**МИНИСТЕРСТВО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

# Основные архитектурные принципы ЭВМ

## Конспект лекций по курсу «Организация ЭВМ и ВС»

Санкт-Петербург

2014

# Глава 1. Архитектура и организация ЭВМ, их виды и основные элементы.

## 1.1. Понятие ЭВМ

Существует множество различных формулировок понятия ЭВМ (электронная вычислительная машина)[1] от достаточно простых и понятных до вычурных, многие из которых, однако, схожи по своей сути.

1. *Компьютер* – это прибор, способный производить вычисления и принимать решения в миллионы или даже в миллиарды раз быстрее человека. Компьютеры обрабатывают данные под управлением наборов команд, называемых компьютерными программами [13].
2. *Цифровой компьютер* – это машина, которая может решать задачи, выполняя данные ей команды. Последовательность команд, описывающих решение определенной задачи, называется программой [16].
3. *ЭВМ* – комплекс электронного оборудования, выполняющий интерпретацию программ в виде физических процессов, назначением которых является реализация математических операций над информацией, представляемой в цифровой форме [2].
4. *ЭВМ* – искусственная (инженерная) система, предназначенная для выполнения вычислений на основе алгоритмов. Принципы построения ЭВМ определяются с одной стороны назначением ЭВМ и с другой – элементной базой (набором элементов, которые используются для создания ЭВМ). Основным назначением ЭВМ является выполнение вычислений на основе алгоритмов, и поэтому свойства алгоритмов предопределяют принципы построения ЭВМ или, точнее, ее

[1] В дальнейшем изложении термины «ЭВМ» и «компьютер» используются как синонимы.

архитектуру (организацию) [4].

### *Краткие сведения о составе ЭВМ*

*Независимо от принадлежности любой ЭВМ к некоторому классу или типу, ее в первом приближении можно разделить на две части: центральную и периферийную.*

*Центральная часть образует ядро ЭВМ и включает в себя центральный процессор, основную память и, возможно, каналы (процессоры) ввода-вывода.*

*Периферийная часть предназначена для связи ядра ЭВМ с внешним миром (пользователями, объектами управления и т. п.) и представляет собой набор разнообразных периферийных устройств (ПУ).*

*Организация обмена между ядром ЭВМ и ПУ возлагается на систему ввода-вывода, которая представляет собой совокупность аппаратных (hardware) и программных (software) средств. К аппаратным средствам системы ввода-вывода в первую очередь относятся контроллеры (адаптеры) ПУ. Основным назначением контроллеров является управление ПУ и организация взаимодействия между конкретными ПУ и центральной частью ЭВМ. Основу программных средств системы ввода-вывода составляют драйверы ПУ, которые входят в состав системного программного обеспечения (ПО) ЭВМ.*

## 1.2. Понятие архитектуры и организации ЭВМ

В силу неоднозначности трактовки термина «архитектура ЭВМ», а так же многообразия различных элементов, включаемых в это понятие, необходимо привести для сравнения несколько формулировок различных авторов.

1. Сложность современных вычислительных машин закономерно привела

к понятию *архитектуры вычислительной машины*, 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 ЭВМ можно условно разделить на вопросы общей структуры, организации вычислительного процесса и общения пользователя с машиной, вопросы логической организации представления, хранения и преобразования информации и вопросы логической организации совместной работы различных устройств, а также аппаратурных и программных средств машины[8].

2. *Архитектура* – описание вычислительной системы на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд и средств пользовательского интерфейса, организации памяти и системы адресации, операций ввода-вывода и управления и т.д. Общность архитектур разных ЭВМ обеспечивает их совместимость с точки зрения пользователя. [7]
3. *Архитектура* – общий термин, обозначающий структуру компьютерной системы или некоторой ее части. Кроме того, к данному понятию относится структура системы программного обеспечения (например, операционной системы), а также комбинация аппаратного и базового программного обеспечения, поддерживающего объединение компьютеров в сеть. Под *архитектурой* компьютера понимается как его общая структура в целом, так и организация его отдельных элементов, необходимые для обеспечения его работоспособности. Таким образом, данный термин охватывает и компьютер, и кристаллы схемы и системные программы – кроме, как правило, приложений, нужных для выполнения конкретных задач [11].
4. *Архитектура ЭВМ* – абстрактное представление или определение

физической системы (микропрограммы и комплекса аппаратных средств) с точки зрения программиста, разрабатывающего программы на машинно-ориентированном языке, или разработчика компилятора.[2] Архитектура определяет принципы организации вычислительной системы и функции процессора и не отражает такие проблемы, как управление и передача данных внутри процессора, конструктивные особенности логических схем и специфика технологии их производства [5].

5. Ранее под термином «архитектура компьютера» подразумевалось описание структуры данных и регистров, необходимое для уяснения системы команд ЭВМ и интерпретации команд. Иначе говоря, этим понятием охватывались те минимальные знания, которые могли понадобиться программисту для составления программы на машинном языке. Однако распространение виртуальной памяти, а также расширение многообразия и повышение эффективности средств управления вводом-выводом явились причиной расширения блоков компьютера, знание которых становилось необходимым для эффективного составления программ. В настоящее время под *архитектурой* обычно понимается структурная организация компьютера в виде совокупности функциональных модулей и определенных связей между ними [6].

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

   - структурную схему ЭВМ;
   - средства и способы доступа к элементам структурной схемы (точнее

[2] *Компилятор* – программа, осуществляющая преобразование пользовательских программ с языков высокого уровня на машинный язык, язык системы команд.

говоря, структуры[3]) ЭВМ;

- организацию и разрядность интерфейсов ЭВМ;
- набор и доступность регистров;
- организацию и способы адресации памяти;
- способы представления и форматы данных ЭВМ;
- набор машинных команд ЭВМ;
- форматы машинных команд;
- обработку нештатных ситуаций (прерываний).

Как видно, понятие архитектуры включает в себя практически всю необходимую для программиста информацию о компьютере [12].

Исходя из вышеизложенного, сделаем обобщение понятия «архитектура ЭВМ». Под *архитектурой ЭВМ* обычно понимается ее представление и описание возможностей с точки зрения пользователя, разрабатывающего программы на машинно-ориентированном языке (ассемблере). Архитектура, как правило, отображает те аспекты структуры и принципов функционирования ЭВМ, которые являются видимыми для пользователя[4] и, следовательно, для разрабатываемых им программ.

Термины *архитектура ЭВМ* и *организация ЭВМ* во многом являются подобными, в связи с чем многие специалисты используют их как синонимы. Одним из сторонников такого подхода является Э. Таненбаум: «Архитектура связана с аспектами, которые видны программисту. Например, сведения о том, сколько памяти можно использовать при написании программы, - часть архитектуры. Аспекты разработки (например, какая технология используется при создании памяти) не являются частью архитектуры. Изучение того, как разрабатываются те части компьютерной системы, которые видны

---

[3] Ред. П. С. Довгого.

[4] В дальнейшем изложении под *пользователем* будем понимать *программиста, разрабатывающего программы на машинно-ориентированном языке (ассемблере)*.

программистам, называется изучением компьютерной архитектуры. Термины «компьютерная архитектура» и «компьютерная организация» означают в сущности одно и то же». [16] Однако, существует и другой подход, при котором эти понятия, если не противопоставляются, то, по крайней мере, различаются. Это различие состоит в том, что если понятие *архитектура ЭВМ* определяет возможности ЭВМ, то понятие *организация ЭВМ* определяет, как эти возможности реализованы в рамках конкретных моделей ЭВМ. Одним из сторонников такого подхода является В. Столлингс:

«При описании компьютерных систем принято различать их *структурную организацию* и *архитектуру*. Хотя точное определение этим понятиям дать довольно трудно, среди специалистов существует общепринятое мнение о смысле этих понятий и различий между ними.

Термин *архитектура компьютерной системы* (или компьютера) относится к тем характеристикам системы, которые доступны извне, т. е. со стороны программы, или, с другой точки зрения, оказывают непосредственное влияние на логику выполнения программы. Под термином *структурная организация компьютерной системы* подразумевается совокупность операционных блоков (устройств) и их взаимосвязей, обеспечивающая реализацию спецификаций, заданных архитектурой компьютера. В число характеристик архитектуры входят набор машинных команд, формат разрядной сетки для представления данных разных типов, механизм обращения к средствам ввода-вывода и метод адресации памяти. Характеристики структурной организации включают скрытые от программиста детали аппаратной реализации системы – управляющие сигналы, аппаратный интерфейс между компьютером и периферийным оборудованием, технологию функционирования памяти.» [15]

Понятие *организация ЭВМ* используется в двух аспектах: структурная организация и функциональная организация [4]. *Структурная организация* определяет, как устроена ЭВМ, т.е. определяет ее структуру на уровне

устройств, входящих в состав ЭВМ, и организации связей между этими устройствами (аппаратные интерфейсы). *Функциональная организация* определяет, в свою очередь, принципы функционирования ЭВМ, т.е. как в ней протекают вычислительные процессы при решении различных задач. При совместном рассмотрении обоих аспектов принято говорить о *структурно-функциональной организации ЭВМ*.

В связи с тем, что возможности ЭВМ постоянно развиваются и совершенствуются, то и понятие «архитектура ЭВМ» включает в себя все большее число аспектов, отражающих принципы построения и функционирования ЭВМ.

## 1.3. Виды архитектуры ЭВМ и их составные элементы

Одним из подходов к уровням представления архитектуры ЭВМ является ее разделение на два вида (класса):

- *программная архитектура*, которая включает в себя аспекты, видимые программистом.
- *аппаратная архитектура*, которая включает в себя аспекты, невидимые программистом (прозрачные как для программиста, так и для программ).

Одним из сторонников подобного подхода к классификации архитектуры является В. Л. Григорьев [10].

В связи с принятым во всем мире делением программистов на прикладных и системных, программную архитектуру также можно разделить на два уровня: *прикладную* и *системную*.

К основным элементам (аспектам) прикладной архитектуры ЭВМ, как правило, относятся:

1) типы, форматы и способы представления данных, аппаратно поддерживаемые в ЭВМ;

2) регистровая структура процессора;
3) адресная структура основной памяти и принципы размещения информации в ней, принципы формирования физического адреса;
4) режимы адресации;
5) структуры и форматы машинных команд;
6) система команд.

Все аспекты прикладной архитектуры естественным образом входят и в системную архитектуру.

К дополнительным аспектам системной архитектуры, как правило, относятся:

1) организация прерываний;
2) организация ввода/вывода;
3) организация виртуальной памяти (сегментная и страничная), принципы преобразования логического (виртуального) адреса в физический;
4) организация защиты памяти;
5) организация многозадачного (многопрограммного) режима работы ЭВМ, организация переключения задач (программ);
6) поддержка механизмов отладки программ на аппаратном уровне;
7) поддержка механизмов проверки (тестирования) отдельных блоков процессора на аппаратном уровне.

К основным аспектам аппаратной архитектуры, как правило, относятся:

1) структурная организация ЭВМ, включающая в себя номенклатуру устройств, входящих в состав ЭВМ, и организацию связей между устройствами на уровне аппаратных интерфейсов;
2) структурная организация процессора, включающая в себя реализацию

конвейера команд и арифметико-логического устройства и принципы построения блока микропрограммного управления;

3) организация кэш–памяти;

4) организация основной памяти на физическом уровне и, в частности, принципы построения многомодульной памяти с расслоением обращений (чередованием адресов);

5) представление аппаратного интерфейса на физическом уровне.

В соответствии с рассмотренным выше принципом разделения понятий *архитектура ЭВМ* и *организация ЭВМ*, а также с разделением архитектуры ЭВМ на программную и аппаратную, можно сопоставить понятия *аппаратная архитектура ЭВМ* и *структурная организация ЭВМ*.

Основные элементы архитектуры и структурной организации представлены на рис 1.

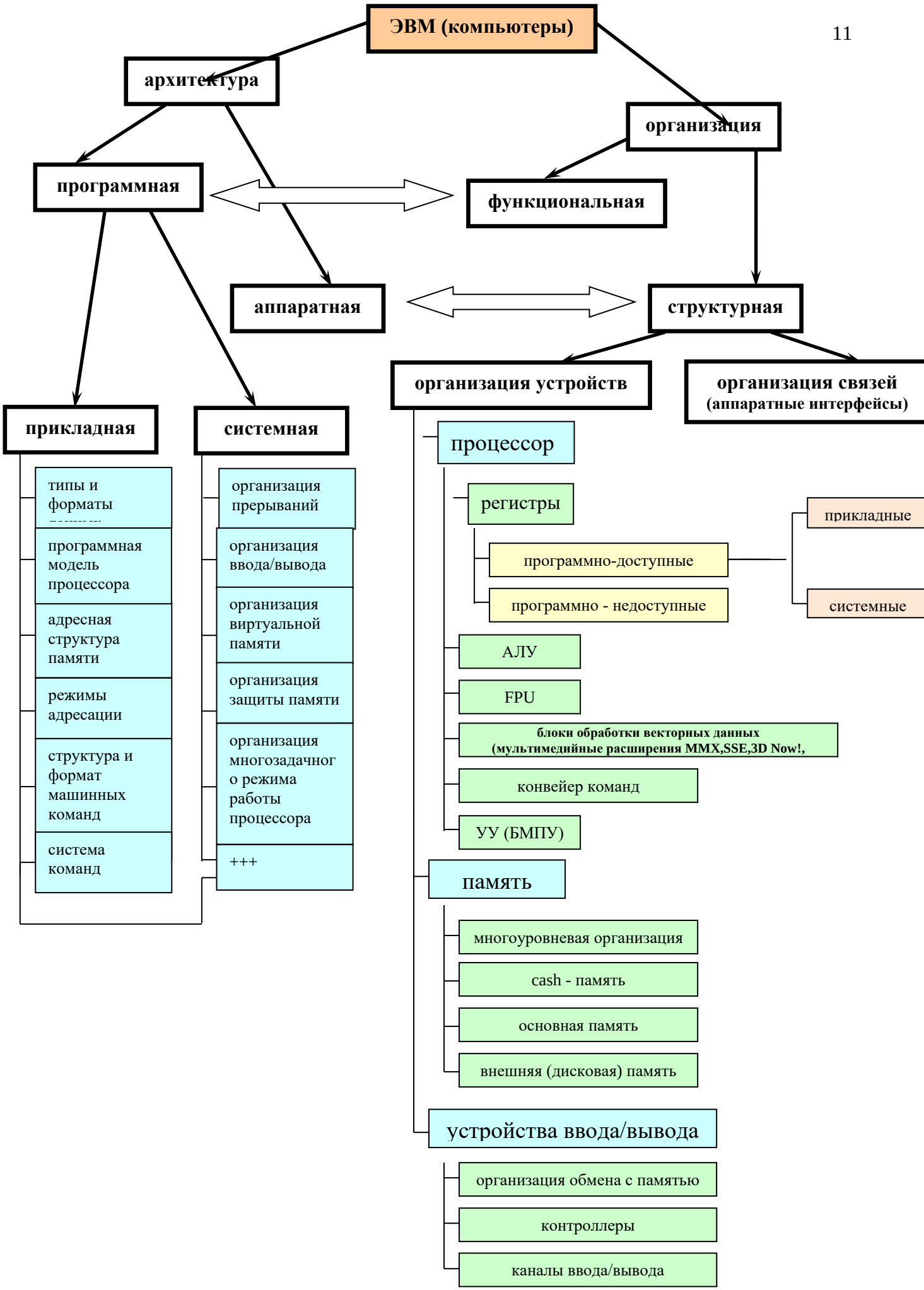



**Рис.1.Обобщенное представление архитектуры, организации ЭВМ и их элементов.**

### *1.4.1. Типы, форматы и способы представления данных, аппаратно поддерживаемые в ЭВМ*

Для многих компьютерных систем основной задачей является обработка данных различного типа, которые внутри компьютера должны быть представлены в определенных форматах.

Под *форматом данных* понимают внутреннее представление данных: разрядность и назначение битов. Например, для знаковых чисел крайний левый бит формата отводится для знака (0 соответствует знаку «+», 1 соответствует знаку «–»).

Для обозначения форматов *стандартной* длины принято использовать следующие наименования:

- *байт* (*B* – byte) – 8 бит;
- *слово* (*W* – word) – 16 бит;
- *двойное слово* (*DW* – double word) – 32 бита.

Одним из важных вопросов в отношении представления данных в ЭВМ является вопрос о том, существует ли аппаратная поддержка для конкретного типа данных. Под *аппаратной поддержкой* подразумевается наличие в системе команд ЭВМ некоторого множества машинных команд, предназначенных для обработки данных определенного типа, представленных в соответствующих форматах.

К основным типам данных, используемых в ЭВМ, относятся *числовые данные* (числа), *символьные данные* и *логические значения*.

Основными видами числовых данных являются *целые числа* – числа, представленные в формате с фиксированной точкой (Fixed Point), и *действительные числа* – числа, представленные в формате с плавающей точкой (Floating Point).

Целые числа, в свою очередь, могут быть представлены как *знаковые* и *беззнаковые*. Отличительной особенностью представления знаковых целых чисел в ЭВМ является использование дополнительного кода. Аппаратная поддержка целых чисел реализуется на уровне арифметических команд. При этом для их представления используются стандартные форматы (B, W, DW). Например, в системе команд процессоров Intel 80x86 имеются специальные парные команды умножения – MUL / IMUL и деления – DIV/IDIV, для выполнения операций над беззнаковыми (MUL, DIV) и знаковыми (IMUL, IDIV) числами.

В формате чисел с плавающей точкой выделяются: бит знака, поле мантиссы, поле порядка. Аппаратная поддержка чисел с плавающей точкой реализована на уровне арифметических команд блока FPU[5]. Числа с плавающей точкой представляются в трех различных форматах:

- короткий – 32 бита;
- длинный – 64 бита;
- расширенный – 80 бит.

Для представления символьных данных каждый символ (цифра, буква, знак и т. д.) кодируется с помощью одного байта. В настоящее время преимущественное распространение получил код ASCII[6].

### *1.4.2. Регистровая структура процессора*

Регистровая структура процессора включает в себя набор программно доступных регистров. Достаточно часто этот аспект прикладной архитектуры называется *программной моделью процессора*. Фактически рассмотрение этого аспекта связано с перечнем программно доступных регистров процессора и описанием их назначения. Программная доступность регистра означает, что со стороны программы с использованием специальных команд

[5] FPU – Floating Point Unit.
[6] ASCII – American Standard Code for Information Interchange.

может осуществляться обращение к этому регистру либо по чтению, либо по записи.

Практически любой процессор современных ЭВМ содержит внутреннюю память для хранения операндов и адресов, а также результатов выполняемых операций. Внутреннюю память обычно называют регистровой, т. к. она состоит из отдельных прямо адресуемых регистров, или сверхоперативной, т. к. время доступа к ней намного меньше, чем к оперативной.

В программной модели процессора Intel 8086 регистровая память включает в себя восемь 16-ти разрядных регистров для хранения операндов и адресов, называемых регистрами общего назначения (РОН). В старших моделях, включая Pentium, число РОН не изменилось, но увеличилась их разрядность до 32 бит.

Кроме того, в программную модель входит регистр состояний, в котором отражается текущее состояние процессора. В процессорах Intel 80x86 регистр состояний называется регистром флагов в связи с тем, что отдельные биты этого регистра называются флагами.

Одним из важнейших регистров, входящих в программную модель любого процессора, является счетчик команд (программный счетчик). В терминологии фирмы Intel этот регистр называется указателем команды (Instruction Pointer – IP). С помощью этого регистра осуществляется управление порядком следования команд при выполнении программы. При выполнении текущей команды в IP находится адрес следующей команды программы.

Общее число программно доступных регистров в процессоре Intel 8086 – 14, в последующих моделях процессора Intel 80x86 их число увеличивается практически в два раза за счет появления системных регистров.

### *1.4.3. Адресная структура основной памяти и принципы размещения информации в ней. Принципы формирования физического адреса*

Иначе этот аспект определяется в виде: «Память, как она видна программисту». Минимальная адресуемая единица в памяти – байт, в связи с этим для программы память представляется в виде массива последовательно адресуемых байтов.

При размещении байтов, составляющих некоторое слово или двойное слово, в адресном пространстве памяти во всех ПК на базе процессоров Intel 80x86 реализован следующий принцип: «Байт с меньшей значимостью размещается по меньшему адресу». В соответствии с этим принципом, адрес любой единицы информации фиксированной длины, например, слова или двойного слова, задается адресом крайнего правого (младшего) байта (рис. 1).

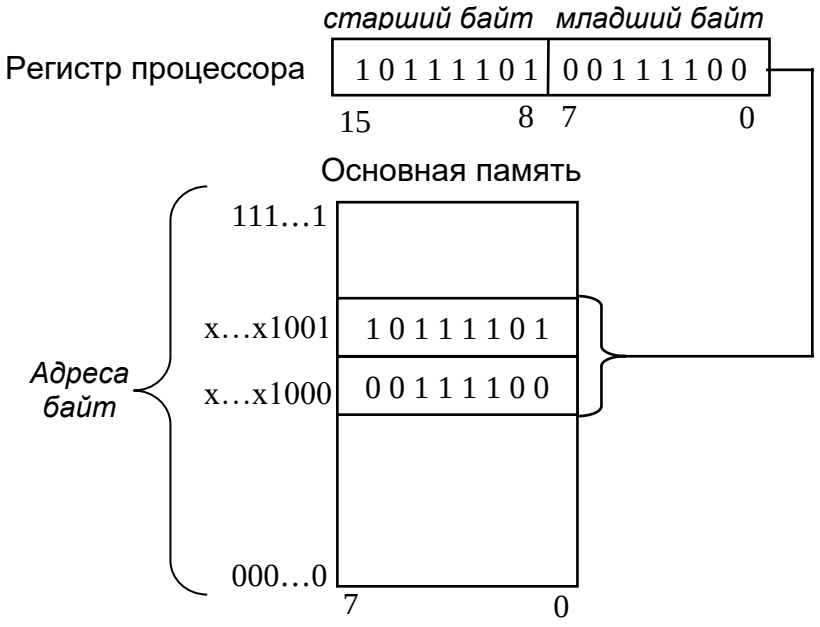



Рис.1

Под физическим адресом понимается адрес, по которому производится обращение к физической памяти. Этот адрес (физический), как правило, формируется из программного адреса на основе механизмов сегментации и, возможно, страничного преобразования. В процессоре Intel 8086

используется простейшая модель сегментированной памяти, на основе которой физический адрес формируется из двух компонент: базового адреса сегмента (выбирается из соответствующего сегментного регистра) и внутрисегментного смещения (программного адреса).

### *1.4.4. Режимы адресации*

При выполнении любой машинной команды, производящей заданное действие (сложение, вычитание, …), числа, над которыми производится операция, называются операндами и задаются своими адресами. С помощью адреса определяется местоположение операнда в памяти (основной или регистровой). Режимы адресации определяют способ формирования программного адреса операнда (результата) на основе информации, находящейся в адресной части команды.

Структура машинной команды, как правило, состоит из двух основных частей: операционной (обязательной) и адресной (необязательной).

*Операционная* часть команды определяет тип операции, инициируемой в процессоре, а точнее в АЛУ, при выполнении данной машинной командой. В связи с этим операционная часть команды обычно называется кодом операции (OpC – Operation Code). Отдельные биты или поля кода операций могут задавать длину (формат) операндов, местоположение результата операции (по первому или второму адресу) и т.п. В простейшем случае машинная команда состоит из единственного кода операции.

Разрядность кода операции определяется мощностью системы команд, реализуемой в рамках данного процессора. Так, например, в системе команд процессоров семейства Intel 80X86 код операции обычно занимает в команде один байт, хотя есть команды с двухбайтным кодом операции.

В *адресной части* команды задается информация, определяющая местоположение операндов и, возможно, результата данной операции. По

информации в адресной части команды с использованием того или иного режима адресации определяются так называемые программные адреса операндов и результата, которые в дальнейшем преобразуются в физические адреса. Для команд переходов адресная часть команды задает адрес перехода.

Простейшими режимами адресации являются прямая и косвенная. При использовании прямой адресации, в адресной части команды задается собственно адрес операнда. При использовании косвенной адресации в адресной части команды задается адрес адреса операнда, т. е. адрес ячейки памяти или регистра, в которой (котором) находится собственно адрес операнда.

### *1.4.5. Структуры и форматы машинных команд*

Структура машинной команды задает основные части команды и определяет их назначение. В свою очередь, формат машинной команды определяет разрядность как всей команды, так и ее отдельных частей (полей).

В зависимости от количества адресов, используемых в машинной команде, команды разделяются на:

- безадресные;
- одноадресные;
- двухадресные;
- трехадресные.

В зависимости от преимущественного использования некоторого числа адресов в машинных командах, системы команд и, соответственно, сами ЭВМ, их использующие, также разделяются на:

- нуль – адресные;
- одноадресные;
- двухадресные;

- трехадресные.

Систему команд процессоров Intel 80x86 принято относить к двухадресным системам команд.

В двухадресной команде для адресации результата операции используется адрес одного из операндов, называемого приемником (destination – dst). Второй операнд команды называется источником (source – src). В соответствии с этим любая двухадресная команда представляется в виде: dst = dst * src (* – знак операции).

Для процессора Intel 8086 длина машинной команды может составлять от 1 до 6 байт. Однобайтная машинная команда содержит единственный байтный код операции.

### *1.4.6. Система команд*

Основной характеристикой системы команд любой ЭВМ является ее мощность, т. е. число разнообразных машинных команд, которые можно использовать при программировании на ассемблере. Все машинные команды, образующие систему команд процессора, принято разделять на отдельные классы по их функциональному назначению. К основным классам машинных команд относятся:

- арифметические;
- логические;
- команды управления программой (переходы, циклы, вызовы, возвраты);
- команды пересылки данных и адресов.

Система команд базовой модели Intel 8086 включает в себя 113 различных мнемокодов. С учетом последующих расширений системы команд в процессорах Pentium ее мощность составляет для модели Pentium IV более 500 команд [14].

В зависимости от мощности используемой системы команд выделяют два вида процессоров:

- CISC[7];
- RISC[8].

Процессоры семейства Intel 80x86 представляют собой классическое направление компьютеров с CISC–архитектурой.

При зарождении RISC–архитектуры (начало 70-х годов) она действительно соответствовала своему наименованию (мощность системы команд составляла не более 100 различных типов команд, в то время как CISC–архитектура имела мощность системы команд до 300 команд). Современные модели RISC–процессоров имеют мощность системы команд, не уступающую CISC. В настоящее время основным признаком RISC–архитектуры является не столько сокращение набора команд, сколько стремление к выполнению подавляющего большинства команд за один машинный такт[9].

Под *машинным тактом* понимается интервал времени между двумя последовательными синхросигналами, поступающими от генератора и задающими темп работы основных схем процессора. Принято считать, что элементарные действия, выполняемые процессором за один такт, лишь в простейшем случае соответствуют одной машинной команде, и носят название *микрооперации*. К простейшим микрооперациям можно отнести пересылку между двумя регистрами, сложение (вычитание) в АУ и т.п.

В процессорах с RISC–архитектурой понятие микрооперация (микрокоманда) становится фактически тождественной понятию машинной

---

[7] Complex Instruction Set Computer – компьютер с полным набором команд. Подробнее см. в главе 7.

[8] Reduced Instruction Set Computer – компьютер с сокращенным набором команд. Подробнее см. в главе 7.

[9] Длительность машинного такта – величина, обратная тактовой частоте. Тактовая частота является одной из основных характеристик процессора и определяет его быстродействие.

операции (команды).

# Литература


1. Дж. Айлиф Принципы построения базовой машины./ Пер с англ. – М.: «Мир», 1973.
2. Майоров С. А., Новиков Г. И. Принципы организации цифровых машин. – Л .: «Машиностроение» (Ленингр. отд-ние), 1974.
3. Королев Л. Н. Структуры ЭВМ и их математическое обеспечение. – М.: Наука, Главная редакция физ.-мат. лит., 1978.
4. Майоров С. А., Новиков Г. И. Структура электронных вычислительных машин. – Л.: «Машиностроение» (Ленингр. отд-ние), 1979.
5. Майерс Г. Архитектура современных ЭВМ: В 2-х кн. Пер. с англ. – М.: Мир, 1985.
6. Компьютеры на СБИС: В 2-х кн. Пер. с японск. / Мотоока Т., Томита С., Танака Х. и др. – М.: Мир, 1988.
7. Толковый словарь по вычислительным системам / Под ред. В. Иллингуорта и др.: Пер. с англ. А. К. Белоцкого и др.; Под ред. Е. К. Масловского. – М.: Машиностроение, 1990.
8. Каган Б. М. Электронные вычислительные машины и системы: Учеб. пособие для вузов. – 3-е изд., перераб. и доп. – М.: Энергоатомиздат, 1991.
9. Амамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект: Пер. с японск. – М.: Мир, 1993.
10. Григорьев В. Л. Микропроцессор i486. Архитектура и программирование: В 4-х кн. – М.: ГРАНАЛ, 1993.
11. Microsoft Press Толковый словарь по вычислительной технике / Пер. с англ. – М.: Издательский отдел «Русская Редакция» ТОО «Channel Trading Ltd.», 1995.

12.В. Юров Assembler – СПб.: Питер, 2000.

13. Харви Дейтел, Пол Дейтел Как программировать на C++: Пер. с англ. – М.: ЗАО «Изд-во БИНОМ», 2000.

14.Гук М., Юров В. Процессоры Pentium 4, Athlon и Duron. – СПб.: Питер, 2001.

15.Столлингс Вильям Структурная организация и архитектура компьютерных систем, 5-е изд.: Пер. с англ. – М.: Изд. дом «Вильямс», 2002.

16. Таненбаум Э. Архитектура компьютера. – СПб.: Питер, 2002.

17.Von Neumann J., Brucks, Goldstine “Preliminary Discussion of the Logical Design of an Electronic Computing Instrument”, 1946.

## Основные понятия.

**Существует множество различных формулировок понятия ЭВМ от достаточно простых и понятных до чрезмерно вычурных, которые, однако, схожи по своей сути.**

**По Э.Таненбауму:**

**Цифровой компьютер – машина, которая может решать задачи, выполняя данные ей команды. Последовательность команд, описывающих решение определённой задачи, называется программой.**

**По Б. Я. Цилькеру:**

**ЭВМ – устройство, которое принимает данные, обрабатывает их в соответствии с хранимой программой, генерирует результаты и обычно состоит из блоков ввода/вывода, памяти, арифметики, логики и управления.**

**Некорректное определение:**

**ЭВМ – функциональный блок, способный выполнять реальные вычисления, включающие множественные арифметические и логические операции без участия человека в этих процессах.**

**По Новикову, Майорову (наилучший вариант):**

**ЭВМ - искусственная (инженерная), предназначенная для вычислений на основе алгоритмов.**

**Принципы построения ЭВМ, с одной стороны, определены назначением ЭВМ и, с другой стороны, элементной базой (набором элементов, которые используются для создания ЭВМ).**

**Основным назначением ЭВМ является выполнение вычислений на основе алгоритмов, и поэтому свойства алгоритмов предопределяют принципы построения ЭВМ или, точнее, ее архитектуру (организацию).**

## Понятие архитектуры и организации ЭВМ.

**В компьютерной литературе существует большое количество разнообразных трактовок понятия архитектура ЭВМ, которые отличаются как по смыслу, так и по разнообразию элементов, включаемых в понятие архитектура. (См. электронный конспект). По мнению специалистов, впервые термин архитектура компьютера (Computer Architecture) был употреблен фирмой IBM (International Business Machines) при разработке семейства машин IBM 360 в середине 60 – ых годов.**

## Обобщенное понятие архитектуры.

**Под архитектурой ЭВМ обычно понимается ее представление и описание возможностей с точки зрения пользователя, разрабатывающего программу на машинно-ориентированном языке.**

**Архитектура, как правило, отображает те аспекты структуры и принципы функционирования ЭВМ, которые являются видимыми для пользователя и, следовательно, для разрабатываемых им программ.**

**Термины архитектура ЭВМ и организация ЭВМ во многом кажутся подобными. В связи с этим, некоторые специалисты используют их как синонимы. Сторонником этого является Э.Таненбаум:**

***"Архитектура связана с аспектами, которые видны программисту. Например, сведения о том, сколько памяти можно использовать при написании программы – часть архитектуры, а аспекты разработки (например, какая технология используется при создании памяти) не является частью архитектуры. Изучение того, как разрабатываются те части компьютерной системы, которые видны программистам, называется изучением компьютерной архитектуры. Термины компьютерная архитектура и компьютерная организация означают, в сущности, одно и тоже…".***

**Однако существует и другой подход, при котором эти понятия если не противопоставляются, то, по крайней мере, отличаются. Это различие состоит в том, что если понятие архитектура ЭВМ определяет возможности ЭВМ, то понятие организация ЭВМ определяет, как эти возможности реализованы в рамках конкретных моделей ЭВМ. Одним из сторонников подобного подхода является я У. Столлингс:**

***"При описании компьютерных систем принято различать их структурную организацию и архитектуру. Хотя точное определение этим понятиям дать довольно трудно, среди специалистов существует общепринятое мнение о смысле этих понятий и различий между ними***

***Термин архитектура компьютерной системы (компьютера) относится к тем характеристикам системы, которые доступны извне, то есть со стороны программы или, с другой точки зрения, оказывает непосредственное влияние на логику выполнения программ.***

***Под термином структурная организация компьютерной системы подразумевается совокупность операционных блоков (устройств) и их взаимосвязей, обеспечивающих реализацию спецификаций, заданных архитектурой компьютера.***

***В число характеристик архитектуры входят набор машинных команд, формат разрядной сетки для представления данных разных типов, механизм обращения к средствам ввода/вывода и метод адресации памяти.***

***Характеристики структурной организации включают скрытые от программиста детали аппаратной реализации системы: управляющие сигналы, аппаратный интерфейс между компьютером и периферийным оборудованием, технологию функционирования памяти".***

**Существенное отличие между архитектурой и структурной организацией ЭВМ проявляется для моделей компьютеров, принадлежащих к одному семейству. Все они, как правило, обладают единой архитектурой, с некоторым расширением от младшим моделей к старшим, при обязательном условии совместимости, но разной структурной организацией, в результате чего старшие модели обладают большей производительностью и, соответственно, стоимостью по сравнению с младшими.**

**В принципе, понятие организация (не только применительно к ЭВМ) обычно используется в двух аспектах: структурная и функциональная.**

**Структурная организация ЭВМ определяет, как устроена ЭВМ, т. е. задает ее структуру на уровне устройств, входящих в состав ЭВМ, и организацию связей между этими устройствами на уровне аппаратных интерфейсов.**

**Функциональная организация определяет, в свою очередь, принципы функционирования ЭВМ, т. е. как в ней протекают вычислительные процессы при решении различных задач.**

**В некотором смысле существует аналогия между понятиями архитектура ЭВМ и функциональная организация. В связи с тем, что возможности ЭВМ постоянно развиваются и совершенствуются, то, и понятие архитектура ЭВМ включает в себя все большее число аспектов, отражающих принципы построения и функционирования ЭВМ.**

## Виды архитектуры ЭВМ и их составные элементы.

**Одним из подходов к уровням представления архитектуры ЭВМ является её разделение на 2 уровня (2 класса):**

**- программная архитектура, которая включает в себя аспекты, видимые программистам и, соответственно, программам**

**- аппаратная архитектура, включающая аспекты, невидимые для программиста.**

**В этом смысле понятие аппаратной архитектуры и структурной организации ЭВМ можно рассматривать как синонимы.**

**В связи с принятым во всем мире деления программистов на прикладных и системных, программную архитектуру также целесообразно разделить на 2 вида: прикладную и системную. Основные элементы архитектуры и структурной организации представлены на рис 1.**

## Краткое представление основных элементов прикладной архитектуры компьютеров.

### *Типы, форматы и способы представления данных, аппаратно поддерживаемых в ЭВМ.*

**Для многих компьютеров основной задачей является обработка данных различного типа, которые внутри компьютера должны представляться определённых форматах.**

**Под форматом данных обычно понимают внутреннее представление данных, в первую очередь, разрядность и назначение битов. Например, для знаковых целых чисел крайний левый бит формата отводится для представления знаков. Для обозначения форматов стандартной длины принято использовать следующие наименования:**

- **Байт (B – Byte) – 8 бит**
- **Слово (W – Word) – 16 бит**
- **Двойное слово (DW – Double Word) – 32 бита**
- **Учетверенное слово (QW – Quadro Word) – 64 бита**

**Ключевым понятием в отношении данных, представляемых в ЭВМ, является наличие или отсутствие аппаратной поддержки для конкретного типа и формата данных. Под аппаратной поддержкой подразумевается наличие в системе команд ЭВМ некоторого множества машинных команд, предназначенных для обработки данных определенного типа, представленных в соответствующих форматах.**

**Применительно к базовой модели Intel 8086 аппаратной поддержкой обладают следующие типы и форматы данных:**

- **Целые знаковые числа**
  **Основные форматы: B, W. Примеры команд: ADD, SUB, IMUL(сомножители ), IDIV(делитель, частное, остаток). Неосновной формат: DW. Примеры команд: IMUL(произведение),IDIV(делимое).**
- **Целые беззнаковые числа**
  **Основные форматы: B, W. Примеры команд: ADD, SUB, MUL(сомножители), DIV(делитель, частное, остаток).**

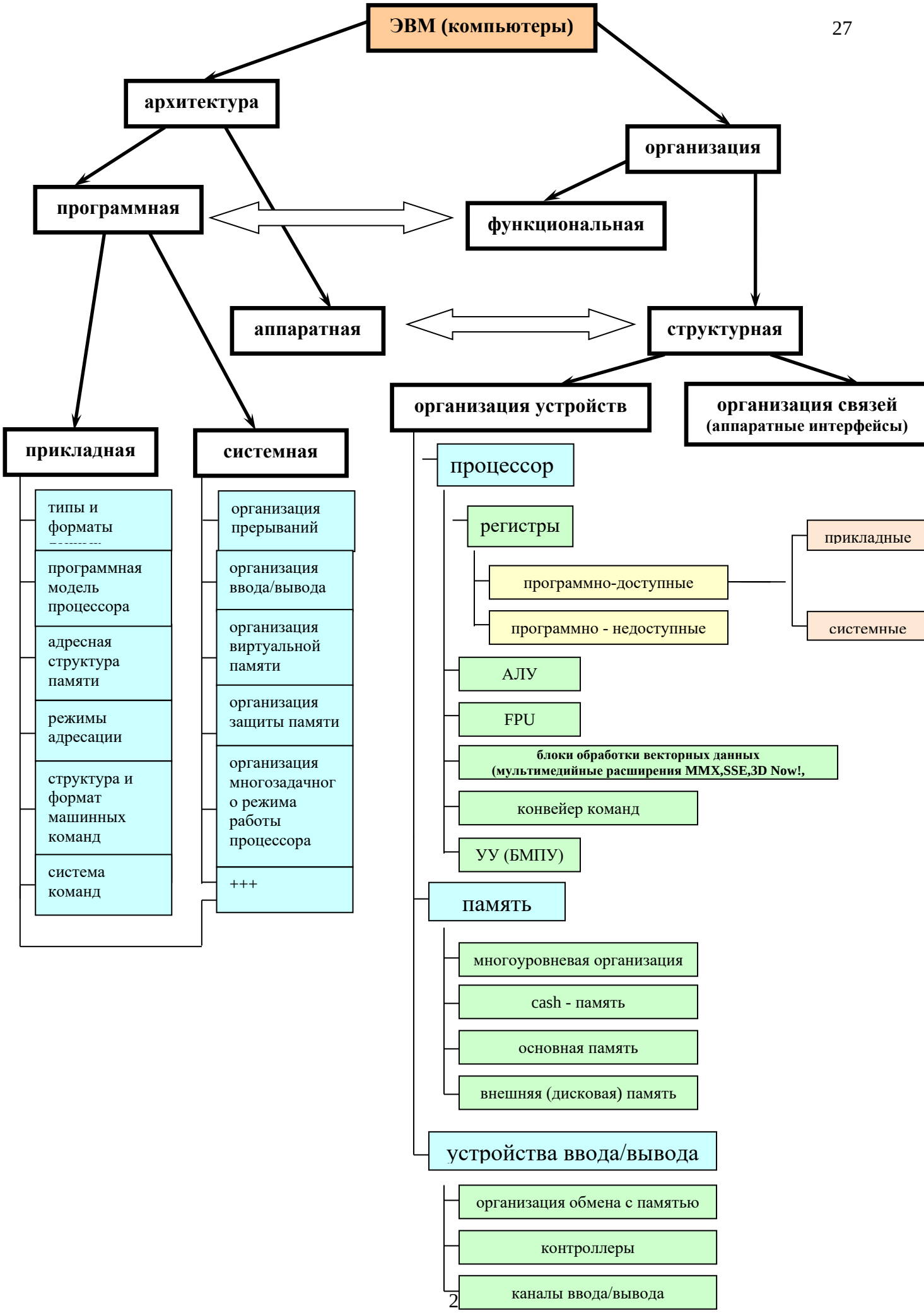



**Рис.1.Обобщенное представление архитектуры, организации ЭВМ и их элементов.**

**Неосновной формат: DW. Примеры команд: MUL(результат),DIV(делимое).**

- **Числа с плавающей запятой**
  **Короткий формат: 32 бита**
  **Длинный формат: 64 бита**
  **Расширенный формат: 80 бит**
  **Примеры команд: FADD, FSUB**
- **Десятичные числа**
  **Упакованный формат – BCD:B. Примеры команд: DAA, DAS**
  **Неупакованный формат – ASCII. Примеры команд: AAA, AAS, AAM, AAD**
- **Логические значения**
  **Основной формат:B, W. Примеры команд: AND, OR, XOR, TEST, NOT**
- **Символьные данные**
  **Основной формат:B. Примеры команд: MOVS, LODS, STOS, CMPS, SCAS**
  **XLAT.**

**Для числовых данных необходимо представлять диапазон и точность.**

## ***Программная модель (регистровая структура) процессора.***

**Регистровая структура процессора включает в себя набор программно доступных регистров. В соответствии с этим, этот аспект достаточно часто называют программной моделью процессора. Фактически, рассмотрение этого аспекта связано с перечислением программно доступных регистров и описанием их назначения для использования.**

**Программная доступность регистров означает, что со стороны программы, с использованием некоторых машинных команд, может осуществляться обращение к этому регистру, либо по чтению, либо по записи.**

**Важной характеристикой регистра является его разрядность. Как правило, именно разрядность внутренних регистров и определяет разрядность самого процессора (Intel 8086 – 16 –разрядный). Разрядность регистра определяет количество бит информации, которое можно представить (хранить) в данном регистре.**

**Любой процессор в современной ЭВМ содержит собственную внутреннюю память для хранения, в основном, операндов и адресов, а также результатов выполняемых операций. Эту внутреннюю память называют регистровой памятью, или сверхоперативной памятью, чтобы подчеркнуть значительно большее её быстродействие по сравнению с оперативной (основной) памятью. Быстродействие памяти определяется так называемым временем доступа (обращения). Время доступа к регистровой памяти – единицы наносекунд, а к оперативной памяти - десятки.**

**В состав регистровой памяти любого процессора входят как программно доступные, так и программно недоступные регистры. Типичным примером программно недоступного регистра может служить регистр команд, в который производится выборка машинной команды из памяти перед её выполнением.**
**Программно доступные регистры, в свою очередь разделяются на прикладные (доступные как прикладным, так и системным программам) и системные**

**(доступные только системным программам). Системные регистры появляются в процессорах семейства Intel 80X86, начиняя с модели i286, в которой впервые был введён защищённый режим.**

**В старших моделях процессора Intel используются следующие группы системных регистров.**

- **Управляющие регистры CR – Control Registers**
- **Регистры управления памятью**
- **Регистры отладки DR – Debug Registers**
- **Регистры проверки TR – Test Registers (аппаратная поддержка механизмов тестирования внутренних блоков)**

**Программная модель базового процессора Intel 8086 включает в себя 14 шестнадцатиразрядных регистров, 8 из них входят в состав Регистров Общего Назначения (РОН) – General Purpose Registers (GPR). Группа этих регистров предназначена как для хранения операндов (результатов), так и адресов.**

**В принципе, существует два диаметрально противоположных подхода к использованию регистров процессора:**

**1) Полная специализация регистров, когда каждый регистр используется только по одному специальному назначению.**

**2) Полная универсализация, когда каждый регистр можно использовать по любому назначению.**

**В процессорах фирмы Intel используется промежуточный подход. Это означает, что, в принципе, за каждым регистром закреплена его определенная функциональная специализация. Например, функционально специализированный регистр CX – Counter Register. Его специализация проявляется при выполнении команд циклов (LOOP), команд сдвигов (SAR) или команд обработки строк (MOVS, CMPS). Эта специализация отражается в наименовании регистра. Однако наличие специализации у регистров не мешает их использованию для других целей (не по прямому назначению). Например, в регистр CX может быть помещён операнд для какой – либо арифметической команды.**

**Использование регистров по их прямому назначению позволяет существенно сократить длину машинного (объектного) кода программы за счёт использования неявной адресации (операнд или адрес не задаётся, а подразумевается по умолчанию).**

**Следующая группа регистров программной модели – 4 Сегментных Регистра – Segment Registers (SR). С использованием этих регистров реализуется простейшая модель сегментирования памяти. В сегментных регистрах содержатся базовые (начальные) адреса 4 сегментов памяти по наименованию регистров:**

- **Code Segment (сегмент кода)**
- **Stack Segment(сегмент стека)**
- **Data Segment(сегмент данных)**
- **Extra Segment(дополнительный сегмент)**

**Модель памяти в процессоре Intel 8086 предполагает формирование физического адреса как суммы двух компонент: базовый адрес сегмента и Offset (внутрисегментное смещение). При суммировании компонент первая составляющая сдвигается влево на 4 разряда, в итоге получается двадцатиразрядная сумма, представляющая собой физический адрес, который и выставляется на внешнюю шину адреса. В процессоре Intel 8086 используется мультиплексированная шина адрес/данные, т. е. по одним и тем же проводам, но в разные моменты времени**

**передаются адрес и данные. В соответствии с принципами формирования физического адреса, в сегментных регистрах находятся старшие 16 – ти разрядные компоненты 20 – ти разрядных базовых адресов сегментов.**

**В соответствии с этим подходом, границы сегментов физической памяти выравниваются на 16 – ти байтную границу (4 младших нуля в адресе), которую принято называть границей параграфа.**

**Выбор внутрисегментного смещения (Offset) определяется видом обращения к памяти. Например, при выборке команды в качестве компонент адреса используется пара CS – IP.**

## *Регистр IP (Instruction Pointer).*

**Любой процессор, входящий в состав компьютера с неймановской архитектурой, в качестве обязательного элемента, содержит так называемый счётчик команд, который иначе называется программным счётчиком PC (Program Counter) или указатель команд.**

**Содержимое IP используется процессором при выборке очередной команды из памяти. В момент выполнения машинной команды, содержимое IP определяет адрес следующей команды.**

**Понятие"следующая", в отношении команды, характеризует последовательность команд не столько в смысле их выполнения, сколько в смысле их положения в памяти. Так, например, при выполнении команд перехода, вызовов, возвратов, содержимое IP изменяется на значение адреса перехода, вызова или возврата.**

## *Регистр FR (Flag Register).*

**Содержимое этого регистра используется, во-первых, для фиксации так называемых признаков результата (арифметические флаги), а так же для управления режимом работы процессора (флаги управления). Содержимое арифметических флагов используется при выполнении команд условных переходов. Значения арифметических флагов изменяются при выполнении большинства арифметических и логических команд, к флагам управления относятся:**

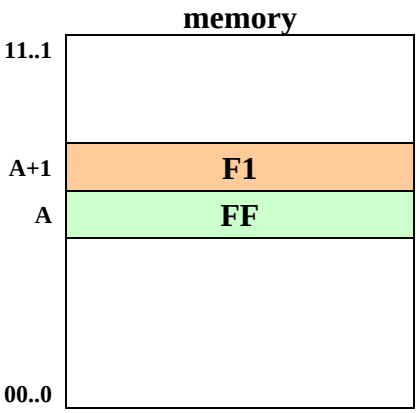
- **IF (Interrupt Flag) – Флаг Прерывания**
- **TF (Trace Flag) – Флаг Трассировки**
- **DF (Direction Flag) – Флаг Направления**

## *Адресная структура основной памяти и принципы размещения информации в ней. Принципы формирования физического адреса.*

**Этот аспект определяет память с позиции программиста, разрабатывающего программу на ассемблере (*память, как она видна программисту).***

**В подавляющем большинстве адресация памяти осуществляется на уровне байт.**

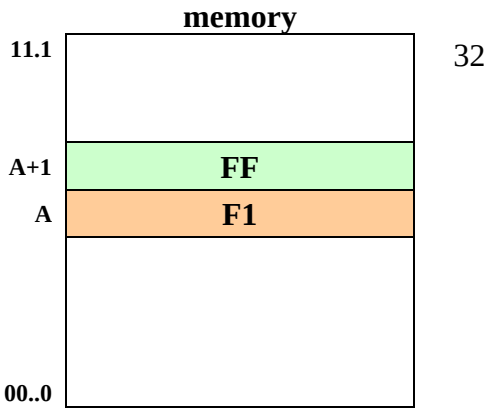
**В соответствии с этим для программы память представляется в виде массивов последовательно адресуемых байтов. При размещении единицы информации, кратных байту, например Word или Doable Word, в памяти компьютера, адрес единицы информации определяется адресом одного из байтов, либо старшего, либо младшего. В зависимости от этого, в англоязычной литературе система адресации, начиная от старшего байта, обозначают термином "Big Endian" , а с младшего байта "Little Endian".Сторонниками первого принципа является фирма IBM и Motorola, сторонниками второго - Intel и DEC. Например: -15 в формате Word**




| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | | | | F | | | | F | | | | 1 | | | |

**Big Endean –**
**байт с большей значимостью по меньшему адресу**

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | | | | F | | | | F | | | | 1 | | | |




**Little Endean –**
**байт с большей значимостью по большему адресу**

**При размещении единицы информации фиксированной длины в памяти компьютера достаточно широко используется понятие целочисленной границы. В некоторых моделях компьютеров несоблюдение целочисленной границы при размещении данных, а иногда даже и команд, приводит к прерыванию программы.**

**Принцип целочисленной границы гласит:**
**Адрес любой фиксированной единицы информации,**
**содержащий $2^k$ байт, должен быть кратен $2^k$ .**

**Это означает, что *k* младших разрядов адреса должны быть равны нулю. В процессорах семейства Intel 80X86 проверка соблюдения целочисленной границы только в отношении данных аппаратно поддерживается, начиная с модели i486.**

**При одном обращении к памяти, количество байт, перемещаемых из процессора в память или обратно, соответствует ширине (разрядности) шины данных. При этом 2 или 4 байта, участвующих в обмене являются структурой, выровненной на целочисленную границу (естественное требование аппаратного интерфейса памяти). В соответствии с этим, для уменьшения числа обращений к памяти необходимо соблюдать целочисленную границу. Реализация этого правила учтена в большинстве компиляторах.**

## *Режимы адресации.*

**При выполнении любой машинной команды, производя заданную операцию, будь это сложение, вычитание, конъюнкция, дизъюнкция и т.д., данные, над которыми производятся операции и называемые операндами, задаются своими адресами. В соответствии с этим, единственным идентификатором данных (операндов) и результатов внутри ЭВМ является их адрес. С помощью адреса определяется местоположение операндов в памяти компьютера (основной или регистровой).**

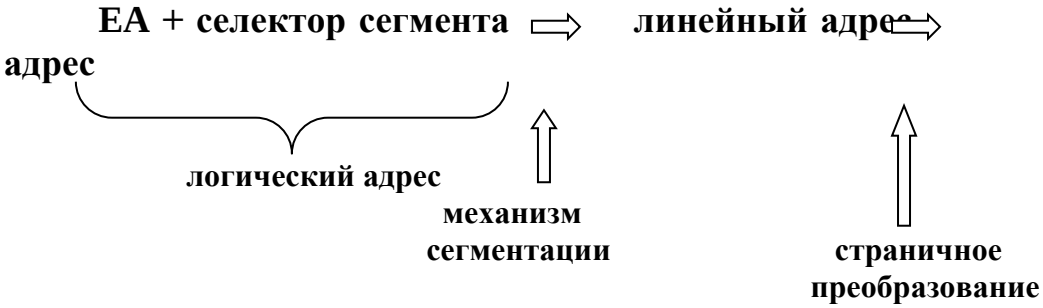
**Под режимом адресации принято понимать способ формирования так называемого исполнительного адреса операнда (или результата) на основе информации, находящейся в адресной части команды.**

**Исполнительный адрес достаточно часто называют программным адресом, т.к. он формируется программой. В терминологии фирмы Intel, исполнительный (программный) адрес называется эффективным адресом Effective Address (EA).**

**В подавляющем большинстве случаев исполнительный адрес не является физическим. Под физическим адресом принято понимать адрес, по которому производится фактическое обращение к памяти. Именно физический адрес выставляется процессором на внешнюю шину адреса.**

**Получение физического адреса на основе исполнительного сопровождается преобразованием последнего с использованием в общем случае механизмов**

**сегментации и страничного преобразования. Применительно к процессору фирмы Intel, работающему в так называемом защищенном режиме, в котором осуществляется поддержка виртуальной организации памяти, как на уровне сегментов, так и на уровне страниц, преобразование эффективного адреса в физический осуществляется по следующей схеме:**




**Структура машинной команды в общем случае состоит из двух основных частей: операционной и**


**адресной:**

**Операционная часть задаёт тип выполняемой операции и обычно называется кодом операции, Operation Code (OpC).**

**Адресная часть задаёт адреса операндов, участвующих в операции, а так же адрес результата.**

**В зависимости от числа операндов, задаваемых в адресной части команды, машинные команды разделяются на трёхадресные, двухадресные, одноадресные и безадресные (нольадресные). В процессорах семейства Intel 80X86 используются безадресные, одно – и двухадресные команды. Использование безадресных команд (однобайтные команды, содержащие только код операции) может быть связано с двумя аспектами:**

- **использование неявной адресации операндов (их адреса не задаются в команде, но подразумеваются по умолчанию)**
- **для выполнения команды операндов не требуется (NOP, HLT)**

**В терминологии фирмы Intel операнды двухадресной команды называются источником Source (SRC) и приемником Destination (DST). Результат операции помещается по адресу операнда – приемника.**

**Основными режимами адресации, используемыми в ЭВМ, принято считать:**

1. **Прямая**
   - **памяти**
   - **регистровая**
2. **Косвенная**
   - **памяти**
   - **регистровая**
3. **Относительная**
   - **базовая**
   - **индексная**
   - **базово - индексная без смещения**
   - **базово – индексная со смещением**
   - **относительно текущего счетчика команд**
4. **Непосредственная**

**5. Неявная.**

**В максимальном случае для процессора 8086 относительный адрес может состоять из трёх компонент EA = Base + Index + Displacement (Disp)**

**В старших моделях процессоров Intel реализовано дальнейшее развитие относительной адресации в виде базово – индексной адресации с масштабированием. При использовании этого режима индексная компонента EA умножается на соответствующий масштаб. $EA = Base + Index * 2^{Scale} + Disp$ , Scale = 0,1,2,3..**

## *Структура и форматы машинных команд.*

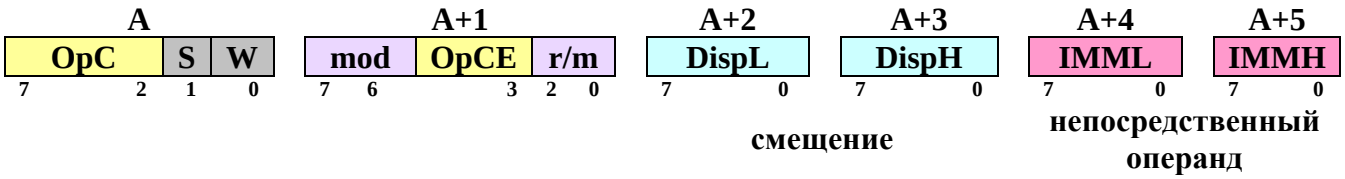
**Структура машинной команды задаёт основные части (поля) машинной команды и определяет их назначение. В свою очередь, формат машинной команды определяет разрядность, как всей команды, так и отдельных её полей. Под полем принято понимать совокупность последовательных битов формата (команды или данных), которые имеют определённое общее назначение. Примерами отдельных полей машинных команд могут являться**

- **поле кода операции (OpC);**
- **адрес базового или индексного регистров (Base, Index);**
- **смещение(Disp);**
- **непосредственный операнд (IMM);**

**Длина машинной команды без учета возможных префиксных байтов составляет от 1 до 6 байт. Префиксные байты имеют специальный код, отличный от кода операции и оказывают то или иное влияние на выполнение только одной машинной команды, следующей за префиксом. Виды префиксов : Rep (повторения), Seg (замена сегмента), Lock ( блокировка шины).**

**В базовой модели Intel 8086 используется более 10 разнообразных форматов команд. Пример формата команды максимальной длины (двухоперандная команда с постбайтом адресации и непосредственным операндом)**

**адреса в ОП**



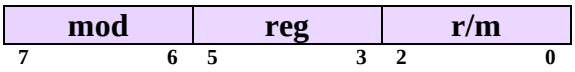

**Специальные биты OpC - W(Word) и S(Sign Extended). Расширение Word определяет длину операндов (W=0 -> байт, W=1 -> слово). А S=1 - необходимость знакового расширения непосредственного операнда.**

**Знаковое расширение операнда имеет место только для комбинации S,W = 1,1, при этом в машинной команде задаётся только один младший байт (IMML) непосредственного операнда, т. к. операнд команды, задаваемый постбайтом адресации, является двухбайтным, то для приведения непосредственного операнда к тому же формату (слово) производится предварительное его расширение на старший**

**байт. При этом операнды рассматриваются как целые знаковые числа, естественно, представленные в дополнительном коде. В связи с чем, старший байт IMMH заполняется нулями для положительного операнда или единицами для отрицательного. Фактически, все биты старшего байта становятся равными знаковому биту младшего байта, что и называется знаковым расширением операнда. При W=0, бит S становится неактуальным.**

**Постбайт адресации используется в команде для задания режимов адресации, в принципе, для обоих операндов. В приведенном примере формата, постбайт адресует только один операнд, поскольку второй из операндов задан непосредственно в команде.**

**Структура постбайта адресации для двухадресной команды имеет вид:**



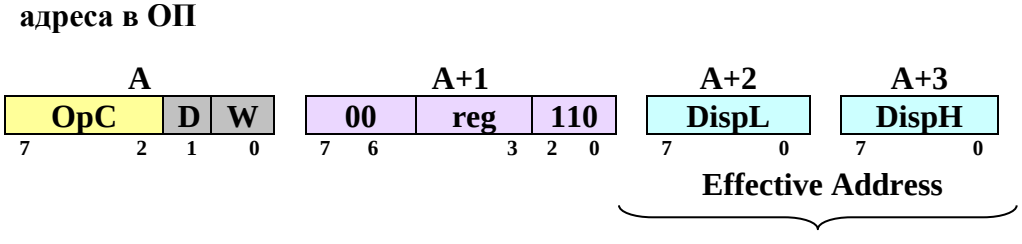

**Т. к. поле reg в рассматриваемом формате не используется, то на его месте задается расширение кода операции. Назначение полей постбайта см. Эл. Консп.**

**Для операнда, находящегося в памяти, поле r/m (register/memory) задаёт компоненты эффективного адреса Base и/или Index неявным образом (см. таблицу кодирования). В базовой модели для базовой компоненты EA могут использоваться только регистры BX и BP. А для индексной компоненты только регистры SI и DI.**

**С помощью постбайта адресации могут быть реализованы следующие режимы адресации:**

- **прямая регистровая;**
- **прямая адресация памяти EA = Disp;**
- **базовая EA = Bese + Disp;**
- **индексная EA = Index + Disp;**
- **базово – индексная без смещения EA = Base + Index;**
- **базово – индексная со смещением EA = Base + Index + Disp;**
- **косвенная регистровая EA = [РОН], где РОН = BX, SI, DI.**

**Прямая адресация памяти реализована, как исключительный случай при следующих значениях полей постбайта адресации: mod = 00, r/m = 110. В этом случае формат команды принимает вид**




**В старших моделях процессоров, точнее, начиная с модели i386, используется 32 – битная адресация и 32 – битные операнды. Для сохранения совместимости с 16 – битными моделями в старших моделях имеется возможность использования, как 16 – битной адресации, так и 16 – битных операндов.**

**Глобально, размер адресации и операндов задаётся специальным битом D – Default Size (размер по умолчанию), этот бит находится в Description Segment (описатель сегмента). Локальную установку разрядности адреса и/или операнда по**

**сравнению с принятой по умолчанию (по значению бита D) можно осуществить с использованием соответствующих префиксов AS – Address Size, OS – Operand Size.**

**Для 32 – битных моделей максимальный размер машинной команды составляет 12 байт (без учета возможных префиксов)**

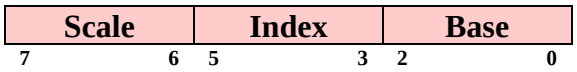
**2байта – код операции**

**1байт – постбайт адресации**

**1байт – дополнительный байт адресации (SIB – Scale Index Base)**

**4 байта – операнд**

**Байт SIB используется для задания базово – индексной адресации с масштабированием**




**В отличие от 16 – битного адреса, где базовый и/или индексные регистры задаются неявно, в 32 – битной адресации поля Base и Index задают прямые адреса РОНов, в которых размещаются базовая и индексная компонента ЕА.**

**Еще одним существенным отличием является возможность использования практически любых РОНов в качестве базы и/или индексов.**

## ***Базовая система команд.***

**Система команд компьютера включает в себя перечень машинных команд, реализуемых непосредственно аппаратными средствами. Именно система команд определяет возможности компьютера в плане решения разнообразных задач обработки данных.**

**Основной характеристикой системы команд является её мощность. Обычно под этим термином понимается количество разнообразных мнемокодов, используемых для символического обозначения на ассемблере.**

**Существует и другой подход к определению мощности системы команд, когда в разнообразие машинных команд включают не только разнообразие мнемокодов, но и разнообразие используемых режимов адресации и форматов команд.**

**Например, с использованием первого подхода, мощность системы команд базовой модели Intel 8086 составляет 113 мнемокодов. С учетом же возможности использования разнообразных режимов адресации и форматов машинных команд при втором подходе, мощность системы команд составляет не менее 4000 машинных команд.**

**При переходе от младших моделей к старшим система команд процессора неуклонно расширяется. Основными причинами расширения для семейств Intel 90X86 Pentium являются:**

1. **Поддержка защищённого режима (i286).**
2. **Внедрение блока FPU (Float Pointer Unit) в один кристалл CPU. Система команд блока FPU, кроме арифметических команд обработки данных с плавающей точкой, включают в себя большой набор трансцендентных команд для вычислений значений большинства элементарных функций ($\sqrt{}$, ln, exp,sin,cos,tg,arctg,arcos,arcsin).**
3. **Включение в Pentium блока MMX (Multimedia Extension), система команд, которая содержит порядка 60 команд для поддержки принципа векторной обработки на уровне целочисленных данных.**

**Отличие векторной обработки и, соответственно, векторных команд от скалярной обработки и, соответственно, скалярных команд, состоит в том, что операндами векторных команд являются вектора, последовательно расположенные в памяти.**

**Суммарная длина векторных данных составляет 64 бита, т. е. количество элементов вектора весьма ограничено. Блок MMX принято считать первым использованием принципов векторной обработки в микропроцессорах. Сами идеи векторной обработки появились еще в 60 – е годы и были реализованы уже в 70 – е годы в первых суперкомпьютерах ILLIAC – IV, CRAY – 1. В отличие от микропроцессорной реализации длина вектора в этих компьютерах составляла 64 элемента. MMX – векторная обработка, но для целочисленных данных.**

**4. Внедрение в кристалл процессора блока SSE (SSE2)**

**SSE – Streaming SIMD Extension - потоковое SIMD расширение**

**SIMD – Single Instruction Multiple Data**

**Внедрение порядка 80 команд, поддерживающих векторную обработку в отношении данных с плавающей запятой.**

**С учетом разнообразных расширений, мощность степени команд последних моделей Pentium составляет более 400 мнемокодов.**

**По функциональному назначению команды базовой модели принято разделять на следующие группы:**

- **арифметические команды(ADD,SUB,IMUL,MUL,DEC,INC,NEG,CMP..);**
- **логические команды (побитовой обработки)(AND,OR,XOR,NOT,TEST);**
- **команды сдвигов(SAR,SAL,SHL,SHR,ROL,ROR,RCL,RCR);**
- **команды управления программами(JMP,J/cond,LOOP,CALL,RET).**

## CISC- и RISC – архитектура.

**Непосредственно с мощностью использования системы команд связаны два основных направления в архитектуре компьютера:**

**CISC – Complex Instruction Set Computer – компьютер с расширенной системой команд**

**RISC – Reduced Instruction Set Computer - компьютер с сокращенной системой команд.**

**Вся история развития компьютеров с CISC - архитектурой сопровождалась постоянным развитием и расширением их системы команд. Статистические исследования, широко проводимые в 70 –е годы в отношении частоты использования различных команд, позволили сформулировать достаточно актуальный в своё время принцип "80X20".Суть которого в том, что на 20% машинных команд из общей системы команд процессор затрачивает порядка 80% своего бюджета, т. е. очень многие машинные команды оказываются практически невостребованными при разработке программных продуктов.**

**Сложность используемой системы команд в первую очередь сказывается на сложности устройства управления. При реализации CISC – процессора на одном кристалле в виде СБИС, на долю устройства управления приходится от 30% до 60% площади кристалла.**

**В соответствии с реализацией принципов микропрограммного управления, используемого во всех CISC - процессорах, в состав устройства управления включена достаточно большая микропрограммная память для хранения микрокодов по реализации различных машинных команд. Например, в ЭВМ VAX – 11 ёмкость микропрограммной памяти составляет 50 Кбайт.**

**Основные особенности RISC – архитектуры.**

**Термин RISC был впервые введён в 1980 г. Дэвидом Паттерсоном, профессором Калифорнийского Университета.**

1) **Использование сравнительно небольшого множества машинных команд, наиболее часто используемых при решении задач обработки данных. Первые модели RISC – процессоров (середина и конец 80- ых г.г.) имели мощность системы команд менее 100, в современных моделях RISC – процессоров - примерно 150. Система команд современного RISC – процессоров включает в себя целочисленную арифметику, арифметику с плавающей точкой, мультимедийные расширения(векторные команды).**
2) **Стремление к выполнению большинства машинных команд за 1 машинный такт (машинный цикл).**

**Под одним машинным тактом понимается интервал времени между двумя последовательными синхросигналами, подаваемыми на вход процессора от генератора (как правило, внешняя микросхема). Величина, обратная длительности машинного такта, называется тактовой частотой, это одна из самых важных характеристик процессора, как и компьютера, определяющая его быстродействие (производительность).**

**За 1 машинный такт в процессоре выполняются элементарные действия, называемые микрооперациями, например, пересылка между двумя регистрами или выполнение элементарных операций в АЛУ, таких как сложение или логическое умножение. Управляющее слово, инициирующее выполнение одной или нескольких совместимых во времени микроопераций, называется микрокомандой.**

**Совокупность микрокоманд для реализации какой – нибудь сложной машинной команды называется микропрограммой. Микропрограммы составляются и отлаживаются при проектировании компьютера и хранятся в постоянной памяти, называемой памятью микропрограммы. Она входит в состав блока (устройства управления). В RISC – процессорах грань между машинной командой и микрокомандой практически стирается.**

3) **(как следствие из 2)) В RISC – процессорах для реализации устройства управления используется принцип жесткой логики как альтернатива принципа микропрограммной логики. Это означает, что устройство управления реализуется как схемный автомат с использованием автоматной модели Мили Мура. В виду упрощения устройства управления площадь, занимаемая им на кристалле для RISC – процессоров составляет 5 – 10 %, а для CISC – процессоров – 30 -50%**

**Освобождённая площадь кристалла используется для:**

- **Увеличения внутренней регистровой памяти. В современных моделях RISC – процессорах число внутренних регистров может быть несколько сотен (до 500).**
- **Увеличение объёма внутренней Кэш – памяти. Типичный размер внутрикристальный Кэш первого уровня 64 – 128 Кбайт.**

**Замечание: в некоторых моделях используется внутрикристальная Кэш – память и второго уровня.**

- **Внедрение в кристалл дополнительных блоков для реализации векторной обработки (мультимедийные расширители).**

4) **Использование большого числа внутренних регистров создаёт дополнительную сложность при выходе на обработку прерывания, связанные с необходимостью сохранения контекста прерываемой программы. Для**

**уменьшения времени издержек на переключение программ в RISC – процессорах зачастую используется механизм регистровых окон. Идея состоит в том, что каждой программе выделяется некоторое подмножество регистров, образующих окно.**

5) **Ограниченное число форматов машинных команд и используемых режимов адресации. Используется 3 – 5 форматов машинных команд фиксированной длины и 2 - 3 основных режима адресации (косвенная адресация в RISC – процессорах не используется). Всё это делается для максимального упрощения процесса декодирования команды и формирования адресов операндов, что в свою очередь приводит на минимизацию затрат на блок управления.**
6) **Все команды обработки данных реализуются только над регистровыми операндами (команды типа reg/reg). Для обмена с памятью используются специальные команды типа LOAD (memory -> reg) и STORE (reg -> memory).**
7) **Широкое использование принципов суперскалярной и суперконвейерной обработки.**

**Основные модели RISC – процессоров:**

| Модель процессора | Полное название | Фирма изготовитель | Полное название |
|---|---|---|---|
| SPARC | Scaleable Processor ARChitecture | SUN Microsystems | SUN Microsystems |
| Micro SPARC | 32 разрядные | | |
| Super SPARC | | | |
| Hyper SPARC | | | |
| Ultra SPARC | 64 разрядный | | |
| ALPHA | ALPHA | DEC/HP | Digital Equipment Corporation/Hewlett Packard |
| Power PC | Performance Optimized With Enchanced RISC | IBM/Motorola | International Business Machines/Motorola |
| PA RISC | Precision Architecture (64разрядный) | HP | Hewlett Packard |

## Режимы работы процессора. Привилегированные команды.

**В целях разграничения доступа к системным ресурсам со стороны прикладных и системных программ, в современных моделях процессоров в том или ином виде существует два основных режима функционирования:**

**- прикладной**
**- системный.**

**В системном режиме допускается исполнение любых машинных команд. В прикладном режиме не допускается исполнение так называемых привилегированных команд. В простейшем случае режим работы процессора задаётся с помощью специального бита, находящегося в каком – либо системном регистре. Например, в процессоре IBM 370 бит режима находится в слове состояния программы PSW (Program Status Word).Два альтернативных состояния процессора называются Task/Supervisor.**

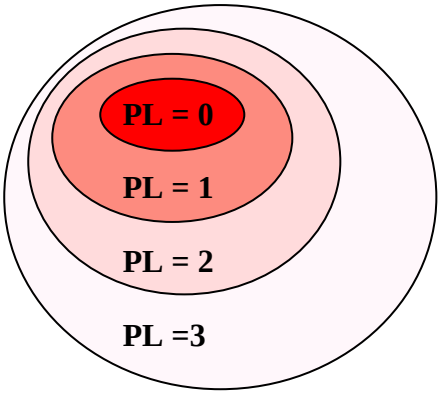
**Аналогичный бит режима System/User имеет место в моделях системы VAX(DEC), который находится в PS (Processor Status).**

**В процессорах семейства 80X86 Pentium используется более сложная интерпретация (способ) для задания режима работы процессора в плане разделения системного и пользовательского режимов. С помощью специального бита PE (Protect Enabled) в управляющем регистре CR0 задаётся или реальный режим (PE = 0) или защищённый режим (PE = 1).**

**При использовании защищённого режима вступают в действие различные средства защиты. Одним из средств защиты, поддерживающимся на аппаратном уровне, является защита по уровням привилегий (кольцам защиты).**

**Аппаратно поддерживаются 4 уровня привилегий для сегментов и 2 уровня привилегий для страниц. Идея защиты по уровням привилегий состоит в присвоении различным сегментным объектам, в частности сегментам кода, данных, стека, определённого уровня привилегий. Этот уровень отражается соответствующим двухбитным полем DPL(Descriptor Privilege Level), размещаемом в дескрипторе (описателе сегмента).**

**Уровень привилегий определяет степень важности и доступности сегмента. Наивысшим уровнем привилегий является PL = 0, и он присваивается программам ядра, наинизший уровень PL = 3, он присваивается прикладным программам.**




**Общее правило защиты предполагает возможность обращений или доступа из внутренних колец. Во внешние. Попытки обращений из внешних колец во внутренние в общем случае пресекаются средствами защиты с выходом на прерывание специального типа Тип 13 (“Нарушение общей защиты”).**

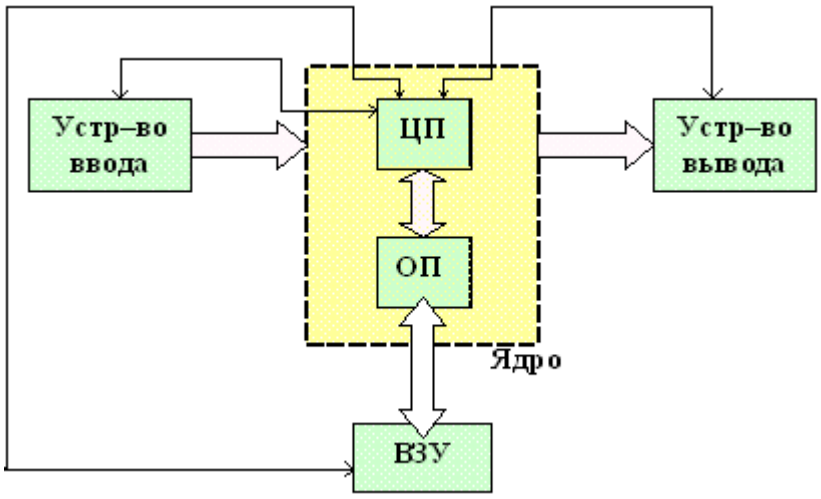
**Пользовательский режим ассоциируется с уровнем привилегии PL = 3, системный режим с уровнем привилегий PL = 0.**

**Современные операционные системы, в частности Windows и Unix поддерживают только 2 уровня привилегий (User/Supervisor). Соответствующие биты для двух уровней используются на страничном уровне и размещаются в страничных дескрипторах. В процессорах семейства Intel к привилегированным командам относятся:**

1) **Команды загрузки и сохранения системных регистров.**
2) **Команды манипуляции флагом IF: CLI (0 - > IF), STI (1 - >IF).**
3) **Команды ввода/вывода: IN/OUT, INS/OUTS.**
4) **Команда останова процессора - HLT.**

**Попытка выполнения привилегированных команд в пользовательском режиме (PL = 3) приводит к выводу на прерывание 13. Почти все привилегированные команды, за исключением команд ввода/вывода и манипуляций флагом IF требует для своего выполнения наивысшего уровня привилегий PL = 0.**

## Упрощенная структура компьютера (ЭВМ).




**Независимо от принадлежности компьютера некоторому классу или типу, его в первом приближении можно разделить на 2 части:**

**1. центральную;**

**2. периферийную.**

**Центральную часть принято называть ядром. В ядро входят два основных устройства компьютера: ЦП и ОП.**

**Периферийную часть можно условно представить устройствами трех типов:**

- **внешние запоминающие устройства, которые образуют внешнюю память (к ним относятся накопители на магнитных дисках и на магнитных лентах);**
- **устройства ввода;**
- **устройства вывода.**

**Обмен информацией между ядром и периферией, а так же между устройствами ядра осуществляется на уровне аппаратных интерфейсов. Организация обмена между ядром и периферийной частью компьютера возлагается**

**на систему ввода/вывода (I/OS – Input/Output System). Система ввода/вывода представляет собой аппаратно - программный комплекс.**

**Аппаратная часть I/O S включает в себя:**

1. **собственно периферийные устройства (разделённые на УВ/В и ВЗУ);**
2. **контроллеры ПУ(устройства управления);**
3. **контроллеры для организации обмена, в частности, контроллер DMA – Direct Memory Accses (ПДП-прямой доступ к памяти) PIC (Program Interrupt Controller – Программируемый Контроллер Прерываний);**
4. **интерфейсы (шины);**
5. **система прерываний.**

**Программная часть I/O S включает в себя:**

1. **супервизор (Supervisor) в/в;**
2. **драйверы ВУ.**

**Для современных программных средств I/O S типичным свойством является многоуровневая (иерархическая) организация, в частности, многоуровневые драйверы.**

**Программное обеспечение I/O S разделяется на устройство-зависимую часть и устройство независимую часть. Устройство-независимое ПО выполняет следующие функции:**

- **буферизация;**
- **защита (сообщения об ошибках);**
- **блокирование (блочный характер передачи);**
- **обеспечивание единообразного программного интерфейса для драйверов устройств.**

## Организация ввода/вывода.
## Понятие, основные характеристики и уровни представления интерфейса.

**В общем плане под интерфейсом принято понимать способ сопряжения и взаимодействия между несколькими объектами. В отношении компьютеров принято рассматривать множество понятий интерфейса: аппаратный, программный, пользовательский. В отношении аппаратных интерфейсов используются следующие понятия: интерфейс памяти, интерфейс ввода/вывода, интерфейс периферийных устройств (малый интерфейс). Существует большее количество подходов к определению аппаратного интерфейса. Основными компонентами в различных понятиях аппаратного интерфейса, являются:**

**1) Совокупность линий, шин, обеспечивающих обмен информацией между устройствами.**

**2) Алгоритм (протокол) обмена, определяющий последовательность организации передачи информации по линиям интерфейса.**

**3) Разделение интерфейса на ряд уровней представлений.**

**На обобщённой структуре двойными линиями обозначаются структуры, по которым осуществляются передача информации данных между компонентами компьютера, а одинарными – обозначаются связи по управлению. Тем самым подчёркивается, что центральный процессор выполняет в компьютере двойную функцию: как устройство обработки (выполняет заданные программы) и как устройство управления всеми компонентами компьютера.**

**От ЦП к остальным устройствам по линиям связи передаются управляющие сигналы (приказы, команды в/в); в обратную сторону передаются сигналы о состоянии устройств, в частности об их готовности к обмену, а также запросы прерываний (например, для идентификации момента завершения операции в/в).**

**Основными типами линий (шин), входящих в состав аппаратного интерфейса, являются:**

- ⇒ **шина адреса;**
- ⇒ **шина данных;**
- ⇒ **шина управления (для передачи сигналов, управляющих обменом);**
- ⇒ **линии синхронизации (по шинам передаются сигналы, синхронизирующие передачу информации по интерфейсу);**
- ⇒ **линии запросов прерываний (по ним передаются сигналы прерывания от устройств, подключаемых к интерфейсу);**
- ⇒ **линии питания;**
- ⇒ **линии заземления.**

**Основные характеристики интерфейса:**

1. **Пропускная способность определяется максимальным количеством бит или байт данных, передаваемых по интерфейсу за одну секунду.**
2. **Информационная ширина (количество бит или байт данных, передаваемых параллельно по шине данных, т.е. разрядность линии).**
3. **Максимально возможное удаление устройств, подключаемых к интерфейсу.**

**Алгоритм (протокол) обмена обычно представляется с помощью временных диаграмм. Определяется порядок следования и допустимые параметры(амплитуда, длительность) для управляющих и информационных сигналов при работе интерфейсов в различных режимах.**

**Уровни представления интерфейсов:**

- **Логический уровень определяет состав, наименование, назначение шин интерфейса, а также порядок передачи информации по этим линиям (протокол обмена).**

- **Физический уровень определяется параметрами электрических, оптических сигналов, передаваемых по линиям интерфейса.**

- **Конструктивный уровень определяет физическую реализацию шин интерфейса: скрученная (витая) пара, коаксиальный кабель; а также определяет виды разъемов и распределение линий интерфейсов по контактам разъема.**

## Шины (интерфейсы) персональных компьютеров на базе процессоров Pentium.

**Структура современных ПК отличается большим разнообразием используемых шин или интерфейсов.**

| Название шины | Полное название | Перевод | Последовательная /параллельная | Комментарий |
|---|---|---|---|---|
| FSB | Front – Side Bus | Шина переднего плана | параллельная | обеспечивает связь между ЦП и ОП |
| BSB | Back – Side Bus | Шина заднего плана | параллельная | обеспечивает связь между ЦП и внешнего КэшL2, отличается большой пропускной способностью |
| PCI Intel - 1990 | Peripheral Component Interconnect | Соединение периферийных компонент | параллельная | шина расширения |
| ISA | Industry Standard Architecture | Стандартная промышленная архитектура | параллельная | шина расширения; ISA –16разрядн. EISA-32разрядн. Для подключения принтера, модема, звуковой карты |
| SCSI | Small Computer System Interface | Интерфейс малых вычислительных систем | параллельная | для подключения периферийных интерфейсов (в частности, магнитных дисков) |
| USB | Universal Serial Bus | Последовательная универсальная шина | последовательная | для подключения медленных устройств (например клавиатуры) |

## Основные аспекты организации ввода/вывода.

**1.Структура компьютера в плане организации связей между ядром и периферийными устройствами:**

**а) структура с единым интерфейсом (с магистральным интерфейсом, с общей шиной);**

**б) многошинная структура, рис. 1.11 (Таненбаум);**

**в) структура с каналами (процессорами) ввода/вывода (Цилькер);**

**2. Адресация к ВУ или ПУ. Основным аспектом, связанным с адресацией ВУ, является объединение или разделение адресных пространств памяти и ввода/вывода.**

**3.Способ организации ввода/вывода:**

**а) программный (программно управляемый, программируемый) ввод/вывод;**

**б) ввод/вывод по прерыванию (управляемый прерываниями);**

**в) ввод/вывод с использованием прямого доступа к памяти;**

**г) канальный ввод/вывод.**

## Упрощенная структура компьютера с единым интерфейсом.

**Ещё в конце прошлого века подобная структура являлась канонической и стандартной для большинства моделей мини ЭВМ, микро ЭВМ и ПК. Примером единого интерфейса может служить стандартный интерфейс Unibas, который использовался в компьютерах фирмы DEC(PDP – 11, VAX – 11, СМ ЭВМ).**

**Основными особенностями компьютеров с единым интерфейсом являются:**

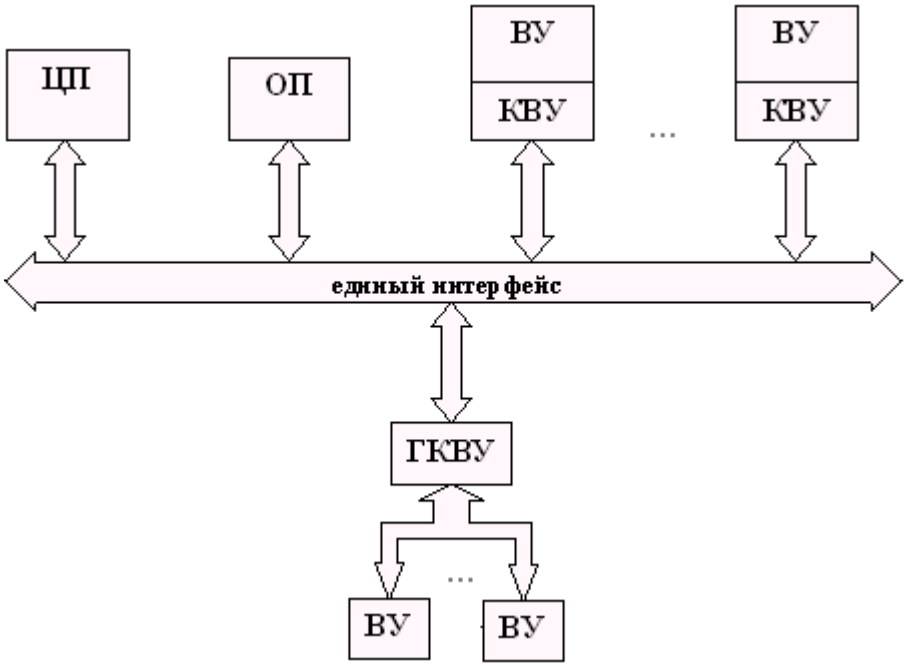
**1) все устройства, как центральные, так и периферийные, для связи между собой используют одни и те же шины адреса, данных и управления;**

**2) в любой момент времени по единому интерфейсу может быть организована передача данных только между двумя устройствами;**

**3) в соответствие с п.2 при наличии большого числа устройств, единый интерфейс становится “узким местом” (bottle neck), в связи с чем подобная структура усовершенствовалась путём использования локальных дополнительных шин;**

**4) как правило, использование единого интерфейса предполагает единообразие операции с памятью (чтение и запись) и ввода/вывода, в связи с этим предполагается использование объединённого адресного пространства для памяти и ввода/вывода (ввод/вывод, отображённый на память).**

**В современных ПК подобная структура не используется.**



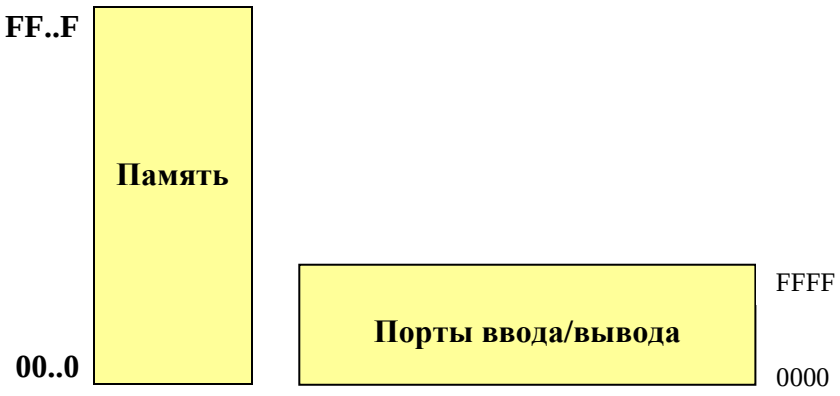

## Адресация ВУ.

**Адресация собственно ВУ в современных компьютерах используется достаточно редко. Примером использования фактического адреса ВУ могут служить команды ввода/вывода IBM 370. В современных ПК адресация ВУ осуществляется на уровне программно доступных регистров контролеров ВУ, которые называются портами ввода/вывода.**

## Способы адресации портов ввода/вывода и их сравнительный анализ.

**Различие способов адресации связанно с использованием раздельного или единого адресного пространства для памяти и портов ввода/вывода.**

**Раздельное адресное пространство.**




**Использование раздельного адресного пространства предполагает, что одни и те же адреса могут применяться как для адресации ячеек памяти, так и для**

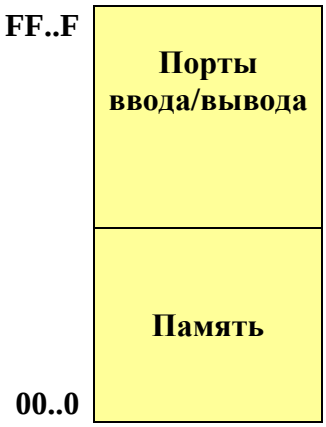
**адресации портов ввода/вывода. При этом в системе команд процессора должны использоваться специальные команды для ввода/вывода, отличные от команд обмена с памятью. Достоинством подобного подхода принято считать возможность использования более короткого адреса порта ввода/вывода по сравнению с адресом ячейки памяти.**

**Так, например, в системе команд процессора Intel для адресации портов ввода/вывода может использоваться либо 1 байт (при прямой адресации порта), либо 2 байта (при косвенной адресации порта).**

**При косвенном задании адреса используется регистр DX (неявно адресуемый), в котором и находится адрес порта ввода/вывода.**

## Единое адресное пространство.

**Использование единого адресного пространства существенно влияет как на систему команд процессора, так и на управление вводом/выводом на аппаратном уровне.**




**Некоторая область адресного пространства в старших адресах используется не для адресации памяти, а для адресации портов ввода/вывода. Подобная идея была впервые реализована в мини ЭВМ PDP – 11(DEC). Подобный способ хорошо вписывается в рамки так называемого магистрального интерфейса.**

**В системе команд отсутствуют специальные команды ввода/вывода, и обмен между памятью и портами ввода/вывода реализуется по аналогии с обменом между процессором и памятью с использованием обычной команды типа MOV.**

**Организация ввода/вывода с отображением на память обладает следующими достоинствами:**

1) **Не требуются специальные команды ввода/вывода (упрощение системы команд).**
2) **Возможность использования любых видов обработки (реализуется и/или логическими командами применительно к содержимому портов ввода/вывода).**

**Недостатками использование совмещенного адресного пространства являются:**

1) **Усложнение управления кэшированием, связанное с необходимостью запрета кэширования адресного пространства, выделенного для портов ввода/вывода.**
2) **Сложность реализации этого способа для многошинной архитектуры.**

**Для одношинной архитектуры любой адрес, выставляемый на шину адреса, сравнивается всеми модулями памяти и ввода/вывода на предмет собственной принадлежности. При раздельных шинах памяти и ввода/вывода требуются дополнительные затраты для проверки адреса на его принадлежность к памяти или вводу/выводу.**

**Возможные способы решения проблемы:**

- **первоначальный запрос направляется к памяти по быстрой шине, а затем, если память не может ответить на него, то запрос перенаправляется в шину ввода/вывода;**

- **фильтрация адресов специальной микросхемой, отделяющей адреса памяти от адресов портов ввода/вывода; в частном случае подобную функцию может выполнять мост PCI, в состав которого входят специальные регистры диапазона; при этом все адреса, попадающие в мост и принадлежащие выделенному диапазону, передаются в дальнейшем не к памяти, а непосредственно в шину PCI, которая служит шиной ввода/вывода.**

## Способы организации ввода/вывода.

### Программно управляемый ввод/вывод (ПУВВ). (гл.8 Цилькер)

**Ввод/вывод осуществляется при непосредственном участии ЦП. Реализация ввода/вывода производится специальной программой (драйвером ВУ), в котором выполняются следующие действия:**

1) **Пересылка порции данных между ОП и портом ввода/вывода (как правило, в качестве промежуточного звена используется какой – либо регистр ЦП).**
2) **Проверка готовности ВУ к обмену (сводится к опросу регистра состояния контроллера ВУ).**
3) **Ожидание готовности ВУ.**
4) **Изменение (модификация) параметров пересылки, в частности текущего адреса области ОП для ввода/вывода и счетчика длины пересылаемого блока.**
5) **Проверка завершения передачи путем сравнения счетчика с некоторым конечным значением (например, с нулем для декрементного счетчика).**

**Достоинства и недостатки ПУВВ PIO(Programming Input/Output):**
**Основным достоинством PIO принято считать относительную простоту его реализации, а основным недостатком – неэффективное использование ресурсов ЦП.**

### Ввод/вывод по прерыванию.

**Использование этого способа организации ввода/вывода позволяет устранить основной недостаток предыдущего способа.**

**Основная идея: после осуществления элементарных действий по пересылке очередной порции данных, ПЦ вместо переключения в состояние активного ожидания с опросом готовности ВУ (как в первом способе) переходит к выполнению другой программы (планировщик заданий Операционной Системы запускает другой процесс); в свою очередь, устройство ввода/вывода о своей готовности продолжать операцию ввода/вывода с очередной порцией данных сообщает ПЦ с помощью сигнала прерывания; сигналы прерывания от ВУ в ПЦ Intel приходят на специальный вход INTR(Interrupt Request); при получении сигнала от ВУ ПЦ выполняет следующие действия:**

1) **завершает выполнение текущей команды программы;**
2) **сохраняет контекст прерываемой программы;**
3) **идентифицирует источник прерывания и преобразует код источника в начальный адрес программы обработчика прерываний;**
4) **загружает адрес обработчика в счетчик команд;**
5) **переходит к выполнению программы обработчика прерываний.**

**Замечание: обработчики прерываний от устройств ввода/вывода связаны с драйверами соответствующих устройств.**

**Несмотря на очевидное преимущество этого способа по сравнению с предыдущим, его недостатком являются большие издержки времени, связанные с контекстным переключением из прерванной программы на обработчик прерываний и обратно. Для устройств с посимвольным обменом эти переключения имеют место после пересылки, если не каждого байта, то сравнительно небольшого их числа (4-8ба), определяемых шириной шины данных.**

## Прямой доступ к памяти – DMA (Direct Memory Access).

**Основное отличие DMA от предыдущих способов в том, что участие ПЦ в организации ввода/вывода сводится к минимуму. ПЦ организует лишь так называемую инициализацию DMA, а также реакцию на завершение операции ввода/вывода.**

**Режим DMA используется для организации так называемых блочных пересылок. Типичным ВУ с блочным обменом являются накопители на магнитных дисках и магнитных лентах. Управление обменом в режиме DMA осуществляется специальным устройством (микросхемой), называемым контроллером DMA – DMAC. При этом контроллер DMA реализует обмен не на программном, а чисто на аппаратном уровне (DMAC – микропрограммный автомат).**

**DMAC содержит некоторое число программно доступных регистров, представленных для ПЦ адресуемыми портами ввода/вывода.** (гл.8 Цилькер,Орлов) **Инициализация DMA сводится к заданию режима работы и необходимых адресов путем пересылки требуемой информации из ЦП в регистры контроллера DMA. На этапе инициализации задаются следующие основные данные:**

- **начальный адрес блока памяти (области памяти), используемого при обмене;**
- **объем передаваемого блока памяти в байтах (типичный размер блока при обмене с жестким диском составляет 512 байт, длина или объем сектора);**
- **код операции обмена (ввод или вывод);**
- **адрес устройства прямого доступа (адрес ВУ задается в связи с тем, что стандартный контроллер DMA включает в себя 8 каналов прямого доступа).**

**DMA может быть реализован в одном из следующих основных режимов (по Цилькеру):**

| Название режима | Описание режима |
|---|---|
| Блочная пересылка | Захват шины на весь период пересылки блока; на весь период DMA ЦП не имеет доступа к шине памяти, в этот период процессор может продолжать работу по программе с обращением к КЭШ – памяти. |
| Пропуск цикла | После пересылки слова DMAC освобождает шину на один цикл, предоставляя ее ЦП. |
| Прозрачный режим | DMAC имеет доступ к шине только в тех циклах, в которых ЦП в ней не нуждается. |

1) **Поддержка DMA на аппаратном уровне в процессоре фирмы Intel осуществляется на уровне входного сигнала HOLD(захват шины) и выходного сигнала HLDA(Hold Acknowledgment) – сигнал подтверждения захвата. Сигнал HOLD инициализирует DMAC при начале цикла обмена; ЦП, получив этот сигнал, отключается от шины и выставляет активный уровень сигнала подтверждения HLDA, получив этот сигнал, DMAC начинает цикл блочного обмена.**

**2) Стандартные контроллеры DMA позволяют реализацию следующих видов обмена:**

**Port -> Mem**
**Mem -> Port**
**Mem -> Mem (обмен с видеопамятью)**
**Port - > Port**

2) **В современных моделях ПК для обмена с жесткими дисками наряду с DMA также используется и PIO.**

## Канальный ввод/вывод (КВВ).

**Канальный ввод/вывод основан на использовании в архитектуре ЭВМ специализированных процессоров, ориентированных на организацию ввода/вывода. Эти процессоры обычно называются <u>каналами</u> ввода/вывода. Канальный ввод/вывод является программно управляемый, реализуется с помощью специальной программы, которая носит название <u>канальной</u>.**

**Канальные программы для организации обмена с различными типами ВУ хранятся в основной памяти. В связи с тем, что КВВ является процессором, правда специализированным, управление порядком выполнения команд канальной программы осуществляется с помощью своеобразного счетчика команд.**

**Команды канальной программы называются УСК (Управляющими Словами Канала). УСК содержат следующую основную информацию:**

**1.код команды (например:прочитать или записать, проверить состояние ВУ,т.п.);**

**2. начальный адрес области ОП, с которой осуществляется обмен;**

**3. длина передаваемого блока в байтах;**

**4. различные идентификаторы и признаки, влияющие на организацию обмена:**

- **<u>признак цепочки данных;</u>**

**при его установке следующая команда канальной программы выполняет то же действие, что и данная, но с другой областью памяти; с помощью цепочки данных обеспечивается непрерывный обмен с несмежными областями ОП без привлечения ЦП;**

- **<u>признак цепочки команд;</u>**

**установка этого признака в текущей команде указывает каналу на необходимость продолжения канальной программы после завершения текущей команды путем выборки следующей команды; сброшенный признак цепочки команд означает, что текущая команда является последней в канальной программе (аналог - HLT);**

- **<u>признак программно управляемого прерывания;</u>**

**установка этого признака в какой – либо команде сопровождается выдачей сигнала прерывания из КВВ в ЦП в момент выборки этой команды на исполнение; получив сигнал прерывания, ЦП может,**

**например, приступить к обработке блока данных, передача которого завершилась при выполнении предыдущей части канальной программы.**

## Основные функции КВВ:

**1) Функции по установлению логической связи между ВУ и ОП.**
- **а) прием и декодирование команд ввода/вывода от ЦП;**
- **б) инициирование выполнения канальной программы при получении команды SIO (Start Input/Output) от ЦП;**
- **в) проверка состояния ВУ, участвующего в обмене, и передача в ЦП информации о его готовности или неготовности к обмену;**

**2) Функции, связанные с непосредственной передачей данных между ВУ и ОП.**
- **а) последовательная выборка команд канальной программы из ОП, их декодирование и выполнение;**
- **б) обеспечение приема, передачи, контроля и промежуточного хранения данных при обмене между ОП и ВУ;**
- **в) формирование текущих адресов ОП, по которым записываются или считываются передаваемые данные;**
- **г) согласование форматов данных, передающихся по интерфейсу ввода/вывода, с форматом интерфейса ОП (как правило, ширина интерфейса ввода/вывода составляет 1, 2 или 4 байта, что меньше ширины интерфейса ОП: 4, 8, 16 байт);**
- **д) подсчет числа передаваемых байт данных с целью определения момента завершения передачи блока данных;**
- **е) выработка последовательности синхронизирующих и управляющих сигналов в соответствии со стандартом интерфейса ввода/вывода;**
- **ж) анализ особых ситуаций в ВУ во время обмена (ошибка передаваемых данных, сбой устройства и т.п.) и информирование ЦП об этих ситуациях (с помощью запроса прерывания);**

**3) Функции, связанные с завершением обмена и разрушением логической связи между ВУ и ОП.**
- **а) определение момента завершения в программе по организации обмена между ВУ и ОП;**
- **б) передача в ЦП сигнала прерывания о завершении обмена.**

## Участие ЦП в организации КВВ сводится к выполнению следующих функций:

- **Инициирование операции ввода/вывода (реализуется командой SIO основной программы).**
- **Проверка состояния канала ввода/вывода (реализуется командой TCH – Test Channel).**
- **Проверка состояния ВУ (реализуется командой TIO – Test Input/Output).**
- **Остановка операций ввода/вывода (реализуется командой HIO – Halt Input/Output) возможно, для инициирования более приоритетной операции.**

**Команды ввода/вывода являются привилегированными, т.е. могут выполняться только программами операционной системы в режиме Supervisor (SVR).**

**Использование каналов ввода/вывода в архитектуре компьютеров, относительно к классу Main Frame и суперкомпьютеров, является мощной**

**аппаратной поддержкой реализации мультипрограммного режима обработки. Архитектура компьютера с каналами ввода/вывода, даже при наличии одного ЦП, позволяет параллельно реализовать выполнение одной программы в ЦП и процессов ввода/вывода для нескольких других программ с использованием ресурсов КВВ.**

## Классификация КВВ.

**По режиму функционирования, они разделяются на мультиплексные и селекторные. Мультиплексный канал обеспечивает параллельную работу многих ВУ. Селекторный канал работает в монопольном режиме, обеспечивая работу единственного ВУ.**

**Взаимодействие мультиплексного канала с одним из ВУ, подключенным к ВУ, принято называть сеансом связи с ВУ. В зависимости от порции данных, передаваемых через канал между ВУ и ОП за один сеанс связи, мультиплексные каналы разделяются на два вида: байт-мультиплексные и блок-мультиплексные.**

**Часть ресурсов мультиплексного канала, используемая отдельным ВУ, называется подканалом. В каждом подканале используется некоторая область памяти канала, в которой хранится информация о текущем состоянии обмена с данным ВУ. Переключение мультиплексного канала с обслуживания одного ВУ на другое сопровождается сохранением информации в памяти подканала для текущего ВУ и выборкой информации из памяти подканала для следующего ВУ.**

## Сравнение Канального ВВ с PIO и с DMA.

**Так как КВВ осуществляет организацию обмена по собственной программе, то КВВ следует считать программно управляемой, однако, в отличие от PIO, программу выполняет не ЦП, а КВВ.**

**Многие авторы сопоставляют КВВ и DMA. Аналогия между КВВ и DMA состоит в том, что оба эти способа обмена реализуются практически без участия ЦП. Так же, как и для DMA, КВВ требует участия ЦП на этапе инициализации. В частности, при инициализации от ЦП в КВВ передается начальный адрес канальной программы. Существенным же отличием КВВ от DMA является программная реализация первого и чисто аппаратная второго.**

# Организация прерываний.

**<u>Основные отличия организации прерываний в защищенном режиме по сравнению с реальным режимом.</u>**

**Организация прерываний в реальном режиме практически ничем не отличается от организации прерываний в процессоре Intel 8086. В защищенном режиме, а также в его модификации в виде виртуального режима (V – режима), механизм прерываний при особых случаях, сохранив общую реакцию на их возникновение, значительно усовершенствован.**

**Основные усовершенствования сводятся к следующим.**

1) **Трансформация таблицы векторов прерываний в дескрипторную таблицу прерываний (IDT – Interrupt Description Table).**

2) **Более сложный процесс перехода к обработчику особого случая или прерывания с привлечением системных объектов в виде шлюзов.**
3) **Возможность передачи обработчику дополнительной информации о причине возникновения особого случая с помощью так называемого кода ошибки (Error Code).**
4) **Использование дополнительных видов особых случаев, связанных исключительно с защищенным режимом, например, неприсутствие страницы или сегмента в ОП, нарушение общей защиты и т.п.**

**Для защищенного режима используется следующая классификация программных прерываний, которые обобщаются единым наименованием exception, (особый случай). В зависимости от способа возникновения особых случаев и возможности перезапуска (рестарта ЦП) после их обработки с вызвавшей их командой, принято различать три вида особых случаев:**

**а) *Fault (нарушение)* – особые случаи, которые выявляются и обслуживаются либо перед выполнением, либо во время выполнения "виновной" команды.**

**При обнаружении нарушения, сохраняемые в стеке значения CS, EIP, указывают на команду, вызвавшую это нарушение, а не на следующую команду программы. Это дает возможность осуществлять рестарт программы после устранения причины нарушения именно с виновной команды.**

**Типичными примерами нарушений могут служить отсутствие сегмента или страницы в ОП.**

**б) *Trap (ловушка)* – особый случай, который возникает непосредственно после выполнения команды, вызвавшей этот особый случай. Значение регистров CS и EIP, сохраняемые при срабатывании ловушек, указывают на команду следующую по отношению к команде, вызвавшую это срабатывание.**

**Типичным примером ловушки может служить ловушка пошагового выполнения программы, ее генератором является установленный флаг TF.**

**в) *Abort (авария, выход из процесса)* – особый случай, который не позволяет точно локализовать вызвавшую его команду осуществить рестарт программы.**

**Примером такого особого случая может служить так называемая двойная ошибка, которая имеет место в том случае, когда при обработке нарушений, связанных с неприсутствием сегмента или страницы, возникает новое нарушение. Назначением аварии является устранение возможности бесконечных циклов прерываний.**

**Сначала шанс выйти из цикла дается программе обработки исключений по двойной ошибке, а если этого не достигается, т. е. при ее выполнении обнаруживается еще одно исключение, происходит аппаратное отключение процессора, из состояния отключения процессор может быть выведен сигналом сброса или немаскируемым прерыванием.**

## Программируемый контроллер прерываний (PIC i8259A).

**Одна микросхема PIC может обслуживать 8 запросов прерываний. В современных компьютерах на базе процессоров Intel используются две микросхемы PIC, объединенных с помощью так называемого каскадного включения, что позволяет в принципе обслуживать до 15 источников прерывания.**

**Одна из микросхем является ведущей, а вторая – ведомая. Ведущий PIC связан с CPU, а ведомый PIC с ведущим. Максимальные возможности каскадного включения PIC позволяют обслуживать до 64 внешних источников запросов прерываний. Связь между ведущим PIC и CPU осуществляется по двум линиям: 1-ая линия INT PIC – INTR(CPU), 2 – ая линия INTA (CPU) – INTA (PIC).**

## Основные функции PIC.

1) **Фиксация запросов, поступающих от подключенных к нему ВУ в специальном регистре запросов - IRR.**
2) **Осуществление внутреннего маскирования запросов с помощью специального регистра – маски IMR (0 – разрешение, 1 - запрет).**
3) **Выделение наиболее приоритетного запроса из всех поступивших и незамаскированных запросов.**
4) **Выдача в CPU сигнала о наличии хотя бы одного незамаскированного запроса (по линии 1).**
5) **Выдача в CPU номера (кода) запроса в цикле подтверждения прерывания, который, в свою очередь, модифицируется CPU в адрес вектора прерываний (начальный адрес программы обработчика соответствующего прерываний).**

## Внутренняя структура PIC.

**В состав PIC входят 7 байтных регистров, основными из которых являются:**

- **IRR – Interrupt Request Register**
- **IMR – Interrupt Mask Register**
- **ISR – Interrupt Service Register (фиксируются запросы, принимаемые на обслуживание или обработку в CPU)**

**Кроме регистров, в состав PIC входят комбинационные схемы, в частности:**

1. **Схема выделения наиболее приоритетного незамаскированного запроса.**
2. **Шифратор выделенного запроса.**

**Шифратор представляет собой комбинационную схему, осуществляющую преобразование двоичного унитарного кода (код с единственной единицей) в двоичный позиционный код, в данном случае шифратор имеет 8 входов и 3 выхода.**

3. **Схема для реализации каскадирования.**

**Внешние запросы о ВУ поступают на входы ir0,..ir7.**

## Основные режимы работы PIC.

**1) FNM (Fully Nested Mode – Режим вложенных прерываний).**

**В этом режиме наивысшим приоритетом обладает запрос irq0, наинизшим irq7. Допускается прерывание прерываний, это означает, что поступление на вход PIC запроса с более высоким приоритетом, чем обрабатываемый, вызывает генерацию активного уровня выходного сигнала INT, который поступает в CPU.**

**В регистре ISR может быть несколько установленных битов.**

**Недостаток этого режима – достаточно сильная дискриминация запросов низшего уровня. В связи с этим, используется следующий режим.**

**2) ARM (Automatic Rotation Mode – Режим автоматического сдвига приоритета запросов).**

**Приоритеты линейно упорядочены и изменяются после обработки очередного запроса таким образом, что уровень обработанного запроса становится низшим, а следующий за ним уровень – высшим.**

**3) SRM (Specific Rotation Mode – Режим программно управляемых приоритетов).**

**Уровень запроса наивысшего приоритета устанавливается извне путем передачи соответствующего приказа из CPU в PIC.**

**4) PM (Polling Mode – Режим опроса).**

**PIC лишь фиксирует поступающие запросы в IRR. Анализ содержимого IRR и соответствующая реакция на него осуществляется CPU. При этом IRR предварительно считывается в CPU с помощью команды IN <порт ввода/вывода>.**

## Взаимодействие между CPU и ведущим PIC.

**1) При наличии хотя бы одного незамаскированного запроса прерываний, PIC выставляет активный выходной сигнал INT, который поступает на вход INTR CPU.**
**2) CPU завершает текущую команду программы и проверяет состояние внешних входов NMI и INTR.**
**3) Если флаг IF установлен, процессор переходит к обработке запроса. При сброшенном флаге IF обработка запроса временно откладывается до выполнения процессором команды STI (Set Interrupt).**
**4) При IF = 1 процессор генерирует активный уровень выходного сигнала подтверждения прерывания INTA.**
**5) При получении сигнала INTA, PIC выполняет следующие действия:**
**а) сбрасывает бит обрабатываемого запроса в IRR;**
**б) устанавливает бит обрабатываемого запроса в ISR;**
**в) выставляет на внешнюю шину данных (в ее младший байт) номер обрабатываемого запроса;**
**6) CPU принимает номер запроса по шине данных и модифицирует этот номер в адрес соответствующего вектора прерываний (модификация осуществляется путем умножения на 4 – сдвиг влево на 2 разряда).**
**7) Текущее значение регистра флагов, CS и IP последовательно помещаются в стек, и тем самым сохраняется минимальный контекст прерываемой программы.**
**8) Два последовательных слова из таблицы векторов прерываний загружаются в регистр IP(слово по меньшему адресу) и CS (слово по большему адресу).**
**9) Н аппаратном уровне осуществляется сброс флага IF.**
**10) Процессор переходит к выполнению первой команды обработчика прерываний.**

## Основы программирования PIC.

**Доступ к ведущему PIC осуществляется с помощью портов с адресами 20h, 21h, а к ведомому A0h, A1h. PIC имеет достаточно большое число программируемых битов (устанавливаемых программой), которые содержатся в 7 байтных регистрах. Эти регистры разделяются на две группы:**

| Приказы инициализации | Рабочие приказы |
|---|---|
| ICW – Initialization Control Word | OCW – Operation Control Word |
| ICW1, ICW2, ICW3, ICW4 | OCW1, OCW2, OCW3 |

**Слова приказов инициализации устанавливаются процедурой инициализации при включении системы и в дальнейшем не изменяются. Слова рабочих приказов используются для динамического управления обработкой прерываний и могут изменяться в ходе работы системы.**

**В порт с четным адресом выводятся слова ICW1, OCW2, OCW3. В порт с нечетным адресом выводятся остальные слова приказов.**

**Идентификация типа слова (приказ инициализации или рабочий приказ) осуществляется специальными битами. В связи с чем, неоднозначности их интерпретации не происходит. Инициализация PIC начинается выводом в порт с четным адресом приказа ICW1. Далее, в процессе инициализации контроллер принимает приказ ICW2 (в порт с четным адресом) и далее, при использовании каскадного включения, - приказ ICW3 (при использовании единственной микросхемы этот приказ не выводится). Необходимость вывода последнего приказа ICW4 определяется значением специального бита приказа ICW1.**

**При программировании PIC имеет место такая особенность, что один и тот же порт используется для инициализации нескольких регистров контроллера.**

**Приказы инициализации.**

**ICW1 имеет следующий формат:**

| A0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | 1 | LTIM | - | SNGL | IC4 |

**Бит 0 (IC4) определяет будет ли выводиться приказ ICW4 в процессе инициализации (1 – выводится, 0 – не выводится). В современных моделях ПК на базе процессоров Intel бит IC4 установлен.**

**SNGL (Single) указывает на наличие в системе единственной микросхемы PIC, если он установлен, и нескольких, если он сброшен.**

**LTIM определяет режим запуска по входам. При сброшенном бите запуск осуществляется по фронту сигналов, при установленном бите запуск осуществляется по уровню сигнала. Режим запуска фронтом вызывает сброс бита в регистре IRR, когда устанавливается соответствующий бит ISR. Режим запуска оказывает влияние на способ установки битов в регистре запросов IRR при появлении сигнала запроса на соответствующем входе irq.**

**Установленный бит 4 является идентификатором приказа инициализации, отличающим его от рабочих приказов, выводимых в тот же четный порт.**

**ICW2 определяет базовый адрес последовательности векторов прерываний, размещаемых в таблице векторов прерываний. Собственно, под базовый адрес отводятся старшие 5 битов приказа(3-7).**

**Для ведущего PIC базовый адрес инициализируется на значении 08h, для ведомого PIC на значение 70h. Младшие три бита определяются номером источника запроса и фиксируются с помощью шифратора приоритетов. Значение базового адреса, дополненное уровнем обслуживаемого запроса, и выставляется микросхемой PIC на внешнюю шину данных в цикле подтверждения прерывания. Фактически, это значение задает номер (тип) обрабатываемого прерывания, который модифицируется процессором в адрес вектора прерываний.**

**ICW3 определяет связи микросхем PIC при их каскадном включении. Для ведущего PIC установленные биты определяют к каким входам irq подключаются ведомые PIC. Сброшенное значение бита означает, что к соответствующему входу подключается ВУ либо этот вход не используется. Для ведомых PIC младшие три бита приказа являются кодом идентификации и задают номер линии запроса ведущего PIC, к которой подключается выход INT ведомого контроллера.**

**ICW4** **Наиболее важным битом приказа ICW4 является бит 1, именуемый AEOI – Automatic End Of Interrupt.**

**В обычном режиме работы (при сброшенном бите AEOI) процедура обработки аппаратного прерывания должна перед своим завершением очистить собственный бит в ISR специальной командой, иначе новые прерывания этого же типа не будут обрабатываться.**

**При использовании же режима AEOI (соответствующий бит установлен) бит, соответствующий виду запроса в ISR сбрасывается автоматически в тот момент, когда начинается обработка этого прерывания. Сложность работы в этом режиме обусловлена тем, что процедура обработки аппаратного прерывания должна обеспечивать свойства повторной входимости (реентерабельности), т. к. во время их работы могут повторно возникнуть запросы того же уровня.**

**После инициализации PIC готов к работе в заданном режиме, т.е. выполнять следующие действия:**
**1) Маскировать и размаскировать аппаратные прерывания.**
**2) Изменять приоритеты уровней.**
**3) Издавать команду завершения обработки аппаратного прерывания (AEOI).**
**4) Переводить контроллер в режим опроса и считывать состояния регистров ISR и IRR с помощью вывода в соответствующий порт одного из слов рабочих приказов.**

## Слова рабочих приказов.

**OCW1** **представляет собой маску запросов прерываний. Маскирование запросов осуществляется единичным значением соответствующего бита. Этот приказ при выводе в нечетный порт пересылается в регистр IMR.**

**OCW2** **предназначено для следующих целей:**

1) **вывод команды завершения обработки аппаратного прерывания (EOI – End Of Interrupt);**
2) **циклический сдвиг или явное изменение уровней приоритетов.**

**Структура:**

| A0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | R | SL | EOI | 0 | 0 | $L_2$ | $L_1$ | $L_0$ |

**R – Rotation – вращение приоритетов,**
**SL – Set Level – установка уровня приоритета,**
**EOI – End Of Interrupt – приказ конца прерывания,**
**$L_0$, $L_1$,$L_2$ - уровень приоритета (значение битов актуально только при SL=1).**

**Биты 3 и 4 – биты идентификации. Нулевое значение четвертого бита отличает рабочий приказ от приказа инициализации ICW1, выводимого в тот же четный порт. Нулевое значение третьего бита отличает рабочий приказ OCW2 от OCW3, для которого этот бит установлен.**

**Бит EOI непосредственным образом связан с аналогичным битом AEOI (1 бит в ICW4).Единичное значение бита AEOI означает автоматический конец прерывания и приводит к тому, что установленный запросом прерывания бит в регистре ISR автоматически сбрасывается в цикле подтверждения прерывания. При нулевом значении бита AEOI бит в регистре ISR необходимо сбрасывать специальным приказом конца прерывания. Выдача этого приказа заключается в передаче OCW2 с установленным битом EOI в порт с четным адресом (ведущего 20, ведомого A0).**

**Выдача этого приказа возлагается на программу обработчик прерывания и команды, связанные с этой выдачей размещаются в самом конце обработчика.**

**При выдаче приказа EOI комбинация битов R и SL определяют возможное изменение приоритетов запросов прерывания.**

| R | SL | Описание режимов |
|---|---|---|
| 0 | 0 | Режим обычных приоритетов. После сброса последнего установленного бита в регистре ISR, приоритеты запросов остаются стандартными, т. е. irq0 имеет наивысший приоритет, а irq7 – наинизший. |
| 0 | 1 | Специальный режим конца прерывания, в ISR сбрасывается бит с заданным в поле $L_0$, $L_1$,$L_2$ номером или уровнем. |
| 1 | 0 | После сброса бита наивысшего уровня производится циклическое изменение приоритетов, причем обслуживаемый уровень опускается на дно приоритетного кольца. |
| 1 | 1 | Т.е. сброс бита в ISR с заданным уровнем с изменением его приоритета на низший. |

**В принципе, биты R и SL являются актуальными и при нулевом значении бита EOI. Например, комбинация 10 предполагает разрешение вращения уровней приоритетов, причем изменение приоритетов происходит каждый раз при автоматическом сбросе бита запроса в ISR (AEOI = 1). Аналогично, комбинация 11 осуществляет принудительную установку дна приоритетного кольца при каждом сбросе бита в регистре ISR. Возврат к обычному режиму приоритетов осуществляется посылкой нулевого байта в порт с адресом 20.**

**OCW3 позволяет выполнить следующие действия:**

1) **установка и отмена так называемого режима специального маскирования;**
2) **установка и сброс режима опроса (полинга);**
3) **разрешение чтения регистров IRR и ISR контроллера.**

**Структура:**

| A0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | ESMM | SMM | 0 | 1 | P | RR | RIS |

**RIS – бит управления чтением регистров. 0–читается регистр IRR, 1 – ISR,**
**P – бит поллинга, разрешающего опрос,**
**RR – бит разрешения чтения регистров,**
**SMM – Special Mask Mode - режим специального маскирования,**
**ESMM – Enable Special Mask Mode – режим разрешения специального маскирования.**
**Установка битов RR, P, ESMM является взаимоисключающей.**

| ESMM | SMM | Описание режимов |
|---|---|---|
| 1 | 1 | Режим специального маскирования устанавливается битом ESMM и реализует следующие действия, в зависимости от бита SMM: осуществляется обработка незамаскированных запросов по мере их появления(порядок приоритетов игнорируется – FIFO(FCFS)). |
| 1 | 0 | Восстановление приоритетного обслуживания. |

**P=1 - Режим поллинга. В этом режиме предполагается, что процессор не воспринимает запросов прерываний по входу INTR, вход замаскирован (IF = 0).При**

**этом необходимо опрашивать наличие запросов в регистре IRR. По активному уровню сигнала чтения, поступающему в PIC либо от CPU, либо от контроллера шины, происходит установка соответствующего бита в ISR, как будто получен сигнал INTA и выдача в регистр AL процессора байта следующего формата:**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| I | X | X | X | X | $L_2$ | $L_1$ | $L_0$ |

**Старший бит I=1 показывает наличие незамаскированного запроса прерывания, а младшие три бита содержат уровень запроса наивысшего приоритета из поступивших и незамаскированных. Сигнал Read генерируется при выполнении команды IN.**

**При P=0, содержимое регистров IRR и ISR можно прочитать в регистр AL, используя команду IN, при предварительно установленном бите RR.**

## Организация центральных процессоров.

**CPU – Central Processing Unit.**

**ЦП выполняет в компьютере двоякую функцию:**

1) **Как обрабатывающее устройство: ЦП осуществляет выполнение программ, связанных с какой-либо обработкой данных.**
2) **Как управляющее устройство: ЦП осуществляет координацию остальных устройств компьютера, а также связь компьютера с внешним миром.**

**Чтобы подчеркнуть одну из основных функций CPU, в некоторых случаях в компьютерной литературе для его обозначения используется аббревиатура ISP – Instruction Set Processor ( процессор команд). ISP =CPU.**

**По мнению некоторых специалистов, термин «Central» в аббревиатуре CPU является устаревшим.**

**В состав CPU, как правило, входят следующие блоки:**

**Регистры, которые в свою очередь можно разделить на программно доступные и программно недоступные. Программно доступные регистры можно разделить на прикладные и системные.**

**ALU (Arithmetic and Logic Unit). В этом блоке осуществляется обработка целых чисел и логических значений. В связи с этим аббревиатуру ALU иногда заменяют IU(Integer Unit).**

**FPU (Floating Point Unit). Блок или устройство обработки чисел с плавающей запятой.**

**Блоки, ориентированные на так называемую мультимедийную обработку:**
**MMX (Multimedia extension),**
**SSE (Stream SIMD Extension - потоковое SIMD расширение),**
**SIMD (Single Instruction Multiple Date - одиночный поток команд, множественный поток данных).**

**Отметим, что основной особенностью блоков мультимедийной обработки является использование не скалярных, а векторных данных и, соответственно, векторных команд (одной векторной командой задается однотипная обработка для нескольких данных, трактуемых как элементы вектора).**

**CU (Control Unit - блок (устройство) управления).**

**Блоки, входящие в состав конвейера команд:**
**PFU (Prefetch Unit - блок предварительной выборки команд),**
**DU (Decode Unit- блок декодирования команд),**
**BTB (Branch Target Buffer - блок предсказания ветвлений).**

**Блок для связи CPU с системной шиной:**
**BIU (Bus Interface Unit - блок сопряжения с шиной**
**или BU – Bus Unit).**

**MMU – (Memory Management Unit - блок управления памятью).**

- **PU (Page Unit -блок управления страницами).**
- **SU (Segment Unit –блок управления сегментацией).**

**В некоторых источниках внутрикристалльную Кэш-память (КЭШ L1) включают также в состав CPU, что, по мнению Довгия П.С. является некорректным.**

**Связи между отдельными блоками (устройствами) CPU осуществляются с использованием внутренних шин. Примерами структур процессора i386, i486, Pentium, Pentium Pro (P6) см. раздаточный материал.**

## Принципы построения и функционирования конвейеров команд.

**Конвейеры команд следует рассматривать в качестве одного из актуальных структурных средств улучшения производительности центральных процессоров и, соответственно, всего компьютера вцелом.**

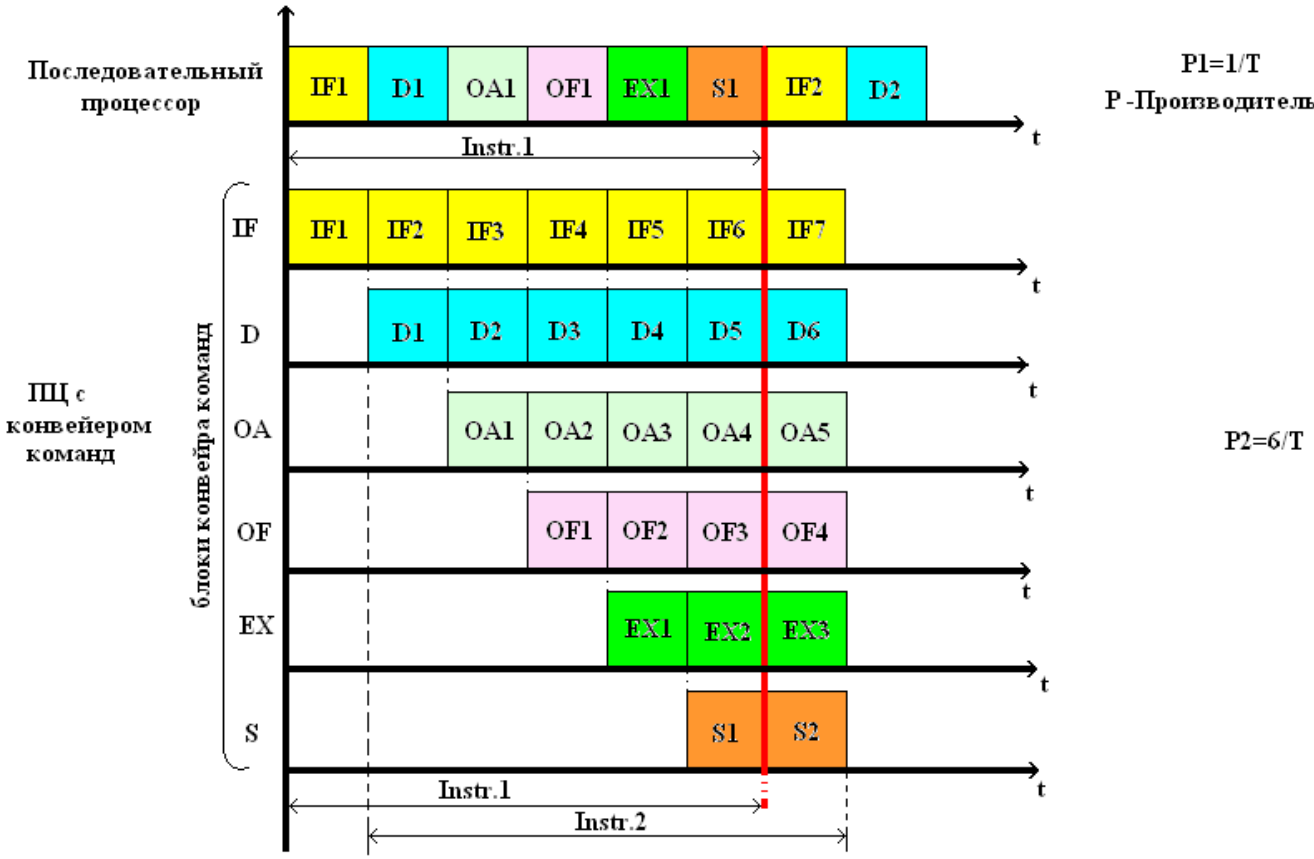
**Принципы построения конвейеров команд базируются, во-первых, на разделении, выполнения любой машинной команды на ряд последовательных этапов и, во-вторых, на разделении аппаратуры CPU на ряд независимых функциональных блоков, способных функционировать параллельно и выполнять один или несколько смежных этапов машинной команды.**

**При классическом подходе выполнение основных машинных команд (например, арифметических или логических), связанных с обработкой данных, можно разделить на 6 этапов (фаз, стадий):**

1. **IF (Instruction Fetch - выборка команды).**
2. **D (Decode - декодирование команды).**
3. **OA (Operand Address - формирование адресов операнда).**
4. **OF (Operand Fetch - выборка операнда).**
5. **EX (Executive - выполнение операции).**
6. **S (Store - запись результата).**

## Сравнение производительности последовательного процессора (без конвейера команд) и «параллельного» процессора (с конвейером команд).

**Конвейеризация на уровне машинных команд представляет собой низкоуровневый параллелизм на уровне машинных команд.**




**В идеальном случае производительность процессора с N - ступенчатым конвейером команд в N раз больше производительности последовательного процессора, т.е. без конвейера команд.**

**Сравнительная оценка конвейерного и без конвейерного процессора по производительности производилась для идеального случая, который включает в себя следующие допущения:**

1. **длительности всех фаз конвейера одинаковы;**
2. **обеспечивается постоянная загрузка вех блоков конвейера (без изменения порядка действия);**
3. **в конвейере отсутствуют так называемые сбои, связанные с выполнением команд переходов, т.е. работа конвейера рассматривается на достаточно длинном, линейном участке программы.**

**В реальности производительность конвейеров команд оказывается намного ниже идеального случая. К основным причинам снижения производительности конвейера команд принято относить:**

1) **Наличие в программе так называемых зависимостей по управлению (конфликты по управлению).**

**Эта зависимость проявляется в использовании команд, нарушающих естественный порядок следования команд программы. К таким командам относятся команды перехода (безусловные и условные), команды вызовов и возвратов, команды циклов, команды прерываний.**

**Конвейер команд осуществляет предварительную выборку команд из памяти, как правило, руководствуясь линейным порядком их следования (предварительная выборка осуществляется по последовательным адресам). Когда в конвейер попадает одна из перечисленных выше команд, то факт наличия перехода по тому или иному адресу распознается на фазе EX (исполнения).**

**Для команд с безусловным переходом (JMP, CALL, RET, INT, IRET) оказывается, что все последующие команды, накопленные к этому моменту, остановятся неактуальными. Инициализируется так называемый сброс конвейера, формально связанный с очисткой его ступней от последующих команд программы, который сопровождается реинициализацией конвейера, начиная с выборки команды по сформированному адресу перехода.**

**При выполнении команд условного перехода (команды типа JA, JB, JG,.., LOOP, JCXZ), проверка выполнения или невыполнения условия перехода осуществляется на фазе EX и реинициализация конвейера требуется только в том случае, если условие перехода выполняется, в противном случае работа конвейера продолжается в естественном порядке следования команд программы.**

2) **Наличие в программах зависимостей по данным (конфликты по данным).**

**Зависимость по данным проявляется при использовании некоторой программы некоторой команды результата предыдущей команды в качестве операнда или адреса операнда. При этом имеет место следующая ситуация в конвейере: когда зависимая команда доходит до фазы выборки операнда, предыдущая команда еще не успела сформировать и записать в память свой результат.**

**Подобная ситуация не приводят к сбою конвейера, а приводят лишь к его приостановке.**

3) **Использование различными блоками конвейера одного и того же ресурса (структурные конфликты).**

**Как правило, в роли разделяемого ресурса фигурирует память. Обращения к памяти могут возникать в блоках конвейера IF, OF, S. Учитывая факт использования Кэш-памяти и более того, разделение внутренней Кэш-памяти на два независимых блока (Кэш-команд и Кэш-данных), принято считать, что структурные конфликты снижают реальную производительность конвейера команд весьма незначительно.**

4) **Наличие при выполнении программы особых случаев, приводящих к прерыванию.**

**Возникновение прерывания приводит к сбросу конвейера. Организация прерываний с учетом конвейерной обработки команд имеет ряд дополнительных нюансов, связанных с наличием нескольких машинных команд в процессоре, находящихся на различных фазах обработки в момент асинхронного прерывания. Основная проблема в том, адрес какой машинной команды сохраняется в качестве возврата и что делать с недовыполненными командами.**

5) **Различное время выполнения отдельных фаз машинных команд.**

**Наиболее трудоёмкой фазой является фаза EX для так называемых длинных команд типа умножение, деление.**

6) **Большой разброс длительности фазы EX для различных машинных команд.**

## Основные действия, выполняемые процессором на различных фазах (этапах) команды.

### IF - выборка команды.

**С позиции простейшего классического процессора при реализации этой фазы используются следующие регистры:**

**PC – Program Counter;**
**IR - Instruction Register**
**MAR – Memory Address Register;**
**MBR - Memory Buffer Register.**

**Последние два регистра обеспечивают интерфейс процессора с основной памятью.**

**Простейший классический процессор не имеет следующих средств:**

- **Кэш-памяти;**
- **средств преобразования программного адреса в физический;**
- **конвейера команд;**
- **команды и данные имеют единый формат, совпадающий с разрядностью шины данных (шириной выборки из ОП);**

**Этап выборки команды состоит в извлечении текущей команды из памяти с последующей ее пересылкой в регистр команд (IR). Адрес, по которому осуществляется выборка команды, находится в счетчике команд. Как правило, в этот же этап включается модификация счетчика команд (PC) путем увеличения на длину выбираемой команды.**

**Выборку команды можно представить последовательностью действий вида:**

**1. PC → MAR;**
**2. RDM: ОП [MAR] → MBR; PC + 1 → PC;**
**3. MBR → IR;**

**Комментарии:**

**Использование механизмов сегментации и страничного преобразования создает необходимость предварительного преобразования программного адреса команды, который является частью логического, в физический адрес.**

**На этапе сегментации логический адрес команды, представляющий пару CS:IP с использованием дескрипторных таблиц сегментов (GDT-Global Descriptor Table, и возможно LDT- Local Descriptor Table) преобразуется в так называемый линейный адрес.**

**На этапе страничного преобразования линейный адрес преобразуется в физический адрес с использованием таблиц двух уровней:**

- **первый уровень – каталог таблиц страниц (PD – Page Directory);**
- **второй уровень – таблицы страниц (PT – Page Table).**

**Для ускорения преобразования адреса из логического в физический используются специальные аппаратные средства в виде теневых регистров (Shadow Register) на этапе сегментации и буфера ассоциативного преобразования. TLB (Translate Look aside Buffer) на этапе пейджинг (paging).**

**Теневые регистры представляют собой аппаратные расширения (64-разрядные) сегментных регистров недоступные программам. В эти расширения помещаются дескриптор соответствующего сегмента при любом изменении**

**содержимого сегментного регистра. Тем самым предотвращается задержка, связанная с обращением к дескрипторным таблицам, находящимся в памяти. Фактически, обращение к таблицам производится только один раз при перезагрузке сегментного регистра. Достаточного часто теневые регистры называют Кэш-регистрами дескрипторов, в связи с тем, что их использование может рассматривать как один из способов кэширования.**

**TLB – Translate Look aside Buffer представляет собой небольшой блок ассоциативной памяти, в котором содержится информация о страницах, которые использовал процессор последнее время.**

**Отметим, что основным отличием ассоциативной памяти от классической адресной памяти является принцип обращения не по адресу, а по признаку (тегу, ассоциации). Обычно в ассоциативной памяти выделяют два блока: блок признаков (ключей) и блок данных.**

**При обращении к ассоциативной памяти на ее вход поступает тег запрашиваемых данных. Входной тег сравнивается параллельно со всеми тегами из памяти тегов. При обнаружении совпадений осуществляется выборка данных, соответствующих совпавшему тегу из памяти данных, при этом формируется сигнал попадания. При отсутствии информации с заданным тегом формируется сигнал промаха. Принципы организации TLB точно такие же, как и внутренней Кэш-памяти, в соответствии с чем, структурно эти блоки включаются в кэш-память.**

**Так как TLB используется для преобразования линейного адреса в физический, то тегом для поиска TLB служит линейный адрес, точнее, 20 старших его разрядов. В свою очередь в памяти данных TLB содержатся физические адреса наиболее часто используемых страниц, точнее 20 их старших разрядов. Кроме физического адреса в памяти данных TLB содержатся также некоторые атрибуты страниц, связанные с их защитой и возможностью кэширования. Младшие 12 разрядов линейного адреса являются смещением внутри страницы и не подлежат преобразованию, то есть прямо копируются в младшие разряды физического адреса.**

**В физическом пространстве памяти, впрочем как и в линейном, страницы выравниваются на 4-х Кб границу из этого следует, что базовый или начальный адрес страницы содержит 12 младших нулей.**

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

# Конспект лекций по курсу «Организация ЭВМ и ВС»

Санкт-Петербург

2012

# Организация прерываний

## Концепции системы прерываний

По образному выражению П. Нортона: «Прерывание – это движущая сила компьютера». В связи с этим, прерывания следует рассматривать не только и не столько как реакцию процессора на аномальные явления, а как естественный процесс, с помощью которого реализуется поддержка большинства механизмов, таких как ввод / вывод, виртуальная память, мультизадачность.

Система прерываний является неотъемлемой частью любого компьютера и предназначена для быстрой реакции процессора на ряд ситуаций,

требующих его внимания, которые могут возникать как при выполнении программы, так и при работе аппаратуры.

Система прерываний представляет собой комплекс аппаратных и программных средств.

Аппаратные средства системы прерываний либо входят в состав CPU и называются тогда ***блоком прерываний***, либо реализуются в виде отдельного устройства, называемого ***контроллером прерываний***. Первый подход является типичным для больших ЭВМ класса Mainframe, второй – для компьютеров на базе микропроцессоров.

Программные средства системы прерываний представляют собой так называемые ***обработчики прерываний***, которые входят в состав операционной системы.

В первом приближении прерывания разделяют на два больших класса: 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, т.е. являются асинхронными по отношению к выполняемой программе. С помощью аппаратных прерываний осуществляется взаимодействие процессора с периферийными устройствами, а также сообщается о различных аппаратных ошибках.

## Основные причины, приводящие к прерыванию программы

1. Особые случаи, возникающие при выполнении программы. К ним относятся:

**а)** ошибки, возникающие при выполнении арифметических операций:
- переполнение при сложении / вычитании целых чисел,
- переполнение или исчезновение порядка при выполнении арифметических операций над числами с плавающей запятой,
- некорректность деления в операции деления целых чисел (частное не помещается в формате делителя);

**б)** различные некорректности, имеющие место в машинных командах:
- некорректный код операции,

- некорректный адрес операнда или команды, в частности, нарушение целочисленной границы,
- некорректные данные ( например, в качестве десятичной цифры используются комбинации 1010÷1111);

**в)** особые случаи, связанные с защитой памяти:

- попытка обращения к данным по записи при их доступности только для чтения,
- выход обращения за пределы заданного диапазона (нарушение границы сегмента);

**г)** особые случаи, связанные с организацией виртуальной памяти, например: неприсутствие сегмента или страницы;

**д)** выполнение специальных команд, имитирующих прерывание, для вызова различных служебных функций операционной системы.

Для процессоров Intel такой командой является команда INT n, где n – байтный тип прерываний. Классическим примером является команда INT 21h, используемая в реальном режиме для вызова функций DOS. В вычислительных системах IBM/370 подобную функцию выполняет команда SVC – SuperVisor Call.

2. Запросы прерываний от внешних (периферийных) устройств. Эти запросы могут иметь место в следующих случаях:

а) ВУ, готовое к обмену, требует реакции процессора на организацию передачи данных;

б) завершение работы ВУ по передаче какой-либо порции данных (например, блока);

в) особая (аварийная) ситуация в ВУ, например, нарушение контроля передаваемых данных.

3. Сбои аппаратуры, обнаруживаемые встроенными средствами аппаратного контроля, например, ошибка памяти.

## Функции системы прерываний и их реализация на аппаратном и программном уровнях

Функции системы прерываний:

**1)** прием и хранение запросов прерываний от многих источников;

**2)** выделение наиболее приоритетного запроса из множества поступивших;
**3)** проверка возможности обработки выделенного запроса центральным процессором;
**4)** сохранение состояния (контекста) прерываемой программы;
**5**) вызов соответствующего обработчика прерываний;
**6)** обработка прерывания (выполнение программы-обработчика);
**7)** восстановление состояния (контекста) прерванной программы и возобновление ее выполнения.

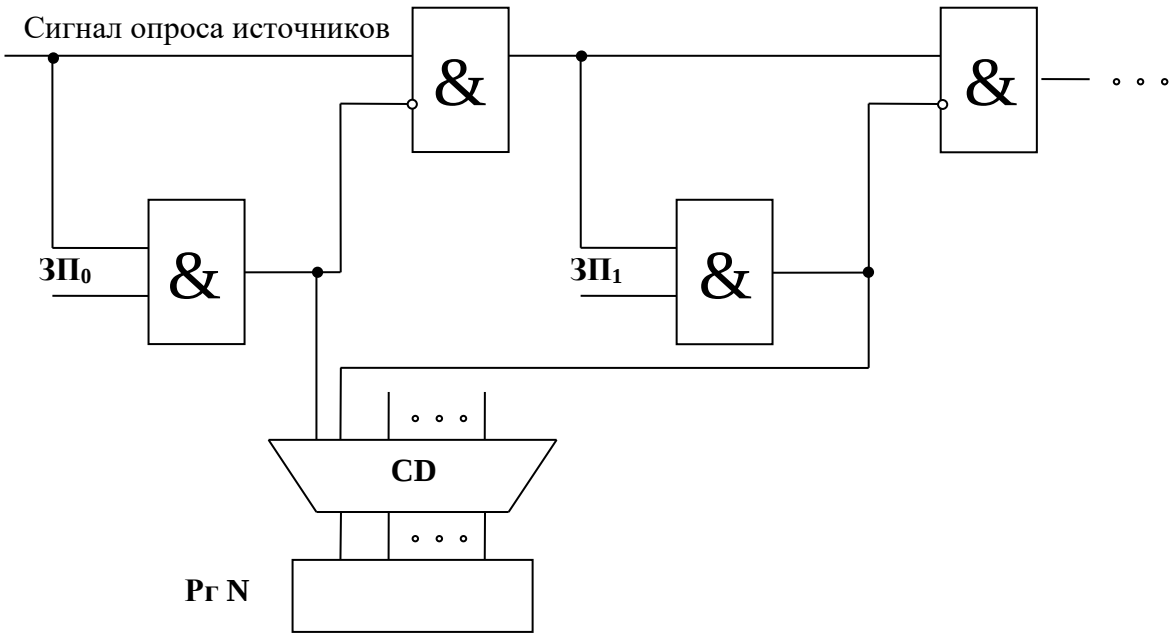
## Реализация функций на аппаратном и программном уровнях

*1. Прием и хранение запросов прерываний от многих источников.*

Эта функция реализуется на аппаратном уровне путем установки соответствующих битов специального регистра запросов при появлении этих запросов. Например, в микросхеме PIC (Intel 8259A) имеется специальный регистр IRR (Interrupt Request Register – регистр запросов прерываний), который является восьмибитным (по числу обслуживаемых запросов). Каждый бит этого регистра соответствует определенному источнику прерываний (например, запрос от таймера или клавиатуры), и установка этого бита свидетельствует о наличии запроса от источника. При количестве источников запросов больше восьми применяется так называемая ***схема каскадного подключения микросхем PIC***. В типовой комплект ПК входят две микросхемы PIC, одна из них называется ведущей, а другая – ведомой.

*2. Выделение наиболее приоритетного запроса из множества поступивших.*

Процедура опроса источников прерываний с целью выделения наиболее приоритетного из них обычно называется ***полинг (polling)***. В принципе, эта процедура может быть реализована как на аппаратном, так и на программном уровнях. Аппаратная реализация полинга, как правило, осуществляется с помощью цепочной однотактной схемы, именуемой ***дейзи-цепочкой***.




ЗП – запросы прерываний от источников 0,1,2,…
Наиболее приоритетным является $ЗП_0$
CD – кодер, преобразующий унитарный код запроса в позиционный;
Позиционный код запроса сохраняется в Рг N.

Программный полинг реализуется специальной программой, которая последовательно опрашивает разряды регистра запросов с целью выделения крайней левой или крайней правой единицы (в зависимости от упорядочивания запросов по приоритетам).

Для ускорения работы программы полинга могут быть использованы специальные команды сканирования битов с мнемоникой BSF – Bit Scan Forward (прямое сканирование) или BSR – Bit Scan Reverse (обратное сканирование), с помощью которых можно выделить крайний левый (BSF) или крайний правый (BSR) бит, установленный в операнде-источнике. Эти команды введены в систему команд процессоров Intel, начиная с Intel 80386, и возвращают в качестве результатов номер позиции бита.

В стандартной микросхеме PIC встроен механизм аппаратного полинга на основе дейзи-цепочки, но имеется возможность реализации и программного полинга.

*3. Проверка возможности обработки выделенного запроса центральным процессором*

Отношение ЦП к поступающим запросам прерываний выражается с помощью двух основных механизмов:

- механизм масок;
- механизм порога.

***Механизм масок*** используется в ПК на базе процессоров Intel, а также в мэйнфреймах фирмы IBM.

***Механизм порога*** используется в мини-компьютерах с архитектурой DEC – Digital Equipment Corporation (PDP-8, VAX-11), а также в ПК на базе процессоров Motorola.

Механизм масок основан на использовании специального бита для каждого запроса прерывания, с помощью которого разрешается или запрещается обработка этого запроса. Как правило, единичное значение бита маски определяет разрешение обработки (прерывание не замаскировано), а нулевое значение – запрещение обработки (прерывание замаскировано). В принципе, возможен и обратный подход.

В микросхеме PIC имеется соответствующий регистр IMR – Interrupt Mask Register, в котором маскирование запросов осуществляется единичным значением бита Mask.

Дальнейшим развитием механизма Mask является использование иерархического подхода к маскированию запросов прерываний. В качестве примеров иерархии масок, используемых в процессорах фирмы Intel, могут являться:

• Маскирование внешних запросов. Локальные маски для каждого из запросов сосредоточены в регистре IMR микросхемы PIC. Глобальная маска представляет собой флаг IF. При установленном флаге разрешается обработка внешних прерываний, при сброшенном – запрещается. С помощью флага IF маскируются только те запросы, которые поступают на вход INTR процессоров (Interrupt Request). В свою очередь запросы, поступающие на внешний вход NMI (Non-Maskable Interrupt), принимаются к обслуживанию независимо от состояния флага IF.

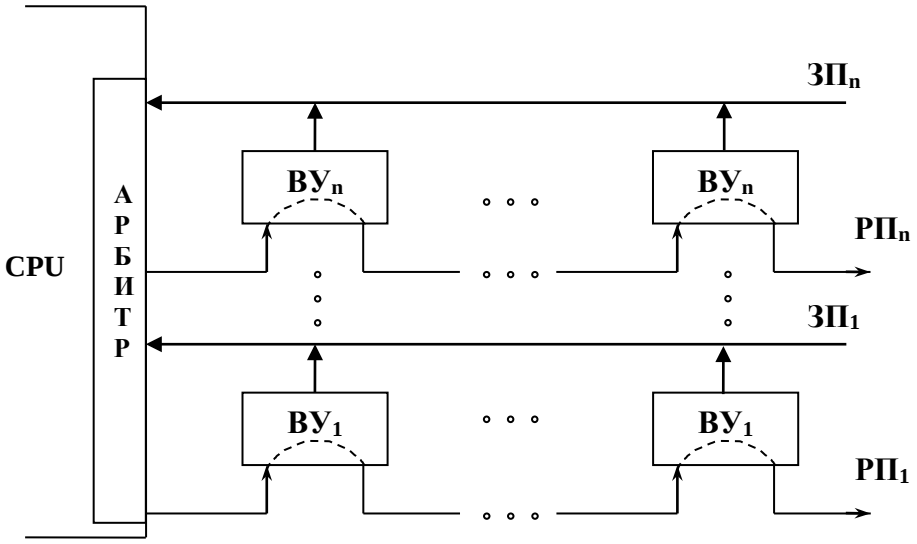
• В качестве другого примера можно привести маскирование особых случаев в FPU. В управляющем регистре CR (Control Register) FPU крайние правые 6 бит являются масками особых случаев, к которым относятся:

- недействительная операция;
- денормализованный операнд;
- деление на ноль;
- переполнение порядка;
- исчезновение порядка (антипереполнение);
- потеря точности.

Кроме того, в этом же регистре имеется глобальная маска IEM, с помощью которой маскируются все особые случаи.

Порог прерываний представляет собой собственный приоритет процессора, точнее, уровень приоритета выполняемой им программы. Порог отражается с помощью специального трехбитного поля, находящегося в слове состояний процессора PS (Processor Status).

В интерфейсе Unibus (общая шина), используемом в компьютерах с архитектурой DEC, выделяются специальные линии запросов прерываний от ВУ и линий разрешения прерываний, которые являются однонаправленными. Упрощенная схема подключения к этим линиям имеет вид:




Все ВУ, в зависимости от их важности (приоритета), подключаются параллельно к соответствующей линии запроса прерываний $ЗП_1 \ldots ЗП_n$. Предполагается, что приоритет увеличивается с увеличением номера.

В свою очередь, линии разрешения прерываний проходят последовательно через ВУ каждого уровня, что соответствует так называемому ***цепочному интерфейсу***. В CPU имеется специальный блок, называемый ***арбитром***, который выделяет линию с наиболее приоритетным запросом. Если приоритет этой линии выше порога прерываний, то арбитр посылает сигнал разрешения прерывания по соответствующей линии этого уровня. Сигнал разрешения последовательно проходит через ВУ этого уровня и блокируется первым же ВУ, пославшим запрос на линию ЗП. В соответствии с этим, в подобной схеме подключения ВУ реализуется двумерная система приоритетов. Это означает, что приоритет ВУ во-первых зависит от уровня линий ЗП и РП, к которым оно подключается, во-вторых – от степени его электрической близости по линии РП к арбитру.

Функция проверки возможности обработки выделенного запроса центральным процессором реализуется чисто на аппаратном уровне.

*4. Сохранение состояния (контекста) прерываемой программы*

Эта функция обычно реализуется с использованием как аппаратного, так и программного уровней. На аппаратном уровне сохраняется лишь

минимальная часть контекста, в частности: обязательный адрес возврата и не очень обязательный регистр состояний (флагов). Содержимое остальных регистров процессора, которые могут быть востребованы программой-обработчиком прерываний, сохраняются на программном уровне. Действия, связанные с сохранением этих регистров, составляют начальную фазу программы-обработчика прерываний.

Применительно к процессорам фирмы Intel, исключительно удобной для этих целей (сохранение контекстов) является команда PUSHA, по которой сохраняются в стеке все РОНы (8 штук). В процессорах фирмы Intel на аппаратном уровне происходит сохранение в стеке содержимого регистра флагов FR, сегмента кода CS и IP. Последняя пара и представляет собой полный адрес возврата.

В тех случаях, когда выход на обработку прерываний сопровождается переключением задач, сохранение всего контекста прерываемой программы (задачи) реализуется на аппаратном уровне с использованием специального системного сегмента TSS – Task State (Status) Segment.

*5. Вызов соответствующего обработчика прерываний*

Эта функция реализуется чисто на аппаратном уровне и предполагает загрузку начального адреса обработчика, обычно называемого вектором прерываний, в соответствующие регистры процессора (для процессоров Intel это регистры CS и IP).

*6. Обработка прерывания (выполнение программы-обработчика)*

Эта функция реализуется на программном уровне.

*7. Восстановление состояния (контекста) прерванной программы и возобновление ее выполнения*

Эта функция является обратной функции сохранения состояния (контекста) прерываемой программы.

## Программируемый контроллер прерываний PIC (микросхема Intel 8086)

Одна микросхема PIC может обслуживать 8 запросов прерываний. В современных компьютерах на базе процессоров Intel используются две микросхемы PIC, объединенные с помощью так называемого каскадного включения, что позволяет, в принципе, обслужить до 15 источников прерываний. В реальном подключении их меньше. Одна из микросхем PIC является ведущей, а другая – ведомой. Ведущий PIC непосредственно связан с CPU, а ведомый PIC – только с ведущим.

## Основные функции PIC

1) Фиксация запросов прерываний, поступающих в PIC от ВУ в специальном регистре запросов.

2) Осуществление внутреннего маскирования запросов с помощью специального регистра маски IMR (нулевое значение бита маски является разрешением запроса, а единичное – запретом).

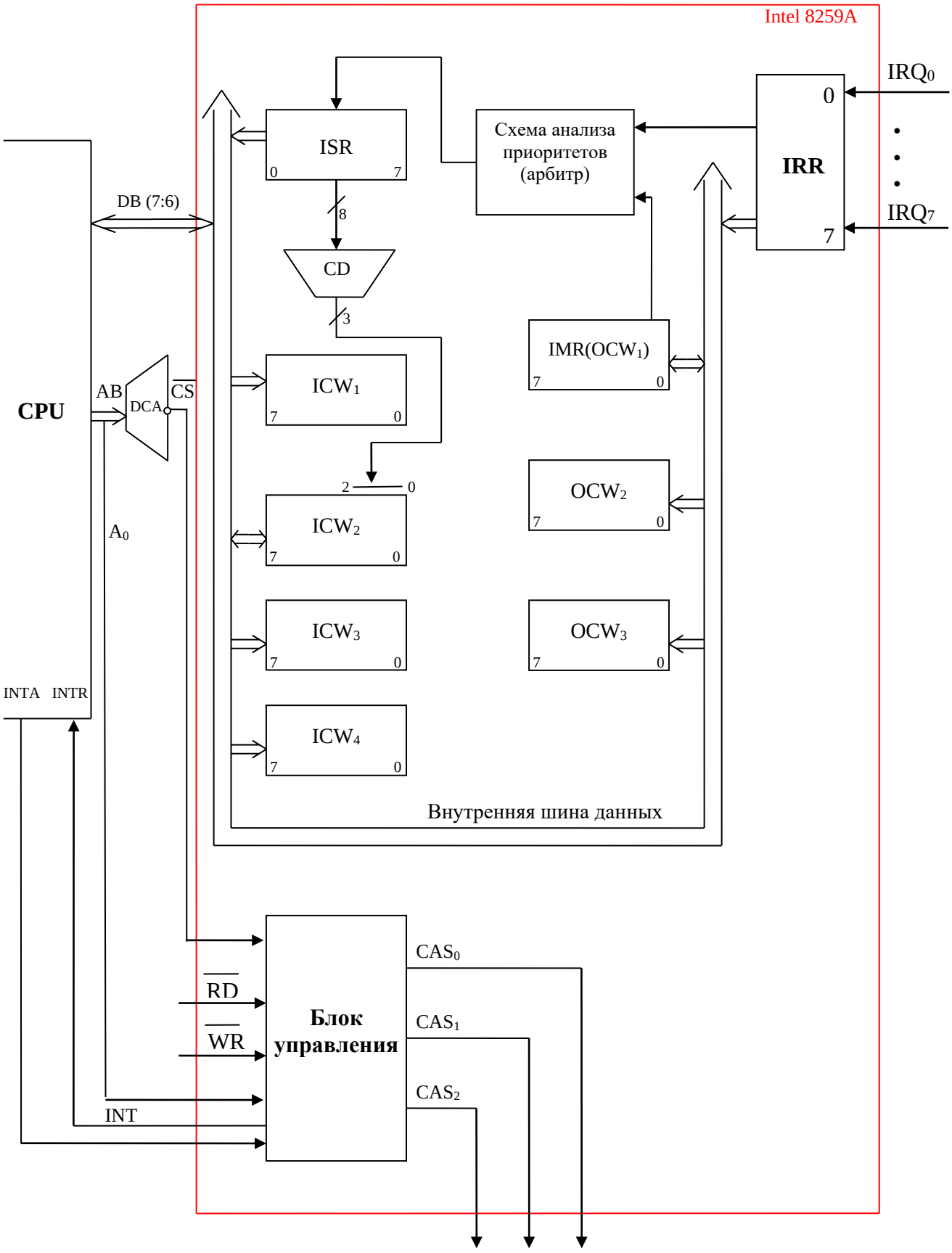
3) Выделение наиболее приоритетного запроса из всех поступивших и не замаскированных.

4) Выдача в CPU сигнала о наличии хотя бы одного незамаскированного запроса прерывания.

5) Выдача в CPU номера (кода) запроса в цикле подтверждения прерывания, который, в свою очередь, модифицируется CPU в адрес вектора соответствующего прерывания (начальный адрес обработчика прерываний для выделенного ВУ).

6) Возможность изменения приоритетов запросов прерываний.

## Упрощенная структурная схема PIC




## Описание схемы

Структура PIC включает в себя следующие байтные регистры:

• IRR – регистр запросов прерываний - связан с внешними входами запросов ($IRQ_0$ – $IRQ_7$);
• IMR – регистр маски запросов;
• ISR – Interrupt Service Register – регистр обслуживаемых запросов;
• $ICW_1$-$ICW_3$ – Initialization Control Word – управляющее слово инициализации (приказы инициализации);
• $OCW_1$-$OCW_3$ - Operation Control Word – операционное управляющее слово (рабочие приказы);
$OCW_1$ = IMR

Кроме регистров в состав PIC входят: блок управления и схема анализа приоритетов (арбитр).

Назначением блока управления является выработка внутренних и внешних сигналов управления, с помощью которых осуществляются те или иные элементарные действия (микрооперации) внутри микросхемы. Например, запись байта из внешней шины данных в один из регистров контроллера.

Сигналы $CAS_0$-$CAS_2$ используются для реализации каскадирования микросхемы.

Входной сигнал CS (Chip Select – выбор кристалла) генерируется в том случае, если на внешней шине адреса (AB) зафиксированы адреса, относящиеся к контроллеру прерываний. Программирование контроллера осуществляется по стандартным адресам портов ввода / вывода.

## Основные режимы работы PIC

1. **FNM** (Fully Nested Mode) – режим вложенных прерываний (фиксированных приоритетов). В этом режиме высшим приоритетом обладает запрос, поступающий на вход $IRQ_0$, низшим – запрос, поступающий на вход $IRQ_7$. В этом режиме допускается прерывание в прерывании, т.е. поступление на вход PIC запроса с более высоким приоритетом, чем обрабатываемый, вызывает генерацию активного уровня выходного сигнала INT (поступает на вход INTR CPU). Если процедура обработки прерывания предусматривает программную установку флага IF (при выходе на обработку прерывания этот флаг автоматически сбрасывается), то тем самым разрешается обработка более приоритетного запроса, прерывающая обработку менее приоритетного. При этом режиме в регистре ISR могут иметь место несколько установленных битов. Основной недостаток этого режима – сильная дискриминация запросов с низким уровнем приоритета.

2. **ARM** (Automatic Rotation Mode) – режим автоматического сдвига приоритетов. В этом режиме приоритеты запросов прерываний линейно

упорядочены и изменяются после обработки очередного запроса таким образом, что уровень обработанного запроса становится низшим, а следующий за ним уровень – высшим. Так, например, после обработки запроса $IRQ_3$ линейка приоритетов примет следующий вид:

$IRQ_4, IRQ_5, \ldots\ldots\ldots\ldots\ldots IRQ_3$

приоритет

3. **SRM** (Specific Rotation Mode) – режим адресуемых (программно-управляемых) приоритетов. В этом режиме уровень запроса наивысшего приоритета устанавливается извне путем передачи соответствующего приказа из CPU в PIC.

4. **PM** (Polling Mode) – режим опроса (полинга). С помощью этого режима реализуется программный полинг. В этом режиме PIC лишь фиксирует поступающие запросы в IRR и не посылает внешнего сигнала INT на вход CPU. Анализ содержимого IRR, а также регистра маски IMR, осуществляется программным путем путем предварительной пересылки содержимого этих регистров в CPU с помощью команды ввода INT с указанием адреса соответствующего порта ввода/вывода.

## Порядок взаимодействия между CPU и PIC (ведущим контроллером)

1. При наличии хотя бы одного незамаскированного запроса прерываний PIC выставляет активный уровень выходного сигнала INT, который поступает на вход INTR CPU.

2. CPU завершает текущую команду программы и проверяет состояние внешних входов, в том числе и INTR.

3. Если флаг IF установлен (внешние прерывания от PIC разрешены), процессор генерирует активный уровень выходного сигнала INTA (INTerrupt Answer – подтверждение прерывания). При сброшенном флаге IF обработка внешнего запроса прерывания временно откладывается (в частности, до выполнения процессором специальной команды STI – SeT Interrupt – разрешение прерывания, действие которой сводится к установке флага IF).

4. При получении сигнала INTA PIC выполняет следующие действия:

а) сбрасывает бит запроса, принятого к обслуживанию в IRR;
б) устанавливает бит обрабатываемого запроса в регистре ISR;
в) выставляет на внешнюю шину данных (точнее, в младший ее байт) номер (тип) обрабатываемого запроса.

5. CPU принимает номер запроса от PIC по шине данных и модифицирует этот номер в адрес соответствующего вектора прерываний (модификация номера в адрес осуществляется путем умножения номера на 4).

6. Текущее значение регистра флагов, сегмента кода (CS) и счетчика команд (IP) помещаются в стек, и тем самым сохраняется минимальный контекст прерываемой программы.

7. Два последовательных слова из таблицы векторов прерываний загружаются в регистр IP (слово по меньшему адресу) и CS (слово по большему адресу), тем самым настраивая CPU на выполнение первой команды программы-обработчика прерываний.

8. На аппаратном уровне производится сброс флага IF в целях временного запрещения поступления других запросов от PIC.

9. Процессор переходит к выполнению программы-обработчика соответствующего прерывания.

Описанная выше последовательность является типичной для базовой модели процессора Intel 8086, либо для старших моделей, функционирующих в R-режиме (реальном режиме). Процессор определяет, в каком режиме он функционирует, с помощью специального бита PE.

## Основы программной инициализации и изменения режимов работы PIC

Программный доступ к PIC осуществляется с помощью специальных команд ввода/вывода INT/OUT, адресующих порты ввода/вывода, зарезервированные за PIC. Доступ к ведущему PIC осуществляется с помощью двух портов с адресами 20h и 21h. Доступ к ведомому PIC осуществляется с помощью портов с адресами 0A0h и 0A1h.

Использование всего пары адресов для программного взаимодействия на большое число программно-доступных регистров PIC объясняется во-первых строгим заданием порядка следования слов при инициализации, во-вторых – использованием специальных битов идентификации, с помощью которых различаются слова инициализации и рабочие приказы.

Слова приказов инициализации устанавливаются общей процедурой инициализации при включении компьютера и в дальнейшем не изменяются. Слова рабочих приказов используются для динамического управления обработкой прерываний и могут изменяться в ходе работы компьютера. В порт с четным адресом выводятся слова $ICW_1$, $OCW_2$, $OCW_3$. В порт с нечетным адресом выводятся остальные слова приказов.

Инициализация PIC начинается выводом в порт с четным адресом приказа $ICW_1$, далее контроллер принимает приказ $ICW_2$ в порт с нечетным адресом. Необходимость ввода последующих приказов инициализации определяется единичными значениями соответствующих бит приказа $ICW_1$.

Для стандартных схем ПК на базе процессоров Intel процесс инициализации включает в себя вывод всех четырех слов.

## Основные функции и назначения основных битов приказов инициализации

**$ICW_1$** определяет особенности последовательности приказов инициализации. Два специальных бита определяют, будут ли присутствовать слова $ICW_3$ и $ICW_4$ в последовательности приказов. Один из битов определяет режим запуска по входам $IRQ_0$-$IRQ_7$. Режим запуска задает способ установки битов в регистре IRR при появлении запроса на соответствующем входе $IRQ_i$. При сброшенном бите запуск осуществляется по фронту сигналов, при установленном бите – по уровню сигналов.

**$ICW_2$** – содержимое этого регистра задает базовый адрес последовательности векторов прерываний, размещаемых в таблице векторов. Собственно под базовый адрес отводятся пять старших бит приказов, младшие три бита определяются номером источника запроса и фиксируются с помощью шифратора приоритета (см. схему). Для ведущего PIC базовый адрес инициализируется на значение 08h, для ведомого PIC – на значение 70h. Значение базового адреса дополняется уровнем обслуживаемого запроса и выставляется микросхемой PIC на внешнюю шину данных в цикле подтверждения прерывания. Фактически, содержимое регистра $ICW_2$ и является номером (типом) обрабатываемого прерывания.

**$ICW_3$** – определяет связи микросхем PIC при их каскадном включении. Для ведущего PIC установленные биты определяют, к каким входам IRQ подключаются ведомые контроллеры. В свою очередь, сброшенное значение бита для ведущего PIC означает, что к соответствующему входу подключается запрос от ВУ, либо этот вход вообще не используется. Для ведомых PIC младшие три бита приказа являются кодом идентификации и задают номер линии запроса ведущего PIC, к которой подключается выход INT ведомого контроллера.

**$ICW_4$** – наиболее важным битом этого приказа является бит, именуемый AEOI – Automatic End Of Interrupt (автоматический конец прерывания). Установленный бит задает соответствующий режим автоматического конца прерывания. В этом режиме выделенный бит обслуживаемого запроса в регистре ISR автоматически сбрасывается в тот момент, когда начинается

обработка соответствующего этому биту прерывания. Следствием этого является возможность приема к обслуживанию запроса того же типа до окончания обработки предыдущего запроса. Сложность работы в этом режиме обусловлена тем, что процедура обработки должна обеспечивать свойство реентерабельности (повторной входимости).

После инициализации PIC готов к работе в заданном режиме, т.е. способен выполнять следующие действия:

1) маскировать или размаскировать аппаратные прерывания;
2) изменять приоритеты уровней;
3) издавать команду завершения обработки аппаратного прерывания (AEOI);
4) переводить контроллер в режим опроса и считывать состояния регистров IRR и ISR с помощью вывода в соответствующий порт одного из слов рабочих приказов.

## Слова рабочих приказов

**$OCW_1$** – это слово представляет собой маску запросов прерываний. Маскирование (запрещение) запросов осуществляется единичным значением соответствующего бита маски. Этот приказ при выводе в нечетный порт пересылается в регистр IMR.

**$OCW_2$** – этот приказ предназначен для выполнения следующих функций:

1) вывод команды завершения обработки аппаратного прерывания (EOI);
2) циклический сдвиг или явное изменение уровней приоритетов.

Формат $OCW_2$:

| **R** | **SL** | **EOI** | **0** | **0** | **$L_2$** | **$L_1$** | **$L_0$** |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

R – Rotation – вращение приоритетов;
SL – Set Level – установка уровней приоритетов;
EOI – End Of Interrupt – приказ конца прерывания;
$L_2$,$L_1$,$L_0$ – уровень приоритета (это поле является актуальным только при SL=1)
Биты 3 и 4 являются битами идентификации $OCW_2$.

Бит EOI непосредственным образом связан с аналогичным по смыслу битом AEOI (Automatic EOI). Единичное значение бита AEOI означает

автоматический конец прерывания, что приводит к тому, что установленный запросом прерывания бит в регистре ISR автоматически сбрасывается в цикле подтверждения прерывания.

При нулевом значении бита AEOI установленный в регистре ISR бит, соответствующий обслуживаемому запросу прерывания, необходимо сбрасывать специальным приказом конца прерывания. Выдача этого приказа возлагается на программу-обработчик прерываний, и команды, связанные с этой выдачей, размещаются в самом конце программы-обработчика (как правило, непосредственно перед командой возврата IRET). Для ведущего PIC выдача этого приказа осуществляется командами:

```
MOV AL, 20H; пересылка приказа EOI в регистр AL
OUT 20H, AL; вывод приказа EOI в PIC
```

Для ведомого контроллера используется команда OUT 0A0H, AL.

После вывода этого приказа PIC работает в режиме обычных приоритетов (соответствует описанному выше режиму FNM).

**R SL**

**0 0** - Режим обычных приоритетов (FNM).

**0 1** - В регистре ISR производится сброс бита, соответствующего заданному в приказе уровню приоритета, после чего устанавливается обычный режим приоритетов.

**1 0** - После сброса в ISR бита, соответствующего запросу с наивысшим приоритетом, запросы этого уровня опускаются на дно приоритетного кольца (им присваивается низший приоритет), в то время как наивысший приоритет переходит к следующему по порядку уровню. Так, например, после запроса с уровнем $IRQ_3$ линейка запросов принимает вид:

$IRQ_4, IRQ_5, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots IRQ_1, IRQ_2, IRQ_3$



приоритет

**1 1** - Сброс в ISR бита, соответствующего запросу с заданным в приказе уровнем, после чего этому уровню присваивается наинизший приоритет. Наивысший приоритет получает следующий по порядку уровень.

В принципе, биты R и SL являются актуальными и при нулевом значении бита EOI, т.е., не посылая приказа конца прерывания, можно, тем не менее, реализовать изменение приоритетов. Так, например, комбинация 10 предполагает разрешение вращения уровней приоритетов, причем изменение приоритетов происходит каждый раз при автоматическом сбросе бита запроса в ISR (AEOI=1).

Аналогично комбинация 11 осуществляет принудительную установку дна приоритетного кольца так же при каждом сбросе бита в регистре ISR.

Возврат к обычному режиму приоритетов при условии, что AEOI=1, осуществляется выводом нулевого байта в порт с адресом 20.

**$OCW_3$** – с помощью этого приказа могут быть реализованы следующие функции:

1) установка и отмена режима специального маскирования;
2) установка и отмена режима опроса (полинга);
3) разрешение чтения регистров IRR и ISR контроллера.

Формат $OCW_3$:

| **0** | **ESMM** | **SMM** | **0** | **1** | **P** | **RR** | **RIS** |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

ESMM (Enable Special Mask Mode) – разрешение специального режима маскирования;
SMM (Special Mask Mode) – специальный режим маскирования;
P (Polling) - бит разрешения полинга;
RR – Read Register – разрешение чтения регистров; RIS (“0” – чтение регистра IRR, “1” – чтение регистра ISR);
Биты 3 и 4 – биты идентификации.

При включении режима специального маскирования запросы прерываний, поступающие в регистр IRR, обслуживаются в порядке их поступления во времени (дисциплины обслуживания FIFO (FCFS – First Come First Served)).

В режиме опроса (P=1) контроллеру запрещается автоматически прерывать работу CPU при появлении запроса прерывания от ВУ. В связи с этим CPU не воспринимает запросов прерываний по входу INTR (IF=0). Для того чтобы CPU мог узнать о наличии запроса на прерывание, он должен подать команду ввода с указание четного адреса порта: IN AL , 20H.

При выполнении этой команды в регистр AL будет помещен байт следующего формата:

| **I** | **XXXX** | **L** |
|---|---|---|
| 7 | 6 3 | 2 0 |

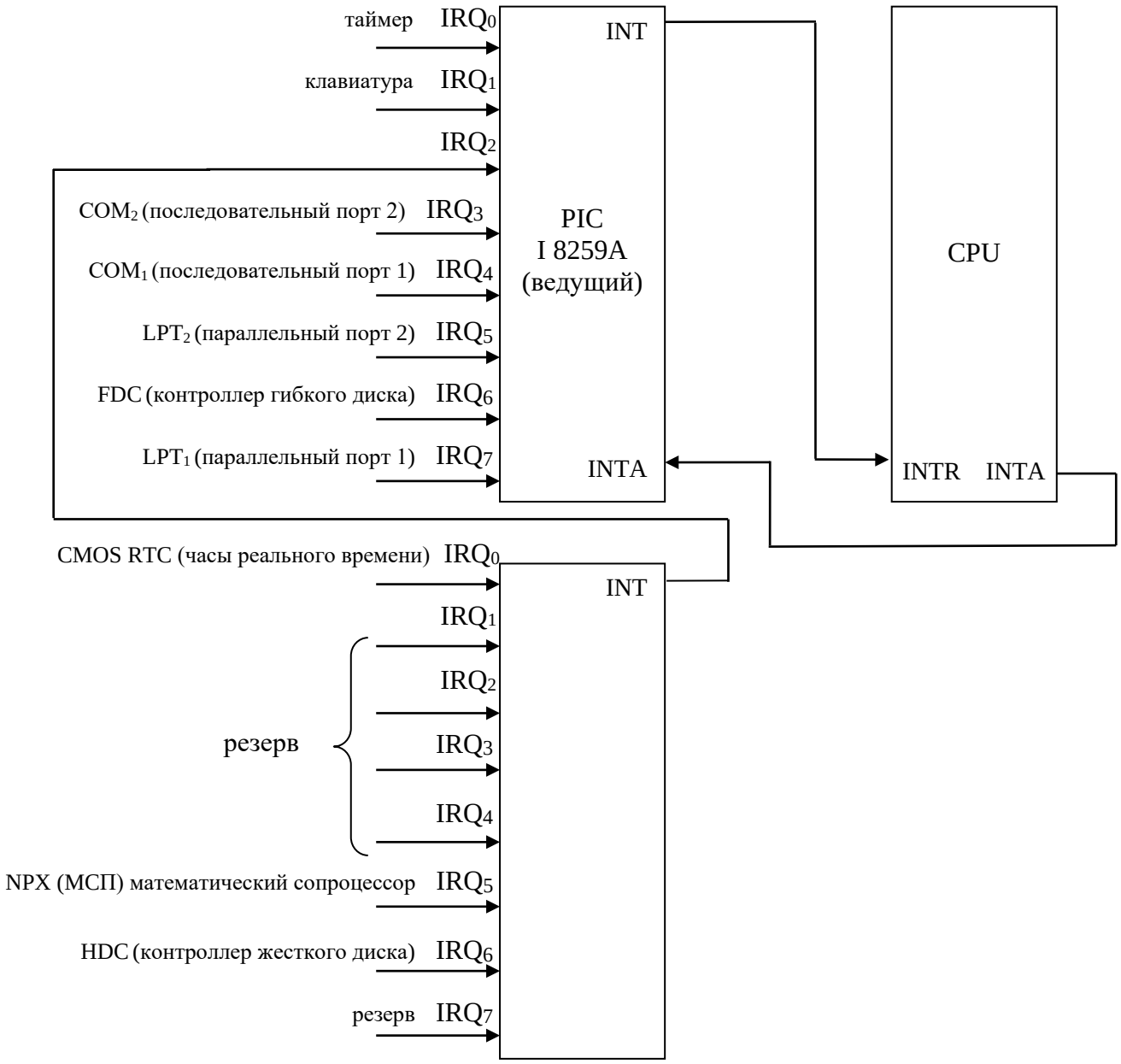
Старший бит I (Interrupt) установлен при наличии незамаскированного запроса в IRR. Младшие 3 бита содержат уровень выделенного запроса с наивысшим приоритетом. По активному уровню сигнала чтения (RD), сгенерированному командой IN, происходит установка соответствующего бита в ISR, как будто был получен сигнал подтверждения прерывания INTA. Путем программной модификации номера уровня выделенного запроса номер (тип) прерывания CPU программным путем осуществляет вызов обработчика соответствующего аппаратного прерывания. В этом режиме

процедура полинга, как выделение наиболее приоритетного из незамаскированных запросов прерываний, реализуется непосредственно в PIC, т.е. на аппаратном уровне. Для реализации программного полинга могут использоваться специальные приказы чтения регистров IRR и IMR с целью последующего программного анализа их содержимого.

В режиме разрешения чтения регистров (RR=1) содержимое регистров IRR и ISR можно прочитать в регистр AL, используя команду ввода из порта с четным адресом (см. выше).

*Замечание:* перечисленные выше три режима, реализуемые с помощью $OCW_3$, являются взаимоисключающими, из чего следует, что только один из трех битов (ESMM, R, RR) могут быть установлены.

## Стандартная схема подключения PIC к CPU и запросов различных ВУ в PIC



COM-порт (COM – Communications Port).

В принципе, компьютер может иметь до четырех последовательных портов ($COM_1$-$COM_4$), из которых на вход $IRQ_3$ подключаются $COM_3$ и $COM_4$, а на вход $IRQ_4$ подключаются порты $COM_1$ и $COM_2$. Назначением COM-портов является подключение коммуникационного оборудования (например, модемов) для связи с другими компьютерами, сетями и периферийными устройствами (ПУ). К порту могут подключаться ПУ (ВУ) с последовательным интерфейсом (например, мышь). Как правило, используется стандартный последовательный интерфейс RS-232C Serial Interface (Serial Port).

LPT (Line PrinTer)

Наиболее распространенное применение LPT-порта – подключение принтера, кроме того, возможно подключение сканеров, внешних накопителей, а также адаптеров ЛВС. Как правило, используется стандартный параллельный интерфейс IEEE-1284.

CMOS – Complimentar Metal Oxide Semiconductor (комплиментарная МОП-структура).

RTC – Real Time Clock.

В любом компьютере имеется небольшая энергонезависимая память, которая питается от аккумулятора. В этой памяти, в частности, хранятся текущие дата и время. Кроме того, в CMOS-памяти хранится некоторая информация о конфигурации компьютера. Во время загрузки компьютера дата и время считываются в область данных BIOS. Дальнейший отсчет времени после загрузки системы ведется уже с помощью таймера. Таймер представляет собой отдельную микросхему на системной плате, которая периодически (примерно 18 раз в секунду) генерирует сигнал, поступающий на вход $IRQ_0$ контроллера прерываний. Во время работы компьютера этот сигнал обрабатывается соответствующей программой BIOS, которая ведет счет текущего времени.

## Организация прерываний в базовой модели процессора Intel 8086

В этой модели зарезервировано 5 типов прерываний с минимальными номерами:

**Тип 0** – ошибка деления. Может иметь место при выполнении команд DIV/IDIV в том случае, когда либо делитель равен 0, либо частное, как результат операции, не помещается в формате делителя. Этот особый случай распознается на начальных шагах алгоритма деления.

**Тип 1** – пошаговое прерывание (прерывание отладочного режима). Генератором этого прерывания является установленный флаг TF – Trace Flag, Trap Flag. В отладочном режиме (TF=1) завершение выполнения любой машинной команды приводит к отладочному прерыванию. Отладочный режим является прерогативой программ-отладчиков.

**Тип 2** – внешнее прерывание по входу NMI (вход немаскированного прерывания). С помощью сигнала NMI сообщается о различных достаточно катастрофических ситуациях, примерами которых могут служить сбой питания (снижение питающего напряжения до некоторого критического уровня), ошибка памяти (при считывании данных из памяти фиксируется ошибка с помощью схем машинного контроля (например, контроль по четности/нечетности)).

**Тип 3** – прерывание по однобайтной команде INT (тип 3 присвоен этой команде по умолчанию и в самой команде не задается).

**Тип 4** – прерывание по переполнению. Генератором этого прерывания является соответствующая команда INTO, функцией которой является проверка флага OF, при единичном значении которого осуществляется выход на прерывание. При нулевом значении флага OF команда INTO передает управление следующей команде.

**Основные причины прерываний**

1) Внешние прерывания:

а) по входу INTR (запрос маскируется по флагу IF);
б) по входу NMI (немаскированный запрос).

2) Внутренние прерывания:

а) некорректное выполнение операции (ошибка деления, переполнение – OF);
б) команды-генераторы прерывания: INT(3), INT type, INTO;
в) отладочное прерывание (по флагу TF – режим отладки TF=1).

В основном, процессор реализует реакцию на всевозможные причины прерывания между выполнениями двух последовательных команд программы. Это означает, что опрос состояний внешних входов, связанных с запросами прерываний, осуществляется после завершения очередной команды. Из этого правила есть некоторые исключения, одним из примеров которых может служить выполнение цепочной команды с префиксом повторения REP.

В связи с тем, что число обрабатываемых элементов может быть очень большим (max – $2^{16}$), то для обеспечения быстрой реакции на некоторые внешние ситуации допускается выход на прерывание внутри команды после обработки очередного элемента строки.

Организация прерываний в процессоре Intel 8086 базируется на следующих основных положениях (концепциях):

**1.** Сохранение минимального контекста прерываемой программы на аппаратном уровне в стеке. В аппаратно-сохраняемую часть контекста включается три слова (6 байт) по порядку их загрузки в стек:

1 – регистр флагов FLAGS - состояние CPU;
2 – сегмент кода CS } адрес возврата.
3 – указатель команды IP }

В качестве адреса возврата, сохраняемого в стеке, при выходе на обработку прерывания фиксируется адрес не той команды, на которой это прерывание произошло, а следующей по порядку команды. В связи с этим в процессоре Intel 8086 возможность рестарта прерванной программы с команды, являющейся причиной прерывания, отсутствует. Такое возможно только в защищенном режиме. Остальная часть контекста прерываемой программы сохраняется в случае необходимости на программном уровне (начальная фаза программы-обработчика).

**2.** Использование системной таблицы векторов прерываний для вызова обработчиков прерываний.

Каждый вид прерывания, характеризуемый собственным обработчиком, идентифицируется уникальным номером (типом) прерывания. В таблице векторов прерываний размещаются начальные адреса программ-обработчиков, упорядоченные по типам прерываний. В данном контексте под вектором прерывания понимается полный адрес: пара seg:offset соответствующей программы-обработчика. Максимальная длина таблицы векторов прерываний рассчитана на 256 обработчиков и занимает 256*4=1024 байта в младших адресах.

При выходе на обработку прерывания тип произошедшего прерывания однозначно модифицируется в адрес соответствующего вектора прерываний путем умножения на 4.

Для каждого уникального прерывания существуют следующие возможности задания его идентификатора (типа):

**а)** тип вырабатывается аппаратно (схемно) – для всех зарезервированных типов прерываний;
**б)** тип прерывания задается во втором байте соответствующей команды (команда INT Type);

**в)** прием байтного типа в шине данных от PIC в цикле подтверждения прерывания.

В цикле вызова обработчика прерывания фактически производится загрузка вектора прерывания в регистры:

IP – слово по меньшему адресу;

CS – слово по большему адресу.

**3.** Возможность учета приоритетов запросов при их обработке.

В связи с тем, что на момент завершения очередной команды выполняемой программы могут существовать несколько причин для ее прерывания, в базовой модели принят следующий порядок приоритетов в обслуживании запросов прерываний:

1) программные прерывания (по ошибке деления или по различным модификациям команды INT);
2) внешнее прерывание по входу NMI;
3) внешнее прерывание по входу INTR;
4) пошаговый режим (TF=1).

В принципе, предусматривается возможность обработки так называемых ***вложенных прерываний*** (прерывание в прерывании). В частности, такая возможность обеспечивается для внешних прерываний по входу INTR путем принудительной установки флага разрешения прерывания IF в начальной фазе обработчика прерывания с помощью команды STI. При выходе на обработку прерывания осуществляется автоматический (аппаратный) сброс флага IF.

# Организация прерываний в реальном и защищенном режимах в старших моделях семейства Intel 80x86, Pentium

## Реальный режим и его основные особенности

В реальном режиме процессор старшей модели выполняет программы, составленные для базовой модели Intel 8086 или для реального режима процессоров младших моделей. С точки зрения программиста процессор старшей модели в R-режиме представляет собой более быстрый процессор Intel 8086 с расширением набора команд и регистров до уровня процессора старшей модели.

Основная особенность R-режима состоит в формировании физического адреса на основе простейшей модели сегментированной памяти (как в базовой модели Intel 8086). Физический адрес формируется как сумма двух 16-разрядных компонент, первая из которых представляет собой содержимое одного из сегментных регистров и трактуется как старшая часть базового

адреса сегмента, вторая компонента представляет собой внутрисегментное смещение (offset).

$$\text{ФА (физический адрес)} = \underbrace{\text{seg}*16}_{\text{16 бит}} + \underbrace{\text{offset}}_{\text{16 бит}}$$

20 бит

Размеры сегментов фиксированы и составляют $2^{16}$байта=64 Ki-байта (кибибайта) (по разрядности поля offset).

После включения процессора программа инициализации автоматически вводит его в R-режим. Аналогичная процедура выполняется и по сигналу сброса RESET. Переход из R-режима в защищенный режим (P-режим) может осуществляться командой MOV, загружающей управляющий регистр CR0 новым состоянием с установленным крайним правым битом (PE=1). Перед этим в R-режиме должна быть проведена загрузка необходимых регистров, в частности GDTR и IDTR (IDT – дескрипторная таблица прерываний – аналог таблицы векторов прерываний для защищенного режима) и таблиц, в частности GDT и IDT, используемых в P-режиме.

Обратное переключение из P-режима в R-режим выполняется также командой MOV, загружающей регистр CR0 новым состоянием со сброшенным битом PE. Предварительно необходимо выполнить некоторые подготовительные процедуры, обеспечивающие сохранение правильного функционирования при переходе к реальному режиму, в частности:

• отключить механизм страничной трансляции адресов, перейдя к использованию линейных адресов, равных физическим;

• установить для всех сегментов размер, равный 64 Ki-байта.

После выполнения команды MOV, сбрасывающей в CR0 бит PE, следует перейти к программе, выполняемой в R-режиме с помощью команды межсегментного перехода JMP FAR, которая очищает очередь команд (очередь команд – это необходимый элемент конвейера команд). Затем в сегментные регистры загружается новое содержимое, обеспечивающее формирование физических адресов в реальном режиме.

## Основные отличия R-режима от процессора Intel 8086

1. Возможность использования расширенной системы команд. Не допускается использование сравнительно небольшой группы команд, связанных непосредственно с P-режимом. К ним относятся:

SLDT – загрузка регистра LDT;

LLDT – сохранение регистра LDT;

STR – загрузка регистра задач;

LTR – сохранение регистра задач;

ARPL – корректировка запрашиваемого уровня привилегий;

LAR – загрузка прав доступа;

LSL – загрузка предела сегмента;

VERR – проверка возможности чтения;

VERW - проверка возможности записи.

Попытка выполнения этих команд в R-режиме приводит к прерыванию стандартного типа 6 – «некорректный код команды».

2. Возможность использования расширенного набора регистров. Допускается использование регистров, отсутствующих в базовой модели Intel 8086, в частности: дополнительных сегментных регистров (FS и GS), регистров системных адресов GDTR и IDTR (но не LDTR и TR), регистров отладки (DR0-DR7), управляющего регистра CR).

3. Возможность использования 32-разрядных операндов. По умолчанию в R-режиме используются 8- и 16-разрядные операнды. Возможность использования 32-разрядных операндов обеспечивается наличием специального префикса OS – Operand Size перед командой.

4. Возможность использования 32-разрядных адресов обеспечивается за счет префикса AS – Address Size перед командой. Однако, при выходе адреса за пределы стандартного сегмента генерируется прерывание стандартного типа 13 «нарушение общей защиты, выход за пределы сегмента».

5. Возможность использования физических адресов, превышающих 1 Mi байт. Максимальное значение физического адреса, формируемое в R-режиме при максимальных значениях 16-разрядных компонент равно:

$$\begin{array}{ll} \text{seg*16} & F\ F\ F\ F\ 0 \\ \text{offset} & +\ \underline{0\ F\ F\ F\ F} \\ & (1\ 0\ F\ F\ E\ F)_{16} = (1114095)_{10} \end{array}$$

Так как разрядность шины адреса – 32 или даже 36 бит, то в R-режиме возможна адресация за пределами 1 Mi байта памяти.

В процессоре Intel 8086 используется 20-разрядная шина адреса, в связи с чем при суммировании подобных компонент производится так называемое «заворачивание» адреса, поэтому для данного примера на шину адреса будет выставлен адрес $(0FFEF)_{16}$.

## Особенности организации прерываний в реальном режиме

Сохраняется общая идеология организации прерываний в процессоре Intel 8086, которая касается следующих моментов:

1) использование таблицы векторов прерываний, содержащей 4-байтные указатели на обработчики прерываний различных типов;

2) сохранение в стеке минимального контекста прерываемой программы в виде трех слов (FLAGS, CS, IP) на аппаратном уровне;

3) аппаратный сброс флага IF при выходе на обработку прерываний.

### Основные отличия обработки прерываний в R-режиме по сравнению с базовой моделью Intel 8086.

1. Увеличение числа зарезервированных типов прерываний. К новым типам прерываний в R-режиме относятся:

**тип 5** – нарушение границы массива (источником прерывания является специальная команда BOUND – проверка границы);

**тип 6** – некорректный код операции;

**тип 7** – недоступный сопроцессор;

**тип 8** – выход за пределы таблицы векторов прерываний;

**тип 13** – выход адреса операнда или команды за пределы сегмента или превышение длины машинной команды максимального предела 15 байт (такое возможно только при некорректном использовании префиксов);

**тип 16** – ошибка сопроцессора.

Имеет место при генерации одного из незамаскированных особых случаев математического сопроцессора или блока FPU (перечень особых случаев в разделе «Иерархия масок прерываний»).

2. Возможность обеспечения рестарта «виновной» команды после обработки прерываний. Это означает, что в качестве адреса возврата в стеке сохраняется именно адрес недовыполненной команды, а не следующей команды программы как в процессоре Intel 8086.

3. Использование для входа в таблицу векторов прерываний регистра IDTR. В реальном режиме базовый адрес этой таблицы инициализируется на начало памяти, т.е. равен нулю. Задаваемый в этом же регистре IDTR предел (limit) используется для проверки возможного выхода обращения за пределы таблицы (прерывание с типом 8).

## Организация прерываний в защищенном режиме

### Основные положения

В P-режиме, а также в его модификации в виде V-режима (режим виртуального процессора 8086), механизм прерываний и особых случаев, сохранив общую реакцию на их возникновение, значительно усовершенствован. Эти усовершенствования сводятся к следующему:

1) трансформация таблицы векторов прерываний в дескрипторную таблицу прерываний (IDT);

2) более сложный процесс перехода к обработчику особого случая или прерывания с привлечением системных объектов в виде шлюзов;

3) передача обработчику прерывания или особого случая дополнительной информации о причине возникновения в виде так называемого кода ошибки (Error Code);

4) использование дополнительных видов особых случаев, связанных исключительно с защищенным режимом, например, таких как неприсутствие сегмента, неприсутствие страницы, нарушение общей защиты и т.п.

## Расширенная классификация прерываний

В защищенном режиме термином «прерывание» принято обозначать только аппаратные прерывания, в то время как для программных прерываний принято использовать термин «особые случаи» или «исключения» (exception).

В зависимости от способа возникновения особых случаев и возможности перезапуска (рестарта) CPU после их обработки с вызвавшей их команды принято различать три вида особых случаев:

**Нарушение (fault)** – это особые случаи, которые выявляются и обслуживаются либо перед выполнением, либо во время выполнения «виновной» команды. При обнаружении нарушения, сохраняемые в стек значения CS и EIP указывают на команду, вызвавшую это нарушение, для возможности осуществить рестарт программы после устранения нарушения, связанного с «виновной» командой. Типичными примерами нарушений (отказов) могут служить неприсутствие сегмента или страницы.

**Ловушка (trap)** – это особый случай, который возникает непосредственно после команды, вызвавшей этот особый случай. Значения регистров CS и EIP, сохраняемые в стеке при обрабатывании ловушки, указывают на команду, следующую по отношению к команде, вызвавшей это срабатывание. Типичными примерами ловушек могут служить: ловушка пошагового исполнения программы (ее генератором является установленный флаг IF), команды-генераторы прерываний (с мнемоникой INT).

**Авария (abort)** (выход из процесса) – является особым случаем, который не позволяет точно локализовать вызвавшую его команду и осуществить рестарт программы. Аварии используются для сообщений о крупных ошибках, таких как сбои аппаратуры или ошибки в системных таблицах.

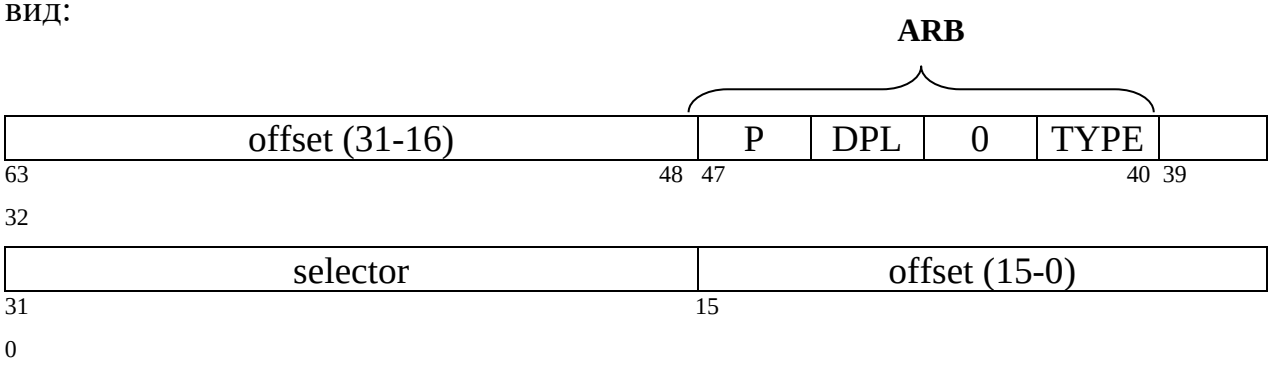
### 1. Дескрипторная таблица прерываний и ее элементы

В отличии от таблицы векторов прерываний, местоположение которой в памяти является строго регламентированным (она находится в младших адресах), IDT, в принципе, может размещаться в любом месте линейного адресного пространства, однако, как правило, для сохранения преемственности ее также располагают в младших адресах. Локализация IDT в линейном адресном пространстве и ее допустимый размер задаются содержимым системного регистра IDTR (этот регистр, так же как и GDTR является 48-разрядным: 32 бита – Base, 16 бит – Limit). Максимальный размер IDT должен быть рассчитан на 256 типов прерываний и составляет 2 Ki байта. В свою очередь минимальный размер IDT должен быть рассчитан по крайней мере на 32 зарезервированных типов прерываний. Элементами IDT являются 8-байтные дескрипторы, представляющие собой системные объекты в виде шлюзов.

**2. В IDT могут находиться шлюзы трех видов**:

• шлюзы прерываний;
• шлюзы ловушек;
• шлюзы задач.

Структура шлюзов прерываний и ловушек идентична и имеет следующий вид:

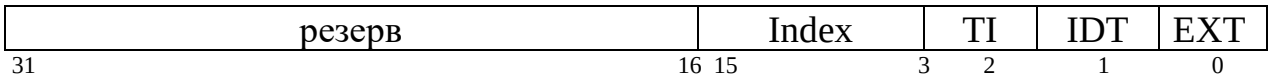



Для шлюза прерываний TYPE = Eh.
Для шлюза ловушек TYPE = Fh.

При вызове обработчика прерывания или особого случая через шлюз прерывания или ловушки 32-битное поле offset, задающее смещение в сегменте кода обработчика, загружается в регистр процессора EIP. В свою очередь 16-битное поле селектора, предназначенное для выбора сегмента кода обработчика, загружается в процессоре в регистр CS (сегмент кода). После этой загрузки в процессоре полностью определен начальный адрес обработчика прерывания или особого случая. Основным отличием использования шлюза прерывания и шлюза ловушки для вызова обработчика является аппаратный сброс флага IF при вызове обработчика через шлюз прерывания. В свою очередь вызов обработчика через шлюз ловушки не оказывает аппаратного воздействия на флаг IF.

В отличие от шлюзов прерывания и ловушек, в шлюзе задач поле offset не используется. Вызов обработчика прерывания через шлюз задачи сопровождается переключением задач с использованием системной структуры данных в виде TSS – Task State Segment (сегмент состояния задачи). В связи с этим обработчик прерывания трактуется как отдельная задача, в отличие от обработчика, вызываемого через шлюз ловушки или прерывания, при котором обработка прерывания или особого случая реализуется в контексте прерываемой задачи. В связи с тем, что в TSS обработчика прерывания задается содержимое регистра IP, то и поле offset в шлюзе задачи является невостребованным. В свою очередь поле селектора шлюза задачи должно обязательно определять дескриптор TSS, иначе генерируется особый случай.

**3.** При реализации некоторых особых случаев в стек обработчика дополнительно заносится код ошибки (после сохранения адреса возврата). Структура кода ошибки имеет вид:




Младшее слово кода ошибки практически совпадает с селектором сегмента или системного объекта. Поле индекса указывает на дескриптор, использование которого вызвало особый случай. Биты TI и IDT указывают на таблицу, в которой находится «виновный» дескриптор.

IDT = 1 → IDT
IDT = 0, TI = 0 → GDT
IDT = 0, TI = 1 → LDT

Установка бита EXT=1 означает, что особый случай вызван не выполняемой программой, а внешним сигналом прерывания. С использованием кода ошибки обработчик прерывания может проанализировать «виновный» дескриптор, извлекая его из соответствующей таблицы.

В тех случаях, когда обработчик прерывания располагается на другом уровне привилегий (в другом кольце защиты) по сравнению с прерываемой программой, в стеке обработчика, помимо всего прочего, сохраняется адрес вершины стека прерываемой программы в виде пары SS:ESP. Сохранение вершины стека осуществляется до включения в стек содержимого регистра флагов. Как правило, обработчики прерываний - особых случаев стараются размещать на наивысшем уровне привилегий (PL=0). Возможен также вариант оформления обработчиков в виде подчиненных (конформных) сегментов кода.

## Виды прерываний и особых случаев Р-режима

| № | Мнемоническое обозначение | Наименование | Причины | Вид особого случая/прерывания | Формирование кода ошибки | Класс особого случая/прерывания |
|---|---|---|---|---|---|---|
| **0** | **#DE** (Drive Error) | Ошибка деления | Команды DIV/IDIV | Нарушение | Нет | B |
| **1** | **#DB** (Debug) | Отладка | Пошаговый режим(TF=1); контрольные точки останова | Нарушение/ ловушка | Нет | A |
| **2** | - | Немаскируемое прерывание | Внешний сигнал NMI | Прерывание | Нет | A |
| **3** | **#BP** (Break Point) | Точка останова | Команда INT 3 | Ловушка | Нет | A |
| **4** | **#OF** (Overflow) | Переполнение | Команда INTO при OF=1 | Ловушка | Нет | A |
| **5** | **#BR** (BOUND Range Exceeded) | Выход за границы (нарушение контроля диапазона) | Команда BOUND | Нарушение | Нет | A |
| **6** | **#UD** (Undfined Opcode) | Недопустимый код операции | Неверный код операции или адрес | Нарушение | Нет | A |
| **7** | **#NM** (No Math Coprocessor) | Сопроцессор недоступен | Команда сопроцессора (FPU) при EM=1 или TS=1 | Нарушение | Нет | A |
| **8** | **#DF** (Double Fault) | Двойное нарушение (двойная ошибка) | При обработке одного нарушения появляется другое | Авария | Да (0) | - |
| **9** | Не используется | | | | | |
| **10** | **#TS** (Invalid TSS) | Ошибочный TSS | Переключение задачи с некорректным TSS | Нарушение | Да | B |
| **11** | **#NP** (segment Not Present) | Отсутствие сегмента | Обращение к дескриптору сегмента, в котором P=0 (кроме сегмента стека) | Нарушение | Да | B |
| **12** | **#SS** (Stack Segment Fault) | Нарушение стека | Некорректность при обращении к сегменту стека: отсутствие сегмента, выход за пределы сегмента и т.п. | Нарушение | Да | B |
| **13** | **#GP** (General Protection) | Нарушение общей защиты (основное нарушение защиты) | Все случаи нарушения защиты, не входящие в #TS, #NP, #SS, #PF | Нарушение | Да | B |
| **14** | **# PF** (Page Fault) | Страничное нарушение | Попытка обращения к неприсутствующему каталогу или странице, а также нарушение правил защиты на страничном уровне | Нарушение | Да | C |
| **15** | Не используется | | | | | |
| **16** | **#MF** (Math Fault) | Ошибка сопроцессора (FPU) | Различные виды ошибок при работе FPU (6 типов) | Нарушение | Нет | A |
| **17** | **#AC** (Alignment Check) | Нарушение контроля выравнивания | Нарушение правил выравнивания операндов на целочисленную границу (проверка реализуется при AC=1 (EFLAGS) и AM=1 (CR0)) | Нарушение | Да (0) | A |
| **18** | **#MC** (Machine Check) | Нарушение машинного контроля | Возникновение аппаратных ошибок; контролируемых средствами машинного контроля: ошибка обращения к системной шине; ошибка при обращении к памяти; ошибка контроля четности при передаче адреса или данных; ошибка кэш-памяти, в том числе TLB | Авария | Нет | A |
| **19** | **#XF** (XMM Fault) | Нарушение в блоке XMM (SSE – Streaming SIMD Extension) | Ошибки при обработке операндов с плавающей точкой, такие же как в блоке FPU (6 видов) | Нарушение | Нет | A |

**Тип 0 (#DE)**. В Р-режиме этот особый случай классифицируется как нарушение, что, в принципе, позволяет осуществить рестарт команды деления.

**Тип 1 (#DB)**. В отличие от процессора Intel 8086, этот особый случай может иметь место не только при установке флага TF, но и при переключении задач в том случае, когда в TSS входящей (новой) задачи установлен специальный бит T (бит ловушки – Trap). Кроме того, особый случай отладки может иметь место при использовании контрольных точек останова, задаваемых с помощью регистров отладки DR0-DR7 (начиная с процессора Intel 386). В регистрах DR0-DR3 задаются линейные адреса точек остановов. В регистре DR7 задаются режим и специфика остановов. В свою очередь, в регистре DR6 фиксируется состояние после останова. В принципе, с помощью регистров отладки можно реализовать три вида остановов:

1. при выборке команды по заданному адресу;
2. при чтении операнда (ячейки) по заданному адресу;
3. при записи результата по заданному адресу.

В зависимости от вида возникающего останова в заданной контрольной точке особый случай может трактоваться либо как нарушение, либо как ловушка. Например, при отладочном останове в контрольной точке по выборке команды необходимо осуществить рестарт этой команды, т.е. особый случай должен трактоваться как нарушение.

**Тип 3 (#BP)**. Генератором этого особого случая является однобайтная команда INT, которой по умолчанию присваивается тип прерывания 3. Эту команду обычно вставляют в текст отлаживаемой программы для ее останова в заданной точке и анализа текущих результатов выполнения. Особый случай – ловушка.

**Тип 5 (#BR)**. Источником этого особого случая является команда BOUND, с помощью которой осуществляется проверка возможного выхода текущего значения индекса за пределы массива. Эта команда использует три операнда: текущий индекс, верхняя граница и нижняя граница. При несоблюдении индексом границ массива осуществляется выход на особый случай этого типа, трактуемый как нарушение.

**Тип 6 (#UD).** Этот особый случай имеет место на этапе декодирования машинной команды при обнаружении недопустимого или зарезервированного кода операции. Кроме того, он может иметь место при некорректном задании адреса, например, в том случае, если в команде JMP FAR с косвенной адресацией перехода в постбайте адресации задается регистр, а не память.

Еще одним примером может служить некорректное использование префикса LOCK (захват шины) перед командой, для которой его использование является некорректным.

**Тип 7 (#NM)**. В управляющем регистре CR0 содержится два бита, оказывающих влияние на выполнение команд FPU:

*1- EM – EMulation – эмуляция;*

*2 – TS – Task Switched – задача переключена.*

Бит EM появился в ранних моделях для целей обеспечения возможности программной эмуляции системы команд математического сопроцессора. В связи с этим, установка бита EM означала, что для выполнения команды сопроцессора требуется вызвать программный эмулятор.

Бит TS устанавливается при переключении задачи. Стандартное переключение задач предполагает переключение контекста только для CPU, но не для FPU. Таком образом, проверка бита TS перед выполнением команды FPU позволяет реализовать на программном уровне переключение контекста FPU со старой задачи на новую.

Перед выполнение любой команды FPU CPU проверяет биты EM и TS, и если хотя бы один из них установлен, генерируется нарушение #NM.

CPU распознает команды FPU по специальному коду ESC в старшем байте команды (OPC). Для команд FPU старшие 5 бит кода операции имеют значение (11011 = код ESC).

Реакцией CPU при выделении команды FPU при установленном флаге TS является сохранение контекста FPU (содержимого его основных регистров) в дополнительной части сегмента TSS выходящей задачи и выборка нового содержимого этих регистров из дополнительной части сегмента TSS входящей задачи.

**Тип 8 (#DF)**. Обычно, когда CPU обнаруживает особый случай при попытке вызвать обработчик предыдущего особого случая, два особых случая обрабатываются последовательно. Если же CPU не может обработать их последовательно, генерируется особый случай двойной ошибки, классифицируемый как авария. Для выделения ситуаций, приводящих к двойной ошибке, особые случаи разделяются на три класса:

- A – легкие;
- B – тяжелые;
- C – страничное нарушение.

Возможность обработки последовательных особых случаев определяется таблицей:

| Первый особый | Второй особый случай |
|---|---|

| случай | A | B | C |
|---|---|---|---|
| A | последовательно | последовательно | последовательно |
| B | последовательно | #DF | последовательно |
| C | последовательно | #DF | #DF |

Процессор всегда включает код ошибки в стек обработчика. Однако этот код содержит полный ноль.

Двойное нарушение классифицируется как авария в связи с тем, что, как правило, оказывается невозможным осуществить рестарт виновной команды. Если при попытке вызвать двойное нарушение возникает любое другое нарушение, CPU переходит в режим отключения (shutdown). Этот режим аналогичен состоянию CPU после выполнения команды останова HLT (эта команда является привилегированной CPL=0).

Из этого состояния CPU выводится только аппаратно: сигналом NMI, который оставляет процессор в R-режиме, либо сигналом RESET, который переводит процессор в R-режим.

**Тип 10 (#TS).** Этот особый случай может иметь место только при переключении задач. Переключение задач может инициироваться следующими событиями:

1. Текущая задача выполняет команды JMP FAR или CALL FAR со ссылкой на дескриптор TSS (прямое переключение задач). Адрес перехода или вызова содержит селектор, индексирующий дескриптор TSS обязательно в GDT.

2. текущая задача выполняет команды JMP FAR или CALL FAR со ссылкой на шлюз задачи (косвенное переключение задач). Адрес перехода или вызова содержит селектор, индексирующий дескриптор шлюза задачи в GDT или LDT. В свою очередь, селектор шлюза задачи индексирует дескриптор TSS.

3. обработчик прерывания или особого случая векторизируется через шлюз задачи, расположенный в IDT.

4. текущая задача выполняет команду IRET для возврата в предыдущую задачу при установленном флаге NT (Nested Task –). Флаг NT является актуальным только для P-режима и устанавливается, если переключение задач вызвано командой CALL FAR или выходом на прерывание / особый случай. С помощью флага NT реализуется цепь вложенных задач (по аналогии с вложенными подпрограммами).

**<u>Основными действиями при переключении задач являются:</u>**

**1)** Сохранение контекста выходящей задачи. Для этого CPU выбирает базовый адрес сегмента TSS из теневого регистра, расширяющего системный регистр TR, и копирует в этот сегмент основные регистры процессора (РОНы, сегментные регистры, флаги EFLAGS, EIP). , образующие динамические поля обязательной части TSS. Статические поля TSS, которые не изменяются при выполнении задачи, не копируются. К основным статическим полям относятся:

1. селектор LDT;
2. содержимое управляющего регистра CR#;
3. полные указатели стеков $SS_k$, $ESP_k$ для трех высших уровней привилегий.

**2)** Сохранение селектора TSS выходящей задачи в специальное поле TSS входящей задачи (поле называется селектором возврата) для обратного переключения задач.

**3)** Загрузка регистра TR селектором и его теневого регистра дескриптором TSS входящей задачи.

**4)** Установка бита TS в регистре CR0.

**5)** Загрузка контекста входящей задачи из ее сегмента TSS в регистры процессора.

**6)** Переход к выполнению входящей задачи (начальный адрес программы предварительно загружен из TSS в регистровую пару CS:EIP.

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

# Микропроцессорные системы

## Конспект лекций

**Лектор:** доцент Довгий Павел Семенович

Санкт-Петербург
2007

# 1. Основные понятия и определения

Микропроцессор (МП) **– это устройство, которое осуществляет прием, обработку и выдачу информации. Конструктивно МП выполнен, как правило, на одной интегральной схеме (ИС) и выполняет действия, определенные программой, хранящейся в памяти.**

Микропроцессорная система (МПС) **– это вычислительная, контрольно-измерительная или управляющая система, основным устройством обработки информации в которой является МП.**

**Как правило, МПС строится из набора микропроцессорных БИС (СИС), входящих в состав микропроцессорного комплекта (МПК).**

Микропроцессорный комплект (МПК) **– совокупность интегральных схем (как правило, БИС и СИС), совместимых по электронным, информационным и конструктивным параметрам и предназначенных для построения электронно-вычислительной аппаратуры и/или МПС.**

**Классическим примером МПК может служить комплект фирмы Intel для ПК на базе процессора Intel 8086. Он включает в себя следующие микросхемы:**

| | |
|---|---|
| i8086 | **микросхема ЦП (CPU)** |
| i8087 | **математический сопроцессор** |
| i8088 | **аналог процессора i8086 с 8-ми разрядной внешней шиной данных (ШД)** |
| i8089 | **специализированный процессор ввода/вывода (сопроцессор ввода/вывода)** |
| i8284 | **генератор тактовых импульсов** |
| i8288 | **системный контроллер** |
| i8289 | **арбитр системной шины** |
| i8282 | **регистр-защелка** |
| i8286 | **шинный формирователь** |
| i8259(A) | **программируемый контроллер прерываний (PIC)** |
| i8237 | **контроллер DMA** |
| i8254 | **таймер** |
| i8251 | **программируемый связной адаптер (интерфейс)** |
| i8255 | **программируемый периферийный адаптер (интерфейс)** |

**Кроме того, в состав типовых МПК могут входить: контроллер динамической памяти (DRAM), контроллеры периферийных устройств (ПУ), модули ОЗУ и ПЗУ.**

**В современной микросхемотехнике наборы типовых микросхем, входящие в состав МПК, объединяются в рамках чипсетов (*chipset*).**

**Мультимикропроцессорная (мультипроцессорная) система – это система, которая образуется объединением некоторого числа универсальных или специализированных микропроцессоров (процессоров), благодаря чему обеспечивается параллельная обработка информации и распределенное управление.**

**Микроконтроллер (МК) содержит в одном кристалле все компоненты МПС: процессор, память данных и память команд (*Гарвардская архитектура*), программируемые интерфейсы и т.д.**

**Фактически, МК можно рассматривать в качестве простейшего варианта однокристальной микро ЭВМ.**

**В настоящее время термин «микроконтроллер» практически заменил устаревшее понятие «однокристальная микро ЭВМ».**

## Основные особенности МК:

- **Система команд ориентирована на выполнение задач управления и регулирования.**
- **Алгоритмы, реализуемые в МК, как правило, являются многоразветвленными с зависимостью ветвлений от значений внешних сигналов, которые поступают от датчиков управляемого объекта.**
- **Данные, которыми оперируют МК, являются малоразрядными (как правило, 8 или 16 бит, реже – 24 бита, гораздо реже – 32 бита).**
- **Схемная реализация системы управления или регулирования на базе МК не должна являться сложной и дорогостоящей.**
- **Универсальность и возможность расширения функций управления в системах на базе МК значительно ниже, чем в системах с универсальными МП.**

**Основное отличие МК и МП:**

**МК включает в свой состав *память для программ и данных*, а микропроцессор *такой памяти не содержит*.**

## Понятие микроконтроллера
## *(по Мелехину)*

**Микроконтроллер (МК) – это программируемое однокристальное вычислительное устройство со встроенным набором средств ввода/вывода, применяемое для решения задач управления и первичной обработки данных в технических системах (по учебнику В.Ф. Мелехина и Е.Г. Павловского).**

**По сведениям из учебного пособия: годовой объем выпуска МК превышает 2 млрд. экземпляров, превосходя на порядок объем выпуска универсальных микропроцессоров. Номенклатура выпускаемых МК содержит несколько тысяч типов.**

**Характерной особенностью МК является размещение на одном кристалле с ЦПУ внутренней памяти для программ и данных, и большого набора устройств и блоков для связи с периферийным оборудованием, к которому относятся:**

**- параллельные и последовательные порты ввода/вывода;**
**- таймерный блок;**
**- АЦП;**
**- контроллер дисплея и т.п.**

**Благодаря использованию внутренней памяти и большому набору устройств для связи с периферией, система управления на базе МК содержит минимальное количество дополнительных элементов.**

**В связи с широким диапазоном решаемых задач управления, требования, предъявляемые к производительности процессора, объему внутренней памяти команд и данных, а также набору периферийных устройств, оказываются весьма разнообразными. Для удовлетворения разнообразных запросов потребителей выпускается большая номенклатура МК, которую принято разделять на 8-ми, 16-ти и 32-х разрядные МК.**

## 8-разрядные МК

**8-разрядные МК представляют собой наиболее многочисленную группу МК (порядка 50% рынка МК в стоимостном выражении, поэтому они еще и гораздо дешевле).**

**Они имеют относительно низкую производительность, которой оказывается достаточно для решения широкого круга задач управления в относительно несложных объектах и устройствах массового выпуска.**

**<u>Основными областями применения 8-разрядных МК являются</u>:**

**• бытовая и измерительная техника;**
**• промышленная автоматика;**
**• автомобильная электроника;**
**• теле-, видео-, аудио- аппаратура;**
**• средства связи.**

**Для МК этого типа характерной чертой является использование *Гарвардской архитектуры.***

**Внутренняя память команд и данных имеет небольшую емкость (единицы килобайт), однако имеется возможность подключения дополнительной (не кристальной) памяти команд и данных.**

**Набор команд относительно небольшой (50-100 команд) с использованием простейших режимов адресации.**

**В некоторых моделях реализованы принципы RISC-архитектуры, обеспечивающие выполнение большинства команд за один машинный такт.**

## 16-разрядные МК

**– МК во многих случаях являются усовершенствованной модификацией своих 8-разрядных прототипов.**

**Они характеризуются не только увеличенной разрядностью обрабатываемых данных, но и расширенной системой команд, режимов адресации, увеличенным набором регистров и объемом адресуемой памяти, а также рядом других дополнительных возможностей, что позволяет повысить производительность МК и расширить область их применения.**

**Используемый объем памяти программ и данных обычно расширяется до нескольких Мбайт путем подключения внешних микросхем памяти.**

**Во многих случаях реализуется программная совместимость с младшими 8-разрядными моделями.**

**<u>Основные области применения 16-разрядных МК:</u>**

- **сложная промышленная автоматика;**
- **телекоммуникационная аппаратура;**
- **медицинская и измерительная техника.**

## 32-разрядные МК

**– содержат высокопроизводительный процессор, соответствующий по своим возможностям младшим моделям универсальных МП.**

**В ряде случаев процессор, используемый в 32-разрядных МК, аналогичен процессорам, которые выпускались ранее этой фирмой в качестве *универсальных*.**

**Например, в 32-разрядных МК фирма Intel использует процессор i386. В МК фирмы Motorola широко применяется процессор 680x0. В ряде других МК в качестве процессорного ядра служат RISC-процессоры типа PowerPC, выпускаемые фирмами IBM и Motorola.**

**На базе этих универсальных процессоров были реализованы различные модели ПК и, следовательно, наработан большой объем прикладного и системного программного обеспечения, которое может быть использовано в МК.**

**Кроме 32-разрядного процессора на кристалле МК размещается внутренняя память команд емкостью несколько десятков Кбайт, память данных емкостью в несколько Кбайт, сложные устройства для управления периферией:**

- **таймерный процессор;**
- **коммуникационный процессор;**
- **модуль последовательного обмена и др.**

**МК работает с расширенной некристальной памятью, емкостью порядка 16 Мбайт и более.**

**Структурная организация МК включает в себя различные средства увеличения производительности путем использования низкоуровневого параллелизма на уровне машинных команд (этот вид параллелизма принято обозначать *ILP – Instruction Level Parallelism*). К таким средствам относится конвейер команд, а также блоки для реализации суперскалярной обработки.**

**32-разрядные МК находят применение в системах управления сложными объектами промышленной автоматики (робототехнические системы, средства комплексной автоматизации производства), контрольно-измерительной аппаратуре, телекоммуникационном оборудовании.**

## Основные модели МК

**• MCS (MCS-51) – разработан фирмой Intel в 1980 году. Содержит 128 тысяч транзисторов на кристалле. В дальнейшем его производством занималось много фирм, таких как Atmel, Philips.**

**Советский аналог – ОМК 1816 ВЕ 51**

**• HC – фирма Motorola.**

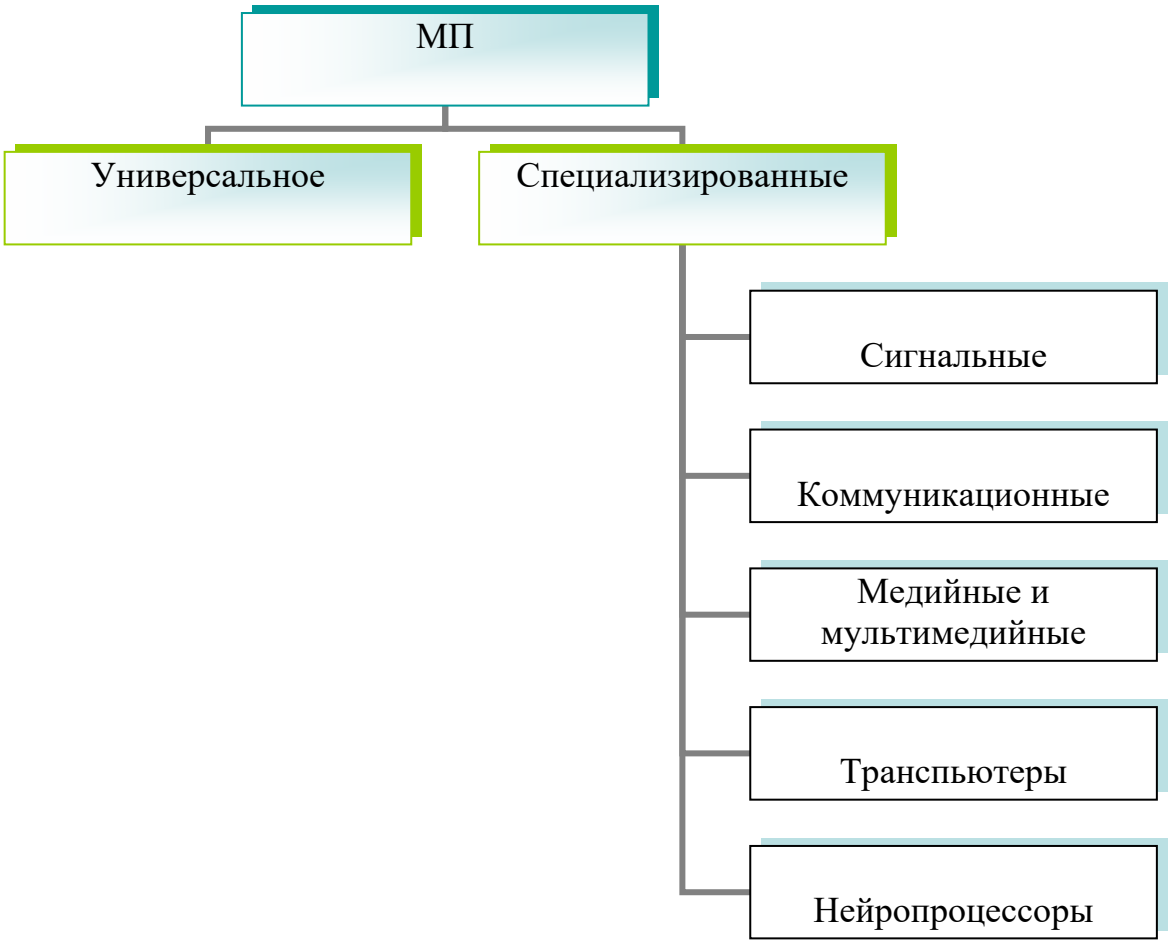
**• PIC (PIC 16, PIC 17, PIC 18). Фирма MicroChip.**
***PIC – Peripheral Interface Controller***

**• AVL фирмы ATMEL.**

**• SX фирмы Scenix.**

## Классификация микропроцессоров (МП)

**Одним из основных признаков классификации является *область применения* (назначения) МП.**




## Краткая характеристика классов

Универсальные МП **предназначены для решения широкого класса задач вычисления, обработки данных и управления.**

Специализированные МП **предназначены для решения определенного, достаточно узкого круга задач.**

• Сигнальные МП **предназначены для цифровой обработки сигналов в реальном масштабе времени. С их помощью решаются следующие задачи:**

**- фильтрация сигналов;**

**- вычисление корреляционной функции;**

**- преобразование сигналов (в частном случае прямое и обратное преобразование Фурье).**

• Коммуникационные МП **используются в системах связи и телекоммуникационных системах. В отличие от сигнальных МП, которые ориентированы на реализацию протоколов физического и канального уровней, с помощью коммуникационных МП осуществляется реализация сетевого и транспортного уровней.**

• Медийные и мультимедийные МП **предназначены для обработки аудиосигналов, графической информации, а также для решения ряда задач в мультимедиа-компьютерах, бытовой технике и игровых приставках.**

• Транспьютеры **предназначены для реализации массовых параллельных вычислений и работы в мультипроцессорных системах. Характерным свойством транспьютеров является наличие внутренней памяти и встроенного межпроцессорного интерфейса для связи с другими МП.**

• Нейропроцессоры.

**Нейрон – это нервная клетка, из которой состоит мозг человека (примерно 100 млрд. шт.). Назначение: оперативное управление организмом человека.**

**Попытки создания математических моделей нейрона и их физическая реализация начались уже в середине прошлого века.**

**Простейшая модель нейрона представляет собой пороговый элемент, который срабатывает, если суммарный сигнал на входах превышает некоторое пороговое значение.**

**Наиболее распространенными функциями активации нейрона являются:**

**- ступенчатая пороговая функция:**

**- линейная пороговая функция;**

**- линейная функция;**

**- сигмоидная функция;**

**- функция гауссиана.**

**Нейронная сеть создается путем объединения выходов одних нейронов со входами других. Межнейронные соединения снабжаются определенными весовыми функциями. Вид графа межнейронных**

**соединений является одним из классификационных признаков типа нейронной сети, по которым они разделяются на два вида:**

**- сети без циклов (ациклические);**

**- циклические сети (сети с обратными связями).**

**Приняв некоторое соглашение о тактировании нейронной сети, можно получить аппарат для задания алгоритмов посредством нейронной сети. Разнообразие этих алгоритмов ничем не ограничено, т.к. можно использовать нейроны с различными функциями активации, различными функциями состояния и различными весовыми функциями входов/выходов.**

**Нейросетевой подход показал свою эффективность как при решении плохо формализованных задач распознавания, кластеризации, ассоциативного поиска, так и при решении хорошо формализованных, но трудоемких задач оптимизации и аппроксимации функций многих переменных.**

**Общая идея применения нейронных сетей (НС) для решения плохо формализованных задач основана не на выполнении предписанного алгоритма, а на запоминании сетью предъявленных ей примеров на этапе создания сети и выработки результатов, согласованных с запомненными примерами, на этапе решения задачи нейросетью.**

**Практическая реализация нейросетевых вычислений подразумевает, во-первых, минимизацию объема памяти, требуемой для запоминания эталонных примеров, и, во-вторых, быстрое использование запомненных примеров, исключающее применение традиционных типов памяти.**

**При построении нейронной сети осуществляется и процесс ее обучения на основе исходных (базовых) примеров. При дальнейшей работе с сетью, путем решения на ней задач происходит ее совершенствование в плане самообучения на основе получаемого от решения задач опыта. Следовательно, НС обладает свойствами искусственного интеллекта.**

**<u>В настоящее время используется 3 направления для реализации нейронных вычислений:</u>**

**1) на базе каскадного соединения универсальных RISC или CISC микропроцессоров (программная эмуляция нейросети);**

**2) на базе программируемых ПЛИС или специализированных процессоров с параллельной обработкой данных, например, сигнальных процессоров (TMS, DSP, ADSP);**

**3) на специализированной элементной базе в виде однобитовых процессоров, называемых *нейрочипами*. Нейрочипы, выпускаемые в настоящее время, могут быть трех типов: цифровые, аналоговые, гибридные (более детально см. Корнеев, Кисилев «Современные МП»).**

## Основные модели специализированных МП

| Специализированные МП | Модель | Фирма-производитель |
|---|---|---|
| Сигнальные МП | **TMS** | **Texas Instrument (TI)** |
| | **ADSP** | **Analog Devices** |
| | **DSP** | **Motorola** |
| Коммуникационные МП | **MPC** | **Motorola** |
| | **IPX** | **Intel** |
| Медийные и мультимедийные МП | **Media Proc** | **Micro Unity** |
| | **Trimedia** | **Philips** |
| | **NVI** | **NVIDIA** |
| | **Media GX** | **Cyrix** |
| Транспьютеры | **T-2, T-4, T-8** | **Inmos** |
| Нейропроцессоры | **NM** | **Neuro Matrix** |

Вторым, достаточно важным признаком классификации МП может служить их внутренняя архитектура. По этому признаку МП можно разделить на 3 вида:

1) CISC – Complex Instruction Set Computer;
2) RISC – Reduced Instruction Set Computer;
3) VLIW - Very Long Instruction Word.

## Основные особенности RISC-процессоров

**Зарождение RISC-архитектуры относится к середине 70-х годов прошлого века.**

**Многочисленные исследования частоты использования машинных команд различных типов в современных для того времени компьютерах послужили основанием для формулировки принципа *«20x80»*. Суть его состоит в том, что лишь 20% машинных команд используют порядка**

**80% объема машинных программ. Из этого следует, что остальные 80% команд либо очень малы, либо вообще не востребуются.**

**В соответствии с этим была предпринята попытка построения компьютеров с минимальной системой команд, получивших название RISC.**

**В принципе, первой разработкой RISC-архитектуры является миникомпьютер IBM-801, основным разработчиком которого является Дон Кок (1979г.).**

**Более важными разработками в плане появления самого термина RISC являются разработки под руководством Дэвида Патерсона, проводившиеся в Калифорнийском университете и получившие названия RISC-I (1981г.) и RISC-II (1983г.). Усовершенствованная модель RISC-II положила начало популярному семейству RISC-процессоров SPARC (Sun Microsystems).**

**Практически в то же время в Стенфорском университете проводились аналогичные разработки под руководством Джона Хеннеси. Там в 1982г. была создана модель MIPS:**

MIPS – Microprocessor without Interlocked Pipeline Stages **– микропроцессор без блокировок фаз (стадий) конвейера. Само название модели говорит о том, что основное внимание разработчики уделяли конвейерному принципу выполнения команд и использованию различных приемов увеличения производительности конвейера. Эта разработка получила дальнейшее развитие в RISC-процессорах семейства MIPS одноименной фирмы MIPS Technologies Inc.**

## Основные особенности RISC-архитектуры

1. Ограниченное число машинных команд.

**В принципе, в систему команд RISC-процессоров включаются только команды, наиболее часто используемые при решении задач обработки данных.**

**Первые модели RISC-процессоров имели мощность системы команд, равную примерно 50÷70 команд.**

**Для современных моделей RISC-процессоров имеется явная тенденция к расширению системы команд, и мощность системы команд определяется значениями порядка 150÷200.**

**Система команд современных RISC-процессоров включает в себя:**

**- целочисленную арифметику (может быть без команды деления);**

**- арифметику с плавающей запятой;**

**- мультимедийные расширения (векторные команды).**

2. Выполнение почти всех машинных команд за один такт.

**Фактически, в RISC-процессорах стираются грани между машинными командами и микрокомандами. В соответствии с этим**

**тактовая частота RISC-процессора может являться прямой оценкой пиковой производительности. Так, например, при тактовой частоте 1ГГц пиковая производительность приближается к 1000 MIPS (Millions of Instructions Per Second).**

3. Отсутствие блока микропрограммной памяти в устройстве управления (УУ).

**Устройство управления RISC-процессоров реализуется в виде так называемого схемного автомата (в отличие от микропрограммного автомата в CISC-процессорах).**

**В RISC-процессорах машинные команды выполняются в режиме прямого управления, а не в режиме интерпретации, как в CISC-процессорах.**

4. Фиксированная длина машинной команды.

**32 бита = 4 байта.**

5.Небольшое число разнообразных форматов команд (3-5).

**Самая короткая команда – IA32 – 1-байтная.**

6. Небольшое число используемых режимов (способов) адресации.

**Типичными режимами адресации в RISC-процессорах являются:**

- **прямая регистровая;**
- **относительная (регистр + смещение);**
- **непосредственная;**
- **косвенная регистровая.**

7. Как следствие из пунктов 3,4,5,6 – упрощение устройства управления в RISC-процессорах **приводит к тому, что оно занимает 5-10% площади кристалла, в отличие от CISC-процессоров, где оно занимает 30-50% площади кристалла.**

**Освобожденная площадь кристалла используется для следующих целей:**

1) **увеличение числа внутренних регистров (минимальный объем регистровой памяти в RISC-процессорах включает в себя 32 РОН и 32 РПЗ (регистр с плавающей запятой), не считая специальных регистров);**
2) **увеличение объема кэш-памяти уровня L1;**
3) **увеличение блоков конвейера команд, дублирование или троирование конвейера.**

8. Большое число регистров в RISC-процессорах **порождает проблему их сохранения и восстановления при входах в программы и выходах из**

**них, а также при обработке прерываний. Кроме того, существует проблема совместного использования различных регистров многими процедурами.**

**Для решения этих проблем специально для RISC-процессоров разработан механизм регистровых окон.**

9. Все команды обработки данных используют только регистровые операнды**, т.е. являются командами типа «reg ↔ reg». Обмен с памятью реализован с помощью специальных выделенных команд:**

**LOAD (mem → reg);**

**STORE (reg → mem).**

**Впервые подход с выделенным доступом к памяти был реализован С. Креем в компьютерах CDC-6600, CDC-7600 и, позднее, в суперкомпьютере CRAY-1 (70-е годы).**

10. Широкое использование принципов конвейерной и суперскалярной обработки. **Эти принципы основаны на параллельном выполнении нескольких последовательных команд машинной программы (самый низкоуровневый параллелизм).**

**Благодаря использованию конвейерной и суперскалярной обработки увеличивается производительность процессора и компьютера в целом.**

**Конвейеризация команд базируется на двух основных принципах:**

**1) Разделение выполнения машинной команды на ряд последовательных этапов (фаз, стадий) для классического конвейера команд разделяется на 6 фаз:**

**1. выборка команды – IF (Instruction Fetch);**
**2. декодирование команды – D (Decode);**
**3. формирование адресов операндов – OA (Operand Address);**
**4. выборка операнда – OF (Operand Fetch);**
**5. выполнение операции – EX (EXecutive);**
**6. запись результата – S (Store).**

**2) Выделение в аппаратуре CPU нескольких относительно независимых блоков, способных функционировать параллельно во времени, выполняя одну или несколько последовательных фаз машинной команды.**

**Параллелизм на уровне машинных команд, реализуемый конвейером, принято обозначать *ILP (Instruction Level Parallelism)*.**

Под суперскалярной обработкой понимается возможность параллельной обработки нескольких команд на фазе выполнения EX.

**Необходимым условием реализации суперскалярной обработки является наличие большого числа обрабатывающих блоков (устройств) в составе CPU, которые способны выполнять различные команды на фазе EX. К таким блокам прежде всего относятся:**

**- ALU (IU – Integer Unit);**
**- FPU;**
**- SSE.**

**В RISC-процессорах, по сравнению с CISC-процессорами, конвейеризация команд и суперскалярная обработка реализуются за счет гораздо меньших аппаратных затрат.**

## Сравнительный анализ CISC и RISC архитектуры

| Архитектурное свойство | CISC | RISC |
|---|---|---|
| **Мощность системы команд** | **400÷500** | **150÷200** |
| **Средняя длительность выполнения машинных команд (в тактах)** | **1÷50** | **1,2÷1,6** |
| **Количество внутренних регистров** | **50÷70** | **100÷500** |
| **Количество форматов команд** | **10÷20** | **3÷5** |
| **Длина машинной команды (в байтах)** | **1÷15** | **4** |
| **Количество режимов адресации** | **10÷15** | **3÷5** |
| **Местоположение операндов** | **Любое** | **Reg-reg для всех команд обработки, reg-mem для команд типа STORE, mem-reg для команд типа LOAD** |
| **Сложность устройства (блока) управления (% от площади кристалла)** | **30-50%** | **5-10%** |
| **Тактовая частота** | **2÷3 ГГц** | ? |
| **Длина компилируемой машинной программы** | **Меньше** | **Примерно на 30% больше** |

## Основные модели CISC и RISC процессоров

CISC-процессоры:

1. **Вычислительные системы IBM/360, IBM/370, IBM/390, … (начало выпуска этих моделей – 1964г.).**

**Для сохранения преемственности в XXI веке IBM выпустила модель процессора G5, которая является совместимой по системе команд с компьютерами семейств IBM/360, IBM/370, IBM/390, …**

**Блок вычислений с плавающей точкой модели G5 поддерживает две стандартные формы представления с плавающей точкой:**

- **IEEE-754, основание порядка S=2;**
- **HFP (стандарт IBM/360), S=16.**

2. **Семейство миникомпьютеров VAX-11, которые выпускались фирмой DEC (70е – 90е гг.).**

**VAX – Virtual Address eXtended (расширенная виртуальными адресами).**

3. **Микропроцессоры фирмы Intel семейства 80x86, Pentium.**

4. **Микропроцессоры фирмы AMD (NexGen):**

**K5 (1996г.);**
**K6 (1997г.);**
**K7 (1999г.) Athlon, Duron;**
**K8 (2001г.) Hammer (64-разрядная архитектура, совместим с x86).**

5. **Микропроцессоры фирмы Cyrix.**
**Основные модели: 5x86, 6x86.**

6. **Процессоры фирмы Motorola: M68xxx (аналог VAX-11).**

RISC-процессоры:

1. ALPHA **(компания DEC). Начало выпуска моделей – с 1992г.**
**Модели: 2106x, 21164, 21264, 21364 (2002г.).**
**В 1992г. компания DEC была куплена фирмой Compaq, которая затем объединилась с HP. С 2004г. разработка процессоров ALPHA была прекращена в связи с тем, что HP переключилась на поддержку архитектуры IA-64.**

2. **Процессоры** SPARC **(фирма Sun Microsystems).**
**SPARC – Scalable Processor ARChitecture.**
**Начало выпуска моделей – с 1985г.**
**Являясь разработчиком архитектуры SPARC, компания Sun Microsystems предоставляет лицензии на производство микропроцессоров этой архитектуры в соответствии с предлагаемой фирмой спецификацией. Производством процессоров с архитектурой SPARC занимаются компании Fujitsu, Philips,…**
**К настоящему времени выпущено 4 основных вида моделей: Micro SPARC, Super SPARC, Hyper SPARC (32-разрядный), Ultra SPARC (64-разрядный).**

3. **Модель** POWER **(IBM), модель POWER PC (IBM, Motorola, Apple).**
**POWER – Performance Optimized with Enhanced RISC (оптимизация производительности с использованием RISC).**
**Архитектура POWER разрабатывается фирмой IBM с конца 80х годов прошлого столетия. Основная область применения – высокопроизводительные серверы и суперкомпьютеры (MPP-системы и кластеры).**
***MPP - Massively Parallel Processing* (массивная параллельная обработка).**
**Архитектура POWER PC разрабатывается с 1991г. Областью ее применения являются ПК и рабочие станции низкого уровня. Первые модели POWER PC - 32-разрядные, начиная с модели POWER PC-620 –**

**64-разрядные. В моделях POWER PC реализация векторной обработки осуществляется с помощью блока Altivec.**

**Одна из последних моделей – POWER 4 (2001г.)- представляет собой двухпроцессорную систему на одном кристалле. Наряду с традиционным ILP-параллелизмом (используется 4 конвейера команд), реализован также параллелизм *TLP (Thread Level Parallelism)* на уровне трэдов (нитей).**

**В компьютере допускается одновременное выполнение до восьми команд в каждом процессоре.**

4. **Микропроцессоры семейства** PA-RISC **(HP).**
***PA – Precision Architecture*.**

5. **Микропроцессоры компании MIPS (SGI).**
**Модели: R2000, R3000, …, R 12000, R14000 (2001г.).**

## Основные особенности VLIW-архитектуры

**VLIW - Very Long Instruction Word.**

**Основная идея VLIW-архитектуры базируется на использовании так называемых *длинных машинных команд,* в которые упаковываются несколько (как правило, 3-10) простых команд. При этом предполагается, что все простые команды, входящие в одну составную команду, могут быть выполнены параллельно. Тем самым предполагается, что в структуре VLIW-процессора имеется несколько обрабатывающих (исполнительных) блоков, способных функционировать параллельно во времени.**

**Использование VLIW-архитектуры предполагает, что компоновка длинных команд из простых производится на этапе компиляции. Такой подход обычно называют статическим ILP (*статическим параллелизмом на уровне команд*). Таким образом, вся тяжесть по реализации ILP переносится с аппаратуры на ПО в виде компилятора. В этом плане задача построения оптимизирующих компиляторов до сих пор далека от идеальных решений.**

**Зарождение и реализация идей VLIW-архитектуры относится к 70-м годам прошлого века. Первоначально VLIW-архитектура была реализована в векторных сигнальных сопроцессорах, которыми оснащались суперкомпьютеры и мощные мэйнфреймы. Первые полноценные VLIW-компьютеры появились в 80-е годы и были реализованы в виде минисуперкомпьютеров компаний Multi Flow, Culler, Cydrome, однако они не имели коммерческого успеха.**

**В настоящее время VLIW-архитектура используется:**

**- в некоторых моделях специализированных микропроцессоров (например, ADSP-TS001);**

**- в универсальных процессорах фирмы Intel (Itanium) – реализация в виде EPIC, процессор Crusoe фирмы Transmeta.**

## Реализация VLIW-архитектуры в виде EPIC-архитектуры в процессорах Itanium

EPIC – Explicitly Parallel Instruction Computing **(явный параллелизм на уровне команд).**

**Тот факт, что в одну большую команду (включающую несколько сотен бит) объединяются несколько простых команд, по сути сближает VLIW-архитектуру и RISC-архитектуру. По мнению многих специалистов, VLIW-архитектура является дальнейшим развитием RISC-архитектуры, в связи с чем ее (VLIW) часто определяют как Post-RISC архитектуру.**

**Основное отличие в реализации ILP во VLIW-архитектуре – статический подход к распараллеливанию, создаваемому заранее на этапе компиляции (отсюда – повышенные требования к компиляторам).**

**В свою очередь, суперскалярные процессоры реализуют ILP динамически, преобразуя последовательную программу в параллельную аппаратными средствами на фазе исполнения, возможно, с изменением порядка команд.**

## Сравнительный анализ суперскалярной (IA-32 (P6)) и VLIW-архитектуры (IA-64(Itanium))

| № | Суперскалярная архитектура | VLIW(EPIC)-архитектура |
|---|---|---|
| **1** | **Использование сложных команд переменной длины, обрабатываемых по одной** | **Использование простых команд, сгруппированных по 3 в длинную команду с возможностью параллельного выполнения** |
| **2** | **Переупорядочивание и оптимизация команд во время выполнения** | **Переупорядочивание и оптимизация команд во время компиляции** |
| **3** | **Попытки предсказания направления переходов с учетом предыстории их выполнения** | **Исполнение нескольких последовательностей команд одновременно без предсказания переходов (параллельное выполнение обеих ветвей программы с последующим анализом их актуальности)** |
| **4** | **Загрузка данных из памяти по мере их необходимости (в момент фактической ссылки на них из программы)** | **Загрузка данных до того, как они фактически потребуются (предварительная загрузка)** |

## Основные особенности архитектуры EPIC

**1. Наличие явного параллелизма в машинном коде.**

**2. Большое количество внутренних регистров.**

**3. Масштабирование архитектуры путем наращивания числа функциональных блоков.**

**4. Предикация – команды разных ветвей программы снабжаются разными предикатами (тегами, ярлыками, признаками) и запускаются на выполнение параллельно.**

**5. Предварительная (спекулятивная) загрузка данных.**

Подробнее:

**1.** Явный параллелизм **основан, во-первых, на объединении трех простых команд в рамках одной длинной команды, называемой «bundle» («связка»).**

**В Itanium всего 3 команды упаковываются в одну связку.**

**Структура связки имеет вид:**

| | | | **T** |
|---|---|---|---|
| **Instruction slot 2** | **Instruction slot 1** | **Instruction slot 0** | **template** |
| **41 бит** | **41 бит** | **41 бит** | **5 бит** |

**T – это шаблон или маска, указывающая на возможность параллельного исполнения простых команд данной связки, а также связок, следующих за ней.**

**В рамках одной связки можно задать следующие комбинации параллельного или последовательного выполнения ее команд:**

**I2 || I1 || I0**
**I2 & I1 || I0**
**I2 || I1 & I0**
**I2 & I2 & I0**

**Структура слота команды содержит 5 полей:**

| **OpC** | **PR** | **R2** | **R1** | **R0** |
|---|---|---|---|---|
| **14 бит** | **6 бит** | **7 бит** | **7 бит** | **7 бит** |

**OpC – Operation Code**
**PR – Predicate**
**R2, R1, R0 – адреса регистров операндов и результата**

**2.** Регистры **Itanium:**

**1) 128 РОН (GPR), каждый по 64 бита;**
**2) 128 регистров с плавающей точкой РПТ (FPR – Floating Point Register), каждый по 80 бит;**
**3) 64 регистра предикатов РП (PD) – по 1 биту;**
**4) 8 регистров адреса РА (AR) – по 64 бита;**
**5) 128 прикладных регистров ПР (AR – Application Register) – по 64 бита.**

**32 РОНа с младшими номерами реализованы как обычные адресные регистры, а остальные 96 РОНов организованы в виде регистровых окон.**

**3.** Свойство масштабируемости **специалисты Intel и HP именуют как inherently scalable instruction set (наследственно масштабируемый набор команд).**

**Одна длинная команда (связка), состоящая из трех простых команд соответствует набору из трех функциональных устройств процессора, способных функционировать параллельно и объединяемых в один функциональный блок.**

**Процессоры с архитектурой IA-64 (EPIC) могут содержать разное количество таких блоков, оставаясь при этом совместимыми по коду.**

**Благодаря тому, что в шаблоне поля T указана зависимость не только между простыми командами этой связки, но и между связками, процессору с N одинаковыми блоками из трех функциональных устройств будет соответствовать командное слово из N связок или 3×N простых команд.**

**В архитектуре EPIC обеспечивается возможность увеличения числа функциональных устройств путем блочного наращивания по 3 устройства.**

**В процессоре Itanium-2 имеется 23 функциональных устройства. Из них:**

**- 6 устройств IU (Integer Unit);**

**- 4 устройства FPU – для обработки данных с плавающей точкой;**

**- 6 MMU (MultiMedia Unit) – устройства мультимедийной обработки;**

**- 4 блока LSU (Load/Store Unit) – загрузка и выгрузка данных;**

**- 3 модуля ветвления (выполнение команд перехода).**

**4.** Предикация **является альтернативой предсказания ветвлений.**

**В архитектуре большинства современных процессоров заложена возможность предсказания наиболее вероятного направления ветвления и выполнения команд программы именно по этому направлению. Такая особенность называется *исполнением по предположению* (speculative execution) и реализуется на основе аппаратного средства, называемого BTB – Branch Target Buffer (буфер меток (адресов) ветвлений).**

**Возможная ошибка предсказанного направления ветвления обходится необходимостью сброса конвейера команд и его реинициализации по истинному направлению ветвления.**

**Использование предикации предполагает, что на этапе компиляции все команды программы, относящиеся к одному пути ветвления, помечаются уникальным идентификатором, называемым *предикатом (predicate)*, а команды альтернативного направления – другим предикатом.**

**Каждая простая команда содержит 6-битное поле для хранения значений предикатов. На этапе исполнения команды разных ветвей могут выполняться параллельно, что задается соответствующим значением поля шаблона. Когда процессор вычисляет значение условия перехода и определяет его истинное направление, происходит запись единичного значения в соответствующий предикатный регистр и сброс предикатного регистра, соответствующего ложному направлению. К этому моменту процессор уже, возможно, выполнил некоторое количество команд, соответствующих обоим путям, но до сих пор не сохранил результат этого выполнения, чему препятствовало неопределенное значение соответствующих предикатных регистров. Перед тем, как сделать запись результата, процессор проверяет соответствующий предикатный регистр. Если он установлен, то результат записывается, если сброшен – не записывается.**

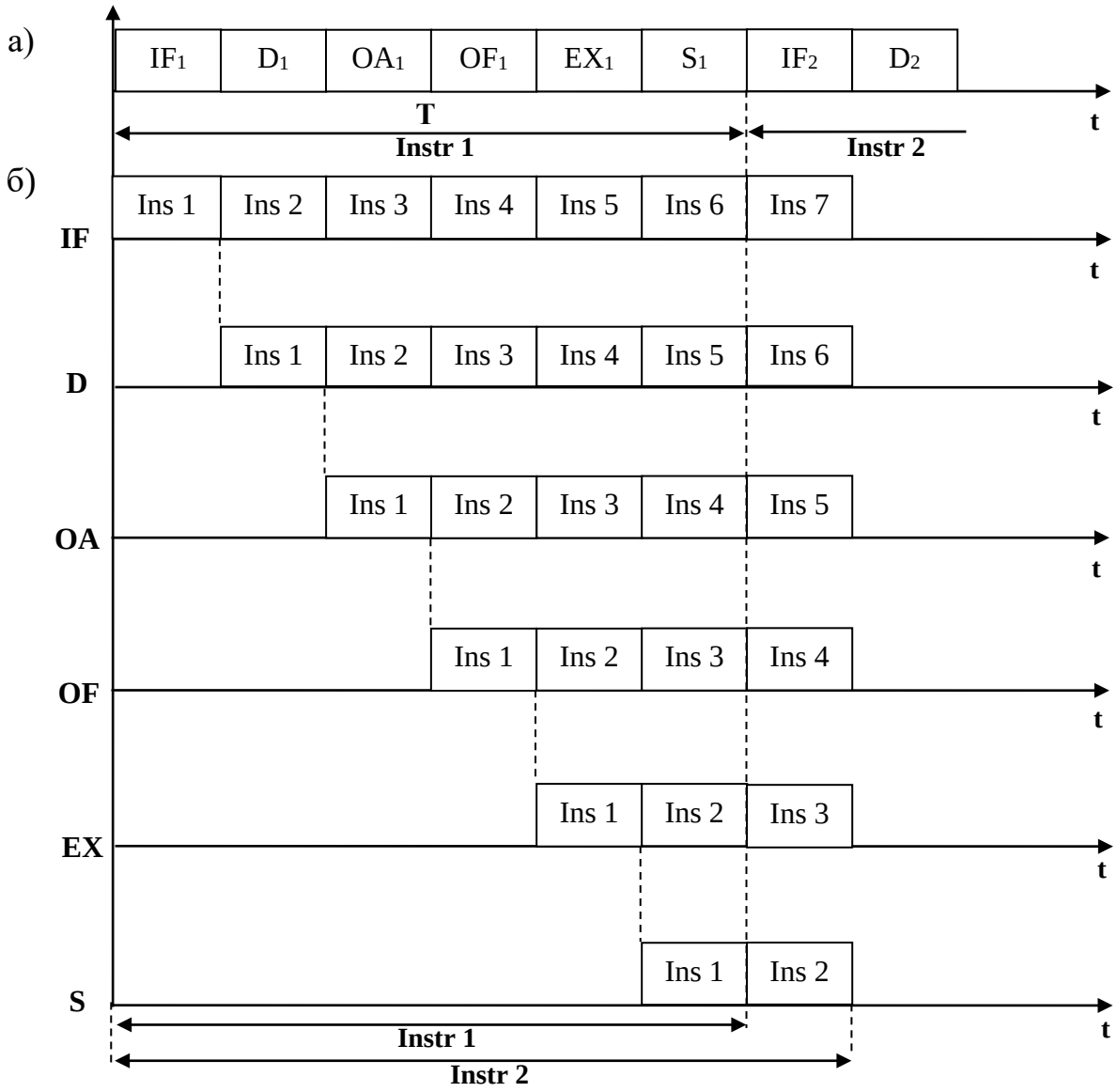
**5.** Предварительная загрузка данных **– загрузка по предположению (speculative loading) позволяет снизить потери производительности конвейера команд, связанные с простоями при обращении к памяти. Компилятор, просматривая код программы, определяет команды, использующие данные из памяти. Везде, где это возможно, добавляется команда предварительной загрузки на достаточно большом расстоянии перед командой, непосредственно использующей эти данные, а также команда проверки загрузки (speculative check) непосредственно перед командой, использующей эти данные.**

**На этапе выполнения программы процессор обнаруживает команду предварительной загрузки и пытается заранее загрузить данные из памяти, делая это в тот момент, когда системная шина свободна для обмена.**

# 2. Принципы организации конвейеров команд

**Конвейеризация (pipelining) команд является одним из важнейших способов реализации низкоуровневого параллелизма (уровень машинных команд ILP). Зарождение конвейеров команд было связано с попыткой совмещения стадий предварительной выборки команд и фактического их выполнения. По российским источникам, впервые идея совмещенного выполнения операции в АЛУ и опережающей выборки команд была реализована в 1956г. академиком С.А. Лебедевым в ЭВМ М-20. По зарубежным источникам аналогичная идея была реализована в 1959г. (компьютер Stretch фирмы IBM). В свою очередь, полноценный четырехступенчатый конвейер команд был впервые реализован в 1961г. в вычислительной машине Atlas Манчестерского университета.**

## Сравнение производительности последовательного (бесконвейерного) процессора и процессора с шестиступенчатым конвейером команд




**Производительность конвейера команд, также как и производительность процессора определяется числом команд, выполняемых за единицу времени. Стандартной оценкой этой характеристики является MIPS. Производительность последовательного процессора определяется как:**

$$P_1 = \frac{1}{T},$$

**где *T* – длительность выполнения одной машинной команды**

**Для оценки производительности конвейера команд необходимо учитывать, что, несмотря на то, что длительность выполнения каждой команды по сравнению с последовательным процессором остается прежней, темп схода выполняемых команд с конвейера составляет $\frac{n}{T}$, где *n* – число ступеней конвейера. В связи с этим производительность**

$P_n = \frac{n}{T}$ **n-ступенчатого конвейера в идеале оказывается в *n* раз больше производительности процессора без конвейера.**

**Идеализация приведенной оценки базируется на следующих допущениях:**

1. **длительности всех фаз конвейера одинаковы;**
2. **время переключения фаз конвейера (передачи команды с одной фазы на другую) пренебрежимо мало по сравнению с длительностью самих фаз;**
3. **все фазы конвейера максимально загружены, что может иметь место только на линейных участках программы (естественно, в работе конвейера предполагается, что блок IFU осуществляет свою работу по выборке команд по последовательным адресам).**

**Согласно статистическим данным, длина линейного участка программы составляет 5-8 машинных команд.**

**В реальных условиях производительность конвейера команд оказывается намного ниже по сравнению с идеальным случаем.**

## Основные причины снижения производительности конвейеров команд

**Конфликтные ситуации, имеющие место в конвейерах команд, в англоязычной литературе именуются «hazard» (риск, сбой).**

**В современной литературе в основном описываются три вида конфликтов (рисков), приводящих к снижению производительности конвейеров команд:**

1. **конфликты по ресурсам (структурный риск);**
2. **конфликты по данным;**
3. **конфликты по управлению.**

Конфликты по ресурсам **имеют место, когда несколько блоков конвейера пытаются одновременно использовать один и тот же ресурс. Обычно таким ресурсом является память. Действительно, сразу трем блокам классического конвейера команд (IF, OF, S) может понадобиться одновременное обращение к памяти, что приведет к приостановке конвейера.**

**Однако, следует иметь в виду, что подавляющее большинство обращений к памяти реализуется через кэш. И только в случае достаточно редкого кэш-промаха требуется обращение к ОП. Кроме того, в современных процессорах внутренняя кэш-память (кэш L1) организована по принципу Гарвардской архитектуры, в связи с чем блок выборки команд, обращаясь к кэш-памяти команд, не конфликтует**

**с блоками выборки операндов и записи результатов, которые обращаются к кэш-памяти данных.**

**В свою очередь, для устранения конфликтов при обращении к кэш-памяти данных, оно обычно реализуется как многопортовая (в частном случае - двухпортовая) память, что допускает параллельное обращение к ней по двум разным адресам (простейшая реализация – раздельные порты для чтения и записи).**

**Как отмечается большинством специалистов, конфликты по ресурсам оказывают гораздо меньшее негативное влияние на производительность конвейеров команд по сравнению с другими видами конфликтов.**

Конфликты по данным **являются следствием наличия так называемой зависимости по данным, присущей программам. Эта зависимость имеет место между соседними или близкорасположенными командами и состоит в том, что результат выполнения предыдущей команды программы является операндом или его адресом для последующей команды. Если не учитывать зависимости по данным, то может оказаться, что последующая команда программы использует необновленное (неверное) значение операнда или его адреса, так как произведет выборку из некоторой ячейки памяти или регистра до того момента, как предыдущая команда запишет туда верное значение.**

**Для устранения подобного конфликта необходима приостановка зависящей по данным команды в конвейере до момента завершения записи результата предыдущей командой.**

Конфликты по управлению **обусловлены наличием в программах зависимостей по управлению, которые связаны с командами, изменяющими естественный порядок следования машинных команд программы (по порядку возрастания их адресов). К таким командам относятся команды безусловных и условных переходов, вызовов процедур и возвратов из них, команды циклов и программных прерываний.**

**Конвейер команд осуществляет выборку команд, расположенных по последовательным адресам памяти. Когда в конвейер попадает одна из перечисленных выше команд, факт наличия перехода распознается только на фазе EX. Если переход действительно имеет место, то все предыдущие наработки конвейера команд, связанные с последующими за переходом командами, оказываются неактуальными. Простейшей реакцией на подобную ситуацию является сброс конвейера и его реинициализация, начиная с выборки команды по адресу перехода (это не очень хороший подход, так как сброс конвейера осуществляется через каждые 5-8 команд).**

**Для уменьшения влияния зависимостей по управлению используются различные методы, основными из которых являются**

**методы предсказания переходов. Эти методы разделяются на *статические* и *динамические*. В статических методах предсказание перехода - однонаправленное. Так, например, наиболее вероятным направлением перехода для команды LOOP *cycle* является адрес команды, отмеченной именем *cycle*, то есть продолжение цикла. В свою очередь, переход к следующей команде программы имеет гораздо меньшую вероятность.**

**Динамическое предсказание переходов достаточно часто базируется на использовании в процессоре дополнительного блока BTB (Branch Target Buffer). Основу этого блока составляет ассоциативная кэш-память, входом в которую (тэгом) является адрес команды перехода. При наличии информации об этой команде в BTB на выход выдается наиболее вероятный адрес перехода. Емкость этого буфера в современных процессорах составляет 256, 512 единиц (адресов).**

**Кроме ассоциативной памяти, BTB содержит блок, хранящий предысторию переходов. В первых моделях процессоров Pentium предыстория каждого перехода кодировалась двумя битами, в более поздних моделях - четырьмя, для увеличения вероятности верного предсказания. По результатам исследований блок BTB выдает верный адрес в 90% случаев.**

**Блок BTB появился в первой модели процессора Pentium. Он принимает участие в организации первого этапа конвейера команд (IF), совмещенного с блоком IFU, который в некоторых моделях принято называть PFU (Pre Fetch Unit). Все адреса команд, по которым осуществляется выборка из памяти, поступают в BTB. При кэш-попадании BTB выдает адрес наиболее вероятного перехода.**

**Для некоторых команд (таких как безусловный переход - JMP, вызов процедуры - CALL, возврат из процедуры - RET, аппаратное прерывание - INTn) этот адрес является безальтернативным и поэтому точным. Именно по этому адресу, выбираемому из BTB, осуществляется дальнейшая выборка и, возможно, выполнение последующих команд программы.**

**При отсутствии информации о команде передачи управления в BTB, эта команда переходит на стадию декодирования, на которой определяется ее принадлежность к данному классу.**

**Для команд условного перехода осуществляется статическое предсказание направления перехода и, если предсказано направление по следующему за командой адресу (отсутствие перехода), то конвейер продолжает свою работу по последовательным адресам. Если же предсказан переход или декодируется команда с безальтернативным переходом, то дальнейшая выборка команд приостанавливается до момента формирования адреса перехода и пересылки его в счетчик команд (IP, PC). Во всех случаях информация о новой (впервые выполняемой) команде передачи управления заносится в BTB.**

**Окончательное решение о правильности выбранного направления перехода принимается на фазе исполнения (EX), на которой осуществляется проверка истинности или ложности условия перехода.**

**Неправильное предсказание направления перехода не приводит к ошибочному выполнению программы, так как ни одна из последующих команд программы не могла еще записать результат своего выполнения, однако, неверное предсказание чревато значительными издержками на реинициализацию конвейера по правильному адресу. Эти издержки тем больше, чем больше число ступеней в конвейере. В современных процессорах число ступеней значительно превосходит 5-6 (типично для ранних моделей) и доходит до 20. Архитектура конвейера с числом ступеней, превышающим 6 (как для классического конвейера), называется суперконвейерной (superpipeline architecture) и характеризуется числом ступеней, равным 8-14. Для числа ступеней ≥ 15 – гиперконвейерная архитектура (hyperpipeline architecture).**

**Кроме рассмотренных выше трех видов конфликтов (по ресурсам, по данным, по управлению) еще одной существенной причиной снижения производительности конвейеров команд является *большой разброс длительности выполнения некоторых фаз конвейера для различных машинных команд*. В основном это относится к фазе EX. При попадании "длинной" (по времени выполнения) команды на фазу исполнения она надолго "задерживается", препятствуя продвижению по конвейеру последующих команд программы, что приводит к непроизводительному простою многих блоков конвейера команд.**

**<u>Длина некоторых машинных команд (архитектура P5 – Pentium, Pentium MMX):</u>**

**IMUL/MUL 10-11 тактов**

**IDIV/DIV 17-46 тактов (зависит от разрядности делителя)**

**FMUL 3 такта**

**FDIV 40 тактов**

**FCOS/FSIN 18-124 такта**

**FBSTP * 150 тактов**

**Наиболее эффективным способом борьбы с издержками, описанными выше, является использование межблочных буферов.**

**При использовании буфера между двумя соседними блоками конвейера команд их работа становится во многом независимой друг от друга, так как предыдущий блок может выполнять свою работу над**

---

* пересылка результата из регистра с плавающей точкой в память с преобразованием в десятичную форму

**последовательными командами программы до полного заполнения своего выходного буфера.**

**В свою очередь, последующий блок конвейера, использующий этот буфер как входной, может выполнить свою работу при наличии хотя бы одного элемента команды в этом буфере.**

**Уже в первой реализации простейшего двухступенчатого конвейера команд в процессоре Intel 8086 имелся подобный буфер емкостью 6 байт, называемый очередью команд - IQ (Instruction Queue). В последующих моделях этот буфер сохранился, однако возросла его длина. Примером другого буфера может служить очередь декодированных команд, которая является результатом работы блока конвейера DU (Decode Unit).**

**Еще одной причиной снижения производительности конвейера команд являются запросы аппаратных прерываний (в частности, по вводу/выводу). Вследствие их асинхронного характера по отношению к выполняемой программе каких-либо методов их предсказания и уменьшения степени их влияния предусмотреть невозможно.**

## Основные способы увеличения производительности конвейера команд

**Простейшие способы были рассмотрены ранее в качестве мер, уменьшающих конфликты различных видов.**

1. **Использование буфера BTB на этапе выборки команд для организации спекулятивной выборки и выполнения команд.**

2. **Использование межблочных буферов для сглаживания различий в длительностях фаз конвейера.**

3. **Разделение трудоемких фаз конвейера на более мелкие ступени, т.е. переход к супер- и гипер- конвейерной архитектуре. За счет этого можно увеличить тактовую частоту и, следовательно, производительность процессора. (*Временная диаграмма работы суперконвейера и его сравнение с обычным конвейером - Цилькер и Орлов стр. 445-446).***

4. **Расширение уровней параллелизма в конвейере команд.**

**В обычном линейном конвейере команд, даже в том случае, если он супер- или гипер- могут параллельно выполняться *n* машинных команд (*n* - число ступеней конвейера), каждая на своей фазе конвейера. Принципиально можно использовать 2 подхода для увеличения возможностей ILP (распараллеливания) в конвейере команд:**

- **использование нескольких параллельных конвейеров команд;**

**• использование одного конвейера команд и нескольких параллельно работающих исполнительных (обрабатывающих) блоков на фазе EX линейного конвейера.**

**И тот и другой подходы переводят процессор в ранг суперскалярных (Super Scalar Architecture).**

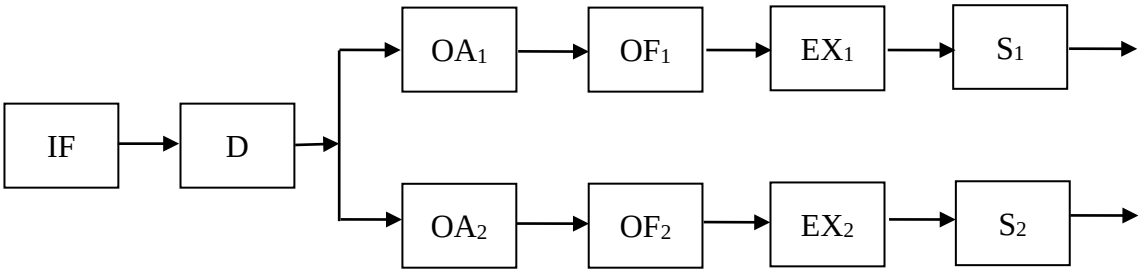



**Рисунок 1 (для первого подхода). Использование двух параллельных конвейеров команд.**

**Подобный подход реализован первой модели процессора Pentium.**

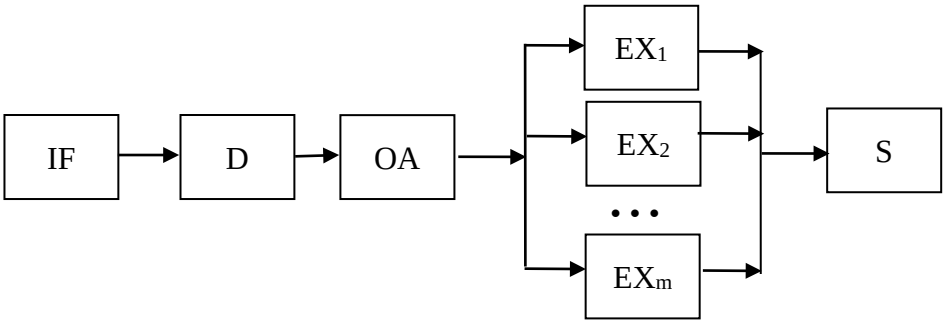



**Рисунок 2 (для второго подхода). Использование одного конвейера команд и нескольких параллельно работающих исполнительных (обрабатывающих) блоков на фазе EX линейного конвейера.**

**Подобный подход впервые реализован фирмой Intel в модели Pentium Pro (архитектура P6).**

**В качестве исполнительных блоков, реализующих параллельно фазу EX для нескольких команд, могут использоваться:**

**- несколько (2-3) целочисленных ALU (IU);**

**- блок FPU;**

**- блоки мультимедийной (векторной) обработки: MMX, SSE, AltiVec, 3DNow! и т.д.**

## Понятия суперскалярной архитектуры и суперскалярных процессоров.

**1. (по Столлингсу) Под суперскалярной реализацией архитектуры процессора понимается такая реализация, в которой обычные машинные команды - арифметики, загрузки, сохранения и условного перехода - могут запускаться на выполнение одновременно и выполняться независимо друг от друга.**

**2. (по Цилькеру и Орлову) Суперскалярным называется центральный процессор, который одновременно выполняет более чем одну скаляруню команду. Это достигается за счет включения в состав ЦП нескольких самостоятельных функциональных (исполнительных) блоков, каждый из которых отвечает за свой класс операций и может присутствовать в процессоре в нескольких экземплярах. Суперскалярность предполагает параллельную работу максимального числа исполнительных блоков, что возможно лишь при одновременном выполнении нескольких скалярных команд. Последнее условие хорошо сочетается с конвейерной обработкой, при этом желательно, чтобы в суперскалярном процессоре было несколько конвейеров (например, 2 или 3).**

**3. (по Мелехину и Павловскому) Процессор, содержащий в своем составе несколько операционных блоков, которые обеспечивают одновременное выполнение более одной скалярной команды, называется суперскалярным процессором.**

**Термин "суперскалярный" впервые был использован в 1987 году в техническом докладе сотрудников фирмы IBM, посвященном увеличению производительности RISC-процессоров.**

**Термин "скалярный" используется для того, чтобы подчеркнуть отличие процессоров этого типа от векторных процессоров.**

# Процессор Pentium (архитектура P5)

## Основные архитектурные особенности (по сравнению с i486)

**1) *Суперскалярная архитектура*. Использование двух конвейеров команд обеспечивает возможность выполнения более 1й машинной команды в каждом такте процессора. В модели i486 - линейный конвейер.**

**2) *Разделение внутрикристалльной кэш-памяти уровня L1 на 2 блока* - кэш команд и кэш данных по 8 Кбайт каждый.**

**В i486 - единый кэш для команд и данных.**

**3) *Использование блока BTB для динамического предсказания перехода*.**

**В i486 BTB отсутствует.**

**4) *Расширение шины данных до 64 бит*.**

**В i486 ША и ШД – 32 бита.**

**5) *Существенное ускорение операций FPU за счет его конвейеризации*. (Производительность FPU в 2-10 раз выше, чем в процессоре i486).**

**Обрабатывающие блоки FPU (в первую очередь умножитель и делитель, во вторую - сумматор и вычитатель) реализованы как конвейер операций. Конвейеры операций используются на ступени EX конвейера команд и хорошо сочетаются с последними. При этом предполагается, что выполнение типовой операции разделяется на ряд последовательных этапов, каждый из которых реализуется самостоятельным блоком операционного устройства. Например, в конвейерном устройстве сложения/вычитания FPU можно выделить 4 последовательных ступени и, соответственно, самостоятельных блоков для реализации таких этапов, как сравнение порядков, уравнивание порядков, сложение мантисс, нормализация результата.**

**Конвейеры операций дают максимальную эффективность при обработке векторных данных.**

**6) *Использование нового режима процессора* - режима системного управления *SMM (System Management Mount)*. Переключение в этот режим, в частности, обеспечивает перевод системы в состояние пониженного энергопотребления.**

**7) *Введение средств поддержки мультипроцессорности* (процессор Pentium как элемент SMP-системы (*Symmetric MultiProcessing*)).**

**В средства поддержки мультипроцессорности входят:**

**• внутрикристальный локальный контроллер прерываний для обеспечения совместной работы нескольких процессоров;**

**• двухвходовой (двухканальный) контроллер кэш-памяти второго уровня, позволяющий двум процессорам совместно использовать общую кэш-память;**

**• реализация протокола MESI (Modified Exclusive Shared Invalid - 4 возможных состояния блока (строки) кэш памяти) во внутренней кэш-памяти данных для поддержки когерентности (согласованности) содержимого кэшей многих процессоров SMP-системы.**

**8) *Использование средств мониторинга производительности*.**

## Pairing Rules (правила спаривания)

Все команды процессора разделяются на 4 класса:

**1. NP – Non Paring – команды не могут спариваться ни при каких условиях и выполняются только в основном конвейере (U-конвейере). К таким командам относятся:**

**- команды сдвигов со счетчиком в регистре CL;**

**- длинные арифметические команды (MUL, DIV);**

**- большинство команд FPU;**

**- межсегментные команды типа CALL FAR, PUSH sr;**

**- дополнительные команды типа RET, ENTER, PUSHA, MOVS, STOS, LOOPNZ и т.п.**

**2. UV – команды этого класса могут спариваться, направляясь в любой из конвейеров – U или V. К ним относятся:**

**- большинство простых арифметических и логических команд, таких как ADD, CMD, INC, XOR, TEST и т.п. В данном контексте под простыми понимаются команды, требующие для своего выполнения не более четырех тактов CPU;**

**- стековые команды с регистровым операндом: PUSH reg, POP reg.**

**3. PU – команды этого класса не могут исполняться в V-конвейере, но могут спариваться с другими командами, исполняемыми параллельно в V-конвейере. К этому классу относятся:**

**- арифметические команды с переносом или заемом, типа ADC, SBB;**

**- команды сдвига с непосредственным операндом;**

**- команды с префиксами повторения и/или замены сегмента;**

**- некоторые FPU-команды (могут спариваться только с командой обмена FXCH) – FADD, FMUL, FDIV, FLD.**

**4. PV – команды этого класса могут выполняться в любом из конвейеров, но имеют возможность спариваться с другими командами только когда направляются в V-конвейер. К этому классу относятся:**

**- команды внутрисегментной передачи управления (JMP near, CALL near, Jcc);**

**- команда FXCH.**

### Условия спаривания (Paring Rules)

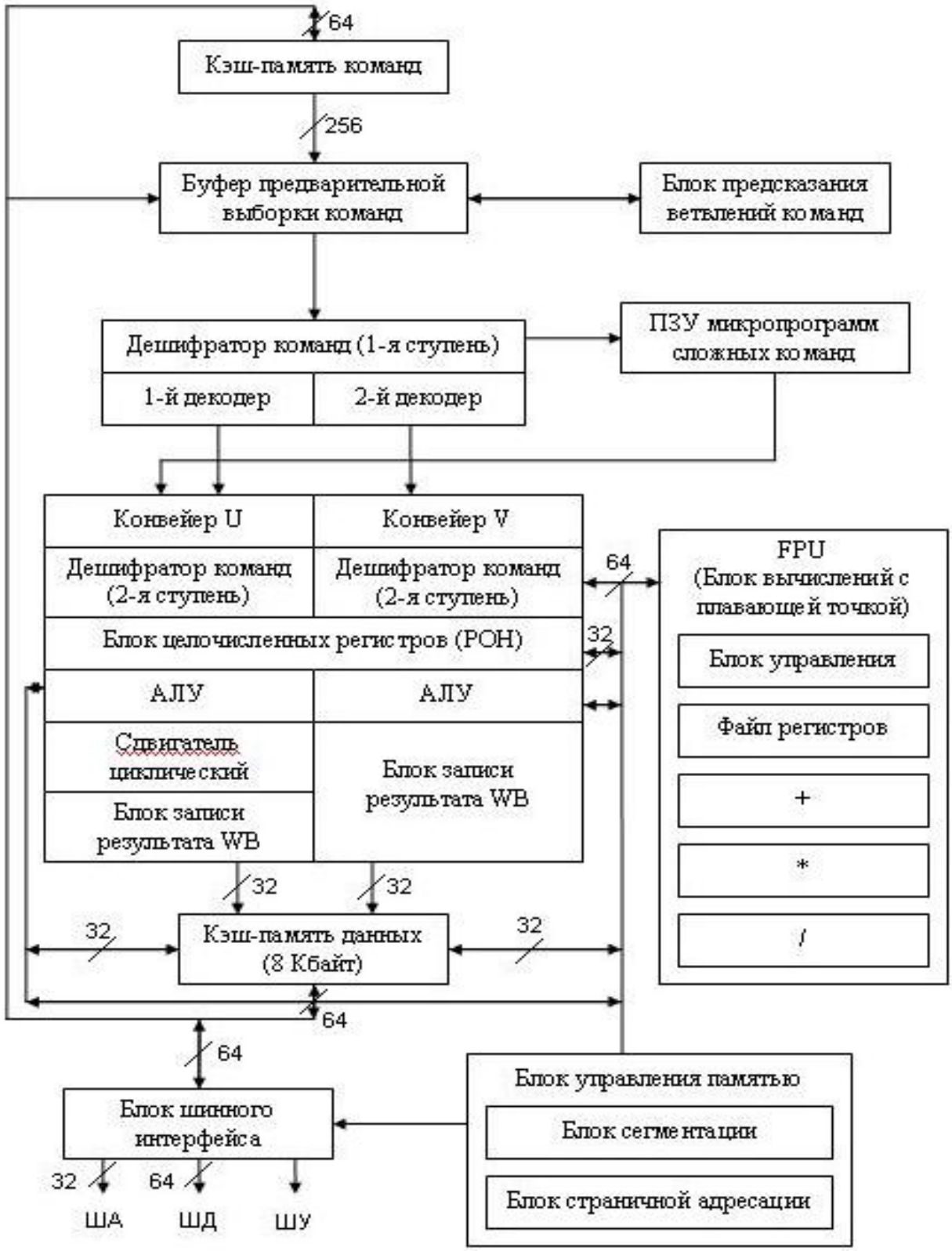
**Условием одновременного выполнения двух команд является их принадлежность к соответствующим классам: команда, которая направляется в U-конвейер, должна принадлежать классу UV или PU, а команда, которая направляется в V-конвейер – классу UV или PV.**

**Вторым важным условием спаривания команд является их взаимная независимость по данным. Спаривание двух соседних команд**

**невозможно, если какой-либо регистр служит операндом-приемником для первой команды пары и операндом-источником для второй команды, или если обе команды используют один и тот же регистр в качестве операнда-приемника. Использование одного и того же регистра в качестве операнда-источника для двух соседних команд или в качестве источника для первой команды и приемника для второй не препятствует спариванию этих команд.**

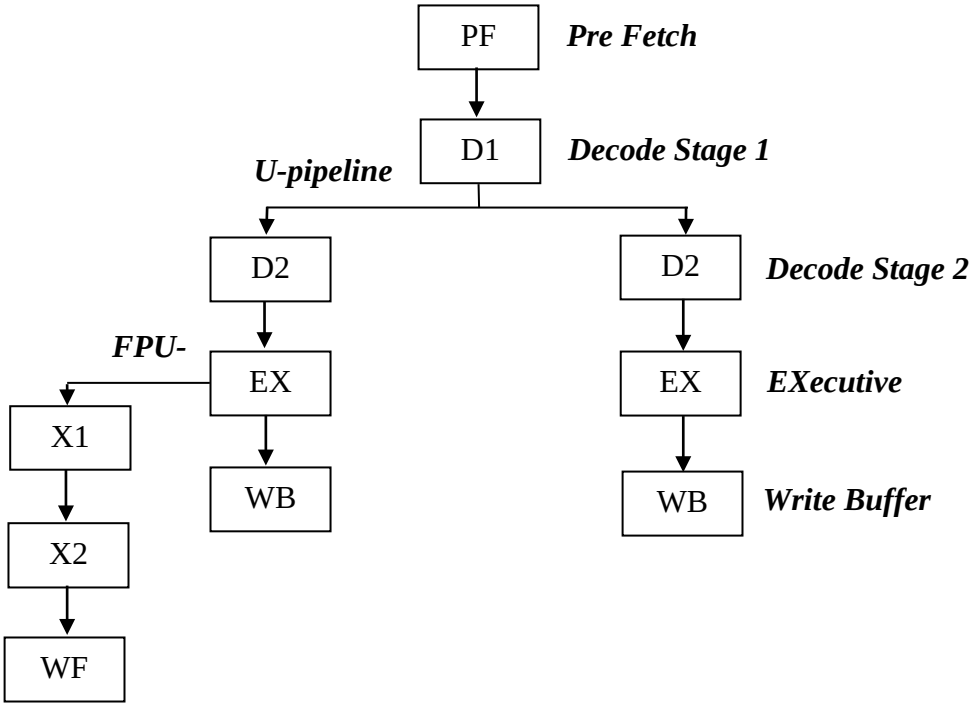
Упрощенная схема процессора

**Из учебника В.Ф. Мелехина и Е.Г. Павловского «Вычислительные машины, системы и сети»:**



## Организация конвейера команд

Упрощенная схема конвейера команд




**Конвейер команд процессора Pentium является пятиступенчатым, как и в предыдущей модели i486. Начиная со второй фазы декодирования (D2), конвейер становится параллельным, разделяясь на основной (U-конвейер) и вспомогательный (V-конвейер).**

**U-конвейер может выполнять любые целочисленные команды и команды FPU. Имеется ответвление от фазы EX для трехступенчатого FPU-конвейера. V-конвейер может выполнять простые целочисленные команды и команду обмена FPU – FXCH.**

**Выделение команды FXCH для спаривания с другими командами FPU связано с довольно частым ее использованием для перестановки данных в регистрах FPU. Регистры данных FPU (8 регистров по 80 бит каждый) организованы как регистровый стек. Команды FPU по умолчанию используют операнды из вершины стека и помещают результат операции также в вершину стека. В отличие от организации стека в памяти, использование операнда из вершины стека, а также запись результата в вершину не сопровождаются инкрементом или декрементом указателя вершины стека. Этот указатель является трехбитным и размещается в управляющем регистре FPU – CR.**

## Описание фаз конвейера

**1)** Фаза PF (Pre Fetch) **– предварительная выборка команд. На стадии PF процессор осуществляет выборку команд программы из кэш-памяти команд или (при кэш-промахе) из ОП.**

**Блок предварительной выборки (PFU) содержит два буфера емкостью по 32 байта каждый (емкость буфера совпадает с длиной блока (строки) кэш-памяти).**

**В любой момент времени выборка команд производится только в один из буферов. При этом выборка команд осуществляется по последовательным адресам. Если в выбираемом фрагменте машинной программы встречается команда ветвления, то определение дальнейшего направления выборки команд осуществляется с помощью BTB. BTB выдает адрес наиболее вероятного перехода на основании двухбитовой предыстории переходов.**

**Если предсказание соответствует случаю отсутствия перехода, то последовательная выборка команд продолжается. Если же имеет место предсказанный переход (нарушение последовательности адресов выполняемых команд), то происходит переключение на другой буфер, в который осуществляется выборка команд по новому адресу, полученному из BTB. Окончательное решение о правильности выбранного направления перехода переносится на стадию EX команды условного перехода, на которой, собственно, и осуществляется анализ заданного в команде условия перехода.**

**При неправильном предсказании направления перехода содержимое буферов и всего конвейера аннулируется, и выборка команд начинается с необходимого адреса. При этом осуществляется корректировка соответствующей строки BTB.**

**2)** Фаза D1 (Decode Stage 1). **Эта фаза реализуется отдельным блоком конвейера – декодером команд (DU – Decode Unit). Декодер из актуального буфера команд (одного из двух) выбирает 2 последовательные команды и определяет возможность их параллельного исполнения в соответствии с их парабельностью (возможностью спаривания) и независимостью по данным. На этой же фазе осуществляется преобразование кодов команд во внутренний формат, используемый устройством управления (CU – Control Unit) для интерпретации машинных команд в последовательность микрокоманд. Для сложных команд (реализуемых по микропрограмме) во внутреннем формате задается начальный адрес ПЗУ (ROM) микропрограмм, с которого начинается микропрограмма интерпретации этой команды.**

**3)** Фаза D2 (Decode Stage 2). **Фактически, на фазе D2 производится генерация (вычисление) адресов операндов и их выборка из памяти. В связи с тем, что фаза D2 является первой из параллельных фаз, то она**

**реализуется двумя идентичными блоками AGU (Address Generate Unit) U- и V-конвейеров.**

**Для возможности обеспечения параллельной выборки операндов двумя блоками AGU из кэш-памяти данных, она реализована как двухпортовая память.**

**4)** Фаза исполнения EX. **Эта фаза реализуется в сдвоенном блоке целочисленного ALU (IU). В связи с тем, что блок циклического сдвига не дублирован, а имеется только в U-конвейере, спариваемые команды обязательно относятся к классу PU.**

**5)** Фаза WB (Write Buffer). **На последней ступени конвейера осуществляется буферизованная запись в память. Буферы записи используются для увеличения производительности при последовательных операциях записи в память. Пересылка результата операции в буфер позволяет продвигаться по конвейеру следующей команде, не дожидаясь фактической записи результата предыдущей команды в память.**

**На внешнюю шину данных содержимое буферов направляется по мере ее освобождения от других транзактов (пересылок), но именно в том порядке, в котором осуществлялась запись в буфер.**

## Особенности FPU-конвейера

**В технической литературе отмечается, что FPU-конвейер состоит из 7 фаз (по рисунку из предыдущей темы – с учетом того, что он является составной частью U-конвейера).**

На фазе X1 **осуществляется преобразование данных к внутреннему формату FPU. При этом предполагается, что чтение операндов из памяти осуществляется на фазе EX (только для команд FPU).**

На фазе X2 **производится фактическое исполнение команд.**

На фазе WF **осуществляется округление результата операции и запись его в регистр.**

**В отличие от целочисленных команд, которые целиком выполняются на стадии EX U- или V-конвейера, команды FPU начинают выполняться на стадии EX U-конвейера (кроме команды FXCH, которая может быть обработана в V-конвейере), а затем переходят на стадию X1 конвейера FPU, где продолжается их выполнение. FPU-команды не могут спариваться с целочисленными командами в начале конвейра, но после того, как FPU-команда перейдет на стадию X1, следующие за ней по программе целочисленные команды смогут продвигаться дальше по конвейеру.**

**Например, если запустить в конвейер сначала команду FMUL, то следующие за ней целочисленные команды смогут продолжить выполняться параллельно с FMUL. Если же сначала запустить в**

**конвейер целочисленную команду MUL,то она «застрянет» на стадии EX на несколько тактов, блокируя дальнейшее продвижение следующих за ней команд по обоим конвейерам.**

**Для сравнения, команде FMUL для своего исполнения требуется 3 такта, а целочисленной команде MUL – 10 тактов.**

## Процессор Pentium Pro (первая модель с архитектурой P6)

**Основные особенности модели:**

**1. *Суперконвейерная и суперскалярная архитектура*. Суперконвейерность проявляется в том, что число ступеней конвейера команд по разным источникам равно 10-14.**

**Реализация суперскалярной архитектуры основана на втором принципе, в соответствии с которым используется единственный конвейер команд с распараллеливанием фазы исполнения путем использования большого числа обрабатывающих (исполнительных) блоков, способных параллельно выполнять несколько команд программы.**

**Существенной особенностью суперскалярной обработки в процессорах с архитектурой P6 является ее реализация не на уровне машинных команд, а на уровне микрокоманд. В связи с этим, во многих источниках отмечается, что процессоры с этой архитектурой, являясь по сути CISC-процессорами, имеют в своем составе RISC-ядро.***

**2. *Реализация кэш-памяти второго уровня (кэш L2) в одном корпусе с CPU.***

**Кристалл кэш-памяти L2 содержит 15,5 млн. транзисторов для емкости 256 Кбайт или 31 млн. транзисторов для емкости 512 Кбайт, в то время, как кристалл CPU содержит всего (включая кэш L1) 5,5 млн. транзисторов. Основной причиной такого объединения является увеличение производительности процессора.**

**Кэш L2 связан с CPU специально выделенной шиной шириной 64 бита и работает на той же тактовой частоте, что и CPU.**

**Подобный подход получил дальнейшее распространение в виде так называемой двойной независимой шины *DIB (Dual Independent Bus)*. В соответствии с концепцией DIP для связи CPU с памятью и системной шиной используется *FSB (Front Side Bus)*, в то время как для связи CPU с кэшем L2 используется *BSB (Back Side Bus)*.**

## Основные особенности реализации суперскалярной обработки в процессорах P6

* По мнению Павла Семеновича то, что называется RISC-ядром – это обычный микрокомандный уровень, без которого не обойтись в CISC-процессорах при реализации сколь-нибудь сложных машинных команд.

**Основная идея состоит в следующем: суперскалярный процессор извлекает из памяти несколько команд, а затем пытается отыскать среди соседних команд такие, которые не зависят друг от друга и, следовательно, могут выполняться параллельно при условии обеспечения этой параллельности аппаратными средствами (наличие большого числа исполнительных блоков).**

**Если входные данные (операнды) очередной команды зависят от выходных данных (результата) одной из предшествующих команд, то последующая команда должна оставаться в стадии ожидания фазы выполнения до момента разрешения конфликта по данным. После того, как процессор проанализирует наличие подобных зависимостей между выбранными командами, он может приступить к их выполнению, причем порядок выполнения команд может отличаться от порядка их следования в машинном коде программы. Такая особенность суперскалярных процессоров называется *Out-of-Order Execution* (исполнение с изменением порядка команд).**

**По мнению Столлингса основными аппаратными методами, которые используются в суперскалярных процессорах, являются:**

**1. дублирование ресурсов;**

**2. свободный порядок запуска команд на исполнение;**

**3. переименование регистров.**

**1)** Дублирование ресурсов **позволяет сокращать так называемые конфликты по ресурсам. Типичными примерами такого дублирования могут служить использование двух или трех блоков ALU (IU) в составе процессора, а также использование многопортовой кэш-памяти (в основном – L1).**

**2)** Свободный порядок запуска команд на выполнение**. При прямом порядке запуска команд процессор декодирует и далее выполняет команды в порядке их следования в машинной программе, пока не обнаружит зависимость между очередной командой и предыдущими или конфликт по ресурсам. После обнаружения конфликта какого-либо вида дальнейшие команды не декодируются до тех пор, пока обнаруженный конфликт не будет устранен. При использовании стратегии прямого порядка запуска команд процессор лишен возможности просмотра последующих команд программы, которые, в принципе, можно было запустить в конвейер в обход конфликта, поскольку они не зависят от команд, уже находящихся в конвейере. В соответствии с этим для уменьшения простоев конвейера возникла идея реализации стратегии свободного порядка запуска команд на фазу исполнения. Для этой цели между фазами декодирования и исполнения помещается промежуточный буфер, который обычно называют *окном команд* (Instruction Window). Каждая декодированная команда помещается в это окно, и до тех пор, пока буфер в виде окна не заполнится, процессор**

**может продолжать выборку и декодирование очередных команд программы. Выдача команд из буфера на исполнение определяется не столько последовательностью их поступления в буфер, сколько мерой готовности этих команд к выполнению. Иначе говоря, любая команда, для которой определены операнды и свободен требуемый для исполнения функциональный блок, немедленно выдается из окна на фазу исполнения. Таким образом, организация свободного порядка запуска команд позволяет просматривать программу вперед и отыскивать в ней команды, которые можно с опережением запустить на выполнение. Единственным ограничением является необходимость сохранения корректности результатов внеочередного выполнения команд. Это означает, что очередная команда программы не может быть запущена на исполнение, если имеется какая-либо зависимость этой команды от ранее запущенных команд или конфликт в использовании ресурсов.**

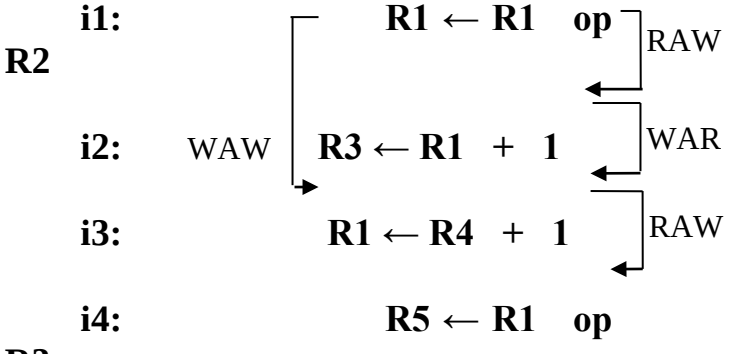
**Основным же отличием свободного порядка запуска команд от прямого является значительное расширение числа команд, из которых производится выбор на выполнение, за счет чего уменьшаются простои конвейера и, следовательно, увеличивается его производительность.**

**При использовании стратегии свободного порядка запуска команд возникает необходимость анализировать еще один вид зависимости по данным, который принято называть обратной зависимостью (antidependence), в отличие от рассмотренной ранее прямой зависимости.**

**Если прямая зависимость относится к виду RAW – Read After Write, то обратная зависимость относится к виду WAR – Write After Read. Кроме того, необходимо учитывать и зависимость вида WAW – Write After Write. Последовательность RAR – Read After Read никогда не вызывает конфликтов.**

**Пример зависимостей из Столлингса:**

**Команды:**

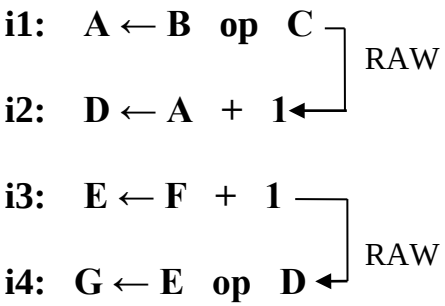



**op – произвольная операция над двумя регистровыми операндами.**

**Для устранения обратных зависимостей (WAR и WAW) наиболее эффективным методом является переименование регистров (*Renaming Registers*).**

**3) По сути,** метод переименования регистров **можно считать расширением метода дублирования ресурсов в отношении регистровой памяти процессора. Суть метода состоит в динамическом отображении логических регистров, имена (адреса) которых формируются компилятором на физические регистры. Распределение физических регистров и их отождествление с логическими реализуется на аппаратном уровне. При этом, как правило, с одним логическим регистром может быть связано несколько физических регистров, содержимое каждого из которых соответствует значению некоторой программной переменной в один из моментов выполнения программы.**

**Рассмотренный выше фрагмент после переименования регистров принимает вид:**




**Из примера видно, что после переименования регистров появляется возможность выполнения команды i3 раньше, чем i2 и даже i1, с которыми команда i3 имела зависимость по данным.**

**Естественно, реализация метода переименования регистров предполагает ведение учета свободных физических регистров и возвращение в этот список освобождающихся регистров.**

## Принципы организации суперскалярной архитектуры в процессоре Pentium Pro (P6)

Процесс выполнения программы выглядит следующим образом:

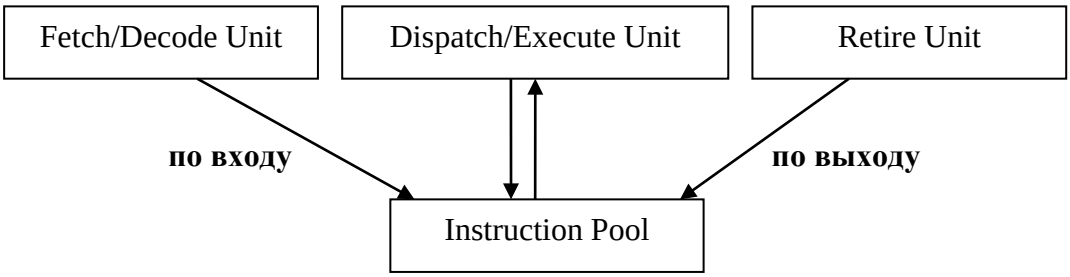
**1. процессор извлекает команды из памяти в прямом порядке;**

**2. каждая команда преобразуется в последовательность микрокоманд фиксированной длины (подобие RISC-команд);**

**3. микрокоманды выполняются в свободном порядке;**

**4. процессор фиксирует результаты выполнения микрокоманд в регистрах и памяти в том порядке, который соответствует исходному потоку команд.**

**В соответствии с описанным порядком в ядре процессора Pentium Pro выделяются 3 отдельных независимых блока (устройства), связанные с конвейером команд.**




- Первый блок (Fetch/Decode Unit) **выборки и декодирования реализует начальные ступени конвейера команд, обрабатывая команды в прямом порядке (In-Order Front End).**
- Средний блок (Dispatch/Execute Unit) **диспетчирования и выполнения образует ядро исполнения с изменением последовательности (Out-of-Order Core).**
  **Этот блок является основным для фазы собственно исполнения команд.**
- Последний блок (Retire Unit)**, называемый блоком отката, производит выдачу результатов выполнения команд в произвольном порядке в соответствии с прямым порядком их следования в программе.**

## Блок выборки-декодирования

**Этап выборки команд разделен на 3 ступени (фазы) и реализуется последовательными блоками конвейера IFU1, IFU2, IFU3 (Instruction Fetch Unit).**

IFU1 **извлекает из кэш-памяти команд строку объемом 32 байта и сохраняет её в собственном буфере.**

IFU2 **принимает из буфера IFU1 порции по 16 байт. Основной функцией IFU2 является выделение страниц между командами. При обнаружении команды условного перехода (IFU2 осуществляет частичное декодирование), ее адрес пересылается в BTB.**

IFU3 **принимает из IFU2 порции по 16 байт и на основе частичного декодирования производит анализ стоимости команд для выделения для них подходящих декодеров с целью эффективной загрузки следующего блока конвейера.**

**Этап декодирования разделен на 2 ступени (фазы) и реализуется последовательными блоками конвейера ID1 и ID2.**

**Блок ID1, реализующий первую фазу декодирования, может параллельно обрабатывать 3 машинных команды (по числу декодеров, входящих в состав этого блока). В блоке ID1 каждая машинная команда преобразуется в последовательность микрокоманд, каждая из которых может состоять из одной, двух, трех или четырех микрокоманд. Микрокоманда имеет длину 118 бит. По Столлингсу, микрокоманда напоминает команду RISC-процессора, однако она значительно длиннее.**

**После фазы конвейера ID1 единицей обработки в нем уже является не команда, а микрокоманда. В связи с этим многие специалисты утверждают, что в архитектуре процессора P6 реализовано так называемое RISC-ядро.**

**Только один из трех декодеров блока ID1 может обрабатывать сложные машинные команды, которые преобразуются в две, три или четыре микрокоманды. Два других декодера могут обрабатывать только простые команды, преобразующиеся в одну единственную микрокоманду. Для того, чтобы команды из буфера IFU2 были переданы на соответствующие декодеры блока ID1, блок IFU3 переставляет команды таким образом, чтобы первая из них была сложная, а вторая и третья – простые. Если все три команды в буфере простые, то никаких перестановок от IFU3 не требуется (декодер сложных команд может с успехом декодировать и простые команды). Если же в буфере содержится более одной сложной команды, они распределяются таким образом, чтобы на второй и третий декодеры попадали только простые команды.**

**В системе команд процессоров с архитектурой P6 имеется достаточно большое число команд, для реализации которых требуется более четырех микрокоманд. При декодировании таких команд (естественно, через декодер сложных команд) реализуется обращение к блоку микропрограммной памяти, именуемому *MIS* (*Microcode Instruction Sequencer* – последовательность микрокода) из которого выбираются последовательности микрокоманд, представляющие алгоритм реализации сложных команд. Сформированная блоками ID1 и MIS последовательность микрокоманд передается в блок ID2 (вторая фаза декодирования).**

**В связи с использованием трех параллельных декодеров команд на фазе ID1 обеспечивается возможность параллельного декодирования трех машинных команд в одном такте. В соответствии с этим (по мнению Д.П.С.) конвейер команд процессоров P6 достаточно часто называют трехпотоковым.**

**Блок ID2 образует очередь микрокоманд в соответствии с исходным порядком следования команд в программе и включает в себя буфер на 6 микрокоманд.**

**Следующим блоком конвейера является блок распределения регистров *RAT (Register Allocation (Alias) Table)*, функцией которого является назначение физических регистров логическими. Количество физических регистров равно 40, они объединяются в регистровый файл с наименованием *RRF* (*Retirement Register File* – регистровый файл отката). При реализации этой фазы конвейера в микрокоманде адрес логического регистра заменяется адресом физического, то есть выполняется функция переименования регистров. Трансформированные микрокоманды пересылаются в следующий блок конвейера, называемый *ROB* (*Re-Order Buffer* – блок изменения последовательности). В состав ROB входят 40 регистров, образующих циклический буфер, каждый из которых предназначен для хранения одной микрокоманды.**

**Буфер ROB можно считать аналогом Instruction Window (окно команд). В каждом регистре буфера хранится следующая информация о микрокоманде:**

**1. состояние микрокоманды;**

**2. адрес микрокоманды программы, «породившей» данную микрокоманду;**

**3. собственно код микрокоманды (определяет действия над операндами);**

**4. операнды, используемые микрокомандой (если операнды поставляются в микрокоманду другими микрокомандами, то в данной микрокоманде лишь резервируется место для них и сбрасывается соответствующий флаг готовности операнда, тем самым порядок выполнения микрокоманд задается не только и не столько порядком их следования по коду программы, сколько готовность операндов для них);**

**5. результат (пересылается в ROB только после завершения исполнения микрокоманды).**

**Микрокоманды, находящиеся в ROB, могут идентифицироваться одним из семи возможных состояний:**

1. **RS – Ready for Scheduling;**
2. **SD – Scheduling;**
3. **DP – Dispatching;**
4. **EX – Executing;**
5. **WB – Writing Back to ROB;**
6. **RR – Ready to Retire;**
7. **RT – Retining.**

**Микрокоманды поступают в ROB в том порядке, который задается исходной программой, но на исполнение они могут направляться в свободном порядке. Критерием принятия микрокоманды на исполнение**

**является готовность данных (операндов) и доступность необходимых ресурсов (исполнительных (обрабатывающих) блоков).**

**Определением порядка запуска команд на исполнение занимается блок *RS* (*Reservation Station* – блок распределения). RS выбирает микрокоманды, готовые к исполнению, из буфера ROB, распределяет их между исполнительными устройствами и записывает результаты исполнения обратно в регистры ROB.**

**За один такт RS может обеспечить анализ и распределение до пяти микрокоманд. Число 5 определяется наличием пяти портов, которые связывают RS с исполнительными блоками.**

**В Pentium Pro используется следующее распределение исполнительных блоков (устройств) по портам:**

- ***порты 0 и 1*** **используются для микрокоманд обработки данных и обеспечивают связь с блоками ALU и FPU;**
- ***порты 2, 3 и 4*** **обеспечивают обращение к памяти.**

**После завершения микрокоманды соответствующий элемент ROB обновляется, а освободившийся исполнительный блок становится доступным для обработки ожидающих микрокоманд.**

**Заключительные две фазы конвейера осуществляют фиксацию результатов выполнения микрокоманд в нужном порядке. Для этой цели осуществляется просмотр буфера ROB (его хвостовой части) на предмет поиска микрокоманд, завершивших исполнение и готовых к записи результата в память и последующему удалению из ROB.**

**В связи с тем, что ROB является буфером FIFO, то и удаляются микрокоманды всегда в том порядке, в каком они поступали в буфер и, следовательно, в каком они расположены в программе.**

# Конспект

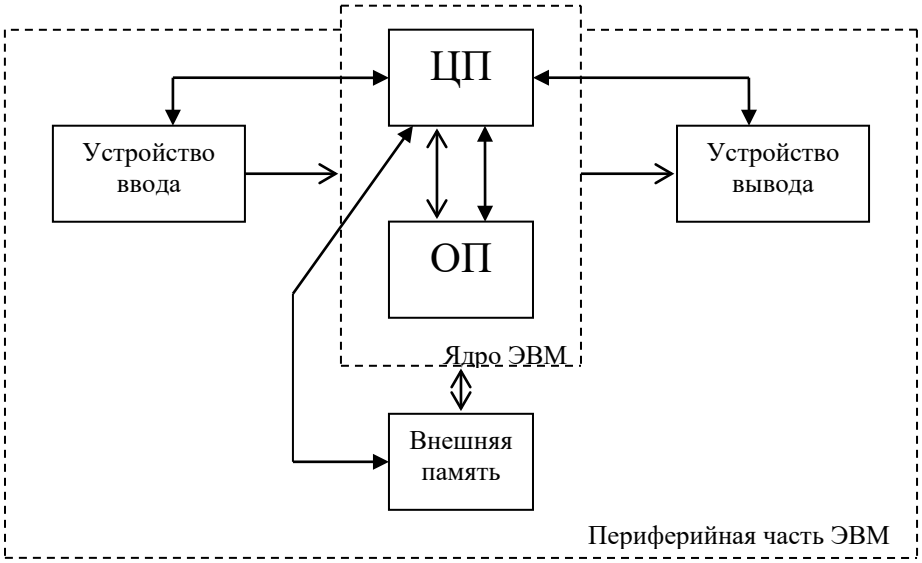
# по курсу

# «Организация ЭВМ и систем»

# Структурная организация ЭВМ

# 1. Структурная организация ЭВМ.

## Упрощенная (каноническая) структура компьютера. Основные устройства компьютера и их характеристика.



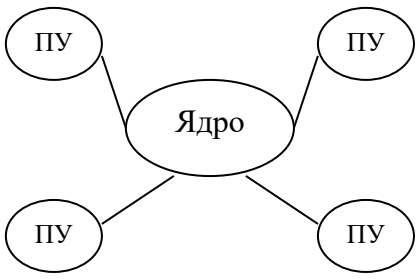

Связь по управлению

Линия передачи данных

По линиям управления от ЦП к остальным устройствам компьютера передаются управляющие сигналы. С помощью этих сигналов инициируются соответствующие действия. В свою очередь от устройства компьютера к ЦП по этим линиям передаются сигналы о состоянии устройств (сигналы о готовности устройств к обмену).

Во многих учебниках и монографиях подобная структура носит название структуры неймановской ЭВМ.

В классификации структур ЭВМ по их топологии подобная структура относится к классу структур с топологией звезда:




ПУ – периферийное устройство

Основные устройства компьютера, которыми являются ЦП и ОП, образуют так называемую центральную часть ЭВМ – ядро ЭВМ. Связь между ядром ЭВМ и её периферийной частью реализуется на основе аппаратных интерфейсов.

*Основное устройство компьютеров* – ЦП (CPU) выполняет двойную функцию: с одной стороны ЦП является обрабатывающим устройством, т.к. выполняет функции по обработке данных в соответствии с заданной программой; с другой стороны ЦП является управляющим устройством, в связи с тем, что на него возлагаются функции: во-первых, по управлению программой, во-вторых, по управлению остальными устройствами ЭВМ.

Управление периферийными устройствами со стороны ЦП, как правило, сводится к обеспечению реакции на запросы ПУ и к организации обмена между ПУ и ядром ЭВМ. Основными устройствами (блоками) ЦП являются, во-первых, *АЛУ (ALU),* во-вторых, *устройство управления (CU).*

АЛУ реализует функцию ЦУ по обработке и предназначено для выполнения арифметических и логических операций над целыми числами, логическими значениями и символьными данными. В некоторых современных моделях компьютеров это устройство называется IU для того, чтобы подчеркнуть основной тип обрабатываемых данных.

Функцией устройства управления (УУ) является выработка сигналов управления, с помощью которых осуществляется выполнение элементарных операций в АЛУ или периферийных устройствах, которые называются *микрооперациями*.

УУ, во-первых, обеспечивает выполнение команд программы, реализуя выборку команд из памяти, их декодирование, формирование адресов операндов и их выборку из памяти, настройку АЛУ на выполнение заданной операции и запись результата операции в память. С другой стороны УУ реализует функции по управлению взаимодействия периферийных устройств ЭВМ с его ядром, обеспечивая реакцию на запросы ПУ по организации обмена между ними и памятью (ОП). Для обеспечения быстрой реакции на запросы ПУ в ЦП используется система, представляющая собой комплекс аппаратных и программных средств.

*Аппаратные средства* системы прерываний в ПК реализуется с помощью специализированных микросхем PIC, а *программные* - обработчиками прерываний, входящими в состав операционной системы (ОС).

Кроме АЛУ и УУ в состав ЦП входит *внутренняя регистровая память*. Регистры ЦП обычно разделяют на *программно-доступные* и *программно-недоступные.*

Программно-доступные обычно рассматриваются как программная модель процессора. Например, в базовой модели процессора Intel 8086 – 14 16-разрядных

регистров, из них 8-РОН, 4-сегментных, FR – флаговый регистр и IP-указатель команд. Типичными примерами программно-недоступных регистров могут служить:

- IR (РК) - регистр команд (instruction register);
- MAR(РА) – регистр адреса (memory address register);
- MDR (РД) - регистр данных (memory data register).

Последние два регистра входят в состав интерфейса и служат для обмена между ЦП и ОП.

Основными характеристиками ЦП являются:

1. *Тактовая частота* в некотором смысле характеризует *быстродействие* ЦП. Быстродействие оценивается числом операций в секунду. Величина обратная тактовой частоте представляет собой *длительность одного такта процессора* $\tau = 1/f$.

Для RISC процессоров тактовую частоту можно отождествить с *пиковой (предельной) производительностью* при условии, что в процессоре отсутствуют средства суперскалярной обработки. Это утверждение базируется на свойстве RISC архитектуры: выполнение подавляющего большинства машинных команд за 1 такт процессора. Таким образом, тактовая частота 1ГГц для RISC процессора без средств *суперскалярной обработки* соответствует производительности 1000 MIPS (Million Instruction Per Second).

В простейшем смысле под ***суперскалярной обработкой*** понимается возможность выполнения ЦП более одной машинной команды в каждом такте. Суперскалярность обеспечивается способностью обрабатывающих устройств в ЦП функционировать параллельно, обеспечивая тем самым возможность схода с конвейера команд в каждом такте более одной готовой команды.

Простейшим способом реализации суперскалярной обработки является использование двух параллельных конвейеров команд как, например, в процессоре Intel Pentium.

Все современные универсальные процессоры имеют средства суперскалярной обработки, с учетом этого для преобразования тактовой частоты в производительность в MIPS`ах необходимо ее умножить на коэффициент суперскалярности, определяющий среднее число машинных команд завершающихся в каждом такте процессора.

При оценке пиковой производительности ЦП и в принципе всего компьютера в целом кроме MIPS используется также MFLOPS (Million Floating Point Operation Per Second) и его производные GFLOPS и TFLOPS. Именно оценка производительности во FLOPS`ах является основанием для формирования рейтинга TOP 500 самых высокопроизводительных вычислительных систем.

2. *Разрядность CPU* определяется максимальной разрядностью обрабатываемых в АЛУ данных. Современные модели высокопроизводительных процессоров являются 64-х разрядными. Из процессоров фирмы Intel к таким относится Itanium.

3. В принципе существует 2 подхода к оценке *мощности системы команд*. В первом из них мощность определяется количеством уникальных мнемоник на Assembler`e. При втором подходе мощность оценивается числом разнообразных машинных кодов команд с учетом различных кодов команд и режимов адресации. Для примера мощность системы команд базовой модели Intel по числу мнемоник имеет значение 113, а по числу разнообразных машинных кодов ~3800. В дальнейшем будем использовать первый подход к оценке мощности. Именно по этой характеристике осуществляется деление процессоров и соответственно компьютеров на 2 класса: CISC и RISC.

Мощность системы команд в современных CISC процессорах составляет ~350-450, а RISC процессорах ~100-150.

Для примера рассмотрим состав системы команд процессора Pentium IV:

1. Команды CPU:
   - команды пересылки данных и адресов (их порядка 40);
   - арифметические (~20);
   - логические (5);
   - сдвиги (10);
   - побитовой обработки (6);
   - условная установка байта (~20);
   - команды управления программой (циклы, переходы, вызовы, возвраты ~30);
   - строковые команды (7);
   - манипуляция с флагами (13);
   - системные команды (~30).

   В сумме команд CPU ~180
2. FPU (~100)
3. MMX (~60)
4. XMM (SSE 2) (~100)

---

~440

## 1.1. Основная память.

Основной функцией памяти является хранение информации и обеспечение селективного доступа к ней. Основная память представляет собой в основном память типа RAM (Random Access Memory – с произвольным доступом).

Произвольность доступа означает, что время обращения к любой ячейке памяти по заданному адресу инвариантно к этому адресу. Память типа RAM является энергозависимой.

Для сохранения наиболее важной информации часть адресного пространства ОП реализуется в виде ROM (Read Only Memory – постоянная память).

Память ROM с возможностью перезаписи называется PROM (Programming ROM – полупостоянная память).

Для современной реализации ОП типичным свойством является байтная адресация. Это означает, что адресация информации (команд, данных) размещенной в ОП производится на уровне байт.

Для адресации единицы информации содержащей несколько байт (например, слов или двойных слов) адрес этой единицы задается адресом младшего байта (меньшим из адресов байт, составляющих эту единицу). Для адресации байт внутри единиц может быть принят один из двух принципов, которые в англоязычной литературе именуются Big Endian и Little Endian.

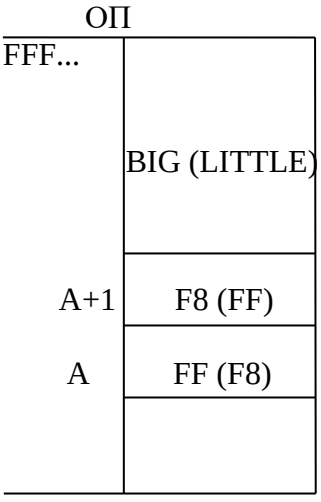
Использование принципа Big Endian предполагает, что байт с большей значимостью располагается в слове по меньшему адресу.

Пример:

Число $(-8)_{10}$ в формате WORD

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | | | | | | | | | | | | | | 0 |

(FFF8)




В процессорах фирмы Intel используется принцип Little Endian. В свою очередь Big Endian используется в больших компьютерах (Main Frame, IBM 360,Motorola).

000...

При адресации единиц информации фиксированной длины (слова, двойного слова и т.д.) адрес этой единицы задается наименьшим адресом байта (адресом крайнего левого байта). Для адресации элементов информации переменной длины, например строк, как правило, используется задание адреса граничного байта (левого или правого) а также длины этой строки.

В базовой модели процессора Intel реализована аппаратная поддержка структуры данных типа строки, при этом по умолчанию начальный адрес строки задается в регистре SI для строки источника и в регистре DI для строки приемника. Количество элементов строки по умолчанию задается в регистре CX (регистре счетчика). Направление обработки строки задается флагом DF: слева - направо (от меньших адресов к большим DF=0), справа - налево (от больших адресов к меньшим DF=1). Элементами строк могут быть байты или слова. Длина элемента определяется крайним правым битом кода операции обозначенным W.

$$W=\begin{cases}0\text{-Byte}\\1\text{-Word}\end{cases}$$

В мнемонике строковых команд длина элемента задается суффиксом мнемоники (B и W).

В базовую систему команд включено 5 строковых команд:

MOVS – пересылка строки;

LODS – пересылка строки из памяти в аккумулятор;

STOS – пересылка строки из АК в память;

CMPS – сравнение строк;

SCAS – сканирование строки;

В более старших моделях (начиная с Intel 80186) строковые команды расширились на INS и OUTS (ввод/вывод строки) (поддержка блочных пересылок).

Сами по себе строковые команды выполняют соответствующие действия только с одним элементом строки. Для обработки всех элементов строки (или их части) строковая команда снабжается предшествующим ей префиксом повторения REP или его модификациями.

Еще одним важным принципом размещения единиц информации фиксированной длины в ОП является соблюдение целочисленной границы. Реализация этого принципа позволяет уменьшить число обращений к памяти.

Сам по себе этот принцип гласит:

Адрес единицы информации, содержащей $2^k$ байт, должен быть кратен $2^k$.

Фактическая проверка соблюдения целочисленной границы для единицы информации из $2^k$ байт сводится к проверке k младших разрядов адреса на равенство 0.

Принцип целочисленной границы нашел реализацию во многих моделях компьютеров. Применительно к процессорам Intel проверка соблюдения целочисленной границы реализована, начиная с модели Intel i486. Эта проверка осуществляется при соблюдении следующих условий:

1) бит AM (Alignment Mask) в управляющем регистре CMO;
2) установлен флаг AF в регистре EFLAGS;
3) процессор осуществляет выборку или запись данных, но не команд на самом нижнем (прикладном) уровне привилегий PL=3 (Privilege Level).

Защита по уровням привилегий (кольцом защиты) является одной из составных частей общего механизма защиты.

Старшие модели процессоров Intel поддерживают 4 уровня привилегий на уровне сегментов: PL0- наивысший (уровень ядра ОС), PL3 – уровень прикладных программ-низший и два уровня U/S (user/supervisor) для страниц.

При соблюдении всех условий проверки целочисленной границы обнаружение факта ее нарушений приводит к прерыванию соответствующего типа.

### 1.1.1. Основные характеристики основной памяти.

1. Объем или емкость.

В последнее время к единицам измерения емкости памяти применяется модифицированное наименование в виде добавки к основным приставкам: Kilo, Mega, Giga, Tera, Peta, Exa слова Binary.

Предлагается также использовать новые единицы измерения Kib, Mib и т.п., именующиеся Кибибайт, Мебибайт, для того чтобы отличать кило, мега от степени 2.

2. Время доступа.
3. Удельная стоимость.

Определяется стоимостью хранения 1-го бита.

4. Длина слова памяти или ширина выборки.

Совпадает с разрядностью шины данных.

Разрядность шин адреса и данных для Intel 80x86 Pentium

| **модель** | **ША** | **ШД** | **М/Р** | **$V_{фап\ max}$** |
|---|---|---|---|---|
| 8086 | 20 | 16 | М | $2^{20}$ байт, 1 Мібайт |
| 80286 | 24 | 16 | Р | 16 Мібайт |
| 80386 | 32 | 32 | Р | 4 Gібайт |
| Pentium (в 1993 г.) | 32 | 64 | Р | 4 Gібайт |
| Pentium Pro (P6) | 36 | 64 | Р | 64 Gібайт |

Признак мультиплексируемости (или разделимости ША/ШД) – М/Р.

При использовании мультиплексированной (М) шины А/Д (адреса/данных) по одним и тем же линиям (проводам шины) передаются как адреса, так и данные, с разделением передач по времени.

$V_{фап\ max}$ – максимальный объем физического адресного пространства. Определяется разрядностью шины адреса (ША).

### 1.1.2. Рассмотрение основной памяти. Принципы организации взаимодействия между ЦП и ОП.

В современных моделях персональных компьютеров связь между ЦП и ОП организуется по системной шине, которая включает в себя ША, ШД, ШУ (шина управления).

В ПК на базе последних моделей Pentium системная шина разделена на 2 независимые шины:

- FSB (Front Side Bus) – переднего плана, для связи ЦП и ОП;
- BSB (Back Side Bus) – заднего плана, для связи ЦП с кэш–памятью L2.

Подобное разделение системной шины принято называть архитектурой DIB (Duo Independence Bus).

Посредником в организации взаимодействия между ЦП и ОП является контроллер (устройство управления ОП).

Для организации управления основной памятью со стороны ЦП используется интерфейсные сигналы управления для инициирования соответствующих операций в ОП:

- чтение (выборка) - read (RDM);
- запись - write (WRM).

Кроме приведенных обозначений в различной литературе используется также:

- LOAD (загрузка);
- FETCH (выборка);
- STORE (сохранение).

Обозначение операций с памятью LOAD/STORE часто используется в системах команд процессоров для обозначения соответствующих команд обмена с памятью.

В системе команд процессоров Intel команды с похожей мнемоникой LODS/STOS используется в отношении данных типа строка.

Для обмена с памятью данными целого типа используется универсальная команда MOV (пересылка), в которой в качестве источника (src) или/и приемника (dst) могут быть заданы следующие пары:

- Reg -> reg (регистр- reg);
- Reg -> mem (mem- память); *write*
- Mem -> reg. *read*

В состав ЦП обычно входят 2 интерфейсных регистра для организации обмена с памятью (регистр адреса – MAR (Memory Address Register) и регистр данных – MDR (Memory Data Register)).

Как правило, эти регистры являются программно-недоступными.

### 1.1.3. Элементная база ОП.

В качестве таковой в современных компьютерах используется *сверхбольшие интегральные схемы* (СБИС) на основе *CMOS технологий* (комплементарная МОП технология).

В качестве элементарной ячейки элементов памяти используется элемент типа **DRAM – Dynamic RAM**.

Основу элементов DRAM составляют один транзистор и конденсатор. Наличие заряда на конденсаторе соответствует хранению 1, отсутствие- 0.

Вследствие способности конденсатора к разрядке, содержимое динамической памяти требует постоянной перезаписи (Refresh).

Средний интервал между двумя последовательными перезарядками составляет 8-64 мксек.

Альтернативой динамических элементов памяти DRAM являются статические элементы **SRAM (Static RAM),** основу которых составляет статический триггер, создаваемый на 4-х или 6-ти транзисторах.

Если сравнивать элементы SRAM и DRAM по основным характеристикам, то можно утверждать следующее:

1) *элементы DRAM значительно дешевле;*
2) *элементы SRAM значительно более скоростные;*
3) *элементы DRAM характеризуются большей простотой и соответственно гораздо большей плотностью упаковки на кристалле.*

Областью применения элементов DRAM в современных компьютерах является основная память, в то время как на элементах SRAM реализуется кэш-память.

### 1.1.4. Основные принципы иерархической организации памяти.

Требования к памяти со стороны пользователя являются противоречивыми, что в принципе и является предпосылкой построения памяти компьютера по многоуровневой (иерархической) схеме.

Иерархический подход к организации памяти компьютера означает, что память представляет собой совокупность запоминающих устройств, отличающихся как своими основными характеристиками, так и принципами действия. В простейшем случае иерархичная структура памяти рассматривает состояние из четырех уровней (уровни нумеруются по степени и близости к ЦП):

1. Регистровая память ЦП(CPU).
2. Кэш-память.
3. Основная память.
4. Внешняя память на жестких магнитных дисках.

Дополнение к упрощенной схеме уровней:

1. Кэш-память сама является многоуровневой (три, и даже четыре уровня).
2. Ниже уровня четвертого может использоваться дополнительно архивная память на сменных носителях.
3. Промежуточный уровень между третьим и четвертым в виде дискового кэша.

От верхнего уровня к нижнему емкость возрастает, время доступа увеличивается (быстродействие уменьшается), удельная стоимость уменьшается.

Иерархичный подход к организации памяти можно рассмотреть как разумный компромисс к обеспечению противоречивых требований со стороны пользователей к основным характеристикам памяти.

## 1.2. Периферийные (внешние) устройства компьютера.

Номенклатура этих устройств весьма разнообразна и включает в себя три вида устройств:

1. Внешние запоминающие устройства.
2. Устройства ввода.
3. Устройства вывода.

Внешние запоминающие устройства образуют внешнюю память компьютера, основным назначением этих устройств является долговременное хранение большого объема программ, данных и другой информации, необходимой для обеспечения функционирования в течение длительного времени. Во внешней памяти хранится практически все программное обеспечение компьютера.

Отличительными особенностями внешней памяти по сравнению с основной памятью являются:

- энергонезависимость;
- отсутствие доступа со стороны ЦП;
- малая скорость обмена;
- практически неограниченный объем;
- относительная дешевизна хранения данных.

Основные устройства (ВЗУ) в составе ВП:

1. Накопители на жестких дисках (винчестеры).
2. Накопители на гибких магнитных дисках.
3. Накопители на оптических дисках (CD ROM).
4. Накопители на кассетной магнитной ленте.
5. Устройства флэш-памяти.

### 1.2.1. Основные принципы организации обмена между ядром и ПУ.

Организация обмена возлагается на так называемую систему ввода/вывода (IOS). Система ввода/вывода (СВВ) представляет собой комплекс аппаратных и программных средств.

Аппаратные средства СВВ:

- ПУ;
- контроллеры (адаптеры) ПУ;
- специализированные контроллеры для организации обмена (DMAC – direct memory access controller);
- аппаратные интерфейсы;

- система прерываний (точнее ее аппаратная часть обычно представлена специализированным контроллером PIC).

Программные средства СВВ:

- супервизор ввода-вывода;
- драйверы ПУ.

Под вводом данных обычно понимается их передача из ПУ в основную память. Под выводом данных – передача данных из ОП в ПУ.

### 1.2.2. Основные способы организации ввода/вывода.

1. Программно-управляемый ввод-вывод (В/В) (PIO).

Иногда разделяют на синхронный и асинхронный.

2. В/В по прерыванию
3. В/В в режиме DMA (прямой доступ к памяти).

При использовании первых двух способов все управления В/В организует ЦП. При этом регистры ЦП обычно являются промежуточным звеном при пересылке данных между ОП и ПУ. При использовании третьего способа организацию обмена осуществляет контроллер DMA без участия ЦП. По завершению операции обмена контроллер DMA информирует об этом ЦП через систему прерываний.

Как правило, контроллеры ПУ включает в свой состав:

- регистр данных;
- регистр приказов (регистр команд);
- регистр состояний.

для доступа к ним со стороны ЦП.

### 1.2.3. Аппаратная поддержка системы ввода/вывода процессора Intel 80x86, Pentium. Уровень системы команд.

В базовой системе команд есть две команды IN и OUT, с помощью которых реализован обмен между регистром-аккумулятором ЦП AX(AL) и адресуемым портом ввода/вывода. Использование специальных команд В/В предполагает наличие раздельного адресного пространства памяти и В/В. Фактически один и тот же адрес может быть использован и как адрес ячейки памяти и как адрес порта В/В. В командах IN/OUT используется два способа адресации портов В/В:

1) прямая (адресация порта задается во втором порте команды);

2) неявная (машинный код команды занимает 1 байт и состоит из единственно кода операции, а адрес порта находится в регистре DX (данных)).

В некоторых монографиях второй способ адресации портов называется косвенным. Использование косвенной адресации существенно расширяет объем адресного пространства В/В для однобайтных портов В/В до $0 – 2^{16} -1=65535$, для двухбайтных - до $0 – 2^{15}-1=32767$, для четырехбайтных - до $0 – 2^{14}-1 = 16383$. При адресации портов В/В используется принцип целочисленной границы. В расширенной системе команд, начиная с Intel 80186, используется дополнительно две команды INS/OUTS, с помощью которых обеспечивается блочный В/В при использование префикса REP или его модификации.

### 2.2.4. Аппаратная поддержка на уровне управляющих или осведомительных сигналов.

1. M/$\overline{IO}$ (выходной) – Memory/Input-Output – с помощью этого сигнала разделяются обращение к памяти (высокий уровень) и к портам В/В (низкий уровень), с помощью этого сигнала реализуется поддержка раздельных адресных пространств памяти и В/В.

2. RDY (входной) – ReaDY – сигнал готовности адресованного устройства к взаимодействию с ЦП. Наличие активного уровня этого сигнала на входе ЦП означает, что адресуемое ВУ выставило данные на шину при вводе или восприняло данные с шины при выводе.

3. INTR (входной) – INTerrupt request - запрос прерывания. Как правило, этот вход ЦП связан с выходом INT программируемого контроллера прерывания (PIC) в свою очередь к PIC подключаются запросы прерываний от подключаемых к компьютеру ВУ. PIC выдает соответствующий сигнал процессору при наличии хотя бы одного незамаскированного запроса прерываний на его входах. PIC имеет внутренние схемы маскирования (разрешения или запрещения) для запросов прерываний от ВУ, поступающих на его вход.

ЦП реагирует на активный уровень входного сигнала INTR по завершению выполнения каждой машинной команды и при условии, что внешние прерывания не замаскированы по этому выходу. (IF=1 - разрешение прерывания).

4. $\overline{INTA}$ (выходной) – INTerrupt Acknowledgment – процессор выставляет активный уровень этого сигнала, если после завершения очередной машинной команды, он обнаружил незамаскированный сигнал на своем входе INTR. При получении сигнала INTA PIC выставляет на шину данных (ее младший бит) код (тип) прерывания,

идентифицирующий наиболее приоритетный источник запросов. В свою очередь ЦП приняв идентификатор (тип прерывания) модифицирует его в адрес вектора прерываний, который однозначно определяет начальный адрес программы обработчика этого прерывания. Надчеркивание над сигналом INTA означает, что его активным уровнем является низкий. (В более современной литературе INTA#).

5. HOLD (входной) – запрос захвата шины от внешней подсистемы (ВУ или контроллера DMA).

6. HLDA (выходной) – HoLD Acknowledgment – подтверждение захвата шины. Этот сигнал выдается ЦП в ответ на сигнал HOLD, после перевода мультиплексируемой шины адреса данных и некоторых управляющих сигналов в Z – состояние (высокоимпедансное). ЦП пребывает в состоянии отключения от шины до момента перевода входного сигнала HOLD в пассивный (нижний) уровень. Во время захвата шины, как правило, для организации обмена с ВУ в режиме DMA ЦП может продолжать выполнение команд, используя для этого буфер команд и внутренние регистры для выборки операндов в базовой модели, а также внутрикристальную кэш-память для старших моделей.

В старших моделях ЦП практически все сигналы базовой модели в том или ином виде присутствуют, исключение составляет сигнал подтверждения прерывания INTA, что компенсируется инициированием специального цикла шины, называемого подтверждением прерывания в ответ на получение незамаскированного запроса по входу INTR.

Реализация цикла подтверждения прерывания сводится к передаче по шине данных от PIC к ЦП типа (кода) прерывания.

## 2.3. Общие представления об аппаратных интерфейсах. Понятие интерфейса.

В дальнейшем под интерфейсом будем понимать *аппаратный интерфейс*.

В литературе существует достаточно большое число определений интерфейса. В общем плане под интерфейсом принято понимать *способ сопряжения и взаимодействия между несколькими объектами и субъектами.*

В отношении ЭВМ принято рассматривать множество понятий интерфейсов, например, *аппаратный, программный, пользовательский* и т.д.

В обобщенном плане под *аппаратным интерфейсом* принято понимать совокупность линий и шин электрических схем и алгоритмов, предназначенную для осуществления обмена информацией между устройствами.

***Аппаратные интерфейсы***, используемые в ЭВМ, как правило, обладают свойством *унифицируемости*, подчиняются определенным стандартам.

Унификация затрагивает следующие моменты:

- унифицированность набора линий и шин по составу и назначению;
- унифицированность сигналов и протоколов обмена по линиям (шинам) интерфейса;
- унифицированность конструктивных характеристик средств сопряжения.

Многие авторы отождествляют понятие интерфейса и шины, кроме того, неоднозначность термина тоже имеет место и в отношении шин различных типов. Так, например системная шина достаточно часто называется главной шиной, шиной процессора, шиной памяти и т.п.

В определение интерфейса не вписываются так называемые беспроводные интерфейсы.

Основные виды линий (шин) входящие в состав интерфейсов:

- шина адреса;
- шина данных;
- шина управления (для передачи сигналов управляющих обменом);
- линии синхронизации (для передачи сигналов, синхронизирующих обмен данных по интерфейсу);
- линии запросов прерывания (для передачи сигналов прерываний от ВУ подключенных к интерфейсу в ЦП или PIC);
- линии разрешения прерывания (для передачи сигналов разрешения от ЦП или PIC к ВУ);
- линии питания;
- линии заземления.

Общее число линий в современных интерфейсах составляет ~150-200.

### 2.3.1. Уровни представления интерфейсов.

1) ***логический***: определяет состав, наименование и предназначение линий (шин), а также порядок передачи информации (сигналов) по этим линиям (протокол обмена, который обычно представляется в виде временных диаграмм).

2) ***физический***: определяет параметры сигналов (электрических, оптических), переданных по линиям интерфейсов;

3) ***конструктивный:*** определяет физическую реализацию шин интерфейсов (печатные проводники, витая пара, коаксиальный кабель), а также виды разъемов и распределения линий интерфейсов по контактам разъемов.

### 2.3.2. Классификация интерфейсов.

1. *По способу соединения компонент:*
   - магистральный;
   - радиальный;
   - цепочный;
   - комбинированный (смешанный).

С помощью *цепочного* интерфейса обычно реализуются линии разрешения прерывания, которые проходят последовательно через ряд подключенных к ним ВУ. *Сигнал разрешения*, проходя последовательно через подключенные к интерфейсу ВУ, может быть заблокирован первым из ВУ на этой линии, которая предварительно послала запрос прерывания.

Реализация цепочного интерфейса предоставляет преимущество в обслуживании (приоритет) тем устройствам, которые находятся ближе по электрической связи к источнику сигнала разрешения. Этим источником, как правило, является **арбитр** – специализированный блок, входящий в состав ЦП. Как правило, линии цепочного интерфейса объединяются с линиями магистрального интерфейса, образуя комбинированный.

2. *По способу передачи информации:*
   - параллельные;
   - последовательные;
   - параллельно-последовательные.
3. *По принципу обмена:*
   - синхронный;
   - асинхронный.
4. *По режиму передачи информации*:
   - односторонний (симплексный);
   - двухсторонний (дуплексный);
   - двухсторонний - поочередный (полудуплексный).
5. *По функциональному назначению:*
   - системные;
   - интерфейсы периферийных устройств (малые интерфейсы);

- интерфейсы ввода/вывода;
- интерфейсы программно-управляемых модульных систем и приборов (приборные).

### 2.3.3. Основные характеристики интерфейсов.

**1. Пропускная способность.** Определяется максимальным количеством бит (чаще байт), передаваемых по интерфейсу за единицу времени (за 1 сек.).

**2. Информационная ширина.** Определяется числом бит (реже байт), переданных по линиям интерфейса параллельно (разрядность шины данных).

**3. Максимальное возможное удаление устройств подключенных к устройству.**

# 3. Эволюция структурной организации компьютеров.

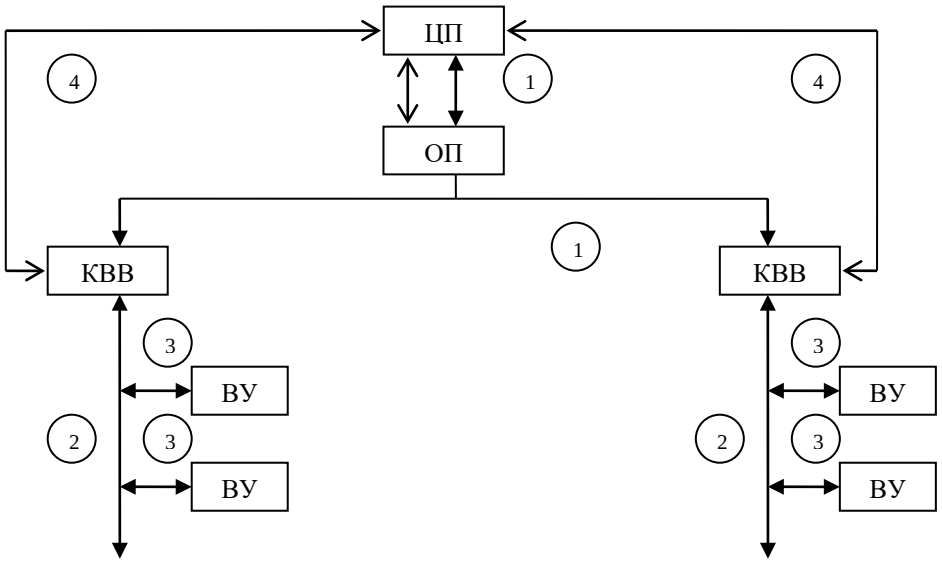
Для ЭВМ I-го и II-го поколения, как правило, использовалась так называемая структура с непосредственными связями, характеризующим признаком этой структурной организации являлось наличие отдельных шин (интерфейсов) для каждого из ВУ с ядром ЭВМ (процессором и ОП).

По мере развития ЭВМ и усовершенствования парка внешних устройств (ВУ), существенным оказалась необходимость разгрузки ЦП от рутинных операций по вводу/выводу. В связи с этим в середине 60-х была анонсирована структурная организация ЭВМ с каналами ввода/вывода.

Подобную структуру называют также иерархической, так как организация ввода/вывода в ней реализована по иерархической схеме: ЦП ⟷ КВВ ⟷ ВУ.

В свое время подобная структура являлась типичной для больших универсальных ЭВМ, основные из которых являлись разработками фирмы IBM 360/370/390. немного позднее в 90-е годы ЭВМ подобного типа получили название Main Frame.

## 3.1. Упрощенная структура ЭВМ с каналами ввода/вывода.




В структуре с каналами ввода/вывода используется *несколько видов интерфейсов*:

1. Интерфейс ОП – обеспечивает связь между основной памятью и центральным процессором с одной стороны, а также каналами ввода/вывода с другой стороны. Информационная ширина этого интерфейса определяется длиной слова ОП (шириной

выборки из ОП). Для различных моделей ЭВМ IBM 360/370/390 типичными значениями ширины выборки являлись 16/32/64 бита.

2. Интерфейс ввода/вывода – предназначен для организации обмена между каналами ввода/вывода и ВУ, подключенного к ним. Посредником в организации этого обмена со стороны ВУ является устройство управления (контроллер ВУ).

3. Интерфейс внешних (периферийных) устройств.

4. Интерфейс ЦП – КВВ – по этому интерфейсу передача данных не осуществляется. По нему от ЦП к КВВ передаются управляющие сигналы, а в обратном направлении – осведомительные сигналы (сигналы о состоянии КВВ и ВУ, подключенных к нему).

Интерфейс ввода/вывода является стандартизованным, в свою очередь интерфейсы ОП и внешних устройств являются специализированными. Интерфейс ОП является моделезависимым, то есть его характеристики изменяются от модели к модели. Интерфейсы периферийных устройств являются разнообразными в зависимости от вида подключения по ним ВУ.

## 3.2. Организация обмена (ввода/вывода) в ЭВМ с иерархической структурой.

Ввод/вывод, реализованный в этой структуре, принято называть канальным В/В. Он основан на использовании в структуре ЭВМ специальных процессоров, ориентированных на операции В/В. Эти процессоры обычно называются каналами В/В.

Канальный В/В является программно-управляемым, т.е. реализуются специальные программы, которые носят название *канальной программы*. Они для организации обмена с ВУ различных типов хранятся в ОП и выполняются каналами В/В. В связи с тем, что канал В/В является процессором, правда специализированным, управление порядком следования команд канальной программы осуществляется с помощью своеобразного счетчика команд.

## 3.3. Основные функции каналов ввода/вывода.

**1. Функции по установлению связи между ВУ и ОП**

1) прием и декодирование команд В/В от ЦП;

2) инициирование канальной программы при получении команды SIO (Start Input/Output) от ЦП;
3) проверка состояния ВУ, участвующего в обмене, и передача в ЦП информации о его готовности или неготовности к обмену.

**2. Функции, связанные с непосредственной передачей данных между ВУ и ОП:**

1) последовательная выборка команд конкретной программы из ОП, их декодирование и выполнение;
2) обеспечение приема, передачи, контроля и промежуточного хранения данных при обмене между ОП и ВУ;
3) формирование текущих адресов ОП, по которым записываются или считываются передаваемые данные;
4) согласование форматов данных передающихся по интерфейсу. В/В с форматом интерфейса ОП. (Как правило, ширина интерфейса В/В меньше ширины интерфейса ОП, в связи с чем возникает необходимость упаковки/распаковки данных при передаче.);
5) подсчет числа передаваемых байт данных с целью определения момента завершения передачи блока данных;
6) выработка последовательности синхронизирующих управляющих сигналов в соответствии со стандартом интерфейса В/В;
7) анализ особых ситуаций ВУ во время обмена (ошибка четности передаваемых данных, сбой устройства) и информирование ЦП об этих ситуациях (с помощью запроса прерывания).

**3. Функции, связанные с завершением обмена и разрешением логической связи между ВУ и ОП:**

1) определение момента завершения, т.е. окончания выполнения программы;
2) передача в ЦП сигнала прерывания о завершении обмена.

Участие ЦП в организации канального ввода/вывода сводится к выполнению следующих функций:

1) инициирование операции В/В (реализуется командой SIO);
2) проверка состояния канала В/В (ВУ подключается к нему (реализуется специальной программой TCH – Test CHanel или TIO – Test I/O));

3) остановка операции В/В (реализуется специальной программой HIO – Halt I/O), возможно для инициализации более приоритетной операции в канале В/В.

Перечисленные выше команды В/В являются привилегированными, т.е. могут выполняться только программами операционных систем, при этом процессор должен находиться в состоянии супервизора (одно из альтернативных состояний T/S – Task/Supervisor).

## 3.4. Классификация каналов ввода/вывода.

По режиму функционирования КВВ разделяются на два вида:

- селекторные;
- мультиплексные.

Селекторный канал функционирует в монопольном режиме, обеспечивает работу с единственным ВУ из множества подключенных к нему.

Мультиплексный канал обеспечивает параллельную работу с несколькими ВУ в течение некоторого интервала времени, при этом мультиплексный канал переключается с обслуживания одного ВУ на обслуживание другого ВУ, причем это переключение выполняется достаточно быстро, настолько, что обеспечивает возможность параллельного обслуживания нескольких ВУ.

Взаимодействие мультиплексного канала с одним из конкретных ВУ называется *сеансом связи с ВУ*. В зависимости от порций данных, передаваемых через канал за один сеанс связи мультиплексные каналы разделяются на два вида:

- байт- мультиплексный (несколько байт за один сеанс);
- блок- мультиплексный (за один сеанс связи через канал передается блок данных, длина блока совпадает с длиной страницы и обычно составляет от 1 до 8 Кб).

Часть ресурсов мультиплексного канала, использующаяся отдельным ВУ, называется *подканалом*. В каждом подканале используется область памяти (канала), в которой хранится информация о текущем состоянии обмена с данным ВУ.

Переключение мультиплексности канала с одного ВУ на другой сопровождается сохранением информации в памяти подканала для текущего ВУ и выборкой информации из памяти подканала для следующего ВУ.

Каналы В/В, используемые в современных компьютерах типа Mainframe, являются универсальными и позволяют производить переключение на один из трех режимов.

Байт-мультиплексный режим используется для обмена с ВУ с байтным характером передачи данных (например, с клавиатурой), а блок-мультиплексный режим для обмена с ВУ с блочным характером передачи (магнитный диск).

## 3.5. Сравнение канального ввода/вывода с Programmable I/O.

Так как канал В/В осуществляет организацию обмена с ВУ по собственной программе, то КВВ следует считать программно управляемым, однако, в отличие от PIO программу, связанную с обменом, выполняет не ЦП, а специализированной процессор – КВВ.

## 3.6. Сравнение с DMA.

Многие специалисты сопоставляют (как синонимы) канальный В/В и DMA. Аналогия между ними состоит в том, что оба этих способа организации В/В реализуются практически без участия ЦП. Так же как и для DMA КВВ требует некоторого участия ЦП лишь на этапе инициализации В/В, при этом из ЦП в КВВ передается начальный адрес программы в памяти, а также при особых ситуациях в работе канала или ВУ. Существенным отличием КВВ от DMA является программная реализация первого и чисто аппаратная второго.

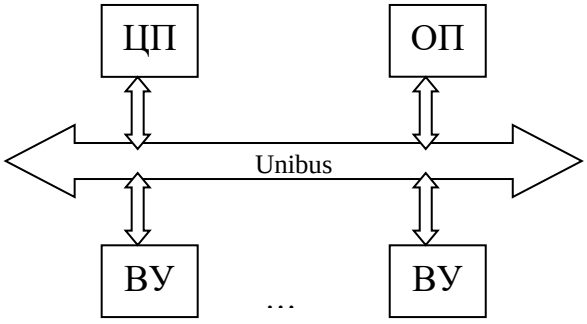
# 4. Магистральная структура компьютера

Такая структура является типичной для мини- и микро- ЭВМ, в том числе и персональных компьютеров 70-80 гг. XX века.

В качестве примеров реализации единого интерфейса можно привести следующие:

- omnibus (PDP - 8) – DEC;
- unibus (PDP – 11) – DEC;
- multibus (Intel 8086);
- IBM PC (PC/XT) – 8086;
- ISA (PC/AT) – INT 80286.

## 4.1. Обобщенная структура компьютера с общей шиной.




*Основные особенности:*

1. Для связи между любыми компонентами используются одни и те же линии интерфейса (ША, ШД, ШУ и т.п.), а также сигналы, управляющие передачей. Этот факт в значительной степени упрощает организации связи между устройством и обеспечивает простоту наращивания структуры путем подключения дополнительных устройств.

2. Использование единого интерфейса предполагает, что в любой момент времени по нему может быть организован обмен только между двумя устройствами, одно из которых является ведущим, а другое исполнительным, ведомым. Ведущим устройством не может быть ОП. Подобная структура является источником конфликтов между различными активными устройствами, требующими практически одновременного взаимодействия с шиной для передачи данных. Компьютеры с подобной структурой не предполагают значительного наращивания числа устройств. Подобная структура для увеличения производительности требует введения дополнительных шин для разгрузки основной.

3. В компьютерах с общей шиной могут быть реализованы различные способы организации В/В, такие как PIO, В/В по прерыванию, а также В/В в режиме DMA.

В структуре с общей шиной могут быть реализованы оба подхода (в рамках конкретной модели – один из них) к адресации ВУ (также портов В/В):

1) использование раздельного адресного пространства для памяти и портов В/В;
2) использование единого адресного пространства.

Первый способ адресации является типичным для ПК, второй типичным для PIC.

Использовании единого адресного пространства для памяти и портов В/В предполагает единообразие операций обмена с памятью и ВУ. Это означает, что в системе команд компьютеров, предполагающих использование единого адресного пространства, отсутствуют специальные команды В/В (типа IN и OUT). Это означает, что В/В, т.е. пересылка данных из регистра процессора в регистр контроллера ВУ и в обратном направлении реализуются той же универсальной командой (типа MOVE), как и обмен между процессором и памятью.

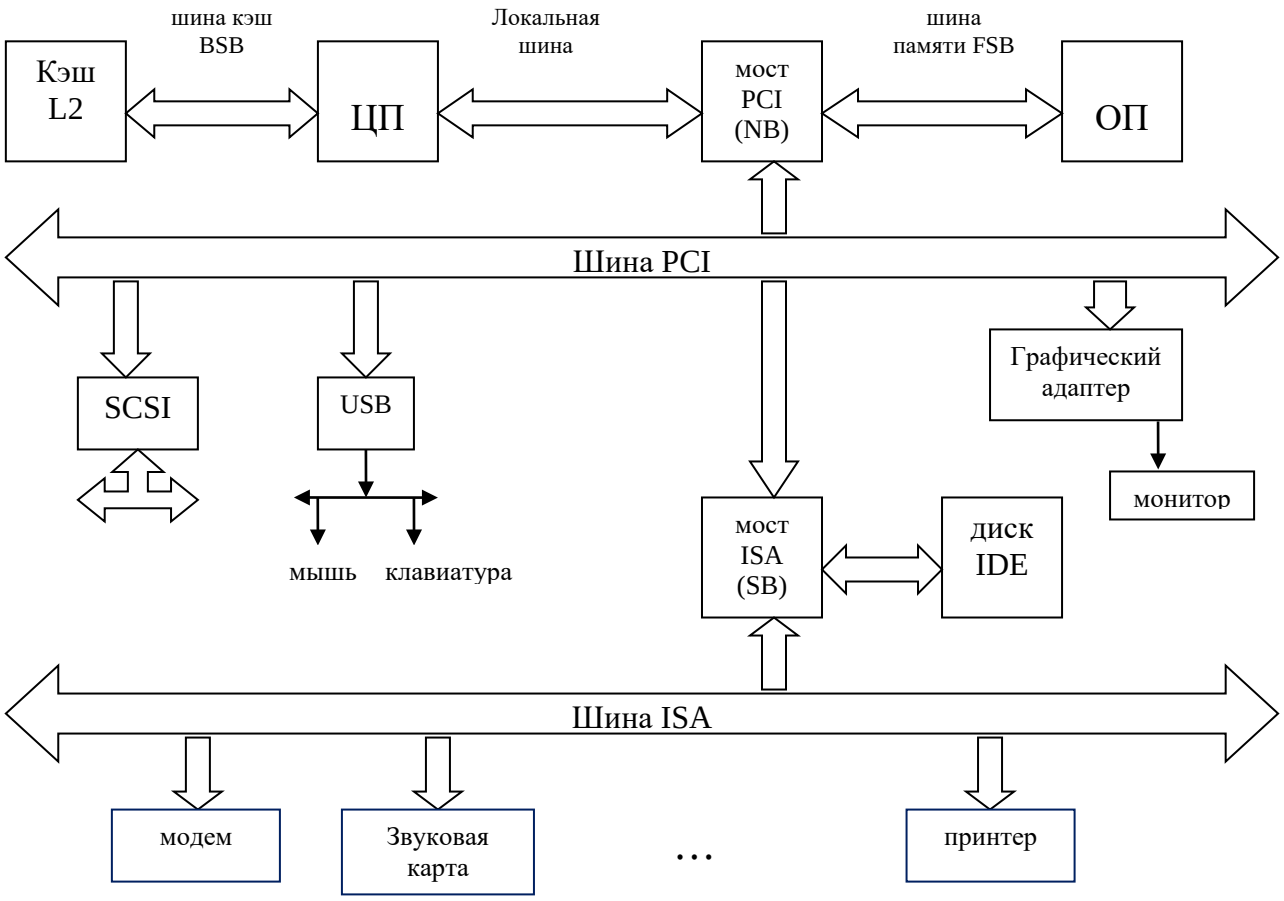
## 4.2. Усовершенствования структуры с единым интерфейсом.

Вследствие наличия существенного недостатка у одноименной структуры компьютера в 80-е годы прошлого столетия в различных моделях компьютера была реализована модификация этой структуры, сначала структура с двумя видами шин, далее структура с тремя видами шин и наконец - многошинная структура компьютера, используемая в современных компьютерах. В структуре с двумя видами шин производится дополнение общей шины (системная шина) дополнительными шинами В/В. Примерами шин (интерфейсов) В/В могут служить:

- SCSI – Small Computer System Interface;
- IDE/ATA – Integrated Drive Electronics(устройство с встроенным контроллером)/AT Attachment (AT – Advanced Technology).

Дальнейшее развитие структуры привело к использованию дополнительного вида шины, называемой *шиной расширения*. Шина расширения является промежуточным звеном между шиной процессор-память (основной шиной) и шиной В/В. Примером шины (интерфейса расширения) может служить PCI (Peripheral Component Interconnect).

## 4.3. Пример многошинной структуры ПК на базе первых моделей процессора Pentium(середина-конец 90-х гг.).




DIB – Dual Independent Bus – двойная независимая шина:

- FSB – Front Side Bus – шина переднего плана;
- BSB – Back Side Bus – шина заднего плана;

NB – North Bridge

SB – South Bridge

USB – Universal Serial Bus – универсальная последовательная шина;

ISA – Industrial Standart Architecture

По мнению М.Гука мост представляет собой аппаратное средство для соединения шин (разнородных или однородных), мосты входят в состав чипсетов системных плат. Мосты являются программируемыми устройствами. При программировании задается диапазон адресов пространств памяти и В/В, отведенных устройствам, соединяемым мостом шин. Если адрес целого устройства текущей транзакции на одной шине (стороне моста) относится к шине противоположной стороны, мост перенаправляет транзакцию на

соответствующую шину и выполняет действие по согласованию протоколов шин. Таким образом, совокупность мостов выполняет маршрутизацию транзакций по связанным шинам.

*По Гуку:* чипсет – набор специализированных интегральных схем, при соединении которых формируется функциональный блок компьютера.

Чипсеты применяются в системных платах, графических контроллерах и других устройствах, функции которых нельзя реализовать в одной микросхеме.

В данной структуре используется 7 видов интерфейсов (шин), из которых 5 (ISA, PCI, IDE, SCSI, USB) являются стандартными и 2 (FSB, BSB) – моделезависимыми (нестандартными).

## 4.4. Основные характеристики и особенности в стандартных интерфейсах ПК.

### 4.4.1. ISA/EISA.

Является разработкой фирмы IBM, как развитие более ранних интерфейсов Microbus, Multibus применительно к ПК IBM PC.

Первоначальная версия ISA включала восьмиразрядную шину данных (ШД) и 20-ти разрядную шину адреса (ША), т.е. предназначалась для ПК на базе процессора Intel 8088. Дальнейшее развитие было произведено в 1984 году в связи с появлением модели (ориентированной на модель) Intel 80286, в этом случае ШД 16-ти разрядная, а ША по сравнению с 86 была увеличена до 24 разрядов. Пропускная способность шины в зависимости от модификации составляет от 4 до 16 Мб/сек.

*Основные недостатки* шины ISA:

1. Неспособность обеспечивать режим автоконфигурирования. В результате этого пользователю приходится вручную устанавливать номера прерываний и адреса устройств, что требует соответствующей квалификации. От этого недостатка свободна, например шина PCI.
2. Низкая пропускная способность шины. Обменяется тем, что передача данных по ней реализована без подтверждения (квитирования), в связи, с чем передача всегда выполняется со скоростью самого медленного устройства.

В связи с этими недостатками согласно спецификации PC’99, принятой фирмами Intel, Microsoft и другими производителями ПК и ОП, шина ISA не должна использоваться в ПК.

Расширением шины ISA является **EISA,** которая имеет ШД и ША по 32 бита. EISA

является разработкой большой группы фирм, в которые входит в частности HP, Compac, NEC и т.д., как альтернатива шине MCA(Micro Chanel Architecture), разработанной фирмой в 1987 году.

Основное достоинство по сравнению с MCA – возможность подключения ранее разработанных для ISA контроллеров (адаптеров) ВУ. Тем не менее, MCA находит в настоящее время применение в мощных файл-серверах, где требуется высоконадежный и производительный В/В.

### 4.4.2. PCI.

Представляет собой типичный пример шины расширения. Она была разработана фирмой Intel, которая, запатентовав шину, организовала промышленный консорциум PCI SIG, в который вошли все ведущие фирмы в 1990 г.

Занимает особое место в архитектуре ПК, являясь высокоскоростной шиной расширения, соединяет системную шину с шинами (интерфейсами) ВВ.

Именно шина PCI считают своеобразной “центральной шиной (экватором)” структуры ПК при определении наименования моста.

В принципе шина PCI разрабатывалась применительно к ПК на базе процессоров Pentium, однако, являясь независимой от процессора, она находит также применение в компьютерах компании SUN Micro system, в серверах на процессорах Alpha и Rower PC.

Используются *две версии* шины PCI с 32-х и 64-х разрядной шиной данных, 132/264 Мб/с с частой 33 МГц. Более поздние версии с частотой PCI 2.1 обеспечивают пропускную способность 528 Мб/с на частоте 66 МГц.

Шина PCI сейчас является *самой высокоскоростной* шиной расширения в ПК (не считая AGP – Accelerated Graphic Port), которая используется только для высокоскоростных графических мониторов.

### 4.4.3. IDE (ATA). SCSI.

При использовании IDE основной контроллер диска встроен в чипсет (входит в состав южного моста), ответная часть контроллера размещена в самом устройстве (накопителе на жестком диске). К интерфейсам IDE можно подключать до четырех устройств.

По сравнению с интерфейсом SCSI, который требует отдельного контроллера. Скоростная возможность IDE мало уступает SCSI, однако IDE дисковода примерно в два раза дешевле. В связи с этим в большинстве случаев в ПК в качестве дискового интерфейса используется IDE. В свою очередь SCSI используется в серверах в качестве

интерфейса высокоскоростных дисков, а также сканеров и стримеров (ленточных накопителей).

Различные версии IDE/ATA имеют существенно различающиеся пропускные способности. Одна из первых версий интерфейса 1986 года имела пропускную способность 4 Мб/с. Последняя версия ATA/ATAPI-4 (PI – Package Interface) имеет пропускную способность 66 – 100 Мб/с.

Разработка SCSI – 1986 г., интерфейс стандартизован ANSI – American National Standarts Institute. Используются две основные версии: Narrow (ШД – 8 б.) и Wide (ШД – 16 б.). Более современная версия 32 бита, определена стандартом, однако, обладает высокой стоимостью, из-за чего практически не используется.

Интерфейс использует последовательное (шлейфное) подключение устройств в количестве до 8 или 16, в зависимости от версии. Максимальное удаление до 25 метров.

Одним из устройств, подключенных к шине SCSI, является SCSI-контроллер (хост-адаптер), обеспечивающий связь с шиной SCSI с шиной расширения или системной шины. С учетом этого фактическое число подключаемых к SCSI внешних устройств равно 7 или 15 в зависимости от модификации.

*Сравнение интерфейсов IDE/ATA и SCSI:*

*Достоинства SCSI:*

1) большее число подключаемых ВУ по сравнению с IDE (в IDE до четырех);
2) возможность большого удаления подключения устройств до 25 метров (для IDE не более 0,5 метров);
3) возможность параллельной работы всех устройств, подключенных к интерфейсу (для IDE может быть активным только одно устройство).

*Недостатки SCSI:*

Высокая стоимость, в частности дорогие кабели, примерно в 2 раза дороже, чем у IDE.

### 4.4.4. USB.

Является промышленным стандартом расширения архитектуры ПК, ориентированным на интеграцию с телефонией и устройствами бытовой электроники. Шина разработана рядом компьютерных и коммерческих компаний, в число которых входят Intel, Microsoft, HP, Philips в 1996 г.

*Основными целями разработки USB являлись:*

1. Возможность “горячего” подключения (без выключения ПК) различных внешних (находящихся вне корпуса) устройств с низким и средним трафиком (скоростью обмена). К ним относятся: мышь, модем, принтер, клавиатура,

джойстик, сканер, цифровая камера, звуковые колонки, цифровой телефон, флэш-память и другие. Шина USB является полным воплощением концепции PnP.

2. Замена коммуникационных портов, для которых характерны более низкие скорости и отсутствие “горячего” подключения.
3. Сокращение общей длины кабелей при подключении ВУ. Каждое USB-устройство может содержать разъем для последовательного подключения в цепочку других устройств. Более того, одно USB-устройство можно сделать концентратором (хабом), к которому можно подключить несколько других устройств.

Шина USB позволяет организовать многоуровневое каскадирование подключенных устройств и обеспечивает логическую топологию “дерево”, вершиной которой является корневой хаб (хост-контроллер).

В иерархии USB шин и устройств имеется единственный управляющий блок в виде хост-контроллера, который подключен к одной из шин расширения компьютера (обычно к PCI). Контроллер USB входит в состав южного моста и является двух портовым. К каждому порту может быть подключены ВУ или промежуточный хаб при этом допускается до пяти уровней подключения ВУ к хост-контроллеру через промежуточные хабы. Общее число подключаемых устройств к USB может достигать 127.

#### 4.4.4.1. Характеристики USB.

Версия **USB 1.0** имеет два режима передачи: низкоскоростной, пропускная способность составляет 1,5 Мбит/с и полноскоростной, с пропускной способностью 12 Мбит/с.

Версия **USB 2.0** в высокоскоростном режиме передачи обеспечивает пропускную способность 480 Мбит/с, что существенно расширяет класс устройств, подключаемых к шине.

Программная поддержка в виде драйвера шины вошла в состав ОС Windows 98, что являлось переломным моментом в истории шин.

#### 4.4.4.2. Физическая реализация шины.

Последовательная шина USB по своей организации существенно отличается от параллельных шин (интерфейсов), в ней нет отдельных линий для данных, адреса управления. Все протокольные функции связаны с обменом по шине, выполняются с помощью одной пары сигнальных проводов путем пересылки определенным образом организованных цепочек байт, называемых пакетами.

Кабель USB состоит всего из четырех проводов. Два из них предназначены для передачи питающего напряжения. Физическая реализация кабеля USB - экранированная витая пара с длиной сегмента до 5 метров для полной скорости передачи или

неэкранированная невитая пара с длиной сегмента до 3-х метров для низкой скорости передачи.

Линии питания в USB обеспечивают подачу питающего напряжения на устройства, подключенные к ней, и не требует для этих устройств дополнительного источника питания. Передача данных по соответствующим линиям USB осуществляется в полудуплексном режиме.

## 4.5. Функции мостов.

Северный мост иначе называется системным контроллером (TSC). Северный мост как системный контроллер выполняет функции по взаимодействию и обмену между устройствами, подключенными к нему: ЦП, ОП, внешняя кэш память L2 и шина PCI.

**SB** называется контроллером шин и выполняет следующие *системные функции*:

1) организация моста между шинами PCI и ISA с согласованием частот синхронизации;
2) реализация высокопроизводительного (обычно двух канального) дискового интерфейса IDE/ATA;
3) реализация стандартных для ПК средств для В/В: два контроллера прерывания PIC, два контроллера доступа к памяти DMAC, трехканальный счетчик таймера, логика немаскируемого прерывания NMI;
4) коммутация запросов прерывания от устройств на шинах PCI и ISA, а также устройств на материнской плате на входы запросов контроллеров прерывания (PIC);
5) коммутация каналов DMA;
6) реализация моста с внутренней шиной X-bus используется традиционно в ПК для подключения контроллера клавиатуры, БИС энергонезависимой памяти с BIOS (Flash BIOS), часов реального времени;
7) реализация контроллера интерфейса USB;
8) поддержка системного мониторинга (управление SM Bus – System Monitoring Bus).

## 4.6. Средства мониторинга. System Monitoring Bus.

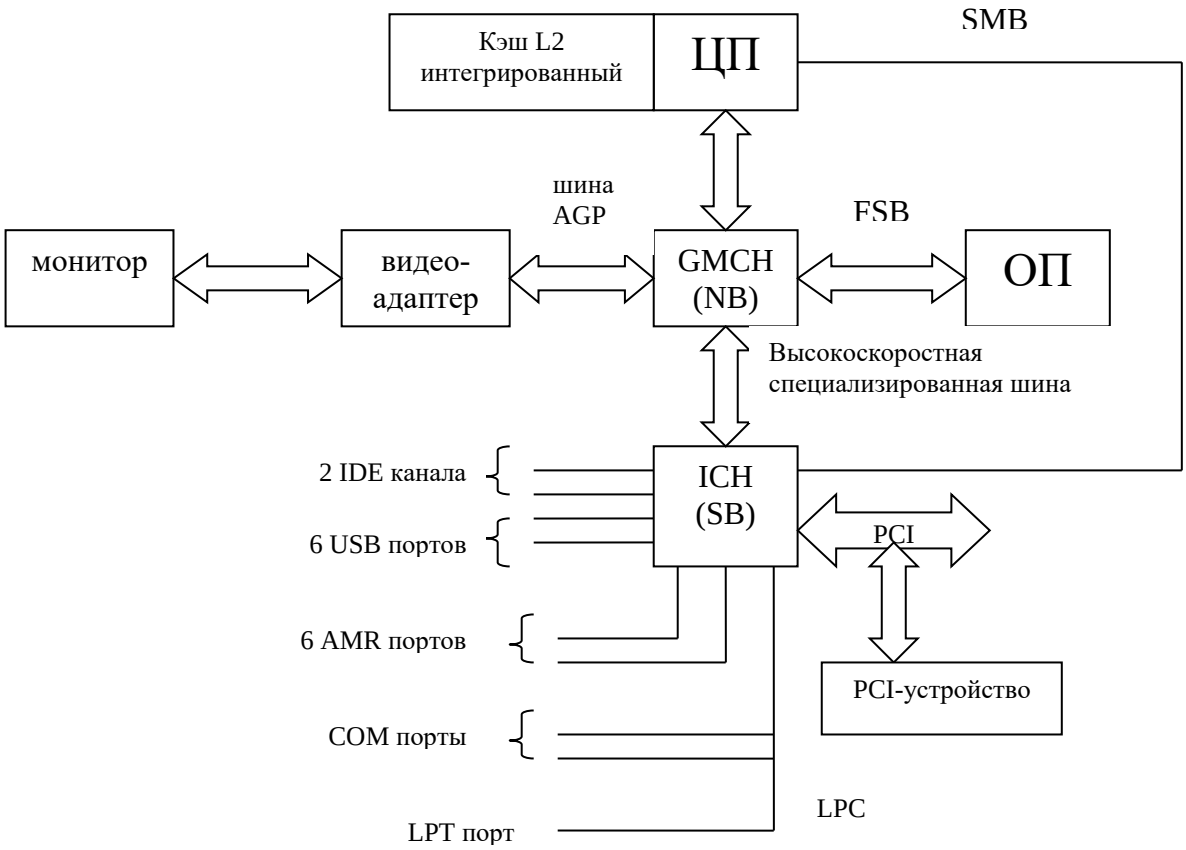
Современные чипсеты включают в состав, как правило, встроенный модуль мониторинга. Реализация мониторинга производится путем непрерывного контроля значений, снимаемых с датчиков температуры (до 8 штук) и напряжения (до 8 штук) и сравнение их с пороговыми значениями. При выходе за пределы включается сигнализация.

Реализация расширенного мониторинга подразумевает использование обратных связей, в частности снижения частоты ЦП до нормализации температуры. Наиболее развитой является обратная связь от датчиков температуры к управлению вентиляторами (до 3 штук).

## 4.7. Основные недостатки многошинной структуры ПК на базе процессора Pentium.

1. Использование для связей мостов NB и SB сравнительно низкоскоростной шины PCI.
2. Использование морально устаревшей шины ISA в качестве дополнительной шины расширения для подключения некоторых низкоскоростных устройств.

## 4.8. Упрощенная многошинная структура компьютера на базе процессора Pentium 4.




### 4.8.1.Основные особенности структуры

1. Расширены функции мостов, в результате чего в рамках приведенной структуры они называются не мостами, а концентраторами (хаб). Для NB используется наименование GMCH – Graphics and Memory Controllers Hub – концентратор контроллеров графики и памяти. Для SB – ICH – Integrated Controllers Hub или Input/Output Controllers Hub – концентратор контроллеров В/В.

2. Связь между хабами GMCH и ICH осуществляется не по стандартизованной шине PCI, как в предыдущей структуре, а по отдельной высокоскоростной специализированной шине (шина является закрытой).

3. Отсутствует шина ISA, вместо нее введена шина LPC – Low Pin Count – малое число контактов, особенностью которой является отсутствие слотов (разъемов).

4. Добавлены AHR-порты для подключения модемов и звуковых карт через интерфейс AC-Link. В соответствии со спецификацией AC’97 – Audio Codec фирмы Intel на архитектуру и параметры звуковых карт существует разделение модемов и звуковых карт на аналоговые и цифровые чипы. Это позволяет создавать дешевые модемы и звуковые карты, на которых присутствует только аналоговый чип. Цифровая обработка реализуется ЦП. Для таких звуковых карт разработан AMR-порт, что

означает Audio Modem Riser. Интерфейс порта AC-Link встраивается в хаб, при этом реализуется поддержка шести портов (каналов) AMR.

5. Использование специальной шины AGP для подключения монитора. AGP – Accelerated Graphics Port – ускоренный графический порт. Разработан фирмой Intel в 1997 году, является специализированным портом В/В для реализации высокопроизводительной графики.

AGP предназначен только для подключения видеоадаптера, позволяя ему использовать ОП и избавляя его от необходимости делить с другими устройствами шину PCI. На практике у современных видеоадаптеров имеется большой объем локальной видеопамяти, в результате чего поток данных циркулирует внутри видеоадаптера, слабо нагружая внешнюю шину, однако при построении 3D-изображения видеоадаптеру становится “тесно” в ограниченном объеме видеопамяти и его поток данных выходит на внешнюю шину, по составу сигналов напоминающую PCI. Чипсет системной платы через GMCH связывает AGP с ОП через системную шину FSB, не пересекаясь с “узким местом” в виде шины PCI. Ускоренность AGP обеспечивается специальными особенностями принципов ее функционирования:

1) конвейеризация обращений к памяти;
2) сдвоенная передача данных;
3) демультиплексирование шин адреса и данных.

Максимальная пропускная способность шины в режиме счетверенной передачи составляет 1064 Мб/с. Реализация стандарта шины AGP потребовала существенного усложнения чипсета по сравнению с базовым вариантом подключения адаптера к шине PCI, но при этом существенно снизилась нагрузка на шину PCI.