Семинар 12.02.2021

Повторение

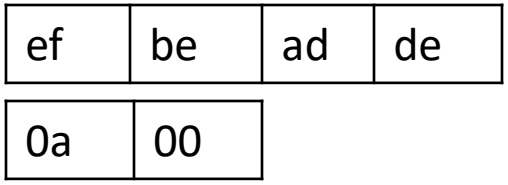
```
section .data/.rodata
     a db/dw/dd/dq 0b10_10/12o/10d/0ha/0xa

section .bss
     b resb/resw/resd/resq 4

section .text
     L: op op1, op2 ; comment
```

Повторение: little endian

```
section .data
      a dd 0xdeadbeef
      b dw 10
```



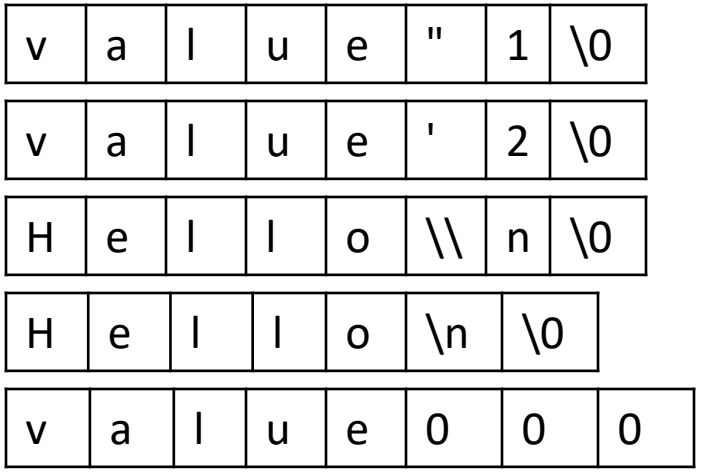
Порядок размещения байтов - обратный

Конструкция повторения times

```
section .data
      a times 64 db 0 ; 64 нулевых байта
```

Символьные/строковые константы

```
section .data
      a db 'value"1', 0
      b db "value'2", 0
      c db "Hello\n", 0
      d db `Hello\n`, 0
      e dd 'value'
```



Размещаются в памяти в прямом порядке

Регистры

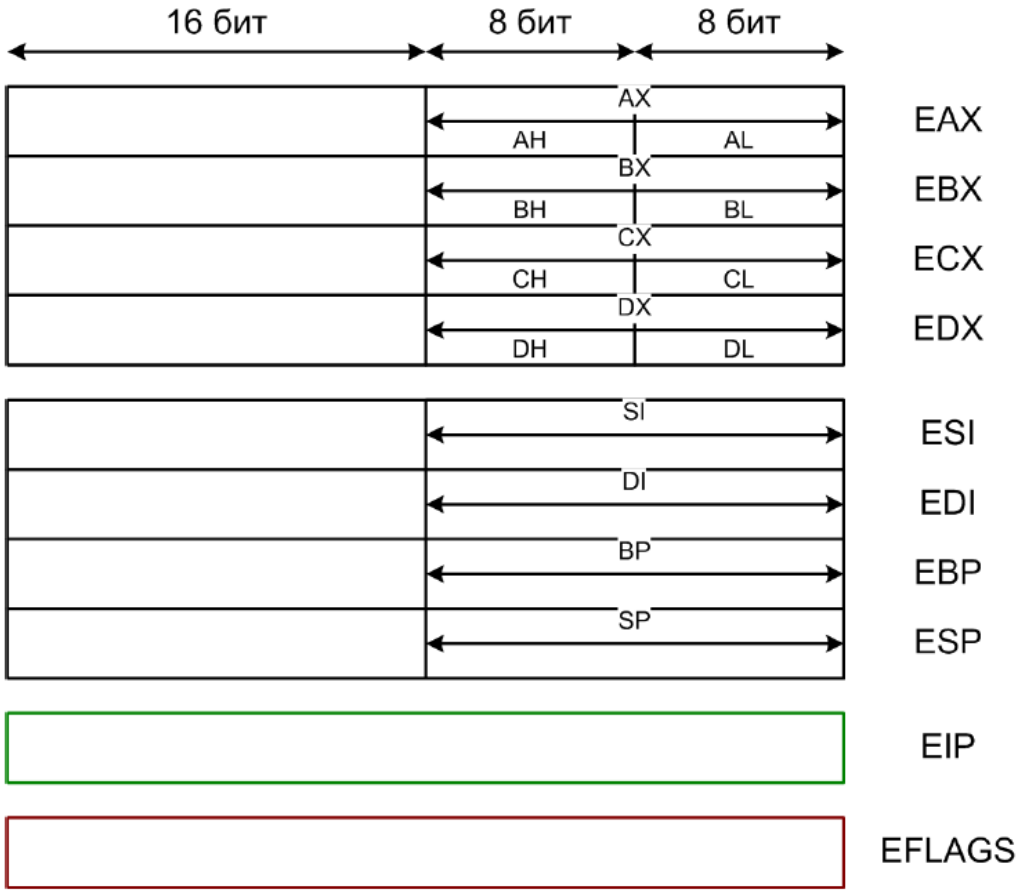


Рисунок 3 – Основные регистры IA-32.

Макросы ввода-вывода io.inc

```
PRINT_UDEC/PRINT_DEC/PRINT_HEX size, data
PRINT_CHAR/PRINT_STRING data
NEWLINE

GET_UDEC/GET_DEC/GET_HEX size, data
GET_CHAR data
GET_STRING data, maxsz
```

Документация: http://asmcourse.cs.msu.ru/?page_id=169, справка SASM

Задача

Задача 1-3 Директивы описания данных

Используя только директивы DB, привести описания, эквивалентные заданным.

```
; a
DW 185Ah
; b
DW 90
; c
DW 9000
; d
DW 10000, -10000
```

```
; e
DW 'AB'
; f
DW "NASM"
; g
DW "Hello"
; h
DW "w", "orld", "!"
```

```
; i
DD 97
; j
DD "x"
; k
DD 'CD'
; l
DD "Hello, world!"
```

Обращения к памяти

```
спецификатор размера[адресное выражение]
```

Спецификатор размера: `byte/word/dword`

Адресное выражение (effective address): `[base_reg + index_reg_not_esp * 1/2/4/8 + constant]`

Примеры:

```
byte[a], word[eax + 4], dword[eax + 2 * ebx], dword[b + 5 * ebx + 10]
```

lea - load effective address

```
lea r32/r16, [effective address]
```

Примеры:

```
lea eax, [a + 4 * eax]
lea ebx, [9 * esi]
```

mov - пересылка (move)

```
mov dst, src
```

размеры dst и src должны совпадать, dst всегда слева

Варианты операндов:

```
mov r, imm
mov r1, r2
mov r, m
mov m, r
mov m, imm
```

~~`mov m, m`~~ - запрещено для всех команд

Примеры:

```
mov eax, 10
mov ebx, ecx
mov ecx, dword[ebx]
```

Пример

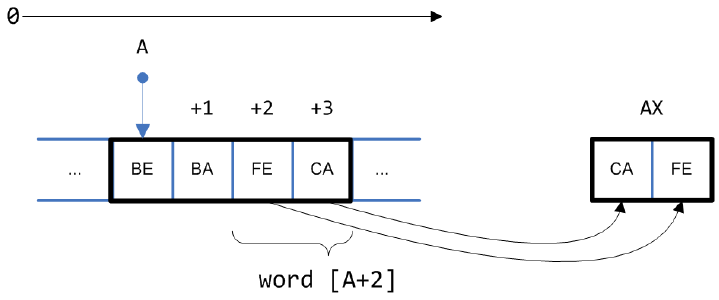
Пример 1-2 Определение значения регистра

Пусть ассемблерная переменная A имеет значение 0x CAFE BABE. Требуется выписать в шестнадцатеричном виде значение регистра AX после выполнения следующих инструкций.

```
MOV AX, WORD [A + 2]
ADD AX, 3
```

Решение

Рассмотрим расположение в памяти переменной A и определим, что будет в регистре AX, после выполнения первой инструкции. Поскольку в архитектуре IA-32 используется обратное расположение байтов в памяти, то получаем следующее:



Байт с адресом A+2 будет иметь значение 0xFE, следующий за ним – 0xCA. При пересылке в регистр байты поменяются местами, и регистр AX будет иметь значение 0xCAFE. После того, как к этому значению будет прибавлено 3, оно станет равным 0xCB01.

xchg - обмен (exchange)

```
xchg op1, op2
```

размеры op1 и op2 должны совпадать

Варианты операндов:

```
mov r1, r2
mov r, m
mov m, r
```

Примеры:

```
xchg eax, ebx
xchg dword[a], ecx
```

movzx/movsx - пересылка с расширением (move zero/sign extend)

movsx используется для знаковых чисел и расширяет число знаковым битом

movzx используется для беззнаковых чисел и расширяет число нулями

```
movzx/movsx r16, r/m8
movzx/movsx r32, r/m8
movzx/movsx r32, r/m16
```

Примеры:

```
movzx eax, bl
movsx esi, word[eax]
movzx edi, byte[a + ecx]
```

Расширение целочисленных типов при приведении в языке Си

Расширение транслируется в команды movzx/movsx в зависимости от знаковости числа.

```
static short a;
static int b;
b = a;

movsx eax, word[a]
mov dword[b], eax
```

```
static unsigned short a;
static unsigned int b;
b = a;

movzx eax, word[a]
mov dword[b], eax
```

Сужение целочисленных типов при приведении в языке Си

Сужение транслируется в mov с обращением к младшей части регистра/памяти.

```
static short a;
static int b;
a = b;
```

```
mov ax, word[b]
mov word[a], ax
```

Задача

Пример 1-3 Переворот байтов в двойном слове

В памяти последовательно расположены 4 переменных a, b, c и d размером 1 байт каждая в заданном порядке. Требуется сформировать 32-битное число в регистре EAX таким образом, чтобы старший байт числа совпадал со значением переменной a, следующий за ним байт – со значением b, следующий – со значением c, и, наконец, младший байт – со значением d. Пусть, для примера, значения a, b, c и d равны 1, 2, 3, 4 соответственно. Тогда в регистре EAX будет размещено число 0x01020304.

Задача

```
b resb 1
w resw 1
y resw 1
d resd 1
```

Вычеркнуть синтаксически неверные инструкции.

```
; a
mov b, 1
; b
mov byte [b], 1
; c
mov word [b], 1
; d
mov ax, bx
```

```
; e
mov ecx, cx
; f
mov bh, cl
; g
mov dword [d], esi
; h
mov byte [w], ch
```

```
; i
mov word[w], word[y]
; j
add 15, bx
; k
sub word [y], 8
; l
sub eax, dword [d]
```

Задача

Задача 1-5 Определение значения регистра

Выписать в шестнадцатеричном виде значение регистра EAX, после выполнения каждой помеченной инструкции. Следует отметить, что в каждой последовательности команд «а»-«ж» и «з»-«к» необходимо учитывать уже имеющееся значение регистра EAX.

```
section .data
   a dw 0xDEAD
   b dw 0xF00D
   c dw 0xCAFE
   d dw 0xBABE

section .text
   movsx eax, word [a + 1] ; (а)
   movsx ax,  byte [b]     ; (б)
   movzx eax, word [c]     ; (в)
   movsx eax, byte [b + 2] ; (г)
   movzx ax,  byte [d + 1] ; (д)
   movsx eax, word [b + 1] ; (е)
   mov   bx,  word [a + 1] ;
   movsx ax,  bh           ; (ж)
```

Задача

Задача 1-6 Перемещение данных

Привести фрагмент кода, осуществляющего пересылку данных, как это указанно на рисунке.