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.

<u>Структурная организация</u> 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.

<u>Функциональная организация</u> 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.

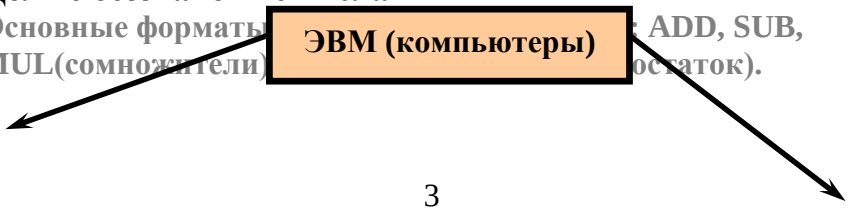
Под форматом данных обычно понимают внутреннее представление данных, в первую очередь, разрядность и назначение битов. Например, для знаковых целых чисел крайний левый бит формата отводится для представления знаков. Для обозначения форматов стандартной длины принято использовать следующие наименования:

- **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: ADD, SUB, MUL(сомножители) остаток).



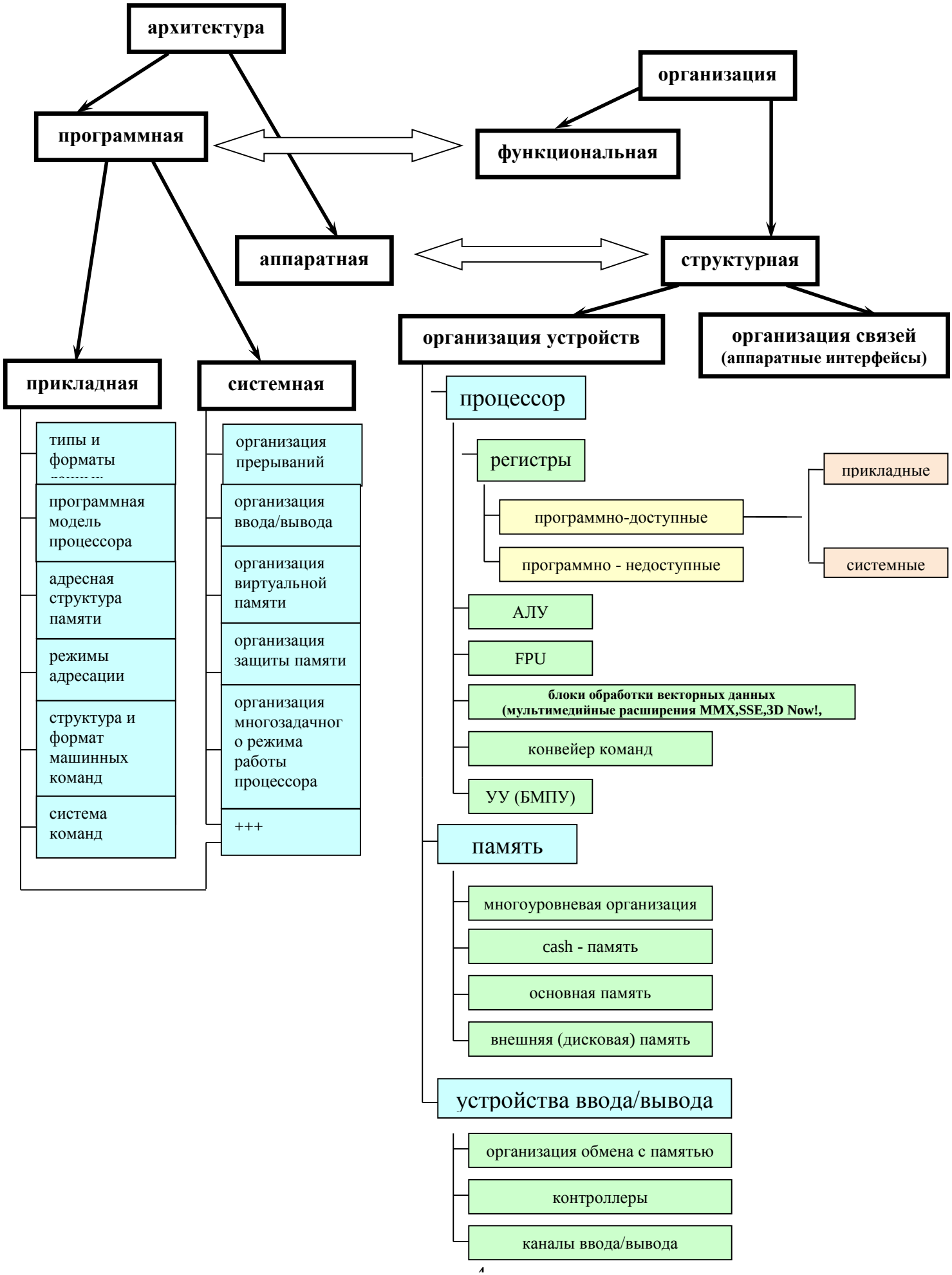


Рис.1.Обобщенное представление архитектуры, организации ЭВМ и их элементов.

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**

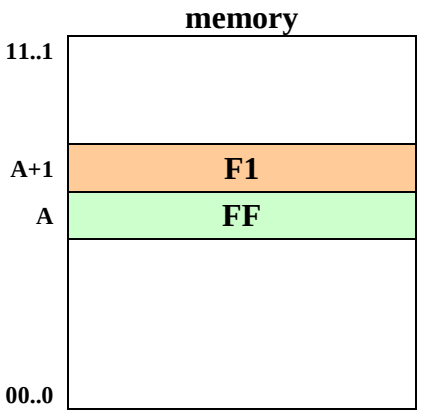
Адресная структура основной памяти и принципы размещения информации в ней. Принципы формирования физического адреса.

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

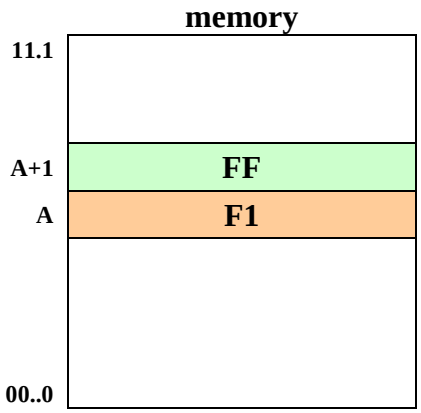
В подавляющем большинстве адресация памяти осуществляется на уровне байт. В соответствии с этим для программы память представляется в виде массивов последовательно адресуемых байтов. При размещении единицы информации, кратных байту, например Word или Doable Word, в памяти компьютера, адрес единицы информации 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.

<u>Принцип целочисленной границы гласит:</u>
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.

<u>Режимы адресации.</u>

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.

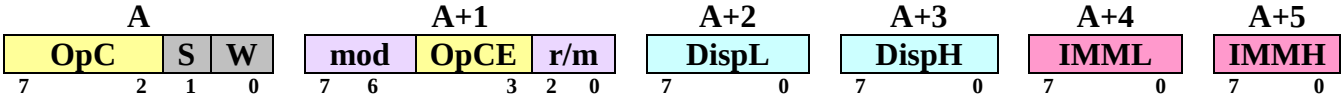
Структура машинной команды задаёт основные части (поля) машинной команды и определяет их назначение. В свою очередь, формат машинной команды определяет разрядность, как всей команды, так и отдельных её полей. Под полем принято понимать совокупность последовательных битов формата (команды или данных), 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

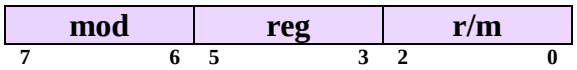


Специальные биты OpC - W(Word) и S(Sign Extended). Расширение Word определяет длину операндов (W=0 -> байт, W=1 -> слово). А S=1 - необходимость знакового расширения непосредственного операнда.

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.

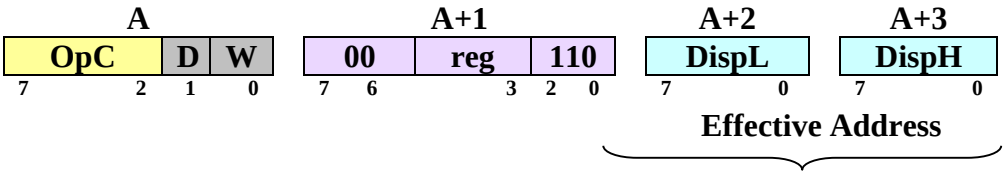
Для операнда, находящегося в памяти, поле r/m (register/memory) задаёт компоненты эффективного адреса Base и/или Index неявным образом (см. таблицу кодирования). В базовой модели для базовой компоненты EA могут использоваться только регистры BX и BP. А для индексной компоненты только регистры SI и DI.

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

адреса в ОП



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:

<table>
<tr><th>Модель процессора</th><th>Полное название</th><th>Фирма изготовитель</th><th>Полное название</th></tr>
<tr><td>SPARC</td><td>Scaleable Processor ARChitecture</td><td rowspan="5">SUN Microsystems</td><td rowspan="5">SUN Microsystems</td></tr>
<tr><td>Micro SPARC</td><td rowspan="3">32 разрядные</td></tr>
<tr><td>Super SPARC</td></tr>
<tr><td>Hyper SPARC</td></tr>
<tr><td>Ultra SPARC</td><td>64 разрядный</td></tr>
<tr><td>ALPHA</td><td>ALPHA</td><td>DEC/HP</td><td>Digital Equipment Corporation/Hewlett Packard</td></tr>
<tr><td>Power PC</td><td>Performance Optimized With Enchanced RISC</td><td>IBM/Motorola</td><td>International Business Machines/Motorola</td></tr>
<tr><td>PA RISC</td><td>Precision Architecture (64разрядный)</td><td>HP</td><td>Hewlett Packard</td></tr>
</table>

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.

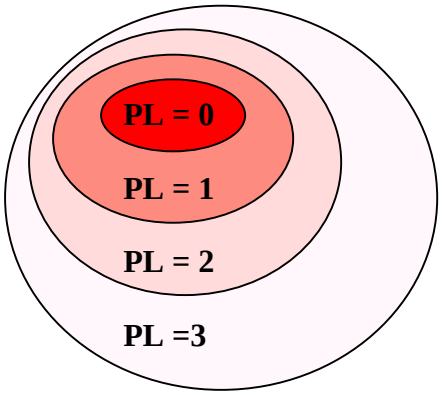
Аналогичный бит режима System/User имеет место в моделях системы VAX(DEC), который находится в PS (Processor Status).

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.

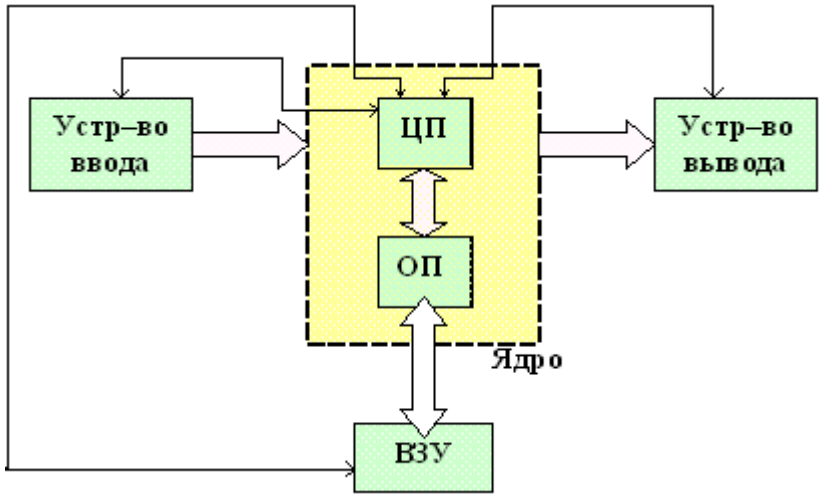
Современные операционные системы, в частности Windows и Unix поддерживают только 2 уровня привилегий (User/Supervisor). Соответствующие биты для двух уровней используются на страничном уровне и размещаются в страничных дескрипторах. В процессорах семейства Intel к привилегированным командам относятся:

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:

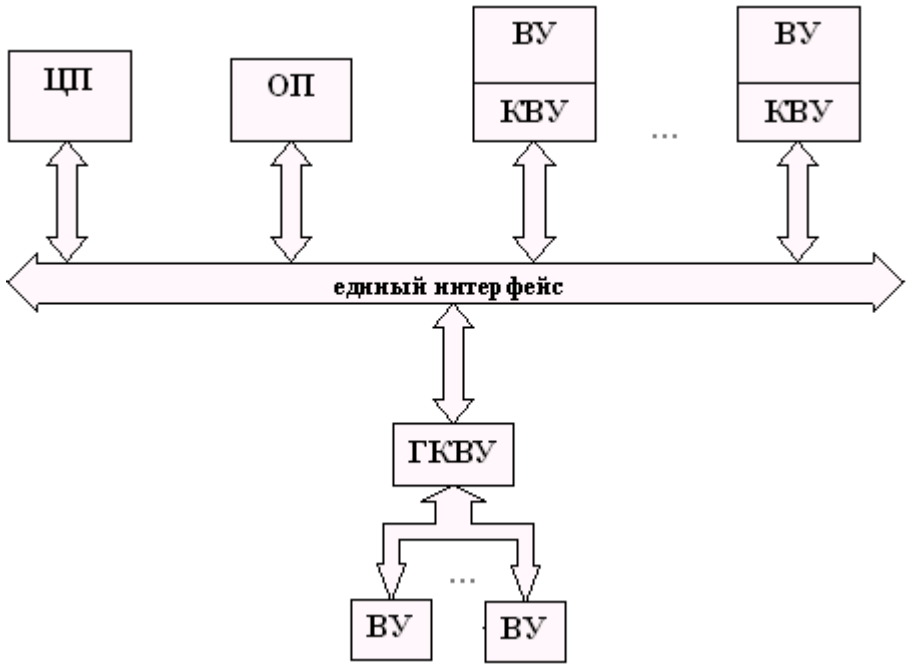
1) все устройства, как центральные, так и периферийные, для связи между собой используют одни и те же шины адреса, данных и управления;

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.

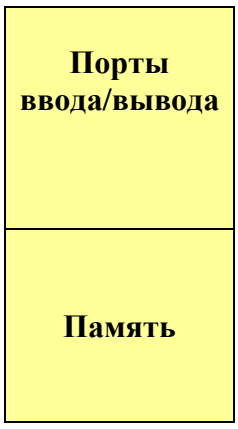
Использование раздельного адресного пространства предполагает, что одни и те же адреса могут применяться как для адресации и так и для адресации портов ввода/вывода. При этом в системе команд процессора должны использоваться специальные команды для ввода/вывода, отличные от команд обмена с памятью. Достоинством 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.



а

Некоторая область адресного пространства в старших адресах используется не для адресации памяти, 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.

Основные отличия организации прерываний в защищенном режиме по сравнению с реальным режимом.

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.

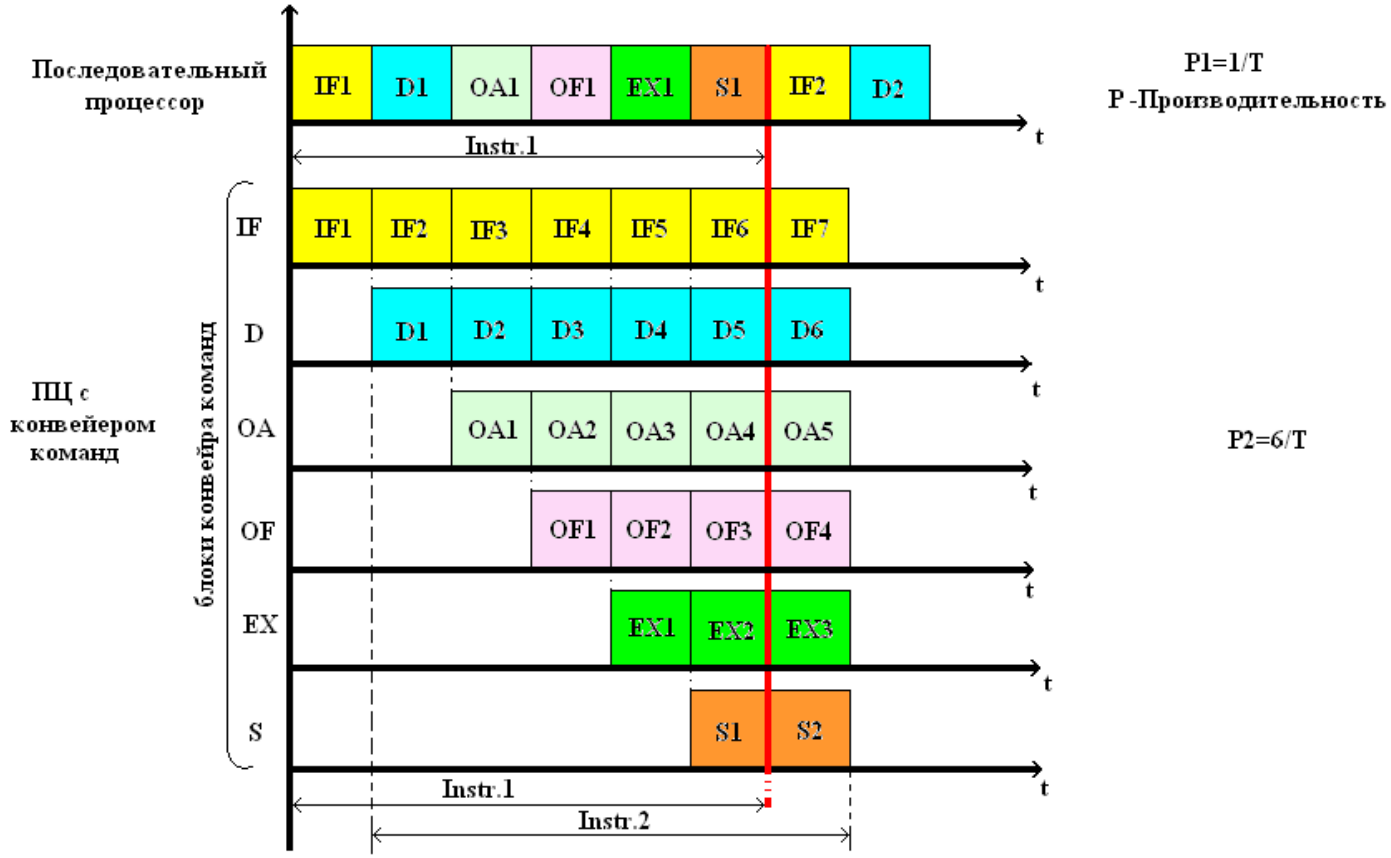
При классическом подходе выполнение основных машинных команд (например, арифметических или логических), связанных с обработкой данных, можно разделить на 6 этапов (фаз, стадий):

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.

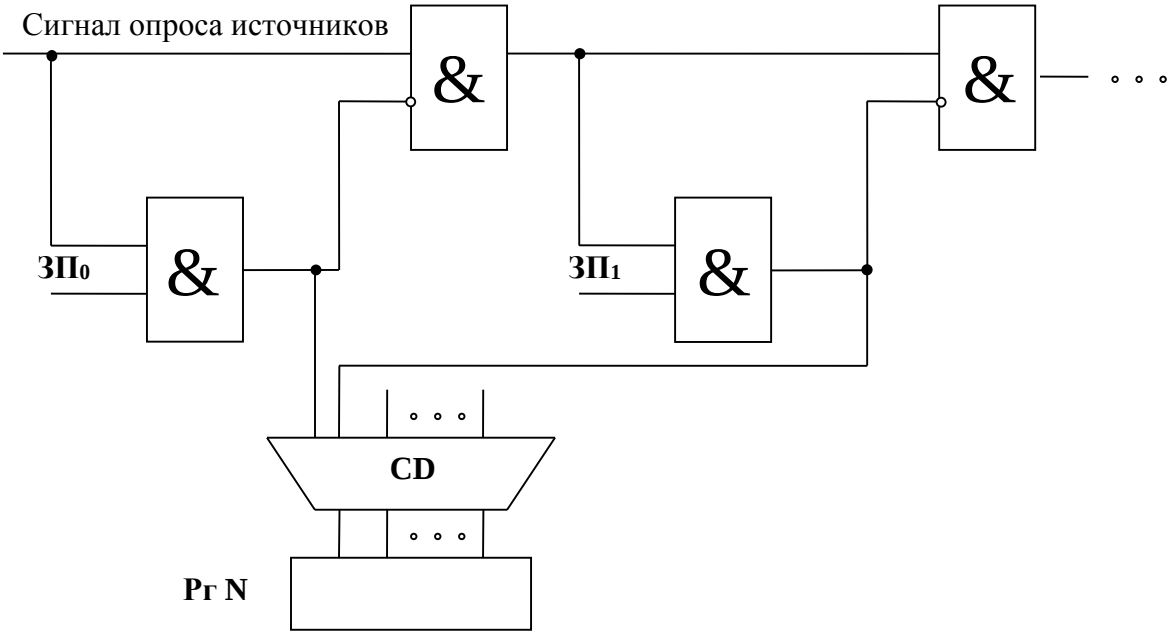
Реализация функций на аппаратном и программном уровнях

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:

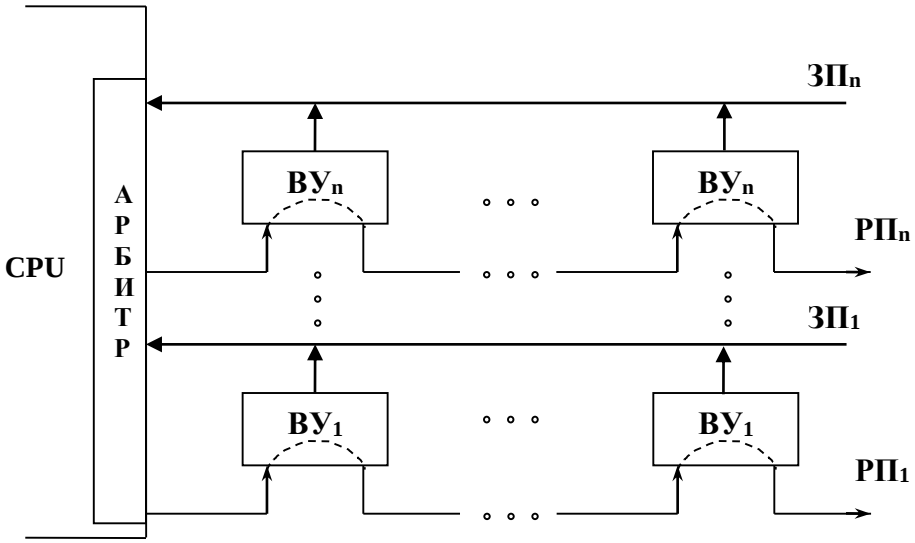
- недействительная операция;
- денормализованный операнд;
- деление на ноль;
- переполнение порядка;
- исчезновение порядка (антипереполнение);
- потеря точности.

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).

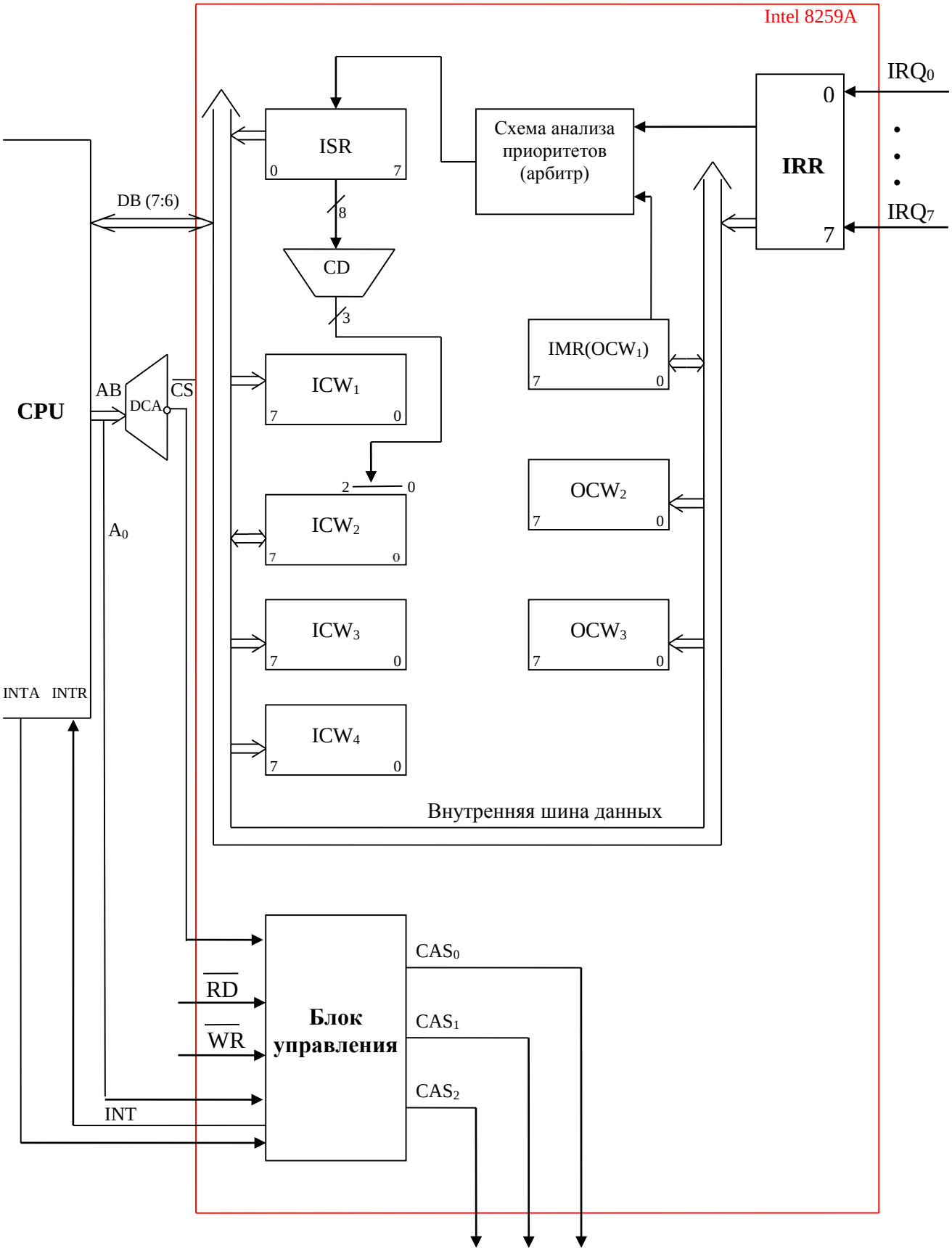
3) Выделение наиболее приоритетного запроса из всех поступивших и не замаскированных.

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.

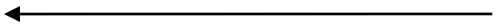
Основные режимы работы PIC

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:

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

$\longleftarrow$

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

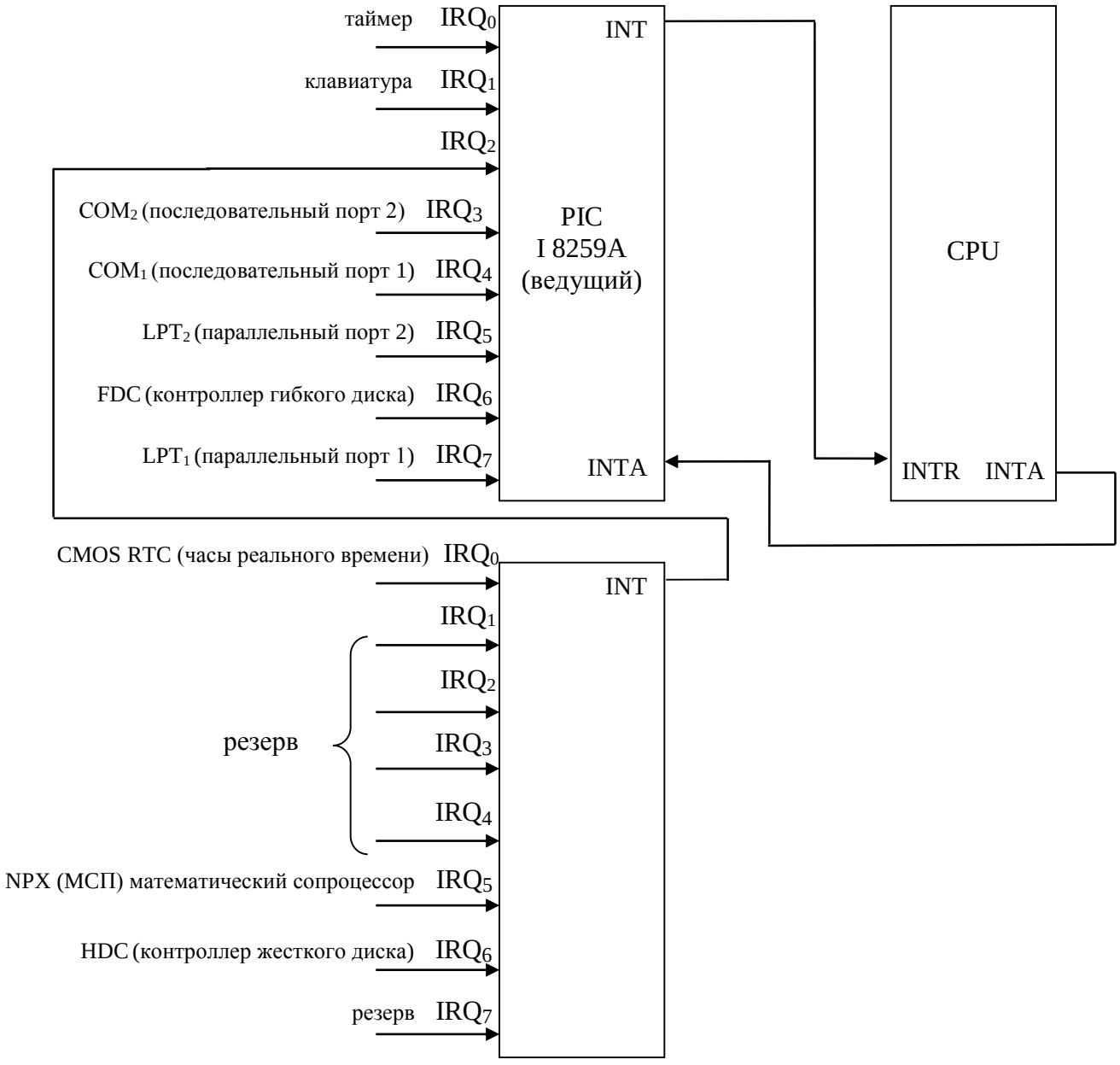
Старший бит I (Interrupt) установлен при наличии незамаскированного запроса в IRR. Младшие 3 бита содержат уровень выделенного запроса с наивысшим приоритетом. По активному уровню сигнала чтения (RD), сгенерированному командой IN, происходит установка соответствующего бита в ISR, как будто был получен 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
3 – указатель команды IP
} адрес возврата.

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.

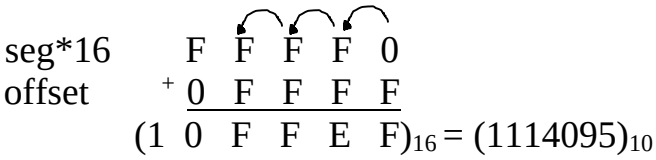
Попытка выполнения этих команд в R-режиме приводит к прерыванию стандартного типа 6 – «некорректный код команды».

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:



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

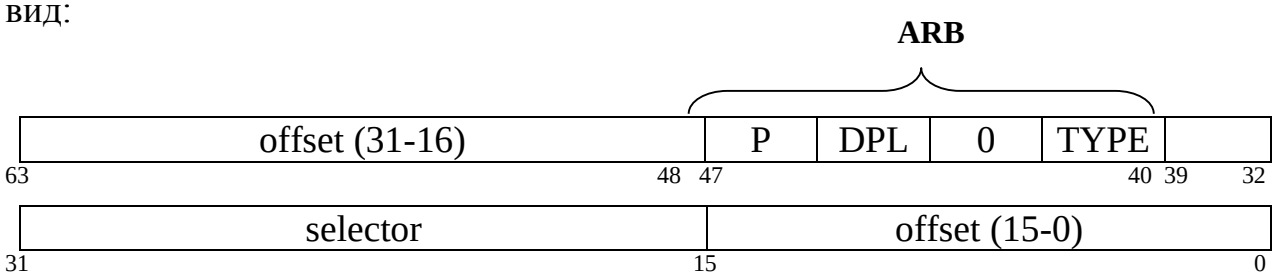
В отличии от таблицы векторов прерываний, местоположение которой в памяти является строго регламентированным (она находится в младших адресах), IDT, в принципе, может размещаться в любом месте линейного адресного пространства, однако, как правило, для сохранения преемственности ее также располагают в 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:



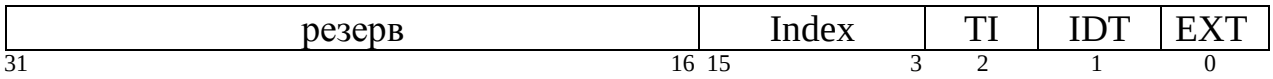
Для шлюза прерываний TYPE = Eh.
Для шлюза ловушек TYPE = Fh.

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.

№	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 (CRO))	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 случай	Второй особый 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).

Основными действиями при переключении задач являются:

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.

Прикладная архитектура процессора Intel 8086

СОДЕРЖАНИЕ

1. ТИПЫ И ФОРМАТЫ АППАРАТНО ПОДДЕРЖИВАЕМЫХ ДАННЫХ

В первом приближении информацию, используемую в ЭВМ, можно разделить на команды, адреса и данные.

Под *аппаратной поддержкой данных* определенного типа, представленных в некотором формате, понимается наличие в системе команд ЭВМ таких команд, которые предназначены для обработки данных этого типа, представленных в соответствующем формате.

Классификация данных см. рис.1.1.

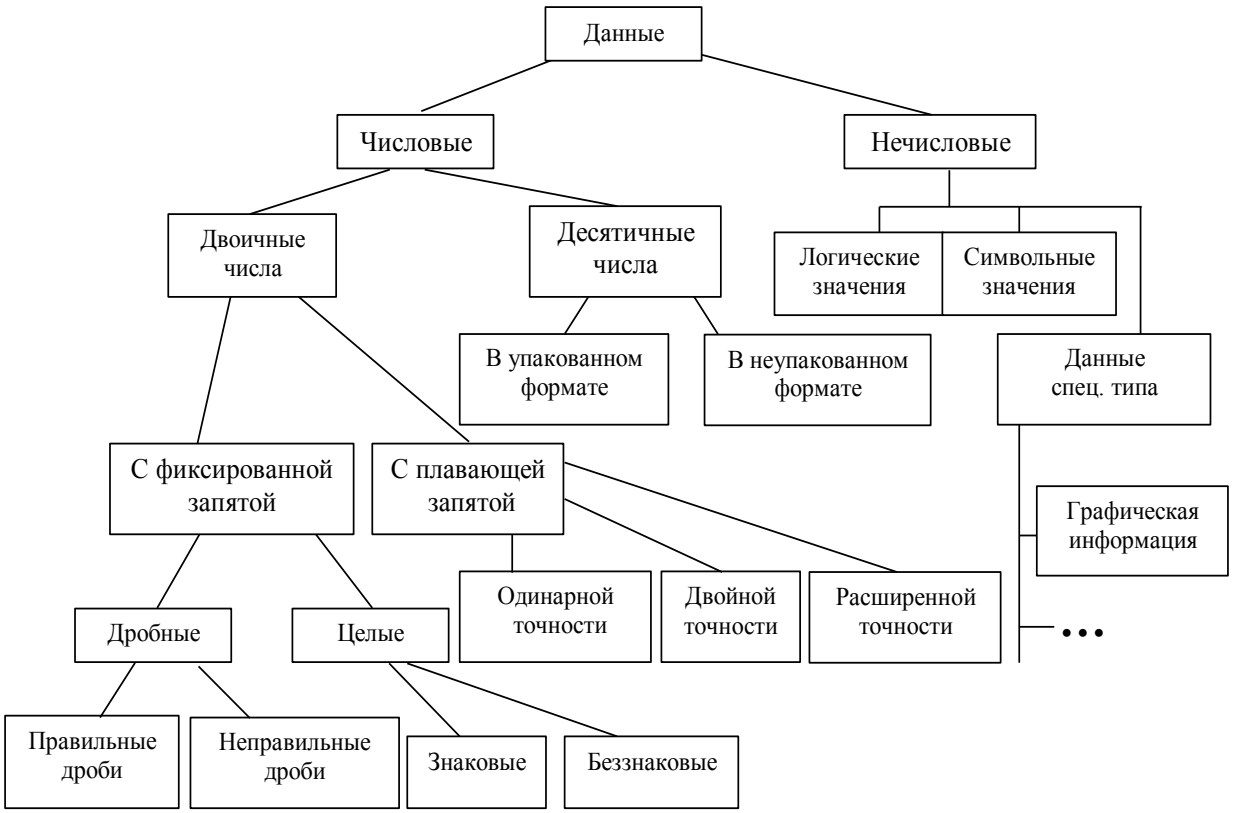


Рис. 1.1. Классификация данных

1.1. ЧИСЛА С ФИКСИРОВАННОЙ ЗАПЯТОЙ

Их деление на 2 типа (дробные и целые) определяется местоположением запятой в числе: слева (перед старшим разрядом) – дробные числа, справа (после младшего разряда) – целые числа.

Дробные числа как таковые в современных ЭВМ не используются. Они используются лишь для представления мантисс в числах с плавающей запятой.

В правильных дробях целая часть нулевая, в неправильных –не нулевая.

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

Особенностью представления знаковых целых чисел является использование дополнительного кода. Дополнительный код n-разрядного целого числа X определяется по правилу:

$$[X]_{ДК} = \begin{cases} X, & при\ X \geq 0 \\ 2^n - |X|, & при\ X < 0 \end{cases} \qquad (1.1)$$

В свою очередь под прямым кодом знакового числа подразумевается его представление в виде:

$$[X]_{ПК} = \begin{cases} X, & при\ X \geq 0 \\ 2^{n-1} + |X|, & при\ X < 0 \end{cases} \qquad (1.2)$$

Во многих литературных источниках предлагается считать, что дополнительный код положительного числа совпадает с его прямым кодом.

Прямой код отрицательного числа образуется путем записи единицы в знаковый разряд и модуля числа в цифровые разряды.

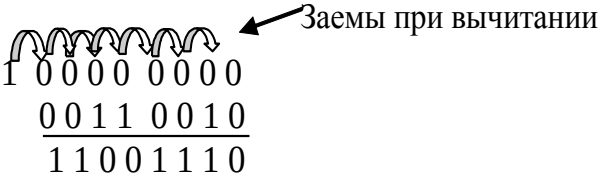
Пример 1.1.а. – Представление чисел [+50] и [-50] в байтном формате (n = 8) в прямом коде.

$[+50]_{ПК}$ = 0.0110010
$[-50]_{ПК}$ = 1.0110010

Прямой код для представления отрицательных чисел в современных ЭВМ не используется, его целесообразно использовать лишь при ручных операциях для проверки корректности отрицательного результата, представленного в дополнительном коде.

Исходя из приведенного выше правила получения дополнительного кода отрицательного числа, необходимо выполнить вычитание модуля числа из константы 2^n, представленной единицей в n+1 разряде и n нулями.

Пример 1.1.б. – Представить число [-50] в дополнительном коде.



2^n = 1 0000 0000
$[+50]_{ПК}$ = − 0011 0010
$[-50]_{дк}$ = 1100 1110

На принципе, рассмотренном в примере, основывается аппаратная поддержка преобразования чисел из прямого кода в дополнительный или из дополнительного в прямой. В процессоре Intel 8086 это преобразование реализуется с помощью команды

NEG (изменение знака). Выполнение этой команды сводится к вычитанию операнда из нуля, что дает такой же результат как в примере 1.1.б.

Для ручного преобразования из прямого в дополнительный код можно использовать один из следующих способов:

1. Инвертирование всех разрядов прямого кода с последующим добавлением единицы в младший разряд.
2. Младшие нули, включая первую младшую единицу, прямого кода сохраняются и в дополнительном коде, а остальные разряды инвертируются.

Примечания:

1. Если инвертирование распространяется на старший (крайний левый) разряд, интерпретируемый как знак, то преобразование из прямого кода в дополнительный меняет знак числа.
2. Если инвертирование не распространяется на старший (крайний левый) разряд, интерпретируемый как знак, то преобразование из прямого кода в дополнительный не меняет знак числа.
3. Преобразование из дополнительного кода в прямой осуществляется по аналогичным правилам, что и преобразование из прямого кода в дополнительный код.

1.2. ДИАПАЗОН ПРЕДСТАВЛЕНИЯ ЦЕЛЫХ ЧИСЕЛ

Диапазон для знаковых чисел:

$$-2^{n-1} \le A_{ц}^{зн} \le 2^{n-1} - 1. \qquad (1.3)$$

Диапазон представления целых знаковых чисел (см. формулу 1.3) не симметричен относительно нуля, как бы сдвинут на 1 единицу в отрицательную область, что объясняется тем фактом, что к области положительных чисел относится и ноль. Из этого следует, что максимальное по модулю отрицательное число не имеет аналога в области положительных чисел. Попытка изменения знака у этого числа (например, с помощью команды NEG) приводит к переполнению формата.

Для байтного формата (n = 8) диапазон знаковых целых чисел:

$$-2^{7} \le A_{ц}^{зн} \le 2^{7} - 1, \qquad (1.4.а)$$
$$-128 \le A_{ц}^{зн} \le 127.$$

Для двухбайтного формата (n = 16) диапазон знаковых целых чисел:

$$-2^{15} \le \grave{A}_{ö}^{çí} \le 2^{15} - 1, \qquad (1.4.б)$$
$$-32768 \le \grave{A}_{ö}^{çí} \le 32767.$$

Представление границ диапазона знаковых чисел в байтном формате:

-128 = | 10000000 |
|---|
| 7 0 |

127 = | 01111111 |
|---|

7 0

Диапазон для беззнаковых чисел:

$$0 \le A_{ö}^{á/çí} \le 2^n - 1 \qquad (1.5)$$

Для байтного формата (n = 8) диапазон беззнаковых целых чисел:

$$0 \le A_{ц}^{б/зн} \le 2^8 - 1 = 255.$$

Для двухбайтного формата (n = 16) диапазон беззнаковых целых чисел:

$$0 \le A_{ц}^{б/зн} \le 2^{16} - 1 = 65535.$$

Аппаратная поддержка целых чисел как знаковых, так и беззнаковых, осуществляется на уровне арифметических команд, причем в командах сложения ADD и вычитания SUB отсутствует разделение представляемых операндов и, соответственно, результатов на знаковые и беззнаковые целые. Соответствующая интерпретация используемых чисел при программировании на ASSEMBLER возлагается на программиста.

Единственное аппаратное отличие знаковых чисел от беззнаковых проявляется в способе фиксации переполнения при сложении. Для знаковых чисел переполнение фиксируется с помощью флага OF, а для беззнаковых чисел с помощью флага CF. Для команды вычитания флаг OF фиксирует переполнение при знаковой интерпретации чисел. Установка же флага CF при беззнаковой интерпретации свидетельствует о том, что результат вычитания отрицательный (уменьшаемое меньше вычитаемого) и представлен в дополнительном беззнаковом коде.

Пример 1.2. – Выполнить операцию сложения чисел A=59 и B=73 с одинаковыми знаками.

n = 8
|A|=59=$(111011)_2$
|B|=73=$(1001001)_2$

1) +A +B

Переносы при сложении

	Знаковая интерпретация (ЗИ)	Беззнаковая интерпретация (БЗИ)
+A= 0 0 1 1 1 0 1 1	+ 59	59
+B= 0 1 0 0 1 0 0 1	+ 73	+ 73
$C_{ДК}$= 1 0 0 0 0 1 0 0		132
$C_{ПК}$= 1 1 1 1 1 1 0 0	-124?	верно
	переполнение	

Для знаковой интерпретации полученный результат является некорректным вследствие возникшего переполнения, для беззнаковой интерпретации результат операции корректен.

2) (-A) + (-B)

	ЗИ	БЗИ
$[-A]_{дк}$= 1 1 0 0 0 1 0 1	(-59)	197
$[-B]_{дк}$ = 1 0 1 1 0 1 1 1	(-73)	183
C = 0 1 1 1 1 1 0 0	+124?	124?
	переполнение	переполнение

В приведенном примере сложения отрицательных знаковых операндов результат оказывается некорректен как для знаковой интерпретации, так и для беззнаковой интерпретации чисел. О некорректности беззнакового сложения можно судить по наличию переноса из старшего разряда, который аппаратно фиксируется во флаге CF. О наличии переполнения при знаковом сложении можно судить с использованием одного из двух способов:

1. Сравнение знаков операндов и результата.
   Если знаки операндов одинаковы, а знак суммы отличается от них – фиксируется переполнение.
2. Сравнение переносов из старшего цифрового разряда в знаковый и из знакового разряда за пределы формата.
   Если один из этих переносов имеет место, а другой отсутствует, то фиксируется переполнение. Именно этот способ используется для фиксации переполнения в процессорах фирмы Intel.

Если при сложении переполнение может иметь место только при одинаковых знаках операндов, то при вычитании - только при разных знаках операндов.

Пример 1.3. - Выполнить операцию вычитания чисел *A*=59 и *B*=73 с разными знаками. Фиксация переполнения в операциях знакового вычитания по аналогии с операцией

1) (+A) – (-B)

	заемы при вычитании	ЗИ	БЗИ
[+A] =	0 0 1 1 1 0 1 1	+59	59
$[-B]_{дк}$ =	1 0 1 1 0 1 1 1	-73	183
$C_{ДК}$ =	1 0 0 0 0 1 0 0		132?
$C_{ПР}$ =	1 1 1 1 1 1 0 0	-124?	некорректно
		переполнение	

2) (-A) – (+B)

		ЗИ	БЗИ
$[-A]_{дк}$ =	1 1 0 0 0 1 0 1	-59	195
[B]=	0 1 0 0 1 0 0 1	+73	73
C=	0 1 1 1 1 1 0 0	+124?	124
		переполнение	верно

сложения может осуществляться одним из двух способов:

1. Сравнение знаков операндов и результата.
   Если знаки операндов различны и знак результата отличается от знака первого операнда (уменьшаемого), фиксируется переполнение.

2. Сравнение заемов из знакового разряда в старший цифровой и в знаковый разряд из-за пределов формата.
   Если один из этих заемов имеет место, а другой отсутствует, фиксируется переполнение. Если оба заема либо отсутствуют, либо имеют место, результат знакового вычитания корректен.

1.3. ЧИСЛА С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

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

$$A = (-1)^{signA} \cdot M_A \cdot S^{P_A} \qquad (1.6)$$

где *signA* – знак числа *A* (0 – для положительных, 1 – для отрицательных),
M_A – мантисса числа *A*,
S – основание порядка,
P_A – порядок числа *A*.

В классическом представлении мантисса числа представляется правильной дробью со старшей значащей цифрой.

Порядок числа представляет собой целое число со знаком.

В качестве основания порядка в современных ЭВМ используется *S* = 2 (двоичное представление мантиссы) и *S* = 16 (16-ричное представление мантиссы). Основание 2 используется в миникомпьютерах, а 16 – в компьютерах типа Main Frame, а также в суперЭВМ.

$(-15{,}87)_{10} = (-1)^1 \cdot 0{,}1587 \cdot 10^{+2}$
$(+0{,}0052)_{10} = (-1)^0 \cdot 0{,}52 \cdot 10^{-2}$

- примеры записи чисел в форме (1.6) с десятичным представлением мантиссы (S=10).

Основными особенностями представления чисел с плавающей запятой в современных ЭВМ являются:

1. Преимущественное использование так называемых нормализованных чисел.

Число с плавающей запятой называется *нормализованным*, если старшая цифра его мантиссы является значащей (не ноль).

Подавляющее использование именно нормализованных чисел связано с требованиями повышения точности при выполнении различных операций, т.к. именно нормализованные числа обладают наименьшей погрешностью по сравнению с ненормализованными.

2. Порядок числа представляется не как целое число со знаком в явном виде, а в виде беззнакового числа, называемого смещенным порядком или характеристикой. При этом характеристика отличается от порядка на некоторую фиксированную для данного формата величину, называемого смещением (или смещением порядка).

$X_A = P_A + d$ (X_A – характеристика числа *A*, *d* – смещение порядка). (1.7)

Существует 2 подхода к выбору величины смещения:

1) Величина смещения равна весу старшего разряда смещенного порядка (характеристики).

2) Величина смещения равна весу старшего разряда смещения порядка, уменьшенного на 1.

Первый подход используется в основном в больших ЭВМ, второй в миникомпьютерах, в том числе в ПК.

3. Для представления чисел с плавающей запятой в рамках конкретной модели ЭВМ используется несколько форматов, как правило, 2 или 3, отличающихся разрядностью мантиссы и, в некоторых случаях, порядка. Эти форматы, как правило, носят название:

1) короткий формат (32 бита) или формат одинарной точности;

2) длинный формат (64 бита) – формат двойной точности;

3) расширенный формат (128 или 80 бит) – формат расширенной точности.

Для последнего формата длина 128 бит является типичной для больших ЭВМ, а 80 бит – для миникомпьютеров, в том числе и для ПК.

Использование разнообразных форматов позволяет реализовать различные требования к точности и, возможно, диапазону представления чисел. При выборе формата для конкретных вычислений следует исходить из точности вычислений и их скорости.

В использовании различных форматов в современных ЭВМ существуют 2 тенденции к расширению форматов:

1) формат расширяется только за счет увеличения разрядности мантиссы (разрядность порядка сохраняется).

2) формат расширяется за счет увеличения как мантиссы, так и порядка.

В первом случае расширение формата сопровождается только увеличением точности при неизменном диапазоне представления чисел.

Во втором случае расширение формата приводит как к увеличению точности, так и диапазона представления чисел.

Первый подход к расширению формата в основном используется в больших ЭВМ, а второй – в ПК.

4. Независимо от знака числа мантисса чисел с плавающей запятой представляется в прямом коде.

1.4. ОСОБЕННОСТИ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ В ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРАХ

Эти особенности регламентируются международным стандартом IEEE-754. Этот стандарт, кроме особенности представления чисел, оговаривает также все особые ситуации, связанные с обработкой чисел с плавающей запятой и стандартные реакции на эти ситуации.

1. В качестве основания порядка используется $S = 2$.

2. Мантисса числа представляется неправильной дробью, имеющей одну цифру в целой части. Вследствие того, что, как правило, используются только нормализованные числа с обязательной единицей в целой части мантиссы, эта единица в формате не представляется, а лишь подразумевается и носит название *скрытая единица* (скрытый разряд).

Скрытая единица имеет место только в коротком и длинном форматах. В расширенном формате она представляется явно.

3. Различные форматы имеют следующую структуру (см. рис.1.2):

Рис. 1.2. Форматы чисел

4. Порядок числа представляется со смещением в виде беззнакового целого числа, называемого *характеристикой*. Величина смещения равна весу старшего разряда характеристики, уменьшенному на 1.

Для различных форматов величина смещения различна и составляет:

КФ: $d = 2^7 - 1 = 127$,

ДФ : $d = 2^{10} - 1 = 1023$,

РФ: $d = 2^{14} - 1 = 16383$.

1.5. ДИАПАЗОН ПРЕДСТАВЛЕНИЯ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Его принято определять в отношении модуля нормализованного числа.

В общем виде диапазон представляется следующим двойным неравенством:

$$M_{A\min}^{H} \cdot S^{P_{AMIN}} \le \left|A_{пз}^{H}\right| \le M_{A\max}^{H} \cdot S^{P_{AMAX}} \qquad (1.8)$$

Диапазон представления нормализованной мантиссы, представленной неправильной m-разрядной дробью с обязательной единицей в целой части, имеет вид:

$$1 \le M_{A}^{H} \le 2 - 2^{-(m-1)} \qquad (1.9)$$

С учетом того, что единица целой части является скрытой для короткого и длинного форматов, диапазон представления мантиссы преобразуется:

$$1 \le M_{A}^{H} \le 2 - 2^{-m} \qquad (1.10)$$

При достаточно больших значениях m правую границу мантиссы можно считать с большой степенью точности равной 2.

Диапазон порядка как целого знакового числа определяется из диапазона характеристики.

При использовании *n* разрядов под характеристику ее диапазон как целого беззнакового числа определяется в виде:

$$0 \le X_{A} \le 2^{n} - 1 \qquad (1.11)$$

Стандартом IEEE 754 (854) предусматривается зарезервирование крайних значений характеристики для представления специальных значений типа NAN (Not A Number), ±∞, ¤ (неопределенность), ±0, а также денормализованных чисел. Таким образом, при определении реального диапазона характеристики и, следовательно, порядка следует исходить из неравенства (1.12):

$$1 \le X_{A} \le 2^{n} - 2$$
$$1 - d \le P_{A} \le 2^{n} - 2 - d \qquad (1.12)$$

1) *КФ*

$$1 \le X_{A} \le 254$$
$$-126 \le P_{A} \le 127$$
$$1{,}18 \cdot 10^{-38} \approx 1 \cdot 2^{-126} \le \left|A_{пз}^{H}\right| \le 2 \cdot 2^{127} \approx 3{,}4 \cdot 10^{38}$$

2) *ДФ*

$$2{,}23 \cdot 10^{-308} < \left|A_{пз}^{H}\right| < 1{,}79 \cdot 10^{308}$$

3) *РФ*

$$3{,}37 \cdot 10^{-4932} < \left|A_{пз}^{H}\right| < 1{,}18 \cdot 10^{4932}$$

На числовой оси диапазон представления располагается симметрично относительно нуля:

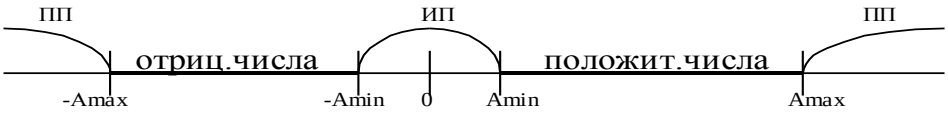


Рис. 1.3. Диапазон представления чисел

При выполнении арифметических операций результат операции может выходить за диапазон представления чисел.

Получение результата, меньшего по модулю минимального представления числа, приводит к особому случаю исчезновения порядка (ИП), а получение результата, по модулю большего максимального представления числа, – к особому случаю переполнения порядка (ПП).

Эти случаи фиксируются специальными флагами особых случаев, которые размещаются в регистре состояний АСП (FPU) – арифметического сопроцессора (Floating Point Unit).

Возникновение особых случаев может служить причиной для прерывания выполняемой программы.

В терминологии фирмы Intel особый случай исчезновения порядка называется *антипереполнением*.

Как правило, стандартной реакцией на особый случай антипереполнения является присвоение результату значения 0. В стандарте предусматриваются 2 возможных значения 0 – положительное и отрицательное, отличающихся значением знакового разряда при нулевых значениях характеристики и мантиссы.

1.6. ТОЧНОСТЬ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

В общем случае числа с плавающей запятой представляются в ограниченном формате приближенно. Точность представления чисел принято характеризовать абсолютной и относительной погрешностью.

Абсолютная погрешность является знаковой величиной и определяется как разность между приближенным (машинным) представлением числа и его точным значением.

$$\Delta A = A_M - A_T \qquad (1.13)$$

В свою очередь *относительная погрешность* есть беззнаковая величина, определяемая в виде:

$$\delta A = \left| \frac{\Delta A}{A_T} \right| \qquad (1.14)$$

В отношении форматов точность представления, как правило, задается максимальной относительной погрешностью представления чисел, инвариантной к способу их округления.

Для различных форматов значения погрешности:

$$\begin{aligned} &КФ : \delta A_{\max} \approx 10^{-7} \\ &ДФ : \delta A_{\max} \approx 10^{-16} \\ &РФ : \delta A_{\max} \approx 10^{-19} \end{aligned} \qquad (1.15)$$

1.7. ДЕСЯТИЧНЫЕ ЧИСЛА

Десятичные числа представляются в двоично-кодированной форме с использованием либо упакованного (PACK), либо неупакованной (UNPACK) формата.

В упакованном формате в каждом байте числа кодируются две цифры, в неупакованном – одна.

Для кодирования десятичных цифр используется в основном естественный двоичный код, обычно называемый 8421.

В этом коде:

0 – 0000
1 – 0001
…

9 – 1001

Частным случаем неупакованного формата является код ASCII (American Standart Code for Interchange Information), используемый в ПК. В этом коде десятичная цифра представляется в младшей тетраде байта, а старшая тетрада принимает стандартное значение 0011.

Упакованный формат обычно называют BCD-форматом (или BCD-числом – Binary Coded Decimal).

Пример 1.4.

785

BCD	0000 0111	1000 0101	
ASCII	0011 0111	0011 1000	0011 0101

Для кодирования знаков используются специальные ASCII-символы.

Десятичные числа используются в ЭВМ на этапах ввода и вывода информации.

Классическая схема выполнения действий над числами в ЭВМ имеет вид (см. рис.1.4):

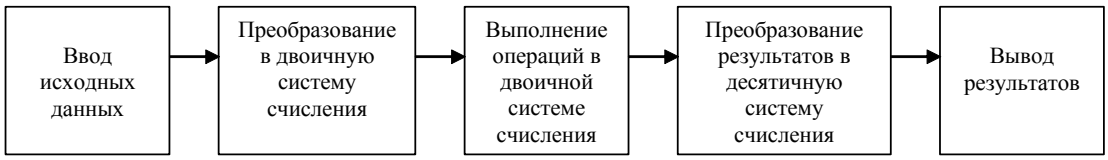


Рис. 1.4. Схема выполнения действий над числами в ЭВМ

В классической схеме (рис. 1.4) предполагается, что компьютер содержит двоичное АЛУ для выполнения операций над числами в двоичной системе счисления. Эта схема требует 2-кратного преобразования данных на этапе ввода и на этапе вывода. В зависимости от класса ЭВМ эти преобразования могут быть реализованы либо на аппаратном, либо на программном уровне.

При аппаратной реализации преобразования в системе команд процессора должны быть соответствующие команды. Так, например, система команд классической вычислительной системы 70-х годов IBM/370 включала в себя 2 команды - CDB (Convert Decimal to Binary), CBD (Convert Binary to Decimal) для преобразования десятичных чисел в двоичные (CDB) и двоичных чисел в десятичные (CBD).

В ПК подобных команд нет, т.е. преобразование реализуется на программном уровне.

Классическая схема обработки данных оказывается целесообразной только при решении так называемых научно-технических задач, которые характеризуются большим объемом вычислений при сравнительно небольшом объеме исходных данных.

Альтернативой научно-технических задач являются так называемые коммерческие задачи, которые характеризуются сравнительно небольшим объемом вычислений при очень большом объеме обрабатываемых данных. Для задач этого типа более целесообразно выглядит следующая схема обработки данных (рис.1.5):

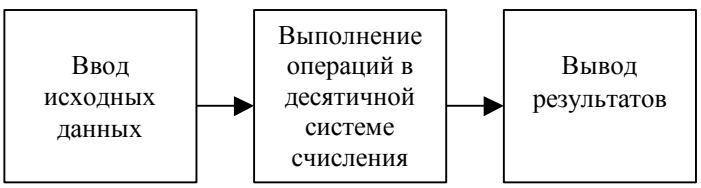


Рис.1.5. Схема обработки данных

Реализация этой схемы в рамках архитектуры компьютера требует наличия в составе центрального процессора десятичного АЛУ (естественно, наряду с двоичным). Как правило, наличие десятичного АЛУ является особенностью ЭВМ класса Main Frame (ЭВМ общего назначения). В ЭВМ этих классов, как правило, используется 3 специализированных АЛУ соответственно для выполнения операций над целыми числами, над числами с плавающей запятой и над десятичными числами.

В процессорах семейства Intel 80X86 аппаратная поддержка десятичных данных реализуется на уровне команд десятичной (BCD) и ASCII-коррекции. Основным назначением этих команд является приведение результата двоичной операции над десятичными числами в упакованном или неупакованном форматах в корректную десятичную форму. Это означает формальную возможность программной реализации псевдодесятичной арифметики с использованием команд двоичной обработки и последующим применением команд коррекции.

1.8. НЕЧИСЛОВЫЕ ДАННЫЕ

Логические значения – в них каждый бит стандартного формата несет собственную смысловую нагрузку. Аппаратная поддержка данных этого типа реализована на уровне логических команд, осуществляющих побитовую обработку.

Символьные значения – для описания любого символа представляющего собой букву, цифру, знак препинания и т.п., используется стандартная единица, длиной в байт. На уровне системы команд поддерживаются не столько одинарные символьные данные, сколько их элементарная структура, называемая строкой (цепочкой). Эти команды относятся к специальному классу команд обработки строк.

2. РЕГИСТРОВАЯ СТРУКТУРА (ПРОГРАММНАЯ МОДЕЛЬ) ПРОЦЕССОРА

2.1. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ

Регистровая структура процессора включает в себя 14 16-разрядных программнодоступных регистров и может быть представлена в следующем виде:

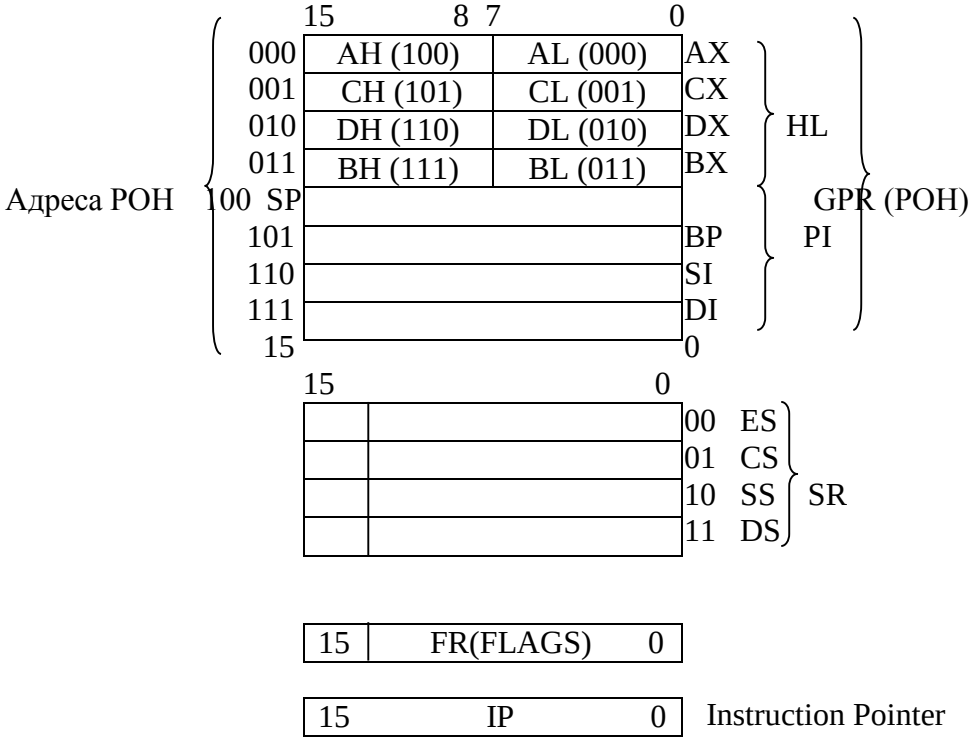


Рис.2.1. Регистровая структура процессора

Программно-доступные регистры разделяются на:

- регистры общего назначения (GPR – General Purpose Registers), группа включает 8 регистров;
- сегментные регистры (SR – Segment Registers), группа включает 4 регистра;
- регистр флагов (Flags);
- указатель команды (Instruction Pointer).

2.2. РЕГИСТРЫ ОБЩЕГО НАЗНАЧЕНИЯ

В отношении функционального назначения регистров, образующих внутреннюю регистровую память процессоров, существуют 2 противоположных подхода, реализуемых в архитектуре ЭВМ:

1. Полная специализация регистров, т.е. каждый регистр используется только по одному конкретному назначению.

2. Полная универсализация регистров, т.е. каждый регистр может использоваться по любому назначению.

В процессорах фирмы Intel используется промежуточный подход, сочетающий в себе частичную специализацию и частичную универсализацию регистров. Это означает, что по умолчанию любой регистр используется как специализированный для определенной цели, и в то же время его можно использовать и для других целей как универсальный регистр.

Использование любого регистра по его прямому назначению сокращает длину объектного кода программы по сравнению с любым другим использованием регистра, так как использование регистра по назначению, как правило, предполагает его неявную адресацию (адрес регистра не задается, но подразумевается).

Функциональная специализация РОНов отражается в их наименованиях:

AX – Accumulator (регистр-аккумулятор) – по умолчанию используется для задания одного из операндов команды и для представления результата.

CX – Counter (счетчик) – по умолчанию используется, во-первых, как счетчик числа повторения циклов в команде "организация цикла" (LOOP); во-вторых, для задания числа сдвигов в командах сдвигов (его младший байт – CL); в-третьих, для задания числа элементов обрабатываемых строк (цепочек) в командах обработки строк (MOVS, CMPS и т.д.).

DX –Data (регистр данных) – по умолчанию используется как расширение аккумулятора со стороны старших разрядов в командах умножения и деления.

BX –Base (базовый регистр) – по умолчанию используется как базовая компонента эффективного адреса операнда, находящегося в памяти. (В терминологии фирмы Intel под эффективным адресом – Effective Address (EA) – понимается адрес операнда, формируемый программой (программный адрес).) Для получения физического адреса ячейки памяти, в которой находится операнд, осуществляется преобразование EA на основе простейшей модели сегментированной памяти (механизма сегментации).

SP – Stack Pointer (указатель стека) – по умолчанию используется для адресации вершины стека.

Вершина стека указывает на адрес последнего элемента, записанного в стек.

Стек представляет собой сегмент памяти. Стек растет в область младших адресов. Это означает, что при записи (включении в стек), например, по команде PUSH, сначала производится декремент SP (уменьшение) на 2, а затем запись в память по новому значению SP как адреса.

При чтении (извлечении) из стека, например, по команде POP, сначала производится чтение слова по адресу из SP, а затем инкремент (увеличение) содержимого SP на 2.

Работа со стеком реализуется на уровне слов, но не байт.

BP – Base Pointer (указатель базы) – по умолчанию используется как базовая компонента эффективного адреса операнда в памяти по аналогии с BX.

Отличие в использовании содержимого регистров BX и BP как базовых компонент EA состоит в том, что при использовании BX подразумевается обращение к сегменту данных, а при использовании BP – к сегменту стека (но не через его вершину).

Подобный способ работы со стеком не через его вершину используется в программах на ASSEMBLER для обращения к параметрам процедуры, передаваемым через стек.

SI – Source Index (индекс источника) – по умолчанию используется для задания индексной компоненты EA, а также для адресации элементов строки-источника в командах обработки строк.

DI – Distination Index (индекс приемника) – по умолчанию используется аналогично SI для задания индексной компоненты EA, а также для адресации элементов строки-приемника в командах обработки строк.

Группу из 8 РОН принято делить на 2 части:
- группа HL (High – Low);
- группа PI (Pointer – Index).

Группу HL иногда называют регистрами данных, подразумевая ее преимущественное использование для операндов и результатов команд.

Регистры группы HL могут использоваться в командах в 2-байтном (полном) и в байтном (неполном) варианте.

Отдельные байты этих регистров используют то же наименование, что и полный регистр (A, C, D, B) с добавлением приставки L – младший, H – старший, для соответствующих байтов регистра.

Группа PI или группа указателей-индексов может использоваться только в 2-байтном варианте.

Для адресации РОН, как полных, так и не полных, в машинной команде используются 3 двоичных разряда.

Двоичные адреса полных РОН приведены на рис.2.1 слева, а их отдельных байт - в скобках.

2.3. СЕГМЕНТНЫЕ РЕГИСТРЫ

CS – Code Segment (сегмент кода – машинной программы),
SS – Stack Segment (сегмент стека),
DS – Data Segment (сегмент данных),
ES – Extra Segment (дополнительный сегмент).

Сегментные регистры используются для аппаратной поддержки простейшей модели сегментированной памяти, принятой в процессоре 8086. Их содержимое определяет базовые (начальные) адреса соответствующих сегментов в физической памяти. Использование 4 сегментных регистров предполагает, что в любой момент времени программа может работать (иметь доступ) с 4 сегментами памяти. С учетом того, что длина каждого сегмента составляет 2^{16} байт = 64 Кбайт (16-разрядный адрес) физическое адресное пространство, доступное программе, составляет 256 Кбайт.

Шина адреса процессора Intel 8086 является 20-разрядной, что обеспечивает емкость адресного пространства 2^{20} байт = 1 Мбайт. При формировании 20-разрядного физического адреса, содержимое соответствующего сегментного регистра смещается в сторону старших разрядов путем сдвига на 4 разряда влево. Таким образом, содержимое сегментных регистров представляет собой не сам физический начальный адрес сегмента, а его значение, уменьшенное на 16 (сегменты выровнены в памяти на границу параграфа).

Содержимое одного из сегментных регистров привлекается по умолчанию каждый раз при обращении процессора к памяти для формирования физического адреса, который и выставляется на шину адреса. Например, на этапе выборки команды по умолчанию привлекается регистр CS, при обращении к стеку – регистр SS. При обращении за операндом или при записи результата – регистр DS.

2.4. РЕГИСТР ФЛАГОВ

XXXX	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF
15 12	11	10	9	8	7	6	5	4	3	2	1	0

В регистре флагов актуальными являются только 9 битов, 6 из которых представляют собой арифметические флаги (флаги состояний) и 3 – флаги управления. Остальные 7 бит являются незадействованными (обозначаются X).

Арифметические флаги формируются в основном арифметическими командами, определяя результат арифметических операций (точнее, они являются признаками

результата). Кроме того, на арифметические флаги оказывают влияние логические команды и команды сдвигов. Флаги управления оказывают влияние на процесс выполнения программы.

К арифметическим флагам относятся:

CF – Carry Flag (флаг переноса). В нем фиксируется перенос из старшего разряда при сложении и заем в старший разряд при вычитании. При умножении значения флага CF определяет возможность (CF=0) или невозможность (CF=1) представления результата (произведения) в том же формате, что и сомножители.

С помощью флага переноса фиксируется переполнение при сложении беззнаковых чисел.

PF – Parity Flag – флаг паритета (четности). Он устанавливается при наличии четного числа единиц в младшем байте результата, в противном случае – сбрасывается. Этот флаг используется как аппаратная поддержка контроля по четности (нечетности).

AF – Auxiliary Carry Flag (флаг вспомогательного переноса). В этом флаге фиксируется межтетрадный перенос при сложении и межтетрадный заем при вычитании. Значение этого флага используется командами десятичной и ASCII-коррекции сложения и вычитания. Этот флаг можно рассматривать как аппаратную поддержку операций над десятичными числами.

ZF – Zero Flag (флаг нуля). Он устанавливается при нулевом результате операции, в противном случае (ненулевой результат) - сбрасывается.

SF – Sign Flag (флаг знака). В него копируется старший (крайний левый) бит результата, интерпретируемый как знак.

OF – Overflow Flag (флаг переполнения). Он устанавливается в командах сложения и вычитания в случае, если результат операции не помещается в формате операндов. При этом как операнды, так и результат интерпретируются как знаковые целые числа. В аппаратную установку этого флага положен принцип фиксации переполнения по сравнению переносов из двух старших разрядов при сложении или заемов в два старших разряда при вычитании. Если один из переносов (заемов) имеет место, а другой отсутствует, то происходит переполнение формата (разрядной сетки). В командах умножения флаг OF выполняет ту же функцию, что и флаг CF (их значения совпадают).

К флагам управления относятся:

TF – Trace (Trap) Flag (флаг трассировки (ловушки)). При установке флага TF процессор переводится в так называемый отладочный (покомандный, пошаговый) режим работы. В этом режиме завершение выполнения любой команды сопровождается выходом на прерывание специального типа (стандартный тип 1 – прерывание по отладке).

DF – Direction Flag (флаг направления). Его значение используется командами обработки строк (цепочек) и определяет направление обработки: от меньших адресов к большим (слева на право) при DF=0 или от больших адресов к меньшим (справа на лево) при DF=1.

IF – Interrupt Flag (флаг прерываний). С помощью этого флага разрешаются (IF=1) или запрещаются (IF=0) внешние прерывания, запросы которых поступают на специальный вход INTR (Interrupt Requester). Этот вход обычно связан с микросхемой PIC (Programmable Interrupt Controller) – программируемый контроллер прерываний.

2.5. РЕГИСТР IP (Instruction Pointer)

Содержимым регистра IP является так называемый продвинутый адрес команды. Это означает, что в момент выполнения какой-либо команды регистр IP указывает на адрес следующей команды.

В качестве адреса команды фигурирует адрес ее младшего байта (по адресации).

В связи с тем, что в базовой модели используется простейший двухступенчатый конвейер команд (I ступень – выборка команды; II ступень - остальные фазы (этапы) выполнения команды, к которым относятся: декодирование команды (определение типа), формирование адресов операндов, выборка операндов, выполнение операции в АЛУ, запись результата), фактическое содержимое регистра IP, который входит в блок предварительной выборки команд, указывает на очередной байт команды, выбираемый из памяти и помещаемый в специальный буфер, называемый очередью команд (IQ – Instruction Queue). Несмотря на этот факт, для выполняемой программы IP содержит адрес следующей команды. Фактически, на аппаратном уровне при необходимости использования IP (например, для его сохранения как адреса возврата) осуществляется соответствующая коррекция его содержимого с учетом числа байт, выбранных в IQ.

Конвейер команд служит одним из важнейших средств увеличения производительности компьютера. С его помощью реализуется параллелизм на уровне машинных команд. Это означает, что в любой момент времени в процессоре на стадии одновременного выполнения находится несколько последовательных машинных команд (по возрастанию адресов). Для аппаратной реализации конвейера команд отдельные фазы команды реализуются с помощью отдельных блоков конвейера. Блоки конвейера могут функционировать параллельно во времени независимо друг от друга. При использовании классического 6-ступенчатого конвейера команд (по числу фаз выполнения команды) и условии, что каждая фаза требует одинакового времени для реализации, полная загрузка конвейера команд в принципе обеспечивает 6-кратное увеличение производительности по сравнению с последовательным (бесконвейерным) процессором. В некотором смысле конвейер команд аналогичен производственному конвейеру.

3. ОСНОВНЫЕ РЕЖИМЫ АДРЕСАЦИИ, ИСПОЛЬЗУЕМЫЕ В ЭВМ

Под режимом адресации обычно понимается способ формирования так называемого исполнительного адреса операнда и/или результата на основе информации, содержащейся в адресной части команды.

Фактически, исполнительный адрес является программным адресом. В некоторых случаях исполнительный адрес совпадает с физическим, т.е. именно по нему производится обращение к памяти. Однако в подавляющем большинстве случаев исполнительный адрес не является физическим, а требует дальнейшего преобразования в физический адрес, которое, как правило, осуществляется с использованием механизма сегментации и, возможно, страничной организации. (В терминологии фирмы Intel исполнительный адрес называется эффективным адресом – Effective Address – EA.)

3.1. КЛАССИФИКАЦИЯ ОСНОВНЫХ РЕЖИМОВ АДРЕСАЦИИ

К основным режимам адресации принято относить следующие:

1. Прямая адресация;
2. Относительная адресация;
3. Косвенная адресация;
4. Непосредственная адресация;
5. Неявная адресация.

Прямая адресация и ее виды. Достоинства и недостатки прямой адресации

При использовании *прямой адресации* в адресной части команды задается сам адрес операнда. Различают 2 вида прямой адресации:

- прямая регистровая адресация;
- прямая адресация памяти.

Последнюю иногда называют абсолютной адресацией.

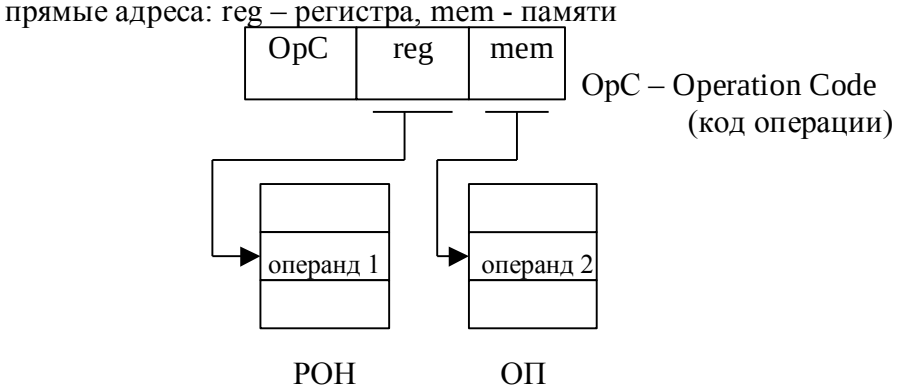


Рис.3.1. Прямая адресация

При иллюстрации режимов адресации предполагается, что исполнительный адрес является в то же время и физическим.

Основным достоинством прямой адресации является отсутствие каких-либо операций, связанных с формированием адреса.

Основным недостатком прямой адресации памяти является большая разрядность поля mem, которая рассчитана на адресацию всего адресного пространства памяти. В свою очередь, это приводит к увеличению длины машинной команды.

Относительная адресация и ее частные случаи

При использовании *относительной адресации* в адресной части команды задается, во-первых, адрес регистра, содержимое которого является главной частью адреса и, во-вторых, смещение, являющееся добавкой к главной части адреса. Как правило, разрядность смещения меньше полной разрядности адреса, что позволяет уменьшить длину адресной части команды по сравнению с использованием прямой адресации.

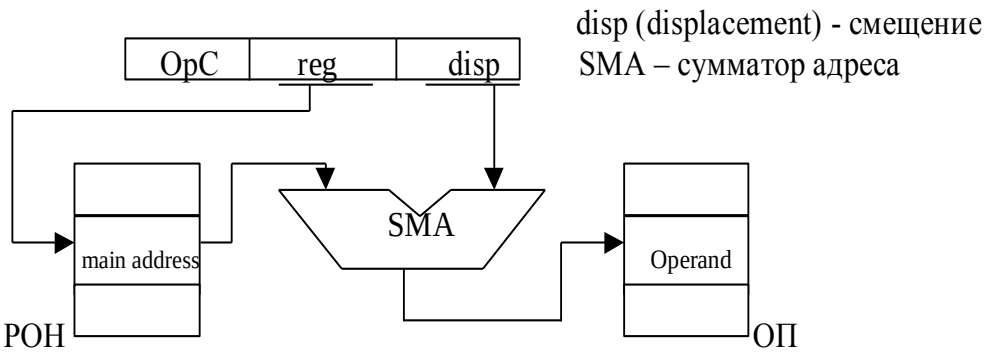


Рис.3.2. Относительная адресация

В зависимости от вида функциональной специализации используемого регистра и соответствующей трактовки его содержимого различают следующие виды относительной адресации:

1. *Базовая адресация* (адресуемый в команде регистр трактуется как базовый);
2. *Индексная адресация* (регистр трактуется как индексный);
3. *Адресация относительно текущего значения счетчика команд.*

В последнем случае адрес регистра в команде не задается, т.к. счетчик команд адресуется неявно.

Последний вид относительной адресации, как правило, используется для адресации команд, хотя в некоторых ЭВМ он может использоваться и для адресации операндов, например, в ЭВМ с архитектурой DEC (Digital Eqipment Corporation).

Базово-индексная адресация и ее развитие в виде базово-индексной адресации с масштабированием

Более сложным видом относительной адресации является *базово-индексная адресация* (Рис.3.3). При ее использовании в адресной части команды выделяются три поля: база (base), индекс (index) и смещение (displacement), а исполнительный адрес формируется как сумма из трех компонент.

Поля base и index команды содержат адреса РОН, в которых в свою очередь находятся компоненты исполнительного адреса – базовый адрес (Base) и индекс (Index).

Частным случаем базово-индексной адресации является отсутствие смещения и формирование адреса из 2 компонент (база и индекс). Для уточнения этого случая соответствующий режим адресации принято называть *базово-индексной адресацией без смещения*.

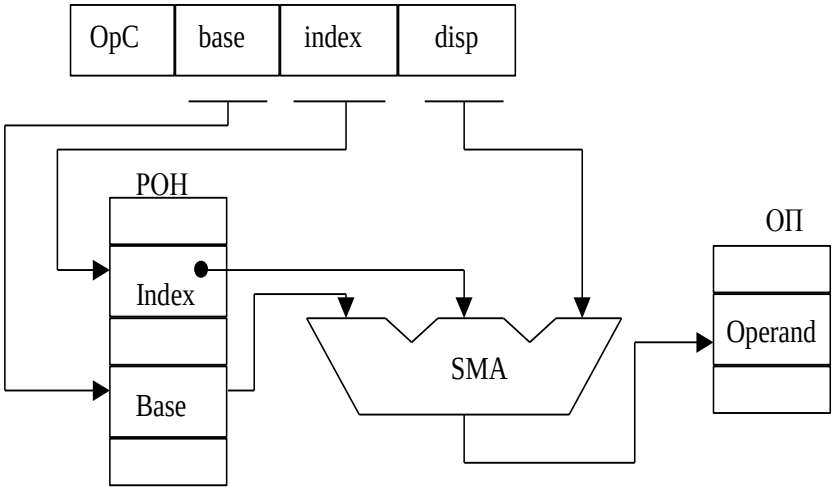


Рис.3.3. Базово-индексная адресация со смещением

Дальнейшим развитием базово-индексной адресации является *базово-индексная адресация с масштабированием,* которая используется в старших моделях семейства 80x86 (начиная с 80386). При использовании масштабирования индекс как одна из компонент адреса предварительно умножается на заданный в машинной команде масштаб. Типичными значениями масштаба являются 2, 4, 8.

Косвенная адресация

При использовании *косвенной адресации* в адресной части команды задается не адрес операнда, а адрес адреса операнда. Принято различать 2 вида косвенной адресации:

а) *косвенная регистровая* (в команде задается адрес регистра, содержащего адрес операнда, находящегося в памяти).

б) *косвенная адресация с использованием памяти* (в команде задается адрес ячейки памяти, в которой содержится адрес операнда). В принципе адрес ячейки памяти может быть как прямым, так и относительным.

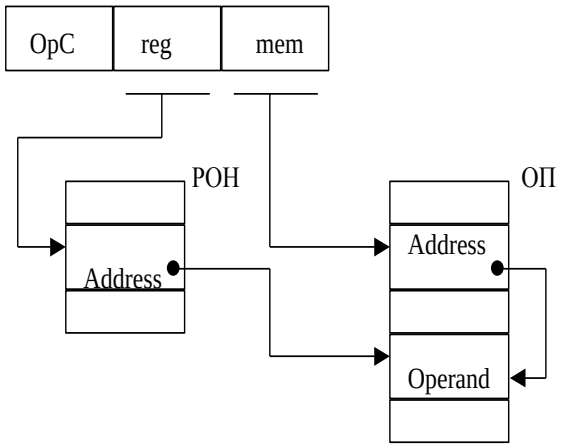


Рис.3.4. Косвенная адресация

Дальнейшим развитием косвенной регистровой адресации являются так называемые автоинкрементная и автодекрементная адресации. При автоинкрементной адресации

содержимое адресуемого в команде регистра сначала используется как адрес операнда, а затем наращивается (инкрементируется) на длину операнда. При автодекрементной адресации содержимое адресуемого регистра сначала уменьшается (декрементируется) на длину операнда, а затем используется как адрес этого операнда. Эти режимы адресации находят широкое применение в ЭВМ с архитектурой DEC (M6800).

Автоинкрементную и автодекрементную адресации целесообразно использовать при работе со стеком.

Непосредственная адресация

При использовании *непосредственной адресации* в адресной части команды задается не адрес операнда, а сам операнд. Непосредственный операнд может являться только программной константой, но не переменной, т.к. его значение должно формироваться на этапе компиляции. Использование непосредственной адресации по сравнению с прямой для программных констант имеет следующие преимущества:

1) Экономия памяти, т.к. операнд представляется в команде и не требует дополнительной ячейки памяти как при прямой адресации.

2) Экономия времени. При выборке операнда не тратится время на обращение к памяти, как при прямой адресации, т.к. операнд выбирается вместе с командой на этапе выборки команды.

Неявная адресация

При использовании *неявной адресации* либо адрес операнда, либо сам операнд не задаются, а лишь подразумеваются. Использование неявной адресации позволяет в значительной степени сократить длину машинного (объектного) кода программы за счет отсутствия в нем адресов неявных операндов, а также самих неявных операндов.

Классическими примерами неявной адресации могут служить:

1) Аккумуляторные команды, в которых адрес аккумулятора, где находится сначала операнд, а затем и результата операции, не задается.

В базовой модели Intel 8086 байтным аккумулятором является регистр AL, двухбайтным – AX, четырехбайтным – комбинация регистров AX (младшие разряды) и DX (старшие разряды).

2) Стековые команды, в которых адрес указателя стека (SP) задается неявно.

Примерами использования неявного операнда могут служить:

1) те же стековые команды, в которых константа изменения указателя стека, равная 2, не задается, а подразумевается.

2) команды инкремента и декремента (INC и DEC), в которых константа изменения операнда, равная 1, подразумевается, а не задается.

3.2. РЕЖИМЫ АДРЕСАЦИИ ПРОЦЕССОРА INTEL 8086 И СПОСОБЫ ИХ ЗАДАНИЯ

В основном для задания режимов адресации используется специальный байт, который принято называть *постбайтом адресации* (он обязательно следует за байтом кода операции) или байтом mod, r/m (по наименованию основных полей, которые являются неизменными в этом байте).

Режимы адресации, реализуемые на основе постбайта адресации, принято называть постбайтными режимами.

Для двухадресной команды постбайт адресации имеет следующую структуру:

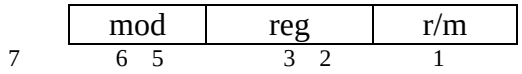


Рис.3.5

Байт с подобной структурой осуществляет адресацию двух операндов, один из которых, задаваемый полем reg, является регистровым. Из этого следует, что машинные команды, использующие постбайт адресации, реализуют операции следующих типов:

- reg – reg (регистр – регистр);
- reg – mem (регистр – память);
- mem – reg (память – регистр);

и не реализуют операции типа mem- mem (память – память).

Операции типа "память – память" реализуются в командах обработки строк (цепочек). Например, команда MOVS (пересылка строки) осуществляет пересылку строки-источника в строку-приемник. Естественно, обе строки находятся в памяти.

Поле reg постбайта задает прямой регистровый адрес операнда, находящегося в РОН.

Поле r/m (register/memory) задает адрес второго операнда команды, находящегося либо в регистре, либо в памяти. Факт принадлежности поля r/m к адресации регистра или памяти определяется значением поля mod (режим). При mod = $(11)_2$ поле r/m трактуется как адрес регистра, при mod ≠ $(11)_2$ поле r/m участвует в адресации памяти. Фактическое значение двоичного кода поля r/m в этом случае неявным образом определяет базовую и (или) индексную компоненту ЕА. При адресации памяти значение поля mod задает длину смещения disp в байтах:

mod	disp
00	-
01	1 байт
10	2 байт

Смещение интерпретируется как целое число со знаком, естественно, представляемое в дополнительном коде. В соответствии с этим при сложении байтного смещения с 16-разрядными компонентами в виде базы и (или) индекса на этапе формирования ЕА производится предварительное расширение смещения путем копирования знакового бита во все биты старшего байта. Таким образом, старший байт содержит все нули в случае положительного смещения и все единицы в случае отрицательного смещения.

При использовании двухадресной команды адрес одного из операндов используется и как адрес операнда, и как адрес результата. Этот операнд называется приемником (dst - destination).

Использование одного из операндов в качестве приемника определяется значением специального бита *d* (direction) кода операции. При *d* = 1 приемником является операнд, адресуемый с помощью поля reg, при *d* = 0 – операнд, адресуемый с помощью поля r/m.

Бит d является предпоследним слева (вторым справа) битом байта кода операции OpC. В свою очередь, крайний правый бит кода операции *w* (word) определяет длину операндов. При *w* = 0 длина операндов – байт, при *w* = 1 – 2 байта (слово).

Интерпретация поля r/m для регистровых операндов и операндов, размещаемых в памяти, может быть представлена следующей таблицей:

Таблица 3.1

код r/m	mod=11		mod≠11	
	w=0	w=1	Base	Index

000	AL	AX	BX	SI
001	CL	CX	BX	DI
010	DL	DX	BP	SI
011	BL	BX	BP	DI
100	AH	SP	-	SI
101	CH	BP	-	DI
110	DH	SI	BP	-
111	BH	DI	BX	-

Как следует из таблицы 3.1, для базовой компоненты EA могут быть использованы только регистры BX и BP, а для индексной компоненты - только регистры SI и DI. При использовании 32-разрядной адресации, начиная с процессора 80386, эти ограничения снимаются, т.е. в качестве базового и индексного регистра может использоваться любой РОН, за исключением SP.

Исключением из общего правила является комбинация:

mod = 00, r/m = 110.

По общим правилам, в соответствии с таблицей, приведенной комбинации должна соответствовать косвенная регистровая адресация с использованием BP. На самом деле эта комбинация используется как исключительный случай для кодирования прямой адресации памяти; при этом за постбайтом адресации в команде следуют 2 байта смещения, которые в этом случае интерпретируются как EA.

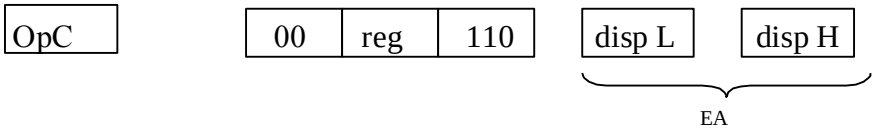


Рис.3.6

С помощью постбайта адресации реализуются следующие режимы адресации:

1) *Регистровая прямая.*

Реализуется всегда для операнда, адресуемого полем reg, а также для операнда, адресуемого полем r/m при mod = 11.

2) *Прямая адресация памяти.*

Реализуется как исключительный случай при mod = 00, r/m = 110.

3) *Косвенная регистровая адресация.*

Имеет место при mod = 00 и r/m = 100 (SI), r/m = 101 (косвенный адрес в DI), r/m = 111 (BX).

4) *Базовая адресация.*

(EA = BASE+disp): mod = 01 или mod = 10, r/m = 110 или 111.

5) *Индексная адресация.*

(EA = Index+disp):

mod = 01 или 10, r/m = 100 или 101

6) *Базово-индексная адресация без смещения.*

(EA = Base+Index):

mod = 00, r/m = 000, 001, 010, 011

7) *Базово-индексная адресация со смещением.*

(EA=Base+Index+disp):

mod = 01 или 10, r/m = 000, 001, 010, 011.

Непосредственная и неявная адресации задаются не с помощью постбайта, а с помощью кода операции, в связи с чем эти режимы не относятся к постбайтным.

Косвенная адресация с использованием памяти реализуется в командах перехода JMP (jump) и в командах вызова CALL, но этот режим используется не для адресации операнда, а для задания адреса команды (перехода или вызова).

4. ОСНОВНЫЕ ФОРМАТЫ КОМАНД

Система команд процессора Intel 8086 насчитывает более 10 разнообразных форматов команд, отличающихся как длиной формата (машинная команда может занимать от 1 до 6 байт, не считая возможных предшествующих ей префиксов), так и распределением полей в отдельных байтах команды.

Используются 3 вида префиксов (префиксных байтов), которые предшествуют команде и определенным образом влияют на ее выполнение.

К префиксам относятся:

1) seg – префикс замены сегмента;

2) rep – префикс повторения;

3) lock – префикс блокировки шины.

1) Префикс замены сегмента используется для переназначения стандартных сегментов, используемых по умолчанию при обращении к памяти за операндом и (или) при записи результата.

Адрес переназначения сегмента занимает 2 средних бита в префиксном байте (адресацию сегментных регистров см. выше: регистровая структура (программная модель) процессора).

2) Префикс повторения используется исключительно перед командами обработки строк и заставляет повторять ее выполнение многократно в целях поэлементной обработки всей строки.

Использование префикса rep позволяет организовывать цикл по последовательной обработке элементов строки на аппаратном, а не на программном уровне.

3) При выполнении команды с предшествующим ей префиксом lock на все время выполнения команды блокируется шина, связывающая процессор с памятью и портами ввода-вывода.

Действие любого префикса распространяется только на одну машинную команду, которая следует непосредственно за ним.

Форматы команд

1. Однобайтная безадресная команда:

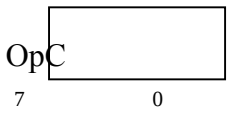


Рис.4.1. Однобайтная одноадресная команда

Подобный формат используется либо командами с неявной адресацией, либо командами, не использующими операндов.

Примерами команд первого типа могут служить команды обработки строк, в которых строка–источник и строка-приемник неявно адресуются с использованием регистров SI и DI соответственно.

К ним относятся:

MOVS – пересылка строки,

LODS – загрузка строки,

STOS – запись в память строки,

CMPS – сравнение строк,

SCAS – сканирование строки.

Примером команды 2 типа может служить команда RET (return) возврата из процедуры (подпрограммы), которая в зависимости от вида возврата: типа NEAR (ближний или внутрисегментный) или FAR (дальний, межсегментный) восстанавливает из стека либо только содержимое регистра IP (NEAR), либо содержимое IP и CS (FAR).

2. Однобайтная одноадресная команда

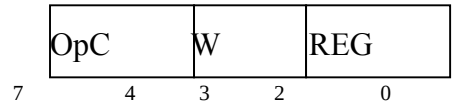


Рис.4.2 . Однобайтная одноадресная команда

Эта команда задает прямой адрес регистрового операнда (поле reg). Бит w задает длину операнда (1 – слово, 0 – байт).

Примерами использования подобного формата могут служить команды INC (+1) (инкремент) или DEC (-1) (декремент), а также команда XCHG (exchange) – команда с аккумуляторным операндом.

3. Двухадресная команда с постбайтом адресации

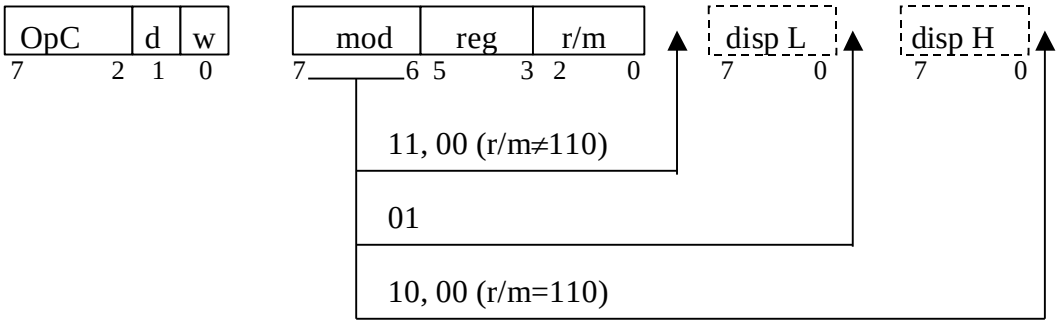


Рис.4.3. Двухадресная команда с постбайтом адресации

Бит d кода операции (direction - направление) определяет, по какому адресу записывается результат операции (при $d = 1$ – в регистр reg, при $d = 0$ – в регистр или память, адресуемые полем r/m).

Подобный формат широко используется для разнообразных арифметических и логических команд.

4. Одноадресная команда с постбайтом адресации

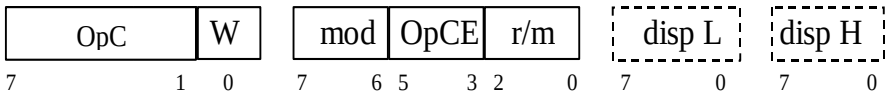


Рис.4.4. Одноадресная команда с постбайтом адресации

В отличии от предыдущего формата, среднее поле постбайта адресации является расширением кода операции (E – Extended).

Подобный формат используется, во-первых, для одноопернадных команд (например, INC, DEC, NEG – negative – изменение знака, NOT – инвертирование) и, во-вторых, для

двухоперандных команд, в которых один из операндов адресуется неявно (например, MUL/IMUL – умножение, DIV/IDIV – деление, в которых один из операндов является аккумуляторным, а также команды сдвигов, в которых счетчик числа сдвигов адресуется неявно регистром CL).

Замечание: в командах сдвигов расширением кода операции определяет вид сдвига (арифметический, логический или циклический, вправо, влево). Так как среднее поле 3-х битовое – 8 видов сдвигов.

SAR – арифметический сдвиг вправо (shift arifmetic right);
SAL - арифметический сдвиг влево (shift arifmetic left);
SHR – логический сдвиг вправо;
SHL – логический сдвиг влево;
ROR – циклический сдвиг вправо (rotation right);
ROL – циклический сдвиг влево (rotation left);
RCR - циклический сдвиг вправо с включением в кольцо флага CF;
RCL – циклический сдвиг влево с включением в кольцо флага CF.

5. Двухоперандная команда с постбайтом адресации с непосредственным операндом.

Команда этого формата может занимать в памяти от 3 до 6 байт в зависимости как от длины смещения, так и от длины непосредственного операнда.

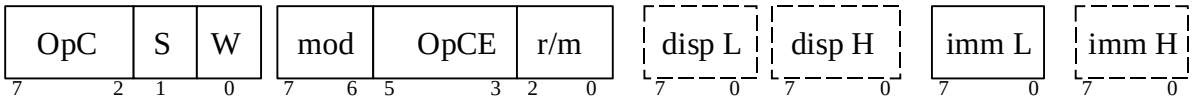


Рис.4.5. Двухоперандная команда с постбайтом адресации
с непосредственным операндом (imm (immediatly) – непосредственный операнд)

Бит *S* – специальный бит кода операции (*s* – sign extended – знаковое расширение) является актуальным только при w = 1 (длина операнда – слово). При *S* = 1 непосредственный операнд занимает в памяти 1 байт (imm L) и требует знакового расширения до слова (операнды должны иметь одинаковую длину). При *S* = 0 длина операнда – 2 байта.

Примерами использования подобного формата могут служить арифметические (сложение и вычитание) и логические команды.

5. ПРИНЦИПЫ РАЗМЕЩЕНИЯ В ОП ЕДИНИЦ ИНФОРМАЦИИ ФИКСИРОВАННОЙ ДЛИНЫ

Эти принципы охватывают 2 аспекта:

1. Как размещаются байты внутри слова, двойного слова, учетверенного слова.
2. Как размещаются сами единицы в адресном пространстве ОП.

В отношении **первого аспекта** используются 2 принципа распределения байт внутри слова, двойного слова, учетверенного слова.

Первый принцип:

Байт с большей значимостью располагается по меньшему адресу. Этот принцип в основном используется в больших ЭВМ.

Второй принцип:

Байт с меньшей значимостью размещается по меньшему адресу. Этот принцип используется в миникомпьютерах и, в частности, в PC.

Предположим, что в двухбайтном регистре размещено число (-10), естественно, представленное в дополнительном коде.

$(10)_{10} = (1010)_2$

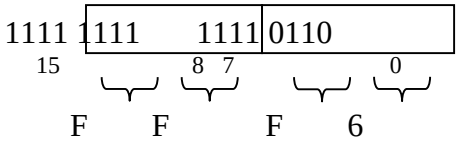


Рис. 5.1

Передача содержимого регистра в память по адресам байт A и (A+1) в зависимости от принципа размещения байт представлена на рисунках.

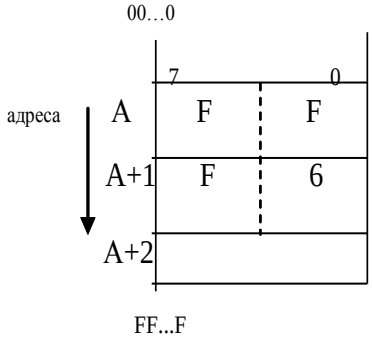


Рис.5.2. Байт с большей значимостью по меньшему адресу

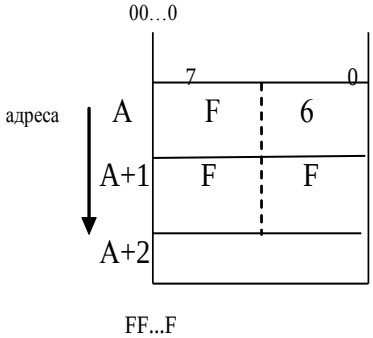


Рис.5.3. Байт с меньшей значимостью по меньшему адресу

Замечание.

Принципы размещения байт в словах, двойных словах, учетверенных словах необходимо учитывать в программах на ASSEMBLER в тех случаях, если реализуется побайтная выборка единиц информации.

Второй аспект связан с тем, какие адреса для размещения единиц информации фиксированной длины являются допустимыми, а какие нет. Принято считать, что адресом структурной единицы информации, содержащей несколько байт, является адрес ее левого (старшего) байта при использовании первого принципа размещения, или правого (младшего) байта при использовании второго принципа. Это означает, что в любом случае

размещения единицы информации в памяти по адресам A, A+1, … адресом этой единицы считается A.

В некоторых моделях компьютеров налагаются ограничения на размещение единиц информации фиксированной длины, составляющей 2^k байт ($k>0$). Это ограничение принято называть *целочисленной границей*. Правила целочисленной границы формулируются в виде:

Адрес единицы информации длиной 2^k байт должен быть кратен 2^k. Это означает, что адрес слова (2 байта) должен быть кратен 2, т.е. быть четным. Адрес двойного слова должен быть кратен 4. Адрес учетверенного слова кратен 8.

Формально проверка соблюдения целочисленной границы схемно реализуется сравнением соответствующего числа младших разрядов адреса с нулем. Несоблюдение целочисленной границы приводит к прерыванию соответствующего типа.

Требование соблюдения целочисленной границы связано с уменьшением числа обращений к ОП при выборке операндов фиксированной длины, например, при ширине выборки в 4 байта (шина данных 32 разряда) интерфейс памяти позволяет при одном обращении выбрать из памяти или записать в память 4 байта с адресами $4m$, $4m+1$, $4m+2$, $4m+3$ ($m \in N$), т.е. меньший адрес, по которому выбирается двойное слово, кратен 4. Таким образом, если двойное слово размещается в ОП по целочисленной границе, то обращение к нему по чтению или записи реализуется за 1 цикл памяти. В противном случае – за 2 цикла.

Понятие целочисленной границы широко используется в отношении как команд, так и данных в больших компьютерах. Начиная с модели i486, проверка целочисленной границы используется в отношении данных, но не команд, и имеет место только в том случае, когда установлен флаг AC (Alignment Check – контроль выравнивания) в регистре EFLAGS и, кроме того, установлен бит AM (Alignment Mask – маска выравнивания) в управляющем регистре CR0 – Control Register.

Управляющие регистры относятся к так называемым системным регистрам, что означает возможность их использования только системными, но не прикладными программами.

6. ПРИНЦИПЫ ФОРМИРОВАНИЯ ФИЗИЧЕСКОГО АДРЕСА. СТАНДАРТНОЕ НАЗНАЧЕНИЕ СЕГМЕНТОВ

В процессоре Intel 8086 поддерживается простейшая модель сегментированной памяти. Использование сегментации позволяет, во-первых, сделать машинные программы инвариантными к месту их конкретной привязки (загрузки) в физической памяти, и, во-вторых, расширить объем физического адресного пространства до 1Мбайта (2^{20} байт) по сравнению с возможностями 16-битной адресации, которая обеспечивает адресное пространство всего 2^{16} байта = 64 Кбайта.

Физический адрес формируется как сумма двух компонент: базового адреса сегмента, который выбирается из соответствующего сегментного регистра, и внутрисегментного смещения (offset), которое определяет относительный адрес внутри сегмента. Обе компоненты физического адреса являются 16-разрядными, однако они складываются со смещением друг относительно друга, в результате чего формируется 20-разрядная сумма, представляющая собой физический адрес.

Схему формирования физического адреса можно представить в следующем упрощенном виде:

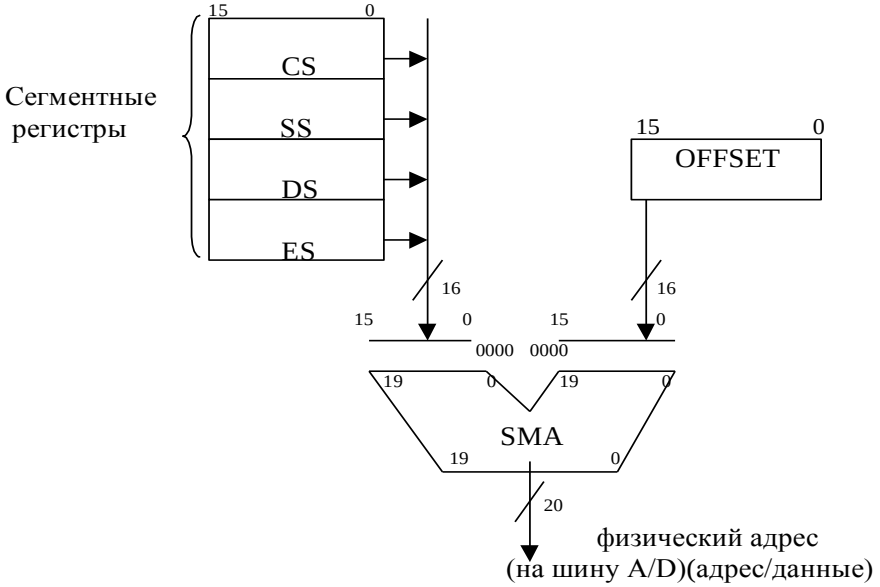


Рис.6.1. Схема формирования физического адреса

Так как при сложении к базовому адресу сегмента приписываются 4 младших нуля (справа), то в физической памяти автоматически происходит выравнивание сегментов на 16-байтную границу, которую принято называть *границей параграфа*.

Тип используемого внутрисегментного смещения (offset) определяется видом обращения к памяти. Собственно говоря, вид обращения к памяти определяет также и используемый сегмент и, соответственно, сегментный регистр, участвующий в формировании физического адреса.

Сегментные регистры и внутрисегментные смещения, используемые в соответствии с их стандартным назначением (по умолчанию) в зависимости от вида обращения к памяти, представлены в таблице:

Таблица 6.1.

N	Вид обращения к памяти	Стандартное назначение		Возможность переопределения
		seg	offset	
1	Выборка команды	CS	IP	-
2	Выборка операнда и/или запись результата при использовании регистра BX в качестве базового при формировании EA	DS	EA	CS, SS, ES
3	То же, что и (2), но при использовании регистра BP	SS	EA	CS, DS, ES
4	Стековая операция (обращение к стеку)	SS	SP	-
5	Обращение к строке (цепочке)-источнику	DS	SI	CS, SS, ES
6	Обращение к строке (цепочке)-приемнику	ES	DI	-

Для переопределения сегмента, отличного от стандартного назначения, используется префикс замены сегмента. Он предшествует команде и имеет стандартную структуру следующего вида:

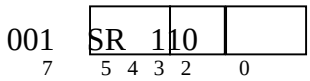
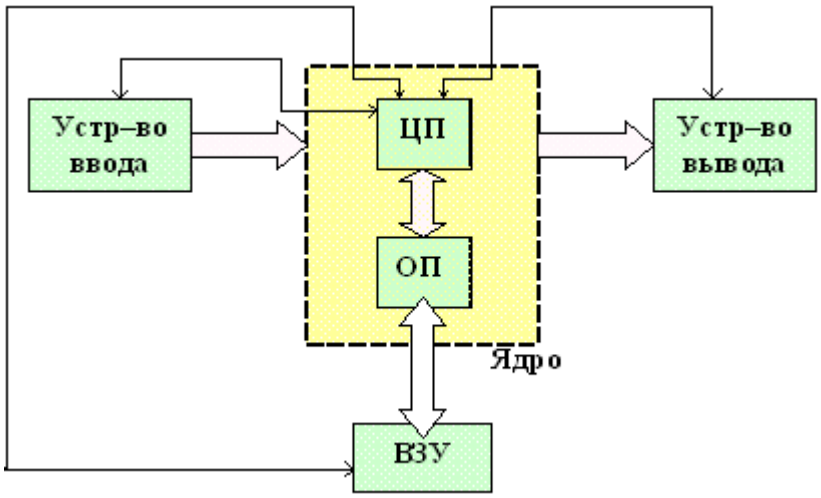


Рис.6.2. Структура префикса замены сегмента

Поле SR определяет адрес используемого сегментного регистра.

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.

Периферийные (внешние) устройства компьютера.

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;
- файловая организация данных и последовательный доступ к данным в файлах ;
- низкая скорость обмена как следствие использования механики и последовательного доступа;
- неограниченный объем по сравнению с ОП;

- минимальная стоимость хранения данных по сравнению с другими типами памяти.

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.

Под вводом данных обычно понимается их передача из ПУ в основную память. Под выводом данных – передача данных из ОП в ПУ.

Периферийные устройства ввода/вывода.

Устройства могут выполнять только функции ввода, специальные устройства вывода и могут совмещать эти функции. Стандартная серийная периферия ЭВМ и разнообразные специальные устройства в прикладных системах.

Устройства ввода – клавиатура, устройство считывания с перфоленты и перфокарт, мышь, микрофон, джойстик, сканер. Источниками данных могут быть различные бытовые приборы и устройства – фотокамеры, кинокамеры, видеокамеры

Устройства вывода – принтер, дисплей, перфоратор перфоленты или перфокарт, графопостроитель.

Устройства, совмещающие эти функции – телетайп,

Большое многообразие специальных устройств ввода/вывода – разнообразные датчики в контрольно-измерительных приборах, системах промышленной автоматики – нагреватели, вентиляторы.

Прямое управление элементами и блоками устройств – двигатели шаговые , постоянного тока, переменного тока, реле, пускатели

Непосредственный контроль – состояния контактов, переключателей, датчики положения, преобразователи угол-код,

Основные способы организации ввода/вывода.

1. Программно-управляемый ввод-вывод (В/В) (PIO).
2. Каналы передачи данных
3. Мультишина в режиме DMA (прямой доступ к памяти).
4. Процессоры В/В

Ввод/вывод с прямым программным управлением (PIO).

Для первых Неймановских ЭВМ использовалась структура с непосредственными связями и прямое программное управление В/В. Как наиболее простой и быстрый сохраняется во встроенных применениях – микроконтроллерах.

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

6) Пересылка данных между ОП и ВУ включает две стадии – чтение данных из ОП в регистры CPU , запись или чтение данных либо из адресуемых внешних регистров ВУ или непосредственно с датчиков в составе ВУ.

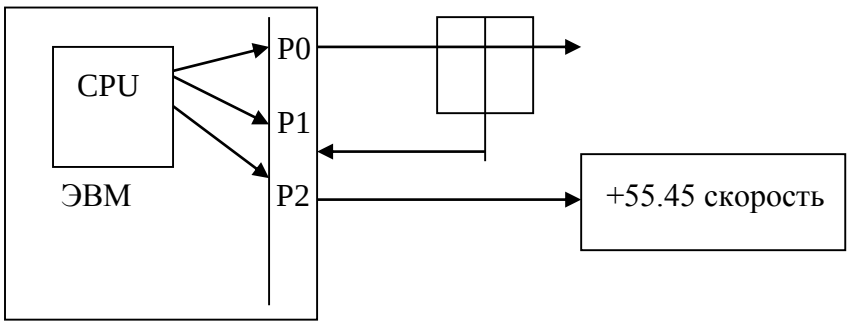
7) Преимущественно асинхронный принцип обмена, связанный с использованием механики, электромеханических элементов и существенными не предсказуемыми задержками в переходных процессах управления.

При запуске принтера ожидаем его готовности - разгон двигателя, контроль наличия бумаги в устройстве подачи и др. По готовности передаются команды управления обменом и контролируются результат их выполнения.

8) Использование квитирования и/или прерываний в асинхронном обмене

Значительные задержки ожидания случайных по продолжительности и времени возникновения событий завершения переходных процессов в ПУ устраняются использованием **прерываний** по их заверешении Процессоры при этом могут параллельно выполнять обработку данных.

Схема включения простых средств управления клавиатурой по прерываниям и ЖКИ-индикатором с микроэвм (контроллером) MCS51 и организация локального пульта/



ЭВМ сканирует строки матрицы клавиш и при нажатии на клавишу принимает запрос прерывания, идентифицируя нажатую клавишу и принимая решение по ее значению.
Очевидно, что событие нажатия клавиши случайное и не предсказуемое по длительности.
Механическое нажатие клавиши сопровождается дребезгом, который устраняется программным ожиданием.
Стандартный модуль ЖКИ-дисплея включает контроллер управления разверткой изображения на дисплее, использует коды разверток во внутреннем ПЗУ и память для хранения ASCii –кодов идндицируемых символов. Программное управление ЖКИ включает передачу и выполнение команд адресации по строкам и столбцам, специальные режимы инициализации, сброса и их завершение контролируется либо асинхронно - по сигналам завершения в слове состояния, либо временными задержками.
В современных ЭВМ функции управления клавиатурой отделены от CPU и выполняются специальным устройством со встроенным локальным управлением микроконтроллером, где все вопросы решаются программно и независимо от работы центральной машины.
Это устройство формирует прерывания, когда после нажатия символ уже распознан и его код передается в программу редактирования или командного управления.
Однако и с такой клавиатурой ожидание сохраняется и связано уже с принципиально медленной работой оператора. Высокопроизводительные ЭВМ, выполняя одну эту работу, простаивают и их стараются загрузить параллельно другой работой, а средства ручного ввода и индикации выносят в автономные терминалы под управлением ОС.
Если при этом работают несколько терминалов и некоторые задачи обращаются к внешним файлам, то параллельно-последовательное их прямое обслуживание не возможно и может привести к потерям данных и событий.

Основное назначение В/В – обмен данными с ОП и эти операции могут быть выполнены независимыми друг от друга .

Таким образом, фирма IBM впервые обратила внимание на эти проблемы и нашла, по тому времени, эффективное ее решение.

Управление вводом-выводом с использованием **каналов и мультипрограммная загрузка** ЭВМ обеспечивают до 90% загрузки Процессора и Памяти как наиболее дорогостоящего оборудования ЭВМ.

Каналы В/В IBM360/370

В первую очередь, рутинные операции прямого управления электро-механическими устройствами (включение, выключение, управление двигателями, контроль состояния старт-стопных устройств и т.д.) выделены в **контроллеры Увв**. **Унифицированы интерфейсы и система кодирования управляющих команд контроллеров**. Таким образом, получена единая схема (интерфейс) включения различных УВВ. После этого выделена **система команд автономного управления и аппаратура** для реализации программного управления Увв в виде **оборудования каналов**.

Фирма IBM в системах IBM360/370 использовала для этой цели систему каналов и мультиплексор коммутации потоков данных.

В IBM360 эта проблема решалась на аппаратном уровне – совместным использованием общего оборудования CPU:

1)выделяется блок мультиплексора данных, в котором коммутируются потоки данных **CPU←→ Mem, Mem←→ I/O**.

2) Если в схеме Неймана предполагается прямое управление I/O, что вносит существенную задержку и даже остановку (недогрузку) Процессора, то в архитектуре IBM для управления Увв организуются процессоры В/В в виде **(Мультиплексных**) и **(Селекторных**) Каналов.

Каналы являются программно-независимыми блоками управления с прямым доступом к контроллерам устройств ввода-вывода. Каналы работают по своей **канальной программе**, состоящей из специальных канальных команд, хранящихся в Основной памяти. Эту программу инициирует CPU, выполняя команду ввода-вывода, которая устанавливает связь с Увв через Канал.

Канал проверяет состояние готовности к вводу-выводу , запускает электромеханические элементы Увв (Принтер, Перфоратор, Лента) и, в случае их готовности, обращается в Основную память к собственной программе. Обмен осуществляется **информационными блоками байтов в** режиме **приостановки** Процессора на время быстрого обмена байтами между Памятью и Каналом. Канал параллельно и независимо от Процессора считывает или записывает байт в Увв.

Выполнение канальной программы завершается **прерыванием** от Канала, при этом завершается ввод-вывод или включается новая программа Канала.

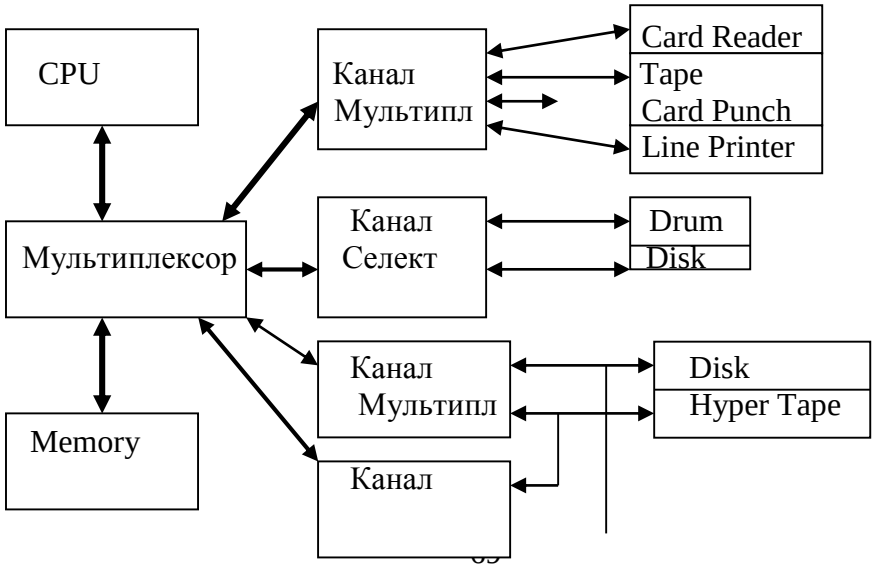


Рис.4. Структура IBM7094

Приостановка Процессора или занятие цикла (такта) – выполнение одной микрокоманды записи или чтения байта из ОП. Практически для выполнения программы эта задержка не существенна в случае медленных Увв(Принтер, Лента, Считыватель с перфокарт, Дисплей с клавиатурой,..)

Мультиплексный канал может обслуживать несколько Увв последовательно. 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.

В случае быстрых Увв (Диск, Барабан) работает только одно Увв через **Селекторный Канал**. Быстрые схемы Селекторного Канала размещаются в CPU и с учетом разделенного использования БМУ монополизируют и останавливают работу CPU.

Работа канала

а) 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);

Участие ЦП 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).

В дальнейшем такое распределение оборудования и **совмещенное** использование **CU** Процессором и несколькими Каналами становится в СБИС не возможным - управление вводом/выводом локализовано в контроллерах, память и контроллеры функционально и конструктивно разделены в современных процессорах и управление каналами заменяет режим **Прямого Доступа .**

В современных архитектурах класса **Main Frame** выделенные процессоры В/В **объединяются с CPU в мультипроцессорную систему и работают с общей или локальной памятью, где хранятся их программы и общие данные.** Процессоры управления вводом/выводом.для высокопроизводительных систем позволяют разделить и распараллелить быструю обработку данных и медленные операции ввода.

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

В начале 70-х годов фирма DEC разработала **Мини-ЭВМ** PDP8-11, принципиально отличающуюся от компьютеров IBM . Архитектура ЭВМ имеет модульную организацию на основе стандартного системного интерфейса **Общая шина:**

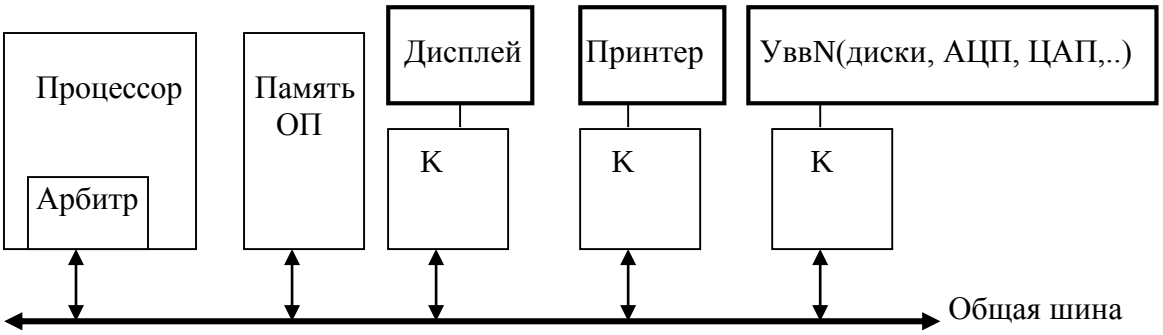


Рис.5. Структура PDP 11.

Процессор включает 16-битное АЛУ, что определяет **основной формат (разрядность)** данных. Стандартная система команд включает арифметические и логические операции с 16-битовыми данными с фиксированной точкой.

Общая шина – стандартный системный интерфейс , обеспечивающий единообразное электрическое и конструктивное подключение блоков ЭВМ в виде модулей. Корзина модулей с Общей шиной может быть сконфигурирована пользователем. Шина содержала 16- битную шину данных , 20-бит шину адреса и сигнальные линии, используемые **арбитром** шины для подключения асинхронных запросов к ОП.

Выделенную линию Захвата Шины с открытым коллектором контролирует Арбитр. Контроллер Увв, захвативший шину, закорачивает ее на себя и освобождает по окончании обмена- устанавливается высокий уровень. Арбитр принимает Запрос шины по линиям Запроса и в соответствии с приоритетами разрешает доступ к шине конкретному Внешнему устройству.

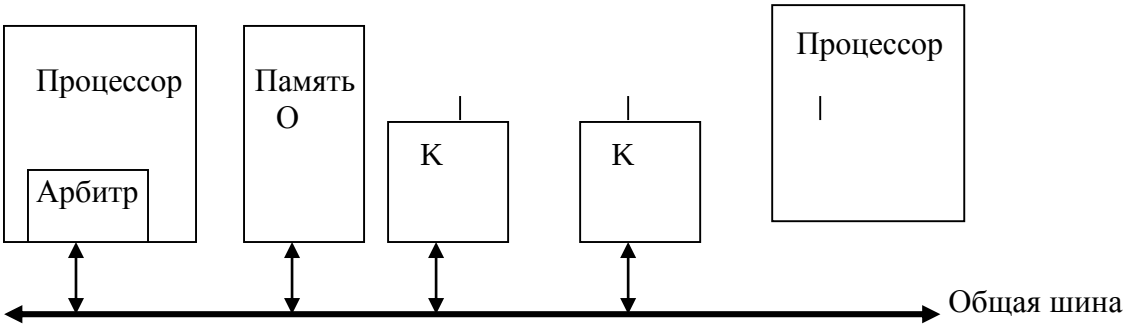
Общая шина допускает обмен данными CPU←→ОП и Увв←→ОП в режиме **прямого доступа к памяти(ПДП)** с приостановкой процессора. При этом CPU не участвует в обменах Увв с ОП., обеспечивается высокая скорость обмена данными при минимальных затратах оборудования. Завершение многобайтовых операций ввода-вывода также контролируется по прерыванию и каждый модуль с контроллером ввода-вывода содержит схемы управления ПДП.

ЦП инициализирует DMA и завершает операцию ввода/вывода.

В ОШ могут работать несколько активных устройств (процессоров) и таким образом, реальным фактом стало создание мультипроцессорных или мультимашинных систем.

Потенциальная возможность квазипараллельной работы нескольких процессоров с ОП через общую шину - огрнаниченная по времени занятость шины при многотактном исполнении команд. Только 2-3 такта 12-тактной команды занята шина обменом данными с процессором, остальные такты шина свободна и может быть захвачена другим активным

устройством и процессором в мультипроцессорной системе. Если для каждого процессора выделить локальную память для хранения собственной программы и локальные устройства ввод/вывода, общую память сохранить только для данных, то количество обращений к шине и ее загрузка в мультимшинной системе становится еще меньше и параллельная работа становится более производительной.



Режим DMA используется как для передачи одиночных байтов , так и для блочного обмена. 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.

DMAC содержит некоторое число программно доступных регистров, представленных для ЦП 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):

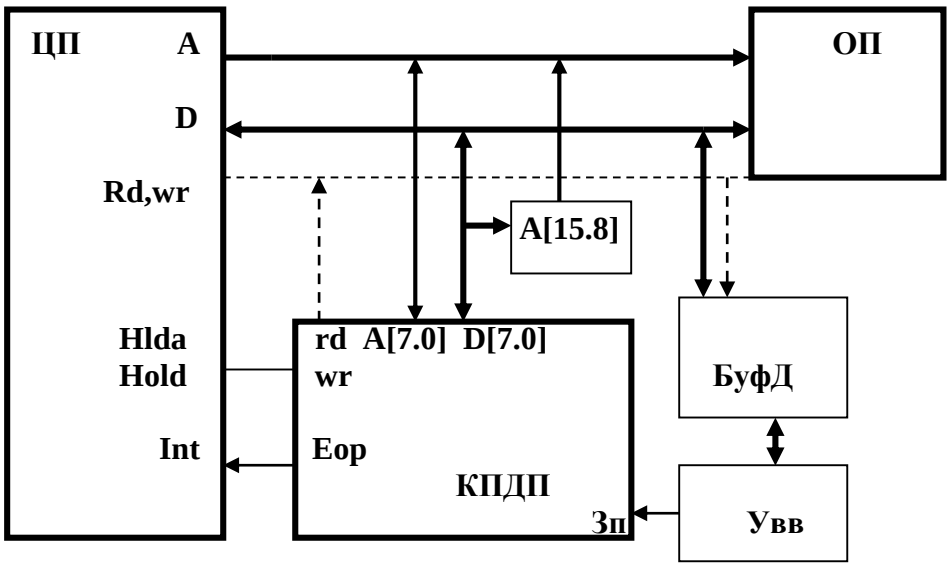
Название режима	Описание режима
Блочная пересылка	**Захват шины на весь период пересылки блока; на весь период DMA ЦП не имеет доступа к шине памяти, в этот период процессор может продолжать работу по программе с обращением к КЭШ – памяти.**

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.**

Архитектура современной ЭВМ с микропроцессором напоминает архитектуру PDP11 – Общую шину заменили другие стандарты – **MultiBus** (Intel), **VME**(Motorolla) и **иерархическая мультишина** ПК. Модули, подключаемые к системной шине, - **микророцессор, ОП , БИС** сопряжения шины с периферийными устройствами – **параллельные и последовательные порты, контроллер ПДП и прерываний.**

В процессорах фирмы Intel предусмотрен входной сигнал HOLD(захват шины) и сигнал подтверждения захвата HLDA(Hold Acknowledgment). 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.
Арбитраж осуществляет специальный внешний модуль контроллера DMA, например, схема **К1810ВТ37** (отечественный аналог) [Подклетнов Г.С.] управляет четырьмя каналами.
Каждый канал содержит регистр базового адреса ОП, регистр текущего адреса ОП, счетчик байтов, регистр режима, регистр команд, слово состояния.
Основные режимы – чтение, запись, проверка. При выполнении операции выбирается режим инкремента или декремента адреса. Передача – одиночная, блочная и каскадный – расширение числа каналов.
Команда управления – разрешение или запрет общего ПДП, приоритет фиксированный или вращение, обмен Память-Память или обычный Память-Увв
Регистр маски каналов – разрешение или запрет i-ого канала.

Схема включения.



КПДП инициализируется и программируется ЦП записью управляющих байтов в регистры по адресам на линиях **A[3.0].** По запросу ПДП из Увв (**сигнал Зп**) формируется сигнал **Hold** и подтверждение от ЦП – **Hlda**. КПДП выставляет адрес памяти за два такта – в первом старшие разряды **A[15.8]** сохраняются в буферном регистре, во втором – младшие разряды адреса A[7.0] , сигналы **rd** или **wr**, читаются или записываются данные,разрешенные буферу **БуфД** для Увв. Завершение обменом одиночных байтов – снимается запрос Hold. При обмене блоками по сбросу счетчика формируется прерывание сигналом **Eop**.

Контроллеры DMA позволяют реализовать следующие передачи байтов:

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

Контроллеры ПДП берут на себя только непосредственно операции обмена данными и контроль обмена блоками данных. Вместе с тем каналы В/В выполняли и другие необходимые для ввода/вывода вспомогательные операции, например, кодирование и декодирование, упаковку байтов в форматы, согласованные с форматами слова ОП , быструю независимую инициализацию. Эти задачи решали программы каналов, а теперь их должны выполнять CPU.

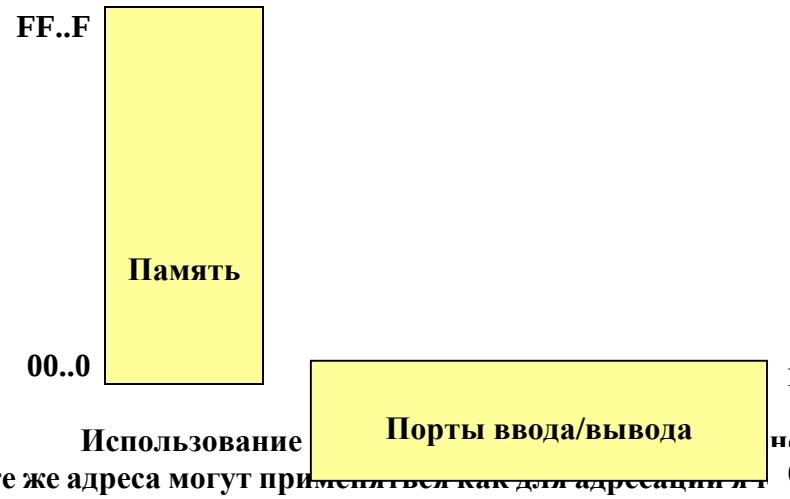
Функции каналов реализуют специализированные процессоры ввода/вывода.

Процессор ввода/вывода К1810ВМ89 (аналог микросхемы фирмы Intel).

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 ...нства предполагает, что одни и те же адреса могут применяться как для адресации ... ак 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.

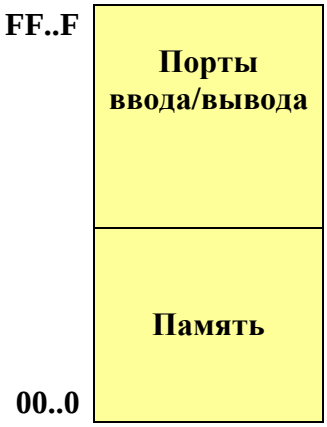
Достоинством подобного подхода принято считать возможность использования более короткого адреса порта ввода/вывода по сравнению с адресом ячейки памяти.

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:

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

Недостатками использование совмещенного адресного пространства являются:

5) **Усложнение управления кэшированием, связанное с необходимостью запрета кэширования адресного пространства, выделенного для портов ввода/вывода.**
6) **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:

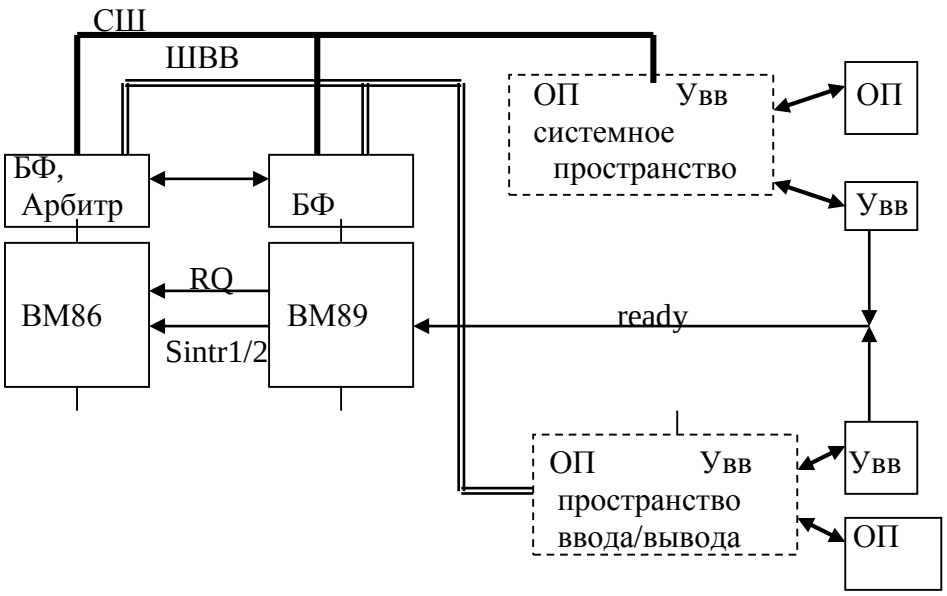
- **первоначальный запрос направляется к памяти по быстрой шине, а затем, если память не может ответить на него, то запрос перенаправляется в шину ввода/вывода;**
- **фильтрация адресов специальной микросхемой, отделяющей адреса памяти от адресов портов ввода/вывода; в частном случае подобную функцию 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.

ПРОЦЕССОР ВВОЛА -ВЫВОДА

Процессор использовался в мультипроцессорной конфигурации с i86. (Подклетнов Г.С.) Процессор может параллельно работать с двумя каналами ПДП со скоростью 0.28-1.5Мбайт. Процессор имеет 16-битовую шину данных и 20-битовую адресную шину, позволяют сопрягать 8- и 16-битовые шины. Используются две конфигурации включения – локальная и удаленная.

Локальная конфигурация рис.



СШ – системная шина, адресное пространство с Увв, отображаемыми на ОП, 1Мбайт
ШВВ – шина ввода/вывода и адрсное пространство 64 Кбайт
RQ – сигнал запроса шины.
Sintr1/2 – сигналы запросов прерываний
Ready – запрос ПДП
Две Multibas-шины СШ и ШВВ с двумя процессорами, каждый из которых может быть Host, Slave – всегда память ОП или Увв.
Буферами (усилителями) шин управляют два контроллера шин ВГ88 и Арбитр Multibas ВБ89.

Каждый канал в локальной конфигурации может выполнять ПДП-обмен по 8/16 битовой шине данных, выполнять программу, отвечать на запросы готовности или приостанавливаться.
20-битовые регистры каналов содержат 2 адреса (источников или приемников), адрес таблицы перекодировки, счетчик байтов, битовую маску для сравнения выделенных разрядов байта с заданным значением

x x x 1 1 1 1 x 1 0 1 0 1 1 0 1 = x x x 0 1 1 0 x замаскированное значение

маска код сравнения

Регистр управления каналом определяет,

- откуда и куда передаются данные (M-M, Y-M, M-Y, Y-Y)

-перекодировка (да, нет) – производится заменой передаваемого байта через выделенную таблицу 256 байт

-способ синхронизации (асинхронная, синхронная от источника или приемника)

-в каком регистре адрес источника(один из двух регистров)

-одиночный обмен или по счетчику

-как заканчивается обмен – по внешнему сигналу со смещением перехода 0, 4, 8

- по счетчику со смещением перехода 0, 4, 8

- по маскированному сравнению со смещением перехода (да, нет)

Команды выбираются из ОП линейно – по байтам . Смещение задается в байтах

Блок управляющих параметров подготавливается в системном пространстве для каждого канала. ЦП выставляет адрес A[19..1]= сигналы CA,CA запроса готовности канала и вместе с ними сигналы SEL(A[0]) – выбора канала и запуск инициализации

Первая пара сигналов

10 – если используются несколько процессоров, то выбирается как ведущий

11 - …….. как ведомый

Далее процессор В/В вводит последовательно первые байты инициализации процессора

- настройка на заданную шину ШВВ или ШС

- выбирает ширину шины

Вторая пара сигналов

10 – выбран первый канал

11- выбран второй канал

Далее процессор В/В вводит последовательно байты инициализации выбранного канала и начинает выполнять программу ввода/вывода.

Система команд процессора В/В включает 53 команды следующих типов

Mov dst,src
Add dst,src
And dst,src
Or dst,src
Not dst
Setb bit
Clr bit
Call addr
Jmp addr,
Jz(nz,ce,nce,b, nb,) addr
Hlt - останов программы
Xfer - запуск ПДП (завершается автоматически по условиям - прерыванием или командами перехода, останова)

Адресация косвенная регистровая – через внутренние регистры, длина команды 2-6 байт

Иерархия интерфейсов PC на базе процессора Pentium 4

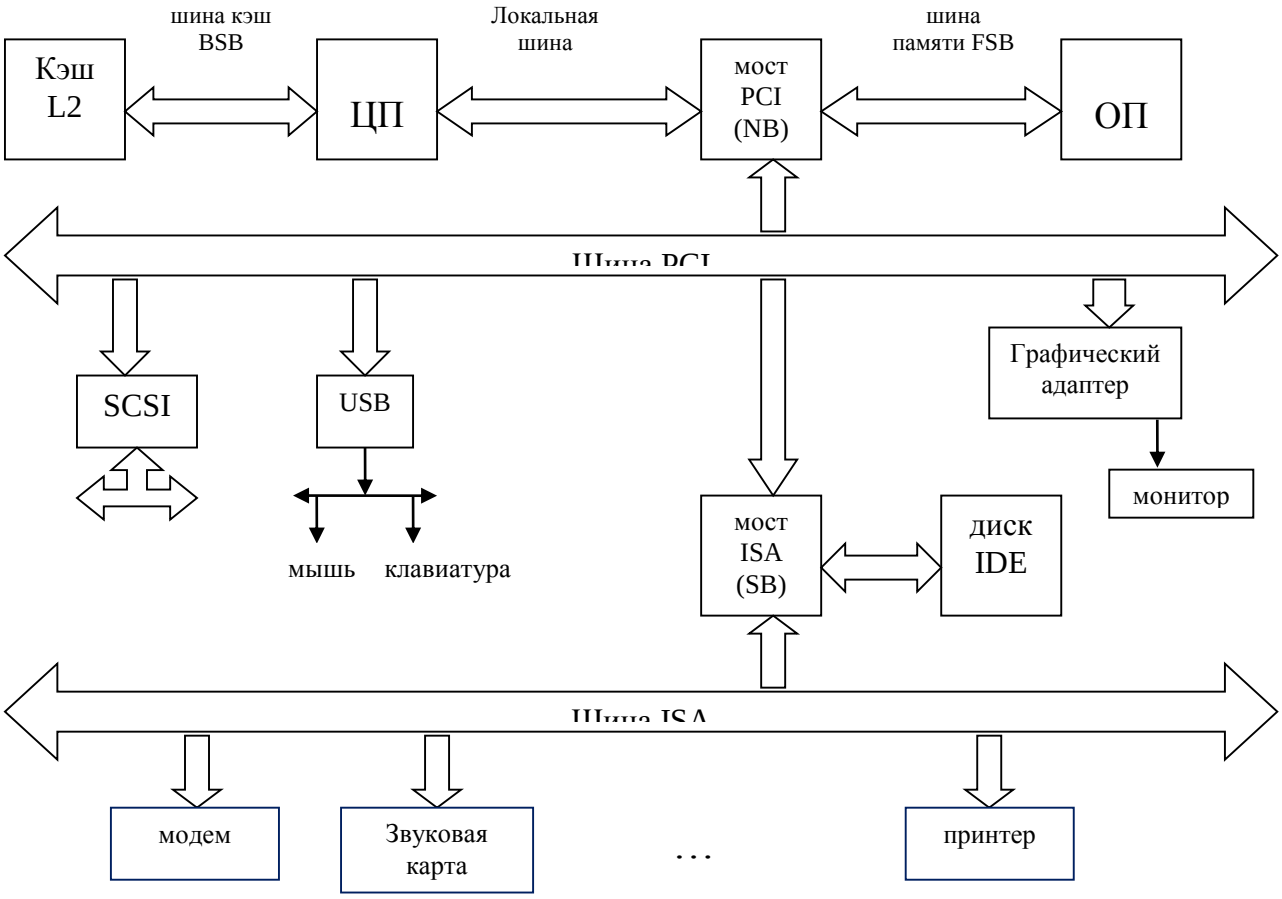
Вследствие наличия существенного недостатка у одноименной структуры компьютера в 80-е годы прошлого столетия в различных моделях компьютера была реализована

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).

3.1. 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.

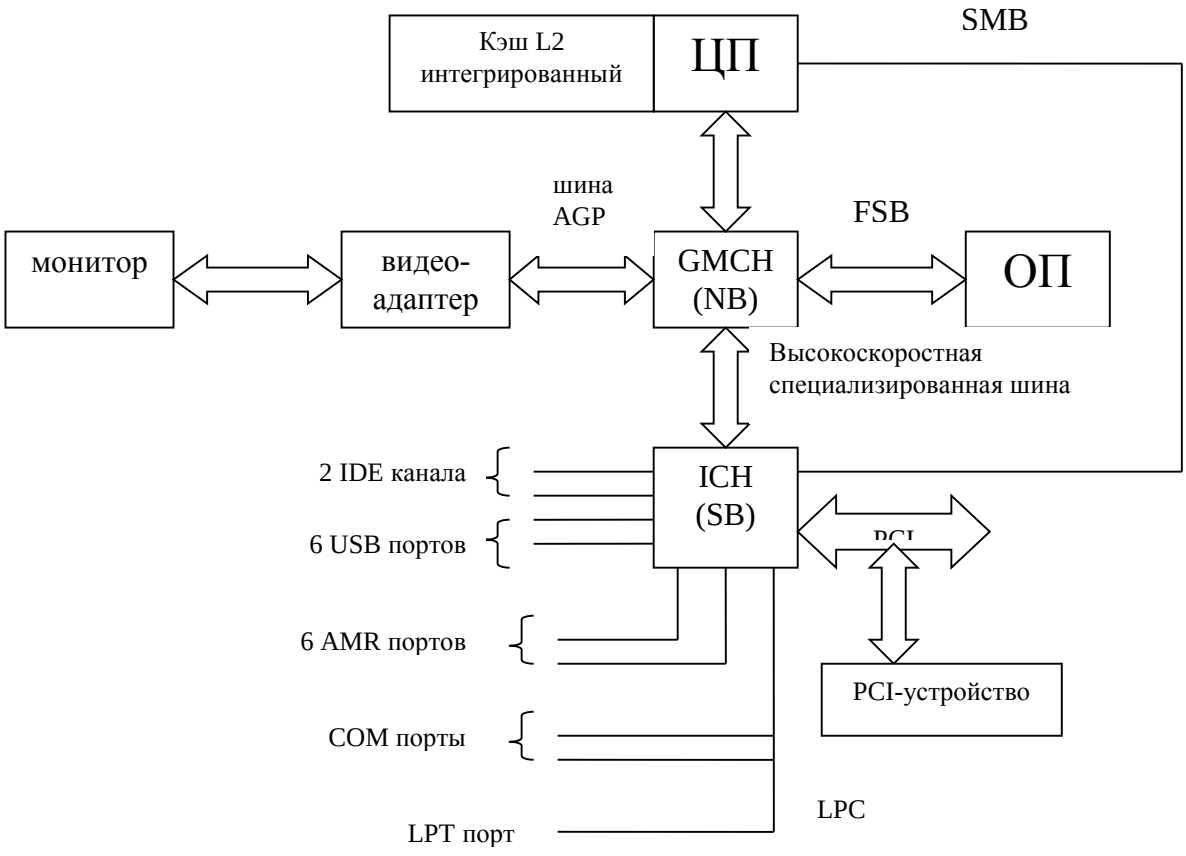
По Гуку: чипсет – набор специализированных интегральных схем, при соединении которых формируется функциональный блок компьютера.

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).

Основная тенденция в архитектуре высокопроизводительных систем – многоуровневая иерархия интерфейсов, ориентированных на обслуживание обменов, существенноразличающихся динамикой (скоростью и интенсивностью)

На рис приведена иерархия интерфейсов PC



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)**

3.1.1. 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

.

3.2. Основные характеристики и особенности в стандартных интерфейсах ПК.

3.2.1. 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.

3.2.2. 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.

3.2.3. 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.

3.2.4. 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.

3.2.4.1. 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.

3.2.4.2. 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.

3.3. 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).

3.4. 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).

1.2. 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 управления, которые проходят последовательно через ряд подключенных к ним ВУ (в частности, сигналы разрешения). *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.

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.

3.5. 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.

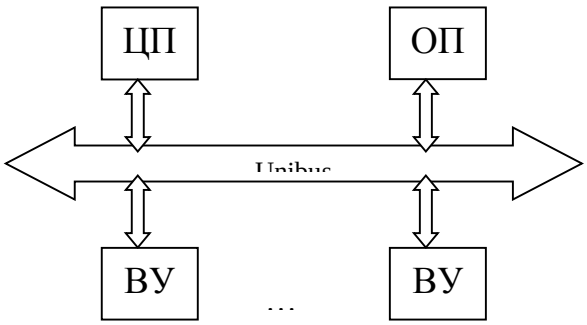
4. Магистральная структура компьютера

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.

4.3. 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.

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:

4. **Пропускная способность определяется максимальным количеством бит или байт данных, передаваемых по интерфейсу за одну секунду.**
5. **Информационная ширина (количество бит или байт данных, передаваемых параллельно по шине данных, т.е. разрядность линии).**
6. **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.

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:

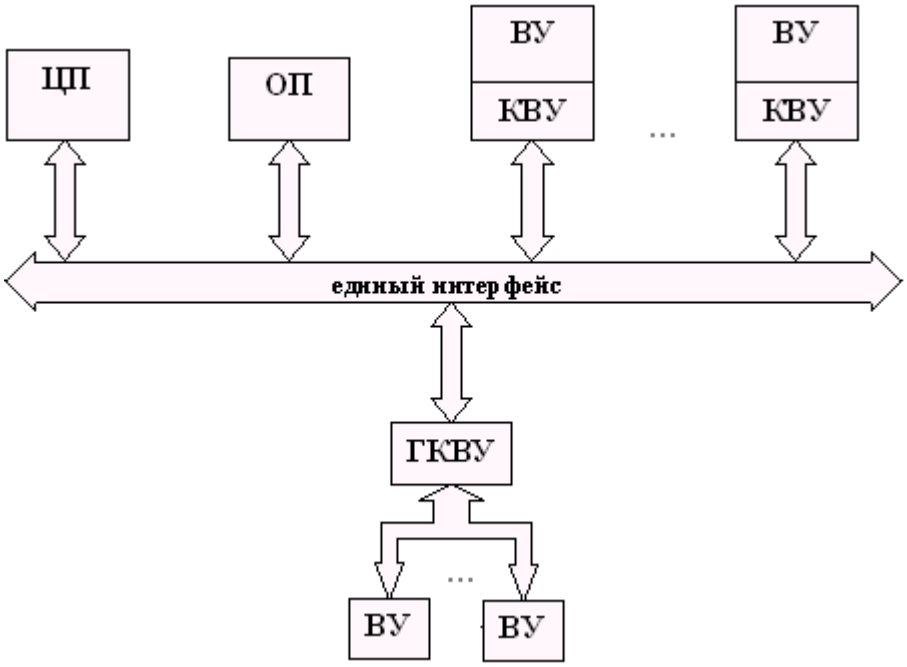
1) все устройства, как центральные, так и периферийные, для связи между собой используют одни и те же шины адреса, данных и управления;

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 организации ввода/вывода.

КУА

Сравнение ПВВ 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.

1.1.1. 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:

2) прямая (адресация порта задается во втором порте команды);
3) 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.

3.2.4. 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. $\overline{\text{INTA}}$ (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.

Многоуровневая (иерархическая) организация памяти компьютера

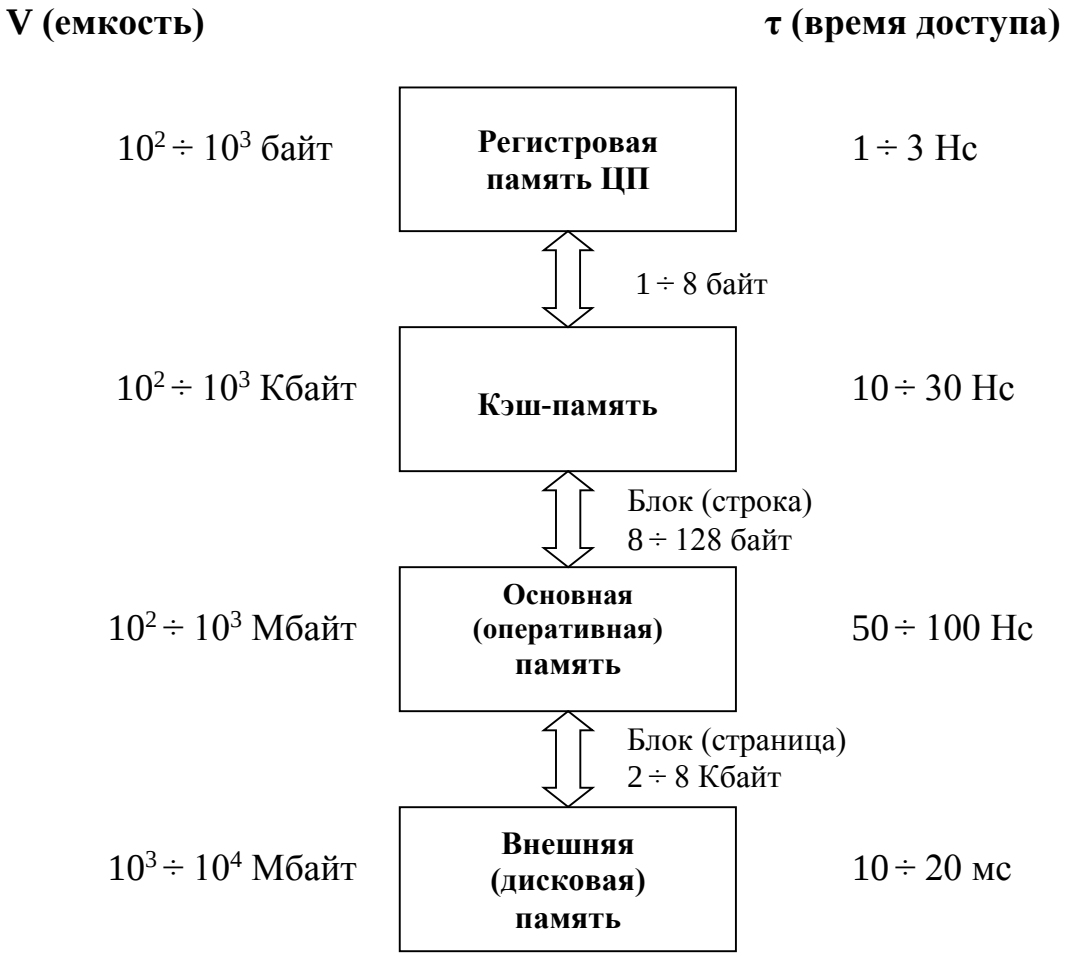
Память компьютера строится из достаточно большой совокупности разнообразных запоминающих устройств, обладающих различным принципом действия, элементной базой и характеристиками.

Основными характеристиками запоминающих устройств являются:

- емкость (объем);
- быстродействие (время доступа);
- удельная стоимость (стоимость хранения единицы информации).

Для построения памяти компьютера используется иерархический принцип (разделение ЗУ на ряд уровней), что обусловлено противоречивостью требований со стороны пользователей к различным характеристикам памяти (больше объема, меньше время доступа и дешевле).

Упрощенная схема иерархии памяти



Обмен данными в многоуровневой памяти осуществляется только между двумя соседними уровнями. В принципе, могут иметь место исключения из этого правила, например, в виде пересылок между ОП и регистрами ЦП, минуя кэш-память.

Изменение характеристик от уровня к уровню имеет следующие закономерности:

- сверху вниз емкость памяти увеличивается;
- сверху вниз быстродействие и удельная стоимость уменьшаются.

Для двух соседних уровней все необходимые для верхнего уровня данные обязательно размещаются на нижнем уровне. Передача данных между уровнями инициируется в том случае, если на верхнем уровне при обращении к нему необходимых данных не обнаружено.

В отношении Кэш-памяти (в качестве примера) обнаружение требуемой информации называется ***cache-hit*** , отсутствие требуемой информации – ***cache-miss***.

Пересылка блоков между тремя верхними уровнями реализуется чисто аппаратными средствами, в то время как пересылка между основной и внешней памятью реализуется совместно аппаратными и программными средствами.

Эффективность использования модели иерархической памяти во многом, если не во всем, объясняется существованием так называемого ***принципа локальности обращений*** (локальности ссылок). Этот принцип рассматривается в двух аспектах: пространственном и временном в отношении как команд, так и данных.

Пространственный аспект принципа локальности в отношении команд означает, что вероятность выборки команды по следующему адресу по сравнению с адресом исполняемой команды намного больше, чем по любому другому адресу. Этот принцип проявляется на линейных участках программ. По статистике, средняя длина линейного участка составляет 5-8 машинных команд.

Пространственный аспект принципа локальности в отношении данных означает, что вероятность обращения к данным по следующему адресу по сравнению с предыдущим обращением намного больше вероятности обращения по любому другому адресу. Этот принцип наиболее ярко проявляется при обработке структур данных типа массив.

Временной аспект принципа локальности в отношении команд и данных состоит в большей вероятности повторных обращений по одним и тем же адресам за командами или данными в течение небольшого промежутка времени. В отношении команд этот принцип наиболее ярко проявляется в циклах, а в отношении данных – при повторной обработке одной и той же структуры (например, массива).

Дополнение к упрощенной структуре иерархической памяти

1) Кэш-память, как правило, сама является многоуровневой (двух- или трехуровневой): L_1, L_2, L_3.

2) Использование дискового КЭШа как промежуточного уровня между ОП и ВП.

Возможные реализации:

- программная, путем выделения некоторого буфера в ОП;

• аппаратная, в виде самостоятельного ЗУ, включаемого в состав НМД (накопителя на магнитном диске).

Организация виртуальной памяти

Концепции виртуальной памяти

Под ***виртуальной памятью***, по мнению ДПС, понимается такой способ организации двухуровневой памяти (первый уровень – ОП, второй – внешняя дисковая память), при котором эта память воспринимается пользователем и, соответственно, программами, как большая одноуровневая память. При этом все пересылки между уровнями памяти являются невидимыми (прозрачными) для выполняемых программ.

Поддержка механизмов виртуальной памяти реализуется специальными аппаратными и программными средствами. Аппаратные средства в современных компьютерах обычно представлены специальным блоком **MMU – Memory Management Unit**, входящим в состав центрального процессора (CPU). В отношении процессоров семейства Intel 80x86 этот блок впервые появился в процессоре Intel 80286 и был предназначен для реализации сегментированной виртуальной памяти. Начиная со следующей модели Intel 80386, была реализована аппаратная поддержка как сегментной, так и страничной организации памяти. При этом блок MMU был разделен на две относительно независимые части в виде SU – Segment Unit и PU – Page Unit.

Программные средства поддержки виртуальной памяти входят в состав ОП, точнее, в ее ядро и обычно называются ***супервизором памяти***.

Виртуальная организация памяти базируется на разделении как основной, так и внешней памяти на блоки. Переменная длина блока является типичной для сегментной организации, фиксированная длина блока – для страничной.

Предполагается, что во внешней дисковой памяти содержатся все используемые блоки, в то время как в основной памяти - лишь копии некоторых из них. Естественно, что в любой момент времени в основной памяти содержатся копии именно тех блоков, к которым в последнее время осуществлялись обращения со стороны ЦП.

Пересылка очередного блока из внешней памяти в основную инициируется в том случае, если при обращении к этому блоку со стороны ЦП было обнаружено его отсутствие в основной памяти. Как правило, пересылка очередного блока из внешней памяти в основную предваряется процедурой освобождения места в основной памяти для этого блока. С этой целью производится выбор блока-кандидата на удаление из основной памяти. В современных компьютерах для этой цели используется стратегия LRU – Least Recently Used (наиболее давно использованный). В соответствии с этой стратегией удалению подлежит тот блок, к которому наиболее давно не было обращения. В тех случаях, когда блок-кандидат на удаление за время его нахождения в основной памяти подвергался модификации (производились обращения по записи), требуется предварительная его пересылка

из основной памяти во внешнюю для актуализации его копии во внешней памяти. Только после этого на его место пересылается новый блок.

Сегментная организация виртуальной памяти опирается на логическую структуру программы (представление программы в виде совокупности логических единиц, называемых ***сегментами***, например, сегмент кода, данных, стека). При этом, естественно, допускается использование в программе многих сегментов кода и данных, например, реализация процедур или программ в виде отдельных сегментов.

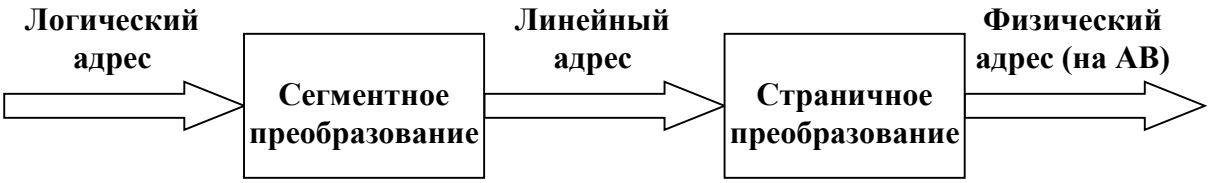
Основным недостатком сегментной организации памяти является так называемый ***эффект фрагментации***.

В свою очередь страничная организация памяти более привязана к физической реализации, но совершенно не отражает логику самой программы.

Первое упоминание об организации прозрачных для пользователя пересылок между основной и внешней памятью появилось при разработке вычислительной машины Atlas в 1961 году.

При использовании виртуальной памяти одним из базовых механизмов является ***механизм преобразования*** (трансляции) логического (виртуального) адреса в физический.

В большинстве современных компьютеров поддерживается двухступенчатая схема преобразования следующего вида:



AB – Address Bus

Для процессоров Intel сегментное преобразование является обязательным, а страничное может быть включено с помощью специального бита PG (PaGing), находящегося в управляющем регистре CR0 (CR – Control Register).

PG = 1 → включено страничное преобразование;

PG = 0 → выключено страничное преобразование, линейный адрес используется как физический.

Реализация виртуальной памяти в процессорах Intel

Организация виртуальной памяти на уровне сегментов

Структура логического адреса

Логические адреса формируются программой и состоят из двух основных частей: сегмент и смещение (seg:offset).

Первая часть адреса seg представляет собой содержимое соответствующего сегментного регистра.

Вторая часть offset представляет собой внутрисегментное смещение. Фактически, это относительный адрес байта внутри сегмента (относительно начала сегмента или его базового адреса).

Сегментные регистры являются неотъемлемой аппаратной поддержкой механизма сегментации. В младших моделях 8086-80286 использовалось четыре сегментных регистра:

CS – Code Segment – сегмент кода;
DS – Data Segment – сегмент данных;
SS – Stack Segment – сегмент стека;
ES – Extra Segment – дополнительный сегмент.

Начиная с модели 80386, появилось два дополнительных сегментных регистра: FS и GS.

Независимо от модели процессора, сегментные регистры являются 16-разрядными.

Содержимое сегментных регистров трактуется по-разному в зависимости от режима работы процессора. Основными режимами являются:

- реальный режим (RM – Real Mode);
- защищенный режим (PM – Protect Mode).

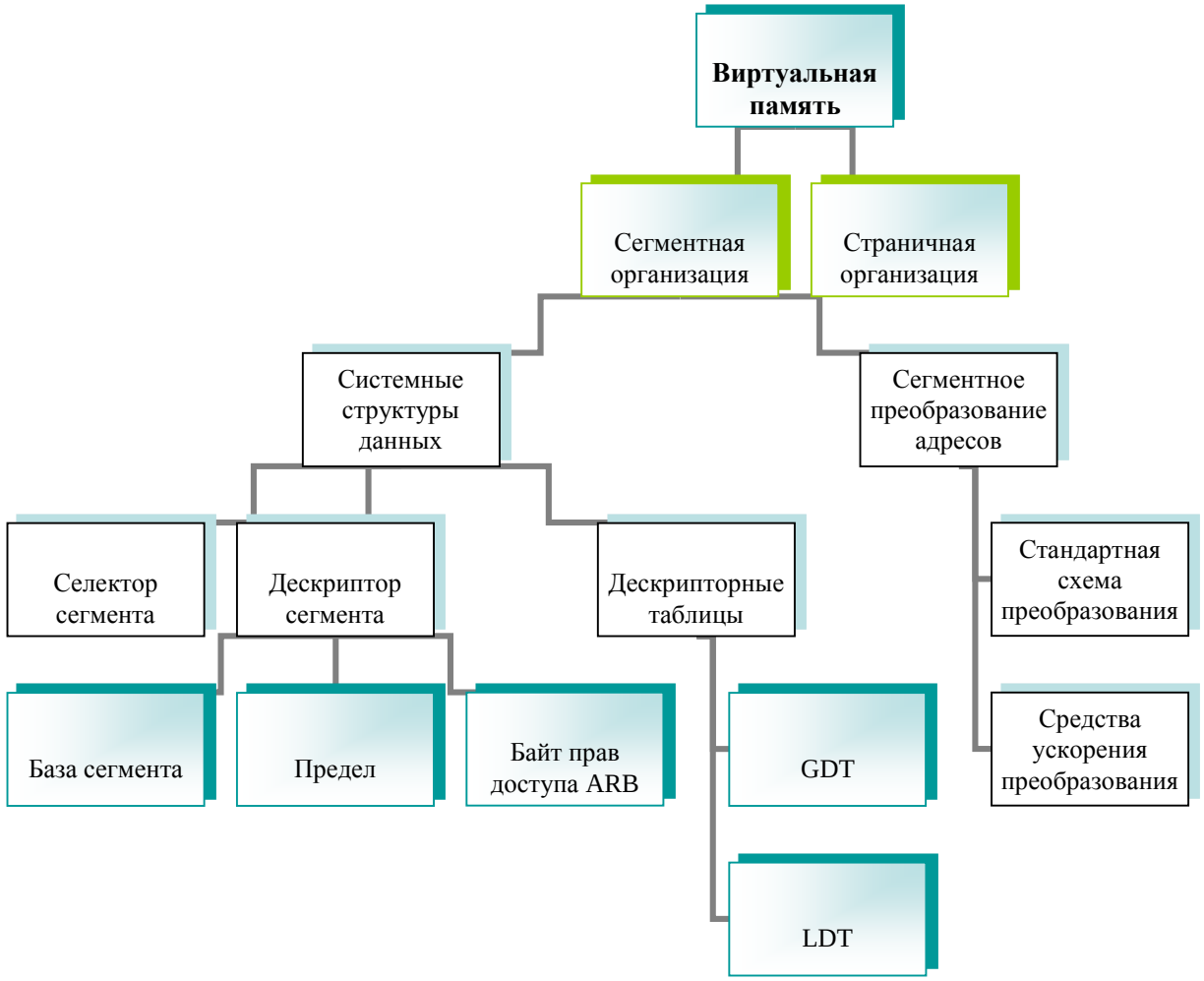
Вид используемого режима определяется специальным битом PE – Protect Enable в управляющем регистре CR0.

PE = 0 → реальный режим RM;
PE = 1 → защищенный режим PN.

При инициализации процессора он автоматически переводится в режим RM, в котором не используется виртуальная память.

В R-режиме содержимое сегментного регистра трактуется как базовый адрес сегмента, указывающий его местоположение в физической памяти. Точнее говоря, это 16 старших разрядов 20-разрядного физического адреса, задающие так называемую границу параграфа.

Для P-режима содержимое сегментного регистра трактуется как селектор сегмента.



TLB – буфер ассоциативного преобразования.

Страничная организация памяти в процессорах Intel Pentium (дополнение к разделу)

Детализированная структура TLB

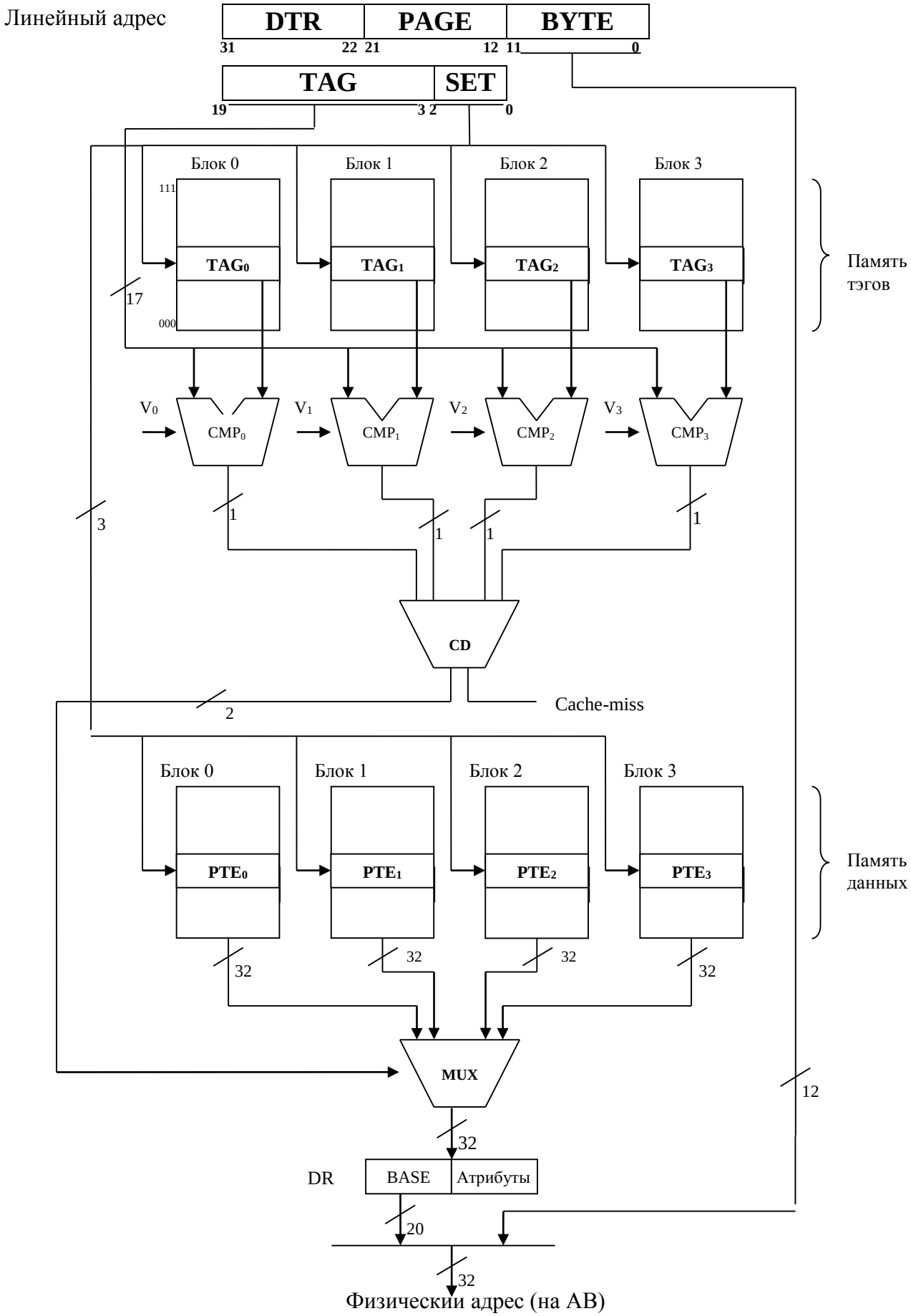
TLB построена по принципу ассоциативной по множеству кэш-памяти.

На вход TLB поступает 20-разрядный линейный адрес. На выходе TLB в случае удачного обращения появляется 20-разрядный физический адрес страницы со всеми атрибутами, сопровождающими эту страницу.

Описание схемы

Принцип работы блока TLB, как ассоциативный по множеству Кэш-памяти, обычно обозначается аббревиатурой 4WSA (четырехканальная по множеству Кэш-память, WSA - Way Set Associative).

В соответствии с этим принципом, как память тэгов, так и память данных разделяется на 4 независимых блока с возможностью обеспечения параллельного обращения к ним. Каждый из блоков содержит по 8 элементов (для памяти тэгов это тэги, а для памяти данных это элементы PTE – Page Table Entry из таблиц страниц). Обращение к блокам осуществляется по трехразрядному адресу, представляющему собой поле SET.


Линейный адрес
DTR
PAGE
BYTE
31
22 21
12 11
0
TAG
SET
19
3 2
0
Блок 0
Блок 1
Блок 2
Блок 3
111
000
TAG0
TAG1
TAG2
TAG3
17
Память тэгов
V0
V1
V2
V3
CMP0
CMP1
CMP2
CMP3
3
1
1
1
1
CD
2
Cache-miss
Блок 0
Блок 1
Блок 2
Блок 3
PTE0
PTE1
PTE2
PTE3
Память данных
32
32
32
32
MUX
12
32
DR
BASE
Атрибуты
20
32
Физический адрес (на AB)

Из памяти тэгов по адресу осуществляется параллельная выборка четырех тэгов: TAG_0, … , TAG_3, которые поступают на правые входы компараторов (CMP_0, … , CMP_3). Компараторы представляют собой схемы сравнения (совпадения), на выходе которых формируется сигнал логической единицы при совпадении операндов на входах и логического нуля при их несовпадении. На управляющий (разрешающий) вход каждого компаратора поступают значения битов достоверности (валидности) V_0, … , V_3. Эти биты поступают из специального блока LRU / достоверности, которого на схеме не показано. В этом блоке содержатся 8 элементов, обращение к которым происходит по тому же адресу SET. Каждый элемент является 7-битным и включает в себя 4 бита достоверности для каждого из блоков выбранного множества и 3 бита LRU: B_0, B_1, B_2. С помощью битов LRU выбирается один из четырех элементов в рамках данного множества, к которому наиболее долго не было обращений, следовательно, он является кандидатом на удаление.

При переключении задач в общем случае содержимое TLB объявляется недействительным путем сброса всех битов валидности.

При наличии 0 на разрешающем входе компаратора, на его выходе будет 0, даже если происходит совпадение операндов.

В той ситуации, когда содержимое всех элементов TLB актуально (все биты валидности установлены), имеет место ситуация кэш-попадания: на выходе одного и только одного компаратора формируется единичный сигнал. В соответствии с этим 4^x-разрядный выход всех компараторов фиксирует унитарный двоичный код (код с единственной единицей) номера блока, в котором произошло совпадение.

С помощью кодера (CD) осуществляется преобразование унитарного двоичного кода в позиционный в соответствии с таблицей:

Унитарный код	**Позиционный код**
$Б_0$ $Б_1$ $Б_2$ $Б_3$	
1 0 0 0	00 (0)
0 1 0 0	01 (1)
0 0 1 0	10 (2)
0 0 0 1	11 (3)

Параллельно с выборкой из памяти тэгов осуществляется выборка из памяти данных по адресу множества (SET). Выбранные 4 элемента подаются на входы мультиплексора.

Определение. *Мультиплексор* – это операционный элемент, осуществляющий коммутацию многих входов на один выход в зависимости от значения на адресном (управляющем) входе. На этот вход мультиплексора подается позиционный код номера (адреса) выбранного блока.

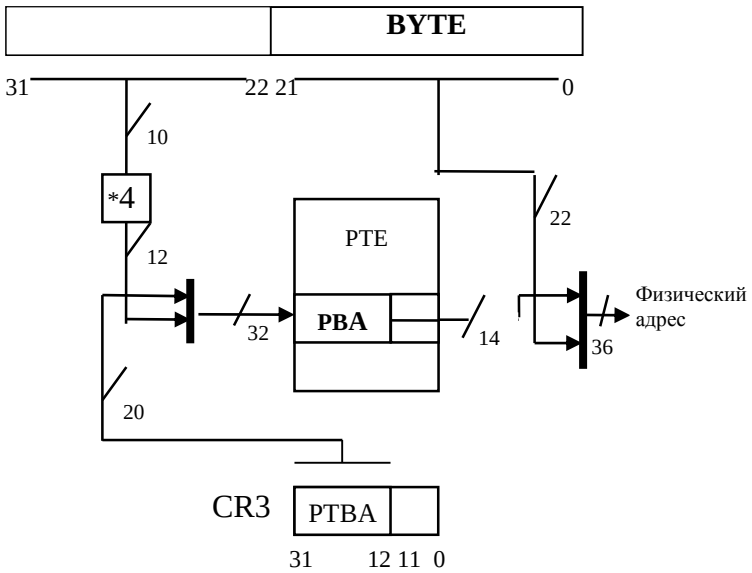
Таким образом, при кэш-попадании на выходе TLB формируется физический 32-разрядный адрес, и, кроме того, необходимые атрибуты страницы, содержащиеся в выбранном элементе PTE.

Усовершенствование страничного механизма в процессорах Intel Pentium

Это усовершенствование включает в себя:
1) возможность использования больших страниц (объемом 2-4 MB);
2) возможность использования расширенной (36 бит) шины адреса;
3) возможность объявления некоторых страниц как глобальных и сохранения информации о них в TLB при переключении задач.

Большие или расширенные страницы появились, начиная с первой модели Pentium. Для управления размером страницы используется специальный бит PSE (Page Size Extension) в управляющем регистре CR4. При PSE=0 используется обычная страница объемом 4Кб и традиционная схема преобразования. При PSE=1 разрешается использовать страницы расширенного размера. Фактическое задание размера страницы осуществляется специальным битом PS (Page Size), который включается в элемент PTE (седьмой бит).

Схема преобразования линейного адреса в физический для PSE=1 и PS=1



В процессорах с архитектурой P6 (Pentium PRO, Pentium 2, Pentium 3) поддерживается механизм расширения физического адреса. Шина адреса в этих процессорах является 36-разрядной, что позволяет увеличить объем физического адресного пространства до 64GB. Включение механизма расширенного адреса осуществляется установкой специального бита PAE (Physical Address Extension), находящегося в CR4.

При PAE=0 старшие 4 бита шины адреса принудительно обнуляются.

Поскольку архитектура P6, относящаяся к IA-32, предполагает использование 32-разрядного линейного адреса, формирование старших четырех битов расширенного 36-разрядного физического адреса возможно только при работе страничного механизма.

При PAE=1 элементы таблиц страниц (PTE) и каталога таблиц страниц (PDE) являются 64-разрядными. При этом в старшем двойном слове ир 32-х битов используются только 4 под старшие разряды базового адреса, либо таблицы страниц (PDE), либо, соответственно, страницы (PTE).

Схема преобразования линейного адреса в физический для расширенного размера адреса зависит также от размера страницы.

Схема преобразования для PS=0 (обычный размер страницы) и PAE=1 (расширенный)

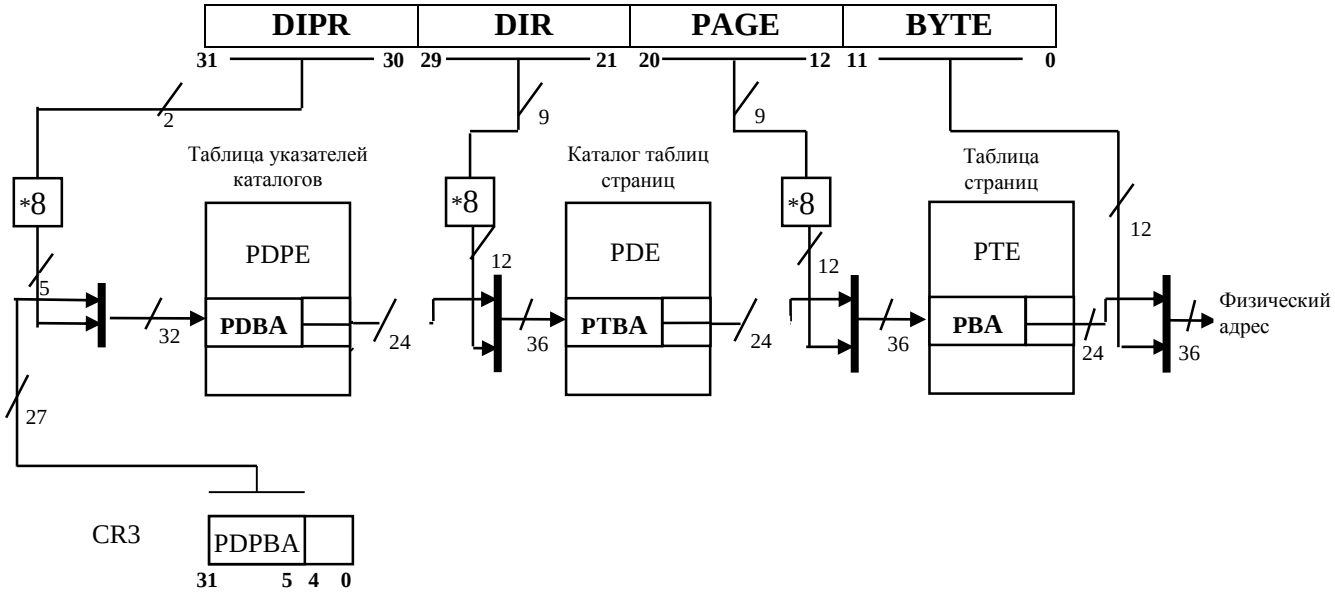


Схема преобразования является трехступенчатой. Количество элементов в таблице указателей каталогов - 4, а в каталоге таблиц страниц и, соответственно, в таблице страниц – 512.

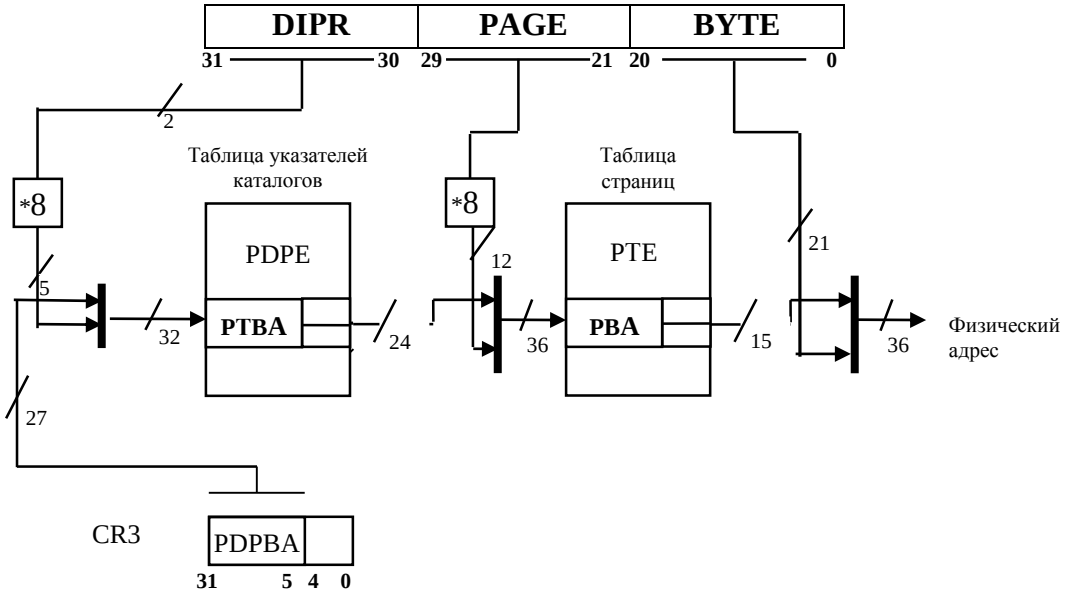
DIRP – Directory Pointer (указатель каталога);

PDPE – Page Directory Pointer Entry.

Использование 32-битного адреса для обращения к таблице указателей каталогов объясняется обязательным ее размещением в младших адресах физической памяти (младших 4-х MB).

Схема преобразования для расширенного физического адреса и расширенной страницы (PAE=1, PS=1).

В отличие от предыдущей схемы для больших страниц, в этой схеме в этой схеме размер страницы не 4, а 2 MB (2^{21} байта).



Бит глобальности

В элементы PDE и PTE включается дополнительный бит G (Global), который оказывает влияние на очистку TLB при переключении задач. Для страниц, отмеченных единичным значением этого бита, информация в TLB сохраняется при переключении задач.

Типичными страницами, которые отмечаются как глобальные, могут являться страницы кода ядра операционной системы.

СОДЕРЖАНИЕ:

1. Иерархическая организация памяти ЭВМ

Память компьютера состоит из некоторого числа разнообразных по своему действию запоминающих устройств, объединенных по многоуровневому (иерархическому) принципу (рис. 1.1). Использование подобного иерархического подхода к построению памяти связано с противоречивостью требований пользователя, предъявляемых к объему, быстродействию и

стоимости памяти. Быстродействие памяти, как правило, оценивается временем доступа. Удельная стоимость памяти определяется как отношение общей стоимости к объему.

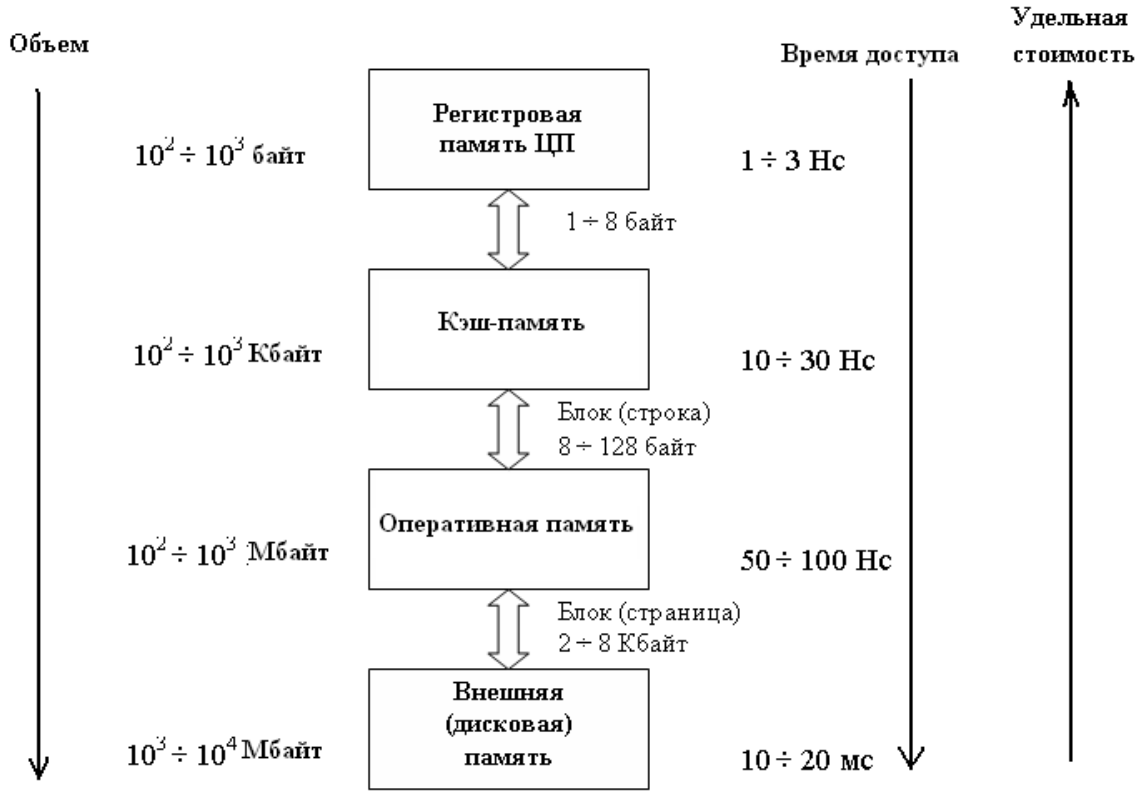


Рис. 1.1. Упрощенная схема иерархии памяти

В принципе, существует определенная зависимость между быстродействием процессора и емкостью основной памяти, при которой обеспечивается наиболее эффективное согласование этих основных устройств ЭВМ. В 70-х годах был сформулирован эмпирический закон, называемый *законом Амдала*, сущность которого состоит в следующем: для эффективной работы ЭВМ каждой единице быстродействия ЦП в 1 MIPS требуется одна единица емкости памяти – 1 Мбайт.

Основные дополнения к упрощенной схеме иерархии памяти:

1) разделение кэш-памяти на уровни;
2) использование дисковой кэш-памяти в качестве промежуточного уровня между оперативной и внешней памятью;
3) дополнение внешней (дисковой) памяти так называемой архивной памятью, в качестве которой используется ВЗУ на магнитных лентах.

2. Концепции кэш-памяти

1. Кэш-память является чисто аппаратным средством, “прозрачным” для выполняемых программ и представляет собой своеобразный буфер между основной памятью и ЦП (рис.1.2). Как правило, соотношение между емкостью кэш-памяти и ОП составляет 1/100 – 1/1000 – в зависимости от типа ЭВМ. Соотношение между временем доступа (циклом памяти) соответственно составляет 1/2 – 1/5. Как правило, передача данных между основной памятью и

процессором производится через кэш-память, хотя в принципе возможна прямая передача (на рис.1.2 показана пунктиром) .

Рис. 1.2. Передача данных между ОП и процессором

2. Кэш-память строится на основе элементов статической памяти (SRAM – Static Random Access Memory), а ОП – на элементах динамической памяти (DRAM – Dynamic RAM). Элементы SRAM представляют собой триггеры, а DRAM – по принципу действия подобны конденсатору. В связи со стеканием заряда во времени память типа DRAM требует периодической перезаписи (refresh).

3. Кэш-память и основная память разделяются на блоки одинакового объема, размер которых обычно составляет 8÷128 байт. Обычно блок ОП и кэш-памяти называют *строкой (line)*. Обмен между ОП и кэш-памятью носит блочный характер (рис. 1.2). В кэш-памяти содержатся копии тех блоков ОП, к которым в последнее время выполнялись обращения со стороны ЦП.

4. При любом обращении процессора к основной памяти определяется наличие блока, к которому производится обращение, в кэш-памяти. При нахождении блока в кэш-памяти (такая ситуация называется *кэш-попаданием – cache-hit*), осуществляется быстрое обращение (чтение или запись) со стороны ЦП в кэш-память.
Обмен между ЦП и кэш-памятью осуществляется не на уровне блоков, а на уровне слов (рис. 1.2). Под словом в данном случае понимается объём данных, участвующих в одной пересылке между ЦП и кэш-памятью или ЦП и ОП. Разрядность слова определяется разрядностью шины данных между ЦП и кэш-памятью или ЦП и основной памятью.
При отсутствии в кэш-памяти бдока, к которому осуществляется обращение (такая ситуация называется *кэш-промахом – cache-miss*), сначала инициируется пересылка блока, содержащего затребованное слово, из ОП в кэш-память, а затем осуществляется обращение к этому слову из кэш-памяти. Как правило, подобный подход имеет место при обращении по чтению. При обращении по записи в случае отсутствия блока в кэш-памяти, запись может производиться и непосредственно в оперативную память без предварительной пересылки блока в кэш-память (рис. 1.2).

5. Эффективность использования кэш-памяти определяется так называемым *принципом локальности ссылок* (доступа, обращений). Причем выделяются два вида локальности: *пространственная* и *временная*. Кроме того, принцип локальности рассматривается как в отношении команд, так и в отношении данных.
Пространственная локальность в отношении команд характеризуется тем, что вероятность выборки команды по следующему адресу, по сравнению с адресом исполняемой команды, намного больше, чем вероятность выборки команды по любому другому адресу. Этот принцип проявляется на линейных участках программы. По статистическим данным, средняя длина линейных участков большинства программ научно-технического профиля составляет 5-7 машинных команд.
Пространственная локальность в отношении данных выражается в том, что вероятность обращения к слову данных по следующему адресу намного больше вероятности обращения к данным по любому другому адресу. Этот принцип проявляется, например, при обработке массивов данных.
Временной аспект принципа локальности обращений в отношении команд предполагает большую вероятность обращения к команде по одному и тому же адресу в течение небольшого интервала времени. Этот аспект проявляется при выполнении программных циклов.

В отношении данных временной аспект принципа локальности обращений означает большое значение вероятности обращений к одному и тому же слову данных в течение небольшого интервала времени. Этот аспект проявляется при многократной обработке массивов данных. В соответствии с принципом локальности ссылок, к слову, однажды прочитанному из ОП в кэш-память, будет выполняться несколько повторных обращений (временной аспект) и, кроме того, так как вместе с одним словом из ОП в кэш-память передается целый блок, состоящий из нескольких последовательных слов, то с большой вероятностью, ряд последующих обращений будет локализован в пределах этого блока (пространственный аспект).

6. Численной оценкой эффективности принятого принципа построения кэш-памяти является процент удачных обращений (процент кэш-попаданий), определяемый как отношение числа обращений к памяти, реализуемых через кэш, к общему числу обращений. Как правило, в современных компьютерах процент кэш-попаданий составляет 95÷98 %.

7. При построении кэш-памяти необходимо решить следующие задачи, определяющие её организацию:

1) выбор принципа отображения блоков основной памяти на блоки кэш-памяти (стратегия отображения \ распределения);
2) выбор принципа удаления блоков из кэш-памяти (стратегия замещения);
3) выбор принципа поддержания актуальности копий блоков кэш-памяти в блоках основной памяти (стратегия обновления ОП).

3. Стратегии отображения (распределения)

Двумя основными стратегиями отображения блоков основной памяти на блоки кэш-памяти являются:

- прямое отображение;

- полностью ассоциативное отображение.

При использовании первого принципа любой блок ОП может отображаться только на один конкретный блок кэш-памяти. При использовании второго принципа любой блок ОП может быть отображен на любой блок кэш-памяти.

3.1. Кэш-память с прямым отображением

Упрощенная структура кэш-памяти с прямым отображением представлена на рис. 1.3 а.

Кэш-память разделяется на две части: память тегов и память данных. Память данных состоит из 128 блоков (по разрядности поля *b* в адресе основной памяти), каждый блок содержит 16 слов (по разрядности поля *c*). В частном случае, слово может соответствовать байту. Емкость кэш-памяти составляет 2^{11}слов = 2 Кслов.
Емкость основной памяти равна 2^{18} слов = 256 Кслов, что в 128 раз больше емкости кэш-памяти. Основная память состоит из 2^{14} = 16384 блоков. Адрес блока (14 разрядов) разделяется на два поля: 7 старших разрядов образуют тег (идентификатор блока), а 7 младших – индекс блока. В соответствии со значениями тега и индекса, основную память можно представить в виде матрицы блоков, в которой номер столбца определяет значение тега, а номер строки – значение индекса блока (рис. 1.3 б). На один блок кэш-памяти может отображаться любой из 128 блоков ОП с одинаковым значением индекса, то есть блоки одной строки матрицы.

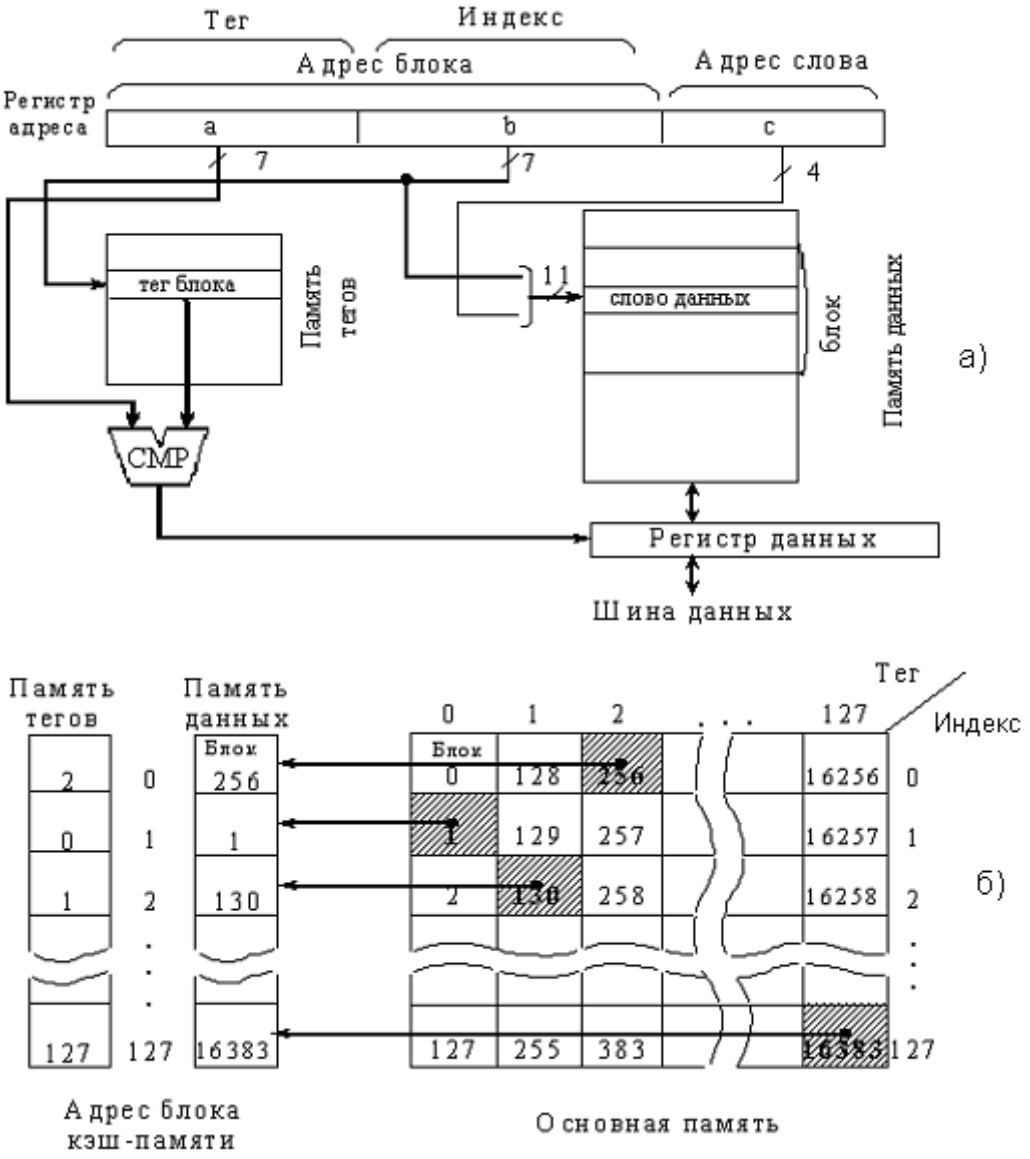


Рис. 1.3. Кэш-память с прямым отображением.

а) – структура кэш-памяти, б) – карта отображения блоков

При обращении к ОП осуществляется выборка тега из памяти тегов по адресу, представляющему собой индекс блока (поле *b*). Выбранный тег блока поступает на один из входов схемы сравнения (компаратора – СМР), на другой вход которого поступает тег запроса, то есть тег блока, к которому осуществляется текущее обращение (поле *a* из регистра адреса). При совпадении тегов (кэш-попадание) на выходе компаратора формируется сигнал, по которому разрешается запись слова из памяти данных в регистр данных при операции чтения, или запись слова из регистра данных в память данных при операции записи. При несовпадении тегов формируется сигнал кэш-промаха. При обращении по чтению реакцией на этот сигнал будет пересылка блока из ОП в кэш-память с последующим чтением из кэш-памяти. При обращении по записи могут иметь место два подхода:

- запись слова только в ОП;
- пересылка блока из ОП в кэш-память с последующей записью слова в кэш-память.

3.2. Кэш-память с полностью ассоциативным отображением

Кэш-память с полностью ассоциативным отображением представлена на рис. 1.4.

При использовании принципа полностью ассоциативного отображения любой блок ОП может быть помещен на место любого блока кэш-памяти. По аналогии с кэш-памятью с прямым отображением в кэш-памяти с полностью ассоциативным отображением можно выделить память тегов и память данных.

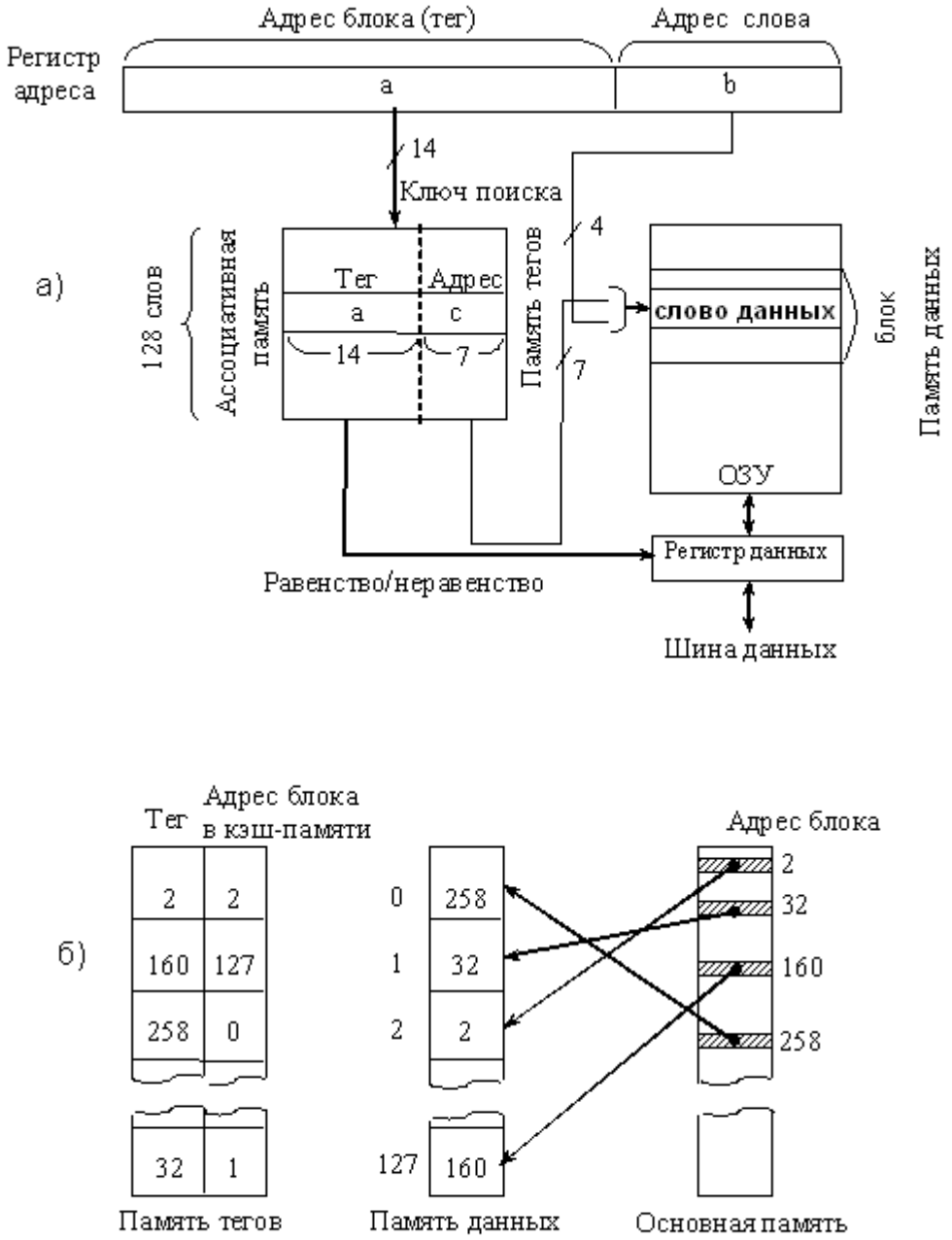


Рис. 1.4. Кэш-память с полностью ассоциативным отображением

а – структура кэш-памяти, б – карта отображения блоков

Память тегов реализуется как ассоциативная, то есть обращение к ней производится не по адресу, а по некоторому ключу (признаку, тегу). Ключом поиска в памяти тегов является 14-разрядный адрес блока. Реализацию ассоциативной памяти можно пояснить рисунком 1.5.

В ассоциативной памяти выделяются два регистровых блока:

- регистры тегов (TR_1, …, TR_n);
- регистры данных (DR_1, …, DR_n).

Между теговыми регистрами и регистрами данных существует взаимно однозначное соответствие. Для поиска информации в ассоциативной памяти во входной регистр ITR (Input Tag Register) записывается тег, по которому осуществляется поиск данных. Этот тег поступает на один из входов схем совпадений (компараторов) CMP_1, …, CMP_n. На другой вход каждого компаратора подается тег из соответствующего тегового регистра. При совпадении содержимого одного из теговых регистров с входным тегом на соответствующем компараторе будет выработан сигнал разрешения обращения E_1, …, E_n к соответствующему регистру данных. Выходы всех компараторов

объединяются на элементе ИЛИ, выход которого связан с входом разрешения обращения выходного регистра данных ODR (Output Data Register).

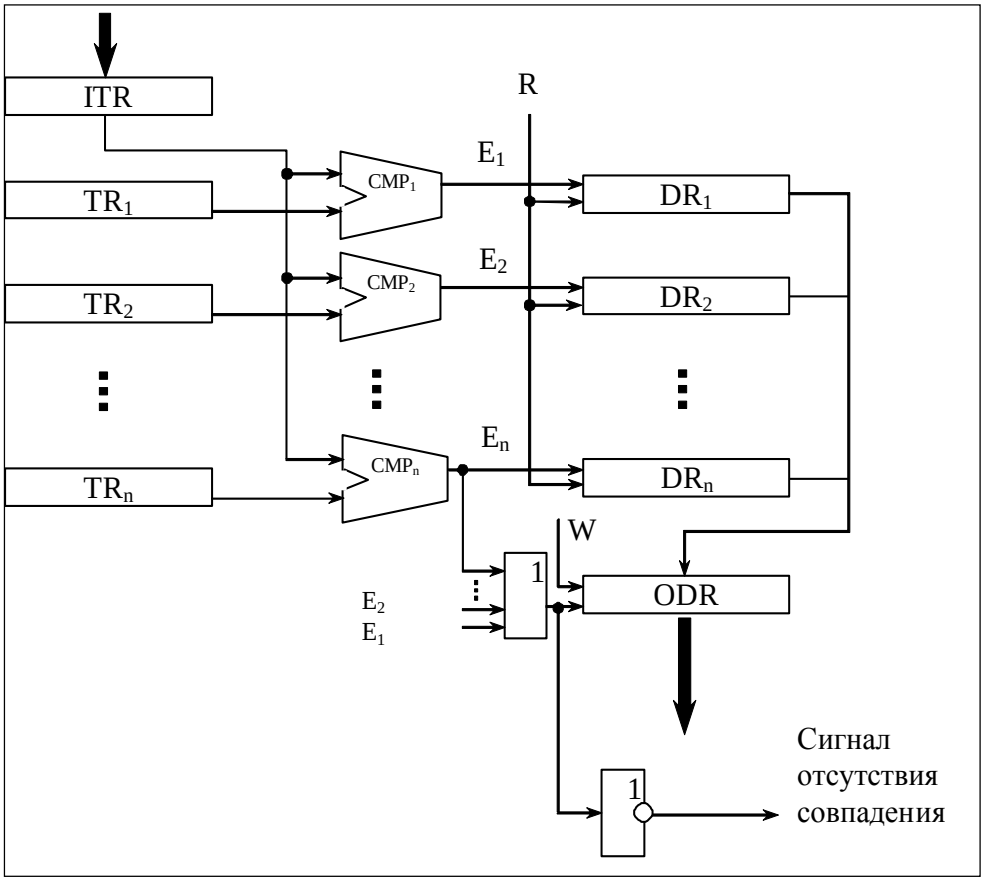


Рис. 1.5. Упрощенная схема ассоциативной памяти

При подаче сигнала чтения R в регистры данных и сигнала записи W в выходной регистр данных, на выходе ассоциативной памяти появятся данные с тегом, соответствующим входному.

В случае несовпадения входного тега с содержимым теговых регистров вырабатывается соответствующий сигнал на выходе инвертора. Этим сигналом отмечается факт отсутствия искомой информации в ассоциативной памяти.

Схема, приведенная на рис. 1.4, соответствует основной памяти емкостью 2^{18} слов = 256 Кслов так же, как и для кэш-памяти с прямым отображением (рис.1.3). Однако, в отличие от кэш-памяти с прямым отображением, где тег является 7-разрядным, в кэш-памяти с полностью ассоциативным отображением тег занимает 14 старших разрядов адреса и представляет собой адрес блока ОП. По тегу блока, используемого в качестве ключа поиска, из ассоциативной памяти тегов в случае кэш-попадания осуществляется выборка адреса блока в кэш-памяти. Для обращения к слову данных, адрес блока дополняется адресом слова в блоке со стороны младших разрядов.

Рассмотренные две стратегии отображения блоков ОП на блоки кэш-памяти в чистом виде практически не применяются. Для построения кэш-памяти, как правило, используется некоторый промежуточный вариант – кэш-память с частично ассоциативным отображением – сочетающий в себе низкую стоимость кэш-памяти с прямым отображением и высокий процент кэш-попаданий для полностью ассоциативной кэш-памяти. Наиболее часто этот промежуточный подход реализуется в виде кэш-памяти с множественно-ассоциативным отображением (кэш-памяти, ассоциативной по множеству). Другим вариантом сочетания принципов прямого и полностью ассоциативного отображения является кэш-память с распределением секторов.

3.3. Кэш-память с множественно-ассоциативным отображением

Кэш-память с множественно-ассоциативным отображением представлена на рис. 1.6.

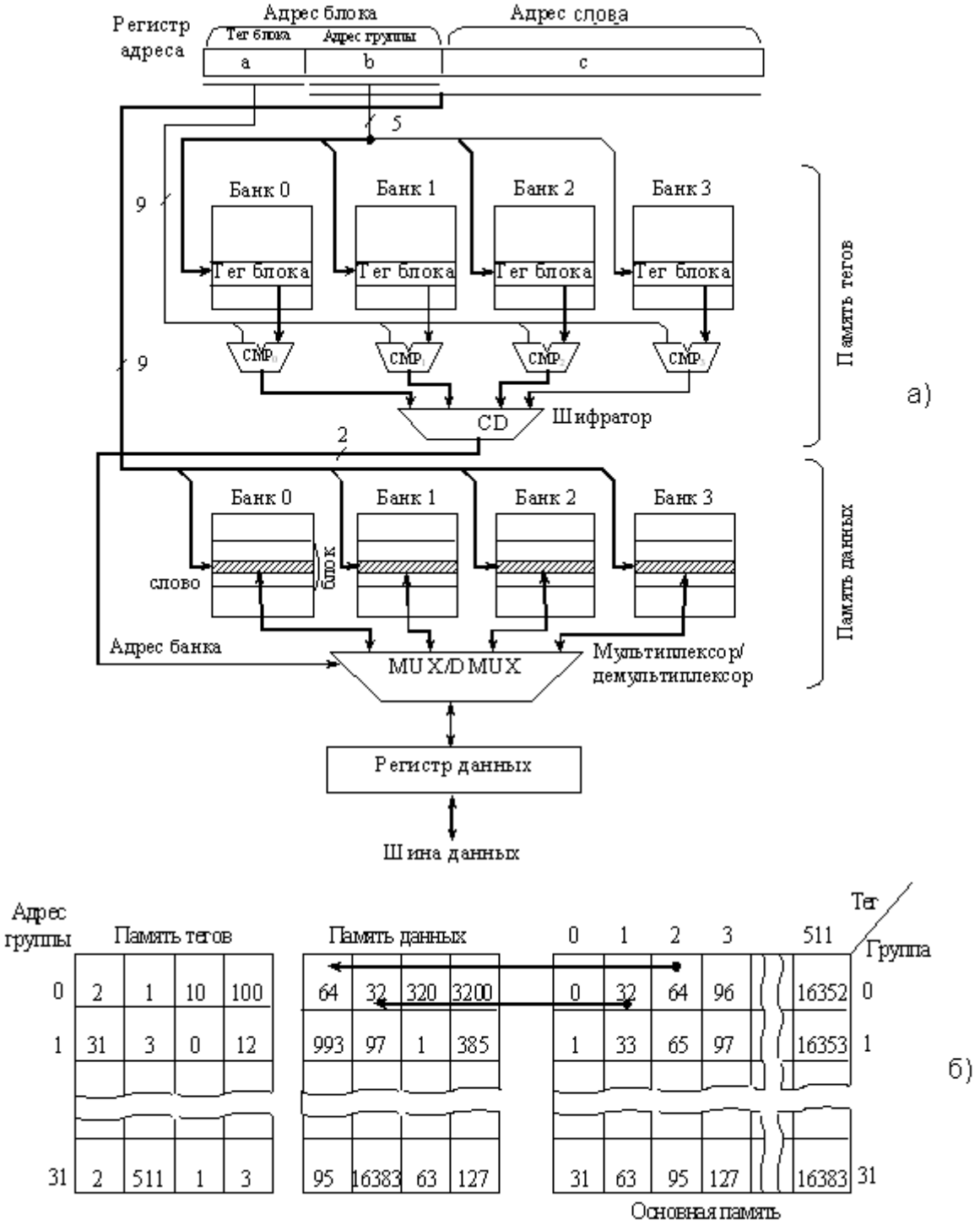


Рис. 1.6. Кэш-память с множественно-ассоциативным отображением

а – структура кэш-памяти, б – карта отображения блоков

В кэш-памяти с множественно-ассоциативным отображением осуществляется разделение блоков кэш-памяти и, соответственно, основной памяти на ряд множеств (групп). Для блоков ОП, принадлежащих одному множеству, реализуется принцип прямого отображения, то есть все блоки ОП из одного множества должны отображаться на определенное число блоков кэш-памяти соответствующего множества гораздо меньшей мощности. В свою очередь принцип отображения блоков ОП на блоки кэш-памяти внутри выбранного множества является ассоциативным: любой блок ОП, принадлежащий выбранному множеству, может помещаться на место любого блока кэш-памяти, принадлежащего соответствующему множеству.

Разрядность поля *c* определяется величиной (размером) блока в байтах, разрядность поля *b* – количеством множеств. Поле *a* занимает оставшиеся старшие разряды физического адреса.

Память данных (рис. 1.6 б) состоит из 128 блоков, которые разделяются на 32 множества (группы) по 4 блока в множестве. Каждому блоку памяти данных соответствует определенный тег из памяти тегов. Память тегов представляет собой матрицу из 32 строк и 4 столбцов.

В свою очередь, ОП состоит из 32 множеств (групп) по 512 блоков в множестве. Каждый из них претендует на занятие одного из 4-х мест в кэш-памяти. Подобная реализация называется кэш-памятью с четырехканальным доступом (4 WSA - 4 Way Set Associative cache). Так как число блоков ОП более, чем на два порядка, превосходит число блоков кэш-памяти, то существует конкуренция между блоками ОП на захват блоков кэш-памяти.

При обращении к кэш-памяти из памяти тегов выбираются 4 тега блоков из одного множества (по одному из каждого банка) для параллельного сравнения с входным тегом из регистра адреса на компараторах $СМР_0$ – $СМР_3$. При совпадении одного из тегов блоков с входным тегом на выходе шифратора (кодера) формируется адрес банка для управления пересылкой данных между регистром данных и памятью данных через мультиплексор/демультиплексор. Банк соответствует столбцу памяти тегов и памяти данных. Каждый банк снабжается собственными схемами управления доступом, что допускает параллельное обращение ко всем банкам по одному адресу.

3.4. Кэш-память с распределением секторов

Кэш-память с распределением секторов представлена на рис. 1.7.

Кэш-память состоит из 8-ми секторов по 16 блоков в каждом, а ОП – из 1024 секторов, также, по 16 блоков в каждом. В отношении секторов реализован принцип полностью ассоциативного отображения, в то время как для блоков внутри каждого из секторов используется принцип прямого отображения. Дополнительным элементом кэш-памяти является память битов достоверности. Каждый бит этой памяти отмечает наличие копии блока из соответствующего сектора ОП в кэш-памяти. Память битов достоверности представляет собой битовую матрицу размером 16x8, число строк которой соответствует числу блоков в секторе, а число столбцов – количеству секторов.

При обращении к кэш-памяти, прежде всего, проверяется наличие сектора в кэш-памяти (по ассоциативному принципу). В случае кэш-попадания определяется наличие блока путем проверки соответствующего бита достоверности. При отсутствии затребованного сектора в кэш-памяти (кэш-промах по сектору), осуществляется выделение места под сектор кэш-памяти путем выбора сектора – кандидата на удаление. Все биты достоверности данного сектора сбрасываются. Далее осуществляется пересылка затребованного блока из ОП в кэш-память с установкой бита достоверности этого блока, после чего обращение к этому блоку становится возможным.

При отсутствии затребованного блока (кэш-попадание по сектору, но кэш-промах по блоку) осуществляется пересылка блока из ОП в кэш-память в рамках имеющегося в кэш-памяти сектора. Любая пересылка блока из ОП в кэш-память сопровождается установкой бита достоверности этого блока.

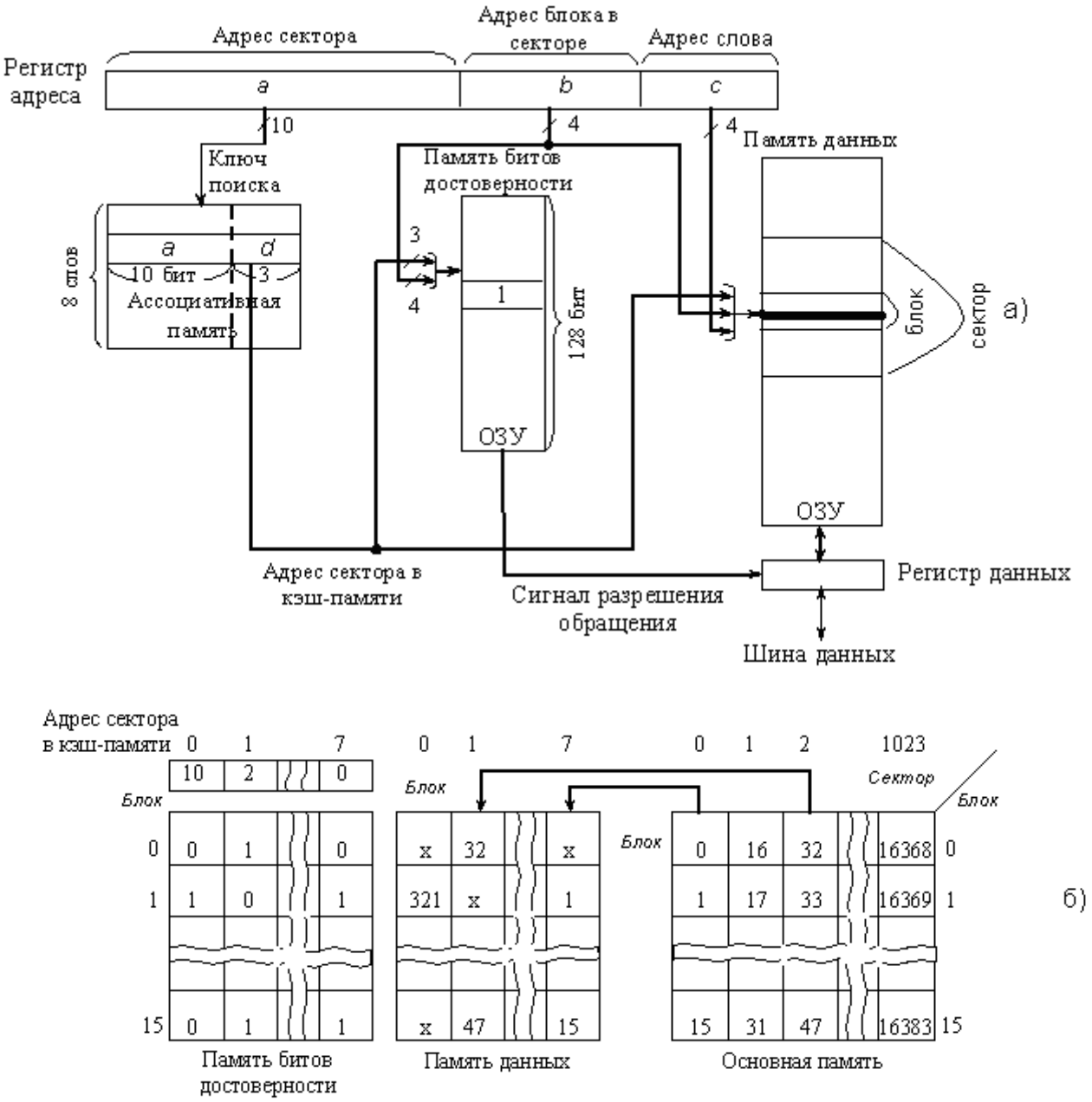


Рис. 1.7. Кэш-память с распределением секторов

а – структура кэш-памяти, б – карта отображения блоков

4. Стратегии замещения блоков в кэш-памяти

При загрузке компьютера все блоки кэш-памяти являются свободными. По мере работы происходит заполнение кэш-памяти, в результате чего наступает момент, когда свободные блоки отсутствуют. В этом случае при очередном кэш-промахе возникает проблема выбора блока – кандидата на удаление из кэш-памяти – с целью освобождения места для блока, пересылаемого из ОП. Эта проблема полностью отсутствует в кэш-памяти с прямым отображением, поскольку при таком способе организации существует единственная строка кэша, в которую может быть помещен новый блок.

В зависимости от используемого принципа отображения выбор блока – кандидата на удаление из кэш-памяти – осуществляется:

a) на всем множестве блоков для полностью ассоциативной кэш-памяти;

b) среди блоков, принадлежащих одному множеству для ассоциативной по множеству кэш-памяти;

c) среди множества секторов для кэш-памяти с распределением секторов.

К основным видам стратегии замещения блоков принято относить:

1) случайный выбор (RAND);
2) FIFO – First In First Out;
3) LFU – Least Frequency Used;
4) LRU – Least Recently Used.

При использовании принципа случайного выбора не предполагается выполнения анализа предыстории блоков, находящихся в кэш-памяти. Согласно стратегии RAND блок – кандидат на удаление – выбирается случайным образом (т.е. удален может быть любой из допустимого множества блоков[1] кэш-памяти). Реализация принципа случайного выбора может быть осуществлена с помощью счетчика, содержимое которого инкрементируется с каким-либо периодом. Частным случаем такого счетчика может быть системный таймер. Значение в счетчике, в принципе, является случайным по отношению к процессу замещения блоков в кэш-памяти и может использоваться как номер блока – кандидата на удаление.

При реализации принципа FIFO удалению подлежит тот блок из допустимого множества блоков кэш-памяти, который раньше других был помещен в кэш. Физическая реализация этой стратегии может быть осуществлена с использованием принципа циклического буфера, в котором блоки выстраиваются в порядке их поступления в кэш. Блок – кандидат на удаление выбирается из вершины буфера.

Принцип LFU предполагает, что удалению из допустимого множества блоков кэш-памяти подлежит блок с наименьшей частотой обращений. Для реализации этого принципа необходимо каждый блок кэш-памяти снабдить счетчиком обращений, значение которого инкрементируется при обращении к блоку. Тогда при необходимости записать новый блок из ОП на место одного из блоков кэш-памяти анализируются значения счетчиков всех блоков из допустимого множества. Удалению подлежит блок с наименьшим значением счетчика обращений. Процесс выбора блока – кандидата на удаление завершается обнулением (сбросом) счетчиков обращений всех блоков кэш-памяти.

Наиболее употребительной при реализации кэш-памяти является стратегия LRU или её упрощенная модификация в виде Pseudo LRU. Данная стратегия предполагает, что удалению подлежит тот блок из допустимого множества блоков кэш-памяти, к которому наиболее долго не было обращения.

Физическая реализация этой стратегии может быть осуществлена с использованием LRU-стека. Отличия LRU-стека от классического стека столь велики, что, как правило, аппаратная реализация LRU-стека основывается не на стековой организации, а на автоматной модели. Простейшая автоматная модель для выбора одного из трех блоков приведена на рис. 1.8.

Вершины графа переходов соответствуют состояниям автомата, при этом состояние **abc** (a, b, c = 0, 1, 2) представляет собой номера блоков в множестве кэш-памяти в соответствии с порядком обращения к ним. Состояние **abc** показывает, что к блоку **a** обращение было последним, а к блоку **c** – наиболее давним. Число состояний равно 3! = 6.

[1] Под допустимым множеством здесь и далее имеется в виду множество блоков для полностью ассоциативной кэш-памяти и кэш-памяти ассоциативной по множеству и множество секторов для кэш-памяти с распределением секторов

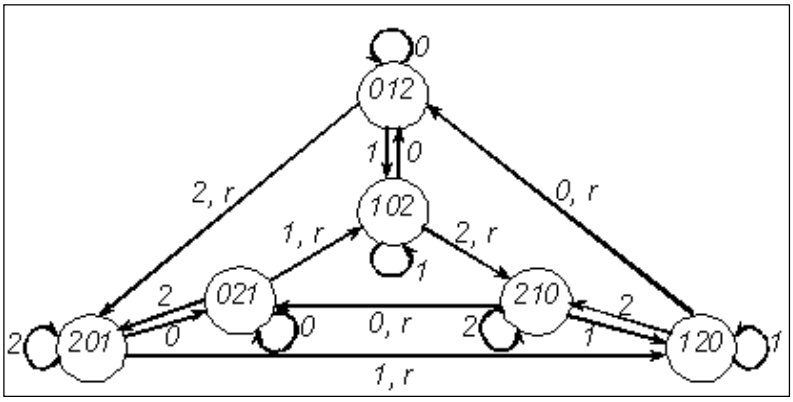


Рис. 1.8. Простейшая автоматная модель для выбора одного из трех блоков

Переходы между состояниями осуществляются, во-первых, при очередном обращении в пределах данного множества и, во-вторых, в ситуации кэш-промаха. Дуги графа идентифицированы номерами блоков, при обращении к которым осуществляется соответствующий переход. Переходы, связанные с кэш-промахом и выбором блока – кандидата на удаление – отмечаются символом **r**. Блоком, подлежащим удалению в состоянии **abc**, является блок **c.**

В виду существенного увеличения сложности автомата с возрастанием числа блоков, на множестве которых определяется блок – кандидат на удаление, использование автоматных моделей для реализации LRU-стека не получило широкого распространения в кэш-памяти.

Одним из подходов к реализации принципа LRU является использование некоторой совокупности битов с определенным алгоритмом их установки и сброса для каждого множества блоков кэш-памяти, среди которых происходит выбор блока – кандидата на удаление. Так, например, во внутренней кэш-памяти процессоров Intel 80x86 с организацией 4WSA, используются 3 бита для каждого множества: b_0, b_1, b_2, называемые битами LRU. Алгоритм установки этих битов, в зависимости от последовательности обращений к блокам (строкам, линиям) L_0, L_1, L_2, L_3 приведен на рис.1.9.

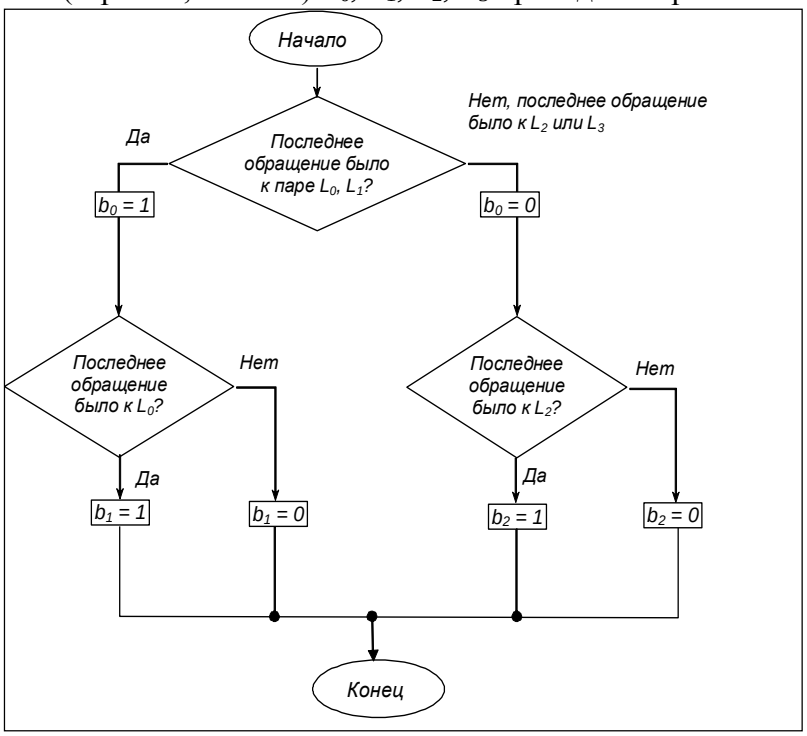


Рис. 1.9. Алгоритм установки битов LRU

Выбор блока – кандидата на удаление – по текущему состоянию битов LRU производится в соответствии с табл. 1.1.

Таблица 1.1

Состояния битов LRU			Кандидат на удаление
b_0	b_1	b_2	
0	0	×	L_0
0	1	×	L_1
1	×	0	L_2
1	×	1	L_3

× – неопределенное состояние

Использование этого алгоритма выбора блока – кандидата на удаление приводит к реализации видоизмененной стратегии LRU, называемой Pseudo LRU. В отличие от классического принципа LRU, где удалению подлежит блок, к которому дольше других не было обращений, стратегия Pseudo LRU в ряде ситуаций в качестве кандидата на удаление выбирает предпоследний блок в цепочке обращений.
Рассмотрим подобную ситуацию. Пусть последовательность обращений к блокам во времени имеет вид, представленный на рис. 1.10.

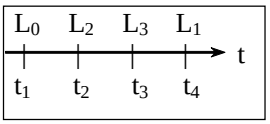


Рис. 1.10. Последовательность обращений к блокам во времени

Таблица 1.2

Моменты времени	Состояния битов LRU		
	b_0	b_1	b_2
t_1	1	1	×
t_2	0	1	1
t_3	0	1	0
t_4	1	0	0

× – неопределенное состояние

Тогда в соответствии с порядком обращений к блокам во времени состояния битов LRU будут изменяться согласно табл. 1.2. Блоком – кандидатом на удаление – после момента времени t_4 по принципу LRU должен быть блок L_0, в то время как в соответствии с используемой стратегией (см. табл. 1.1) выбирается блок L_2 – предпоследний в цепочке обращений. Следует отметить, что на практике в 1/3 случаев при выборе блока – кандидата на удаление с использованием стратегии Pseudo LRU выбирается предпоследний блок.
По мнению большинства специалистов, сложность реализации возрастает от стратегии RAND к стратегии LRU, точно также возрастает процент кэш-попаданий (эффективность стратегии). Однако, как показали исследования на моделях реальных процессов, реализация принципа случайного выбора лишь незначительно снижает эффективность использования кэша по сравнению со стратегиями, учитывающими тем или иным способом предысторию обращений к блокам кэш-памяти. Данное утверждение можно проиллюстрировать приведенной ниже таблицей, в которой представлено сравнение долей промахов для стратегий LRU и RAND при нескольких размерах кэша и разных размерах множества для ассоциативной по множеству кэш-памяти. Размер блока кэш-памяти составляет 16 байт.

Таблица 1.3

Ассоциативность:	**2-канальная (2 WSA)**		**4-канальная (4 WSA)**		**8-канальная (8 WSA)**	
Размер кэш-памяти	LRU	Random	LRU	Random	LRU	Random
16 KB	5.18%	5.69%	4.67%	5.29%	4.39%	4.96%
64 KB	1.88%	2.01%	1.54%	1.66%	1.39%	1.53%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Как видно из таблицы 1.3 при большом объеме кэш-памяти процентное соотношение кэш-промахов при использовании принципа случайного выбора практически приближается к процентному соотношению кэш-промахов при использовании алгоритма LRU, а следовательно не является менее эффективным.

5. Проблема поддержки актуальности копий и способы её решения. Стратегии обновления основной памяти

Обращения к блокам кэш-памяти со стороны центрального процессора могут производиться как по чтению, так и по записи. При модификации блока кэш-памяти после обращения по записи, его копия в оперативной памяти на некоторое время становится неактуальной. Так как к этому блоку оперативной памяти с модифицированной копией могут осуществляться обращения со стороны других устройств (процессоров и/или устройств ввода-вывода), то проблема согласования блоков оперативной памяти и кэш-памяти является весьма актуальной. Для решения данной проблемы предусмотрены методы обновления ОП, которые можно разделить на две большие группы:

- метод сквозной записи (Write Through – WT);
- метод обратной записи (Write Back – WB).

По WT обычно обновляется слово, хранящееся в ОП. Если в кэш-памяти существует копия этого слова, то она так же обновляется. Если же в кэш-памяти отсутствует копия этого слова, то либо из ОП в кэш-память пересылается блок, содержащий это слово (метод WTWA – WT With Allocation – Сквозная запись с распределением), либо этого не делается (WTNWA – WT Non With Allocation – Сквозная запись без распределения). При любом обращении к памяти по записи, обращение идет к ОП, то есть содержимое ОП остается актуальным. При использовании любой из модификаций метода сквозной записи нет необходимости копировать удаляемый из кэш-памяти блок в ОП, так как его копия в ОП поддерживается актуальной. Несмотря на большое достоинство метода WT, связанное с отсутствием проблемы согласования содержимого ОП и кэш-памяти, его существенным недостатком является высокий процент обращений к ОП. В пределе, доля обращений к ОП стремится к общей доле обращений по записи. По статистике, доля обращений к ОП по записи составляет порядка 10-15% от общего числа обращений.
По методу обратной записи модификация содержимого блока в кэш-памяти при обращении по записи не является причиной изменения копии этого блока в ОП. При обращении по записи к блоку, отсутствующим в кэш-памяти, обязательно осуществляется пересылка блока из ОП в кэш-память с последующей его модификацией только в кэш-памяти. Метод WB требует копирования блока из кэш-памяти в ОП только в момент удаления блока из кэш-памяти, то есть когда этот блок становится кандидатом на удаление. Чистая стратегия WB требует обязательной пересылки блока из кэш-памяти в ОП при его удалении из кэш-памяти, независимо от того, подвергается ли он модификации за время своего нахождения в кэш-памяти.
Существуют разновидности стратегий WB, одна из которых FWB (флаговая обратная запись) – предусматривает наличие специального флага модификации (UPDATE) у каждого блока. Установка этого флага производится при обращении к блоку по записи. При удалении блока из кэш-памяти его копирование в ОП осуществляется только при установленном флаге модификации.

В целях повышения эффективности кэш-памяти, выражающейся, в данном случае, в ускорении обращения к затребованному блоку со стороны ЦП, может быть использована еще одна модификация стратегии WB в виде метода FRWB (флаговая регистровая обратная запись). Использование этого метода связано с наличием дополнительного буфера между ОП и кэш-памятью, в который предварительно пересылается удаляемый из кэш-памяти блок, после чего происходит запись нового блока из ОП в кэш-память и этот блок становится доступным для обращения со стороны ЦП. Запись блока из буфера в ОП реализуется после пересылки блока из ОП в кэш-память.

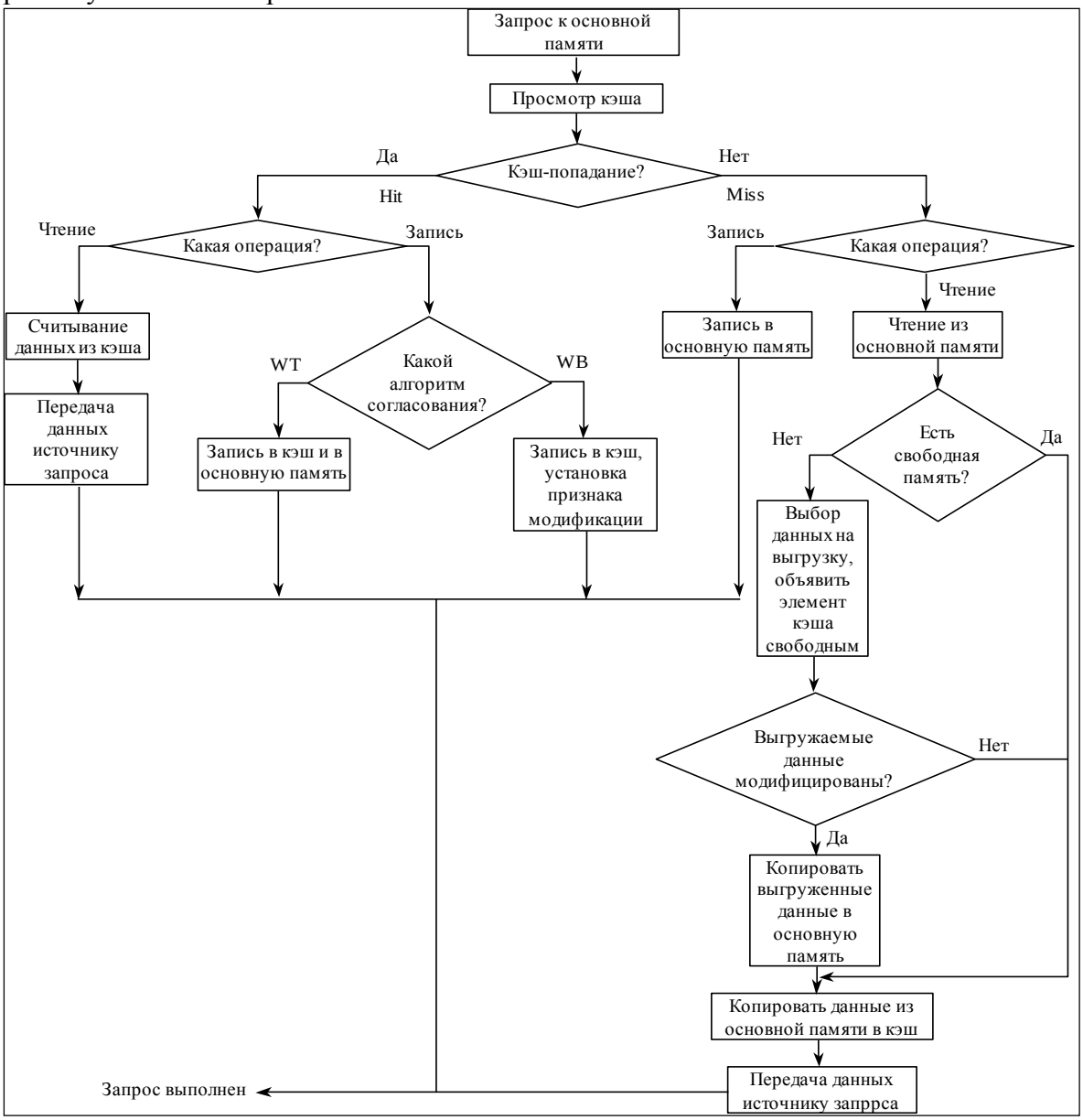


Рис. 1.11. Упрощенная схема организации запроса к памяти

(Схема заимствована из книги «Сетевые операционные системы», авторов Н.А. Олифер, В.Г. Олифер)

Стратегия WT требует больших временных затрат, однако содержимое блоков ОП полностью согласовано с содержимым блоков кэш-памяти. Стратегия WB требует значительно меньших затрат времени на реализацию, однако, на какое-то время копии некоторых блоков в ОП становятся не актуальными по сравнению с их модификациями в кэш-памяти. Однако, к этим блокам может потребоваться обращение со стороны устройств ввода/вывода, поэтому при использовании двухуровневой кэш-памяти для первого уровня обычно используется стратегия WT, а для второго уровня – WB.
Рис. 1.11 демонстрирует, каким образом выбор стратегии обновления влияет на формирование запроса к ОП.

На схеме реализована модификация метода WT в виде WTNWA, так как при кэш-промахе осуществляется запись только в ОП. В свою очередь, представлена модификация стратегии WB в виде метода FWB, что видно из наличия действия – «установка признака модификации» при записи.

6. Обобщение понятия кэширования

В общем плане, кэширование следует рассматривать как универсальный метод, используемый для ускорения доступа к ОП, к диску и другим видам запоминающих устройств. Если кэширование применяется для уменьшения среднего времени доступа к ОП, то в качестве кэш-памяти используют быстродействующую статическую память (SRAM). Если кэширование используется системой ввода/вывода для ускорения доступа к данным, хранящимся на диске, то в этом случае роль кэш-памяти выполняют буферы в ОП, в которых оседают наиболее активно используемые данные. Виртуальную память так же можно считать одним из вариантов реализации принципа кэширования данных, при котором ОП выступает в роли кэш-памяти по отношению к внешней дисковой памяти. Однако, в этом случае кэширование используется не столько для того, чтобы уменьшить время доступа к данным, сколько для того, чтобы заставить дисковую память подменить ОП за счет перемещения временно неиспользуемых кода и данных на диск с целью освобождения места для активных процессов в ОП. В результате, наиболее интенсивно используемые данные оседают в ОП, в то время как остальная информация хранится в более объемной и менее дорогостоящей внешней памяти.

_Организация центральных процессоров.

CPU – Central Processing Unit.

ЦП выполняет в компьютере двоякую функцию:

3) **Как обрабатывающее устройство: ЦП осуществляет выполнение программ, связанных с какой-либо обработкой данных.**
4) **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.

1. 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.

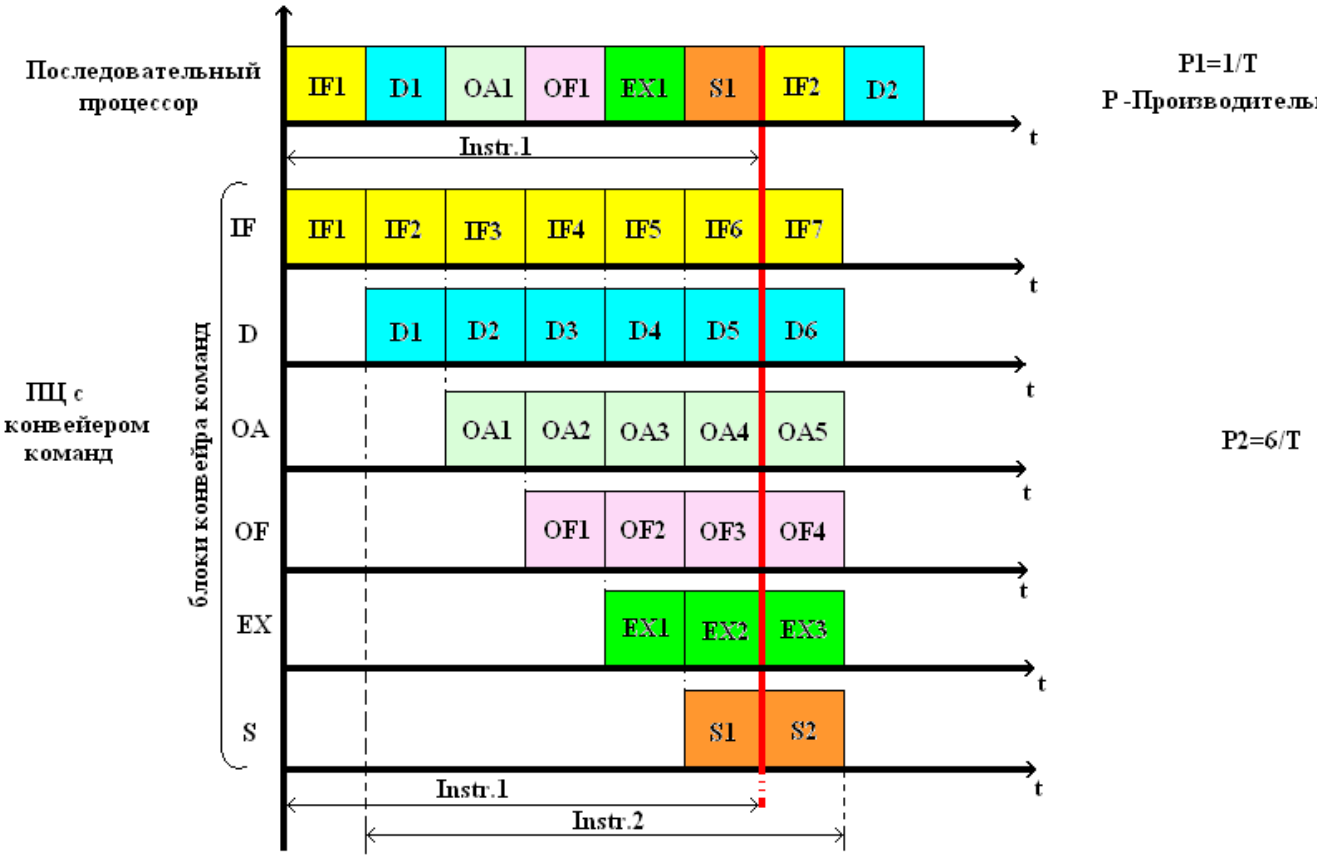
При классическом подходе выполнение основных машинных команд (например, арифметических или логических), связанных с обработкой данных, можно разделить на 6 этапов (фаз, стадий):

7. **IF (Instruction Fetch - выборка команды).**
8. **D (Decode - декодирование команды).**
9. **OA (Operand Address - формирование адресов операнда).**

10. **OF (Operand Fetch - выборка операнда).**
11. **EX (Executive - выполнение операции).**
12. **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:

4. **длительности всех фаз конвейера одинаковы;**
5. **обеспечивается постоянная загрузка вех блоков конвейера (без изменения порядка действия);**
6. **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.