

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

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

Студентка гр. 9383

\_\_\_\_\_

Чебесова И.Д.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Найти и закомментировать ошибки в предоставленной программе на языке программирования Ассемблер. Изучить различные режимы адресации и формировании исполнительного адреса в языке программирования Ассемблер.

### **Текст задания.**

#### **Вариант 8**

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 ее выполнение должно производиться под управлением отладчика в пошаговом режиме.

В программу введен ряд ошибок, которые необходимо объяснить в отчете по работе, а соответствующие команды закомментировать для прохождения трансляции.

Необходимо составить протокол выполнения программы в пошаговом режиме отладчика по типу таблицы 1 предыдущей лабораторной работы и подписать его у преподавателя.

На защите студенты должны уметь объяснить результат выполнения каждой команды с учетом используемого вида адресации. Результаты, полученные с помощью отладчика, не являются объяснением, а только должны подтверждать ваши объяснения.

### **Исходный код.**

```
; Программа изучения режимов адресации процессора IntelX86
EOL EQU '$'
ind EQU 2
n1 EQU 500
n2 EQU -50

; Стек программы
AStack SEGMENT STACK
```

```

        DW 12 DUP(?)
AStack  ENDS
;Данные программы
DATA    SEGMENT ;
Директивы описания данных
mem1    DW    0
mem2    DW    0
mem3    DW    0
vec1    DB    28,27,26,25,21,22,23,24
vec2    DB    20,30,-20,-30,40,50,-40,-50
matr    DB    -8,-7,3,4,-6,-5,1,2,-4,-3,7,8,-2,-1,5,6
DATA    ENDS

; Код программы
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:AStack

; Головная процедура
Main    PROC  FAR
        push  DS
        sub   AX,AX
        push  AX
        mov   AX,DATA
        mov   DS,AX

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ СМЕЩЕНИЙ
; Регистровая адресация
        mov   ax,n1
        mov   cx,ax
        mov   bl,EOL
        mov   bh,n2
; Прямая адресация
        mov   mem2,n2
        mov   bx,OFFSET vec1
        mov   mem1,ax
; Косвенная адресация
        mov   al,[bx]
        mov   mem3,[bx]
; Базированная адресация
        mov   al,[bx]+3
        mov   cx,3[bx]
; Индексная адресация
        mov   di,ind
        mov   al,vec2[di]
        mov   cx,vec2[di]
; Адресация с базированием и индексированием
        mov   bx,3
        mov   al,matr[bx][di]
        mov   cx,matr[bx][di]
        mov   ax,matr[bx*4][di]

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ
; Переопределение сегмента

```

```

; ----- вариант 1
    mov ax, SEG vec2
    mov es, ax
    mov ax, es:[bx]
    mov ax, 0
; ----- вариант 2
    mov es, ax
    push ds
    pop es
    mov cx, es:[bx-1]
    xchg cx, ax
; ----- вариант 3
    mov di, ind
    mov es:[bx+di], ax
; ----- вариант 4
    mov bp, sp
    mov ax, matr[bp+bx]
    mov ax, matr[bp+di+si]
; Использование сегмента стека
    push mem1
    push mem2
    mov bp, sp
    mov dx, [bp]+2
    ret 2
Main      ENDP
CODE      ENDS
END Main

```

### **Ошибки, обнаруженные в коде.**

Lab2.asm(42): error A2052: Improper operand type (Неподходящий тип операнда)

```
mov mem3,[bx]
```

Объяснение: нельзя читать из памяти и писать в память одной командой (передавать данные из памяти в память)

Lab2.asm(49): warning A4031: Operand types must match

```
mov cx,vec2[di]
```

Объяснение: размер элементов массива vec2 1 байт (число), а cx - 2 байта (слово)

Lab2.asm(53): warning A4031: Operand types must match

```
mov cx,matr[bx][di]
```

Объяснение: размер элементов матрицы matr 1 байт (число), а cx - 2 байта (слово)

Lab2.asm(54): error A2055: Illegal register value

```
mov ax,matr[bx*4][di]
```

Объяснение: нельзя умножать 16-битные регистры

Lab2.asm(73): error A2046: Multiple base registers

```
mov ax,matr[bp+bx]
```

Объяснение: нельзя использовать более одного базового регистра

Lab2.asm(74): error A2047: Multiple index registers

```
mov ax,matr[bp+di+si]
```

Объяснение: нельзя использовать более одного индексного регистра

```
push mem1
```

```
push mem2
```

Объяснение: после этих двух команд в вершине стека уже не лежит смещение и программа не может завершиться

### Листинг успешной программы

```
Microsoft (R) Macro Assembler Version 5.10                                10/11/20
18:20:5                                                                    Page
1-1
```

```
1                                ; Программа изучения режимов адресации
                                ; процессора IntelX86
2 = 0024                        EOL EQU '$'
3 = 0002                        ind EQU 2
4 = 01F4                        n1 EQU 500
5 =-0032                        n2 EQU -50
6                                ; Стек программы
7 0000                          AStack SEGMENT STACK
8 0000 000C[                     DW 12 DUP(?)
9      ????
```

```

11
12 0018                      AStack ENDS
13                          ; Данные программы
14 0000                      DATA SEGMENT
15                          ; Директивы описания данных
16 0000    0000                      mem1 DW 0
17 0002    0000                      mem2 DW 0
18 0004    0000                      mem3 DW 0
19 0006    1C 1B 1A 19 15 16 vec1 DB 28,27,26,25,21,22,23,24
20                          17 18
21 000E    14 1E EC E2 28 32 vec2 DB 20,30,-20,-30,40,50,-40,-50
22                          D8 CE
23 0016    F8 F9 03 04 FA FB matr DB -8,-7,3,4,-6,-5,1,2,-4,-
3,7,8,-
                                2,-1,5,6
24          01 02 FC FD 07 08
25          FE FF 05 06
26 0026                      DATA ENDS
27                          ; Код программы
28 0000                      CODE SEGMENT
29                          ASSUME CS:CODE, DS:DATA, SS:AStack
30                          ; Головная процедура
31 0000                      Main PROC FAR
32 0000    1E                      push DS
33 0001    2B C0                      sub AX,AX
34 0003    50                      push AX
35 0004    B8 ---- R                      mov AX,DATA
36 0007    8E D8                      mov DS,AX
37                          ; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ
                                СМЕЩЕНИЙ
38                          ; Регистровая адресация
39 0009    B8 01F4                      mov ax,n1
40 000C    8B C8                      mov cx,ax
41 000E    B3 24                      mov bl,EOL
42 0010    B7 CE                      mov bh,n2
43                          ; Прямая адресация
44 0012    C7 06 0002 R FFCE mov mem2,n2
45 0018    BB 0006 R                      mov bx,OFFSET vec1
46 001B    A3 0000 R                      mov mem1,ax
47                          ; Косвенная адресация
48 001E    8A 07                      mov al,[bx]
49                          ; mov mem3,[bx]
50                          ; Базированная адресация
51 0020    8A 47 03                      mov al,[bx]+3

```

1-2

```
52 0023 8B 4F 03          mov cx,3[bx]
53                      ; Индексная адресация
54 0026 BF 0002          mov di,ind
55 0029 8A 85 000E R      mov al,vec2[di]
56                      ;mov cx,vec2[di]
57                      ; Адресация с базированием и индексиров
                        анием
58 002D BB 0003          mov bx,3
59 0030 8A 81 0016 R      mov al,matr[bx][di]
60                      ;mov cx,matr[bx][di]
61                      ;mov ax,matr[bx*4][di]
62                      ; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ С УЧЕТОМ С
                        ЕГМЕНТОВ
63                      ; Переопределение сегмента
64                      ; ----- вариант 1
65 0034 B8 ---- R        mov ax, SEG vec2
66 0037 8E C0            mov es, ax
67 0039 26: 8B 07        mov ax, es:[bx]
68 003C B8 0000          mov ax, 0
69                      ; ----- вариант 2
70 003F 8E C0            mov es, ax
71 0041 1E              push ds
72 0042 07              pop es
73 0043 26: 8B 4F FF      mov cx, es:[bx-1]
74 0047 91              xchg cx,ax
75                      ; ----- вариант 3
76 0048 BF 0002          mov di,ind
77 004B 26: 89 01        mov es:[bx+di],ax
78                      ; ----- вариант 4
79 004E 8B EC            mov bp,sp
80                      ;mov ax,matr[bp+bx]
81                      ;mov ax,matr[bp+di+si]
82                      ; Использование сегмента стека
83 0050 FF 36 0000 R      push mem1
84 0054 FF 36 0002 R      push mem2
85 0058 8B EC            mov bp,sp
86 005A 8B 56 02          mov dx,[bp]+2
87 005D CA 0002          ret 2
88 0060                  Main ENDP
89 0060                  CODE ENDS
90                      END Main
```

Symbols-1

Segments and Groups:

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

Symbols:

	N a m e	Type	Value	Attr
	EOL . . . . .	NUMBER		0024
	IND . . . . .	NUMBER		0002
0060	MAIN . . . . .	F PROC		0000 CODE Length =
	MATR . . . . .	L BYTE		0016 DATA
	MEM1 . . . . .	L WORD		0000 DATA
	MEM2 . . . . .	L WORD		0002 DATA
	MEM3 . . . . .	L WORD		0004 DATA
	N1 . . . . .	NUMBER		01F4
	N2 . . . . .	NUMBER		-0032
	VEC1 . . . . .	L BYTE		0006 DATA
	VEC2 . . . . .	L BYTE		000E DATA
	@CPU . . . . .	TEXT	0101h	
	@FILENAME . . . . .	TEXT	lab2	
	@VERSION . . . . .	TEXT	510	

83 Source Lines  
83 Total Lines  
19 Symbols

47308 + 461999 Bytes symbol space free

0 Warning Errors  
0 Severe Errors



## Протокол работы на компьютере.

AX 0000	SI 0000	CS 1A0A	IP 0000	Stack +0 0000	Flags 7202
BX 0000	DI 0000	DS 19F5		+2 0000	
CX 00B0	BP 0000	ES 19F5	HS 19F5	+4 0000	OF DF IF SF ZF AF PF CF
DX 0000	SP 0018	SS 1A05	FS 19F5	+6 0000	0 0 1 0 0 0 0 0

CMD >

1

0 1 2 3 4 5 6 7

DS:0000 CD 20 FF 9F 00 EA F0 FE

DS:0008 AD DE 1B 05 C5 06 00 00

DS:0010 18 01 10 01 18 01 92 01

DS:0018 01 01 01 00 02 FF FF FF

DS:0020 FF FF FF FF FF FF FF FF

DS:0028 FF FF FF FF EB 19 C0 11

DS:0030 A2 01 14 00 18 00 F5 19

DS:0038 FF FF FF FF 00 00 00 00

DS:0040 05 00 00 00 00 00 00 00

DS:0048 00 00 00 00 00 00 00 00

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00
DS:0010	18	01	10	01	18	01	92	01	01	01	01	00	02	FF	FF	FF
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	C0	11	
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	00	00	00	00
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

= f.⊕■

⋮.+. . . . .

.....f. . . . .

δ. L. . . . .

ó. . . . .J. . . . .

..... . . . . .

1 Step

2ProcStep

3Retrieve

4Help ON

5BRK Menu

6

7 ↑

8 ↓

9 ←

10 →

Рисунок 1 – Начальные значения регистров и ячеек памяти.

Таблица 1 – Результаты пошагового выполнения *Lab2.asm*.

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	PUSH DS	1E	(SP) = 0018 (DS) = 19F5 Stack: +0 0000	(SP) = 0016 (DS) = 19F5 Stack: +0 19F5
0001	SUB AX,AX	2BC0	(AX) = 0000	(AX) = 0000
0003	PUSH AX	50	(AX) = 0000 (SP) = 0016 Stack: +0 19F5 Stack: +2 0000	(AX) = 0000 (SP) = 0014 Stack: +0 0000 Stack: +2 19F5
0004	MOV AX, 1A07	BB071A	(AX) = 0000	(AX) = 1A07
0007	MOV DS,AX	8ED8	(DS) = 19F5 (AX) = 1A07	(DS) = 1A07 (AX) = 1A07
0009	MOV AX, 01F4	B8F401	(AX) = 1A07	(AX) = 01F4

000C	MOV CX, AX	8BC8	(CX) = 00B0 (AX) = 01F4	(CX) = 01F4 (AX) = 01F4
000E	MOV BL, 24	B324	(BX) = 0000	(BX) = 0024
0010	MOV BH, CE	B7CE	(BX) = 0024	(BX) = CE24
0012	MOV [0002], FFCE	C7060200CEFF	DS: 0002 = 00 DS: 0003 = 00	DS: 0002 = CE DS: 0003 = FF
0018	MOV BX, 0006	BB0600	(BX) = CE24	(BX) = 0006
001B	MOV [0000], AX	A30000	(AX) = 01F4 DS: 0000 = 00 DS: 0001 = 00	(AX) = 01F4 DS: 0000 = F4 DS: 0001 = 01
001E	MOV AL, [BX]	8A07	(AX) = 01F4 (BX) = 0006 DS: 0006 = 1C	(AX) = 011C (BX) = 0006 DS: 0006 = 1C
0020	MOV AL, [BX+03]	8A4703	(AX) = 0108 (BX) = 0006 DS: 0009 = 19	(AX) = 0119 (BX) = 0006 DS: 0009 = 19
0023	MOV CX, [BX+03]	8B4F03	(CX) = 01F4 (BX) = 0006 DS: 0009 = 19 DS: 000A = 15	(CX) = 1519 (BX) = 0006 DS: 0009 = 19 DS: 000A = 15
0026	MOV DI, 0002	BF0200	(DI) = 0000	(DI) = 0002
0029	MOV AL, [000E+DI]	8A850E00	(AX) = 0119 (DI) = 0002 DS: 0010 = EC	(AX) = 01EC (DI) = 0002 DS: 0010 = EC
002D	MOV BX, 0003	BB0300	(BX) = 0006	(BX) = 0003
0030	MOV AL, [0016+BX+DI]	8A811600	(AX) = 01EC (BX) = 0003 (DI) = 0002 DS: 001B = FB	(AX) = 01FB (BX) = 0003 (DI) = 0002 DS: 001B = FB
0034	MOV AX, 1A07	B8071A	(AX) = 01FB	(AX) = 1A07

0037	MOV ES, AX	8EC0	(ES) = 19F5 (AX) = 1A07	(ES) = 1A07 (AX) = 1A07
0039	MOV AX, ES:[BX]	268B07	(AX) = 1A07 (ES) = 1A07 (BX) = 0003 DS: 0003 = FF DS: 0004 = 00	(AX) = 00FF (ES) = 1A07 (BX) = 0003 DS: 0003 = FF DS: 0004 = 00
003C	MOV AX, 0000	B80000	(AX) = 00FF	(AX) = 0000
003F	MOV ES, AX	8EC0	(AX) = 0000 (ES) = 1A07	(AX) = 0000 (ES) = 0000
0041	PUSH DS	1E	(DS) = 1A07 (SP) = 0014 Stack: +0 0000 Stack: +2 19F5 Stack: +4 0000	(DS) = 1A07 (SP) = 0012 Stack: +0 1A07 Stack: +2 0000 Stack: +4 19F5
0042	POP ES	07	(ES) = 0000 (SP) = 0012 Stack: +0 1A07 Stack: +2 0000 Stack: +4 19F5	(ES) = 1A07 (SP) = 0014 Stack: +0 0000 Stack: +2 19F5 Stack: +4 0000
0043	MOV CX, ES:[BX-01]	268B4FFF	(CX) = 0105 (ES) = 1A07 (BX) = 0003 DS: 0002 = CE DS: 0003 = FF	(CX) = FFCE (ES) = 1A07 (BX) = 0003 DS: 0002 = CE DS: 0003 = FF
0047	XCHG AX, CX	91	(AX) = 0000 (CX) = FFCE	(AX) = FFCE (CX) = 0000
0048	MOV DI, 0002	BF0200	(DI) = 0002	(DI) = 0002
004B	MOV ES:[BX+DI], AX	268901	(ES) = 1A07 (BX) = 0003 (DI) = 0002	(ES) = 1A07 (BX) = 0003 (DI) = 0002

			(AX) = FFCE DS: 0005 = 00 DS: 0006 = 1C	(AX) = FFCE DS: 0005 = CE DS: 0006 = FF
004E	MOV BP, SP	8BEC	(BP) = 0000 (SP) = 0014	(BP) = 0014 (SP) = 0014
0050	PUSH [0000]	FF360000	DS:0000 = F4 DS:0001 = 01 (SP) = 0014 Stack: +0 0000 Stack: +2 19F5 Stack: +4 0000	DS:0000 = F4 DS:0001 = 01 (SP) = 0012 Stack: +0 01F4 Stack: +2 0000 Stack: +4 19F5
0054	PUSH [0002]	FF360200	DS:0002 = CE DS:0003 = FF (SP) = 0012 Stack: +0 01F4 Stack: +2 0000 Stack: +4 19F5 Stack: +6 0000	DS:0002 = CE DS:0003 = FF (SP) = 0010 Stack: +0 FFCE Stack: +2 01F4 Stack: +4 0000 Stack: +6 19F5
0058	MOV BP, SP	8BEC	(BP) = 0014 (SP) = 0010	(BP) = 0010 (SP) = 0010
005A	MOV DX, [BP+02]	8B5602	(DX) = 0000 (BP) = 0010 Stack: +2 01F4	(DX) = 01F4 (BP) = 0010 Stack: +2 01F4
005D	RET FAR 0002	CA0200	(SP) = 0010 (CS) = 1A0A Stack: +0 FFCE Stack: +2 01F4 Stack: +4 0000 Stack: +6 19F5	(SP) = 0016 (CS) = 01F4 Stack: +0 19F5 Stack: +2 0000 Stack: +4 0000 Stack: +6 0000

### **Выводы.**

Были найдены и закомментированы ошибки в предоставленной программе на языке программирования Ассемблер. Были изучены различные режимы адресации и формирования исполнительного адреса в языке программирования Ассемблер.

# ПРИЛОЖЕНИЕ А

## ИСПРАВЛЕННЫЙ КОД ПРОГРАММЫ

### Название файла: lab2.asm

```
; ПРОГРАММА ИЗУЧЕНИЯ РЕЖИМОВ АДРЕСАЦИИ ПРОЦЕССОРА INTELX86
EOL EQU '$'
IND EQU 2
N1 EQU 500
N2 EQU -50
; СТЕК ПРОГРАММЫ
ASTACK SEGMENT STACK
DW 12 DUP(?)
ASTACK ENDS
; ДАННЫЕ ПРОГРАММЫ
DATA SEGMENT
; ДИРЕКТИВЫ ОПИСАНИЯ ДАННЫХ
MEM1 DW 0
MEM2 DW 0
MEM3 DW 0
VEC1 DB 28,27,26,25,21,22,23,24
VEC2 DB 20,30,-20,-30,40,50,-40,-50
MATR DB -8,-7,3,4,-6,-5,1,2,-4,-3,7,8,-2,-1,5,6
DATA ENDS
; КОД ПРОГРАММЫ
CODE SEGMENT
ASSUME CS:CODE, DS:DATA, SS:ASTACK
; ГОЛОВНАЯ ПРОЦЕДУРА
MAIN PROC FAR
PUSH DS
SUB AX,AX
PUSH AX
MOV AX,DATA
MOV DS,AX
; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ СМЕЩЕНИЙ
; РЕГИСТРОВАЯ АДРЕСАЦИЯ
```

```

MOV AX,N1
MOV CX,AX
MOV BL,EOL
MOV BH,N2
; ПРЯМАЯ АДРЕСАЦИЯ
MOV MEM2,N2
MOV BX,OFFSET VEC1
MOV MEM1,AX
; КОСВЕННАЯ АДРЕСАЦИЯ
MOV AL,[BX]
    ;MOV MEM3,[BX]
; БАЗИРОВАННАЯ АДРЕСАЦИЯ
MOV AL,[BX]+3
MOV CX,3[BX]
; ИНДЕКСНАЯ АДРЕСАЦИЯ
MOV DI,IND
MOV AL,VEC2[DI]
    ;MOV CX,VEC2[DI]
; АДРЕСАЦИЯ С БАЗИРОВАНИЕМ И ИНДЕКСИРОВАНИЕМ
MOV BX,3
MOV AL,MATR[BX][DI]
    ;MOV CX,MATR[BX][DI]
    ;MOV AX,MATR[BX*4][DI]
; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ
; ПЕРЕОПРЕДЕЛЕНИЕ СЕГМЕНТА
; ----- ВАРИАНТ 1
MOV AX, SEG VEC2
MOV ES, AX
MOV AX, ES:[BX]
MOV AX, 0
; ----- ВАРИАНТ 2
MOV ES, AX
PUSH DS
POP ES
MOV CX, ES:[BX-1]

```

```
XCHG CX,AX

; ----- ВАРИАНТ 3

MOV DI,IND

MOV ES:[BX+DI],AX

; ----- ВАРИАНТ 4

MOV BP,SP

    ;MOV AX,MATR[BP+BX]

    ;MOV AX,MATR[BP+DI+SI]

; ИСПОЛЬЗОВАНИЕ СЕГМЕНТА СТЕКА

PUSH MEM1

PUSH MEM2

MOV BP,SP

MOV DX,[BP]+2

RET 2

MAIN ENDP

CODE ENDS

END MAIN
```