

Введение в язык Ассемблера

План лекции:

- формат команды ассемблера;
- типы операндов команды ассемблера;
- команды пересылки;
- команды целочисленного сложения и вычитания;
- выравнивание на границу 2, 4, 8, 16 байтов;
- операторы OFFSET, TYPE, PTR, SIZEOF, LENGTHOF.

1. Формат команды ассемблера:

[имя_метки:]	мнемоника_команды	[операнд(ы)]	[;комментарий]
--------------	-------------------	--------------	----------------

Основные типы операндов:

- непосредственное значение;
- регистр;
- память.

2. Команды пересылки данных

2.1. Команда MOV копирует данные из операнда-источника в операнд-получатель.

MOV	получатель	источник
-----	------------	----------

Оба операнда должны быть одного типа и иметь одинаковую длину.
Оба операнда не могут одновременно быть памятью.
В качестве получателя нельзя использовать регистры CS, EIP и IP

MOV – мнемоника команды

Операнды:

- «регистр–регистр»;
- «регистр–память»;
- «память–регистр»;
- «регистр–число (непосредственно значение)»;
- «память– число».

Использование команды MOV с операндами «память-регистр» и «регистр-память»:

1	.586	; система команд(процессор Pentium)	
2	.MODEL FLAT, STDCALL	; модель памяти, соглашение о вызовах	
3	includelib kernel32.lib	; компоновщику: компоновать с kernel32	
4	ExitProcess PROTO : DWORD	; прототип	
5	.STACK 4096	; выделение	
6	.CONST	; сегмент к	
7	.DATA	; сегмент д	
8	dda dd 00112233h		
9	ddb dd 00000000h		
10	dwA dw 4455h		
11	dwB dw 0000h		
12	ba byte 66h		
13	bBh byte 00h		
14	bB1 byte 00h		
15			
16	.CODE	; сегмент кода	
17	main PROC	; точка входа main	
18	mov eax, dda	; dda(4 байта)->eax	
19	mov ddb, eax	; eax(4 байта)->ddb	
20	mov ax, dwA	; dwA(2 байта)->ax	
21	mov dwB, ax	; ax(2 байта)->dwB	
22	mov ah, ba	; ba(1 байт)->ah	
23	mov bBh, ah	; ah(1 байт)->bBh	
24	mov al, ba	; ba(1 байт)->al	
25	mov bB1, al	; al(1 байт)->bB1	
26			
27	push 0	; код возврата проце	
28	call ExitProcess	; так завершается лк	
29	main ENDP	; конец процедуры	
30	end main	; конец модуля main	
31			

Имя	Значение	Тип
eax	0x00112233	unsigned
ddb	0x00112233	unsigned
dwB	0x0000	unsigned
bBh	0x00 \0'	unsigned
bB1	0x00 \0'	unsigned

Точка останова 1

Имя	Значение	Тип
eax	0x00114455	unsigned
ddb	0x00112233	unsigned
dwB	0x4455	unsigned
bBh	0x00 \0'	unsigned
bB1	0x00 \0'	unsigned

Точка останова 2

Имя	Значение	Тип
eax	0x00116655	unsigned
ddb	0x00112233	unsigned
dwB	0x4455	unsigned
bBh	0x66 'f'	unsigned
bB1	0x00 \0'	unsigned

Точка останова 3

Имя	Значение	Тип
eax	0x00116666	unsigned
ddb	0x00112233	unsigned
dwB	0x4455	unsigned
bBh	0x66 'f'	unsigned
bB1	0x66 'f'	unsigned

Точка останова 4

Использование команды MOV с операндами «регистр-число»:

12	ba byte 66h	
13	bBh byte 00h	
14	bB1 byte 00h	
15		
16	.CODE	; сегмент кода
17	main PROC	; точка входа main
18		
19	mov ebx, 00010203h	; 00010203h->ebx
20	mov bx, 0001h	; 0001h->bx
21	mov bh, 11h	; 11h->bh
22	mov bl, 11h	; 11h->bl
23		
24	push 0	; код возврата проце
25	call ExitProcess	; так завершается лк
26	main ENDP	; конец процедуры
27	end main	; конец модуля main

Имя	Значение	Тип
ebx	0x00010203	unsigned int

Имя	Значение	Тип
ebx	0x00010001	unsigned int

Имя	Значение	Тип
ebx	0x00011101	unsigned int

Имя	Значение	Тип
ebx	0x00011111	unsigned int





Использование команды MOV с операндами «память-число»:

```
.DATA; сегмент данных
dda dd    00112233h
ddb dd    00000000h
dwA dw    4455h
dwB dw    0000h
ba  byte  66h
bBh byte  00h
bBl byte  00h

.CODE          ; сегмент кода
main PROC     ; точка входа main

mov ddb, 00010203h ; 00010203h->ddb
mov dwB, 0001h    ; 0001h->dwB
mov bBh, 11h      ; 11h->bBh
mov bBl, 11h      ; 11h->bBl
```

Контрольные значения 1

Имя	Значение	Тип
 ddb	0x00010203	unsigned long
 dwB	0x0001	unsigned short
 bBh	0x11 '\x11'	unsigned char
 bBl	0x11 '\x11'	unsigned char

2.2. Команда расширения целых беззнаковых чисел MOVZX (*move with zero-extend*) копирует содержимое исходного операнда в больший по размеру регистр получателя данных (используется для беззнаковых чисел).

Оставшиеся *неопределенными битами* регистра-получателя сбрасываются в ноль:

- слово → 32-разрядный регистр (старшие 16 бита сбрасываются в ноль);
- байт → 32-разрядный регистр (старшие 24 бита сбрасываются в ноль);
- байт → 16-разрядный регистр (старшие 8 бита сбрасываются в ноль).

<pre> .586 .MODEL flat, stdcall includelib kernel32.lib ExitProcess PROTO :DWORD .stack 4096 .const .data ; сегмент данных ddb dd 00000000h dwA dw 4455h dwb dw 0000h bA1 byte 66h bA2 byte 77h bBh byte 00h bB1 byte 00h .CODE ; сегмент кода main PROC ; начало процедуры mov ecx, 0fffffffh movzx ecx, dwA ; dwA -> ecx mov ecx, 0fffffffh movzx ecx, bA1 ; bA1 -> ecx mov ecx, 0fffffffh movzx cx, bA2 ; bA2 -> ch push 0 ; код возврата (параметр) call ExitProcess ; завершение процесса Win32 main ENDP ; конец процедуры end main ; конец модуля, точка входа </pre>	<div>Контрольные значения 1</div> <table border="1"> <thead> <tr> <th>Имя</th> <th>Значение</th> <th>Тип</th> </tr> </thead> <tbody> <tr> <td>ecx</td> <td>0xffffffff</td> <td>unsigned int</td> </tr> </tbody> </table> <div>Контрольные значения 1</div> <table border="1"> <thead> <tr> <th>Имя</th> <th>Значение</th> <th>Тип</th> </tr> </thead> <tbody> <tr> <td>ecx</td> <td>0x00004455</td> <td>unsigned int</td> </tr> </tbody> </table> <div>Контрольные значения 1</div> <table border="1"> <thead> <tr> <th>Имя</th> <th>Значение</th> <th>Тип</th> </tr> </thead> <tbody> <tr> <td>ecx</td> <td>0xffffffff</td> <td>unsigned int</td> </tr> </tbody> </table> <div>Контрольные значения 1</div> <table border="1"> <thead> <tr> <th>Имя</th> <th>Значение</th> <th>Тип</th> </tr> </thead> <tbody> <tr> <td>ecx</td> <td>0x00000066</td> <td>unsigned int</td> </tr> </tbody> </table> <div>Контрольные значения 1</div> <table border="1"> <thead> <tr> <th>Имя</th> <th>Значение</th> <th>Тип</th> </tr> </thead> <tbody> <tr> <td>ecx</td> <td>0xffffffff</td> <td>unsigned int</td> </tr> </tbody> </table> <div>Контрольные значения 1</div> <table border="1"> <thead> <tr> <th>Имя</th> <th>Значение</th> <th>Тип</th> </tr> </thead> <tbody> <tr> <td>ecx</td> <td>0x0fff0077</td> <td>unsigned int</td> </tr> </tbody> </table>	Имя	Значение	Тип	ecx	0xffffffff	unsigned int	Имя	Значение	Тип	ecx	0x00004455	unsigned int	Имя	Значение	Тип	ecx	0xffffffff	unsigned int	Имя	Значение	Тип	ecx	0x00000066	unsigned int	Имя	Значение	Тип	ecx	0xffffffff	unsigned int	Имя	Значение	Тип	ecx	0x0fff0077	unsigned int
Имя	Значение	Тип																																			
ecx	0xffffffff	unsigned int																																			
Имя	Значение	Тип																																			
ecx	0x00004455	unsigned int																																			
Имя	Значение	Тип																																			
ecx	0xffffffff	unsigned int																																			
Имя	Значение	Тип																																			
ecx	0x00000066	unsigned int																																			
Имя	Значение	Тип																																			
ecx	0xffffffff	unsigned int																																			
Имя	Значение	Тип																																			
ecx	0x0fff0077	unsigned int																																			

Команда **mov** – источник и получатель должны быть одной длины:

```

bA1 byte 66h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC

    mov     eax, 00112233h
    mov     ebx, 44556677h
    mov     ecx, 8899AABBh
    mov     eax, ebx
    mov     ax, cx
    mov     ah, bh
    mov     al, bl

    push    0
    call    ExitProcess
main ENDP
end main

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x44556677	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x4455aabb	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x44556677	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Применение команды **movzx**:

```

.586
.MODEL flat, stdcall
includelib kernel32.lib
ExitProcess PROTO :DWORD
.stack 4096
.const
.data
; сегмент данных
ddb      dd      00000000h
dwA      dw      4455h
dwb      dw      0000h
bA1      byte    66h
bA2      byte    77h
bBh      byte    00h
bB1      byte    00h

.CODE
; сегмент кода
main PROC
; начало процедуры

mov     eax, 00112233h
mov     ebx, 44556677h
movzx   eax, bx
movzx   eax, bh
movzx   eax, bl
mov     eax, 00112233h

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

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00006677	unsigned int
ebx	0x44556677	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00000066	unsigned int
ebx	0x44556677	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00000077	unsigned int
ebx	0x44556677	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int

2.3. Команда расширения целых беззнаковых чисел MOVSX (*move with sing-extend* – переместить и дополнить знаком) копирует содержимое исходного операнда в больший по размеру регистр получателя данных. При этом оставшиеся неопределенными биты регистра-получателя дополняет значением **знакового бита** исходного операнда.

Assembly code snippet:

```

ddB dd 00000000h
dwA dw 4455h
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    mov eax, 00112233h
    mov ebx, 0011F2F3h
    mov ecx, 44556677h
    movsx eax, bx
    movsx eax, bh
    movsx eax, bl
    movsx eax, dwA
    movsx eax, bA1

    push 0
    call ExitProcess
main ENDP
end main

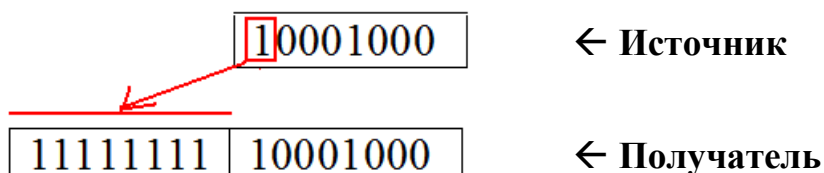
```

Control Values 1 (EAX, EBX, ECX) at various stages:

- Initial: EAX=0x00112233, EBX=0x0011f2f3, ECX=0x44556677
- After `mov ebx, 0011F2F3h`: EAX=0xfffff2f3, EBX=0x0011f2f3, ECX=0x44556677
- After `mov ecx, 44556677h`: EAX=0xffffffff, EBX=0x0011f2f3, ECX=0x44556677
- After `push 0`: EAX=0xffffffff, EBX=0x0011f2f3, ECX=0x44556677
- After `call ExitProcess`: EAX=0x00004455, EBX=0x0011f2f3, ECX=0x44556677

Bit extension diagram:

10001000
11111111 10001000



2.4. Команда XCHG (*exchange data* – обмен данными) обменивает содержимое двух операндов (! длины операндов должны совпадать !).

Допустимые операнды:

- «регистр-регистр»;
- «регистр-память»;
- «память-регистр».

Применение для операндов «регистр - регистр»:

Обмен содержимого 32/16/8 разрядных регистров:

```
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC

    mov eax, 00112233h
    mov ebx, 44556677h
    mov ecx, 8899AABBh
    xchg eax, ebx
    xchg ax, cx
    xchg ah, bh
    xchg al, bl

    push 0
    call ExitProcess
main ENDP
end main

; xchg eax, ddA
; xchg ax, dwA
; xchg ah, bA1
; xchg al, bA2
; xchg ddA, eax
; xchg dwA, ax
; xchg bA1, ah
```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x44556677	unsigned int
ebx	0x00112233	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x4455aabb	unsigned int
ebx	0x00112233	unsigned int
ecx	0x88996677	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x445522bb	unsigned int
ebx	0x0011aa33	unsigned int
ecx	0x88996677	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x44552233	unsigned int
ebx	0x0011aabb	unsigned int
ecx	0x88996677	unsigned int

Применение для операндов «регистр-память», «память регистр».

Обмен содержимого операнда заданной длины и регистра:

```

.const
.data
ddA dd 0CCCCCCCch
ddB dd 00000000h
dwA dw 4455h
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC

    mov eax, 00112233h
    xchg eax, ddA
    xchg ax, dwA
    xchg ah, bA1
    xchg al, bA2
    xchg ddA, eax
    xchg dwA, ax
    xchg bA1, ah
    xchg bA2, al

    push 0
    call ExitProcess
main ENDP
end main
        
```

The screenshots show the following states (from top to bottom):

- Initial state:** eax=0x00112233, ddA=0xCCCCCCC, dwA=0x4455, bA1=0x88, bA2=0x77.
- After `xchg eax, ddA`:** eax=0xCCCCCCC, ddA=0x00112233.
- After `xchg ax, dwA`:** eax=0xCCCC4455, ddA=0x00112233, dwA=0xCCCC.
- After `xchg ah, bA1`:** eax=0xCCCC8855, ddA=0x00112233, dwA=0xCCCC, bA1=0x44.
- After `xchg al, bA2`:** eax=0xCCCC8855, ddA=0x00112233, dwA=0xCCCC, bA1=0x44, bA2=0x55.
- After `xchg ddA, eax`:** eax=0x00112233, ddA=0xCCCC8877, dwA=0xCCCC, bA1=0x44, bA2=0x55.
- After `xchg dwA, ax`:** eax=0x00112233, ddA=0xCCCC8877, dwA=0x2233, bA1=0x44, bA2=0x55.
- After `xchg bA1, ah`:** eax=0x00114455, ddA=0xCCCC8877, dwA=0x2233, bA1=0xCC, bA2=0x55.
- After `xchg bA2, al`:** eax=0x00114455, ddA=0xCCCC8877, dwA=0x2233, bA1=0xCC, bA2=0xCC.

3. Целочисленное сложение и вычитание

3.1. Команды **INC** и **DEC** прибавляют и вычитают единицу из указанного операнда соответственно.

INC: операнд регистр:

```
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; инкремент
    mov eax, 00112233h
    inc eax
    mov eax, 0011FFFFh
    inc ax
    mov eax, 0011FFFFh
    inc ah
    mov eax, 0011FFFFh
    inc al

    push 0
    call ExitProcess
main ENDP
end main
```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112234	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x0011ffff	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00110000	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x0011ffff	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x001100ff	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x0011ffff	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x0011ff00	unsigned int

INC: операнд память:

```

.model flat,stdcall      ; модель памяти, соглашение
includelib kernel32.lib  ; компоновщик: компоновать
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096              ; сегмент стека объемом 4096
.const                   ; сегмент констант
.data                   ; сегмент данных
ddA dd 0CCCCCCCCh
ddB dd 00000000h
dwA dw 4455h
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; инкремент
    inc ddA
    inc dwA
    inc bA1

    push 0
    call ExitProcess

```

Контрольные значения 1

Имя	Значение	Тип
ddA	0xCCCCCCCc	unsigned long
dwA	0x4455	unsigned short
bA1	0x88 '€'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
ddA	0xCCCCCcd	unsigned long
dwA	0x4456	unsigned short
bA1	0x89 '%'	unsigned char

```

.data
ddA dd 0CCCCCCCCh
ddB dd 00000000h
dwA dw 4455h
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; декремент
    mov eax, 00112233h
    mov ebx, 44550000h
    mov ecx, 77880000h
    dec eax
    dec bx
    dec ch
    dec cl

    push 0
    call ExitProcess
main ENDP
end main

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44550000	unsigned int
ecx	0x77880000	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112232	unsigned int
ebx	0x4455ffff	unsigned int
ecx	0x7788ffff	unsigned int

```

.data
ddA dd 0CCCCCCCCh
ddB dd 00000000h
dwA dw 4455h
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC
; декремент
dec ddA
dec dwA
dec bA1

push 0
call ExitProcess
; так должен заканчиваться лю

```

Контрольные значения 1

Имя	Значение	Тип
ddA	0xCCCCCCCc	unsigned long
dwA	0x4455	unsigned short
bA1	0x88 '€'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
ddA	0xCCCCCCcb	unsigned long
dwA	0x4454	unsigned short
bA1	0x87 '÷'	unsigned char

3.2. Команда **ADD** прибавляет значение операнда-источника к значению операнда получателя.

! Команда **ADD** изменяет значения флагов переноса, переполнения, знака и др. (о флагах позже).

```
ExitProcess PROTO :DWORD : прототип функции
.stack 4096
.const
.data
ddA dd 0CCCCCCCCh
ddB dd 00000000h
dwA dw 4455h
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; декремент
    mov eax, 00112233h
    mov ebx, 44556677h
    mov ecx, 8899AABBh
    add eax, ebx
    add bx, cx
    add ch, bl
    add cl, bh

    push 0
    call ExitProcess
main ENDP
end main
```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x446688aa	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x446688aa	unsigned int
ebx	0x44551132	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x446688aa	unsigned int
ebx	0x44551132	unsigned int
ecx	0x8899dcbb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x446688aa	unsigned int
ebx	0x44551132	unsigned int
ecx	0x8899dccc	unsigned int

```

.const                                ; сегмент констант
.data                                ; сегмент данных
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; декремент
    mov eax, 00112233h
    mov ebx, 44556677h
    mov ecx, 8899AABBh
    add eax, ddA
    add bx, dwA
    add ch, bA1
    add cl, bA2

    push 0
    call ExitProcess

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x11223344	unsigned int
ebx	0x44558899	unsigned int
ecx	0x8899dfff	unsigned int

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

```

ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; декремент
    mov eax, 00112233h
    mov ebx, 44556677h
    mov ecx, 8899AABBh
    add ddA, eax
    add dwA, bx
    add bA1, ch
    add bA2, cl

    push 0
    call ExitProcess
main ENDP
end main

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11111111	unsigned long
dwA	0x2222	unsigned short
bA1	0x33 '3'	unsigned char
bA2	0x44 'D'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11223344	unsigned long
dwA	0x8899	unsigned short
bA1	0xdd 'Э'	unsigned char
bA2	0xff 'я'	unsigned char

```

.stack 4096                ; сегмент стека объемом 4096
.const                     ; сегмент констант
.data                     ; сегмент данных
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; сумма
    mov eax, 00112233h
    mov ebx, 44556677h
    mov ecx, 8899AABBh
    add ax, 0011h
    add bh, 77h
    add bl, 22h
    add ddA, 11h
    add dwa, 12h
    add bA1, 13h
    add bA2, 14h

    push 0
    call ExitProcess
main ENDP
end main

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11111111	unsigned long
dwa	0x2222	unsigned short
bA1	0x33 '3'	unsigned char
bA2	0x44 'D'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112244	unsigned int
ebx	0x4455dd99	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11111122	unsigned long
dwa	0x2234	unsigned short
bA1	0x46 'F'	unsigned char
bA2	0x58 'X'	unsigned char

; конец процедуры

; конец модуля, main - точка в

3.3. Команда SUB вычитает *операнд-источник* из *операнда получателя* данных. Процессор заменяет ее на команду *сложения* с отрицательным числом (отрицательные числа представляются в дополнительном коде).
Длины операндов должны быть равны.

```

dw 0000h
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
; сумма
mov eax, 0011223
mov ebx, 4455667
mov ecx, 8899AAB

sub eax, ebx
sub ebx, ddA
sub ch, 2h
sub cl, bA1

sub ddA, ebx
sub dwA, bx
sub bA1, b1

push 0
call ExitProcess
main ENDP
end main

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11111111	unsigned long
dwA	0x2222	unsigned short
bA1	0x33 '3'	unsigned char
bA2	0x44 'D'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
eax	0xbbbbbbbbc	unsigned int
ebx	0x33445566	unsigned int
ecx	0x8899a888	unsigned int
ddA	0x11111111	unsigned long
dwA	0x2222	unsigned short
bA1	0x33 '3'	unsigned char
bA2	0x44 'D'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
eax	0xbbbbbbbbc	unsigned int
ebx	0x33445566	unsigned int
ecx	0x8899a888	unsigned int
ddA	0xddccbbab	unsigned long
dwA	0xccbc	unsigned short
bA1	0xcd 'H'	unsigned char
bA2	0x44 'D'	unsigned char

3.4. Команда **NEG** изменяет знак числа на противоположный.

```

dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; сумма
    mov eax, 00112233h
    mov ebx, 44556677h
    mov ecx, 8899AABBh
    neg eax
    neg bx
    neg ch
    neg cl
    neg ddA
    neg dwA
    neg bA1

    push 0
    call ExitProcess
main ENDP
end main

```

Контрольные значения 1

Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11111111	unsigned long
dwA	0x2222	unsigned short
bA1	0x33 '3'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
eax	0xffeeddcd	unsigned int
ebx	0x44559989	unsigned int
ecx	0x88995645	unsigned int
ddA	0xeefefefef	unsigned long
dwA	0xddde	unsigned short
bA1	0xcd 'H'	unsigned char

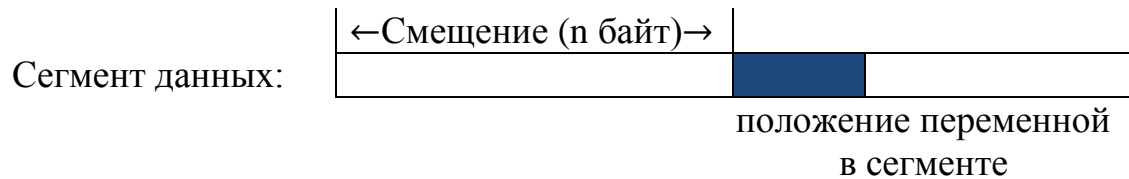
```

; конец процедуры
; конец модуля, main - точка входа

```

4. Смещение в сегменте данных

Опреатор **OFFSET** возвращает смещение метки данных относительно начала сегмента. Под смещением понимается то количество байтов, которое отделяет метку данных от начала сегмента:



```
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bBl byte 00h
.code
main PROC
    ; смещение
    mov eax, OFFSET ddA
    mov ebx, OFFSET dwA
    mov ecx, OFFSET bA2
    push 0
    call ExitProcess
```

Имя	Значение	Тип
eax	0x001b4000	unsigned int
ebx	0x001b4008	unsigned int
ecx	0x001b400d	unsigned int

5. Выравнивание на границу 2, 4, 8, 16

```
.data                                ; сегмент данных
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
      ; align 2
bA2 byte 44h
bBh byte 00h
      ; align 16
bB1 byte 00h
.code
main PROC
      ; смещение
mov eax, OFFSET bA2
mov ebx, OFFSET bB1
```

Имя	Значение	Тип
eax	0x00c2400d	unsigned int
ebx	0x00c2400f	unsigned int

Директива **ALIGN** используется для выравнивания адреса переменной на границу *байта* (1 байта), *слова* (2 байта), *двойного слова* (4 байта), *учетверенного слова* или параграфа (16 байтов).

```
.data                                ; сегмент данных
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
bA1 byte 33h
align 2                               ; выравнивание на 2 байта
bA2 byte 44h
bBh byte 00h
align 16                             ; выравнивание на 16 байтов
bB1 byte 00h
.code                                ; сегмент кода
main PROC                           ; начало процедуры
      ; смещение
mov eax, OFFSET bA2
mov ebx, OFFSET bB1
```

Имя	Значение	Тип
eax	0x00c7400e	unsigned int
ebx	0x00c74010	unsigned int

6. Директива LABEL (определение дополнительных имен)

Директива LABEL определяет в программе метку и назначает ей нужный атрибут длины без распределения памяти под переменную. Используется для определения дополнительных имен (псевдонимов).

```
.const                                ; сегмент констант
.data                                ; сегмент данных
lab3 label dword
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
lab1 label byte
lab2 label dword
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; LABEL
    movzx eax, lab1
    mov     ebx, lab2
    mov     ecx, lab3 + 8
```

Контрольные значения 1

Имя	Значение
eax	0x00000033
ebx	0x00004433
ecx	0x00002222

7. Оператор TYPE (возвращает размер указанной переменной в байтах)

```

; segment constant
; сегмент констант

; segment data
; сегмент данных

.data
lab3 label dword
ddA dd 11111111h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
lab1 label byte
lab2 label dword
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ; TYPE
    mov eax, type ddA
    mov ebx, type lab1
    mov ecx, type bB1
main ENDP

```

Контрольные значения 1

Имя	Значение
eax	0x00000004
ebx	0x00000001
ecx	0x00000001

8. Оператор PTR

Если размеры операндов не совпадают, то можно переопределить размер операнда. Например, для загрузки коротких переменных в больший регистр. Для такого уточнения можно использовать оператор переопределения типа **PTR**. Типы могут быть **byte**, **word**, **dword**.

```
.const                                ; сегмент констант
.data                                 ; сегмент данных
lab3 label dword
ddA dd 11223344h
ddB dd 00000000h
dwA dw 2222h
dwB dw 0000h
lab1 label byte
lab2 label dword
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ;TYPE
    mov eax, 0h
    mov ebx, 0h
    mov ecx, 0h
    mov eax, dword ptr bA2
    mov bx, word ptr ddA
    mov ch, byte ptr lab2
```

Контрольные значения 1

Имя	Значение
eax	0x00000044
ebx	0x00003344
ecx	0x00003300

9. Операторы SIZEOF и LENGTHOF

Опреатор LENGTHOF определяет количество элементов в массиве, перечисленного в одной строке.

Опреатор SIZEOF возвращает значение равное произведению значений, возвращаемых операторами LENGTHOF и TYPE.

```
dwB dw 0000h

mdwA dw 10 dup(?)
mbyA byte 15 dup(?)

lab1 label byte
lab2 label dword
bA1 byte 33h
bA2 byte 44h
bBh byte 00h
bB1 byte 00h
.code
main PROC
    ;TYPE
    mov eax, lengthof mdwA
    mov ebx, type mdwA
    mov ecx, sizeof mdwA

    mov eax, lengthof mbyA
    mov ebx, type mbyA
    mov ecx, sizeof mbyA

    push 0
```

Контрольные значения 1

Имя	Значение	Тип
eax	0x0000000a	unsigned int
ebx	0x00000002	unsigned int
ecx	0x00000014	unsigned int

Контрольные значения 1

Имя	Значение	Тип
eax	0x0000000f	unsigned int
ebx	0x00000001	unsigned int
ecx	0x0000000f	unsigned int

Память 2

Адрес: 0x00A64000

0x00A64000 ?? ?? ?? ?? ?? ?? ?? ??