2024

Машинно-зависимые языки и основы компиляции

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

Структура дисциплины

Лекции (34 часа):

- структура процессора IA-32 и система машинных команд,
- язык ассемблера NASM,
- связь разноязыковых модулей,
- основы построения компиляторов,
- макросредства ассемблера

Семинары (7 занятий): подготовка к лабораторным работам.

Лабораторные работы: 4 занятия по 4 часа – 5 лабораторных работ.

Домашние задания: 1-е – 8 неделя, 2-е – 15 неделя.

Контроль знаний:

Рк1: Структура машинной команды (4 неделя) – 6..10 баллов.
КР2: Ветвления и циклы (10 неделя) – 18..30 баллов.
Рк3: Правила передачи параметров (13 неделя) – 3..5 баллов.
КР4: Основы конструирования компиляторов (16 нед.) – 15..25 баллов.
Экзамен - 18..30 баллов.

Цели и задачи дисциплины

Цель дисциплины:

изучение программирования на языках низкого уровня и основ конструирования компиляторов

Задачи дисциплины:

- знакомство с программной моделью и системой машинных команд процессора семейства IA-32;
- освоение основ программирования на ассемблере Nasm;
- изучение основ связи разноязыковых модулей;
- изучение основ конструирования компиляторов;
- изучение макросредств ассемблера.

Литература не покрывает курса!

Основная литература

1. Г.С. Иванова, Т.Н. Ничушкина. Главы 1-4. Учебные пособия в эл. виде.

Дополнительно:

1. Костюк Д.А., Четверкина Г.А. Программирование на ассемблере в GNU/Linux: методическое пособие. Брест: изд-во БрГТУ, 2013. 68 с. Способ доступа: https://www.bstu.by/uploads/attachments/metodichki/kafedri/EVMiS_Assembler_v_GNU-Linux.pdf.
2. Столяров А.В. Программирование на языке ассемблера NASM для ОС Unix: Уч. Пособие. - М.: МАКС Пресс, 2011. - 188 с. Способ доступа: http://www.stolyarov.info/books/pdf/nasm_unix.pdf.
3. Ирвин К. Язык ассемблера для процессоров Intel. – М.: ИД «Вильямс», 2005.
4. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. – М.: Мир, 1975.
5. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции.
6. Аблязов Р. Программирование на ассемблере на платформе x86-64. – М.: ДМК Пресс, 2011.
7. Зубков С.В. Assembler. Для Dos, Windows и Unix. – М.: ДМК Пресс, 2015.

2024

Глава 1 *Организация ядра ЭВМ на базе IA-32*

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

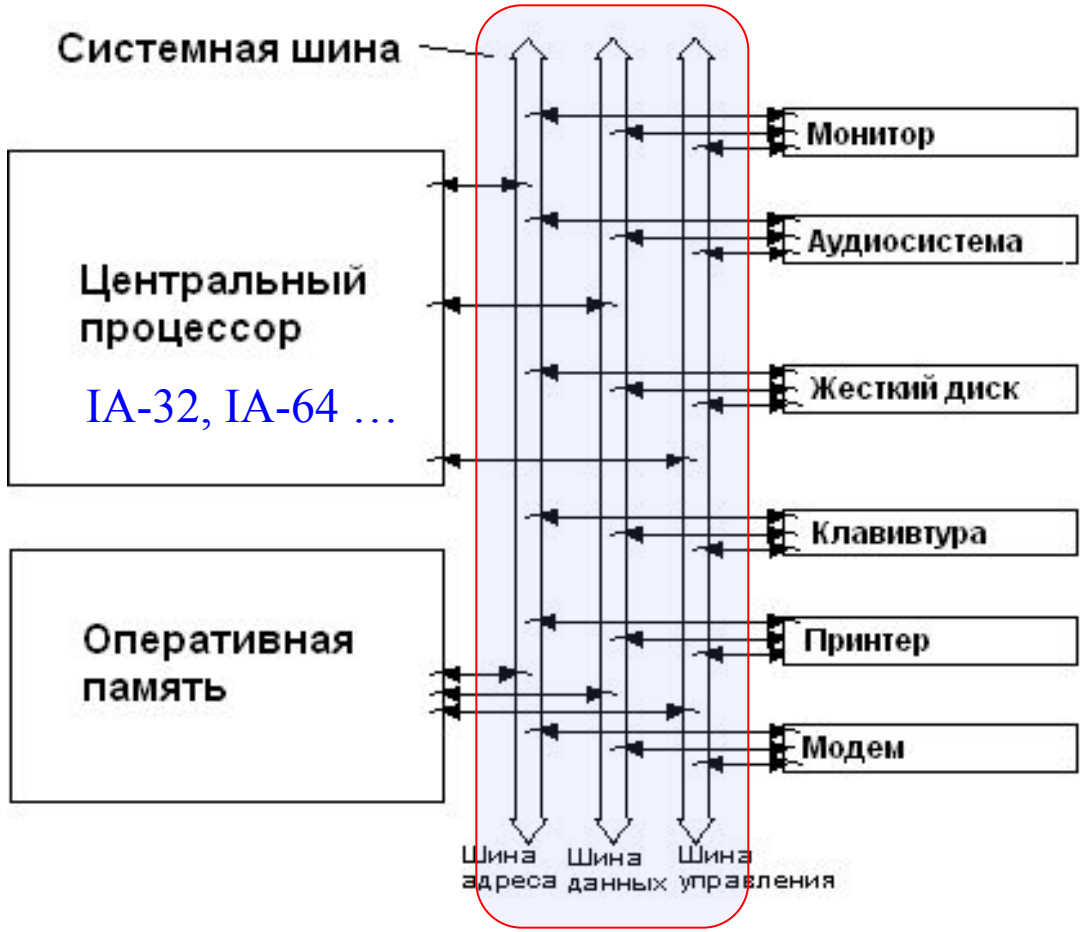
1.1 *Архитектура вычислительной системы на базе процессоров IA-32*

Архитектурой ВС называют совокупность основных характеристик системы, определяющих особенности ее функционирования.

Архитектура «с общей шиной» предполагает, что основные устройства ВС взаимодействуют через единственную шину, называемую *системной,* которая включает*:*

- шину адреса;
- шину данных;
- шину управления.

Шина – соединение компьютерных устройств. Различают механический, электрический и логический уровни.



Семейство процессоров IA-32

Модель (Intel)	РОНы/ Данные/ Адрес	Основные характеристики
I8086 (1978) - I80186	16/16/20	4-8 МГц, 1 Мб
I80286 (1982)	16/16/24	8-20 МГц,защищенный режим
I80386 (1985)	32/32/32	20-40 МГц, виртуальный режим
I80486 (1989)	32/32/32	20-100 МГц, внутренний сопроцессор
Pentium (1993)	32/64/32	60-150 МГц, конвейерные и суперскалярные схемы
Pentium Pro (1995)	32/64/32	100-200 МГц
Pentium II (1997)	32/64/32	233-300 МГц
Pentium III (1999)	32/64/36	450-500 МГц
Pentium IV (2001)	32/64/36	2.8 ГГц
Pentium IV 3,2 (2003)	32/64/36	3.2 ГГц
Pentium Core 2 Extreme QX9775 (2008)	32/64/36	3.2 (до 3.73) ГГц, двухядерные, частота системной шины до 1.6 ГГц
Pentium 4 670		3.8 ГГц, одноядерный
Pentium Core i7 970 Extreme		3.33 (до 3.6) ГГц, шестиядерный, частота системной шины 6.400 ГГц

64 разрядные процессоры

Различали 2 совершенно разные, несовместимые друг с другом, микропроцессорные архитектуры:

- Intel 64 или x86-64 ;
- IA-64 – это семейства Itanium и Itanium 2, предназначены для серверов и больше не выпускаются.

Процессоры архитектуры x86-64 поддерживают два режима работы:

- *Long mode* — позволяет выполнять 64-битные программы; есть поддержка выполнения 32-битных приложений, но устранены сегментная модель памяти, аппаратная поддержка мультизадачности и т.п.;
- *Legacy mode* («*наследственный*», режим совместимости с x86) — предоставляет полную совместимость с 32/16-битным кодом и операционными системами.

Таблица - Процессоры (февраль 2010)

Процессор	Тактовая частота, ГГц	Частота системной шины	Коэффициент умножения	Объем кэш-памяти второго уровня, Мбайт	Цена при поставке партиями по 1000 штук, долл.
Intel Pentium 4 Extreme Edition 3,73 ГГц	3,73	1066	14	2	999
Intel Pentium 4 660	3,60	800	18	2	605
Intel Pentium 4 650	3,40	800	17	2	401
Intel Pentium 4 640	3,20	800	16	2	273
Intel Pentium 4 630	3,00	800	15	2	224

Модели процессоров (2020 г.)

	Частота, ГГц	Количество ядер/потоков	Мощность, Вт	Стоимость, руб.
Intel Core i9-9900K	3,6	8/16	95	32 000
AMD Ryzen Threadripper 1950X	3,4	16/32	180	36 000
AMD Ryzen 9 7950X OEM	4,5	16/32	170	66 640 (новая цена)

1.2 *Программная модель процессора i8086*

Под ***программной моделью*** процессора понимается совокупность его характеристик, существенных для разработки программного обеспечения.

В общем случае, программная модель процессора включает описание:

- способов адресации памяти,
- форматов данных,
- регистров,
- системы команд.

Регистры – внутренняя, сверхбыстрая, но ограниченная по объему память процессора, используемая для хранения адресов и данных во время выполнения программы.

Оперативная память

Оперативная память ВС организована как *последовательность байтов*, которым соответствуют номера – целые числа от 0: 0,1,2 и т.д.

Объем оперативной памяти современного компьютера: 4, 8, 16, 32 ГБ.

Номер байта является его ***физическим адресом***.

Для рассматриваемого класса систем приняты следующие обозначения:

- *слово* (word) – 2 байта = 16 битов;
- *двойное слово* (double word - dword) – 2 слова = 32 бита;
- *учетверенное слово* (qword) – 2 двойных слова = 64 бита.

Кроме этого могут использоваться обозначения:

- параграф – 16 байтов;
- страница – 256 (512, 4096) байтов.

Младшие части чисел в оперативной памяти располагаются в младших адресах. В регистрах, как обычно, сначала записывается старшая часть, затем – младшая.

	Младший байт								Старший байт							
Номера битов в байте:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Номера битов в слове:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Пример: число 258 =	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Типы обрабатываемых данных

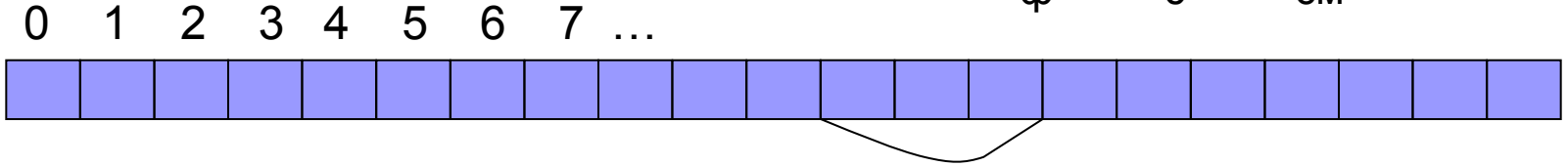
- ***целые числа*** – представляются в двоичной системе счисления, со знаком или без знака, длиной 1, 2, 4 байта, если число – со знаком, то старший бит содержит знак;
- ***вещественные числа*** – представляются в двоичной системе счисления в виде мантиссы со знаком и порядка общей длиной от 4 до 10 байтов;
- ***двоично-кодированные десятичные числа*** со знаком, длиной до 16 байтов – тип больше не используется;
- ***символы*** (кодировки ASCII, ANSI, Unicode и др.), длиной 1 или 2 байта.

Структура процессора i8086

От УУ цепи идут ко всем блокам процессора (на рисунке не показаны).

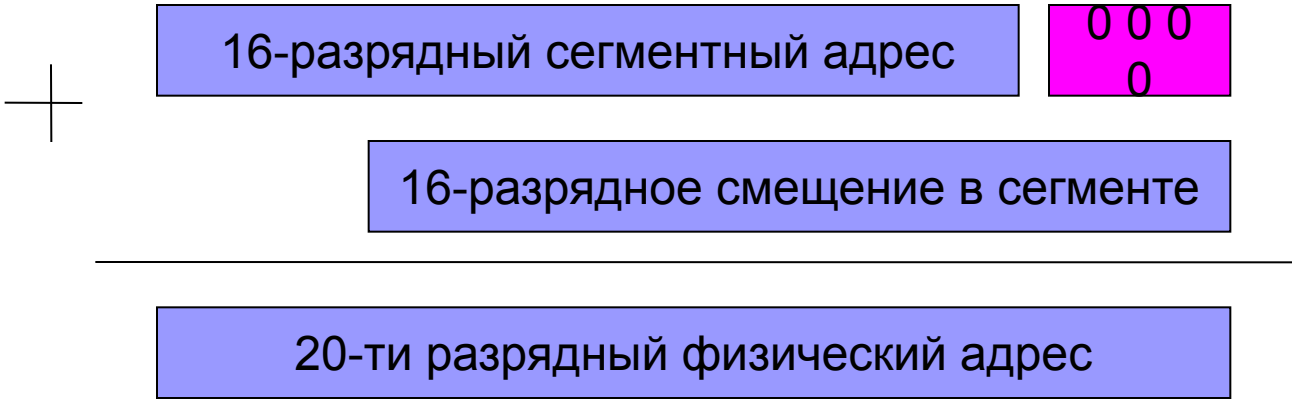
Сегментная модель 16-ти разрядной адресации памяти

Схема адресация «база +смещение»: $A_ф = A_б + A_{см}$



Сегментная схема адресации *микропроцессора i8086*:

Добавляются аппаратно



Сегмент при 16-ти разрядной адресации – фрагмент памяти размером не более 64 кбайт, который начинается с адреса, кратного 16.

Адресация сегментов различных типов

Программа размещается в сегментах трех типов, каждый из которых адресуется одним из сегментных регистров и регистром или регистрами, содержащими смещение.

1. ***Сегмент кода***: **CS: IP**
2. ***Сегмент стека***: **SS:SP**
3. ***Основной и дополнительный сегменты данных:***

Сегментный адрес

Смещение в сегменте

По умолчанию для данных

BX + DI + <Непосредственное смещение>

BX + SI + <Непосредственное смещение>

BP + DI + <Непосредственное смещение>

DS: BP + SI + <Непосредственное смещение>

ES: BX + <Непосредственное смещение>

BP + <Непосредственное смещение>

SI + <Непосредственное смещение>

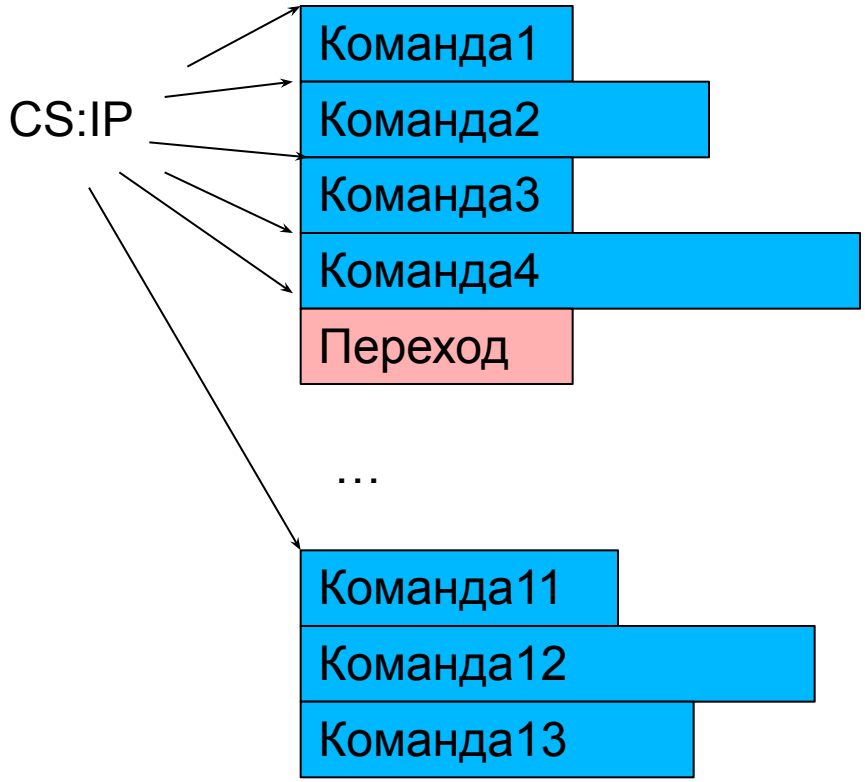
DI + <Непосредственное смещение>

База

Индекс

Выполнение программы

Программа записывается в память единой строкой без разделителей и промежутков. Физический адрес первой команды грузится в регистры CS:IP

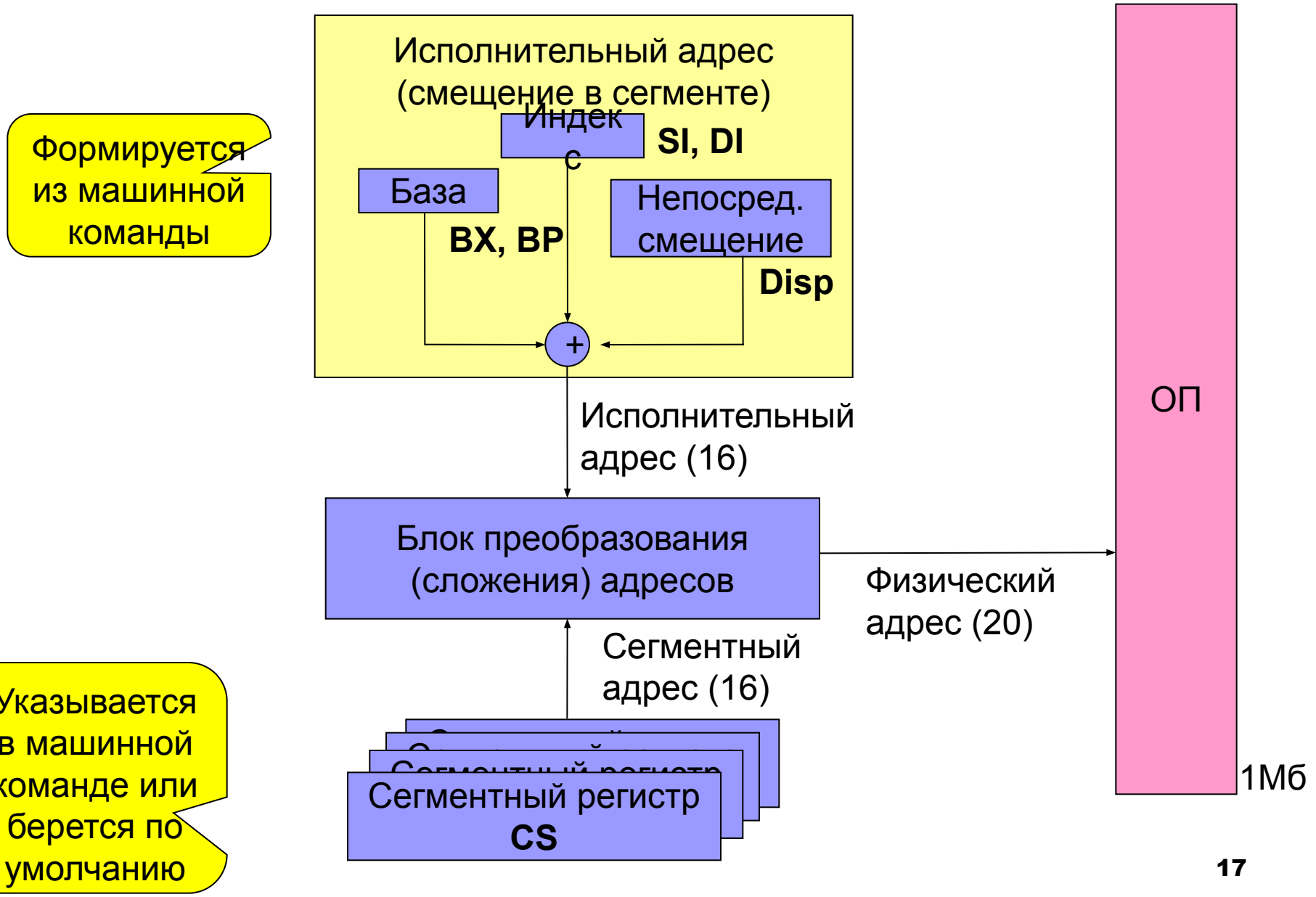


При выполнении команды содержимое IP увеличивается на длину команды

При выполнении команды интрасегментного перехода в регистр IP записывается новое смещение

При выполнении команды перехода в другой сегмент обновляются оба регистра

Схема 16-ти разрядной адресации данных



Система машинных команд i8086. Форматы команды MOV

Префиксы (замены сегмента и повторения)	Код операции	1 байт адресации	2 байта смещения	2 байта данных

Рег-р/память ↔ регистр	**100010**DW	Mod Reg R/M	Смещение млад. байт	Смещение стар. байт	
Литерал → рег-р/память	**1100011**W	Mod **000** R/M	Смещение млад. байт	Смещение стар. байт	Данные

D - 1- в регистр, 0 - из регистра
W - 1- операнды-слова, 0 - байты

Mod - 00 – смещение Disp=0 байт
01 – смещение Disp=1 байт
10 – смещение Disp=2 байта
11 – операнды-регистры

	W=1	W=0
Reg	**000 AX**	**000 AL**
	001 CX	**001 CL**
	010 DX	**010 DL**
	011 BX	**011 BL**
	100 SP	**100 AH**
	101 BP	**101 CH**
	110 SI	**110 DH**
	111 DI	**111 BH**

Sr	
00	**ES**
01	**CS**
10	**SS**
11	**DS**

M =	**000**	**EA=(BX)+(SI)+Disp**
	001	**EA=(BX)+(DI)+Disp**
	010	**EA=(BP)+(SI)+Disp**
	011	**EA=(BP)+(DI)+Disp**
	100	**EA=(SI) + Disp**
	101	**EA=(DI) + Disp**
	110	**EA=(BP) + Disp ***
	111	**EA=(BX) + Disp**

Примеры машинных команд

Команды с регистрами AL, AX имеют особый формат (см. Метод. ук. к лаб.раб.!

Примеры:

1) `mov BX,CX`

8 9	C B

100010DW Mod Reg Reg

10001001 11 001 011

2) `mov CX,[BX+6]`

8 B	4 F	0 6

100010DW Mod Reg Mem См.мл.байт

10001011 01 001 111 00000110

3) `mov BYTE[BX+6],10`

C 6	4 7	0 6	0 A

1100011W Mod 000 Mem См.мл.байт Данные

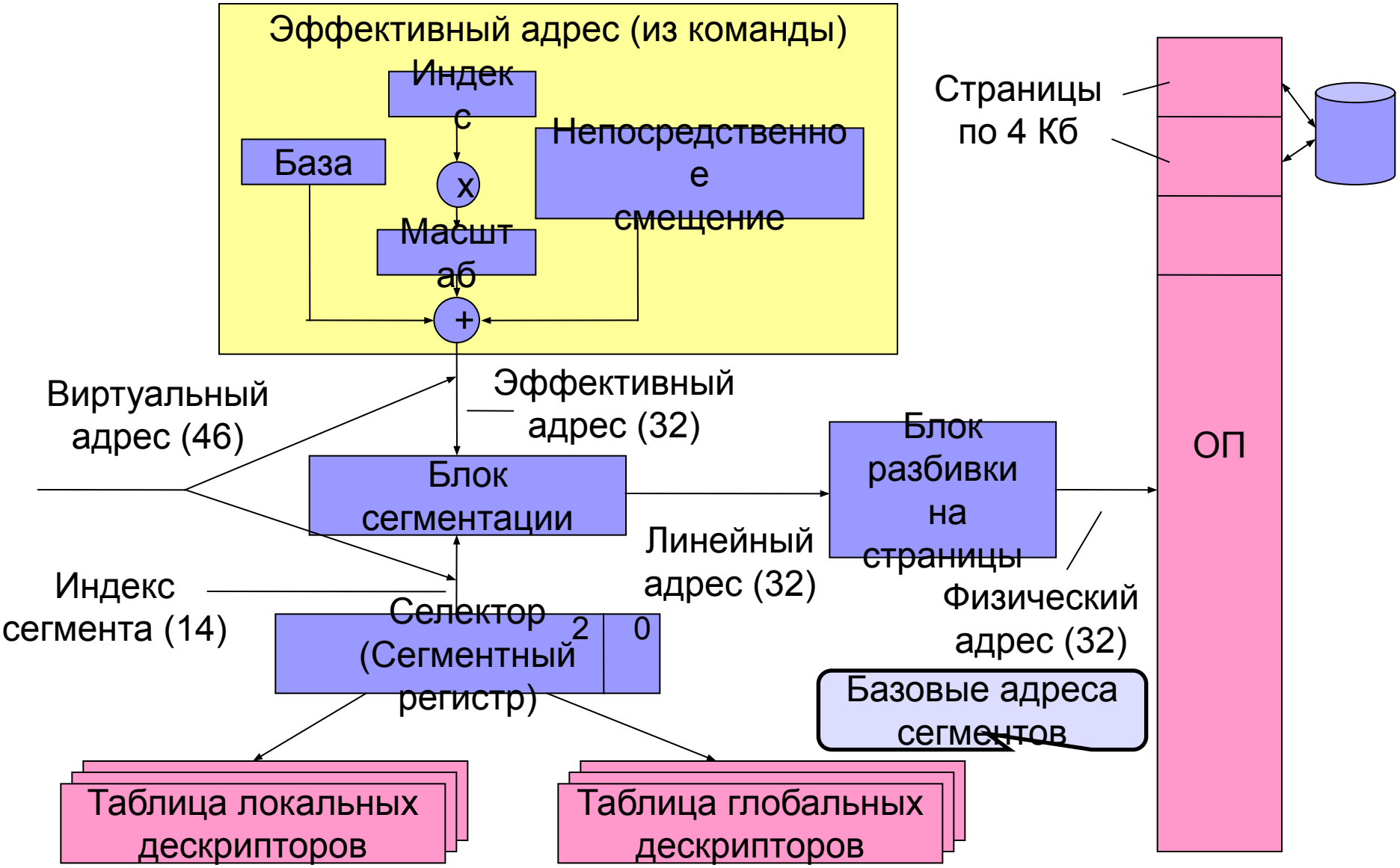
11000110 01 000 111 00000110 00001010

1.3 *Программная модель процессоров IA-32*

Процессоры IA-32 могут функционировать в одном из трех режимов:

- **реальной адресации (Real address mode)** – процессор работает как процессор i8086, адресует только 1 Мб памяти, с использованием 32-х разрядных расширений, например, 32-х разрядных регистров или команд перехода в защищенный режим – используется при начальной загрузке;
- **защищенном (Protected mode)** – процессор работает с 32-х разрядными адресами и при этом использует сегментную и, как правило, страничную модели памяти – обычный режим работы;
- **управления системой (System Management mode)** – для выполнения действий с возможностью их полной изоляции от прикладного программного обеспечения и операционной системы. Переход в режим возможен только аппаратно - используется для выполнения таких операции, как переход в энергосберегающее состояние.

Схема 32-х разрядной адресации данных



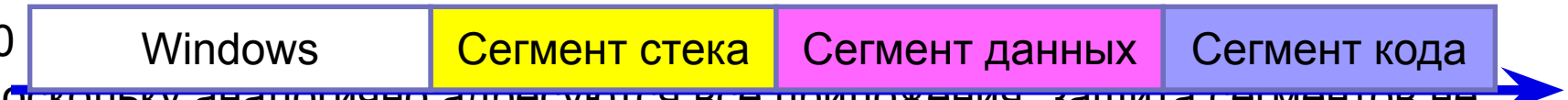
Flat – "плоская" модель памяти

Модель памяти Flat используется в приложениях Windows:

- база = 0;
- граница совпадает с объемом доступной оперативной памяти;
- сегмент кода, сегмент данных и сегмент стека располагаются в одном и том же пространстве адресов, которое разделено между указанными сегментами.

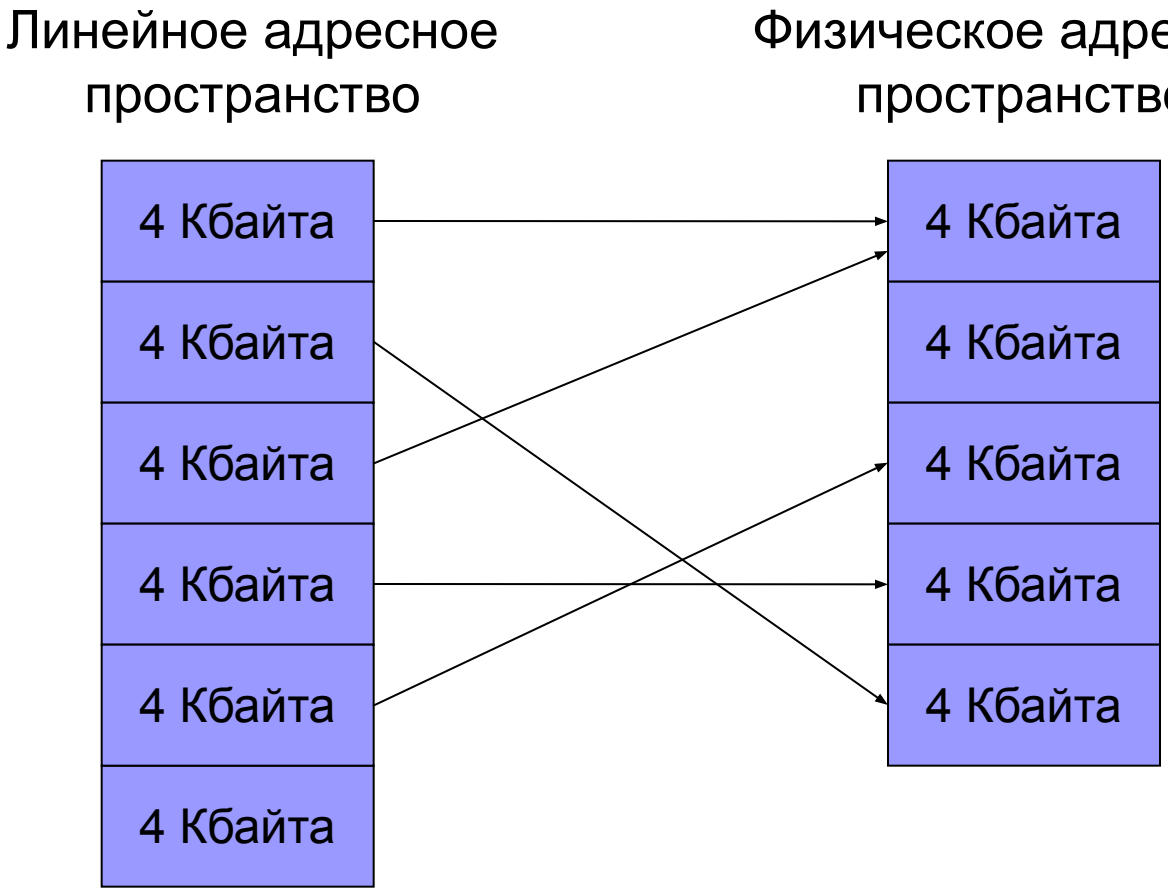
Начальные адреса памяти отводятся для размещения операционной системы. В связи с этим все модули компонуются не с начальных адресов, а с базового адреса в сегменте - 0x400000. в PE формате, несмотря на то, что сам формат позволяет выравнивать секции на 512 байт, используется выравнивание секций на границу 4 кб, меньшее выравнивание в Windows не считается корректным.

Адресное пространство приложения



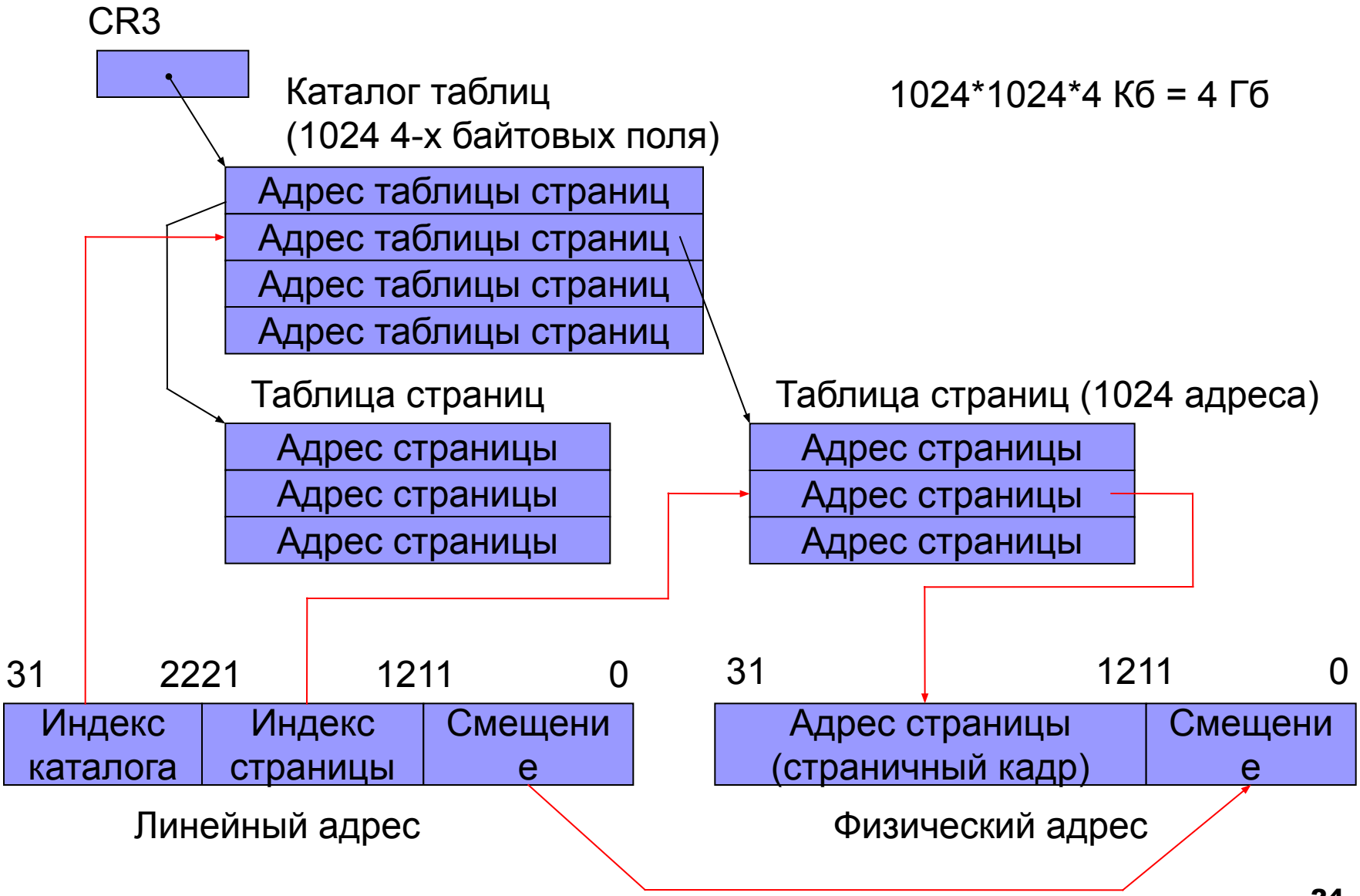
Поскольку аналогично адресуются все приложения, защита сегментов не работает (нельзя писать в сегмент кода, но можно в сегмент данных, расположенный в том же пространстве). Работает только *защита страниц*.

Страничная организация памяти

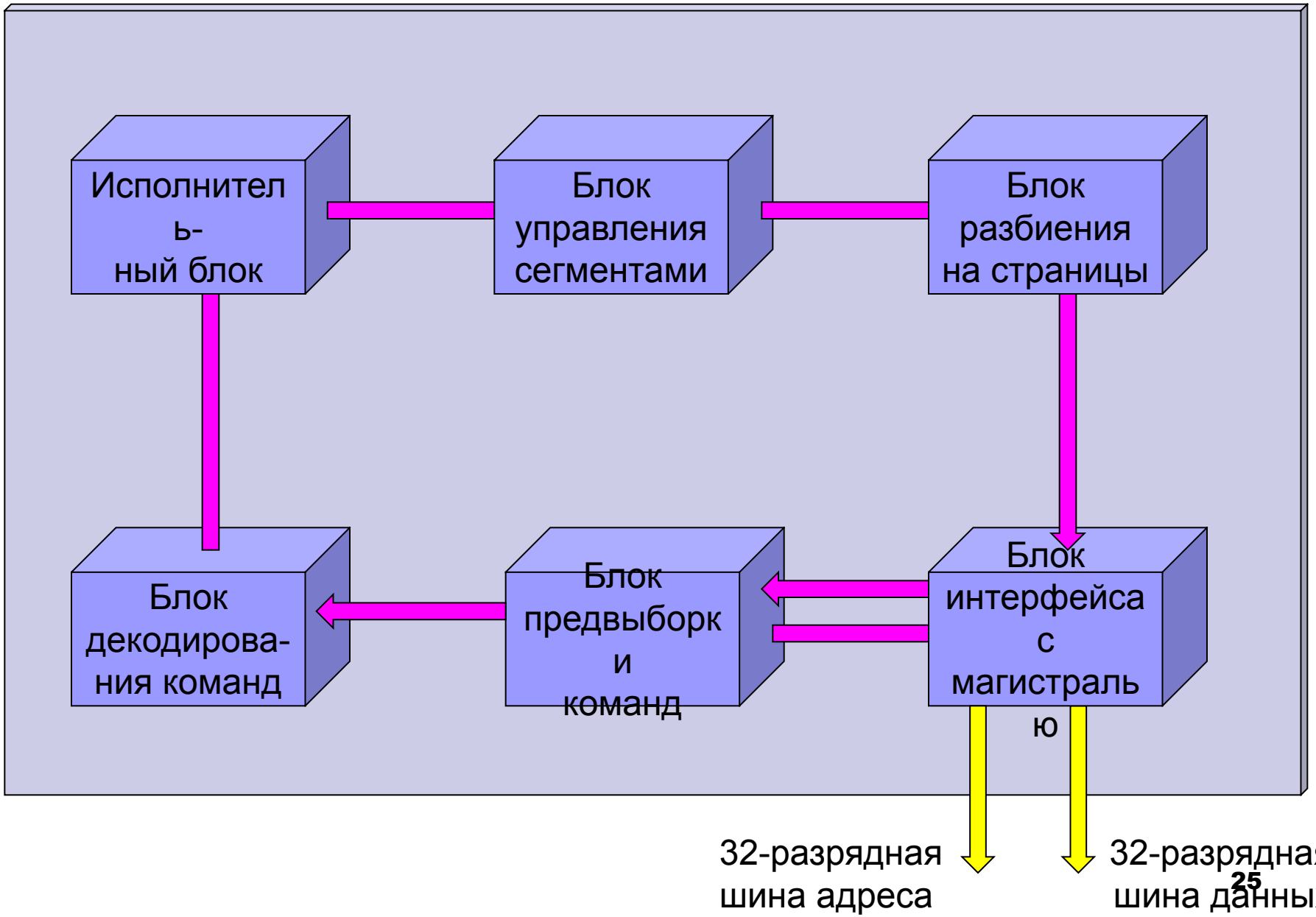


Линейное пространство отображается в физическое пространство посредством специальной таблицы, отдельной для каждого приложения

Схема страничной адресации

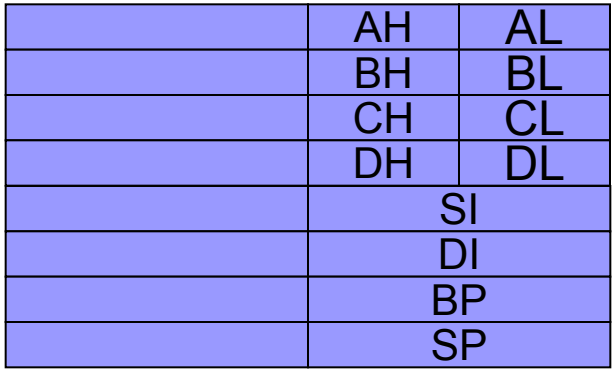


Основные блоки процессора IA-32

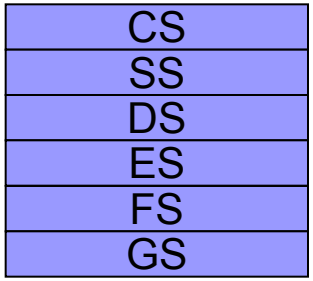


Регистры процессоров семейства IA-32

1. **Регистры данных** (32-х разр.):



2. **Селекторы** (16-ти разр.):



3. **Регистр указатель команд** (32):

4. **Слово системных флагов** (32):

5. **Управляющие регистры**: **CR0..CR3**

6. **Регистры системных адресов**:

GDTR – регистр адреса таблицы глобальных дескрипторов;

LDTR – регистр адреса таблицы локальных дескрипторов;

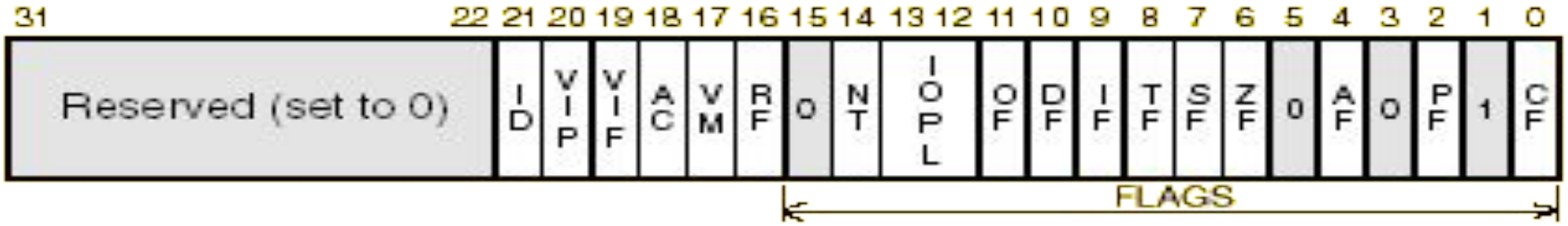
IDTR – регистр адреса таблицы дескрипторов прерываний;

TR – регистр состояния задачи;

7. **Отладочные регистры**

8. **Тестовые регистры**

Системные флаги



Флаги статуса CF, PF, AF, ZF, SF и OF отражают статус выполнения арифметических инструкций (таких как ADD, SUB, MUL, DIV).

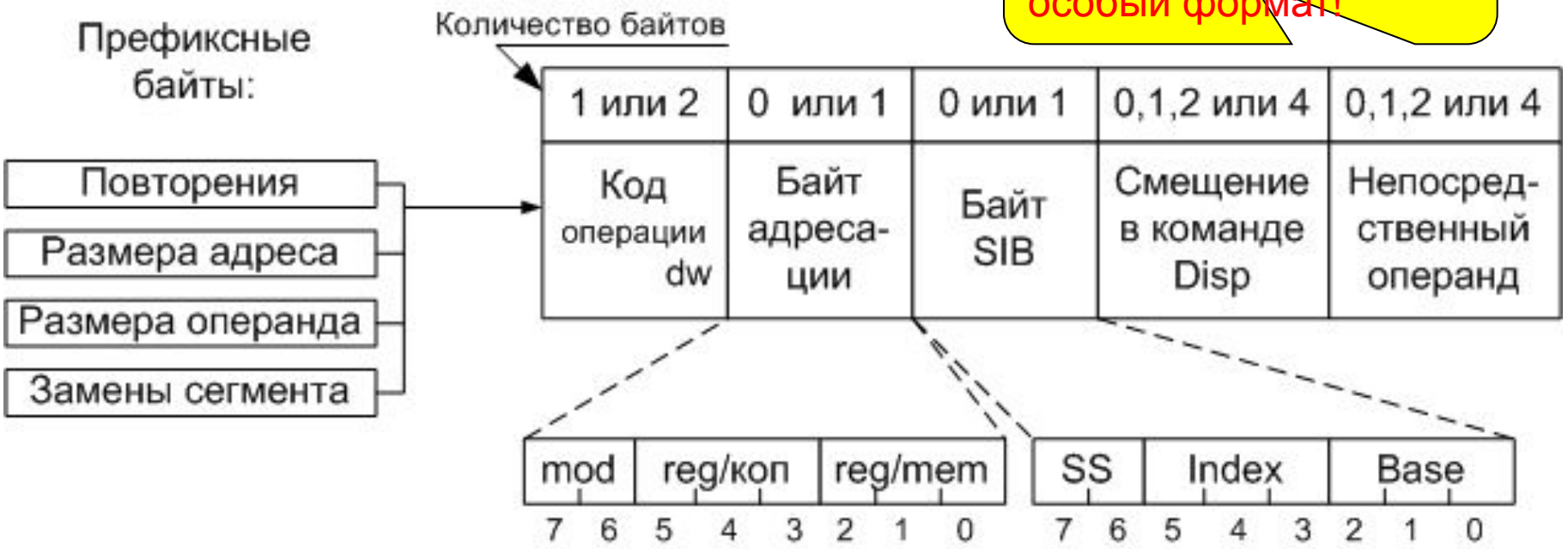
- ***CF*** *– флаг переноса (Carry Flag).*
- ***ZF*** *– флаг нуля (Zero Flag).*
- ***SF*** *– флаг знака (Sign Flag).*
- ***OF*** *– флаг переполнения (Overflow Flag).*
- ***DF*** *– флаг направления (Direction Flag)*

Системные флаги и поле IOPL влияют на процесс исполнения задачи, и поэтому не должны изменяться прикладной программой.

Система команд семейства процессоров IA-32

Размер команды от 1 до 15 байт:

Команды с регистрами AL,AX,EAX имеют особый формат!



d – направление: 1 – в регистр, 0 – из регистра;
w – 1 – операнды – 4 байта, 0 – 1 байт;
mod - 00 - Disp=0 – смещение в команде 0 байт;
01 - Disp=1 – смещение в команде 1 байт;
10 - Disp=2 – смещение в команде 4 байта;
11 - операнды-регистры.

Sib присутствует, если:
- операнд находится в памяти;
- поле m = 100.

Регистровые команды

w=1		w=0	
000	EAX	000	AL
001	ECX	001	CL
010	EDX	010	DL
011	EBX	011	BL
100	ESP	100	AH
101	EBP	101	CH
110	ESI	110	DH
111	EDI	111	BH

Примеры:

1) `mov EBX,ECX`

8 9	C B

100010dw mod reg reg

10001001 11 001 011

2) `mov BX,CX`

6 6	8 9	C B

префикс1 100010dw mod reg reg

01100110 10001001 11 001 011

Определение размера операнда

Размер операнда или адреса определяется на основе режима работы, бита размера операнда D дескриптора используемого сегмента и наличия в инструкции префиксов:

Режимы работы:

	RM, V86, SMM				PM			
Бит размера операнда	0	0	0	0	1	1	1	1
Префикс размера оп. (066h)	нет	нет	есть	есть	нет	нет	есть	есть
Префикс размера адр. (067h)	нет	есть	нет	есть	нет	есть	нет	есть
Размер операнда	16	16	32	32	32	32	16	16
Размер адреса	16	32	16	32	32	16	32	16

Схемы адресации памяти без байта Sib

Поле r/m	Эффективный адрес второго операнда		
	mod = 00B	mod = 01B	mod = 10B
000B	EAX	EAX+Disp8	EAX+Disp32
001B	ECX	ECX+Disp8	ECX+Disp32
010B	EDX	EDX+Disp8	EDX+Disp32
011B	EBX	EBX+Disp8	EBX+Disp32
100B	Определяется Sib	Определяется Sib	Определяется Sib
101B	Disp32	SS:[EBP+Disp8]	SS:[EBP+Disp32]
110B	ESI	ESI+Disp8	ESI+Disp32
111B	EDI	EDI+Disp8	EDI+Disp32

Адрес операнда не зависит от содержимого EBP

1) **mov ECX, [DS:EBX+6]**

100010dw mod reg mem см.мл.байт

10001011 01 001 011 00000110

8 B	4 B	0 6

2) **mov CX, [DS:EBX+6]**

префикс 100010dw mod reg mem см.мл.байт

01100110 10001011 01 001 011 00000110

6 6	8 B	4 B	0 6

3) **mov CX, [ES:EBX+6]**

префикс1 префикс2 100010dw mod reg mem смещение мл.байт

01100110 00100110 10001011 01 001 011 00000110

6 6	2 6	8 B	4 B	0 6

Схемы адресации памяти без байта Sib (2)

Поле r/m	Эффективный адрес второго операнда		
	mod = 00B	mod = 01B	mod = 10B
000B	EAX	EAX+Disp8	EAX+Disp32
001B	ECX	ECX+Disp8	ECX+Disp32
010B	EDX	EDX+Disp8	EDX+Disp32
011B	EBX	EBX+Disp8	EBX+Disp32
100B	Определяется Sib	Определяется Sib	Определяется Sib
101B	Disp32	SS:[EBP+Disp8]	SS:[EBP+Disp32]
110B	ESI	ESI+Disp8	ESI+Disp32
111B	EDI	EDI+Disp8	EDI+Disp32

Адрес операнда не зависит от содержимого EBP

4) **mov ECX, [DS:BX+6]**

префикс 100010dw mod reg mem см.мл.байт

6 7	8 B	4 B	0 6

01100111 10001011 01 001 011 00000110

Размеры регистра и Disp регулируются независимо!

5) **mov ECX, [A]** ; адрес A соответствует $00403000

100010dw mod reg mem смещение

8 B	4 B	0 0	3 0	4 0	0 0

10001011 01 001 101 00000100 00110000 01000000 00000000

Схемы адресации памяти с байтом Sib

Поле base	Эффективный адрес второго операнда		
	mod = 00B	mod = 01B	mod = 10B
000B	EAX+ss*index	EAX+ss*index +Disp8	EAX+ss*index +Disp32
001B	ECX+ss*index	ECX+ss*index +Disp8	ECX+ss*index +Disp32
010B	EDX+ss*index	EDX+ss*index +Disp8	EDX+ss*index +Disp32
011B	EBX+ss*index	EBX+ss*index +Disp8	EBX+ss*index +Disp32
100B	SS:[ESP+ ss*index]	SS:[ESP+ ss*index]+ Disp8	SS:[ESP+ ss*index] +Disp32
101B	Disp32+ss*index	SS:[EBP+ss*index +Disp8]	SS:[EBP+ss*index +Disp32]
110B	ESI+ss*index	ESI+ss*index +Disp8	ESI+ss*index +Disp32
111B	EDI+ss*index	EDI+ss*index +Disp8	EDI+ss*index +Disp32

ss – масштаб;
index – индексный регистр;
base – базовый регистр

Адрес операнда не зависит от содержимого базы в EBP!

SS:
00 - 1 байт
01 - 2 байта
10- 4 байта
11- 8 байт

Пример:

```
mov  ECX,[EBX+EDI*4+6]
```

8B	4C	B B	0 6

100010dw mod reg mem SS Ind Base см.мл.байт

10001011 01 001 100 10 111 011 00000110

Машинные команды с регистрами AL|AX|EAX

Команды mov, один из операндов которых размещен в регистрах **AL|AX|EAX**, а второй задан смещением в памяти, имеют особый формат. Это связано с тем, что указанные команды унаследованы от более старого прототипа – 16-ти разрядного процессора 8080. Указанные команды не содержат байта адресации и имеют коды операции $A0_{16}$ .. $A3_{16}$, два последних бита которых также расшифровываются, как D и W.

Внимание! При этом D=1 соответствует «из регистра», а D=0 – «в регистр», а W имеет те же значения, что и в рассмотренных ранее форматах.

Сразу за кодом операции этих команд следуют 4-х байтовые (с учетом 32-х разрядной адресации) смещения.

Примеры:

1) **mov AL,[A] ;** при адресе A соответствующем $00403000

101000dw смещение 32 разряда

10100000 00000000 00110000 01000000 00000000

A0	00	30	40	00

2) **mov [B],AX** ; при адресе B соответствующем $00403004

префикс1 101000dw смещение 32 разряда

01100110 10100011 00000100 00110000 01000000 00000000

66	A3	04	30	40	00