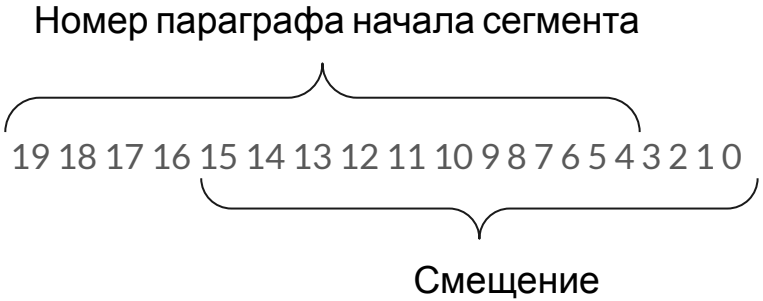
Машинно-зависимые языки программирования, лекция 2

Каф. ИУ7 МГТУ им. Н. Э. Баумана, 2024 г.

Логическая структура памяти. Сегменты

- Сегмент кода (регистр CS)
- Сегменты данных (основной регистр - DS, для дополнительных сегментов - ES, FS, GS)
- Сегмент стека (регистр SS)

Память в реальном режиме работы процессора - пример



[SEG]:[OFFSET] => физический адрес:

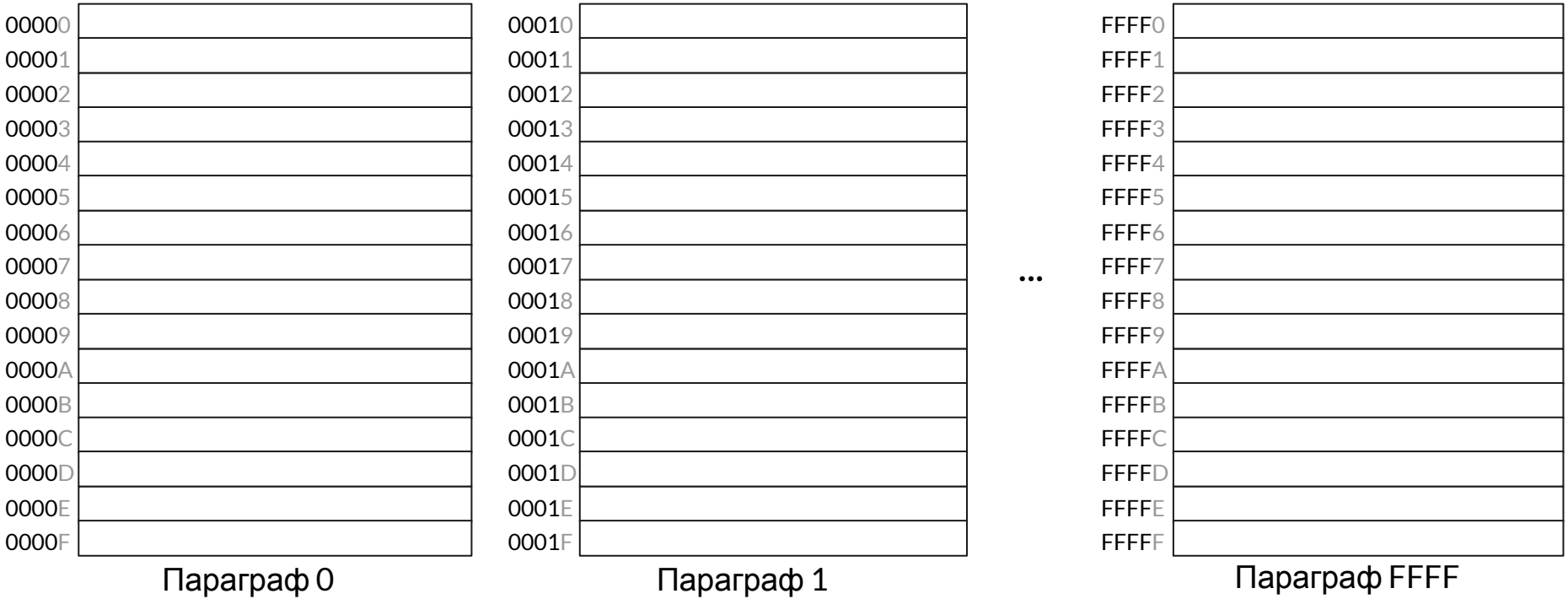
1. SEG необходимо побитово сдвинуть на 4 разряда влево (или умножить на 16, что тождественно)
2. К результату прибавить OFFSET

5678h:1234h =>

56780
+ 1234
579B4

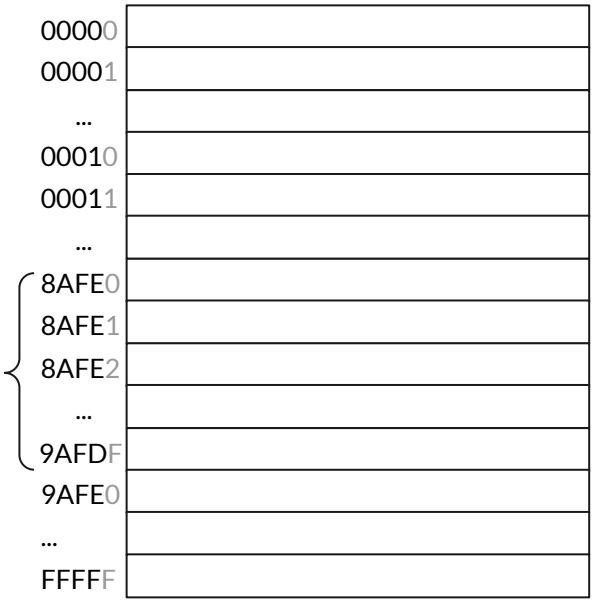
Вычисление физического адреса выполняется процессором аппаратно, без участия программиста.
Распространённые пары регистров: CS:IP, DS:BX, SS:SP

Память 8086 (20-разрядная адресация)



Сегментная модель памяти 8086

Максимальный размер сегмента с начальной сегментной частью адреса 8AFE



8AFE:0000 = 8AFE0
8AFE:001F = 8AFFF
8AFE:1000 = 8BFE0
8AFE:FFFF = 8AFE0+10000-1 = 9AFDF

Целочисленная арифметика (основные команды)

- ADD <приёмник>, <источник> - выполняет арифметическое сложение приёмника и источника. Сумма помещается в приёмник, источник не изменяется.
- SUB <приёмник>, <источник> - арифметическое вычитание источника из приёмника.
- MUL <источник> - беззнаковое умножение. Умножаются источник и AL/AX/EAX/RAX, в зависимости от размера источника. Результат помещается в AX либо DX:AX/EDX:EAX/RDX:RAX.
- IMUL <источник>; IMUL <приёмник>, <источник>; IMUL <приёмник>, <источник1>, <источник2> - знаковое умножение
- DIV <источник> - целочисленное беззнаковое деление. Делится AL/AX/EAX/RAX на источник. Результат помещается в AL/AX/EAX/RAX, остаток - в AH/DX/EDX/RDX.
- IDIV <источник> - знаковое деление
- INC <приёмник> - инкремент на 1
- DEC <приёмник> - декремент на 1

Побитовая арифметика (основные команды)

- AND <приёмник>, <источник> - побитовое “И”. AND al, 00001111b
- OR <приёмник>, <источник> - побитовое “ИЛИ”. OR al, 00001111b
- XOR <приёмник>, <источник> - побитовое исключающее “ИЛИ”. XOR AX, AX
- NOT <приёмник> - инверсия

Структура программы на ассемблере

(Зубков С. В. Assembler для DOS, Windows, Unix, глава 3)

- Модули (файлы исходного кода)
  - Сегменты (описание блоков памяти)
    - команды процессора;
    - инструкции описания структур данных, выделения памяти для переменных и констант;
    - макроопределения.

Полный формат строки:

```
метка        команда/директива        операнды        ; комментарий
```

Метки

В коде

- Пример:

```
    mov cx, 5
label1:
    add ax, bx
    loop label1
```

- Метки обычно используются в командах передачи управления

В данных

- label
  - *метка* `label` *тип*
  - Возможные типы: BYTE, WORD, DWORD, FWORD, QWORD, TBYTE, NEAR, FAR.
- EQU, =
  - *label* `EQU` *выражение*
  - макрос
  - вычисляет выражение в правой части и приравнивает его метке

Директивы выделения памяти

- Директива - инструкция ассемблеру, влияющая на процесс компиляции и не являющаяся командой процессора. Обычно не оставляет следов в формируемом машинном коде.
- Псевдокоманда - директива ассемблера, которая приводит к включению данных или кода в программу, но не соответствующая никакой команде процессора.
- Псевдокоманды определения данных указывают, что в соответствующем месте располагается переменная, резервируют под неё место заданного типа, заполняют значением и ставят в соответствие метку.
- Виды: DB (1), DW (2), DD (4), DF (6), DQ (8), DT (10).
- Примеры:
  - a DB 1
  - float_number DD 3.5e7
  - text_string DB ‘Hello, world!’
- DUP - заполнение повторяющимися данными
- ? - неинициализированное значение
- uninitialized DW 512 DUP(?)

Описание сегментов программы

- Любая программа состоит из сегментов
- Виды сегментов:
  - сегмент кода
  - сегмент данных
  - сегмент стека
- Описание сегмента - директива SEGMENT:

```
имя SEGMENT [READONLY] [выравнивание] [тип] [разрядность] [‘класс’]

    …

имя ENDS
```

Параметры директивы SEGMENT

Выравнивание

- BYTE
- WORD
- DWORD
- **PARA**
- PAGE

Тип

- PUBLIC
- STACK
- COMMON
- AT
- **PRIVATE**

Класс - любая метка, взятая в одинарные кавычки. Сегменты одного класса будут расположены в памяти друг за другом.

Модели памяти

```
.model модель, язык, модификатор
```

- Модели:
  - TINY - один сегмент на всё
  - SMALL - код в одном сегменте, данные и стек - в другом
  - COMPACT - допустимо несколько сегментов данных
  - MEDIUM - код в нескольких сегментах, данные - в одном
  - LARGE, HUGE
- Язык - C, PASCAL, BASIC, SYSCALL, STDCALL. Для связывания с ЯВУ и вызова подпрограмм.
- Модификатор - **NEARSTACK**/FARSTACK
- Определение модели позволяет использовать сокращённые формы директив определения сегментов.

Завершение описания модуля. Точка входа

.

.

.

END [точка_входа]

- точка_входа - имя метки, объявленной в сегменте кода и указывающее на команду, с которой начнётся исполнение программы.
- Если в программе несколько модулей, только один может содержать точку входа.

Сегментный префикс. Директива ASSUME

- Для обращения к переменной процессору необходимо знать обе составляющие адреса: и сегментную, и смещение.

  Пример полной записи - DS:Var1

- Директива `ASSUME регистр:имя сегмента` устанавливает значение сегментного регистра по умолчанию

```
Data1 SEGMENT WORD 'DATA'
Var1  DW    0
Data1 ENDS

Data2 SEGMENT WORD 'DATA'
Var2  DW    0
Data2 ENDS

Code  SEGMENT WORD 'CODE'
        ASSUME  CS:Code
ProgramStart:
        mov   ax,Data1
        mov   ds,ax
        ASSUME  DS:Data1
        mov   ax,Data2
        mov   es,ax
        ASSUME  ES:Data2
        mov   ax,[Var2]

        .
        .
        .
Code  ENDS
END   ProgramStart
```

Прочие директивы

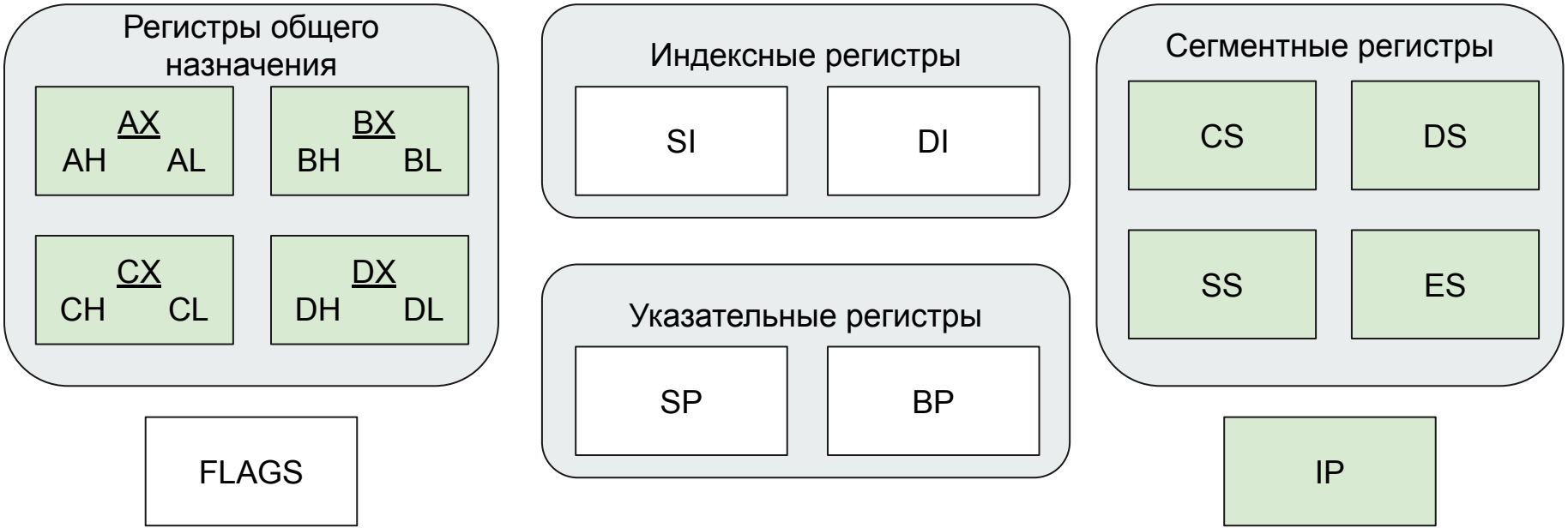
- Задание набора допустимых команд: .8086, .186, .286, ..., .586, .686, ...
- Управление программным счётчиком:
  - ORG значение
  - EVEN
  - ALIGN значение
- Глобальные объявления
  - public, comm, extrn, global
- Условное ассемблирование

```
IF выражение
    ...
ELSE
    ...
ENDIF
```

Виды переходов для команды JMP

- short (короткий) -128 .. +127 байт
- near (ближний) в том же сегменте (без изменения регистра CS)
- far (дальний) в другой сегмент (с изменением значения в регистре CS)
- Для короткого и ближнего переходов непосредственный операнд (константа) прибавляется к IP
- Операнды - регистры и переменные заменяют старое значение в IP (CS:IP)

Архитектура 8086 с точки зрения программиста (структура блока регистров)



Индексные регистры SI и DI

- SI - source index (индекс источника)
- DI - destination index (индекс приёмника)
- Могут использоваться в большинстве команд, как регистры общего назначения
- Применяются в специфических командах поточной обработки данных

Способы адресации

(Зубков С. В. Assembler для DOS, Windows, Unix, глава 2)

- непосредственная адресация (mov ax, 2)
- регистровая адресация (mov ax, bx)
- прямая адресация (mov ax, ds:[0032])
- регистровая косвенная адресация (mov ax, [bx])
- адресация по базе со сдвигом (mov ax, [bx]+2; mov ax, 2[bx]).
- адресация по базе с индексированием (допустимы BX+SI, BX+DI, BP+SI, BP+DI):
  - mov ax, [bx+si+2]
  - mov ax, [bx+2][si]
  - mov ax, 2[bx][si]
  - mov ax, [bx][si]+2
  - mov ax, [bx][si+2]
- адресация с масштабированием mov ax, [si*4]
- адресация с масштабированием и смещением mov ax, [bx][si*4]+10h

Регистр FLAGS

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

Флаги состояния:

- CF (carry flag) - флаг переноса
- PF (parity flag) - флаг чётности
- AF (auxiliary carry flag) - вспомогательный флаг переноса
- ZF (zero flag) - флаг нуля
- SF (sign flag) - флаг знака
- OF (overflow flag) - флаг переполнения

Управляющий флаг:

- DF (direction flag) - флаг направления

Системные флаги:

- IF (interrupt enable flag) - флаг разрешения прерываний
- TF (trap flag) - флаг трассировки
- IOPL (I/O privilege flag) - уровень приоритета ввода-вывода
- NT (nested task) - флаг вложенности задач

Команда сравнения CMP

CMP <приёмник>, <источник>

- Источник - число, регистр или переменная
- Приёмник - регистр или переменная; не может быть переменной одновременно с источником
- Вычитает источник из приёмника, результат никуда не сохраняется, выставляются флаги CF, PF, AF, ZF, SF, OF

Команды условных переходов Jcc

(Зубков С. В. Assembler для DOS, Windows, Unix, глава 2)

cc - condition code

- Переход типа short или near
- Обычно используются в паре с CMP
- Термины “выше” и “ниже” - при сравнении беззнаковых чисел
- Термины “больше” и “меньше” - при сравнении чисел со знаком

Виды условных переходов (часть 1)

Команда	Описание	Состояние флагов для выполнения перехода
JO	Есть переполнение	OF = 1
JNO	Нет переполнения	OF = 0
JS	Есть знак	SF = 1
JNS	Нет знака	SF = 0
JE, JZ	Если равно/если ноль	ZF = 1
JNE, JNZ	Не равно/не ноль	ZF = 0
JP/JPE	Есть чётность / чётное	PF = 1
JNP/JPO	Нет чётности / нечётное	PF = 0
JCXZ	CX = 0	-

Виды условных переходов (часть 2)

Команда	Описание	Состояние флагов для выполнения перехода	Знаковый
JB JNAE JC	Если ниже Если не выше и не равно Если перенос	CF = 1	нет
JNB JAE JNC	Если не ниже Если выше или равно Если нет переноса	CF = 0	нет
JBE JNA	Если ниже или равно Если не выше	CF = 1 или ZF = 1	нет
JA JNBE	Если выше Если не ниже и не равно	CF = 0 и ZF = 0	нет

Виды условных переходов (часть 3)

Команда	Описание	Состояние флагов для выполнения перехода	Знаковый
JL JNGE	Если меньше Если не больше и не равно	SF <> OF	да
JGE JNL	Если больше или равно Если не меньше	SF = OF	да
JLE JNG	Если меньше или равно Если не больше	ZF = 1 или SF <> OF	да
JG JNLE	Если больше Если не меньше и не равно	ZF = 0 и SF = OF	да

Прерывания

- Прерывание - особая ситуация, когда выполнение текущей программы приостанавливается и управление передаётся программе-обработчику возникшего прерывания.
- Виды прерываний:
  - аппаратные (асинхронные) - события от внешних устройств;
  - внутренние (синхронные) - события в самом процессоре, например, деление на ноль;
  - программные - вызванные командой INT.

Прерывание DOS 21h

- Аналог системного вызова в современных ОС
- Используется наподобие вызова подпрограммы
- Номер функции передаётся через AH

Прерывание DOS - вывод на экран в текстовом режиме

Функция	Назначение	Вход	Выход
02	Вывод символа в stdout	DL = ASCII-код символа	-
09	Вывод строки в stdout	DS:DX - адрес строки, заканчивающейся символом $	-

Прерывание DOS - ввод с клавиатуры

Функция	Назначение	Вход	Выход
01	Считать символ из stdin с эхом	-	AL - ASCII-код символа
06	Считать символ без эха, без ожидания, без проверки на Ctrl+Break	DL = FF	AL - ASCII-код символа
07	Считать символ без эха, с ожиданием и без проверки на Ctrl+Break	-	AL - ASCII-код символа
08	Считать символ без эха	-	AL - ASCII-код символа
10 (0Ah)	Считать строку с stdin в буфер	DS:DX - адрес буфера	Введённая строка помещается в буфер
0Bh	Проверка состояния клавиатуры	-	AL=0, если клавиша не была нажата, и FF, если была
0Ch	Очистить буфер и считать символ	AL=01, 06, 07, 08, 0Ah	