

БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования Введение в язык Ассемблер

1. Косвенная адресация (EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP). Чаще всего используются регистры ESI (индекс источника) EDI (индекс получателя).

Пример прямой адресации:

```
MAS DB 'HELLO'
MOV AL, MAS      ;AL='H'
```

Косвенная адресация:

```
.data                                ; сегмент данных
ddMS  dd 1,2,3,4,5,6,7
ddMD  dd 7 dup(?)
.code                                ; сегмент кода
main PROC                          ; начало процедуры

    mov esi, offset ddMS           ; смещение ddMS -> esi
    mov edi, offset ddMD           ; смещение ddMD -> edi
    mov eax, [esi]                 ; 4 байта по адресу в esi -> eax
    mov [edi], eax                 ; eax-> по адресу в edi

    add esi, 4
    add edi, 4
    mov eax, [esi]                 ; 4 байта по адресу в esi -> eax
    mov [edi], eax                 ; eax-> по адресу в edi

    add esi, 4
    add edi, 4
    mov eax, [esi]                 ; 4 байта по адресу в esi -> eax
    mov [edi], eax                 ; eax-> по адресу в edi

    push 0
    call ExitProcess
```

*(8ddMD+0)	1
*(8ddMD+1)	2
*(8ddMD+2)	3

(параметр ExitProcess)

... так должно выглядеть любой процесс Windows

```

.const          ; сегмент констант
.data           ; сегмент данных
dwMS    dw 1,2,3,4,5,6,7
dwMD    dw 7 dup(?)
.code       ; сегмент кода
main PROC  ; начало процедуры

    mov esi, offset dwMS    ; смещение ddMS -> esi
    mov edi, offset dwMD    ; смещение ddMD -> edi
    mov ax, [esi]           ; 2 байта по адресу в esi -> ax
    mov [edi], ax           ; ax-> по адресу в edi

    add esi, 2
    add edi, 2
    mov eax, [esi]          ; 2 байта по адресу в esi -> ax
    mov [edi], eax          ; ex-> по адресу в edi

    add esi, 4
    add edi, 4
    mov eax, [esi]          ; 2 байта по адресу в esi -> ax
    mov [edi], eax          ; ax-> по адресу в edi

    push 0
    call ExitProcess        ; процесс (параметр ExitProcess )
                             ; заканчиваться любой процесс Windows
main ENDP
end main                   ; конец модуля, main - точка входа

```

Имя	Значение
*(&dwMD+0)	1
*(&dwMD+1)	2
*(&dwMD+2)	3

```

bMS    byte 1,2,3,4,5,6,7
bMD    byte 7 dup(?)
.code   ; сегмент кода
main PROC ; начало процедуры

    mov esi, offset bMS    ; смещение ddMS -> esi
    mov edi, offset bMD    ; смещение ddMD -> edi
    mov al, [esi]           ; 1 байт по адресу в esi -> al
    mov [edi], al           ; al-> по адресу в edi

    inc esi                 ; ++esi
    inc edi                 ; ++edi
    mov al, [esi]           ; 1 байт по адресу в esi -> al
    mov [edi], al           ; al-> по адресу в edi

    inc esi                 ; ++esi
    inc edi                 ; ++edi
    mov al, [esi]           ; 1 байт по адресу в esi -> al
    mov [edi], al           ; al-> по адресу в edi

    push 0
    call ExitProcess        ; так должен заканчиваться любой процесс Windows
main ENDP
end main                   ; конец модуля, main - точка входа

```

Имя	Значение
*(&bMD+0)	1 '\x1'
*(&bMD+1)	2 '\x2'
*(&bMD+2)	3 '\x3'

```

.model flat,stdcall ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компоновщик: компоновать с kernel32.lib
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD byte 7 dup(?)
.code ; сегмент кода
main PROC ; начало процедуры

    mov esi, offset ddMS ; смещение ddMS -> esi
    mov eax, [esi]
    add esi,4
    add eax, [esi]
    add esi,4
    add eax, [esi]

    push 0 ; код возврата процесса (параметр ExitProcess )
    call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры
end main ; конец модуля, main - точка входа

```

Имя	Значение
eax	6

Операнды с индексом, синтаксис первой формы представления:

имя_переменной[индексный_регистр]

```

ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD byte 7 dup(?)
.code ; сегмент кода
main PROC ; начало процедуры

    mov esi,0
    mov eax,0
    add eax, ddMS[esi]
    add esi,4
    add eax, ddMS[esi]
    add esi,4
    add eax, ddMS[esi]

    push 0 ; код возврата процесса (параметр ExitProcess )
    call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры
end main ; конец модуля, main - точка входа

```

Имя	Значение
eax	6

Операнды с индексом, синтаксис второй формы представления:

[имя_переменной+индексный_регистр]

```

.stack 4096                ; сегмент стека объемом 4096
.const                    ; сегмент констант
.data                     ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD byte 7 dup(?)
.code                     ; сегмент кода
main PROC                 ; начало процедуры

    mov esi,0
    mov eax,0
    add eax, [ddMS]
    add esi,4
    add eax, [ddMS+esi]
    add esi,4
    add eax, [ddMS+esi]

    push 0
    call ExitProcess        ; так должен заканчиваться любой процесс Windows
main ENDP                  ; конец процедуры
end main                   ; конец модуля, main - точка входа

```

```

.const                    ; сегмент констант
.data                     ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD byte 7 dup(?)
.code                     ; сегмент кода
main PROC                 ; начало процедуры

    mov esi,0
    mov eax,0
    add eax, [ddMS]
    add eax, [ddMS+4]
    add eax, [ddMS+8]

    push 0
    call ExitProcess        ; код возврата процесса (параметр ExitProcess )
main ENDP                  ; конец процедуры
end main                   ; конец модуля, main - точка входа

```

Имя	Значение
eax	6

2. Указатели

```
.const                ; сегмент констант
.data                ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD dd 7 dup(?)
pddMS dd offset ddMS ; указатель на ddMS
pddMD dd offset ddMD ; указатель на ddMD
.code                ; сегмент кода
main PROC            ; начало процедуры

    mov esi,pddMS
    mov edi,pddMD
    mov eax,[esi]
    mov [edi], eax

    add esi,4
    add edi,4
    mov eax,[esi]
    mov [edi], eax

    add esi,4
    add edi,4
    mov eax,[esi]
    mov [edi], eax

    push 0            ; код возврата процесса (параметр ExitProcess )
    call ExitProcess  ; так должен заканчиваться любой процесс Windows
main ENDP             ; конец процедуры
end main              ; конец модуля, main - точка входа
```

Имя	Значение
pddMS	9453568
pddMD	9453596
*(&ddMD+0)	1
*(&ddMD+1)	2
*(&ddMD+2)	3

3. Команды переходов

3.1 Синтаксис:

JMP

метка_перехода

```
.const          ; сегмент констант
.data           ; сегмент данных
ddMS  dd  1,2,3,4,5,6,7
ddMD  dd  7 dup(?)
pddMS dd offset ddMS    ; указатель на ddMS
pddMD dd offset ddMD    ; указатель на ddMD
.code         ; сегмент кода
main PROC     ; начало процедуры

    mov esi,pddMS
    mov edi,pddMD
    mov eax,[esi]
    mov [edi], eax
    jmp  L1          ; переход по адресу L1

    add esi,4
    add edi,4
    mov eax,[esi]
    mov [edi], eax
L1:          ; метка
    add esi,4
    add edi,4
    mov eax,[esi]
    mov [edi], eax

    push 0           ; код возврата процесса (параметр ExitProcess )
    call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP           ; конец процедуры
```

3.2 Синтаксис:

LOOP

метка_перехода

<метка_перехода>:

... ;

loop <метка_перехода >

Регистр ECX используется в качестве счетчика.

В ECX загружается количество повторений цикла.

- На каждом шаге выполнения цикла значение ECX автоматически уменьшается на 1 и сравнивается с 0.
- Если не ноль – переходим по метке
- В противном случае выполняется следующая по порядку команда.

```
.data                                ; сегмент данных
ddMS  dd  1,2,3,4,5,6,7
ddMD  dd  7 dup(?)

.code                                ; сегмент кода
main PROC                            ; начало процедуры

    mov esi, offset ddMS
    mov edi, offset ddMD

    mov ecx, 7                        ; счетчик
CYCLE:                               ; метка
    mov eax, [esi]
    mov [edi], eax
    add esi, 4
    add edi, 4
    loop CYCLE                       ; --ecx, if (ecx != 0) goto CYCLE

    push 0                           ; код возврата процесса (параметр ExitProcess)
    call ExitProcess                 ; так должен заканчиваться любой процесс

main ENDP                            ; конец процедуры
end main                            ; конец модуля, main - точка входа
```

Имя	Знач
*(&ddMD+0)	1
*(&ddMD+1)	2
*(&ddMD+2)	3
*(&ddMD+3)	4
*(&ddMD+4)	5
*(&ddMD+5)	6
*(&ddMD+6)	7

```

; сегмент констант
; сегмент данных
.data
ddMS dd 1,2,3,4,5,6,7
ddMD dd 7 dup(?)

; сегмент кода
main PROC
; начало процедуры

mov esi, offset ddMS
mov edi, offset ddMD

mov ecx, lengthof ddMS ; счетчик
CYCLE: ; метка
mov eax, [esi]
mov [edi], eax
add esi, type ddMS
add edi, type ddMD
loop CYCLE ; --ecx, if (ecx != 0) goto CYCLE

push 0 ; код возврата процесса (параметр ExitProcess )
call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры
end main ; конец модуля, main - точка входа

```

Имя	Знач
*(&ddMD+0)	1
*(&ddMD+1)	2
*(&ddMD+2)	3
*(&ddMD+3)	4
*(&ddMD+4)	5
*(&ddMD+5)	6
*(&ddMD+6)	7

```

; сегмент данных
.data
ddMS dd 1,2,3,4,5,6,7
ddMD dd 7 dup(?)

; сегмент кода
main PROC
; начало процедуры

call proc1 ; поместить в стек адрес следующей
; команды и jmp proc1

push 0 ; код возврата процесса (
call ExitProcess ; так должен заканчиваться
main ENDP ; конец процедуры

proc1 PROC ; начало процедуры
mov esi, offset ddMS
mov edi, offset ddMD
mov ecx, lengthof ddMS ; счетчик
CYCLE: ; метка
mov eax, [esi]
mov [edi], eax
add esi, type ddMS
add edi, type ddMD
loop CYCLE ; --ecx, if (ecx != 0) goto CYCLE
ret ; рор адрес возврата и jmp
proc1 ENDP ; конец процедуры

end main ; конец модуля, main - точка входа

```

Имя	Зн
*(&ddMD+0)	1
*(&ddMD+1)	2
*(&ddMD+2)	3
*(&ddMD+3)	4
*(&ddMD+4)	5
*(&ddMD+5)	6
*(&ddMD+6)	7

4. Операции со стеком: PUSH, POP, PUSHAD, POPAD, CALL, RET, регистр ESP

Ассемблер	Комментарии	Адрес	Операция	Операнды
call proc1				
push 0				
call ExitProcess				
main ENDP				
proc1 PROC uses esi edi ecx				
mov esi, offset ddMS		008E1027	call	_ExitProcess@4 (08E105)
mov edi, offset ddMD		008E102C	push	esi
mov ecx, lengthof ddMS		008E102D	push	edi
CYCLE:		008E102E	push	ecx
mov eax, [esi]		008E102F	mov	esi, 8E4000h
mov [edi], eax		008E1034	mov	edi, 8E401Ch
add esi, type ddMS		008E1039	mov	ecx, 7
add edi, type ddMD		008E103E	mov	eax, dword ptr [esi]
loop CYCLE		008E1040	mov	dword ptr [edi], eax
ret		008E1042	add	esi, 4
proc1 ENDP		008E1045	add	edi, 4
end main		008E1048	loop	CYCLE (08E103Eh)
		008E104A	pop	ecx
		008E104B	pop	edi
		008E104C	pop	esi
		008E104D	ret	

```

ddd    dd    1
ddw    dw    2
ddesp0 dd    0
ddesp1 dd    0
ddesp2 dd    0
ddesp3 dd    0
ddesp4 dd    0
ddesp5 dd    0
ddesp6 dd    0
ddesp7 dd    0
code                                ; сегмент кода
main PROC                          ; начало процедуры
mov eax, 01020304h
mov ddesp0, esp
push eax
mov ddesp1, esp
push ax
mov ddesp2, esp
push 0101h
mov ddesp3, esp
push 1
mov ddesp4, esp
push word ptr 1
mov ddesp5, esp
push ddd
mov ddesp6, esp
push ddw
mov ddesp7, esp









```

ddesp0	0x00b7fbd4
ddesp1	0x00b7fbd0
ddesp2	0x00b7fbce
ddesp3	0x00b7fbca
ddesp4	0x00b7fbc6
ddesp5	0x00b7fbc4
ddesp6	0x00b7fbc0
ddesp7	0x00b7fbbe

```

pop    ddw
mov    ddesp7, esp
pop    ddd
mov    ddesp6, esp
pop    ax
mov    ddesp5, esp
pop    eax
mov    ddesp4, esp
pop    eax
mov    ddesp3, esp
pop    ax
mov    ddesp2, esp
push   eax
mov    ddesp1, esp







```







Имя	Значение
 ddesp0	0x003efed4
 ddesp1	0x003efecc
 ddesp2	0x003efed0
 ddesp3	0x003efece
 ddesp4	0x003efeca
 ddesp5	0x003efec6
 ddesp6	0x003efec4
 ddesp7	0x003efec0







```

mov    eax, 77
mov    ecx, 77
mov    edx, 77
mov    ebx, 77
mov    esi, 77
mov    edi, 77
pushad
mov    eax, 88
mov    ecx, 88
mov    edx, 88
mov    ebx, 88
mov    esi, 88
mov    edi, 88
popad
push    0
call    ExitProcess

```

Имя	Значение
 eax	77
 ecx	77
 edx	77
 ebx	77
 esi	77
 edi	77

Имя	Значение
 eax	88
 ecx	88
 edx	88
 ebx	88
 esi	88
 edi	88

Имя	Значение
 eax	77
 ecx	77
 edx	77
 ebx	77
 esi	77
 edi	77

возрата процесса (должен заканчивать)

5. Логические команды AND, OR, XOR, NOT

Синтаксис:

AND	получатель	источник
-----	------------	----------

Таблица истинности для операции логического И:

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

```
.code
main PROC
    mov eax, 00011011h
    mov ebx, 01010110h
    and eax, ebx
    ;
```

eax 0x00010010

```
main PROC ; начало процедуры
    mov eax, 11111011h
    mov ebx, 01010110h
    and ax, bx
    ;
```

eax 0x11110010

```
.code ; сегмент кода
main PROC ; начало процедуры
    mov eax, 11111011h
    mov ebx, 00000010h
    and al, bl
    ;
```

eax 0x11111100

```
.data ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD dd 7 dup(?)
ddAND dd 11111111h
dwAND dw 1111h
bAND byte 11111111b

.code ; сегмент кода
main PROC ; начало процедуры
    mov eax, 10101001h
    and ddAND, eax
    and eax, ddAND
    and dwAND, ax
    and ax, dwAND
    and al, bAND
    and bAND, ah
    ;
```

Синтаксис:

OR	получатель	источник
----	------------	----------

Таблица истинности для операции логического ИЛИ:

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

```
.code ; сегмент кода
main PROC ; начало процедуры
    mov eax, 10101001h
    mov ebx, 01011000h
    or ebx, eax
```

ebx	0x11111001
-----	------------

```
ddAND dd 11111111h
dwAND dw 1111h
bAND byte 11111111b
```

```
.code ; сегмент кода
main PROC ; начало процедуры
    mov eax, 10101001h
    or ddAND, eax
    or eax, ddAND
    or dwAND, ax
    or ax, dwAND
    or al, bAND
    or bAND, ah
    or eax, 2
    or ddAND, 2
    or dwAND, 2
    or al, 5
```


Синтаксис:

XOR	получатель	источник
------------	-------------------	-----------------

Таблица истинности для операции исключающего ИЛИ:

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

```
.code                                ; сегмент кода
main PROC                            ; начало процедуры
    mov eax, 10101001h
    mov ebx, 01011000h
    xor ebx, eax
```

 ebx 0x11110001

Синтаксис:







NOT	операнд
------------	----------------

Таблица истинности для операции отрицания:

X	NOT X
0	1
1	0
0	1
1	0

```
ddMS    dd 1,2,3,4,5,6,7
ddMD    dd 7 dup(?)
ddAND    dd 11111111h
dwAND    dw 1111h
bAND     byte 11111111b

.code                                ; сегмент кода
main PROC                            ; начало процедуры
    mov eax, 10101001h
    mov ebx, 01011000h
    mov ecx, 11111111h
    not eax
    not bx
    not ch
    not ddAND
    not dwAND
    not bAND
```

Имя	Значение
 eax	0xefefeffe
 ebx	0x0101efff
 ecx	0x1111ee11
 ddAND	0xffffffff
 dwAND	0xffff
 bAND	0x00 '\0'