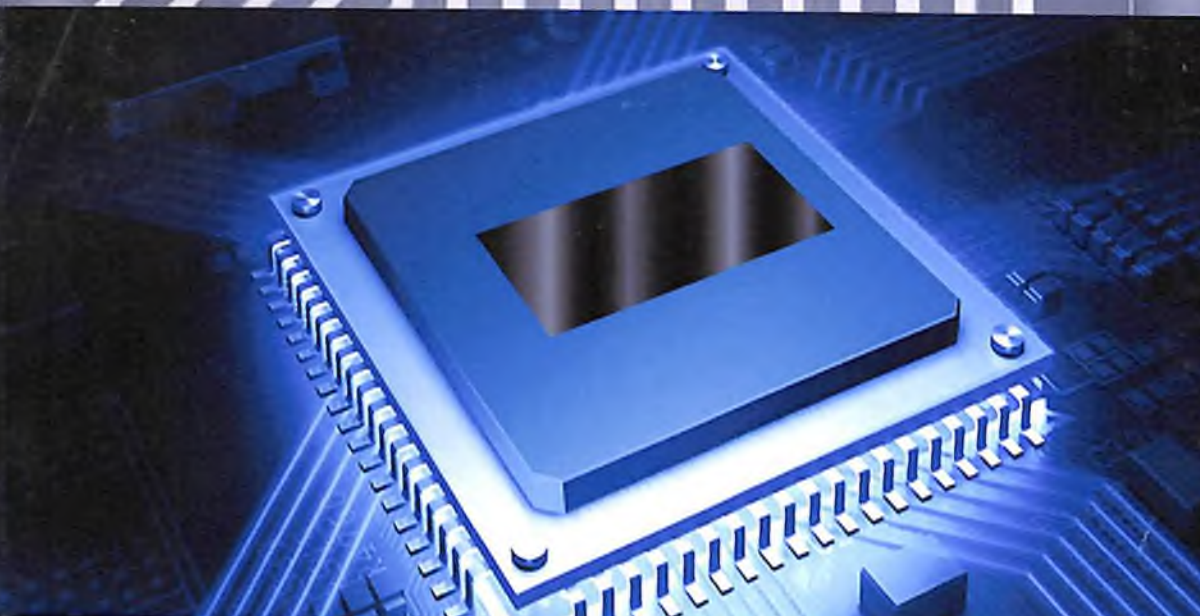
М.М.МУСАЕВ

ПРОЦЕССОРЫ СОВРЕМЕННЫХ КОМПЬЮТЕРОВ



МИНИСТЕРСТВО ПО РАЗВИТИЮ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

М.М.МУСАЕВ

ПРОЦЕССОРЫ СОВРЕМЕННЫХ КОМПЬЮТЕРОВ

(Учебное пособие)

ТАШКЕНТ-2020

УДК: 004.31(075.8)
ББК: 32.973
М 916
М.М.Мусаев. Процессоры современных компьютеров: (Учебное пособие). – Т.: «Aloqachi», 2020. – 416 с.
ISBN 978–9943–6396–2–1

В книге изложены теоретические вопросы построения и функционирования процессоров современной вычислительной техники. Рассмотрены схемы и принципы функционирования различных по архитектуре и назначению процессоров: многоядерных процессоров персональных компьютеров, сигнальных процессоров систем реального времени, графических процессоров видеосистем компьютеров. Особое внимание уделено организации вычислительного процесса, оперативной памяти, увеличению производительности и средствам расширения функций процессоров, их аппаратной реализации. Материал подготовлен в соответствии с учебной программой дисциплины «Архитектура компьютеров». Материалы учебного пособия могут быть использованы при изучении смежных курсов: «Параллельные вычисления», «Системы реального времени», «Встроенные системы», «Микропроцессоры».

Материалы книги могут быть полезны для специалистов по проектированию компьютерных и коммуникационных средств информационных технологий.

Предназначено для студентов направления подготовки бакалавров 5330500-Компьютерный инжиниринг ("Компьютерный инжиниринг", "ИТ-сервис", "Мультимедийные технологии") и магистров 5А330501-«Компьютерный инжиниринг («Проектирование компьютерных систем», «Проектирование прикладных программных средств», «Информационные и мультимедийные технологии», «Информационная безопасность, криптография и криптоанализ»).

Рекомендовано Учебно-методическим советом ТУИТ в качестве учебного пособия для бакалавров и магистров соответствующих специальностей.

УДК: 004.31(075.8)
ББК: 32.973

Рецензенты:

Ю.Г. Шипулин – профессор кафедры «Атоматизация и управление» ТГТУ имени Ислама Каримова, д.т.н., проф.;

Д.С. Яхшибоев – Декан факультета «Компьютер инжиниринг» ТУИТ имени Мухамада ал-Хоразмий, доктор философии, доцент.

ISBN 978–9943–6396–2–1

ВВЕДЕНИЕ

Трудно представить себе рост производительности труда и эффективную организацию работы человека без применения компьютеров в таких областях, как планирование и управление производством, проектирование машин, приборов и оборудования, наука и образование. Наиболее эффективным направлением применения компьютеров по-прежнему являются задачи, требующие сложных вычислений и преобразований при разработке новых технологий и развитии таких отраслей, как машиностроение, авиация, военная техника, строительство масштабных зданий и сооружений.

Произошли фундаментальные изменения в сфере научных исследований и разработок. Возможности вычислительных машин ускорили развитие методов моделирования сложных процессов и алгоритмов численного эксперимента. Это позволило изучать процессы и явления, которые слишком сложные для исследования аналитическими методами или проведением натурного эксперимента. Стало возможным исследовать в реальном времени процессы физико-химических и ядерных реакций, изучать глобальные атмосферные процессы, прогнозировать экономическое и промышленное развитие регионов. При этом производительность и вычислительные ресурсы компьютеров обеспечили решение таких масштабных задач в заданное время. Таким образом, производительность и вычислительные ресурсы стали основными характеристиками компьютеров независимо от их архитектуры и сферы применения.

Расширение сферы применения компьютеров постоянно требует развития его архитектуры и повышения скорости обработки данных. В свою очередь, возрастание производительности компьютерной техники позволяет повышать сложность решаемых задач и постоянно расширять круг исследуемых проблем. Сложность и трудоемкость задач, которые в настоящее время решаются с помощью компьютеров, являются огромными и на порядки превышают их возможности в недавнем прошлом. Росту производительности компьютеров способствует и постоянное совершенствование технологий создания средств компьютерной техники.

Технологические решения в области компьютерной техники постоянно совершенствовались. На первом этапе развития ЭВМ предполагалось, что увеличение производительности связано, прежде всего, с развитием элементной базы процессоров, памяти и других средств обработки. Переход на транзисторы (в 50-е годы) и интегральные схемы (в конце 60-х годов) оправдывал эти предположения. Однако ряд научных идей по совершенствованию архитектуры (конвейерная обработка, виртуальная и кэш-память, рост числа процессоров) показал, что и организационные решения могут во многом определять характеристики вычислительных машин. Одновременно с совершенствованием элементной базы развивались и научные идеи, связанные с логическим построением узлов обработки, хранения и передачи данных, способами организации выполнения операций, принципами управления компьютерными ресурсами - всего того, что впоследствии получило название архитектуры.

На всех стадиях развития и совершенствования самих компьютеров и технологий обработки данных, основным узлом, определяющим показатели производительности, были и являются процессоры (или микропроцессоры). Появление дополнительных обрабатывающих узлов: процессоров плавающей запятой, математических сопроцессоров, контроллеров прямого доступа к памяти позволило увеличить производительность за счет совершенствования архитектуры, но кардинального прорыва в ускорении вычислений это не дало. Однако, идеи распараллеливания вычислений получили свое развитие.

До недавнего времени рост производительности обеспечивался во многом повышением тактовой частоты основных вычислительных элементов – процессоров. Но возможности такого подхода оказались не безграничными – после некоторого рубежа дальнейшее увеличение тактовой частоты требует больших технологических усилий, сопровождается ростом энергопотребления и наталкивается на непреодолимые проблемы теплорегуляции.

В таких условиях практически неизбежным явилось кардинальное изменение основного принципа производства компьютерной техники – вместо создания сложных высокочастотных процессоров новая постановка проблемы стала состоять в разработке

«составных» процессоров, состоящих из множества равноправных и сравнительно простых вычислительных элементов – ядер. Максимальная производительность процессоров в этом случае приблизительно является равной сумме производительности вычислительных ядер, входящих в процессоры. Тем самым, упаковывая в чипах процессоров все большее количество ядер, можно добиваться роста производительности без повышения тактовой частоты. Вместо гонки частот наступила и продолжается эпоха многоядерных процессоров.

Идея распараллеливания процессов обработки между исполнительными узлами компьютера получила свое развитие благодаря появлению технологий мультимедиа, когда данные для обработки могут представляться в различных форматах в виде звука, речи, аудиосигналов, статических и динамических изображений. Появились процессорные узлы в аудио и видеокартах, коммуникационные контроллеры обмена данными между скоростными и относительно медленными узлами компьютера, контроллеры управления шинами. Конечно, они уступали основному, центральному процессору в полноте реализуемых функций, но имели уже необходимые средства приема, временного хранения и обработки данных.

В данной книге архитектура процессоров рассмотрена с учетом всех вышеперечисленных принципов их организации. Так как процессор управляет ресурсами всего компьютера, дается начальное представление об архитектуре самой вычислительной машины, ее отдельных компонентах и программном обеспечении. Детально рассмотрены архитектуры наиболее широко применяемых многоядерных и сигнальных процессоров.

ГЛАВА 1. АРХИТЕКТУРА, КОМПОНЕНТЫ И ПАРАМЕТРЫ КОМПЬЮТЕРОВ

1.1. Понятие архитектуры и организация обработки данных

ЭВМ, компьютер — это комплекс аппаратных и программных средств, предназначенных для приема, хранения, обработки и выдачи информации при решении вычислительных и информационных задач.

При расширенном представлении в понятие вычислительная машина, компьютер входят такие понятия, как состав устройства, число регистров процессора, емкость памяти, тактовая частота центрального процессора. Этот круг вопросов принято определять понятием структурная организация компьютера.

Архитектура ЭВМ – это абстрактное представление компьютера, которое отражает его структурную, схемотехническую и логическую организацию. Понятие архитектуры является комплексным и включает в себя:

- структурную схему компьютера;
- средства и способы доступа к элементам структурной схемы компьютера;
- организацию и разрядность интерфейсов;
- набор и доступность регистров;
- организацию и способы адресации памяти;
- способы представления и форматы данных;
- набор команд компьютера и их форматы;
- обработку нештатных ситуаций (прерываний).

Можно рассматривать компьютер как совокупность технических средств, служащих для автоматизированной обработки цифровых данных по заданному алгоритму.

Алгоритм - одно из фундаментальных понятий математики и вычислительной техники. Международная организация стандартов (ISO) формулирует понятие *алгоритм* как «конечный набор предписаний, определяющий решение задачи посредством конечного количества операций». Помимо этой стандартизированной формулировки существуют и другие определения. Основными свойствами алгоритма являются: дискретность, определенность, масштабируемость и результативность.

Дискретность выражается в том, что алгоритм описывает действия над дискретной (цифровой) информацией (например, числовой или символьной), причем сами эти действия также дискретны.

Свойство **определенности** означает, что в алгоритме указано все, что должно быть сделано, причем ни одно из действий не должно трактоваться двояко.

Масштабируемость алгоритма подразумевает его применимость к множеству значений исходных данных, а не только к каким-то их уникальным значениям.

Результативность алгоритма состоит в возможности получения результата за конечное число шагов.

Рассмотренные свойства алгоритмов предопределяют возможность их реализации на компьютере, при этом процесс, порождаемый алгоритмом, называют **вычислительным процессом.**

В основе архитектуры современных компьютеров лежит представление алгоритма решения задачи в виде программы последовательных вычислений. Если программа — это упорядоченная последовательность команд, подлежащая обработке, то архитектура – это множество взаимосвязанных компонент: аппаратного обеспечения (hardware), программного обеспечения (software), прикладных программ обработки данных, средств приема, преобразования, хранения и передачи данных, а также распределения функций между компонентами при выполнении общей задачи обработки.

Структурную организацию компьютера можно рассматривать как комплекс аппаратно-программных средств, взаимодействующих в процессе решения вычислительной задачи (рис.1.1). Блок-схема компьютера представлена как обобщенная схема построения вычислительной машины.

Схема состоит из устройства хранения, устройств обработки и устройств управления. Кроме того, для доступа к внутренним ресурсам компьютера используется устройства ввода и вывода данных. Это схема универсальных компьютеров в классическом варианте.

Рис.1.1. Обобщенная блок-схема компьютера

Основным элементом архитектуры компьютера является центральный процессор (или микропроцессор) – программно–управляемое устройство, выполненное в виде конструктивно самостоятельного элемента – чипа или набора чипов. Именно технические характеристики процессора – длина слова, тактовая частота, системы исполняемых команд, во-многом определяют параметры компьютера в целом: функциональные возможности, производительность, класс решаемых задач, точность представления результатов.

Процессор является основным вычислительным блоком компьютера, в котором сосредоточены:

- арифметико-логическое устройство (АЛУ) для выполнения арифметических и логических операций;
- блок управления работой процессора и режимом обмена с внешними устройствами;
- внутрипроцессорная память, выполняющая роль сверхоперативной памяти в виде внутренних регистров общего назначения (РОН).

Процессор технологически исполняется в виде полупроводникового кристалла – как большая интегральная схема

(БИС). Набор интегральных схем процессора, оперативной памяти, контроллеров и адаптеров внешних устройств, размещаемых на материнской плате, называется микропроцессорным комплектом.

Оперативная память предназначена для хранения программ управления работой компьютера и данных для обработки. Чаще всего это отдельная БИС, размещаемая на материнской плате вместе с процессором. Это увеличивает скорость обмена данными между процессором и оперативной памятью.

Линии связи (внутримашинный интерфейс) служат для сопряжения центральных узлов машины с ее внешними устройствами.

Внешние устройства обеспечивают доступ пользователя к ресурсам компьютера, а также взаимодействие на уровне ввода исходных данных и вывода результатов обработки. В состав внешних устройств входят также узлы внешней памяти, чаще всего дисковые накопители, и устройства передачи по каналам связи.

На рис. 1.2. представлена обобщенная схема взаимодействия основных аппаратных компонентов компьютера.

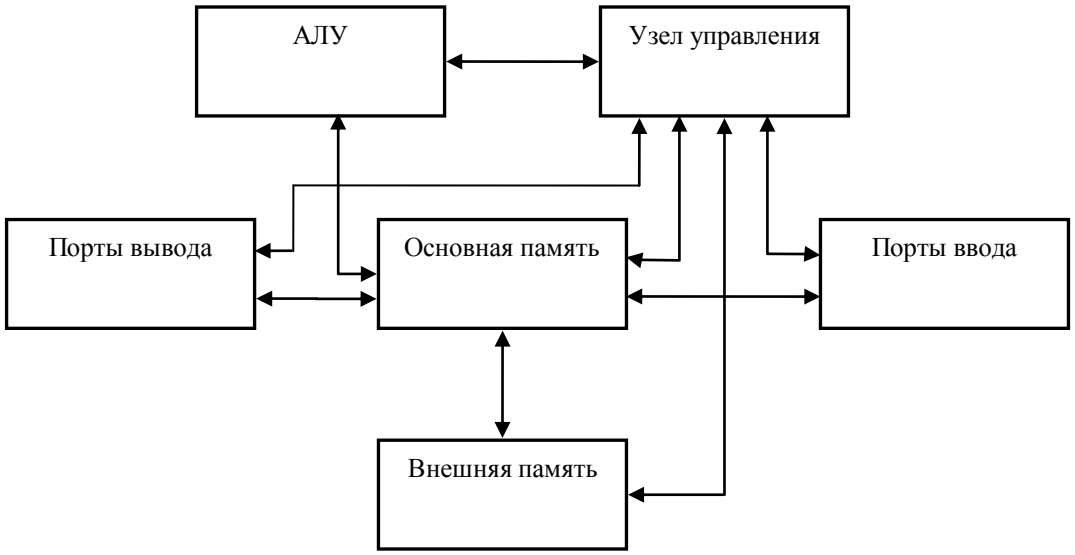


Рис.1.2. Схема взаимодействия основных узлов компьютера

Большинство современных компьютеров построены по принципу программного управления. Этот принцип предполагает наличие в архитектуре трех основных узлов: основной памяти, узла управления и вычислительного узла (АЛУ).

Эти три узла образуют вычислительное ядро и обеспечивают режим программного управления, при этом:

- команды программы располагаются в основной памяти;
- ячейки имеют последовательную сквозную нумерацию (адреса);
- команды читаются и выполняются в естественной последовательности, заложенной в программе.

Арифметико-логическое устройство (АЛУ) производит арифметические и логические операции над поступающими из памяти данными, формирует выходной результат, который либо записывается в основную память, либо временно хранится в одном из сверхбыстродействующих регистров общего назначения РОН.

Основная память состоит из адресуемых (основная память – ОЗУ, постоянная память - ПЗУ) ячеек с произвольным доступом и сквозным адресным пространством, и неадресуемых ячеек, образующих кэш – память. Кэш – память по объему уступает ОЗУ, но превосходит по быстродействию. При считывании из ОЗУ содержимое ячейки не меняется и может быть повторная выборка. При записи в ячейку код стирается и на его место записывается новый код (число или команда). Объем основной памяти определяется количеством хранимых слов, а его быстродействие – временим обращения, т.е. продолжительностью записи и считывания нужного слова.

Процесс исполнения алгоритма и соответствующей компьютерной программы координирует **узел управления** (УУ). В соответствии с текущими командами УУ руководит процессом выборки очередной, подлежащей исполнению команды, выборки необходимых в процессе обработки данных, выполнения нужной операции в АЛУ, размещения полученного результата по указанному адресу.

В процессе выполнения текущей команды программы УУ управления формирует последовательность управляющих сигналов для устройств, участвующих в процессе реализации команды. Если выполнению подлежит команда взаимодействия с внешними устройствами – загрузка данных с внешней памяти, ввод или вывод

данных, то узел управления формирует соответствующие запросы и команды синхронизации процесса обмена.

1.2. Классификации компьютеров, история развития

Существуют различные классификации компьютеров, при помощи которых проводится анализ возможностей аппаратных и программных средств, при решении вычислительных задач.

По назначению компьютеры можно разделить на три группы: универсальные (общего назначения), проблемно-ориентированные и специализированные. **Универсальные** компьютеры предназначены для решения самых различных инженерно-технических, экономических, математических, информационных задач, отличающихся сложностью алгоритмов и большим объемом обрабатываемых данных.Они широко применяются в центрах обработки данных крупных компаний и государственных организаций, в качестве серверов в корпоративных сетях и телекоммуникационных узлах. К категории универсальных относятся и широко используемые персональные компьютеры.

Характерными чертами универсальных компьютеров являются:

- высокая производительность;
- большое разнообразие выполняемых арифметических, логических и специальных операций по обработке данных;
- большая емкость оперативной памяти;
- развитая система ввода-вывода информации, обеспечивающая подключение разнообразных внешних устройств.

Проблемно-ориентированные компьютеры предназначены для решения более узкого круга задач, связанных,как правило,с управлением технологическими объектами,с регистрацией, накоплением и обработкой относительно небольших объемов данных в реальном масштабе времени, с выполнением расчетов по относительно несложным алгоритмам. Они обладают ограниченными по сравнению с универсальными компьютерами аппаратными и программными ресурсами.

Специализированные компьютеры предназначены для решения определенного, узкого круга задач или реализации строго определенной группы функций. Такая узкая ориентация компьютеров

позволяет четко специализировать их структуру, существенно снизить их сложность и стоимость при сохранении высокой производительности и надежности их работы.

К специализированным компьютерам можно отнести, например, адаптеры и контроллеры, выполняющие логические функции управления отдельными несложными техническими устройствами, агрегатами и процессами, устройства согласования и сопряжения работы узлов вычислительных и телекоммуникационных систем (аппаратура аудио и видеокарт компьютеров, процессоров игровых приставок, телестудий, электронных АТС).

Выделяют следующие поколения развития компьютеров и их элементной базы.

Первое поколение (1937-1953) – переход от механических к электронным компонентам. На роль первой в истории электронной вычислительной машины в разные периоды претендовало несколько разработок. Общим у них было использование схем на базе электронно-вакуумных ламп вместо ранее применявшихся электромеханических реле. Предполагалось**,** что электронные ключи будут значительно надежнее электромеханических, поскольку в них отсутствуют движущиеся части, но технология того времени была настолько несовершенной, что по надежности электронные лампы оказались ненамного лучше, чем реле. Однако у электронных компонентов имелось одно важное преимущество: выполненные на них ключи могли переключаться примерно в тысячу раз быстрее своих электромеханических аналогов.

Второе поколение (1954-1962) – переход к полупроводникам. Второе поколение характеризуется рядом достижений в элементной базе, структуре и программном обеспечении. Принято считать, что поводом для выделения нового поколения вычислительных машин стали технологические изменения, переход от электронных ламп к полупроводниковым диодам и транзисторам с высокой частотой переключения**.**

Третье поколение (1963-1972) – переход к интегральным микросхемам. Третье поколение ознаменовалось резким увеличением вычислительной мощности машин, ставшим следствием больших успехов в области развития архитектуры и технологии

программного обеспечения. Основные технологические достижения связаны с переходом от дискретных полупроводниковых элементов к интегральным микросхемам и началом применения полупроводниковых запоминающих устройств, начинающих вытеснять запоминающие устройства на магнитных сердечниках. Существенные изменения произошли и в архитектуре вычислительных машин. Это, прежде всего, микропрограммирование - как эффективная техника построения устройств управления сложными вычислительными алгоритмами, а также наступление эры конвейеризации и параллельной обработки. В области программного обеспечения определяющими вехами стали первые операционные системы и реализация режима разделения времени.

Четвертое поколение (1972-1984) – применение сверхбольших интегральных схем. Отсчет четвертого поколения обычно ведут с перехода на интегральные микросхемы большой (large-scaleintegration, LSI) и сверхбольшой (verylarge-scaleintegration, VLSI) степени интеграции. К первым относят схемы, содержащие около 1000 транзисторов на кристалле, в то время как число транзисторов на одном кристалле VLSI имеет порядок 100 000. При таких уровнях интеграции стало возможным уместить в одну микросхему не только центральный процессор (ЦП) и его внутреннюю память, но и дополнительные узлы - основную память и систему ввода/вывода.

Конец 70-х и начало 80-х годов — это время становления и последующего победного шествия микропроцессоров и микро-ЭВМ, что, однако, не снижает важности изменений, произошедших в архитектуре других типов вычислительных машин и систем.

Пятое поколение (1984-1990) – применение дополнительных вычислительных узлов (процессоров). Главным поводом для выделения вычислительных систем второй половины 80-х годов в самостоятельное поколение стало стремительное развитие вычислительных систем с множеством процессоров, ставшее побудительным мотивом для прогресса в области параллельных вычислений. Ранее параллелизм вычислений выражался лишь в виде конвейеризации, векторной обработки и распределения работы между небольшим числом узлов обработки. Вычислительные системы

пятого поколения обеспечивают такое распределение задач по множеству процессоров, при котором каждый из процессоров может выполнять задачу отдельного пользователя. Это уже постепенный переход к параллельным методам вычисления.

В рамках пятого поколения в архитектуре вычислительных систем сформировались два принципиально различных подхода: архитектура с совместно используемой памятью и архитектура с распределенной памятью.

Шестое поколение (1990-2006) – создание многоядерных процессоров, развитие технологии параллельных вычислений. На ранних стадиях эволюции вычислительных средств смена поколений ассоциировалась с революционными технологическими прорывами. Каждое из первых четырех поколений имело четко выраженные отличительные признаки и вполне определенные хронологические рамки. Последующее деление на поколения уже не столь очевидно и может быть понятно лишь при ретроспективном взгляде на развитие вычислительной техники. Пятое и шестое поколения в эволюции компьютеров — это отражение нового качества, возникшего в результате последовательного накопления частных достижений, главным образом в архитектуре вычислительных систем и, в несколько меньшей мере, в сфере технологий. Переход к многоядерным процессорам позволяет реализовать истинно параллельные вычисления, в отличие от псевдопараллельных (вроде конвейерной обработки).

Следующее поколение (2006-) — это архитектуры массового параллелизма. Поводом для начала отсчета нового поколения стали значительные успехи в области параллельных вычислений, связанные с широким распространением вычислительных систем с массовым параллелизмом. Это совокупность большого количества (до нескольких тысяч) взаимодействующих, но достаточно автономных вычислительных узлов. По вычислительной мощности такие системы уже успешно конкурируют с супер -ЭВМ.

Появление вычислительных систем с массовым параллелизмом дало основание говорить о производительности, измеряемой в триллионы операций в секунду, степени интеграции в миллионы транзисторов на кристалл, объеме адресуемой памяти в терабайты.

Классификация компьютеров по типу архитектуры. В 1966 г. М. Флинном была предложена классификация архитектур компьютеров и вычислительных систем, в основу которой положено понятие потока, или последовательности элементов (команд или данных), обрабатываемых процессором. Соответствующая система классификации, основанная на рассмотрении числа потоков команд и потоков данных, приводи к четырем базовым классам (рис. 1.3).

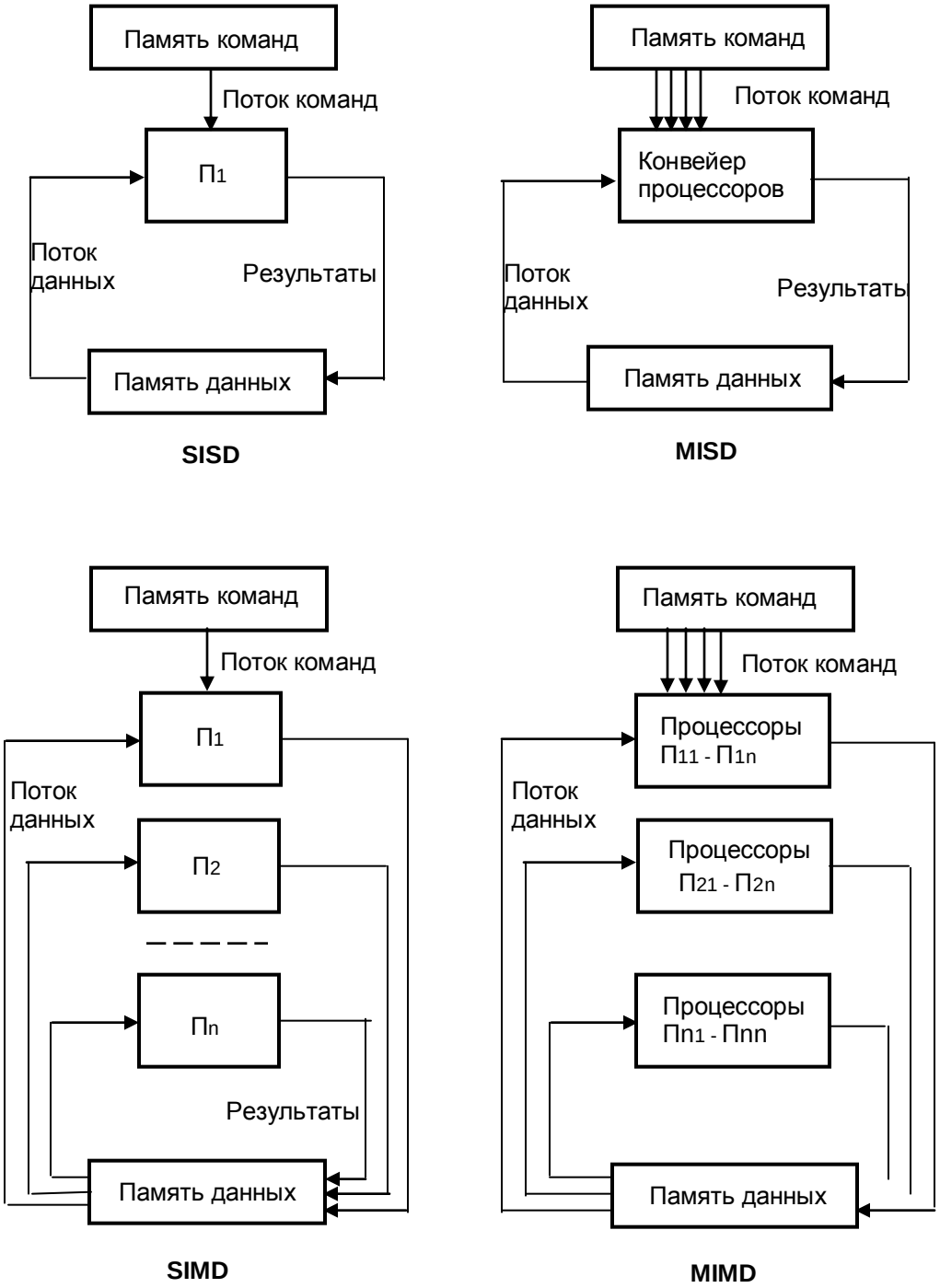


Рис.1.3. Классификация Флинна

Кратко опишем характерные особенности каждой из архитектур.

Архитектура SISD (SISD – Single Instruction Single Datastream) - архитектура одиночного потока команд и одиночного потока данных**.** Она охватывает все однопроцессорные и одномашинные варианты систем. Все компьютеры классической структуры попадают в этот класс. Представителями этого класса являются фон-неймановские архитектуры, где имеется только один поток команд, команды обрабатываются последовательно и каждая команда реализует одну операцию с одним потоком данных.

Архитектура SIMD (SIMD–Single Instruction Single Datastream) – архитектура одиночного потока команд и множественного потока данных**.** Машины данной архитектуры позволяют выполнять одну арифметическую операцию сразу над многими данными – элементами вектора. Системы этого типа однородные, т.е. процессорные элементы, входящие в систему, идентичны, и все они управляются одной и той же последовательностью команд. Однако каждый процессор обрабатывает свой поток данных. Под эту схему хорошо подходят задачи обработки матриц или векторов.

Достоинства SIMD состоят в том, что все параллельные исполнительные устройства синхронизированы, и все они реагируют на одну и ту же инструкцию, которая выдается при работе одного и того же счетчика команд. С точки зрения программиста это весьма близко к уже знакомому SISD. Хотя каждое исполнительное устройство будет выполнять одну и ту же инструкцию, у каждого из них есть свои собственные регистры адресов, и поэтому у каждого устройства разные адреса данных.

Архитектура MISD (Multiple Instruction Single Datastream) – архитектура множественного потока команд и одиночного потока данных. Данная архитектура предполагает построение своеобразного процессорного конвейера, в котором результаты обработки передаются от одного процессора к другому по цепочке. В современных компьютерах по этому принципу реализована схема совмещения операции, в которой параллельно работают различные функциональные блоки, и каждый из них делает свою часть в общем цикле обработки команды. Компьютерные архитектуры этого типа не получили распространения.

Архитектура MIMD (Multiple Instruction Single Datastream) – архитектура множественного потока команд и множественного потока данных**.** В архитектуре этого типа предполагается, что все процессоры системы работают по своим программам с собственным потоком команд. В простейшем случае они могут быть автономны и независимы. Такая схема использования вычислительных систем часто применяется во многих крупных вычислительных центрах для увеличения пропускной способности при обработке больших потоков данных. Класс очень широк, поскольку включает в себя всевозможные мультипроцессорные, мультимашинные и многоядерные компьютерные системы.

Итак, с точки зрения организации архитектуры:

SISD – единственный поток инструкций, единственный поток данных. Реализация в виде одноядерных процессоров. Это классическая архитектура Фон-Неймана.

SIMD – единственный поток инструкций, несколько потоков данных. Реализация может быть в виде многоядерных процессоров и систологических вычислительных схем обработки сигналов.

MISD – множественный поток инструкций, единственный поток данных. Может быть реализована в виде конвейера процессоров, пока архитектура не получила успешной аппаратной реализации.

MIMD – множественный поток инструкций, множественный поток данных. Реализация в виде мультипроцессорной системы.

Схема классификации Флинна является распространенной и широко используется при первоначальной оценке той или иной компьютерной системы, т.к. позволяет сразу оценить базовый принцип ее работы. Наряду с этой классификацией существуют и другие, но они меньше используются.

1.3. Технологии построения компьютерных систем

Хотя во всех компьютерах, начиная с «умных» бытовых приборов и сотовых телефонов и заканчивая самыми большими суперкомпьютерами, используется одинаковый набор технологий аппаратного обеспечения, различие в сферах их применения ведет к разным конструктивным требованиям и к разным методам использования базовых технологий аппаратного обеспечения. В

общем, по характеру использования компьютеры делятся на несколько классов.

Суперкомпьютеры. Самые мощные компьютерные системы, предназначенные для решения сложных, многопараметрических задач. Они имеют колоссальную производительность, измеряемую терафлопсами (триллионы операций в секунду). Основное назначение суперкомпьютеров – моделирование сложных природных явлений, технических и социальных процессов. Для решения этих задач от суперкомпьютеров требуется способность к одновременному выполнению множества вычислений, поэтому они технологически построены из множества параллельно работающих процессоров, каждый из которых решает свою подзадачу общей задачи. Несмотря на размеры и количество исполнительных блоков они также относятся к категории компьютеров, т.к. все компоненты суперкомпьютеров решают одну, общую для всех задачу.

Серверы. Они отличаются большой производительностью, обслуживают корпоративные системы и сети, имеют несколько проблемных ориентаций:

- файл-серверы;
- серверы баз данных;
- серверы приложений;
- Интернет-серверы.

Это, как правило, многопроцессорные компьютеры с высокой надежностью, круглосуточным графиком работы, возможностью замены основных блоков без выключения системы, широкими возможностями по наращиванию вычислительной мощности.

Персональные компьютеры, рабочие станции. Эти компьютеры являются непременным атрибутом офисов и учебных аудиторий. Для офисных компьютеров характерно предоставление отдельным пользователям хорошей вычислительной производительности при низкой стоимости и использование компьютерных программ независимых производителей. История развития этих машин насчитывает около 40 лет, их совершенствование способствовало развитию многих вычислительных технологий. Когда необходимо решать сложные

задачи, модульный принцип построения позволяет оснащаться дополнительными возможностями памяти и внешних устройств.

Встроенные компьютеры представляют собой самый большой класс компьютеров и имеют самый широкий спектр применения. К встроенным компьютерам относятся микропроцессоры, которые можно найти в автомобиле, сотовых телефонах, видеоиграх, сетях телекоммуникаций, медицинской аппаратуре, в системах управления современными станками или транспортными средствами. Встроенные компьютерные системы сконструированы для запуска одного приложения или набора взаимосвязанных приложений, которые обычно интегрированы с аппаратной частью и поставляются в виде единой системы.

Таков круг основных компьютерных систем. По сути, в настоящее время каждый автономно работающий компьютер является вычислительной системой, так как содержит множество вычислительных, управляющих, коммуникационных узлов и элементов, каждый из которых также является сложным электронным компонентом.

По характеру **пространственного распределения** элементов компьютерных систем делятся на системы **сосредоточенного (локального)** и **распределенного** типов. Обычно такое деление касается только многомашинных или многопроцессорных вычислительных систем: в этом классе можно найти вычислительные системы как распределенного, так и локального типов. Как правило, многопроцессорные системы относятся к системам локального типа. Если взаимодействие в составе многомашинной системе распределенного типа организуется с помощью специальных линий связи и определенных протоколов взаимодействия, такую компьютерную систему называют **вычислительной сетью.**

Сложность современных вычислительных машин закономерно привела к понятию **архитектуры компьютера**, охватывающей комплекс общих вопросов ее построения, существенных в первую очередь для пользователя, интересующегося главным образом возможностями машины, а не деталями ее технического исполнения. Круг вопросов, подлежащих решению при разработке архитектуры компьютера можно условно разделить на вопросы общей структуры,

организации вычислительного процесса и общения пользователя с машиной, вопросы логической организации представления, хранения и преобразования информации. **Архитектура** – описание вычислительной системы на некотором общем уровне, включающее взаимодействие аппаратной и программной частей компьютера, описание пользовательских возможностей программирования, системы команд и средств пользовательского интерфейса, а также организации памяти.

1.4. Основные блоки и параметры компьютеров

Рассмотрим особенности организации основных блоков компьютера в наиболее широко используемом варианте применения – в персональных компьютерах (ПК). В самом общем виде структурная схема ПК приведена на рис.1.4. В основе работы любого компьютера лежит функционирование процессора. Под процессором мы будем понимать полупроводниковый чип, на котором размещены основные узлы основного, или центрального процессора. В компьютере, на материнской плате, как правило, размещается кроме чипа самого процессора много дополнительных компонент, которые вместе образуют микропроцессорный комплект или микропроцессор. На рисунке показан основной мозг компьютера – микропроцессор (МП).

Память в компьютере бывает двух видов: постоянная (ПЗУ) и оперативная (ОЗУ). Постоянная память служит для хранения неизменяемых программ и констант, с их помощью компьютер приводится в рабочее состояние, данные из нее могут только считываться. Оперативная память доступна как для считывания, так и для записи, в ней находятся исполняемые программы и обрабатываемые данные.

Для того, чтобы считывать программы и обрабатываемые данные из памяти предусмотрены специальные линии, по которым происходит передача. Набор этих линий образует шину данных (ШД). Для передачи адресов в память, где находятся команды программ и данные для обработки, служит шина адреса (ША). Процессор подает по ША адрес, после чего данные по ШД могут быть считаны или занесены в память. МП руководит работой узлов и блоков компьютера через соответствующую шину управления (ШУ). МП

считывает из памяти команды программы, последовательно исполняет эти команды, обрабатывает хранящуюся в ОЗУ информацию и записывает результаты.

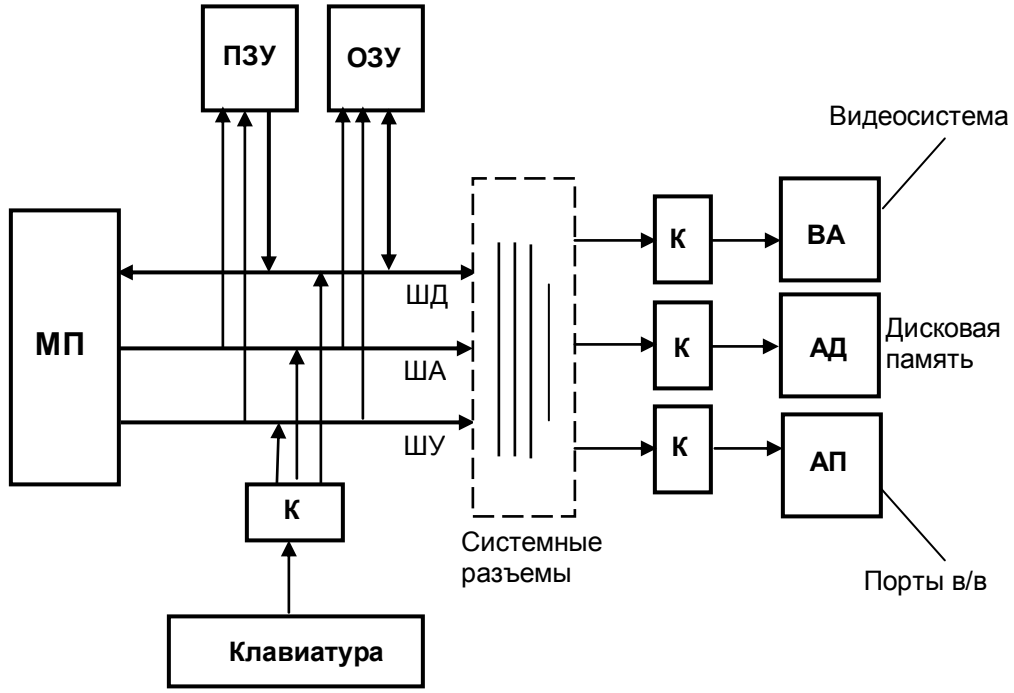


Рис.1.4. Организация персонального компьютера

Для управления работой компьютера через экран монитора используется клавиатура (К) и мышка (на схеме не показана). Монитор входит в состав видеосистемы компьютера (ВА). В составе компьютера имеются специальные управляющие блоки – контроллеры (К), выполняющие функции реализации команд управления внешними устройствами. К внешним устройствам относятся, кроме монитора, накопители большой емкости на дисках и порты ввода-вывода данных.

С развитием технологий создания компьютеров был сделан переход к модульной организации и унификации узлов и блоков, с тем, чтобы пользователь имел возможность конфигурировать вычислительную систему.

Представленная организация ПК содержит несколько самостоятельных технологических частей:

- материнская плата, на которой размещаются МП, ПЗУ, контроллер клавиатуры и мышки;
- плата модулей оперативной памяти;
- видеоадаптер (ВА), обслуживающий видеосистему;

- адаптер дисковых накопителей (АД);
- адаптер устройств (портов) ввода-вывода (АП).

Все электронные блоки адаптеров и контроллеров размещаются на отдельных платах и подключаются к системе через один из системных разъемов. Кроме перечисленных блоков в состав компьютера входят корпус, блок питания, необходимые установочные панели, соединительные кабели.

Функции процессоров по организации вычислений.

Основной задачей центрального процессора является выборка команд из памяти, анализ типа выполняемой операции, выборка данных из памяти, выполнение команды, занесение в память получаемых результатов.

Все действия процессора реализуются с помощью команд. Команда – это совокупность двоичных кодовых слов, содержащих информацию о том, какое действие нужно выполнить (код операции), каким образом определяется местоположение операндов в основной памяти (режим адресации), где расположены операнды (адресная часть). Команда, как и данные, хранится в оперативной памяти, поэтому номер ячейки памяти, где она расположена, читается процессором. После чтения команды начинается процесс ее исполнения функциональными узлами компьютера.

Начальная программа работы компьютера запускается нажатием клавиатуры или запуском с машины, т.е. начинается исполнение программы начального пуска, которая хранится в ПЗУ – это комплекс программ BIOS (BasicInputOutputSystem).В состав BIOS входят следующие программы:

- начального тестирования системы;
- организации начальной загрузки;
- проверки исправности ОЗУ и других узлов.

Механизм конфигурирования – это задание номенклатуры подключенных устройств и их настройка на рабочие режимы. Для выполнения конфигурирования предусмотрены энергонезависимая память и специальная программа Setup в составе BIOS. Данные из энергонезавимой памяти считываются и используются для настройки системы. Процедура конфигурирования построена по следующему принципу:

- считывание сведений о конфигурации ПК и настройка устройств на рабочие режимы во время выполнения начальной загрузки;
- внесение изменений в конфигурацию системы путем перезагрузки.

Часы реального времени (RealTimeClock – RTC). Наличие в составе компьютерной системы электрической батарейки позволило включить в состав ПК еще один блок – электронные часы. Так появилась возможность работать с вычислительными процессами, разнесенными во времени.

Канал параллельной передачи данных (LTP).Предназначен для подключения принтеров, плоттеров, сканеров, внешних накопителей. Системная поддержка режима параллельной передачи выполняет поиск имеющихся портов, их инициализацию, опрос состояния подключенных устройств, прием и передачу.

Канал последовательной передачи данных (COM). Предназначен для ведения последовательного асинхронного обмена данными по стандарту RS-232C.Системная поддержка режима параллельной передачи выполняет поиск имеющихся портов, их инициализацию, опрос состояния устройств, прием и передачу. Обмен с компьютерной системой порты ведут в режиме прерываний.

Видеосистема. Обеспечивает визуализацию процесса обработки данных в реальном времени. Системный сервис устанавливает видеорежим, осуществляет вывод символов на экран. Для взаимодействия со стандартами системами типа EGA и VGA в состав видеосистемы введено расширение в виде программ «VideoBIOS», предназначенных для настройки видеоконтроллера на работу в конкретных видеорежимах.

1.5. Параметры и технические характеристики компьютеров

Производительность - это показатель скорости работы компьютера. Ранее этот параметр измерялся количеством выполненных операций в секунду (опер/сек). Однако позже, учитывая, что различные по типу команды выполняются с различной скоростью, приняли два основных типа оценок производительности. Оценка MIPS (Million Instruction Per Second) соответствует миллиону

реализуемых в секунду операций над числами с фиксированной запятой. Оценка MFLOPS (Millionsof Floating point Operation Per Second) соответствует миллиону реализуемых операций над числами с плавающей запятой (точкой). Вторая оценка в связи с бурным ростом производительности многопроцессорных систем имеет варианты GFLOPS (миллиард операций в секунду) и TFLOPS (триллион операций в секунду) для чисел с плавающей запятой.

Для отдельно взятых процессоров компьютеров производительность оценивается через тактовую частоту - КГц, МГц, ГГц (соответственно тысяча, миллион и миллиард переключений транзисторов в секунду). Например, процессор с тактовой частотой 100 МГц обеспечивает выполнение 20 млн. простых арифметических операций в секунду.

Емкость памяти. У современного компьютера виды памяти отличаются объемом, скоростью чтения/записи и аппаратной реализацией.

Оперативная память ОЗУ выполняется по технологии DRAM (Dynamic Random Access Memory), то есть динамическая память с произвольным доступом. Несколько DRAM-модулей используются вместе для хранения инструкций и данных программы. В отличие от последовательного доступа к памяти, осуществляемого, к примеру, с использованием магнитной ленты, часть аббревиатуры DRAM означает, что доступ к любому фрагменту памяти занимает практически одинаковое количество времени.

Оперативная память, располагается в виде отдельной БИС на материнской плате и является самым скоростным видом адресуемой памяти. Ее емкость измеряется в Кбайт или Мбайт (тысяча и миллион байт). Емкость накопителей на жестких магнитных дисках (так называемый винчестер) измеряется обычно в гигабайтах (миллиард байт).

В компьютере имеется еще один тип памяти — кэш-память. Она состоит из быстродействующей памяти небольшого объема, работающей в качестве буфера DRAM-памяти (техническое определение слова «кэш» — тайник.).Кэш-память построена с использованием другой технологии памяти - SRAM (Static Random Access Memory), то есть статической памяти с произвольным

доступом. SRAM имеет более высокое быстродействие, чем оперативная память, но меньшую плотность элементов. Это тип неадресуемой памяти, ее работа полностью регулируется операционной системой, к тому же способ адресации и выборки отличаются от адресуемой памяти. Емкость кэш-памяти различных уровней составляет от единиц (внутрикристалльная память) до сотен мегабайт (на материнской плате).

К разряду неадресуемой памяти относятся также регистры общего назначения (РОН), разрядность которых соответствует длине машинного слова, а общее количество колеблется от 16 до 32.

Более подробно виды памяти будут рассмотрены позднее.

Разрядность (длина машинного слова) - это число двоичных разрядов в стандартном машинном формате слова, которым оперирует компьютер. В современных компьютерах она составляет 32 или 64 двоичных разряда. Длина слова влияет на многие параметры компьютера - быстродействие, точность вычислений, скорость чтения или записи в память. Она влияет на точность вычислений особенно при выполнении таких трудоемких операций, как умножение, деление, возведение в степень, извлечение корня. Это связано, в основном, с ошибками округления результатов.

Большое влияние длина машинного слова оказывает на адресное пространство адресуемой памяти. При непосредственной адресации объем адресуемой памяти пропорционален разрядности машинного слова компьютера. Для искусственного увеличения адресного пространства используется виртуальная память и различные способы косвенной адресации.

Пропускная способность относится к характеристикам подсистем ввода и вывода данных при обмене информацией с внешними устройствами или другими компьютерами. Пропускная способность измеряется максимальным количеством единиц информации (бит), передаваемых за единицу времени.

Надежность функционирования оценивается следующими показателями:

- вероятность безотказной работы в течение заданного времени;
- время наработки до первого отказа;
- среднее время восстановления работоспособности.

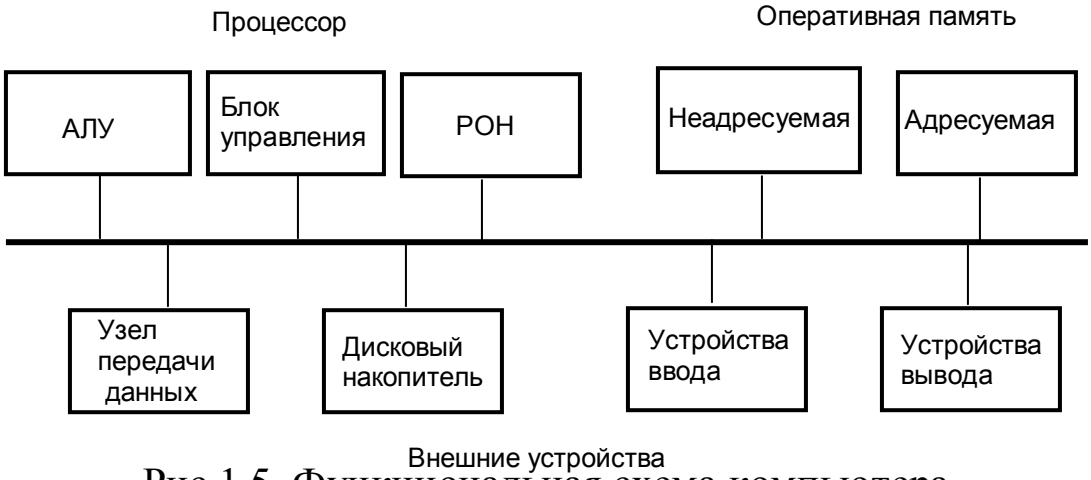


Рис.1.5. Функциональная схема компьютера

С учетом вышеперечисленных компонентов и их параметров можно представить функциональную схему компьютера (рис. 1.5.). Связи между устройствами осуществляются через систему шин. Представленная шина изображена в обобщенном виде, предполагается, что в эту систему входят как локальные (например «процессор-память»), так и общесистемные шины. Каждые из комплекса устройств – процессор, оперативная память, устройства ввода/вывода, накопители и узлы передачи данных будут более подробно рассмотрены в последующих главах.

Наряду с техническими параметрами современные компьютеры оцениваются с точки зрения их функциональных возможностей, влияющих на их качество и цену:

- номенклатура и характеристики внешних устройств хранения, обмена и ввода/вывода информации (принтеры, сканеры, плоттеры);
- номенклатура, емкость и быстродействие всех видов запоминающих устройств (кэш, ОЗУ, ПЗУ, дисковая память);
- типы и пропускная способность узлов сопряжения (портов) с внешними устройствами;
- режимы обработки данных (однопрограммный, многопрограммный, многопользовательский);
- характеристики используемой операционной системы;
- полнота прикладных программ пользователя;
- программная совместимость с другими типами компьютеров;
- возможность работы в составе вычислительной системы или сети.

В настоящее время основные цели использования компьютера - информационное обслуживание и управление, то есть решение информационно-вычислительных задач. Поэтому высокопроизводительные современные компьютеры представляют собой, по сути, вычислительную систему - совокупность одного или нескольких процессоров и контроллеров, их программного обеспечения и периферийного оборудования, организованных для совместного решения задач обработки данных.

1.6. Поколения развития процессоров

Процессор является основным обрабатывающим устройством компьютера, от его технических показателей зависит общая производительность и функциональные возможности. Процессоры также как и компьютеры в целом прошли долгий путь архитектурного и технологического совершенствования, прежде чем достигли нынешних показателей. Кратко рассмотрим историю развития архитектуры процессоров на примере процессоров компании Intel, которая на протяжении многих лет является технологическим лидером в производстве процессоров. Однако, мы не будем придерживаться деления на те или иные поколения, а рассмотрим последовательность совершенствования самой архитектуры и технологических параметров.

Начнем с процессора Intel 80486, архитектурные идеи которого до сих пор составляют базу развития процессоров. Используемая в нем архитектура до сих пор кратко называется 486. Процессор работал на частоте 50 МГц и мог выполнять 40 миллионов команд в секунду. Процессор имел 8 Кб кэша первого уровня, а для изготовления использовался техпроцесс 1000 нанометров (расстояние между транзисторами в чипе).

Следующей архитектурой была Pentium. Эти процессоры появились в 1993 году, здесь был увеличен кэш до 32 кб, частота до 60 МГц, а техпроцесс уменьшен до 800 нм. В дальнейшем размер кэша возрос 32 Кб, а частота достигла 450 МГц. Техпроцесс был уменьшен до 180 нм.

Далее компания начала выпускать процессоры на архитектуре NetBurst. Здесь использовалось 16 Кб кэша первого уровня на каждое

их двух ядер, и до 2 Мб кэша второго уровня. Частота выросла до 3 ГГц, а техпроцесс остался на том же уровне - 180 нм. Уже здесь появились 64 битные процессоры, которые поддерживали адресацию большего количества памяти. Также было внесено множество расширений команд, а также добавлена технология Hyper-Threading, которая позволяла создавать два потока из одного ядра, что повышало производительность.

На смену NetBurst в 2006 году пришла архитектура Intel Core. Одной из причин разработки этой архитектуры была невозможность увеличения частоты в NetBrust, а также ее очень большое тепловыделение. Эта архитектура была рассчитана на разработку многоядерных процессоров, размер кэша первого уровня был увеличен до 64 Кб. Частота осталась на уровне 3 ГГц, но зато была сильно снижена потребляемая мощность, а также техпроцесс до 60 нм.

Процессоры на архитектуре Intel Core поддерживали аппаратную виртуализацию Intel-VT, а также некоторые расширения команд, но не поддерживали Hyper-Threading, поскольку были разработаны на основе архитектуры Pentium, где такой возможности еще не было. Все следующие архитектуры – это улучшенные версии Intel Core.

Архитектура Nehalem пришла на смену Intel Core, у которой были некоторые ограничения, такие как невозможность увеличить тактовую частоту. Она появилась в 2007 году. Здесь используется 45 нм технологический процесс и была добавлена поддержка технологии Hyper-Therading. Процессоры Nehalem имеют размер L1 кэша 64 Кб, 4 Мб L2 кэша и 12 Мб кэша L3. Кэш доступен для всех ядер процессора. Также появилась возможность встраивать графический ускоритель в процессор. Частота не изменилась, зато выросла производительность и размер печатной платы.

Архитектура Sandy Bridge появилась в 2011 году для замены Nehalem. Здесь уже использовался технологический процесс 32 нм, кэш первого уровня содержал 64 Кб, имелось 256 Мб кэша второго уровня и 8 Мб кэша третьего уровня. В экспериментальных моделях использовалось до 15 Мб общего кэша.Также теперь все устройства начали выпускаться со встроенным графическим ускорителем. Была увеличена максимальная частота, а также общая производительность.

Следующий этап развития - процессоры Ivy Bridge. Они работают быстрее, чем Sandy Bridge, а для их изготовления используется техпроцесс 22 нм. Они потребляют на 50% меньше энергии чем предыдущие модели, а также дают на 25-60% высшую производительность. Также процессоры поддерживают технологию Intel Quick Sync, которая позволяет кодировать видео в несколько раз быстрее.

Поколение процессора Intel Haswell было разработано в 2012 году. Здесь использовался тот же техпроцесс - 22 нм, изменен дизайн кэша, улучшены механизмы энергопотребления и немного производительность. Но зато процессор поддерживает множество новых разъемов: LGA 1150, BGA 1364, LGA 2011-3, технологии DDR4 и так далее. Основное преимущество Haswell в том, что эта технология может использоваться в портативных устройствах из-за очень низкого энергопотребления.

Технология Broadwell - это улучшенная версия архитектуры Haswell, которая использует технопроцесс 14 нм. Кроме того, в архитектуру было внесено несколько улучшений, которые позволили повысить производительность в среднем на 5%.

Следующая архитектура процессоров IntelCore - Skylake вышла в 2015 году. Это одно из самых значительных обновлений архитектуры IntelCore. Для установки процессора на материнскую плату используется сокет LGA 1151, теперь поддерживается память DDR4, но сохранилась поддержка DDR3. Поддерживается шина DMI 3.0, которая дает в два раза большую скорость. И уже по традиции была увеличенная производительность, а также снижено энергопотребление.

Технология- KABY LAKE вышла в 2017году, первые процессоры появились в середине января. Здесь было не так много изменений. Сохранен техпроцесс 14 нм, а также тот же сокет LGA 1151. Поддерживаются планки памяти DDR3L SDRAM и DDR4 SDRAM, шины PCI Express 3.0, USB 3.1. Кроме того, была немного увеличена частота, а также уменьшена плотность расположения транзисторов. Максимальная частота 4,2 ГГц.

Мы рассмотрели архитектуры процессора Intel, которые использовались раньше, а также те, которые применяются сейчас.

Дальше компания планирует переход на техпроцесс 10 нм и это поколение процессоров intel будет называться CanonLake. Поэтому в 2017 планируется еще выпустить улучшенную версию SkyLake под кодовым именем Coffe Lake. Также, возможно, будут и другие микроархитектуры процессора Intel пока компания полностью освоит новый техпроцесс.

1.7. Перспективы совершенствования архитектуры компьютеров

Совершенствование архитектуры вычислительных машин и систем началось с момента их появления и не прекращается по сей день. Каждое изменение в архитектуре направлено на повышение производительности или на более эффективное решение задач определенного класса. Эволюцию архитектур определяют самые различные факторы, главные из которых показаны на рис. 1.6.

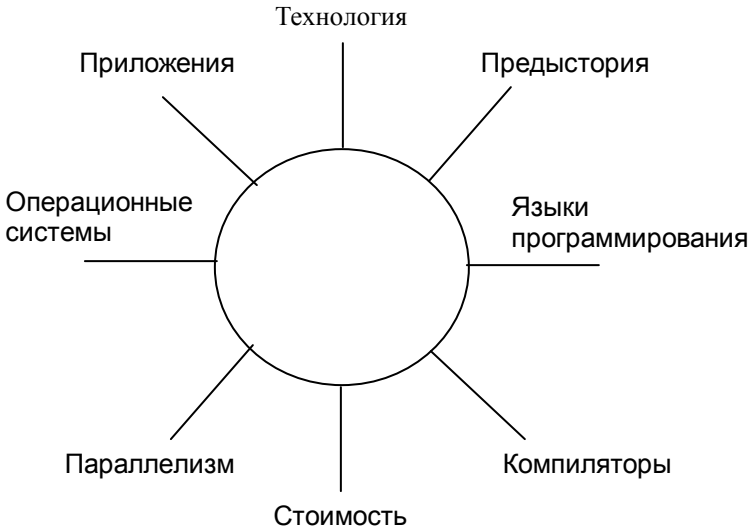


Рис. 1.6. Факторы, определяющие развитие архитектуры компьютеров

Наиболее очевидные успехи в области средств вычислительной техники все же связаны с технологическими достижениями.

Одним из перспективных направлений развития компьютеров является интеграция в рамках одной архитектуры разнородных средств обработки, т.е. создание гетерогенной среды вычислений. Это гибрид архитектурных решений, характерных для сигнальных процессоров и графических ускорителей в сочетании с традиционными решениями универсальных микропроцессоров.

В новых поколениях многоядерных процессоров IntelCore предусмотрены такие новшества, как доведение числа ядер до 16-32, увеличение объемов внутрикристалльной кэш-памяти, встраивание в чип процессора графических ускорителей, контроллеров работы с шинами и памятью, поддержка виртуальной обработки потоков команд и данных, переход к коммутируемым каналам обмена между внешними устройствами и процессором.

Необходимость обработки больших объемов видеоданных (например, для применения в игровых приставках) потребовало создания специальных медиа-процессоров с расширенным набором мультимедийных команд. Поддержка режимов мультимедийной и виртуальной обработки, расширения объемов внутрикристалльной памяти, увеличение числа ядер требуют модернизации операционных систем и компиляторов, совершенствования режимов конвейерной и поточной обработки.

Эти новшества во всех типах процессоров ведущих компаний базируются на совершенствовании производства полупроводниковых кристаллов по 10-15нм технологии при снижении энергопотребления и доведения степени интеграции чипа до сотен миллионов транзисторов на квадратный сантиметр поверхности.

Основной проблемой, возникающей при простом увеличении тактовой частоты, является разогрев кристалла за счет сопротивления в проводниках и отвод тепла. Поэтому для повышения **тактовой частоты** работы современных микропроцессоровиспользуют более совершенный технологический процесс с меньшими проектными нормами, усовершенствованную схемотехнику меньшей каскадности и более плотную компоновку функциональных блоков кристалла.Все эти приемы ориентированы на снижение сопротивления в проводниках за счет уменьшения уровней питающих напряжений, на сокращение длины проводников, что в конечномитоге ведет к уменьшению рассеиваемой на кристалле мощности.

Проблемы увеличения**пропускной способности подсистем памяти**, снабжающей функциональныеустройства процессора работой, включают:

- создание кэш-памяти нескольких уровней;

- увеличение пропускной способности интерфейсов междупроцессором и кэш-памятью, а также между процессором и основной памятью.

Наиболее часто используемое решение состоит в реализации иерархии кэшей. Как правило, на кристалле располагаютсяраздельные кэш первого уровня для данных и команд семкостью в 16 Кб или 32 Кб каждый. Мощные процессоры имеют на кристалле и объединенную кэш-память команд и данныхвторого уровня, емкость которой может составлять от нескольких сотен килобайт до нескольких мегабайт. Кроме этого, какправило, в схему процессора включается интерфейс, позволяющий подключать кэш-память второго и третьего уровней.

Скорость передачи данных в подсистемах памяти определяется количеством передаваемой информации в байтах за единицу времени, поэтомусовершенствование интерфейсов реализуется как увеличениемпропускной способности шин (путем повышения частоты работы шины и/или ее ширины), так и введением дополнительныхшин, «расшивающих» конфликты между процессором, кэш-памятью и основной памятью. В последнем случае одна шинаработает на частоте процессора с кэш-памятью, а вторая – начастоте работы основной памяти.

Каждое семейство процессоров демонстрирует в следующем поколении увеличение числа функциональных исполнительных устройств и улучшение их характеристик, как временных (сокращение числа ступеней конвейера и уменьшениедлительности каждой ступени), так и функциональных, например, введением MMX-расширений системы команд.

В настоящее время процессоры могут выполнять до 10 операций за такт, длина конвейера может составлять 18 этапов. При этом широко используются операции переименования регистров, предсказания переходов, устранения зависимости между командами по даннымыпо управлению.

Устоявшихся решений в этой области практически нет, таккак каждый процессор демонстрирует изобретательностьего создателей по интеграцииаппаратных средств и компиляторадля статического и динамического устранения зависимостеймежду командами.

В процессорах с явно параллельным выполнениемкоманд количество одновременно функционирующих конвейеров от модели к модели растет, что можно объяснить постояннорастущими технологическими возможностями.

В архитектуре современных процессоров разныхкомпаний-производителей имеется много общего. В силу экономической целесообразно

фирмы вынуждены объединять свои финансовые, технологические и интеллектуальные ресурсы для удовлетворения постоянно растущего спроса на функциональные возможности и производительность новых моделей процессоров. Все это приводит к унификации архитектур.

Традиционно считалось, что основными направлениями развития компьютерной техники являются три сферы деятельности человека:

- автоматизация вычислений, отличительной особенностью которой является наличие хорошей математической основы и широкий спектр приложений;
- системы управления, где компьютеры должны не только обеспечивать вычисления, но и автоматизировать сбор данных и распределять результаты обработки;
- системы искусственного интеллекта, где преобладают робототехника, доказательство теорем, машинный перевод, обработка речи, моделирование сложных процессов.

Для эффективного технического обеспечения этих направлений требуются качественно новые архитектурные решения.

Вопросы для контроля

1. Что понимается под архитектурой компьютера?
2. В чем основные свойства алгоритма?
3. Назовите основные устройства компьютера?
4. Перечислите основные узлы компьютера и их функции.
5. Каковы характерные черты универсальных компьютеров?
6. Охарактеризуйте шестое поколение компьютеров.
7. К какой из архитектур Флинна относятся многоядерные процессоры?
8. Какой класс компьютеров имеет самое широкое применение?
9. В чем суть модульной организации персональных компьютеров?
10. Какие программы входят в состав комплекса BIOS?
11. Охарактеризуйте функции канала RS-232?
12. Какими параметрами определяется производительность компьютеров?
13. Какой процессор стал базой развития современных компьютеров?
14. Перечислите факторы, определяющие перспективы развития архитектуры компьютеров.

ГЛАВА 2. БАЗОВЫЕ ПРИНЦИПЫ ОРГАНИЗАЦИИ СИСТЕМ ОБРАБОТКИ

2.1. Архитектуры с общей и распределенной памятью

Различие между быстродействием процессора и памяти всегда было проблемой в однопроцессорных вычислительных машинах. Многопроцессорность приводит еще к одной проблеме - проблеме одновременного доступа к памяти со стороны нескольких процессоров. Разнообразие принципов взаимодействия процессоров с памятью приводит к различию в архитектурах построения компьютеров, что в свою очередь ведет к различию в реализации алгоритмов параллельной обработки при наличии нескольких процессоров и множества вариантов организации памяти.

Современные компьютеры, по существу, являются многопроцессорными вычислительными системами, так как имеют в своем составе, кроме основных процессоров, дополнительные средства обработки в виде видеопроцессоров, плат аудиообработки, сетевых адаптеров, процессоров плавающей запятой, контроллеров прямого доступа в память, контроллеров связи с внешними устройствами. Другими словами, современные компьютеры стали многокомпонентными системами обработки. Поэтому, ранее рассмотренный класс MIMD-систем был расширен, в него включены новые архитектуры компьютеров (рис. 2.1).

Рис.2.1. Архитектуры компьютеров класса MIMD

Для анализа входящих в класс MIMD архитектур компьютеров используется способ организации оперативной памяти. Данный подход позволяет различать два важных типа многопроцессорных вычислительных систем: мультипроцессоры или системы с общей

памятью, и мультикомпьютеры или системы с распределенной памятью.

В системах с общей памятью (ее часто называют также совместно используемой или разделяемой памятью) память вычислительной системы рассматривается как общий ресурс, и каждый из процессоров имеет полный доступ ко всему адресному пространству (рис.2.2). Подобное построение вычислительных систем имеет место как в классе SIMD, так и в массе MIMD.

Рис. 2.2. Архитектура системы с общей памятью

В варианте с распределенной памятью каждому из процессоров придается собственная память (рис.2.3). Процессоры объединяются в сеть и могут при необходимости обмениваться данными, хранящимися в их памяти, передавая друг другу так называемые сообщения. Такие слабо связанные системы встречаются как в классе SIMD, так и в классе MIMD.

Рис. 2.3. Архитектура системы с распределенной памятью

В некоторых случаях вычислительные системы с общей памятью называют мультипроцессорными, а системы с распределенной памятью — мулътикомпьютерными.

2.2. Симметричные мультипроцессорные системы

Архитектуры MIMD с совместно используемой общей памятью. Вычислительные системы с общей памятью, где доступ любого процессора к памяти производится одинаково и занимает одинаковое время, называются системами с однородным доступом к

памяти UMA (Uniform Memori Access). Это наиболее распространенная архитектура параллельных систем с общей памятью. Простейший путь построения таких систем - это объединение нескольких процессоров (Пр) с единой памятью посредством общей шины (рис.2.4). Однако, в этом случае в каждый момент времени обмен по шине может вести только один из процессоров, остальные должны ожидать, пока шина освободится.

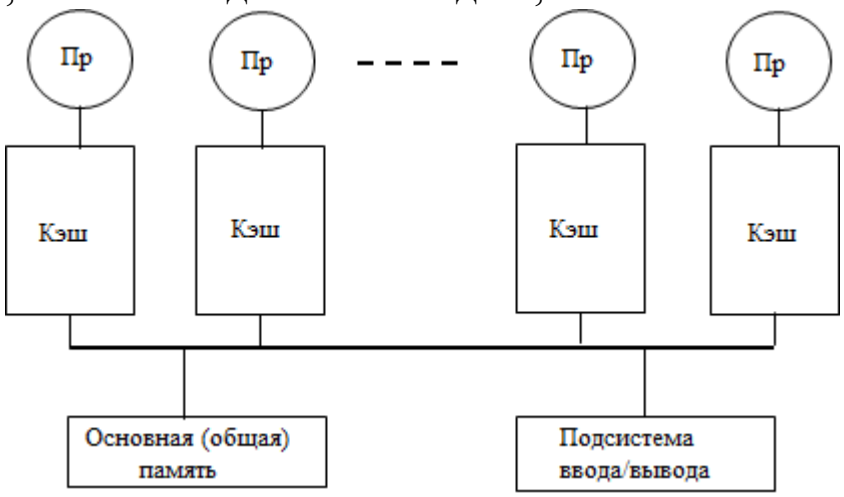


Рис.2.4. Архитектура компьютеров архитектуры UMA

Если в систему входят только два процессора, они в состоянии работать с производительностью, близкой к максимальной, поскольку их доступ к шине можно чередовать: пока один процессор декодирует и выполняет команду, другой вправе использовать шину для выборки из памяти следующей команды. Однако, когда добавляется третий процессор, производительность начинает падать. Память служит также для передачи сообщений между процессорами, при этом все вычислительные устройства при обращении к памяти имеют равные права и одну и те же адресацию для всей памяти. Вычислительные системы с такой архитектурой называются SMP - symmetric multiprocessing (симметричная многопроцессорная архитектура). Использование общей памяти увеличивает скорость обмена данными, пользователь имеет доступ ко всему объему памяти, что обеспечивает простоту программирования.

Однако с ростом числа процессоров снижается производительность, общая память и общая линия связи становятся причиной системных потерь времени.

Именно поэтому архитектура UMA не очень хорошо масштабируется. Наиболее распространенные системы содержат 4-8 процессоров, значительно реже 24-32 процессора. В системах большего масштаба используется технология NUMA.

2.3. Системы с неоднородным доступом к памяти (NUMA)

Другим подходом к построению компьютеров с общей памятью является **неоднородный доступ к памяти**, обозначаемый как NUMA (Non-Uniform Memory Access). Здесь по-прежнему фигурирует единое адресное пространство, где каждый процессор (Пn) имеет локальную память и узлы ввода-вывода (рис.2.5). Доступ процессора к собственной локальной памяти производится напрямую, что намного быстрее, чем доступ к удаленной памяти через коммуникационную среду. Такая система может быть дополнена глобальной памятью, тогда локальные запоминающие устройства играют роль быстрой кэш-памяти для глобальной памяти.

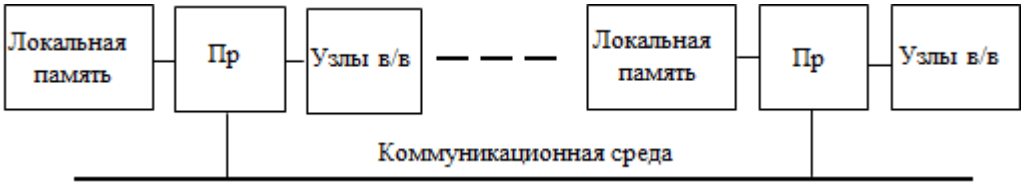


Рис.2.5. Типовая архитектура NUMA

При наличии у процессоров локальной кэш-памяти существует высокая вероятность того, что нужные команда или данные уже находятся в локальной памяти. Разумная вероятность попадания в локальную память существенно уменьшает число обращений процессора к глобальной памяти и, таким образом, ведет к повышению производительности.

2.4. Массивно-параллельные системы обработки

Модели архитектур распределенной памяти. В системе с распределенной памятью каждый процессор обладает собственной памятью и способен адресоваться только к ней. Некоторые авторы называют этот тип систем многомашинными вычислительными системами или мультикомпьютерами, подчеркивая тот факт, что блоки, из которых строится система, сами по себе являются небольшими вычислительными системами с процессором и памятью.

Модели архитектур с распределенной памятью принято обозначать как архитектуры без прямого доступа к удаленной памяти, каждый процессор имеет доступ только к своей локальной памяти. Доступ к удаленной памяти (локальной памяти другого процессора) возможен только путем обмена сообщениями с процессором, которому принадлежит адресуемая память.

Подобная организация характеризуется рядом достоинств. Во-первых, при доступе к данным не возникает конкуренции за шину или коммутаторы — каждый процессор может полностью использовать связь с собственной локальной памятью. Во-вторых, отсутствие общей шины означает, что нет и связанных с этим ограничений на число процессоров. В-третьих, снимается проблема когерентности кэш-памяти. Каждый процессор вправе самостоятельно менять свои данные, не заботясь о согласовании копий данных в собственной локальной кэш-памяти с кэшами других процессоров. Основной недостаток систем с распределенной памятью заключается в сложности обмена информацией между процессорами. Если какой-то из процессоров нуждается в данных из памяти другого процессора, он должен обменяться с этим процессором сообщениями. Такого рода архитектуры называются MPP-massive parallel processing (массивно-параллельная архитектура).

Это приводит к двум видам издержек:

- требуется время для того, чтобы сформировать и переслать сообщение от одного процессора к другому;
- для обеспечения реакции на сообщения от других процессоров принимающий процессор должен получить запрос прерывания и выполнить процедуру обработки этого прерывания.
- распространение получили два основных типа архитектурной организации массива процессорных элементов с распределенной памятью (рис.2.6).

В первом варианте, известном как архитектура типа «процессорный элемент-процессорный элемент» (ПЭ-ПЭ), где N процессорных элементов связаны между собой сетью соединений (рис.2.6, а), при этом каждый ПЭ — это процессор с свой независимой локальной памятью. Процессорные элементы выполняют команды, получаемые из контроллера по шине данных, обрабатывают данные

как хранящиеся в их локальной памяти, так и поступающие от контроллера. Обмен данными между процессорными элементами производится по сети соединений, в то время как шина ввода/вывода служит для обмена информацией между ПЭ и устройствами ввода/вывода.

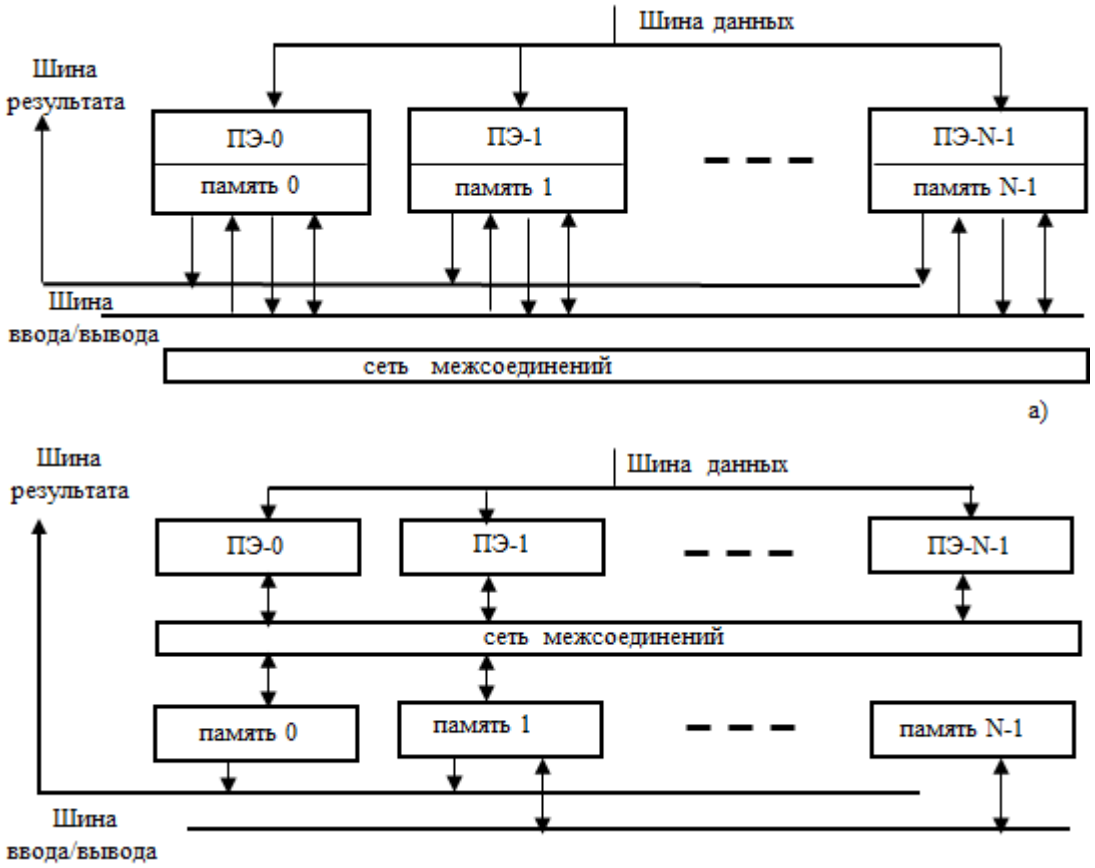


Рис.2.6. Организация массива процессоров с распределенной памятью

Второй вид архитектуры — «процессор-память» на рис.2.6,б. В такой конфигурации двунаправленная сеть соединений связывает N процессоров с M модулями памяти. Процессоры управляются контроллером через шину данных. Обмен данными между процессорами осуществляется как через сеть, так и через модули памяти. Пересылка данных между модулями памяти и устройствами ввода/вывода обеспечивается шиной ввода/вывода. Для передачи данных из конкретного модуля памяти в контроллер служит шина результата. В большинстве подобных систем используются технологии SIMD, а в качестве процессорных элементов применяются

простые RISC-процессоры с локальной памятью ограниченной емкости.

2.5. Векторные вычислительные системы

Основным отличием параллельных векторно-конвейерных систем PVP (Parallel Vector Process) является наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов и матриц. В таких вычислительных системах несколько (1-16) однотипных процессоров работают одновременно в режиме использования общей памяти в рамках многопроцессорных конфигураций. Программирование для данных систем подразумевает исполнение вычислительных циклов в виде векторных последовательностей для эффективной загрузки исполнительных процессоров. Кроме того, алгоритмы распараллеливания направлены на одновременную загрузку нескольких процессоров, реализующих прикладную задачу.

Векторно-конвейерный процессор содержит несколько конвейерных АЛУ, что позволяет параллельно исполнять смежные арифметико-логические операции. При этом широко используются внутренние регистры общего назначения (РОН) и специальные регистровые команды. Устройство управления отличается тем, что при выполнении векторной команды код операции не изменяется.

Векторные инструкции (команды программы) обладают рядом важных свойств по сравнению с обычной архитектурой набора инструкций. Одной векторной инструкцией определяется большой объем работы, она эквивалентна выполнению целого цикла, при этом диапазон извлекаемых и декодируемых инструкций существенно сужается.

Векторные архитектуры и компиляторы отличаются тем, что по сравнению с многопроцессорными MIMD-системами они существенно облегчают создание эффективных приложений при наличии параллелизма на уровне данных.

Проверку на наличие конфликтов данных оборудованию нужно вести только лишь между двумя векторными инструкциями, и делать это нужно для каждого векторного операнда, а не для каждого элемента внутри векторов. Поскольку весь цикл заменяется одной

векторной командой с установленным вычислительным алгоритмом, конфликты управления, которые, как правило, будут возникать из-за условных переходов в цикле, отсутствуют.

Экономия на диапазоне инструкций и на проверке наличия конфликтов плюс эффективное использование диапазона памяти дают векторной архитектуре преимущества по сравнению со скалярной архитектурой. По этим причинам векторные операции могут проводиться быстрее, чем последовательность скалярных операций с одинаковым количеством элементов данных, и разработчики имеют вполне серьезные основания для включения векторных блоков.

Необходимость столь подробного рассмотрения базовых принципов организации вычислений в сосредоточенных и распределенных системах объясняется тем, что современные процессоры представляют собой системы обработки с множеством вычислительных контроллеров, ядер и процессоров. В режимах функционирования современных компьютеров во многом повторяются правила взаимодействия процессоров и различных уровней памяти, аналогичные рассмотренным выше архитектурам. Кроме многочисленных узлов обработки в архитектуре современных компьютеров предусмотрено много уровней памяти разного объема и быстродействия. Кроме того, в структуре аппаратных решений имеется множество линий взаимодействия (шин) для обеспечения скоростных коммуникаций.

Поэтому, изучение базовых принципов распараллеливания вычислительного процесса с целью его ускорения является актуальной как для компьютерных систем, так и для отдельных компьютеров и их процессоров.

2.6. Типы и форматы аппаратно-поддерживаемых данных

В первом приближении информацию, используемую в компьютере, можно разделить на команды, адреса и данные. Под аппаратной поддержкой данных определенного типа, представленных в некотором формате, понимается наличие в системе таких команд, которые предназначены для обработки данных этого типа, представленных в соответствующем формате. Классификация аппаратно-поддерживаемых данных приведена на рис.2.7.

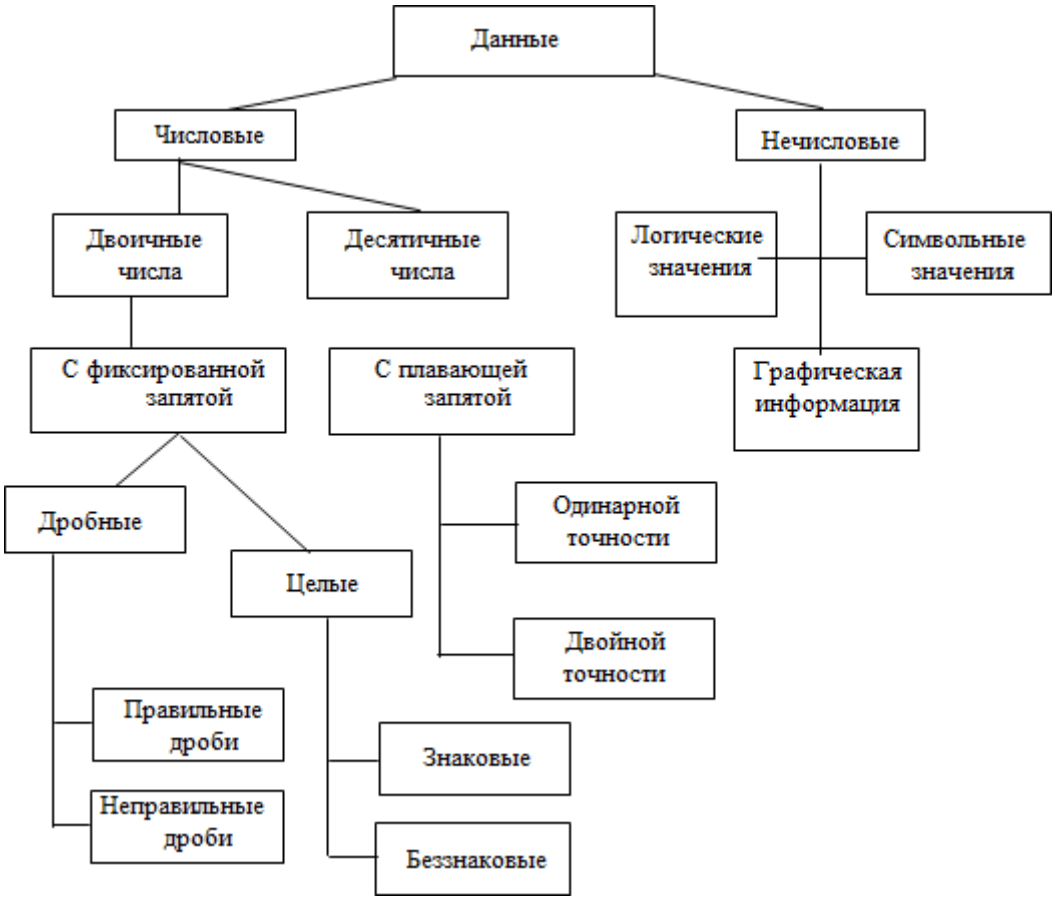


Рис.2.7. Классификация аппаратно-поддерживаемых данных

Числа с фиксированной запятой. Их деление на два типа (дробные и целые) определяется местоположением запятой в числе: слева (перед старшим разрядом) – дробные числа, справа (после младшего разряда) – целые числа. Дробные числа как таковые в современных компьютерах не используются. Они используются лишь для представления мантисс в числах с плавающей запятой. В правильных дробях целая часть чисел нулевая, в неправильных – не равна нулю.

Отличие знаковых и беззнаковых чисел состоит в интерпретации крайнего левого (старшего) бита числа. В знаковых целых числах он интерпретируется как знак числа (0 – "+", 1 – "-"), в беззнаковых целых числах – как старшая цифра числа. Особенностью представления знаковых целых чисел является использование дополнительного кода.

Числа с плавающей запятой. Форма с плавающей запятой используется для представления так называемых действительных чисел, которые в обобщенном виде представляются как комбинация

знака числа, его мантиссы, основания порядка и порядка числа. В классическом представлении мантисса числа представляется правильной дробью со старшей значащей цифрой. Порядок числа представляет собой целое число со знаком. В качестве основания порядка в современных компьютерах используется S = 2 (двоичное представление мантиссы) и S = 16 (16-ричное представление мантиссы).

Точность представления чисел с плавающей запятой. В общем случае числа с плавающей запятой представляются в ограниченном формате приближенно. Точность представления чисел принято характеризовать абсолютной и относительной погрешностью. Абсолютная погрешность является знаковой величиной и определяется как разность между приближенным (машинным) представлением числа и его точным значением.

В свою очередь относительная погрешность есть беззнаковая величина, определяемая как частное от деления абсолютной погрешности на точное значение самого числа. В отношении форматов точность представления, как правило, задается максимальной относительной погрешностью представления чисел, инвариантной к способу их округления.

При обработке данных с одинарной точностью используется длина слова компьютера (32,64 разряда). При работе с двойной точностью в одной операции длина слова используется дважды, то есть используется алгоритм обработки в два этапа.

Вопросы для контроля

1. Какие компьютеры входят в состав архитектуры MIMD?
2. В чем различие систем с общей и распределенной памятью?
3. В чем различие архитектур UMA и NUMA?
4. Назовите особенности векторно-конвейерных архитектур.
5. По каким признакам различают знаковые и без знаковые числа?
6. Чем определяется точность представления чисел с фиксированной и плавающей запятой?

ГЛАВА 3. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ И СИСТЕМА КОМАНД

3.1. Программное управление вычислительным процессом

Основной задачей центрального процессора является выборка команд из памяти (чтение команды), анализ типа выполняемой операции (дешифрации команды), выборка данных из памяти (чтение данных - операндов), выполнение команды (исполнение), занесение в память получаемых результатов (запись).

Таким образом, работа процессора состоит из пяти основных этапов:

- чтение команды;
- дешифрация команды;
- чтение операндов;
- исполнение команды;
- запись результатов.

Все действия процессора реализуются с помощью команд. Команда – это совокупность двоичных кодовых слов, содержащих информацию о том, какое действие нужно выполнить (код операции), каким образом определяется местоположение операндов в основной памяти (система адресации), где расположены операнды (адресная часть). Формат двухадресной команды приведен на рис.3.1.

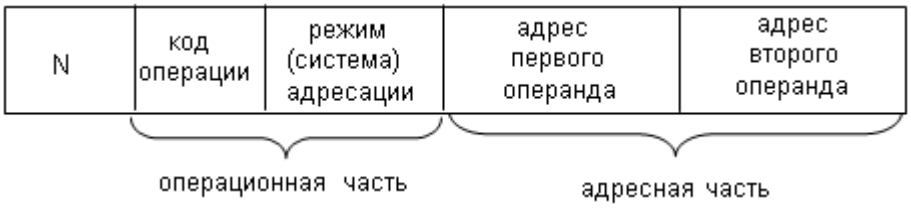


Рис.3.1. Формат команды компьютера

Команда, как и данные, хранится в основной памяти, поэтому N на рисунке обозначает номер (адреса) ячейки памяти, где она расположена и откуда она читается процессором. После чтения команды начинается процесс ее исполнения функциональными узлами компьютера. Для правильного понимания процесса исполнения команд, а значит и программ обработки, рассмотрим организацию вычислений через взаимодействия четырех основных блоков: узла управления, АЛУ, основной памяти и модуля ввода/вывода, объединяющего внешние порты. Внешняя память

непосредственного участия в процессе исполнения команд не принимает, т.к. необходимые в данной программе данные заданные переписываются в основную память или кэш-память.

Первые две части формата команды составляют операционную половину, две последующие - адресную половину команды. В общем случае адресная часть может содержать 4 адреса (адрес операнда 1, адрес операнда 2, адресация результата; адрес следующей команды). Быстрее выполняются команды с меньшим количеством адресов. Адресная часть содержит адреса операндов в оперативной памяти или номера РОН, откуда читаются операнды и куда пишутся результаты.

При исполнении команды ее отдельные части попадают в различные исполнительные устройства: оперативную память (ОП), устройство управления (УУ), АЛУ, модули ввода/вывода (МВВ), счетчики, регистры команд и данных. Адресная часть команды через регистр адреса поступает в память для чтения операндов и их направления в АЛУ.

Программа обработки данных реализуется процессором путем последовательного исполнения поступающих из памяти команд. Вся цепочка последовательного исполнения от момента формирование адреса самой команды до записи результатов ее исполнения называется циклом команды. Стандартный цикл команды включает в себя все основные этапы ее исполнения, изложенные выше, но для правильного понимания режимов обработки цикл представляется в развернутом виде с обозначением устройств, участвующих в исполнении (рис.3.2):

- формирование адреса самой команды (УУ),
- выборка (чтение) команды из памяти (ОП, УУ),
- декодирование кода операции и вида адресации (УУ),
- формирование адресов операндов (УУ, ОП),
- выборка операндов (ОП, УУ, АЛУ),
- выполнение действий над операндами (АЛУ),
- запись операнда результата (ОП),
- проверка запроса прерывания (УУ, АЛУ, МВВ),
- формирование адреса следующей команды (УУ).

В некоторых командах (например, прием-передача) отдельные шаги стандартного цикла могут отсутствовать, в то же время

увеличивается число шагов по управлению: проверка готовности внешнего устройства (ВУ), проверка запросов на прерывание.

Как видно из перечня шагов цикла команды основное взаимодействие по внутренним и внешним шинам идешь по линиям «процессор – ОП» и «процессор – МВВ» Через модуль МВВ осуществляется не только прием – передача между процессором и внешними устройствами, но и принимаются запросы на прерывание работающей программы от ВУ.

На рис. 3.2. представлена частота взаимодействия УУ с основными узлами процесса, участвующими в выполнении одного цикла команды. Основным узлом, контролирующим весь ход цикла выполнения, является УУ. В соответствии с вышеприведенным перечнем, только в одном цикле выполнения УУ как минимум 4 – 5 раз взаимодействует с основной памятью: передача адреса команды – чтение самой команды – передача адреса операнда – выборка операнда – запись результата.

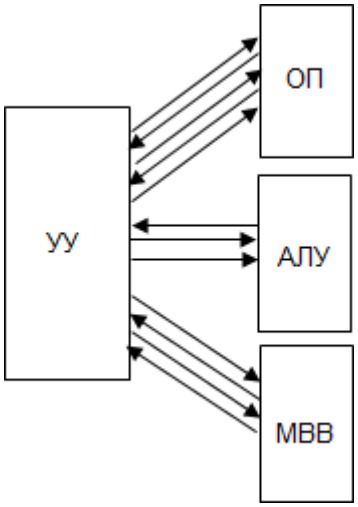


Рис.3.2. Реализация цикла команды

В ранних моделях процессоров с раздельными кэшами L1, L2 и ОП, как правило, команды читаются из кэш–памяти, а операнды (массивы данных) из ОП. В связи с таким порядком изменяются только уровни взаимодействия («регистр – регистр», «регистр – память»), но отдельные шаги по исполнению сохраняются. Именно такая высокая плотность взаимодействия ОП при реализации каждой команды объясняет необходимость наличия высокоскоростной локальной шины «процессор – память». В более поздних моделях процессоров кэш-память разделена на кэш-команд и кэш-данных.

Взаимодействие УУ с АЛУ не столь интенсивное как с ОП: запись операндов – выполнение действия по обработке – выдача результата в память. Но есть еще один такт, который обязателен для УУ – это чтение состояния «регистра флагов» или «регистра признаков» (РП). Этот регистр сигнализирует о нештатных ситуациях в АЛУ при выполнении вычислений (нулевой результат, переполнение). Наличие «1» в разрядах РП сигнализирует о необходимости прерывания действующего порядка следования команд (сигнал прерывания).

Такой же сигнал на прерывание может поступить из МВВ, когда одно из ВУ через МВВ выдает ранее упомянутые сигналы извещения о готовности к приему данных от процессора. Сигнал прерывания может поступить и от относительно медленных ВУ, например, от клавиатуры или мышки по действию пользователя.

3.2 Архитектура системы команд компьютера

Система команд – это полный перечень реализуемых компьютером команд, с помощью которых программист реализует свой алгоритм обработки данных, т.е. решает свою задачу. Набор команд, их разнообразие, по сути, определяют возможности компьютера его функциональную ориентацию.

Применение языков высокого уровня значительно облегчает процесс составления программ, но для их отладки и реализации требуются значительные ресурсы компьютера для хранения, компиляции и исполнения. Между обычными машинными операциями и операторами языков высокого уровня существует семантический разрыв, ведущий к увеличению непрямых затрат ресурсов, что в свою очередь ведет к понижению эффективности выполнения программы на компьютере.

С целью увеличения производительности при сохранении уровней программирования разработчики ищут пути совершенствования архитектуры системы команд. В настоящее время получили свое развитие три типа архитектур системы команд:

- традиционная CISC- архитектура (Complex Instruction Set Computer) с наиболее полным набором команд;

- RISC – архитектура (Reduced Instruction Set Computer) с сокращенным набором команд;

- VLIW – архитектура (Very Long Instruction Word) с командами сверхбольшой длины.

До середины 80–х годов почти все компьютеры имели CISC–архитектуру, которой придерживалась компания IBM в своих майнфреймах с мощными вычислительными ресурсами, а также компания Intel в своих моделях i8086 и Pentium. Для CISC-архитектуры типичны:

- наличие в процессоре сравнительно небольшого числа регистров общего назначения;
- большое количество машинных команд, некоторые из них аппаратно реализуют сложные операторы языков высокого уровня;
- разнообразие способов адресации операндов;
- множество форматов команд различной разрядности;
- наличие команд, где обработка совмещается с обращением к памяти.

К типу CISC можно отнести практически все компьютеры, выпускавшиеся до середины 1980-х годов, и значительную часть производящихся в настоящее время.

Рассмотренный способ решения проблемы семантического разрыва вместе с тем ведет к усложнению аппаратуры, главным образом устройства управления, что, в свою очередь, негативно сказывается на производительности в целом. Опыт применения и использования возможностей CISC–архитектур показали, что мощная система команд чаще всего полностью не используется, аппаратные средства, участвующие в исполнении многих команд, используются неэффективно. Микропрограммная память, которая формирует сигналы управления, получается слишком громоздкой. Процент постоянно, интенсивно используемых команд оказывается небольшим. Так как каждая команда преобразуется в последовательность микроинструкцией для определенных функциональных узлов, то устройство управления получается весьма сложным и медленным в работе.

Для компьютеров с CISC–архитектурой характерны не только большое количество самих типов и модификаций, но и разнообразие

способов адресации и форматов команд различной разрядности. Такой подход оправдан для универсальных компьютеров широкого применения со значительными стоимостными показателями.

Был предпринят комплекс исследований, в результате которых обнаружилось, что доля дополнительных команд, эквивалентных операторам языков высокого уровня, в общем объеме программ не превышает 10-20%, а для некоторых наиболее сложных команд даже 0,2%. В то же время объем аппаратных средств, требуемых для реализации дополнительный команд, возрастает весьма существенно. Так, емкость микропрограммной памяти при поддержании сложных команд может увеличиваться на 60%.

Это привело к серьезному пересмотру традиционных решений, следствием чего стало появление RISC-архитектуры. Термин RISC впервые был использован Д. Паттерсоном в 1980 году. Идея заключается в ограничении списка команд наиболее часто используемыми простейшими командами, оперирующими данными, размещенными только в регистрах процессорах. Обращение к памяти допускается лишь с помощью специальных команд чтения и записи. Резко уменьшено количество форматов команд и способов указания адресов операндов. Сокращение числа форматов команд и их простота, использование ограниченного количества способов адресации, отделение операций обработки' данных от операций обращения к памяти позволило существенно упростить аппаратные средства и повысить их быстродействие.

RISC–архитектура появилась как альтернатива CISC–архитектуре с точки зрения увеличения быстродействия. Сокращенный набор команд обеспечивает исполнение многих команд за один такт процессора, при этом команды работают с операндами, размещаемыми во внутренних регистрах процессора (РОН). В RISC–процессорах уменьшено число форматов команд, соответственно ограничены способы адресации. Для упрощения выполнения большинства команд и сведения их операции «регистр - регистр» компьютер должен располагать значительным числом РОН. В современных RISC–процессорах минимальное число РОН составляет 32. В архитектуре RISC удалось достичь высокого быстродействия за счет:

- стандартной для всех команд длины слова,
- исполнения большинства команд (более 75%) за один цикл,
- ограничения общего числа команд (до 128);
- малого количества форматов команд (не более 4);
- малого числа способов адресации (не более 4);
- обеспечение доступа к ОП только через команды «Чтение», «Запись» (другие команды используют для хранения в РОН, при этом общее число РОН может превышать несколько сотен).

Элементы RISC-архитектуры впервые появились в супер -ЭВМ компании Cray Research. Достаточно успешно реализуется RISC-архитектура и в современных процессорах Alpha фирмы DEC, серии PA фирмы Hewlett-Packard, семействе PowerPC.

Концепция VLIW базируется на RISC-архитектуре, где несколько простых RISC-команд объединяются в одну сверхдлинную команду и выполняются параллельно. Архитектура VLIW сравнительно мало отличается от RISC, появился лишь дополнительный уровень параллелизма вычислений, в силу чего архитектуру VLIW логичнее адресовать не к вычислительным машинам, а к вычислительным системам. Идея процессоров VLIW базируется на том, что возможно одновременное исполнения несколько команд на параллельно работающих функциональных блоках (упрощенных вариантах АЛУ).

Для того, чтобы работа блоков по времени не пересекалась усовершенствованный компилятор в процессе обработки исходной программы выделяет те команды, которые, принципе могут быть реализованы параллельно. Выделив такие команды, компилятор может их объединить в одну сверхдлинную команду, состоящую из нескольких простых и реализуемых параллельно. Количество команд, объединяемых в одну сверхдлинную, должно быть равно количеству исполнительных функциональных блоков. Длина VLIW- команды составляет 256 - 1024 бит, т.е. от 32 до 128 байт, она содержит несколько полей по числу составляющих простых команд.

В качестве простых команд, образующих сверхдлинную команду обычно используются команды RISC-типа, поэтому VLIW-архитектура считается дальнейшим развитием RISC–систем. Число полей команды в сверхдлинной команде равно числу реализующих их

устройств и колеблется в пределах от 3 до 20. Упрощенная схема доступа к данным, организованным в виде одного регистрового файла, со стороны функциональных блоков, реализующих одну команду, значительно упрощает процессы адресации и чтения операндов.

Узким местом VLIW-архитектуры является сложность программы-компилятора, способного исследовать исходную программу, найти и объединить в длинные командные слова взаимозависимые инструкции. Увеличение сложности транслирующей программы увеличивает время формирования сверхдлинной команды. Программисты к внутренним VLIW–командам доступа не имеют, все программное обеспечение базируется на низкоуровневых программах трансляции команд CISC – процессоров в команды VLIW.

Классификация архитектуры по месту хранения операндов.

Количество команд и их сложность являются важными факторами, однако не меньшую роль играет место хранения операндов и способ доступа. С этих позиций различают следующие виды архитектур системы команд:

- стековую организацию команд;
- аккумуляторную организацию;
- регистровую;
- с выделенным доступом к памяти.

Выбор той или иной архитектуры влияет на принципиальные моменты: сколько адресов будет содержать адресная часть команд, какова будет длина этих адресов, насколько просто будет происходить доступ к операндам и какой, в конечном итоге, будет общая длина команд.

Стековая архитектура. Стеком называется память, которая формируется как множество логически взаимосвязанных ячеек (рис.3.3), взаимодействующих по принципу «последним вошел, первым вышел» (LIFO, Last In First Out).

Верхнюю ячейку называют вершиной стека. Запись возможна только в верхнюю ячейку стека, при этом все хранящиеся в стеке команды предварительно проталкиваются на одну позицию вниз. Чтение допустимо также только из вершины стека. Извлеченная

команда удаляется из стека, а оставшееся его содержимое продвигается вверх.

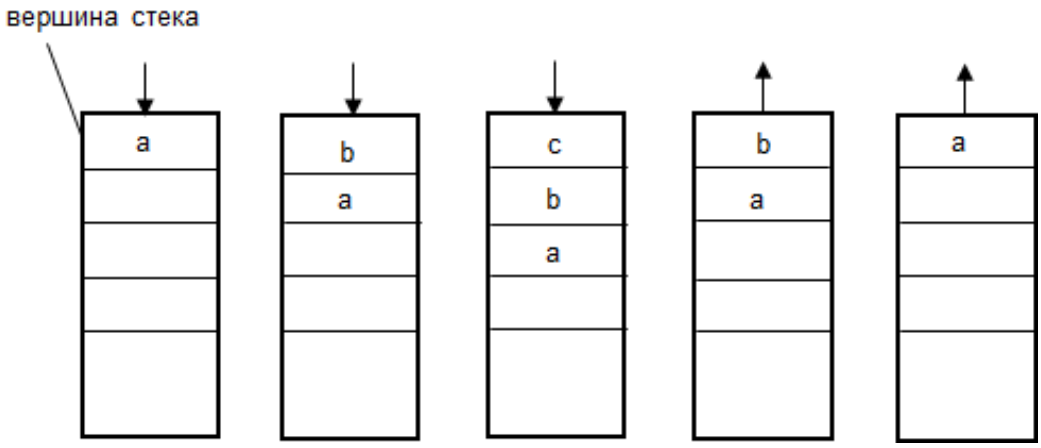


Рис. 3.3. Принцип действия стековой памяти

Аккумуляторная архитектура. Архитектура на базе аккумулятора исторически возникла одной из первых. В ней для хранения результата арифметической или логической операции в процессоре имеется выделенный регистр — аккумулятор. Это один из узлов операционного устройства - АЛУ. Изначально оба операнда хранятся в основной памяти, и до выполнения операции один из них нужно загрузить в аккумулятор. Для выполнения следующей операции достаточно считать из памяти второй операнд. После выполнения команды обработки результат находится в аккумуляторе и, если он не является операндом для последующей команды, его требуется отправить в ячейку памяти.

Типичная архитектура на базе аккумулятора показана на рис. 3.4.

По команде загрузки информация считывается из ячейки памяти, выход памяти подключается к входам аккумулятора и происходит занесение считанных данных в аккумулятор. Запись содержимого аккумулятора в ячейку осуществляется командой, при выполнении которой выходы аккумулятора подключаются к шине, после чего информация с шины записывается в память.

Для выполнения операции в АЛУ производится считывание одного из операндов из памяти в регистр данных. Второй операнд находится в аккумуляторе. Выходы регистра данных и аккумулятора подключаются к соответствующим входам АЛУ. По окончании арифметической или логической операции результат с выхода АЛУ заносится в аккумулятор.

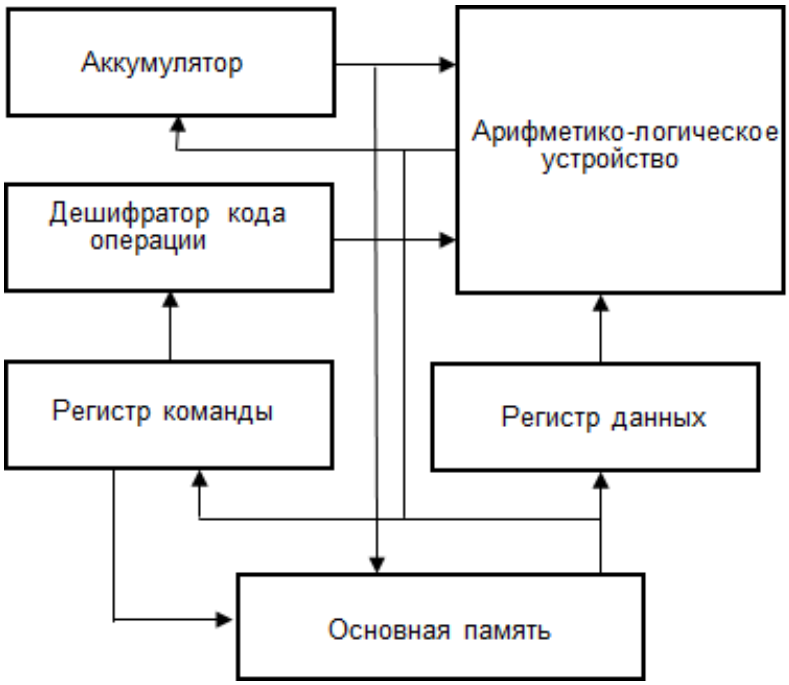


Рис.3.4. Реализация аккумуляторной архитектуры

Достоинствами аккумуляторной АСК можно считать короткие команды и простоту декодирования команд. Однако наличие всего одного регистра порождает многократные обращения к основной памяти. АСК на базе аккумулятора была популярна в ранних моделях компьютеров (IBM 7090, DEC PDP-8).

Регистровая архитектура. В машинах данного типа процессор включает в себя массив регистров общего назначения (РОН). Эти регистры можно рассматривать как кэш-память для хранения недавно использовавшихся данных.

Размер регистров обычно фиксирован и совпадает с размером машинного слова. К любому регистру можно обратиться, указав его номер. Количество РОН в архитектурах типа CISC обычно невелико (от 8 до 32), и для представления номера конкретного регистра необходимо не более пяти разрядов, благодаря чему в адресной части команд обработки допустимо одновременно указать номера двух, а зачастую и трех регистров (двух регистров операндов и регистра результата).

Регистровая архитектура допускает расположение операндов в одной из двух запоминающих сред: основной памяти или регистрах. С учетом возможного размещения операндов в рамках регистровых архитектур выделяют три подвида команд обработки:

- регистр-регистр;
- регистр-память;
- память-память.

В варианте «регистр-регистр» операнды могут находиться только в регистрах. В них же засылается и результат. Подвид «регистр-память» предполагает, что один из операндов размещается в регистре, а второй в основной памяти. Результат обычно замещает один из операндов. В командах типа «память-память» оба операнда хранятся в основной памяти. Результат заносится в память.

Вариант «регистр-регистр» является основным в вычислительных машинах типа RISC. Команды типа «регистр-память» характерны для CISC-машин. Наконец, вариант «память-память» считается неэффективным, хотя и остается в наиболее сложных моделях машин класса CISC.

Возможную структуру и информационные тракты вычислительной машины с регистровой архитектурой команд иллюстрирует рис. 3.5.

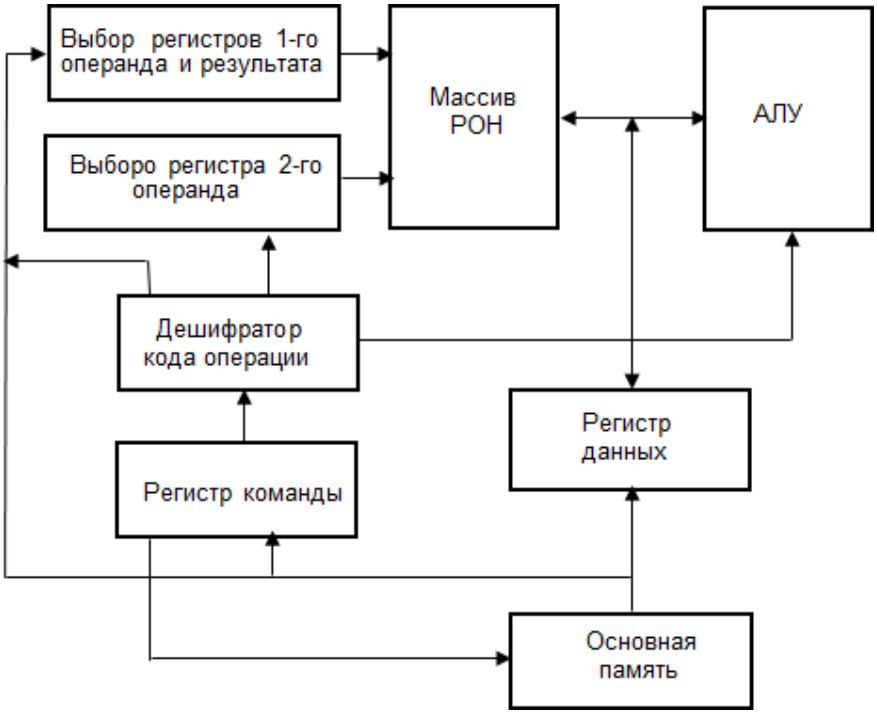


Рис.3.5. Регистровая архитектура

Операции загрузки регистров из памяти и сохранения содержимого регистров в памяти идентичны таким же операциям с

аккумулятором. Отличие состоит в этапе выбора нужного регистра, обеспечиваемого соответствующими селекторами.

Выполнение операции в АЛУ включает в себя:

- выбор регистра первого операнда;
- определение расположения второго операнда (память или регистр);
- подачу на вход АЛУ операндов и выполнение операции;
- выбор регистра результата и занесение результата операции из АЛУ.

Архитектура с выделенным доступом к памяти. В этой архитектуре обращение к основной памяти возможно только с помощью двух специальных команд: ***load*** *и* ***store***. В английской транскрипции данную архитектуру называют Load/Store architecture. Команда ***load*** (загрузка) обеспечивает считывание значения из основной памяти и занесение его в регистр процессора (в команде обычно указывается адрес ячейки памяти и номер регистра). Пересылка информации в противоположном направлении производится командой ***store*** (сохранение). Операнды во всех командах обработки информации могут находиться в РОН. Результат операции также заносится в регистр. В архитектуре отсутствуют команды обработки, допускающие прямое обращение к основной памяти. При выборке данных из основной памяти регистр данных не используется.

3.3. Типы и форматы операндов

Машинные команды оперируют данными, которые в этом случае принято называть **операндами**. К наиболее общим (базовым) типам операндов можно отнести: адреса, числа, символы и логические данные. Помимо них ВМ обеспечивает обработку и более сложных информационных единиц: графических изображений, аудио, видео и анимационной информации. Такая информация является производной от базовых типов данных и хранится в виде файлов на внешних запоминающих устройствах. Для каждого типа данных в ВМ предусмотрены определенные форматы.

Числовая информация. Среди цифровых данных можно выделить две группы:

- целые типы, используемые для представления целых чисел;
- вещественные типы для представления рациональных чисел.

В рамках первой группы имеется несколько форматов представления численной информации, зависящих от ее характера. Для представления вещественных чисел используется форма с плавающей запятой.

Числа в форме с фиксированной запятой.

Представление числа в форме с фиксированной запятой (ФЗ), которую иногда называют также естественной формой (рис.4.6), включает в себя знак числа и его модуль в q-ичном коде. Здесь q - основание системы счисления. Для современных компьютеров характерна двоичная система $(q = 2)$, но иногда используются также восьмеричная $(q = 8)$ или шестнадцатеричная $(q = 16)$ системы счисления. Запятую в записи числа называют соответственно двоичной, восьмеричной или шестнадцатеричной. Знак положительного числа кодируется двоичной цифрой 0, знак отрицательного числа - цифрой 1.

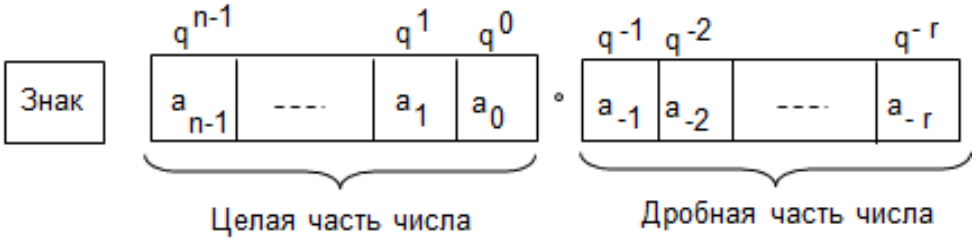


Рис.3.6. Формат чисел с фиксированной запятой

При фиксации запятой перед старшим цифровым разрядом могут быть представлены только правильные дроби. Фиксация запятой перед старшим разрядом встречалась в ряде машин второго поколения, но в настоящее время практически отжила свое.

При фиксации запятой после младшего разряда представимы лишь целые числа. Это наиболее распространенный способ, поэтому в дальнейшем понятие ФЗ будет связываться исключительно с целыми числами, а операции с числами в форме ФЗ будут характеризоваться как целочисленные. Здесь также возможны числа со знаком и без знака. Целочисленные форматы с фиксированной запятой приняты в микропроцессорах фирмы Intel. Целые числа применяются также для работы с адресами.

Представление чисел в формате ФЗ упрощает аппаратурную реализацию компьютера и сокращает время выполнения машинных операций, однако при решении задач необходимо постоянно следить за тем, чтобы все исходные данные, промежуточные и окончательные результаты не превышали допустимый диапазон формата, иначе возможно переполнение разрядной сетки и результат вычислений будет неверным.

Упакованные целые числа. В системе команд современных процессоров имеются команды, оперирующие целыми числами, представленными в упакованном виде. Связано это с обработкой мультимедийной информации, в основном, изображений. Формат предполагает упаковку в пределах достаточно длинного слова (обычно 64 -разрядного) нескольких небольших целых чисел, а соответствующие команды обрабатывают все эти числа параллельно. Если каждое из чисел состоит из четырех двоичных разрядов, то в 64-разрядное слово можно поместить до 16 таких чисел. Неиспользованные разряды заполняются нулями.

В микропроцессорах фирмы Intel, начиная с Pentium MMX, присутствуют специальные команды для обработки мультимедийной информации (MMX-команды), оперирующие целыми числами, упакованными в квадрослова (64-разрядные слова). Предусмотрены три формата: упакованные байты (восемь 8-разрядных чисел); упакованные слова (четыре 16-разрядных числа) и упакованные двойные слова (два 32-разрядных числа).

Десятичные числа. В ряде задач, главным образом, учетно-статистического характера, приходится иметь дело с хранением, обработкой и пересылкой десятичной информации. Особенность таких задач состоит в том, что обрабатываемые числа могут состоять из различного и весьма большого количества десятичных цифр. Традиционные методы обработки с переводом исходных данных в двоичную систему счисления и обратным преобразованием результата зачастую сопряжены с существенными накладными расходами. По этой причине в компьютерах применяются иные специальные формы представления десятичных данных. В их основу положен принцип кодирования каждой десятичной цифры эквивалентным двоичным числом из четырех битов (тетрадой), то

есть так называемым двоично-десятичным кодом (BCD - Binary Coded Decimal). Из задействованных шести четырехразрядных двоичных комбинаций ($2^4 = 16$) две служат для кодирования знаков «+» и «-».

Числа в форме с плавающей запятой. От недостатков ФЗ в значительной степени свободна форма представления чисел *с* плавающей запятой (ПЗ), известная также под названиями нормальной или полулогарифмической формы. В данном варианте каждое число разбивается на две группы цифр. Первая группа цифр называется ***мантиссой,*** вторая - ***порядком.*** Число представляется в виде произведения $X = \pm m\ q^{\pm P}$, где m — мантисса числа, p — порядок числа, q — основание системы счисления.

Для представления числа в форме с ПЗ требуется задать знаки мантиссы и порядка, их модули в двоичном коде, а также основание системы счисления (рис.3.7). Нормальная форма неоднозначна, так как взаимное изменение m и приводит к «плаванию» запятой, чем и обусловлено название этой формы.

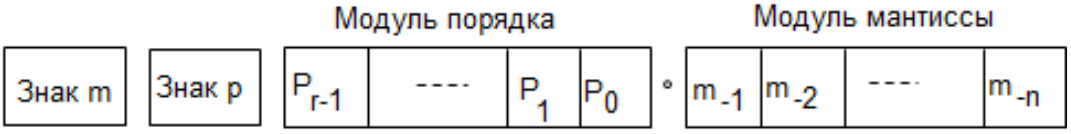


Рис.3.7. Формат представления чисел с плавающей запятой

Диапазон и точность представления чисел с ПЗ зависят от числа разрядов, отводимых под порядок и мантиссу.

Рекомендуемый для всех компьютеров формат представления чисел с плавающей запятой определен стандартом IEEE 754. Этот стандарт был разработан с целью облегчить перенос программ с одного процессора на другие, и нашел широкое применение.

Упакованные числа с плавающей запятой. В микропроцессорах фирмы Intel, начиная с Pentium III, для мультимедийных целей поддержаны команды, реализующие технологию SSE, также ориентированную на параллельную обработку упакованных чисел с ПЗ. Здесь числа объединяются в группы длиной 128 бит, и это позволяет упаковать в одну группу четыре 32-разрядных числа с ПЗ с одинарной точностью. Позже, в технологии SSE2, которую можно считать дальнейшим развитием SSE, появился формат, где в группу из 128 бит упаковываются два 64-разрядных числа с ПЗ, то есть числа, представленные с двойной точностью.

Символьная информация. В общем объеме вычислительных действий все большая доля приходится на обработку символьной информации, содержащей буквы, цифры, знаки препинания, математические и другие символы. Каждому символу ставится в соответствие определенная двоичная комбинация. Совокупность возможных символов и назначенных им двоичных кодов образует **таблицу кодировки.** В настоящее время применяется множество различных таблиц кодировки. Объединяет их весовой принцип, при котором веса кодов цифр возрастают по мере увеличения цифры, а веса символов увеличиваются в алфавитном порядке. Так вес буквы «Б» на единицу больше веса буквы «А». Это способствует упрощению обработки. До недавнего времени наиболее распространенными были кодовые таблицы, в которых символы кодируются с помощью восьмиразрядных двоичных комбинаций (байтов), позволяющих представить 256 различных символов.

Современный стандартный код ASCII—7-разрядный, восьмая позиция отводится для записи бита четности. Это обеспечивает представление 128 символов, включая все латинские буквы, цифры, знаки основных математических операций и знаки пунктуации. Позже появилась европейская модификация ASCII, в которой используются все 8 разрядов. Дополнительные комбинации (коды 128-255) в новом варианте отводятся для представления специфических букв алфавитов западно-европейских языков, символов псевдографики, некоторых букв греческого алфавита, а также ряда математических и финансовых символов. Именно эта кодовая таблица считается мировым стандартом де-факто, который применяется с различными модификациями во всех странах.

Хотя код ASGII достаточно удобен, он все же слишком тесен и не вмещает множества необходимых символов. По этой причине в 1993 году консорциумом компаний Apple Computer, Microsoft, Hewlett-Packard, DEC и IBM был разработан 16-битовый, стандарт ISO 10646, определяющий универсальный набор символов (UCS, Universal Character Set). Новый код, известный под названием Unicode, позволяет задать до 65 536 символов, то есть дает возможность одновременно представить символы всех основных «живых» и «мертвых» языков.

Видеоинформация. Видеоинформация бывает как статической, так и динамической. Статическая видеоинформация включает в себя текст, рисунки, графики, чертежи, таблицы и др. Рисунки делятся также на плоские — двумерные и объемные — трехмерные.

Динамическая видеоинформация — это видео, мульт и слайд-фильмы. В их основе лежит последовательное экспонирование на экране в реальном масштабе времени отдельных кадров в соответствии со сценарием. Динамическая информация используется либо для передачи движущихся изображений (анимация), либо для последовательной демонстрации отдельных кадров (слайд-фильмы).

В вычислительной технике существует два способа представления графических изображений: **матричный (растровый)** и **векторный.** Матричные (bitmap) форматы хорошо подходят для изображений со сложными гаммами цветов, оттенков и форм, таких как фотографии, рисунки, отсканированные данные. Векторные форматы более приспособлены для чертежей и изображений с простыми формами, тенями и окраской.

В матричных форматах изображение представляется прямоугольной матрицей точек — **пикселов** (picture element), положение которых в матрице соответствует координатам точек на экране. Помимо координат каждый пиксел характеризуется своим цветом, цветом фона или градацией яркости. Количество битов, выделяемых для указания цвета пиксела, изменяется в зависимости от формата. В высококачественных изображениях цвет пиксела описывают 24 битами, что дает около 16 млн. цветов. Основной недостаток матричной (растровой) графики заключается в большой емкости памяти, требуемой для хранения изображения, из-за чего для описания изображений прибегают к различным методам сжатия данных. В настоящее время существует множество форматов графических файлов, различающихся алгоритмами сжатия и способами представления матричных изображений, а также сферой применения.

Векторное представление, в отличие от матричной графики, определяет описание изображения не пикселами, а кривыми - сплайнами. Сплайн - это гладкая кривая, которая проходит через

опорные точки, управляющие формой сплайна. В векторной графике наиболее распространены сплайны на основе кривых Безье.

Хотя это может показаться более сложным, но для многих видов изображений использование математических описаний является более простым способом. В векторной графике для описания объектов используются математические формулы. Это позволяет при рисовании объектов вычислять, куда необходимо помещать реальные точки изображения. Имеется ряд простейших объектов, или **примитивов,** например эллипс, прямоугольник, линия. Эти примитивы и их комбинации служат основой для создания более сложных изображений. В простейшем случае изображение может быть составлено из отрезков линий, для которых задаются начальные координаты, угол наклона, длина, толщина линии, цвет линии и цвет фона.

Основное достоинство векторной графики - описание объекта, является простым и занимает малый объем памяти. Векторная графика в сравнении с матричной имеет следующие преимущества:

- простота масштабирования изображения без ухудшения его качества;

- независимость емкости памяти, требуемой для хранения изображения, от выбранной цветовой модели.

Недостатком векторных изображений является их искусственность, заключающаяся в том, что любое изображение необходимо разбить на конечное множество составляющих его примитивов. Как и для матричной графики, существует несколько форматов графических векторных файлов.

Аудиоинформация. Понятие «аудио» связано со звуками, которые способно воспринимать человеческое ухо. Частоты аудиосигналов лежат в диапазоне от 15 Гц до 20 КГц, а сигналы по своей природе являются непрерывными (аналоговыми). Прежде чем быть представленной в компьютере, аудиоинформация должна быть преобразована в цифровую форму (оцифрована). Для этого значения звуковых' сигналов (выборки, samples), взятые через малые промежутки времени, с помощью аналого-цифровых преобразователей (АЦП) переводятся в двоичный код. Обратное действие выполняется цифро-аналоговыми преобразователями

(ЦАП). Чем чаще производятся выборки, тем выше может быть точность последующего воспроизведения исходного сигнала, но тем большая емкость памяти требуется для хранения оцифрованного звука. Цифровой эквивалент аудиосигналов обычно хранится в виде файлов, причем широко используются различные методы сжатия такой информации. Как правило, к методам сжатия аудиоинформации предъявляется требование возможности восстановления непрерывного сигнала без заметного ухудшения его качества. В настоящее время распространен целый ряд форматов хранения аудиоинформации.

3.4. Типы команд процессора

Одна из наиболее важных абстракций — это интерфейс между аппаратной частью и программным обеспечением самого низкого уровня. В силу ее важности она имеет специальное название: архитектура набора команд, или просто архитектура компьютера. Архитектура набора команд содержит все, что нужно знать программисту для создания работоспособных программ на двоичном машинном языке, включая инструкции, характеристики устройства ввода-вывода. Как правило, операционная система будет скрывать в своих модулях подробности осуществления ввода-вывода, распределения памяти и выполнения других низкоуровневых системных функций, поэтому прикладным программистам беспокоиться не стоит. Сочетание основного набора инструкций и интерфейса операционной системы называется двоичным интерфейсом приложений (application interface, **API).**

В отличие от программ, написанных на языках программирования высокого уровня, операнды арифметических команд ограничены; они должны состоять из ограниченного количества специальных мест, построенных непосредственно в аппаратных средствах, называемых регистрами. Регистры легко используются в дизайне аппаратного обеспечения, который также очевиден для программиста, когда компьютер собран, так что вы можете думать о регистрах как о кирпичиках компьютерной конструкции. В структуре процессоров размер регистра составляет 32 бита; тридцати двух битные группы встречаются так часто, что их

называют «слово» в архитектуре системы машинных команд. Основное различие между переменными языка программирования и регистрами является ограниченное число регистров, обычно 32 на современных компьютерах.

Многие программы имеют больше переменных, чем количество регистров у компьютеров. Вследствие этого компилятор пытается сохранить наиболее часто используемые переменные в регистрах и остальное в памяти, используя загрузку и хранение для перемещения переменных между регистрами и памятью. Процесс помещения менее используемых переменных (или тех, которые понадобятся позже) в память называется сбросом регистров**.** Принцип аппаратного обеспечения, касающийся размера и скорости, предполагает, что память должна быть медленнее, чем регистры, так как регистров меньше. Это действительно так; данные доступны быстрее, если они расположены в регистрах, а не в памяти.

Команды традиционного машинного уровня можно разделить на несколько типов, которые показаны в таблице 3.1.

Табл. 3.1

Тип операции	**Примеры**
Арифметические и логические	Целочисленные арифметические и логические операции: сложение, вычитание, логические сложение, логическое умножение
Пересылка данных	Операции загрузки/записи
Управление потоком команд	Безусловные и условные переходы, вызов процедур и возвраты
Системные операции	Системные вызовы, команды управления виртуальной памятью
Операции с плавающей точкой	Операции сложения, вычитания, умножения и деления над вещественными числами
Десятичные операции	Десятичное сложение, умножение, преобразование форматов
Операции над строками	Пересылка, сравнения и поиск строк
SIMD-команды	Одна инструкция – много данных

Основную часть кода команды занимает адресная часть. Если процессор одноадресный, то этот единственный адрес указывает местоположение в памяти обрабатываемого операнда. Оттуда его необходимо считать, выполнить над ним действие и записать полученный результат по тому же адресу. Если процессор имеет двухадресную структуру команды, то оба адреса указывают расположение в памяти операндов, над которыми нужно произвести действие.

После действия результат записывается либо по одному из адресов, либо сохраняется на одном из внутренних регистров процессора. При трехадресном формате обычно два операнда выбираются по первому и второму адресу, а результат записывается по третьему адресу.

Так как процессор исполняет действия только на своем внутреннем машинном языке, перед запуском программы она должна транслироваться с мнемонической формы записи на машинный код. Именно эта программа трансляции и называется Ассемблер - служебная программа, хранимая в памяти и запускаемая в работу только при трансляции.

Одной команде языка ассемблера соответствует одна команда программы, что для сложных задач обработки не эффективно. Позже стали применяться более удобные записи программ на естественных языках пользователей (С, ПАСКАЛЬ).

Во всех трех случаях в качестве источников данных и приемников результатов выполненных операций могут выступать внутренние регистры процессора – аккумулятор, регистры данных, сегментные регистры.

Арифметические и логические команды. В данную группу входят команды, обеспечивающие арифметическую и логическую обработку информации в различных формах ее представления. Для каждой формы представления чисел в системе команд предусматривается набор стандартных операций.

Помимо вычисления результата выполнение арифметических и логических операций сопровождается формированием в АЛУ признаков (флагов), характеризующих этот результат. Наиболее часто фиксируются такие признаки, как: Z (Zero) - нулевой результат; N

(Negative) - отрицательный результат; V (Verflow) — переполнение разрядной сетки; C (Carry) — наличие переноса.

Команды пересылки данных. Это наиболее распространенный тип машинных команд. В таких командах должна содержаться следующая информация:

- адреса источника и получателя операндов — адреса ячеек памяти, номера регистров процессора или информация о том, что операнды расположены в стеке;
- длина подлежащих пересылке данных (обычно в байтах или словах), заданная явно или косвенно;
- способ адресации каждого из операндов, с помощью которого содержимое адресной части команды может быть пересчитано в физический адрес операнда.

Рассматриваемая группа команд обеспечивает передачу информации между процессором и ОП, внутри процессора и между ячейками памяти. Пересылочные операции внутри процессора имеют тип «регистр-регистр». Передачи между процессором и памятью относятся к типу «регистр-память», а пересылки в памяти — к типу «память-память».

Управление потоком команд. Концепция компьютера предполагает, что команды программы, как правило, выполняются в порядке их расположения в памяти. Для получения адреса очередной команды достаточно увеличить содержимое счетчика команд на длину текущей команды. В то же время основные преимущества процессора заключаются именно в возможности изменения хода вычислений в зависимости от возникающих в процессе счета результатов. С этой целью в систему команд компьютера включаются команды, позволяющие нарушить естественный порядок следования и передать управление в иную точку программы. В адресной части таких команд содержится адрес точки перехода (адрес той команды, которая должна быть выполнена следующей). Переход реализуется путем загрузки адреса точки перехода в счетчик команд (вместо увеличения содержимого этого счетчика на длину команды).

В системе команд компьютера можно выделить три типа команд, способных изменить последовательность вычислений:

- безусловные переходы;
- условные переходы (ветвления программ);
- вызовы процедур и возвраты из процедур.

Согласно приведенным данным, среди команд рассматриваемой группы доминируют условные переходы. Команда безусловного перехода обеспечивает переход по заданному адресу без проверки каких-либо условий.

Это может быть директива пользователя или заранее предусмотренное изменение хода выполнения программы (изменение режима в системах управления).

Условный переход происходит только при соблюдении определенного условия, в противном случае выполняется следующая по порядку команда программы. Условием, на основании которого осуществляется переход, чаще всего выступают признаки результата предшествующей арифметической или логической операции. Каждый из признаков фиксируется в нужном разряде регистра флагов процессора. Возможен и иной подход, когда решение о переходе принимается в зависимости от состояния одного из регистров общего назначения, куда предварительно помещается результат операции сравнения. Третий вариант — это объединение операций сравнения и перехода в одной команде.

В системе команд для каждого признака результата предусматривается своя команда ветвления (иногда — две: переход при наличии признака и переход при его отсутствии). Большая часть условных переходов связана с проверкой взаимного соотношения двух величин или с равенством (неравенством) некоторой величины нулю.

Команды управления системой. Команды, входящие в эту группу, являются привилегированными и могут выполняться, только когда центральный процессор компьютера находится в привилегированном состоянии или выполняет программу, находящуюся в привилегированной области памяти (обычно привилегированный режим используется лишь операционной ' системой). Так, лишь эти команды способны считывать и изменять состояние ряда регистров устройства управления.

Операции с числами в форме с плавающей запятой. Для работы с числами, представленными в форме с плавающей запятой, в системе команд большинства компьютеров предусмотрены:

- основные арифметические операции: сложение, вычитание, умножение и деление;
- операции сравнения, обеспечивающие сравнение двух вещественных чисел с выработкой признаков “больше“, “меньше”, “равно”;
- операции преобразования: формы представления (между фиксированной и плавающей запятой), формата представления (с одинарной и двойной точностью).

Операции с десятичными числами. Десятичные числа представляются в компьютере в двоично-кодированной форме. В вычислительных машинах первых поколений для обработки таких чисел предусматривались специальные команды, обеспечивавшие выполнение основных арифметических операций (сложение, вычитание, умножение и деление). В системе команд современных компьютеров машин подобных команд обычно нет, а соответствующие вычисления имитируются с помощью команд целочисленной арифметики.

Команды для работы со строками. Для работы со строками в архитектуре системы команд обычно предусматриваются команды, обеспечивающие перемещение, сравнение и поиск строк. В большинстве машин перечисленные операции просто имитируются за счет других команд.

SlMD-команды. В отличие от обычных команд, оперирующих двумя числами, SIMD-команды обрабатывают сразу две группы чисел (в принципе их можно называть групповыми командами). Операнды таких команд обычно представлены в одном из упакованных форматов.

С 1992 года команды типа SIMD становятся неотъемлемым элементом системы команд микропроцессоров фирм Intel и AMD. Поводом послужило широкое распространение мультимедийных приложений. Видео, трехмерная графика и звук в компьютерах представляются большими массивами данных, элементы которых чаще всего обрабатываются идентично. Так, при сжатии видео и

преобразовании его в формат MPEG один и тот же алгоритм применяется к тысячам битов данных. В трехмерной графике часто встречаются операции, которые можно выполнить за один такт: интерполирование и нормировка векторов, вычисление скалярного произведения векторов, интерполяция компонентов цвета. Включение SIMD-команд в систему команд позволяет существенно ускорить подобные вычисления. К такому типу относятся команды мультимедийного расширения MMX, SSE и ее последнее усовершенствование – команда SSE2. Все они позволяют проводить параллельную обработку упакованных целых чисел.

3.5. Формат команды

Существуют разные методы и приемы, с помощью которых код команды манипулирует обрабатываемыми данными, т.е. определяет, откуда считывать входные операнды и куда помещать результаты обработки (выходные операнды). Эти методы называются методами адресации. Чем более разнообразны источники приемники операндов с точки зрения скорости и простаты взаимодействия процессора с ними, чем шире функциональные возможности компьютера и выше его производительность.

Именно разнообразие способов адресации расширяет круг действий с данными. Для одного и того же кода операции (например сложения) могут быть применены различные вышеописанные варианты манипулирования данными с применением различных видов памяти - ОЗУ, ПЗУ, кэш, внутренних регистров. За счет большого разнообразия способов адресации число команд процессора всегда больше числа реализуемых операций.

Длина команды процессора зависит от объема адресного пространства, а система команд от сферы применения. Чем больше объем внутренней памяти процессора, тем шире его функциональные возможности, однако большие объемы памяти требуют увеличения длины машинного слова и соответственно длина адресной части команды. Например, для адресации к 128 Кбайтам ОЗУ длина адресного кода в команде должна быть 20 разрядов. Разряды кода адресации могут быть заданы как непосредственно в поле кода операции, так и в самостоятельном поле кода команды.

Для ограничения безграничного роста адресного кода и эффективного манипулирования возможностями всех внутренних и внешних устройств хранения данных применяются различные способы, исключающие прямую пропорциональную зависимость между общим объемом памяти и длиной адресной части команды. Количество способов адресации в различных процессорах может быть от 4 до 16.

Типовая команда, в общем случае, должна указывать:

- подлежащую выполнению операцию;
- адреса исходных данных (операндов), над которыми выполняется операция;
- адрес, по которому должен быть помещен результат операции.

Формат команды определяет ее структуру, то есть количество двоичных разрядов, отводимых под всю команду, а также количество и расположение отдельных полей команды. При создании компьютера выбор формата команды влияет на многие характеристики будущей машины. Оценивая возможные форматы, нужно учитывать следующие факторы:

- общее число различных команд;
- общую длину команды;
- тип полей команды (фиксированной или переменной длины) и их длина;
- простоту декодирования;
- адресуемость и способы адресации;
- стоимость оборудования для декодирования и исполнения команд.

Длина команды и разрядность полей адресации и операндов. Это важнейшее обстоятельство, влияющее на организацию и емкость памяти, структуру шин, сложность и быстродействие процессора. С одной стороны, удобно иметь в распоряжении как можно больше кодов операций, операндов, способов адресации, и максимальное адресное пространство. Однако, все это требует выделения большего количества разрядов под каждое поле команды, что приводит к увеличению ее длины. Вместе с тем, для ускорения выборки из памяти желательно, чтобы команда была как можно короче, а ее длина была равна или кратна ширине шины данных. Для упрощения

аппаратуры и повышения быстродействия компьютера длину команды обычно выбирают кратной байту, поскольку основная память организована в виде 8-битовых ячеек. В рамках системы команд одной машины могут использоваться разные форматы команд. Обычно это связано с применением различных способов адресации. В таком случае в состав кода команды вводится поле для задания способа адресации (СА), и обобщенный формат команды приобретает вид, показанный на рис.3.8.

Рис. 3.8. Основные части формата команды

Длины полей операционной и адресной частей определяются различными факторами, которые целесообразно рассмотреть по отдельности.

Количество двоичных разрядов, отводимых под код операции, выбирается так, чтобы можно было представить в коде любую из операций. При заданной длине кода команды приходится искать компромисс между разрядностью поля кода операции и адресного поля. Большее количество возможных операций предполагает длинное поле кода операции, что ведет к сокращению адресного поля, то есть к сужению адресного пространства. Для устранения этого противоречия иногда длину поля кода операции варьируют. Изначально под код операции отводится некое фиксированное число разрядов, однако для отдельных команд это поле расширяется за счет нескольких битов, отнимаемых у адресного поля. Так, например, может быть увеличено число различных команд пересылки данных. Необходимо отметить, что «урезание» части адресного поля ведет к сокращению возможностей адресации, и подобный прием рекомендуется только в тех командах, где подобное сокращение может быть оправданным.

В адресной части команды содержится информация о местонахождении исходных данных и месте сохранения результата операции. Обычно местонахождение каждого из операндов и результата задается в команде путем указания адреса соответствующей ячейки основной памяти или номера регистра

процессора. Принципы использования информации из адресной части команды определяет поле СА - система адресации. Система адресации задает число адресов в команде команды и принятые способы адресации.

В максимальном варианте числа разрядов адресного поля указываются три компонента: адрес первого операнда, адрес второго операнда и адрес ячейки, куда заносится результат операции. В принципе может быть добавлен еще один адрес, указывающий место хранения следующей инструкции (четырехадресный формат команды). Во многих моделях компьютеров необходимость в четвертом адресе отпадает, поскольку команды располагаются в памяти в порядке их выполнения, и адрес очередной команды может быть получен за счет простого увеличения адреса текущей команды в счетчике команд. Это позволяет перейти к трехадресному формату команды.

Если взять в качестве адреса результата адрес одного из операндов, то можно получить двухадресный формат команды. Команду можно еще более сократить, перейдя к одноадресному формату, что возможно при выделении определенного стандартного места для хранения первого операнда и результата. Обычно для этой цели используется специальный регистр (аккумулятор) процессора.

Применение единственного регистра для хранения одного из операндов и результата является ограничивающим фактором, поэтому помимо аккумулятора часто используют и другие регистры процессора. Так как число регистров невелико, для указания одного из них в команде достаточно иметь сравнительно короткое адресное поле.

При выборе количества адресов в адресной части команды обычно руководствуются следующими критериями:

- емкостью запоминающего устройства, требуемой для хранения программы;
- временем выполнения программы;
- эффективностью использования ячеек памяти при хранении программы.

Время выполнения одной команды складывается из времени выполнения операции и времени обращения к памяти.

3.6. Методы адресации операндов

Непосредственная адресация.

При непосредственной адресации в адресном поле команды вместо адреса содержится непосредственно сам операнд (рис. 3.9).

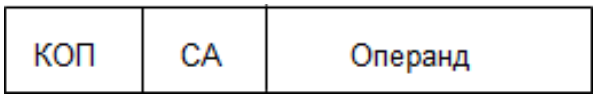


Рис. 3.9. Непосредственная адресация

Этот способ может применяться при выполнении арифметических операций, операций сравнения, а также для загрузки констант в регистры.

Когда операндом является число, оно обычно представляется в дополнительном коде. При записи в регистр, имеющий разрядность, превышающую длину непосредственного операнда, операнд размещается в младшей части регистра, а оставшиеся свободными позиции заполняются значением знакового бита операнда.

Помимо того, что в адресном поле могут быть указаны только константы, еще одним недостатком данного способа адресации является то, что размер непосредственного операнда ограничен длиной адресного поля команды, которое в большинстве случаев меньше длины машинного слова. В 50-60% команд с непосредственной адресацией длина операнда не превышает 8 бит, а в 75-80% - 16 бит. Таким образом, в подавляющем числе случаев шестнадцати разрядов вполне достаточно, хотя для вычисления адресов могут потребоваться и более длинные константы.

Наиболее интенсивно данный' вид адресации используется в арифметических операциях и командах сравнения. Непосредственная адресация сокращает время выполнения команды, так как не требуется обращение к памяти за операндом. Кроме того, экономится память, поскольку отпадает необходимость в ячейке для хранения операнда.

Прямая адресация. При прямой адресации адресный код (Ак) прямо указывает номер ячейки памяти, к которой производится обращение (рис. 3.10), то есть адресный код совпадает с исполнительным адресом.

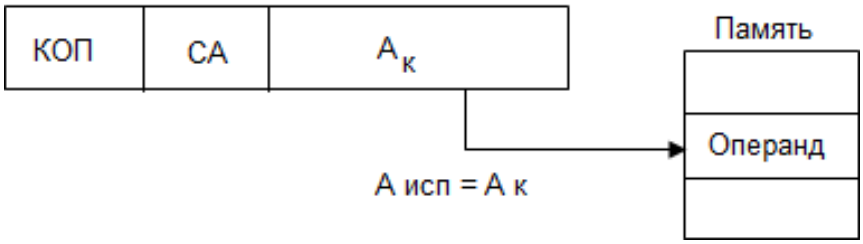


Рис.3.10. Прямая адресация

При всей простоте использования способ имеет существенный недостаток - ограниченный размер адресного пространства, так как для адресации к памяти большой емкости нужно «длинное» адресное поле. Однако более существенным можно считать то, что адрес, указанный в команде, не может быть изменен в процессе вычислений (во всяком случае, такое изменение не рекомендуется). Это ограничивает возможности по произвольному размещению программы в памяти.

Косвенная адресация. Одним из путей преодоления проблем, свойственных прямой адресации, может служить прием, когда с помощью ограниченного адресного поля команды указывается адрес ячейки, в свою очередь, содержащей полноразрядный адрес операнда (рис.3.11). Этот способ известен как косвенная адресация. При косвенной адресации содержимое адресного поля команды остается неизменным, в то время как косвенный адрес в процессе выполнения программы можно изменять. Это позволяет проводить вычисления, когда адреса операндов заранее неизвестны и появляются лишь в процессе решения задачи. Дополнительно такой прием упрощает обработку массивов и списков, а также передачу параметров подпрограммам.

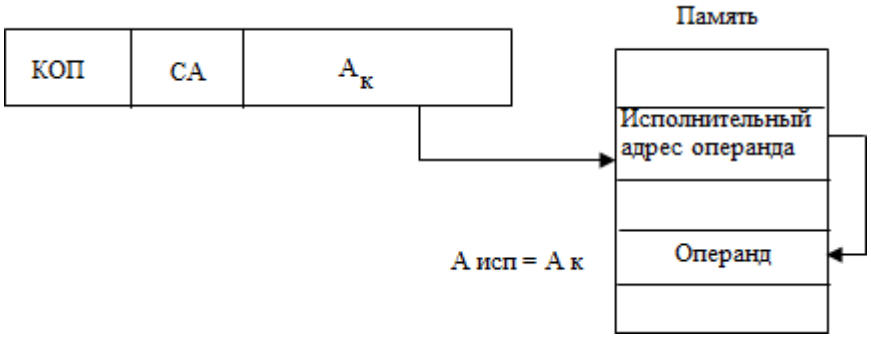


Рис.3.11. Косвенная адресация

Недостатком косвенной адресации является необходимость в двухкратном обращении к памяти: сначала для извлечения адреса операнда, а затем для обращения к операнду. Сверх того,

задействуется лишняя ячейка памяти для хранения исполнительного адреса операнда.

Регистровая адресация. Она напоминает прямую адресацию. Различие состоит в том, что адресное поле инструкции указывает не на ячейку памяти, а на регистр процессора (рис.3.12). Идентификатор регистра обозначен буквой *R*. Обычно размер адресного поля в данном случае составляет три или четыре бита, что позволяет указать соответственно на один из 8 или 16 регистров общего назначения (РОН).

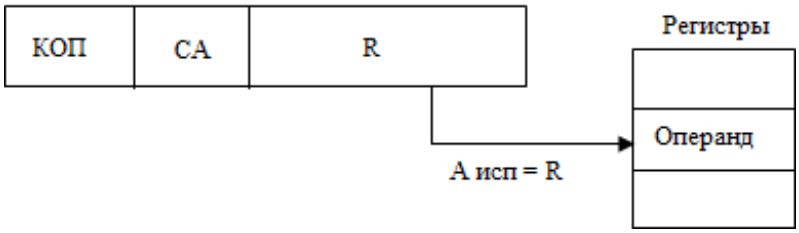


Рис.3.12. Регистровая адресация

Двумя основными преимуществами регистровой адресации являются: короткое адресное поле в команде и исключение обращений к памяти. Малое число РОН позволяет сократить длину адресного поля команды**.** К сожалению, возможности по использованию регистровой адресации ограничены малым числом РОН в составе процессора.

Косвенная регистровая адресация. Косвенная регистровая адресация представляет собой косвенную адресацию, где исполнительный адрес операнда хранится не в ячейке основной памяти, а в регистре процессора. Соответственно, адресное поле команды указывает не на ячейку памяти, а на регистр (рис.3.13). Достоинства и ограничения косвенной регистровой адресации те же, что и у обычной косвенной адресации, но благодаря тому, что косвенный адрес хранится не в памяти, а в регистре, для доступа к операнду требуется на одно обращение к памяти меньше.

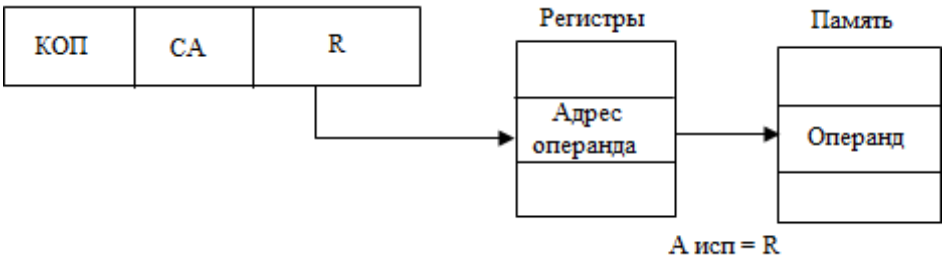


Рис.3.13. Косвенная регистровая адресация

Адресация со смещением. При адресации со смещением исполнительный адрес формируется в результате суммирования содержимого адресного поля команды с содержимым одного или нескольких регистров процессора (рис.3.14).

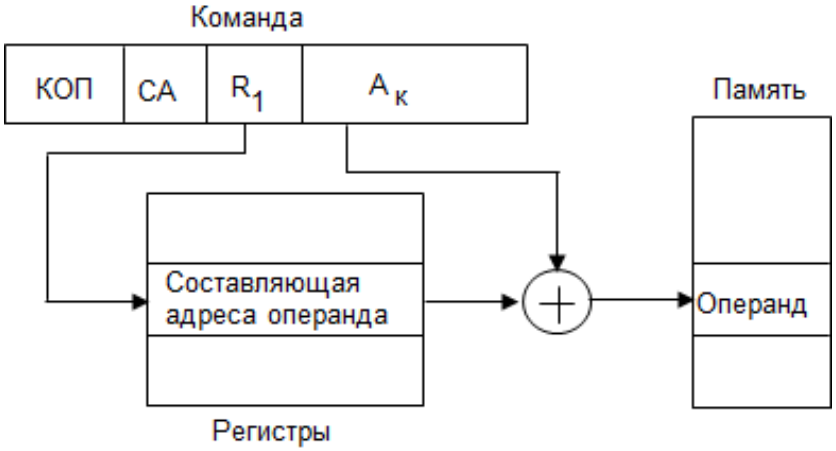


Рис.3.14. Адресация со смещением

Адресация со смещением предполагает, что адресная часть команды включает в себя как минимум одно поле (Ак). В нем содержится константа, смысл которой в разных вариантах адресации со смещением может меняться. Константа может представлять собой некий базовый адрес, к которому добавляется хранящееся в регистре смещение. Допустим и прямо противоположный подход: базовый адрес находится в регистре процессора, а в поле Ак указывается смещение относительно этого адреса. В некоторых процессорах для реализации определенных вариантов адресации со смещением предусмотрены специальные регистры, например базовый или индексный. Использование таких регистров предполагается по умолчанию, поэтому адресная часть команды содержит только поле Ак. Если же составляющая адреса может располагаться в произвольном регистре общего назначения, то для указания конкретного регистра в команду включается дополнительное поле R1 (при составлении адреса более чем из двух составляющих в команде будет несколько таких полей). В наиболее общем случае адресация со смещением подразумевает наличие двух адресных полей: Ак и R1.

В рамках адресации со смещением имеется еще один вариант, при котором исполнительный адрес вычисляется не суммированием, а конкатенацией (присоединением) составляющих адреса. Здесь одна

составляющая представляет собой старшую часть исполнительного адреса, а вторая — младшую.

Ниже рассматриваются основные способы адресации со смещением, каждый из которых, впрочем, имеет собственное название.

Относительная адресация. Для получения исполнительного адреса операнда при относительной адресации содержимое подполя Ак команды складывается с содержимым счетчика команд (СК) (рис. 3.15). Таким образом, адресный код в команде представляет собой смещение относительно адреса текущей команды. Следует отметить, что в момент вычисления исполнительного адреса операнда в счетчике команд может уже быть сформирован адрес следующей команды, что нужно учитывать при выборе величины смещения. Обычно подполе Ак трактуется как двоичное число в дополнительном коде.

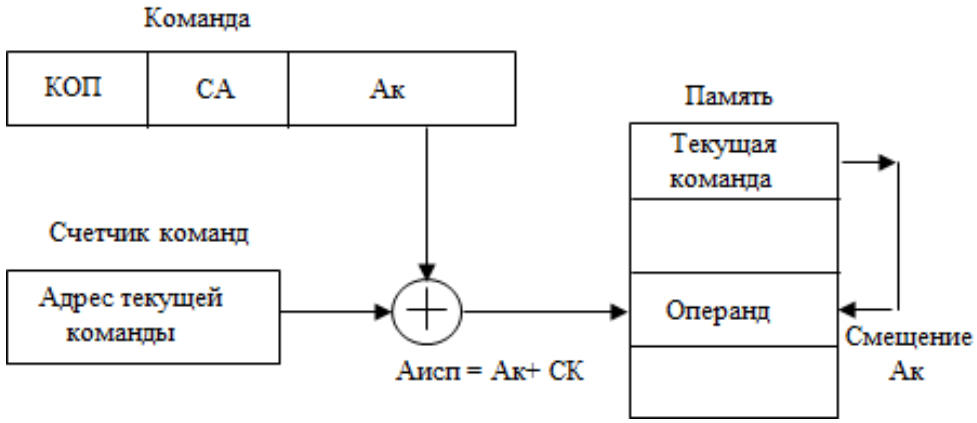


Рис.3.15. Относительная адресация

Адресация относительно счетчика команд базируется на свойстве локальности, выражающемся в том, что большая часть обращений происходит к ячейкам, расположенным в непосредственной близости от выполняемой команды. Это позволяет сэкономить на длине адресной части команды, поскольку разрядность подполя Ак может быть небольшой. Главное достоинство данного способа адресации состоит в том, что он делает программу перемещаемой в памяти: независимо от текущего расположения программы в адресном пространстве взаимное положение команды и операнда остается неизменным, поэтому адресация операнда остается корректной.

В случае **базовой регистровой адресации** регистр, называемый базовым, содержит полноразрядный адрес, а подполе Ас — смещение относительно этого адреса. Ссылка на базовый регистр может быть явной или неявной. В некоторых компьютерах имеется специальный базовый регистр и его использование является неявным, то есть подполе *R* в команде отсутствует (рис. 3.16).

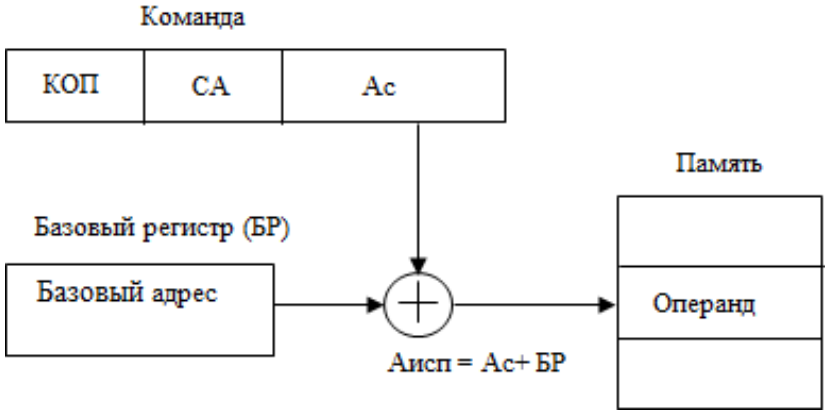


Рис.3.16. Базовая регистровая адресация с базовым регистром

Более типичен случай, когда в роли базового регистра выступает один из регистров общего назначения (РОН), тогда его номер явно указывается в подполе *R* команды (рис.3.17).

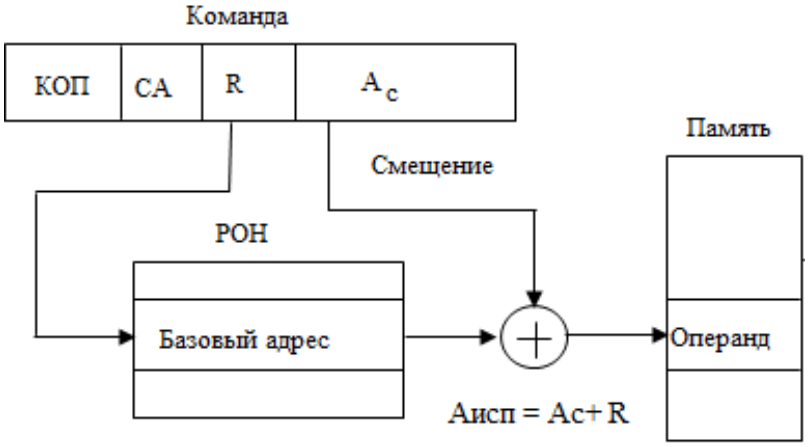


Рис.3.17. Базовая регистровая адресация с использованием РОН

Базовую регистровую адресацию обычно используют для доступа к элементам массива, положение которого в памяти в процессе вычислений может меняться. В базовый регистр заносится начальный адрес массива, а адрес элемента массива указывается в подполе команды в виде смещения относительно начального адреса массива. Достоинство данного способа адресации, в том, что смещение имеет меньшую длину, чем полный адрес, и это позволяет

сократить длину адресного поля команды. Короткое смещение расширяется до полной длины исполнительного адреса путем добавления слева битов, совпадающих со значением знакового разряда смещения.

При индексной адресации подполе содержит адрес ячейки памяти, а регистр (указанный явно или неявно) — смещение относительно этого адреса (рис.3.18). Как видно, этот способ адресации похож на базовую регистровую адресацию.

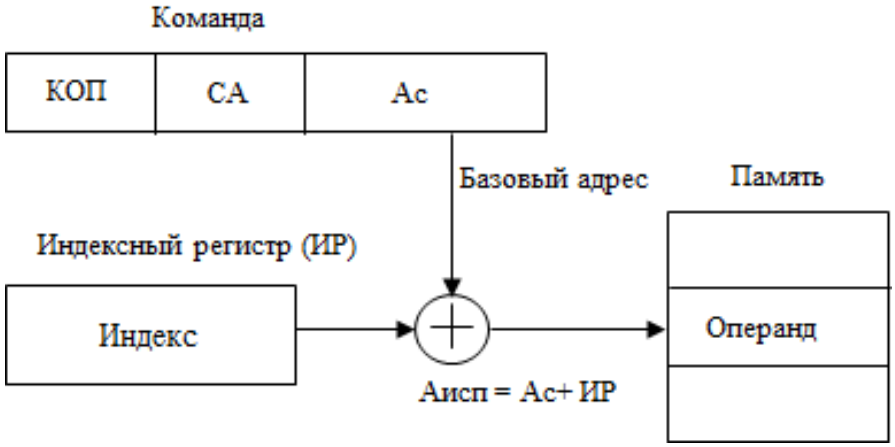


Рис.3.18. Индексная адресация

Поскольку при индексной адресации в поле находится полноразрядный адрес ячейки памяти, играющий роль базы, длина этого поля больше, чем при базовой регистровой адресации. Вычисление исполнительного адреса операнда производится аналогично схемам на рис. 3.17 и 3.18**.** Иногда вместо индексного регистра может использоваться один из РОН (рис.3.19).

Индексная адресация - удобный механизм для организации итеративных вычислений. Пусть имеется массив чисел, расположенных в памяти последовательно, начиная с адреса *N*, нужно увеличить на единицу все элементы массива. Для этого требуется извлечь каждое число из памяти, прибавить к нему 1 и вернуть обратно, а последовательность исполнительных адресов будет следующей: *N*, *N*+ 1, *N*+ 2 и т. д., вплоть до последней ячейки, занимаемой рассматриваемым массивом.

Значение *N* берется из подполя A_c команды, а в выбранный регистр, называемый **индексным регистром**, сначала заносится 0. После каждой операции содержимое индексного регистра увеличивается на 1.

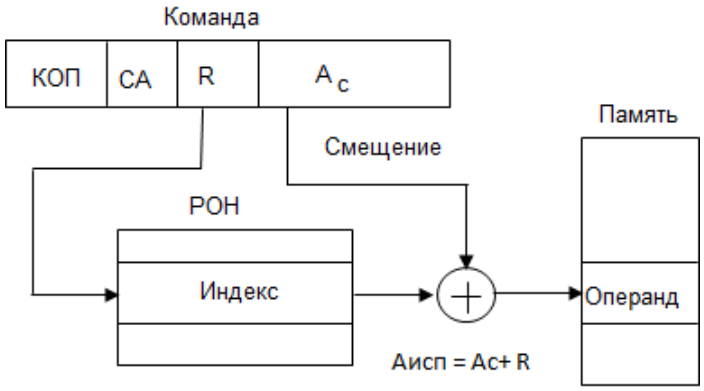


Рис. 3.19. Индексная адресация с использованием РОН

Страничная адресация. Предполагается разбиение адресного пространства на страницы. Страница определяется своим начальным адресом, выступающим в качестве базы. Старшая часть этого адреса хранится в — регистре адреса страницы (РАС). В адресном коде команды указывается смещение внутри страницы, (младшая часть исполнительного адреса). Исполнительный адрес образуется присоединением Асм к содержимому РАС, как показано на рис. 3.20. На рисунке символ || обозначает операцию присоединения.

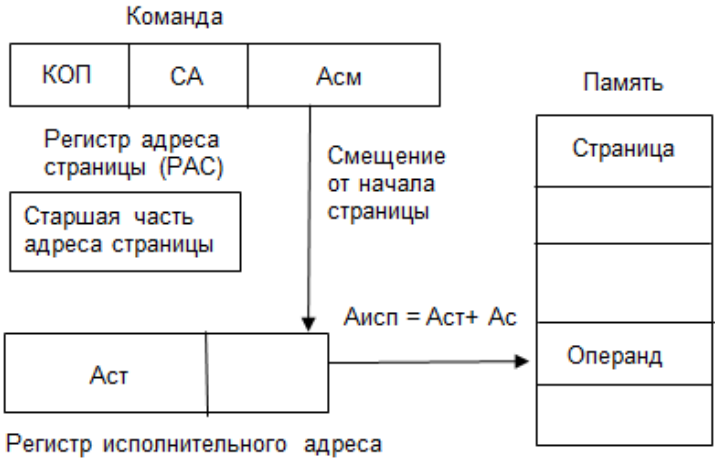


Рис.3.20. Страничная адресация

Блочная адресация. Блочная адресация используется в командах, для которых единицей обработки служит блок данных, расположенных в последовательных ячейках памяти. Этот способ очень удобен при работе с внешними запоминающими устройствами и в операциях с векторами. Для описания блока обычно берется адрес ячейки, где хранится первый или последний элемент блока, и общее количество элементов блока, заданное числом байтов или ячеек. Вместо длины блока может использоваться специальный признак «конец блока», помещаемый за последним элементом блока

Стековая адресация. Данный вид адресации был рассмотрен при описании стековой архитектуры системы команд.

Распространенность различных видов адресации. Частота использования различных способов адресации существенно зависит от типа архитектуры системы команд. Для машин со стековой архитектурой команд очевидно - основным способом адресации является стековая адресация. Для машин с аккумуляторной архитектурой команд главные способы адресации — это прямая и непосредственная. Для RISC-архитектуры преимущественным способ адресации является регистровая адресация.

Более сложным является вопрос о частоте использования различных видов адресации в регистровых машинах. В рамках этой архитектуры существует множество машин с самыми разнообразными списками команд и различными сочетаниями способов адресации, в силу чего дать однозначный ответ относительно наиболее распространенных вариантов практически невозможно. Основными способами адресации в командах управления потоком команд являются прямая и относительная.

Вопросы для контроля

1. Перечислите пять этапов выполнения команды.
2. Из каких частей состоит формат команды?
3. Перечислите этапы цикла команды.
4. Назовите три типа архитектур систем команд.
5. За счет чего в RISC-архитектурах достигнуто высокое быстродействие?
6. Назовите архитектуры систем команд наилучшие с точки зрения хранения операндов и способа доступа.
7. Рассмотрите особенности регистровой архитектуры команд.
8. Опишите формат чисел с ФЗ?
9. Каков формат чисел с ПЗ?
10. Что такое «пиксел» и какими параметрами он характеризуется?
11. Какие команды могут влиять на последовательность вычислений?
12. Перечислите основные методы адресации.

ГЛАВА 4. ОРГАНИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

4.1.Функции процессоров по организации вычислений

Наиболее важными характеристиками процессора являются:

- тактовая частота, которая в современных процессорах изменяется от сотен МГц до нескольких ГГц;
- адресное пространство внешней памяти, которое определяется разрядностью шины данных и шины команд;
- архитектура системы машинных команд и разрядность длины машинного слова;
- объем внутрикристальной кэш-памяти первого и второго уровней;
- количество сопроцессоров и контроллеров шин на кристалле;
- степень интеграции БИС процессора (число транзисторов на кристалле).

Основной задачей центрального процессора является выборка команд из памяти (чтение команды), анализ типа выполняемой операции (дешифрации команды), выборка данных из памяти (чтение данных - операндов), выполнение команды (исполнение), занесение в память получаемых результатов (запись).

Все действия процессора реализуются с помощью команд. Команда – это совокупность двоичных кодовых слов, содержащих информацию о том, какое действие нужно выполнить (код операции), каким образом определяется местоположение операндов в основной памяти (режим адресации), где расположены операнды (адресная часть).

Команда, как и данные, хранится в оперативной памяти, поэтому номер ячейки памяти, где она расположена, читается процессором. После чтения команды начинается процесс ее исполнения функциональными узлами компьютера. Для правильного понимания процесса исполнения команд, а значит и программ обработки, рассмотрим организацию вычислений через взаимодействия четырех основных блоков: узла управления, АЛУ, оперативной (основной) памяти и модуля ввода/вывода, объединяющего внешние порты (рис.4.1). Внешняя память непосредственного участия в процессе исполнения команд не принимает, т.к. необходимые в данной

программе данные заданные переписываются в основную память или кэш-память.

Устройство управления обеспечивает вес цикл исполнения команды. Перед началом вычисления адрес стартовой команды программы (т.е. номер ячейки памяти, где хранится команда) заносится в специальный счетчик команд процессора. В процессе выполнения каждой команды путем увеличения содержимого счетчика на единицу формируется адрес следующей подлежащей выполнению команды. В такой последовательности в оперативной памяти (ОП) располагаются команды исполняемый программы вычислений. При изменении естественного порядка выполнения команд программы (переход другую точку по условию или без условия) в процессоре формируется стартовый адрес той команды, с который начинается исполнения требуемого участка общей программы вычислений.

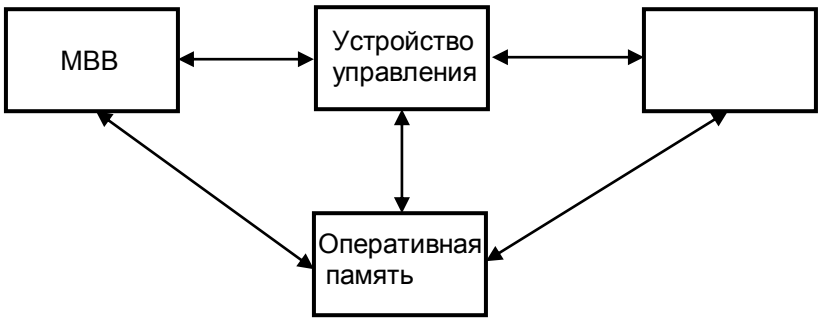


Рис.4.1. Схема взаимодействия блоков процессора

Программа запускается нажатием клавиатуры или запуском с машины, т.е. начинается исполнение программы начального пуска, которая хранится в постоянной памяти (ПЗУ). Завершив программу начального пуска процессор формирует адрес первой исполняемой команды. После реализации первой исполняемой команды результаты вычисления из АЛУ записываются опять в оперативную память.

Модуль ввода/вывода (МВВ) в конкретной аппаратной реализации имеет много вариантов исполнения, но его основной функцией является обеспечение подключения к компьютеру различных по назначению периферийных устройств и синхронизация обмена данными.

Рассмотрим принципы работы современных процессоров на примере простейшей структуры классического одноядерного

процессора.В его основе лежит неймановская архитектура, однако, имеются и существенные отличия. Они связаны с параллельной и конвейерной обработкой данных. Более подробно принципы конвейеров и параллельных вычислений будут рассмотрены в последующих разделах.

4.2. Структурная схема процессора

Первым микропроцессором, на базе которого бал создан первый компьютер, был Intel 8080, выпущенный в 1974 году. Он имел 8-разрядную шину данных и 16-ти разрядную шину адреса. Система команд состояла из 256 операций. Был разработан также целый комплекс сопутствующих микросхем различных генераторов, контроллеров, счетчиков для дополнения функций процессора и построения приборного интерфейса.

В 1979 году был создан процессор Intel 8086, который стал родоначальником направления микропроцессоров на многие годы. Он имел 16-ти разрядную длину слова, внешнюю шину данных той же разрядности, 20-ти разрядную шину адреса, позволявшую работать с адресным пространством в 1 мегабайт. Для обеспечения работы процессора был выпущен набор сопутствующих микросхем и программируемых контроллеров, арбитров шины, процессоров ввода-вывода, математического сопроцессора для автономной реализации обработки с плавающей запятой. Начальные модификации это серии процессоров работали на частоте 4,7 МГц, последующие модели имели частоту уже 8 МГц. На базе этих процессоров были созданы компьютеры массового применения IBM PC.

С тех пор внутренняя организация процессоров претерпела много изменений и совершенствований: изменились конфигурации корпусов и материнских плат, системные шины и видеосистемы, интерфейсы, однако тип процессора оставался неизменным.

В состав классического процессора входят блоки (рис.4.2):

- кэш команд и кэш данных;
- предпроцессор;
- исполнительные блоки;
- постпроцессор;
- блок записи результатов в память.

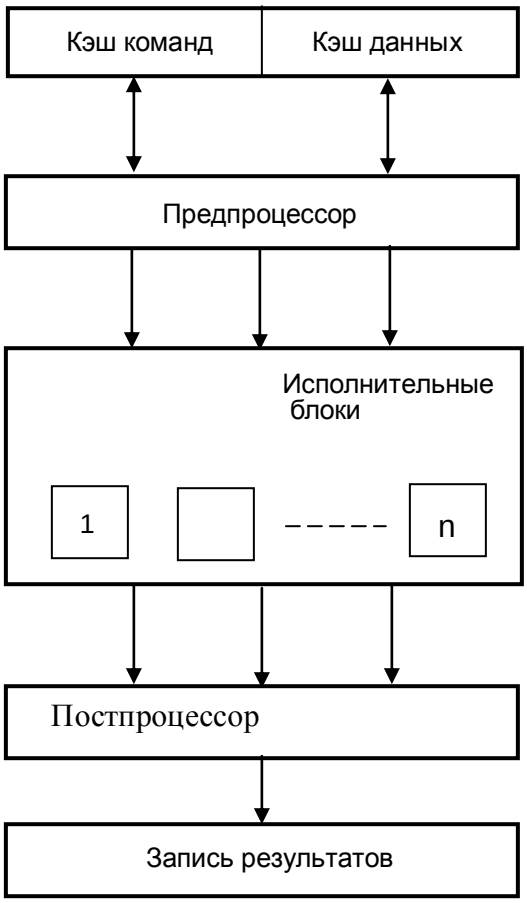


Рис.4.2. Схема классического процессора

Внутренние блоки процессора можно условно разделить на две части: блоки декодирования и интерпретации команд (предпроцессор) и исполнительные блоки.

Команды программы хранятся в кэш-памяти команд и извлекаются оттуда предпроцессором. Это процедура выборки. Затемпроисходит декодирование команд на примитивные микрокоманды,которые воспринимаются функциональными устройствами процессора. Когда микрокоманда получает из кэша данных свои операнды,она готова к исполнению. Декодированные микрокоманды образуютв предпроцессоре очередь к исполнительным блокам. Исполнительные блоки реализуются в виде конвейеров, образуя параллельную вычислительную среду. При неупорядоченном исполнении (технологии суперскалярной обработки) команды подаются на исполнительные блоки не в порядке следования в программе, а по мере готовности их операндов.

Команды поступают в исполнительные блоки и выполняются. В силу различной скорости выполнения операций в конвейерах

происходит переупорядочение команд и выдачи их результатов. Постпроцессор следит за готовностью результатов на выходе исполнительныхблоков и осуществляет возврат к естественной последовательности команд, то есть к их исходному расположению в программе.

Результат данной команды считается готовым, если завершились все предыдущие команды и их результаты признаны готовыми.

4.3. Общие алгоритмы организации вычислений

Типичная компьютерная вычислительная задача состоит из цепочки шагов, определяемых последовательностью команд программы. Каждая команда разбивается процессором на элементарные операции для реализации алгоритма вычислений.

Процессор выполняет при этом следующие функции:

- вычисление адресов команд и операндов;
- выборку и дешифрацию команд из основной памяти (ОП);
- выборку данных из ОП, внутренних регистров процессора и регистров адаптеров внешних устройств;
- прием и обработку запросов и команд от адаптеров на обслуживание внешних устройств;
- обработку данных и их запись в память, регистры процессора и регистры адаптеров внешних устройств;
- выработку управляющих сигналов для всех прочих узлов и блоков;
- переход к следующей команде.

Основными параметрами процессоров являются:

- разрядность;
- рабочая тактовая частота;
- размер кэш-памяти;
- состав инструкций (команд);
- конструктивы и рабочие напряжения.

Разрядность слова процессора определяет количество разрядов, над которыми одновременно могут выполняться операции; разрядность шины адреса процессора определяет его адресное пространство.

Адресное пространство — это максимальное количество ячеек основной памяти, которое может быть непосредственно адресовано процессором.

Рабочая тактовая частота процессора во многом определяет его внутреннее быстродействие, поскольку каждая команда выполняется за определенное количество тактов. Быстродействие (производительность) компьютера зависит также и от тактовой частоты шины системной платы, с которой работает процессор.

Кэш-память, устанавливаемая на плате процессора, имеет два уровня:

- L1 — память 1-го уровня, находящаяся внутри основной микросхемы (ядра) процессора и работающая всегда на его полной тактовой частоте;

- L2 — память 2-го уровня, кристалл, размещаемый на плате процессора и связанный с ядром внутренней процессорной шиной. Память L2 может работать на полной или половинной частоте процессора. Эффективность этой кэш-памяти зависит и от пропускной способности процессорной шины.

Состав инструкций — перечень, вид и тип команд, автоматически исполняемых процессором. От типа команд зависит классификационная группа процессора (CISC, RISC, VLIW). Перечень и вид команд определяют непосредственно те процедуры, которые могут выполняться над данными в процессоре, и те категории данных, к которым применимы эти процедуры. Дополнительные инструкции в небольших количествах вводились во многих процессорах, но существенное изменение состава инструкций произошло в процессорах i386 (этот состав далее принят за базовый), Pentium MMX, Pentium III, Pentium 4.

Конструктив подразумевает те физические разъемные соединения, в которые устанавливается процессор, и которые определяют пригодность материнской платы для установки процессора. Разные разъемы имеют разную конструкцию (Slot — щелевой разъем, Socket — разъем-гнездо), разное количество контактов, на которые подаются различные сигналы и рабочие напряжения.

Рабочие напряжения также являются фактором пригодности материнской платы для установки процессора.

Рассмотрим схему организации вычислений.

Во время исполнения программа должна находится в памяти с произвольным доступом (Random-Access Memory, RAM). Время доступа к памяти в современных устройствах RAM составляет с десяток наносекунд.

Большинство компьютерных операций выполняется в АЛУ. Любые арифметические или логические операции, в том числе умножение, деление и сравнение чисел, начинаются с пересылки этих чисел в процессор, точнее в буферных элементах памяти, называемых регистрами. Каждый регистр может хранить одно слова данных.

Затем данные поступают во входные регистры АЛУ, где над ними выполняется нужная операция. Управляющие и арифметико-логические устройства с буферными регистрами работают намного быстрее, чем все остальные устройства компьютера, поэтому процессор может одновременно контролировать множество внешних устройств: клавиатуру, диски, дисплеи и различные сенсорные устройства. Работу всех перечисленных устройств, а также узлов ввода и вывода данных координирует управляющее устройство (УУ). Оно генерирует сигналы управления и отслеживает состояние всех устройств компьютера. Соответствующие синхронизирующие сигналы генерируются специальными схемами, входящими в состав УУ. На практике узлы и функциональные элементы УУ рассредоточены по разным местам процессора, и УУ не исполняются в виде отдельного блока.

Управление процессором осуществляется с помощью сигналов, которые называются **тактовыми импульсами** (clock) и выдаются через фиксированные интервалы времени. Промежуток времени между двумя тактовыми импульсами составляет тактовый цикл или просто такт. Длительность одного такта является важнейшей характеристикой, определяющей производительность процессора. Количество тактов в секунду называется тактовой частотой. Современные процессоры имеют тактовую частоту от сотен миллионов до миллиарда в секунду. Это соответствует единицам ГГц.

Компьютер может работать с высокой скоростью при условии, чтобы полное слово данных обрабатывалось за установленное время. Для пересылки данных требуется несколько параллельных линий. Эта группа линий называется шиной (bas). Шина также содержит линии для передачи адреса и управляющих сигналов. Это внутренние шины процессора, расположенные в кристалле самого процессора. Они обозначаются как шина адреса, шина данных и шина управления. Кроме внутрикристалльных шин есть еще локальные и системные шины самого процессора, о которых речь пойдет в других разделах.

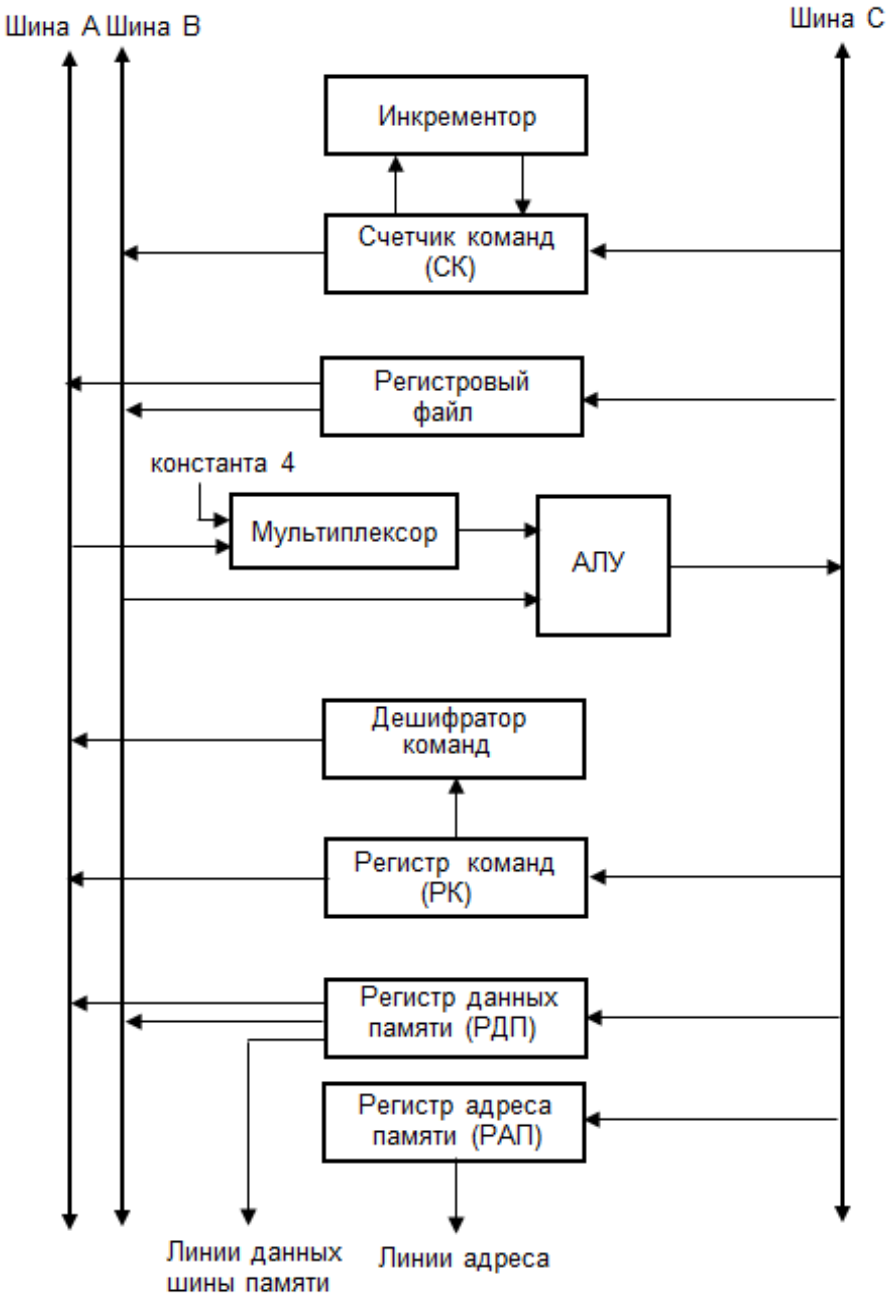


Рис.4.3. Схема взаимодействия АЛУ и регистров

Рассмотрим типичный процесс выполнения программы компьютером. Выполнение начинается с записи в счетчик команд (СК) адреса первой исполняемой команды (рис.4.3). Содержимое счетчика пересылается в регистр адреса памяти (РАП), а в память направляется управляющий сигнал Read. Адресуемое слово считывается из памяти и загружается в регистр данных памяти (РДП) и пересылается в регистр команды (РК). Команда готова к декодированию и выполнению.

Операнды располагаются либо в ОП, либо в кэш-памяти, либо в одном из регистров общего назначения (РОН). Адрес первого операнда читается из двоичного слова команды, расположенной в РК, и пересылается в регистр РАП и затем читается в АЛУ. Аналогично выбираются и другие операнды. Если результат должен быть сохранен, он будет записан в РДП, затем перемещен в РАП, после чего будет инициирован цикл записи Write (в случае работы с ОП). При работе с регистрами в качестве источника и хранения данных используется Регистровый файл. Как только завершится выполнение текущей команды, содержимое СК увеличивается, и он начинает указывать на следующую подлежащую выполнению команду.

Нормальное выполнение программы может быть прервано при возникновении в вычислениях непредусмотренных результатов. Чтобы отреагировать на эту ситуацию процессор должен прервать выполнение текущей программы. С этой целью генерируется сигнал прерывания (interrupt) – запрос на предоставление процессорного времени и исполнения программы обработки прерывания и ликвидации нештатной ситуации. При этом состояние СК и РОН сохраняется для того, чтобы после ликвидации сбоя, состояние процессора восстановилось, и прерванная программа могла продолжить работу.

4.4. Арифметико-логическое устройство и регистры операндов

Чтобы сократить количество шагов, необходимых для выполнения команды, в большинстве современных процессоров для параллельной пересылки информации используется несколько внутренних каналов обмена. На рис.4.3. показана трехшинная схема

соединения АЛУ и регистров, при этом все РОН организованы в виде регистрового файла. Регистры объединены в матрицу запоминающих ячеек с произвольным доступом. Имеется возможность обращаться к двум разным регистрам и помещать их содержимое на шины A и B. Третий порт Регистрового файла дает возможность во время того же тактового цикла загрузить данные с шины C (данных) в третий регистр.

Шины A (управления) и B (адреса) используются для пересылки исходных операндов на входы АЛУ, где над ними производятся арифметические и логические операции. Результат пересылается в регистр назначения по шине C. Если нужно, АЛУ может передать один из двух операндов на шину C без изменения. Особенностью архитектуры является наличие регистра-инкрементатора, используемого для увеличения содержимого СК на 4. Необходимость автоматического добавления в СК числа 4 связано с тем, что каждая команда процессора занимает 4 байта (32-х разрядное слово) и нужно сдвигать адреса очередных вызываемых из памяти команд на 4 шага. Такой шаг может использоваться для приращения других адресов, например адресов памяти в командах групповой загрузки и сохранения. **АЛУ** – основной узел обработки чисел (операндов), имеет два входа операндов и один выход результата (рис.4.4).

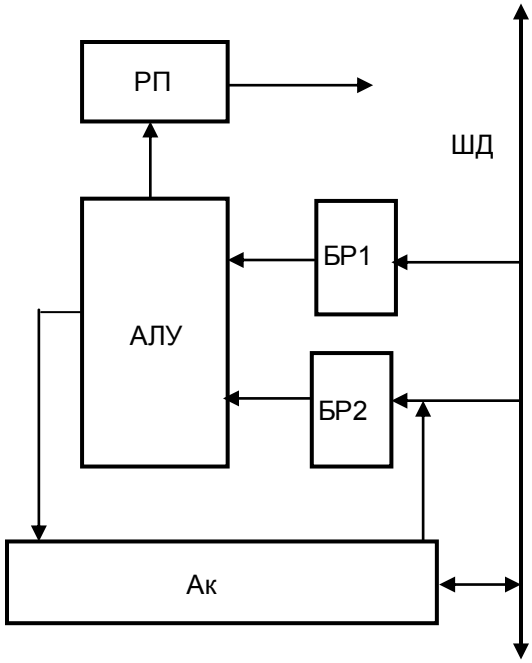


Рис.4.4. Схема АЛУ и вспомогательных регистров

АЛУ обычно реализует 10-15 типичных арифметических и логических операций, при этом общее число команд с модификациями может достигать нескольких десятков и даже сотен. Разрядность АЛУ обычно в два раза больше разрядности обрабатываемых операндов. Входными и выходными узлами АЛУ являются буферные регистры Рх и Ру для ввода операндов и аккумулятор (Ак) для временного хранения промежуточных результатов или их передачи в Ру. При необходимости Ак может производить элементарные операции сдвига, инверсии, присвоения знака, обнуления. Разрядность аккумулятора равна разрядности АЛУ.

Операционный блок представляет собой ту часть АЛУ, которая, собственно, и выполняет арифметические и логические операции над поданными на вход операндами. Выбор конкретной операции из возможного списка операций для операционного блока определяется кодом операции команды. Операционные блоки современных АЛУ строятся как комбинационные схемы, то есть они не обладают внутренней памятью и до момента сохранения результата операнды должны присутствовать на входе блока.

Перечень функций АЛУ зависит от типа процессора и различен для машин различных типов. Некоторые АЛУ способны выполнять большое число различных по характеру операций, у других набор операций ограничен. Функции АЛУ определяют архитектуру процессора в целом.

Типичными операциями, выполняемыми АЛУ большинства процессоров, являются: СЛОЖЕНИЕ, ВЫЧИТАНИЕ, И, ИЛИ, ИНВЕРСИЯ, СДВИГ ВПРАВО, СДВИГ ВЛЕВО, ПРИРАЩЕНИЕ ПОЛОЖИТЕЛЬНОЕ, ПРИРАЩЕНИЕ ОТРИЦАТЕЛЬНОЕ, на базе которых реализуются более сложные, многотактные операции умножения/деления.

В АЛУ реализуются операции по непосредственной обработке данных. Операнды по шине В поступают в буферные регистры Рх и Ру. В операционном блоке, который в аппаратном смысле является многофункциональным сумматором, выполняются арифметические и логические функции. Результат обработки передается в аккумулятор $А_к$ – один из узлов хранения и обработки. С него на Ру подаются данные результата выполнения очередной операции, необходимые на

следующем шаге выполнения программы. В состав АЛУ включены только комбинационные схемы, поэтому при поступлении исходных данных на входе АЛУ сразу появляются результирующие данные на его выходе, которые размещаются в Ак. Для выполнения операции следующего шага программы эти данные поступают сразу на вход Ру. Буферные регистры Рх и Ру недоступны для программиста.

Аккумулятор Ак - главный регистр процессора при различных манипуляциях с данными. Процессор может выполнять некоторые действия над данными непосредственно в аккумуляторе. Его содержимое может быть очищено записью нулей во все разряды, сдвинуто влево или вправо, получено инвертированное значение содержимого. Данные могут поступать в аккумулятор также с внутренней шины данных, куда направляются результаты вычислений. Количество разрядов аккумулятора соответствует длине слова процессора, хотя иногда могут быть использованы аккумуляторы двойной длины. Дополнительные разряды используются при выполнении операций умножения с целью избегания потерь в точности.

Одним из ключевых регистров процессора является связанный с выходами АЛУ регистр признаков результата РП. Он предназначен для фиксации и хранения признаков (флагов) результатов арифметических и логических операции. К таким признакам относятся равенство результата нулю, знак полученного результата, наличие переноса из старших разрядов, переполнение разрядной сетки. Устройство управления учитывает указанные признаки для реализации условных переходов программы по результатам операций, выполняемых в АЛУ. Каждый из разрядов РП фиксирует один из предусмотренных признаков результата. В смысле аппаратной реализации РП может и не быть функционально цельным, автономным узлом. Чаще он представляет собой сочетание рассредоточенных по схеме триггеров состояний, каждый из которых фиксирует состояние узлов процессора при выполнении операций над двоичными числами. Фиксация результатов проверок очень важна при реализации процедур перехода - нарушения естественной последовательности выполнения команд.

Регистр РП предоставляет программисту возможность организовать работу процессора так, чтобы при определенных условиях менялся порядок выполнения программы. Команды, меняющие порядок, называются командами условного перехода. Эти команды предназначены для изменения хода выполнения программы в соответствии со значением, применяемым тем или иным разрядом регистра состояния. Традиционный способ использования этих специальных команд предполагает загрузку счетчика команд СК новым содержимым, если значение определенного разряда РП становится равным единице.

Чаще всего используются следующие разряды регистра РП.

Разряд переноса. Единичное состояние данного разряда указывает, что последняя выполненная операция сопровождалась переносом или заемом (отрицательным переносом). Единица в разряде переноса фиксирует наличие переноса из последнего двоичного разряда результата в АЛУ при сложении двух чисел. Отрицательный перенос (заем) фиксируется в регистре состояния при вычитании большего числа из меньшего.

Разряд нулевого результата. Принимает единичное значение, если после окончания операции во всех разрядах результата в АЛУ оказываются двоичные нули независимо от того, какая выполнялась операция.

Знаковый разряд. Принимает единичное значение, когда старший значащий бит двоичного результата в АЛУ становится равным единице. При выполнении арифметических операций с числами в дополнительном коде единица в старшем значащем бите свидетельствует, что в регистре находится отрицательное число.

Часть разрядов управляющие, которые запускают команды прерывания. К ним относятся флаг системного прерывания, по которому процессор переходит в режим пошагового выполнения программы, флаг общего прерывания от внутреннего или внешнего устройства. Выход РП связан со входом микропрограммного автомата управления, который формирует сигналы условного перехода в соответствии с битами состояния регистра РП.

4.5. Работа устройства управления процессора

Для выполнения команд программы процессор должен генерировать последовательности управляющих сигналов. Возможны два варианта технической реализации этого процесса: аппаратный, с помощью специально разработанных цифровых узлов, и программный, с помощью записанных в памяти специальных управляющих программ.

На рис.4.5 представлен первый, аппаратный вариант реализации последовательности управляющих шагов.

Каждый шаг этой последовательности выполняется за один такт. Для отслеживания управляющих шагов применяется специальный счетчик. При формировании управляющих сигналов на общем выходе устройства учитываются следующие данные:

- содержимое счетчика управляющих шагов;
- содержимое регистра команды;
- значения флагов кодов условий;
- внешние входные сигнала (например, запрос на прерывание).

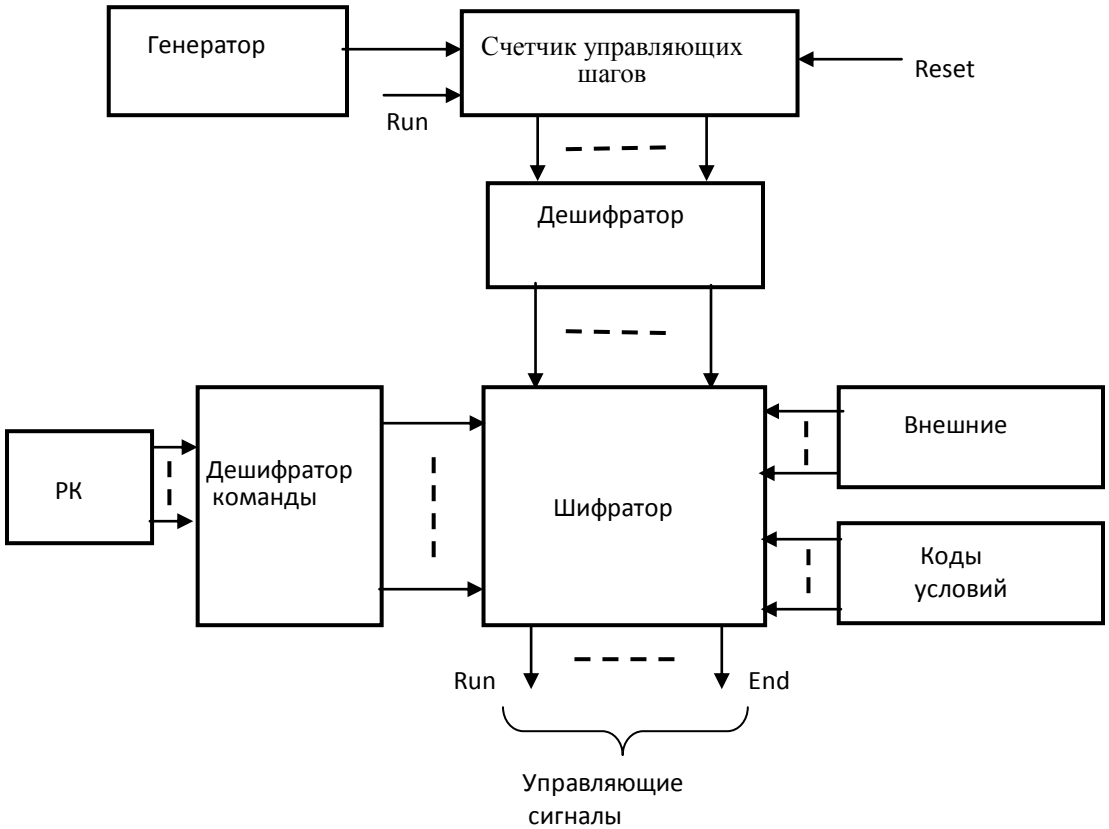


Рис.4.5. Схема устройства управления процессора

Комбинация цифровых узлов шифратор/дешифратор представляет собой комбинационную схему, генерирующую управляющие сигналы на основании состояния всех ее входов. Каждому шагу управляющей последовательности соответствует отдельная сигнальная линия дешифратора шага. Аналогичным образом на выходе дешифратора команд имеется отдельная линия для каждой машинной команды.

Для любой машинной команды, загруженной в РК, одна из выходных линий дешифратора устанавливается в 1, а все остальные линии устанавливаются в положение 0. Если дешифратор расшифровывает входной код и выдает сигнал 1 на одном из своих выходов, соответствующих этому коду, то шифратор выполняет обратную задачу - сигнал как своем единственном входе шифрует в набор управляющих сигналов для рассылки их по исполнительным устройствам процессора. Управляющие сигналы выдаются группами в строгой последовательности для временной синхронизации исполнительных узлов.

Сигнал **End** начинает новый цикл выборки команды, сбрасывая счетчик управляющих шагов командой **Reset** в начальное состояние. Командой **Run** в конце каждого тактового цикла в счетчик управляющих шагов добавляется 1. Если Run становится равным нулю, отчет шагов прекращается. Ограниченность количества входов и выходов шифратора и дешифратора не позволяют увеличить количество и сложность системы команд.

На рис. 4.6 приведена схема построения процессора, в котором выделены связи УУ с АЛУ, регистровой памятью и основной памятью.

В схеме введены следующие обозначения:

УУ –устройство управления процессора;

АЛУ – арифметико-логическое устройство;

БРП – блок регистровой памяти;

БУР – блок управления регистров;

БСОП – блок связи с основной памятью (ОЗУ, ПЗУ, кэш).

В данной схеме УУ формирует и подает во все блоки сигналы управления, обусловленные спецификой выполняемой операции и результатами предыдущих операций, формирует адреса ячеек памяти,

используемых в операции. Опорную последовательность синхронизирующих импульсов УУ получает от тактового генератора.

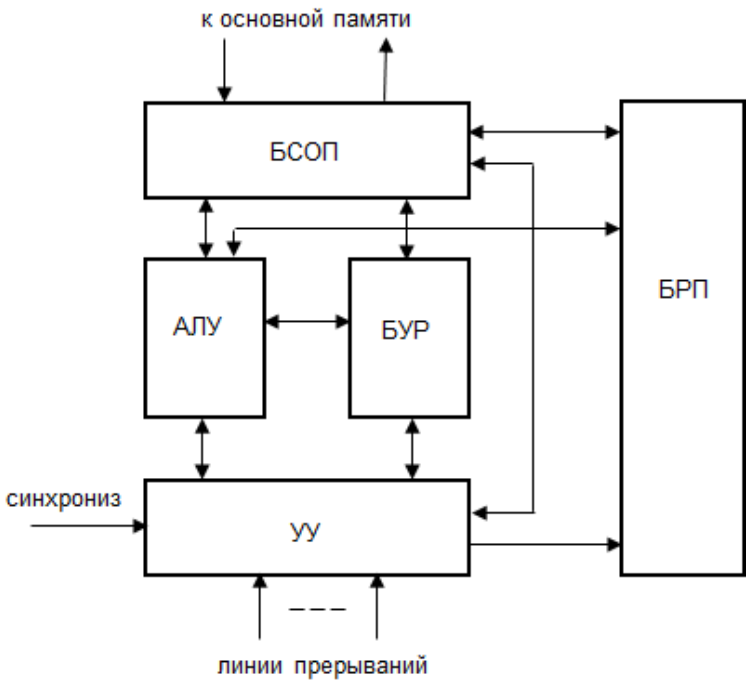


Рис. 4.6. Структура связей УУ с основной и регистровой памятью

БРП предназначен для временного хранения, записи и выдачи обрабатываемой информации, используемой в ближайших тактах работы процессора. Это сверхбыстродействующая память процессора.

Интерфейсная схема БСОП предназначена для связи с основной памятью, в которую входят ОЗУ, ПЗУ и кэш-память. Входящие в ее состав буферные регистры отражены в схема на рис.4.3.

Рассмотрим второй микропрограммный вариант технической реализации команд программы. Это метод микропрограммного управления, при котором сигналы управления генерируются программой, состоящей из команд, идентичных машинному коду.

Управляющая слово (ControlWord) – это слово, отдельные биты которого представляют различные управляющие сигналы, генерируемые схемой на рис.4.7. Каждый шаг управляющей последовательности команд определяет комбинацию нулей и единиц в управляющем слове. Последовательность микропрограммы управляющих слов составляет микропрограмму машинной команды.

В свою очередь, микропрограммы всех команд процессора хранятся в специальной управляющей памяти.

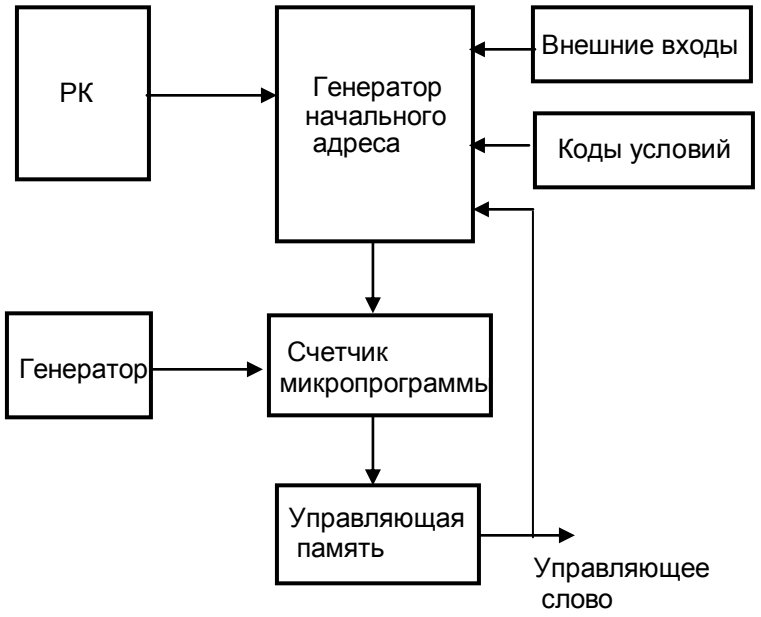


Рис.4.7. Организация управляющего блока

Прежде чем сгенерировать сигналы команды, управляющий блок последовательно считывает из управляющей памяти слова соответствующей микропрограммы. Последовательное чтение слов из управляющей памяти обеспечивается счетчиком микропрограммы. При каждой загрузке в РК новой команды в счетчик микропрограмм загружается выходное значение блока, названного в схеме генератором начального адреса. После этого на очередном такте выполняется автоматическое приращение содержимого счетчика микропрограммы для выбора из управляющей памяти очередной порции управляющих слов, находящихся в РК.

Для поддержки ветвления в микропрограммах на генератор начального адреса возлагается дополнительная функция – генерирование адреса перехода. По указанию микрокоманды этот блок загружает в счетчик микропрограммы новый адрес.

Для поддержки режимов условного перехода на входы генератора начального адреса подаются сигналы с внешних входов, коды условий и содержимое регистра команд. После каждой выборки микрокоманды из управляющей памяти происходит приращение значения счетчика микропрограммы.

Следует отметить, что практических вариантов аппаратно-программной реализации устройства управления много, это зависит от разработчика архитектуры самого процессора. Выше рассмотрены наиболее общие моменты взаимодействия основных узлов процессора при выполнении текущих команд: устройства управления, АЛУ и различных уровней памяти.

4.6. Организация режима прерывания основной программы

Прерывания являются движущей силой компьютера, их следует рассматривать не столько как реакцию процессора на нештатные ситуации в текущей работе компьютера, а как естественный процесс, с помощью которого реализуется поддержка большинства необходимых механизмов, в частности, таких как ввод/вывод, виртуальная память, режим разделения памяти.

Система прерываний является неотъемлемой частью любого компьютера и предназначена для обеспечения быстрой реакции процессора на ряд ситуаций, требуемых его внимания, которые могут возникнуть не только в самом компьютере, но и за его пределами. Система прерываний является комплексом аппаратных и программных средств. Прерывания принято разделять на два основных типа: программные и аппаратные. **Программные** прерывания связаны с выполняемой программой и являются синхронными по отношению к этой программе.

Аппаратные прерывания могут возникать в произвольные моменты времени и являются асинхронными по отношению к выполняемой программе. С помощью аппаратного прерывания осуществляется взаимодействие процессора с периферийными устройствами (клавиатура, магнитные диски, таймер), а также сообщается о различных ошибках аппаратуры (ошибка памяти, ошибка передачи данных по шине). Реагируя на аппаратные прерывания, процессор должен идентифицировать его источник, сохранить минимальный контекст прерываемой программы и переключиться на специальную программу - обработчик прерывания (interrupthandler).

Действие обработчика прерывания, называемое «обслуживанием прерываний», заключается в том, чтобы правильно отреагировать на

прерывание от конкретного источника, например, поместить символ нажатой клавиши в буфер, считать сектор с диска, произвести инкремент системных часов. После завершения обслуживания прерывания процессор должен возвратиться к обслуживанию прерванной программы, и она должна продолжаться таким образом, как будто прерывания не было вовсе.

Ситуации, возникающие в компьютерах и приводящих к прерыванию программы, можно разделить на следующие виды:

1. Случаи, возникающие при выполнении программы. К ним относятся:

- ошибки при выполнении арифметических операций (переполнение при сложении целых чисел, ошибка деления, переполнение порядка при выполнении арифметических операций над числами с плавающей запятой;
- некорректности в машинных командах (некорректный код операции, некорректный адрес команды или операнда. некорректные данные;
- нарушения защиты памяти, точнее, защиты программ и данных;
- случаи, связанные с организацией виртуальной памяти (например, отсутствие сегмента или страницы, к которым производятся обращения в основной памяти).

2. Запросы прерываний от внешних (периферийных) устройств или каналов (процессоров) ввода/вывода в следующих случаях:

- внешнее устройство, готовое к обмену, требует реакции процессора для организации передачи данных или оно завершило работы по передаче данных, или ситуация отсутствия бумаги в принтере.

3. Запросы прерывания от внутренних таймеров процессора.

4. Запросы от объекта управления (например, от робота-манипулятора), который функционируют в режиме реального времени (realtime). Это означает, что темп работы программы задается извне объектом управления.

5. Запросы прерывания от других процессоров или компьютеров для обеспечения синхронизации вычислительных процессов,

протекающих в рамках многопроцессорной или многомашинной системы.

Рассмотрим одну из характерных типов команд компьютера – цикл команды прерывания. Во всех компьютерах предусмотрены команды перехода программы, т.е. прерывание естественного порядка ее исполнения. Причиной прерывания основной работающей программы может быть заранее введенное в программу условие (условный переход) или непредусмотренные ранее действие (безусловный переход), которые ведут к ветвлению программы. Еще одним действием, требующий прерывания хода программы, является вызов процедуры.

С целью реализации указанных случаев - изменения естественного хода вычислений, в адресной части команд перехода программы (независимо от условий) содержится адрес точки перехода, т.е. адрес той команды, куда нужно перейти и которая должна быть выполнена следующей.

На рис.4.8. представлен пример реализации команды условного перехода.

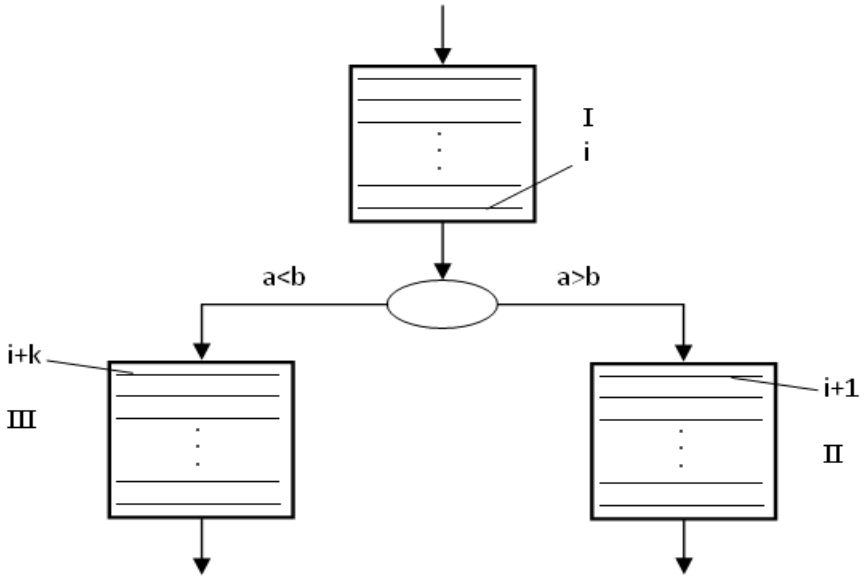


Рис.4.8. Реализация команды условного перехода

В результате завершения выполнения команд участка I программы процессор анализирует полученный результат (операнд «a»). Предложим, условием дальнейшего продолжения естественного хода программы является условие «a>b». Если это условие выполняется, программа переходит на участок II и

естественный ход ее выполнения сохраняется. Выполняется следующая по порядку команда с номером i+1.Если же в результате проверки условия окажется «a<b», то программа должна нарушить естественный ход и перейти к выполнению участка III, т.е. к выполнению команды с номером i+к, где к – величина смещения по номеру команды. Как указывалось ранее, величина смещения фиксируется содержимым счетчика команд СК.

Другой причиной условного перехода может быть состояние регистра признаков (РП) в операционном узле процессора. Опрос битов его состояния (нулевой результат, отрицательный результат, переполнение) по результатам предыдущей арифметической или логической операции может послужить причиной перехода программы.

Во всех компьютерах предусмотрены средства, благодаря которым процессор по инициативе внешних устройств (мышка, клавиатура, накопители) может прервать выполнение текущей программы с последующим возвратом к прерванной точке выполнения. Этот процесс состоит из следующих шагов: от периферийного устройства через модули ввода/вывода процессор получает сигнал запроса прерывания. Получив сигнал запроса, процессор запоминает номер последней выполненной команды и переходит к выполнению программы обработки прерывания, т.е. к выполнению действий, запрошенных периферийным устройством.

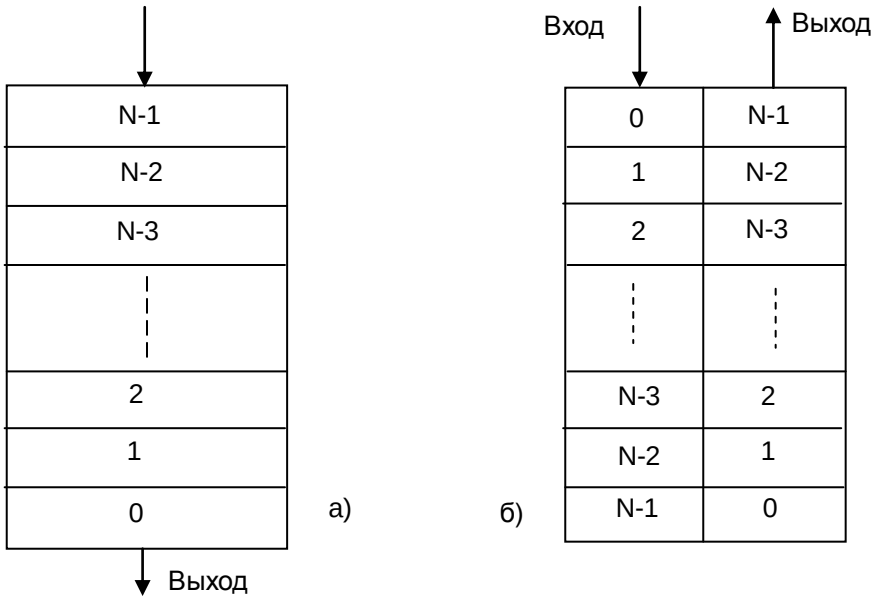


Рис.4.9. Принцип использования стековой памяти

По завершении этих действий процессор восстанавливает вычислительный процесс путем перехода к следующей (после запомненной) команде программы.

Для запоминания места прерывания основной программы компьютера часто используется показанная на рис. 4.9. стековая память. Запись в стек позволяет запомнить порядок ветвления основной программы (номер ячейки i+1) и передать в счетчик команд СК процессора для продолжения прерванного вычислительного процесса.

Если до завершения работы программы обработки прерывания возникает необходимость в новом прерывании, появляются непредусмотренные вложенные циклы и требуется запомнить место ухода на вложенную программу. После выполнения вложенной программы необходимо вернуться к выполнению предыдущей программы обработки прерывания, а после ее завершения к выполнению основной программы (рис.4.10). При нескольких таких ветвлениях возвращение к работе основной программы происходит в последовательности, обратной последовательности прерываний.

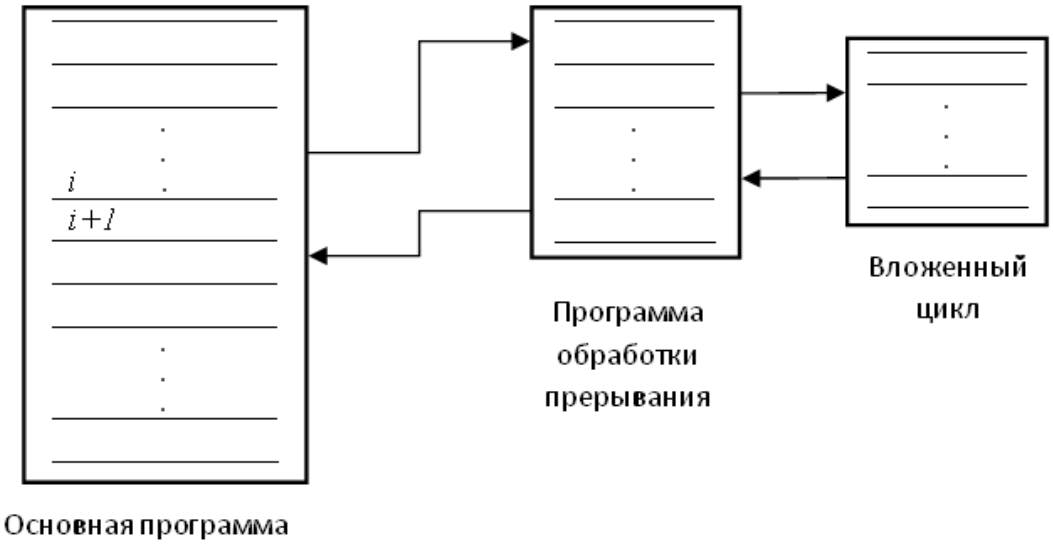


Рис.4.10. Реализация вложенного цикла прерывания

Этот механизм автоматического восстановления временно приостановленных программ, обеспечение порядка ветвлений и возврата к основной программе реализуется с помощью принципа LIFO, заложенного в стековую память – «первым пришел – последним вышел». Адреса команд записываются в порядке возникновения ветвлений, а читаются в обратном порядке.

Таким образом, система прерывания позволяет компьютеру реагировать на программно–независимые события путем временного прекращения выполнения текущей программы и передачи управления программе (или подпрограмме), специально предназначенной для обработки такого события.

4.7. Оценка производительности вычислений

Скорость выполнения программы является важнейшим критерием оценки производительности компьютера. Скорость выполнения программ зависит от архитектуры компьютера, в особенности от скорости взаимодействия по линии «процессор-память». Поскольку программы обычно пишутся на языке высокого уровня, производительность завит и от того, насколько удачно компилятор переводит их на машинный язык. Таким образом, для достижения высокой производительности нужно, чтобы компилятор, состав машинных команд и средства обеспечения реализации программ имели оптимальную структуру.

Общее время выполнения заданной программы является мерой производительности всего компьютера. Оно зависит от быстродействия процессора и устройств памяти. Суммарное время выполнения прикладных программ и программ операционной системы называется процессорным временем. Процессорное время зависит от аппаратного обеспечения, участвующего в выполнении конкретных машинных команд.

Примером эффективного использования времени процессора при обращении к оперативной памяти является использование в качестве буфера кэш-памяти. По мере стартового выполнения программы ее команды по одной выбираются из памяти и по шине передаются в процессор, а их копии перемещаются в кэш. Когда для выполнения требуются данные, расположенные в основной памяти, они также пересылаются в процессор, а их копии помещаются в кэш. Если позднее та же команда или элемент данных потребуются еще раз, они будут прочитаны не из основной памяти, а из кэш-памяти. Так как кэш-память имеет несколько уровней, в которых размещаются часто используемые файлы программ и данных, а также то, что кэш-память

частично расположена в кристалле самого процессора, экономия времени на загрузку процессора получается значительной.

Скорость выполнения команд с использованием кэш-памяти намного выше, чем скорость выборки команд и данных из памяти. В программных циклах при многократном выполнении одной и той же группы команд применение кэш-памяти дает максимальное ускорение вычислений.

Компьютеры часто работают в режиме разделения времени, когда процессор выполняет сразу несколько программ. В таких случаях система может стремиться не к минимизации общего времени, затраченного на выполнение одной программы, а к оптимизации пропускной способности. Поэтому часто возникает потребность в проведении черты между общим затраченным временем и временем работы процессора в наших интересах - процессорное время (CPU time). Эта черта учитывает время, которое центральный процессор затрачивает на работу над текущей задачей, время ожидания операций ввода-вывода или время, затраченное на выполнение других вспомогательных программ.

Необходимо сохранять различие между производительностью, основанной на общем затраченном времени, и производительностью, основанной на времени выполнения задачи центральным процессором. Общее затраченное время на работу системы обозначается термином «производительность системы», а для ссылки на процессорное время пользовательской программы используется термин «производительность центрального процессора».

Рассмотрим оценку влияния компонентов программного обеспечения на количество инструкций и производительность центрального процессора.

Производительность программы зависит от алгоритма, языка, компилятора, архитектуры и используемого аппаратного обеспечения. Оценка влияния этих компонентов на факторы, фигурирующие в уравнении производительности центрального процессора.

Алгоритм. Алгоритм определяет количество инструкций, необходимых для выполнения исходной программы процессором. Алгоритм может также влиять на быстродействие за счет реализации более медленных или более быстрых инструкций. Например, если

алгоритм использует больше операций с плавающей точкой, это может привести к более высокому быстродействию.

Язык программирования. Язык программирования влияет на количество инструкций, поскольку инструкции языка транслируются в инструкции процессора, определяющие общее количество выполняемых инструкций. Язык может также влиять на быстродействие: язык с мощной поддержкой абстракций данных (к примеру, Java) может потребовать применения опосредованных вызовов, использующих более затратные, с точки зрения быстродействия, инструкции.

Компилятор. Эффективность работы компилятора влияет и на количество инструкций, и на среднее количество тактов, приходящееся на одну инструкцию, поскольку компилятор определяет ход трансляции инструкций языка в инструкции процессора. Компилятор может играть весьма сложную роль и осуществлять комплексное воздействие на быстродействие процессора.

Архитектура набора команд. Архитектура набора команд воздействует на все три аспекта производительности центрального процессора, поскольку она влияет на инструкции, на количество циклов, необходимое для выполнения каждой инструкции, и, в общем, на быстродействие процессора.

О тактовой частоте процессора, говорилось ранее. Рассмотрим основную формулу вычисления производительности.

Одним из показателей общего времени выполнения программы является процессорное время. Предположим, что для реализации программы, написанной на одном из языков высокого уровня, требуется T секунд процессорного времени. Компилятор генерирует соответствующую объектную программу на машинном языке. Пусть для полного выполнения такой программы нужно произвести N команд машинного языка. Значение N – это количество машинных команд, которые будут реально выполнены, оно не обязательно равно количеству команд в объектной программе. Некоторые команды могут выполняться более одного раза, например в циклах. Другие команды могут вообще не выполняться.

Пусть среднее количество базовых шагов, необходимых для выполнения одной машинный команды, равно S, при этом каждый базовый шаг производится за один такт процессора. Если тактовая частота процессора равна R тактам в секунду, то время выполнения программы составит:

$$T = (NxS)/R \qquad (4.1)$$

Это равенство часто называют основной формулой вычисления производительности. Из нее вытекает, что для увеличения показателя производительности необходимо предельно уменьшить значения N и S и увеличить значение R. Значение N уменьшается, когда исходная программа компилируется в объектную программу с меньшим количеством команд. Значение S уменьшается, когда процесс выполнения команды состоит из меньшего количества базовых шагов или же если некоторые шаги команд могут выполняться одновременно. С повышением тактовой частоты повышается значение R и сокращается время выполнения базового шага команды.

Параметры N, S и R не являются независимыми, т.к. изменение одного из них может повлиять на величину другого. Процессор с частотой 900 МГц не всегда будет быстрее процессора с частотой 700 МГц, если у него будет другое значение параметра S.

Вопросы для контроля

1. Перечислите наиболее важные характеристики процессора.
2. В чем основная задача процессора в компьютере?
3. Какие блоки входят в состав процессора?
4. Перечислите основные параметры процессоров.
5. Рассмотрите типичный процесс выполнения программы?
6. Каковы функции регистра признаков результата в АЛУ?
7. Каков порядок взаимодействия УУ с основной и регистровой памятью?
8. В чем заключаются функции управляющего блока?
9. В чем причины аппаратных и программных прерываний?
10. Каков цикл команды прерывания программы?
11. Рассмотрите принцип использования стековой памяти.
12. Каково влияние компилятора на быстродействие процессора?

ГЛАВА 5. ОРГАНИЗАЦИЯ ОПЕРАТИВНОЙ ПАМЯТИ КОМПЬЮТЕРОВ

5.1. Иерархическая организация памяти

Память — совокупность устройств, служащих для приема, хранения и выдачи данных в центральный процессор или внешнюю среду компьютера. Основные операции с памятью — запись и чтение. В вычислительных системах память является одним из основных компонентов, определяющим как быстродействие, так и функциональные возможности всей системы. Организация памяти имеет сложный характер и строится по иерархическому принципу. Основная идея иерархии памяти – согласование скоростей работы операционных устройств, в первую очередь процессора, с запоминающими устройствами. Энергозависимая память используется для хранения данных и программ во время работы, энергонезависимая память используется для хранения данных и программ между сеансами работы. Для энергозависимой используется термин **основная**, или DRAM-память (Dynamic Random Access Memory – динамическая память с произвольным доступом), а для энергонезависимой используется термин **вторичная память.**

Таким образом, под основной памятью мы будем иметь в виду все ее виды (ОЗУ, РОН, кэш-память всех уровней), кроме дисковой и флэш-памяти. Под оперативной памятью (ОП) будем иметь в виду адресуемые ОЗУ и ПЗУ. Главным энергонезависимым запоминающим устройством, серверных компьютеров и на рабочих станций, является жесткий диск. Флэш-память, энергонезависимая полупроводниковая память, используется вместо дисков в мобильных устройствах, таких как сотовые телефоны, и она все чаще заменяет диски в музыкальных плеерах и даже в ноутбуках. Подробное строение и принцип их работы представлен в последующих разделах. Иерархия уровней памяти на рис.5.1. Основными характеристиками памяти являются:

- скорость чтения и записи;
- емкость;
- монтажное местоположение;
- метод доступа.

Рис.5.1. Уровни размещения видов памяти компьютера

Первый и самый скоростной тип памяти относительно небольшого объема составляют РОН и кэш L1, находящиеся на первом уровне иерархии. Оба эти вида памяти технологически размещаются непосредственно в кристалле процессора. Оба виды памяти недоступны программисту, это функции операционной системы. Это сверхоперативная неадресуемая память, причем если число регистров РОН относительно невелико (общий объем сотни байт), то кэш–память L1 в современных компьютерах уже имеет емкость до 256 Кб. Второй уровень памяти имеет кэш уровня L2, адресуемые оперативную (ОЗУ) и постоянную (ПЗУ) память. Они технологически располагаются на материнской плате компьютера (в некоторых более современных процессорах кэш L2 располагается в кристалле процессора). Объем кэш L2 составляет 5-10 Мбайт.

Третий уровень содержит кэш L3 и дисковую память, расположенные отдельно как внешние устройства и самостоятельные платы. Объем памяти L3 составляет десятки Мбайт. Объемы оперативной и дисковой памяти изменяются уже в Гигабайтах и Терабайтах.

По месту расположения запоминающие устройства делятся на процессорные, внутренние и внешние (рис.5.2).

К внешним запоминающим устройствам относятся дисковые накопители большой емкости. Это CD/DVD – накопители на оптических дисках, HDD – накопители на жестких магнитных дисках, SSD–накопители на скоростных флэш-микросхемах. Они подключаются к компьютеру аналогично устройствам ввода/вывода через шину внешних устройств. Из-за относительной низкой скорости ввода и вывода данный тип памяти в процессе оперативной реализации команд программы участия не принимает, т.к. программы и данные, которые они хранят, перед непосредственным применением переносятся во внутреннюю, основную память.

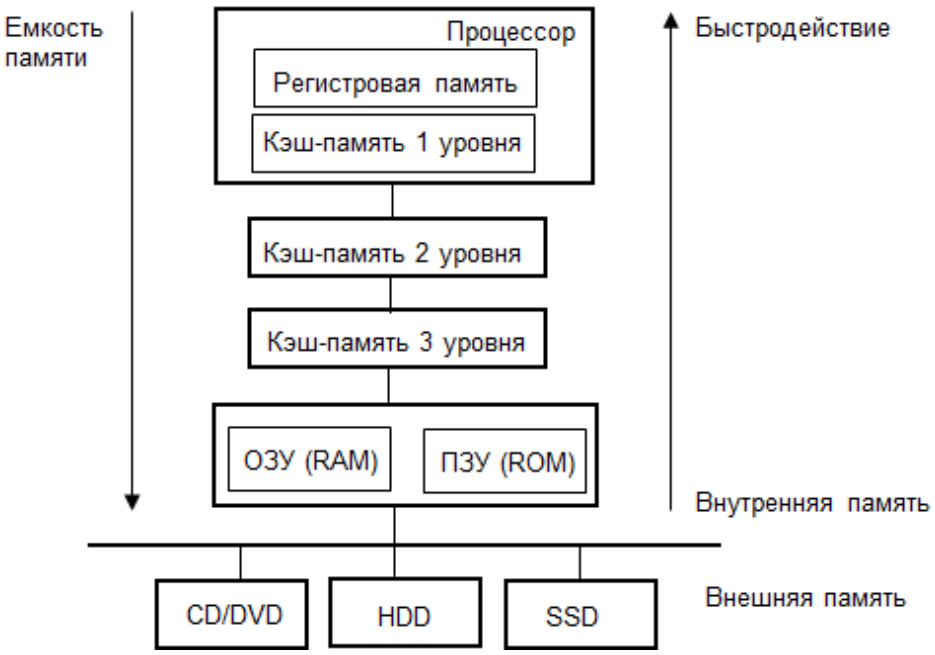


Рис.5.2. Уровни иерархии памяти

Программы компьютера обычно представляют в виде блоков – функционально автономных групп команд и данных, обрабатываемых в соответствии с общим алгоритмом вычислений. Эти блоки (фрагменты общей программы) заранее переписываются из более медленных видов памяти (дисков, магнит лент) в основную память. Будучи один раз переписаны из медленной в более быструю память эти блоки могут многократно использоваться и обеспечивать более высокую скорость обработки. Такой принцип хранения и перемещения данных используется на всех иерархических уровнях памяти от сверх оперативных РОН до медленных дисковых накопителей большой емкости.

Иерархическая организация памяти показывает, что на более высоких уровнях иерархии расположены устройства с меньшей емкостью памяти, но с большим быстродействием.

5.2. Взаимодействие процессора и различных уровней памяти

Уровни иерархии памяти взаимосвязаны: все данные на одном уровне могут быть также найдены на более низком уровне, и все данные на этом более низком уровне могут быть найдены на следующем нижележащем уровне и так далее. В каждый момент времени идет обмен с двумя близлежащими уровнями. Минимальная

единица информации, которая может либо присутствовать, либо отсутствовать в двухуровневой иерархии, называется блоком. Размер блока может быть либо фиксированным, либо переменным. Если этот размер зафиксирован, то объем памяти является кратным размеру блока.

Успешное или неуспешное обращение к более высокому уровню называются соответственно попаданием (hit) или промахом (miss).

Попадание – обращение к объекту в памяти, который найден на более высоком уровне, в то время как промах означает, что он не найден на этом уровне. Доля попаданий или коэффициент попаданий есть доля обращений, найденных на более высоком уровне. Доля промахов есть доля обращений, которые не найдены на более высоком уровне.

Потери на промах – время для замещения блока в более высоком уровне на блок из более низкого уровня плюс время для пересылки этого блока в требуемое устройство (обычно в процессор). Потери на промах включают в себя две компоненты: время доступа – время обращения к первому слову блока при промахе, и время пересылки – дополнительное время для пересылки оставшихся слов блока. Время доступа связано с задержкой памяти более низкого уровня, в то время как время пересылки связано с полосой пропускания канала между устройствами памяти двух смежных уровней.

Инициатором обращения к памяти практически всегда является процессор. Исключение – режим прямого доступа к памяти, когда организуется процесс передачи файлов между основной памятью и внешней памятью через соответствующую шину, минуя процессор.

В процессе выполнения программы процессор обрабатывает каждую команду и определяет исполнительный адрес операнда. При этом процессор не знает, на каком уровне памяти находится этот исполнительный адрес, поэтому сразу формируется обращение к ОЗУ. Далее происходит поиск данных по исполнительному адресу на различных уровнях памяти.

Сложность в том, что на каждом уровне данные представлены различными способами. Будем рассматривать блочную организацию (хранение) данных. В регистровой памяти данные хранятся блоками в виде слов длиной 16, 32, 64 и 128 бит. В кэш-памяти блоком является

строка длиной 16, 32 или 64 байта. В ОЗУ чаще всего в качестве блоков хранения используются страницы по 4 – 8 КВ. На жестких дисках блоки – это сектора по 512 байт. Как правило, размер страницы ОЗУ кратен длине сектора винчестера.

Если в системе есть кэш-память, то контроллер кэша проверяет, содержит ли кэш запрашиваемый исполнительный адрес. Если данные с таким адресом есть, то блок с этими данными считывается из кэш-памяти в процессор, а обращение к ОЗУ блокируется. Если в кэш-памяти нет данных с таким адресом, то нужный блок ищется в оперативной памяти, затем загружается в кэш-память и одновременно передается в процессор.

Аналогично, при обращении к основной памяти при попадании блок данных передается в процессор. При промахе данные загружаются с жесткого или оптического диска в ОЗУ.

5.3. Адресуемая память

Сегодня, есть три основных технологии, используемые в создании иерархий памяти. Основная память осуществляется из памяти DRAM , в то время как уровень, находящийся ближе к процессору (кэш) использует память **SRAM** (Static Random Access Memory - статическая память с произвольным доступом). DRAM является менее затратным в расчете на бит, чем статические ОЗУ, хотя и существенно медленнее. Разница в цене объясняется тем, что DRAM использует значительно меньшие площади, приходящейся на бит памяти, и DRAM, таким образом, имеют большую емкость при том же количестве кремния. Третья технология, используемая, для реализации самого большого и самого медленного уровня в иерархии, является обычно магнитным диском. Время доступа и стоимость немного различается среди этих технологий. Производители DRAM вчетверо увеличивали емкость каждые три года — 60% в год на протяжении 20 лет. В последние годы темпы роста снизились, стали близки к удвоению емкости каждые два-три года и достигли нескольких Гбайт.

Основная память в ее адресуемой части содержит ОЗУ и ПЗУ. Они размещаются в виде набора чипов на материнской плате. Адресуемую часть основной памяти образуют микросхемы с

произвольным доступом, т.е. обращение к любой ячейке занимает одно и то же время и производится в произвольной последовательности. Каждая ячейка имеет уникальный адрес и содержит фиксированное число запоминание элементов, соответствующих битам двоичных чисел.

Принцип построения ОЗУ и ПЗУ и способ доступа идентичен. В схемах памяти реализуется координатный принцип адресации ячеек.

Емкость основной адресуемой памяти достигла величин Гбайт, поэтому она технологически реализуется в виде нескольких больших интегральных схем, а увлечение разрядности ОЗУ или ПЗУ достигается за счет объединения по адресному входу нескольких микросхем памяти.

Обобщенная схема адресуемой памяти компьютера представлена на рис.5.3. Она состоит из полупроводниковой матрицы ячеек хранения двоичных слов и регистров доступа по адресу (РАП) и чтению/записи (РДП). Так как полупроводниковая матрица представляет собой трехмерный куб, то по одной координате (x) идет поиск адресов по строкам, по другой координате (y) поиск по столбцам. Третья координата – это биты читаемых слов.

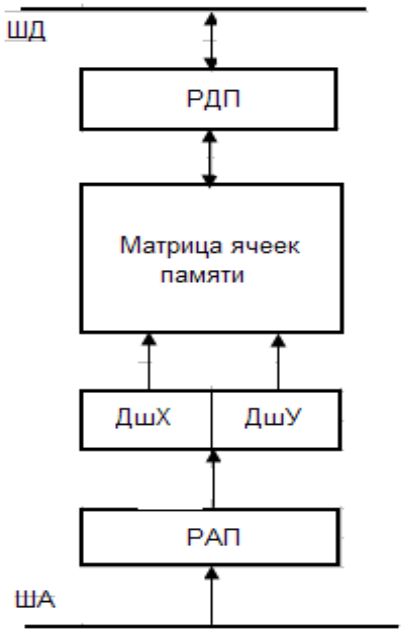


Рис.5.3. Обобщенная схема адресной памяти

Совокупность микросхем, образующих основную память (ОП) называют модулем памяти. На рис.5.4. представлена схема модульной организация памяти, где несколько микросхем модулей M_0, M_1,…, M_n

соединены по адресному входу A_0, …, A_n и по входу управления («Запись», «Чтение»).

По коду адреса, поступающему на входы всех модулей, выбирается один из модулей, который срабатывает либо на выдачу, либо на занесение числа через выходы $Д_0$, …, $Д_n$.

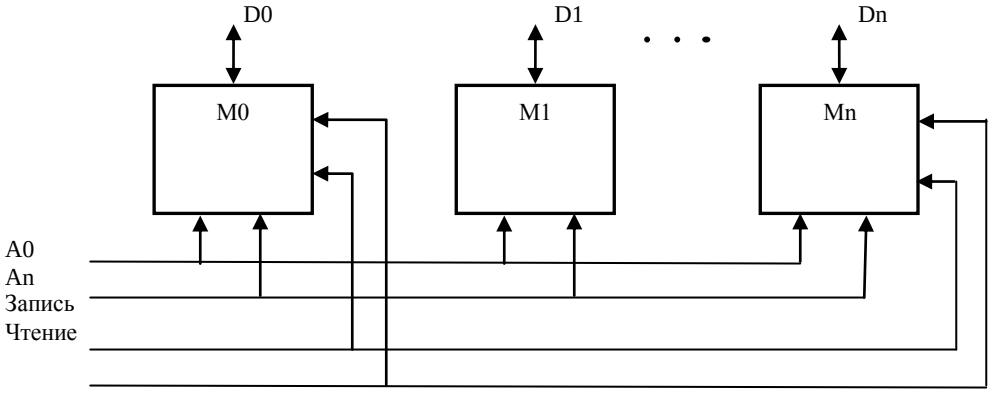


Рис. 5.4. Модульная организация памяти

Укрупненная структура оперативной памяти, состоящей из модулей, объединенных в блоки, представлена на рис.5.5.

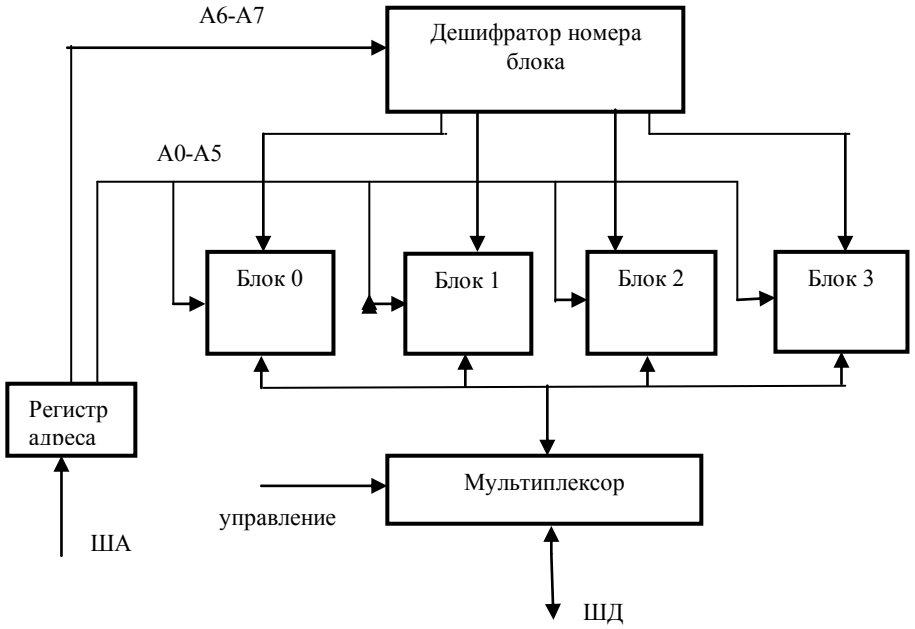


Рис.5.5. Структурная схема блоков оперативной памяти

Общее адресное пространство разбито на группы последовательных адресов, каждая группа сосредоточена в одном из блоков от 0 до 3. Семь младших разрядов адреса (A_0 – A_6) выбирают в каждом из блоков одну ячейку. С помощью двух старших разрядов (A_6 – A_7) осуществляется выбор одного из блоков для чтения или

записи данных. Такая структура ускоряет поиск необходимых данных или их размещение.

На рис. 5.6. показана структурная схема адресного запоминающего устройства. Запоминающая матрица имеет две координаты: строки и столбцы. Схема управления управляет устройствами ЗУ, получая извне сигналы: RAS, CAS, CE, WE и OE:

RAS – сигнал строба строки;

CAS - сигнал строба столбца;

WE – сигнал разрешения записи;

OE – разрешение выдачи выходных сигналов;

CS – выбор микросхемы.

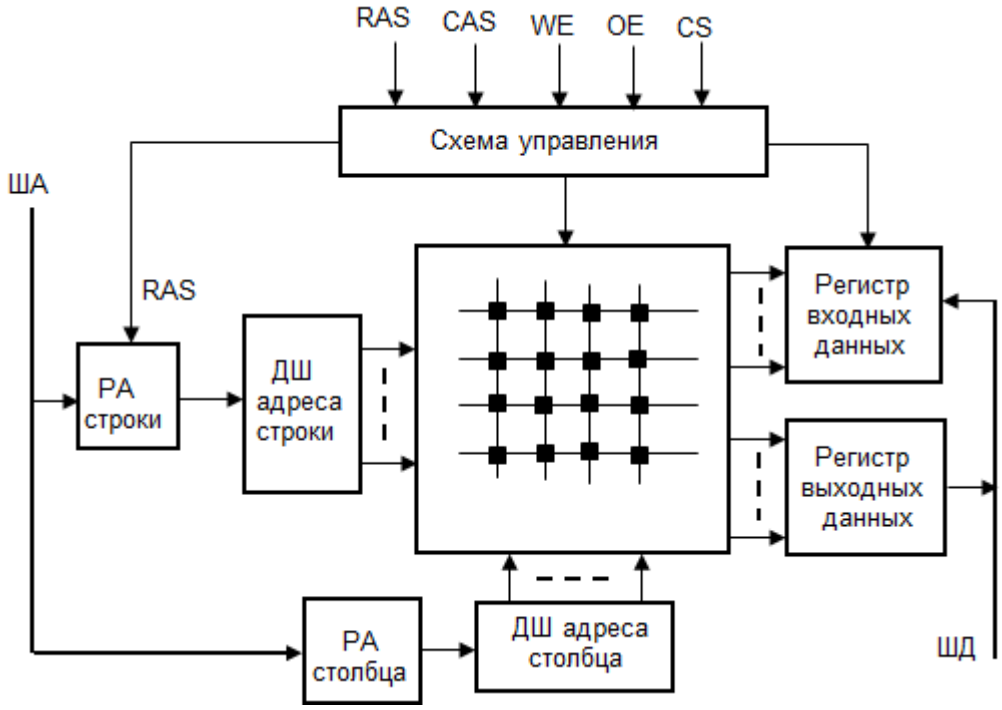


Рис.5.6. Схема организации памяти в отдельном блоке

Сигнал выбора микросхемы CE разрешает работу именно этой микросхемы памяти. Режим чтения или записи определяется сигналом WE. На все время, пока микросхема не использует шину данных ШД, информационные выходы регистров переводятся сигналом OE в третье состояние с высоким выходным сопротивлением.

Адрес строки на шине ША сопровождается сигналом RAS, разрешающим прием адреса и его дешифрацию (ДШ). После этого сигнал CAS разрешает прием и дешифрацию адреса столбца.

Управление операциями с памятью осуществляется контроллером памяти. На каждую операцию требуется, как минимум, пять тактов.

1. Указание типа операции (чтение или запись) и установка адреса строки.
2. Формирование сигнала RAS.
3. Установка адреса столбца.
4. Формирование сигнала CAS.
5. Запись или выдача данных и возврат сигналов RAS и CAS в неактивное состояние.

Под латентностью понимают задержку между поступлением команды в память и ее выполнением. Память не может мгновенно переходить из одного состояния в другое. Для стабильного функционирования памяти необходим пропуск нескольких циклов при изменении состояния ячейки памяти.

Например, после выполнения команды чтения должна следовать задержка CAS (CAS Latency). Это и есть латентность – наиболее важная характеристика памяти. Чем меньше латентность, тем быстрее работает память. В течение последних 25-ти лет латентность оперативной памяти уменьшилась всего в три раза. При этом тактовая частота процессоров возросла в сотни раз.

Организация постоянной памяти ПЗУ отличается от организации ОЗУ только отсутствием режима записи, существует только цикл чтения.

5.4. Ассоциативная память

В рассмотренных ранее видах запоминающих устройств доступ к информации требовал указания адреса ячейки. Зачастую значительно удобнее искать информацию не по адресу, а опираясь на какой-нибудь характерный признак, содержащийся в самой информации. Такой принцип лежит в основе устройства, известного как ассоциативное запоминающее устройство.

Понятие «ассоциация» относится, прежде всего, к памяти, в которой выборка осуществляется не по адресному принципу, а по содержанию.

Ассоциативная память использует запись и чтение данных таким образом, чтобы обеспечить выборку слов, имеющих заданное содержание определенных полей. Поиск ведется с использованием ассоциативных признаков. Структура такой памяти представлена на рис. 5.7.

Принятые на схеме обозначения:

ЗМ – запоминающая матрица;

ШП – шина признака;

ШД – шина данных;

УРВ – устройство разрешения выборки.

Память хранит M ячеек для m+1 – разрядных слов, имеющих значения признаков. Служебный m +1-й разряд показывает: 0 – ячейка свободна для записи, 1 - ячейка занята. Значения ассоциативного признака формируются регистром маски из полей признаков, поступающих из ШП в регистр ассоциативного признака.

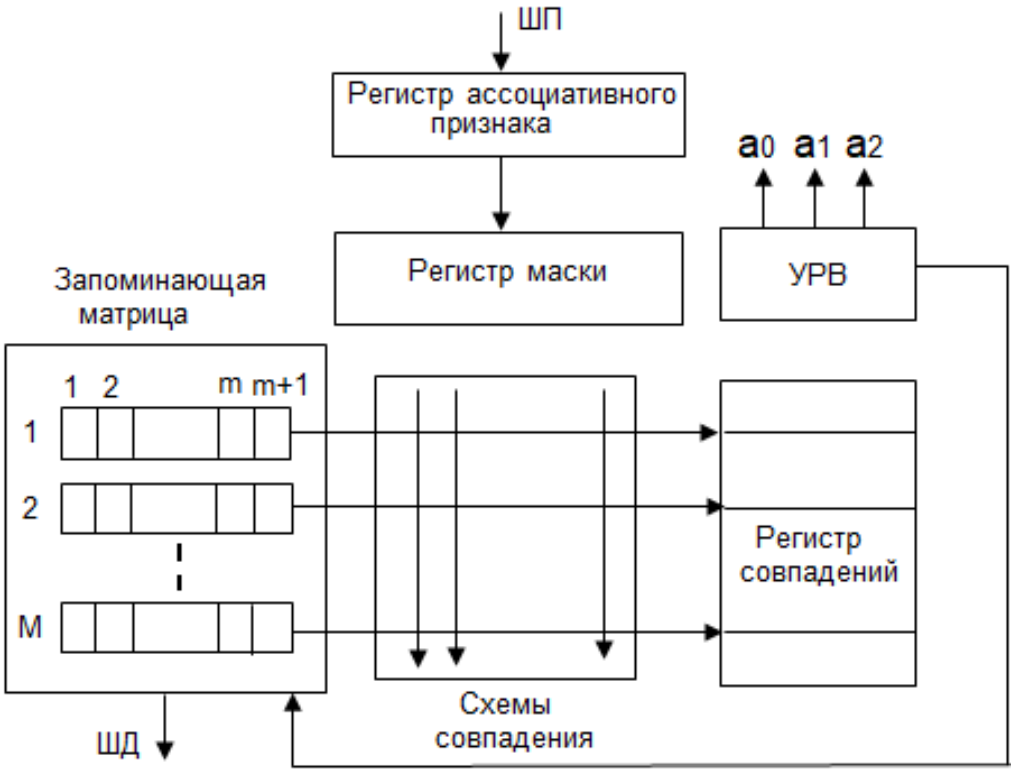


Рис.5.7. Схема ассоциативной памяти

Поиск в запоминающей матрице выполняется за один такт одновременно по полям ассоциативных признаков всех хранящихся слов. Это является отличительной чертой ассоциативных устройств

памяти. Реализация поиска осуществляется комбинационными схемами совпадения на базе элементов сложения по «модулю 2».

Схемы совпадения параллельно сравнивают каждый бит хранимых слов с соответствующим битом признака поиска. В регистре совпадений каждой строке ЗМ соответствует один разряд. В него заносится единица, если биты данной строки совпали со всеми одноименными битами признака поиска.

При выборке в регистре совпадений отмечаются строки с данными, имеющими одинаковые признаки. Таких данных может быть несколько, например, несколько команд.

Устройство разрешения выборки (УРВ) выбирает одно из этих данных (например, первое). Значения управляющих сигналов:

$a1 = 1$ – в ЗМ нет слов, совпадающих с признаком;

$a2 = 1$ – в ЗМ только одно слово, совпадающее с признаком;

$a2 = 1$ – в ЗМ несколько слов, совпадающих с признаком.

Управляющая схема выдает нужную строку из ЗМ на шину данных ШД.

Преимущества ассоциативной памяти в быстроте поиска слов и возможности выполнения логических операций над словами во время их поиска (поиск минимального или максимального элемента в массиве, поиск слов, заключенных в заданные границы). Недостаток: сложность аппаратной реализации блока одновременного сравнения слов с ассоциативными признаками.

5.5. Постоянные запоминающие устройства

Слово «постоянные» в названии этого вида запоминающих устройств относится к их свойству хранить информацию и служить источником данных только для чтения (ROM – Read-Only Memory) . Микросхемы ПЗУ построены по принципу матричной структуры накопителя, где в узлах расположены перемычки в виде проводников, полупроводниковых диодов или транзисторов, одним концом подключенных к адресной линии, а другим — к разрядной линии считывания.

В такой матрице наличие перемычки может означать 1, а ее отсутствие — 0. В некоторых типах ПЗУ элемент, расположенный на

перемычке, исполняет роль конденсатора. Тогда заряженное состояние конденсатора означает. 1, а разряженное — 0.

Основным режимом работы ПЗУ является считывание информации, которое мало отличается от аналогичной операции в ОЗУ как по организации, так и по длительности. В то же время запись в ПЗУ по сравнению с чтением обычно сложнее и связана с большими затратами времени и энергии.

Занесение информации в ПЗУ называют программированием или «прошивкой». Последнее название напоминает о том, что первые ПЗУ выполнялись на базе магнитных сердечников, а данные в них заносились путем прошивки соответствующих сердечников проводниками считывания. Современные ПЗУ реализуются в виде полупроводниковых микросхем, которые по возможностям и способу программирования разделяют на:

- программируемые при изготовлении;
- однократно программируемые после изготовления;
- многократно программируемые в процессе использования.

ПЗУ, программируемые при изготовлении.

Эту группу образуют так называемые масочные устройства и именно к ним принято применять аббревиатуру ПЗУ. Для масочных ПЗУ обозначением является ROM, совпадающее с общим названием всех типов ПЗУ.

Занесение информации в масочные ПЗУ составляет часть производственного процесса и заключается в подключении или неподключении запоминающего элемента к разрядной линии считывания. В зависимости от этого из памяти будет всегда извлекаться 1 или 0. В роли перемычки выступает транзистор, расположенный на пересечении адресной и разрядной линий. Какие именно запоминающие элементы должны быть подключены к выходной линии, определяет маска, «закрывающая» определенные участки кристалла.

В начальный период масочные микросхемы были дороги, однако сейчас это один из наиболее дешевых видов ПЗУ. Для ROM характерна высокая плотность упаковки запоминающих элементов на кристалле и высокие скорости считывания информации. Основной сферой применения являются устройства, требующие хранения

фиксированной информации. Так, подобные ПЗУ часто используют для хранения шрифтов в лазерных принтерах.

Однократно программируемые ПЗУ. Создание масок для ROM оправдано при производстве большого числа копий. Если требуется относительно небольшое количество микросхем с данной информацией, разумной альтернативой являются однократно программируемые ПЗУ.

В микросхемах типа PROM (Programmable ROM - программируемые ПЗУ) информация может быть записана только однократно. Первыми такими ПЗУ стали микросхемы памяти на базе плавких предохранителей. В исходной микросхеме во всех узлах адресные линии соединены с разрядными. Занесение информации в PROM производится электрически, путем пережигания отдельных перемычек, и может быть выполнено поставщиком или потребителем спустя какое-то время после изготовления микросхемы. Позже появились микросхемы, где в перемычку входили два диода, соединенные навстречу. В процессе программирования удалить перемычку можно было с помощью электрического пробоя одного из этих диодов. В любом варианте для записи информации требуется специальное оборудование - программаторы. Основными недостатками данного вида ПЗУ были большой процент брака и необходимость специальной термической тренировки после программирования, без которой надежность хранения данных была невысокой.

Многократно программируемые ПЗУ. Процедура программирования таких ПЗУ обычно предполагает два этапа: сначала производится стирание содержимого всех или части ячеек, а затем производится запись новой информации. В этом классе постоянных запоминающих устройств выделяют несколько групп:

- EPROM (Erasable Programmable ROM - стираемые программируемые ПЗУ);
- EEPROM (Electrically Erasable Programmable ROM - электрически стираемые программируемые ПЗУ);
- флэш-память.

В EPROM запись информации производится электрическими сигналами, так же как в PROM, однако перед операцией записи

содержимое всех ячеек должно быть приведено к одинаковому состоянию (стерто) путем воздействия на микросхему ультрафиолетовым облучением. Процесс стирания может выполняться многократно. Каждое стирание занимает порядка 20 мин.

Более привлекательным вариантом многократно программируемой памяти является электрически стираемая программируемая постоянная память EEPROM. Стирание и запись информации в эту память производятся побайтно, причем стирание - не отдельный процесс, а лишь этап, происходящий автоматически при записи. Операция записи занимает существенно больше времени, чем считывание — несколько сотен микросекунд на байт. В микросхеме используется тот же принцип хранения информации, что и в EPROM. Программирование EPROM не требует специального программатора и реализуется средствами самой микросхемы.

Еще один вид полупроводниковой памяти - флэш-память будет рассмотрена при изучении внешних устройств компьютера.

5.6. Организация кэш-памяти

Основная задача кэш-памяти – согласование работы быстрого процессора и медленной оперативной памяти ОП. Кэш-память исполняет роль буфера между оперативной памятью и процессором (рис. 5.8.).

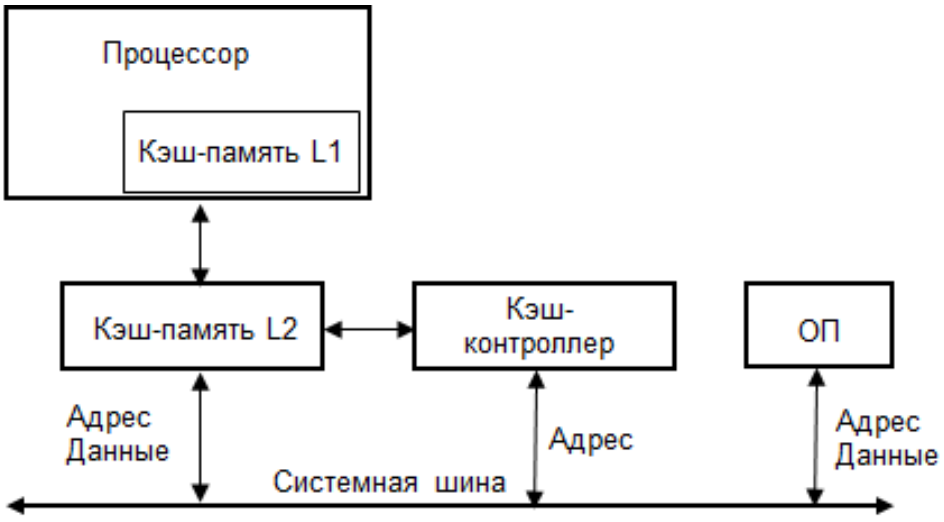


Рис. 5.8. Организация кэш-памяти в компьютере

Использование кэш-памяти базируется на принципе локальности ссылок. Это означает, что следующее обращение к памяти в

программе с большой вероятностью произойдет к тому же блоку данных, который находится в данный момент в кэш-памяти.

Кэш разбивается на строки по 16 или 32 байта, соответствующие одному стандартному пакетному циклу обращения к оперативной памяти. Обмен информацией между ОП и кэш-памятью осуществляется строками, даже если необходимо передать только один байт. Процессор, выполняя команду, запрашивает операнд по некоторому адресу в адресном пространстве оперативной памяти. Кэш-контроллер проверяет, есть ли в кэше строка данных, соответствующая этому адресу. На такие же строки условно разделяются и страницы оперативной памяти.

В случае наличия искомой строки ситуация называется кэш-попадание. Если нужной строки в кэш-памяти нет, то происходит кэш-промах, и кэш-контроллер инициирует обращение к оперативной памяти для переписи из нее нужной строки в кэш-память.

В связи с этим возникает проблема замены какой-либо строки в кэше на новую строку из оперативной памяти. Для этого используют специальные дисциплины замещения строк. Таким образом, функциями кэш-контроллера являются:

- хранение информации об адресах строк данных, находящихся в кэш-памяти;
- хранение предыстории обращений к строкам в кэш-памяти;
- замещение строк в кэш-памяти в случаях кэш-промахов;
- контроль системной шины для выявления обращений к ОП со стороны других устройств.

На рис.5.9. приведена типовая структура внутрипроцессорной кэш–памяти. Оперативная память состоит из 2^n адресуемых слов со своими n–бит адресами. Область оперативной памяти разбивается на М блоков фиксированной длины по К слов в каждом блоке.

Кэш-память состоит из С блоков (строк), каждый из которых имеет размер (длину) в К слов, т.е. в одной строке кэш–памяти помещается один блок оперативной. При считывании один блок оперативной памяти копируется в одну строку кэш. Так как объем оперативной памяти намного больше числа строк в кэш, то блоки данных из ОЗУ размещаются в текущем режиме (в последовательности выполнения, заданной общим алгоритмом

решения задачи) в свободные строки кэш. Сведения о том, какой блок ОЗУ размещен в данной строке кэш, содержится в разряде Тег (признак блока).

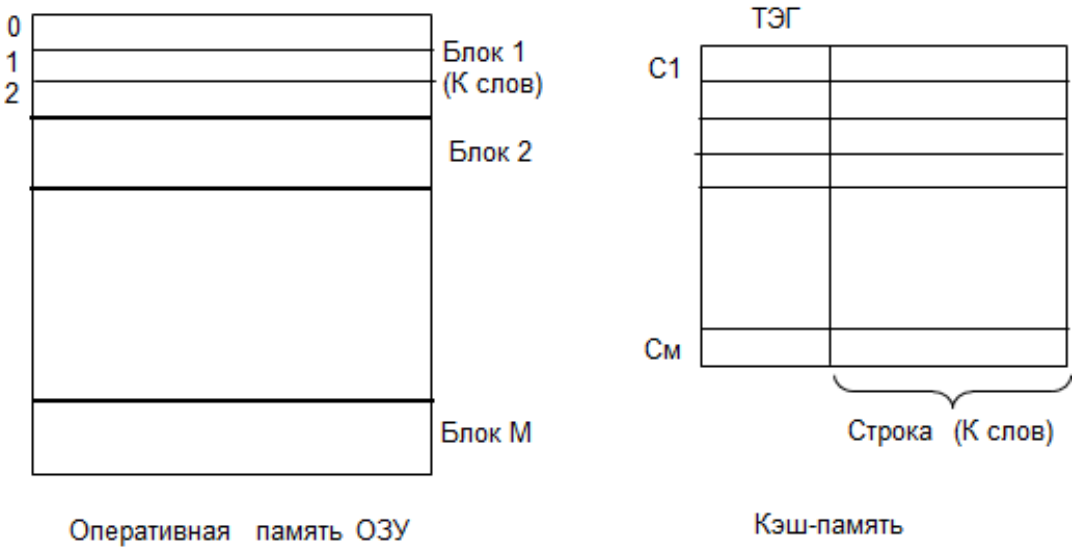


Рис.5.9. Типовая структура кэш-памяти

Типы кэш-памяти.

В современных вычислительных системах используются три типа организации кэш-памяти.

1. Кэш с прямым отображением.
2. Полностью ассоциативная кэш-память.
3. Множественно-ассоциативная кэш-память.

Рассмотрим организацию самого простого типа кэш-памяти.

Кэш с прямым отображением.

Каждая строка кэш-памяти может содержать строку основной памяти только из определенного подмножества адресов, причем эти подмножества не пересекаются. Поиск состоит из следующих шагов:

- определение, в какое из подмножеств адресов основной памяти попадает адрес строки, выработанный процессором;
- обращение к единственной соответствующей строке и сравнение ее тега с адресом от центрального процессора для определения, является ли эта строка искомой.

На рис.5.10. приведен пример структуры кэш-памяти с прямым отображением.

Для простоты рассмотрим оперативную память, содержащую 16 строк данных, и кэш-память объемом в четыре строки. Собственно

микросхема кэш-памяти содержит только данные в виде строк. При этом в одной строке находятся несколько слов с последовательными адресами. В кэш-контроллере организована память тегов и индексов строк, а также блоки выборки строк и сравнения тегов.

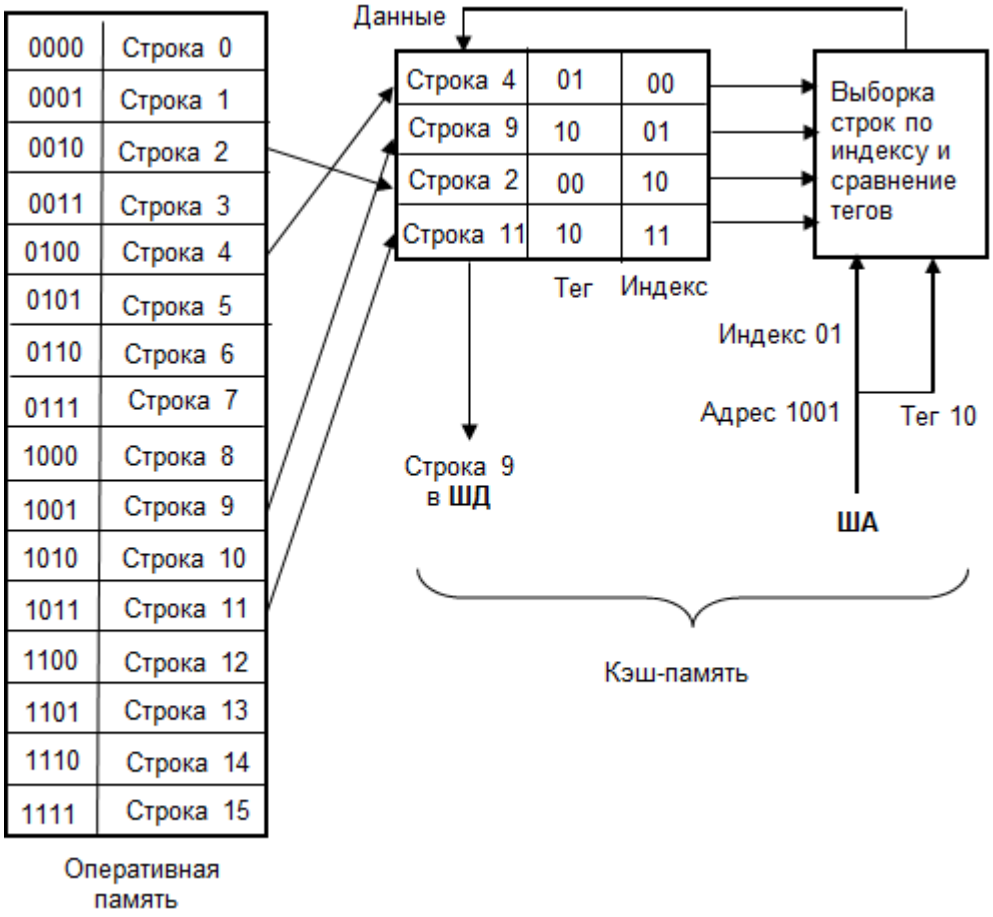


Рис.5.10. Структура кэш-памяти с прямым отображением

Все строки оперативной памяти, имеющие S одинаковых младших разрядов, объединяются в подмножества. Эти подмножества могут отображаться в кэш-памяти только в строке с индексом, равным коду этих разрядов. В нашем примере индекс образуют два младших разряда. Следовательно, например, строки 1, 5, 9 и 13 могут находиться только в строке кэш-памяти с индексом 01 и ни в какой другой строке.

В общем случае, если разрядность адреса ОЗУ равна N, а разрядность индекса – n, то адресные теги содержат оставшиеся N-n разрядов адреса строки.

Преимущество такой кэш-памяти в простоте организации и низкой стоимости. Основной недостаток – ограниченное число

комбинаций строк в кэш-памяти, что приводит к увеличению процента кэш-промахов. Например, строки 5 и 9 не могут одновременно находиться в кэш-памяти, даже если есть свободные места в строках с другими индексами.

Технологии кэш-памяти с другими типами отображения отличаются способом логического соответствия адресов и данных в ОЗУ и кэш-памяти.

В полностью ассоциативной памяти любая строка ОЗУ может находиться в любой строке кэш-памяти и входить при этом в любые комбинации с другими строками. Комбинационные схемы сравнения одновременно анализируют все теги строк, находящихся в кэше в данный момент, и сравнивают их с адресом, поступившим с шины адреса от процессора.

При кэш-попадании найденная строка считывается в шину данных. При кэш-промахе происходит замещение строки в кэш-памяти на требуемую строку, находящуюся в ОЗУ. Преимущество данной памяти в высокой скорости считывания. Недостаток – сложность аппаратной реализации. Поэтому полностью ассоциативная кэш-память чаще всего используется в специализированных буферах, таких, как буфер адресов переходов, с небольшим объемом строк.

Множественно-ассоциативный кэш является промежуточным между двумя выше рассмотренными типами памяти. В нем сочетаются простота кэша с прямым отображением и скорость ассоциативного поиска. Кэш-память делится на непересекающиеся подмножества строк. Каждая строка основной памяти может попадать в любое место только одного подмножества кэша. Для поиска подмножеств используется прямое отображение, а для поиска внутри подмножества используется полностью ассоциативный поиск. Число строк в подмножестве кэша определяет число входов (портов) самого кэша.

В современных процессорах используется 4-х, 6-ти и 8-ми канальная кэш-память. Увеличение числа каналов (входов) кэш-памяти приводит к быстрому увеличению сложности аппаратной реализации той части кэша, которая обеспечивает ассоциативный поиск строк по тегам.

Смешанная и разделенная кэш-память. Когда в микропроцессорах впервые стали применять внутреннюю кэш-память, ее обычно использовали как для команд, так и для данных; Такую кэш-память принято называть смешанной, а соответствующую архитектуру - **Принстонской**, по названию университета, где разрабатывались компьютеры с единой памятью для команд и данных, то есть соответствующие классической архитектуре фон-Неймана. Позже стало обычным разделять кэш-память на две — отдельно для команд и отдельно для данных. Подобная архитектура получила название **Гарвардской**, поскольку именно в Гарвардском университете был создан компьютер «Марк-1», имевший раздельные ЗУ для команд и данных.

Смешанная кэш-память обладает тем преимуществом, что при заданной емкости' ей свойственна более высокая вероятность попаданий по сравнению с разделенной, поскольку в ней оптимальный баланс между командами и данными устанавливается автоматически. Так, если в выполняемом фрагменте программы обращения к памяти связаны в основном с выборкой команд, а доля обращений к данным относительно мала, кэш-память имеет тенденцию насыщаться командами, и наоборот.

С другой стороны, при раздельной кэш-памяти выборка команд и данных может производиться одновременно, при этом исключаются возможные конфликты. Последнее обстоятельство существенно в системах, использующих' конвейеризацию команд, где процессор извлекает команды с опережением и заполняет ими буфер или конвейер.

Идея кэширования распространена и на другие устройства компьютера. В первую очередь это касается жестких дисков. Возможны два варианта:

- операционная система использует часть оперативной памяти в качестве кэша дисковых операций для внешних устройств, не обладающих собственной кэш-памятью, в том числе жестких дисков, flash памяти и гибких дисков;

- используется отдельная дисковая кэш-память объемом от 8 до 64 Мбайт.

Во втором случае контроллер дисковой кэш-памяти пересылает между ОЗУ и винчестером файлы, содержащие сектора или дорожки диска. Устройства чтения CD/DVD/BD-дисков также кэшируют прочитанную информацию для ускорения повторного обращения. Применение кэширования внешних накопителей обусловлено следующими факторами:

- скорость доступа процессора к оперативной памяти во много раз больше, чем к памяти внешних накопителей;
- некоторые блоки памяти внешних накопителей используются несколькими процессами одновременно, и имеет смысл прочитать блок один раз, а затем хранить его копию блока в оперативной памяти для всех процессов.

Дисковая кэш-память использует ассоциативный принцип поиска данных, а также обнаружение тройных и исправление двойных ошибок.

Одноуровневая и многоуровневая кэш-память. Кэш-память первого уровня (L1) размещается в кристалле процессора и строится по технологии статического ОЗУ. Емкость ее не превышает сотен Кбайт. Увеличение емкости L1 приводит к снижению скорости чтения/записи из-за усложнения схем управления и схем дешифратора адреса.

Кэш-память второго уровня (L2) размещается на материнской плате и является расширением кэш-памяти первого уровня. Емкость L2 на порядок больше емкости L1, а быстродействие несколько ниже. Память второго уровня тоже строится как статическое ОЗУ. Типичная емкость L2 составляет 512 Кб – 1 Мбайт, реализуется она в виде отдельной микросхемы.

Следует отметить, что в настоящее время для компьютеров уже не выпускаются одноядерные процессоры, преобладают в зависимости от сферы применения двухъядерные, четырехъядерные и восьмиядерные процессоры. В них для каждого их ядер предусматриваются свои уровни размещения кэш-памяти. Поэтому, если кэш-память L1 имеет емкость 64 Кбайт, то это значит, что каждое ядро располагает таким объемом памяти. Тогда общий объем кэш-памяти первого уровня восьмиядерного компьютера будет составлять 512 Кбайт.

Как правило, кэш второго уровня L2 также придается каждому ядру, исключая случаи, когда компьютер имеет всего два уровня кэш-памяти и L2 является общей для всех ядер процессора. Кэш-память третьего уровня размещается в виде отдельной микросхемы на материнской плате компьютера, является общей памятью для всех ядер процессора и имеет объем до нескольких десятков Мбайт.

Многоуровневые устройства кэш-памяти позволяют легче использовать дополнительную оптимизацию благодаря двум обстоятельствам. Во-первых, параметры конструирования кэш-памяти нижнего уровня отличаются от технологий для кэш-памяти первою уровня. Например, поскольку кэш память нижнего уровня будет иметь значительно больший размер, появляется возможность использования более крупных блоков. Во-вторых, кэш память нижнего уровня не используется процессором на постоянной основе, в отличие от кэш-памяти первого уровня. Это позволяет рассмотреть возможность использования низкоуровневой кэш-памяти для какой-нибудь другой работы в период, когда к ней нет обращения.

Еще одним подходом является предварительная выборка (предвыборка), при которой блок запрашиваемых данных переносится в кэш-память еще до того, как к нему последует обращение. Многие современные процессоры используют средства аппаратной предвыборки, пытаясь предсказать обращения, которые может быть трудно заметить при использовании программных средств.

Когерентность данных в мультипроцессорных системах. Когерентность данных в памяти многопроцессорной системы обеспечена, если каждая операция чтения по какому-либо адресу, выполненная любым из процессоров, возвращает значение, занесенное в ходе выполнения последней операции записи по этому адресу, независимо от того, какой процессор выполнял эту операцию последним. Проблема когерентности данных в различных устройствах памяти особенно остра в мультипроцессорных системах с разделяемой памятью. На рис. 5.11 приведена структура компьютерной системы, содержащей процессоры с локальными кэш, а также общую (разделяемую) основную память.

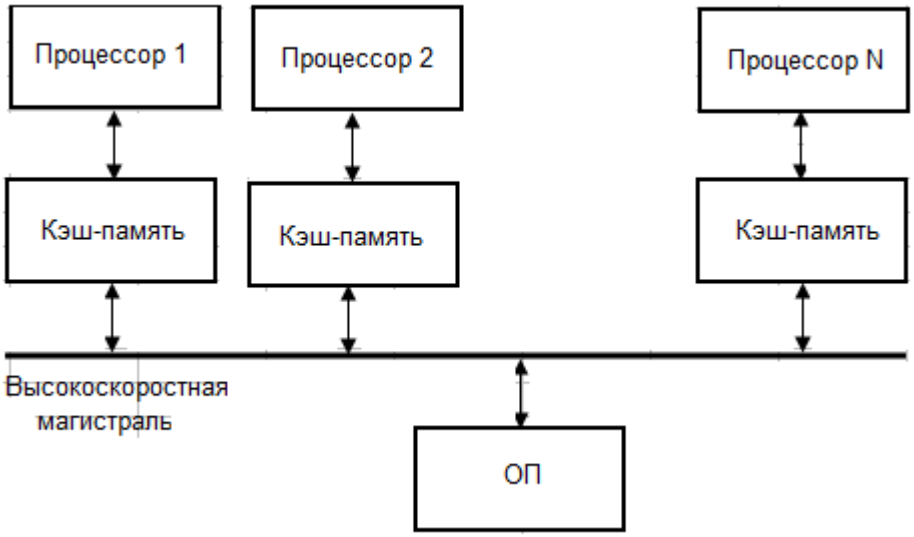


Рис. 5.11. Общая структура мультипроцессорной системы с разделяемой памятью

Доступ к локальным кэшам возможен от других процессоров через высокоскоростную сеть. Кэш-память каждого процессора может содержать данные двух типов: частные и разделяемые. С частными данными работает только один процессор. Разделяемые данные могут использоваться несколькими процессорами и загружаться сразу в несколько кэшей. Если информация в основной памяти и в кэш-памятях совпадает, то говорят, что они когерентны. Потеря этого свойства происходит при записи процессором данных в разделяемый блок.

Обеспечение когерентности предполагает, что любой процессор при обращении к разделяемым данным должен считывать последнее, записанное в них значение. Запись нескольких процессоров в свои кэши необходимо согласовать. Когерентность можно обеспечить при соблюдении следующих условий:

- если чтение и запись в одну и ту же ячейку памяти достаточно отделены друг от друга по времени;
- если несколько операций записи в одну и ту же ячейку выполняются строго последовательно.

Эти условия регламентируются и поддерживаются системными протоколами. Такие протоколы называются протоколами когерентности кэш-памяти.

Существует два класса протоколов:

- протоколы наблюдения;

- протоколы на основе справочников.

Протоколы наблюдения являются наиболее распространенными в современных системах, так как они предназначены для структур, подобных архитектуре современных компьютеров: процессоры с кэш-памятью, соединенные шиной или сетью связи с общей оперативной памятью. Каждый кэш хранит строку данных и информацию об ее состоянии (признак). Этот признак показывает:

а) разделяемая строка или нет;

б) модифицирована строка или нет.

Кэши располагаются на общей (разделяемой) шине, и их контроллеры наблюдают за шиной с целью обнаружения передачи разделяемых строк. Контроллер каждого кэша содержит блок слежения за системной шиной, который контролирует операции записи в кэш-память. При необходимости обновления информации в некотором кэше его процессор захватывает шину и передает по ней адрес строки. Все остальные процессоры анализируют этот адрес и проверяют, нет ли у них такой строки. Если она есть, то необходимо скорректировать свою информацию. Доступ к шине строго последовательный, поэтому все операции выполняются последовательно. Наибольший эффект в работе рассматриваемых систем достигается при использовании двухуровневых кэшей. Наблюдение за шиной выполняется кэшем второго уровня, а процессор, в основном, работает с первичной кэш-памятью. Такой подход позволяет снизить требования к полосе пропускания памяти.

В любых системах основные проблемы возникают при записи в разделяемую строку. Эта операция может выполняться одним из двух методов (протоколов):

1) записью с аннулированием;

2) записью с обновлением.

1. Запись с аннулированием. Если какой-либо процессор производит изменения в одной из строк своей локальной кэш-памяти, все имеющиеся копии этой строки в других локальных кэшах помечаются как недостоверные или аннулируются (бит достоверности обнуляется). Если другой процессор обращается к такой строке, то происходит кэш-промах и замещение корректным значением из той локальной памяти, где произошла модификация данных.

2. Запись с обновлением. Любая запись в локальную кэш-память дублируется в остальные локальные кэши, содержащие копии изменяемой строки. При этом дублирование в основную память может быть отложено. Этот метод требует широковещательной передачи данных по сети связи.

Рассмотренные методы имеют следующие достоинства и недостатки. Первый из них приостанавливает работу процессоров из-за конфликтов, а второй – требует увеличения полосы пропускания памяти. В последнем случае можно снизить интенсивность обмена за счет использования соответствующего признака строки («разделяемая» или нет). Наличие такого признака при записи с аннулированием также ускоряет работу системы (если строка не разделяемая, то аннулирование не нужно). В процессе выполнения программ статус строки может меняться. Если один из процессоров обнаружил, что другой обращается к неразделяемой строке в его кэше (по совпадению адресов оперативной памяти), то признак строки принимает значение «разделяемый».

Эксклюзивная и инклюзивная организация кэш-памяти.

В многоядерных процессорах структура кэш-памяти на кристалле содержит три уровня: L1, L2 и L3. Для обеспечения когерентности используются разные способы организации взаимодействия памятей различных уровней.

1. Эксклюзивная организация. Данные размещаются только на одном уровне и не дублируются на других уровнях. При первоначальной загрузке блок данных поступает в кэш L1, минуя кэш L2. При замещении этот блок из кэша L1 переписывается в кэш L2. При повторном обращении процессора к этому блоку данных он удаляется из кэша L2 и поступает в кэш L1. Аналогично взаимодействуют кэш-памяти L2 и L3. Преимущество эксклюзивного кэша в том, что общий размер кэшируемой информации равен суммарному объёму кэшей всех уровней. Это позволяет более эффективно использовать объем кэш-памяти.

2. Инклюзивная организация. Нижние уровни кэш-памяти гарантированно содержат данные, присутствующие в верхних уровнях кэш-памяти (т.е. расположенных ближе к процессорному ядру). Таким образом, при инклюзивной организации блоки данных

дублируются на всех уровнях. В процессорах AMD Phenom используется эксклюзивный кэш L3. В многоядерных процессорах Intel Core i7/i5 кэш L2 построен инклюзивно по отношению к кэшу L3. В последнем случае упрощается проверка когерентности: если данных в кэше L3 нет, то их нет и в кэшах L1 и L2. Если блок данных присутствует в кэше L3, то к нему привязаны четыре бита, показывающие в кэше какого ядра дублируется этот блок. В процессоре с архитектурой Nehalem при емкости кэше L3, равной 8 Мб, на дублирование данных потребуется максимум 1,25 Мб.

5.7. Виртуальная память

Для большинства типичных применений компьютеров размещение всей программы в оперативной памяти невозможно из-за ее большого размера. В этом, однако, и нет принципиальной необходимости, поскольку в каждый момент времени «внимание» машины концентрируется на определенных сравнительно небольших участках программы. Таким образом, в ОП достаточно хранить только используемые в данный период части программ, а остальные части могут располагаться на внешних запоминающих устройствах (ВЗУ). Сложность подобного подхода в том, что процессы обращения к ОЗУ и ВЗУ существенно различаются, и это усложняет задачу программиста. Выходом из такой ситуации было появление в 1959 году идеи виртуализации памяти, под которой понимается метод автоматического управления иерархической памятью, при котором программисту кажется, что он имеет дело с единой памятью большой емкости и высокого быстродействия. Эту память называют виртуальной (кажущейся) памятью. По своей сути виртуализация памяти представляет собой способ аппаратной и программной реализации концепции иерархической памяти.

В рамках идеи виртуализации основная память рассматривается как линейное пространство N адресов (физическое пространство памяти). Для задач, где требуется более чем *N* ячеек, предоставляется значительно большее пространство адресов (обычно равное общей емкости всех видов памяти), называемое виртуальным пространством, в общем случае не обязательно линейное. Адреса виртуального пространства называют виртуальными, а адреса физического

пространства - физическими. Программа пишется в виртуальных адресах, но поскольку для её выполнения нужно, чтобы обрабатываемые команды и данные находились в ОЗУ, требуется, чтобы каждому виртуальному адресу соответствовал физический. Таким образом, в процессе вычислений необходимо, прежде всего, переписать из ВЗУ в ОЗУ ту часть информации, на которую указывает виртуальный адрес (отобразить виртуальное пространство на физическое), после чего преобразовать виртуальный адрес в физический (рис. 5.12).

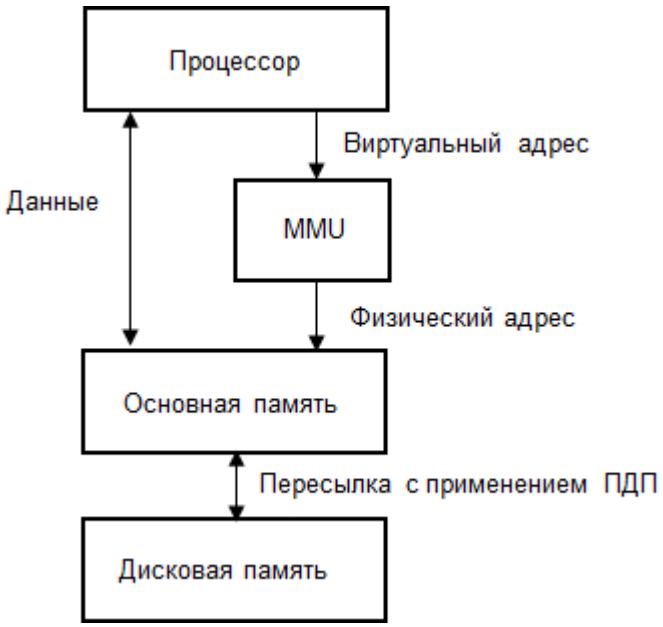


Рис.5.12. Отображение виртуального адреса на физический

Простейший способ преобразования виртуальных адресов в физические основывается на предположении, что все программы и данные состоят из сегментов фиксированной длины, называемых страницами, которые, в свою очередь, состоят из блоков, последовательно расположенных в памяти.

Страничная организация виртуальной памяти.

Страница является базовой единицей информации, перемещаемой между оперативной памятью и диском по требованию механизма преобразования адресов. Размер страницы обычно варьируется от 2 до 16 Кбайт. Найденные данные пересылаются со скоростью несколько мегабайт в секунду.

Все это напоминает концепции кэш-памяти, которая сглаживает разницу в быстродействии процессора и оперативной памяти, а

механизм управления виртуальной памятью делает то же самое в отношении оперативной памяти и дискового устройства.

Блок основной памяти, соответствующий странице, часто называют страничным кадром или фреймом (page frame). Страницам виртуальной и физической памяти присваивают номера. Процесс доступа к данным по их виртуальному адресу иллюстрирует рис.5.13.

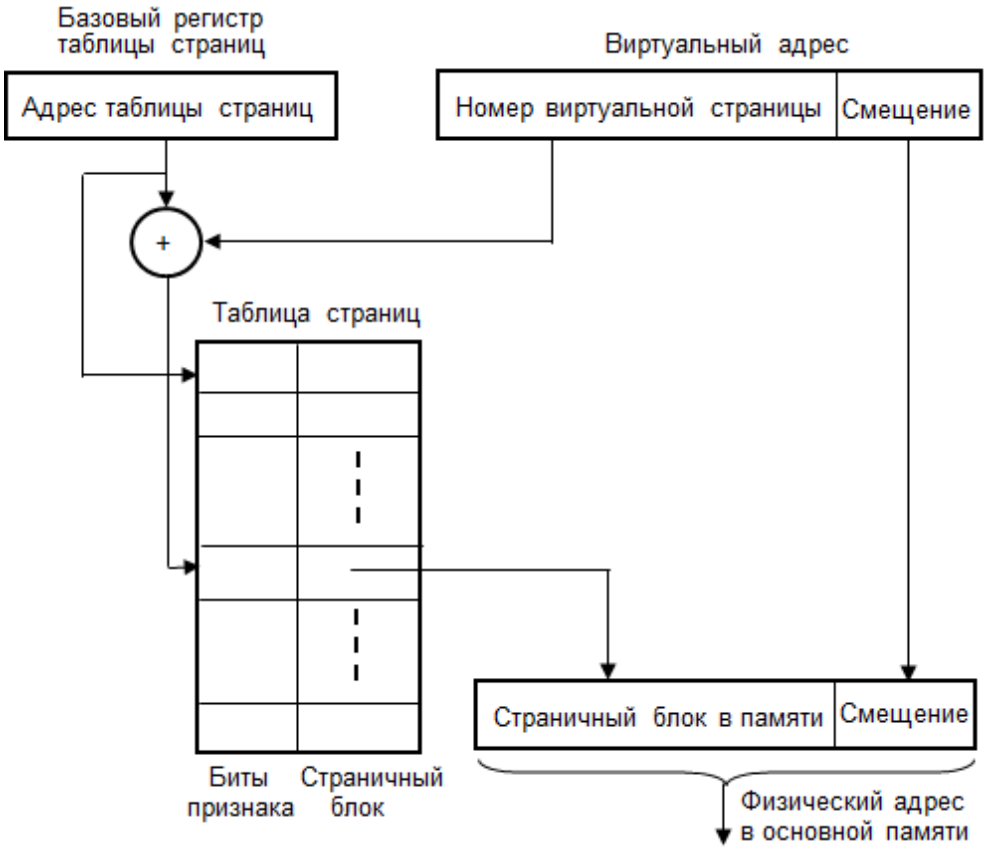


Рис.5.13. Преобразование адресов виртуальной памяти

Каждый сгенерированный процессором виртуальный адрес выборки команды или записи/чтения операнда, интерпретируется как номер виртуальной страницы (старшие разряды) и смещение (младшие разряды) байта или слова от начала страницы. Информация о местонахождении каждой страницы в оперативной памяти содержится в Таблице страниц. Она включает адрес оперативной памяти, по которому хранится страница, и данные о ее текущем состоянии. Область оперативной памяти, где может находиться одна страница, называется страничным блоком. Начальный адрес таблицы страниц хранится в базовом регистре таблицы страниц. Добавив номер виртуальной страницы к содержимому этого регистра можно

получить адрес нужного элемента таблицы страниц. В этом элементе хранится начальный адрес страницы в оперативной памяти.

Кроме адреса страницы каждый элемент таблицы страниц содержит несколько битов признака, которые определяют состояние страницы, находящейся в оперативной памяти. Один бит указывает, хранится ли страница в памяти. Этот бит позволяет операционной системе пометить страницу как отсутствующую в памяти, но не удалять ее. Еще один бит указывает, была ли страница изменена, пока находилась в оперативной памяти. Как и в случае кэш-памяти, на основании этой информации принимается решение о том, записывать ли страницу снова на диск перед ее удалением из оперативной памяти (когда нужно освободить место для другой страницы). Остальные управляющие биты действуют в соответствии с ограничениями, налагаемыми на доступ к странице. Например, программе могут быть предоставлены полные права на чтение и запись или же только на чтение.

Вопросы для контроля

1. В чем различие между основной и вторичной памятью?
2. Перечислите основные характеристики памяти.
3. В чем различие статической и динамической памяти?
4. Каков принцип работы адресной памяти и ее блоков?
5. Расскажите об организации памяти в отдельном блоке.
6. Рассмотрите принцип работы ассоциативной памяти?
7. Опишите особенности работы программируемых ПЗУ?
8. Каким образом организована кэш-память процессора?
9. Как обеспечивается когерентность данных в мультипроцессорных системах?
10. В чем суть и назначение виртуальной памяти компьютера?
11. Опишите процесс преобразования адресов виртуальной памяти?

ГЛАВА 6. АРХИТЕКТУРА ВНУТРЕННИХ И СИСТЕМНЫХ ИНТЕРФЕЙСОВ

6.1. Подключение систем ввода/вывода к процессору

Системный интерфейс, как одна из его подсистем, представляет собой средства сопряжения в единое целое основных компонентов компьютера: процессора, модулей основной и внешней памяти, периферийных устройств. Этот интерфейс в основном реализуется на базе системных и локальных шин, служащих для передачи информации между процессором и другими компонентами компьютера. Эти шины характеризуются разрядностью, то есть, числом бит данных, передаваемых за один цикл шины, и частотным параметром, показывающим, с какой частотой они передаются.

Кроме процессора и памяти, третьим ключевым элементом архитектуры компьютера является система ввода/вывода (СВВ). Ранее было сказано, что имеются внутрипроцессорные средства обмена данными – шина адреса, шина данных и шина управления. Эти коммуникационные компоненты связывали между собой самые скоростные элементы вычислительного ядра – процессор и средства внутренней оперативной памяти. Для связи со значительно более медленными, но более разнообразными по принципу действия внешними устройствами требуется самостоятельная и более мощная система ввода/вывода. Система ввода/вывода призвана обеспечить обмен информацией между ядром компьютера и разнообразными внешними устройствами (ВУ). Технические и программные средства СВВ обеспечивают физическое и логическое сопряжение процессорного ядра и ВУ. Они предназначены для реализации скоростного интерфейса и уменьшения существенного разрыва в производительности процессора и памяти, с одной стороны, и скоростью ввода/вывода - с другой.

Технически система ввода/вывода реализуется комплексом модулей ввода/вывода (МВВ). Модуль ввода/вывода выполняет сопряжение ВУ с ядром компьютера и различные коммуникационные операции между ними. Две основные функции МВВ:

- обеспечение интерфейса с процессором и памятью ("большой"интерфейс),

- обеспечение интерфейса с одним или несколькими периферийными устройствами ("малый" интерфейс).

Анализируя архитектуру известных компьютеров, можно выделить три основных способа подключения СВВ к ядру процессора (рис.6.1, а,в,с).

В варианте с раздельными шинами памяти и ввода/вывода (рис.6.1, а*)* обмен информацией между процессором и памятью физически отделен от ввода/вывода, поскольку обеспечивается полностью независимыми шинами.

Рис.6.1.а. Вариант с раздельными шинами памяти и ввода/вывода

Это дает возможность выполнять обращение к памяти одновременно с выполнением ввода/вывода. Данный архитектурный вариант компьютера позволяет специализировать каждую из шин, учесть формат пересылаемых данных, особенности синхронизации обмена. Шина ввода/вывода, с учетом характеристик реальных ВУ, может иметь меньшую пропускную способность, что позволяет снизить затраты на ее реализацию. Недостатком решения можно считать большое количество точек подключения к процессору. Второй вариант — с совместно используемыми линиями данных и адреса (рис.6.1, в).

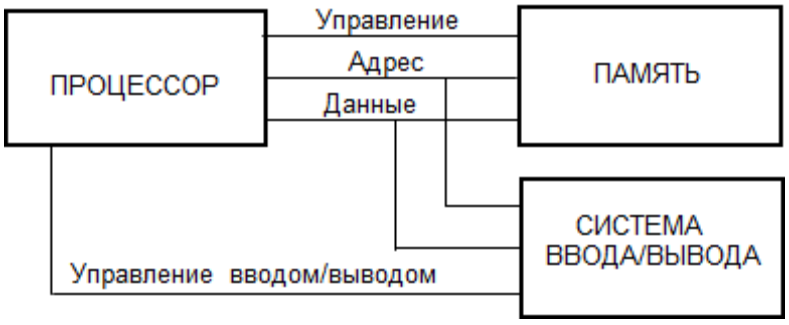


Рис.6.1,в. Совместно используемые линии данных и адреса

Память и СВВ имеют общие для них линии адреса и линии данных, разделяя их во времени. В то же время управление памятью и СВВ, а также синхронизация их взаимодействия с процессором осуществляются независимо по раздельным линиям управления. Это позволяет учесть особенности процедур обращения к памяти и к модулям ввода/вывода и добиться наибольшей эффективности доступа к ячейкам памяти и внешним устройствам.

Последний тип архитектуры предполагает подключение СВВ к системной шине на общих правах с процессором и памятью (рис.6.1,с).

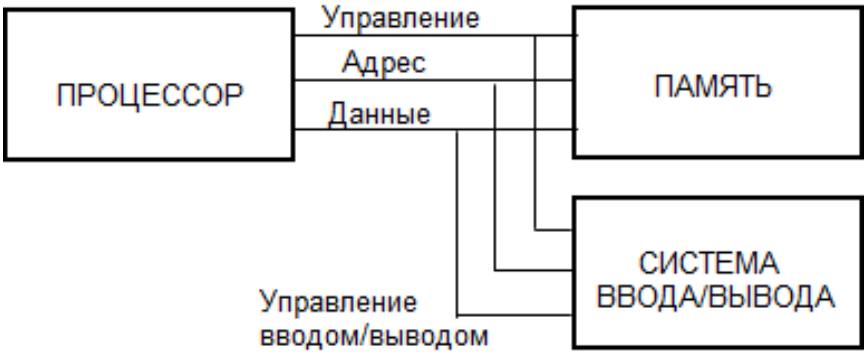


Рис.6.1,с. Подключение на общих правах с процессором и памятью

Преимущества и недостатки такого подхода заключаются в усложнении оборудования за счет дублирования функций процессора. Потенциально возможен вариант подключения внешних устройств к системной шине напрямую, без использования МВВ. Во-первых, в этом случае процессор пришлось бы оснащать универсальными схемами для управления любым ВУ.

При большом разнообразии внешних устройств, имеющих к тому же различные принципы действия, такие схемы оказываются чересчур сложными и избыточными.

Во-вторых, пересылка данных при вводе и выводе происходит значительно медленнее, чем при обмене между процессором и памятью, и было бы невыгодно задействовать для обмена информацией с ВУ высокоскоростную системную шину. И, наконец, в ВУ часто используются иные форматы данных и длина слова, чем в компьютерах, к которым они подключены.

6.2. Аппаратура модуля ввода/вывода

Основным назначением модуля ввода/вывода является обеспечение взаимодействия ядра компьютера, т.е. процессора и оперативной памяти с разнообразными по принципу действия, формату данных и быстродействию внешними устройствами ввода, передачи, отображения, хранения и регистрации информации.

Процедуры ввода/вывода реализуются с помощью модуля состоящего из узла внутреннего(«большого») интерфейса, обеспечивающего связь внешних устройств (ВУ) с процессором и памятью, узлов внешнего («малого») интерфейса (портов), подключаемых непосредственно к периферийным устройствам (рис.6.2).

Рис.6.2. Структура и состав модуля ввода/вывода

В регистре данных буферизуются данные, передаваемые в модуль и из него, что компенсирует различие в быстродействии внешних устройств. Разрядность регистра данных совпадает с шириной шины со стороны «большого» интерфейса (со стороны внутренних узлов компьютера). Это 2,4,8 байт. Интерфейс $ВУ_1$, …, ВУn как правило побайтовый, поэтому в нем имеется узел упаковки/распаковки. При большом количестве ВУ может быть несколько регистров данных.

Регистр управления фиксирует команды взаимодействия модуля с ВУ (очистка регистров, сброс ВУ, начало чтения, начало записи). Регистр состояния служит для хранения битов состояния модуля и ВУ (готовность, занятость, режим приема/переработки).

Каждому модулю в адресном пространстве (совмещенным или разделенным с ОП) выделяется группа адресов. Адреса с помощью селектора проверяются на правильность, и с помощью дешифратора ДС выбирается конкретное ВУ для работы. Узел управления выполняет функции по координации работы всех узлов. Связь модуля с процессором реализуется по линии управления (синхронизация прерывания).

Со стороны внешнего («малого») интерфейса структура модуля ввода/вывода достаточно унифицирована, т.к. все ВУ работают по определенным протоколам и имеют интерфейсные узлы. В качестве интерфейсных узлов применяются **контроллеры,** которые в последних моделях компьютеров размещаются на материнской плате.

Модуль ввода/вывода играет важную роль при организации сетевого обмена, т.е. через контролеры соответствующих портов идет интенсивный обмен информацией, при этом другие компьютеры рассматриваются как внешние устройства по отношению к данному компьютеру сети. Поэтому, следует более детально рассмотреть функции модулей ввода/вывода.

1. Адресная пересылка данных. Команда обращения к внешним устройствам содержит адресную часть. Каждому модулю ввода/вывода назначается определенный диапазон адресов внешних устройств ($ВУ_1$, …, $ВУ_n$). Старшие разряды кода адреса диапазона выбирают сам модуль, младшие разряды выбирают регистр, через который идет обмен с ВУ (селектор адреса).

2. Управление и синхронизация обмена. Процессор может взаимодействовать сразу с несколькими ВУ, имеющими различное быстродействие, в асинхронном режиме, т.е. данные передаются только после подтверждения готовности.

3. Обмен информацией. Модули ввода/вывода обмениваются информацией с одной стороны с процессором, с другой стороны с ВУ. Процедура обмена включает:

- дешифрацию команды, поступившей по линии управления (запись, чтение);

- пересылку данных между процессором и ВУ.

4. Обнаружение ошибок. Ошибки могут возникнуть как по техническим причинам (износ деталей, помехи в электрических сетях), так и по ошибкам в системном программном обеспечении. Наиболее часто встречающиеся источники ошибок:

- непредвиденная последовательность команд;
- малые размеры буферов ввода/вывода;
- некорректные действия пользователей.

Поскольку получателем данных является основная память, блок обнаружения ошибок осуществляется чаще всего в корректирующих цепях основной памяти.

6.3. Методы управления вводом/выводом

В современных компьютерах находят применение три способа организации ввода/вывода:

- программно-управляемый ввод/вывод;
- ввод/вывод по прерываниям;
- прямой доступ к памяти.

При **программно-управляемом** вводе/выводе все связанные с этим действия происходят по инициативе процессора и под его полным контролем. Процессор выполняет программу, которая обеспечивает прямое управление процессом ввода/вывода, включая проверку состояния устройства, выдачу команд ввода или вывода. Ввод/вывод реализуется специальной процедурой ввода/вывода. В этой процедуре процессор с помощью команды ввода/вывода сообщает модулю ввода/вывода, а через него и внешнему устройству о предстоящей операции. Адрес модуля и ВУ, к которому производится обращение, указывается в адресной части команды ввода или вывода. Модуль исполняет затребованное действие, после чего устанавливает в единицу соответствующий бит в своем регистре состояния.

Для определения момента завершения операции или пересылки очередного элемента блока данных процессор должен периодически опрашивать и анализировать содержимое регистра состояния МВВ.

Несмотря на это, процессор, выдав в МВВ команду, должен ожидать завершения ее выполнения. Из-за разницы в скорости работы процессора и ввода/вывода происходит потеря времени.

Ввод/вывод по прерываниям во многом совпадает с программно-управляемым методом. Отличие состоит в том, что после выдачи команды ввода/вывода процессор не должен циклически опрашивать МВВ для выяснения состояния устройства. Вместо этого процессор может продолжать выполнение других команд до тех пор, пока не получит запрос прерывания от МВВ, извещающий о завершении выполнения ранее выданной команды ввода/вывода. Как и при программно-управляемом методе процессор отвечает за извлечение данных из памяти (при выводе) и запись данных в память (при вводе).

Процедура ввода блока данных по прерываниям реализуется следующим образом. Процессор выдает команду чтения, а затем продолжает выполнение другой программы. Получив команду, МВВ приступает к вводу элемента данных с ВУ. Когда считанное слово оказывается в регистре данных модуля,

МВВ формирует на управляющей линии сигнал прерывания работы процессора. Выставив запрос, МВВ помещает введенную информацию на шину данных, после чего он готов к следующей операции ввода/вывода. Процессор в конце каждого цикла команды проверяет наличие запросов прерывания. Когда от МВВ приходит такой сигнал, процессор сохраняет контекст текущей программы и обрабатывает прерывание. В рассматриваемом случае процессор читает слово из модуля, записывает его в память и выдает модулю команду на считывание очередного слова. Далее процессор восстанавливает контекст прерванной программы и возобновляет ее выполнение.

Наиболее эффективен метод прямого доступа к памяти (direct memory access - DMA).

В режиме **прямого доступа к памяти (ПДП**) на системной шине функционирует дополнительный модуль – контроллер ПДП (КПДП), который берет на себя функции процессора по управлению системной шиной и обеспечивает прямую пересылку информации

между основной памятью и ВУ без участия других устройств (рис.6.3).

Инициализация контроллера КПДП со стороны процессора включает занесение в контроллер следующих 4-х параметров:

- вида запроса (чтение или запись),
- адрес ВУ,
- адреса начальных ячеек оперативной памяти, куда будет записываться (или куда будет считываться) информация,
- количество читаемых/записываемых слов (размер блока).

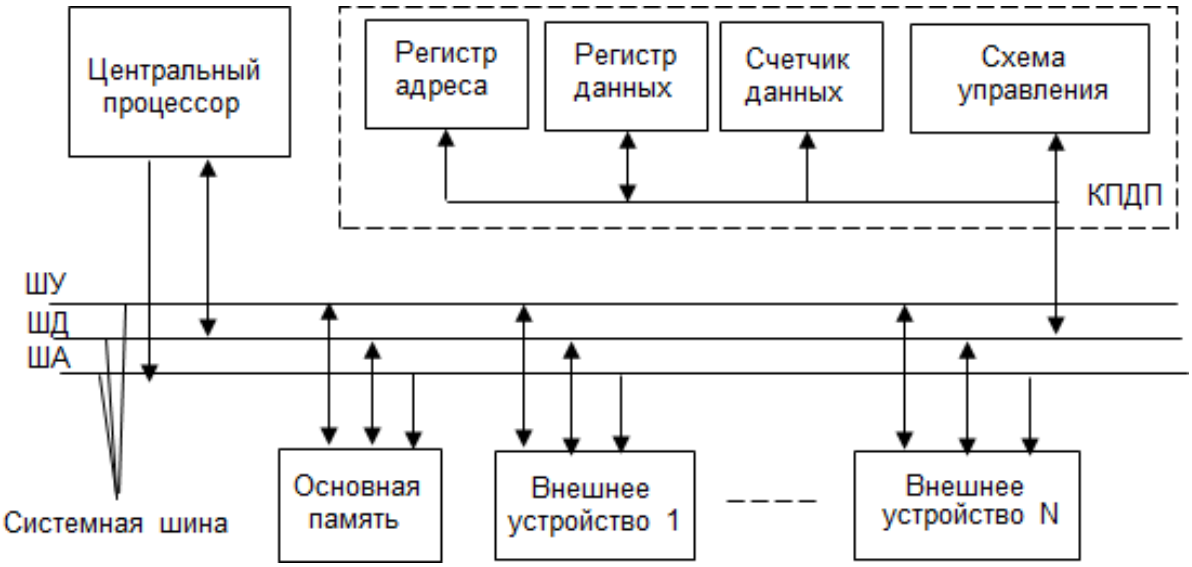


Рис.6.3. Организация прямого доступа к памяти

Вид запроса попадает в схему управления контроллера, туда же посылается и адрес ВУ. Адрес начальной ячейки оперативной памяти хранится в регистре, содержимое которого по мере чтения/записи увеличивается на единицу.

Размер блока заносится в счетчик данных, содержимое которого после передачи каждого слова уменьшается на единицу. Нуль в счетчике означает конец операции чтение/записи.

При подаче сигнала со стороны ВУ контроллер КПДП посылает процессору «запрос ПДП», процессор освобождает ША, ШД линии ввода/вывода, затем от процессора идет сигнал подтверждение КПДП на начало пересылки данных.

При чтении и записи происходит буферизация пересылаемого слова в Регистр данных, т.к. скорости работы оперативной памяти и ВУ различаются. Регистр адреса и счетчик команд синхронно

увеличиваются/уменьшаются соответственно на единицу после каждого такта обмена.

Вмешательство в работу процессора возможно на всех этапах использования текущей команды, кроме момента «чтение команд».

Способ организации взаимодействия ВУ зависит от состояния быстродействия оперативной памяти и ВУ. ВУ бывают быстродействующие (например, диски) со скоростью обмена около 1М байт/с и медленнодействующие (печатающие устройства). Поэтому есть мультиплексный режим обмена (разделения времени канала ввода/вывода между ВУ) и монопольный режим обмена (работает одно ВУ на всем цикле обмена).

В обобщенном виде при передаче в режиме ПДП выполняются следующие действия.

1. Процессор вызывает режим ПДП, передавая параметры устройства, операцию для выполнения на устройстве, адрес памяти, являющийся источником или пунктом назначения для переносимых данных и количество переносимых байтов.

2. Контроллер ПДП запускает на устройстве операцию и передает сигналы для внутреннего соединения. Когда данные доступны (из устройства или памяти), он переносит их. Устройство доставляет адрес памяти для записи или чтения. Если запрос требует более чем одного сеанса передачи (трансфера), контроллер ПДП генерирует следующий адрес и начинает следующий трансфер. Используя этот механизм, контроллер может полностью закончить трансфер, который может составлять тысячи байтов в длину, не тревожа процессор. Многие ПДП-контролеры содержат немного памяти, которая дает им возможность свободно заниматься как задержками при трансфере, так и с теми, которые являются следствием ожидания, чтобы стать главными.

3. Как только ПДП-трансфер окончен, контролер прерывает процессор, который может тогда определить, выполнена ли вся операция успешно, посылая сигналы устройству ПДП или исследуя память.

В компьютерной системе могут быть разнообразные ПДП устройства. Например, в системе с одной шиной процессор-память и различными шинами ввода/вывода каждый контролер шины

ввода/вывода будет скорее всего содержать процессор ПДП, который осуществляет любые переносы между устройством на шине ввода/вывода и памятью.

В отличие от ввода/вывода, управляемого прерываниями, канал ПДП может быть использован для сообщения с жестким диском без потребления всех циклов процессора для однократного ввода/вывода. Конечно, если процессор будет бороться за память, он будет отложен, когда память занята ПДП-трансфером. Благодаря использованию кэш-памяти, процессор может избежать получение доступа к памяти большую часть времени, таким образом оставляя много каналов обращения к памятью свободными для использования устройствами ввода/вывода.

Во многих вычислительных машинах и системах для развязывания входов компьютера с внешними устройствами применяются контроллеры.

Рассмотренные характеристики систем ввода/вывода ведут к появлению различных функций, которые должна обеспечить операционная система (ОС):

- ОС гарантирует, что программа пользователя имеет доступ только к тем частям устройства ввода/вывода, к которым у пользователя есть права. Например, ОС не должна позволять программе считывать или записывать файлы на диск, если владелец файла не давал согласия для доступа к этой программе. В системе с устройствами ввода/вывода совместного пользования защита не могла бы быть обеспечена, если программа пользователя смогла осуществлять ввод/вывод напрямую:

- ОС обеспечивает обобщенный доступ к устройству посредством стандартных программ, управляющих низкоуровневыми операциями устройства;

- ОС управляет прерываниями, генерируемыми устройствами ввода/вывода, как и исключениями, генерируемыми программой;

- ОС пытается обеспечить объективный доступ к общим ресурсам ввода/вывода, так же, как и планирует доступы для увеличения пропускной способности системы.

Чтобы выполнять эти функции от лица пользовательских программ, ОС должна быть в состоянии связываться с устройствами

ввода/вывода и предотвращать прямую связь между программой и устройствами ввода/вывода. Требуются три типа связи:

- ОС должна быть в состоянии давать команды устройствам ввода/вывода. Эти команды являют собой не только такие операции, как считывание и запись, но также и другие операции, такие как поиск диска;
- устройство должно уведомлять ОС, когда устройство ввода/вывода закончило операцию или натолкнулось на ошибку. Например, когда диск закончит поиск, он известит ОС;
- данные должны быть перемещены между памятью и устройством ввода/вывода.

6.4. Организация шин компьютера

Ранее рассмотренные правила взаимодействия между основными узлами и блоками компьютера могут эффективно реализоваться только при наличии электрических, логических и технологических связей между ними. Часть этих связей, обеспечивающих взаимодействие процессорных элементов АЛУ, РОН, узла управления, модулей ввода/вывода и кэш L1, представляются в виде внутрипроцессорных шин: ША, ШД, ШУ. Это электрические проводники, передающие команды, операнды и сигналы управления между узлами процессора. Технологически они встроены в кристалл (чип) процессора, но через модули ввода /вывода имеют выход на другие конструктивные компоненты компьютера (материнскую плату). Указанные три шины отличаются небольшим числом линий связи и высокой тактовой частотой, т.к. связывают два самых быстродействующих узла компьютера: процессор и внутреннюю память.

Однако, в компьютере есть другие многочисленные информационные потоки: адреса периферийных устройств, данные внутренние и внешние, сигналы управления и прерывания. Они на электрическом и информационном уровне связывают центральный процессор и основную память со всеми другими функциональными устройствами компьютера: периферийными устройствами, дисковыми наполнителями средствами сетевого взаимодействия. В отличие от внутрипроцессорных шин, общесистемные шины имеют

большую протяженность, более широкий диапазон передаваемых сигналов большее число каналов (линий) передачи.

Шина - это магистраль, представляющая собой совокупность физических коммуникационных линий с электрическими характеристиками и протоколами передачи данных по ним. Два главных преимущества устройства шины – универсальность и низкая цена. Определив одиночную схему соединения, можно легко добавлять новые устройства, а периферийное оборудование даже можно без проблем конфигурировать внутри компьютерных систем, использующих один и тот же тип шины.

Операции на шине называются транзакциями. Основные типы транзакций – операции чтения или записи. На рис.6.4 представлена структура современного компьютера с шинами. Представленные типы шин в конкретных моделях компьютеров используются в той или иной конфигурации.

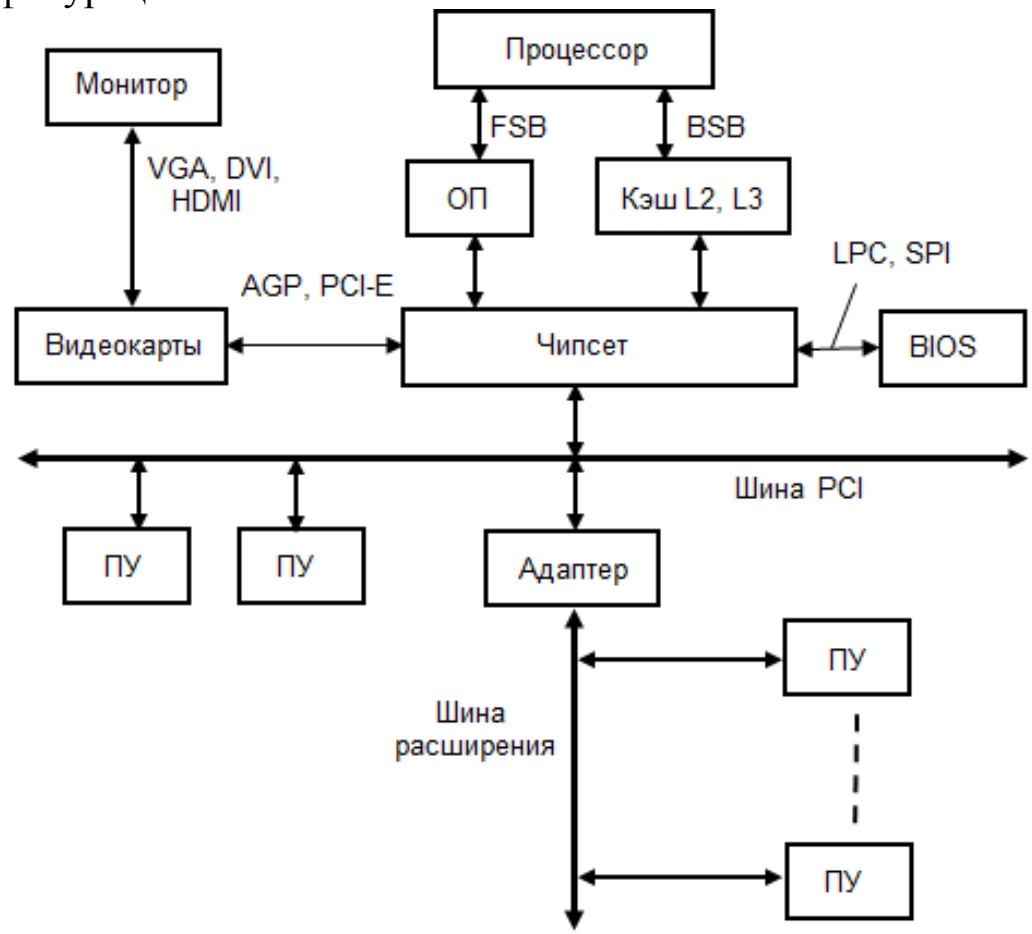


Рис.6.4. Организация шин в компьютере

Введены обозначения:

- FSB (front-side bus), BSB (back-side bus) – шины «процессор-память»;
- AGP – параллельная шина видеосистемы;
- PCI-E – последовательная шина видеосистемы;
- VGA – аналоговый интерфейс монитора;
- DVI, HDMI – цифровые интерфейсы монитора или TV;
- PCI – параллельная шина периферийных устройств ПУ;
- BIOS – базовая система ввода-вывода;
- LPC, SPI – последовательные интерфейсы связи BIOS;
- ОП – оперативная память.

К основным параметрам, характеризующим тот или иной тип шин, относятся:

- ширина шины, т.е. количество адресных линий, но которым передаются данные;
- тактовая частота, определяющая скорость передачи отдельных бит по палам связи;
- протокол обмена, определяющий правила обмена между устройствами (при синхронном обмене все сигналы привязаны к импульсам синхрогенератора, при асинхронном протоколе начало очередного сеанса обмена начинается после окончания предыдущего).

В таблице 6.1 представлены характеристики основных параллельных шин, используемых в компьютерах.

Табл. 6.1.

Шина	Разрядность (бит)	Тактовая частота (МГц)	Пропускная способность (Мбайт/сек)
ISA-16	16	8.3	16.6
EISA	32	8.3	33.3
PCI 1.0	32	33	132
PCI 2.0	64	33	264
PCI 2.1	64	66	528
AGP(1x)	32	66	264
AGP (4x)	64	66	1066
SCSI	8/16	20/40/80/160	20/80/320/640
FSB	64	66-1066	528-8500

Одной из причин, почему спроектировать шину довольно сложно, является тот факт, что максимальная скорость шины существенно ограничена физическими факторами: длиной шины и количеством устройств.

Эти физические ограничения препятствуют быстрой работе шины. Необходимость поддерживать несколько устройств с разными периодами ожидания и скоростями передачи данных также делает проектирование шин затруднительным.

С точки зрения выполняемых функций и целевого назначения все разнообразие шин компьютера можно условно разделить на две категории - локальные и системные. Локальные шины связывают между собой ограниченное число компонентов компьютера: центральный процессор с основной памятью или контролерами и адаптерами внешних устройств. Системная шина предназначена для обеспечения сквозной передачи различными по скорости устройствами ввода/вывода. В последних моделях компьютеров функции системой шины рассредоточились, появилось больше локальных шин.

Любая стандартная шина содержит линии для передачи данных, отдельные линии для передачи адресов (адрес определяет источник или приемник данных), линии аппаратных прерываний, отдельные линии – каналы прямого доступа к памяти, проводники для передачи служебной информации, линии разводки электропитания. Для них определены способы кодирования сигналов (битов), скорости передачи информации, механизмы арбитража (управление пользованием шиной и разрешения возникших ситуаций).

6.5. Системные шины компьютера

В компьютере шины бывают следующих типов:

- скоростные шины «процессор-память», которые связывают процессор с основной памятью и кэш-памятью второго или третьего уровня, расположенных вне процессора;
- системная шина PCI, которая объединяет процессор, память и другие внешние устройства (винчестеры, сетевые карты);
- шины ввода-вывода, подключаемые к системной шине для связи с периферийными устройствами (принтеры, сканеры);

- графическая шина AGP или PCI-E для связи видеосистемы с основной памятью и процессором.

С целью снижения стоимости компьютеры имеют общую шину для памяти и устройств ввода/вывода - системную шину. Системная шина служит для физического и логического объединения всех устройств компьютера. Поскольку основные устройства машины, как правило, размещаются на общей монтажной плате, системную шину часто называют объединительной шиной (backplane bus).

Совокупность линий шины можно подразделить на три функциональные группы (рис. 6.4): линии данных, линии адреса и сигналы управления. К последним, обычно, относят также линии для подачи питающего напряжения на подключаемые к системной шине модули.

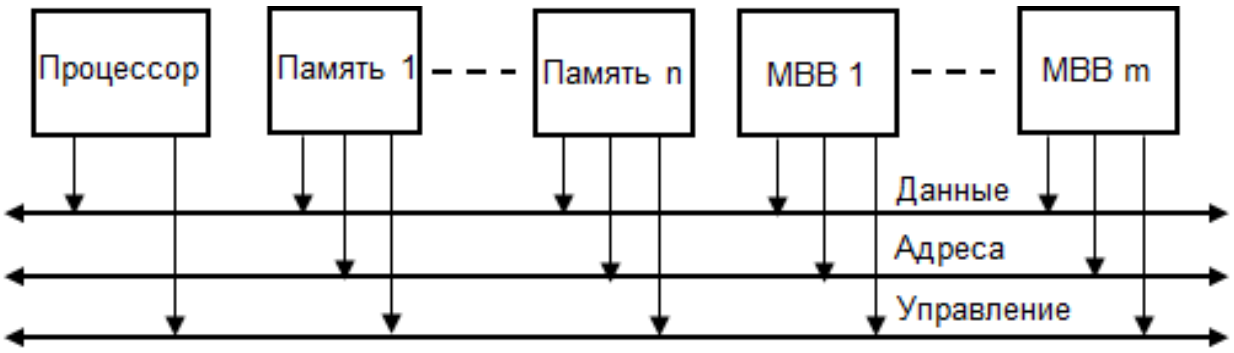


Рис.6.4. Организация системной шины

Режим функционирования системной шины. Для передачи данных в другой произвольный модуль необходимо получить в свое распоряжение шину и только после этого передавать по ней данные. Если какой-то модуль хочет получить данные от другого модуля, он должен получить доступ к шине и с помощью соответствующих линий управления и адреса передать в другой модуль запрос. Далее, он должен ожидать, пока модуль, получивший запрос, пошлет данные.

Физически системная шина представляет собой совокупность параллельных электрических проводников. Этими проводниками служат металлические полоски на печатной плате. Шина подводится ко всем модулям, и каждый из них подсоединяется ко всем или некоторым ее линиям.

Если компьютер конструктивно выполнен на нескольких платах, то все линии шины выводятся на разъемы, которые затем объединяются проводниками на общем шасси.

Внутренняя связь ввода/вывода служит как способ расширения машины и подключения новых периферийных устройств. Чтобы облегчить эту задачу, было разработано несколько стандартов. Стандарты нужны в качестве спецификации для разработчиков компьютеров и периферийных устройств. Стандарт гарантирует компьютерным разработчикам, что периферийные устройства будут доступными для новой машины, а создателей периферии заверяет в том, что пользователи смогут подключить их новое оборудование.

Среди стандартизированных системных шин универсальных компьютеров наиболее широко применимы шины Unibus, Fastbus, Futtirebus, VME, NuBus, Multibus-II. Персональные компьютеры, как правило, строятся на основе системной шины в стандартах EISA или PCI.

В зависимости от количества и номенклатуры внешних устройств количество шин в компьютерах может быть различным как по количеству, так и по назначению. На рис.6.5 представлена структура связей процессора с одной единственной системной шиной, через которую идет обмен данными как с памятью, так и с ВУ. Для такой организации обмена характерны простота и низкая стоимость, но число абонентов шины резко ограничено.

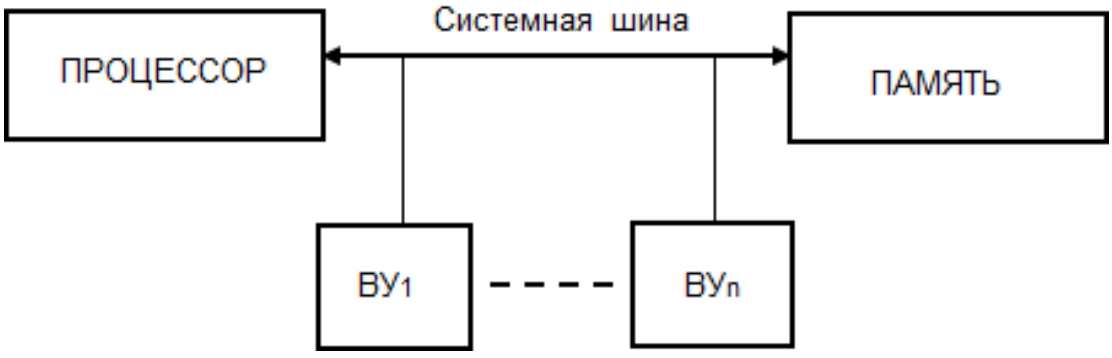


Рис.6.5. Организация связей с одной системной шиной

На рис.6.6. представлена структура связей процессора с двумя видами шин.

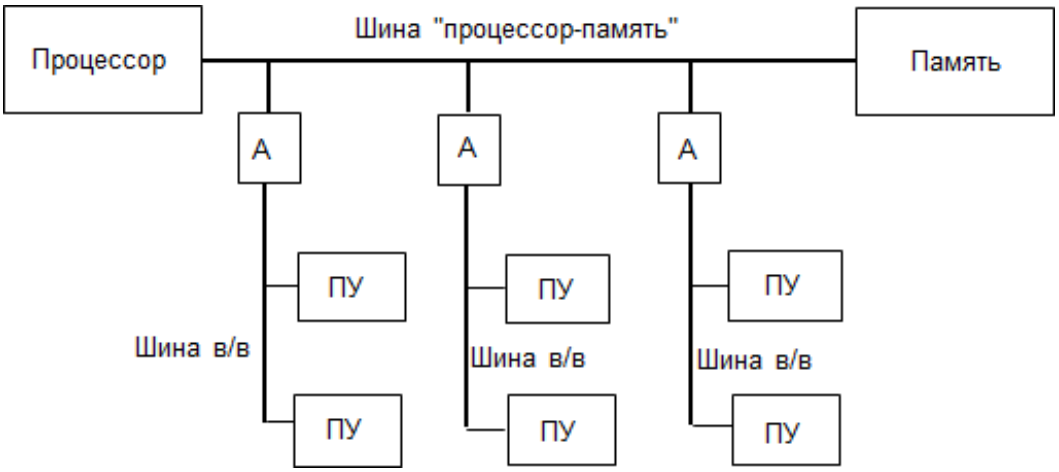


Рис.6.6. Организация связей с двумя шинами

ВУ подключаются к шинам ввода/вывода, которые берут на себя основной трафик, не связанный с выходом на процессор или память. Адаптеры шин (А) обеспечивают буферизацию данных при их пересылке между системной шиной и контроллерами ВУ. Это позволяет компьютеру поддерживать работу множества устройств ввода/вывода и одновременно «развязать» обмен информацией по тракту «процессор-память» и обмен информацией с ВУ.

Для подключения быстродействующих периферийных устройств в систему шин может быть добавлена высокоскоростная шина расширения (рис. 6.7). Здесь ВУ через отдельную шину ввода/вывода подключаются через адаптеры к общей шине расширения, Это позволяет еще больше разгрузить шину «процессор-память».

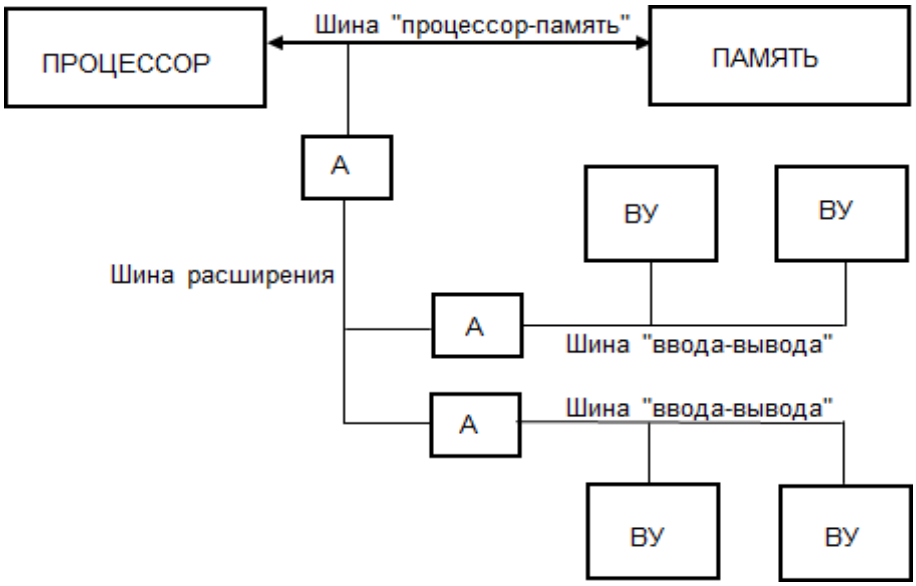


Рис.6.7. Организация связей с тремя шинами

Распределение линий системной шины. Любая транзакция на шине начинается с выставления ведущим устройством адресной информации. Адрес позволяет выбрать ведомое устройство и установить соединение между ним и ведущим. Для передачи адреса используется часть сигнальных линий шины, совокупность которых часто называют **шиной адреса (ША).**

На ША могут выдаваться адреса ячеек памяти, номера регистров процессора, адреса портов ввода/вывода. Многообразие видов адресов предполагает наличие дополнительной информации, уточняющей вид, используемый в данной транзакции. Такая информация может косвенно содержаться в самом адресе, но чаще передается по специальным управляющим линиям шины.

Разнообразной может быть и структура адреса. Так, в адресе может конкретизироваться лишь определенная часть ведомого, например, старшие биты адреса могут указывать на один из модулей основной памяти, в то время как младшие биты определяют ячейку внутри этого модуля.

В некоторых шинах предусмотрены адреса специального вида, обеспечивающие одновременный выбор определенной группы ведомых либо всех ведомых сразу (broadcast). Такая возможность обычно практикуется в транзакциях записи (от ведущего к ведомому), однако существует также специальный вид транзакции чтения (одновременно от нескольких ведомых общему ведущему).

Число сигнальных линий, выделенных для передачи адреса (ширина шины адреса), определяет максимально возможный размер адресного пространства. Это одна из базовых характеристик шины, поскольку от нее зависит потенциальная емкость адресуемой памяти и число обслуживаемых портов ввода/вывода.

Совокупность линий, служащих для пересылки данных между модулями системы, называют **шиной данных (ШД).** Важнейшие характеристики шины данных - ширина и пропускная способность.

Ширина шины данных определяется количеством битов информации, которое может быть передано по шине за одну транзакцию (цикл шины). В современных компьютерах ширина шины данных составляет обычно 32,64 или 128 бит. В любом случае, ширину ШД выбирают кратной целому числу байтов.

Элемент данных, охватывающий всю ширину ШД, принято называть словом, хотя в архитектуре некоторых ВМ понятие «слово» трактуется по-другому, то есть слово может иметь разрядность, не совпадающую с шириной ШД. В большинстве шин используются адреса, позволяющие указать отдельный байт слова. Это свойство оказывается полезным, когда желательно изменить в памяти лишь часть полного слова.

Ширина шины данных существенно влияет на производительность компьютера. Так, если шина данных имеет ширину вдвое меньшую, чем длина команды, процессор в течение каждого цикла команды вынужден осуществлять доступ к памяти дважды.

Пропускная способность шины характеризуется количеством единиц информации (байтов), которые допускается передать по шине за единицу времени (секунду), а определяется физическим построением шины и природой подключаемых к ней устройств. Очевидно, что чем шире шина, тем выше ее пропускная способность.

Помимо трактов пересылки адреса и данных, неотъемлемым атрибутом любой шины являются линии, по которым передается управляющая информации и информация о состоянии участвующих в транзакции устройств. Совокупность таких линий принято называть **шиной управления (ШУ**), хотя такое название представляется не совсем точным. Сигнальные линии, входящие в ШУ, можно условно разделить на несколько групп.

Первую группу образуют линии, по которым пересылаются сигналы **управления транзакциями,** то есть сигналы, определяющие:

- тип выполняемой транзакции (чтение или запись);
- количество байтов, передаваемых по шине данных, и, если пересылается часть слова, то какие байты;
- какой тип адреса выдан на шину адреса;
- какой протокол передачи должен быть применен.

На перечисленные цели обычно отводится от двух до восьми сигнальных линий.

Ко второй группе отнесем линии **передачи информации состояния** (статуса). В эту группу входят от одной до четырех линий,

по которым ведомое устройство может информировать ведущего о своем состоянии или передать код возникшей ошибки.

Третья группа – **линии арбитража.** Вопросы арбитража рассматриваются несколько позже. Арбитраж необходим для выбора одного из нескольких ведущих, одновременно претендующих на доступ к шине. Число линий арбитража в разных шинах варьируется от 3 до 11.

Четвертую группу образуют **линии прерывания**. По этим линиям передаются запросы на обслуживание, посылаемые от ведомых устройств к ведущему. Под собственно запросы обычно отводятся одна или две линии, однако при одновременном возникновении запросов от нескольких ведомых возникает проблема арбитража, для чего могут понадобиться дополнительные линии.

Пятая группа - линии для организации последовательных локальных сетей. Наличие от 1 до 4 таких линий стало общепринятой практикой в современных шинах. Обусловлено это тем, что последовательная передача данных протекает значительно медленнее, чем параллельная, и сети значительно выгоднее строить, не загружая быстрые линии основных шин адреса и данных. Кроме того, шины этой группы могут быть использованы как полноценный, хотя и медленный, избыточный тракт для замены ША и ШД в случае их отказа.

В некоторых ШУ имеется шестая группа сигнальных линий — от 4 до 5 линий позиционного кода, подсоединяемых к специальным выводам разъема. С помощью перемычек на этих выводах можно задать уникальный позиционный код разъема на материнской плате или вставленной в этот разъем дочерней платы. Такой код может быть использован для индивидуальной инициализации каждой отдельной платы при включении или перезапуске системы.

Наконец, в каждой шине обязательно присутствуют линии, которые в нашей классификации входят в седьмую группу**,** которая, по сути, является одной из важнейших. Это группа линий **тактирования и синхронизации.** При проектировании шины таким линиям уделяется особое внимание. В состав группы, в зависимости от протокола шины (синхронный или асинхронный), входят от двух до шести линий.

В довершение необходимо упомянуть линии для подвода питающего напряжения и линии заземления. Большое количество линий в шине[1] предполагает использование разъемов со значительным числом контактов. В некоторых шинах разъемы имеют сотни контактов, где предусмотрены подключение вспомогательных шин специального назначения, свободные линии для локального обмена между дочерними платами, множественные параллельно расположенные контакты для «размножения» питания и «земли».

Связь компьютера с периферийным устройством организуется со стороны процессора и может быть реализована в одном из трех режимов: симплексном, полудуплексном и дуплексном.

В **симплексном режиме** информация передается в одном направлении: один передает, другой принимает.

В **полудуплексном режиме** идет обмен данными в обоих направлениях, но реализуется поочередно, в каждый момент времени передача может вестись в одном направлении. Передающая и принимающая сторона могут поменяться ролями.

Дуплексный режим позволяет вести передачу и прием одновременно в двух направлениях.

Установленные стандартом правила обмена в любом режиме называются протоколом.

6.6. Арбитраж шин

В реальных системах на роль ведущего вправе одновременно претендовать сразу несколько из подключенных к шине устройств, однако управлять шиной в каждый момент времени может только одно из них. Чтобы исключить конфликты, шина должна предусматривать определенные механизмы арбитража запросов и правила предоставления шины одному из запросивших устройств. Решение обычно принимается на основе приоритетов претендентов.

Схемы приоритетов. Каждому потенциальному ведущему присваивается определенный уровень приоритета, который может оставаться неизменным **(статический приоритет)** либо изменяться по какому-либо алгоритму **(динамический приоритет).**

Основной недостаток статических приоритетов в том, что устройства, имеющие высокий приоритет, в состоянии полностью блокировать доступ к шине устройств с низким уровнем приоритета. Системы с динамическими приоритетами дают шанс каждому из запросивших устройств рано или поздно получить право на управление шиной, то есть в таких системах реализуется принцип равнодоступности.

Наибольшее распространение получили следующие алгоритмы динамического изменения приоритетов:

- простая циклическая смена приоритетов;
- циклическая смена приоритетов с учетом последнего запроса;
- смена приоритетов по случайному закону;
- схема равных приоритетов;
- алгоритм наиболее давнего использования.

В алгоритме **простой циклической смены приоритетов** после каждого цикла арбитража все приоритеты понижаются на один уровень, при этом устройство, имевшее ранее низший уровень приоритета, получает наивысший приоритет.

В схеме **циклической смены приоритетов с учетом последнего запроса** все возможные запросы упорядочиваются в виде циклического списка. После обработки очередного запроса обслуженному ведущему назначается низший уровень приоритета. Следующее в списке устройство получает наивысший приоритет, а остальным устройствам приоритеты назначаются в убывающем порядке, согласно их следованию в циклическом списке.

В обеих схемах циклической смены приоритетов каждому ведущему обеспечивается шанс получить шину в свое распоряжение, однако большее распространение получил второй алгоритм.

При **смене приоритетов по случайному закону** после очередного цикла арбитража с помощью генератора псевдослучайных чисел каждому ведущему присваивается случайное значение уровня приоритета.

В **схеме равных приоритетов** при поступлении к арбитру нескольких запросов каждый из них имеет равные шансы на обслуживание. Возможный конфликт разрешается арбитром. Такая схема принята в асинхронных системах.

В **алгоритме наиболее давнего использования** (LRU, Least Recently Used) после каждого цикла арбитража наивысший приоритет присваивается ведущему, который дольше чем другие не использовал шину.

Помимо рассмотренных существует несколько алгоритмов смены приоритетов, которые не являются чисто динамическими, поскольку смена приоритетов происходит не после каждого цикла арбитража. К таким алгоритмам относятся:

- алгоритм очереди (первым пришел — первым обслужен);
- алгоритм фиксированного кванта времени.

В **алгоритме очереди** запросы обслуживаются в порядке очереди, образовавшейся к моменту начала цикла арбитража. Сначала обслуживается первый запрос в очереди, то есть запрос, поступивший раньше остальных. Аппаратурная реализация алгоритма связана с определенными сложностями, поэтому используется он редко.

В **алгоритме фиксированного кванта времени** каждому ведущему для захвата шины в течение цикла арбитража выделяется определенный квант времени. Если ведущий в этот момент не нуждается в шине, выделенный ему квант остается не использованным. Такой метод наиболее подходит для шин с синхронным протоколом.

6.7. Типы шин современных компьютеров и систем

Среди стандартизированных системных шин универсальных вычислительных машин наиболее известны шины общего применения VME, Multibus-II, Summit. Персональные компьютеры, как было указано, строятся на основе системной шины в стандартах ISA, EISA или PCI.

1. Магистрально-модульная архитектура VME используется для построения вычислительных систем различного диапазона производительности, от настольных компьютеров до многопроцессорных суперкомпьютеров, от промышленных контроллеров до систем управления телекоммуникациями.

В соответствии с требованиями стандарта шина имеет разрядность адресов и данных до 32 бит, обеспечивает возможность адресации до 4 Гбайт при скорости обмена до 40 Мбайт/сек.

Появившись в 1981г. Шина VME стала наиболее популярным открытым стандартом для встраиваемых микропроцессорных систем реального времени.

2. Шина Multibus-II применяется в компьютерах фирмы Intel в качестве системной шины с мультиплексированием данных. Данные передаются с разрядностью 32бит, протокол обмена синхронный тактовая частота 10 МГц.

Максимальное количество обслуживаемых внешних устройств 21, максимальная длина шины 0,5 м.

3. Шина Summit разработана для высокопроизводительных серверов компании HP, имеет разрядность адреса 48 бит, разрядность данных до 512 бит, протокол обмена синхронный, тактовая частота 60 МГц.

4. Шина ISA (Industry Standart Architecture – промышленная стандартная архитектура) относится к мультиплексированным 16–разрядным системным магистралям. Обмен по ней осуществляется 8/16 разрядными данными. На шине реализован раздельный доступ к памяти компьютера и к устройствам ввода-вывода. Максимальный объем адресуемой памяти составляет 16 Мбайт (24 адресные линии). Максимальное адресное пространство для ввода/ вывода -64 Кбайт (16 адресных линий).

5. Шина EISA (Extended ISA) – это 32–разрядная шина данных и 32 – разрядная шина адреса, создана в 1989 году. Адресное пространство шины 4 Гбайт, работает на частоте 8 МГц.

Теоретическая пропускная способность шины – 33 Мбайт/с, причем скорость обмена по каналу «процессор – кэш - оперативная память» определяется параметрами микросхем памяти; увеличено число разъемов расширений – теоретически может подключаться до 15 устройств, практически до 10. Улучшена система прерываний, поддерживается режим единоличного управления шиной со стороны любого из устройств на шине Bus Master, имеется система арбитража для управления доступом устройств к шине.

EISA – дорогая, но оправдывающая себя архитектура, применяющаяся многозадачных системах, на файл-серверах и везде, где требуется высокоэффективное расширение шины ввода-вывода.

6. Шина MCA (MicroChannel Architecture - микроканальная архитектура) была введена фирмой IBM для своих компьютеров PS/2 в 1987 году. Обеспечивает быстрый обмен данными между отдельными устройствами, в частности с оперативной памятью. Шина MCA абсолютно несовместима с ISA/EISA и другими адаптерами. Состав управляющих сигналов, протокол и архитектура ориентированы на асинхронное функционирование шины и процессора, что снимает проблемы согласования скоростей процессора и периферийных устройств. Архитектура позволяет эффективно и автоматически конфигурировать все устройства программным путем.

7. Шина PCI (Peripheral Component Interconnect-соединение внешних компонентов) - самый распространенный и универсальный интерфейс для подключения различных устройств. Разработана в 1993 году фирмой Intel. Шина PCI допускает подключение до 10 устройств, имеет свой адаптер, позволяющий ей настраиваться на работу с любым процессором от 80486 до современных Pentium. Тактовая частота PCI – 33 МГц, разрядность – 32 разряда для данных и 32 разряда для адреса с возможностью расширения до 64 бит. Теоретическая пропускная способность шины 132 Мбайт/с, а в 64-битовом варианте – 264 Мбайт/с. Модификация 2.1 локальной шины PCI работает на тактовой частоте до 66 МГц и при разрядности 64 имеет пропускную способность до 528 Мбайт/с. Осуществлена поддержка режимов Plug and Play, Bus Mastering и автоконфигурирования адаптеров.

На одной шине PCI может быть не более четырех устройств (слотов). Мост шины PCI (PCI Bridge) – это аппаратные средства подключения шины PCI к другим шинам. Host Bridge – главный мост – используется для подключения PC1 к системной шине (шине процессора или процессоров). Peer-to-Peer Bridge - одноранговый мост – используется для соединения двух шин PCI. Две и более шины PCI применяются в мощных северных платформах, т.к. дополнительные шины PCI позволяют увеличить количество подключаемых устройств.

Шина PCI все обмены трактует как пакетные: каждый кадр начинается фазой адреса, за которой может следовать одна или

несколько фаз данных. Количество фаз данных в пакете неопределенно, но ограничено таймером, определяющим максимальное время, в течении которого устройство может пользоваться шиной. Каждое устройство имеет собственный таймер, значения для которого задается при конфигурировании устройств шины.

В каждом обмене участвуют два устройства - инициатор обмена (Initiator) и целевое устройство (Target). Арбитражем запросов на использование шины занимается специальный функциональный узел, входящий в состав чипсета (микропроцессорного комплекта) материнской (системной) платы.

В отличие от адаптеров остальных шин, компоненты карт PCI расположены на левой поверхности плат. По этой причине крайний PCI–слот обычно разделяет использование посадочного места адаптера с соседним ISA-слотом. Шина PCI являлась до последнего времени второй (после ISA) по популярности. В современных системах происходит отказ от шин ISA, и шина PCI выходит на главные позиции.

Появление процессоров Pentium III с поддержкой трехмерной графики, а затем Pentium IV потребовало уменьшения нагрузки на PCI, вызванной передачей видеоданных. В модернизованной PCI добавлен магистральный интерфейс AGP.

9. Шина AGP (Accelerated Graphics Port–ускоренный графический порт) это интерфейс для подключения видеоадаптера к отдельной магистрали AGP, имеющей выход непосредственно на основную память. Шина AGP может работать с частотой системной шины до 133 МГц и обеспечивает высокую скорость передачи графических данных. Ее пиковая пропускная способность в режиме четырехкратного умножения (AGP-4x - передаются 4 блока данных за один такт) имеет величину 1066 Мбайт/с, а в режиме восьмикратного умножения AGP8x - 2112 Мбайт/с. По сравнению с шиной PCI, в шине AGP устранена мультиплексированность линий адреса и данных (в PCI адрес и данные передаются по одним и тем же линиям) и усилена конвейеризация операций чтения–записи, что позволяет устранить влияние задержек в модулях памяти на скорость выполнения этих операций.

Конструктивно шина AGP представляет собой отдельной разъем, размещенный на материнской плате для установки печатной платы графического ускорителя.

В отличие от первых моделей компьютеров, имевших довольно ограниченный набор внешних устройств (принтер, сканер, клавиатура), к тому относительно низкому быстродействия, исключая дисководы, современные компьютеры благодаря широкому внедрению сетевых и мультимедийных технологий получили свое распоряжения широкий набор внешних устройств, отличающихся форматами, скоростями и объемами циркулирующих данных. В связи с этим особое место в архитектуре стали занимать локальные (периферийные) шины, обеспечивающие непосредственную связь процессора и памяти со скоростными устройствами периферию: сетевыми адаптерами, графическими адаптерами, дисководами DVD.

10. Шина FSB – соединяет процессор с основной памятью, подключается к северному мосту чипсета. В некоторых компьютерах для соединения процессора с внешней кэш-памятью второго уровня используется отдельная шина BSB. В процессорах Intel применяется шина Quad Pumped FSB. Это 64-х разрядная шина, передающая за один такт четыре пакета данных.

11. В новой архитектуре Nehalem процессоров Intel на смену шинам FSB пришла шина QPI (Quick Path Architecture). Шина организована по принципу «точка-точка» и позволяет обеспечить высокоскоростной обмен данными между процессором и внешней памятью, а также между процессором и контроллером ввода-вывода. Шина представляет собой два 20-ти битных соединения для передачи в прямом и обратном направлении.

Периферийные шины компьютера более разнообразны из–за разнообразия внешних устройств. Периферийные шины IDE (Integrated Drive Electronics), ATA (AT Attachment – подключаемый к AT), EIDE (Enhanced IDE), SCSI (Small Computer System Interface малый компьютерный интерфейс) используются чаще всего в качестве интерфейса только для внешних запоминающих устройств.

12. ATAPI (ATA Package Interface) – стандарт, созданный с тем, чтобы напрямую подключать к интерфейсу ATA не только жесткие диски, но и дисководы CD – ROM, стримеры, сканеры. Версии

интерфейса ATA-3 и Ultra ATA обслуживают диски большей емкости, имеют при этом скорость обмена до 33 Мбайт/с, поддерживают технологию SMART (Self Monitoring Analysis and Report Technology - технологию самостоятельного следящего анализа и отчета), позволяющую устройствам сообщать о своих неисправностях, и ряд других сервисов.

13. Шина SCSI (Small Computer System Interface – системный интерфейс малых компьютеров) была стандартизована в 1986 году. Интерфейс предназначен для соединения устройств различных классов – памяти прямого и последовательного доступа, CD-ROM, оптических дисков однократной и многократной записи, устройств автоматической смены носителей информации, принтеров, сканеров, коммуникационных устройств и процессоров. С точки зрения шины все устройства могут быть равноправными и является как инициаторами обмена, так и целевыми устройствами, однако чаще всего в роли инициатора выступает хост-адаптер. К одному контроллеру может подключаться несколько периферийных устройств, по отношению которым контроллер может быть как внутренним, так и внешним. Широкое распространение получили периферийные устройства со встроенным контроллером SCSI, к которым относятся накопители на жестких дисках, CD-ROM, стримеры.

Шина SCSI распространена в больших серверных системах, системах обработки графических данных. В персональных компьютерах шина SCSI из–за своей дороговизны широкого распространения не получила.

Структурная схема системной платы с сочетанием шин PCI, SCSI, AGP, FSB (шина «процессор – память») и независимой скоростной шиной процессора (локальная шина) представлена на рис.6.8.

На этом рисунке в качестве примера представлена компоновка шин в персональном компьютере Pentium IV. Центральное место занимает мост между шинами, соединяющий пять компонент компьютера: процессор, оперативную память, графический принтер, шину PCI и контроллер дисков стандарта ATAPI.

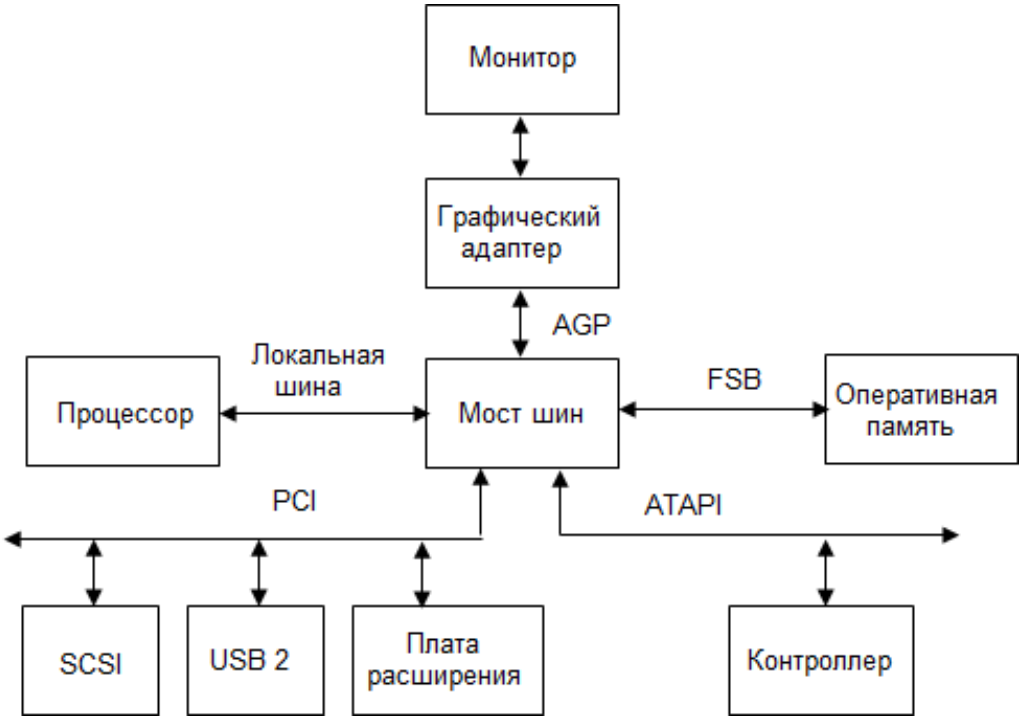


Рис.6.8. Компоновка шин в компьютере Pentium IV

Это уже конкретный компьютер, который может использоваться в качестве рабочей станции, сервера локальной сети или мощного персонального компьютера. В структуре этого компьютера присутствуют все основные, широко применяемые шины и соответствующие интерфейсы.

Системная шина, которая в более ранних моделях компьютеров присутствовала в качестве устройства, связывающего основные узлы – процессор, основную память и модули ввода/вывода, здесь уже отсутствует, ее заменят локальные шины и шины ввода/вывода. Основным связывающим узлом выступает мост локальных шин, который обеспечивает связь процессора с основной памятью и графическим адаптером, а также через шины ввода/вывода PCI, SCSI и контролер ATAPI с внешними устройствами.

Тенденция перехода от системных шин к локальным привела к появлению в современных компьютерах сочетания узлов коммутации с набором локальных шин. Примером этому может быть стандарт PCI-Express (PCI–E). Это программно совместимый с PCI масштабируемый, высокоскоростной интерфейс, обеспечивающий систему связи «точка – точка». Такой подход устраняет необходимость арбитража в многоточечных системных шинах,

обеспечивает малое время ожидания, упрощает оперативное подключение дополнительных системных устройств. Вариант стандартной компоновки компьютера с использованием системы PCI–E показан на рис.6.9.

Это практически переход от шинной организации взаимодействия устройств компьютера к технологии локальных сетей (коммутируемых сетей типа Ethernet).

В отличие от шины PCI, использовавшей для передачи данных общую шину, PCI-Express является пакетной сетью с топологией типа звезда. Устройства шины PCI - Express взаимодействуют между собой через среду, образованную коммутаторами. При этом каждое устройство связано напрямую соединением типа «точка-точка» с коммутатором. В отличие от параллельной шины PCI шина PCI-Express - последовательная асинхронная шина.

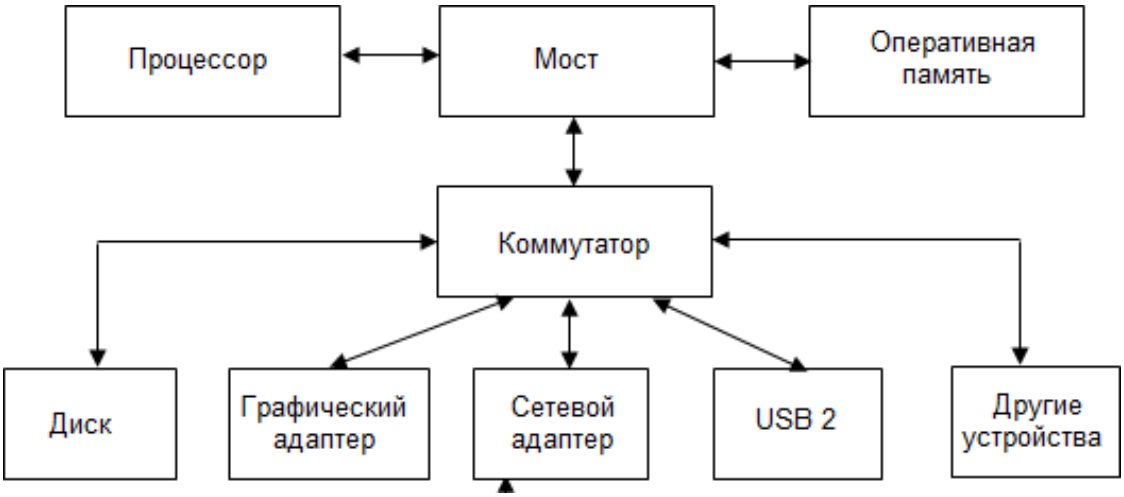


Рис.6.9. Компоновка шин с использованием стандарта PCI-Express

14.Шина Hyper Transport (HT) – это последовательно-параллельная шина с высокой пропускной способностью и малыми задержками. Технология шины HT используется различными производителями:

- компаниями AMD и Transmeta в процессорах Intel-8086;
- фирмами NVIDIA, VIA, AMD и HP в наборах системной логики ПК;
- фирмами IBM, HP, Sun Microsystems в серверах;
- компанией Cisco Systems – в сетевых маршрутизаторах.

Шина работает на частотах от 200 МГц до 3,2 ГГц, каждый пакет состоит из 32-разрядных слов, данные передаются в обеих направлениях.

Типы последовательных шин Serial ATA и USB. Интерфейс Serial ATA (SATA) — последовательного типа, служит для обмена данными с дисковыми накопителями. SATA является развитием параллельного интерфейса ATA. По сравнению с Parallel ATA, интерфейс SATA – более быстрый и надежный, имеет более низкое напряжение и меньшее число выводов. Serial ATA заменяет широкий и плоский ленточный кабель интерфейса Parallel ATA тонким и гибким кабелем последовательной связи, длина которого может достигать до 1 метра. Увеличенная длина позволяет более удобно размещать устройства хранения в корпусе. Serial ATA осуществляет проверку ошибок и их исправление. Вариант eSATA (External SATA) — интерфейс подключения внешних устройств, поддерживающий режим «горячей замены» (Hot-plug). Был создан после SATA в середине 2004 года. Основные особенности eSATA:

- разъёмы менее хрупкие и конструктивно рассчитаны на большее число подключений;
- требует для подключения два провода: шину данных и кабель питания. В новых спецификациях планируется отказаться от отдельного кабеля питания для выносных eSATA устройств;
- длина кабеля увеличена до 2 м (по сравнению с метровым кабелем у SATA);
- существенно снижается нагрузка на центральный процессор;
- уменьшены требования к сигнальным напряжениям по сравнению с SATA.

Шина USB (Universal Serial Bus) разработана для подключения к персональному компьютеру внешних периферийных устройств по технологии PLUG&PLAY(«подключай и работай»:

- подключение устройства к работающему компьютеру;
- автономное распознавание устройства сразу после подключения;
- автономная загрузка драйверов.

Данная шина предполагает использование одного общего для всех разъема, заменяющего многие порты компьютера. Отдельные

USB-устройства, например сканеры и цифровые камеры, могут работать без кабелей питания, тат как питание к ним подается кабелю USB. Стандарт USB позволяет подключать и отключать аппаратуру без перезагрузки компьютера.

По шине может подаваться питание для маломощных периферийных устройств. Преимущество — требование только одного прерывания в компьютере не зависимо от числа подключенных устройств. В настоящее время реализована высокая скорость передачи — 480 MB/s при длине кабеля до пяти метров. В настоящее время данный вид интерфейса также широко применяется для сопряжения с компьютером автономных модулей сбора и обработки измерительныхсигналов, в том числе биомедицинских.

Шина USB может использоваться для подключения любых устройств, кроме высокоскоростных накопителей на жестких дисках. Число подключаемых устройств может быть равным пяти. При увеличении числа подключаемых устройств используются USB-концентраторы. Системы, построенные на шинах USB, имеют топологию дерева. В настоящее время при создании локальных сетей и компьютерных систем сбора и обработки сигналов с удаленных датчиков (обычно не более 100 м) применяют коммуникационный беспроводный интерфейс, в частности технологии Wi-Fi, Bluetooth и ZigBee.

6.8. Адаптеры системного интерфейса

Принцип открытой архитектуры позволяет к основным блокам компьютера (процессор, материнская плата) подключать схемы управления различными элементами с использованием контроллеров. Они могут изготавливаться либо в виде отдельных конструктивных элементов (адаптеров), либо размещаться на материнской плате.

Адаптеры – это печатные платы, на которых размещены все необходимые для решения поставленных задач электронные компоненты, установочные разъемы и средства соединения с другими блоками. Адаптеры устанавливаются в разъем соответствующей шины, к ним подсоединяются управляемые устройства. Их можно условно различать либо по типу используемой шины, либо по типу размещенного на адаптере контроллера.

ISA - адаптеры прошли длинный путь вместе с персональными компьютерами PC - архитектуры. В свое время для ISA-шины были выпущены разнообразные адаптеры: управления дисками, видео, периферийными устройствами. Позже для COM и LPT-портов, управления работой винчестера и дисковода, джойстиком был разработан универсальный адаптер (multicard), имевший для подключения многопроводные плоские кабели.

Задачу формирования видеосигнала и передачи его на монитор обеспечивает видеоадаптер (часто называемая видеокартой). Для каждого типа мониторов имеется своя карта: MDA, CGA, EGA, VGA, SVGA карты. Для придания компьютеру дополнительных функций используются специализированные адаптеры: звуковые и сетевые карты, модемы. Начиная с модели i486, электронные блоки управляющих контроллеров стали устанавливаться на материнских платах. Началось с дисковых накопителей, к ним добавились порты ввода-вывода.

PCI-адаптеры. К моменту появления шины PCI многие контроллеры уже размещались на материнской плате, потребность в адаптерах уменьшилась. Эта шина способствовала появлению 3D, USB, FireWire, SerialATA. Сначала контроллеры изготавливались в виде PCI-адаптера. Постепенно эти контроллеры начали размещаться на материнской плате. Адаптеры локальных шин распространены меньше. Видео и звуковые карты рассмотрены в последующих разделах.

Вопросы для контроля

1. Что такое системный интерфейс?
2. В чем функции «малого» и «большого» интерфейсов?
3. Опишите функции регистров внутреннего интерфейса?
4. В чем заключаются функции модуля ввода/вывода?
5. В чем суть режима ввода/вывода по прерываниям?
6. Какие параметры заносятся в контроллер ПДП?
7. Опишите действия при передаче в режиме ПДП?
8. Каковы основные параметры компьютерных шин?
9. Каков режим функционирования системной шины?
10. Перечислите группы сигнальных линий шины управления.

ГЛАВА 7. УВЕЛИЧЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОЦЕССОРОВ

Всю историю развития вычислительной техники и связанных с ней областей можно описать как историю погони за наивысшей скоростью решения задач, так как практическая потребность всегда ставила перед человеком сложные задачи, на которые нужно было получать ответы. Это задачи строительства высотных заданий, проектирование новых типов самолетов, прогноз погодных явлений, поиски и разработка полезных ископаемых, создание новых лекарственных препаратов. Решение такого рода задач большой размерности на персональных компьютерах невозможно, нужны машины с очень высоко скоростью вычислений.

Методы повышения скорости вычислений.

1. За счет повышения быстродействия элементной базы (тактовой частоты). Быстродействие процессора растет пропорционально росту тактовой частоты, не требуется изменения системы программирования и пользовательских программ.
2. За счет увеличения числа одновременно работающих в одной задаче процессоров, АЛУ, умножителей, параллельных шин, то есть за счет параллелизма выполнения операций. Это требует использования методов параллельного программирования.

Параллельные системы по архитектуре разделяются на два класса:

- **конвейерные системы**, когда несколько специализированных блоков одновременно работают над частями одного потока команд.
- **параллельные системы**, когда множество команд одной программы одновременно выполняются множеством АЛУ или процессоров.

Рассмотрим некоторые наиболее широко применяемые методы увеличения производительности. Приводимые методы параллельной обработки в равной мере базируются на возможностях как аппаратной, так и программной частей процессоров.

Разработчики архитектуры компьютеров издавна прибегали к методам проектирования, известным под общим названием «совмещение операций», при котором аппаратура компьютера в

любой момент времени выполняет одновременно более одной базовой операции. Этот общий метод включает два понятия: параллелизм и конвейеризацию, которые отражают два совершенно различных подхода. При параллелизме высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

7.1. Конвейеризация вычислений

Конвейерная обработка команд. Конвейер команд основан на разделении подлежащей исполнению функции на более мелкие части (этапы) и выделении для каждой из них отдельного блока аппаратуры. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняется несколько команд. Конвейеризация увеличивает пропускную способность процессора (количество команд, завершающихся в единицу времени), но она не сокращает время выполнения отдельной команды.

Одна команда программы выполняется с помощью блоков различного функционального назначения: память, АЛУ, внутренние регистры, узлы управления. В обычном режиме отдельные составляющие одной команды - операционная и адресная часть - выполняются последовательно. Одновременная работа указанных блоков позволяет параллельно обрабатывать сразу несколько смежных команд. Например, одна арифметическая операция может быть разбита на логически самостоятельные микрооперации. Команда сложения с плавающей запятой может быть разделена на следующие микрооперации: вычитание порядков, выравнивание мантисс, сложение, нормализация.

В общем случае отдельные части команды, то есть ее номер, тип операции, сектор вида адресации, адресные части (от одной до трех) выполняются на отдельных аппаратных узлах процессора - счетчике команд, регистре команды, схеме управления, регистрах и дешифраторах адреса. Сам цикл исполнения команды также состоит из этапов: выборка команды, ее дешифрация, выборка данных, вычисление, запись результата (рис.7.1).

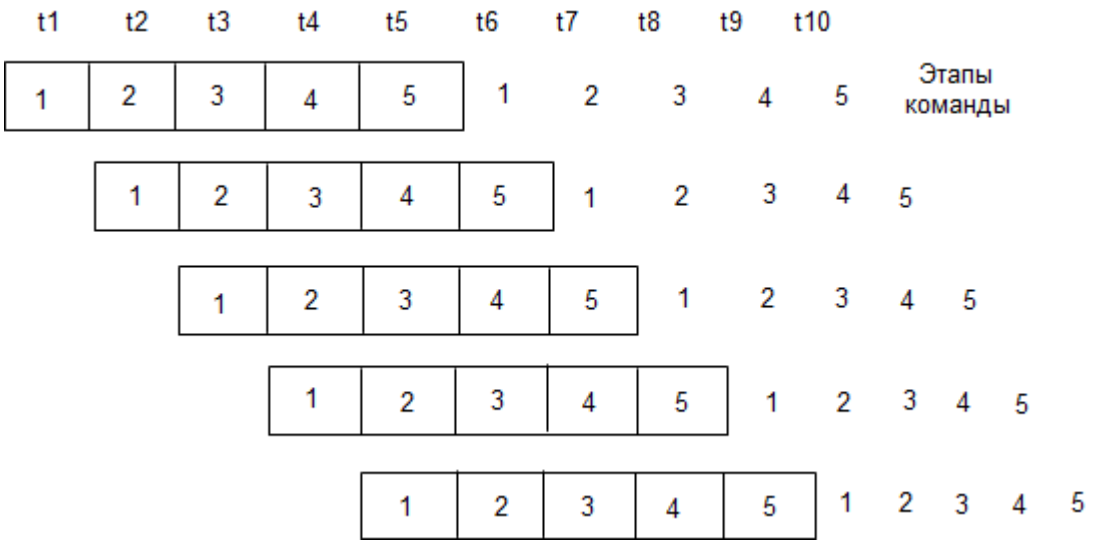


Рис.7.1. Принцип конвейерной обработки последовательности команд

Известно, что команды программы исполняются в пять этапов: выборка; дешифрация; вызов операндов; выполнение операции и запись результата в память.

Указанные этапы выполнения команды действуют по очереди в установленном порядке: этап 1 - выборка команды, этап 2 - ее дешифрация, этап 3 - выборка данных, этап 4 - исполнение операции, этап 5 — запись результата. Если условно представить каждый этап отдельно, то можно образовать последовательность их работы в виде таблицы.

В первый момент времени t1 реализуется этап выборки команды с помощью соответствующих узлов процессора. После завершения этапа 1 процессор с использованием других узлов переходит к выполнению этапа 2 этой же команды (дешифрация), а с помощью предыдущих освободившихся узлов - к выполнению этапа 1 уже следующей команды программы (момент t2). Таким образом в момент t2 идет исполнение этапа 2 первой команды и одновременно этапа 1 второй команды.

В момент t3 процессор реализует этап 3 первой команды (выборка данных), этап 2 второй команды (дешифрация) и этап 1 третьей команды(выборка команды).

В момент t4 уже одновременно на различных узлах процессора реализуется четыре этапа следующих друг за другом команд программы.

Наконец, в момент времени t5 процессор исполняет последний 5 этап первой команды и одновременно реализует 1 этап пятой по счету команды программы. Таким образом, начиная с момента t5 оказываются задействованными все пять основных функциональных узлов процессора и идет одновременная конвейерная обработка пяти очередных, следующих друг за другом команд программы. При обычном, неконвейерном способе исполнения команд в каждый момент времени t реализуется одна команда программы. Таким образом, при неконвейерном способе 5 команд, состоящих из пяти частей были бы реализованы за 25 шагов процессора. При реализации конвейерной обработки (как показано на рис.7.1) обработка будет занимать всего 9 тактов процессора.

Конвейеризация эффективна только тогда, когда исполняемые команды имеют приблизительно равное время реализации. Например, раздельное выполнение одной операции в несколько этапов при конвейерной обработке эффективно только тогда, когда загрузка конвейера близка к полной, а скорость подачи очередных операндов программы соответствует максимальной производительности аппаратных средств. Если возникнет задержка, количество параллельно выполняемых операций уменьшится и суммарная производительность снизится. При этом могут возникать конфликтные ситуации, которые могут быть ликвидированы с помощью операционной системы. Операционная система во всех этих ситуациях должна иметь механизмы уменьшения временных потерь от внезапного возникновения конфликтных ситуаций.

Конфликты возникают в том случае, когда отдельные аппаратные блоки не могут поддерживать все возможные комбинации (чередования) команд в режиме одновременного выполнения с совмещением. Это возникает из-за того, что последовательность команд программы может содержать как быстро реализуемые логические команды, так и долго исполняемые арифметические команды двойной длины. Последние могут вызвать задержки цикла исполнения, что потребует вмешательства операционной системы. Задержка может объясняться как лишним тактом в обработке, так и лишним тактом в обращении к памяти. В этом случае система управления конвейером обязана приостановить работу на один такт.

При структурном риске чаще всего одновременно используются один и тот же ресурс – память. В обычном конвейере сразу три этапа (Чтение команды, Выбор операндов, Запись результатов) связанны с обращением к памяти. Такие конфликты разрешаются за счет:

- модульного построения памяти, при наличии нескольких модулей высока вероятность того, что будет обращение к разным модулям;
- наличие кэш–памяти, обращение может разделится между ОП и кэш– памятью (иногда кэш–данных и кэш–память разделены);
- сама память ОП также разделена на память данных и память команд (гарвардская архитектура).

Скорость конвейера определяется самой длительной ступенью, где бы она ни располагалась в конвейере. Время заполнения или загрузки конвейера сказывается на производительности тем меньше, чем больше команд им обрабатывается. Остановка конвейера приводит к затратам времени на его повторное заполнение. Причинами остановок конвейера являются конфликты. При конвейерной обработке возникают конфликты, когда очередная команда не может быть обработана в предназначенном ей такте.

Существуют три типа конфликтов:

1. Структурные.
2. Конфликты по данным.
3. Конфликты по управлению.

Структурные конфликты. Структурные конфликты вызваны недостаточностью ресурсов вычислительной системы для обеспечения и обработки возможных комбинаций команд. Их причинами могут быть:

- не полностью конвейерные функциональные устройства системы;
- недостаточное дублирование ресурсов системы;
- кэш-промахи;
- общий конвейер для команд и данных.

На рис. 7.2 приведена ситуация с одновременным обращением двух команд Ск и Сп к одновходовой памяти, что приводит к структурному конфликту. В случае одновременного запроса к памяти и нехватки портов происходит приостановка конвейера до

завершения обращения к памяти предыдущей команды. Образуется «конвейерный пузырь». Во многих случаях приостановка конвейера может быть вызвана не синхронной деятельностью других конвейерных устройств.

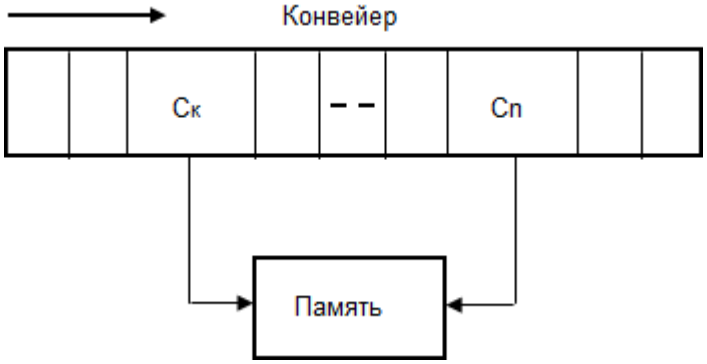


Рис.7.2. Одновременное обращение двух команд к одной ячейке памяти

Подобных конфликтов можно избежать за счет модульного построения памяти или использования кэш-памяти, т.к. имеется вероятность того, что команды будут обращаться либо к разным модулям памяти, либо одна из них будет обращаться к ОП, а другая к кэш-памяти.

На рис. 7.3 показано решение структурного конфликта путем использования двух дополнительных конвейеров К1 и К2 для одновременной обработки двух команд Ск и Сп.

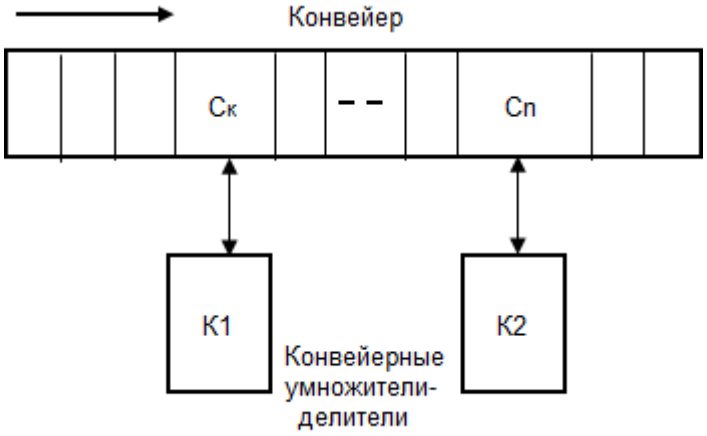


Рис.7.3. Одна из схем решения структурного конфликта

Другим выходом из положения при структурных конфликтах является использование многовходовой памяти.

Конфликты по данным. Конфликт по данным возникает при наличии логических межкомандных зависимостей, т.е. при

использовании одной командой результата выполнения другой команды.

Предположим, что две команды в конвейере Ск и Cn предусматривают обращение к родной и той же переменной *X,* причем команда Ск предшествует команде Cn . При этом могут возникнуть три типа конфликтов:

- «чтение после записи», когда команда Cn читает *X* до того, как команда Ск успела записать новое значение переменной *X*;

- «запись после чтения», когда команда Cn записывает новое значение *X* до того, как команда Ск успела прочитать *X*;

- «запись после записи», когда команда Cn записывает новое значение *X* до того, как команда Ск успела записать в качестве *X* свое новое значение.

Чаще всего бывает конфликт по данным - «чтение после записи», т.к. операция чтения в цикле команды (такт 3 – вызов операндов)) предшествует операции записи (такт 5 – запись результата).

Для борьбы с конфликтами по данным применяются как программные, так и аппаратные средства.

Программные методы борьбы ориентированы на устранение самой возможности конфликта еще на стадии компиляции программы. Оптимизирующий компилятор создает такой объектный код, чтобы между командами, склонными к конфликтам, находилось достаточное количество нейтральных команд, или команд отсутствия операции.

Наиболее часто применяемым аппаратным методом является остановка команды Cn на несколько тактов с тем, чтобы команда Ск успела пройти ступень конвейера, где мог произойти конфликт.

Простейший конвейер ориентирован на так называемые линейные программы, когда ступень выборки извлекает команды из последовательных ячеек памяти в соответствии с состоянием счетчика команд процессора. В линейной программе адрес очередной команды формируется автоматически за счет прибавления к содержимому счетчика команд числа, равного длине текущей команды в байтах. Однако реальные программы редко бывают линейными, в них обязательно присутствуют команды управления,

изменяющие последовательность вычислений: команды условного и безусловного перехода, команды вызова процедуры и возврата из процедуры. Такие команды перехода в программах могут приводить к приостановке конвейера на несколько тактов, из-за чего производительность процессора снижается.

Суперконвейерные процессоры. Увеличение темпа работы конвейера можно добиться разбиением каждой ступени конвейера на N подступеней при одновременном увеличении тактовой частоты внутри конвейера. Каждая из пяти ступеней конвейера на рис.7.1 разбивается на две более простые подступени. Выполнение операций на каждой полступени занимает половину тактового периода, при этом тактирование операций внутри конвейера производится с частотой, вдвое превышающей частоту подачи команд на вход конвейера. Тогда на каждой ступени конвейера можно в пределах одного внешнего тактового периода выполнить две команды.

При таком подходе главным требованием к аппаратуре процессора является возможность реализации операции в каждой подступени наиболее простыми техническими средствами с минимальными затратами времени. К примеру, длина конвейера в процессорах UltraSPARC III и Pentium 4 составляет соответственно 14 и 20 ступеней.

На рис.7.4 приведена схема взаимодействия многоступенчатых конвейеров процессора Pentium 4 с внутренней кэш-памятью и регистрами общего назначения РОН.

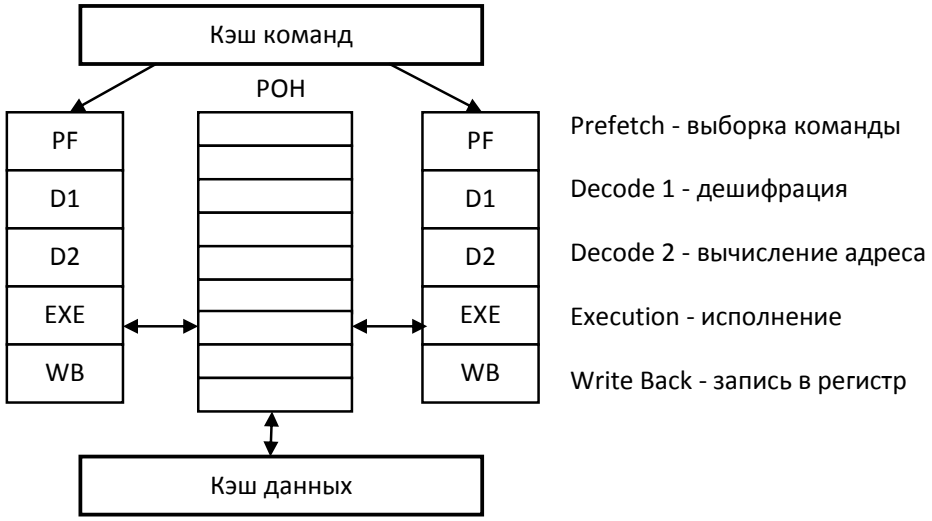


Рис.7.4. Схема взаимодействия компонентов Pentium 4

Каждый конвейер процессора разбит на 14 ступеней, при этом каждая ступень реализует свой шаг в выполнении алгоритма обработки.

7.2. Гарвардская архитектура процессоров обработки

Это один из методов увеличения производительности за счет эффективного использования памяти.

Обычно высокая производительность при существенных затратах на аппаратном и программном уровне бывает необходима в двух случаях:

- при решении вычислительных задач, когда исходные данные хранятся в памяти большого объема, а программное обеспечение решает задачу большой размерности как модели реального процесса;
- при обработке большого потока входных переменных в системах реального времени, когда скорость обработки компьютера должна обеспечить получение текущих результатов в темпе их поступления.

Для второго типа обработки, когда встала задача увеличения скорости вычислений в АЛУ за счет ускорения обмена данными между оперативной памятью и АЛУ, была применена так называемая гарвардская архитектура процессоров.

Для увеличения быстродействия АЛУ стали использоваться различные архитектурные приемы: матричные умножители, сопроцессоры плавающей запятой, аппаратные узлы реализации процедур округления, преобразования формата, вычисления абсолютной величины. Таким образом, время цикла исполнения одной команды было сведено к минимуму.

Следующим шагом совершенствования архитектуры с целью сокращения времени обмена «память – АЛУ» было решение разделить оперативную внутреннюю память процессора на отдельную память команд и отдельную память данных со своими аппаратными средствами доступа. Это позволило обеспечить параллельную выборку команды и необходимых для ее выполнения данных с одновременной загрузкой в исполнительные устройства. Другими словами, такты выборки команды и выборки данных исполняются одновременно.

7.3. Архитектура процессоров с сокращенным набором команд

Расширенная система команд (несколько десятков и даже сотен) универсальных процессоров увеличивает возможности компьютера с точки зрения класса решаемых задач и качества оказываемых услуг пользователю. Опыт применения и использования возможностей процессоров с расширенной системой команд (**CISC–архитектура**) показали, что мощная система команд чаще всего полностью не используется, аппаратные средства, участвующие в исполнении многих команд, используются неэффективно. Микропрограммная память, которая формирует сигналы управления, получается слишком громоздкой. Процент постоянно, интенсивно используемых команд оказывается небольшим. Так как каждая команда преобразуется в последовательность микроинструкцией для определенных функциональных узлов, то устройство управления получается весьма сложным и медленным в работе.

Для компьютеров с CISC–архитектурой характерны не только большое количество самих типов и модификаций, но и разнообразие способов адресации и форматов команд различной разрядности. Такой подход оправдан для универсальных компьютеров широкого применения со значительными стоимостными показателями.

RISC – архитектура (архитектура с сокращенным набором команд) появилась как альтернатива CISC–архитектуре с точки зрения увеличения быстродействия. Особенностью RISC–архитектуре является сокращенный набор команд, которые в большинстве исполняются за один такт процессора, при этом команды работают с операндами, размещаемыми во внутренних регистрах процессора (РОН). Для обращения к оперативной памяти требуются специальные команды. В RISC–процессорах значительно уменьшено число форматов команд, соответственно ограничены способы адресации. Это позволило упростит аппаратуру и повысить быстродействие.

Идеология построения RISC-процессоров (RISC- Reduced Instruction Set Computing) складывалась в конце 1970-х - начале 1980-х годов, когда потребовались новые идеи для повышения производительности процессоров. Выводы различных групп исследователей были обобщены в виде так называемого правила

"80/20": 80 % времени выполнения программ занимает выполнение 20 % команд, входящих в состав системы команд. То есть в определении производительности процессора основную роль играет лишь пятая часть всех команд, остальные же команды встречаются достаточно редко, и время их выполнения существенного влияния на производительность процессора не оказывает. Исходя из этого было принято решение разработать процессор, в котором выделенная небольшая группа команд выполнялась бы максимально быстро за счет ее аппаратной реализации, а остальные команды либо вообще удалялись из системы команд, либо реализовывались на микропрограммном уровне.

Сложившаяся в результате этого RISC-архитектура опиралась на следующие принципы:

- набор команд сокращен до 70-100 команд (вместо нескольких сотен у CISC-микропроцессоров);
- большинство команд выполняется за 1 такт, и лишь немногие - за несколько или даже несколько десятков тактов;
- все команды обработки данных оперируют только содержимым регистров процессора, а для обращения к более медленной оперативной памяти предусмотрены исключительно инструкции вида "загрузить в регистр" и "записать в память";
- команды имеют простой, четко заданный формат;
- из набора команд исключены редко используемые инструкции, а также команды, не вписывающиеся в принятый формат;
- состав системы команд должен быть удобным для применения оптимизирующих компиляторов с языков высокого уровня.

Такой подход позволил уменьшить объем аппаратуры процессора за счет сокращения блока управления примерно в 10 раз, существенно увеличить тактовую частоту работы процессора и снизить его тепловыделение.

Несмотря на свое название, основой RISC-архитектуры является то, что вся обработка сосредоточена только во внутренних регистрах процессора.

Так как вся обработка проходит в регистрах, отпадает необходимость в большом количестве режимов адресации операндов, а в системе команд можно применять трехадресные команды,

наиболее эффективные с точки зрения организации вычислительного процесса и в то же время не имеющие их главного недостатка - большой длины команды. Простой формат команды легко поддается декодированию на соответствующей ступени работы конвейера.

Естественно, что этот подход потребовал использования в микропроцессоре регистровой памяти большого объема (до 128 регистров). А для обеспечения согласованной работы быстрых внутренних конвейеров и относительно медленной оперативной памяти в RISC-процессорах предусматривается кэш-память большой емкости.

Наличие большого количества регистров создает хорошую основу для работы оптимизирующих компиляторов, которые эффективно используют все конвейеры процессора.

Простой формат команды и ориентация на регистровую обработку позволили безболезненно внедрить в RISC-процессорах конвейерный принцип обработки информации.

Все это обеспечило преобладание процессоров такого типа в тех областях, где производительность являлась основополагающим фактором, например, в серверах. В то же время они не нашли своего места на наиболее развитом рынке вычислительной техники - рынке персональных компьютеров. Тому есть несколько причин:

- дороговизна RISC-процессоров и систем на их основе, т.к. разработчики использовали в них решения, слишком дорогие для персональных компьютеров, дешевые варианты RISC-компьютеров стоили гораздо дороже сравнимых с ними ПК на базе процессоров Intel по причине малых объемов производства;

- отсутствие широких наработок в области программного обеспечения, различные RISC-платформы обычно использовали несовместимые между собой разновидности операционной системы Unix, для которых существовало значительно меньше программ, главным образом научно-технических;

- RISC-процессоры по своему основополагающему положению обладают несовместимыми с i8086 наборами команд, поэтому единственным способом исполнения кода i8086 была эмуляция, которая снижала производительность от десятков до сотен процентов, что сводило на нет скоростные преимущества RISC-процессоров;

Развитие архитектуры RISC-процессоров шло по нескольким направлениям. За счет повышения технологических возможностей производства процессоров смягчились требования к составу и форматам используемых команд. В настоящее время их системы команд расширились с первоначальных 70-100 до 100-120. Увеличилось также и количество используемых форматов команд. Однако при этом основной принцип RISC-архитектуры остается неизменным: обработка данных выполняется только над содержимым внутренних регистров процессора без обращения к оперативной памяти.

Вместо требования выполнения команды за один такт используется требование получения очередного результата в очередном такте работы, то есть фактически закреплен принцип конвейерной обработки данных. Для обработки данных процессоры получили не один, а несколько конвейеров со своими исполнительными устройствами.

Наиболее известными RISC-процессорами в сейчас являются семейства процессоров SPARC фирмы Sun Microsystems, процессоры Alpha 21x64 фирмы Digital EquIPment и Rx000 фирмы MIPS Computer Systems. Среди фирм, выпускающих RISC-процессоры, находятся также Intel, Hewlett Packard.

Совместный проект компаний Apple, Motorola и IBM – микропроцессор Power PC (Performance Optimization With Enhanced RISC) - был ориентирован на создание недорогого, но мощного RISC-процессора и платформы для него. Рассмотрим организацию работы RISC-процессора на примере модели Power4 фирмы IBM. Процессоры Power4 содержат свыше 180 млн. транзисторов на кристалле и выпускаются с частотами до 1,7 ГГц.

Структура процессора Power4 представлена на рис.7.5.

Являясь RISC-процессором, Power4 имеет много особенностей.

Во-первых, Power4 стал первым процессором, в котором реализована идея размещения нескольких процессорных ядер на одной микросхеме. В состав Power4 входят два однотипных процессорных ядра.

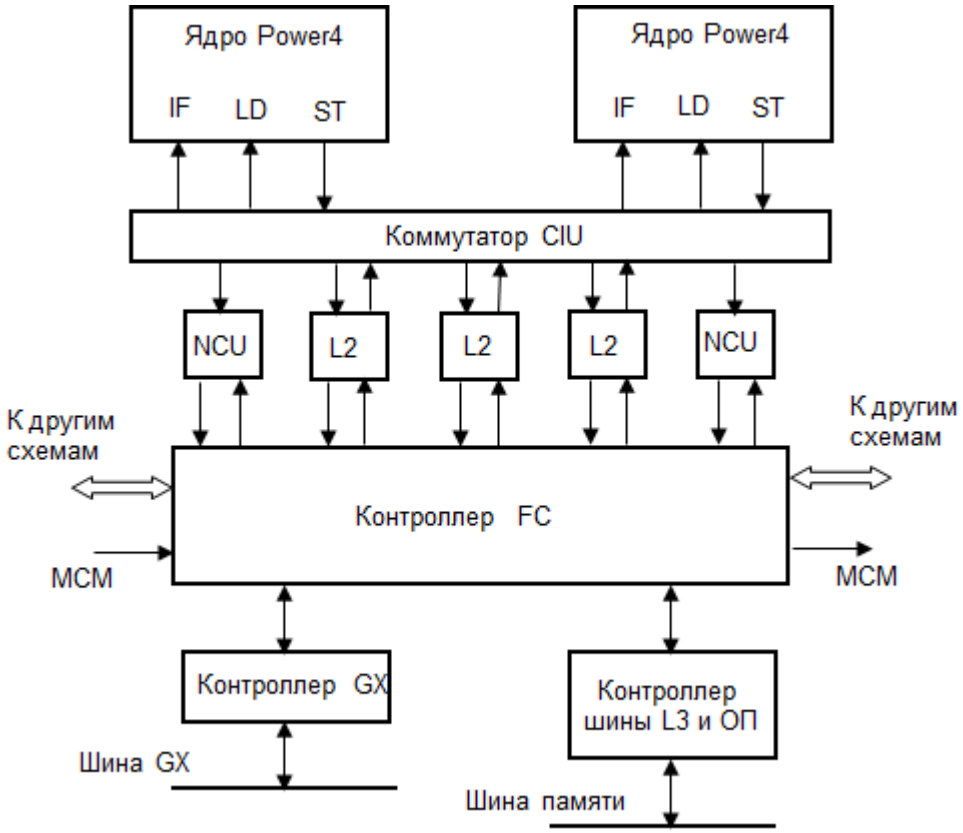


Рис.7.5.Структурная схема процессора Power4

Во-вторых, на кристалле процессора Power4 содержатся аппаратные средства, определяющие архитектурные черты процессорных систем на его основе. Это порты каналов, напрямую связывающих Power4 с другими процессорами внутри так называемого микросхемного модуля-MCM (Multi-Chip Module), канал ввода-вывода (шина GX), контроллер шины оперативной памяти и кэша третьего уровня L3, разделяемого всеми процессорами MCM. Каждый двухъядерный процессор Power4 упакован в керамический мультипроцессорный модуль вместе с тремя другими процессорами. Такой микросхемный модуль MCM, содержащий четыре процессора Power4, в итоге объединяет восемь процессоров.

Каждое из ядер имеет свой кэш первого уровня, отдельный для команд и данных. Кэш емкостью 64 Кбайт является прямо адресуемым и может читать или писать 32 байта за такт, что эквивалентно 8 командам. Кэш второго уровня L2 у ядер общий. Он является уже 8-канальным и имеет относительно небольшую емкость 1,4 Мбайт. Это, однако, компенсируется использованием с Power4

кэша L3 большой емкости (32 Мбайт на двухъядерную микросхему Power4 или 128 Мбайт на микросхемный модуль MCM из четырех).

Кэш второго уровня организован в виде трех независимых блоков, каждый со своим контроллером и интерфейсом доступа со стороны ядра CIU (core Interface unit). Распределение обработки запросов в кэш второго уровня между тремя блоками способствует увеличению производительности. Каждое ядро имеет по три порта к CIU (для выборки команд - IF, загрузки регистров - LD и сохранения данных - ST). Контроллеры кэша L2 работают одновременно, обрабатывая по 32 байта за такт процессора. Для поддержания согласованного представления данных в L2 используется расширенный протокол MESI, включающий 7 состояний. Кэш L2 разделяется не только двумя процессорами кристалла, но также и внешними процессорами других кристаллов микросхемного модуля через линии связи шириной 16 байт, работающими на частоте более 500 МГц. Логической частью кэша второго уровня являются два так называемых некэширующих устройства NCU (Non-Caching Unit), каждое приписанное к своему процессору. NCU-устройства отвечают за сериализацию команд и выполнение некэшируемых операций в иерархии памяти.

Через специальный контроллер FC (fabric conTRoller) данные могут быть направлены в кэш третьего уровня L3 и оперативную память. Весь ввод-вывод осуществляется через контроллер шины GX,имеющий порт к FC.

Порт кристалла Power4, предназначенный для подключения кэш-памяти L3, имеет ширину 16 байт для каждого из двух направлений пересылки данных. Порт функционирует на 1/3 от тактовой частоты процессоров кристалла, что обеспечивает пропускную способность к памяти на уровне 10 Гбайт/с. Кэш L3 является общим для модуля MCM, который может содержать кроме микросхемы кэша третьего уровня четыре микросхемы Power4. Для передачи данных между микросхемами Power4, входящими в MCM, в FC предусмотрены выделенные порты, ориентированные на создание мультипроцессорной системы.

Процессор Power4 несет в себе все черты RISC-архитектуры. Он имеет 32 регистра общего назначения и 32 регистра для работы с

числами с плавающей точкой. Все вычисления производятся в регистрах, а не в основной памяти. Система команд - трехадресная. Каждая команда имеет длину 32 бита. Первые 6 бит определяют код операции, а остальные имеют различное значение, зависящее от команды. Тот факт, что команды имеют фиксированную длину, позволяет процессору выполнять их более эффективно.

7.4. Параллельное выполнение нескольких команд программы

VLIW-архитектура (Very Large Instruction Word) процессоров работают практически по правилам: команда, выдаваемая процессору на каждом цикле, определяет не одну операцию, а сразу несколько. Команда VLIW-процессора состоит из набора полей, каждое из которых отвечает за свою операцию, например, за активизацию функциональных устройств, работу с памятью, операции с регистрами. Если какая-то часть процессора на данном этапе выполнения программы не востребована, то соответствующее поле команды не задействуется.

Для того, чтобы работа устройств по времени не пересекалась усовершенствованный компилятор процессе обработки исходной программы выделяет те команды, которые, принципе могут быть реализованы параллельно. Выделив такие команды, компилятор может их объединить в одну сверхдлинную команду, состоящую из нескольких простых и реализуемых параллельно. Количество команд, объединяемых в одну сверхдлинную, должно быть равно количеству исполнительных функциональных блоков. Длина VLIW- команды составляет 256 - 1024 бит, т.е. от 32 до 128 байт, она содержит несколько полей по числу составляющих простых команд.

В качестве простых команд, образующих сверхдлинную команду обычно используются команды RISC- типа, поэтому VLIW-архитектура считается дальнейшим развитием RISC–систем. Число полей команды в сверхдлинной команде равно числу реализующих их устройств и колеблется в пределах от 3 до 20. Упрощенная схема доступа к данным, организованным в виде одного регистрового файла, со стороны функциональных блоков, реализующих одну

команду, значительно упрощает процессы адресации и чтения операндов.

Узким местом VLIW-архитектуры является сложность программы-компилятора, способного исследовать исходную программу, найти и объединить в длинные командные слова взаимозависимые инструкции. Увеличение сложности транслирующей программы увеличивает время формирования сверхдлинной команды. Программисты доступа к внутренним VLIW –командам не имеют, все программное обеспечение базируется на низкоуровневых программах трансляции команд CISC – процессоров в команды VLIW.

Пример реализации подобной архитектуры приведен на рис.7.6, где представлен сигнальный процессор TMS320C62.

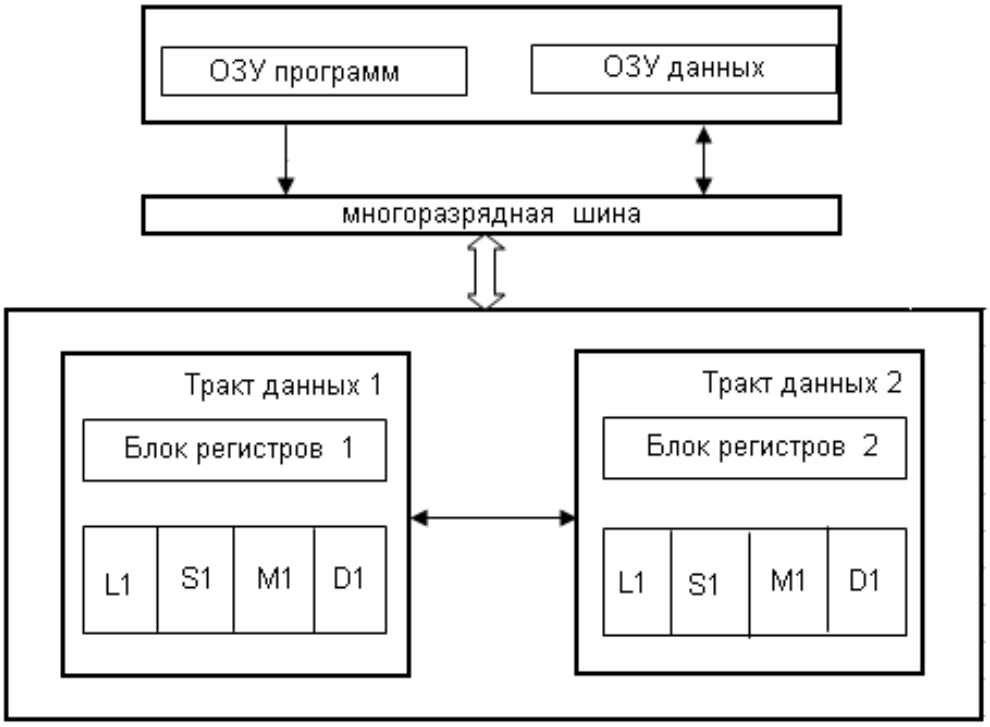


Рис.7.6. Обобщенная схема VLIW – процессора

Технология VLIW – это параллельное выполнение нескольких команд. В основе лежит применение «разумных» компиляторов. В процессе предварительного анализа программы компилятор объединяет параллельно выполняемые команды (без конфликтов между ними) в пакеты. Эти пакеты представляют собой сверхдлинные команды.

Правила объединения:

- количество простых команд, объединяемых в одну сверхдлинную команду, должно быть равно числу имеющихся в процессоре исполнительных блоков (L1, S1, M1, D1);

- в сверхдлинную команду входят только такие простые команды, которые исполняются разными функциональными блоками, то есть обеспечивается одновременное выполнение всех составляющих сверхдлинной команды.

Функциональные блоки могут быть следующими: блок сложения с ПЗ (L1), блок умножения с ПЗ (S1), арифметическое действие с фиксированной запятой (M1), блок реализации переходов (D1). Могут быть и другие типы функциональных блоков.

7.5. Суперскалярная обработка

Суперскалярная обработка предполагает предварительное переупорядочение команд программы с целью составления потоков однотипных команд. Однотипные команды одного потока проще реализуются с помощью конвейерной обработки, поэтому суперскалярные процессоры представляют собой структуры с несколькими конвейерами обработки (рис.7.7).

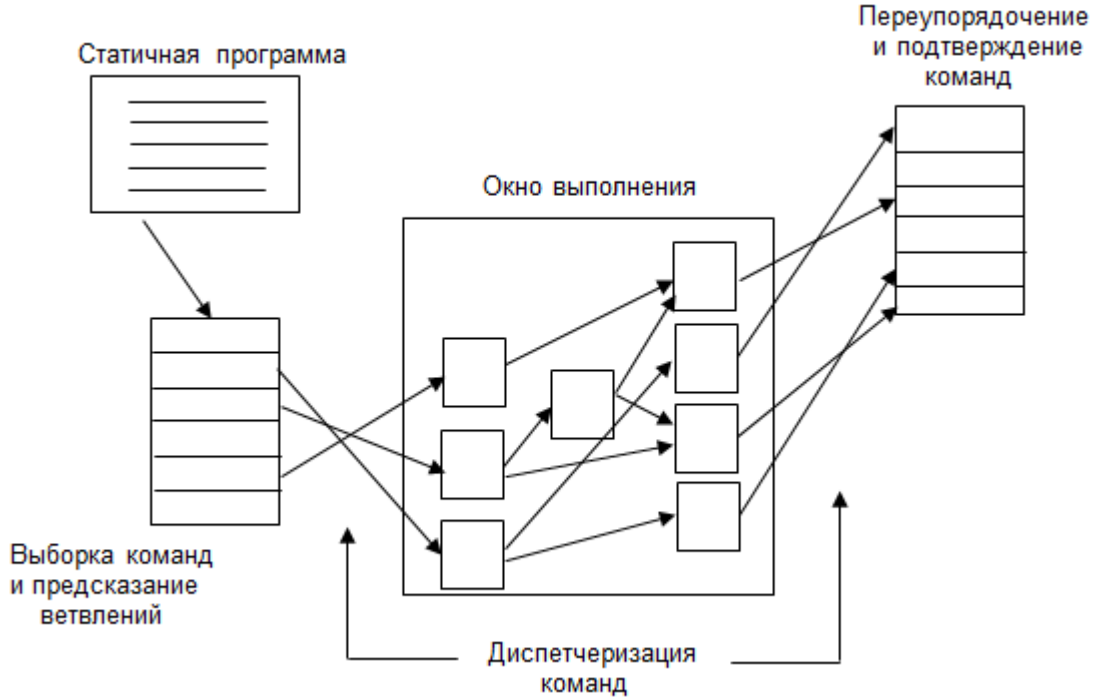


Рис.7.7. Принцип работы суперскалярного процессора

По классификации Флинна к классу SIMD относятся векторные и матричные процессоры, в которых параллельно обрабатываются операнды, являющиеся компонентами векторов и массивов. В отличие от них суперскалярным называется процессор, в котором могут параллельно обрабатываться две или более различные команды.

Выше отмечалось, что среднее время выполнения команды в одном конвейере стремится к одному такту при увеличении числа команд в обрабатываемой последовательности.

Для дальнейшего увеличения производительности используют несколько параллельных конвейеров. При реализации множества однотипных команд конвейер работает быстрее и ритмичнее. Переупорядочением команд и восстановлением первоначального порядка их исполнения является функцией компилятора, т.к. именно компилятор занимается преобразованием программы с языка высокого уровня в машинный код.

Рассмотрим особенности суперскалярной архитектуры на примере процессора Pentium Pro (рис. 7.8).

В этом процессоре реализованы новые решения:

- двухпотоковая суперскалярная организация, допускающая параллельное выполнение двух простых команд;
- наличие двух независимых двухвходовых множественно-ассоциативных кэшей для команд и для данных;
- динамическое прогнозирование условных переходов;
- конвейерная организация (восемь ступеней) устройства с плавающей точкой;
- совместимость с процессорами младших моделей.

Блоки в процессоре выполняют следующие функции.

ALU (U) – арифметико-логическое устройство для выполнения простых и сложных команд;

ALU (V) – арифметико-логическое устройство для выполнения только простых команд

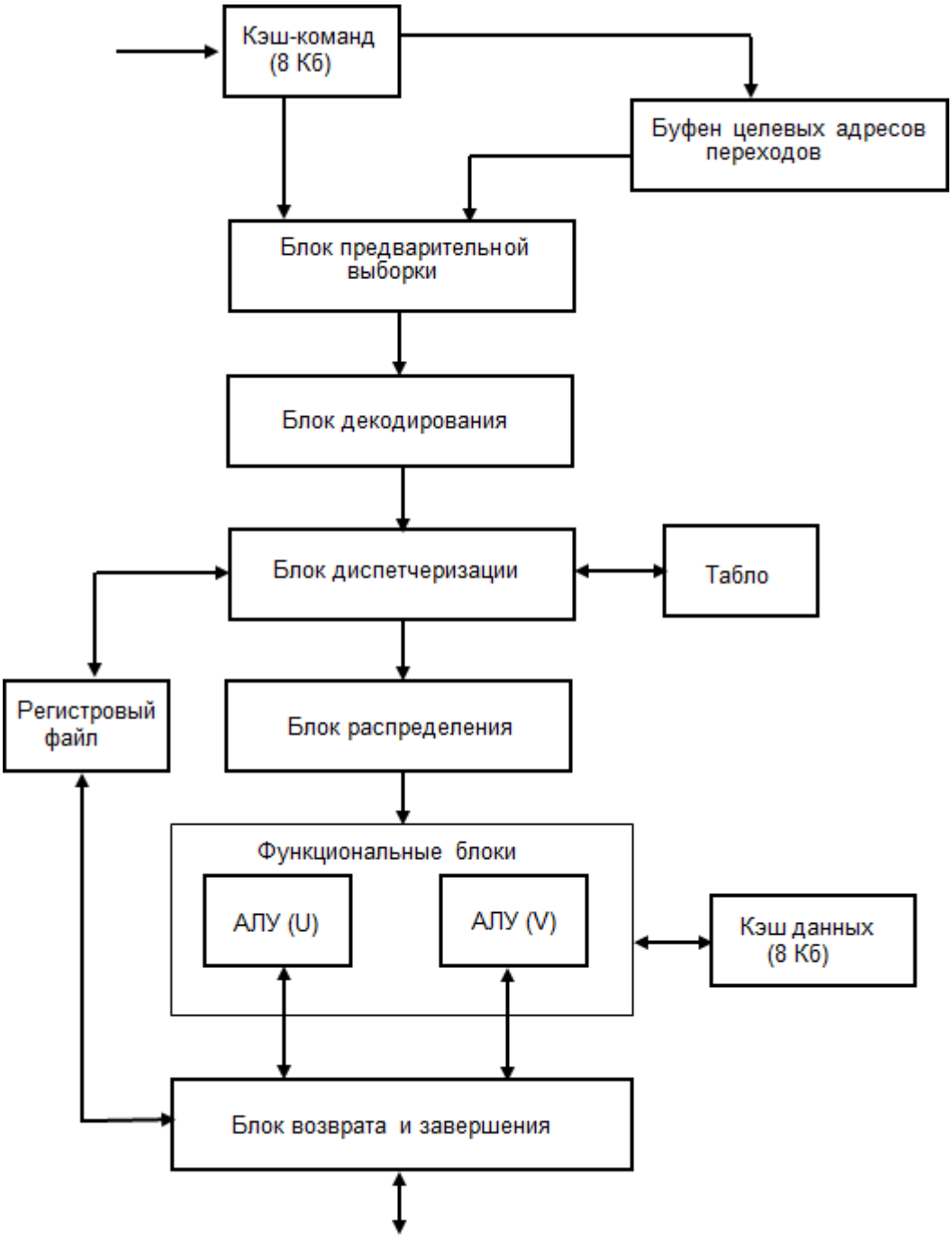


Рис.7.8. Схема работы процессора Pentium Pro

1. Блок выборки команд извлекает их из основной памяти через кэш-память команд, хранит с помощью счетчика команд очередь выбранных команд и обрабатывает команды условного перехода.

2. Блок декодирования расшифровывает код операции, содержащийся в извлеченных из кэш-памяти командах, и преобразует их в соответствующую последовательность микрокоманд.

3. Блоки диспетчеризации и распределения взаимодействуют между собой и в совокупности играют роль контроллера трафика, управляют потоком команд. Очередь блока распределения часто рассредотачивается по нескольким самостоятельным буферам – накопителям, предназаначенным для хранения команд, которые уже декодированы, но не выполнены.

4. Блок возврата и завершения восстанавливает естественную последовательность команд и подготавливает полученные результаты для ожидающих их команд.

Каждый накопитель команд связан со своим функциональным блоком, поэтому число накопителей обычно равно числу функциональных блоков. По отношению к блоку диспетчеризации накопители команд выступают в роли виртуальных функциональных устройств. Оба вида очередей показаны на рис.7.9.

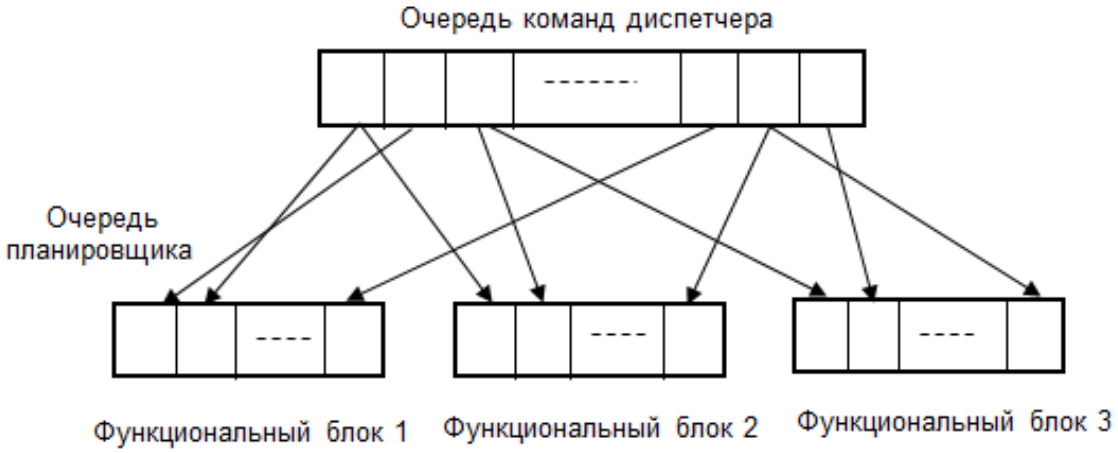


Рис.7.9. Очереди диспетчеризации и распределения

В дополнение к очереди, блок диспетчеризации хранит также список свободных функциональных блоков в специальном блоке – табло. Оно используется для отслеживания состояния очереди распределения. Табло определяет порядок выдачи команд на свободные функциональные блоки.

Один раз за цикл блок диспетчеризации извлекает команды из своей очереди, считывает из регистров операнды этих команд, после чего, в зависимости от состояния табло, помещает команды и операнды в очередь распределения. Эта операция выдачи команды. Блок распределения в каждом цикле проверяет каждую команду в своих очередях на наличие операндов и начинает выполнение таких команд в соответствующем функциональном блоке.

Исполнение операций обработки реализуется в узле функциональных блоков. Примерами функциональных блоков в различных типах процессоров могут служить целочисленные операционные блоки, блоки умножения и сложения с плавающей точкой, блоки доступа к памяти. Когда исполнение команд завершается, ее результат записывается и анализируется блоком возврата и завершения.

Суперскалярная обработка предполагает параллельную работу максимального числа исполнительных блоков, что возможно только при одновременном выполнении нескольких скалярных команд. Последнее условие хорошо сочетается с конвейерной обработкой, при этом желательно, чтобы в суперскалярном процессоре было несколько конвейеров.

Подобный подход реализован в процессоре Intel Pentium, где имеются два конвейера, каждый со своим АЛУ.

Вопросы для контроля

1. Назовите методы повышения скорости вычисления компьютеров?
2. Рассмотрите принцип конвейерной обработки?
3. Назовите типы конфликтов при конвейерной обработке?
4. Каким образом разрешаются конфликты по данным при конвейерной обработке?
5. Каковы особенности Гарвардской архитектуры процессоров?
6. Перечислите основные принципы построения и особенности архитектуры RISC-процессоров?
7. В чем особенности и преимущества архитектуры VLIW-процессоров?
8. Рассмотрите пример конкретной реализации VLIW-архитектуры.
9. Опишите принцип организации вычислительного процесса при суперскалярной обработке данных.
10. Перечислите блоки процессора Pentium Pro и их основные функции?

ГЛАВА 8. СРЕДСТВА РАСШИРЕНИЯ ФУНКЦИЙ ПРОЦЕССОРОВ

Под функциональным расширением понимается применение дополнительных вычислительных средств поддержки работы процессора. В качестве средств функционального расширения и поддержки основного процессора компьютера выступают дополнительные вычислительные устройства в виде видеоадаптеров, сетевых карт и сопроцессоров. В режимах обработки звуковых и речевых файлов, видеоданных, приема и отправления сообщений из сети, выполнения наиболее трудоемких вычислительных алгоритмов указанные средства выполняют большую по объему часть работы, оставляя основному процессору функции синхронизации и управления. Функции средств расширения специальные, а возможности ограничены. Управление их работой, обеспечение ввода и вывода результатов обработки, программную поддержку обеспечивает основной процессор компьютера.

8.1. Звуковые платы

Звуковые или аудио платы (sound blaster) используются для создания, записи и воспроизведения различных звуковых сигналов: музыки, речи, шумовых эффектов. В режиме создания звука плата действует как музыкальный инструмент. Звук, создаваемый с помощью звуковой платы, называют «синтезированным».

Источниками сигналов являются микрофоны. Магнитофоны, музыкальные инструменты. Они подключаются к компьютеру через звуковые адаптеры и аудиоплаты. Аудиоплата в соответствии со стандартом обработки музыкальной информации преобразуют сигналы в цифровую форму с частотой 44 – 48 КГц и сохраняет их в виде файлов в памяти компьютера.

В режиме воспроизведения звука плата работает аналогично цифровому аудиоплейеру, преобразуя считанные из памяти цифровые сигналы в аналоговые звуковые.

Функционально плата содержит модули записи и воспроизведения звука, модуль синтезатора звука, модуль интерфейсов.

Модуль записи и воспроизведения звука использует для оцифровки звука аналого-цифровые преобразователи (АЦП), а для обратного преобразования — цифро-аналоговые преобразователи (ЦАП). На качество звука и в том и в другом случае существенно влияет количество двоичных разрядов преобразователей и частота дискретизации.

Цифровой сигнал (его двоичный код) записывается в память машины. При воспроизведении оцифрованного звука в ЦАП двоичные коды заменяются дискретными значениями сигнала для последующего их усиления и воспроизведения через акустическую систему.

Модуль синтезатора звука. Для синтеза звукового сигнала используется два основных метода:

- синтез с помощью частотной модуляции, или FM-синтез;
- синтез с использованием таблицы волн (Wave Table), или табличный WT-синтез.

FM-синтез звука осуществляется с использованием специальных генераторов сигналов, называемых операторами. В операторе можно выделить два базовых элемента: фазовый модулятор и генератор огибающей. Фазовый модулятор определяет частоту (высоту) тона, а генератор огибающей — его амплитуду (громкость).

Амплитуда сигнала у разных музыкальных инструментов различна, она сначала возрастает (attack), затем несколько спадает (decay), после чего следует короткий равномерный участок (sustain) и, наконец, происходит спад амплитуды (release). Вышеназванные фазы сигнала реализуются генератором огибающей, который по первым буквам английских терминов этих фаз часто называют генератором ADSR. В общем случае, для воспроизведения голоса одного инструмента достаточно двух операторов:

- первый генерирует колебания несущей частоты, то есть основной тон;
- второй — модулирующую частоту, то есть обертоны.

Современные звуковые платы способны воспроизводить несколько голосов, например, синтезатор с 18 операторами может имитировать 9 разных голосов. Звук, синтезированный FM-методом, имеет необычный оттенок, то есть не похож на звук настоящего музыкального инструмента.

Акустическая система (колонки) является не обязательным, но желательным компонентом мультимедийной системы — при ее использовании восприятие звуковой информации существенно улучшается.

Активные акустические системы оборудованы усилителем и могут подключаться как к линейному выходу звуковой платы, так и к выходу ее усилителя. Источником питания для встроенного в колонки усилителя может являться внутренний аккумулятор или блок питания, который, в свою очередь, может быть и внутренним, и внешним. Кроме регулятора громкости активные колонки имеют обычно и 3-полосный эквалайзер.

8.2. Программные средства звуковых технологий

Программы для работы со звуком можно составляю две большие группы: программы-секвенсоры и программы, ориентированные на цифровые технологии записи звука, так называемые звуковые редакторы.

MIDI-секвенсоры предназначены для создания музыки. С помощью секвенсоров выполняется кодировка музыкальных произведений. Они используются для аранжировки, позволяют формировать отдельные мелодии, назначать тембры инструментов, выстраивать уровни и балансы каналов (треков).

В отличие от обычного сочинения музыки эффективное использование секвенсора требует от композитора специальных инженерных знаний. Программы звуковых редакторов позволяют записывать звук в режиме реального времени на жесткий диск компьютера и преобразовывать его, используя возможности цифровой обработки и объединения различных каналов. Программ-секвенсоров на рынке много: Cakewalk Pro Audio, Cubase VST, Logic Audio Platinum и многие другие.

Программа **Sound Forge** много лет является одним из лидеров среди звуковых редакторов. Она обладает мощными функциями редактирования, позволяет встраивать любые подключаемые модули, имеет удобный современный интерфейс. Включает две дополнительных компонента: Batch Converter, позволяющий объединить группу файлов в один общий файл, и Spectrum Analysis, представляющий данные в двух видах (спектр и фонограмма), используя быстрое преобразование Фурье. Поддерживает современные звуковые форматы, в том числе RealAudio. Среди конкурентов этого звукового редактора можно назвать CoolEdit Pro , WaveLab , PowerTracks Pro.

Программа **Cool Edit Pro -** профессиональная студия звукозаписи фирмы Syntrillium Software. Она позволяет записывать звук через звуковую карту от микрофона, CD-проигрывателя или другого источника, считывать и записывать файлы в популярном формате MP3, редактировать полученные звуковые файлы и добавлять в них разнообразные фантастические эффекты.

Данная прикладная программа позволяет использовать эффекты: реверберatор, эхо, эквалайзер, компрессор, шумоподавление, изменение высоты тона и темпа, анализ спектров и амплитудно-частотных характеристик. Программа обеспечивает работу с мультимедиа-сайтами, подготовку звука для MP3, RealAudio, DVD, позволяет объединять до 64 каналов, создавая файлы объемом до 2 Гбайт и сохраняя их в одном из 25 различных форматов.

Программа **Wave Lab - с**тереоредактор фирмы Steinberg входит в группу лидеров среди звуковых программ-редакторов. Это самый быстрый пакет для редактирования звука. Он обладает множеством эффектов, обеспечивает запись, анализ спектров, имеет возможность работы со встроенными подключаемыми модулями DirectX и VST, поддерживает многие форматы звуковых файлов, в том числе и MP3. Программа открывает звуковой файл в двух окнах: первое — для общего обзора, а второе — для конкретного редактирования. Существует возможность открывать несколько файлов одновременно. Они могут быть сведены в группу и сохранены как проект (project). Большой массив звуковых файлов можно объединить в базу данных (database).

8.3. Системы речевой обработки

Существует две технологии речевого общения с компьютером:

- системы распознавания речи;
- системы синтеза речи.

В системах распознавания речи выполняется оцифровка звуковой информации, ее идентификация с кодами, содержащимися в электронных тезаурусных (иногда многоязычных) словарях, необходимая автоматическая коррекция кодов и генерация соответствующих им символов, слов и предложений, возможный вывод текстов на экран для ручной их коррекции (иногда звуковое воспроизведение) и запись текстов в память машины либо исполнение заданных (услышанных) команд.

По характеру распознаваемой речи системы речевого ввода можно разделить на:

- системы, ориентированные на распознавание отдельных слов, команд и вопросов;
- системы распознавания предложений и связной речи;
- системы идентификации по образцу речи.

Системы, ориентированные на распознавание отдельных слов, команд и вопросов часто называют системами речевого управления, поскольку их основная задача — обеспечить выполнение компьютерной системой действий, задаваемых голосом.

Наибольшее распространение такие системы получили в автоматических телефонных службах (call-центрах). В них можно ввести голосом номер телефона вызываемого абонента или его имя, можно задать простой вопрос автоматической справочной службе.

Наиболее разработаны системы распознавания чисел, которые можно отнести к системам первого поколения. Человек сначала говорит свой числовой пароль, затем свой числовой идентификатор и только после этого может назвать число, кодирующее сущность запроса.

К средствам распознавания второго поколения относятся системы распознавания имен. Основаны эти системы на использовании ключевых слов (имен), хранимых в базе данных системы. Множество хранимых слов и ограничивает возможные имена, распознаваемые команды и вопросы. Например, система Voice

Writer позволяет распознавать около 10 000 слов английского языка, которые после идентификации преобразуются в соответствующие ASCII-последовательности и либо исполняются машиной (если это команды), либо заносятся в файл

Существенно сложнее системы третьего поколения, строящие диалог с пользователем с помощью системы голосовых меню. Такие системы основаны на идее обучения: в течение некоторого времени система обучается на большом количестве типовых речевых диалогов. В ходе этого обучения строится рабочий словарь и база данных отношений между отдельными словами.

Системы распознавания предложений и связной речи. Активный словарь системы насчитывает десятки тысяч слов и может пополняться пользователем по его профессиональной тематике. В системе дополнительно анализируются спектральные (частотные) характеристики каждой буквы, выделяются и хранятся ее отдельные фонемы (элементы спектра). На основе этого анализа создаются фонетические модели букв и формируемых из них слов. Точность распознавания достигает 90 %, а после проверки по словарю еще значительно повышается.

Наиболее сложные проблемы возникают при распознавании связной речи. При произнесении связной речи больше сказывается эмоциональная составляющая вводимой информации, и при слитном произношении слов несколько изменяется их звучание - это затрудняет распознавание.

Системы идентификации по образцу речи. Идентификация по образцу речи относится к биометрическим технологиям идентификации человека по его уникальным физическим признакам, таким как отпечатки пальцев, рисунок радужной оболочки глаз. Речь, подобно подписи, характеризуется множеством постоянных физических параметров. Цель систем по образцу речи — распознать конкретного известного системе пользователя.

Механизм распознавания речи состоит обычно из четырех основных блоков: препроцессора, экстрактора, компаратора, интерпретатора.

Препроцессор или модуль сбора данных обеспечивает приведение речевого сигнала к наиболее качественному виду

(производится автоматическая регулировка усиления, подавление эхо-сигнала, фиксация наличия или отсутствия речи и интонационного конца фразы).

Экстрактор выполняет спектральный анализ сигнала. Поток звуков разбивается на короткие кадры (длительностью примерно по 10 мс) и выявляются спектральные характеристики каждого кадра. Компаратор выполняет акустическое сравнение выявленных характеристик каждого кадра с имеющимися образцами.

Интерпретатор решает задачу наилучшего разбиения полученного потока на слова и фразы.

Системы синтеза речи (речевого вывода информации) базируются либо на выборке из словаря готовых оцифрованных звуковых последовательностей, либо на синтезаторах речи. Самым простым вариантом является выборка готовых звуковых последовательностей (как в автоответчике), но ввиду большого размера звуковых файлов, вывод большого числа слов в этом случае практически невозможен. В таких простых системах часто используются меню, по которым пользователь может выбрать те высказывания, которые он бы хотел услышать. При наличии нужных записей в базе данных их текст озвучивается. Такие системы используются, например, в будильниках, в автомобильных навигационных системах.

Формирование речевого вывода более функционально полными синтезаторами речи выполняется в несколько этапов.

Задачей первого этапа является отфильтровать шумовые символы текста (знаки препинания, кавычки, тире, скобки). На втором этапе модуль преобразования переводит текст из орфографического в фонетический формат (из букв в звуки). Модуль анализа выполняет одновременно лексикографическую и синтаксическую обработку для выбора варианта произношения, ритма и интонации.

Фонетический модуль, получив от модуля анализа фонетическое представление исходного текста, обогащает звучание речи дифтонгами, трифтонгами и другими полезными составляющими.

Модуль обработки звука преобразует фонетические данные в звуковые сигналы: генерируемые волновые последовательности (с частотой порядка 10 кГц) модулируются фонетическим потоком. На

этой стадии выполняется управление громкостью, скоростью речи, тембром голоса.

8.4. Средства обеспечения видеотехнологий

Основная задача компьютерной видеосистемы – представить информацию в привычной для пользователя форме в виде букв, цифр, рисунков. При этом необходимо решить следующие вопросы:

- способ построения изображения;
- эффективный метод кодирования;
- принцип размещения кадра изображения в памяти;
- выбрать механизм подготовки кадров изображения;
- определить методы организации визуальных эффектов (объем, движение).

Разработчиками компьютерного видео были установлены следующие положения.

1. Изображение на экране строится из точек, которые размещаются на экране в виде матрицы.

2. Точка описывается цветом, кадр изображения составляет множество точек разного цвета.

3. Сведения о цвете всех точек экранной матрицы хранятся в специальном блоке памяти (кадровом буфере).

4. Экранные кадры готовятся заранее, выводятся на экран в готовом виде без изменений.

Возможности компьютерного видео на начальном этапе создания видеосистем были очень ограниченными, изображение было статичным с плохим качеством. Прогресс сдерживался малым объемом динамической памяти, отсутствием специальных графических процессоров, недостаточной проработкой вопросов трансформации изображений и описания самих изображаемых объектов.

На рис. 8.1 представлена структура видеосистемы персонального компьютера.

1. **Отображающая панель.** Местом размещения видеоинформации является экран монитора. Широко применяются плоские мониторы с отображающей панелью на жидких кристаллах.

В последнее время началось использование плазменных плоских панелей.

2. **Видеопамять.** Блок специализированной памяти, хранящий в кодированном виде видеокадр отображения. Чаще всего размещается на плате контроллера, иногда для создания видеопамяти выделяется блок из массива оперативной памяти. Сначала проводится запись всего массива, затем считывание всего объема.

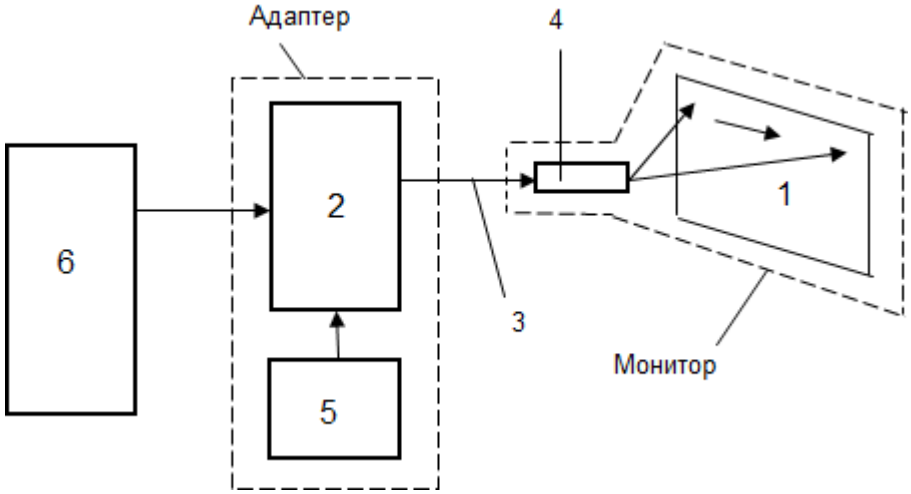


Рис.8.1. Структура видеосистемы персонального компьютера

3. **Видеоинтерфейс**. Информационный канал, по которому данные изображения из видеопамяти передаются на выводящую систему монитора. Интерфейсы бывают аналоговыми (системы VGA и SVGA) и цифровые (EGA, интерфейс DVI в системах SVGA).

4. **Блок вывода.** Обеспечивает последовательный вывод изображения на экран, управляет в плоских мониторах светопроводимостью слоя жидких кристаллов, в плазменных панелях управляет механизмом воздействия на ионизированный газ.

5. **Управляющий контроллер**. Видеоконтроллер может быть выполнен в виде отдельного адаптера или интегрирован в материнскую плату. Адаптер подключается к материнской плате с помощью одной из шин ISA, PCI, AGP, PCI-E (PCI-Express) материнской платы.

Адаптер (видеокарта) является внутрисистемным устройством, преобразующим данные в сигнал, отображаемый монитором. Видеокарта содержит: графический процессор, растровую оперативную память (видеопамять), микросхемы преобразования данных для монитора (рис.8.2).

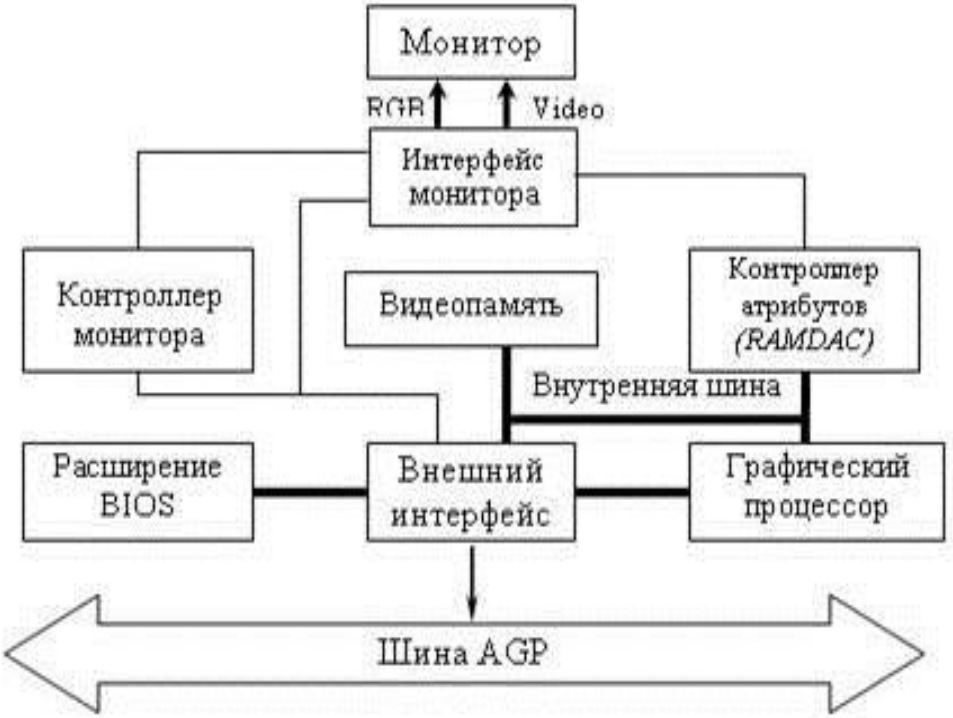


Рис.8.2. Функциональная схема видеокарты

Контроллер монитора (специализированный процессор) формирует управляющие сигналы для монитора и управляет выводом закодированного изображения из видеопамяти, регенерацией ее содержимого, взаимодействием с основным процессором. Контроллер с аппаратной поддержкой некоторых функций, позволяющей освободить центральный процессор от выполнения части типовых операции, называется акселератором (ускорителем). Акселераторы эффективны при работе со сложной графикой: многооконным интерфейсом, трехмерной (3D) графикой.

Основными компонентами специализированного процессора являются: SVGA-ядро, ядро 2D-ускорителя, ядро 3D-ускорителя, видеоядро, контроллер памяти, интерфейс системной шины (внешний интерфейс). Аппаратно большая часть этих компонентов реализуется на одном кристалле видеоконтроллера.

2D-ускоритель — устройство, осуществляющее обработку графики в двух координатах на одной плоскости.

3D-ускоритель — устройство, осуществляющее построение и обработку трехмерных (3D) изображений. В процессе формирования 3D-изображения аппаратный 3D-ускоритель взаимодействует с программным обеспечением. Основные аппаратные элементы 3D-

ускорителя: геометрический процессор, механизм установки и механизм закраски примитивов. Характеристиками ускорителей являются максимальная пропускная способность (треугольников в секунду), максимальная производительность закраски (точек в секунду), скорость (кадров в секунду).

RAMDAC – блок, отвечающий за преобразование двоичных значений цвета и яркости точки в аналоговый сигнал.

Микросхема VGA BIOS отвечает за стандартные функции начальной загрузки компьютера (к ней обращается BIOS системной платы) и ряд специфичных функций, присущих конкретному видеоадаптеру.

Графический процессор предназначен для манипулирования данными, считываемыми или записываемыми процессором и видеопамятью, а также обеспечивает независимость вывода информации на экран от работы с процессором. Для более мощных средств работы с графикой, позволяющих создавать изображения с использованием большого количества цветов, обрабатывать и воспроизводить полноэкранные анимации (последовательность графических изображений, создающая эффект движения объектов на экране), программа разрабатывается с помощью специальных пакетов.

Видеопамять - часть оперативной памяти, в которой хранится электронный образ изображения. Обычно размещается на плате видеоадаптера. Видеоадаптер считывает содержимое видеопамяти и передает на монитор с частотой 56-160 раз в секунду. Каждая ячейка видеопамяти соответствует определенной позиции на экране и любое изменение содержимого видеопамяти вызывает соответствующее изменение на экране.

Важная характеристика — емкость видеопамяти, она определяет количество хранимых в памяти пикселов и их атрибутов. Видеокарта должна обеспечить естественное качественное изображение на экране монитора, что возможно при большом числе воспроизводимых цветовых оттенков, высокой разрешающей способности и высокой скорости вывода изображения на экран.

Под разрешающей способностью здесь (так же как и для мониторов) понимается то количество выводимых на экран монитора

пикселов, которое может обеспечить видеокарта. При разрешении 1024x768 на экран должно выводиться 786 432 пиксела, а при разрешении 2048x1536 выводится 3 145 728 пикселов. Для каждого пиксела должна храниться и его характеристика — атрибут.

Количество воспроизводимых цветовых оттенков (глубина цвета) зависит от числа двоичных разрядов, используемых для представления атрибута каждого пиксела. Выделение четырех битов информации на пиксел (контроллеры CGA) позволяло отображать 24=16 цветов, 8 бит (контроллеры EGA и VGA) — 28 = 256 цветов, 16 бит (стандарт High Color), 24 и 25 бит (стандарт True Color в контроллерах SVGA), соответственно, 16 бит - 65 536, 24бита - 16 777 216 и 25бит- 33 554 432 цвета. В стандарте True Color для отображения каждого пиксела обычно используется 32 бита, из них 24 или 25 для характеристики цветового оттенка, а остальные для служебной информации.

Скорость вывода изображения на экран зависит от скорости обмена данными видеопамяти со специализированным процессором. В качестве видеопамяти могут использоваться различные типы памяти универсальные, специализированные (синхронные графические), двухпортовые типы видеопамяти.

Скорость обмена данными с центральным процессором определяется пропускной способностью шины, через которую осуществляется обмен. В современных компьютерах вместо шины PCI используется более скоростная шина AGP (в частности, AGP 4x).

ЦАП (контроллер атрибутов) RAMDAC преобразует результирующий цифровой поток данных, поступающих от видеопамяти, в уровни интенсивности, подаваемые на соответствующие электронные пушки трубки монитора — красную, зеленую и синюю.

От частоты зависит, какое максимальное разрешение и при какой частоте кадровой развертки монитора сможет поддерживать видеокарта. Разрядность определяет, сколько цветов может поддерживать видеокарта.

Наиболее распространено 8-битное представление характеристики пиксела на каждый цветовой канал монитора (суммарная разрядность 24).

Основные характеристики видеокарты:

- режимы работы (текстовый и графический);
- воспроизведение цветов (монохромный и цветной);
- разрешающая способность (число адресуемых на экране монитора пикселов по горизонтали и вертикали);
- емкость и число страниц в буферной памяти;
- размер матрицы символа (количество пикселов в строке и столбце матрицы, формирующей символ на экране монитора);
- разрядность шины данных, определяющая скорость обмена данными с системной шиной.

Общепринятый стандарт формируют следующие видеокарты:

- EGA — улучшенный графический адаптер (Enhanced Graphics Adapter);
- VGA — видеографический адаптер (Video Graphics Adapter), иногда его называют видеографической матрицей (Video Graphics Array);
- SVGA, VESA SVGA — улучшенный видеографический адаптер (Super VGA);
- PGA — профессиональный графический адаптер (Professional GA).

В настоящее время практически используются видеокарты только типа SVGA. Современные SVGA-видеокарты поддерживают разрешение до 2048 x 1536, число цветовых оттенков более 16,7 млн. (наиболее продвинутые 32-разрядные — более 33 млн.), имеют емкость видеобуфера до 64 Мбайт.

Видеокарта устанавливается на материнской плате в свободный разъем AGP или PCI. Некоторые видеокарты имеют вход для подключения телевизионной антенны (TV-in) и тюнер, то есть позволяют через ПК просматривать телепередачи, видеофильмы с видеомагнитофона и видеокамеры.

Ряд видеокарт имеют разъем для подключения телевизора (TV-out), для просмотра видео.

Видеоадаптеры работают в одном их 2-х режимов.

Текстовый режим. Экран условно разбивается на отдельные участки – знакоместа (прямоугольники 9´16), чаще всего на 25 строк по 80 символов. В каждое знакоместо может быть выведен один из

256 заранее заданных символов, т.е. минимальным объектом на экране является символ ASCII. Содержимое видеопамяти является точным электронным образом изображения на экране. Адаптер 60-70 раз в секунду считывает содержимое видеопамяти и посылает в монитор для вывода на экран. Каждая ячейка видеопамяти соответствует определенной позиции на экране. При работе в текстовом режиме адаптер рассматривает видеопамять как последовательность ячеек на экране. Каждой ячейке соответствуют 2 байта видеопамяти: 1 байт – ASCII-код выводимого символа, 2 байт – код цветового атрибута данного символа (цвет изображения и цвет фона).

Текстовый режим – основной видеорежим. Во время начальной загрузки вывод информации на экран осуществляется в текстовом режиме с разрешением 720´400. Графические элементы создаются с использованием псевдографических символов.

Графический режим. Экран представляет собой матрицу пикселей, а изображение на экране – совокупность пикселей разного цвета. Минимальным объектом является пиксель. Изображение хранится в собственной памяти адаптера. Количество поддерживаемых цветов и разрешение зависят от типа видеосистемы и объема памяти адаптера. Современные адаптеры имеют 512-1024 Мб памяти. Доступ к памяти осуществляется с помощью специальных команд.

8.5. Ускорение графических функций

Стандартная видеокарта персонального компьютера VGA представляет собой набор жесткой логики и видеопамяти. Все, что записывается основным процессором в видеопамять, по определенным алгоритмам преобразуется в аналоговый сигнал, который подается на монитор. Основному процессору необходимо самому рассчитывать параметры всех точек на экране и загрузить все данные в видеопамять.

В связи с развитием технологий мультимедиа и появлением многочисленных двух- и трехмерных видеоигр была усовершенствована аппаратная часть видеокарты, были разработаны устройства, которые получили название графических ускорителей

(графических процессоров). В настоящее время графические процессоры взяли на себя многие функции основных процессоров.

Требование повышения качества изображения привело к тому, что простой графический ускоритель в виде контроллера превратился мощный специализированный графический процессор (GPU – Graphics Processing Unit), мало уступающий по производительности основному процессору.

Так как расчет трехмерных изображений – это множество математических преобразований с плавающей запятой, архитектура графических процессоров стала усложняться, появились дополнительные вычислительные узлы в виде простейших процессоров. Работой этих устройств стал управлять отдельный управляющий процессор, каждый процессор имеет свою локальную память, несколько таких процессоров образуют мультипроцессор МП (рис.8.2).

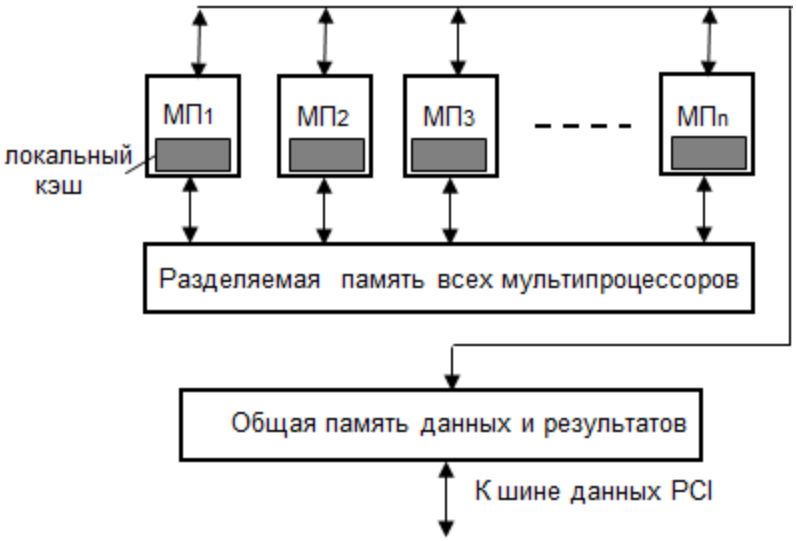


Рис.8.2. Общая архитектура графического процессора

Линейка мультипроцессоров образует графический процессор с разделяемой кэш-памятью и общей памятью данных и результатов, которая выходит на шину PCI компьютера. Все необходимые функции математических преобразований и формирования изображений распределились между графическим GPU-процессором и основным CPU-процессором. В связи с возросшими функциями графических процессоров по скоростной обработке данных, более подробные их характеристики будут представлены далее.

Для видеокарт необходима мощная программная поддержка для превращения инструкции (команд) программиста в изображение на экране монитора, тем более, что написание программ на уровне машинных кодов очень сложно. Появились два стандарта общепринятых интерфейсов графического программирования API (Application Programming Interface): международный открытый стандарт OpenGL (Graphics Library - графическая библиотека) и Microsoft DirectX, привязанный к ОС Windows.

8.6. Сетевые карты

Основные функции сетевых карт. Сетевой адаптер**,** или сетевая интерфейсная карта (Network Interface Card, NIC), вместе со своим драйвером реализует канальный уровень модели OSI в конечном узле сети — компьютере. Обычно сетевые функции распределяются между процессором и адаптером. Это устройство, устанавливаемое в сетевую станцию для организации физической связи с коммуникационным оборудованием сети. Как устройство доступа сетевая карта формирует адрес для сопровождения данных в сеть, передает эти данные и воспринимает адресуемые ему данные из среды передачи. Сетевая карта совместно с драйвером выполняют две операции: передачу и прием кадра.

Передача кадра из компьютера в кабель требует выполнения нескольких этапов.

Прием кадра данных через интерфейс вместе с адресной информацией. Обычно взаимодействие внутри компьютера происходит через буферы, расположенные в оперативной памяти. Данные для передачи в сеть помещаются в эти буферы из дисковой памяти либо из файлового кэша с помощью подсистемы ввода-вывода операционной системы.

Оформление кадра данных, заполнение адресов приемника и источника, вычисление контрольной суммы.

Выдача сигналов в кабель в соответствии с принятым линейными помехозащищенными кодами.

Прием кадра из кабеля в компьютер включает следующие действия.

Прием из кабеля сигналов, кодирующих битовый поток.

Выделение сигналов на фоне шума. Эту операцию могут выполнять различные специализированные микросхемы или процессоры DSP. В результате в приемнике адаптера образуется некоторая битовая последовательность, с большой степенью вероятности совпадающая с той, которая была послана передатчиком.

Проверка контрольной суммы кадра. Если контрольная сумма неверна, то кадр отбрасывается и передается соответствующий код ошибки. Если контрольная сумма верна, то кадр передается в память для дальнейшего использования.

Обычно сетевые карты делятся на карты для клиентских компьютеров и карты для серверов.

В сетевых картах для клиентских компьютеров значительная часть работы перекладывается на драйвер, тем самым адаптер оказывается проще и дешевле. Однако при этом увеличивается степень загрузки основного процессора компьютера рутинными работами по передаче кадров из оперативной памяти компьютера в сеть.

Карты, предназначенные для серверов, обычно снабжаются собственными процессорами, которые самостоятельно выполняют большую часть работы по передаче кадров из оперативной памяти в сеть и обратно.

В зависимости от того, какой протокол реализует сетевая карта, они делятся на карты для технологий Ethernet, Token Ring, FDDI.

В сетевых картах реализована конвейерная схема обработки кадров, то есть процессы приема кадра из оперативной памяти компьютера и передачи его в сеть совмещаются во времени. Таким образом, после приема нескольких первых байтов кадра начинается их передача. Это существенно (на 25-55 %) повышает производительность цепочки оперативная память — карта — физический канал — карта — оперативная память. Адаптеры базируются на специализированных интегральных схемах, что повышает производительность и надежность адаптера при одновременном снижении его стоимости.

Вопросы для контроля

1. Назовите основные модули звуковой карты и их функции.
2. В чем заключаются задачи программ-секвенсоров?
3. Назовите наиболее популярные программы обработки звука.
4. Перечислите основные типы систем речевого ввода.
5. Назовите блоки анализа и распознавания речи.
6. Какие функции выполняют средства видеотехнологий?
7. Опишите основные элементы видеосистемы персонального компьютера и их функции.
8. Их каких аппаратных компонент состоит видеокарта?
9. Перечислите основные характеристики видеокарт?
10. Рассмотрите архитектуру графического процессора.
11. Каковы основные функции сетевых карт компьютеров?

ГЛАВА 9. МИКРОПРОЦЕССОРНЫЕ КОМПЛЕКТЫ МАТЕРИНСКОЙ ПЛАТЫ

В предыдущих разделах рассматривались функции по управлению вычислительным процессом со стороны основного элемента компьютера – процессора. Под процессором понимали полупроводниковый кристалл (чип), в котором размещался как сам процессор, так и его внутрикристальные элементы: регистры общего назначения, кэш-память первого уровня, внутренние шины команд и данных.

Однако, на материнской плате компьютера размещается много других функциональных узлов компьютера: оперативная память, контроллеры портов, шин и доступа, схемы внутренних узлов сопряжения с периферией и
многие другие микросхемы. В современных компьютерах такой набор функциональных узлов носит название микропроцессорной системы (МПС). Многие авторы по традиции называют универсальные процессоры компьютеров микропроцессорами, хотя принципиальной разницы в этом нет. Мы будем применять термин микропроцессор по отношению комплектам дополнительных узлов, обеспечивающих эффективное функционирование процессора.

Микропроцессор (МП) - это программно-управляемое устройство, которое предназначено для обработки цифровой информации и управления процессом этой обработки и выполнено в виде одной или нескольких больших интегральных схем (БИС). Микропроцессор характеризуется большим количеством параметров и свойств, так как он является, с одной стороны, функционально сложным вычислительным устройством, а с другой - электронным прибором, изделием электронной промышленности. Как средство вычислительной техники он характеризуется прежде всего своей **архитектурой**, то есть совокупностью программно-аппаратных свойств, предоставляемых пользователю. Сюда относятся система команд, типы и форматы обрабатываемых данных, режимы адресации, количество и распределение регистров, принципы взаимодействия с оперативной памятью и внешними устройствами (характеристики системы прерываний, прямой доступ к памяти), т.е.

теми же характеристиками, как и процессоры, рассмотренные в предыдущих разделах.

9.1.Структура микропроцессорной системы

МПС - сложная система, включающая в себя большое количество различных устройств. Основой ее является микропроцессор. Связь устройств компьютера между собой осуществляется с помощью сопряжений, которые в вычислительной технике называются интерфейсами.

Интерфейс - это совокупность программных и аппаратных средств, предназначенных для передачи информации между компонентами компьютера, и включающих в себя электронные схемы, линии, шины и сигналы адресов, данных и управления, алгоритмы передачи сигналов и правила интерпретации сигналов устройствами.

В широком смысле интерфейс включает также механическую часть (совместимость по типоразъемам) и вспомогательные схемы, обеспечивающие электрическую совместимость устройств по уровням логических сигналов, входным и выходным токам. Основным способом организации МПС является магистрально-модульный (рис.9.1).

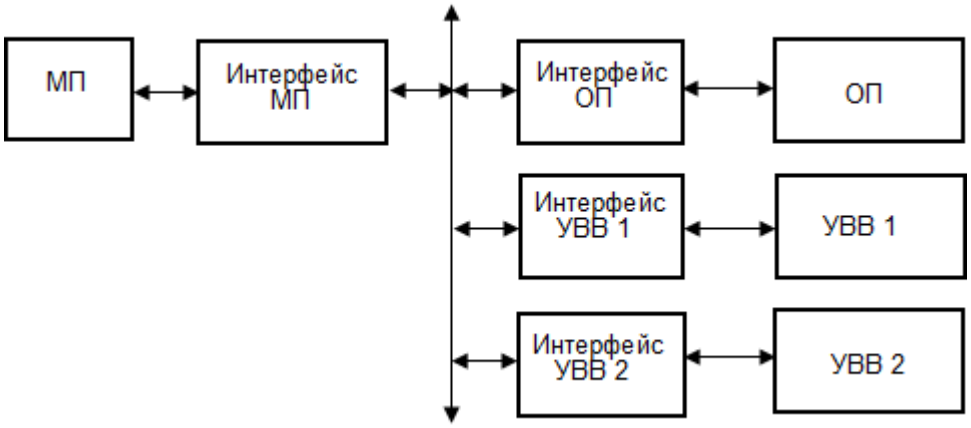


Рис.9.1. Магистрально-модульный принцип построения МПС

Все устройства, включая и микропроцессор, представляются в виде модулей, которые соединяются между собой общей магистралью. Обмен информацией по магистрали удовлетворяет требованиям некоторого общего интерфейса, установленного для

магистрали данного типа. Каждый модуль подключается к магистрали посредством специальных интерфейсных схем.

На интерфейсные схемы модулей возлагаются следующие задачи:

- обеспечение функциональной и электрической совместимости сигналов и протоколов обмена модулей и системной магистрали;
- преобразование внутреннего формата данных модуля в формат данных системной магистрали и обратно;
- обеспечение восприятия единых команд обмена информацией и преобразование их в последовательность внутренних управляющих сигналов.

Эти интерфейсные схемы могут быть достаточно сложными. Обычно они выполняются в виде микросхем специализированных контроллеров. Контроллеры обладают высокой степенью автономности, что позволяет обеспечить параллельную во времени работу периферийных устройств и выполнение программы обработки данных микропроцессором. Кроме того, предварительно буферируя данные, контроллеры обеспечивают пересылку сразу для многих слов, расположенных по подряд идущим адресам.

Возможны два способа организации адресного пространства микропроцессорной системы:

- с общим адресным пространством внешних устройств и оперативной памяти;
- с независимыми адресными пространствами.

В первом случае к портам ввода/вывода можно обращаться как к ячейкам оперативной памяти. Достоинством такого подхода является возможность использовать различные режимы адресации при обращении к внешним устройствам, а также выполнять над содержимым портов ввода/вывода различные арифметико-логические операции. Но при этом сокращается емкость адресуемой оперативной памяти и снижается защищенность системы, так как она лишается дополнительных средств защиты, связанных с выполнением команд ввода/вывода. К тому же нарушение в логике работы программы (формирование неверного адреса оперативной памяти) может привести к ложному срабатыванию внешнего устройства.

Если первый недостаток не столь существенен при современных объемах запоминающих устройств, то второй может весьма негативно сказаться на работе МПС. Возможность использования сложных режимов адресации при обращении к внешним устройствам для микропроцессорных систем на основе универсальных МП не столь важна. Поэтому в настоящее время при построении МПС предпочтение отдается второму подходу.

9.2. Назначение и функции чипсета в микропроцессорной системе

Чипсет (chipset) - это набор БИС (несколько микросхем), функционально эквивалентный схемам, входящим в стандартную конфигурацию микропроцессорной системы. Чипсет проектируют вместе с процессором, с целью максимального использования его возможностей. Основная функция — управление обменом данных через шины между основными устройствами компьютера. Чипсет устанавливается на материнской плате и содержит набор контроллеров, связывающих центральный процессор, память, видеоадаптеры и другие периферийные устройства.

Как правило, чипсет интегрирует в себе функции следующих устройств:

- контроллера оперативной памяти;
- контроллеров кэш-памяти 2-го и/или 3-го уровня;
- контроллеров ПДП;
- контроллеров приоритетных прерываний;
- контроллера клавиатуры;
- контроллера мыши;
- контроллера инфракрасного порта;
- таймера реального времени;
- моста шины PCI;
- моста шины ISA.

Обычно в составе чипсета выделяют:

- северный мост (North Bridge) - системный контроллер, в который входит контроллер системной шины, шин AGP и PCI, ОЗУ и кэш-памяти (для наборов под обычный Pentium);

- южный мост (SOUTh Bridge) - периферийный контроллер, включающий контроллеры EIDE, клавиатуры, моста PCI, последовательных/параллельных портов, шины USB и других подобных устройств;

- Firmware Hub (FWH) – микросхема BIOS.

Все микросхемы связаны между собой высокоскоростной магистралью. Укрупненная структурная схема компьютера с чипсетом приведена на рис. 9.2.

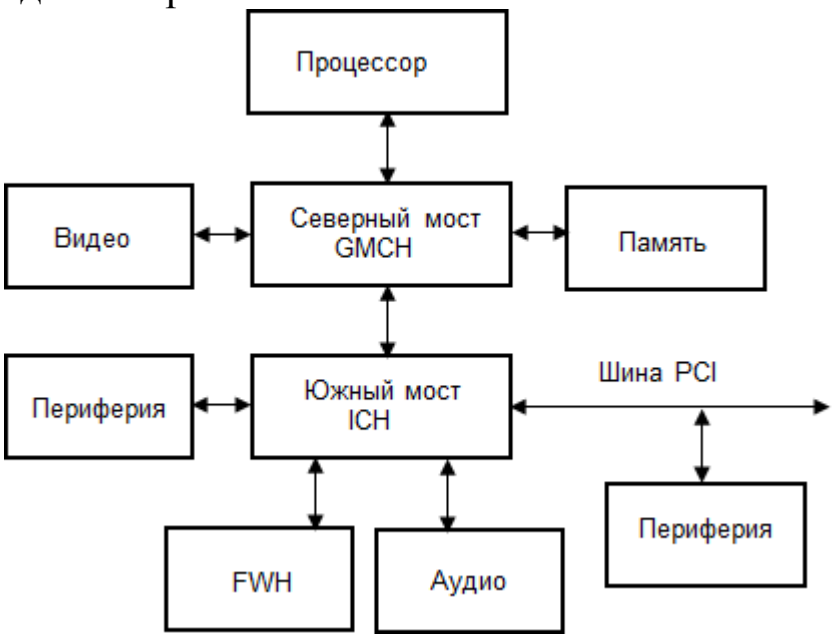


Рис. 9.2. Структурная схема компьютера с чипсетом

Северный мост GMCH (Graphic & Memory Controller Hub) содержит контроллеры FSB, памяти и графики. Южный мост ICH (Input-Output Controller Hub) содержит контроллеры вводы – вывода для периферийных устройств.

Начиная с чипсета Intel-945, в состав «южного моста» включен контроллер специализированной последовательной шины SPI (Serial Peripheral Interface). Он обеспечивает перепись или распаковку BIOS в оперативную память. Для процессоров Pentium Pro, Pentium 2 и Pentium 3 были разработаны наборы чипсет i440 и i810, а для Pentium 4 – наборы i865, i915, i925.

Выбор чипсета во многом определяет конфигурацию МПС и ее производительность. Если МП можно заменить, а емкость ОЗУ увеличить, то замена чипсета однозначно связана с заменой системной платы, а ограничения чипсета также однозначно

ограничивают возможности замены других элементов МПС: МП, ОЗУ, внешних устройств.

Чипсет накладывает ограничения на следующие функциональные характеристики системы в целом: тип памяти, тип кэш-памяти второго и/или третьего уровня, тип МП, максимальную частоту системной шины, тип шины PCI (32/64-разрядная); поддержку многопроцессорной конфигурации и некоторые другие характеристики.

Практика показывает, что разница в производительности системных плат разных фирм, построенных с применением одного и того же чипсета, составляет от силы несколько процентов, между тем как тот же параметр для различных чипсетов может отличаться на порядок. Рассмотрим использование чипсета на примере организации микропроцессорной системы на базе моделей Pentium (рис. 9.3).

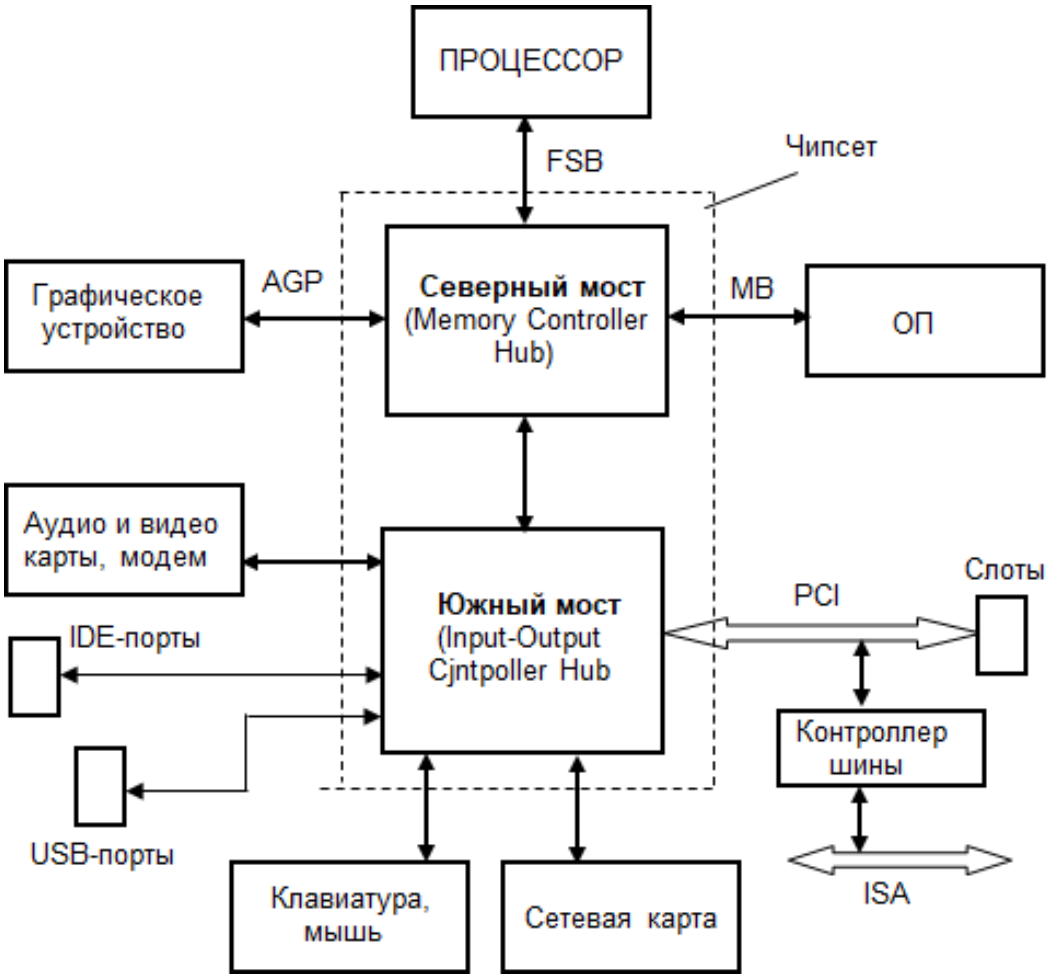


Рис.9.3. Структурная схема компьютера на базе чипсета i-815

В структуре этого компьютера присутствуют все основные, широко применяемые шины и соответствующие интерфейсы, показаны конкретные микросхемы устройств, взаимодействие между которыми осуществляется с помощью данной модели чипсета.

Системная шина, которая в более ранних моделях компьютеров присутствовала в качестве устройства, связывающего основные узлы – процессор, основную память и модули ввода/вывода, здесь уже отсутствует, ее заменят локальные шины и шины ввода/вывода.

Структура типовой материнской платы обычно включает:

- процессор, устанавливаемый в специальный разъем с радиатором и охлаждаемый вентилятором;
- микросхемы кэш-памяти второго уровня. В современных конструкциях эти микросхемы устанавливаются на плату картриджа центрального процессора;
- слоты для установки модулей оперативной памяти;
- слоты для установки карт расширения. Как правило на материнскую карту устанавливаются разъемы для карт стандарта ISA PCI. Современные модели материнских плат оборудованы дополнительным слотом AGP. Наличие слотов и возможность установки в них любых карт расширения (видеоадаптера, звуковой карты, модема, карты АЦП) определяет открытую архитектуру компьютера;
- микросхема программируемой памяти, в которой хранятся программы BIOS, программы тестирования компьютера, загрузки операционной системы, драйверы устройств, начальные установки;
- разъемы для подключения накопителей HDD, FDD.

Все компоненты материнской платы связаны между собой системой проводников (линий), по которым происходит обмен информацией.

Эту совокупность линий называют шинами (Bus).

Поскольку быстродействие различных компонентов (процессора, памяти, адаптеров для шин ISA, EISA, VLB, PCI) существенно различаются, в компьютерах на процессорах класса i486 и старше применяется деление опорной частоты для синхронизации шин ввода/вывода и внутреннее умножение частоты в процессорах.

Различают следующие частоты:

- Host Bus Clock – частота системной шины (внешняя частота шины процессора. Эта частота является опорной для всех других.

- Core Speed – внутренняя частота процессора, на которой работает его внутреннее ядро. Современные технологии позволили существенно повысить предельные частоты интегральных компонентов, в связи с чем широко применяется внутреннее умножение частоты на 1,5, 2, 2,5, 3, 3,5. 4 значения.

- PCI Bus Clock – частота шины PCI, которая должна составлять от 33,3 до 66,6 МГц. Она обеспечивается делением Host Bus Clock на 2 или 3 значения.

- VLB Bus Clock – частота шины VLB, определяемая аналогично PCI Bus Clock. Платы с шиной VLB обычно имеют джампер, переключаемый в зависимости от того, превышает ли системная частота значение 33,3 МГц.

- ISA Bus Clock частота шины ISA, которая должна быть близкой к 8 МГц. Она обычно задается в BIOS Setup через коэффициент деления системной частоты.

Прямое подключение к магистралям различны адаптеров, контроллеров, плат расширения осуществляется через механические соединители, называемые слотами. Современные компьютеры оснащены большим числом магистралей, связывающих различные компоненты чипсетов МПК, например, магистрали процессора FSB (Front Side Bus) и оперативной памяти MB (Memory Bus). В зависимости от места расположения и числа подключаемых устройств магистрали бывают локальные и периферийные. Локальные магистрали предназначены для обслуживания небольшого числа специальных устройств, например, для связи процессора с ОП. Шины периферийной магистрали PSI, ISA(шины расширения ввода-вывода) нужны для информационного обмена с множеством разнообразных внутренних и внешних устройств.

Структурная организация различных магистралей материнских плат имеет много общего. Любая стандартная магистраль содержит шину данных, адресную шину, линии аппаратных прерываний, каналы прямого доступа, проводники для передачи служебной информации и разводки электропитания.

Для магистралей определены способы кодирования сигналов, скорости передачи информации и механизмы арбитража. В общем случае самостоятельно управлять шиной может не только МП, но и устройство, подключенное к ней. На время обмена оно становится ведущим устройством. Такой режим предусмотрен для материнских плат, поддерживающих стандарт Bus Mastering.

Развитие чипсетов Intel В соответствии с политикой фирмы Intel каждые два года продвигается новая архитектура ядер процессоров. Под них разрабатываются новые серии чипсетов, позволяющие наиболее эффективно использовать архитектурные новинки, внедренные в процессоры. Более современные чипсеты Intel 4 Series Express имеют следующие возможности:

- возможность поддержки FSB с частотой 1600 МГц в чипсете X48;
- чипсет P45 поддерживает шину PCI Express спецификации 2.0;
- чипсет G45 обладает наиболее мощным ядром Intel GMA X4500HD;
- все новые северные мосты комплектуются одним и тем же южным мостом ICH10(R);
- наборы микросхем Intel 4 Series Express допускают использование двухканальной памяти DDR3 с частотой 1066 МГц. Таким образом, эта серия чипсетов ориентирована на мощные мультимедийные системы в домашних компьютерах.

Чипсеты модели Intel 5 Series Express предназначены для поддержки новых многоядерных процессоров Core i7/i5/i3, основанных на архитектуре Nehalem.

Архитектура чипсета значительно изменилась по сравнению с серией Intel 4x. Для процессоров на ядре Bloomfield разработан чипсет X58 Express (кодовое название Tylersburg). В дальнейшем для ядра Lynnfield был создан чипсет P55 Express (кодовое название Ibex Peak). В чипсете P55 отказались от северного моста, так как контроллер шины PCI Express был введен в состав процессора, а контроллер памяти еще раньше был помещен в кристалл процессора. Теперь чипсет P55 фактически представляет собой прежний улучшенный южный мост. Вместо традиционной аббревиатуры ICH (I/O Controller Hub) мост P55 носит название PCH — Platform

Controller Hub. Чипсеты Intel серии 6x разработаны для второго поколения процессоров Intel Core i7/i5/i3 архитектуры Sandy Bridge. Первые чипсеты этой серии H61 Express и H67 Express будут иметь встроенное графическое ядро. В последних моделях чипсетов Intel некоторые контроллеры встраиваются непосредственно в кристаллы процессоров, видеокарт, некоторых скоростных портов.

9.3. Дополнительные блоки материнской платы

Наряду с электронными блоками, входящими в общий комплект микропроцессора, на материнской плате размещаются чипы, с помощью которых расширяется круг решаемых задач.

Постоянное запоминающее устройство и его контроллер. Постоянная память (ROM – Read Only Memory) предназначена для хранения следующих программ: загрузки системы BIOS (Basic Input Output System), начального тестирования - POST (Power On-Self-Test), сохранения и изменения информации в энергонезависимой памяти - Setup.

Оперативное запоминающее устройство – система с произвольным доступом (RAM – Random Access Memory), которая включает контроллер оперативной памяти и средства подключения модулей. Задача ОЗУ – размещение набора предназначенных для исполнения программ и обрабатываемых данных. Блок ОЗУ выполняется в виде отдельных модулей на базе динамических методов хранения. Блок ОЗУ напрямую связан с процессором, он может выполнять только программы, находящиеся в ОЗУ. На дисках и других накопителях программы могут только сохраняться, перед исполнением они должны быть переписаны в ОЗУ.

RTC (Real Time Clock) – часы реального времени. Они продолжают функционировать и после выключения компьютера. Электропитание обеспечивается той же батарейкой, что и энергонезависимая память.

Блок обработки прерываний. Реализует механизм прерывания исполнения текущей программы по сигналу от процессора или подключенных периферийных устройств. Сигналы могут поступать от более чем десяти инициативных устройств.

Блок прямого доступа в память (ПДП). Обеспечивает блочную передачу данных объемом в несколько десятков килобайт миную процессор.

Таймер. Предназначен для отсчета временных интервалов и передачей по специальным линиям управляющих сигналов управляющих сигналов. Таймер располагает несколькими рабочими каналами. Каждый из них может формировать синхросерии сигналов, подавать одиночные импульсы.

Тактовый генератор. Все действия устройств на материнской плате должны выполняться в нужной последовательности, время работы должно измеряться в одинаковых интервалах в соответствии с тактовой частотой. Формированием частоты синхронизации занимается тактовый генератор.

Вопросы для контроля

1. Дайте определение интерфейса микропроцессорной системы.
2. Каковы задачи интерфейсных схем?
3. Что понимается под словом «чипсет»?
4. Какие контроллеры входят в состав чипсета?
5. Проанализируйте функции «северного» и «южного» мостов чипсета?
6. Перечислите аппаратные компоненты материнской платы.
7. Какие дополнительные блоки могут входить в комплекты материнской платы компьютера?
8. Какие преимущества имеют чипсеты фирмы Intel последних моделей?

ГЛАВА 10. АРХИТЕКТУРА БАЗОВОЙ МОДЕЛИ ПРОЦЕССОРОВ INTEL

Под архитектурой компьютера обычно понимается ее представление и описание возможностей с точки зрения пользователя, изучающего средства вычислительной техники, использующего компьютеры в своей практической деятельности и разрабатывающего прикладные программы решения задач. Архитектура, как правило, отображает те аспекты структуры и принципов функционирования компьютеров, которые являются видимыми для пользователя и, следовательно, понятными при разработке им программ.

Основные архитектурные особенности компьютеров, рассмотренные в предыдущих разделах, отражают в значительной степени количественные и качественные характеристики процессоров марки **Intel.**

В данном разделе под пользователем понимается программист, разрабатывающий программы на языке ассемблера. В связи с принятым во всем мире делением программистов на прикладных и системных, архитектуру программного обеспечения также можно разделить на два уровня: прикладной и системный.

К основным элементам рассматриваемой далее **прикладной архитектуры** компьютера, как правило, относятся:

- типы и форматы представления данных, аппаратно поддерживаемые в компьютере;
- регистровая организация структуры процессора;
- адресная структура основной памяти и принципы размещения данных;
- режимы адресации;
- система, структуры и форматы машинных команд;
- система прерывания.

В качестве наглядного примера реализации прикладной архитектуры выбрана архитектура наиболее популярных и широко используемых в нашей стране процессоров компании **Intel.** Из вышеперечисленных элементов прикладной архитектуры будут рассмотрены регистровая организация и структура команд, т.к.

остальные элементы были рассмотрены ранее или мало отличаются от таких элементов других типов процессоров.

10.1. История развития архитектуры процессоров Intel

Микропроцессоры с архитектурой i86 - это своего рода феномен второй половины XX века. С 1978 года, когда фирма Intel выпустила первый микропроцессор Intel-8086 и затем его упрощенный вариант 8088, сменилось восемь поколений таких устройств. В настоящее время практически во всех областях промышленности и компьютерной сере доминируют процессоры, в той или иной мере совместимы с архитектурой i86.

Самая первая IBM PC была построена на базе микропроцессора 8088, имела 64-Кбайт ОЗУ и была оснащена НГМД для односторонних дисков емкостью 160 Кбайт. Продажа IBM PC началась в октябре 1981 г., а уже к концу этого же года было продано более 35 тыс. машин. Этот процессор стал "мозгом" первого персонального компьютера "Альтаир",названного по имени звезды, к которой был запущен межпланетный корабль из телесериала "Космическая одиссея".

Принятие корпорацией IBM на вооружение архитектуры 8086/88 при разработке первого персонального компьютера резко взвинтило ценность этого наименования, сохраненного последующими процессорами в виде 5-значного обозначения: 80286, 80386 и, наконец, 80486.

Появившийся в 1982 году 286-й, известный также под наименованием 80286, стал первым процессорои Intel, способным выполнять любые программы, написанные для его предшественников. С тех пор такая программная совместимость остается отличительным признаком семейства процессоров Intel. Спустя 6 лет с момента выпуска 286-го, количество персональных компьютеров на базе этого процессора оценивалось в 15 миллионов по всему миру.

Микропроцессор Intel 386, вышедший в свет в 1985 году, насчитывал уже 275 тыс. транзисторов, число которых, по сравнению с первыми моделями увеличилось более чем в 100 раз. Это был 32-разрядный "многозадачный" процессор с возможностью одновременного выполнения нескольких программ.

В 1989 году бал разработан микропроцессор Intel 486 DX. Поколение процессоров 486 ознаменовало переход от работы на компьютере через командную строку к режиму "укажи и щелкни". Intel 486 стал первым микропроцессором со встроенным математическим сопроцессором, который существенно ускорил обработку данных, выполняя сложные математические действия вместо центрального процессора.

Именно на переходе от 80-х к 90-м г. сформировался альянс Wintel. Когда Intel выпустила микропроцессор 486, производители компьютеров не стали дожидаться примера со стороны IBM или Compaq. Началась гонка, в которую вступили десятки фирм. Но все новые компьютеры были очень похожи друг на друга, т.к. их роднила совместимость с Windows и процессоры от Intel.

Микропроцессор Pentium.

Следующим шагом в развитии микропроцессорной техники было появление 586-го процессора **Pentium** в 1993 году. Pentium при полной совместимости с предыдущими моделями имел производительность свыше 100 млн. операций в секунду, два отдельных кэш-устройства по 8 Кбайт для команд и данных, что позволило снизить число обращений к внешней памяти компьютера и увеличить производительность системы. Разрядность шины увеличилась до 64 разрядов, что позволило за один такт передавать вдвое больше информации.

Блок предсказаний ветвлений позволил предсказывать ход выполнения программы. Два параллельных конвейера позволили выполнять одновременно две команды. В математическом сопроцессоре аппаратно были реализованы умножение, деление и сложение, благодаря чему большинство операций с плавающей запятой выполнялись за один акт.

Pentium стал работать в 2 раза быстрее, а на задачах с плавающей арифметикой – в 5 раз быстрее, чем 486-ой процессор. Процессор Pentium научил компьютеры работать с атрибутами "реального мира" — такими, как звук, голосовая и письменная речь, фотоизображения.

В 1997 появился усовершенствованный процессор модели **Pentium II**, имеющий 7,5 миллионов транзисторов. Процессор

Pentium II использует технологию Intel, обеспечивающую эффективную обработку аудио, визуальных и графических данных. Кристалл высокоскоростной кэш-памяти был помещен в корпус с односторонним контактом, который на системной плате устанавливается с помощью одностороннего разъема — в отличие от прежних процессоров, имевших множество контактов.

Процессор дал пользователям возможность вводить в компьютер и обрабатывать цифровые фотоизображения, пересылать их через Internet, создавать и редактировать тексты, музыкальные произведения и даже сценки для домашнего кино, передавать видеоизображения по обычным телефонным линиям и по Internet.

Позже появились модели процессоров Pentium II Xeon (аналогичный по характеристикам Pentium II) и Intel Celeron, упрощенный и более дешевый вариант процессора.

В 1999 году появился процессор Pentium III, имеющий 9.5 миллионов транзисторов. Его основными характеристиками являлись 512 Кб кэш-памяти, картридж с односторонним контактом в основе корпуса, системная шина с частотой 100 МГц и разрядностью 64 бит. Процессор обладал адресуемой памятью 64 Гбайт. Pentium III применялся как для офисных работ, так и для домашних компьютеров, одно- и двухпроцессорных серверов и рабочих станций.

Процессор Pentium 4 был разработан в 2000 году. Пользователи персональных компьютеров на базе процессора Pentium 4 могли создавать профессионально оформленные видеофильмы; смотреть видео телевизионного качества через Internet; общаться друг с другом с передачей речи и изображения; воспроизводить трехмерную графику в режиме реального времени; быстро оцифровывать музыку для mp3-плееров; одновременно запускать несколько мультимедийных приложений при активном соединении с Internet. Процессоры этого поколения содержат 42 млн. транзисторов, а ширина проводников составляет всего 0,18 микрон. Частота системной шины достигла 400 МГц.

В 2001 году вышел процессор **Intel Itanium** - новый 64-разрядный процессор Intel, который позволяет предприятиям вывести свои компьютеры на новый уровень производительности, функциональности и надежности. В его основе лежит новая

архитектура EPIC (Explicitly Parallel Instruction Computing - обработка команд с явным параллелизмом). Благодаря усилиям сотен компаний, разрабатывающих системы и приложения для этого процессора, Intel Itanium обеспечил значительный прогресс в наиболее требовательных к вычислительным ресурсам областях применения компьютеров.

Процессор Intel Core 2 - восьмое выпущенное корпорацией Intel поколение процессоров архитектуры i86, основанное на совершенно новой процессорной архитектуре, которая называется Intel Core. Это потомок микроархитектуры Intel P6 на которой, построено большинство процессоров

Intel. На рынок поступили процессоры Pentium Dual-Core и Core Celeron.

Первые процессоры Core 2 официально представлены в июле 2006 года. Также как и их предшедственники Intel Core, они делятся на модели Solo (одноядерные), Duo (двухъядерные), Quad (четырёхъядерные) и Extreme (двух- или четырёхъядерные с повышенной частотой и разблокированным множителем).

В отличие от микропроцессоров архитектуры Pentium 4, в архитектуре Core 2 ставка делается не на повышение тактовой частоты, а на улучшение других параметров процессоров, таких как кэш, эффективность и количество ядер. Рассеиваемая мощность этих микропроцессоров значительно ниже, чем у настольной линейки Pentium.

Четырехъядерный МП Intel Core2 Quad обеспечивает высочайшую скорость выполнения ресурсоемких задач в многозадачных средах и максимальную производительность многопоточных приложений. Этот процессор стал идеальным решением для развлекательных приложений. Кодирование, рендеринг, редактирование и потоковая передача – далеко не все возможности мультимедийных приложений профессионального уровня с персональными компьютерами на базе процессора Intel Core2 Quad.

Четыре ядра процессора, до 8 МБ общей кэш-памяти 2 уровня и системная шина с частотой 1066 МГц обеспечивают эффективность и производительность многозадачной нагрузки, что позволяет

пользователям превратить свой компьютер в мощное средства параллельной обработки сигналов и изображений.

Настольные платформы на базе **процессоров Intel Core 2 Duo** открывают новый уровень производительности, быстродействия и энергосбережения. Арсенал новейших технологий, обеспечивающих повышение производительности, до 4 МБ общей кэш-памяти 2 уровня и частота системной шины до 1066 МГц открывают пользователю дорогу в будущее компьютерной техники..

Настоящая многозадачность этих процессоров заключается в том, что у пользователя появилась возможность выполнять больше задач параллельно. Это задачи цифровой обработки сложных (например, речевых) сигналов и динамических изображений, распознавания объектов и сцен, выполнение мультимедийных приложений.

Микророцессор Intel Core i3 — еще одно семейство процессоров начального и среднего уровня цены и производительности. В новом модельном ряду призваны заменить устаревшие Pentium Dual-Core на архитектуре Intel Core 2. По уровню производительности стоят на самой низкой ступени, перед более дорогими и производительными Core i5.

Они имеют встроенный контроллер памяти и поддерживают технологию Turbo Boost (автоматический разгон процессора под нагрузкой). Имеют встроенный графический процессор.

Первые Core i3 представлены в январе 2010 года и используют ядро Clarkdale. Обладают встроенным графическим процессором (в корпусе процессора, но на отдельном кристалле). Core i3 имеют: 2 ядра (4 потока), L2-кэш 256 Кбайт на ядро, L3-кэш 4 МБ; двухканальный чип памяти.

Микропроцессор Intel Core i5 позиционируется как семейство процессоров среднего уровня цены и производительности, между более дешёвым Intel Core i3 и более дорогим Core i7. Они имеют встроенный контроллер памяти и поддерживают технологию Turbo Boost.

Микропроцессоры Intel Core i7 — первое семейство, использующее микроархитектуру Intel Nehalem. Он также является

преемником семейства Intel Core 2. Все три модели микропроцессоров являются четырехъядерными.

С момента появления первого процессора архитектуры i86 различные фирмы во всем мире занимаются совершенствованием совместимых устройств. Такие процессоры, как правило, полностью копируют архитектуру приборов Intel и совместимы с их программным обеспечением. Процессоры-клоны существуют для всех устройств серии i86. Наиболее известны процессоры-клоны: Cyrix - IBM, K6 - AMD, Winchip - IDT и некоторые другие. Специфика этих процессоров состоит в том, что они предназначены для применения в компьютерах средней производительности и не претендуют на лидерство в сфере мощных многопроцессорных сверхвысокопроизводительных систем.

10.2. Регистровая сруктура процессора

Регистровая структура процессора включает в себя 14 программно-доступных 16-ти разрядных регистров: регистров общего назначения (РОН), сегментных регистров, регистра флагов и указателя флагов.

Регистры общего назначения (РОН) (рис.10.1):

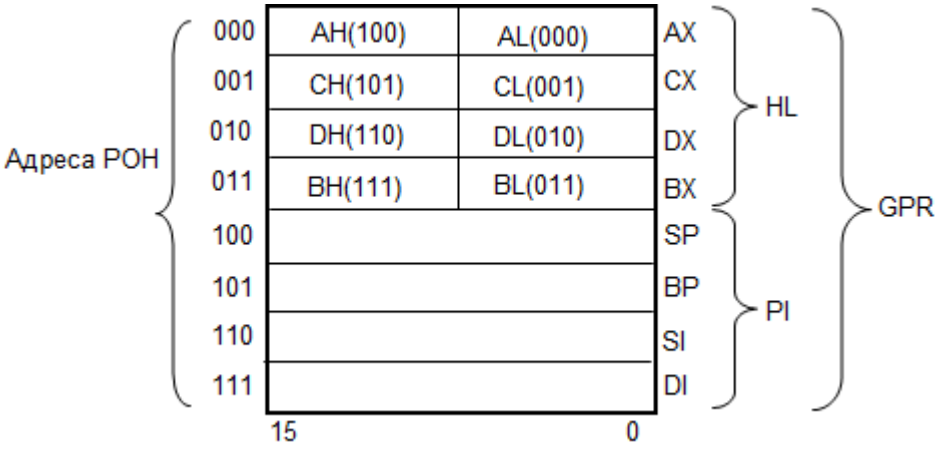


Рис.10.1. Схема регистров общего назначения

В процессорах фирмы Intel используется подход, сочетающий в себе частичную специализацию и частичную универсализацию регистров. Это означает, что по умолчанию любой регистр используется и как специализированный и как универсальный регистр.

Использование любого регистра по его прямому назначению сокращает длину объектного кода программы по сравнению с любым другим использованием регистра, так как использование регистра по назначению, как правило, предполагает его неявную адресацию (адрес регистра не задается, но подразумевается).

Функциональная специализация РОН отражается в их наименованиях:

AX – Accumulator (регистр-аккумулятор) – по умолчанию используется для задания одного из операндов команды и для представления результата.

CX – Counter (счетчик) – по умолчанию используется, во-первых, как счетчик числа повторения циклов в команде "организация цикла" (LOOP); во- вторых, для задания числа сдвигов в командах сдвигов (его младший байт –CL); в-третьих, для задания числа элементов обрабатываемых строк (цепочек) в командах обработки строк (MOVS, CMPS).

DX – Data (регистр данных) – используется как расширение аккумулятора со стороны старших разрядов в командах умножения и деления.

BX – Base (базовый регистр) – используется как базовая компонента эффективного адреса операнда, находящегося в памяти. В терминологии фирмы Intel под эффективным адресом – Effective Address (EA) – понимается адрес операнда, формируемый программой (программный адрес). Для получения физического адреса ячейки памяти, в которой находится операнд, осуществляется преобразование EA на основе простейшей модели сегментированной памяти (механизма сегментации).

SP – Stack Pointer (указатель стека) – используется для адресации вершины стека. Вершина стека указывает на адрес последнего элемента, записанного в стек.

BP – Base Pointer (указатель базы) – по умолчанию используется как базовая компонента эффективного адреса операнда в памяти по аналогии с BX.

SI – Source Index (индекс источника) – по умолчанию используется для задания индексной компоненты EA, а также для адресации элементов строки источника в командах обработки строк.

DI – Distination Index (индекс приемника) – по умолчанию используется аналогично SI для задания индексной компоненты EA, а также для адресации элементов строки-приемника в командах обработки строк.

Группу из восьми РОН принято делить на две части:

- группа HL (High – Low);
- группа PI (Pointer – Index).

Группу HL иногда называют регистрами данных, подразумевая ее преимущественное использование для операндов и результатов команд. Регистры группы HL могут использоваться в командах в двухбайтном (полном) и в байтном (неполном) варианте.

Группа PI или группа указателей-индексов может использоваться только в двухбайтном варианте.

Для адресации РОН, как полных, так и не полных, в машинной команде используются три двоичных разряда. Двоичные адреса полных РОН приведены на рис.10.1 слева, а их отдельных байт - в скобках.

Сегментные регистры. Сегментные регистры используются для аппаратной поддержки простейшей модели сегментированной памяти, принятой в процессоре 8086. Их содержимое определяет базовые (начальные) адреса соответствующих сегментов в физической памяти (рис.10.2).

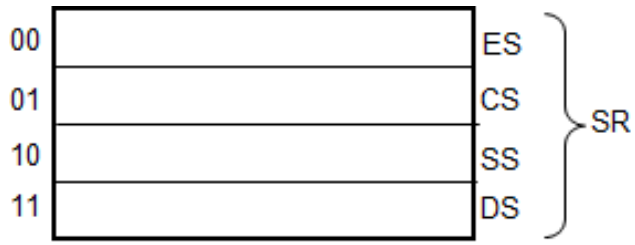


Рис.10.2. Схема сегментных регистров процессора

В состав сегментных входят следующие регистры:

CS – Code Segment (сегмент кода – машинной программы),

SS – Stack Segment (сегмент стека),

DS – Data Segment (сегмент данных),

ES – Extra Segment (дополнительный сегмент).

Использование четырех сегментных регистров предполагает, что в любой момент времени программа может работать (иметь доступ) с четырьмя сегментами памяти.

С учетом того, что длина каждого сегмента составляет 216 байт = 64 Кбайт (16-разрядный адрес) физическое адресное пространство, доступное программе, составляет 256 Кбайт.

Шина адреса процессора Intel 8086 является 20-разрядной, что обеспечивает емкость адресного пространства 220 байт - 1 Мбайт. При формировании 20-разрядного физического адреса, содержимое соответствующего сегментного регистра смещается в сторону старших разрядов путем сдвига на четыре разряда влево. Таким образом, содержимое сегментных регистров представляет собой не сам физический начальный адрес сегмента, а его значение, уменьшенное на 16 (сегменты выровнены в памяти на границу параграфа).

Содержимое одного из сегментных регистров привлекается по умолчанию каждый раз при обращении процессора к памяти для формирования физического адреса, который и выставляется на шину адреса. Например, на этапе выборки команды по умолчанию привлекается регистр CS, при обращении к стеку – регистр SS. При обращении за операндом или при записи результата –регистр DS.

Регистр флагов и регистр управления IP.

В регистре флагов актуальными являются только девять битов, шесть из которых представляют собой арифметические флаги (флаги состояний) и три –флаги управления (рис.10.3.). Остальные семь бит незадействованы.

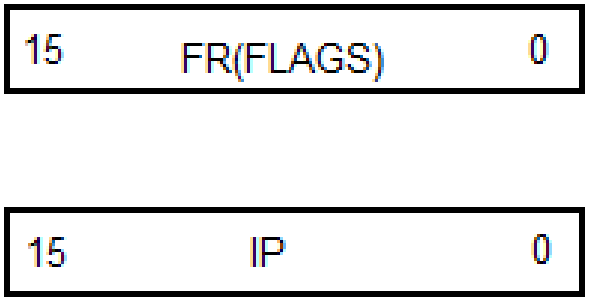


Рис.10.3. Регистр флагов и регистр управления

Арифметические флаги формируются в основном арифметическими командами, определяя результат арифметических операций (они являются признаками результата). Кроме того, на арифметические флаги оказывают влияние логические команды и команды сдвигов. Флаги управления оказывают влияние на процесс выполнения программы.

К арифметическим флагам относятся:

CF – Carry Flag (флаг переноса – 0 разряд регистра). В нем фиксируется перенос из старшего разряда при сложении и заем в старший разряд при вычитании. При умножении значения флага CF определяет возможность (CF=0) или невозможность (CF=1) представления результата (произведения) в том же формате, что и сомножители. С помощью флага переноса фиксируется переполнение при сложении беззнаковых чисел.

PF – Parity Flag (флаг четности – 2 разряд регистра). Он устанавливается при наличии четного числа единиц в младшем байте результата, в противном случае – сбрасывается. Этот флаг используется как аппаратная поддержка контроля по четности (нечетности).

AF – Auxiliary Carry Flag (флаг вспомогательного переноса – 4 разряд). Флаг фиксирует межтетрадный перенос при сложении и межтетрадный заем при вычитании. Значение этого флага используется командами десятичной и ASCII- коррекции сложения и вычитания. Этот флаг можно рассматривать как аппаратную поддержку операций над десятичными числами.

ZF – Zero Flag (флаг нуля – 6 разряд). Он устанавливается при нулевом результате операции, в противном случае сбрасывается.

SF – Sign Flag (флаг знака -7 разряд). В него копируется старший (крайний левый) бит результата, интерпретируемый как знак.

OF – Overflow Flag (флаг переполнения -11 разряд). Он устанавливается в командах сложения и вычитания в случае, если результат операции не помещается в формате операндов. При этом как операнды, так и результат интерпретируются как знаковые целые числа.

В аппаратную установку этого флага положен принцип фиксации переполнения по сравнению переносов из двух старших разрядов при сложении или заемов в два старших разряда при вычитании. Если один из переносов (заемов) имеет место, а другой отсутствует, то происходит переполнение формата (разрядной сетки). В командах умножения флаг OF выполняет ту же функцию, что и флаг CF (их значения совпадают).

К **флагам управления IP (**Instruction Pointer**)** относятся:

TF – Trace (Trap) Flag (флаг трассировки – 8 разряд). При установке флага TF процессор переводится в так называемый отладочный (пошаговый) режим работы. В этом режиме завершение выполнения любой команды сопровождается выходом на прерывание специального типа (стандартный тип 1 – прерывание по отладке).

DF – Direction Flag (флаг направления -10 разряд). Его значение используется командами обработки строк (цепочек) и определяет направление обработки: от меньших адресов к большим (слева на право) при DF=0 или от больших адресов к меньшим (справа на лево) при DF=1.

IF – Interrupt Flag (флаг прерываний – 9 разряд). С помощью этого флага разрешаются (IF=1) или запрещаются (IF=0) внешние прерывания, запросы которых реализуются с помощью специального контроллера прерываний (Programmable Interrupt Controller).

Содержимым **регистра IP** является так называемый продвинутый адрес команды. Это означает, что в момент выполнения какой-либо команды регистр IP указывает на адрес следующей команды. В качестве адреса команды фигурирует адрес ее младшего байта (по адресации). В связи с тем, что в базовой модели используется простейший двухступенчатый конвейер команд (I ступень – выборка команды; II ступень - остальные фазы (этапы) выполнения команды, к которым относятся: декодирование команды (определение типа), формирование адресов операндов, выборка операндов, выполнение операции в АЛУ, запись результата. Фактическое содержимое регистра IP, который входит в блок предварительной выборки команд, указывает на очередной байт команды, выбираемый из памяти и помещаемый в специальный буфер, называемый очередью команд (IQ – Instruction Queue).

Несмотря на этот факт, для выполняемой программы IP содержит адрес следующей команды. Фактически, на аппаратном уровне при необходимости использования IP (например, для его сохранения как адреса возврата) осуществляется соответствующая коррекция его содержимого с учетом числа байт, выбранных в IQ.

Конвейер команд служит одним из важнейших средств увеличения производительности компьютера. С его помощью

реализуется параллелизм на уровне машинных команд. Это означает, что в любой момент времени в процессоре на стадии одновременного выполнения находится несколько последовательных машинных команд (по возрастанию адресов). Для аппаратной реализации конвейера команд отдельные фазы команды реализуются с помощью отдельных блоков конвейера.

Блоки конвейера могут функционировать параллельно во времени независимо друг от друга. При использовании классического шестиступенчатого конвейера команд (по числу фаз выполнения команды) и условии, что каждая фаза требует одинакового времени для реализации, полная загрузка конвейера команд в принципе обеспечивает шестикратное увеличение производительности по сравнению с последовательным (бесконвейерным) процессором.

10.3. Базовая система команд процессоров Intel

Система команд включает в себя более 100 различных мнемокодов команд (мнемокод - буквенное кодирование операций). С учетом разнообразных форматов команд и режимов адресации число команд более 380. По функциональному назначению команды можно разделить на следующие типы:

- команды передачи данных и адресов;
- арифметические команды;
- логические команды;
- команды сдвигов;
- команды управления программой (переходы, вызовы, возвраты и т.д.);
- команды обработки строк (цепочек);
- команды управления процессором.

10.3.1 Команды передачи данных и адресов

Классификация команд передачи данных и адресов приведена на рис.10.4.

Общие передачи данных.

MOV (MOVe data) – пересылка данных. Команда передает значение операнда-источника в операнд-приемник. Команда MOV не

может пересылать данные из одной области памяти в другую, для таких пересылок может быть использована команда MOVS.

В ассемблерной нотации команда представляется в виде: MOV dst, src.

Рис.10.4.Команды передачи данных и адресов

Содержимое src пересылается на место dst (dst – приемник, src – источник).

dst := r, mem, sr (r –регистр, mem – память, sr – сегментный регистр),

src = r, mem, sr, imm (imm – непосредственный операнд).

Примеры*:*

MOV AX,BX – передача содержимого регистра BX в регистр AX;

MOV AL,AH – передача содержимого регистра AH в регистр AL.

Команда не влияет на арифметические флаги.

XCHG (eXHanGe register/memory with register) – обмен данными.

Команда производит взаимный обмен значениями между операндом- источником и операндом-приемником. Ассемблерная нотация команды: XCHG op1, op2. В качестве операндов могут использоваться РОН и память. Оба операнда не могут быть операндами из памяти.

Пример: XCHG AX,BX - меняется местами содержимое регистров AX и BX.

Значения флагов не меняются.

Стековые передачи.

PUSH (PUSH operand onto stack) – загрузка операнда в стек.

Ассемблерная нотация команды: PUSH src. Выполняет декремент указателя стека SP (Stack Pointer) на два и затем заносит слово операнда-источника в вершину стека, адресуемую SP. Операндом может быть регистр общего назначения, сегментный регистр, слово памяти. Команда часто используется для передачи (через стек) параметров вызываемой процедуре или для сохранения значений временных переменных.

Пример: PUSH AX – поместить содержимое регистра AX в стек.

Значения флагов не меняются.

POP – (POP operand from the stack) – восстановление операнда из стека.

Ассемблерная нотация команды: POP dst. Команда восстанавливает в операнде-приемнике слово, хранящееся в вершине стека и производит инкремент указателя стека (SP) на 2. В качестве операнда команды может использоваться РОН, сегментный регистр, слово памяти. Т.к. стек работает со словами, а не с байтами, операнд должен быть 16-разрядным.

Пример: POP AX – извлечь содержимое из вершины стека в регистр AX.

Значения флагов не меняются.

Флажковые передачи.

Все команды флажковых передач являются безадресными, так как используют неявную адресацию операндов.

PUSHF (PUSH Flags register onto stack) – это запись в стек содержимого регистра флагов. Команда PUSHF производит декремент указателя стека на 2 и копирует регистр FLAGS в новую вершину стека.

Значения флагов не меняются.

POPF (POP Flags register from the stack) – извлечение из стека в регистр флагов. Команда POPF извлекает слово из вершины стека и сохраняет его значение в регистре флагов, далее производится инкремент указателя стека (SP) на два.

Меняются значения всех флагов (OF, DF, IF, TF, SF, AF, ZF, PF, CF)

LAHF (Load Flags into AH register) – это загрузка флагов в регистр AH. Команда LAHF передает младший байт регистра флагов в регистр AH. При этом в регистр AH загружаются все арифметические флаги, кроме OF, который размещается в старшем байте регистра флагов.

SAHF (Store AH into Flags) – это сохранить содержимое AH в регистре флагов. Команда SAHF записывает биты регистра AH в регистр флагов. При этом значения битов 0, 2, 4, 6, 7 регистра AH становятся соответственно значениями флагов: переноса (CF), четности(PF), дополнительного переноса (AF), нуля (ZF) и знака (SF).

Табличное преобразование.

XLAT (table look up transLATion) – табличное преобразование.

Команда XLAT - безадресная команда, т.к. использует неявную адресацию операндов. Действие команды состоит в следующем (рис.10.5).

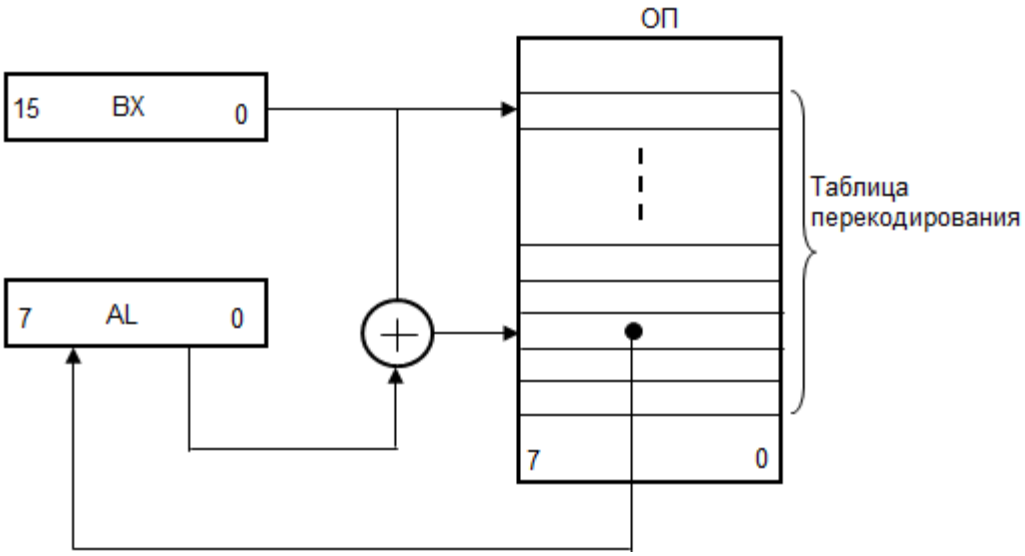


Рис.10.5. Графическое представление действия команды XLAT

К содержимому регистра BX прибавляется содержимое регистра AL, и эта сумма далее используется для обращения к сегменту данных как внутрисегментное смещение. Из сегмента данных выбирается байт, который помещается в регистр AL. Эту команду используют для преобразования символьных данных из одного кода в другой. Для этой цели в памяти, точнее в сегменте данных, предварительно формируется таблица перекодирования, начальный адрес которой загружается в регистр BX. В AL загружается перекодируемый символ. Значение этого символа используется как смещение

относительно начального адреса из BX в таблице перекодирования. По этому адресу реализуется выборка символа в новом коде, загружаемого в AL.

Значения флагов не меняются.

Т.к. в одном байте перекодируемого символа может быть представлено 256 комбинаций, то наибольшая длина таблицы перекодирования составляет 256 байт. Однократное применение команды XLAT перекодирует лишь один символ. Для перекодирования строки символов необходимо организовать программный цикл.

Команды ввода-вывода.

По команде IN (INput from port - ввод из порта) – реализуется передача данных из адресуемого порта ввода/вывода в регистр-аккумулятор (AL - для байтного порта, AX – для двухбайтного). По команде OUT (OUTput into port - вывод в порт) осуществляется передача данных из аккумулятора (AL или AX) в адресуемый командой порт ввода/вывода.

Для адресации портов ввода-вывода используются два подхода:

- прямая адресация (во втором байте задается адрес одного из 256 портов).

- косвенная адресация порта, через регистр DX (65536 портов).

Под портами ввода/вывода следует понимают адресуемые регистры контроллеров устройств ввода/вывода. Эти регистры являются доступными программе при использовании команд ввода/вывода. Адресация портов является стандартной и приводится в справочной литературе. Ввод/вывод является функцией операционной системы, поэтому команды IN и OUT в прикладных программах не используются.

Команды передачи адресов.

LEA dst, src (Load Effective Address) – загрузка эффективного адреса.

Команда LEA формирует эффективный адрес (EA) на основе постбайта адресации и загружает его в регистр с адресом, заданным полем reg постбайта. Эта команда используется для адресации как одиночных элементов, так и структур данных, размещаемых в памяти.

В качестве операнда-приемника (dst) может использоваться только 16-разрядный регистр (по разрядности EA), а в качестве операнда-источника (src) - имя переменной или структуры данных, размещаемых в памяти.

Пример: LEA BX, ARRAY

По этой команде в регистр BX загружается начальный адрес массива с именем ARRAY, размещаемого в памяти (по умолчанию в сегменте данных).

LDS dst, src (Load full pointer using DS) – загрузка полного указателя с использованием регистра DS.

LES dst, src (Load full pointer using ES) – загрузка полного указателя с использованием регистра ES.

Команды LDS и LES загружают полный указатель данных (seg:offset), состоящий из сегментного адреса и смещения, из памяти в пару регистров, одним из которых является сегментный регистр, именуемый мнемоникой команды.

По этим командам из памяти по эффективному адресу операнда источника осуществляется выборка двух слов, первое из которых (по меньшему адресу) загружается в регистр с адресом reg (по адресу операнда-приемника), а второе (по большему адресу) в сегментный регистр DS и ES соответственно.

10.3.2. Арифметические команды

Классификация арифметических команд приведена на рис.10.6. Они делятся на двухоперандные и однооперандные.

ADD (ADD integers) – сложение целых чисел.

Команда ADD прибавляет операнд-источник к операнду-приемнику и помещает результат на место операнда-приемника.

Примеры:

ADD ah, al – сложение al с ah;

ADD ax, bx – сложение ax с bx;

ADD ah, 4 – сложение ah с константой 4.

SUB (SUBtract integers) – вычитание целых чисел.

Команда SUB вычитает операнд-источник из операнда-приемника и помещает результат на место операнда-приемника.

Примеры*:*

SUB ah, al – Вычитание al из ah с записью результата в ah;

SUB al, 4 – Вычитание числа 4 из al с записью результата в al.

ADC (ADd integers with Carry) – сложение целых чисел с переносом.

Рис.10.6. Классификация арифметических команд

Команда ADC складывает операнд-источник, операнд-приемник и значение флага переноса (CF). Результат заносится в операнд-приемник. Команда обычно используется для организации сложения длинных чисел по частям (байтам или словам).

Пример:

ADC ah, al - сложение ah, al и значения флага переноса CF. Результат помещается в ah.

SBB (SuBtract integers with Borrow) – вычитание целых чисел с заемом.

Команда SBB вычитает операнд-источник из операнда-приемника, затем вычитает из результата значение флага заема (CF). Результат заносится в операнд-приемник. Команда SBB обычно

используется для организации вычитания длинных чисел по частям (байтам или словам).

Пример:

SBB AH, AL – из содержимого регистра AH вычитается содержимое регистра AL и значение флага CF.

CMP (CoMPare two operands) - Сравнение двух операндов.

Команда CMP реализуется здесь как вычитание операнда-источника из операнда-приёмника, но в отличии от команды SUB, результат вычитания не сохраняется в приёмнике, а лишь оказывает влияние на арифметические флаги. Команда CMP обычно используется для организации условных переходов по различным арифметическим флагам.

Примеры: CMP AH, CL – сравнение AH с CL;

CMP AX, 8 – сравнение непосредственного операнда (числа 8) с AX;

CMP AH, 0Ah – сравнение непосредственного операнда (шестнадцатеричного числа A) с AH.

Аддитивные команды изменяют значения всех арифметических флагов.

При выполнении команд ADD и ADC для беззнаковых чисел о переполнении формата можно судить по флагу CF, а для знаковых чисел - по флагу OF. При выполнении команд SUB, SBB над беззнаковыми операндами установка флага CF свидетельствует о том, что из меньшего операнда вычитался больший и результат операции представлен в беззнаковом дополнительном коде.

Мультипликативные команды.

Мультипликативные команды разделяются на два вида: беззнаковые и знаковые, что объясняется различием алгоритмов умножения и деления для беззнаковых и знаковых целых чисел. В командах задается единственный операнд (источник), т.к. другой операнд (приемник) использует неявную адресацию аккумулятора (аккумуляторные команды). Операнд-источник может адресовать регистр или память.

В **командах умножения (MUL / IMUL)** результат операции представляется в удвоенном формате по сравнению с форматом операнда.

MUL (unsigned integer MULtiply of AL register or AX register) – беззнаковое умножение целых чисел.

IMUL (signed integer MULtiply) – знаковое умножение целых чисел.

Обе команды производят умножение содержимого аккумулятора (множимого) на операнд-источник (множитель). Если операнд-источник имеет размер 8 бит, в качестве аккумулятора берется регистр AL, и результат помещается в AX. При размере операнда-источника в 16 бит в качестве аккумулятора используется AX, и результат помещается в пару регистров DX.AX (в DX – старшие разряды произведения, в AX – младшие).

Примеры: MUL CH – беззнаковое умножение AL на CH;

IMUL BX − знаковое умножение AX на BX.

Отличие команд MUL и IMUL состоит в интерпретации операндов и результата как беззнаковых целых чисел (команда MUL) или знаковых целых чисел (команда IMUL). О возможности представления произведения в том же формате, что и сомножители свидетельствуют флаги CF и OF, которые устанавливаются одинаково.

Для беззнакового умножения (MUL) эти флаги сбрасываются, если старшая половина произведения равна нулю, в противном случае флаги устанавливаются. Для знакового умножения (IMUL) флаги сбрасываются, если старшая половина произведения является полной копией знакового разряда младшей половины, в противном случае они устанавливаются.

В командах деления (DIV / IDIV) первый операнд (делимое) представляется в удвоенном формате по сравнению со вторым операндом (делителем). Делимое является аккумуляторным операндом. Результат целочисленного деления представляется в виде частного, которое замещает младшую половину делимого, и остатка, который замещает старшую половину делимого.

DIV (unsigned integer DIVide) – беззнаковое деление целых чисел.

IDIV (signed integer DIVide) – знаковое деление целых чисел.

Обе команды производят деление неявно заданного операнда-приемника на указанный в команде операнд-источник. Если размер

делителя 8 бит, в качестве делимого используется содержимое регистра AX. При размере делителя 16 бит делимое находится в паре регистров DX.AX. Частное в первом случае помещается в AL, остаток – в AH, во втором случае частное помещается в AX, а остаток – в DX.

В соответствии с принципами целочисленного деления, результат деления операнда A (делимое) на операнд B (делитель) представляется в виде частного C и остатка R.

Примеры: DIV BL – беззнаковое деление операнда из AX на операнд из BL;

IDIV CX – знаковое деление операнда из пары регистров DX (старшие разряды) и AX (младшие разряды) на операнд из CX.

Отличие команд DIV и IDIV состоит в интерпретации операндов и результатов как целых беззнаковых и знаковых чисел соответственно.

Команды расширения формата.

CBW (Convert Byte to Word) – преобразование байта в слово.

CWD (Convert Word to Double Word) – преобразование слова в двойное слово. CBW и CWD - безоперандные команды, состоящие из единственной мнемоники. Команды являются аккумуляторными, поддерживают знаковое представление целых чисел. Расширение операндов производится путём копирования знакового бита на все старшие разряды.

Команда CBW производит знаковое расширение байта из регистра AL до слова путем копирования (распространения) старшего бита (знакового разряда) регистра AL на все разряды регистра AH. Команда CWD производит аналогичную операцию над DX, расширяя его знаком слова из AX. Команды знакового расширения форматов обычно используются перед командами знакового деления для преобразования делимого в требуемый формат. Эти команды не оказывают влияния на арифметические флаги.

Команды инкремента / декремента.

INC (INCrement by 1) – инкремент (увеличение) на единицу.

DEC (DECrement by 1) – декремент (уменьшение) на единицу.

Команды **INC** и **DEC** являются одноадресными и производят сложение или вычитание единицы по отношению к заданному операнду. Операнд может быть регистровым или находиться в

памяти. По результатам этих команд устанавливаются все арифметические флаги, кроме CF, который сохраняет прежнее значение.

Примеры:INC AX – увеличить значение AX на 1;

DEC CL – уменьшить значение CL на 1.

Команда изменения знака.

NEG (NEGate two's complement) – изменение знака, дополнение до 2.

Команда производит изменение знака адресуемого операнда на противоположный. При этом осуществляется взятие дополнения от исходного операнда путем инвертирования всех его разрядов с добавлением единицы к младшему разряду. Фактически в АЛУ эта операция реализуется вычитанием исходного операнда из нуля. По результату этой команды устанавливаются все арифметические флаги. Флаг OF устанавливается в том случае, если операнд команды представляет собой максимальное по модулю отрицательное число для данного формата. Флаги CF и AF фиксируют соответствующие заемы при вычитании операнда из нуля.

Пример: NEG AX – изменение знака операнда из AX.

Команды десятичной коррекции.

DAA (Decimal Adjust AL after Addition) – десятичная коррекция после сложения. Корректирует результат предыдущего сложения в AL, преобразуя его в упакованное (BCD) двоично-десятичное число. Сравнивается младшая тетрада на условие AL > 9 и флаг AF на его установку, если хотя бы одно из этих условий выполнено, к младшей тетраде добавляется корректирующий код 6. Аналогичная проверка проводится для старшей тетрады со сравнением флага CF. Если значение этой тетрады будет > 9 или флаг CF установлен, добавляется корректирующий код 6 в старшую тетраду. Значения флагов AF и CF после выполнения команды свидетельствует о наличии (установка флага) или отсутствии (сброс флага) коррекции соответствующих тетрад.

Команды ASCII-коррекции.

AAA (ASCII Adjust after Addition) – ASCII-коррекция после сложения.

Корректирует результат сложения в AL преобразуя его в неупакованный (ASCII) формат цифр (0…9). Если младшая тетрада результата в AL > 9 или установлен флаг AF, осуществляется сложение AL с кодом 6, сброс старшей тетрады AL в ноль и установка обоих флагов CF и AF, а также инкремент AH. В противном случае осуществляется сброс флагов AF и CF.

AAS – ASCII Adjust after Subtraction – ASCII-коррекция после вычитания.

Команду AAS нужно выполнять только после команды SUB, которая оставляет байтный результат в регистре AL. Младшие тетрады операндов команды SUB должны находиться в диапазоне 0…9. В этом случае команда AAS корректирует регистр AL так, чтобы он содержал правильную десятичную цифру результата. Если число в AL перед AAS больше 9, то AH уменьшается на 1 и устанавливаются AF и CF. Если число в AL перед AAS меньше 9, то AH не изменяется. В любом случае старшая тетрада регистра AL содержит 0. Флаги AF и CF установлены в 1 при наличии десятичного переноса, а при отсутствии сброшены; флаги OF, SF, ZF, PF – не определены.

AAM (ASCII Adjust AX after Multiply) – ASCII-коррекция после умножения. Эта команда преобразует результат двоичного умножения 2 десятичных цифр из регистра AL в 2-х разрядное десятичное неупакованное число в регистре AX. Фактически ее действия сводятся к делению двоичного числа из AL на 10 и помещению частного как старших цифр в AH, а остатка как младших цифр в AL. Для корректного применения этой команды после команды MUL необходимо предварительно осуществить сброс старших тетрад байтных операндов. Флаги SF, ZF и PF устанавливаются по результату; флаги OF, AF, CPF – не определены.

AAD (ASCII Adjust AX before Division) – ASCII- коррекция регистра перед делением. В отличие от других команд коррекции, команда AAD корректирует не результат, а операнд (делимое), превращая неупакованное десятичное число из регистра AX в двоичное число, которое помещается в регистр AL. Эта команда, как правило, используется перед командой DIV.

Для корректного использования команд в отношении ASCII-кода 2 цифр необходимо после помещения их в регистр AX осуществить сброс старших тетрад. Фактически действие команды сводится к умножению содержимого AH на 10 и к сложению полученного результата с AL.

10.3.3. Логические команды

Логические команды компьютера реализуют поразрядные логические операции над операндами, помещая результат в операнде-приемнике. Типы логических команд представлены на рис.10.7.

AND (logical AND – логическое И) – выполняет операцию поразрядного логического умножения (конъюнкции). Каждый бит результата команды AND равен 1, если соответствующие биты обоих операндов равны 1, иначе бит результата равен 0.

Рис.10.7. Классификация логических команд

OR (logical inclusive OR – логическое ИЛИ) – выполняет операцию поразрядного логического сложения (дизъюнкции). Каждый бит результата команды OR равен 0, если соответствующие биты обоих операндов равны 0, в противном случае бит результата равен 1.

XOR (logical eXclusive OR – логическое ИЛИ) – выполняет операцию исключительного ИЛИ (сложение по модулю два). Каждый бит результата равен 1, если соответствующие биты обоих операндов различны, в противном случае бит результата равен 0.

TEST (logical compare – логическое сравнение) – выполняется как команда неразрушающего логического умножения, единственным результатом которой является установка арифметических флагов для последующего условного перехода. Один из операндов содержит проверяемую величину, а второй – маску, соответствующую проверяемым битам.

NOT (negate, one's complement – инверсия, дополнение до 1, логическое НЕ) – заменяет каждый бит его дополнением, инвертирует операнд. Влияние логических команд на арифметические флаги: все логические команды, кроме команды NOT, оказывают на арифметические флаги следующее влияние: флаги OF и CF сбрасываются, флаг AF принимает неопределенное значение, остальные флаги (ZF, SF, PF) устанавливаются по общим правилам. Команда NOT не изменяет значения флагов.

Примеры:

AND AL, 0FH – выделение младшей тетрады регистра AL;

OR AH, 0AAH – установка нечетных битов регистра AH;

XOR AL, 0F0H – инвертирование старшей тетрады регистра AL;

TEST AH, 0FFH – проверка содержимого регистра AH, например, после команды деления DIV / IDIV на байтный делитель с целью дальнейшего перехода по нулевому значению остатка (по флагу ZF);

NOT AH – инвертирование содержимого регистра AH.

10.3.4. Команды сдвигов

В ассемблерной нотации команды сдвигов используются два операнда, первый задает, собственно, сдвигаемый операнд, который может размещаться в регистре или в памяти, а второй определяет способ задания количества сдвигов. Если этот операнд равен единице, то выполняется так называемый однократный сдвиг (сдвиг на 1 разряд), если же в качестве второго операнда задается CL, то число разрядов, на которое осуществляется сдвиг, выбирается из регистра CL, в этом случае имеет место так называемый многократный сдвиг.

Классификация команд сдвигов приведена на рис.10.8.

SAL/SAR/SHL/SHR (Shift instruction) – команды сдвига.

Отличие команд арифметического сдвига от команд логического сдвига проявляется только для сдвига вправо. При арифметическом сдвиге вправо (SAR) в освобождающийся старший разряд копируется его прежнее значение,что позволяет сохранить знак операнда при его сдвиге вправо. При этом, при выполнении команды логического сдвига вправо (SHR), в свободный старший разряд заносится 0.

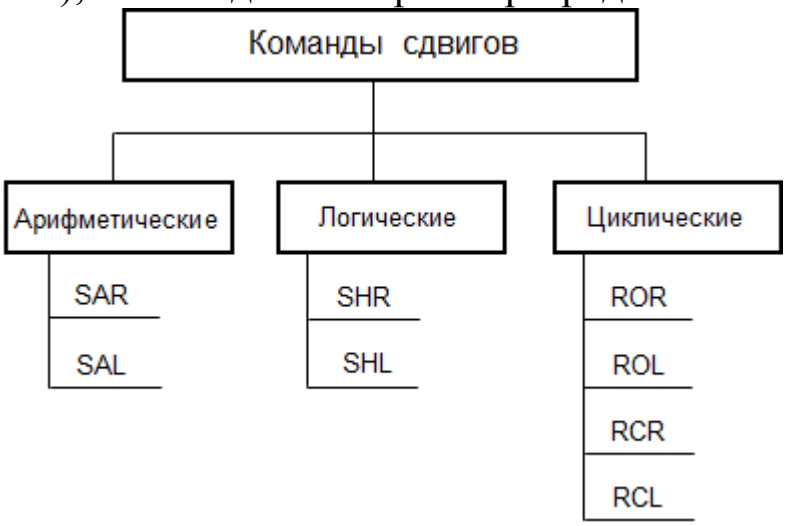


Рис.10.8. Классификация команд сдвигов

Арифметический и логический сдвиги влево реализуются одинаково, в связи с чем в машинной системе команд они имеют одинаковый код операции. Команды арифметического и логического сдвигов, как правило, используются в ассемблерных программах для умножения или деления на степени двойки. Например, сдвиг вправо на два разряда осуществляет деление операнда на 4, а сдвиг влево на два разряда – умножение на 4. При этом арифметический сдвиг используется при знаковом представлении операнда, а логический – при беззнаковом.

RCL/RCR/ROL/ROR – вращение (циклический сдвиг).

Команды циклического сдвига реализуют так называемое вращение (ротацию) операнда, при котором спадающие разряды сохраняются в освобождающихся. Отличие двух модификаций команд циклического сдвига является возможность включения или исключения флага CF в кольцо (из кольца). Для RCR и RCL предполагается, что предварительное значение флага CF (до выполнения команды) равно 0.

Влияние на арифметические флаги: команды арифметического и логического сдвигов оказывают влияние на все арифметические

флаги, кроме AF, значение этого флага после команд сдвигов не определено. Флаг OF является актуальным только для команд однократного сдвига, при сдвигах многократных он принимает неопределенное значение.

При левом сдвиге (SAL , SHL) установка флага OF производится в том случае, если при сдвиге изменяется значение старшего бита операнда, интерпретируемого как знак.

10.3.5. Команды управления программой

По видам передач эти команды разделяются на два типа: внутрисегментные (ближние) передачи типа **NEAR** и межсегментные (дальние) передачи типа **FAR**. При ближней передаче управления в качестве адреса перехода задается адрес команды в том же сегменте кода. В связи с этим реализация передач управления осуществляется модификацией только одного регистра IP. Дальняя передача управления адресует команду в другом сегменте кода.

Дальние передачи управления используются только в отношении команд безусловного перехода (JMP), вызова процедур (CALL) и возврата (RET). Для команд условных переходов и циклов используется только ближняя передача управления.

Классификация команд управления приведена на рис.10.9.

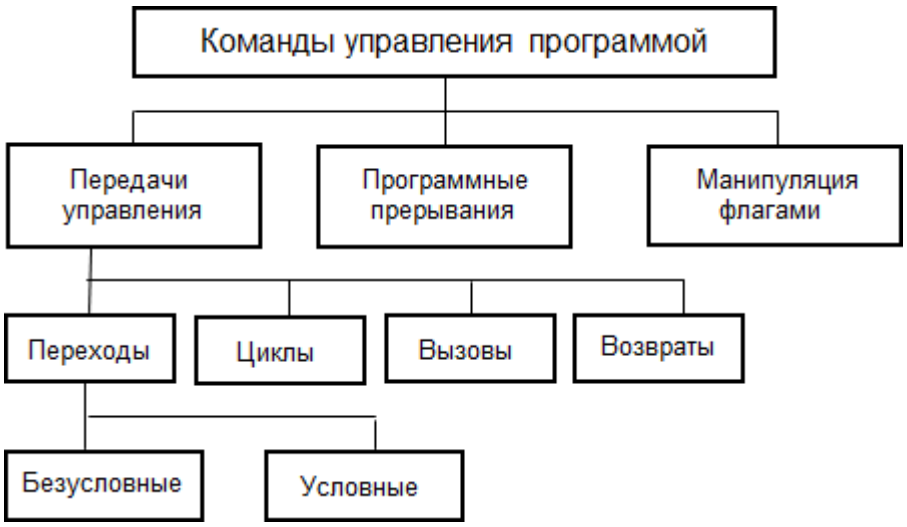


Рис.10.9. Команды управления программой

Команды передачи управления.

Команда **безусловного перехода JMP** (JuMP if condition is met) передаёт управление другой команде управления по заданному адресу

перехода. В JMP могут использоваться все способы задания адреса перехода.

Команды **условного перехода** характеризуются проверкой некоторого заданного кодом операции условия и реализацией перехода по заданному адресу при выполнении условия или к следующей команде при его невыполнении.

Условные переходы принято делить на: знаковые, беззнаковые, по отдельным флагам, по счётчику.

Отличие **знаковых переходов** от беззнаковых состоит в способе интерпретации данных, над которыми производится предшествующая переходу команда, как знаковых, так и беззнаковых целых чисел. Как правило, этой командой является команда сравнения CMP.

В аналогичных по смыслу знаковых и беззнаковых переходах анализируются разные арифметические флаги. В знаковых переходах в мнемонике используются буквы G – Greater (больше), L – Less (меньше), а в беззнаковых переходах буквы A – Above (выше), B - Below (ниже).

В переходах по **отдельным флагам** возможно использование любого арифметического флага, кроме AF, переход к которому отсутствует (JC – по переносу; JO - по переполнению).

Команда перехода по счетчику JCXZ, в отличие от других команд условных переходов, не анализирует арифметические флаги. Эта команда проверяет содержание регистра CX на равенство 0. Если равенство имеет место, тогда происходит передача управления по заданному адресу. Если равенство не выполняется, тогда управление передаётся следующей команде.

Эту команду обычно используют при входе в цикл, который при некоторых условиях может не выполняться ни одного раза, т.е. число повторений будет равно нулю. Использование JCXZ в начале цикла в случае нулевого значения счётчика CX позволяет обойти этот цикл, не выполняя команды, содержащиеся в теле цикла.

Таким образом, команда JCXZ не изменяет содержимого регистра CX, а только проверяет его значение.

Команда организации цикла LOOP осуществляет декремент регистра счётчика CX и сравнение его нового содержимого с нулём. При равенстве осуществляется переход к следующей команде

программы и выход из цикла. При невыполнении равенства осуществляется передача управления по адресу перехода в начало цикла. Для корректного применения команды LOOP необходимо перед первым входом в цикл загрузить в CX число повторений этого цикла.

Команда LOOP является завершающей командой цикла. Поскольку переход в начало цикла осуществляется вверх по программе, то в качестве смещения (diff) задаётся отрицательное число. Команда LOOP и её модификации задаёт единый способ задания адреса перехода, короткий относительный типа SHORT (байтное смещение).

Модификации команды LOOP:

LOOPE (повторять пока равно);

LOOPZ (повторять пока ноль).

В этих командах дополнительным условием выхода из цикла является равенство нулю флага ZF. Эти команды целесообразно использовать для поэтапного сравнения двух массивов до обнаружения первых несовпадающих элементов, при этом предполагается, что в теле цикла используется команда CMP.

Другой модификацией LOOP являются:

LOOPNE (выполнять пока не равно);

LOOPNZ (выполнять пока не ноль).

Для них дополнительным условием выхода из цикла является значение флага ZF=1. Эти команды рационально использовать для поиска первых совпадающих элементов в сравниваемых массивах. Эти парные команды, несмотря на различие мнемоник (E или Z), кодируются одинаковыми машинными кодами (у LOOPE и LOOPZ один код, у LOOPNE и LOOPNZ - другой).

Команда вызова процедуры и возврата из нее CALL может использовать все способы задания адреса перехода вызываемой процедуры, кроме короткого относительного (SHORT). Для обеспечения возможности возврата в основную программу после завершения вызванной процедуры (подпрограммы), выполнение команды CALL сопровождается сохранением адреса возврата в стеке. Адрес возврата состоит из содержимого IP при ближнем вызове

(CALL NEAR) и дополнительно из CS при дальнем вызове CALL FAR.

RET (RETurn) - осуществляет возврат из процедуры в основную программу и является завершающей командой в процедуре.

Вид возврата, ближний или дальний, совпадает с видом предшествующего вызова. Команда RET производит извлечение из стека адреса возврата в виде содержимого IP (ближний возврат), либо в виде пары IP:CS (дальний возврат).

Команды программных прерываний являются генераторами прерываний и включаются в программу для вызова различных служебных функций, команды (**INT n**) DOS или BIOS. Эти команды выполняют функции, аналогичные тем, которые реализуются процессором при обработке внешних аппаратных прерываний.

В основном эти функции сводятся к сохранению состояния прерванной программы и вызову программы-обработчика прерываний.

Действия, выполняемые двухбайтной командой INT n (с заданным типом прерывания):

- последовательное сохранение в стеке регистров FLAGS, CS, IP;
- модификация второго байта команды, номера типа прерывания, в адрес вектора прерывания сдвигом на два разряда влево (умножением на 4);
- чтение из основной памяти двух последовательных слов, адреса программы-обработчика прерываний, с их загрузкой в регистры IP и CS.

Однобайтные команды **INT** по умолчанию используют тип прерывания, равный 3, выполняемые ею действия аналогичны выполняемым выше. Как правило, эту команду используют для реализации останова в контрольной точке при отладке программы.

INTO - предварительно осуществляет проверку значения флага OF. Если он сброшен, то осуществляется передача управления следующей команде про граммы. При установленном флаге реализуются действия, аналогичные определённым выше, для типа прерываний 4 (стандартный тип прерывания по переполнению).

IRET - возврат из обработчика прерывания. Действие этой команды аналогично действию однобайтной команды RET FAR, но дополнительно из стека извлекается содержимое регистра флагов.

Команды манипуляции флагами производят определённое действие:

- сброс **CLC**;
- установку **STC**;
- инвертирование **CMC**.

Эти действия осуществляются только в отношении 3-х флагов: переноса CF, направления DF и прерывания IF, причём инвертирование только в отношении флага переноса CF. Все команды манипуляции флагами являются однобайтными, неявно адресуют операнд в виде соответствующего флага.

Команды обработки строк.

MOVS (MOVe data from String to string) – пересылка элемента строки источника в элемент строки-приемника.

LODS (Load String operand) – загрузка элемента строки-источника в аккумулятор (AL или AX).

STOS – пересылка содержимого аккумулятора (AL или AX) в элемент строки-приемника.

CMPS – сравнение элемента строки-приемника и строки-источника (реализуется как неразрушающее вычитание элемента строки-приемника из элемента строки-источника, по результату которого устанавливаются арифметические флаги).

SCAS – сканирование (просмотр) элемента строки-приемника. Реализуется как неразрушающее вычитание этого элемента из содержимого аккумулятора с установкой арифметических флагов.

Вопросы для контроля

1. Каковы основные элементы прикладной архитектуры компьютера?
2. В чем преимущества программной совместимости очередных моделей процессоров?
3. Какие новшества были введены в процессоры серии Pentium?
4. Какие преимущества появились у процессора Pentium 4?
5. С какой модели процессоров Intel началась эра многоядерных архитектур?
6. В чем преимущества регистровой структуры процессоров?
7. Каковы функции сегментных регистров?
8. Какова роль регистра флагов и регистра управления в реализации выполняемой процессором программы?
9. Назовите типы команд процессоров Intel?
10. Дайте определение командам условного и безусловного переходов.

ГЛАВА 11. ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА НА МНОГОЯДЕРНЫХ ПРОЦЕССОРАХ

Существует два подхода к увеличению производительности процессора. Первый - увеличение тактовой частоты процессора, второй – увеличение количества инструкций программного кода, выполняемых за один такт процессора. Увеличение тактовой частоты не может быть бесконечным и определяется технологией изготовления процессора. При этом рост производительности не является прямо пропорциональным росту тактовой частоты, то есть наблюдается тенденция насыщаемости, когда дальнейшее увеличение тактовой частоты становится нерентабельным.

Разработка более совершенных архитектур процессоров, содержащих большее число функциональных исполнительных устройств, с целью повышения количества команд, одновременно исполняемых за один такт, — традиционный альтернативный росту тактовой частоты путь повышения производительности.

Другим предшественником многоядерного подхода можно считать технологию Intel HyperThreading (HT), где также есть небольшое дублирование аппаратуры и использование двух потоков инструкций, использующих общее ядро.

Многоядерный процессор имеет два или больше исполнительных ядер. Операционная система рассматривает каждое из исполнительных ядер, как дискретный процессор со всеми необходимыми вычислительными ресурсами. Поэтому многоядерная архитектура процессора, при поддержке соответствующего программного обеспечения, осуществляет полностью параллельное выполнение нескольких программных потоков.

К 2006 году все ведущие разработчики микропроцессоров создали двуядерные процессоры. Переход к многоядерным процессорам становится основным направлением повышения производительности вычислительных систем. В связи с этим, знание основ функционирования вычислительных систем на многоядерных процессорах является актуальной.

11.1. Технологии многопоточности

Технология TLP (Thread Level Parallelism).

Основные направления развития архитектуры современных процессоров определяются стремлением к увеличению их производительности. Одно из решений данной проблемы связано с реализацией концепции "параллелизма на уровне тредов (потоков)" - TLP. Если программные коды не в состоянии загрузить работой все или даже большинство функциональных устройств, то можно разрешить процессору выполнять более чем одну задачу (тред, или поток), чтобы дополнительные потоки загрузили простаивающие устройства.

Можно провести аналогию с многозадачной операционной системой: чтобы процессор не простаивал, когда задача оказывается в состоянии ожидания (например, завершения ввода-вывода), операционная система переключает процессор на выполнение другой задачи. Некоторые механизмы диспетчеризации в операционной системе имеют аналоги в многопотоковой архитектуре (MTA – Multi Threading Architecture). Очевидно, архитектура, поддерживающая параллелизм на уровне потоков (TLP), должна гарантировать, что треды не будут использовать одновременно одни и те же ресурсы, для чего требуются дополнительные аппаратные средства (например, дублирование регистровых файлов). Однако оказалось, что можно реализовать MTA на базе современных суперскалярных процессоров, и это требует лишь относительно небольших аппаратных доработок.

При использовании базового типа параллелизма на уровне потоков (TLP) в процессоре необходимо иметь не менее двух аппаратных расширений для потоков. Это регистры общего назначения, счетчик команд, слово состояния процесса. В любой момент времени работает только один поток (тред). Он выполняется до возникновения определенной ситуации (например, выполнения команды загрузки регистра при отсутствии данных в кэш-памяти). В этом случае процессор переключается на выполнение другого потока. Поскольку при непопадании в кэш-память операции с памятью могут потребовать до сотни тактов процессора, его простои по причине ожидания данных могли бы быть весьма значительными.

Технологии SMT (Simultaneous Multi-Threading).

Современные процессоры, имеющие возможности спекулятивного внеочередного выполнения команд, в подобной ситуации могут продолжить выполнение других команд, но на практике число независимых команд быстро исчерпывается и процессор останавливается. Архитектура с одновременным выполнением тредов - SMT (Simultaneous Multi-Threading) допускает одновременное выполнение нескольких потоков. В этом случае на каждом новом такте на выполнение в какое-либо исполнительное устройство может направляться команда любого потока. По сравнению с суперскалярными процессорами, поддерживающими внеочередное спекулятивное выполнение команд и использующими механизм переименования регистров, для SMT необходимы, в частности, следующие аппаратные средства:

- несколько счетчиков команд (по одному на поток) с возможностью выбора любого из них на каждом такте;
- средства, ассоциирующие команды с потоком, которому они принадлежат (необходимы, в частности, для работы механизмов предсказания переходов и переименования регистров);
- несколько стеков адресов возврата (по одному на поток) для предсказания адресов возврата из подпрограмм;
- специальная дополнительная память в процессоре (в расчете на каждый поток) для процедуры удаления из буфера выполненных вне очереди команд.

Одна из основных особенностей SMT у многих современных процессоров - переименование регистров, когда логические (архитектурные) регистры отображаются в физические, с которыми и ведется реальная работа. Техника переименования регистров может, очевидно, применяться для того, чтобы избежать прямого дублирования файлов регистров, как аппаратной принадлежности потока.

Технология Hyper-Threading.

Представленная в 2002 году компанией Intel технология Hyper-Threading – пример многопоточной обработки команд. Данная технология является чем-то средним между многопоточной обработкой, реализованной в мультипроцессорных системах, и

параллелизмом на уровне инструкций, реализованном в однопроцессорных системах. Фактически технология Hyper-Threading позволяет организовать **два логических процессора в одном физическом.** Таким образом, с точки зрения операционной системы и запущенного приложения в системе существует два процессора, что даѐт возможность распределять загрузку задач между ними. Посредством реализованного в технологии Hyper-Threading принципа параллельности можно обрабатывать инструкции в параллельном (а не в последовательном) режиме, то есть для обработки все инструкции разделяются на два параллельных потока. Это позволяет одновременно обрабатывать два различных приложения или два различных потока одного приложения и тем самым увеличить количество инструкций, выполняемых процессором в секунду, что сказывается на росте его производительности.

В среднем, при выполнении кода для типичного набора команд, реально используется только 35% исполнительных ресурсов процессора, а 65% исполнительных ресурсов процессора простаивают, что означает неэффективное использование возможностей процессора. Эффективнее было бы реализовать работу процессора таким образом, чтобы в каждом тактовом цикле максимально использовать его возможности. Именно эту идею и реализует технология Hyper-Threading, подключая незадействованные ресурсы процессора к выполнению параллельной задачи.

Выполняемая программа разбивается на два параллельных потока (threads). На стадии подготовки программы задача компилятора и операционной системы (на этапе выполнения) заключается в формировании таких последовательностей независимых команд, которые процессор мог бы обрабатывать параллельно, загружая по возможности не занятые в данном процессе обработки одним из потоков функциональные блоки. Операционная система воспринимает физический суперскалярный процессор как два логических процессора и организует поступление на эти два процессора двух независимых потоков команд.

В конструктивном плане процессор с поддержкой технологии Hyper-Threading состоит из двух логических процессоров, каждый из которых имеет свои регистры и контроллер прерываний. Это значит,

две параллельно исполняемые задачи работают со своими собственными независимыми регистрами и прерываниями, но при этом используют одни и те же ресурсы процессора для выполнения своих задач. После активизации каждый из логических процессоров может самостоятельно и независимо от другого процессора выполнять свою задачу, обрабатывать прерывания либо блокироваться. Таким образом, от реальной двухпроцессорной конфигурации новая технология отличается только тем, что оба логических процессора используют одни и те же исполняющие ресурсы, одну и ту же разделяемую между двумя потоками кэш-память и одну и ту же системную шину.

На рис. 11.1. представлена архитектура процессора Pentium 4, в которой реализована технология Hyper-Threading в сочетании с суперскалярной и конвейерной обработкой.

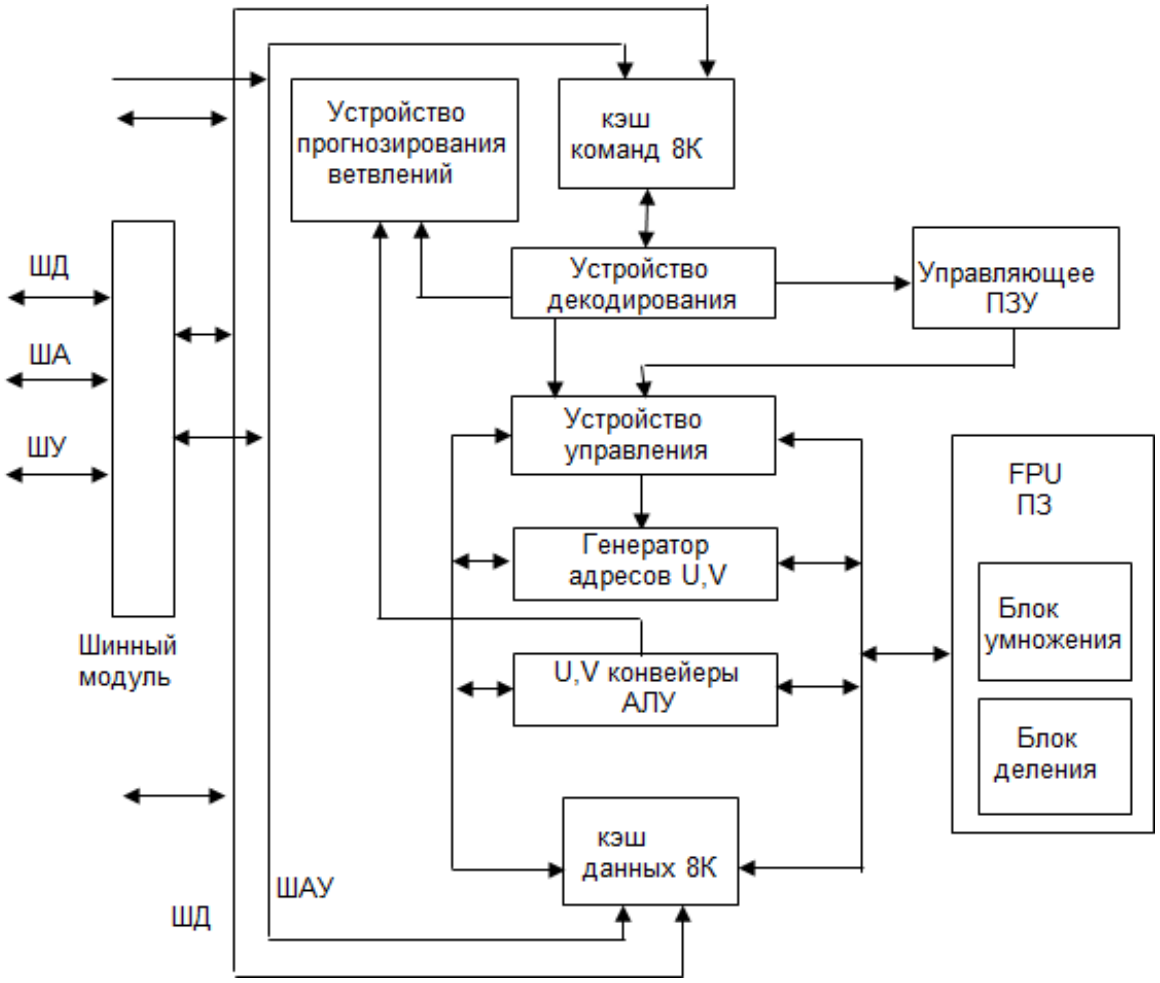


Рис.11.1. Архитектура процессора Pentium 4

В схеме процессора Pentium 4 также имеются идентичные блоки, предназначенные для реализации суперскалярной обработки: два независимых кэша для команд и данных, два АЛУ конвейерного типа, узлы декодирования и диспетчеризации. Имеется дополнительный сопроцессор плавающей запятой. Но при этом устройство прогнозирования ветвлений, устройство управления и управляющее ПЗУ обеспечивают режим многопоточной обработки на одном АЛУ.

Отличием является исполнение процессора в виде двух многоступенчатых конвейеров, а также то, что две параллельно исполняемые задачи работают со своими собственными независимыми узлами, но при этом используют одни и те же ресурсы процессора для выполнения своих задач.

Оба логических процессора совместно используют одну и ту же кэш-память. Если поток, обрабатываемый одним логическим процессором, хочет прочитать данные, кэшированные другим логическим процессором, он может их взять из общего кэша. Из-за этого вероятность конфликтов по памяти и вероятность снижения производительности процессора возрастают.

Наличие только одного вычислительного ядра не позволяет достичь удвоенной производительности, однако за счет большей отдачи всех внутренних ресурсов, общая скорость вычислений существенно возрастает. Это особенно заметно, когда потоки содержат команды разных типов, тогда замедление обработки в одном из них компенсируется большим объемом работ, выполненных в другом потоке. Эффективность технологии Hyper-Threading во многом зависит от работы операционной системы, так как разделение команд на потоки осуществляет именно операционная система.

По оценке Intel, прирост скорости вычислений в многозадачных приложениях может достигать более 20%. Возможны ситуации, когда прирост в быстродействии может быть незаметен, так как обладание ресурсом кэш-памяти оказывает заметное влияние на рост производительности.

Эффективность данной технологии находится в прямой зависимости от характера реализуемого приложения.

11.2. Многоядерные процессоры

Решением реализовать параллелизм на уровне программных потоков не увеличивая существенно стоимость системы обработки является создание **многоядерных процессоров.**

При этом важно понимать, что переход к многоядерности одновременно знаменует и наступление эры параллельных вычислений. На самом деле, задействовать вычислительный потенциал многоядерных процессоров можно только, если осуществить разделение выполняемых вычислений на информационно независимые части и организовать выполнение каждой части вычислений на разных ядрах. Подобный подход позволяет выполнять необходимые вычисления с меньшими затратами времени, и возможность получения максимального ускорения ограничивается только числом имеющихся ядер и количеством "независимых" частей в выполняемых вычислениях. Параллельные вычисления становятся неизбежными и повсеместными.

Численные методы в случае многоядерности должны проектироваться как системы параллельных и взаимодействующих между собой процессов, допускающих исполнение на независимых вычислительных ядрах. Применяемые алгоритмические языки и системное программное обеспечение должны обеспечивать создание параллельных программ, организовывать синхронизацию и взаимоисключение асинхронных процессов. Все перечисленные проблемы организации параллельных вычислений увеличивают существующий разрыв между вычислительным потенциалом современных компьютерных систем и имеющимся алгоритмическим и программным обеспечением применения компьютеров для решения сложных задач. И, как результат, устранение или, по крайней мере, сокращение этого разрыва является одной из наиболее значимых задач современной науки и техники.

Технологии многопоточности, рассмотренные выше, позволяет достичь многопроцессорности на логическом уровне. Затраты на поддержку такой технологии являются сравнительно небольшими, но и получаемый результат достаточно далек от максимально-

возможного – ускорение вычислений от использования многопоточности оказывается равным порядка 20-30%. Дальнейшее повышение быстродействия вычислений, при таком подходе, по прежнему лежит на пути совершенствования процессора, что требует разрешения ряда сложных технологических проблем. Возможное продвижение по направлению к большей вычислительной производительности может быть обеспечено на основе возврата к более простым процессорам, с более низкой тактовой частотой и с менее сложной логикой реализации. Простые процессоры требуют для своего изготовления меньшее количество логических схем, что приводит к освобождению в рамках кремниевого кристалла, используемого для изготовления процессоров, большого количества свободных транзисторов. Эти свободные транзисторы, в свою очередь, могут быть использованы для реализации дополнительных вычислительных устройств, которые могут быть добавлены к процессору. Фактически, данный подход позволяет реализовать в единственном кремниевом кристалле несколько вычислительных ядер в составе одного многоядерного процессора, при этом по своим вычислительным возможностям эти ядра могут не уступать обычным (одноядерным) процессорам.

Важный момент – потенциал производительности многоядерных процессоров может быть задействован только при надлежащей разработке программного обеспечения – программы должны быть очень хорошо распараллелены. Сложность разработки параллельных программ значительно превышает трудоемкость обычного последовательного программирования. Тем самым, проблема обеспечения высокопроизводительных вычислений перемещается теперь из области компьютерного оборудования в сферу параллельного программирования. И здесь нужны новые идеи и перспективные технологии для организации массового производства параллельных программ.

Методы и средства реализации параллельной обработки зависят от того, на каком уровне они реализуются. Можно выделить несколько уровней параллелизма.

Уровень заданий – когда несколько независимых задач одновременно выполняются на разных процессорах независимо друг от друга.

Уровень программ – когда части одной задачи выполняются за счет совместной одновременной работы множества процессоров.

Уровень команд – когда выполнение отдельной команды разделяется на этапы (фазы), исполняемые в конвейерном режиме.

Уровень разрядов – когда многоразрядные коды машинного слова обрабатываются одновременно.

Параллелизм **уровня задания** возможен между независимыми задачами или их фазами решения. Основным средством реализации параллелизма на уровне заданий служат многопроцессорные и многомашинные вычислительные системы, в которых задачи распределяются по отдельным процессорам или машинам.

Параллелизм на **уровне программ** реализуется в двух вариантах: когда в программе могут быть выделены независимые участки, которые допустимо выполнять параллельно, либо когда в пределах одного программного цикла отдельные итерации реализуются независимо друг от друга. В этом случае параллелизм может быть реализован за счет большого количества процессоров или множества функциональных блоков одной машины.

Параллелизм **уровня команд** имеет место, когда обработке нескольких команд или их отдельных этапов перекрываются во времени (совмещение операций или конвейеризация).

Таким образом, уровни параллелизма аппаратных или программных средств реализуются путем разделения общей задачи на отдельные части и их исполнением либо на отдельных узлах, либо на отдельных процессорах, либо на отдельно работающих компьютерах вычислительной системы.

Закон Амдала.

Этот закон оценивает степень повышения скорости вычислений за счет распределения вычислительной нагрузки по множеству параллельно работающих процессоров. В идеальном случае система из и **n** процессоров могла бы ускорить вычисления в **n** раз. В реальности достичь такого показателя по ряду причин не удается. Главная из этих причин заключается в невозможности полного

распараллеливания ни одной из задач. Как правило, в каждой программе имеется фрагмент кода, который принципиально должен выполняться последовательно и только одним из процессоров. Ориентируясь на параллельную вычислительную систему необходимо четко сознавать, что добиться прямо пропорционального числу процессоров увеличения производительности не удастся, и, естественно, встает вопрос о том, на какое реальное ускорение можно рассчитывать. Ответ на этот вопрос в какой-то мере дает закон Амдала.

Джин Амдал - один из разработчиков всемирно известной системы IBM 360 предложил формулу, отражающую зависимость ускорения вычислений, достигаемого на многопроцессорной вычислительной систем, от числа процессоров и соотношения между последовательной и параллельной частями программы. Показателем сокращения времени вычислений служит такая метрика, как ускорение.

Ускорение S - это отношение времени Ts, затрачиваемого на проведение вычислений на однопроцессорной систем (в варианте наилучшего последовательного алгоритма), ко времени Tp, решения той же задачи на параллельной системе (при использовании наилучшего параллельного алгоритма):

$$S=Ts/Tp$$

Проблема рассматривается в следующей постановке (рис. 11.2).

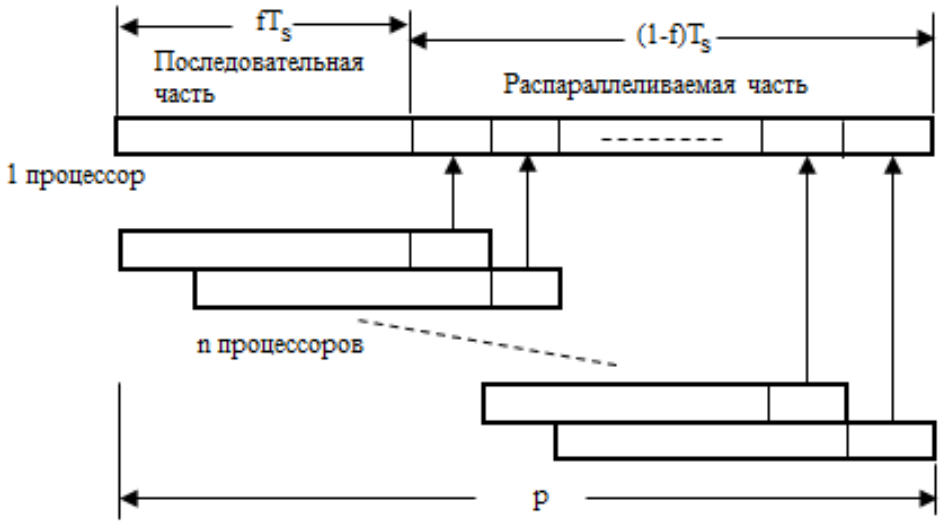


Рис.11.2. Иллюстрация к закону Амдала

Прежде всего, объем решаемой задачи с изменением числа процессоров, участвующих в ее решении, остается неизменным. Программный код решаемой задачи состоит из двух частей: последовательной и распараллеливаемой. Обозначим долю операций, которые должны выполняться последовательно одним из процессоров, через f, где 0 <=f<= 1 (здесь доля понимается не по числу строк кода, а по числу реально выполняемых операций). Отсюда доля, приходящаяся на распараллеливаемую часть программы, составит 1–f. Крайние случаи в значениях f соответствуют полностью параллельным (f = 0) и полностью последовательным (f = 1) программам. Распараллеливаемая часть программы равномерно распределяется по всем процессорам.

С учетом приведенной формулировки имеем:

$$T_p = f \times T_s + \frac{(1-f)\times T_s}{n}.$$

В результате получаем формулу Амдала, выражающую ускорение, которое может быть достигнуто на системе из ***n*** процессоров:

$$S = \frac{T_s}{T_p} = \frac{n}{1+(n-1)\times f}.$$

Формула выражает простую и обладающую большой общностью зависимость. Характер зависимости ускорения от доли последовательной. части программы на одном процессоре показан на рис. 11.3.

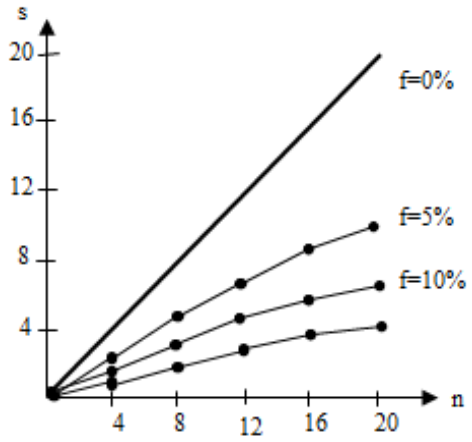


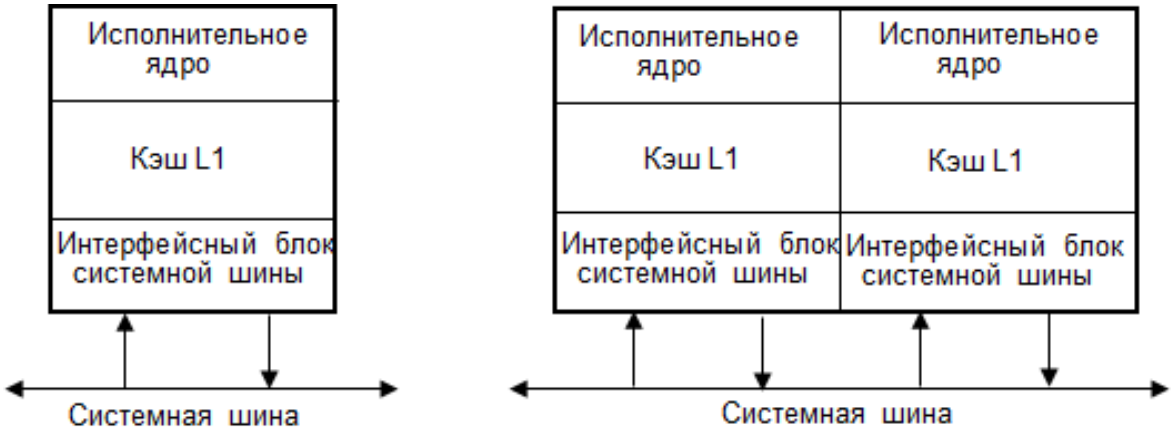
Рис.11.3. Зависимость ускорения от доли последовательной части программы на одном процессоре

Распараллеливание ведет к определенным издержкам (дополнительные операции, связанные с распределением программ по процессорам, обмен информацией между процессорами), которых нет при последовательном выполнении программы.

11.3. Аппаратная платформа параллельных вычислений

Многоядерные процессоры поддерживают режим мультипроцессорной обработки на едином кристалле (полупроводниковом чипе). Это технология **CMP (Chip Multiprocessing**). Проектировщики аппаратуры реализовали исполнительные ядра обработки в архитектуре одного процессора. Эти ядра по сути – отдельные процессоры на одном технологическом кристалле. Исполнительные ядра имеют собственный набор аппаратных и архитектурных ресурсов. В зависимости от конструкции эти процессоры могут совместно использовать свой кэш на том же кристалле. В результате применения многоядерной архитектуры каждая последовательность команд (программный поток) получает аппаратную среду выполнения в свое исключительное распоряжение. Это позволяет каждому потоку выполняться истинно параллельным образом.

На рисунке 11.4 представлены аппаратные конфигурации одноядерного (для сравнения) и двухъядерных процессоров.



«а» «б»

Рис.11.4. Схемы одноядерного («а») и двухъядерного («б») процессоров

На рисунке 11.4, «а» представлена схема одноядерного процессора в составе исполнительного блока, кэш-памяти первого уровня (L1) и интерфейсного блока для связи с системной шиной компьютера, а на рис.11.4, «б» двухъядерный вариант с индивидуальными кэш первого уровня и интерфейсными блоками. Название «исполнительное ядро» означает, что это сочетание функциональных блоков, которые непосредственно участвуют в выполнении команды. Совместно с памятью и интерфейсом они объединяются в «ядро». Количество ядер в процессорах может различаться, но ядра остаются симметричными. Эти обеспечивается регулярность структуры и возможность размещения на кристалле процессора большего числа ядер.

Как известно, команда процессора является основной единицей реализации программного обеспечения. Команда состоит из двух основных частей: операционной части (код операции, система адресации) и адресной части (адреса хранения в памяти обрабатываемых операндов). Именно эти два компонента определяют важность двух основных узлов компьютера – операционного узла и памяти. Наличие этих двух основных узлов обработки плюс интерфейса связи с другими элементами аппаратной архитектуры и определяет минимальную, функционально самостоятельную единицу обработки данных **– ядро.**

Процессор может включать в себя несколько ядер. Он отличается от ядра большим числом режимов обработки, более широкой номенклатурой видов внутренней памяти (кэш-память, ОЗУ, ПЗУ), имеет всевозможные контроллеры и расширители, магистрали обмена данными. Если учесть, что каждое из ядер процессора может обеспечить свой уровень поточной обработки, можно считать многоядерную архитектуру (при относительной простоте аппаратных решений) одним из наиболее эффективных вариантов организации вычислительного процесса.

Для эффективного использования возможностей многоядерных процессоров необходимо понимать особенности поточной модели программирования, а также возможности аппаратной части платформы. Чтобы обеспечить параллельную обработку программ аппаратная платформа должна поддерживать возможность

одновременного выполнения нескольких программных потоков. При грамотной реализации поточная обработка может повысить производительность за счет более эффективного использования аппаратных ресурсов.

Программный поток – это отдельная последовательность связанных между собой команд, которая выполняется независимо от других последовательностей команд. Каждая программа имеет, по крайней мере, один программный поток – главный, который инициализирует программу и начинает выполнение первых команд.

На аппаратном уровне программный поток – это исполнение алгоритма независимо от других аппаратных средств выполнения потоков. Аппаратные исполнительные ресурсы для программных потоков назначает операционная система.

Компьютерные архитектуры и их платформы могут быть классифицированы в двух разных измерениях. Первое измерение – это количество потоков команд, которые данная архитектура способна выполнить в единицу времени. Второе измерение – это количество потоков данных, которые могут быть обработаны в единицу времени. Любая компьютерная система может быть описана в терминах обработки потоков команд и потоков данных. С учетом такого представления процессов обработки родилась известная классификация Флинна, при этом параллельные, в том числе многоядерные компьютеры относятся либо к **категории SIMD**, либо к **категории MIMD.** Программы для них могут использовать параллелизм на уровне данных и на уровне заданий. Эти две архитектуры максимально используют возможности параллельной потоковой обработки с использованием одновременно функционирующих вычислительных узлов. Развитие именно этих двух архитектур в сочетании с достижениями современной микропроцессорной техники и привело к созданию многоядерных процессоров. В основном представленные на сегодняшний день многоядерные процессоры имеют MIMD-архитектуру, но есть также группа процессоров, построенных как SIMD-системы, — это рассмотренные ранее графические процессоры.

Перед использованием многоядерного оборудования системотехники и разработчики программного обеспечения должны выбрать модель обработки, которая способна обеспечить максимальный прирост производительности и свести к минимуму модификацию существующих программных компонентов. Многоядерные кристаллы отвечают этим требованиям, обеспечивая значительно более высокую производительность на единицу мощности, веса и площади, чем традиционные однопроцессорные микросхемы. Кроме того, платы на основе многоядерных кристаллов уменьшают число разъёмов в системе, а, следовательно, снижают её вес и энергопотребление, уменьшают размер корпуса и сокращают расходы на создание системы.

Системотехники и разработчики программного обеспечения должны освоить использование многоядерной технологии не только из-за её преимуществ, но и потому, что многоядерная технология быстро становится основой всё большего числа новых процессорных архитектур. Тем не менее, многоядерность создаёт существенную проблему переноса программного обеспечения. Разработчики должны перейти от модели последовательного исполнения, в которой задачи по очереди выполняются на единственном процессоре, к модели параллельного исполнения, в которой задачи выполняются одновременно. Чем более высокого уровня параллелизма достигают разработчики, тем выше производительность многоядерной системы.

Первое и самое важное решение, которое разработчики должны принять при переходе к многоядерной технологии — это выбор типа параллельной обработки в соответствии с требованиями приложения. Этот выбор определяет, насколько легко будет обеспечить максимальный уровень параллельной работы нового и существующего кода. Существует два основных типа многопроцессорной обработки: асимметричная многопроцессорность (asymmetric multiprocessing — AMP), симметричная многопроцессорность (symmetric multiprocessing — SMP).

При асимметричной многопроцессорности AMP управление каждым ядром осуществляет отдельная операционная система или отдельная копия операционной системы, общей для нескольких ядер. Обычно каждый программный процесс привязан к одному ядру

(например, процесс А выполняется только на ядре 1, процесс Б — только на ядре 2 и т. д.). Асимметричную многопроцессорность также называют архитектурой с независимыми узлами.

При симметричной многопроцессорности (SMP) управление всеми процессорными ядрами одновременно осуществляет единственная операционная система. Процессы могут динамически перемещаться между любыми ядрами, что даёт возможность полной загрузки всех ядер.

Рассмотрим развитие современных технологий параллельной обработки применением потоковых вычислений на примере одной из популярных моделей многоядерных процессоров – архитектуры Nehalem и ее модели Intel Core i7.

Архитектура Nehalem была разрабатана как архитектура, которая может адаптироваться для нужд всех трех основных рынков: мобильного, настольного и серверного. Nehalem изначально была разработана на модульном принципе — наборе базовых "кирпичей", которые можно собирать, чтобы создавать разные версии архитектуры (рис.11.5).

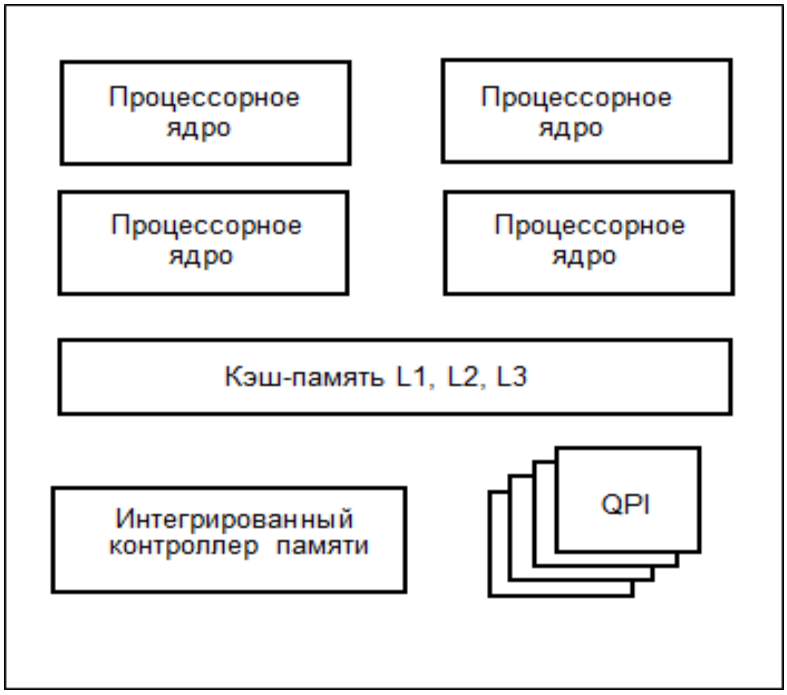


Рис.11.5. Модули реализации архитектуры Nehalem

Базовая архитектура следующая: четырехъядерный процессор, использующий три уровня кэш-памяти, встроенный контроллер памяти, а также высокопроизводительная система интерфейсов

"точка-точка" для связи с периферией и другими процессорами в многопроцессорной конфигурации. Добавлены также новые инструкции SSE.

В архитектуре Nehalem предусмотрен конвейер команд — часть, которая отвечает за считывание инструкций из памяти и подготовку их для выполнения. В конвейере Nehalem присутствуют четыре блока декодирования (три простых и один сложный). Поддерживается функция слияния макроопераций (macro-ops fusion), обеспечивается теоретическая максимальная пропускная способность 4+1 инструкций на ядро за такт. При определении процессором цикла отключаются некоторые части конвейера (чтобы не исполнять пустые операции). Поскольку цикл подразумевает выполнение одинаковых инструкций указанное число раз, отключается выполнение предсказания ветвлений и выборки инструкции из кэша L1 при каждой итерации цикла.

Как и в одноядерных суперскалярных процессорах имеется блок предсказания ветвлений. Эффективность алгоритмов предсказания ветвлений критична для архитектур, где используется высокий уровень параллелизма инструкций. Ветвления разрывают параллелизм, поскольку необходимо ждать результат предыдущей инструкции, прежде чем продолжить выполнение потока инструкций. Предсказание ветвлений прогнозирует, будет взята ветвь или нет, и если ветвь будет взята, то быстро вычисляет дальнейший адрес для продолжения выполнения. Для этого применяется массив (целевой буфер ветвлений), как и в Pentium 4 называемый Branch Target Buffer (BTB), который сохраняет результаты ветвлений по мере продолжения выполнения кода (взята ветвь или нет, а также целевой адрес). К массиву прилагается алгоритм определения результата следующего ветвления.

В архитертуре Nehalem реализована поддержка многопоточности SMT. Поскольку физических ядер на кристалле четыре, некоторые версии Nehalem, которые используют два ядра в одной "упаковке", смогут выполнять до 16 потоков одновременно. Многопоточность реализована по технологии Hyper-Threading.

Технология Hyper-Threading позволяет использовать параллелизм на уровне потоков, чтобы оптимизировать нагрузку

исполнительных блоков ядра, в результате чего на уровне приложений одно физическое ядро превращается в два виртуальных. SMT пытается обеспечить параллелизм инструкций из двух потоков, а не из одного, с целью максимально уменьшить число холостых тактов.

Этот подход оказывается очень эффективным, когда два потока связаны с заданиями разной природы. С другой стороны, если два потока выполняют, например, интенсивные вычисления, это лишь увеличит нагрузку на те же самые вычислительные блоки, которые будут бороться между собой за доступ к кэшу. SMT в данной ситуации может даже негативно повлиять на производительность.

Имеется также модуль интегрированного в процессор интерфейса процессоров с чипсетами и памятью, который представляет собой встроенный контроллер памяти и очень быструю последовательную шину "точка-точка". С технической точки зрения интерфейс является двунаправленным с двумя 20-битными шинами, по одной на каждое направление.

Для связи ядер процессора применяется технология QPI (Quick Path Interconnect). Линии связи имеют по 20 каналов на каждое направление, из которых 16 зарезервировано под данные, а оставшиеся четыре — под функции исправления ошибок или служебную информацию протокола. Это дает максимальную скорость 6,4 ГГб/с или полезную пропускную способность 12,8 Гбайт/с, как на чтение, так и на передачу.

В теоретической ситуации, когда есть только операции чтения или записи, пропускная способность будет идентична FSB. Но следует помнить, что шина FSB использовалась как для доступа к памяти, так и для передачи всех данных на периферию или между процессорами. В случае Nehalem интерфейс QPI исключительно предназначается для передачи данных на периферию, а за работу с памятью отвечает интегрированный в процессор контроллер.

Связь между несколькими CPU в многосокетной конфигурации осуществляется еще одним интерфейсом QPI (рис.11.6). Даже в самой тяжелой ситуации QPI должен показать лучшую производительность, чем FSB.

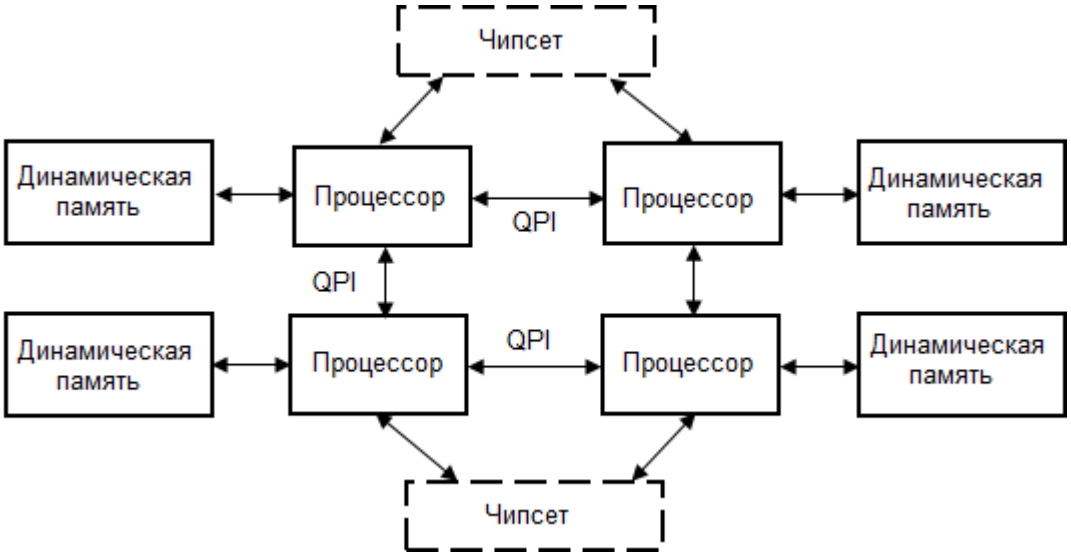


Рис.11.6. Технология связи процессоров QPI

Число доступных интерфейсов QPI меняется в зависимости от ориентации на сегмент рынка — от одного интерфейса для связи с чипсетом (процессорная плата компьютера) в односокетных конфигурациях до целых четырех для четырехсокетных серверов. Это позволяет создавать полносвязные четырехпроцессорные системы, когда каждый процессор может получать доступ к любой области памяти через один QPI, поскольку каждый процессор напрямую подключен к трем остальным.

В архитектуре Nehalem каждое ядро имеет собственный кэш L2 (рис.11.7).

Поскольку он выделен на каждое ядро и относительно мал (256 Кбайт), появилась возможность обеспечить кэш очень высокой производительностью. Присутствует кэш-память третьего уровня (8 Мбайт), отвечающая за связь между ядрами. Если ядро попытается получить доступ к данным и они отсутствуют в кэше L3, то нет необходимости искать данные в собственных кэшах других ядер — там их нет. Напротив, если данные присутствуют, четыре бита, связанные с каждой строчкой кэш-памяти (один бит на ядро), показывают, могут ли данные потенциально присутствовать (потенциально, но без гарантии) в нижнем кэше другого ядра, и если да, то в каком.

Рис.11.7. Организация кэш-памяти ядер процессора Nehalem

Эта техника весьма эффективна для обеспечения когерентности персональных кэшей каждого ядра, поскольку она уменьшает потребность в обмене информацией между ядрами. Кэш третьего уровня работает на других частотах по сравнению с самим чипом, следовательно, задержка доступа на данном уровне может меняться. Здесь она составляет около 40 тактов. Пропускная способность кэша команд L1 составляет 16 байт. Задержка кэша данных достигает четырех тактов, облегчая работу на высоких тактовых частотах. Было увеличено число промахов кэша данных L1, которые архитектура может обрабатывать параллельно.

Как правило, процессоры работают не с физическими адресами основной памяти, а с виртуальными. Такой подход позволяет выделять программе больше памяти, чем есть в компьютере, сохраняя только необходимые на данный момент данные в физической памяти, а все остальное — на жестком диске. При каждом доступе к памяти виртуальный адрес нужно переводить в физический адрес, и для сохранения соответствия приходится использовать огромную таблицу. Для ускорения работы с виртуальной памятью применяется принцип физической адресации с применением небольшой кэш-памяти в процессоре, хранящей соответствие для нескольких недавно запрошенных адресов. Кэш-память называется TLB - Translation Lookaside Buffer.

Таким образом, базовая архитектура Nehalem включает: четырехъядерный процессор, три уровня кэш-памяти, встроенные контроллеры памяти, высокопроизводительный системный интерфейс QPI для связи с периферией и другими процессорами (в случае многопроцессорной конфигурации), двухуровневый TLB, два уровня предсказания ветвлений, поддержку многопоточности Hyper-Threading, а также адаптивную систему управления энергопотреблением. Nehalem представляет собой мощное решение как для персональных компьютеров, так и для серверных и многопроцессорных систем.

11.4. Программная поддержка параллельной обработки

Простота программирования физически параллельных, многоядерных иди многопроцессорных систем выливается на практике в целый ряд сложных методик и условий.

1.Алгоритм должен допускать распараллеливание.

2. При вычленении параллельных участков, как правило, приходится придавать алгоритму специальную форму. Например, если необходимо определить сумму массива, при распараллеливании на первом шаге одновременно суммируются соседние четные и нечетные элементы массива, а на втором попарно суммируются результаты, полученные на первом шаге, и т.д. Компактно записать параллельный вариант на языке Си невозможно. В общем случае приведение алгоритма к форме, позволяющей сократить время вычислений, означает отход от формы, обеспечивающей наиболее наглядное представление.

3. В многоядерной системе разбиение на слишком крупные части не позволяет равномерно загрузить процессоры и внутреннюю память и добиться минимального времени вычислений, а излишне мелкая «нарезка» означает рост непроизводительных расходов на связь и синхронизацию.

4. Физическому параллелизму присуща зависимость глобальной структуры алгоритма от топологии вычислительной платформы. Процесс создания максимально эффективного алгоритма практически не автоматизируется и связан с большими трудозатратами на поиск

специфической структуры алгоритма, оптимальной для конкретной топологии целевой системы. Найденная структура обеспечит минимальное время получения результата, но, скорее всего, будет неэффективна для другой конфигурации. Более суровое следствие зависимости структуры алгоритма от вычислительной платформы — тотальная непереносимость не только исполняемых кодов, но и самого исходного текста.

Важнейшим и одним из наиболее трудоемких этапов создания программы является разработка алгоритма обработки при наличии нескольких процессоров. Процесс разработки параллельного алгоритма можно разбить на четыре этапа.

1. **Декомпозиция.** На данном этапе исходная задача анализируется, оценивается возможность ее распараллеливания. Иногда выигрыш от распараллеливания может быть незначительным, а трудоемкость разработки параллельной программы большой. В этом случае первый этап разработки алгоритма оказывается и последним. Если же применяемый численный метод обработки позволяет разделить задачу и связанные с ней данные на более мелкие части — подзадачи и фрагменты структур данных, это позволяет создать эффективную параллельную программу. Особенности архитектуры конкретной вычислительной системы на данном этапе могут не учитываться.

2. **Проектирование коммуникаций** (обменов данными) между задачами. На этом этапе определяются коммуникации, необходимые для пересылки исходных данных, промежуточных результатов выполнения подзадач, а также коммуникации, необходимые для управления работой подзадач. При выборе алгоритмов и методов коммуникации должна учитываться архитектура многоядерного процессора, особенно разделяемые и неразделяемые вида памяти.

3**. Укрупнение**. Подзадачи могут объединяться в более крупные блоки, если это позволяет повысить эффективность алгоритма и снизить трудоемкость разработки. Основными критериями на данном этапе являются эффективность алгоритма (производительность — в первую очередь) и трудоемкость его реализации.

4**. Планирование вычислений**. На последнем этапе производится распределение подзадач между процессорами.

Основной критерий выбора способа размещения подзадач — эффективное использование процессоров с минимальными затратами времени на обмены данными и максимальное соответствие размеров обрабатываемых файлов размерам неадресуемой памяти.

При переходе на многоядерную архитектуру возникает необходимость поддержания **когерентности** (согласованности) кэш-памяти для всех ядер при использовании общей памяти (shared memory). Кэш-память применяется для ускорения доступа к общей памяти. Теперь в дополнение к поддержанию когерентности между основной памятью и кэш-памятью каждого отдельного ядра необходимо в каждый момент времени поддерживать когерентность основной памяти и кэш-памяти всех ядер, использующих эту общую память при любых операциях чтения-записи. Как правило, эта проблема решается на аппаратном уровне.

Кроме того, не всякая задача допускает физическое распараллеливание. Целые классы задач (например, быстрое Фурье-преобразование, обработка запроса к базе данных, быстрые алгоритмы сортировок, алгоритмы цифровой фильтрации) не получают заметного выигрыша от распараллеливания либо не распараллеливаются в принципе. У разработчиков появляется альтернатива — вместо изменения структуры программы уделять больше внимания локальной оптимизации кода.

11.5. Средства технологий параллельного программирования

При разработке параллельных программ в рамках модели параллелизма задач используются специализированные библиотеки и системы параллельного программирования, Наметилось три основных подхода к реализации этих систем, отличающихся методами взаимодействия параллельных задач. Один подход строится на основе концепции обмена сообщениями (спецификация MPI - Message Passing Interface), второй — на концепции разделяемой памяти (спецификация OpenMP - Open specifications for Multi-Processing), третий – на концепции совмещения инструкций языка

программирования с инструментальными средствами и библиотеками.

На уровне распараллеливания отдельных процедур и алгоритмов (параллельной сортировки, умножения матриц, решения системы линейных уравнений) удобно использовать такую технологию параллельного программирования, как OpenMP.

OpenMP - это набор директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах. В OpenMP используется модель параллельного выполнения "ветвление-слияние". Программа OpenMP начинается как единственный поток выполнения, называемый начальным потоком. Когда поток встречает параллельную конструкцию, он создает новую группу потоков, состоящую из себя и некоторого числа дополнительных потоков, и становится главным в новой группе. Все члены новой группы (включая главный поток) выполняют код внутри параллельной конструкции. В конце параллельной конструкции имеется неявный барьер. После параллельной конструкции выполнение пользовательского кода продолжает только главный поток. В параллельный регион могут быть вложены другие параллельные регионы.

За счет такой постановки процесса разработчик не создает новую параллельную программу, а просто последовательно добавляет в текст последовательной программы OpenMP-директивы.

На данный момент технология OpenMP поддерживается большинством компиляторов языка C/C++ и FORTRAN.

MPI-программа представляет собой набор независимых процессов, каждый из которых выполняет свою собственную программу (не обязательно одну и ту же), написанную на языке C. Процессы MPI-программы взаимодействуют друг с другом посредством вызова коммуникационных процедур. Как правило, каждый процесс выполняется в своем собственном адресном пространстве, однако допускается и режим разделения памяти. MPI не специфицирует модель выполнения процесса - это может быть как последовательный процесс, так и многопотоковый. MPI не предоставляет никаких средств для распределения процессов по

вычислительным узлам и для запуска их на исполнение. Эти функции возлагаются либо на операционную систему, либо на программиста. MPI не накладывает каких-либо ограничений на то, как процессы будут распределены по процессорам, в частности, возможен запуск MPI программы с несколькими процессами на обычной однопроцессорной системе.

MPI - это библиотека функций, обеспечивающая взаимодействие параллельных процессов с помощью механизма передачи сообщений. Это достаточно объемная и сложная библиотека, состоящая примерно из 130 функций, в число которых входят:

- функции инициализации и закрытия MPI процессов;
- функции, реализующие коммуникационные операции типа точка-точка;
- функции, реализующие коллективные операции;
- функции для работы с группами процессов и коммуникаторами;
- функции для работы со структурами данных;
- функции формирования топологии процессов.

Основное применение этой технологии – многопроцессорные системы с распределенной памятью.

Кроме представленных технологий и систем параллельного программирования в настоящее время разработано множество специализированных библиотек, направленных на решение проблемных задач паралелльного программирования.

Intel IPP (Integrated Performance Primitives) — это набор кросс-платформенных библиотек, содержащих большое количество высокопроизводительных функций, которые могут быть использованы для аудио- и видеокодеков (MPEG-4), сжатия изображений (JPEG), обработки изображений (двумерных массивов), обработки сигналов (одномерных массивов или векторов), сжатия естественной речи, систем компьютерного зрения, криптографических приложений. Необходимость создания такого набора функций напрямую вытекает из необходимости разработки прикладного программного обеспечения для систем мультимедиа.

Библиотека MKL (**Intel Math Kernel Library) -** представляет собой набор функций линейной алгебры, быстрого преобразования

Фурье и векторной математики для разработки научного и инженерного прикладного программного обеспечения. Она представлена в вариантах для Windows и Linux, есть версия для Linux-кластеров.

Библиотеки MKL состоят из нескольких двоичных файлов, каждый из которых оптимизирован для определенного семейства процессоров Intel.

На стадии выполнения MKL автоматически определяет модель процессора и запускает соответствующую версию вызываемой функции, что гарантирует максимальное использование возможностей процессора и максимально возможную производительность. Так как библиотека MKL максимально оптимизирована под процессоры корпорации Intel и учитывает их архитектуру, использование компонентов библиотеки более предпочтительно, чем ручное написание кода.

Parallel Extensions to the .NET Framework (PFX) — это библиотека, разработанная фирмой Microsoft для использования в программах на базе управляемого кода. Она позволяет распараллеливать задачи, в которых могут использоваться специальные, координирующие структуры данных. Тем самым библиотека PFX упрощает написание параллельных программ, обеспечивая увеличение производительности при увеличении числа ядер или числа процессоров, исключая многие сложности современных моделей параллельного программирования.

Библиотека **Intel MPI Library -** позволяет разработчику создать эффективную программу для всех многоядерных платформ в виде одного двоичного файла (за счет связывания его с MPI-библиотекой), который может автоматически конфигурироваться для имеющегося на момент выполнения варианта взаимодействия. Библиотека поддерживает файловый ввод/вывод, обобщенные запросы и предварительную поддержку потоков.

Библиотека **Intel Trace Analyzer and Collector** позволяет разработчику анализировать, оптимизировать высокопроизводительные приложения, состоит из сборщика и анализатора трасс. Сборщик взаимодействует с MPI-приложением, собирая информацию о времени выполнения, анализатор служит для

вывода собранной трассировочной информации после выполнения программы с целью ее анализа. Анализатор вычисляет статистику для конкретных интервалов времени, процессов или функций. С целью масштабирования пользователь может перемещаться по разным уровням абстракции данных трассировки: кластер. Узел, процесс, поток, функция.

Рассмотренные инструменты оказывают помощь со стороны производителей для пользователей при кодировке, отладке, настройке и тестировании параллельных приложений на многоядерных процессорах.

Вопросы для контроля

1. В чем суть технологии многопоточности?
2. Каковы преимущества технологии SMT?
3. Опишите подробно технологию Hyper-Threading?
4. Какую технологию обработки реализуют многоядерные процессоры?
5. Рассмотрите уровни параллелизма при обработке данных.
6. Какие задачи решает закон Амдала?
7. Что из себя представляет ядро многоядерного процессора?
8. Перечислите новые технологии в архитектуре Nehalem?
9. В чем оригинальность интерфейса QPI?
10. Перечислите этапы разработки параллельного алгоритма?
11. Каковы инструментальные средства для разработки параллельных программ компьютеров?
12. Рассмотрите подробнее работу пакета OpenMP?

ГЛАВА 12. ПРОЦЕССОРЫ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Цифровые процессоры обработки сигналов (DSP – Digital Signal Processor), или сигнальные процессоры (СП) являются основным обрабатывающим элементом систем реального времени. Системы реального времени — это компьютерные средства контроля и управления объектами или процессами в реальном масштабе времени, то есть процесс сбора, обработки и выдачи информации реализуется в темпе ее поступления от объекта или процесса. Объектом может быть техническая система (робот, автомобиль, самолет, человек), процессом может быть промышленная технология, геофизическое явление, медико-биологический или физико-химический процесс, а также различная аппаратура локальной обработки – студия звукозаписи, система обработки речевых сигналов.

Сигналы являются моделями измеряемого динамического процесса, поэтому они представляются в виде функции с временным аргументом, так как они позволяют определить характер изменения поведения объекта или параметров процесса во времени. После их преобразования из аналоговой (непрерывной) формы в цифровую, дальнейшая обработка является задачей сигнальных процессоров.

Основными задачами цифровой обработки сигналов (ЦОС) являются:

- получение информации об объекте путем измерения параметров сигнала – амплитуды, фазы, частоты, спектра;
- выделение полезных компонент сигнала на фоне помех;
- сжатие (компрессия) сигналов;
- анализ неизвестных сигналов и синтез сигналов с заданными параметрами.

За относительно короткий промежуток времени цифровые процессоры обработки сигналов прошли несколько этапов развития. Первый однокристальный СП (например, TMS32010 фирмы Texas Instruments) имел размер слова 16 бит, разрядность АЛУ - 32 бита, быстродействие 5 млн. операций сложения или умножения в секунду, память 256 слов, ПЗУ программ до 4 килослов, пропускную

способность каналов ввода/вывода 50 Мбит/с, 8 внешних портов по 16 бит.

В последующем в конкурентную борьбу на рынке перспективных электронных технологий вступили такие фирмы, как Motorola, Analog Devices, AT&T, SGS Thomson (США). В результате интенсивных разработок в значительной степени выросли вычислительная производительность и внутренние ресурсы СП, появились мощные программные и аппаратные средства поддержки микропроцессорных систем ЦОС. Уменьшение стоимости и расширение функциональных возможностей процессоров обработки сигналов способствовали широкому практическому использованию методов ЦОС в различных сферах научной и производственной деятельности человека.

Современные СП имеют следующие параметры: тактовая частота – выше 1 ГГц, многоядерная архитектура, наличие двухуровневой кэш-памяти, встроенные многоканальные контроллеры прямого доступа к памяти, быстродействие в тысячи MIPS и MFLOPS.

Традиционно базовыми областями применения техники ЦОС являются цифровая обработка речи, звука, изображений (сжатие, синтез, распознавание, идентификация, закрытие), а также статистическая ЦОС в радиотехнике, связи и управлении (спектральное оценивание, адаптивная фильтрация, цифровая прием/передача).

12.1. Аппаратная реализация и функциональные узлы СП

В цифровых сигнальных процессорах широко используются все известные методы повышения производительности: разделение шин команд и данных, конвейерное выполнение команд, аппаратная реализация программных функций, дублирование функциональных узлов и их параллельное исполнение, встроенная кэш-память, введение специальных команд, ориентированных на цифровую обработку. Кроме этого, СП отличаются разнообразием устройств ввода/вывода, наличием каналов прямого доступа к памяти, встроенными аналого-цифровыми (АЦП) и цифро-аналоговыми

(ЦАП) преобразователями, разнообразными дополнительными модулями.

Таким узлом является **аппаратный умножитель**, выполняющий за один такт операцию перемножения двух операндов (рис.12.1).

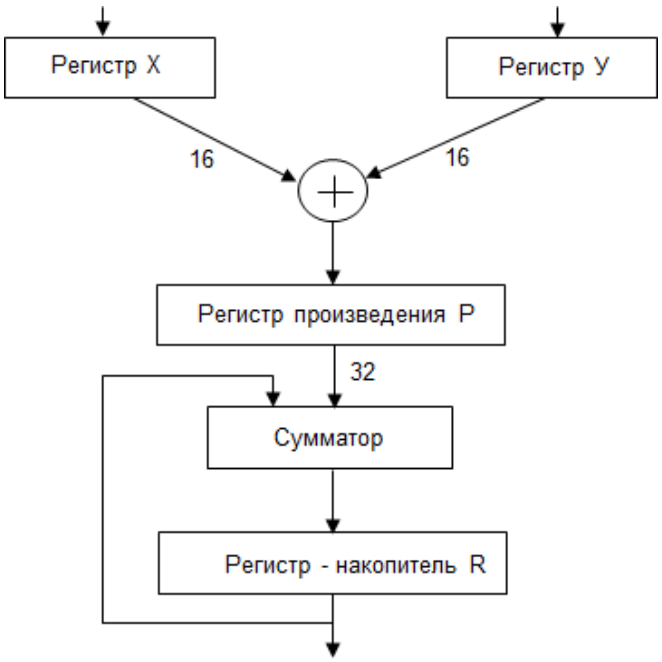


Рис.12.1. Структурная схема умножителя-накопителя

В такой конфигурации умножитель имеет два входных регистра (X и У), которые подключены к входам узла умножения, и 2N-битовый регистр произведения (P) для сохранения результата умножения (N=16 – длина слова процессора). Выход регистра произведения соединяется с накопителем для формирования промежуточных и окончательных результатов выполнения данной операции. Данные на входы умножителя могут поступать как из оперативной памяти, так и из кэш-памяти или внутренних регистров. В отличие от традиционных процессоров умножение выполняется не программным, а аппаратным способом за один такт процессора (обычно за 25нс). Полученное произведение поступает в регистр P. Функции сумматора заключаются в суммировании содержимого регистра P и ранее накопленной суммы (это значение поступает по второму входу сумматора). Полученная в результате общая сумма данного шага выполнения операции формируется в регистре-накопителе.

Схема такого умножителя-накопителя выполняет широко применяемую в обработке сигналов MAC-операцию умножения с

накоплением (MAC- Multiplier-Adder Combination). Эта операция является базовой операцией корреляционного анализа и фильтрации.

Кроме умножителя-накопителя СП имеют в своем составе **аппаратные устройства сдвига.** В принципе, сдвиг операндов влево или вправо на заданное число двоичных разрядов можно осуществлять в АЛУ, однако при этом требуется отдельная команда. Аппаратные сдвигатели СП позволяют производить сдвиг при передаче и загрузке операндов без использования специальных команд. Известно, что сдвиг двоичного числа в сторону старших разрядов дает его удвоение, т.е умножение на 2 в каждом такте, а сдвиг числа в сторону младших разрядов дает деление на 2 в каждом такте. Таким образом, умножение или деление операндов на число кратное 2 может быть обеспечено аппаратным путем с помощью сдвигающих регистров. Это полезная процедура, т.к. размеры файлов обработки, длина слов обработки, сегменты фрагментов сигналов кратны степени двойки.

С точки зрения выполняемых функций сдвигатели делятся на:

- предсдвигатели, выполняющие сдвиг до начала операции или в ходе ее выполнения;

- постсдвигатели, выполняющие сдвиг после операции.

Функции предсдвигателей – это предварительное масштабирование переменных, а также арифметические, логические и циклические сдвиги в ходе исполнения команд.

Функции постсдвигателей – это масштабирование результатов при сохранении в памяти, удаление битов расширения знака, нормализация, выделение одинаковых порядков.

Аккумулятор. Аккумулятор – это регистр, предназначенный для временного хранения промежуточных результатов операций, выполняемых в операционных узлах АЛУ**.** Его функцией при этом является не только временное хранение промежуточных результатов, но и выполнение простейших операций инверсии, сдвига. Аккумулятор связан не только с АЛУ, но и с шиной данных. В АЛУ через один из буферных регистров передается промежуточный результат обработки для последующих шагов цикла, на шину данных передаются для записи в память окончательные результаты обработки.

В архитектуре многих СП предусмотрено два аккумулятора, что позволяет повысить скорость выполнения операций, требующих хранения промежуточных результатов. Технически аккумулятор может состоять из нескольких регистров: регистры расширения, регистры старшего и младшего слова. Их основное назначение – повышение точности вычисления и хранение промежуточных результатов.

Дополнительные арифметические устройства.

Дополнительно к основному АЛУ. выполняющему различные арифметические и логические операции, в процессорах ЦОС применяют вспомогательные арифметические устройства. Они позволяют осуществлять различные математические операции одновременно с основным АЛУ, повышая тем самым производительность системы. Например, в процессорах TMS320 используется второе арифметическое устройство АУВР (арифметическое устройство вспомогательных регистров) для работы со вспомогательными регистрами (рис.12.2).

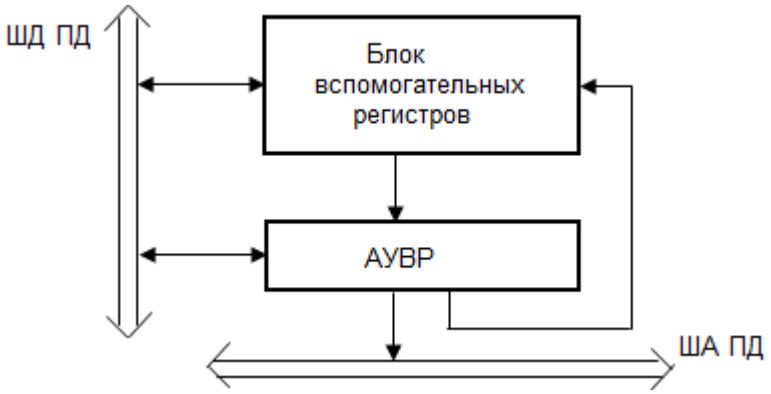


Рис.12.2. Арифметическое устройство с вспомогательными регистрами

Вспомогательные регистры применяются в основном для косвенной адресации данных, а также могут служить в качестве регистров общего назначения. Входной информацией для АУВР может быть содержимое регистров и операнды, передаваемые из памяти данных по шине данных памяти данных ШД ПД. АУВР используется для модификации (изменения) содержимого регистров и получения тех адресов данных, которые передаются на шину адреса памяти данных ША ПД и для сохранения в регистры.

На рис.12.3. приведена функциональная схема умножителя и АЛУ процессоров TMS фирмы Analog Devices. В этих процессорах для накопления результатов умножения (то есть для выполнения MAC-операции) применяется отдельный сумматор — накопитель, который входит в состав устройства умножения-накопления. Результат накопления из регистра MR по специальной шине P (шине результата) передается для дальнейшей обработки в основное АЛУ. Иногда в СП добавляют дополнительное устройство для выполнения только логических операций инверсии, сложения по модулю.

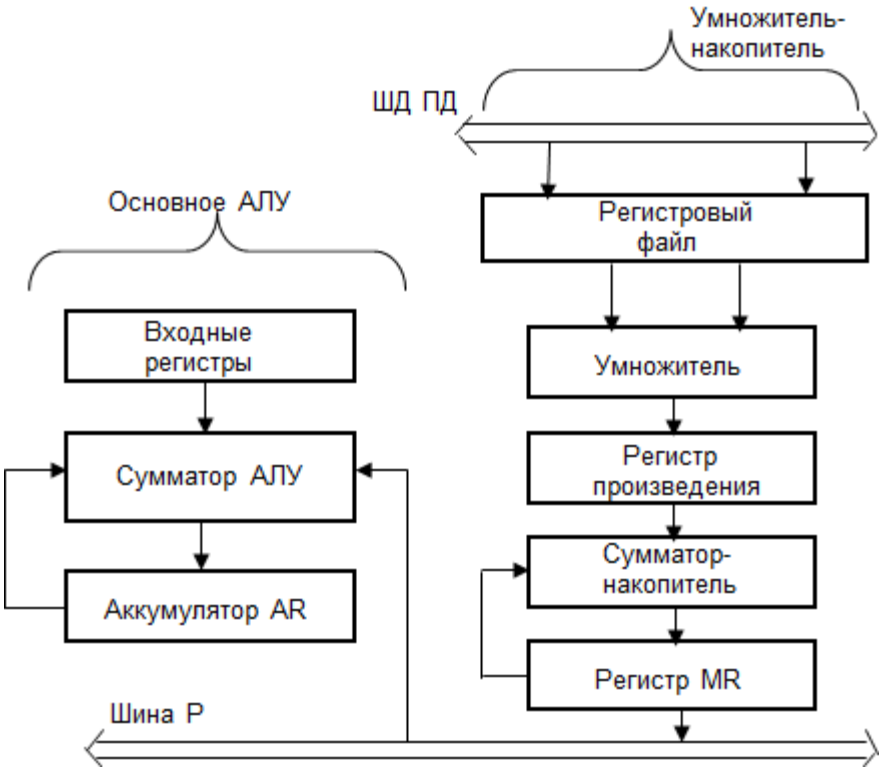


Рис.12.3.Дополнительный умножитель-накопитель в процессорах TMS

Большинство процессоров последних моделей имеют как минимум два аккумулятора, позволяющих сохранять несколько результатов вычислений в АЛУ.

Специализированные устройства генерации адреса.

В процессорах ЦОС используют специализированные устройства для генерации адресов данных (УГА) в памяти данных. Эти устройства формируют или модифицируют адреса для обращения к операндам, размещенным в памяти данных. Формирование адресов при различных методах адресации, особенно специальных, связано с

выполнением вычислений. УГА функционируют параллельно с другими модулями и позволяют одновременно с выполнением операций в АЛУ вычислять адреса операндов для следующей команды. В них используются наборы регистров для хранения адресов и некоторых данных (например, значений индексов или приращений при модификации адреса), а иногда и специализированные АЛУ. Отдельно как устройства генерации адресов данных УГА выделяются в процессорах Analog Devices, Motorola. В этих процессорах используются два устройства, позволяющих формировать одновременно два адреса для двух операндов.

12.2. Особенности архитектуры сигнальных процессоров

Термин «архитектура» используется для описания состава, принципа действия, конфигурации и топологии соединения основных узлов сигнальных процессоров. Понятие архитектура также распространяется на вопросы программирования, форматы представления данных, систему команд, способы адресации, т.е. охватывает весь комплекс аппаратно-программного обеспечения. Ключевым вопросом выбора той или иной архитектуры является высокая производительность, связанная с необходимостью работы в реальном масштабе времени. В свою очередь производительность системы ЦОС определяется возможностями обмена данными между самим процессором и его памятью.

При обработке большого потока входных переменных в системах реального времени скорость обработки должна обеспечить получение текущих результатов в темпе их поступления. В таких системах внутренняя адресуемая память не бывает большого объема, основной упор делается на скорость вычислений и скорость обмена данными между оперативной памятью и АЛУ. Для решения этой проблемы в сигнальных процессора применена известная гарвардская архитектура. Такой подход позволил обеспечить параллельную выборку команды и данных с их одновременной загрузкой в исполнительные устройства. Такты выборки команды и выборки данных исполняются одновременно.

В усовершенствованном варианте гарвардской архитектуры введены раздельные шины для команд и данных, была обеспечена еще более высокая скорость выборки. Дальнейшим развитием гарвардской архитектуры была модифицированная гарвардская архитектура, по которой операнды могут хранится не только в памяти данных, но и в памяти команд вместе с программами. Например, в случае реализации цифровых фильтров коэффициенты фильтра могут храниться в памяти программ, а значения входных сигналов в памяти данных. Коэффициент и данные могут выбираться в одном машинном цикле.

В сигнальных процессорах используется также конвейерная обработка. При конвейерной обработке задача исполнения команды разбивается на несколько этапов реализации. Последовательное соединение этапов и их совмещение во времени выполняется так же, как обычные процессорах. Пропускная способность такой архитектуры определяется числом команд, пропущенных через конвейер в единицу времени. Теоретически среднее время выполнения одной команды вычисляется как время выполнения одной команды деленное на число этапов конвейера. Конвейерная обработка обеспечивает постоянный поток команд на процессор обработки, уменьшается время ожидания отдельных функциональных блоков, увеличивается скорость выполнения операций. При разбиении сложных команд на более простые шаги выполнения количество этапов конвейера может быть 5-10. Такой подход позволяет решить основную проблему конвейерной обработки – различие времени выполнения команд разной сложности.

Применение специальных команд.

В сигнальных процессорах применяют специальные команды, оптимизированные для обработки сигналов. Это команды, поддерживающие базовые операции при обработке:

- оцифровка и запись в память высокочастотных сигналов,
- одновременное умножение с накоплением,
- смещение отсчетов для реализации задержки,
- копирование выборки из памяти.

Имеются также команды, ориентированные на конкретные приложения, например, обеспечения работы в режиме поблочно-

плавающей запятой. Есть также команды, обеспечивающие повторение отдельных команд заданное число раз при организации циклов (в алгоритмах фильтрации или спектрального анализа). Например, в СП марки TMS320C50 команда MAC-D выполняет за один такт несколько действий:

- умножает выборку сигнала из памяти данных на коэффициент, который находится в памяти для хранения программы;
- добавляет предыдущее произведение в накапливающий сумматор,
- реализует единичную задержку отсчета путем его смещения в ячейку со следующим по порядку адресом.

Подобного рода команды эффективно реализуются с помощью аппаратных средств в виде специальных умножителей-накопителей, описанных ранее. Для выполнения последовательно нескольких команд умножения с суммированием произведений вместе с MAC-командой может быть использована команда повторения. Содержимое регистров X и Y перемножается, результат добавляется в накапливающий сумматор, в то же время следующий отсчет сигнала и соответствующий коэффициент читаются их своих регистров для выполнения следующей операции умножения.

Аппаратно-программный параллелизм.

В архитектуре СП для повышения вычислительной эффективности широко используются методы параллельной обработки. Чаще всего используются три метода построения архитектуры:

- архитектура SIMD (Single Instruction Multiple Data);
- архитектура VLIW (Very Long Instruction Word);
- суперскалярная обработка.

Архитектура SIMD позволяет увеличить число математических операций, выполняемых с помощью одной команды. К классу SIMD относятся архитектуры, где множество элементов данных подвергается параллельной, но однотипной обработке. В этом случае общая команда транслируется множеству
процессоров (в простейшем случае АЛУ), каждый из которых обрабатывает свои данные. Такая архитектура может увеличить количество необходимых операционных блоков при изменении

форматов обрабатываемых данных (например, обрабатывать 4 числа по 16 бит или 2 числа по 32 бит). Пример такого операционного блока для SIMD-операций приведен на рис.12.4.

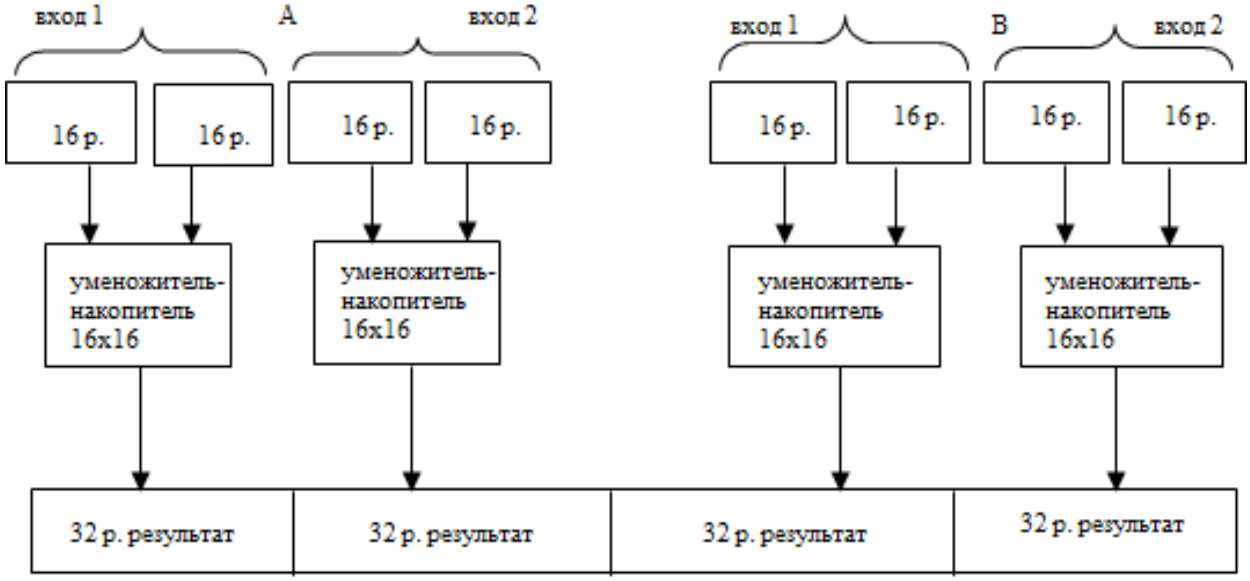


Рис.12.4. Одновременная реализация четырех операций в СП TigerSHARC

Это элемент процессора TigerSHARC, который может одновременно выполнять четыре операции умножения/накопления (умн/нак) формата 16х16 бит. Процессор имеет два операционных блока «А» и «В», на вход которых может поступать два 64-битных числа с шины данных. Каждый из четырех умножителей имеет два 16-битных входа и один 32-битный регистр результата.

Такой подход к организации обработки широко применяется в так называемых систологических процессорах, специально предназначенных для реализации быстрых алгоритмов Фурье-анализа. Кроме систологических процессоров SIMD-архитектура используется в векторных и матричных процессорах, где реализуются скоростные операции умножения векторов и матриц.

Архитектура SIMD реализована практически во всех моделях современных СП: DSP1600, TMS320C62, ADSP-TS001.

Обработка с **командными словами большой длины (VLIW)** позволяет существенно увеличить число команд, обрабатываемых за один такт. Такие команды являются сочетанием нескольких коротких команд, для выполнения которых необходимо несколько параллельно функционирующих блоков. При этом, в отличие от SIMD-

процессоров, VLIW-процессоры обрабатывают в одном такте сразу несколько различных по исполнению команд.

VLIW-технология базируется на том, что компилятор вначале исследует исходную программу, ищет команды, которые могут быть выполнены одновременно, но на различных исполнительных блоках. Затем компилятор объединяет такие команды в пакеты. Каждый пакет представляет собой одну сверхдлинную команду, состоящую только из простых команд, выполняемых одновременно на разных исполнительных блоках процессора. При этом количество простых команд в одной сверхдлинной команде равно числу имеющихся в процессоре функциональных блоков. На рис.12.5 представлена схема обработки данных в процессоре TMS320C62x.

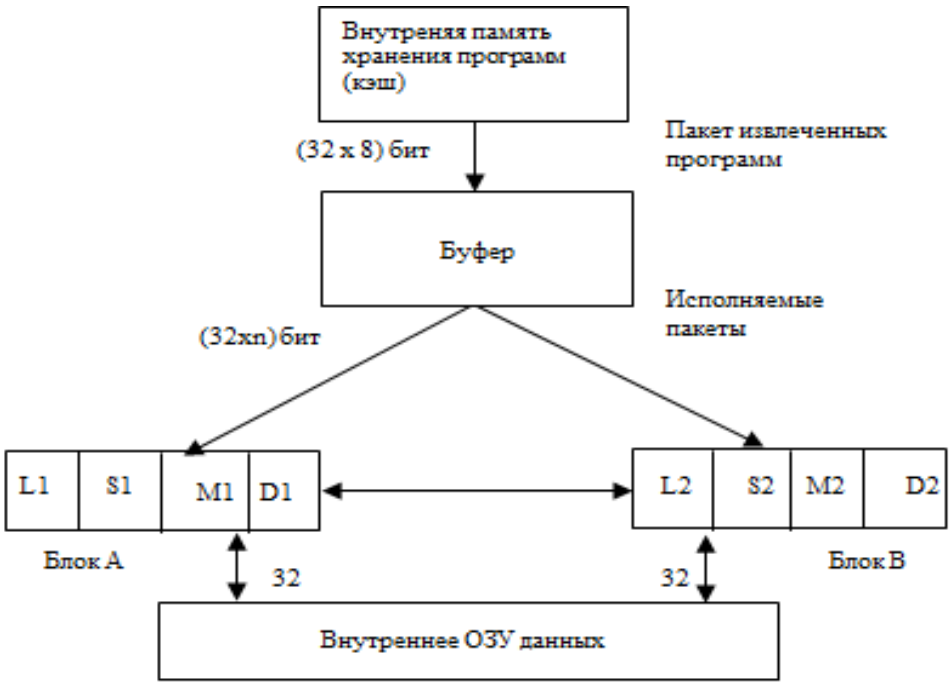


Рис.12.5. Схема потока команд и данных VLIW-процессора

Процессор содержит два тракта передачи данных и восемь независимых операционных блоков, организованных в два узла (блок «А» и бок «В»). Обработка начинается с формирования компилятором пакета команд и чтения команд пакета из внутренней кэш-памяти. Длина пакета – восемь команд по 32 бита каждая. Пакет в 256 бит передается через буфер на восемь операционных блоков различного назначения: L1, L2 – блоки выполнения логических операций, S1, S2 – схемы сдвига, M1, M2 – умножители, D1, D2 – адресные элементы.

Результаты размещаются во внутреннем ОЗУ данных и могут быть использованы в последующих тактах обработки.

Принцип использования компилятора для предварительной подготовки пакета коротких команд требует достаточно сложных «интеллектуальных» компиляторов, однако обеспечивает параллелизм на уровне команд и очень высокую скорость обработки данных, особенно для простых команд программы.

Суперскалярная обработка – это тоже технология увеличения числа одновременно обрабатываемых команд за один такт. Суперскалярные процессоры содержат несколько полноценных операционных блоков для параллельной обработки сразу нескольких команд. Суперскалярная обработка – это объединение технологий SIMD и VLIW.

На рис.12.6 представлена структурная схема узлов обработки и каналов передачи данных супескалярного СП марки TS201S (серии моделей TigerSHARC).

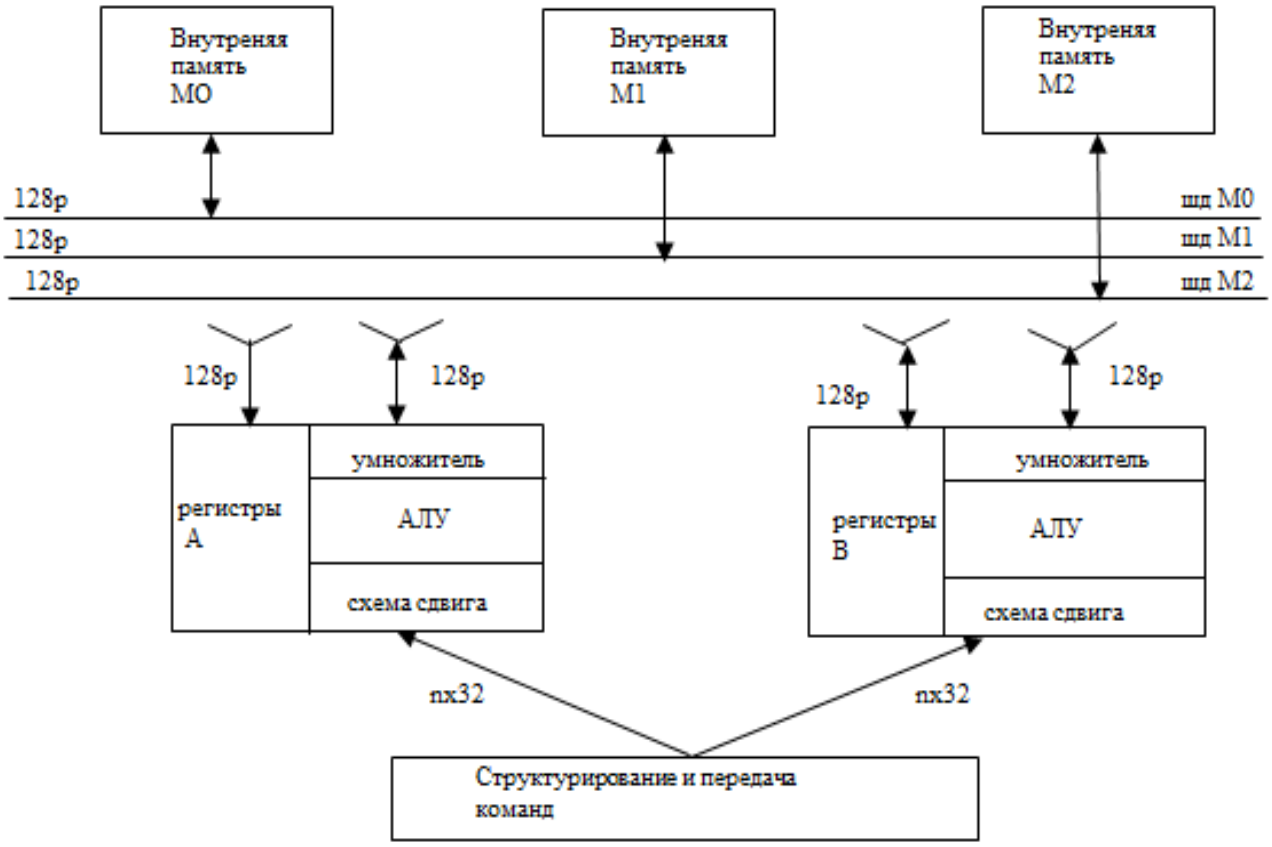


Рис.12.6. Архитектура и поток данных в процессоре TS201S

Имеется три тракта передачи данных по 128 разрядов, два независимых операционных блока (АЛУ, умножитель, схема сдвига), три канала передачи адресной информации по 32р (на схеме не показаны). СП может работать в форматах 8,16 и 32 бит.

В каждом такте из памяти считывается до 4-х 32-битовых команд, которые передаются независимо друг от друга операционным блокам, построенным по архитектуре SIMD (один поток команд - множество потоков данных). Каждый их операционных блоков работает со своим блоком регистров «А» или «В». Команды могут передаваться как одновременно двум блокам, так и каждому блоку независимо. Блоки регистров соединены с тремя трактами передачи данных так, что могут одновременно в одном такте работы процессора считывать два числа из памяти и записывать один результат в память. Это согласуется с базовыми операциями ЦОС – два входных значения и одно выходное.

Процессор может в одном такте выполнять до восьми операций сложения/вычитания и восемь операций умножения/накопления с 16-бит входными данными (или две операции умножения/накопления с 32-бит входными данными).

Возможность обработки смешанных типов данных и разбиения больших командных слов на отдельные команды для операционных блоков позволяет процессору интенсивно использовать параллелизм на уровне команд. Эффективность использования таких возможностей процессора зависит от планирования команд компилятором перед выполнением программы и наличием механизмов управления при ветвлении программы.

Первым шагом перехода к параллелизму в **организации памяти** была гарвардская архитектура процессоров, когда произошло разделение памяти команд и памяти данных. В последующем единая память данных была разделена на самостоятельные «память данных X» и «память данных У». Такое построение памяти оказалось удобным для графических приложений (координаты X и У), при реализации фильтрации и выполнении алгоритмов спектрального анализа – основных процедур обработки сигналов.

В последующих моделях сигнальных процессоров появилась кэш-память, использовалась также двухвходовая память. Встроенный

контроллер прямого доступа к памяти позволил совместить во времени сами вычисления и процедуры обмена данными с памятью.

Дальнейшим развитием принципа параллелизма памяти было широкое использование **регистровой памяти** в различных вариантах. Наиболее часто в архитектуре используются регистры общего назначения, где хранятся промежуточные результаты обработки. Многофункциональными регистрами являются аккумуляторы, накопители, сдвигающие регистры, регистровые файлы.

Примером практического использования параллелизма в памяти является используемый при циклической адресации циркулярный буфер - набор ячеек в памяти данных, обращение к которым производится по циклу, т.е. обращение к записанным в буфере данным производится по замкнутому кругу в одном и другом направлении. Такой циркулярный буфер может использоваться также для организации линии задержки.

Параллелизм в организации шин процессора.

Известно, что основной проблемой является высокое быстродействие процессора и относительно низкая скорость выборки данных из оперативной памяти. Для устранения этой проблемы и разработаны более скоростные виды памяти: регистровая память**,** кэш-память различных уровней, двухвходовая память. Эти приемы ускоряют процесс обработки, но они малоэффективны без соответствующих каналов передачи информации – процессорных шин.

Сигнальные процессоры работают с входными потоками данных, для которых скорость работы системы упирается в скорость обмена данными с конечным ОЗУ. Решение проблемы чтения и записи данных для процессора заключается в правильной организации шин памяти, в сокращении циклов доступа в память. Параллелизм в организации линии взаимодействия «процессор-шина-память» позволяет оптимально загрузить ядро обработки без чрезмерного усложнения памяти.

По мере развития архитектуры СП возрастало как количество самих шин, так и их разрядность (ширина). Разделение функций шин соответствовало разделению функций элементов памяти. Кроме традиционных шины адреса, шины данных и шины управления

появились отдельные **шины данных команд и шины данных операндов, шины данных X и шины данных У.** Это позволило не только оптимально загрузить сам процессор данными, но и обеспечить информацией дополнительные функциональные узлы: АЛУ для фиксированной и плавающей запятой, аккумуляторы и сдвиговые регистры, модули MAC-обработки. На базе параллелизма в структуре шин появилась возможность ввести режим конвейерной обработки, использовать SIMD и VLIW архитектуры. Для поддержки суперскалярной архитектуры и обеспечения необходимого потока данных, например, в процессоре Tiger SHARC имеется три шины данных по 128 бит каждая.

Организацию взаимодействия шин и функциональных узлов можно увидеть на примере структуры ядра обработки сигнального процессора TMS320C55x (рис.12.7).

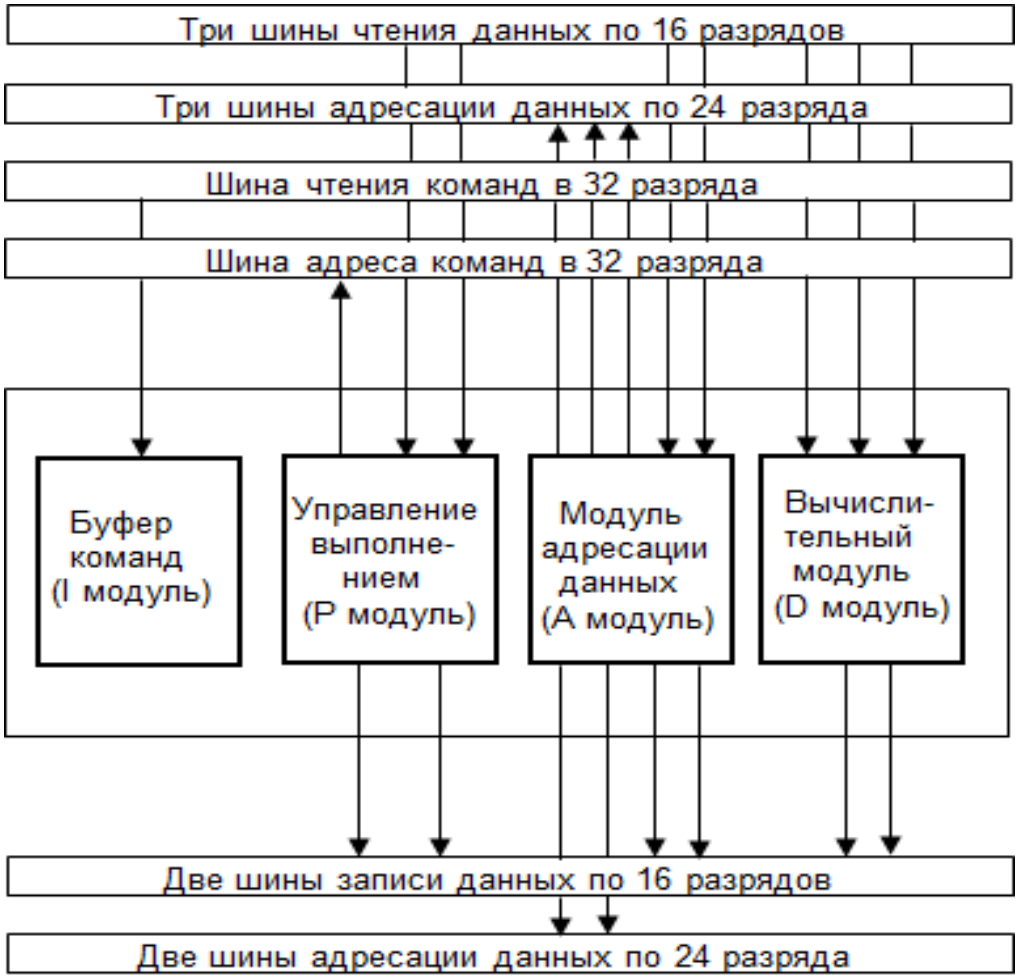


Рис.12.7. Структура процессорного ядра модели TMS 320C55x

Ядро процессора построено по гарвардской архитектуре и содержит 4 функциональных модуля: вычислительный модуль (D); модуль адресации (A); модуль декодера команд (I); модуль управления выполнением (P).

Модули объединены между собой и ресурсами памяти 12 шинами.

Архитектура шин может обеспечить одновременную выборку трех 16-ти разрядных слов и запись двух 16-ти разрядных и одного 32-разрядного слова.

Параллелизм на уровне команд обработки данных.

Большое количество разнообразных дополнительных к основному АЛУ функциональных узлов дает возможность производить в процессорах одновременно несколько действий. Это, в свою очередь, предоставляет возможность ввода и широкого использования **комбинированных команд**, осуществляющих одновременно несколько действий. Комбинированные команды применяются в процессорах среднего класса с относительно невысокой производительностью. В мощных высокопроизводительных процессорах выбрана архитектура с сокращенным набором упрощенных команд типа «регистр – регистр».

Комбинированные команды, прежде всего, используются для выполнения основной операции цифровой обработки сигналов — **умножения с накоплением и различных ее вариантов.** Другим примером комбинированных команд могут служить команды, связанные с действиями в АЛУ, умножителе и с двигателе. В соответствии с такой командой будет выполнено изменение содержимого регистров при выполнении условия, которое является необязательным элементом команды (например, связанное с битом переноса).

12.3. Специализированные СП систем реального времени

Одной из основных особенностей сигнальных процессоров, работающих в системах реального времени, является сочетание алгоритмов цифровой обработки сигналов с общими функциями управления аппаратурой, т.е. внутренними и внешними аппаратными узлами. Технологические особенности СП учитывают области их

применения, поэтому в системах реального времени к ним предъявляются очень жесткие требования по надежности, минимальным габаритно-весовым показателям, минимальной потребляемой мощности и удобным средствам отладки программ. Основное требование к аппаратно-программному обеспечению систем реального времени – высокая скорость обработки при сохранении требуемой точности.

На ранней стадии создания аппаратных средств ЦОС выпускались специализированные процессоры с ориентацией на конкретные алгоритмы обработки. Так как самым распространенным был алгоритмы спектрального анализа, то появилось множество вариантов исполнения Фурье-процессоров как в виде отдельных микросхем и даже вычислительных блоков, так и в виде встраиваемых внутрисхемных узлов. Такие процессоры имели оригинальную архитектуру и выполняли только один алгоритм преобразования. Следующим шагом было создание специализированных процессоров с систологической архитектурой. Они включали множество параллельно функционирующих процессорных элементов – сумматоров на два входа, которые связывались определенным образом и реализовали основные алгоритмы быстрых преобразований или цифровой фильтрации (например, свертку). В более поздних вариантах аппаратной реализации алгоритмов ЦОС в качестве дополнительных внутренних узлов стали применяться матричные умножители, которые с высокой скоростью выполняли процедуры умножения векторов и матриц в алгоритмах свертки цифровых последовательностей, а также в алгоритмах Фурье-анализа.

Однако, по мере расширения сферы применения СП, появления новых, оригинальных методов и алгоритмов, отпала необходимость построения архитектур процессоров с различными специализированными узлами. Для технологии БИС более эффективным оказалось использование новых архитектурных решений на уровне процессора в целом.

Специализированные СП по своим функциональным возможностям и характеристикам относятся к категории младших

моделей процессоров цифровой обработки сигналов. Они специализируются не по архитектуре, а по применению.

Старшие модели СП имеют универсальное назначение и реализуются на основе суперскалярных архитектур с использованием сверхдлинных команд (VLIW-архитектура), имеют структуру SIMD («одна команда – множество данных»), реализуют конвейерную обработку и потоковые вычисления. Они имеют длину слова не менее 32 разрядов, достаточно развитую систему команд, развитый внутренний (шинный) и внешний интерфейс. Последние модели универсальных СП имеют уже **многоядерную архитектуру**, в которой присутствуют несколько параллельно функционирующих, полноценных обрабатывающих узлов.

Младшие модели имеют, как правило, специальное назначение и применяются в системах реального времени при обработке аудио и видеосигналов, в бытовых приборах и системах управления промышленными агрегатами, в системах контроля и мониторинга, мобильной связи, в телекоммуникационной аппаратуре. Рассмотрим некоторые варианты использования младших моделей СП.

Сегодня сигнальный процессор основной элемент практически любого вида бытовой техники. В сотовых телефонах СП формирует ядро процессора связи, выполняя функции аудиокодека, в интерфейсе цифровой абонентской линии связи или в беспроводном маршрутизаторе, СП скрыт в модеме. В каждой портативной мультимединой системе обработки и сжатия изображения – от простых медиаплееров и многофункциональных устройств воспроизведения, таких как iPod, до цифровых видеокамер и камкодеров – везде можно найти специализированные микросхемы СП. Их называют видеопроцессорами или видеокодерами.

Требования к таким сигнальным процессорам постоянно ужесточаются. Переход к средствам отображения видеоинформации высокой точности потребовал увеличения тактовой частоты процессоров изображения, особенно тех, которые используются в мобильных устройствах и телевизионных приставках. А к сигнальным процессорам для аудиосистем с высоким разрешением, в частности для систем "окружающего", пространственного звука, предъявляются требования повышенной точности и скорости.

Расширяются и требования к СП для таких относительно простых систем, как цифровые фотоаппараты с функциями стабилизации изображения и распознавания лица. Сигнальные процессоры необходимы и для телевизионных приставок, IP-телефонов, фотоаппаратов, игровых систем и медиаплееров, подключаемых в различных сочетаниях к Интернету для обмена информацией.

Последние годы в системах ЦОС стали часто применяться программируемые логические интегральные схемы (ПЛИС), специально разработанные для задач обработки сигналов. Они могут работать как автономные узлы обработки или в качестве дополнительных сопроцессоров к обычным СП. ПЛИС отличаются гибкой архитектурой, высоким уровнем параллелизма, достаточной производительностью для систем, выпускаемых малой или средними сериями.

Более простые модели ПЛИС позволяют создавать схемы, эквивалентные сотне корпусов обычных интегральных схем. На них трудно реализовать сложные алгоритмы ЦОС, поэтому они перспективны в интерфейсных схемах сопряжения СП с реальной аппаратурой. Для более сложных задач лучше подходят ПЛИС с архитектурой программируемых самим пользователем вентильных матриц. Такие матрицы имеют тактовую частоту до 300 МГц и возможность реализации сложных параллельных алгоритмов.

Архитектурные особенности ПЛИС эффективно реализуют алгоритмы умножения, свертки, фильтрации. Например, ПЛИС семейства FLEX 6000 фирмы Altera позволяют реализовать алгоритмы БПФ, БИХ и КИХ-фильтров высокого порядка, при этом обеспечивают намного более высокую скорость, чем СП.

Примером использования специализированного процессора в звуковых панелях домашней аппаратуры может быть аудиопроцессор TMS320DA710. В прошлом для виртуализации использовалось несколько СП, которые обеспечивали декодирование, виртуализацию и различные аудиоэффекты. Современная интеграция устройств на одном кристалле, при которой объединяются аналоговые и цифровые компоненты с аудиопериферией, позволяет одному процессору обеспечивать функционирование всей системы звуковой панели.

На рисунке 12.8 показана блок-схема аудиопроцессора TMS320DA710. Это мощный 32-разрядный специализированный процессор с тактовой частотой 300 МГц. Он поддерживает операции с плавающей точкой и обладает производительностью 1800 MFLOPS. Процессор служит для обеспечения функций декодирования и кодирования многоканальных аудиоданных высокого разрешения с их одновременной обработкой через соответствующий аудиоинтерфейс.

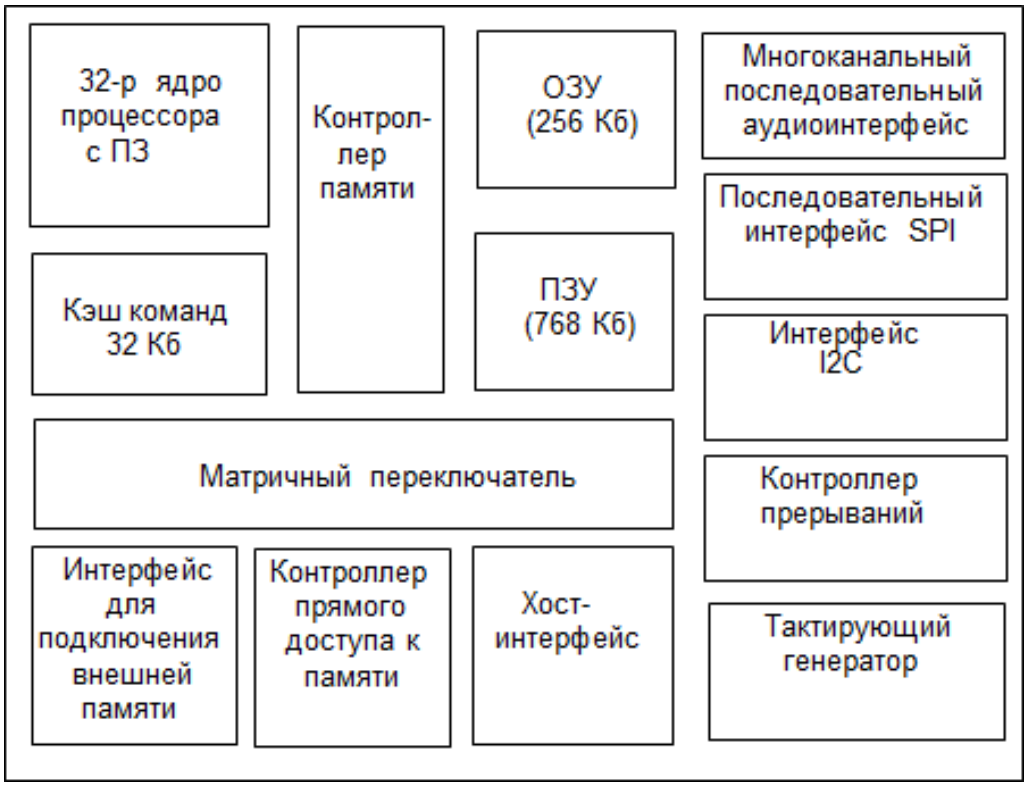


Рис.12.8. Блок-схема аудиопроцессора TMS320DA710

Для программирования процессора предусмотрен комплект средств программного обеспечения DA7x SDK (Software Devetlopment Kit), который не требует от разработчиков звуковых панелей навыков программирования процессора, т.к. программирование реализовано в графической форме.

Специализированные процессоры рассмотренного типа широко применяются в других типах электронной аппаратуры. Современные высокопроизводительные модемы используют средства цифровой обработки для выполнения функций модуляции, демодуляции, обнаружения и исправления ошибок, настройки параметров передачи.

Развитые системы видеонаблюдения постепенно переходят на IP-технологии, которые в отличие от старых систем видеонаблюдения (в составе аналоговой камеры, монитора и видеомагнитофона) возлагают на камеры функции приема изображения, его детектировании, отслеживания траектории движения объекта наблюдения, оповещение пользователя при возникновении определенных событий, сжатия сигнала и его передачи в сеть. К некоторым камерам возможно подключение жесткого диска и карт памяти для сохранения видеоинформации.

Использование сигнальных процессоров в видеотехнологиях и мультимедийных системах промышленного применения породило идею коммуникации простых специальных процессоров и их размещения на одном кристалле с высокой степенью интеграции (термин «система на кристалле»). В этом случае в систему должны быть интегрированы интерфейсы камеры, сенсорных и жидкокристаллических дисплеев, а также распространенных в компьютерной индустрии интерфейсов: высокоскоростного USB 2.0, карт памяти SD, а также интерфейс Gigabit-Ethernet. Важно, чтобы система могла поддерживать аудио- и видеофункции. Это могло быть осуществлено с помощью высокоскоростных процессоров с соответствующими программными кодеками, однако данное решение лишь в малой степени отвечают современным требованиям по скорости. Лучшим решением являются процессоры, в которых в микрочип встроен специальный аппаратный ускоритель.

Компания Renesas разработала специальное семейство 32-битных RISC-процессоров SH772x с низким потреблением энергии для мультимедийных приложений. В данном случае речь идет о решениях в виде мультимедийных «систем на кристалле» с высокой степенью интеграции для обработки аудио- и видеосигналов и восприятия речи, а также для графического ускорения. В такой интегрированной системе компоненты хоть и обладают высокопроизводительным суперскалярным процессорным ядром, но функции мультимедиа обрабатываются частично независимыми периферийными модулями, которые соединены с ядром согласно модели быстрых шин (рис.12.9).

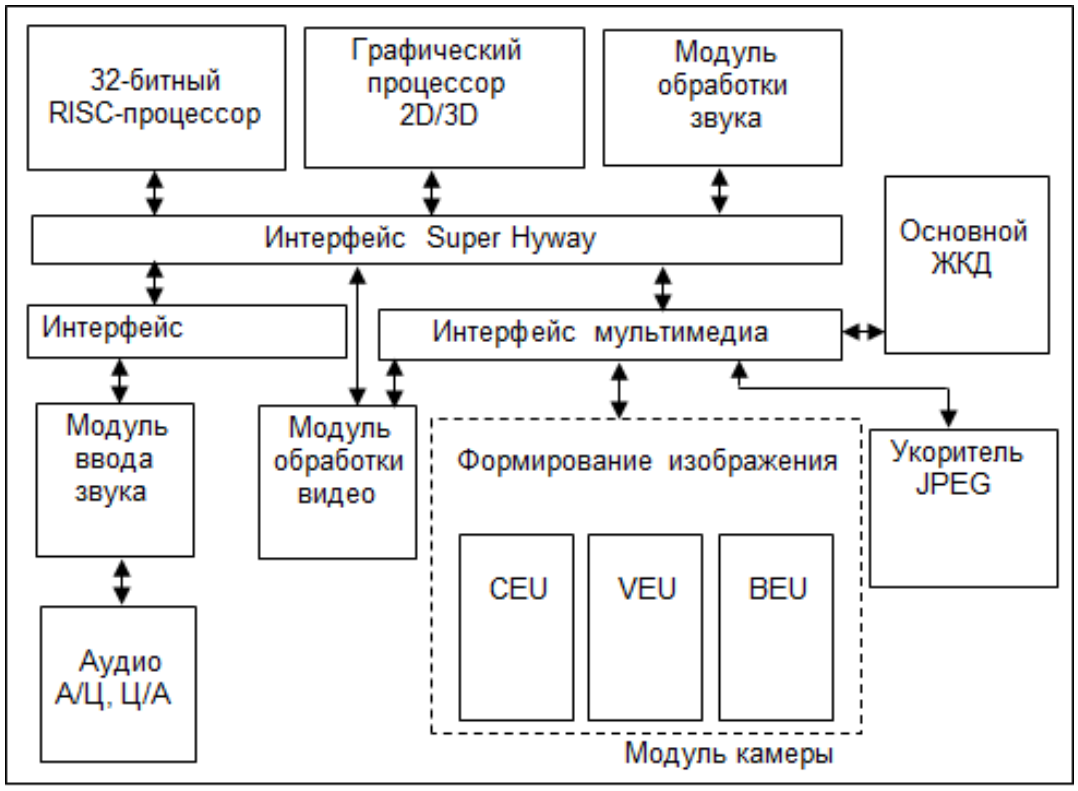


Рис.12.9. Архитектура мультимедийного процессора SH772x

Функциональные возможности этих периферийных процессорных модулей достаточно широки. Модуль видеокамеры предназначен для предварительной обработки видеосигналов и содержит процессор ввода изображения (CEU), модуль фильтрации и изменения размера (VEU), процессор плавного перехода между фрагментами (BEU). В кристалл системы интегрированы также самостоятельные процессорные модули обработки видео, звука, кодирования и декодирования сигналов, графический процессор, жидкокристаллический дисплей (ЖКД). Коммуникация между отдельными блоками обработки осуществляется с помощью шины Super Hyway. Эта двунаправленная шина позволяет передавать данные со скоростью до 3,2 Гбит/с при частоте 200 МГц и ширине 64 бит. Шина поддерживает пошаговую передачу, пакетную передачу, а также передачу маршрутной информации.

Описанная архитектура специализированных микрочипов со встроенными модулями обработки оптимально подходит для использования в портативных проигрывателях и навигационных системах, промышленных компьютерах с полноценными дисплеями, камерах систем безопасности, терминалах видеосистем.

Специализированные или заказные микрочипы занимают большую долю рынка электронных компонентов, однако вследствие их узкой специализации и невозможности перепрограммирования встроенных функций они затрудняют модернизацию аппаратуры и добавление новых возможностей. По сравнению с ними важнейшее преимущество универсальных программируемых сигнальных процессоров – это возможность их применения в различных по назначению устройствах и сокращение сроков их выхода на рынок. Рассмотрим архитектуры универсальных СП ведущих компаний.

12.4. Модели современных сигнальных процессоров компании Texas Instruments

Различные приложения, в которых применяется цифровая обработка сигналов, предъявляют различные, часто противоречивые требования к аппаратуре обработки. Поэтому производители аппаратных средств вынуждены выпускать широкий набор различных моделей СП, чтобы максимально удовлетворить требованиям потребителей. Ведущие мировые производители цифровых сигнальных процессоров: Texas Instruments, Analog Devices, Motorola создают свои модели СП по отработанному технологическому принципу. Вначале создается аппаратное ядро обработки – базовая модель основного процессора с необходимы набором элементов памяти, интерфейса, управления и соответствующим набором команд. На основе процессорного ядра разрабатывается модельный ряд СП, отличающий набором дополнительных модулей для конкретных применений. В пределах модельного ряда СП совместимы по программному коду, среде разработки и отладки программ. Такой подход позволяет быстро создавать требуемые варианты аппаратуры СП для различных сфер применения, не изменяя базовой архитектуры ядра обработки.

Существуют различные классификации СП в соответствии с их архитектурой, форматом слова, назначением и многими другими показателями. Эти показатели в той или мере рассмотрены в предыдущих разделах, поэтому наиболее целесообразно рассмотреть современные СП с точки зрения областей их наиболее эффективного применения. Учитывая, что ведущие компании в целом разделили

между собой рынки сбыта СП, рассмотрим их особенности построения с точки зрения типов оказываемых услуг и класса решаемых задач. Наибольшее внимание будет уделено СП с универсальной архитектурой. Универсальные СП – это высокоскоростные сигнальные процессоры с гарвардской архитектурой, высоким уровнем параллелизма, использованием конвейерной обработки и наборами команд, оптимизированными под операции ЦОС.

Модели сигнальных процессоров компании Texas Instruments.

Компания Texas Instrument (TI) объявила концепцию трех платформ (на базе трех типов ядер обработки), т.к. одна архитектура СП не в состоянии удовлетворить всем противоречивым требованиям при решении широкого класса задач. Любая архитектура не в состоянии обеспечить высокую производительность, минимум потребляемой мощности, максимальную функциональную полноту и низкую стоимость одновременно для любых прикладных задач.

С целью оптимизации архитектуры компания TI объявила о создании трех независимых платформ СП: TMS320C2000, TMS320C5000 и TMS320C6000. В пределах каждой из платформ процессоры совместимы по коду, при этом в распоряжение пользователя предоставлена единая среда разработки и отладки программ. Каждая платформа имеет свою область применения, мало пересекающуюся с другими платформами.

Платформа C2000. Модели этого ряда СП применяются в робототехнике, промышленных автоматах, жестких дисках, оптических сетях в качестве коммутирующих элементов. Процессоры данным моделей обладают гибкостью RISC-архитектуры, достаточной вычислительной мощностью и простотой программирования.

В платформу входят два семейства процессоров: C24x и C28x. Рассмотрим последнюю модель, которая имеет более десяти разновидностей (различие в объемах памяти и периферийных узлах).

Ядро процессора C28x имеет 32-разрядную гарвардскую архитектуру, разделенные шины программ и данных. Шина данных, в свою очередь, разделена на шины чтения и записи в целях увеличения

пропускной способности, ввиду обработки необходимости обработки данных в реальном времени. Процессор позволяет выполнять за один машинный цикл с помощью матричного умножителя 32x32-разрядную MAC-операцию, оперировать с 64-разрядными числами результатов.

Высокая разрядность при фиксированной точке позволяет иметь достаточный динамический диапазон. В составе данного семейства СП есть модели, в которых специально введен процессор плавающей запятой. Процессор выполняет операции чтения, обработки и записи в память за один машинный такт за счет атомарного АЛУ. Производительность увеличена также за счет применения 8-ми ступенчатого конвейера. Оперативная память поделена на несколько областей, каждая из которых может быть использована как память программ, либо как память данных.

Помимо оперативной памяти имеется Flash-ПЗУ объемом до 128 килослов по 16 разрядов. Производительность ядра процессора составляет 150 MIPS при тактовой частоте 150 МГц.

Семейство C28x обладает широким набором периферии, включающим АЦП на 16 каналов по 12-разрядов, большой набор таймеров, раздельных портов ввода/вывода, набор коммуникационных портов. Программирование выполняется на C++.

На рисунке 12.10. представлена схема СП TMS320F28335, входящего в семейство C2000 (буква F свидетельствует о наличии в составе основного ядра отдельного процессора плавающей запятой).

Ядро процессора с производительностью 300 MFLOPS при частоте 150 МГц. включает 32-разрядный умножитель, атомарное АЛУ (одновременно считывает, умножает и записывает результат), процессор плавающей запятой, три таймера по 32 разряда, встроенный модуль отладки JTAG.

Подсистема памяти включает Flash-память объемом 512 Кбайт, память RAM объемом 68 Кбайт и память ROM объемом 68 Кбайт.

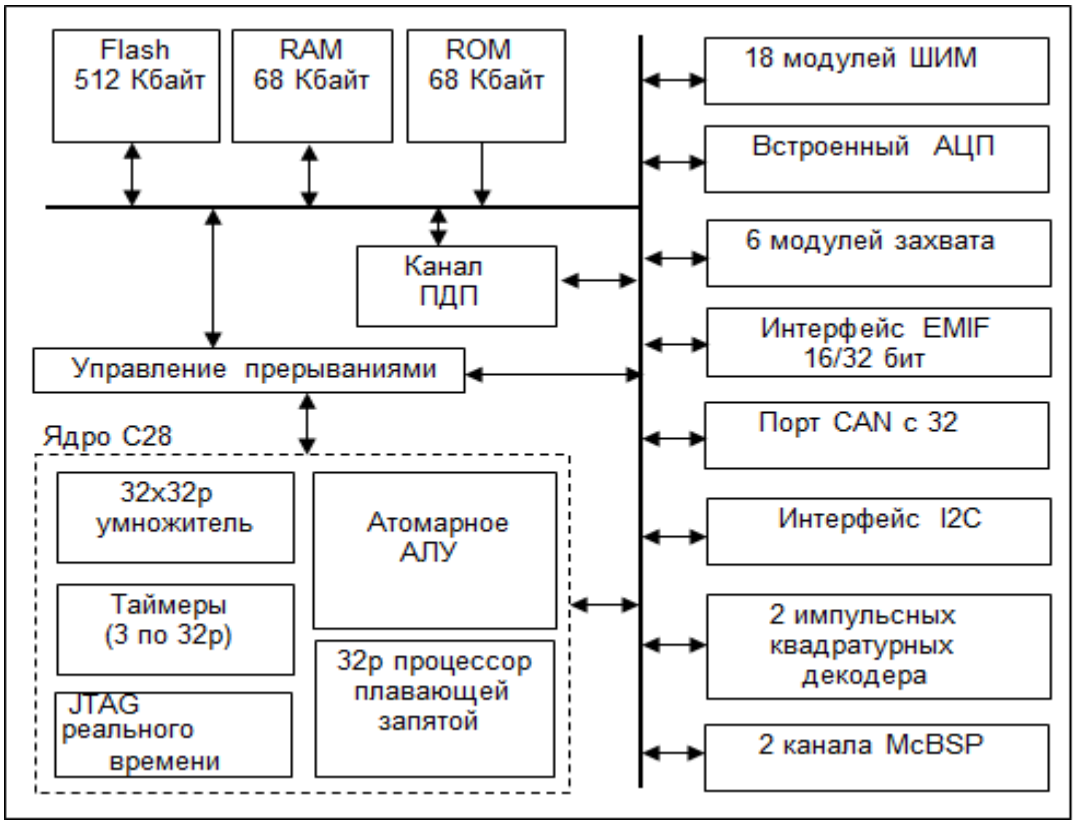


Рис.12.10. Блок-схема процессора TMS320F28335

В составе периферийных модулей: 18 каналов приема-передачи с широтно-импульсной модуляцией (ШИМ), высокоскоростное АЦП (16 каналов, 12 разрядов, время преобразования 80 нс), 6 модулей захвата, интерфейс EMIF с возможностью конфигурации в 16 или 32 разрядном режиме, 6 каналов прямого доступа к памяти (ПДП), поддерживающих интерфейс EMIF, два канала интерфейса многоканального буферного последовательного порта McBSP, порт CAN с 32 почтовыми ящиками (mailboxes), интерфейс I2C со скоростью 400 кбит/с.

Фирма TI предлагает пользователям более десяти вариантов построения таких СП. К стандартному набору ядра базового процессора C28x добавлены команды поддержки операций с плавающей запятой. Это не повлияло на полную совместимость программ для C28x и F28335, а также других моделей данного ряда СП.

Платформа C5000. C5000 – это 16-ти разрядные СП с фиксированной точкой. Предназначены для мобильных средств телекоммуникаций, измерительных устройств, медицинской техники.

Основные технологические преимущества – высокая скорость при малой потребляемой мощности.

Первые модели данного семейства процессоров (например, C54xx) заняли большой рынок мобильной телефонии и использовались в аппаратах фирм Nokia, Ericsson. Среди моделей этой серии есть 2 и 4 ядерные процессоры, применяемые в многоканальных системах обработки, где ядра имеют общую разделяемую память.

Наиболее перспективной считается модель C55xx (она представлена выше в этом же разделе). В ядро процессора добавлено еще одно АЛУ, еще один модуль умножения с накоплением, поэтому процессор способен выполнять до двух вычислительных операций одновременно. Увеличено количество внутренних шин, причем шин данных 5: три для чтения и две для записи. Введена плавающая длина команд. За счет введения конвейерной обработки максимальная производительность увеличена до 600 MIPS.

Процессоры семейства C55xx условно делятся на несколько групп.

В первой группе СП общего назначения: C5501/02/10. Процессор C5510 имеет наибольший объем памяти на кристалле, процессоры C5501 и C5502 имеют меньший объем памяти на кристалле, но содержат равные длине слова внешние шины адреса, что позволяет подключать к ним массивы данных большого объема.

Вторая группа процессоров C5503/07/09 содержит разнообразную периферию, позволяющую легко интегрироваться в интерфейсы мультимедийных систем – USB, Flash-карт.

Третью группу составляют процессоры, частично относящиеся к данному семейству (OMAP5910/12). Они содержат два ядра на одном кристалле: сигнальное и хост-ядро. К данным процессорам адаптирован Linux, что позволяет строить на их основе более сложные системы, где сигнальное ядро выполняет обработку графической, звуковой, видео, телеметрической информации, а второе ядро выполняет функции хост-процессора.

Платформа C6000. Современные передовые технологии ЦОС требуют модульных принципов обработки на базе параллельных исполнительных устройств, выполняющих обработку на частотах

свыше 200 МГц и поддерживающие многошинную архитектуру с полосой пропускания до 400 Мбайт/с.

Процессоры C6000 развиваются по трем направлениям.

Первое направление включает процессоры C62xx и C64xx с фиксированной точкой, второе направление – процессоры C67x с плавающей точкой, третье направление – процессоры C64x с фиксированной точкой, имеющие расширенную архитектуру и более развитую систему команд.

Ядром всех процессоров семейства C6000 является центральное процессорное устройство (ЦПУ), построенное на основе VLIW архитектуры. Упрощенная структура ЦПУ представлена на рис.12.11.

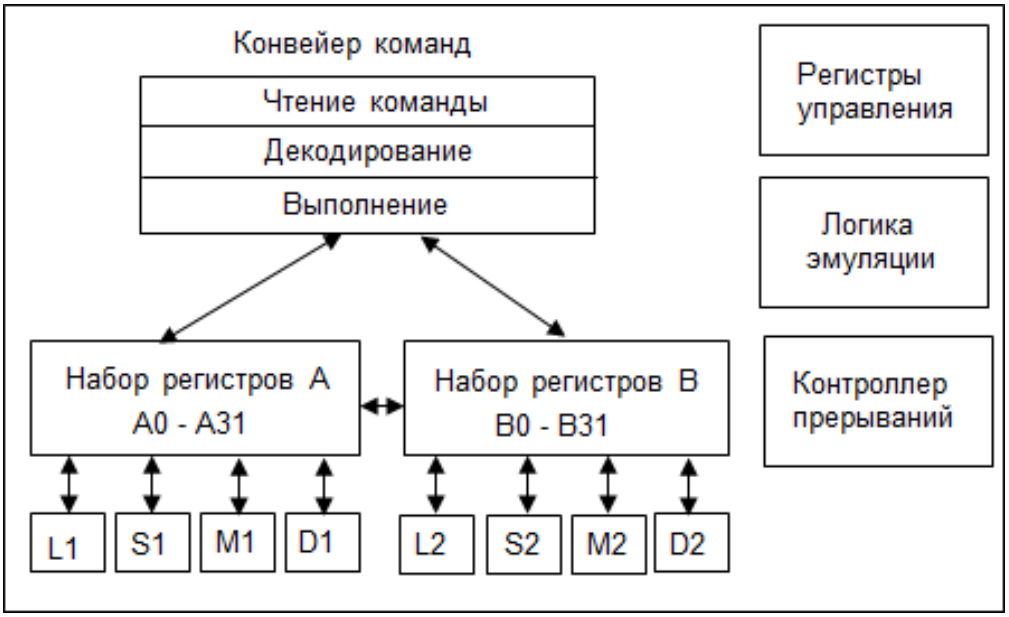


Рис.12.11. Структура ЦПУ процессоров модели C6000

ЦПУ содержит два арифметических блока обработки, в каждом из которых имеется по 4 операционных узла:

- узел выполнения арифметических и логических операций, операций сравнения L1;
- схема реализации операций сдвига и ветвления программ S1;
- умножитель M;
- адресное устройство чтения и записи D1.

Наличие внутреннего конвейера команд, большого количества универсальных регистров и многопоточной архитектуры позволяет добиться одновременной работы всех 8 вычислительных узлов. Кроме

того, все узлы могут выполнять арифметические и логические операции с плавающей запятой.

ЦПУ извлекает из памяти сразу восемь 32 разрядных команд в одном слове длиной 256 бит. Каждый из двух трактов обработки (A и B) имеет собственный блок из 16-ти регистров по 32 разряда каждый, при этом регистры обеих трактов могут оперативно взаимодействовать. Таким образом, процессор может выполнять до 8 команд одновременно (параллельно) в одном такте. Процессор имеет 32 Кбайт кэш первого уровня и 2 Мбайт кэш второго уровня для команд и данных.

Использование прямого доступа в память (DMA) позволяет сортировать данные отдельных каналов в отдельные буферы без участия ресурсов процессора. В некоторых моделях платформы C6000 используется расширенный контроллер EDMA, отличающийся большим числом каналов и схемой приоритетов.

Больший интерес C6000 представляет семейство TMS320C64xx. Семейство построено на базе архитектуры VLIW второго поколения (длинное слово команды, параллельные вычислительные модули).

К семейству C64x относится достаточно обширное количество процессоров, отличающихся по производительности, объему внутренней памяти и набору периферии с идентичным ядром центрального процессора. Таким образом, у разработчика существует возможность оптимально подобрать процессор, удовлетворяющий требованиям прикладной задачи по вычислительной мощности и набору периферии при минимальной цене. СП этого семейства предназначены для построения многопроцессорных систем ЦОС реального времени (радиолокация, видеосистемы распределенного типа, телекоммуникационное оборудование базовых станций).

Отдельная ветвь семейства C64x – процессоры TMS320DM64x, предназначенные для обработки видеоданных. Они содержат многофункциональные видеопорты с возможность потоковой обработки цифрового видео в различных форматах.

Линейка СП с плавающей запятой платформы C6000 состоит из семейства TMS320C67x. Они имеют 64 Кбайт памяти программ и 64 Кбайт памяти данных, а также набор функциональных устройств, аппаратно реализующих вычисления с плавающей запятой.

При частоте синхронизации 160 МГц производительность составляет 1GFLOPS (миллиард операций с плавающей запятой в секунду). Процессор совместим на уровне ассемблерного кода с C64x и C62x. Наиболее эффективное применение – студийное звукозаписывающее оборудование, аудиосинтезаторы.

Двухядерные сигнальные процессоры фирмы TI.

Процессоры TMS320DM64x послужили фундаментом для следующего поколения мультимедийных СП компании TI— двухъядерных процессоров модели DaVinci.

Использовав опыт, накопленный при разработке процессоров TMS320DM64x, компания TI создала мультимедийный процессор на основе интеграции высокопроизводительного семейства процессоров TMS320C6000 и современного дополнительного ядра обработки. В качестве высокопроизводительного основного процессора (DSP-ядро) применен модернизированный аналог ранее рассмотренного C64x процессор TMS320C64xx, а в качестве дополнительного универсального устройства обработки использовано ядро ARM926. Все это позволяет решать практически любую задачу многопотоковой обработки сигналов аудио и видео. Обобщенная структура СП модели Da Vinci представлена на рисунке 12.12.

Наличие DSP-ядра позволяет программно реализовывать практически любые алгоритмы обработки аудио и видеоданных. Ядро ARM позволяет значительно снизить нагрузку на ядро DSP. Интерфейс памяти и устройств хранения гарантирует высокую скорость обмена с внешними подключаемыми модулями.

Рис.12.12. Обобщенная структура СП модели Da Vinci

Дополнительный модуль подключения реализует обмен по линии стандартных внешних портов. Блок последовательных интерфейсов обеспечивает гибкость при реализации конкретных устройств. Рассмотрим отдельные составляющие более подробно.

12.5. Модели СП компании Analog Devices

Сигнальные процессоры семейства ADSP успешно конкурируют с аналогичной продукцией компаний TI и Motorola благодаря высокой производительности и низкой цене, а также наличию развитых аппаратных и программных средств разработки прикладных систем. К наиболее перспективным СП этой фирмы можно отнести:

- 32-разрядные процессоры с плавающей запятой семейства SHARC (Super Harvard Architecture Computer) моделей ADSP-21xx (например, ADSP2126x и ADSP2136x);
- сверхпроизводительные 32-разрядные процессоры с плавающей запятой семейства Tiger SHARC (ADSP-TS101/201/202/203);
- 16-разрядные сигнальные процессоры с фиксированной запятой семейства Blackfin (ADSP-BF531/532/533/535/561).

Рассмотрим более подробно эти модели СП.

Сигнальные процессоры семейства SHARC. В отличие от существующих 32-разрядных процессоров, SHARC-архитектура не имеет существенных ограничений по производительности. Все процессоры этого семейства состоят из пяти основных частей (рис.12.13) – высокопроизводительного ядра, двухпортового статического ОЗУ большого объема, мощного процессора ввода/вывода и порта связи с внешним миром, объединенных внутренними шинами команд и данных через шинный коммутатор.

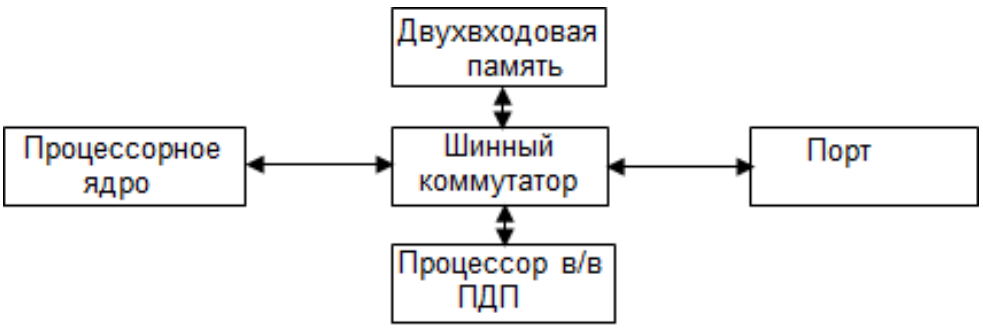


Рис.12.13. Архитектура процессоров семейства SHARC

Ядро имеет в своем составе три независимых параллельно работающих вычислительных устройства, выполняющих арифметические и логические операции над данными, временно хранящимися в специальном регистровом файле. Наличие такого регистрового файла обеспечивает быструю реакцию на прерывания, например в системах реального времени. В ядре имеется два независимых генератора адреса данных, которые обеспечивают вычислительному блоку гибкие режимы адресации памяти, организацию кольцевых буферов. Устройство управления ядра вместе со встроенным кэшем команд осуществляет выборку и исполнение команд за один машинный цикл.

Двухпортовое ОЗУ состоит из двух блоков памяти, каждый их которых можно использовать как для хранения 48-разрядных команд, так и данных, допускающих организацию в виде 8-,16- и 32-разрядных слов. Доступ к информации может быть осуществлен как со стороны ядра процессора, так и любого внешнего устройства.

Процессор ввода/вывода содержит контроллер прямого доступа к памяти (ПДП), управляющего работой 10 каналов прямого доступа, два полностью дуплексных программируемых последовательных портов (поддерживающих многоканальный режим с разделением времени), шести скоростных 4-разрядных линков-портов для оперативного обмена информацией между процессорами в мультипроцессорных системах.

Порт связи с внешним миром содержит мультиплексоры внутренних шин для связи с внешними устройствами, хост-порт для организации многопроцессорной системы, мультипроцессорный интерфейс, позволяющий без дополнительных затрат реализовать одновременную работу до шести процессоров семейства на одной шине.

Процессоры семейства ADSP-21xx имеют пять внутренних шин для повышения эффективности передачи данных. Шины адреса памяти программ и адреса памяти данных используются одновременно в пределах адресных пространств памяти программ и памяти данных. Шины данных памяти программ и шина данных памяти данных используются также одновременно для передачи данных из соответствующих областей памяти. При выводе шин на

корпус за пределы кристалла они объединяются в одну внешнюю шину адреса и в одну внешнюю шину данных. Шина результата используется для пересылки промежуточных результатов напрямую между различными вычислительными блоками.

Данная архитектура позволяет за один машинный цикл:

- сгенерировать адрес следующей команды программы,
- выбрать следующую по счету команду,
- выполнить вычислительную операцию,
- передать или принять данные от двух последовательных портов.

Сигнальные процессоры семейства SHARC предназначены для применения во встраиваемых устройствах для обработки аудиосигналов, высококлассной измерительной и контрольной аппаратуре, в медицинской аппаратуре, бытовой электронике, системах распознавания речи, средствах телекоммуникаций и других устройствах, где требуется большая вычислительная мощность и развитые средства высокоскоростного обмена данными.

К сигнальным процессорам третьего поколения с плавающей запятой относятся более поздние модели процессоров Analog Devices.

Сигнальные процессоры семейства Tiger SHARC.

Наиболее известной моделью этого семейства является процессор ADSP-TS001 (рис.12.14).

Это первый процессор с супескалярной архитектурой, в которой сочетаются возможности RISC-процессоров (фиксированная структура команды, конвейерное выполнение, предсказание переходов) и VLIW-архитектуры (выявление параллелизма уровня команд на этапе компиляции, возможность независимого задания в программе загрузки функциональных блоков).

ADSP-TS001 имеет тактовую частоту всего 150 МГц, но обладает наибольшей производительностью среди процессоров семейства при работе с фиксированной и плавающей запятой, может обрабатывать 8,16,32-разрядные данные с одинаковой скоростью.

При интегрировании в многопроцессорную систему могут быть объединены до 8 процессоров ADSP-TS001.

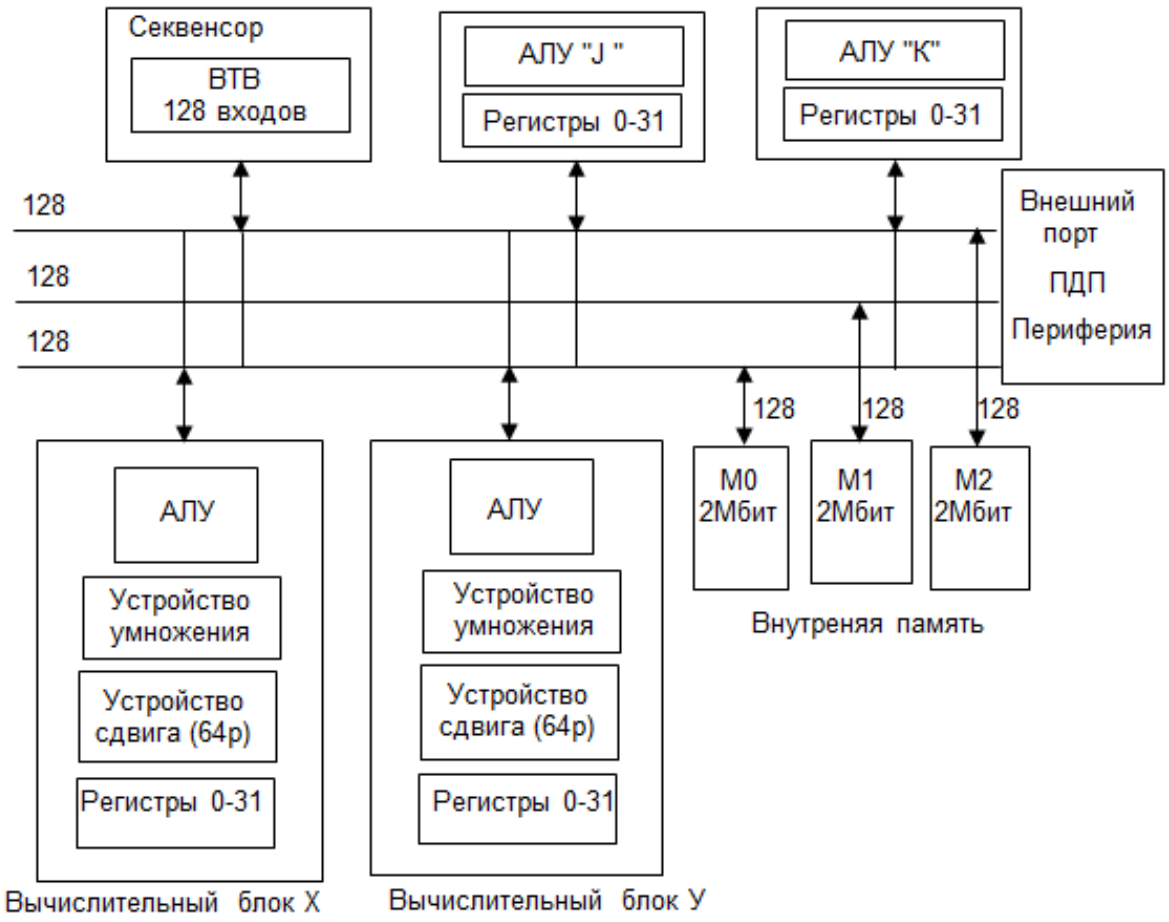


Рис.12.14. Архитектура процессора ADSP-TS001

В состав сигнального процессора ADSP-TS001 входят: 2 вычислительных блока, внутренняя память, 2 АЛУ адресной арифметики (АЛУ «J» и АЛУ «K»), блок формирования последовательности команд (секвенсор), внешние порты, каналы ПДП, периферия.

Эффективная загрузка функциональных устройств СП, планируемая на стадии компиляции программ, позволяет в одном такте выполнять 2 операции умножения с накоплением для 16-разрядных данных, 2 операции умножения с накоплением 16-разрядных комплексных чисел или 2 операции с 80-разрядными числами с накоплением 32-разрядных данных.

АЛУ «J» и АЛУ «K» предназначены для вычисления адресов или выполнения целочисленных операций над данными. Они имеют 32-разрядный регистровый файл, поддерживают циклические буферы и бит-реверсную адресацию.

Секвенсор обеспечивает порядок исполнения команд по результатам предварительно заданных условий. Кроме того, одна и та же команда может быть выполнена двумя блоками одновременно с использованием различных значений данных (SIMD-обработка). В составе секвенсора используется буфер адресов перехода (BTB-Branch Target Bufer) и узел предсказания переходов.

Внутренняя и внешняя память СП организованы в виде единого адресного пространства. Внутренняя память разделена на три 128-разрядных блока по 2 Мбит каждый (M0, M1, M2), что позволяет при обращении к памяти читать в регистровый файл нормальные, длинные и учетверенные слова, а также выбирать в каждом цикле до 4-х 32-разрядных команд. Данные с длиной слова 8,16 и 32 разряда могут записываться в память последовательно в упакованном виде.

Три 128 разрядные шины образуют скоростной канал между внутренними блоками и внешними периферийными устройствами. 64-разрядный интерфейс внешней шины позволяет строить мультипроцессорные системы с 8 процессорами.

Сравнительно сложную организацию памяти имеют процессоры TMS320C620x (рис. 12.15).

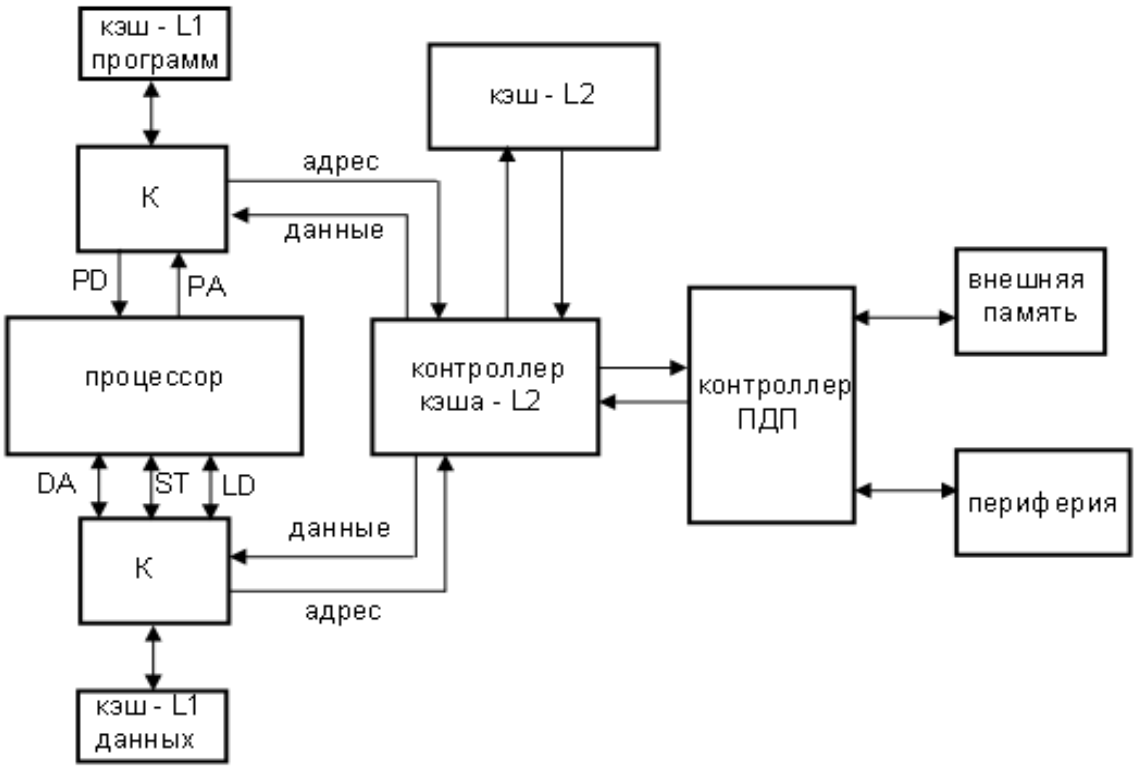


Рис.12.15. Организация памяти в процессорах TMS320C62xx

Они обладают внутренними памятью программ и памятью данных. Внутренняя память программ может работать как обычная адресуемая память программ и как кэш программ.

Кэш в свою очередь имеет несколько режимов работы. Обычный режим — это кэш программ между процессором и внешней памятью. В этом случае происходит следующее: чтение по заданному адресу производится из кэш. Если в кэш нужной информации нет, то выполняется чтение данных из внешней памяти, запись ее в кэш и передача в процессор. При повторном обращении к тем же командам чтение осуществляется уже прямо из кэш.

Внутренняя память имеет кэш двух уровней. Первый уровень кэша L1 образуется из отдельных частей для программ и данных. Указанный кэш управляется двумя контроллерами (К) — для кэш программ и для кэш данных. Процессор связан с кэш программ через шины PD (program data) и PA (program address). Связь процессора с кэш данных происходит через комплекты шин DA (data address), ST (store data) и LD (load data). Общий для данных и программ кэш второго уровня L2 управляется своим контроллером. Он связан с внешней памятью процессора через расширенный контроллер ПДП и интерфейс внешней памяти. Контроллер ПДП позволяет загружать в кэш L2 данные через различные периферийные устройства, в том числе последовательные порты ввода/вывода.

Основные сферы применения сигнальных процессоров этой серии: многоканальные базовые станции для сетей сотовой связи, шлюзы Интернет-телефонии, системы обработки и сжатия трехмерных графических изображений, радары, многоканальные цифровые абонентские линии, устройства промышленного и военного назначения, в которых необходимо обеспечить многоканальную цифровую обработку сигналов в реальном масштабе времени.

Двухъядерные процессоры семейства Blackfin.

Отвечая на потребности рынка компания Analog Devices совместно с компанией Intel разработала архитектуру MSA (Microsignal architecture) и на основе этой архитектуры создала семейство процессоров Blackfin. Данная архитектура совмещает в себе основные достоинства двух классов вычислительных устройств –

СП и микроконтроллеров. В 2005 году это семейство было дополнено двухъядерным процессором модели ADSP BF561.

ADSP BF561 – это симметричный двухъядерный процессор с идентичными и равноправными ядрами обработки. Каждое из ядер включает следующие компоненты:

- вычислительный блок;
- регистровые файлы;
- блок формирования адресов;
- блок управления.

Основу вычислительного блока составляют два умножителя-накопителя (MAC), каждый из которых представляет собой комбинацию 16-разрядного умножителя и 40-разрядного аккумулятора. Стандартные арифметические и логические операции над 16- и 32-разрядными данными производятся двумя 40-разрядными арифметико-логическими устройствами. 40-разрядное устройство сдвига позволяет выполнять операции логического, арифметического и циклического сдвига, осуществлять нормализацию и извлечение экспоненты, манипулировать отдельными битами или наборами битов входных операндов. Кроме того, в состав вычислительного блока входят четыре 8-разрядных видео-АЛУ, поддерживающих некоторые операции над 8-разрядными данными, характерные для задач обработки видеоизображений.

В процессоре ADSP-BF561 используется иерархическая трехуровневая модель памяти. Отдельного пространства ввода/вывода в процессоре нет, и все ресурсы отображены в едином 32-разрядном адресном пространстве. Память первого уровня работает с тактовой частотой ядра, но имеет малый объем. Каждое из ядер имеет собственную память объемом 100 кбайт, доступную только ему. Эти области памяти структурированы следующим образом:

- 32 кбайта памяти команд, из которых 16кбайт могут быть сконфигурированы как кэш команд,
- 64 кбайта памяти данных, из которых 32кбайта могут быть сконфигурированы как кэш данных.

На кристалле также интегрирована менее быстродействующая память второго уровня объемом 128 кбайт. В этой памяти могут храниться как команды, так и данные, она доступна обоим ядрам и не

может быть сконфигурирована как кэш. Для оптимизации обмена между двумя видами памяти в архитектуре процессора предусмотрен специальный контроллер.

Третий уровень в иерархической модели памяти процессора Blackfin занимает внешняя память. В пространстве внешней памяти может отображаться до четырех банков памяти объемом от 16 до 512 Мбайт и до четырех банков асинхронной флэш-памяти объемом 64 Мбайт каждый. Разрядность внешней шины данных процессора ADSP-BF561 составляет 32 бита.

Процессор ADSP BF561 имеет очень богатый набор интегрированных периферийных узлов. Добавление на кристалл второго ядра привело к увеличению, по сравнению с предшествующими одноядерными моделями, количества линий ввода/вывода общего назначения с16 до 48. Появился дополнительный сторожевой таймер, дополнительный таймер ядра и девять дополнительных таймеров общего назначения. Однако наиболее интересным, особенно в свете задач обработки видеоизображений, является наличие в ADSP BF561 второго 16-разрядного параллельного порта PPI.

Параллельные порты позволяют без использования дополнительной логики подключать к процессору многие стандартные АЦП, ЦАП, видеокодеры и декодеры, и присутствие на кристалле сразу двух PPI дает возможность, например, организовать ввод оцифрованного видеосигнала, его обработку «на лету» и последующий вывод на устройство отображения без мультиплексирования внешних выводов.

Набор команд процессора включает в себя как 16-, так и32-разрядные команды, причем наиболее часто исполняемые команды (загрузка/сохранение регистров) кодируются 16-битами, а 32 битами кодируется большинство арифметических команд и команды манипуляции битами. Архитектура ADSP BF561 допускает произвольное размещение 16- и 32-разрядных команд в памяти.

Разветвленная система внутренних шин и большое число вычислительных блоков позволяют каждому ядру процессора в одном цикле выполнять сразу несколько команд, зачет чего достигается повышенная плотность кода.

Эти свойства набора команд в совокупности с поддержкой многих характерных для микроконтроллеров возможностей обеспечивают высокую эффективность при компиляции кода, написанного на языках C/C++, что значительно упрощает разработку программного обеспечения.

Перечисленные архитектурные особенности процессора ADSP BF561 позволяют разработчику строить при помощи одного относительно недорогого процессора сложные системы с интенсивной сигнальной обработкой.

12.6. Модели сигнальных процессоров компании Motorola

Каждая из компаний по производству сигнальных процессоров занимает свою нишу на рынке электронной техники. Продукция компании Analog Devices используется при решении сложных задач, требующих выполнения больших объемов математических вычислений (цифровая фильтрация, корреляционный анализ, обработка изображений), так как их производительность на таких задачах выше, чем у процессоров компаний Texas Instruments и Motorola. Для задач, требующих интенсивного обмена с внешними устройствами (многопроцессорные системы, системы контроля, диагностики, управления), предпочтительнее использовать процессоры Texas Instruments, обладающие высокоскоростным интерфейсом. Компания Motorola является лидером по производству относительно простых 16 и 24 разрядных сигнальных процессоров с фиксированной запятой, применяемых, в основном, в системах связи.

Расширенные коммуникационные возможности, наличие достаточных объемов внутрикристалльной памяти для данных и программ, возможность поддержки режима энергосбережения позволяют использовать эти СП в качестве контроллеров управления технологическими процессами, в бытовых электронных приборах, в автомобильной электронике, в системах мобильно связи.

Рассмотрим одно из перспективных семейств СП компании Motorola – DSP56800, которое имеет наилучшие показатели производительность/качество/стоимость. Обобщенная структурная схема процессора данного семейства представлена на рис.12.16.

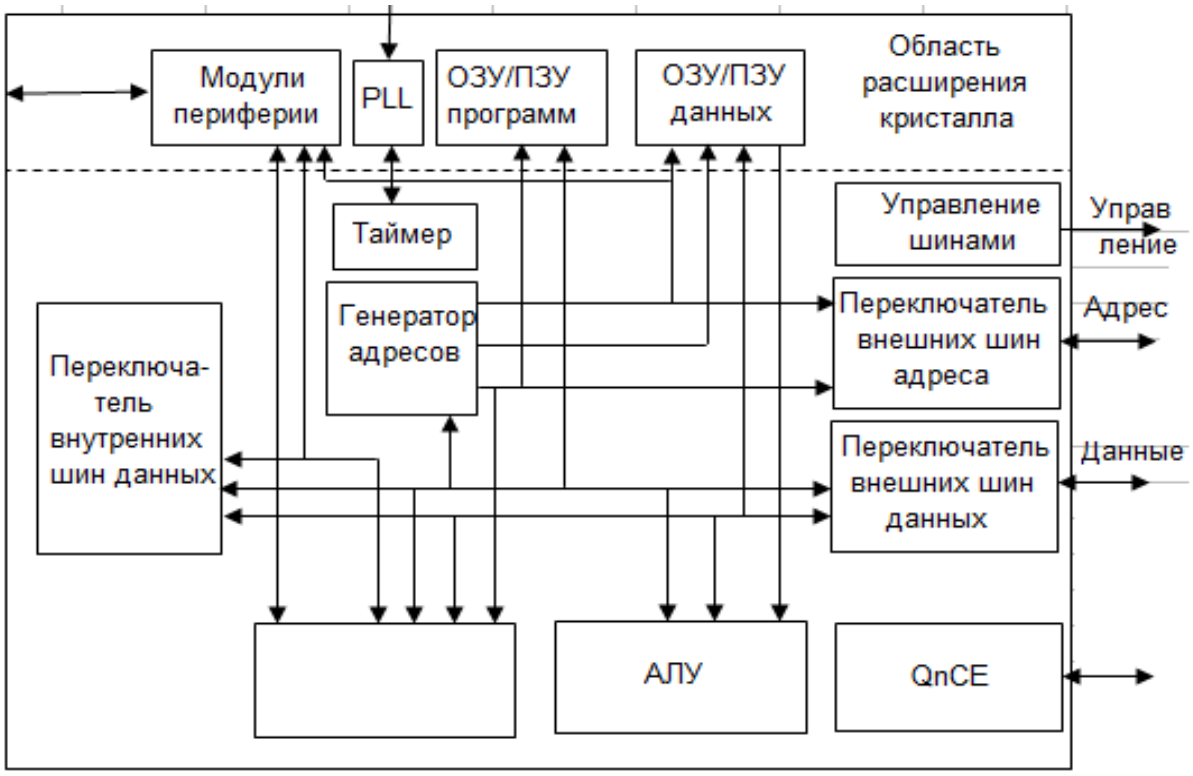


Рис.12.16. Архитектура процессора DSP 568xx

Архитектура семейства объединяет традиционные функции управления с помощью микроконтроллеров с эффективностью СП. Ядро состоит из четырех функциональных устройств: контроллера программ, 16-ти битного устройства генерации адресов, 16-разрядного АЛУ и порта эмуляции QnCE с 8-ми уровневым FIFO буфером и выходом на отладочный порт JTAG. В составе АЛУ имеется устройство умножения с накоплением (MAC) разрядностью 16x16, выполняющее все операции за один цикл, а также два 36-битных аккумулятора с битами расширения. В составе процессора имеются узлы интерфейса внешней памяти (данные, адрес, управление), таймер прерывания реального времени, модуль снижения частоты и потребления энергии (PLL).

Каждое из устройств ядра имеет свой набор регистров и логику управления и организовано таким образом, что может функционировать независимо и одновременно с тремя другими. Внутренние шины адресов и данных связывают между собой память, функциональные и периферийные устройства. Ядро реализует одновременно выполнение нескольких действий: устройство управления выбирает первую команду, устройство генерации адресов

формирует до двух адресов второй команды, а АЛУ выполняет умножение третьей команды.

Гарвардская архитектура СП обеспечивает наличие двух независимых областей памяти – данных и программ в форме ОЗУ и флэш-памяти. Флэш-память в зависимости от типа модели может иметь объем от 8 Кбайт до 64 Кбайт (для программ) и от 2 Кбайт до 8 Кбайт (для данных). Память ОЗУ может иметь объем от 1 Кбайт до 2 Кбайт (для программ) и от 1 Кбайт до 4 Кбайт (для данных).

Периферийные модули имеют различные модификации в зависимости от применения. В их составе могут быть параллельные и последовательные интерфейсные схемы, АЦП, декодеры, таймеры, программируемые входы внешних прерываний. Для каждого процессора фирмы Motorola, как и для процессоров других упомянутых компаний, существуют стандартные отладочные платы, которые осуществляют отладку программного обеспечения через JTAG-порты.

Наряду с рассмотренными моделями современных СП на рынке присутствуют перспективные модели процессоров других производителей. Некоторые из них имеют проблемную ориентацию, но многие предназначены для решения широкого круга задач цифровой обработки сигналов. Среди них большой сектор занимают многоядерные СП, сочетающие в свой архитектуре возможности параллельной обработки и реализации сложных численных методов преобразования данных. Рассмотри некоторые примеры.

Компания Texas Istrument выпустила на рынок новый промышленный DSP-процессор с шестью ядрами TMS320C6472, предназначенный для многоканальных высокопроизводительных приложений. Основные особенности DSP-процессора:

- шесть производительных ядер C64x, имеющих тактовую частоту 500 МГц и полностью совместимый с сигнальными процессорами ранее рассмотренных моделей C64x;

- наличие до 4-х Мбайт RAM L1/L2;

- процессор имеет для каждого ядра кэш L1, разделяемую память программ и данных L2 объемом 768 Кб и диспетчер ее совместного использования обеими ядрами;

- богатые периферийные возможности: гигабитный Ethernet, последовательный телекоммуникационный порт TSIP, интерфейс Host-порта (HPI) и обычные интерфейсы общего назначения.

Цифровые сигнальные процессоры серии «Мультикор» - программируемые многопроцессорные «системы на кристалле» на базе IP-ядерной платформы, разработанной научно-производственным центром «ЭЛВИС» (Риссия). Основное назначение процессоров этой серии – обработка сигналов и изображений в системах управления и высокоточной обработки информации.

К настоящему времени выпущена большая серия СП модели «Мультикор», процессоры которой отличаются рабочей частотой, пиковой производительностью, архитектурой и количеством вычислительных ядер (процессоров).

Последняя модель процессоров этой серии идет под маркой МС-0428и имеет следующие характеристики:

- технология изготовления – 0,18 мкм;
- MIMD-архитектура с 5 процессорами;
- рабочая частота 250-340 МГц;
- пиковая производительность (при 32битах) – 8000 MFLOPS.

Микросхемы в зависимости от модели содержат от 2 до 8 Мбит внутренней памяти, периферийные SHARC-совместимые линки и порты USB, Ethernet, PCI, JTAG, гиперлинки SpaseWire и Serial Rapid IO.

Особый класс сигнальных процессоров составляют «встраиваемые системы» обработки. Они размещаются внутри более сложного устройства обработки и имеют малые габариты, минимальную потребляемую мощность, как правило, размещены на одном кристалле.

12.7. Перспективы развития архитектуры сигнальных процессоров

Повышение параллельности обработки данных в сигнальных процессорах и использование высокоскоростных соединений между процессорами, внешней памятью и другими компонентами позволило разработчикам интегральных схем создать новое поколение плат и систем шлюзов для инфраструктуры.

Некоторые стратегии проектирования очевидны, например интеграция дополнительных сопроцессоров на кристалле устройства и повышение параллелизма операций. Также начинает практиковаться интеграция нескольких ядер процессоров на одном кристалле.

Многоядерные СП имеют очевидные преимущества. Несколько ядер, совместно использующих память, могут иметь меньшую тактовую частоту и меньшее напряжение питания, что приводит к меньшей мощности на канал. Многоядерность также создает возможности, характерные для инфраструктуры сотовой связи и появившейся в последнее время области приложений WiMAX.

В настоящее время можно выделить три основных класса устройств цифровой обработки сигналов – универсальные процессоры младших моделей, сигнальные процессоры (DSP) и устройства цифровой обработки сигналов на основе программируемых логических интегральных микросхем (ПЛИС).

Младшие модели универсальных процессоров – это встраиваемые 8-32 разрядные микроконтроллеры. Старшие модели универсальных процессоров – это центральные процессоры персональных компьютеров, рабочих станций, сетевых серверов, подобные известным процессорам семейств Pentium и Power PC.

Со времен компьютерной революции 1970-х годов возникло несколько тенденций, определивших будущие перемены в развитии систем приема, обработки и передачи данных. Эти тенденции сильно влияют и на развитие архитектуры сигнальных процессоров – важных компонентов систем телекоммуникаций и систем реального времени.

Среди этих тенденций:

- переход от исключительно голосового трафика к совокупному голосовому трафику и трафику данных. Эта тенденция зародилась десятилетия назад и полноценно реализуется в настоящее время;

- добавление мультимедийного трафика, в особенности потокового мультимедиа, к существующему голосовому трафику и трафику данных. Эта тенденция подтверждается переходом поставщиков телекоммуникационных услуг ксервисам Triple Play, обеспечивающим передачу голоса, видео и данных;

- переход от сервисов с фиксированным местоположением к сервисам на дому, а затем и к мобильным услугам. Эволюция комплекса голос/данные/мультимедиа, имевшая место в проводной инфраструктуре, в настоящее время реализуется и в беспроводных системах.

-концентрация разработки новых архитектур СП на двух основных наиболее актуальных направлениях развития: системах телекоммуникаций и системах видеообработки.

Многие высокопроизводительные микропроцессоры, например Pentium, Athlon или PowerPC, имеют достаточно вычислительных ресурсов, чтобы успешно выполнять задачи, связанные с цифровой обработкой сигналов. Наряду с техникой SIMD в них применяются расширенные наборы команд, такие, например, как MMX и SSE. Благодаря использованию 64-разрядных шин данных, регистров и АЛУ микропроцессоры общего назначения иной раз по производительности обгоняют даже самые быстрые ПЦОС. В немалой степени это связано и с тактовой частотой. Как известно, данный параметр для универсальных кристаллов часто превышает 600-650 МГц, а у специализированных ПЦОС не бывает больше 200-250 МГц.

Тем не менее СП применялись и будут применяться впредь для многих приложений. И причина этого весьма проста. Несмотря на то что универсальные микропроцессоры могут обеспечить сравнимую или даже лучшую производительность, ПЦОС предлагают лучшее соотношение производительности, мощности потребления и цены.

Вопросы для контроля

1. Перечислите основные задачи цифровой обработки сигналов.
2. Каковы базовые области применения ЦОС?
3. Опишите работу умножителя-накопителя.
4. В чем особенности архитектуры сигнальных процессоров?
5. Назовите основные архитектуры сигнальных процессоров.
6. Опишите технологию параллелизма в организации шин СП?
7. Каковы архитектурные особенности технологии ПЛИС?
8. В чем основной недостаток заказных чипов СП?
9. Перечислите блоки обработки в ЦПУ процессора C600?
10. Какова оригинальность архитектуры СП модели Da Vinci?
11. Назовите компоненты, входящие в состав ядер СП BF561?
12. Назовите три основных класса устройств ЦОС.

ГЛАВА 13. ПРОЦЕССОРЫ ГРАФИЧЕСКОЙ СИСТЕМЫ КОМПЬЮТЕРА

13.1. Распараллеливание графических вычислений

Во многих областях использования компьютеров существует задача визуализации трехмерных изображений (сцен). Большим стимулом для улучшения обработки графики стала индустрия компьютерных игр, развивавшаяся как на базе персональных компьютеров, так и на специальных игровых консолях. Быстро разраставшийся рынок компьютерных игр подтолкнул многие компании на инвестирование все более крупных средств в развитие быстродействующего графического оборудования. Таким образом, темпы развития графической обработки стати более высокими по сравнению с темпами развития вычислений общего назначения в серийно выпускаемых микропроцессорах.

Трехмерная (3D) визуализация необходима при создании анимации и спецэффектов, а также для инженерного проектирования, наглядного представления физико-математических моделей. Поскольку графические процессоры создавались именно для решения задачи визуализации, рассмотрим ее основные составляющие. Пусть, для определенности, трехмерная сцена представляет собой совокупность поверхностей, разбитых для дискретной компьютерной обработки на плоские треугольники. В некоторых массивах, которые обычно называют текстурами, хранится информация о цветах треугольников. Кроме того, известны положения и характеристики источников света. Задача визуализации состоит в том, чтобы сформировать изображение этой сцены на плоскости экрана, положение которой определяется точкой зрения наблюдателя.

Процесс формирования изображения включает в себя следующие основные этапы:

- проектирование треугольников, представляющих сцену, на плоскость экрана;
- разбиение полученных проекций на отдельные пиксели, для которых будут определяться цвета (стадия растеризации треугольников);

- определение видимого цвета элементов поверхности треугольников с учетом исходного цвета и отражающих свойств самой поверхности, прозрачности других поверхностей, освещенности и теней.

Обрабатываемые сцены обычно состоят из очень многих треугольников, которые разбиваются на еще большее количество пикселей, так что их визуализация требует больших вычислительных ресурсов. Вместе с тем и вершины, и пиксели можно обрабатывать почти или со- всем независимо друг от друга. Соответственно задача визуализации допускает очень эффективное распараллеливание. Именно в целях распараллеливания графических вычислений цен- тральные процессоры персональных компьютеров стали суперскалярными — получили возможность одновременного (векторного) исполнения некоторых операций сразу над несколькими числами (расширения 3DNow! и SSE). Большинство центральных процессоров состоит из нескольких ядер, что дополнительно увеличивает их потенциал как систем для параллельных вычислений. Тем не менее, исторически сложилось так, что наиболее эффективными устройствами для увеличения скорости обработки изображений стали и до сих пор остаются специализированные графические процессоры (GPU). Эти процессоры, разрабатывавшиеся именно для обработки графики, практически полностью ориентированы на параллельную обработку данных.

Рассмотрим некоторые ключевые характеристики, благодаря которым графические процессоры — GPU отличаются от центральных процессоров CPU.

1. GPU являются ускорителями, дополняющими CPU, поэтому им не нужно уметь выполнять все задачи, присущие CPU. Эта роль позволяет им посвятить все свои ресурсы графике. Таким образом, комбинация CPU-GPU является одним из примеров гетерогенной мультипроцессорной обработки, где не все процессоры являются идентичными.

2. Интерфейсами программирования для GPU являются высокоуровневые интерфейсы прикладного программирования (application programming interface, API), такие как OpenGL и разработанный компанией Microsoft DirectX, в совокупности с

высокоуровневыми языками графического затенения, или шейдинга (shading), такими как разработанный компанией NVIDIA язык C для графики (Cg) и разработанный компанией Microsoft язык высокого уровня для программирования шейдеров High Level Shader language (HLSL). Компиляторы языков предназначены не для машинных инструкций, а для промежуточных языков, отвечающих промышленным стандартам. Программный драйвер GPU генерирует оптимизированные машинные инструкции, подходящие к конкретному GPU.

3. Обработка графики предполагает рисование точек (вершин) трехмерных геометрических примитивов, таких как линии и треугольники, и затенение (shading) или прорисовку, или рендеринг (rendering), пиксельных фрагментов геометрических примитивов. Каждая точка (вершина) может быть нарисована независимо, и прорисовка каждого пиксельного фрагмента также может быть произведена независимо. Чтобы быстро прорисовать миллионы пикселов на кадр, создается GPU, способный параллельно выполнять множество потоков от программ, рисующих вершины и пиксельные затенения.

4. К типам графических данных относятся точки (вершины), состоящие из координат (x, y, z, w), и пикселы, состоящие из цветов (красный — red, зеленый — green, синий — blue, альфа alpha). Графические процессоры представляют каждый компонент точки в виде 32-разрядного числа с плавающей точкой.

Большое различие состоит в том, что графические процессоры, в отличие от центральных, не полагаются на использование многоуровневой кэш-памяти для преодоления большой латентности при обращении к памяти. Вместо этого GPU полагаются на наличие достаточно большого количества потоков, позволяющее скрадывать латентность обращения к памяти. То есть, между временем выдачи запроса к памяти и временем поступления данных GPU выполняет сотни или тысячи потоков, которые не зависят от этого запроса.

Развитие этих возможностей серьёзно изменило рынок в дальнейшем, открыв новый сектор – параллельные вычислители на основе видеочипов. Разработчики компании NVIDIA представили программно-аппаратную стратегию параллельных вычислений,

открыв новую долю рынка. И главным преимуществом этой инициативы NVIDIA стало то, что разработчики детально знают все возможности своих GPU, и в использовании графического API нет необходимости, а работать с аппаратным обеспечением можно напрямую при помощи драйвера. Результатом усилий этой команды стала NVIDIA CUDA (Compute Unified Device Architecture) – новая программно-аппаратная архитектура для параллельных вычислений на NVIDIA GPU.

Однако для достижения наилучших результатов необходимо учитывать ряд особенностей.

1) Графический процессор (GPU) состоит из нескольких мультипроцессоров, которые, в свою очередь, состоят из ядер. Каждое ядро одновременно выполняет 32 потока (warp). Например, NVidia GeForce GTX 480 состоит из 15×32=480 ядер и параллельно может выполнять до 15360 легковесных потоков. Потоки объединяются в блоки и сетки блоков. Каждый поток имеет идентифицирующие его координаты.

2) Максимальной производительности удается достичь при выполнении однотипных действий над большим числом обрабатываемых единиц данных.

3) Архитектура памяти имеет сложную организацию: глобальная память (объемная, но медленная), локальная память, разделяемая память (быстрая), память констант и т. д.

4) Особенности доступа к памяти: для получения максимальной пропускной способности все запросы к памяти должны быть выровнены.

В видеочипах NVIDIA основной блок – это мультипроцессор с восемью-десятью ядрами и сотнями ALU в целом, несколькими тысячами регистров и небольшим количеством разделяемой общей памяти. Кроме того, видеокарта содержит быструю глобальную память с доступом к ней всех мультипроцессоров, локальную память в каждом мультипроцессоре, а также специальную память для констант.

Самое главное – эти несколько ядер мультипроцессора в GPU являются SIMD-ядрами (одиночный поток команд, множество потоков данных). И эти ядра исполняют одни и те же инструкции

одновременно. Такой стиль программирования является обычным для графических алгоритмов и многих научных задач, но требует специфического программирования. Зато такой подход позволяет увеличить количество исполнительных блоков за счёт их упрощения. В отличие от современных универсальных CPU, видеочипы GPU предназначены для параллельных вычислений с большим количеством арифметических операций. И значительно большее число транзисторов GPU работает по прямому назначению – обработке массивов данных, а не управляет исполнением (control) немногочисленных последовательных вычислительных потоков (рис.13.1).

В отличие от CPU в GPU имеется большое количество простых и однотипных ALU (в приведенном примере их 16 x 8 = 128).

Ядра традиционнкх процессоров CPU созданы для исполнения одного потока последовательных инструкций с максимальной производительностью. Эти процессоры оптимизированы для достижения высокой производительности единственного потока команд, обрабатывающего и целые числа и числа с плавающей точкой.

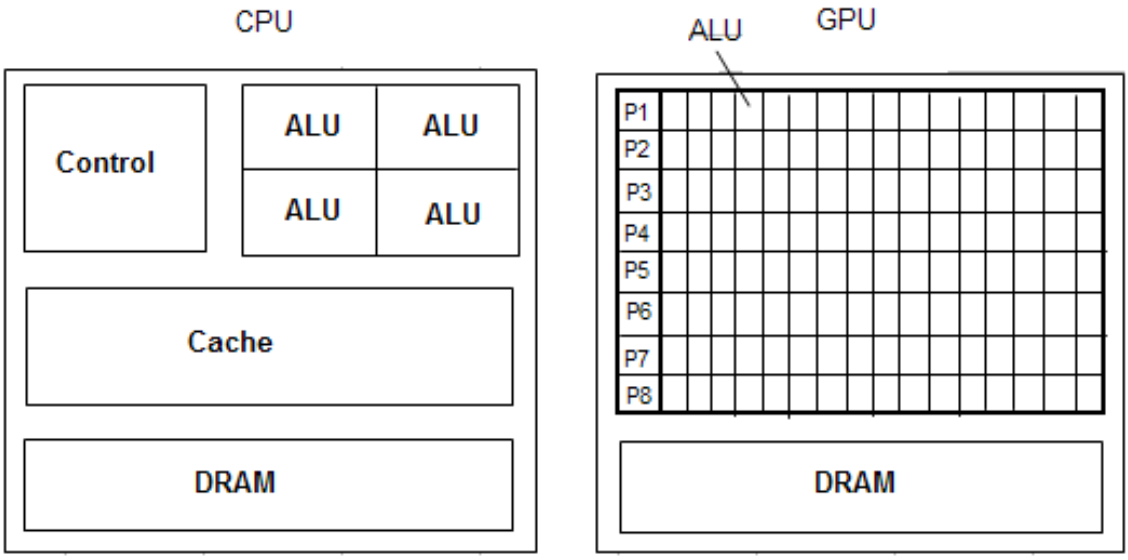


Рис.13.1.Структура универсального и графического процессоров

Разработчики CPU стараются добиться выполнения как можно большего числа инструкций параллельно, для увеличения производительности. Для этого у процессоров Intel появились технологии конвейерной и суперскалярной обработки, обеспечивающие выполнение двух и более инструкций за такт.

Графические процессоры являются наиболее ярким примером неоднородных многоядерных процессоров, т.е. процессоров с ядрами различной конфигурации. По сути, это сочетание многядерных процессоров с параллельно функционируюўими многочисленнқми простыми АЛУ (гетерогенная вычислительная структура).

13.2. Архитектура графического процессора

Конструктивно графический процессор представляет собой вычислительное устройство, работающее отдельно от центрального процессора, параллельно с ним. Обычно графические процессоры размещают на отдельных печатных платах с собственной системой охлаждения, которые часто называют **графическими ускорителями** (или **видеокартами**). Вместе с GPU на плате графического ускорителя расположена **видеопамять**— специализированная оперативная память, в которой хранятся обрабатываемые графическим процессором массивы данных.

Современные графические процессоры реализуются по технологии CUDA - новой архитектуре параллельнқх вычислений. Ее основное назначение графические приложения. Ранее описанные технологии MMX и SSE появились в универсальных процессорах из-за возросших требований графических приложений. Обработка фрагментов изображений и пикселей независима, их можно обрабатывать параллельно, отдельно друг от друга. Поэтому из-за изначально параллельной организации работы в GPU используется большое количество исполнительных блоков – АЛУ.

GPU отличается от CPU по принципам доступа к памяти. Пиксели и фрагменты изображения располагаются в памяти и читаются поочередно, поэтому и формирование адресов значительно проще. К тому же, в GPU всегда имеется несколько контроллеров памяти для ускорения доступа. Сама память выполнена по другой технологии с более быстрым доступом.

На рис.13.2. представлен пример микросхемы GPU, на которой находится 8 мультипроцессоров, под управление каждого из них находится 16 АЛУ. Таким образом, на кристалле находится 128 CUDA- ядер. Мультипроцессоры (МП) имеют свою кэш-память и схемы управления работой каждого АЛУ.

По своей организации память GPU-процессоров намного сложнее и имеет несколько уровней.

Рис.13.2. Кристалл GPU с 8 процессорами и 16 схемами АЛУ

Локальная память — это небольшой объём памяти, к которому имеет доступ только один потоковый процессор (АЛУ).

Разделяемая память кэш-память — это 16-килобайтный блок памяти с общим доступом для всех потоковых процессоров, расположенный в МП. Эта память быстрая, как регистры. Обеспечивает взаимодействие между потоками, управляется разработчиком напрямую. Преимущества этой памяти: использование в виде кэш первого уровня(L1), снижение задержек при доступе исполнительных блоков (ALU) к данным, сокращение количества обращений к глобальной памяти.

Регистровая память – является наиболее простым видом памяти. Каждый мультипроцессор содержит 8192 или 16 384 32-битовых регистров.

Глобальная память — самый большой объём памяти, доступный для всех мультипроцессоров. Она расположена видеочипе, ее размер составляет от 256 мегабайт до 1.5 гигабайт.

Рассмотрим архитектуру графического процессора G80 — одного из процессоров компании NVIDIA. Главной (и общей для всех графических процессоров) характеристикой этой архитектуры является то, что GPU представляет собой систему из параллельных

вычислительных устройств, каждое из которых применяет заданную, единую для всех устройств, программу к различным элементам входных массивов данных, расположенных в общей памяти.

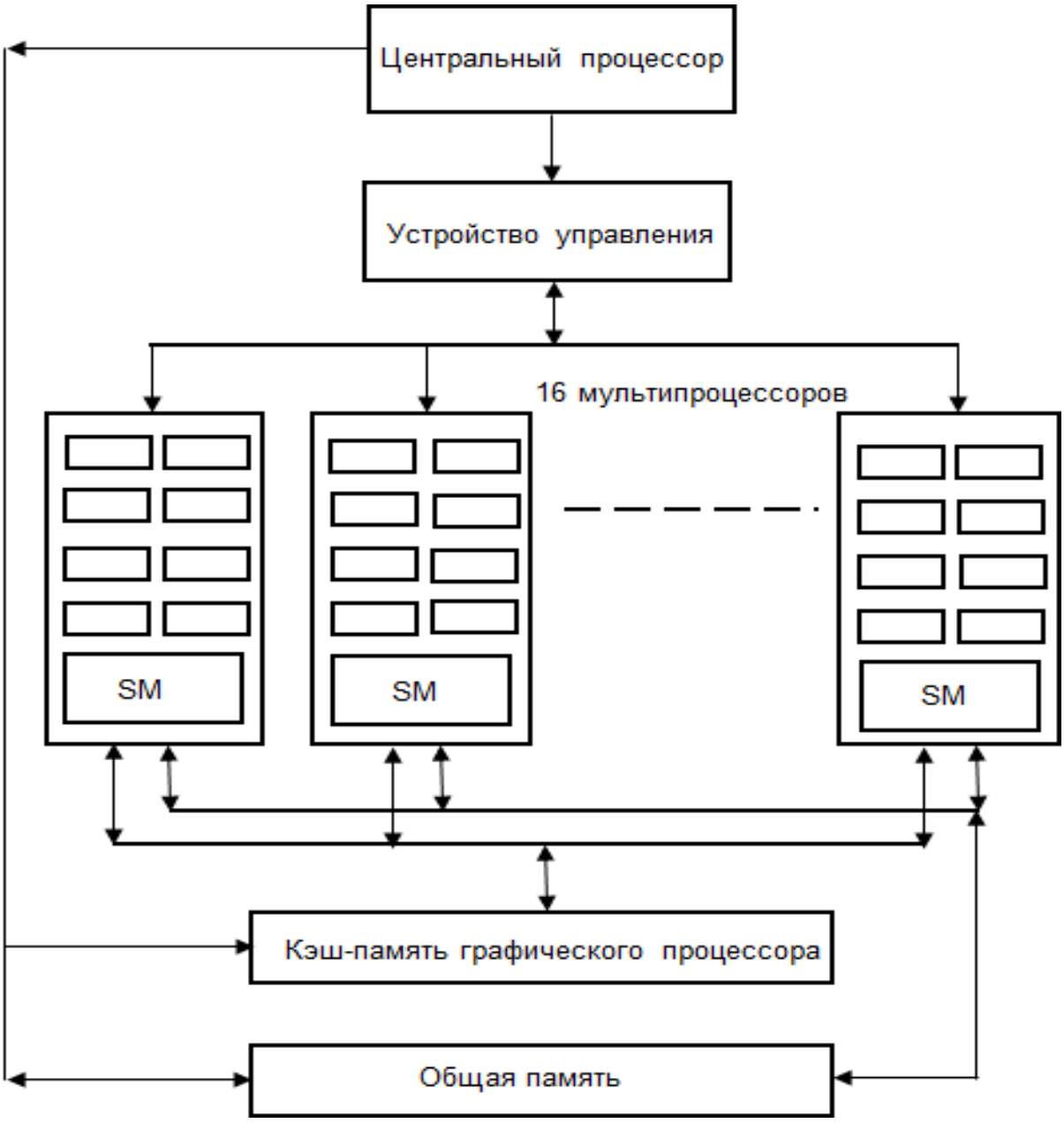
Параллельная архитектура графических процессоров ориентирована на исполнение алгоритмов, в которых элементы больших входных массивов обрабатываются одинаковым образом независимо или почти независимо друг от друга, то есть использующих распараллеливание вычислений по данным. Множества элементов, подвергаемых однотипной независимой обработке, называют потоками (данных либо результатов), так что графические процессоры осуществляют поточно-параллельную обработку данных.

Концепция программирования, заключающаяся в потоковой обработке данных, известна под аббревиатурой SIMD. Процессор, работающий по принципу SIMD, преобразует поток данных в поток результатов, используя программу как функцию преобразования. Выбор концепции SIMD для графических процессоров обусловлен тем, что она обеспечивает параллельное использование большого количества «вычислителей» без явного управления ими: распределения задач, синхронизации вычислений и коммуникации между параллельными расчетами. На рис.13.3 показана структура графического процессора G80 компании NVIDIA.

Графический процессор G80 представляет собой систему из 128 вычислительных устройств (вычислительных ядер – kernel), каждое из которых исполняет единую для всех устройств программу для элементов входных массивов данных, расположенных в общей памяти.

Каждому блоку из 8 ядер придается разделяемая кэш-память данных SM (Shared Memory), доступная всем ядрам одного мультипроцессора.

Кэш-память графического процессора доступна для работы всем 16 мультипроцессорам (центральному процессору эта память доступна частично и только для записи).



13.3. Структура графического процессора G-80

Общая память для размещения входных данных и результатов доступна как всем мультипроцессорам, так и центральному процессору.

Графические процессоры имеют собственную систему команд. Для программирования трехмерных изображений необходимо использовать специализированные прикладные программные библиотеки, составляющие интерфейс прикладного программирования API. Используются две библиотеки - OpenGL (Open Graphics Library) и DirectX.

В качестве языка программирования используется язык Cg на основе C++ (NVIDIA). В 2004 году вышел интерфейс DirectX 9, в настоящее время видеокарты поддерживают DirectX 10 и DirectX 11.

API DirectX выполняет следующие функции:

- программирование двумерной графики;
- создание трехмерной графики;
- работа со звуковыми данными;
- поддержка устройств ввода;
- разработка сетевых игр.

Графические процессоры с полностью программируемым графическим интерфейсом – это поколение DirectX 9. Структура такого процессора приведена на рис. 13.4. Рассмотрены только выполняемые графическим процессором функции по обработке изображения.

Процессор вершин использует специальные программы - вершинные шейдеры для обработки координат вершин, текстурных координат и расчета освещенности. Эти функции в принципе может выполнить и центральный процессор. Но обработать все пикселы изображения за короткое время он не может. Для этого используется специальный пиксельный процессор, который оперирует с цветом, глубиной, освещенностью и текстурными координатами. И вершинный, и пиксельный процессоры представляют собой большое число параллельно работающих конвейеров.

На центральный процессор возлагаются функции по конструированию сцены и преобразованию поверхностей в треугольники.

Затем эти данные через северный мост чипсета передаются по высокоскоростному интерфейсу в видеопамять видеокарты. Затем начинает работать графический конвейер.

После наложения текстур, удаления невидимых поверхностей, исправления и сглаживания дефектов изображения сформированный кадр передается на вывод в блок цифро-аналогового преобразования RAMDAC и/или на цифровой интерфейс вывода.

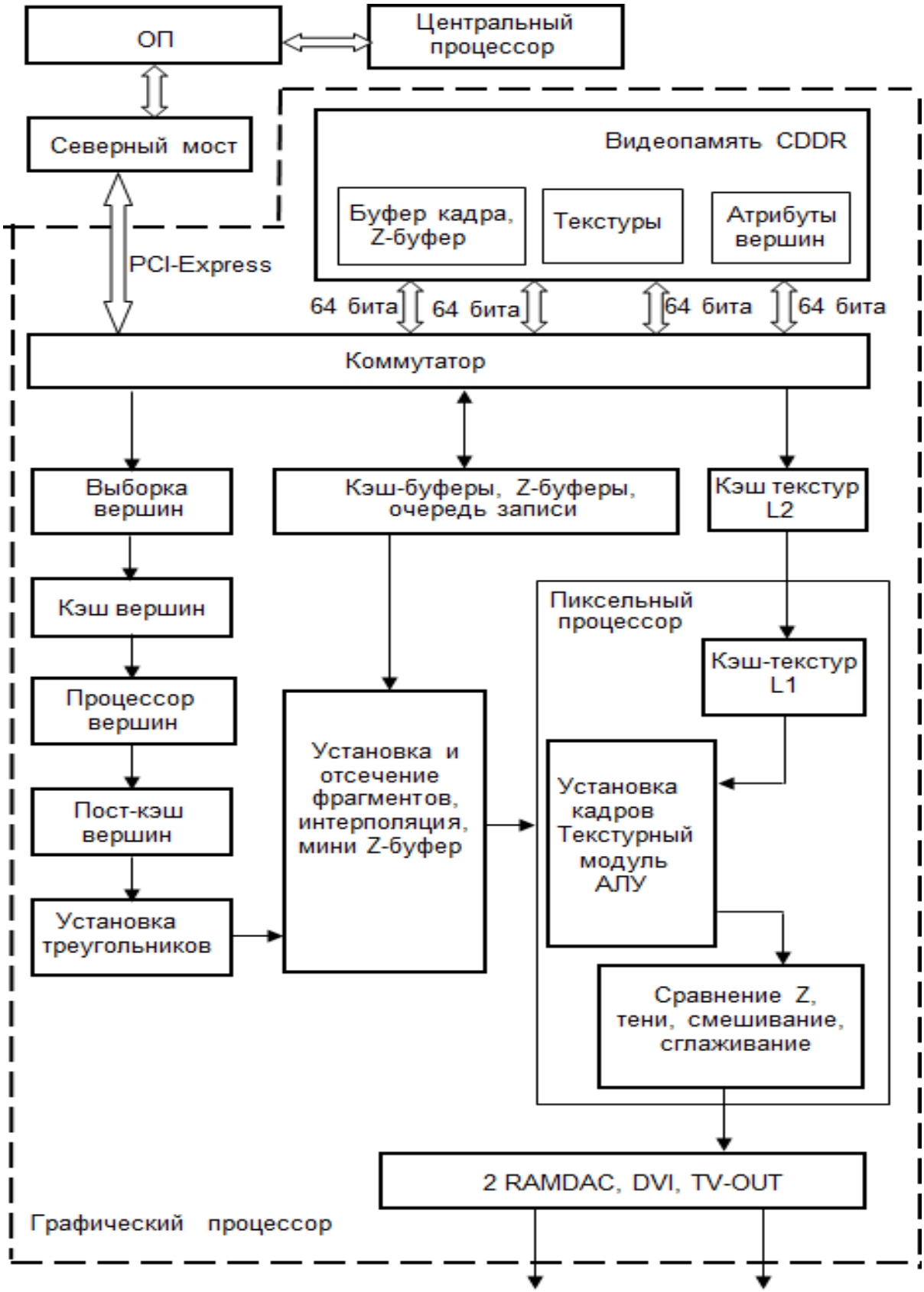


Рис.13.4. Структура графического процессора поколения DirectX 9

Разрешение экрана – это количество точек (пикселов), в изображении по горизонтали и вертикали. Для компьютерных мониторов (4:3) используется разрешение:

- VGA – 640x480;
- SVGA – 800x600;
- XGA – 1024x768;
- SXGA – 1280x1024 (5:4);
- SXGA+ - 1400x1050;
- UXGA – 1600x1200;
- QXGA - 2048x1536.

Для телевизионных и компьютерных панелей (16:9):

- WVGA – 854x480;
- HD 720 - 1280x720; HD 1080 - 1920x1080 (Full HD, HD Ready 1080P).

Глубина цветов определяется количеством бит на передачу каждого цветового канала. Для RGB 24 бита дают 28 x 28 x 28 = 224 = 16,777216 млн. цветовых оттенков. Для CMYK (Голубой, Пурпурный, Желтый, Черный) необходимо 32 бита. В современных ЖК-панелях при передаче цветов RGB по 17 бит на компоненту (всего 51 бит) достигается более 2 квадриллионов цветовых оттенков.

В состав блока RAMDAC входит память RAM. Она построена на статических запоминающих элементах и имеет высокое быстродействие (не путать с видеопамятью). Цифроаналоговый преобразователь DAC имеет три параллельных канала для трех цветовых компонентов. Частота преобразования определяется поддерживаемым разрешением и частотой кадров. Например, для разрешения 1024x768 и частоты кадров 70 Гц частота DAC должна быть не менее 75 МГц. В видеокартах NVIDIA и AMD частота RAMDAC равна 400 МГц при кадровой частоте 85 Гц.

В 2007 году производители GPU перешли от различающихся вершинных и пиксельных конвейеров к унифицированным потоковым процессорам, заменяющим как вершинные, так и пиксельные конвейеры. Графические процессоры 2014–2016 годов выпуска ATI Radeon Fiji XT и NVIDIA GM200–400 включают 4096 и 3072 потоковых процессора (для 32-битовых вычислений одинарной точности) при несколько различной внутренней архитектуре.

При этом производительность графических процессоров продолжает быстро увеличиваться.

13.3. Взаимодействие графического и центрального процессоров

Графический процессор не имеет средств прямого взаимодействия с устройствами ввода-вывода (кроме монитора), а также доступа к оперативной памяти компьютера. Поэтому управление графическим процессором осуществляется только через центральный процессор. Схема взаимодействия CPU и GPU приведена на рис. 13.5.

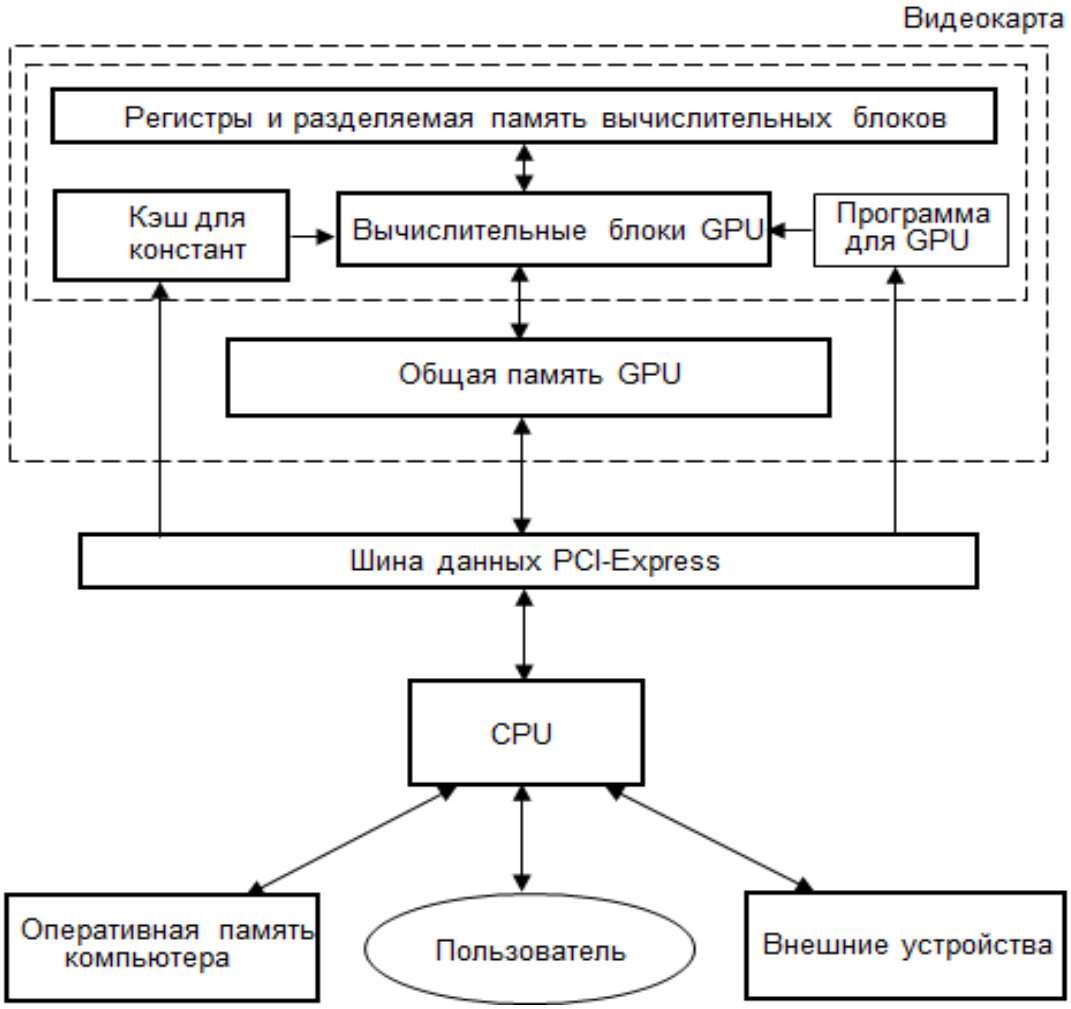


Рис.13.5. Схема взаимодействия CPU и GPU

Графические ускорители подключаются к системной плате персонального компьютера через высокоскоростную шину данных (в настоящее время PCI-Express). Посредством этой шины центральный процессор получает доступ к видеопамяти, а также к некоторым

разделам кеш-памяти, расположенной на самом графическом процессоре. Через эту же шину CPU загружает в графический процессор программу и запускает ее.

Перед запуском программы, исполняемой на GPU, центральный процессор передает графическому процессору данные двух видов:

- значения констант, используемых в программе;
- один или несколько больших массивов данных для потоковой обработки.

К константам необходим постоянный и быстрый доступ, поэтому они записываются в кэш-память (или регистры), расположенную на кристалле графического процессора. Массивы данных часто бывают настолько велики, что целиком в кэш-память не помещаются. Однако, при простейшей потоковой обработке каждый из элементов массивов данных используется только один раз. Так что для хранения этих массивов предназначена видеопамять (общая память), представляющая собой отдельные микросхемы на плате графического ускорителя. Она работает медленнее кэш-памяти и регистров, зато имеет бóльший объем (до нескольких гигабайтов).

Результаты своей работы графический процессор может сразу записывать в раздел видеопамяти, называемый буфером кадра, откуда они передаются на монитор. Но существует также возможность вообще не отображать расчет на экране, а копировать результаты из видеопамяти в оперативную память компьютера, где они становятся доступными для дальнейшей обработки центральным процессором. На этом и основано использование графических процессоров для вычислений общего назначения, не связанных с обработкой графики.

На схеме рисунка 13.5 указаны также типы программ, которые исполняются центральным и графическим процессорами на различных этапах обработки данных.

Иерархия памяти, доступной CPU и GPU процессорам.

Как показано выше, в программах для графических процессоров используется память нескольких разновидностей с различными характеристиками и назначением. Это так, поскольку время, затрачиваемое центральным и графическим процессорами на операции чтения данных из памяти и записи в память, является одним

из важнейших факторов, определяющих быстродействие программ для графических процессоров.

Использование памяти различных типов обусловлено необходимостью баланса между объемом памяти и скоростью доступа к данным. Разновидности памяти, имеющие наибольшую емкость, обычно характеризуются бóльшим временем доступа к данным, и наоборот. Быстродействие памяти, в свою очередь, определяется двумя характеристиками — латентностью и пропускной способностью.

Латентность — это время доступа к памяти, точнее, время ожидания процессором данных после запроса. Латентностью определяется производительность вычислений при решении задач, требующих частого обращения к различным по расположению неупорядоченным ячейкам памяти (произвольный доступ к памяти). Такой обмен с памятью характерен для интерактивных приложений (программы, интенсивно взаимодействующие с другими приложениями и с пользователями), а также для приложений, управляющих сложными процессами. Подобные алгоритмы обычно исполняются центральным процессором.

Пропускная способность памяти характеризует объем данных, которые могут быть переданы к процессору или от процессора за единицу времени. Высокая пропускная способность оказывается эффективнее низкой латентности в задачах, позволяющих организовать последовательный доступ к памяти — считывание (или запись) данных из ячеек памяти, расположенных друг за другом, непрерывным потоком. При поточно-параллельной обработке данных предпочтителен именно последовательный доступ к памяти, поэтому видеопамять, предназначенная для обмена данными с графическим процессором, должна обладать максимальной пропускной способностью даже в ущерб латентности.

На рис. 13.6 приведена еще одна схема, иллюстрирующая основные принципы использования памяти различных типов.

1. Регистры и кеш-память CPU – память небольшого (до нескольких мегабайт) объема, расположенная на кристалле CPU. Характеризуется минимальной латентностью, доступна только

центральному процессору. Этой памятью управляет CPU , записывая в нее данные, используемые наиболее часто.

2. Оперативная память компьютера – память большого (до десятков гигабайт) объема в форме микросхем, на системной плате. Имеет среднюю латентность и пропускную способность. Доступна только центральному процессору. Используется для хранения всех обрабатываемых CPU данных, превышающих объем кэша.

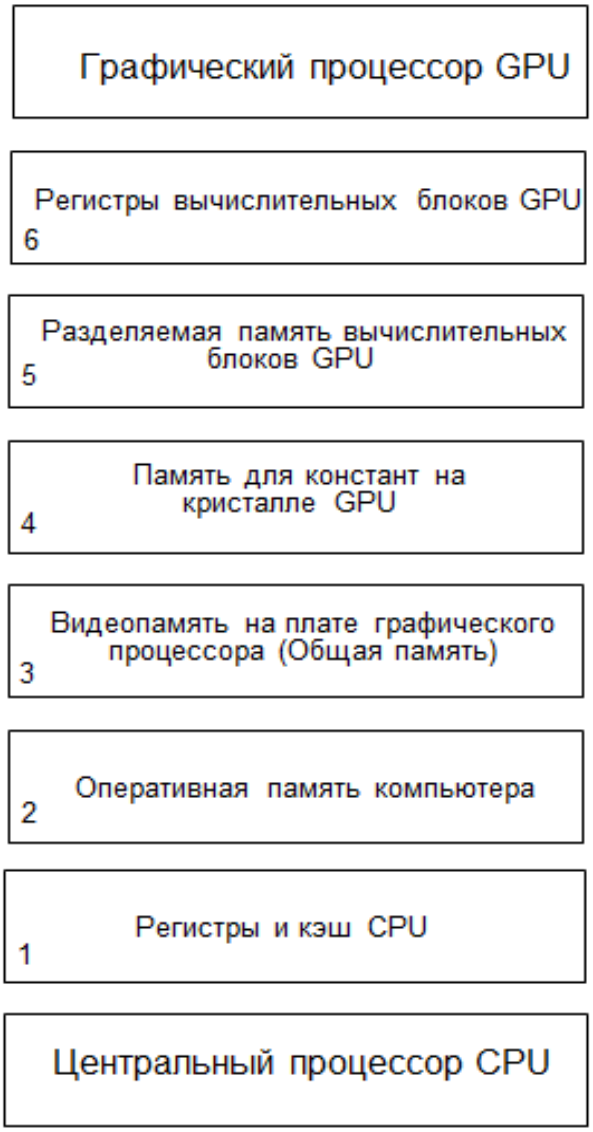


Рис.13.6. Использование памяти CPU и GPU процессорами

3. Видеопамять – память, в которую CPU копирует данные, предназначенные для обработки графическим процессором. Сюда же GPU записывает результаты расчета.

Видеопамять имеет большой объем (до нескольких гигабайт) и максимально возможную пропускную способность для потокового доступа к данным. Расположена на плате видеокарты, но не на самом кристалле GPU. Доступна как графическому, так и центральному процессорам для чтения и записи. Латентность со стороны GPU средняя, со стороны CPU очень большая (медленная передача данных).

4. Память для констант на кристалле GPU. Имеет низкую латентность со стороны GPU и предназначена для быстрого произвольного доступа GPU к константам. Доступна центральному процессору для записи констант, а графическому процессору – для их чтения.

5-6. Разделяемая память и регистры вычислительных блоков GPU. Они расположены на кристалле графического процессора и доступны только ему. Имеют очень низкую латентность (высокую скорость), но небольшой объем (десятки килобайт). Предназначены для произвольного доступа к данным со стороны параллельных вычислительных процессов

Отметим основные принципы разделения памяти между центральным и графическим процессорами. В персональном компьютере для хранения большей части данных и программ предназначена так называемая оперативная память. Микросхемы этой памяти расположены на системной плате (не на самом процессоре), они доступны центральному процессору для произвольных чтения и записи данных. Графическому процессору эта память не доступна. На плате графического ускорителя расположена видеопамять (общая память), технологически и по объему подобная оперативной памяти компьютера. Она доступна графическому процессору для чтения и записи в последовательном и в произвольном режиме, но оптимизирована для последовательного (потокового) доступа, поскольку предназначена для хранения массивов данных, обрабатываемых GPU в поточно-параллельном режиме. Видеопамять является основным средством обмена данными между графическим и центральным процессорами. Центральный процессор тоже имеет к этой памяти доступ для чтения и записи.

Перед началом вычислений на GPU он копирует исходные данные из оперативной памяти в видеопамять, а после окончания расчета копирует результаты обратно в оперативную память.

Обмен данными между центральным процессором и видеопамятью происходит сравнительно медленно, так что проводится обычно только в начале и конце расчета. Поскольку для размещения исходных потоков данных для GPU и результатов его работы видеопамять должна иметь сравнительно большой объем (на современных графических ускорителях до нескольких гигабайтов), она выпускается в виде отдельных микросхем и расположена не на самом кристалле графического процессора, а на плате графического ускорителя. Для повышения производительности GPU оснащены небольшой быстрой памятью (регистры процессора, память для констант, разделяемая память вычислительных блоков), расположенной прямо на кристалле процессора.

Графические процессоры обычно предлагаются в виде семейства микросхем с разными диапазонами стоимости производительности, сохраняющими программную совместимость друге другом. Наборы микросхем на основе Tesla GPU бывают до 16 узлов, которые NVIDIA называет мультипроцессорами. В начале 2010 года самая старшая версия называлась GeForce 8800 GTX, имела 16 мультипроцессоров и тактовую частоту 1,35 ГГц. Каждый мультипроцессор имел восемь многопоточных блоков для работы с числами с плавающей точкой одинарной точности и блоки обработки целых чисел, которые NVIDIA называла потоковыми процессорами.

Каждый из 16 мультипроцессоров в составе GeForce 8800 GTX имеет локальное программно-управляемое запоминающее устройство объемом 16 Кбайт плюс 8192 32-разрядных регистра. Система памяти 8800 GTX состоит из шести сегментов 900 МГц графической DDR3 DRAM-памяти, каждый шириной 8 байт и объемом 128 Мбайт. Общий объем памяти, таким образом, составляет 768 Мбайт.

Чтобы скрыть латентность памяти, каждый потоковый процессор имеет аппаратно-поддерживаемые потоки. Каждая группа из 32 потоков называется варпом (warp). Варп является блоком диспетчеризации, и активные потоки в варпе, до 32, выполняются в параллельном режиме SlMD-способом.

Но многопоточная система справляется с условиями, позволяя потокам расходиться по разным путям условных переходов. Когда потоки варпа идут по расходящимся путям, варп последовательно выполняет код по обоим маршрутам, делая неактивными некоторые потоки, что приводит к более медленному выполнению активных потоков. Как только условная часть завершится, оборудование снова объединяет потоки в полностью активный варп. Для достижения наивысшей производительности все 32 потока варпа нуждаются в совместном параллельном выполнении. Похожим образом оборудование также следит за течением адресов, поступающих от разных потоков, чтобы попытаться объединить отдельные запросы в меньшее количество передач более крупных блоков памяти для увеличения производительности при работе с памятью.

13.4. Системы программирования для GPU

Хотя графические процессоры имеют собственную систему команд, графический процессор управляется пользователем не напрямую, а через центральный процессор компьютера. При этом CPU в большинстве случаев исполняет взаимодействующие программы на разных уровнях управления графическим процессором:

- программа для центрального процессора: пользовательский интерфейс и плохо распараллеливаемые расчеты. Языки высокого уровня общего назначения (C#, C++);
- программа для GPU с использованием специализированных инструментов (HLSL, CUDA). Инициализируется и запускается центральным процессором;
- библиотеки API (DirectX, OpenCL, CUDA) обеспечивают и унифицируют взаимодействие пользовательских приложений с драйвером GPU;
- драйвер графического процессора. Управляет GPU с учетом архитектуры и технологических особенностей конкретного устройства.

Драйвер графического процессора.

На самом близком к графическому процессору уровне исполняется драйвер — программа, которая непосредственно управляет как самим GPU, так и видеопамятью. Фактически драйвер

работает как часть операционной системы. Он получает от других приложений запросы — задачи для обработки на графическом процессоре, после чего передает их на GPU в оптимальном порядке и необходимом формате. Функциями драйвера являются:

- распределение ресурсов графического процессора между несколькими приложениями;
- загрузка данных в видеопамять и регистры графического процессора, копирование данных из видеопамяти в оперативную память компьютера;
- загрузка в графический процессор и запуск на исполнение пользовательских программ;
- автоматическое распределение расчетов между параллельными вычислительными блоками графического процессора;
- исполнение стандартных алгоритмов обработки графики;
- регулирование частоты графического процессора для поддержания рабочей температуры.

Драйверы обычно разрабатываются самими производителями GPU, т.к. это требует детального знания их технического устройства и особенностей. Они обновляются для поддержки процессоров новых моделей. Благодаря существованию драйверов, пользовательские программы могут выполняться на различных GPU без перекомпиляции, если GPU и драйвер поддерживают функции, задействованные в программе.

Программирование приложений.

Для программирования трехмерных изображений необходимо использовать специализированные прикладные программные библиотеки, составляющие интерфейс прикладного программирования API (Application Program Interface)**.** Используются две библиотеки- OpenGL (Open Graphics Library) и DirectX.

Корпорацией Microsoft для Windows выпускается пакет мультимедийных библиотек API, который в целом называется DirectX. В этот пакет входит библиотека DirectCompute, предназначенная для реализации вычислений общего назначения на GPU. Начиная с Windows 98 пакет DirectX поставляется вместе с операционной системой, так что практически входит в ее состав.

Интерфейс программирования приложений DirectCompute поддерживается операционными системами Windows, начиная с Windows Vista. В состав DirectX входит компилятор языка высокого уровня для пользовательского программирования графических процессоров HLSL (High Level Shading Language). Этот язык разработан Microsoft специально для DirectX, он совершенствуется вместе с этим пакетом библиотек.

OpenGL и OpenCL.

OpenGL (Open Graphics Library — открытая графическая библиотека) представляет собой спецификацию, определяющую независимый от языка программирования кросс-платформенный программный интерфейс API для написания приложений, использующих двумерную и трехмерную компьютерную графику. Спецификация не является конкретной библиотекой, она только описывает набор функций и точное их поведение. На основе этой спецификации OpenGL разработчики создают библиотеки функций, называемые реализациями. Реализации по возможности используют аппаратные средства GPU, а другие требуемые функции эмулируют программно. Чтобы называться реализациями OpenGL, библиотеки должны удовлетворять определенным тестам на соответствие (conformance tests). Существует возможность написания на OpenGL вычислительных шейдеров (compute shaders), реализующих вычисления общего назначения на GPU.

Разработана новая спецификация OpenCL (Open Computing Language — открытый язык для вычислений), включающая в себя интерфейсы программирования приложений и язык программирования для гетерогенных систем, имеющих в своем составе центральные и графические процессоры, как и другие процессоры для высокопроизводительных параллельных вычислений. Существуют многочисленные реализации OpenCL, в том числе от Nvidia, AMD, IBM, Apple и Intel.

Главным преимуществом OpenGL и OpenCL перед DirectX является существование их реализаций не только для Windows, но и для большинства других операционных систем, в том числе Unix-подобных систем (Linux, Mac OS), систем игровых приставок. Такие реализации разрабатываются производителями центральных и

графических процессоров, поэтому реализации максимально используют аппаратное ускорение вычислений.

Пользовательские приложения.

На верхнем уровне управления графическим процессором находятся пользовательские приложения — программы, разрабатываемые для решения конечных задач, таких как создание трехмерных сцен, проведение математических расчетов. Структура пользовательского приложения, использующего API типа DirectCompute или OpenCL, показана на рис.13.7.

При использовании DirectCompute или OpenCL приложение состоит по крайней мере из двух частей, для написания которых используются различные языки программирования. Одна из этих частей — это программа для графического процессора, которую обычно называют **вычислительным ядром**, а другая часть — **центральный процессор.**

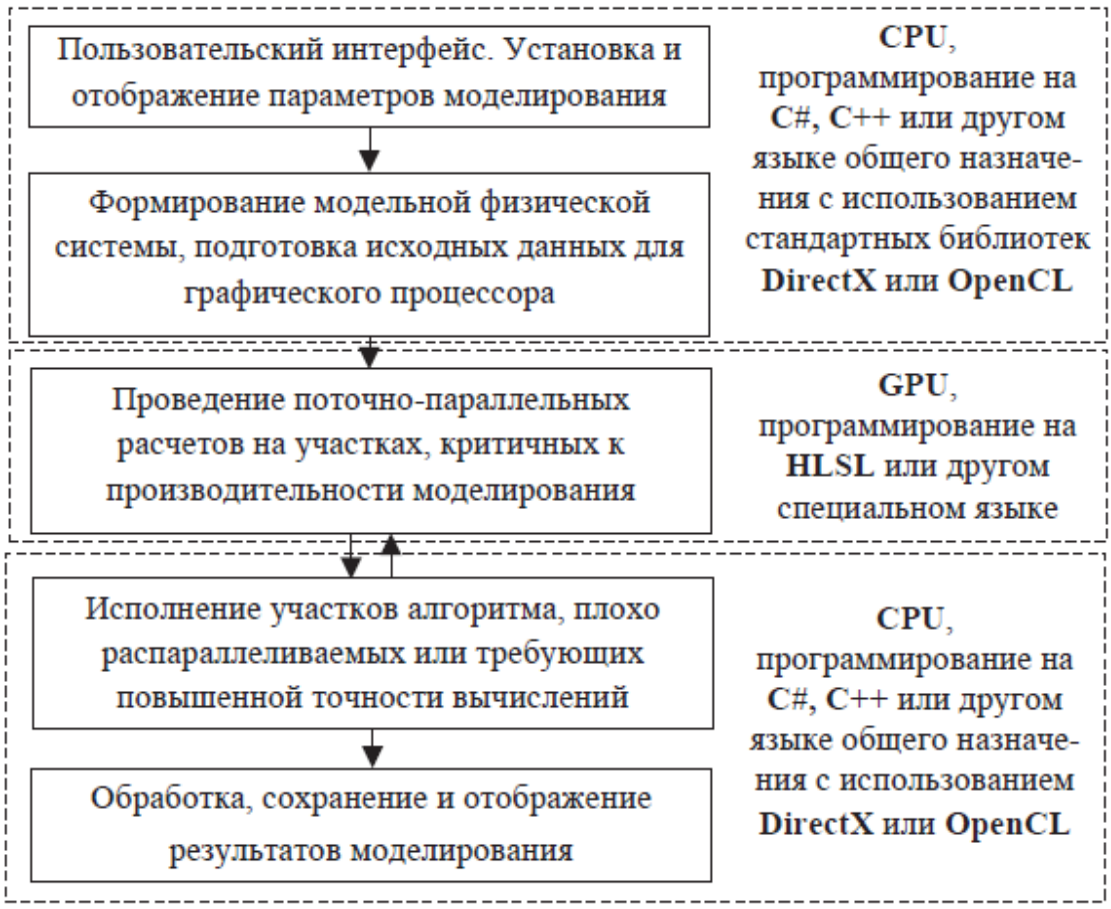


Рис.13.7. Структура пользовательского приложения

Для написания вычислительных ядер используются специальные языки программирования. При работе с DirectX это может быть либо язык высоко уровня HLSL, либо низкоуровневый ассемблер. Вычислительное ядро хранится в отдельном текстовом файле, оно компилируется и загружается в память GPU средствами DirectX или OpenCL непосредственно перед запуском. Эти API используются также для загрузки в графический процессор и видеопамять данных и констант, необходимых для расчета.

Часть пользовательского приложения, которая исполняется на центральном процессоре, обеспечивает:

- взаимодействие с пользователем (пользовательский интерфейс);
- получение исходных данных от пользователя или из внешних источников, сохранение результатов расчета;
- подготовку и запуск расчетов на графическом процессоре;
- исполнение участков расчетного алгоритма, предназначенных для центрального процессора.

Программы для центрального процессора обычно разрабатываются на традиционных языках высокого уровня, в частности, на C++ или C#, которые достаточно легко обращаются к процедурам из OpenCL или DirectX.

Программно-аппаратная платформа NVIDIA CUDA.

Средства программирования GPU, описываемые выше, применимы для программирования графических процессоров всех производителей, в частности, компаний NVIDIA и AMD. Однако внутренние архитектуры GPU разных производителей и поколений существенно различаются между собой, а возможности использования особенностей каждой из архитектур универсальными системами программирования ограничены. Преимущества конкретных GPU доступны при использовании специализированных инструментов программирования, одним из которых является NVIDIA CUDA.

Платформа для параллельных вычислений NVID - архитектура единого вычислительного устройства CUDA, фактически представляет собой программно-аппаратный комплекс, в котором устройство графического процессора и его программная модель

соответствуют подходам к организации параллельных вычислений, разрабатываемым NVIDIA.

Технология CUDA позволяет программировать только совместимые GPU (производства NVIDIA), однако позволяет управлять ими на более низком уровне, чем DirectX и OpenCL. В частности, преимуществами CUDA являются:

- доступ к программируемой разделяемой памяти;
- произвольная адресация при записи в память;
- значительно ускоренное взаимодействие CPU и GPU, некоторые операции выполняются асинхронно.

Интерфейс программирования приложений реализован в форме расширения языка C++.

Можно подвести некоторые итоги. На сегодня возможными платформами программирования GPU для вычислений общего назначения являются NVIDIA CUDA, Microsoft DirectX и OpenCL. Все платформы достаточно функциональны и динамично развиваются.

Преимуществами CUDA являются лучшая управляемость параллельных вычислений, а также использование обычного языка C++ с дополнительными библиотеками.

У среды программирования CUDA также есть своя собственная терминология. Программа CUDA является унифицированной C/C++ программой, предназначенной для гетерогенного центрального процессора и системы GPU. Она выполняется на центральном процессоре и осуществляет диспетчеризацию параллельной работы для GPU. Эта работа состоит из передачи данных от оперативной памяти и из диспетчеризации потоков. Поток является частью программы для GPIJ. Программисты указывают количество потоков в поточном блоке и количество поточных блоков, которое требуется запустить на выполнение на GPU. Программисты заботятся о поточных блоках, поскольку все потоки в блоке подвергаются диспетчеризации для запуска на одном и том же мультипроцессоре, поэтому они совместно используют одну и ту же локальную память. Благодаря этому они могут обмениваться данными через загрузки и сохранения, а не через сообщения.

Компилятор CUDA распределяет регистры каждому потоку при том ограничении, что количество регистров за один поток, умноженное на количество потоков на поточный блок, не должно превышать число, равное 8192 регистрам на один мультипроцессор.

Потоковый блок может иметь до 512 потоков. Каждая группа из 32 потоков в потоковом блоке запакована в варпы. Большие потоковые блоки более эффективны, чем небольшие, и они могут уменьшаться до одного блока. Как уже ранее упоминалось, потоковые блоки и варпы, состоящие менее чем из 32 потоков, работают менее эффективно, чем заполненные.

Аппаратная диспетчеризация старается, по возможности, спланировать выполнение нескольких потоковых блоков на мультипроцессоре. Если это удается, диспетчер также динамически делит 16 Кбайт локального запоминающего устройства между различными потоковыми блоками.

Достоинство платформы Microsoft DirectX (включая язык HLSL и вычислительные шейдеры) — совместимость приложений с графическими процессорами всех ведущих производителей (в частности, AMD и NVIDIA), что актуально, поскольку по теоретической производительности GPU AMD и NVIDIA примерно равны.

В ближайшие годы будет актуальной разработка приложений физико-математического моделирования как для DirectX/OpenCL, так и для CUDA.

Современные графические процессоры могут эффективно решать очень многие задачи. Основное требование к таким задачам — возможность распараллеливания по данным или по задачам, а также алгоритм, в котором вычисления превалируют над условными переходами и обменом данными между параллельными процессами. Вычисления на графических процессорах можно успешно применять в таких областях, как матричные преобразования, астрофизические расчеты, оптимизация, дискретное преобразование Фурье, кодирование видео, обработка речи, визуализация строения вещества, нейронные сети, биофизические расчеты, искусственный интеллект, симуляция реальных процессов.

Вопросы для контроля

1. Каковы основные этапы формирования изображения?
2. В чем особенности аппаратной архитектуры NVIDIA GPU?
3. Чем отличиется GPU от CPU с точки зрения доступа в память?
4. Назовите виды памяти GPU-процессоров?
5. Каков порядок взаимодействия между уровнями памяти GPU?
6. Какие функции выполняет библиотека DirectX10?
7. Опишите схему взаимодействия CPU и GPU?
8. Для каких целей используется видеопамять?
9. Опишите функции драйвера CPU.
10. Дайте краткую характеристку пакетам Open GL и OpenCL.
11. Какая часто пользовательского приложения выполняется на CPU?

ГЛАВА 14. АППАРАТНАЯ РЕАЛИЗАЦИЯ И ПРОГРАММНАЯ ПОДДЕРЖКА РАБОТЫ ПРОЦЕССОРА

Средства автоматизированного проектирования и современные технологии изготовления электронных компонентов дали возможность значительно удешевить аппаратную часть компьютера. В то же время значительное расширение сфер применения, увеличение числа различной категории пользователей, расширение типов услуг со стороны компьютера привело к тому, что стоимость программного обеспечения стала превосходить стоимость аппаратных средств.

В состав программного обеспечения входят как прикладные программы пользователя (так называемые приложения), так и системные программы управления режимами функционирования процессора:

- операционные системы, обеспечивающие доступ к ресурсам процессора;
- сервисные системы, облегчающие работу пользователя;
- инструментальные средства трансляции и редактирования программ;
- тестовые и диагностические системы обслуживания.

Основной задачей программного обеспечения является управление процессом обработки и связь с ресурсами (рис.14.1)

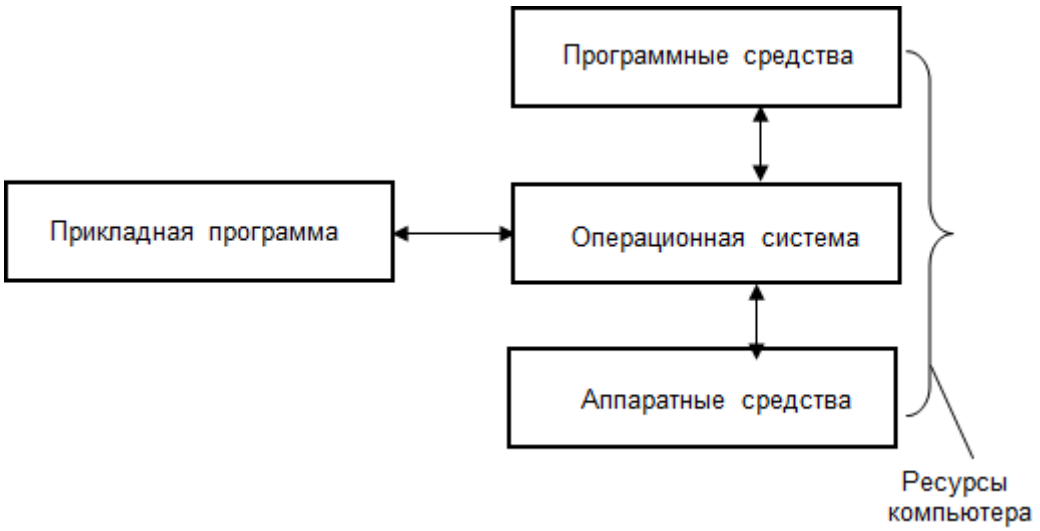


Рис.14.1. Связи между ресурсами компьютера

Под управлением ресурсами понимается упрощение доступа к ресурсам и распределение ресурсов между программами–пользователями. Управление процессом обработки – это обеспечение эффективного функционирования процессора во всех предусмотренных режимах:

- однопользовательский и многопользовательский режим;
- режим разделения времени (диалоговый);
- режим реального времени;
- работа в сетевом режиме.

В настоящее время компьютер представляет собой практически вычислительную систему, состоящую из множества параллельно функционирующих узлов, каждый из которых выполняет свою отдельную задачу обработки и преобразования данных. Компьютерная система состоит из модулей – процессора, памяти и внешних устройств, каждое из которых управляется своим контроллером, соединенных между собой системной шиной. В современных компьютерных системах имеются такие модули, как процессор, память, общая шина PCI, порты – USB, COM, IEEE 1394, SCSI, HDMI. Каждый порт допускает подключение к компьютеру отдельного, самостоятельного типа внешних устройств. Беспроводной интерфейс Bluetooth используется для связи компьютера с мобильным устройством, наушниками, плеером.

Модули компьютерной системы – процессор, память и внешние устройства с их контроллерами – функционируют параллельно (рис.14.2). Так как в архитектуре компьютера много всевозможных вспомогательных процессоров в составе различных адаптеров и карт, на данной схеме процессор, выполняющий систему команд, представлен как центральный процессор компьютерной системы (ЦП).

Каждый контроллер имеет локальный буфер, через который осуществляется обмен с устройством. При необходимости выполнения операции по обмену данными процессор информирует систему об этом и реализуется режим прерывания.

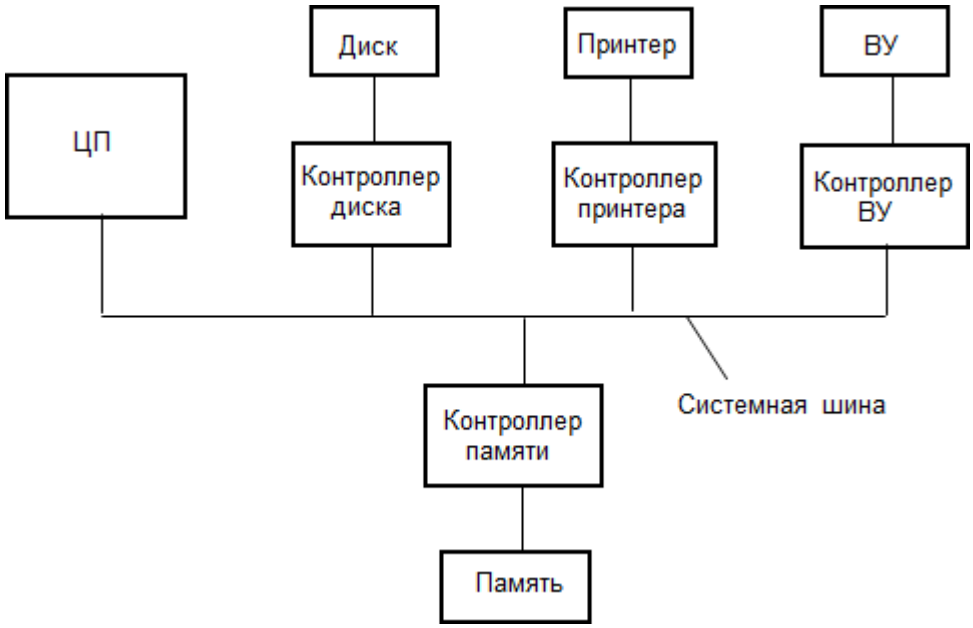


Рис.14.2. Архитектура аппаратных средств компьютера

Адаптеры и контроллеры – это очень упрощенные варианты процессоров, предназначенные для согласованного обмена данными без привлечения ЦП. Одним из основных механизмов обмена данными является обработка прерываний.

14.1. Обработка прерываний от внешних устройств

Во время выполнения процессором текущей программы внутри машины и в связанной с ней внешней среде могут возникать события, требующие немедленной реакции на них со стороны машины. Реакция состоит в том, что процессор прерывает обработку текущей программы и переходит к выполнению некоторой другой программы, специально предназначенной для данного события. По завершении этой программы процессор возвращается к выполнению прерванной программы. Принципиально важным является то, что моменты возникновения событий, требующих прерывания программ, заранее неизвестны и поэтому не могут быть учтены при программировании.

Каждое событие, требующее прерывания, сопровождается сигналом, оповещающим процессор (запросы прерывания). Программу, затребованную запросом прерывания, назовем прерывающей программой, противопоставляя ее прерываемой программе, выполнявшейся процессором до появления запроса.

Запросы на прерывания могут возникать внутри самого компьютера и во внешней среде. К первым относятся, например, запросы при возникновении в процессоре таких событий, как переполнение разрядной сетки, попытка деления на нуль, выход из установленной для данной программы области памяти, а также затребование периферийным устройством операции ввода-вывода, завершение операции ввода-вывода периферийным устройством или возникновение при этой операции особой ситуации. Хотя некоторые из указанных событий порождаются самой программой, моменты их появления, как правило, невозможно предусмотреть. Внутренние события, порожденные работой самой программы (переполнение, деление на нуль, отрицательный результат) рассмотрены ранее при изучении работы процессора. Запросы во внешней среде, о которых речь пойдет далее, могут возникать во внешней среде. В первую очередь это относится к организации параллельной во времени работы процессора и периферийных устройств машины, а также к использованию компьютера для управления в реальном времени технологическими процессами. При работе компьютера в составе сети особенно много запросов прерывания поступает через сетевую карту от других компьютеров этой сети.

Возможность прерывания программ - важное архитектурное свойство компьютера, позволяющее эффективно использовать производительность процессора при наличии нескольких протекающих параллельно во времени процессов. Чтобы компьютер мог, не требуя больших усилий от программиста, реализовывать с высоким быстродействием прерывания программ, машине необходимо придать соответствующие аппаратурные и программные средства, совокупность которых получила название системы прерывания программ.

Основными функциями системы прерывания являются:

- запоминание состояния прерываемой программы и осуществление перехода к прерывающей программе,

- восстановление состояния прерванной программы и возврат к ней.

При наличии нескольких источников запросов прерывания должен быть установлен определенный порядок (дисциплина) в

обслуживании поступающих запросов. Другими словами, между поступающими запросами и соответствующими прерывающими программами должны быть установлены приоритетные соотношения, определяющие, какой из нескольких поступивших запросов подлежит обработке в первую очередь, и устанавливающие, имеет право или не имеет данный запрос (прерывающая программа) прерывать ту или иную программу. Приоритетный выбор запроса для исполнения входит в процедуру перехода к прерывающей программе.

Пример программы обработки прерывания от внешних устройств представлен на рис.14.3.

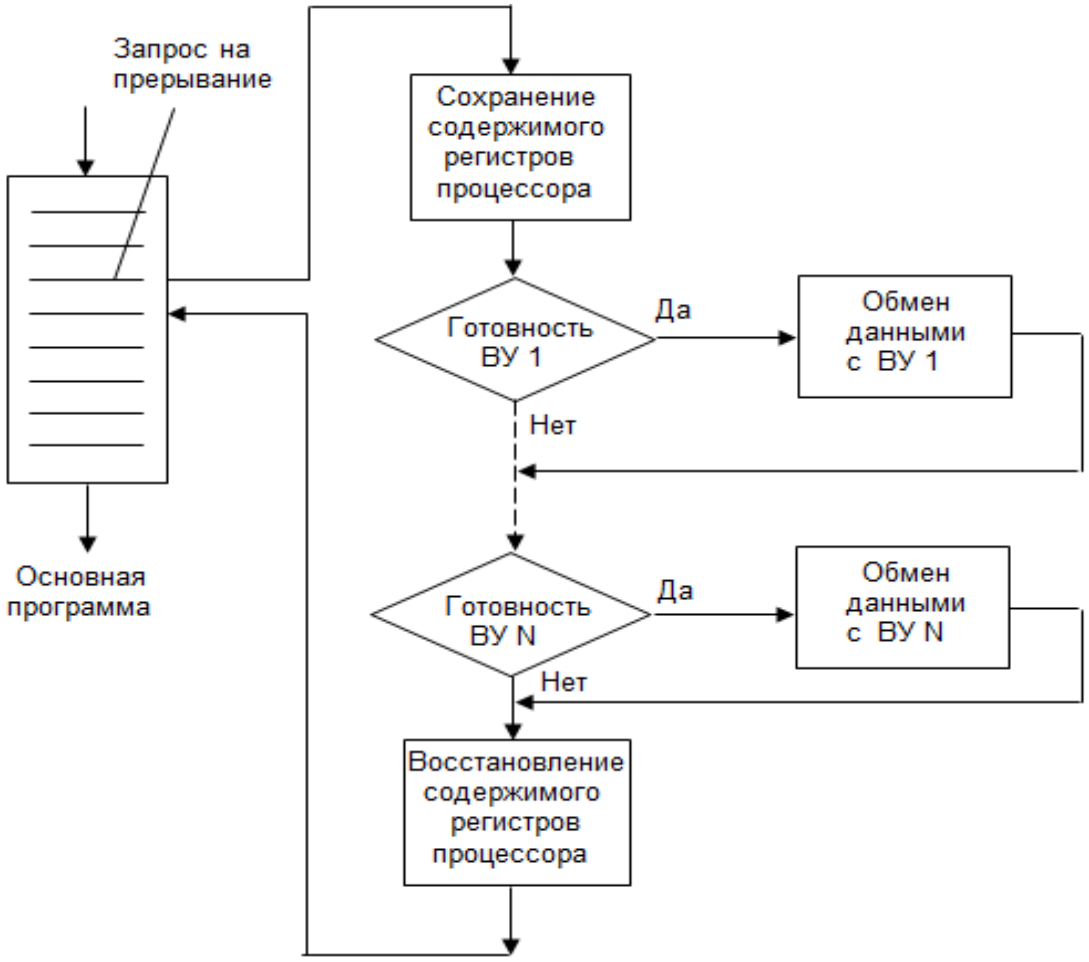


Рис.14.3. Блок-схема программы обработки прерывания

После поступления запроса от ВУ основная программа обработки прерывается, код последней выполненной команды сохраняется, происходит обмен данными с соответствующим внешним устройством, после чего происходит восстановление содержимого следующей после выполненной команды регистра.

Реализация ввода/вывода по прерываниям

При реализации ввода/вывода по прерываниям необходимо дать ответы на два вопроса. Во-первых, определить, каким образом ЦП может выяснить, какой из МВВ и какое из подключенных к этому модулю ВУ выставили запрос. Во-вторых, при множественных прерываниях требуется решить, какое из них должно быть обслужено в первую очередь.

При идентификации устройства возможны три основных метода:

- множественные линии прерывания;
- программная идентификация;
- векторное прерывание.

Наиболее простой подход к решению проблемы определения источника запроса — применение **множественных линий прерывания** между процессором и модулями ввода/вывода, хотя выделение слишком большого количества управляющих линий для этих целей нерационально. Более того, даже если присутствует несколько линий прерывания, желательно, чтобы каждая линия использовалась всеми МВВ, при этом для каждой линии действует один из двух остальных методов идентификации устройства.

При **программной идентификации**, обнаружив запрос прерывания, процессор переходит к общей программе обработки прерывания, задачей которой является опрос всех МВВ с целью определения источника запроса. Для этого может быть выделена специальная командная линия опроса. Процессор помещает на адресную шину адрес опрашиваемого МВВ и формирует на этой линии сигнал опроса. Реакция модуля зависит от того, выставлял он запрос или нет. Возможен и иной вариант, когда каждый МВВ включает в себя адресуемый регистр состояния. Тогда процессор считывает содержимое регистра состояния каждого модуля, после чего выясняет источник прерывания. Когда источник прерывания установлен, процессор переходит к программе обработки прерывания, соответствующей этому источнику. Недостаток метода программной идентификации заключается в больших временных потерях.

Наиболее эффективную процедуру идентификации источника прерывания обеспечивают аппаратурные методы, в основе которых лежит идея **векторного прерывания.** В этом случае, получив

подтверждение прерывания от процессора, выставившее запрос устройство выдает на шину данных специальное слово, называемое «вектором прерывания». Слово содержит либо адрес МВВ, либо какой-нибудь другой уникальный идентификатор, который процессор интерпретирует как указатель на соответствующую программу обработки прерывания. Такой подход устраняет необходимость в предварительных действиях с целью определения источника запроса прерывания. Реализуется он с помощью хранящейся в оперативной памяти таблицы векторов прерывания (рис.14.4), где содержатся адреса программ обработки прерываний.

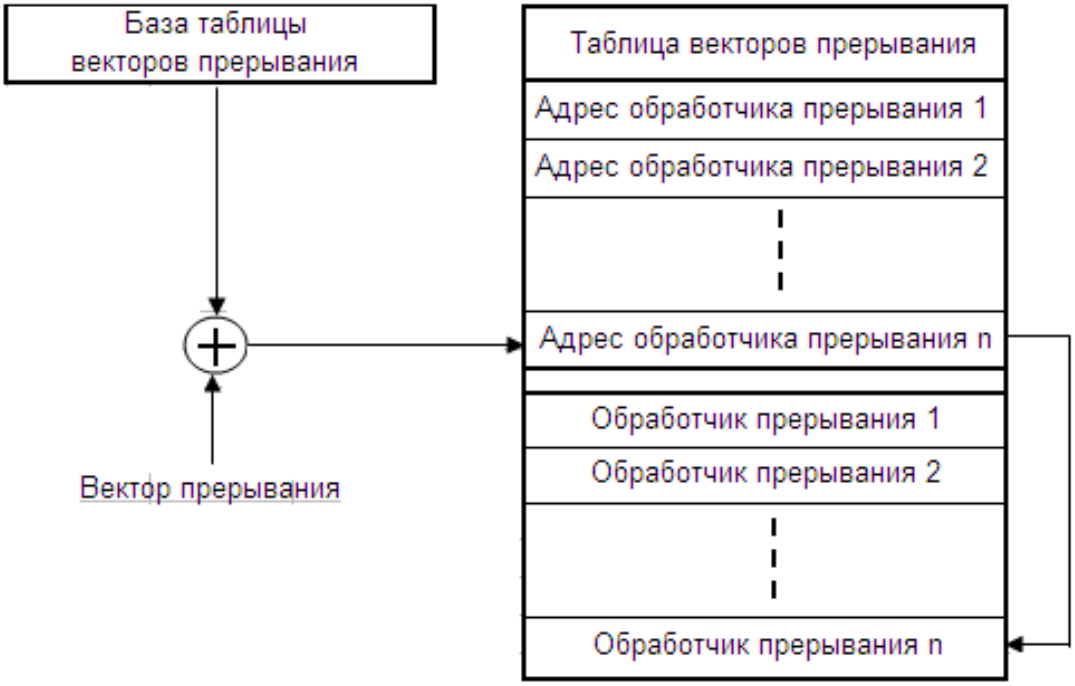


Рис.14.4. Идентификация запроса с помощью вектора прерывания

Входом в таблицу служит вектор прерывания. Начальный адрес таблицы (база) обычно задается неявно, то есть под таблицу отводится вполне определенная область памяти. Внешнее устройство, запросившее прерывание, идентифицирует себя с помощью специального кода, посылаемого процессору по шине. Этот код с помощью созданной базы способен определять начальный адрес программы обработки прерывания. Длина адреса обычно составляет от 4 до 8 разрядов. Оставшаяся часть адреса формируется процессором на основе данных базы.

После анализа кода вектора прерывания из базы выбирается соответствующий адрес программы – обработчика данного прерывания. По найденному адресу из памяти читается программа – обработчик прерывания, которая запускается процессором. После ее исполнения и ликвидации причины прерывания, компьютер возвращается к первой невыполненной команде основной программы.

14.2. Общая схема взаимодействия процессора с внешними устройствами

Устройства ввода-вывода передают в компьютер и получают от него, как правило, большой объем информации, который не может быть временно размешен только в регистрах процессора. Поэтому информация передается из внешнего устройства в оперативную память и поступает на внешнее устройство из оперативной памяти.

Системный интерфейс - это набор цепей, связывающих процессор с памятью и внешними устройствами, алгоритм передачи сигналов по этим цепям, их электрические параметры и тип соединительных элементов. Подключение любого внешнего устройства к компьютеру осуществляется через его устройство управления - **контроллер** ВУ (рис.14.5).

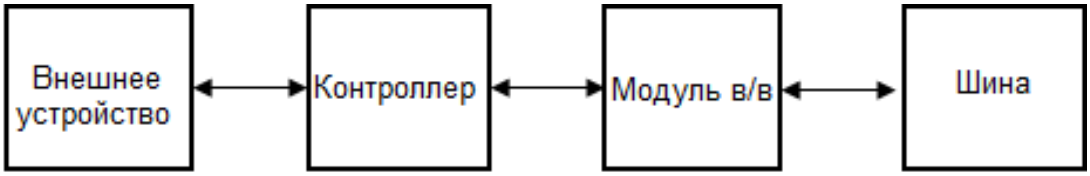


Рис.14.5. Схема подключения ВУ к системной шине

Таким образом, в системах ввода-вывода большинства современных компьютеров можно выделить два уровня сопряжения внешних устройств с процессором и памятью. На первом уровне контроллеры сопрягаются с процессором и памятью через системный интерфейс (модуль ввода/вывода), который обеспечивает объединение отдельных устройств в единую систему.

На втором уровне сопряжения контроллеры посредством шин связи соединяются с внешними устройствами.

Операции обмена информацией.

С помощью периферийных устройств осуществляется связь узлов процессора с различными источниками (УВыв) и

потребителями (УВв) информации. Функции внешних устройств достаточно сложны, однако из них можно выделить две основные:

- хранение информации в том или ином физическом представлении на разных носителях данных,
- преобразование информации соответственно функциям, выполняемым устройством.

На рис. 14.6 представлена упрощенная схема взаимодействия внешних устройств с процессорными узлами. Взаимодействие осуществляется путем использования двух операций:

- операции «Записи» (ВЫВОДА) – перенесения информации из оперативной памяти на внешнее устройство,
- операции «Чтения» (ВВОДА) – перенесения информации из внешнего устройства в оперативную память.

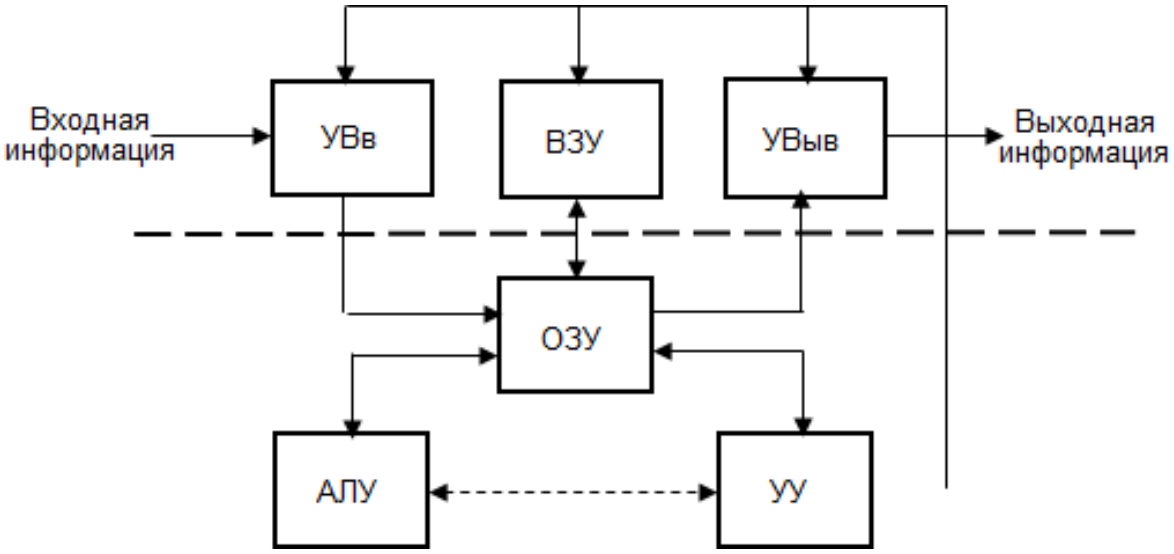


Рис.14.6. Схема взаимодействия внешних устройств с процессором

Так как большинство внешних устройств позволяет человеку общаться с компьютером на языке слов и десятичных чисел, а в компьютере информация представляется в виде последовательности двоичных чисел (0 и 1), в устройствах ввода-вывода производится кодирование (декодирование) пересылаемой информации. Существуют различные способы кодирования символьной информации, поступающей из устройств в компьютер. Следует помнить, что один внешний символ кодируется комбинацией битов, составляющих один байт информации в памяти.

14.3. Программная поддержка режимов функционирования

Основным элементом системных программ компьютера, обеспечивающих доступ к ресурсам компьютера, являются операционные системы (ОС). Вариантов построения ОС для различного типа вычислительных машин много, есть общие, схожие компоненты, есть и различия в структуре и функционировании различных ОС. Для упрощения будут рассматриваться ОС персональных компьютеров.

ОС является такой же программой, как и все другие программы компьютера. Она базируется на системе команд данного процессора, но добавляет к системе команд свои системные вызовы, которые представляют собой команды вызова определенной службы ОС (работа, с файлами, каталогами, памятью, идентификация пользователей, установка приоритетов). Эти уже относятся к категории управления ресурсами и управления данными через основной исполнительный элемент компьютера - процессор.

Основными компонентами ОС являются следующие программы.

Управление процессами. Процесс – это программа пользователя в ходе ее выполнения в компьютерной системе. ОС управляет работой процессов, их распределением по процессорам и ядрам системы, порядком их выполнения и размещения в памяти, их синхронизацией при параллельном решении частей одной и той же задачи разными процессами. Большинство современных компьютеров работает в мультизадачном (мультипрограммном) режиме, т.е. его операционная система организует параллельное выполнение нескольких вычислительных процессов. Эти вычислительные процессы чаще всего не зависят друг от друга. Каждому процессу ОС выделяет затребованные ресурсы, обеспечивая защиту каждого процесса от невмешательства других.

Другими словами, ОС поддерживает их обособленность: у каждого процесса имеется свое виртуальное адресное пространство, свои ресурсы в виде файлов. При этом процессор одновременно используется несколькими процессами обработки.

Большинство современных ОС позволяют управлять процессами динамически, при этом проблемой является обеспечение синхронизации взаимодействующих вычислительных процессов

(взаимодействующие процессы совместно используют некоторые общие переменные) при разделении ресурсов. При этом используются механизмы блокировки памяти (если это разделяемый ресурс), механизмы управления системой прерывания с применением, так называемых почтовых ящиков, семафоров. Правильно организованная синхронизация процессов позволяет избежать конфликтов между процессами. Управление процессами обработки подразумевает не только синхронизацию и поддержку взаимодействия между процессами, но и планирование процессов на основе приоритетов.

Управление памятью. Оперативная (основная) память может рассматриваться как большой массив. Операционная система распределяет ресурсы памяти между процессами, выделяет память по запросу, освобождает ее при явном запросе или по окончании процесса, хранит списки занятой и свободной памяти в системе.

Управление файлами. Файл – это логическая единица размещения информации на внешнем устройстве, например, на диске. ОС организует работу пользовательских программ с файлами, создает файлы, выполняет их открытие и закрытие и операции над ними (чтение и запись), хранит ссылки на файлы в директориях (папках) и обеспечивает их поиск по символьным именам.

Управление системой ввода-вывода. Как уже отмечалось, в компьютерной системе имеется большое число внешних устройств (принтеры, сканеры, устройства управления компакт-дисками), управляемых специальными контроллерами и драйверами – низкоуровневыми программами управления устройствами, выполняемыми в привилегированном режиме. ОС управляет всеми этими аппаратными и программными компонентами, обеспечивая надежность работы внешних устройств, эффективность их использования, диагностику и реконфигурацию в случае их сбоев и отказов. Существует несколько описанных ранее способов организации ввода/вывода: программный, по прерываниям, и с использованием прямого доступа к памяти. В реализации процедур ввода/вывода участвуют модули ввода/вывода и порты. Они имеют в общем случае регистры данных, состояния и управления, а также узлы исполнения по шинам адресов. Доступ к портам устройств имеют управляющие программы – драйверы. Динамическая загрузка,

инициализация, регистрация и выгрузка драйвера программы реализуются диспетчером ввода/вывода ОС. Драйвер развязывает физические устройства от прикладных программ и ОС.

Управление памятью. ОС отвечает за следующие действия, связанные с управлением памятью:

- **о**тслеживание того, какие части памяти в данный момент используются и какими процессами. Как правило, ОС организует для каждого процесса свою виртуальную память – расширение основной памяти путем хранения ее образа на диске и организации подкачки в основную память фрагментов (страниц или сегментов) виртуальной памяти процесса и ее откачки по мере необходимости;

- **с**тратегия загрузки процессов в основную память, по мере ее освобождения. При активизации процесса и его запуске или продолжении его выполнения процесс должен быть загружен в основную память, что и осуществляется операционной системой. При этом, возможно, какие-либо не активные в данный момент процессы отсылаются на диск;

- выделение и освобождение памяти по мере необходимости. Длина участков выделяемой и освобождаемой памяти может быть различной. ОС хранит список занятой и свободной памяти. При интенсивном использовании памяти может возникнуть ее фрагментация – дробление на мелкие свободные части, вследствие того, что при запросах на выделение памяти длина найденного сегмента оказывается немного больше, чем требуется, и остаток сохраняется в списке свободной памяти.

Внешняя память – это расширение оперативной памяти процессора более медленными, но более емкими и постоянно хранящими информацию видами памяти (диски, ленты). При управлении внешней памятью ОС решает задачи, аналогичные задачам управления основной памятью, - выделение памяти по запросу, освобождение памяти, хранение списков свободной и занятой памяти. ОС поддерживает также использование кэш-памяти для оптимизации обращения к внешней памяти.

Поддержка сетей. Очень часто компьютер работает в составе локальных и глобальных сетей. Операционная система обеспечивает использование сетевого оборудования (сетевых карт или адаптеров),

вызов соответствующих драйверов, поддержку удаленного взаимодействия с файловыми системами, находящимися на компьютерах сети, удаленный вход на другие компьютеры сети и использование их вычислительных ресурсов, отправку и получение сообщений по сети, защиту от сетевых атак.

Система защиты. Согласно современным принципам надежных и безопасных вычислений при работе должны быть обеспечены безопасность и защита от внешних атак, конфиденциальность личной и корпоративной информации, диагностика и исправления ошибок и неисправностей. ОС обеспечивает защиту компонент компьютерной системы, данных и программ, поддерживает фильтрацию сетевых пакетов, обнаружение и предотвращение внешних атак, хранит информацию обо всех действиях над системными структурами, полезную для анализа атак и борьбы с ними.

Система поддержки командного интерпретатора. Командный интерпретатор представляет собой находящийся в оперативной памяти код исполняемой программы, реагирующий на символьные команды, вводимые пользователем или читаемые из командного файла. Благодаря ему пользователь взаимодействует с компонентами ОС, локальными и сетевыми компьютерными ресурсами. Типичные команды – это получение информации об окружении, установка и смена текущей рабочей директории, пересылка файлов, компиляция и выполнение программ, получение информации о состоянии системы и выполнении своих процессов.

Графическая оболочка – подсистема ОС, реализующая графический пользовательский интерфейс пользователей и системных администраторов с операционной системой. Разумеется, использование одного лишь командного языка и системных вызовов неудобно, поэтому простой и наглядный графический пользовательский интерфейс с ОС необходим. Имеется много известных графических оболочек для операционных систем, причем их возможности очень похожи друг на друга.

Операционная система представляет собой сложный комплекс управляющих и обрабатывающих программ, поэтому кроме вышеуказанных основных систем имеется множество дополнительных модулей, реализующих те или иные механизмы

организации вычислительного процесса. Эти модули в различных ОС имеют разный состав и назначение в зависимости от цели создания. Однако, все вышеперечисленные процедуры управления вычислительным процессом не могут быть реализованы без непосредственного участия процессора.

Некоторые разновидности современных ОС представлены на рис.14.7.

Классификационный признак	Разновидности операционных систем		
Поддержка мультипроцессорной обработки	однопроцессорные	SMP	ASMP
Предоставление процессорного времени	реального времени	с разделением времени	
Поддержка вычислений	локальные	сетевые	распределенные
Организация	объектно-ориентированные	операционные платформы	прикладные среды
Число обслуживаемых пользователей	однопользовательские	многопользовательские	
Число одновременно решаемых задач	однозадачные	мультизадачные	

Рис.14.7. Разновидности операционных систем

1. С точки зрения масштабируемости ОС могут быть однопроцессорные и мультипроцессорные, поддерживающие выполнение заданий на нескольких процессорах. К таким ОС относятся UNIX, MS Windows NT. Среди них выделяют ОС с симметричной (SMP) и ассиметричной (ASMP) мультипроцессорной обработкой.

2. С точки зрения предоставления времени процессора ОС могут быть системами реального времени с гарантированным временем обслуживания каждого обращения пользователя или внешнего терминала и разделения времени (одновременный диалоговый (интерактивный) доступ нескольких пользователей с разделением

между ними каждого заранее фиксированного интервала машинного времени, или в соответствии с иной дисциплиной обслуживания).

Рассмотрим примеры некоторых наиболее широко применяемых ОС.

Операционная система UNIX. Это многопользовательская, многозадачная ОС, включает в себя достаточно мощные средства защиты программ и файлов различных пользователей. Большая часть системных программ ОС UNIX написана на языке С и она (за исключением небольшого ядра) является машинно- независимой, что обеспечивает ее высокую мобильность и легкую переносимость прикладных программ на компьютеры другого назначения и другой архитектуры. Важной особенностью ОС семейства UNIX является ее модульность и обширный набор сервисных программ, система особенно эффективна для прикладных программистов.

UNIX поддерживает иерархическую файловую структуру, виртуальную память, многооконный интерфейс, многопроцессорные системы, системы управления базами данных, неоднородные вычислительные сети. Большое распространение UNIX и ее версия Linux получили в сети Интернет, где важнейшее значение имеет независимость ОС от аппаратной платформы.

Операционная системы Windows. Операционные системы Windows — это семейство операционных систем, включающее: Windows 9x, Windows NT, Windows 2000, Windows ME, Windows XP. Windows 95 и Windows 98 — это популярные ОС для персонального компьютера с графическим пользовательским интерфейсом. Они относятся к поколению 32-разрядных операционных систем, позволяют более полно использовать потенциал современного персонального компьютера, многие операции в этих версиях Windows выполняются проще и быстрее. ОС Windows 95, 98 — хорошо защищенные многозадачные ОС, обеспечивают эффективную работу в системах мультимедиа и в информационно-вычислительных сетях (в том числе и в Интернете), работу с электронной почтой. При работе с ними 9x можно использовать длинные, достаточно информативные имена файлов, можно перемещать любые объекты в любое место экрана и в любом месте экрана вызвать контекстное меню и получить контекстную помощь.

Windows NT — многопользовательская, многозадачная, многопоточная ОС, она имеет графический пользовательский интерфейс, почти аналогичный интерфейсу Windows 95,98.

Отличительными чертами этой операционной системы являются:

- встроенная сетевая поддержка, в отличие от других ОС, Windows NT
- изначально создавалась с учетом работы в вычислительной сети, поэтому в интерфейс пользователя встроены функции совместного использования сетевых файлов, устройств и объектов;
- приоритетная многозадачность, позволяющая приложениям с более высоким приоритетом вытеснять менее приоритетные приложения, что приводит, в частности, к более эффективному использованию машинного времени ввиду автоматической ликвидации зависания системы при выполнении сбойного приложения;
- присутствие достаточно мощных средств защиты файлов различных пользователей от несанкционированного доступа;
- наличие многоуровневого доступа к ресурсам с назначением пользователям уровня доступа в соответствии с их компетенцией;
- поддержка нескольких файловых систем, кроме того Windows NT имеет собственную файловую систему (NTFS);
- поддержка широкого спектра компьютерных платформ, в том числе и мультипроцессорных вычислительных систем.

В настоящее время ее версии широко применяются самыми разными организациями, банками, промышленностью и индивидуальными пользователями.

Windows 2000 — операционная система, объединяющая возможности Windows NT и Windows 95,98, с расширением многих сервисных функций, но достаточно сложная в использовании. В качестве упрощенного варианта на базе Windows 2000 создана Windows ME — версия ОС, являющаяся развитием Windows 95,98. Разработана новая версия ОС Windows — версия а Windows XP.

14.4. Аппаратные средства процессора

На данный момент процессоры представляют собой схему (микропроцессор) и являются маленькой тонкой пластиной,

квадратной по форме. На такой схеме расположены элементы, обеспечивающие функциональность самого процессора и компьютера в целом. Такая пластина защищена пластмассовым или керамическим корпусом, подсоединенная золотыми проводами с наконечниками из металла. Данная конструкция позволяет присоединить процессор к системной плате. Разработчики компьютерных составляющих знают, что архитектора процессора отражает какие-либо свойства и качества, которые присущи целому семейству процессоров (другими словами, организация процессоров или их внутренняя конструкция). Даже если процессоры имеют одинаковую архитектуру, они могут иметь различия. В первую очередь это, конечно же, различие в процессорных ядрах, которые наделяют сам процессор, какими-либо характеристиками. Отличаться процессоры могут и размерами кэш-памяти и различиями в частоте системной шины.

Кэш-память – это по сути технология, которая во всех современных процессорах является обязательной, еще ее называют быстрой памятью. Кэш технология является буфером между процессором и более медленной оперативной памятью. Буфер является хранилищем блоков данных, которые отрабатываются именно сейчас, таким образом процессору не нужно лишний раз обращаться к ОЗУ. Такое свойство очень хорошо увеличивает производительность процессора. На данный момент существует несколько уровней кэш-памяти.

L1 – кэш-память первого уровня, является самой быстрым и работает напрямую с ядром, так как размещается непосредственно в полупроводниковом кристалле процессора. Далее идет кэш второго уровня — L2, который взаимодействует с L1. Такой кэш по объему намного больше, чем L1 и расположен на материнской плате (в некоторых полседних моделях компьютеров кэш L2 также встроен в чип процессора.

Часто реализуется и кэш третьего уровня – L3. Он достаточно медленный, а по объему еще больше, чем L2, но все же быстрее, чем оперативная память. В некоторых многоядерных процессорах кэш L3 является общей памятью для ядер процессора.

Заметим, что в настоящее время соотношение скорости работы процессора и скорости работы оперативной памяти составляет

приблизительно 1:1000, то есть, если данные, требуемые процессору для дальнейших вычислений, находятся в оперативной памяти (а не в Кэше), то процессор будет вынужден их ожидать, пропуская десятки и сотни операций. Если же данные находятся в Кэше, то они могут быть переданы процессору в ритме, необходимом для его безостановочной работы.

Принцип функционирования кэш-памяти представлен на рис.14.8.

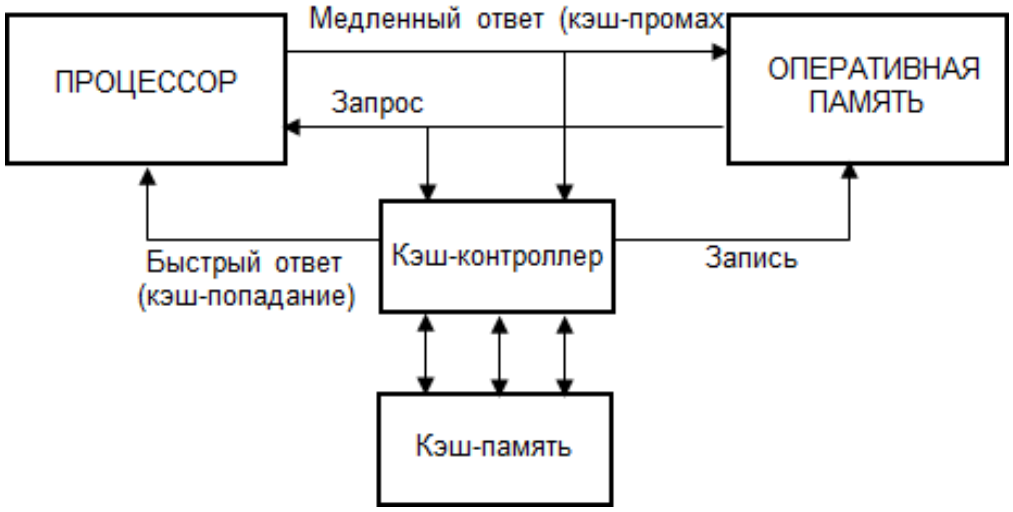


Рис. 14.8. Принцип использования кэш-памяти

Кэш-контроллер управляет кэш-памятью: загружает в неё нужные данные из оперативной памяти и возвращает, когда нужно, обработанные процессором данные в оперативную память. Когда процессор хочет прочесть (или записать) данные по какому-либо адресу оперативной памяти, он передает этот адрес в контроллер кэш-памяти.

Контроллер по некоторому алгоритму определяет, содержатся ли в кэш-памяти данные, соответствующие полученному от процессора адресу. Если данные найдены (это событие называется попаданием в кэш), кэш-контроллер выдает требуемые данные процессору (в случае чтения), либо перезаписывает их полученными от процессора данными (в случае записи).

Если же данные не найдены (промах кэша), то производится обращение к оперативной памяти, и процессор вынужден ждать. Важно понимать, что кэш всегда «полон», так как оставлять часть кэш-памяти «пустой» было бы совершенно нерационально. Поэтому

перед запуском приложения кэш заранее заполняется, а новые данные попадают в кэш только путём вытеснения (замещения) каких-либо старых данных.

Эффективность кэш-памяти зависит от следующих факторов:

1) **Объём**. Чем больше объем кша, тем большую часть требуемых программе данных он может в себе содержать, тем реже будут происходить обращения к оперативной памяти.

2) **Алгоритм функционирования**. Зачастую объема Кэш-памяти недостаточно для того, чтобы вместить все необходимые для вычислений данные. В этом случае кэш-контроллер должен «решить», какие именно данные следует держать в кэше. Поэтому, кроме объема кэша, важным является алгоритм его функционирования: кэш, оснащенный хорошим алгоритмом, будет гораздо эффективнее использовать свой объем, храня меньше ненужных данных.

3) Выполняемая процессором **программа.** Кэш оказывается эффективным потому, что большинство компьютерных программ обращаются к памяти не случайным образом, а закономерно. Чем лучше кэш-контроллер может «предсказать» обращения приложения к памяти, тем выше эффективность.

Основная (оперативная) память ОП – единственная крупная часть памяти, к которой процессор имеет непосредственный доступ. Как известно, содержимое основной памяти не сохраняется после перезагрузки или после выключения компьютера.

Развитие техники привело к тому, что в устройства, требующие высокой скорости обработки данных, стали встраивать собственную память, этим устраняются издержки на передачу данных и обычно в таких случаях используется более скоростная память, чем применяемая в ОЗУ. Примером может служить видеоадаптер, встроенный кэш центрального процессора и так далее.

Даже многие винчестеры имеют сейчас свой внутренний высокоскоростной буфер, позволяющий ускорить операции чтения/запись. Ответ на вопрос, почему эта высокоскоростная память не используется сейчас в качестве оперативной очень простой, некоторые технические сложности, но главное ее дороговизна. Применительно к типичным компьютерам, оперативная память

выпускается в виде модулей, устанавливаемых в специальный разъем материнской платы. Размеры и форма зависят от применяемого стандарта, но в общем случае выглядит в виде планки с установленными на ней чипами. Размеры и форма зависят от применяемого стандарта, но в общем случае выглядит примерно как на рисунке 14.9.

Рис.14.9. Модуль ОП, устанавливаемый на материнской плате

Оперативная память устанавливается в специально предназначенные слоты. Количество слотов колеблется от 1 до 32. Чаще всего встречаются платы с двумя или четырьмя слотами под оперативную память. Современные планки памяти бывают 2-х видов: DDR3 и DDR4. Последняя имеет меньшее энергопотребление и большую скорость передачи данных (частоту).

Если слотов 4 и больше, то слоты работают попарно. Так же они попарно размечены на материнской плате. Пары маркированы разными цветами. Для увеличения производительности следует покупать парные планки памяти и попарно подключать их в разъемы.

На данный момент времени, существует два типа памяти возможных к применению в качестве оперативной памяти в компьютере. Оба представляют собой память на основе полупроводников с произвольным доступом. Другими словами, память позволяющая получить доступ к любому своему элементу (ячейке) по её адресу.

SRAM (Static random access memory) — изготавливается на основе полупроводниковых триггеров и имеет очень высокую скорость работы. Основных недостатков два: высокая стоимость и занимает много места. Сейчас используется в основном для кэша небольшой емкости в микропроцессорах или в специализированных устройствах, где данные недостатки не критичны.

DRAM (Dynamic random access memory) — память, наиболее широко используемая в качестве оперативной в компьютерах. Построена на основе конденсаторов, имеет высокую плотность записи и относительно низкую стоимость. Недостатки вытекают из особенностей её конструкции, а именно, применение конденсаторов небольшой емкости приводит к быстрому саморазряду последних, поэтому их заряд приходится периодически пополнять. Этот процесс называют регенерацией памяти, отсюда возникло и название динамическая память. Регенерация заметно тормозит скорость ее работы, поэтому применяют различные интеллектуальные схемы стремящиеся уменьшить временные задержки.

Развитие технологий идет быстрыми темпами и совершенствование памяти не исключение. Компьютерная оперативная память, применяемая в настоящее время, берет свое начало с разработки памяти DDR SDRAM. В ней была удвоена скорость работы по сравнению с предыдущими разработками за счет выполнения двух операций за один такт (по фронту и по срезу сигнала), отсюда и название DDR (Double Data Rate). Поэтому эффективная частота передачи данных равна удвоенной тактовой частоте.

Сейчас наиболее распространённой памятью является третье поколение DDR3 SDRAM. За счет технологических решений и снижения питающего напряжения удалось снизить энергопотребление и поднять эффективную частоту, составляющую от 800 до 2400 МГц. Несмотря на тот же корпус и 240 контактов, модули памяти DDR2 и DDR3 электрически не совместимы между собой. Для защиты от случайной установки ключ (выемка в плате) находится в другом месте. Как можно догадаться, DDR SDRAM — это старый тип оперативной памяти, DDR4 — самый новый. На сегодняшний день самым распространенным является DDR3. Различаются эти типы памяти между собой производительностью и внешним видом.

Количество памяти, которое можно установить в компьютер зависит от материнской платы. Объем памяти ограничивается как физически количеством слотов для её установки, так и в большей мере программными ограничениями конкретной материнской платы

или установленной операционной системы компьютера. Для домашнего или офисного компьютера будет достаточно иметь память ОЗУ объемом 2 Гб. Для домашнего мультимедийного можно устанавливать от 4 Гб памяти. Если у вас игровой компьютер или вы часто пользуетесь «тяжелыми» профессиональными программами можно установить от 8 и больше Гб оперативной памяти.

Память энергозависима и требует подачи постоянного питания для своей работы, зато она в разы быстрее. Когда процессору требуются данные, эти данные считываются с винчестера и загружаются в ОЗУ и все дальнейшие операции с ними происходят в ней. По завершении работы с ними, если результаты нужно сохранить, то они отправляются обратно на жесткий диск для записи на него, а из оперативной памяти они удаляются, чтобы освободить место для других данных. Если результаты сохранять не нужно, оперативная память компьютера просто очищается.

Так в сильно упрощенном виде выглядит их взаимодействие. Помимо центрального процессора информация из ОЗУ может потребоваться и другим компонентам, например, видеокарте. Естественно одновременно в памяти хранится множество данных, поскольку все программы, которые вы запускаете или открываемые вами файлы загружаются в нее. Файлы браузера, через который смотрят сайты, а так же сама интернет-страница находятся именно в оперативной памяти.

Стоит отметить, что данные с жесткого диска именно копируются в ОЗУ, поэтому пока изменения сделанные с ними не будут сохранены обратно на диске, там будет оставаться их старая версия. Именно по этой причине открыв, например Word-файл и внеся в него какие то изменения в редакторе, пользователю требуется в конце выполнить сохранение, при этом файл загружается обратно на жесткий диск и перезаписывает ранящийся там.

В каждом современном модуле памяти содержится специальная микросхема называемая SPD. Данная аббревиатура расшифровывается как Serial Presence Detect и в эту микросхему производитель записывает всю информацию о данном модуле включая объем, маркировку, производителя, серийный номер, рекомендованные задержки и некоторую другую информацию. Во

время начальной загрузки компьютера эта информация считывается BIOS из микросхемы SPD и в соответствии с указанными настройками, выставляется режим работы памяти.

Связь процессора с оперативной памятью и внешними устройствами осуществляется по **системной шине** (магистрали), предназначенной для передачи кодов данных, адресов, команд и состояний готовности устройств. Внешние устройства механически подключаются к шине через разъем, называемый **слотом**. На аппаратном уровне обмен электрическими сигналами производится через **порты.** Физический порт имеет адрес, по которому подключаемое внешнее устройство распознается.

Порты устройств представляют собой некие электронные схемы, содержащие один или несколько регистров ввода-вывода и позволяющие подключать периферийные устройства компьютера к внешним шинам микропроцессора.

Портами также называют устройства стандартного интерфейса - последовательный, параллельный и игровой порты (или интерфейсы):

- последовательный порт обменивается данными с процессором побайтно, а с внешними устройствами — побитно. К последовательному порту обычно подсоединяют медленно действующие или достаточно удалённые устройства, такие, как мышь и модем;

- параллельный порт получает и посылает данные побайтно. К параллельному порту подсоединяют более "быстрые" устройства — принтер и сканер;

- через игровой порт подсоединяется джойстик.

Клавиатура и монитор подключаются к своим специализированным портам, которые представляют собой просто разъем.

Устройство управления и прерывания реализует временную диаграмму и вырабатывает управляющие сигналы для функционирования всех компонент процессора. Определяющей информацией для устройства управления служат инструкции, генерируемые дешифратором команд. Базовым узлом устройства управления является микропрограммная память, изготавливаемая либо как отдельная микросхема, либо для этого выделяется участок

постоянной памяти. Остальные узлы устройства управления разбросаны по всей системной плате.

На рисунке 14.10 представлена в качестве примера схема размещения электронных компонентов на системной плате процессора Pentium 4, предназначенного для работы в персональных компьютерах.

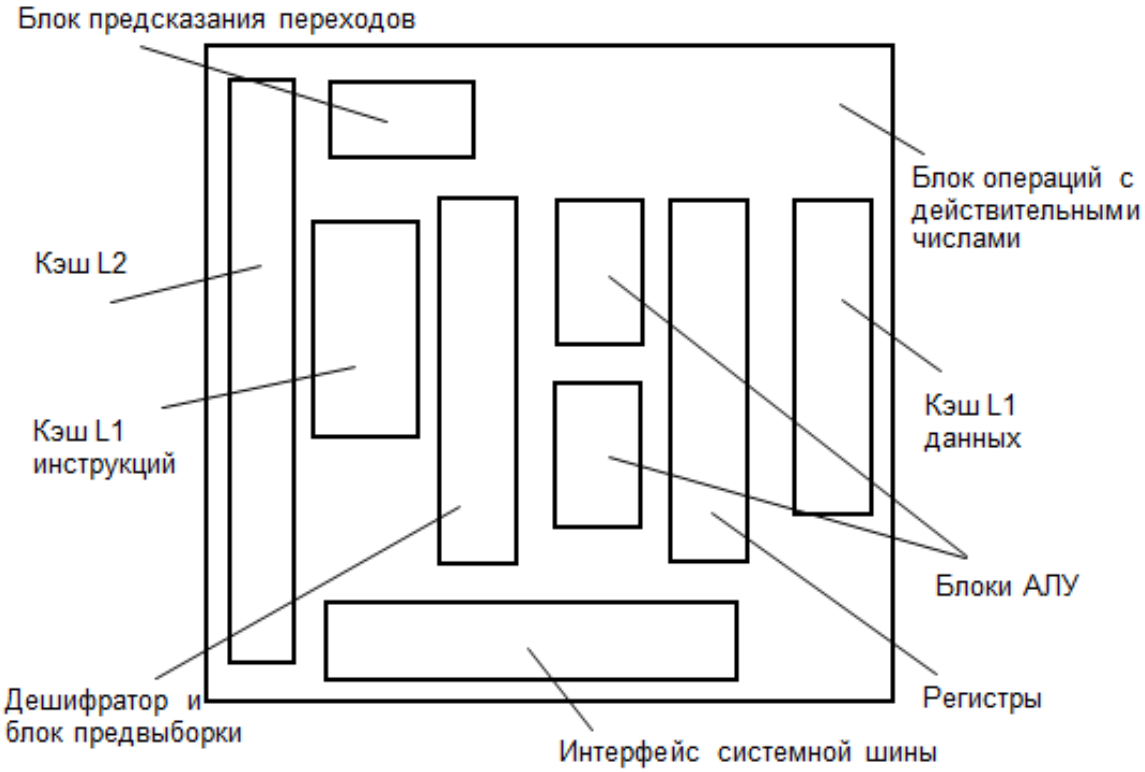


Рис.14.10. Аппаратная модель процессора Pentium 4

Отличительными особенностями данной архитектуры являются:

- наличие высокопроизводительного устройства с плавающей точкой;
- прогнозирование пути ветвления программ, предсказание переходов;
- U-конвейер с полным набором команд;
- V- конвейер с сокращенным набором команд
- раздельный кэш инструкций и данных.

В процессоре Pentium 4 реализована потоковая Hyper-Threading технология обработки, которая обеспечивает максимальную загруженность процессора при конвейерной поддержке вычислений в двадцать ступеней.

14.5. Конструкция системной платы компьютера

Основные электронные компоненты, определяющие архитектуру процессора, размещаются на основной плате компьютера, которая называется **системной (Systemboard)** или **материнской (Motherboard)**:

- центральный процессор;
- постоянная и оперативная память, кэш-память;
- интерфейсные схемы шин;
- гнёзда расширения;
- обязательные системные средства ввода-вывода.

Системные платы исполняются на основе наборов микросхем-**чипсетов** (*chipset*). Часто на системных платах устанавливают и контроллеры дисковых накопителей, видеоадаптер, контроллеры портов.

Эффективная работа вычислительной системы определяется не только параметрами процессора, но и характером связи его с остальными устройствами системы. Чипсет проектируют вместе с процессором, с целью максимального использования его возможностей. Основная функция — управление обменом данных через шины между основными устройствами компьютера. Чипсет устанавливается на материнской плате и содержит набор контроллеров, связывающих центральный процессор, память, видеоадаптеры и другие периферийные устройства (рис.14.11)

Чипсет поддерживает:

- процессоры определенной модели конкретной фирмы;
- шины FSB процессоров;
- память SDRAM;
- шины видеоадаптера;
- клавиатуру и мышь;
- последовательные и параллельные порты для подключения внешних устройств;
- порты подключения сетевых карт;
- устройства шины PCI.

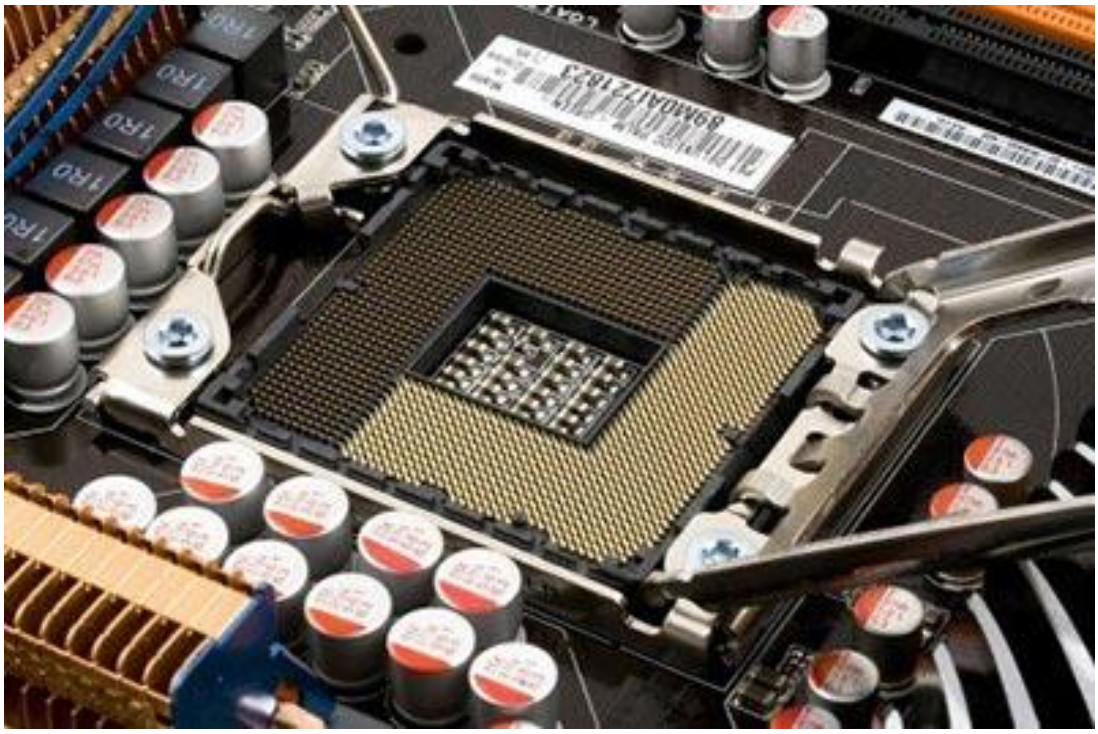


Рис.14.11. Технология размещения компонентов на чипсете

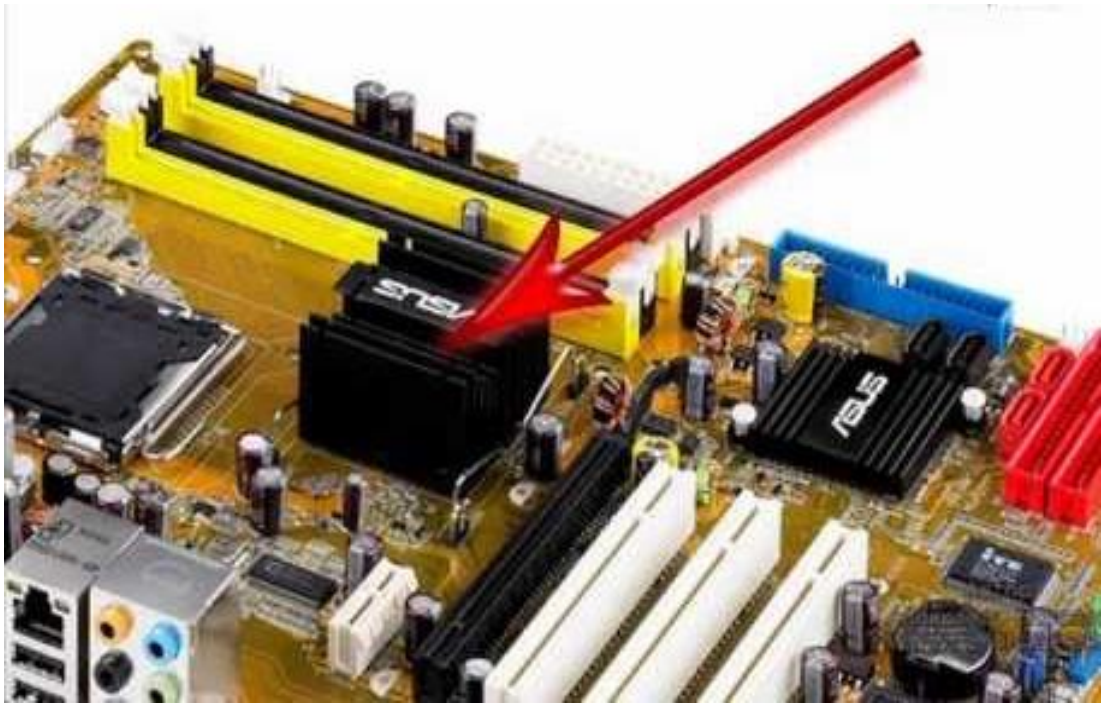


Рис.14.12. Установка чипсета на материнской плате

Современные чипсеты содержат три основные микросхемы:

- **North Bridge** - северный мост;
- **South Bridge** - южный мост;
- **Firmware Hub (FWH)** – микросхема BIOS.

Чипсет – это чип или группа чипов, которые координируют работу подключенного оборудования. Чипсет является очень важным элементом материнской платы. От него зависит максимальная скорость работы и количество разъемов на плате. Чаще всего чипсет прикрыт радиатором (рис.14.12). Они так же делятся по производителю, самые распространенные на данный момент чипсеты компании Intel это чипы 7 серии (Z77 и H77), а чипсеты от AMD представлены 900 серией (990FX, 990X, 970).Разница в чипсетах довольно сильно влияет на цену материнской платы. Также более производительные чипсеты потребляют больше электроэнергии и выделяют больше тепла, следовательно, они более требовательны к охлаждению. На более дешевых чипсетах подключаемое оборудование не сможет полностью себя раскрыть и работать с нужной производительностью.

Все микросхемы связаны между собой высокоскоростной магистралью.

Материнская плата содержит:

- процессор, установленный в специальный разъем и охлаждаемый радиатором с вентилятором;
- микросхемы кэш-памяти второго уровня, устанавливаемые на картридже центрального процессора;
- слоты для установки модулей оперативной памяти;
- слоты для установки карт расширения. Как правило, на материнских платах имеются разъемы для карт стандарта ISA, PCI, а в последних моделях и для стандарта AGP. Наличие слотов и возможность установки в них любых карт расширения (видеоадаптера, звуковой карты, модема, карты АЦП) определяет открытую архитектуру компьютера;
- микросхемы перепрограммируемой памяти, в которой хранятся программы BIOS, программы тестирования компьютера, загрузки ОС, драйверы устройств;
- разъемы для установки накопителей HDD, FDD.

Укрупненная структурная схема системной платы с чипсетом приведена на рис.14.13.

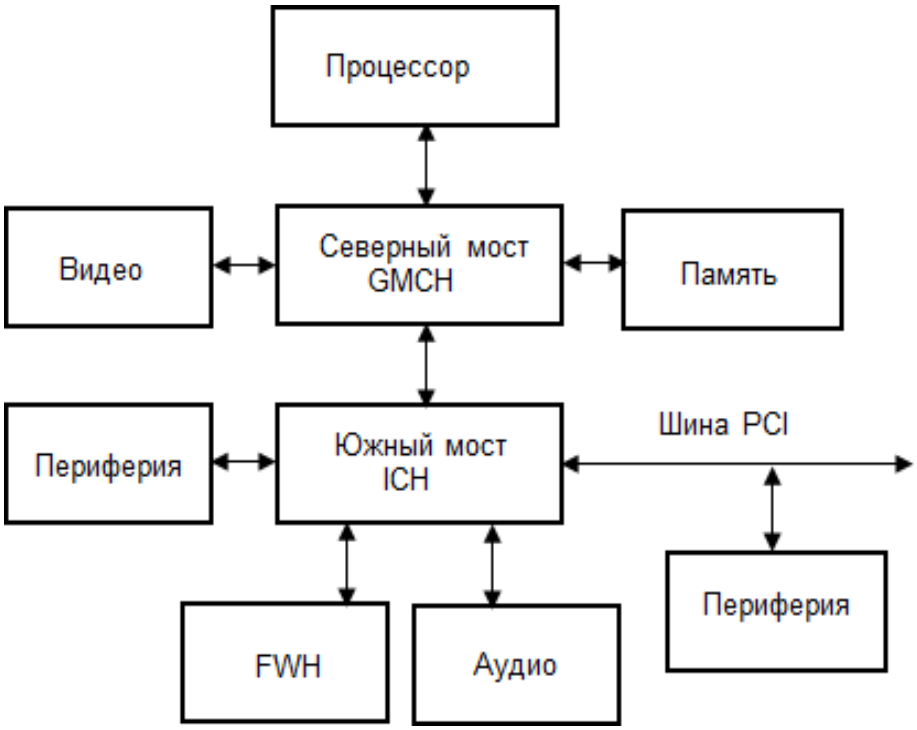


Рис.14.13. Аппаратная структура системной платы с чипсетом

Северный мост GMCH (Graphic & Memory Controller Hub) содержит контроллеры FSB, оперативной памяти и графики. Южный мост ICH (Input-Output Controller Hub) содержит контроллеры ввода–вывода периферийных устройств. Начиная с чипсета Intel 945, в состав ≪южного моста включен контроллер специализированной последовательной шины Serial Peripheral Interface (SPI) с интерфейсом FWH. Он обеспечивает перепись или распаковку BIOS в оперативную память. При обращении к BIOS в сеансе ОС программы работают с оперативной памятью, а не со схемой Flash ROM.

14.6. Форм-фактор системной платы

Форм-фактор системной платы — это стандарт, который определяет размеры платы, места ее крепления к корпусу, места и количество разъемов, подключаемый тип блока питания. Эти спецификации не являются обязательными, но большинство производителей стараются соблюдать их в угоду совместимости с другим оборудованием. На данный момент эти стандарты используются только в персональных компьютерах и не относятся к прочей компьютерной технике, такой как ноутбуки или планшеты.

Разновидностей форм-факторов существует большое количество, но мы остановимся лишь на 4-х наиболее часто

используемых: ATX, Mini-ATX, Flex-ATX и Micro-ATX (рис.14.14). Главное их отличие в размерах и разъемах PCI.

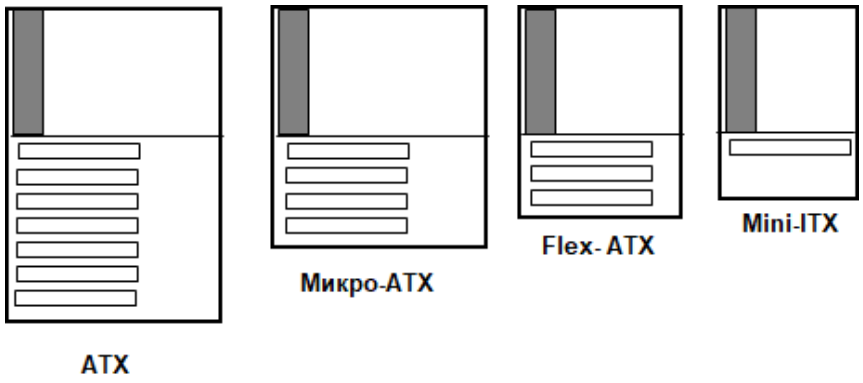


Рис.14.14. Форм-факторы системных плат

На рисунке представлены четыре системные платы следующих размеров:

- 12.00 x 9.625 см (ATX);
- 9.625 x 9.625 см (Микро-ATX);
- 9.00 x 7.50 см (Flex – ATX);
- 6.75 x 6.75 см (Mini – ITX).

Форм-фактор выбирается исходя из необходимости в разъемах для подключаемого оборудования. Так, например, для офисного компьютера будет достаточно Mini-ATX системной платы. Она будет компактней и дешевле полноразмерной. В свою очередь полноразмерная материнская плата формата ATX является предпочтительным вариантом при сборке игрового персонального компьютера или компьютера для работы с графикой. Она вмещает на себе большее количество разъемов, с помощью которых можно подключить дополнительное оборудование. Например, дополнительные планки оперативной памяти, большее количество жестких дисков, 2 видеокарты.

Необходимо помнить, что при выборе форм фактора материнской платы не стоит забывать про размеры корпуса. При сборке будет крайне неприятно вдруг обнаружить, что системная плата не влезает в корпус.

Сокетом процессора называется соединение между процессором и системной платой. Сокет является одним из основных параметров при выборе системной платы. Разъёмы сокета делятся на 2 вида, в зависимости от фирмы производителя процессора. Для

процессоров Intel специфично в названии наличие букв LGA и цифрового обозначения (LGA1155 или LGA775). Для фирмы AMD характерно одно- или двухбуквенное обозначение с цифровой приставкой в 1 или 2 цифры, возможно с символом + (AM3+ или FM2).

Немаловажной составляющей системных плат является ее система управления. Это и есть BIOS. В более новых платах UEFI. UEFI представляет из себя более продвинутую версию BIOS. Она имеет более информативный графический интерфейс и может отображать не только наборы параметров запуска, но и состояние системы в целом и элементов в отдельности, таких как температура, занятые разъемы или количество оперативной памяти.

Внутренние разъемы используются для подключения оборудования, которое остается внутри системного блока. Например оперативная память или жесткий диск. Рассмотрим подробно основные разъемы системной платы.

Слоты оперативной памяти. Оперативная память устанавливается в специально предназначенные слоты. Количество слотов колеблется от 1 до 32. Чаще всего встречаются платы с двумя или четырьмя слотами под оперативную память. Современные планки памяти бывают 2-х видов: DDR3 и DDR4. Последняя имеет меньшее энергопотребление и большую скорость передачи данных (частоту). Если слотов 4 и больше, то слоты работают попарно. Так же они попарно размечены на системной плате. Пары маркированы разными цветами. Для увеличения производительности следует покупать парные планки памяти и попарно подключать их в разъемы (рис.14.15).

Данные разъемы встречаются в 3х основных форм-факторах: PCI, PCI-Express x1 и PCI-Express x16. Количество данных слотов может варьироваться в зависимости от типа материнской платы и производителя.

Разъемы PCI-Express x16 предназначены для оборудования с высокой скоростью передачи данных. Чаще всего используется для подключения различных видеокарт.

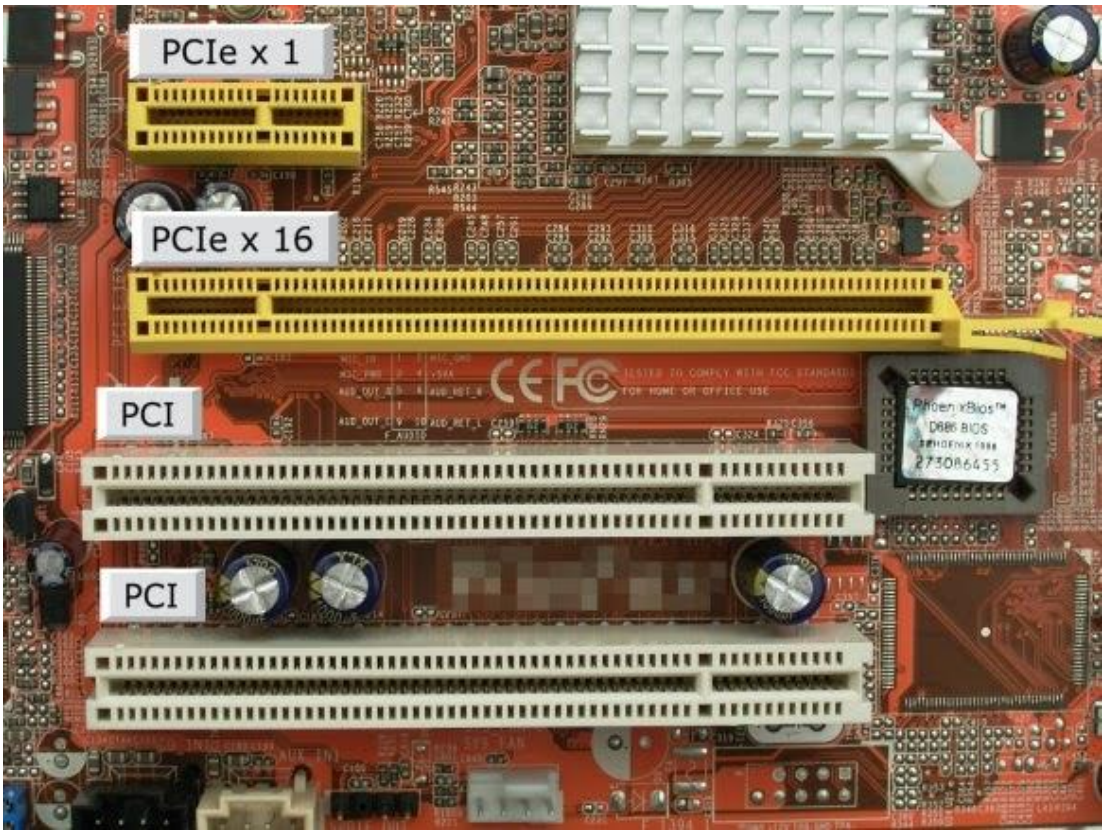


Рис.14.15. Слоты и разъемы системной платы

Разъемы PCI-Express x1 используются для подключения низкоскоростного оборудование, такого как дополнительные контроллеры USB или TV-тюнеры.

Разъем PCI является более устаревшим, чем предыдущие, но все еще использующийся в современных компьютерах. Он имеет меньшую скорость, но все еще активно используется для различных периферийных устройств, таких как сетевые или звуковые карты.

Через разъемы питания на материнскую плату подается выпрямленное напряжение (рис.14.16).

Для работы разных элементов компьютера необходимо разное напряжение, поэтому пинов в этом разъеме так много. Чаще всего встречаются 24 пиновые разъемы. Также часто присутствуют на плате дополнительные 4-х или 6-ти пиновые разъемы питания процессора.

Для подключения систем охлаждения используются небольшие 2-х или 4-х пиновые разъемы питания. 4-х пиновые разъемы имеют датчики скорости и управляются при помощи ШИМ (широтно-импульсной модуляции). Чаще всего разъемов на системной плате может быть от 1 до 4-х. Основной кулер используется для

подключения охлаждения процессора, его разъем питания подписан «cpu_fun».

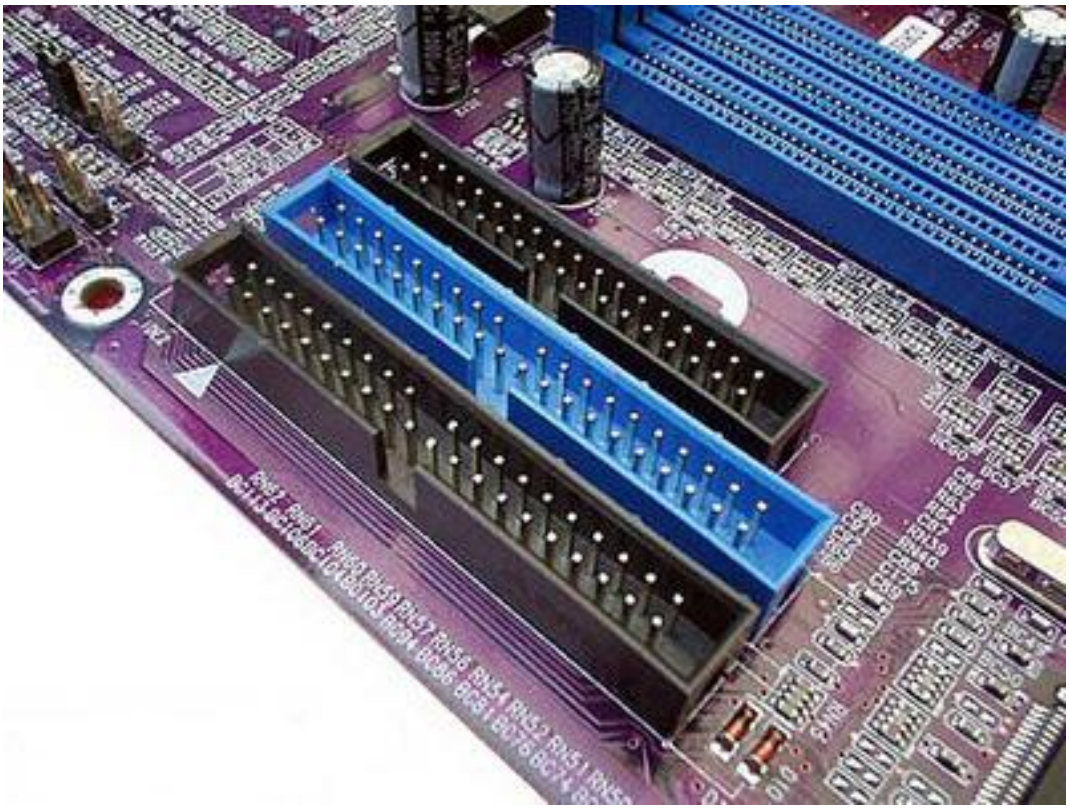


Рис.14.16. Разъемы питания системной платы

В зависимости от типа и класса системной платы, в ней могут быть дополнительные разъемы. Основная группа этих разъемов расположена в нижней части системной платы. Там находятся разъемы для подключения корпусных кнопок включения/выключения и перезагрузки компьютера, выходы на передние аудио разъемы и дополнительные USB + системы мониторинга (загрузка процессора, взаимодействие с жестким диском).

Описание подключения данных разъемов нанесено на материнскую плату рядом с ними или более подробно описано в инструкции к системной плате.

14.7. Аппаратура средств расширения функций процессоров

Звуковая карта, звуковая плата, аудиокарта (sound card) — дополнительное оборудование компьютера, позволяющее обрабатывать звук, т.е. записывать и выводит его на акустические системы. На момент появления звуковые платы представляли собой отдельные карты расширения, устанавливаемые в соответствующий

слот. В современных материнских платах представлены в виде интегрированного в материнскую плату аппаратного кодека (по спецификации Intel AC’97).

Одним из методов сокращения объёмов, занимаемых музыкой, является технология MIDI (Musical Instrument Digital Interface) — способ записи команд, посылаемых инструментам. MIDI-файл (обычно это файл с расширением mid) содержит ссылки на ноты. Когда MIDI-совместимая звуковая карта получает эту ссылку, она ищет необходимый звук в таблице (Wave Table).

Стандарт General MIDI описывает около 200 звуков. Карты, поддерживающие этот стандарт, обычно имеют память, в которой хранятся звуки, либо используют для этого память компьютера. Одной из первых wavetables-карт была GUS (Gravis Ultrasaund). Был выпущен специальный звуковой процессор EMU8K и музыкальная плата на его основе Sound Blaster AWE32, которая была, несомненно, лучшей картой того времени (32 — это количество голосов MIDI-синтезатора в карточке).

С возрастанием мощности процессоров, постепенно стала отмирать шина ISA, на которой работали все предыдущие звуковые карты, и многие производители переключились на выпуск карты для шины PCI.

В 1998 году компания Creative (рис. 14.17) вновь делает широкий шаг выпускает карту Sound Blaster Live на аудиопроцессоре EMU10K и устанавливает новый стандарт для IBM PC, который остаётся (в усовершенствованном виде) актуален и по сей день.

AC’97 (audio codec '97) — это стандарт для аудиокодеков, разработанный компанией Intel в 1997 году. Этот стандарт используется в основном в системных платах, модемах, звуковых картах и корпусах с аудиорешением передней панели. AC’97 поддерживает частоту дискретизации 96 кГц при использовании 20-разрядного стерео кодирования и 48 кГц при использовании 20-разрядного стерео для многоканальной записи и воспроизведения.

AC’97 состоит из встроенного в южный мост чипсета хост-контроллера и расположенного на плате аудиокодека. Хост-контроллер (он же цифровой контроллер, DC’97 - digit controller) отвечает за обмен цифровыми данными между системной шиной и

аналоговым кодеком. Аналоговый кодек — это небольшой чип (4×4 мм, 48 выводов), который осуществляет аналого-цифровое и цифро-аналоговое преобразования в режиме программной передачи. Состоит из узла, непосредственно выполняющего преобразования — АЦП/ЦАП. От качества применяемого АЦП/ЦАП во многом зависит качество оцифровки и декодирования цифрового звука.

Рис.14.17. Звуковая плата Creative Labs Sound Blaster Live

Спецификация HD Audio (high definition audio — звук высокой чёткости) является эволюционным продолжением спецификации AC’97, предложенным компанией Intel в 2004 г., обеспечивающим воспроизведение большего количества каналов с более высоким качеством звука, чем при использовании интегрированных аудиокодеков AC’97. Аппаратные средства, основанные на HD Audio, поддерживают 24-разрядное качество звучания (до 192 кГц в стереорежиме, до 96 кГц в многоканальном режимах — до 8 каналов). Формфактор кодеков и передачи информации между их элементами остался прежним. Изменилось только качество микросхем и подход к обработке звука.

Видеокарта. Все видеокарты можно разделить на две большие группы: интегрированные и дискретные. Интегрированные (встроенные) видеокарты, являются неотъемлемой частью материнской платы или центрального процессора, то есть встроены в

них. Наличие интегрированного видео уменьшает стоимость и энергопотребление компьютера, однако они имеют ограниченную производительность (часто не имеют собственной видеопамяти и используют ОЗУ компьютера) и используются в основном в нижнем и среднем сегментах рынка компьютерных систем (рис.14.18).

Рис.14.18. Современная видеокарта

Дискретная видеокарта, представляет собой отдельную плату расширения, устанавливаемую в специальный слот на материнской плате. Она имеет в себе все необходимое для полноценной работы. Благодаря этому, она может иметь высокую производительность, позволяющую использовать ее в 3D-играх и серьезных графических приложениях. Главными минусами является высокая стоимость и энергопотребление, что особенно важно для ноутбуков.

В свою очередь их можно разделить на два класса, профессиональные и игровые Первые в основном используются обычными людьми для игр, а профессиональные видеокарты нацелены на использование в различных сложных графических приложениях 3D-моделирования, САПР и тому подобное, где они способны дать значительный прирост производительности. Соответственно и стоимость высокопроизводительных моделей может быть высокой.

Основные технические параметры видеокарты.

Интерфейс — служит для передачи данных между графическим ускорителем и центральным процессором. В настоящее время стандартом является шина PCI Express (PCI-E) разных версий, хотя пока применяется и интерфейс AGP. Физически интерфейс

реализован в виде слота на материнской плате компьютера, куда устанавливается дискретный видеоадаптер.

Видеокарты AGP и PCI-E несовместимы друг с другом, поэтому слоты для их установки расположенные на материнской плате имеют разные физические размеры, исключающие случайную установку «чужой» видеокарты.

Тактовая частота видеопроцессора — влияет на производительность видеоадаптера, чем она выше, тем быстрее он работает и тем больше его тепловыделение. Именно поэтому, увеличение рабочей частоты GPU является одним из способов разгона видеокарты.

Частота видеопамяти — измеряется в мегагерцах, и чем она выше, тем быстрее работает подсистема памяти. Так же является одним из способов ускорить работу видеокарты.

Объем видеопамяти — сколько памяти установлено на плате и доступно для хранения данных. В настоящее время измеряется в мегабайтах или гигабайтах и чем ее больше, тем лучше. Однако есть определенный предел. Когда объем памяти, установленный в графическом процессоре, превышает объем данных требуемых для работы, дальнейшее увеличения объема видеопамяти не приводит к ускорению работы.

Тип видеопамяти — сейчас используется несколько типов оперативной памяти, применяющиеся в видеокартах. В современных видеокартах может применяться как DDR, так и специально разработанная для использования в видеокартах память типа GDDR.

Ширина шины памяти — имеет большое влияние на пропускную способность памяти и, следовательно, на общую производительность видеокарты. Определяется числом бит данных передаваемых за один цикл. Чем ширина шины памяти больше, тем выше скорость работы. В очень дешевых видеокартах ширина шины обычно 64 или 128 бит, а в топовых – 256 бит и выше.

Разъемы. Служат для подключения к видеокарте внешних устройств для вывода на них видеосигнала, таких как мониторы, телевизоры, проекторы. Иногда их наличие и количество влияет на выбор конкретной модели. Среди самых распространенных на сегодняшний день стоит отметить DVI (Digital Visual Interface),

который бывает в трех вариантах: DVI-D (цифровой), DVI-A (аналоговый), DVI-I (комбинированный).

Графические процессоры видеокарт -- GPU.

Во второй половине 1990-х годов началось быстрое развитие графических процессоров GPU — дополнительных вычислительных устройств для ускоренного исполнения алгоритмов визуализации трехмерных сцен. Поскольку трехмерная визуализация допускает эффективное распараллеливание расчетов, графические процессоры разрабатывались как поточно-параллельные системы с большим количеством вычислительных блоков, конвейерной обработкой данных и памятью с максимальной пропускной способностью (рис.14.19).

Рис.14.19. Современный графический процессор

Современные графические процессоры выполняют не только стандартные алгоритмы визуализации, но и сложные пользовательские программы, что позволяет решать на них задачи общего назначения, включая физико-математическое моделирование. При параллельных расчетах GPU может обеспечить производительность кластера из сотен обычных персональных компьютеров. По соотношению производительности и цены графические процессоры имеют большое преимущество перед

другими вычислительными системами, в том числе перед специализированными суперкомпьютерами.

Как показано в предыдущих разделах, GPU ориентирован на выполнение программ с большим объемом данных и расчетов и представляет собой массив потоковых процессоров, что состоит из кластеров текстурных процессоров, который в свою очередь состоит из набора мультипроцессоров SM (Streaming Multi-processor), в каждом из которых несколько потоковых процессоров или ядер. В современных процессорах количество ядер превышает 1024.

Сетевая карта, сетевой адаптер, Ethernet-адаптер, NIC (Network Interface Controller) — периферийное устройство, позволяющее компьютеру взаимодействовать с другими устройствами сети. В настоящее время, особенно в персональных компьютерах, сетевые платы довольно часто интегрированы в материнские платы для удобства и удешевления всего компьютера в целом.

По конструктивной реализации сетевые платы делятся на:

- внутренние — отдельные платы, вставляющиеся в ISA, PCI или PCI-E слот;
- внешние, подключающиеся через LPT или USB интерфейс, используются чаще в ноутбуках;
- встроенные в материнскую плату.

На 10-мегабитных сетевых платах для подключения к локальной сети используются 4 типа разъёмов: 8P8C для витой пары; BNC-коннектор для тонкого коаксиального кабеля;15-контактный разъём AUI трансивера для толстого коаксиального кабеля; оптический разъём (en:10BASE-FL и другие стандарты 10 Мбит Ethernet).

Эти разъёмы могут присутствовать в разных комбинациях, иногда даже все три сразу, но в любой данный момент работает только один из них. Рядом с разъёмом для витой пары устанавливают один или несколько информационных светодиодов, сообщающих о наличии подключения и передаче информации.

В зависимости от мощности и сложности сетевой карты она может реализовывать вычислительные функции (подсчёт и генерацию контрольных сумм кадров, параллельно-последовательное кодирование) аппаратно либо программно (драйвером сетевой карты

с использованием центрального процессора). Серверные сетевые карты могут поставляться с двумя и более сетевыми разъёмами. Некоторые сетевые карты, встроенные в материнскую плату, также обеспечивают функции межсетевого экрана (например, Nforce).

Сетевой адаптер перед установкой в компьютер необходимо конфигурировать. При конфигурировании адаптера обычно задаются номер прерывания IRQ, используемого адаптером, номер канала прямого доступа к памяти DMA (если адаптер поддерживает режим DMA) и базовый адрес портов ввода/вывода.

Если сетевой адаптер, аппаратура компьютера и операционная система поддерживают стандарт Plug-and-Play, то конфигурирование адаптера и его драйвера осуществляется автоматически. В противном случае нужно сначала сконфигурировать сетевой адаптер, а затем повторить параметры его конфигурации для драйвера. В общем случае, детали процедуры конфигурирования сетевого адаптера и его драйвера во многом зависят от производителя адаптера, а также от возможностей шины, для которой разработан адаптер.

Выпускаемые сегодня сетевые адаптеры можно отнести к четвертому поколению. В эти адаптеры обязательно входит схема, выполняющая функции MAC-уровня сети, скорость развита до 1 Гбит/сек, а также есть большое количество высокоуровневых функций. В набор таких функций может входить поддержка агента удаленного мониторинга RMON, схема приоритезации кадров, функции дистанционного управления компьютером. В серверных вариантах адаптеров почти обязательно наличие мощного процессора, разгружающего центральный процессор. Примером сетевого адаптера четвертого поколения может служить адаптер компании 3Com Fast EtherLink XL 10/100.

Распределение обязанностей между сетевым адаптером и его драйвером стандартами не определяется, поэтому каждый производитель решает этот вопрос самостоятельно. Обычно сетевые адаптеры делятся на адаптеры для клиентских компьютеров и адаптеры для серверов.

В адаптерах для клиентских компьютеров значительная часть работы перекладывается на драйвер, тем самым адаптер оказывается проще и дешевле. Недостатком такого подхода является высокая

степень загрузки центрального процессора компьютера рутинными работами по передаче кадров из оперативной памяти компьютера в сеть. Центральный процессор вынужден заниматься этой работой вместо выполнения прикладных задач пользователя. Поэтому адаптеры, предназначенные для серверов, обычно снабжаются собственными процессорами, которые самостоятельно выполняют большую часть работы по передаче кадров из оперативной памяти в сеть и в обратном направлении. Примером такого адаптера может служить сетевой адаптер SMC EtherPower со встроенным процессором Intel i960.

Модем (модулятор-демодулятор). Главное назначение любого модема - обеспечение физической связи двух объектов, один из которых передаёт данные другому. Модулятор трансформирует сигнал перед началом передачи в соответствии с требованиями канала связи, а демодулятор на месте приёма производит обратную операцию, предоставляя информацию пользователю в удобном для восприятия виде.

По исполнению модемы бывают как внутренние (рис.14.20), так и внешние (рис.14.21), по принципу работы - аппаратные и программные, по типу сети и соединения и по поддерживаемым протоколам передачи данных.

Рис. 14.20. Внутренний модем

В аппаратных внутренних модемах все операции по преобразованию сигнала и поддержке физических протоколов обмена производятся встроенным в модем вычислителем. Кроме этого, в

аппаратном модеме имеется постоянное запоминающее устройство, в котором записана микропрограмма, управляющая модемом.

Рис.14.21. Внешний USB-модем

Во внешних USB-модемах все операции по кодированию сигнала, контролю ошибок и управлению протоколами реализованы программно и производятся центральным процессором компьютера. В модеме располагаются только входные/выходные аналоговые цепи и преобразователи, а также контроллер интерфейса USB.

Существуют также модемы, в которых функции кодирования/декодирования разделены между модемом и процессором.

Вопросы для контроля

1. Что входит в состав программного обеспечения компьютера?
2. Назовите функции системы прерывания.
3. Перечислите основные компоненты операционной системы.
4. Рассмотрите подробно функции ОС по управлению памятью.
5. Дайте характеристику ОС Unix.
6. Дайте характеристику ОС Windows 2000?
7. Опишите функционирование кэш-памяти.
8. Каково различие в работе SRAM и DRAM?
9. Какие компоненты процессора размещаются на материнской плате?
10. Перечислите микросхемы чипсета.
11. Что такое форм-фактор системной платы?
12. Какое различие между интегрированными и дискретными видеокартами?

ГЛАВА 15. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРОВ

Средства автоматизированного проектирования и изготовления электронных компонентов дали возможность значительно удешевить аппаратную часть компьютера. В то же время значительное расширение сфер применения, увеличение числа различной категории пользователей, расширение типов услуг со стороны компьютера привело к тому, что стоимость программного обеспечения стала превосходить стоимость аппаратных средств

Программное обеспечение (ПО) компьютера – это совокупность программ, обеспечивающих эффективное функционирование компьютера и удобный интерфейс для пользователя при подготовке и решению его задач. ПО компьютера состоит из двух основных частей: **системного программного обеспечения (СПО) и прикладного программного обеспечения (ППО).**

СПО предназначено для организации процесса обработки и эффективного функционирования аппаратной части, а также для обеспечения удобного доступа к ресурсам компьютера. Это обеспечивает программными модулями операционной системы (ОС) и технической поддержки (тестирование, контроль и администрирование программных и аппаратных ресурсов).

ППО предназначено для решения проблемных задач пользователя на всех этапах исполнения: при подготовке, исполнении вычислительного алгоритма и представлении результатов.

15.1. Системное программное обеспечение

Системное программное обеспечение состоит из основного элемента – операционной системы (ОС) и вспомогательных программ, расширяющих возможности ОС и обеспечивающих пользователю удобный доступ к ресурсам компьютера. Вариантов построения ОС для различного типа вычислительных машин много, есть общие, схожие компоненты, есть и различия в структуре и функционировании различных ОС.

Для упрощения будут рассматриваться ОС персональных компьютеров.

В состав СПО персональных компьютеров входят (рис. 15.1);

а) сервисные программы, в основном утилиты, – программы, улучшающие работу аппаратной части и включающие элементы тестирования и настройки всех узлов компьютера (системной платы, аудио и видеосистемы, дисков, внешних устройств и модемов);

б) инструментальные средства, куда входят программы трансляции с языков высокого уровня и средства их отладки;

в) вспомогательные программы обслуживания компьютера.

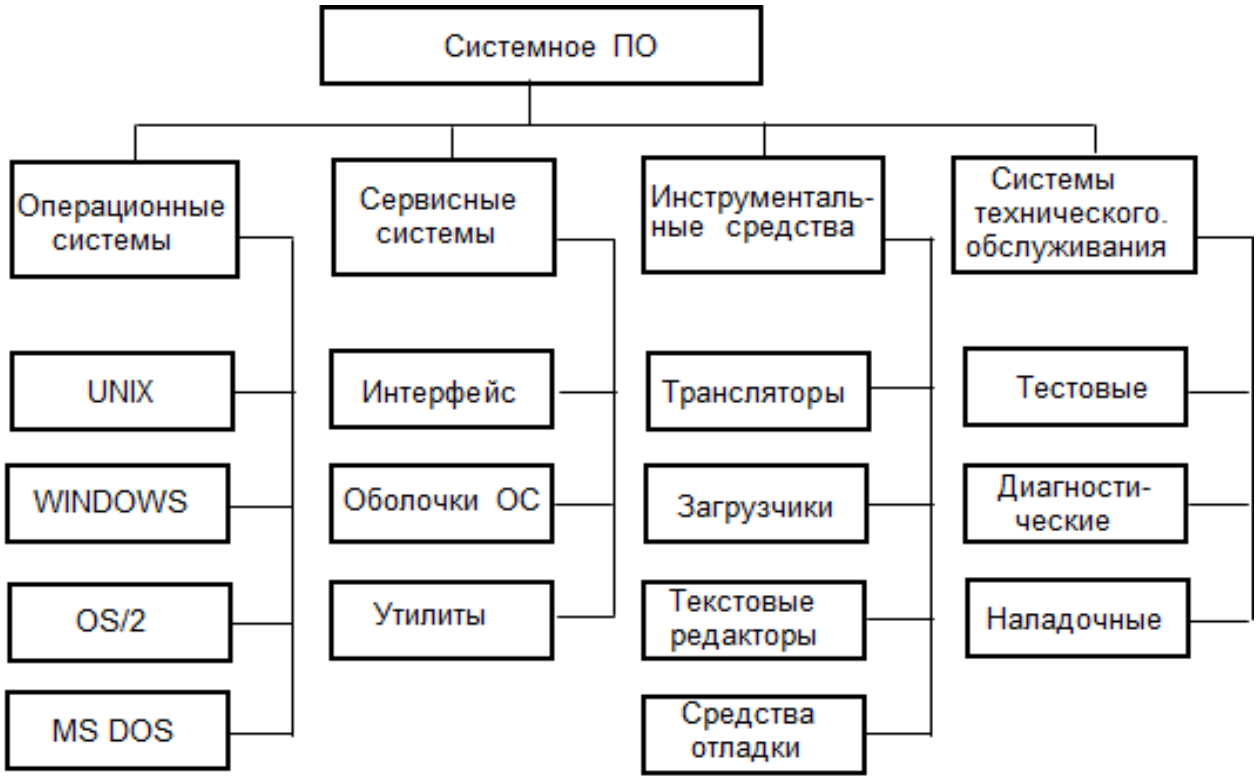


Рис.15.1. Структура системного программного обеспечения

Ядро ОС является фактически связывающим звеном между системными программами и аппаратными средствами. Основной задачей ОС является управление ресурсами компьютера, непосредственно процессом обработки и управление данными.

15.2. Разновидности операционных систем

Современные ОС можно условно разделить на группы по нескольким признакам (рис.15.2).

1. С точки зрения масштабируемости ОС могут быть однопроцессорные и мультипроцессорные, поддерживающие

выполнение заданий на нескольких процессорах. К таким ОС относятся UNIX, MS Windows NT. Среди них выделяют ОС с симметричной (SMP) и ассиметричной (ASMP) мультипроцессорной обработкой.

Классификационный признак	Разновидности операционных систем		
Поддержка мультипроцессорной обработки	однопроцессорные	SMP	ASMP
Предоставление процессорного времени	реального времени	с разделением времени	
Поддержка вычислений	локальные	сетевые	распределенные
Организация	объектно-ориентированные	операционные платформы	прикладные среды
Число обслуживаемых пользователей	однопользовательские	многопользовательские	
Число одновременно решаемых задач	однозадачные	мультизадачные	

Рис.15.2. Разновидности операционных систем

2. С точки зрения предоставления времени процессора ОС могут быть системами реального времени с гарантированным временем обслуживания каждого обращения пользователя или внешнего терминала и разделения времени (одновременный диалоговый (интерактивный) доступ нескольких пользователей с разделением между ними каждого заранее фиксированного интервала машинного времени, или в соответствии с иной дисциплиной обслуживания).

15.3. Сервисные системы

Сервисные системы предназначены для обеспечения эффективного взаимодействия пользователя и компьютера, они являются дополнением и расширением пользовательского интерфейса

операционных систем — выполняют посреднические функции между пользователем и ОС. Сервисные системы чисто условно можно разделить на:

- интерфейсные системы;
- оболочки ОС;
- утилиты.

Интерфейсные системы — это мощные сервисные системы, чаще всего графического типа, совершенствующие не только пользовательский, но и программный интерфейс ОС (сопряжение ОС с прикладными программами), в частности, реализующие некоторые дополнительные процедуры распределения дополнительных ресурсов.

Оболочки ОС предоставляют пользователю качественно новый по сравнению с реализуемым операционной системой интерфейс и делают необязательным знание последнего; оболочки реализуют наиболее дружественный интерфейс с пользователем с помощью системы меню.

Наиболее популярные оболочки ОС MS DOS: Norton Commander, Volkov Commander, DOS Navigator, FAR manager. Для OS/2 - это WPS Shell.

Утилиты автоматизируют выполнение отдельных типовых, часто выполняемых процедур, реализация которых потребовала бы от пользователя разработки специальных программ. Многие утилиты имеют развитый диалоговый интерфейс с пользователем и приближаются по уровню общения к оболочкам. Собственно,оболочки ОС и интерфейсные системы тоже состоят из утилит, но объединенных в единую систему.

Среди наиболее популярных утилит следует отметить средства:

- обслуживания магнитных дисков (форматирование; обеспечение сохранности системной информации на диске и возможности ее восстановления в случае разрушения;
- восстановление ошибочно удаленных файлов и каталогов, а также содержимого файлов и каталогов в случае его разрушения;
- оптимальная компоновка и дефрагментация файлов на диске;
- надежное удаление с диска конфиденциальной информации с невозможностью ее дальнейшего прочтения;

- обслуживания файлов и каталогов (создание, копирование, переименование, пересылка, быстрый поиск, удаление);
- архивирования и разархивирования файлов (архивирование существенно уменьшает размер файла);
- защиты от компьютерных вирусов и многие другие.

15.4. Инструментальные программные средства

Инструментальные программные средства находят применение в ходе разработки, корректировки или расширения других программ и включают в свой состав средства написания программ (текстовые редакторы), преобразования программ к виду, пригодному для выполнения на компьютере (ассемблеры, компиляторы, интерпретаторы, загрузчики и редакторы связей), контроля и отладки программ (средства отладки).

При программировании используются: машинно-ориентированный язык Assembler, процедурно-ориентированные языки высокого уровня: Macro Assembler, Basic, Pascal, Delphi, C, C++, Java, Ada, APL, COBOL, Forth, GPSS, LOGO, Modula, PL/1, Snobol, PRGT и многие другие; проблемно-ориентированные языки (функ циональные языки, непроцедурные языки высокого уровня): dBASE и его про изводные, LISP, PROLOG.

Для написания программы на одном из названных алгоритмических языков полезным помощником является текстовый редактор, позволяющий формировать тексты в символах ASCII. Текстовый редактор умеет редактировать, формировать и объединять тексты программ, а некоторые — и контролировать синтаксис создаваемых программ (примеры популярных текстовых редакторов: MS Word, Lexicon, WordPerfect, XEDIT, TeX, ChiWriter, Norton Editor, MultiEdit и многие другие).

Программа, написанная на алгоритмическом языке, должна быть преобразована (переведена) в объектную программу (объектный модуль) на языке машины (двоичные коды). Подобное преобразование выполняется трансляторами: с языка ассемблер — ассемблером, с языков высокого уровня — компиляторами. Для некоторых алгоритмических языков используются интерпретаторы, не создающие объектный модуль, а при каждом очередном

выполнении программы преобразующие каждую ее отдельную строку или оператор на машинный язык; формирующие машинные команды с последующим непосредственным выполнением предписанных этими командами действий (интерпретаторы, естественно, существенно замедляют выполнение программы, поэтому использование компиляторов для отлаженных регулярно исполняемых программ предпочтительнее).

Объектный модуль затем обрабатывается загрузчиком — редактором связей (Link, TurboLink), преобразующим его в исполняемую машинную программу, с объединением воедино отдельно скомпилированных его частей и привлечением дополнительных системных библиотек, содержащих стандартные подпрограммы и процедуры. На этапах трансляции, интерпретации и редактирования связей выполняется, как правило, синтаксический контроль программы с выдачей сообщений об ошибках.

Интерактивную отладку программы целесообразно осуществлять с помощью специальных программных средств — средств отладки. Средства отладки позволяют выполнять трассировку программ (пошаговое ее исполнение с выдачей информации о результатах исполнения — содержимом регистров и ячеек памяти), производить проверку синтаксиса программы и промежуточных результатов в точках останова, осуществлять модификацию значений переменных в этих точках. Наиболее распространенный отладчик, включаемый в системное программное обеспечение, — Debug (более развитый его вариант Turbo Debugger).

15.5. Прикладное программное обеспечение

Прикладное ПО состоит из модулей стандартных тиражируемых пакетов прикладных программ и модулей оригинальных программ самого пользователя.

К стандартным пакетам прикладных программ (ППП) можно отнести:

- текстовые, графические редакторы, издательские системы;
- табличные процессоры;
- базы данных, поисковые и экспертные системы;
- системы проектирование и дизайна;

- мультимедийные обучающие системы;
- программные пакеты математических вычислений;
- пакеты программ аудио- и видеообработки, анимации, подготовки презентации, машинной графики.

На современном рынке продаж на долью прикладных программ приходится примерно 70% от общего объема, в тоже время на системное ПО приходится примерно 30% объема продаж. Прикладные программы тиражируемого типа – эта комплекс программных средств и документов, предназначенных для реализации функционально завершенных алгоритмов обработки. Они обеспечивают автоматизацию решения задач делового документооборота, формирование отчетов и анализа статистических данных, реализуют научно-исследовательские и инжиниринговые проекты.

Проблемно-ориентированные ППП поддерживают различные информационные технологии и формируют информационную среду для реализации функциональных программ управления.

Задачи, решаемые проблемно-ориентированными ППП, сводятсяк выполнению информационных пррцедур: формирования и организации информации в виде электронных текстовых и графических документов, баз данных и знаний, выполнения аналитического преобразования информации, высокохудожественного представления информации в печатных изданиях и на презентациях.

Функционально-ориентированные ППП обеспечивают реализацию тех или иных конкретных функций управления предприятием. Названные информационные процедуры характерны скорее для офиса предприятия, в то время как функциональные задачи важны для деятельности самого

предприятия, фирмы, корпорации.Поэтому проблемно-ориентированные ППП называются офисными, а функционально-ориентированные — корпоративными.

Прикладные программы для офиса.

Компьютеры в небольших офисах чаще всего используются для выполнения следующих работ:

- обработка входящей и исходящей информации с помощью текстовых редакторов и средств презентационной графики (электронная почта и факсы, письма и запросы, реклама и прочая документация);

- сбора и анализа данных, расчетов и отчетов, выполняемые обычно с использованием электронных таблиц (расчеты и обработка прайс-листов, формирование отчетов по разным направлениям и критериям, анализ и статистическая обработка информации);

- накопление и хранение поступившей информации, обеспечивающие быстрый ее поиск (по различным критериям и признакам) и доступ к ней с применением систем управления базами данных (СУБД).

Выполняются в офисах и экономические, бухгалтерские расчеты, решаются задачи анализа финансового состояния фирм, но по предложенной выше классификации их отнесем к корпоративным задачам.

Программных продуктов, раздельно или интегрированно позволяющих выполнять указанные работы, выпускается великое множество. Выбор конкретных программ для практического использования зависит, конечно, от конкретных условий, но в значительной мере и от знаний и опыта покупателя.

Для реализации первых трех указанных выше задач целесообразно воспользоваться не отдельными программами, а интегрированными пакетами офисного обслуживания. Наблюдается тенденция не просто объединения больших автономных программ в такие пакеты, а их интеграция в прикладные программные комплексы.

Это подразумевает их полную унификацию, то есть единообразные подходы к решению таких типовых задач, как управление файлами, редактирование, форматирование, печать, вычислительная обработка, статистический и финансовый анализ, работа с электронной почтой и факсами и многое другое.

Интегрированные офисные пакеты предлагаются разными фирмами под разными названиями: офисные прикладные программы, офисные пакеты. Сегодня на рынке прикладных офисных программных продуктов lоминируют пакеты **Microsoft Office.**

Широко применяемый пакет Microsoft Office XP поставляется в трех основных вариантах.

1. Стандартный выпуск включает программы:

- Microsoft Word – многофункциональный текстовый редактор;
- Microsoft Exel – программа для создания и обработки электронных таблиц, графиков и рисунков;
- Microsoft PowerPoint – программа подготовки презентаций, включающих графические, текстовые, звуковые и видеоэлементы;
- Microsoft Outlook – мощный офисный менеджер, сочетающий в себе программы электронной почты, подготовки и отправки факсов, записную книжку и многое другое.

2. Профессиональный выпуск включает программы:

- Microsoft Access – программу для создания и редактирования баз данных;
- Microsoft Publisher – программу верстки и дизайна текстовых публикаций;
- Microsoft FrontPage – программу для создания и дизайна страниц Интернета.

Пакет Open Office.

Бесплатный пакет офисных программ. В него входят:

- текстовый редактор (Writer) , электронные таблицы (Calc), графический редактор (Image), система презентаций (Impress) и система работы с векторной графикой (Draw).

Научные и инженерные программы. Их основные функции заключаются в помощи при подготовке научных проектов, проектировании изделий, расчете различных конструкций, при решении сложных математических задач.

Математический пакет MatLab.

Пакет выполняет множество компьютерных задач для поддержки научных и инженерных работ, начиная от сбора и анализа данных до разработки приложений.

Пакет содержит инструменты для:

- сбора, анализа, обработки и визуализации данных;
- цифровой обработки сигналов и изображений;
- разработки алгоритмов и программ реализации численных методов;

- моделирования и имитации процессов;
- проектирования и разработки приложений.

Среда MatLab объединяет математические вычисления, визуализацию и мощный технический язык. Встроенные интерфейсы позволяют получить быстрый доступ и извлекать данные из внешних устройств, файлов, внешних баз данных и программ. Пакет позволяет интегрировать внешние процедуры, написанные на языках C, C++, Java с разработанными приложениями. MatLab фактически стал принятым во всем мире стандартом для технических вычислений.

Пакет автоматизированного проектирования AutoCAD.

С помощью этой программы можно создавать трехмерные объемные модели отдельных деталей и сложных систем. Новейшие технологии, заложенные в этой системе, позволяют применять ее в качестве базового пакета для создания машиностроительных, архитектурных, строительных, геодезических программ, систем инженерного анализа. Функциональные возможность программы: открытая архитектура, широкие возможности по программированию, связь с базами данных, большой выбор совместимых периферийных графических устройств, наличие сотен постоянно развивающихся прикладных программ на его базе.

Графическая программа Adobe Photoshop.

Пакет создан для профессионалов, которым необходимы совершенные инструменты настройки растрового изображения. В данной графической программе имеются следующие возможности:

- возможность создания многослойного изображения, при этом каждый элемент может быть сохранен в собственном, отдельном слое и редактироваться отдельно;
- можно добавлять текстовые вставки в любой участок изображения, набирая текст и редактировать его прямо поверх картинки;
- наличие множества инструментов для рисования, вырезания контуров изображения, редактирования отдельных участков изображения;
- широкие возможности совмещения изображений, работы с десятками популярных графических форматов;

- наличие профессиональных инструментов цветокоррекции, возможность подгонки цветовой гаммы к любым видам мониторов и печатающих устройств.

Кроме вышеуказанных прикладных программ в состав программного обеспечения могут быть включены:

- программы языкового перевода и словари;
- средства обработки звука и речи;
- программы создания и использования баз данных;
- прикладные программы обработки финансовых данных;
- программы сбора и анализа статистических данных;
- средства работы в среде Интернет.

Вопросы для контроля

1. Какие основные элементы входят в состав системного ПО?
2. Перечислите квалификационные признаки ОС?
3. Перечислите наиболее популярные утилиты.
4. Каковы функции инструментальных программных средств?
5. Перечислите решаемые задачи пакетов прикладных программ?
6. Какие инструменты содержит пакет Matlab?

ЗАКЛЮЧЕНИЕ

Вычислительная техника развивается очень быстрыми темпами. Появляются новые процессоры, виды памяти, средства сопряжения, рождаются новые архитектурные решения. В этой ситуации особенное значение имеет знание основных принципов вычислений и их аппаратно-программного ускорения. Хотя процессоры и продолжают оставаться основным вычислительным и управляющим звеном компьютера, но в их технологическом совершенствовании конкурентная борьба между фирмами- разработчиками продолжается с прежней силой. В настоящее время идет, в основном, развитие технологических методов, позволяющих воплотить то или иное архитектурное решение.

Технологический переход от повышения тактовой частоты к многоядерным архитектурам породил задачу разработки приложений, эффективно использующих возрастающее число вычислительных ядер. Это в свою очередь стимулирует процесс совершенствования численных методов обработки данных в системах с общей памятью.

Развитие в последние годы архитектуры графических процессоров ведет к созданию гетерогенных вычислительных структур с разнообразием типов процессорных элементов и иерархией видов памяти. В такой постановке графические процессора превращаются в полноправные сопроцессоры основного процессора, задачи которых выходят за рамки традиционных вычислительных алгоритмов.

С другой стороны, развитие систем искусственного интеллекта повышают требования к алгоритмическим возможностям современных процессоров. Возникает необходимость разработки и совершенствования встроенных функций логико-эвристической обработки при работе с базами знаний, с системами анализа динамических изображений и сцен, при обработке речи и карт земной поверхности. Сюда можно отнести задачи управления роботами, доказательство теорем, машинный перевод, моделирование сложных, быстропротекающих процессов.

Однако, без знания базовых основ вычислительной техники, без освоения архитектур аппаратно-программных средств трудно

осваивать новые направления развития компьютерной техники и технологий.

Автор надеется, что предлагаемое учебное пособие поможет студентам быть готовыми к освоению новых знаний в области компьютерных систем.

Список сокращений

3D – трехмерный
AGP – параллельная шина видеоадаптера
ATA – интерфейс для подключения винчестеров
BHT – буфер целевых адресов переходов
BIOS – базовая система ввода-вывода
BSB – системная шина “back-side bus”
BTB – буфер адресов переходов
BTIC – кэш-память команд в точке перехода
CC-NUMA – архитектура с кэш-когерентной памятью
CD/DVD – накопитель на оптических дисках
CISC – универсальная система команд
COMA – архитектура, содержащая только кэш-памяти
CPU – центральный процессор
DM - распределенная память
DVI – цифровой интерфейс подключения к монитору
EISA – расширенный интерфейс ISA
FSB – системная шина “front-side bus”
GPU – графический процессор
GTL – Gunning Transceiver Logic - параллельная процессорная шина
HDD – накопитель на магнитных дисках
IA-32 – архитектура 32-х разрядных процессоров Intel
IA-64 – архитектура 64-х разрядных процессоров Intel
IDE – Integrated Device Electronic – параллельный интерфейс для винчестеров (другое название - ATA)
ISA – параллельный интерфейс для периферийных устройств
LCD – жидкокристаллический дисплей или экран
LPC –недорогой интерфейс для подключения внешних устройств и связи с BIOS
LSD - Loop Stream Detector –технология раннего обнаружения циклов
MIMD - Множественный поток команд - Множественный поток данных

MISD - Множественный поток команд - Одиночный поток данных

MMX –мультимедийные команды

MPP - системы с массовым параллелизмом

NUMA - системы с неоднородным доступом к памяти

PCI –параллельная шина для подключения различных устройств

PVP –векторно - конвейерные процессоры

QPI –последовательная системная шина фирмы Intel

RISC –сокращенная система команд

ROB –буфер переупорядочивания команд

SaS –последовательный интерфейс SCSI

SATA- высокоскоростной последовательный интерфейс для подключения дисковых накопителей

SCSI –параллельный интерфейс подключения различных устройств компьютера

SIMD - Одиночный поток команд –Множественный поток данных

SISD –Одиночный поток команд - Одиночный поток данных

SM - Shared Memory –общая память

SMP –Symmetric MultiProcessor –симметричные мультипроцессорные системы

SMT - Simultaneous Multi-Threading –одновременная многопоточность

SPI –Serial Peripheral Interface –интерфейс для связи устройств с южным мостом чипсета

SSD –накопитель на флеш-микросхемах

TLB –буфер ассоциативной трансляции виртуальных адресов

UMA - однородный доступ к памяти

USB –универсальный последовательный интерфейс для подключения периферийных устройств

VGA –аналоговый интерфейс подключения к монитору

VLIW –система команд с очень длинным командным словом

АЛУ –арифметико-логическое устройство

БВП –буфер восстановления последовательности

ВС - вычислительная система

ДША –дешифратор адреса

ЗПЗ –запись после записи
ЗПЧ –запись после чтения
ЗУ –запоминающее устройство
ИКП –интегрированный контроллер основной памяти
КОП –код операции
ОЗУ - оперативное запоминающее устройство
ОП – основная (оперативная) память
ОС – операционная система
ПЗУ –постоянное запоминающее устройство
ПЗ – обработка чисел с плавающей запятой
ПО –программное обеспечение
ВУ –периферийное внешнее устройство
ПЭ –процессорный элемент
РОН –регистр общего назначения
РП –регистр предикатов
УУ –устройство управления
ФБ –функциональный блок
ФЗ – обработка чисел с фиксированной запятой
ЧПЗ –чтение после записи
ША –шина адреса
ШД –шина данных
ШУ – шина управления

Список литературы

1. David A. Patterson, John L. Hennessy. «Computer Organization and Desing», 2011. , Morgan Kaufmann, San Francisco, US.
2. Механов В.Б. Особенности архитектуры универсальных микропроцессоров. Учебное пособие – Пенза : изд. ПГУ, 2010. – 176 с.
3. Асмаков,С.В. Железо 2010. КомпьютерПресс / С.В. Асмаков, С.А. Пахомов. – СПб, Питер, 2010.- 416 с. - ISBN 975-5-49807-625-6.
4. Мусаев М.М. Компьютер тизимлари ва тармоклари.Ташкент: Алокачи, 2013. - 394с.
5. Довгий П. С., Поляков В. И. Прикладная архитектура базовой модели процессора Intel. Учебное пособие по дисциплине «Организация ЭВМ и систем». – СПб.: НИУ ИТМО, 2012. – 115 с.
6. М.Н.Головчинер. Введение в архитектуру ЭВМ. Курс лекций - Томск, ТГУ, 2013 – 108с.
7. Гергель В.П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем. Учебное пособие – Нижний Новгород; изд-во ННГУ им. Н.И.Лобачевского, 2010г.- 421с.
8. Орлов С.П., Ефимушкина Н.В.. Организация компьютерных систем: Учебное пособие– Самара: Самар.гос. техн. ун-т, 2011. – 203 с.
9. Шамаева О.Ю. Архитектура компьютера. Конспект лекций. - М. МЭИ, 2015г., -134с.
10. Соломенчук, В.Г. Железо ПК 2010/ В.Г. Соломенчук, П.В. Соломенчук. – СПб, БХВ-Петербург, 2010. – 448 с. - ISBN 978-5-9775-0515-4.
11. Мелехин, В.Ф. Вычислительные машины, системы и сети: учебник для вузов/В.Ф. Мелехин, Е.Г. Павловский. – М.: Издательский центр Академия, 2007. – 560 с. - ISBN 978-5-7695-4485-9.
12. Цилькер, Б.Я.,Орлов С.А. Организация ЭВМ и систем: учебник для вузов/ – СПб.: Питер, 2004. – 586 с. - ISBN 5-94723-759-8.

13. Поворознюк, А.И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств: учеб. пособие/ А.И. Поворознюк. - Ч.1. – Харьков, Торнадо, 2004. – 355 с. - ISBN 966-635-542-6.
14. Ефимушкина, Н.В. Вычислительные системы и комплексы: учеб. пособие для вузов/ Н.В. Ефимушкина, С.П. Орлов. – М.: Машиностроение-1, 2006. – 268 с. - ISBN 5-94275-281-8.
15. Корнеев,В.В. Современные микропроцессоры/ В.В. Корнеев, А.В. Киселев.- 3-е изд. – СПб.: БХВ-Петербург, 2003– 432 с. - ISBN 5-94157-385-5.
16. Сырецкий Г.А. Информатика. Фундаментальный курс. Том 1.- СПб, БХВ-Петербург, 2005.-832с.
17. Буза М.К. Архитектура компьютеров.Учебник для вузов.- Минск.Новое знание.2006,-559с.
18. Бройдо В.Л. Вычислительные системы, сети и телекоммуникации. Учебник для вузов.2-е изд.- СПб.:Питер, 2005 -703с.
19. К.Хамахер. Организация ЭВМ. 5-е изд.- СПб, Питер, БХВ- 2003. - 848с.
20. Жмакин А . П . Архитектура ЭВМ. — СПб.: БХВ-Петербург, 2006. — 320 с.
21. Таненбаум Э. Архитектура компьютера. 5-е изд.— СПб.: Питер, 2007. — 844с.
22. Amdahl, G. Validity of the single processor approach to achieving large scale computing capabilities. In AFIPS Conference Proceedings, Vol. 30, 1967, pp. 483-485, Washington, D.C.: Thompson Books. 39.
23. Gustafson, J.L. Reevaluating Amdahl’s Low/J.L.Gustafson. - CACM, 31(5), 1988, pp.532-533.
24. Andrews, G. R. Foundations of Multithreaded, Parallel, and Distributed Programming.. – Reading, MA: Addison-Wesley, 2000 (русский перевод Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. – М.: Изд. дом "Вильямс", 2003)

25. Alasir Enterprises [Электронный ресурс]. - Болотов П. Принципы работы кэш-памяти. – Режим доступа: http://alasir.com/articles/cache_principles/index_rus.html
26. Intel [Электронный ресурс]. – Процессор IntelR Core™2 Duo. Основа быстрых и эффективных ПК. - Режим доступа: http://www.intel.com/cd/products/services/emea/rus/processors/core 2duo/overview/422521.htm
27. Компьютерра [Электронный ресурс]. - Sandy Bridge: микроархитектура Intel следующего поколения.- Режим доступа:http://www.computerra.ru/ terralab/platform/559825/ .
28. Корпорация Intel – разделы сайта в России [Электронный ресурс]. - An Introduction to the Intel QuickPath Interconnect. - Режим доступа: http :// www . intel . - com / technology / quickpath / introduction . pdf.
29. 3DNews –Новости мира высоких технологий и обзоры компьютеров [Электронный ресурс]. – Современные процессоры Intel: 2-е полугодие 2009. - Режим доступа: http://www.3dnews.ru/guide/desktop_cpu_intel2009
30. 3DNews[Электронный ресурс]. - Новые архитектуры и технологии Intel, часть I: Dunnington, Nehalem. – Режим доступа:http://www.3dnews.ru/cpu/intel dunnington_nehalem/print.
31. [Электронный ресурс]. – Режим доступа:http://www.intel.com
32. [Электронный ресурс]. – Режим доступа: http://www.amd.com
33. [Электронный ресурс]. – Режим доступа: http://www.hp.com
34. PCI-SIG [Электронный ресурс]. – PCI-Express. - Режим доступа: http://www .pcisig. com/specifications/pciexpress/
35. Двухъядерные процессоры Intel и AMD: теория. Часть 1 Страница 2. Многоядерные процессоры http://www.cyberguru.ru/hardware/ processors/intel-2core-1-page2.html
36. hww.ru/wp/2017/11/-arxitekturu-processorov-intel-core-poslednix-pokolenij/.
37. https://worksdoklad.ru/view/1bAaJ6ssxoQhtmlhttp
38. Архитектура ЭВМ и систем [Электронный ресурс]. – Режим доступа: http://www.twirpx.com/file/143/19/

О Г Л А В Л Е Н И Е

М.М.МУСАЕВ

ПРОЦЕССОРЫ СОВРЕМЕННЫХ КОМПЬЮТЕРОВ

(Учебное пособие)

Ташкент – «Aloqachi» – 2020

Редактор: К. Маткурбанов
Тех. редактор: А. Тагаев
Художник: Б. Эсанов
Корректор: Ф. Тагаева
Компьютерная верстка: B.Berdimuradov

Изд. лиц. Ii №176. 11.06.2010.
Разрешено в печать: .
Формат 60x84 1/16. Гарнитура «Times New Roman».
Усл. п.л. 26,5. Изд.п.л. 26,0. Тираж 100. Заказ № 43.

Отпечатано в типографии ОК «Nihol print».
Г. Ташкент, ул. М.Ашрафий, 99/101.