МАШИННО-ЗАВИСИМЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Лекция 2

ИУ7, 4-й семестр, 2020 г.

Важное из 1-й лекции и 1-й л/р

- Машинный код - набор кодов операций конкретной вычислительной машины, которые интерпретируется непосредственно процессором.
- Язык ассемблера - низкоуровневый язык программирования, одна команда которого соответствует одной машинной команде.
- Компиляция - процесс перевода программы с языка программирования в машинный код.
- COM-файл - простейший формат исполняемого файла DOS (и немного Windows), который считывается с диска в ОЗУ без изменений и запускается с 1-го байта. Плюсы: простой. Минусы: размер < 64 Кб; слишком простой.

Память в реальном режиме работы процессора (режим 8086)

1 Мб памяти = 2^{20}

Номер параграфа начала сегмента (сегментная часть адреса, сегмент)

19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Смещение

CS:IP, DS:BX, SS:SP…

[SEG]:[OFFSET] => физический адрес:
SEG*16 + OFFSET

5678h:7890h =>

```
  56780
+  7890
-------
  5E010
```

Логическая структура памяти. Сегменты

- Сегмент кода (CS)
- Сегменты данных (**DS**, ES, FS, GS)
- Сегмент стека (SS)

Команда NOP (no operation)

- Ничего не делает
- Занимает место и время
- Размер - 1 байт, код 90h
- Назначение - задержка выполнения либо заполнение памяти, например, для выравнивания

Структура программы на ассемблере
(Зубков С. В., Assembler для DOS, Windows, …, глава 3)

- Модули (файлы исходного кода)
- Сегменты (описание блоков памяти)
- Составляющие программного кода:
  - команды процессора
  - инструкции описания структуры, выделения памяти, макроопределения
- Формат строки программы:
  - метка команда/директива операнды ; комментарий

Метки

В коде

- Пример:

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

- Метка обычно используется в командах передачи управления

В определении данных

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

Директивы выделения памяти

- Псевдокоманда - директива ассемблера, которая приводит к включению данных или кода в программу, но не соответствует 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(?)

Структура программы

- Любая программа состоит из сегментов
- Виды сегментов:
  - сегмент кода
  - сегмент данных
  - сегмент стека
- Описание сегмента в исходном коде:

  имя SEGMENT READONLY выравнивание тип разряд '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 расположатся в памяти друг за другом.

Директива ASSUME

- ASSUME регистр:имя сегмента
- Не является командой
- Нужна для контроля компилятором правильности обращения к переменным

```
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
```

Модели памяти

.model модель, язык, модификатор

- 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.

Конец программы. Точка входа

.

.

.

END start

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

Прочие директивы

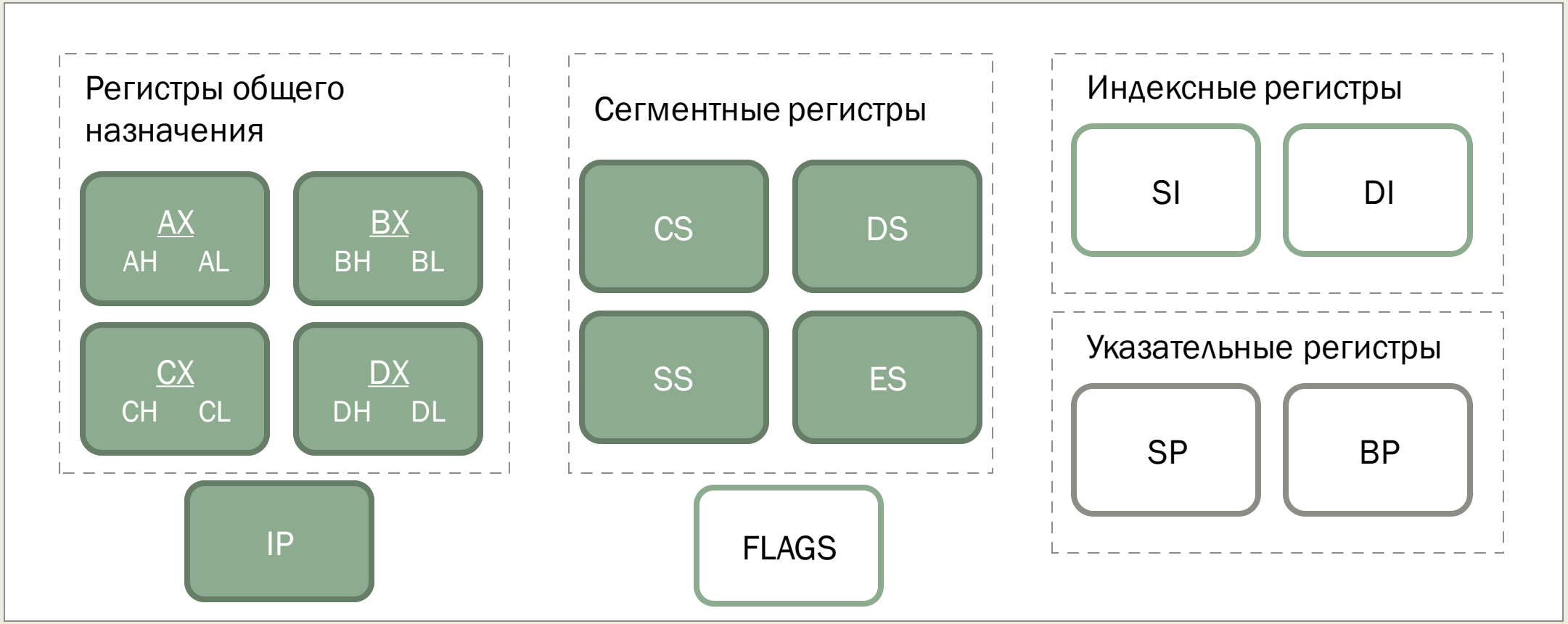
- Задание набора допусимых команд: **.8086,** .186, .286, …, .586, .686, …
- Управление программным счётчиком
  - ORG значение
  - EVEN
  - ALIGN значение
- Глобальные объявления
  - public, comm, extrn, global
- Условное ассемблирование

```
IF выражение
    …
ELSE
    …
ENDIF
```

Виды переходов для команды JMP

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

Регистры. Регистр флагов



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, Assembler, …, глава 2)

- Регистровая адресация (mov ax, bx)
- Непосредственная адресация (mov ax, 2)
- Прямая адресация (mov ax, ds:0032)
- Косвенная адресация (mov ax, [bx]). В 8086 допустимы BX, BP, SI, DI
- 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]

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	-

- 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
- TF (trap flag) - флаг трассировки
- IF (interrupt enable flag) - флаг разрешения прерываний
- DF (direction flag) - флаг направления
- OF (overflow flag) - флаг переполнения
- IOPL (I/O privilege level) - 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 J.. (Зубков, Assembler, ..., глава 2)

- Переход типа short или near
- Обычно используются в паре с CMP
- "Выше" и "ниже" - при сравнении беззнаковых чисел
- "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	Есть 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

Команда TEST

- TEST <приёмник>, <источник>
- Аналог AND, но результат не сохраняется
- Выставляются флаги SF, ZF, PF.

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	