

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ»
ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»
Кафедра МО ЭВМ**

**ОТЧЁТ
по лабораторной работе № 2
по дисциплине «Организация ЭВМ и систем»
Тема: Изучение режимов адресации и формирования
исполнительного адреса**

Студент гр. 1303

Преподаватель

Ягодаров М.А.

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

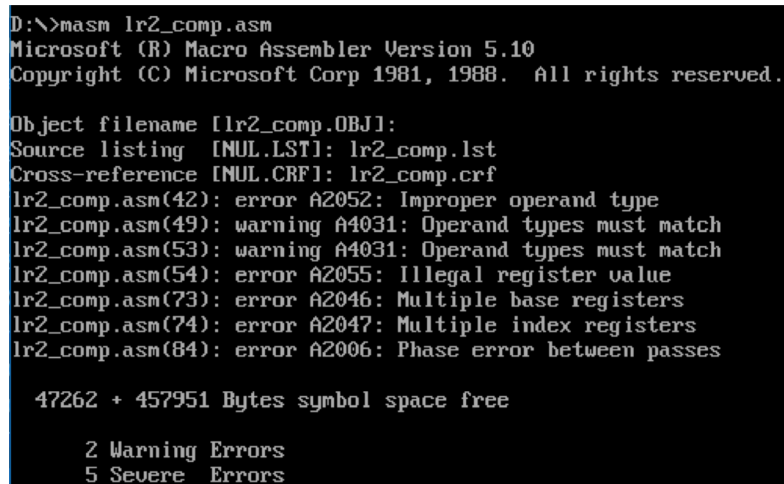
Изучить режимы адресации. Ознакомиться с работой с ними на языке программирования Ассемблер.

Задание.

Лабораторная работа №2 предназначена для изучения режимов адресации, использует готовую программу lr2_comp.asm на Ассемблере, которая в автоматическом режиме выполняться не должна, так как не имеет самостоятельного функционального назначения, а только тестирует режимы адресации. Поэтому её выполнение должно производиться под управлением отладчика в пошаговом режиме.

Выполнение работы.

1. Изменены исходные значения vec1, vec2, matr согласно варианту.
2. Была произведена попытка трансляции файла с получением ошибок, продемонстрированных на рисунке 1.



```
D:\>masm lr2_comp.asm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [lr2_comp.OBJ]:
Source listing [NUL.LST]: lr2_comp.lst
Cross-reference [NUL.CRF]: lr2_comp.crf
lr2_comp.asm(42): error A2052: Improper operand type
lr2_comp.asm(49): warning A4031: Operand types must match
lr2_comp.asm(53): warning A4031: Operand types must match
lr2_comp.asm(54): error A2055: Illegal register value
lr2_comp.asm(73): error A2046: Multiple base registers
lr2_comp.asm(74): error A2047: Multiple index registers
lr2_comp.asm(84): error A2006: Phase error between passes

47262 + 457951 Bytes symbol space free

2 Warning Errors
5 Severe Errors
```

Рисунок 1 – Ошибки при первой трансляции файла.

3. Закомментированы следующие строки:
(a) `mov mem3, [bx]` — запрещено в качестве операндов команды `mov` использовать одновременно две ячейки в памяти; один из операндов должен быть либо регистром, либо значением. (error A2052: Improper operand type)

- (b) `mov cx, vec2[di]` — операнды имеют разную длину: `cx` — 2 байта, `vec2[di]` — 1 байт. (warning A4031: Operand types must match)
- (c) `mov cx, matr[bx][di]` — операнды имеют разную длину: `cx` — 2 байта, `matr[bx][di]` — 1 байт. (warning A4031: Operand types must match)
- (d) `mov ax, matr[bx*4][di]` — масштабирование регистра `bx` при базово–индексированной адресации. (error A2055: Illegal register value)
- (e) `mov ax, matr[bp+bx]` — оба регистра являются базовыми. (error A2046: Multiple base registers)
- (f) `mov ax, matr[bp+di+si]` — два индексных регистра. (error A2047: Multiple index registers)
- (g) Добавлены две операции `por ax` для успешного завершения программы.

4. Проведена успешная трансляция файла.

5. Начальное содержимое регистров:

`CS = 1A0A; DS = 19F5; ES = 19F5; SS = 1A05;`

Таблица 1 – Результат прогона программы `lr2_comp.exe` в отладчике.

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	<code>push DS</code>	1E	IP = 0000 SP = 0018 Stack: +0 0000	IP = 0001 SP = 0016 Stack: +0 19F5
0001	<code>sub AX, AX</code>	2B C0	IP = 0001 AX = 0000	IP = 0003 AX = 0000

0003	push AX	50	IP = 0003 SP = 0016 Stack: +0 19F5 +2 0000	IP = 0004 SP = 0014 Stack +0 0000 +2 19F5
0004	mov AX, 1A07	B8 07 1A	IP = 0004 AX = 0000	IP = 0007 AX = 1A07
0007	mov DS, AX	8E D8	IP = 0007 DS = 19F5	IP = 0009 DS = 1A07
0009	mov AX, 01F4	B8 F4 01	IP = 0009 AX = 1A07	IP = 0009 AX = 01F4
000C	mov CX, AX	8B C8	IP = 000C CX = 00AE	IP = 000E CX = 01F4
000E	mov BL, 24	B3 24	IP = 000E	IP = 0010
0010	mov BH, CE	B7 CE	IP = 0010 BX = 0024	IP = 0012 BX = CE24
0012	mov [0002], FFCE	C7 06 02 00 CE FF	IP = 0012 DS:0002 = 00 DS:0003 = 00	IP = 0018 DS:0002 = CE DS:0003 = FF
0018	mov BX, 0006	BB 0E 00	IP = 0018 BX = CE24	IP = 001B BX = 0006
001B	mov [0000], AX	A3 00 00	IP = 001B DS:0000 = 00 DS:0001 = 00	IP = 001E DS:0000 = F4 DS:0001 = 01
001E	mov AL, [BX]	8A 07	IP = 001E AX = 01F4	IP = 0020 AX = 011F
0020	mov AL, [BX+03]	8A 47 03	IP = 0020 AX = 011F	IP = 0023 AX = 0122
0023	mov CX, [BX+03]	8B 4F 03	IP = 0023 CX = 01F4	IP = 0026 CX = 2622
0026	mov DI, 0002	BF 02 00	IP = 0026 DI = 0000	IP = 0029 DI = 0002

0029	mov AL, [000E+DI]	8A 85 0E 00	IP = 0029 AX = 0122	IP = 002D AX = 01CE
002D	mov BX, 0003	BB 03 00	IP = 002D BX = 0006	IP = 0030 BX = 0003
0030	mov AL, [0016+BX+DI]	8A 81 16 00	IP = 0030 AX = 01CE	IP = 0034 AX = 01FF
0034	mov AX, 1A07	B8 07 1A	IP = 0034 AX = 01FF	IP = 0037 AX = 1A07
0037	mov EX, AX	8E C0	IP = 0037 ES = 19F5	IP = 0039 ES = 1A07
0039	mov AX, ES:[BX]	26 8B 07	IP = 0039 AX = 1A07	IP = 003C AX = 00FF
003C	mov AX, 0000	B8 00 00	IP = 003C AX = 00FF	IP = 0041 AX = 0000
003F	mov ES, AX	8E C0	IP = 003F ES = 1A07	IP = 0041 ES = 0000
0041	push DS	1E	IP = 0041 SP = 0014 Stack: +0 0000 +2 19F5 +4 0000	IP = 0042 SP = 0012 Stack: +0 1A07 +2 0000 +4 19F5
0042	pop ES	07	IP = 0042 SP = 0012 ES = 0000 Stack: +0 1A07 +2 0000 +4 19F5	IP = 0043 SP = 0014 ES = 1A07 Stack: +0 0000 +2 19F5 +4 0000
0043	mov CX, ES:[BX-1]	26 8B 4F FF	IP = 0043 CX = 2622	IP = 0047 CX = FFCE

0047	xchg AX, CX	91	IP = 0047 AX = 0000 CX = FFCE	IP = 0048 AX = FFCE CX = 0000
0048	mov DI, 0002	BF 02 00	IP = 0048 DI = 0002	IP = 004B DI = 0002
004B	mov ES:[BX+DI], AX	26 89 01	IP = 004B DS:0005 = 00 DS:0006 = 15	IP = 004E DS:0005 = CE DS:0006 = FF
004E	mov BP, SP	8B EC	IP = 004E BP = 0000	IP = 0050 BP = 0014
0050	push [0000]	FF 36 00 00	IP = 0050 SP = 0014 Stack: +0 0000 +2 19F5 +4 0000	IP = 0054 SP = 0012 Stack: +0 01F4 +2 0000 +4 19F5
0054	push [0002]	FF 36 02 00	IP = 0054 SP = 0012 Stack: +0 01F4 +2 0000 +4 19F5 +6 0000	IP = 0058 SP = 0010 Stack: +0 FFCE +2 01F4 +4 0000 +6 19F5
0058	mov BP, SP	8B EC	IP = 0058 BP = 0014	IP = 005A BP = 0010
005A	mov DX, [BP+02]	8B 56 02	IP = 005A DX = 0000	IP = 005D DX = 01F4

005D	pop AX	58	IP = 005D AX = FFCE SP = 0010 Stack: +0 FFCE +2 01F4 +4 0000 +6 19F5	IP = 005E AX = FFCE SP = 0012 Stack: +0 01F4 +2 0000 +4 19F5 +6 0000
005E	pop AX	58	IP = 005E AX = FFCE SP = 0012 Stack: +0 01F4 +2 0000 +4 19F5	IP = 005F AX = 01F4 SP = 0014 Stack: +0 0000 +2 19F5 +4 0000
005F	ret Far	CB	IP = 005F CS = 1A0A SP = 0014 Stack: +0 0000 +2 19F5	IP = 0000 CS = 19F5 SP = 0018 Stack: +0 0000 +2 0000
0000	int 20	CD 20		

Выводы

В ходе лабораторной работы были изучены режимы адресации на языке Ассемблера.

ПРИЛОЖЕНИЕ А

ЛИСТИНГИ ПРОГРАММ

Название файла: lr2comp.lst

Microsoft (R) Macro Assembler Version 5.10

10/30/22 02:03:2

Page 1-1

```
= 0024          _eol equ '$'
= 0002          _ind equ 2
= 01F4          _n1 equ 500
=-0032          _n2 equ -50
```

```
assume cs:code, ds:data, ss:my_stack
```

```
0000          my_stack segment stack
0000 000C[      dw 12 dup('?')
      003F
      ]
```

```
0018          my_stack ends
```

```
0000          data segment
0000 0000          mem1 dw 0
0002 0000          mem2 dw 0
0004 0000          mem3 dw 0
0006 1F 20 21 22 26 25      vec1 db 31, 32, 33, 34, 38, 37, 36, 35
      24 23
000E 32 3C CE C4 46 50      vec2 db 50, 60, -50, -60, 70, 80, -70, -80
      BA B0
0016 FC FD 07 08 FE FF      matr db -4, -3, 7, 8, -2, -1, 5, 6, -8, -7,
      3, 4, -6, -5, 1, 2
      05 06 F8 F9 03 04
      FA FB 01 02
```

```
0026          data ends
```

```
0000          code segment
```

```
          ; Main procedure
0000          main proc far
```

```
0000 1E          push ds
0001 2B C0          sub ax, ax
0003 50          push ax
0004 B8 ---- R      mov ax, data
0007 8E D8          mov ds, ax
```

```
          ; Register addressing
0009 B8 01F4        mov ax, _n1
```



```

000C 8B C8                mov cx, ax
000E B3 24                mov bl, _eol
0010 B7 CE                mov bh, _n2

; Direct addressing
0012 C7 06 0002 R FFCE    mov mem2, _n2
0018 BB 0006 R            mov bx, offset vec1
001B A3 0000 R            mov mem1, ax

; Indirect addressing
001E 8A 07                mov al, [bx]
; mov mem3, [bx]

```

```

Microsoft (R) Macro Assembler Version 5.10          10/30/22 02:03:2
                                           Page    1-2

```

```

; Based addressing
0020 8A 47 03            mov al, [bx]+3
0023 8B 4F 03            mov cx, 3[bx]

; Indexed addressing
0026 BF 0002            mov di, _ind
0029 8A 85 000E R        mov al, vec2[di]
; mov cx, vec2[di]

; Addressing with basing and indexing
002D BB 0003            mov bx, 3
0030 8A 81 0016 R        mov al, matr[bx][di]
; mov cx, matr[bx][di]
; mov ax, matr[bx*4][di]

; Segment-based addressing verification
; Redefining a segment
; -----
; - First option
0034 B8 ---- R          mov ax, seg vec2
0037 8E C0              mov es, ax
0039 26: 8B 07          mov ax, es:[bx]
003C B8 0000            mov ax, 0

; - Second option
003F 8E C0              mov es, ax
0041 1E                 push ds
0042 07                 pop es
0043 26: 8B 4F FF        mov cx, es:[bx-1]
0047 91                 xchg cx, ax

; - Third option
0048 BF 0002            mov di, _ind
004B 26: 89 01          mov es:[bx+di], ax

```

```

                                ; - Fourth option
004E 8B EC                      mov bp, sp
                                ; mov ax, matr[bp+bx]
                                ; mov ax, matr[bp+di+si]
                                ; -----

                                ; Using stack segment
0050 FF 36 0000 R              push mem1
0054 FF 36 0002 R              push mem2
0058 8B EC                      mov bp, sp
005A 8B 56 02                  mov dx, [bp]+2

005D 58                        pop ax
005E 58                        pop ax

005F CB                        ret

0060                          main endp
                                ; Main procedure ends
Microsoft (R) Macro Assembler Version 5.10      10/30/22 02:03:2
                                           Page    1-3

```

```

0060                          code ends

                                end main
Microsoft (R) Macro Assembler Version 5.10      10/30/22 02:03:2
                                           Symbols-1

```

Segments and Groups:

N a m e	Length	Align	Combine Class
CODE	0060	PARA	NONE
DATA	0026	PARA	NONE
MY_STACK	0018	PARA	STACK

Symbols:

N a m e	Type	Value	Attr	
MAIN	F PROC	0000	CODE	Length = 0060
MATR	L BYTE	0016	DATA	
MEM1	L WORD	0000	DATA	
MEM2	L WORD	0002	DATA	
MEM3	L WORD	0004	DATA	
VEC1	L BYTE	0006	DATA	

VEC2	L BYTE	000E	DATA
@CPU	TEXT	0101h	
@FILENAME	TEXT	LR2_COMP	
@VERSION	TEXT	510	
_EOL	NUMBER	0024	
_IND	NUMBER	0002	
_N1	NUMBER	01F4	
_N2	NUMBER	-0032	

104 Source Lines

104 Total Lines

19 Symbols

47800 + 459460 Bytes symbol space free

0 Warning Errors

0 Severe Errors

ПРИЛОЖЕНИЕ В

КОД ПРОГРАММ

Название файла: lr2comp.asm

```
_eol equ '$'
_ind equ 2
_n1 equ 500
_n2 equ -50

assume cs:code, ds:data, ss:my_stack

my_stack segment stack
    dw 12 dup('?')
my_stack ends

data segment
    mem1 dw 0
    mem2 dw 0
    mem3 dw 0
    vec1 db 31, 32, 33, 34, 38, 37, 36, 35
    vec2 db 50, 60, -50, -60, 70, 80, -70, -80
    matr db -4, -3, 7, 8, -2, -1, 5, 6, -8, -7, 3, 4, -6, -5, 1, 2
data ends

code segment

; Main procedure
main proc far

    push ds
    sub ax, ax
    push ax
    mov ax, data
    mov ds, ax

; Register addressing
    mov ax, _n1
    mov cx, ax
    mov bl, _eol
    mov bh, _n2

; Direct addressing
    mov mem2, _n2
    mov bx, offset vec1
    mov mem1, ax

; Indirect addressing
    mov al, [bx]
    ; mov mem3, [bx]
```

```

; Based addressing
    mov al, [bx]+3
    mov cx, 3[bx]

; Indexed addressing
    mov di, _ind
    mov al, vec2[di]
    ; mov cx, vec2[di]

; Addressing with basing and indexing
    mov bx, 3
    mov al, matr[bx][di]
    ; mov cx, matr[bx][di]
    ; mov ax, matr[bx*4][di]

; Segment-based addressing verification
; Redefining a segment
; -----
; - First option
    mov ax, seg vec2
    mov es, ax
    mov ax, es:[bx]
    mov ax, 0

; - Second option
    mov es, ax
    push ds
    pop es
    mov cx, es:[bx-1]
    xchg cx, ax

; - Third option
    mov di, _ind
    mov es:[bx+di], ax

; - Fourth option
    mov bp, sp
    ; mov ax, matr[bp+bx]
    ; mov ax, matr[bp+di+si]
; -----

; Using stack segment
    push mem1
    push mem2
    mov bp, sp
    mov dx, [bp]+2

    pop ax
    pop ax

```

```
    ret  
  
main endp  
; Main procedure ends  
  
code ends  
  
end main
```