

# Лекция 4. Сегментная адресация памяти процессорами x86. Сегментация программ.

## Внутрисегментные адреса

---

Если данные (операнды команды) уже размещены или заносятся в регистр процессора, вы указываете символическое имя регистра.

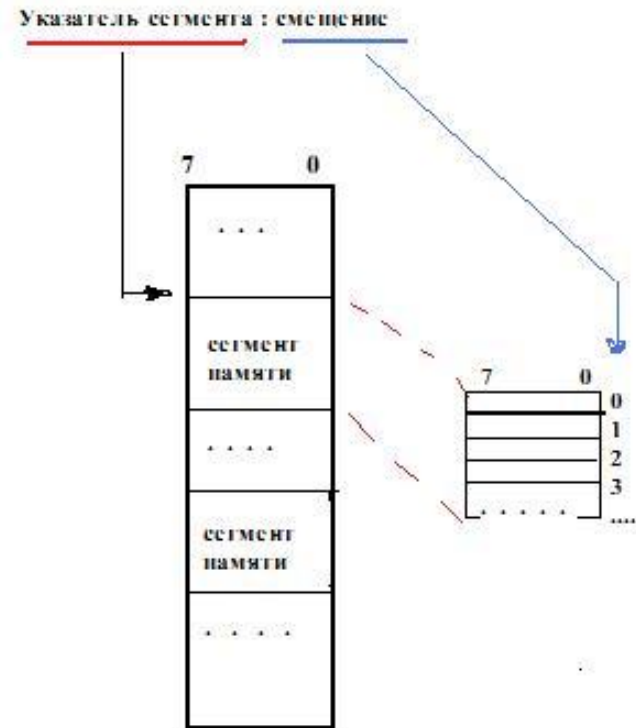
Пример:

- `mov ax, bx` ; переслать код из регистра bx в ax
- `mov eax, 3` ; занести 3 в регистр eax
- `add bx, ax` ; сложить содержимое регистров bx и ax и сумму сохранить в bx

А если ваши операнды размещены в памяти? Как задавать адрес памяти в командах процессора?

# Сегментная адресация памяти процессором

- Процессор «рассматривает» память, как набор блоков (сегментов)
- Программно адрес памяти должен задаваться ему двумя значениями:
  - «указатель на сегмент памяти», в котором находится данный байт
  - «внутрисегментный адрес» - это смещение (в байтах) от начала сегмента до данного байта: 0, 1, 2 ..
- Физический адрес памяти для обращения процессор вычислит сам: по значению «указатель сегмента» определит физический адрес начала сегмента и прибавит к нему внутрисегментный адрес.



# Сегментация программы

---

Сегментная адресация памяти процессором порождает требования к структуре загружаемой ОС в память программы:

- Любой программный объект, предназначенный для загрузки в память, должен быть оформлен, как один или несколько «программных» сегментов. Каждый программный сегмент будет загружаться ОС и размещаться в памяти независимо от другого.
  - Решение о сегментной структуре кода принимает:
    - Компилятор - при компиляции исходного текста в объектный код, если программа написана на ЯВУ
    - Автор - при написании исходного текста программы на низкоуровневом языке (ассемблере)
-

# Регистры-указатели сегментов

## Адрес команды, данных в памяти

---

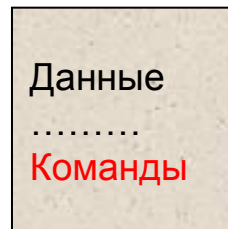
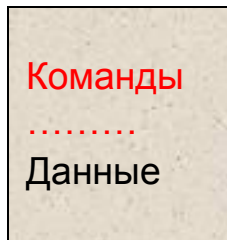
- ❑ Указателями сегментов памяти для процессора являются 16-разрядные регистры:  
**CS** - указатель на «**кодовый**» сегмент – сегмент с командами процессора  
DS, ES SS, GS, FS – указатели для сегментов данных
  - ❑ **Адрес команды** для процессора в памяти – это всегда значения регистров:  
**CS : IP (или EIP)**  
Регистр **IP** или **EIP** – задает внутрисегментный адрес команды. Именно он автоматически увеличивается на длину прочитанной команды, задавая, таким образом, процессору адрес следующей команды в кодовом сегменте
  - ❑ **Адрес данных** для процессора – это любой адрес в виде «указатель сегмента»: «внутрисегментное смещение». Выбор указателя на сегмент с данными и внутрисегментный адрес данных **полностью определяются автором**
-

# Варианты сегментной структуры программы

---

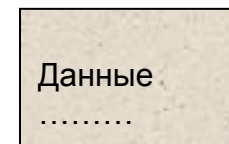
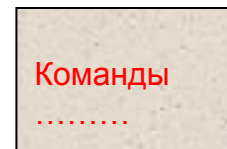
а) **Односегментная структура** . Программа представлена, как **один кодовый сегмент**, в котором размещены команды и (если есть) данные. Указателем на кодовый сегмент должен быть регистр **CS**.

Варианты размещения данных в кодовом сегменте:



б) **Многосегментная структура программы**: один /несколько **кодовых** сегментов с командами и один/несколько сегментов для данных

Пример: Двухсегментная программа



## Примеры записи адреса данных:

---

Пусть с начала сегмента данных последовательно размещены четыре байта: 05 FA 4D 9F. Указателем на сегмент данных выбран регистр `ds`.

- 1) Как записать в команде адрес байта FA? `ds: [ ? ]`
  - 2) Какое значение будет в регистре AX после выполнения команды:  
`mov ax, ds:[2]`; чтение из памяти двух байтов в регистр AX  
AX =
  - 3) Как изменится содержимое сегмента данных после выполнения процессором команды записи в память байта по указанному адресу?  
`mov ds:[1], al`; в сегменте данных: ? ? ? ?
-

# Разрядность внутрисегментных адресов - «разрядность программы»

---

Внутрисегментные адреса в программном сегменте должны иметь конкретную разрядность. Современные процессоры могут оперировать с **внутрисегментными адресами разной разрядности** – 16-ти, 32-х, 64-разрядными.

- Термин «разрядность программы» и означает разрядность внутрисегментных адресов в машинном коде программы

Варианты использования термина «разрядности программы»:

- 16-разрядное приложение,
- 32-разрядный код,
- 64-разрядная операционная системы и т.д.

Это все – про разрядность внутрисегментных адресов в исполняемых кодах этих программ

---

# 1) Как влияет разрядность внутрисегментных адресов на длину команды (в байтах)

---

Пример. Одна и та же команда, в которой используется адрес данных. Только разная разрядность ВА

- `mov vx, ds:[000f]` ; 16-разр.внутрисегментный адрес. В машинном коде команды этот адрес займет 2 байта
  
- `mov vx, ds:[0000000f]` ; 32-разрядный внутрисегментный адрес. В машинном коде команды адрес займет 4 байта
  
- `mov vx, ds:[000000000000000f]` ; 64-разрядный внутрисегментный адрес. В машинном коде команды адрес займет 8 байт.

Таким образом, увеличение разрядности внутрисегментного адреса приводит к **существенному увеличению длины команды и в итоге - объему исполняемой программы в целом.**

---



## 2) Как влияет разрядность внутрисегментного адреса на предельный размер одного программного сегмента

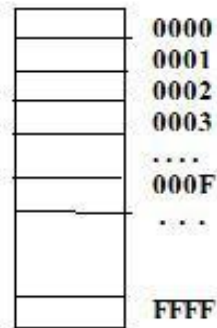
Максимальный размер одного сегмента ограничен  $2^P$ , где  $P$  – разрядность внутрисегментного адреса.

Таким образом, увеличение разрядности ВА приводит к увеличению максимального объема одного сегмента

### внутрисегментные адреса байтов

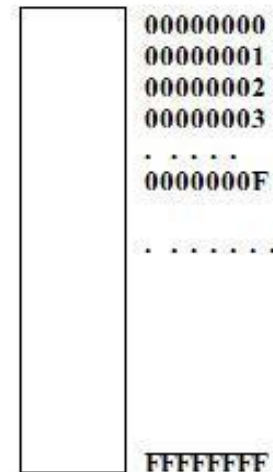
#### 16-разрядные

программный сегмент



Макс.размер сегмента=  
 $2^{16}$  байтов = 64 Кбайт

#### 32-разрядные



Макс.размер сегмента =  
 $2^{32}$  байтов = 4 Гбайт