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

1. Косвеная 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
                          *(&ddMD+0)
                           push 0
                           *(&ddMD+2)
                                                (параметр ExitProcess )
  call ExitProcess
                                          ....ься 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
             Имя
                                Значение
               1
  push 0
                                         процесса (параметр ExitProcess )
               *(&dwMD+1)
   call ExitP
                                         заканчиваться любой процесс Windows
               *(&dwMD+2)
                                3
main ENDP
                                        туры
end main
                           ; конец модуля, main - точка входа
```

```
bMS
      byte 1,2,3,4,5,6,7
      byte 7 dup(?)
bMD
.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 байта 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
                           <u>al-> по адресу</u>в edi
  mov [edi], al
                 Имя
                                   Значени
                   *(&bMD+0)
                                   1 '\x1'
                   2 '\x2'
  push 0
                          call ExitProcess
main ENDP
                          ; конец процедуры
end main
                          ; конец модуля, main - точка входа
```

```
; модель памяти, соглашение о вызовах
.model flat,stdcall
includelib kernel32.lib ; компановщику: 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
                         Имя
                                             Значение
  add eax, [esi]
                           eax
  add esi,4
  add eax, [esi]
  push 0
                            ; код возрата процесса (параметр ExitProcess )
  call ExitProcess
                            ; так должен заканчиваться любой процесс Windows
main ENDP
                            ; конец процедуры
end main
                            ; конец модуля, main - точка входа
```

## Операнды с индексом, синтаксис первой формы представления:

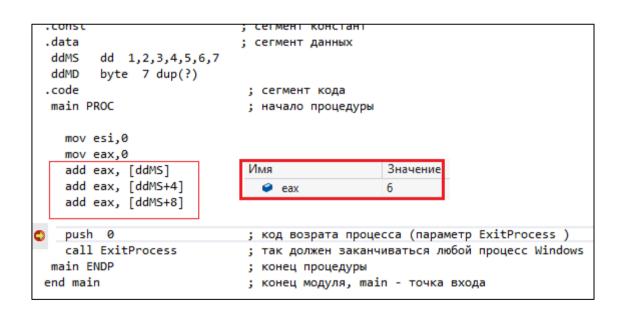
имя\_переменной[индексный\_регистр]

```
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]
                                   Значение оцесса (параметр ExitProcess )
  push 0
                Имя
  call ExitProc
                                             нчиваться любой процесс Windows
                 eax
main ENDP
end main
                            ; конец модуля, main - точка входа
```

## Операнды с индексом, сиинтаксис второй формы представления:

[имя переменной+индексный регистр]

```
.stack 4096
                           ; сегмент стека объемом 4096
                           ; сегмент констант
.const
.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]
                                           роцесса (параметр ExitProcess )
 push 0
  call ExitProcess
                            ; так должен заканчиваться любой процесс Windows
main ENDP
                            ; конец процедуры
end main
                            ; конец модуля, main - точка входа
```



#### 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
                            ; указатель на ddMD
pddMD dd offset ddMD
.code
                            ; сегмент кода
main PROC
                            ; начало процедуры
  mov esi,pddMS
  mov edi,pddMD
  mov eax, [esi]
  mov [edi], eax
                                          Значение
                      Имя
  add esi,4

← pddMS

                                          9453568
  add edi,4
                                          9453596
  mov eax, [esi]

♠ pddMD

  mov [edi], eax
                        *(&ddMD+1)
                                          2
  add esi,4
                        *(&ddMD+2)
                                          3
  add edi,4
  mov eax, [esi]
  mov [edi], eax
  push 0
                            ; код возрата процесса (параметр ExitProcess )
  call ExitProcess
                            ; так должен заканчиваться любой процесс Windows
main ENDP
                            ; конец процедуры
end main
                            ; конец модуля, main - точка входа
```

# 3. Команды переходов

## 3.1 Синтаксис:

**JMP** 

метка перехода

```
; 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 возрата процесса (параметр ExitProcess )
   call ExitProcess
                             ; так должен заканчиваться любой процесс Windows
 main ENDP
                             ; конец процедуры
```

#### 3.2 Синтаксис:



Регистр ЕСХ используется в качестве счетчика.

В ЕСХ загружается количество повторений цикла.

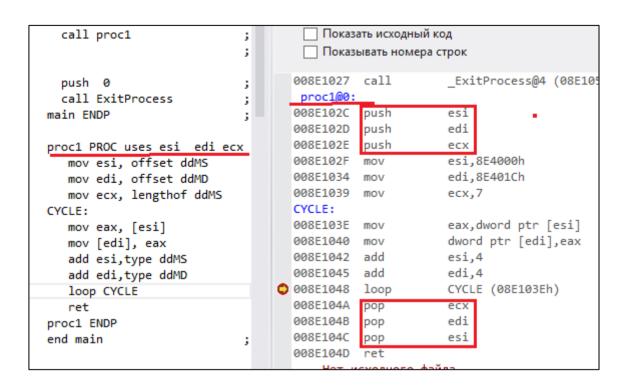
- На каждом шаге выполнения цикла значение ЕСХ автоматически уменьшается на 1 и сравнивается с 0.
- Если не ноль переходим по метке
- В противном случае выпоняется следующая по порядку команда.

```
.data
                           ; сегмент данных
ddMS
       dd 1,2,3,4,5,6,7
       dd 7 dup(?)
ddMD
.code
                            ; сегмент кода
main PROC
                            ; начало процедуры
                                             Имя
                                                                 Знач
  mov esi, offset ddMS
                                               1
  mov edi, offset ddMD
                                               *(&ddMD+1)
                                                                 2
                                               *(&ddMD+2)
                                                                 3
                            ; счетчик
  mov ecx, 7
CYCLE:
                            ; метка
                                               *(&ddMD+3)
                                                                 4
                                                                 5
                                                 *(&ddMD+4)
  mov eax, [esi]
  mov [edi], eax
                                                 *(&ddMD+5)
                                                                 6
  add esi,4
                                                  *(&ddMD+6)
  add edi,4
  loop CYCLE
                            ; --ecx, if (ecx != 0) goto CYCLE
                            ; код возрата процесса (параметр ExitProce
  push 0
  call ExitProcess
                            ; так должен заканчиваться любой процесс \
main ENDP
                            ; конец процедуры
end main
                            ; конец модуля, main - точка входа
```

```
.data
                           ; сегмент данных
 ddMS
       dd 1,2,3,4,5,6,7
 ddMD dd 7 dup(?)
.code
                            ; сегмент кода
 main PROC
                            ; начало процедуры
                                       Имя
                                                          Знач
   mov esi, offset ddMS
                                         1
   mov edi, offset ddMD
                                         *(&ddMD+1)
                                                          2
                                         *(&ddMD+2)
                                                          3
   mov ecx, lengthof ddMS
                            ; счетчик
                                         *(&ddMD+3)
CYCLE:
                            ; метка
                                         5
   mov eax, [esi]
   mov [edi], eax
                                         *(&ddMD+5)
                                                          6
  add esi,type ddMS
                                         *(&ddMD+6)
                                                          7
   add edi, type ddMD
   loop CYCLE
                            ; --ecx, if (ecx != 0) goto CYCLE
                           ; код возрата процесса (параметр ExitProcess )
   push 0
   call ExitProcess
                           ; так должен заканчиваться любой процесс Windows
 main ENDP
                           ; конец процедуры
end main
                            ; конец модуля, main - точка входа
```

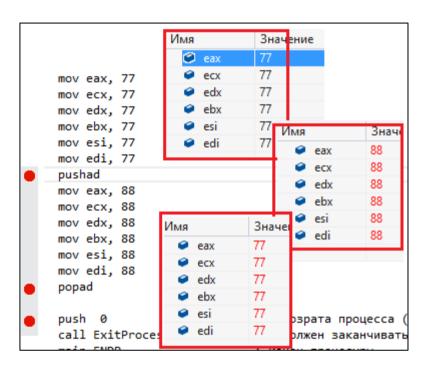
```
; сегмент данных
.data
       dd 1,2,3,4,5,6,7
ddMS
ddMD
       dd 7 dup(?)
.code
                           ; сегмент кода
main PROC
                           ; начало процедуры
  call proc1
                           ; поместить в стек адрес следующей
                           ; команды и jmp proc1
                                                  Имя
                                                    1
                          ; код возрата процесса
  push 0
                                                    *(&ddMD+1)
                                                                     2
  call ExitProcess
                           ; так должен заканчивать
                                                    3
main ENDP
                           ; конец процедуры
                                                    4
                                                                     5
                                                    *(&ddMD+4)
proc1 PROC
                         ; начало процедуры
  mov esi, offset ddMS
                                                    *(&ddMD+5)
                                                                     6
  mov edi, offset ddMD
                                                    *(&ddMD+6)
                                                                     7
  mov ecx, lengthof ddMS
                           ; счетчик
CYCLE:
                           ; метка
  mov eax, [esi]
  mov [edi], eax
  add esi, type ddMS
  add edi, type ddMD
                           ; --ecx, if (ecx != 0) goto CYCLE
  loop CYCLE
  ret
                           ; рор адрес возврата и јтр
proc1 ENDP
                           ; конец процедуры
end main
                           ; конец модуля, main - точка входа
```

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



```
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
                           ddesp0
                                           0x00b7fbd4
mov ddesp1, esp
push ax
                           ddesp1
                                           0x00b7fbd0
mov ddesp2, esp
                           g ddesp2
                                           0x00b7fbce
push 0101h
                                           0x00b7fbca
                           ddesp3
mov ddesp3, esp
                                           0x00b7fbc6
                           ddesp4
push 1
                          ddesp5
                                           0x00b7fbc4
mov ddesp4, esp
                          g ddesp6
                                           0x00b7fbc0
push word ptr 1
                           ddesp7
                                           0x00b7fbbe
mov ddesp5, esp
push ddd
mov ddesp6, esp
push ddw
mov ddesp7, esp
```





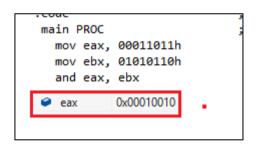
# 5. Логические команды AND, OR, XOR, NOT

Синтаксис:

AND получатель источник

Таблица истинности для операции логичекого И:

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



```
лет процедуры по темент кода процедуры по темент кода процедуры по темент кода процедуры по темент кода по те
```

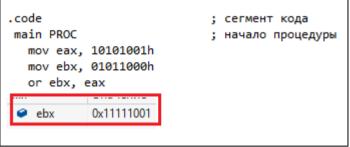
```
.data
                           ; сегмент данных
ddMS dd 1,2,3,4,5,6,7
ddMD dd 7 dup(?)
ddAND dd 11111111h
dwAND dw 1111h
bAND
       byte 11111111b
.code
                            ; сегмент кода
main PROC
                            ; начало процедуры
  mov eax, 10101001h
  and ddAND, eax
  and eax, ddAND
  and dwAND, ax
  and ax, dwAND
  and al, bAND
  and bAND, ah
```

# Синтаксис:



Таблица истинности для операции логичекого ИЛИ:

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



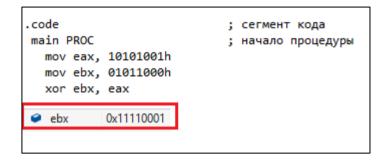
```
ddAND dd 11111111h
dwAND dw 1111h
bAND byte 11111111b
.code
                            ; сегмент кода
main PROC
                            ; начало процедуры
  mov eax, 10101001h
  or ddAND, eax
  or eax, ddAND
  or dwAND, ax
  or ax, dwAND
  or al, bAND
  or bAND, ah
  or eax, 2
  or ddAND, 2
  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



## Синтаксис:

NOT операнд

Таблица истинности для операции отрицания:

X	NOT X
0	1
0	1
1	0
1	0

