Трегубов Владимир Михайлович

Каф. Прикладной математики и информатики, 3 этаж, 7 уч. зд.

Дирекция института ТКИ, 7 уч. зд., ком. 120

Периферийные устройства: программирование на языке Ассемблера

Литература (основная)

1. В.И.Юров. Ассемблер. Учебник для вузов. – СПб.: Питер, 2003. – 637 с.
2. В.И.Юров. Ассемблер. Практикум. 2-е изд. - СПб.: Питер, 2006. – 399 с.
3. А.Ю.Александров, А.Н.Козин, В.М.Трегубов ЛАБОРАТОРНЫЙ ПРАКТИКУМ «Программирование на языке ассемблера». Казань: КГТУ им.А.Н.Туполева, 1997, 47с.
4. А.Ю.Александров, В.М.Трегубов ЛАБОРАТОРНЫЙ ПРАКТИКУМ «Программирование на языке ассемблера» Часть 2. Учебное пособие. Казань: Мастер Лайн, 1997, 36с.
5. Р. Джордейн. Справочник программиста персональных компьютеров типа IBM PC, XT и AT.: Пер. с англ.- М.: Финансы и статистика, 1991.-544 с.

6. С.В. Зубков. Assembler для DOS, Windows и Unix. –М.: ДМК, 1999.-640 с .

7. В.Ю.Пирогов. Ассемблер для Windows. – СПб.: БХВ-Петербург, 2003- 656С.

8. В.А.Авдеев. Периферийные устройства. Интерфейсы, схемотехника, программирование. – М.: ДМК Пресс, 2009. – 848 с.

Дополнительная

1. Рудаков П.И., Финогенов К.Г. Программируем на языке ассемблера IBM PC: Части 1 и 2.-М.:"Энтроп", 1995.-164, 162с

2. Финогенов К.Г. Самоучитель по системным функциям MS-DOS.-М.: "МП Малип", 1993.-264 с.

3. Пильщиков В.Н. Программирование на языке ассемблера IBM PC. М.: ДИАЛОГ МИФИ, 1994. 288с.

4. Данкан. Профессиональная работа в MS DOS: Пер. с англ.- М.:Мир, 1993.-510 с.

5. Магда Ю.С. Ассемблер для процессоров Intel Pentium. – СПб.:Питер, 2006. – 410с.
6. В.Кулаков. Программирование на аппаратном уровне: специальный справочник. - СПб.:Питер, 2006. – 847с.
7.В.Несвижский. Программирование аппаратных средств в Windows. - СПб.:БХВ-Петербург, 2004. – 880с.
8. В. Комиссарова. Программирование драйверов для Windows. - СПб.:БХВ-Петербург, 2004. – 880с.

http://gen.lib.rus.ec/search

Архитектура персонального компьютера

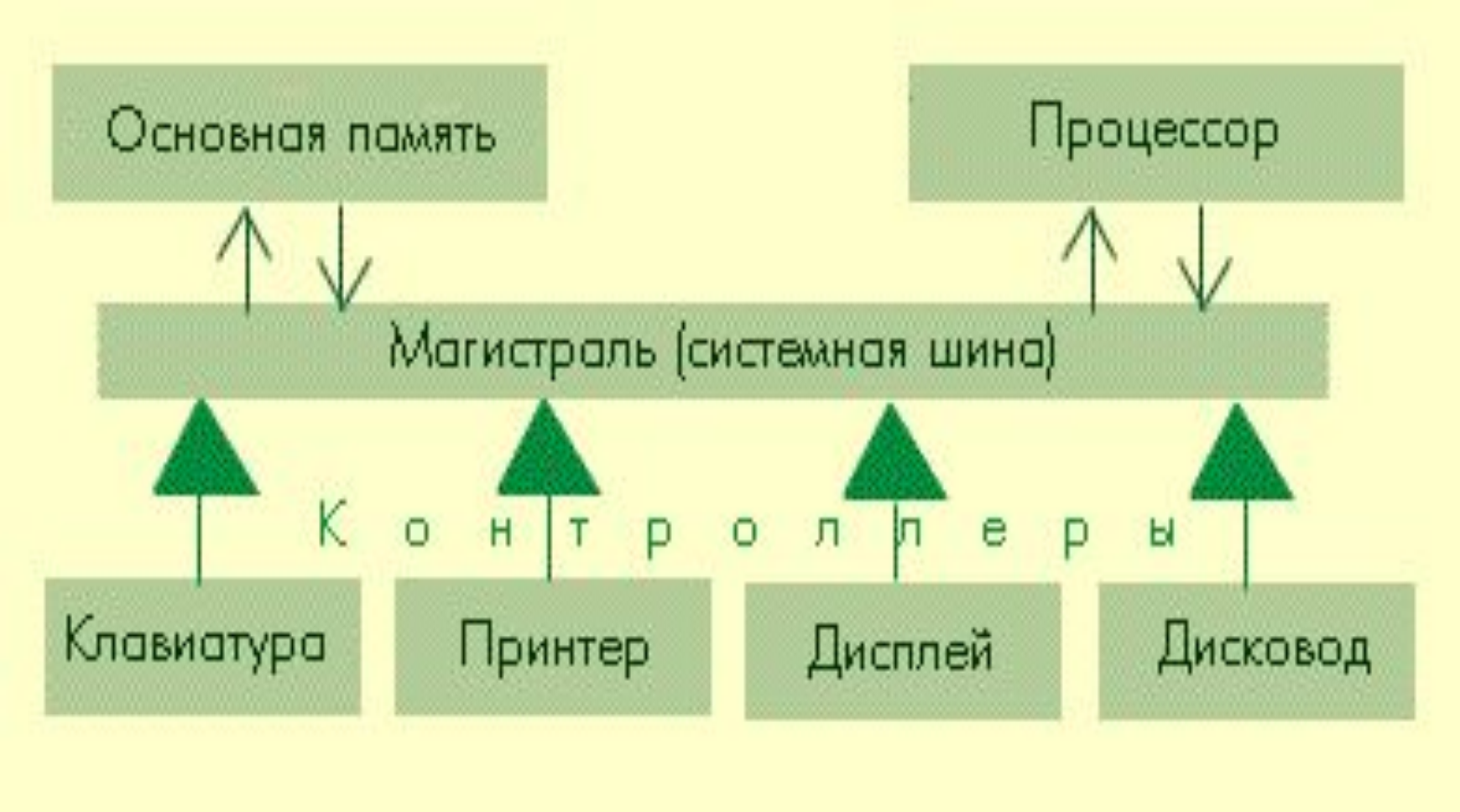
Понятие архитектуры ЭВМ

Архитектурой компьютера называется ее логическая организация, структура и ресурсы, которые может использовать программист.

Элементы архитектуры

- Структурная схема ЭВМ
- Средства и способы доступа к элементам этой структурной схемы
- *Организация памяти и способы её адресации*
- Организация и разрядность интерфейсов ЭВМ, *набор и доступность регистров*
- *Набор и формат машинных команд процессора*
- *Способы представления и форматы данных*
- *Правила обработки прерываний*.

Структурная схема персональной ЭВМ



Память компьютера

Память персонального компьютера

- Состоит из ROM (read-only-memory) памяти и RAM (random-access-memory) памяти.

- В ROM данные хранятся постоянно и не теряются при отключении питания.
- Служит для поддержки процедур первоначальной загрузки компьютера: при включении питания программы из ROM проверяют устройства и загружают в RAM необходимые данные с системного диска.
- В ROM прошит BIOS (base input/output system) – важнейшие программы по организации ввода/вывода данных.

- RAM – оперативная память ПК. Содержимое RAM теряется при отключении компьютера.

Определения

- **Минимальная структурная единица памяти (ячейка памяти) – 1 байт**
- **Байты в памяти пронумерованы от 0000_{16}. Номер байта называется *адресом (физическим адресом)* соответствующей ячейки памяти. Максимальный адрес для 32-разрядных ЭВМ – 2^{32}-1 = 4 ГБ.**
- **Область памяти – непрерывно расположенная последовательность байтов (ячеек памяти).**
- **Длина области памяти – количество байтов в области памяти. Длина области памяти измеряется в байтах.**
- **Адрес области памяти – адрес первого байта области памяти.**
- **Ячейка памяти может состоять также из 2 или 4 и более байтов. Они называются ячейкой памяти длиной слово (двойное слово, учетверенное слово и т.п.).**

Модели памяти.

Сегментированная модель памяти

Память для программы делится на непрерывные области памяти (сегменты) и программа может обращаться **только** к данным, находящимся в этих сегментах.

Страничная модель памяти

Является надстройкой над сегментной моделью. В этой модели оперативная память рассматривается как совокупность блоков длиной 4 КБ. Применяется при организации виртуальной памяти до 4 Тб

Сегменты.

1. При работе с данными в памяти программист должен разделить необходимую ему память на области - сегменты.
2. Каждый сегмент должен иметь свой тип. Тип сегмента является условным. Тип нужен для того, чтобы по умолчанию использовать данные, расположенные в этом сегменте, определенным образом.
3. Тип сегмента определяет программист при создании сегмента в программе средствами языка программирования. Вся программа обязательно должна быть разделена на сегменты.
4. В программе может быть множество сегментов различных типов. При выполнении команд программы в команде всегда должно быть указано, из какого сегмента берутся данные и в какой сегмент данные помещаются.

Преимущество сегментов

1. Расширение возможных диапазонов адресов
2. Защита программ и данных, находящихся в сегментах за счет задания специальных свойств сегментов
3. Упрощение задания адресов данных в программах

Доступ к данным с использованием сегментов (Сегментная адресация памяти)

Программист размещает определенные данные в сегментах программы самостоятельно исходя из назначения данных. Например, команды программы можно разместить в сегменте команд. МП всегда будет пытаться интерпретировать такие данные как команды.

- Если память разбита на сегменты, то адрес любой ячейки памяти может быть составлен из двух частей:
- *- Адреса начала сегмента*
- *- Адреса ячейки относительно начала сегмента (смещение ячейки памяти)*

Вычисление

Адрес ячейки памяти=Адрес_начала_сегмента+смещение

Преимущество: существенное расширение диапазона адресов ячеек памяти (до какой величины?)

Типы сегментов.

- **Сегмент команд.** Обычно содержит команды программы. Как правило, первая выполняемая команда находится в начале сегмента команд и на нее ОС передает управление для запуска программы.

Если в программе имеется несколько сегментов команд, то в программе должно быть четко указано, из какого сегмента в данный момент должна браться команда для выполнения.

- **Сегмент данных.** Обычно содержит обрабатываемые данные.
- **Сегмент стека.** Обычно это один сегмент в программе. Служит для хранения данных, сохранения различных значений и последующего восстановления этих данных и т.п.

Особенность: существует правило автоматической записи и извлечения данных в стек (из стека) специальными командами. В этом случае стек работает по правилу LIFO (Last Input-First Output)- последний пришел – первый ушел. Это обеспечивает упрощение правильной последовательности сохранения/восстановления данных.

Дополнительный сегмент. Вместе с сегментом данных служит для хранения обрабатываемых данных.

Регистры микропроцессора

Регистры – это структурные элементы МП, в которых внутри МП хранятся данные.

16 пользовательских регистров

16 системных регистров

Общая характеристика регистров

- 1. Каждый регистр имеет имя, по которому к нему можно обратиться в программе (записать данные в регистр или извлечь данные из регистра)

- 2. Регистры имеют длину 16 или 32 бита. Биты регистра нумеруются справа налево.

31 30……..15 14 13 ……..3 2 1 0

Нулевой бит является самым младшим, 31 – самым старшим.

Пользовательские регистры

8 регистров общего назначения.

Имена: eax,ebx,ecx,edx,esi,edi,esp,ebp

Длина – 32 бита.

6 регистров сегментов.

Имена: cs, ds, ss, es,fs,gs

Длина – 16 бит.

Регистр указателя команд. Имя: eip

Длина – 32 бита.

Регистр флагов. Имя: eflags

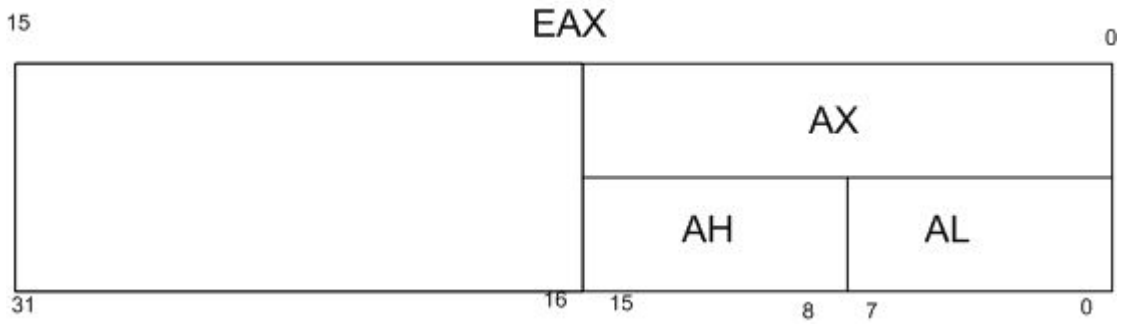
Длина 32 бита.

Регистры общего назначения.

Особенность:
1. Позволяют в программе обращаться к своим младшим частям. Однако к битам 16-31 нельзя обратиться по имени

2. Данные, помещенные в байтовый регистр (напр, AL), автоматически попадают в полный регистр на соответствующее место.

3. Данные, помещенные в полный регистр, могут быть извлечены оттуда по имени соответствующего байтового регистра.



Назначение регистров

- eax/ax/ah/al – аккумулятор. Применяется для хранения промежуточных данных, а также в арифметических операциях, операциях ввода/вывода, операциях обмена данными.

- ebx/bx/bh/bl – базовый регистр. Применяется для операций обмена данными, арифметических операций. Часто применяется для формирования в командах адресов ячеек памяти.

- ecx/cx/ch/cl – регистр счетчика. Применяется для операций обмена данными, арифметических операций. Часто применяется для управления повтором операций (например, управления числа повторений циклов)

- edx/dx/dh/dl – регистр данных. Применяется для операций обмена данными, арифметических операций, операций ввода/вывода.

-

Регистры указателей и индексов

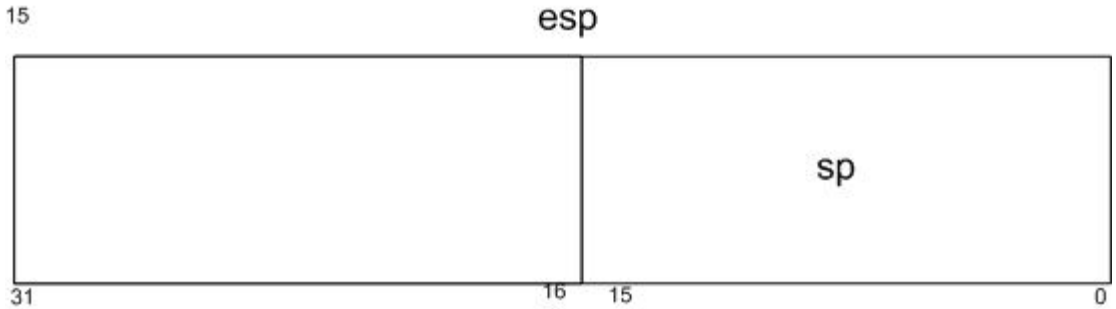
- Имена

esp – указатель стека

ebp – указатель базы

esi – индекс источника

edi– индекс приемника



У этих регистров доступны по имени либо целиком 32-битный регистр, либо его младшее слово.

Назначение регистров

- esp/sp – указатель стека. По умолчанию используется для хранения смещения доступной в текущий момент ячейки памяти в сегменте стека.

Эта ячейка памяти называется **вершиной стека.**

SS:ESP – указатель на вершину стека.

Команды, работающие со стеком (PUSH, POP, PUSHF и др) автоматически помещают данные (извлекают данные) в вершину стека (с вершины стека).

При создании сегмента стека в программе ESP содержит число байтов, отведенных под стек.

Операционная система автоматически меняет содержимое регистра ESP при работе со стеком: при добавлении слова в стек ESP уменьшается на 2 (ESP=ESP-2), при извлечении слова из стека Esp увеличивается на 2 (ESP=ESP+2).

ESP/SP редко используется для других целей.

ebp/bp – указатель базы. Если в команде явно не указано другое, то EBP также указывает смещение ячейки памяти в сегменте стека. В отличие от ESP операционная система не меняет EBP при операциях со стеком. В программах EBP используется для доступа к данным в сегменте стека (по умолчанию), либо для других целей.

esi/si – индекс источника. В командах используется для хранения смещения данных в различных сегментах. В командах обработки строк по умолчанию используется для извлечения данных из сегмента данных. Адрес ячейки источника DS:ESI.

edi/di – индекс приемника. В командах используется хранения смещения данных в различных сегментах. В командах обработки строк по умолчанию используется для помещения данных в дополнительный сегмент. Адрес ячейки-приемника – ES:EDI

Сегментные регистры.

- Применяются для хранения селекторов сегментов в программах и формирования адресов ячеек памяти.
- Имена:
- **CS- регистр сегмента команд**
- **DS – регистр сегмента данных**
- **SS – регистр сегмента стека**
- **ES, GS, FS – регистры дополнительного сегмента.**

Селектор сегмента

- Селектор сегмента – это значение в сегментном регистре, по которому микропроцессор определяет адрес начала соответствующего сегмента.

Назначение сегментных регистров.

Поскольку в программе может быть несколько сегментов одного типа, то в программе должен быть определен сегмент каждого типа, который в данный момент используется программой. **Такой сегмент называется текущим.**

Таким образом, в каждый момент в программе должны быть определены:

- Текущий сегмент команд
- Текущий сегмент данных
- Текущий сегмент стека
- Текущий дополнительный сегмент.

Сегментные регистры хранят селекторы текущих сегментов.

Для того, чтобы сделать некоторый сегмент текущим необходимо селектор этого сегмента занести в соответствующий сегментный регистр.

Обозначение адресов ячеек памяти в программе

Адрес_ячейки_памяти::=

[Селектор_сегмента:]смещение_ячейки внутри сегмента

Селектор_сегмента:смещение_ячейки внутри сегмента – указатель на ячейку памяти

Назначение регистров

- CS – содержит селектор текущего сегмента команд. Команда, которая должна быть выполнена всегда выбирается из памяти по адресу CS: смещение_команды_внутри_сегмента команд. Обычно это смещение находится в указателе команд EIP. Таким образом, текущая команда выбирается МП по адресу CS:EIP.
- Меняя содержимое CS можно изменять сегмент, из которого осуществляется выборка очередной команды программы.

- DS – содержит селектор текущего сегмента данных. Смещение ячейки памяти обычно указывается в программе
- DS:смещение – указатель на ячейку памяти в сегменте данных.

Если в программе при операциях с ячейками памяти селектор сегмента не указан, то по умолчанию считается, что ячейка памяти находится в текущем сегменте данных и селектор этого сегмента находится в регистре DS

SS – содержит селектор текущего сегмента стека.

ES, GS, FS – содержат селекторы дополнительных сегментов данных.

Если в программе по правилам записи команды используются данные из сегмента некоторого типа, а селектор этого сегмента не указан, то всегда считается, что селектор находится в сегментном регистре соответствующего типа.

Регистр адреса команд

Другие названия – указатель команд, регистр командного указателя.

Имя –EIP/IP.

Содержит смещение очередной исполняемой команды из текущего сегмента команд.

Указатель очередной команды – CS:EIP.

Изменением регистров CS и EIP/IP обеспечиваются переходы в программе:

При изменении только EIP/IP осуществляется переход внутри текущего сегмента команд

При изменении CS:EIP осуществляется переход в другой сегмент команд (межсегментный переход)

Регистр флагов (регистр состояния процессора)

Биты регистра показывают некоторые состояния микропроцессора.

Условное имя EFLAGS/FLAGS.

К этому регистру нельзя обратиться по имени.

Условные имена имеют также биты регистра, но обращаться к ним по имени нельзя. Существуют команды, которые меняют содержимое битов регистра флагов, другие команды могут проверить эти биты.

Структура регистра флагов.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NT	IOPL		OF	DF	IF	TF	SF	ZF		AF		PF		CF

CF – флаг переноса. 1 – если был перенос в старший разряд или заем из старшего разряда при выплнении арифметических команд. Используется и для других целей.

PF – флаг четности. 1 – если число единичных битов результата выполнения команды четно.

AF – дополнительный флаг переноса.

ZF – флаг нуля. 1 – если результат выполнения команды равен 0.

SF – флаг знака. 1 – если результат выполнения команды меньше 0

TF – флаг трассировки. Если TF=1, то микропроцессор выполняет трассировку команд, показывая ее результат на экране.

IF – флаг прерываний. Если IF=1, то в микропроцессор может обрабатывать прерывания. Если IF=0, то прерывания запрещены.

DF – флаг направления. Используется, в основном, при обработке строк. Если DF=1, то строки обрабатываются “справа на лево” т.е. от старших адресов к младшим.

OF – флаг переполнения. 1 – если результат выполнения команды превышает максимально допустимое значение для МП, напр, при делении на 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													AC	VM	RF

IOPL (Input-Output Privilege Level) (2 бита) - действует в защищенном режиме и

показывает минимальный уровень привилегий, при котором разрешается выполнение ввода-вывода;

NT (Nested Task) - флажок вложенной задачи, действует только в защищенном

режиме, нужен для переключения задач в многозадачном режиме;

RF (Restore Flag) - флажок возобновления, используется в средствах отладки;

VM (Virtual Mode) - флажок виртуального режима 8086. Если VM=0, то процессор

находится в R- или P- режимах; если VM=1, то процессор работает в V-режиме;

AC (Alignment Check) - флажок контроля выравнивания данных.

Режимы работы МП. Адресация памяти в различных режимах.

- 1. Реальный режим
- 2. Защищенный режим

3. Режим виртуального 8086.

Реальный режим

- 1. Используется для i8086/8088 (IBM PC/XT/AT)
- 2. Эмулируется для всех современных ЭВМ на платформе Intel
- 3. В программе можно получить доступ только к 1 МГб памяти (через специальные драйверы можно работать и с памятью более 1 МГб – для i80286)
- 4. Максимальная длина одновременно обрабатываемых данных – 16 бит (1 слово) (т.е. архитектура шестнадцатиразрядная)

При включении компьютера МП всегда находится в реальном режиме. В реальном режиме доступны все 16 пользовательских регистров.

Недостатки реального режима

1. Малый размер адресуемой памяти
2. Вся память доступна прикладной программе (а не только память, выделенная под программу).

Проблемы адресации памяти в реальном режиме (РР)

- Адреса ячеек памяти в РР имеют значения от 0 до 1048576. В то же время шестнадцатиразрядная архитектура реального режима позволяет обрабатывать значения не больше 65535.

Сегменты в реальном режиме.

- Параграф – область памяти длиной 16 байт.
- Сегмент в реальном режиме – область памяти, начинающаяся на границе параграфа и длиной не более 64 К (65536 б).
- Таким образом, длина сегмента кратна 16, сл-но шестнадцатеричный адрес сегмента всегда заканчивается 0.

Пример

- Пусть сегмент начинается по адресу 045F0h. Ячейка памяти по смещению 00032h имеет физический адрес
  - 045F0h

    +00032h

    04622h

Расширение адресации с помощью сегментирования

Прямое сложения адреса начала сегмента и смещения в 16-разрядной архитектуре не позволяет получать значения большие, чем 128 К.

Для адресации 1 Мгб основной памяти (1024К) в МП используется следующий прием.

Поскольку адрес начала сегмента всегда заканчивается 0, то в программах (соответственно, в микропроцессоре) адреса начала сегментов при создании сегментов хранятся без этого нуля. В таком виде адрес начала сегмента называется *сегментным адресом ячейки памяти. Максимальный размер сегментного адреса – 1 слово т.е. 64 К (65535).*

Адрес ячейки памяти в форме

Адрес ячейки памяти=Сегментный адрес:смещение

называется **указателем ячейки памяти**

Вычисление адреса в реальном режиме

- Для вычислении физического адреса ячейки памяти необходимо указать сегментный адрес ячейки памяти и смещение ячейки памяти внутри сегмента.
- МП автоматически умножает сегментный адрес на 16 (т.е. добавляет к сегментному адресу 0. В такой форме адрес начала сегмента имеет длину 20 бит и максимальный адрес сегмента теперь может иметь значение FFFF0h или 1048560. Таким образом получаем адресацию 1 Мгб памяти.

Формирование структур данных реального режима

1. В программе определяем необходимый сегмент данных и заносим его сегментный адрес в нужный сегментный регистр

Вычисление физического адреса в реальном режиме

1. В сегментных регистрах хранятся сегментные адреса текущих сегментов.

   Селектор сегмента::=сегментный адрес

2. В МП сегментный адрес сдвигается на 4 бита влево (т.е. умножается на 10 шест.) для получения адреса начала сегмента
3. Смещение ячейки памяти задается в команде программы.
4. Сумматор адресов МП суммирует адрес начала сегмента и смещение для получения физического адреса ячейки памяти.

Схема вычисления физического адреса ячейки памяти в реальном режиме

Mov bx, [edx]

занести в регистр bx **содержимое ячейки памяти длиной слово** и находящейся в **текущем сегменте** данных по **смещению**, значение которого находится **в регистре edx**

Схема вычисления физического адреса в реальном режиме

Защищенный режим работы МП.

- Программы, разработанные для i8086 (реального режима), не могут функционировать в защищенном режиме.
- После включения или сброса МП работает в реальном режиме. Для того чтобы МП начал функционировать в защищенном режиме, необходимо перевести МП в защищенный режим. При этом необходимо сформировать определенные структуры данных и настроить определенным образом системные регистры.

Позволяет:

1) Адресовать до 4 Гб оперативной памяти. В защищенном режиме поддерживается несколько моделей памяти: плоская (Flat), многосегментная и страничная, тем самым обеспечивается возможность работы с виртуальной памятью, значительно превосходящей имеющуюся физическую память.
2) Основное достоинство защищенного режима – поддержка многозадачности и режима виртуального i8086. Этот режим позволяет работать параллельно нескольким программам, разработанным для i8086. Многозадачность поддерживается МП на аппаратном уровне с использованием специальных системных регистров. Каждая задача занимает свою область памяти и запрещает обращаться по адресам, не предназначенным задаче.

Многозадачные ОС (Windows, Unix и др.) используют защищенный режим работы МП

Регистры защищенного режима работы (системные регистры).

4 регистра управления

4 регистра системных адресов

8 регистров отладки

Регистры управления

- Имена регистров: CR0, CR1, CR2, CR3.
- Длина – 32 бита.
- Эти регистры предназначены для общего управления системой.
- CR0 содержит системные флаги, управляющие режимами работы микропроцессора и отражающие его состояние глобально, независимо от конкретных выполняющихся задач. (***PE- флаг защищенного режима)***
- CR1 – не используется.
- *CR2 используется при страничной организации оперативной памяти*
- *CR3 также используется при страничной организации памяти.*

Регистры системных адресов.

- Эти регистры еще называют регистрами управления памятью. Они предназначены для доступа к программам и данным в защищенном режиме.

GDTR (48 бит) – регистр таблицы глобальных дескрипторов GDT.

LDTR (16 бит) – регистр таблицы LDT локальных дескрипторов

IDTR (48 бит) – регистр таблицы дескрипторов прерываний IDT

TS (16 бит) – регистр задач TSS

Регистры отладки

Предназначены для отладки программ в защищенном режиме.

DR0 – DR7

Позволяют создавать точки останова в программах и осуществлять трассировку программы.

Сегментная адресация в защищенном режиме

1. Сегмент определяется:
   - адресом начала сегмента
   - размером сегмента (неявно указан в программе в RM)
   - свойствами сегмента (отсутствует в RM)

В отличии от реального режима **эти параметры** для каждого сегмента **должны** быть явно указаны в программе в **дескрипторах сегментов**

Где хранятся дескрипторы сегментов

Таблица GDT – хранит общие для всех задач дескрипторы сегментов (например, дескрипторы сегментов многозадачной операционной системы)

Таблицы LDT – хранит дескрипторы сегментов конкретных задач (для каждой задачи – своя LDT)

Таблица GDT (global descriptor table)

- Таблица, которая содержит описание объектов, общих для всех задач в системе.
- Такие описания называются ***дескрипторами***.
- Каждая строка таблицы GDT содержит описание одного объекта (один дескриптор)
- Максимальное количество дескрипторов в GDT – 8192 (2^{13})
- Адрес начала таблицы должен находиться в регистре GDTR

Типы объектов (дескрипторы) в таблице GDT

1. **Дескрипторы сегментов**
2. *Дескрипторы таблиц LDT (таблиц дескрипторов задач)*
3. *Дескрипторы сегментов состояния задач*
4. *Дескрипторы шлюзов вызовов задач*
5. *Дескрипторы шлюзов задач*

Сегментные регистры

Содержат селекторы сегментов, но:

Реальный режим (Real Mode - RM)	**Защищенный режим (Protected Mode – PM)**
Селектор ::=сегментный адрес	Селектор содержит: • номер дескриптора (номер строки), отсчитанный от 0, в дескрипторной таблице GDT или LDT (биты 3-15 сегментного регистра) •Индикатор используемой таблицы (0 – GDT, 1 – LDT) •Уровень привилегий сегмента RPL

Схема вычисления физического адреса ячейки памяти в защищенном режиме (однозадачный режим работы)

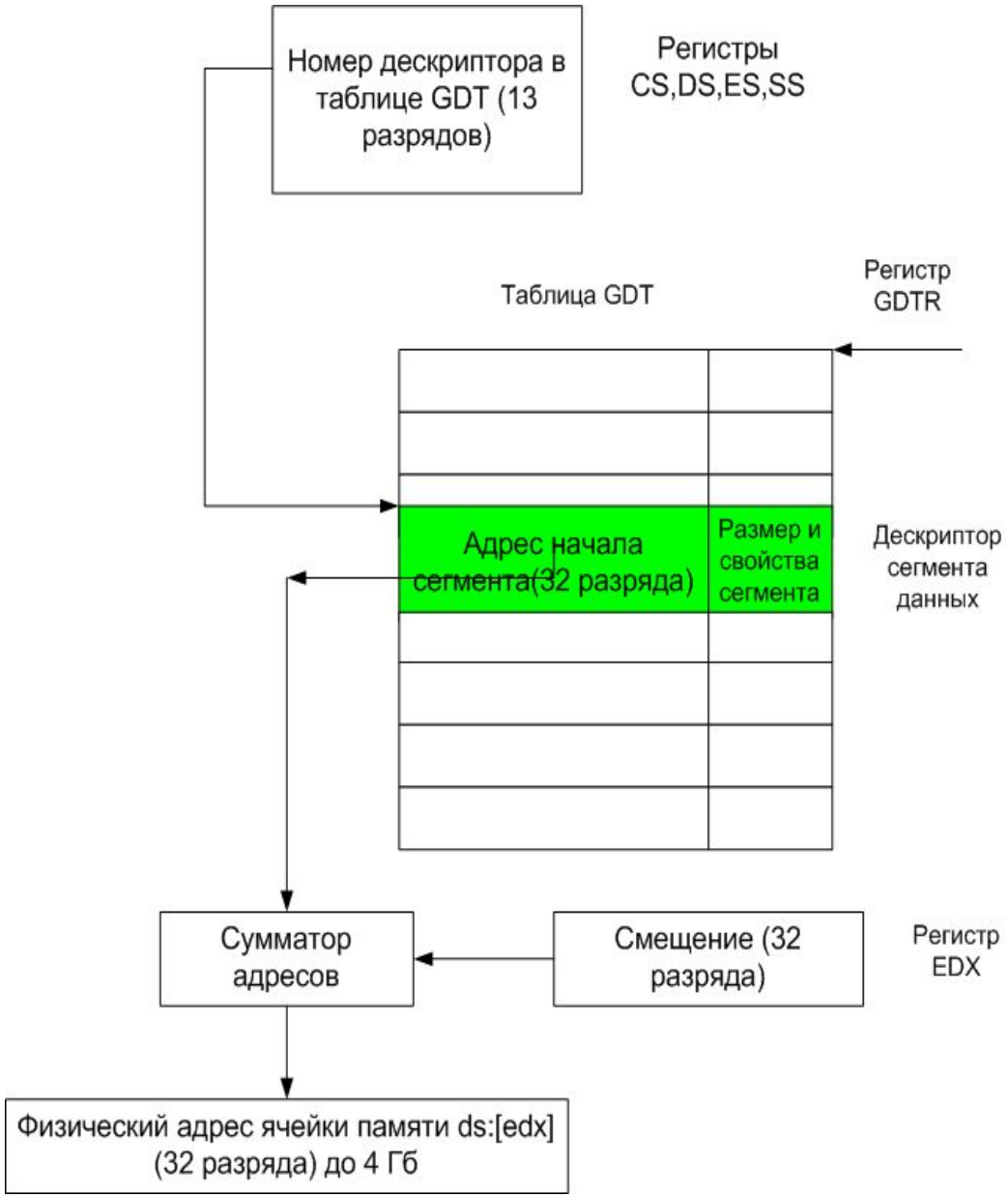
Mov bx, [edx]

занести в регистр bx **содержимое ячейки памяти длиной слово** и находящейся в **текущем сегменте** данных по **смещению**, значение которого находится **в регистре edx**

Формирование структур данных защищенного режима

1. В программе отводим место под таблицу GDT и определяем в ней строки, которые будут содержать необходимые дескрипторы, в том числе дескриптор сегмента данных
2. В программе определяем необходимый сегмент данных и заносим адрес его начала, размер сегмента и его свойства в нужную строку (дескриптор) таблицы GDT
3. С помощью специальной команды заносим в регистр GDTR адрес начала таблицы GDT и размер таблицы GDT.

Как это работает



Хранение слов в памяти компьютера

- 1. Слов имеет длину 2 байта (16 бит).
- Старший_байт Младший байт.
- 2. Линейная модель памяти:
- Байт0_Байт1_Байт2……БайтN
- 3. Слова (двойные слова) в ОП хранятся в обратной последовательности байтов: младший байт (младшее слово) по младшему адресу.

  Младший_байт Старший_байт.
- 4. Адрес ячейки памяти = адресу первого (младшего) байта этого слова.
- 5. При считывании слова в МП (например в регистр), байты в слове (слова в двойных словах) хранятся в прямой последовательности байтов.

Команды МП

- Микропроцессор управляется с помощью команд МП. Совокупность команд образуют систему команд МП.
- Каждая команда имеет числовой код. Также в команде могут быть указаны данные, которые должна обработать команда: содержимое регистров, адреса ячеек памяти, содержимое ячеек памяти, константы.

Форматы допустимых данных МП Intel

Структура регистров и разрядность шины данных определяют формат допустимых данных МП. В командах МП могут использоваться только эти типы данных.

Допустимы следующие 6 типов данных.

1. Целое без знака длиной 1 байт (8 битов). Диапазон 0 – 255 (FFh)
2. Целое со знаком длиной байт. Диапазон (-128) до (127) (80h – 7Fh)

3. Целое без знака длиной слово (16 бит).

Диапазон 0 – 65535 (FFFFh)

4. Целое со знаком длиной слово. Диапазон (-32768) до (32767) (8000h – 7FFFh).

5. Целое без знака длиной двойное слово(32 бита).

Диапазон 0 – 2^{32}-1 (FFFFFFFFh)

6. Целое со знаком длиной двойное слово(32 бита).

Диапазон -2^{31} до 2^{31}-1 (80000000h) до 7FFFFFFFh

Обработка символьных данных

Для обработки символьных данных в программе их необходимо хранить и обрабатывать в стандартном ASCII – коде. В этом коде под 1 символ отводится 1 байт. ASCII – коды символов – целые числа без знака.

Важные ASCII - коды

‘0’ – 30h (00110000)

‘1’ – 31h (00110001)

‘2’ – 32h (00110010)

……

‘9’ – 39h (00111001)

- Обработка этих чисел аппаратурой компьютера (напр., видеоадаптером) приводит к появлению на экране символов чисел.