

Введение в язык Ассемблера

1. Формат команды ассемблера:

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

Основные типы операндов:

- непосредственно 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 констант
7  .DATA                              ; сегмент данных
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 процессору
28 call ExitProcess                   ; так завершается лк
29 main ENDP                          ; конец процедуры
30 end main                           ; конец модуля main

```

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 проц
25  call ExitProcess      ; так завершается л
26  main ENDP            ; конец процедуры
27  end main              ; конец модуля main

```

Контрольные значения 1		
Имя	Значение	Тип
ebx	0x00010203	unsigned int

Контрольные значения 1		
Имя	Значение	Тип
ebx	0x00010001	unsigned int

Контрольные значения 1		
Имя	Значение	Тип
ebx	0x00011101	unsigned int

Контрольные значения 1		
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
main PROC    ; сегмент кода
              ; точка входа main

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

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

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. При этом оставшиеся *неопределенными битами* регистра-получателя (это старшие 16 или 24 бита) сбрасываются в ноль.

```

.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
bBl byte 00h

.CODE
main PROC    ; сегмент кода
              ; начало процедуры

mov ecx, 0xffffffffh
movzx ecx, dwA ; dwA -> ecx
mov ecx, 0xffffffffh
movzx ecx, bA1 ; bA1 -> ecx
mov ecx, 0xffffffffh
movzx cx, bA2 ; bA2 -> ch

push 0 ; код возврата (параметр
call ExitProcess ; завершение процесса Wi
main ENDP ; конец процедуры
end main ; конец модуля, точка вх
    
```

Имя	Значение	Тип
ecx	0x0fffffff	unsigned int

Имя	Значение	Тип
ecx	0x00004455	unsigned int

Имя	Значение	Тип
ecx	0x0fffffff	unsigned int

Имя	Значение	Тип
ecx	0x00000066	unsigned int

Имя	Значение	Тип
ecx	0x0fffffff	unsigned int

Имя	Значение	Тип
ecx	0x0fff0077	unsigned int

```

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

```

.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

```

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 битами регистра-получателя (это старшие 16 или 24 бита) дополняет значением *знакового бита* исходного операнда.

The screenshot shows a debugger window with assembly code on the left and several register windows on the right. The assembly code includes instructions for moving data into registers and then using MOVSX to extend the data. The register windows show the state of eax, ebx, and ecx at different points in the execution. A callout box highlights the bit extension process: the source register's sign bit (1) is used to fill the upper bits of the destination register.

Assembly code (left):

```

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

```

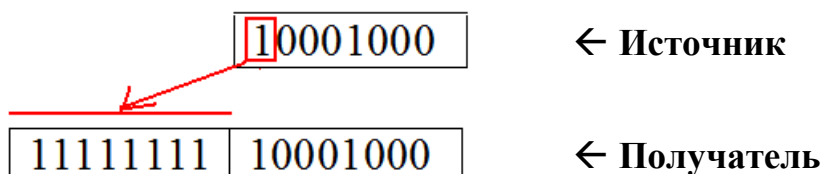
Register windows (right):

- Window 1: eax=0x00112233, ebx=0x0011f2f3, ecx=0x44556677
- Window 2: eax=0xfffff2f3, ebx=0x0011f2f3, ecx=0x44556677
- Window 3: eax=0xffffffff, ebx=0x0011f2f3, ecx=0x44556677
- Window 4: eax=0xffffffff, ebx=0x0011f2f3, ecx=0x44556677
- Window 5: eax=0x00004455, ebx=0x0011f2f3, ecx=0x44556677
- Window 6: eax=0xfffff888, ebx=0x0011f2f3, ecx=0x44556677

Callout box (bottom right):

10001000 (Source bit)

11111111 10001000 (Destination register state)



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»:

The screenshot displays an assembly window on the left and a series of four register window snapshots on the right, all titled "Контрольные значения 1".

Assembly Code:

```
dwB dw 0000h
bA1 byte 88h
bA2 byte 77h
bBh byte 00h
bBl 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
```

Register Window Snapshots:

- Snapshot 1:** Initial state. eax: 0x00112233, ebx: 0x44556677, ecx: 0x8899aabb.
- Snapshot 2:** After `xchg eax, ebx`. eax: 0x44556677, ebx: 0x00112233, ecx: 0x8899aabb.
- Snapshot 3:** After `xchg ax, cx`. eax: 0x4455aabb, ebx: 0x00112233, ecx: 0x88996677.
- Snapshot 4:** After `xchg ah, bh` and `xchg al, bl`. eax: 0x445522bb, ebx: 0x0011aa33, ecx: 0x88996677.
- Snapshot 5:** After `xchg al, bA2`. eax: 0x44552233, ebx: 0x0011aabb, ecx: 0x88996677.

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

Assembly code snippet:

```

.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

```

Multiple screenshots of the "Контрольные значения 1" (Control Values 1) window showing the state of registers and memory locations during execution:

- Initial State:**

Имя	Значение	Тип
eax	0x00112233	unsigned int
ddA	0xCCCCCCCC	unsigned long
dwA	0x4455	unsigned short
bA1	0x88 '€'	unsigned char
bA2	0x77 'w'	unsigned char
- After `mov eax, 00112233h`:**

Имя	Значение	Тип
eax	0xCCCC4455	unsigned int
ddA	0x00112233	unsigned long
dwA	0xCCCC	unsigned short
- After `xchg eax, ddA`:**

Имя	Значение	Тип
eax	0xCCCC8855	unsigned int
ddA	0x00112233	unsigned long
dwA	0xCCCC	unsigned short
bA1	0x44 'D'	unsigned char
bA2	0x77 'w'	unsigned char
- After `xchg ax, dwA`:**

Имя	Значение	Тип
eax	0xCCCC8877	unsigned int
ddA	0x00112233	unsigned long
dwA	0xCCCC	unsigned short
bA1	0x44 'D'	unsigned char
bA2	0x55 'U'	unsigned char
- After `xchg ah, bA1`:**

Имя	Значение	Тип
eax	0x0011CCCC	unsigned int
ddA	0xCCCC8877	unsigned long
dwA	0x2233	unsigned short
bA1	0x44 'D'	unsigned char
bA2	0x55 'U'	unsigned char
- After `xchg al, bA2`:**

Имя	Значение	Тип
eax	0x001144CC	unsigned int
ddA	0xCCCC8877	unsigned long
dwA	0x2233	unsigned short
bA1	0xCC 'M'	unsigned char
bA2	0x55 'U'	unsigned char

3. Целочисленное сложение и вычитание

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

INC: операнд регистр:

The image shows a debugger window with assembly code on the left and a series of 'Контрольные значения 1' (Control Values 1) windows on the right, illustrating the state of the EAX register during an increment operation.

Assembly Code:

```
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
```

Control Values 1 Windows:

Имя	Значение	Тип
eax	0x00112233	unsigned int
eax	0x00112234	unsigned int
eax	0x0011ffff	unsigned int
eax	0x00110000	unsigned int
eax	0x0011ffff	unsigned int
eax	0x001100ff	unsigned int
eax	0x0011ffff	unsigned int
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 409
.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	0xCCCCCCCCh	unsigned long
dwA	0x4455	unsigned short
bA1	0x88 '€'	unsigned char

Контрольные значения 1

Имя	Значение	Тип
ddA	0xCCCCCCCd	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

```

.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
; декремент
dec ddA
dec dwA
dec bA1

push 0
call ExitProcess          ; так должен заканчиваться лк
main ENDP                 ; конец процедуры
end main                   ; конец модуля, main - точка

```

Контрольные значения 1

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

Контрольные значения 1

Имя	Значение	Тип
ddA	0xffffffff	unsigned long
dwA	0x4454	unsigned short
bA1	0x87 '+'	unsigned char

- 3.2. Команда **ADD** прибавляет операнд-источник к операнду получателю. 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	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
bBl 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
; конец процедуры
end main                  ; конец модуля, main - точка в

```

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

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.

```

dwB 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

```

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

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

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, 8899AAB8h

neg eax
neg bx
neg ch
neg cl
neg ddA
neg dwA
neg bA1

push 0
call ExitProcess
main ENDP
end main

```

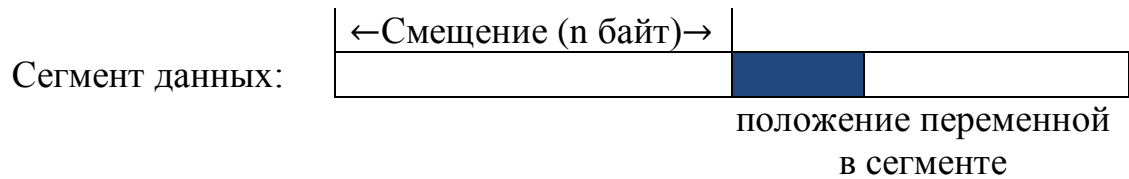
Имя	Значение	Тип
eax	0x00112233	unsigned int
ebx	0x44556677	unsigned int
ecx	0x8899aabb	unsigned int
ddA	0x11111111	unsigned long
dwA	0x2222	unsigned short
bA1	0x33 '3'	unsigned char

Имя	Значение	Тип
eax	0xffeeddcd	unsigned int
ebx	0x44559989	unsigned int
ecx	0x88995645	unsigned int
ddA	0xeeeeeeef	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 относительно 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

```

.586 ; система команд (процессор Pentium)
.model flat,stdcall ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компоновщику: компоновать с kernel32.lib
ExitProcess PROTO :DWORD ; прототип функции
.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
; 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

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	Тип
eax	0x00c2400d	unsigned int
ebx	0x00c2400f	unsigned int

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

```

.586 ; система команд (процессор Pentium)
.model flat,stdcall ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компоновщику: компоновать с kernel32.lib
ExitProcess PROTO :DWORD ; прототип функции
.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
align 2 ; выравнивание на границу 2
bA2 byte 44h
bBh byte 00h
align 16 ; выравнивание на границу 16
bB1 byte 00h
.code ; сегмент кода
main PROC ; начало процедуры
; смещение
mov eax, OFFSET bA2
mov ebx, OFFSET bB1

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	Тип
eax	0x00c7400e	unsigned int
ebx	0x00c74010	unsigned int

6. Директива LABEL (определение дополнительных имен)

```
.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
push 0 ; код возврата проце
call ExitProcess ; так должен заканч
main ENDP ; конец процедуры
end main ; конец модуля, mai
```

Имя	Значение
eax	0x00000033
ebx	0x00004433
ecx	0x00002222

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

```
.586 ; система команд (пр
.model flat,stdcall ; модель памяти, сог
includelib kernel32.lib ; компановщику: комп
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объек
.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
; TYPE
mov eax, type ddA
mov ebx, type lab1
mov ecx, type bB1
push 0 ; код возврата проце
call ExitProcess ; так должен заканч
main ENDP ; конец процедуры
```

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
push 0
call ExitProcess
main ENDP
```

Имя	Значение
eax	0x00000044
ebx	0x00003344
ecx	0x00003300

Адрес	Значение
0x00A64000	0x00000000

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.

The screenshot shows an assembly editor with the following code:

```
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
call ExitProcess
main ENDP
end main
```

Two 'Контрольные значения 1' (Control Values 1) windows are overlaid on the code. The first window shows the state after the first three instructions:

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

The second window shows the state after the next three instructions:

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

Below the windows, a 'Память 2' (Memory 2) window shows the address 0x00A64000 and a list of memory addresses with their corresponding values (mostly unknown, indicated by '??').