

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

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

Студент гр1383

Федорова О.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Изучение режимов адресации на языке Ассемблера.

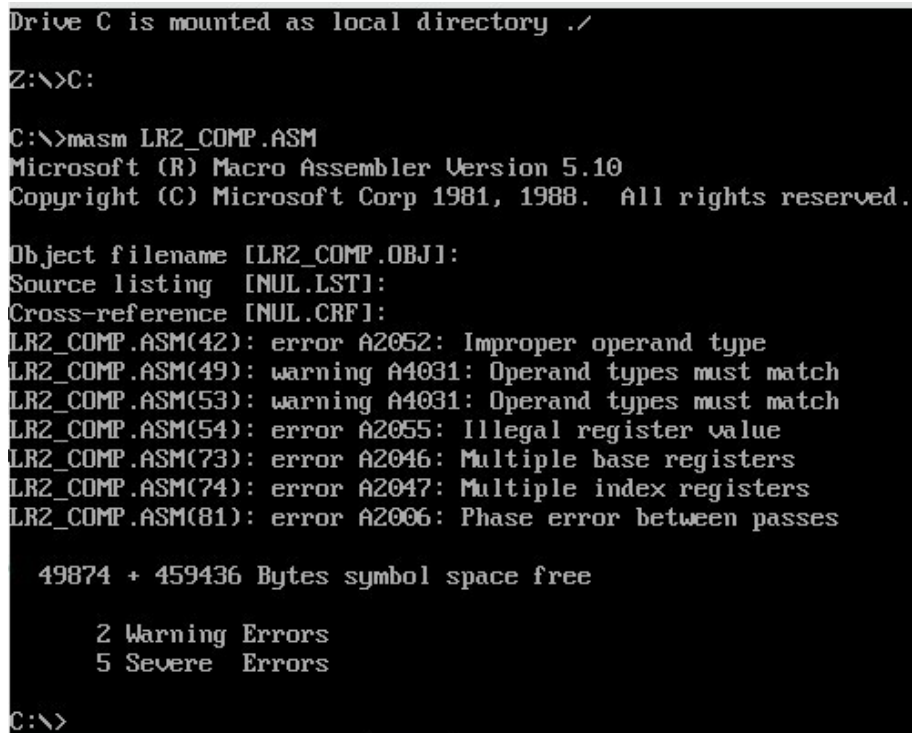
Задание.

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 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.

Выполнение работы.

1. Обнаруженные ошибки.

Выведенные ошибки при компиляции выведены на Рис. 1



```
Drive C is mounted as local directory ./
Z:\>C:

C:\>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]:
Cross-reference [NUL.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(81): error A2006: Phase error between passes

49874 + 459436 Bytes symbol space free

2 Warning Errors
5 Severe Errors

C:\>
```

Комментарии к ошибкам:

- строка 42 `mov mem3,[bx]` — нельзя читать из памяти и записывать в память одновременно.

Warning 49 — Не совпадение типов, в старший байт регистра CX запишется следующий байт — `vec2[di+1] = 14` (14 в 16сс = 20 в 10сс), а в младший — значение, лежащее в `vec2[di] = 0A`

Warning 53 — Не совпадение типов, из-за которого в регистр CX в старший байт дополнительно запишется следующий байт — `matr[bx][di + 1] = FE` (FE в машинном коде, -2 в привычном значении), а в младший — значение `matr[bx][di] = FD`

Error 54 — нельзя масштабировать базовые регистры

Error 73 - нельзя использовать более одного базового регистра

Error 74 — Нельзя использовать больше одного индексного регистра

2. Протокол выполнение программы.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO

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 >
0000 1E          PUSH    DS
0001 2BC0          SUB     AX,AX
0003 50          PUSH    AX
0004 B8071A       MOV     AX,1A07
0007 B8D8        MOV     DS,AX
0009 B8F401       MOV     AX,01F4
000C B8C8        MOV     CX,AX
000E B324        MOV     BL,24
  
```

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FF	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
---------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------

Таблица 2. Протокол main.asm

Адрес Команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти
------------------	------------------------------	--------------------------	--

			До выполнения	После выполнения
0000	push ds	1E	ip = 0000 ds = 19F5 sp = 0018 stack: +0 0000 +2 0000 +4 0000 +6 0000	ip = 0001 ds = 19F5 sp = 0016 stack: +0 19F5 +2 0000 +4 0000 +6 0000
0001	sub ax, ax	2B C0	ip = 0001 ax = 0000	ip = 0003 ax = 0000
0003	push ax	50	ip = 0003	ip = 0004

			ax = 0000 sp = 0016 stack: +0 19F5 +2 0000 +4 0000 +6 0000	ax = 0000 sp = 0014 stack: +0 0000 +2 19F5 +4 0000 +6 0000
0004	mov ax, data	B8 07 1A	ip = 0004 ax = 0000	ip = 0007 ax = 1A07
0007	mov ds, ax	8E D8	ip = 0007 ds = 19F5 ax = 1A07	ip = 0009 ds = 1A07 ax = 1A07
0009	mov ax, n1	B8 F4 01	ip = 0009 ax = 1A07	ip = 000C ax = 01F4

000C	mov cx, ax	8B C8	ip = 000C ax = 01F4 cx = 00B0	ip = 000E ax = 01F4 cx = 01F4
000E	mov bl, EOL	B3 24	ip = 000E bx = 0000	ip = 0010 bx = 0024
0010	mov bh, n2	B7 CE	ip = 0010 bx = 0024	ip = 0012 bx = CE24
0012	mov mem2, n2	C7 06 02 00 CE FF	ip = 0012 mem2 = 0000	ip = 0018 mem2 = FFCE
0018	mov bx, offset vec1	BB 06 00	ip = 0018 bx = CE24	ip = 001B bx = 0006
001B	mov mem1, ax	A3 00 00	ip = 001B mem1 = 0000	ip = 001E mem1 = 01F4
001E	mov al, [bx]	8A 07	ip = 001E bx = 0006 ax = 01F4	ip = 0020 bx = 0006 ax = 0101

0020	mov al, [bx]+3	8A 47 03	ip = 0020 bx = 0006 ax = 0101	ip = 0023 bx = 0006 ax = 0104
0023	mov cx, 3[bx]	8B 4F 03	ip = 0023 bx = 0006 cx = 01F4	ip = 0026 bx = 0006 cx = 0804
0026	mov di, ind	BF 02 00	ip = 0026 di = 0000	ip = 0029 di = 0002
0029	mov al, vec2[di]	8A 85 0E 00	ip = 0029 ax = 0104	ip = 002D ax = 010A
002D	Mov cx, vec2[di]	8B 8D 0E 00	ip = 002D cx = 0804	ip = 0031 ax = 140A
0031	mov bx, 3	BB 03 00	ip = 002D bx = 0006	ip = 0030 bx = 0003

0034	mov al, matr[bx] [di]	8A 81 16 00	ip = 0034 ax = 010A	ip = 0038 ax = 01FD
0038	mov cx, matr[bx] [di]	8B 89 16 00	ip = 0038 cx = 140A	p = 003C cx = FFED
003C	mov ax, seg vec2	B8 07 1A	ip = 0034 ax = 01FD	ip = 003F ax = 1A07
003F	mov es, ax	8E C0	ip = 0037 es = 19F5 ax = 1A07	ip = 0041 es = 1A07 ax = 1A07
0041	mov ax, es:[bx]	26 8B 07	ip = 0039 ax = 1A07	ip = 0044 ax = 00FF
0044	mov ax, 0	B8 00 00	ip = 003C ax = 00FF	ip = 0047 ax = 0000
0047	mov es, ax	8E C0	ip = 003F es = 1A07 ax = 0000	ip = 0049 es = 0000 ax = 0000

0049	push ds	1E	ip = 0041 sp = 0014 ds = 1A07 stack: +0 0000 +2 19F5 +4 0000 +6 0000	ip = 004A sp = 0012 ds = 1A07 stack: +0 1A07 +2 0000 +4 19F5 +6 0000
------	---------	----	---	---

004A	pop es	07	ip = 0042 sp = 0012 es = 0000 stack: +0 1A07 +2 0000 +4 19F5 +6 0000	ip = 004B sp = 0014 es = 1A07 stack: +0 0000 +2 19F5 +4 0000 +6 0000
004B	mov cx, es:[bx-1]	26 8B 4F FF	ip = 0043 cx = 0804	ip = 004F cx = FFCE
004F	xchg cx, ax	91	ip = 0047 ax = 0000 cx = FFCE	ip = 0050 ax = FFCE cx = 0000
0050	mov di, ind	BF 02 00	ip = 0048 di = 0002	ip = 0053 di = 0002
0053	mov es:[bx+di], ax	26 89 01	ip = 004B es:[bx+di] = 0100 ax = FFCE	ip = 0056 es:[bx+di] = FFCE ax = FFCE
0056	mov bp, sp	8B EC	ip = 004E bp = 0000	ip = 0058 bp = 0014

			sp = 0014	sp = 0014
0058	push mem1	FF 36 00 00	ip = 0050 sp = 0014 stack: +0 0000 +2 19F5 +4 0000 +6 0000	ip = 005C sp = 0012 stack: +0 01F4 +2 0000 +4 19F5 +6 0000

005C	push mem2	FF 36 02 00	ip = 0054 sp = 0012 stack: +0 01F4 +2 0000 +4 19F5 +6 0000	ip = 0060 sp = 0010 stack: +0 FFCE +2 01F4 +4 0000 +6 19F5
0060	mov bp, sp	8B EC	ip = 0058 bp = 0014 sp = 0010	ip = 0062 bp = 0010 sp = 0010
0062	mov dx, [bp]+2	8B 56 02	ip = 005A dx = 0000	ip = 0065 dx = 01F4
0065	ret 2	CA 02 00	ip = 005D cs = 1A0A sp = 0010 stack: +0 FFCE +2 01F4 +4 0000 +6 19F5	ip = FFCE cs = 01F4 sp = 0016 stack: +0 19F5 +2 0000 +4 0000 +6 0000

Выводы.

Изучены режимы адресации на языке Ассемблера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

lr2_comp.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 1,2,3,4,8,7,6,5 vec2 DB -10,-

20,10,20,-30,-40,30,40 matr DB 1,2,3,4,-4,-

3,-2,-1,5,6,7,8,-8,-7,-6,-5

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] ;error   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];запишется 140А (0А - по индексуБ 14 - 20в 16йсс -
следующий)

```

```

; Адресация с базированием и индексированием    mov
bx,3    mov al,matr[bx][di];al = FD = -3 = matr[5]    mov
cx,matr[bx][di];cx = FEFD FE = next = matr[6] = -2 ;error
mov ax,matr[bx*4][di] нельзя масштабировать bx ;

```

ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

```

; Переопределение сегмента
; ----- вариант 1    mov ax, SEG
vec2 ;натало DATA    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

```

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

ПРИЛОЖЕНИЕ В

ЛИСТИНГ ПРОГРАММЫ

Lr2.lst

```

; D[?] N[?] D[?] D[?] D[?] D[?]
D[?] D[?] N[?] N[?] D[?] D[?] D[?] N[?] N[?] D[?] D[?]
        C[?] D[?] D[?] D[?] D[?] D[?] D[?] D[?] D[?]
D[?] N[?] D[?] N[?] D[?] N[?] N[?] D[?] D[?] D[?] I
        ntelX86

= 0024                EOL EQU '$'
= 0002                ind EQU 2
= 01F4                n1 EQU 500
=-0032                n2 EQU -50

; D[?] N[?] D[?] D[?] D[?] D[?] D[?] D[?] D[?]
0000                AStack SEGMENT STACK
0000 000C[           DW 12 DUP(?)
        ???
    ]

0018                AStack ENDS

; D[?] D[?] D[?] D[?] D[?]
D[?] N[?] D[?] D[?] D[?] D[?] D[?]
0000                DATA SEGMENT

; D[?] D[?] D[?] D[?] D[?] D[?]
D[?] D[?] D[?] D[?] D[?] D[?] D[?]
        D[?]
0000 0000            mem1 DW 0
0002 0000            mem2 DW 0
0004 0000            mem3 DW 0
0006 01 02 03 04 08 07 vec1 DB 1,2,3,4,8,7,6,5
06 05

```

```
000E F6 EC 0A 14 E2 D8      vec2 DB -10,-20,10,20,-30,-40,30,40  
    1E 28  
  
0016 01 02 03 04 FC FD      matr DB 1,2,3,4,-4,-3,-2,-1,5,6,7,8,-8,-7,-6,-5  
    FE FF 05 06 07 08  
    F8 F9 FA FB  
  
0026                          DATA ENDS  
                                ; Ď#ĚĐŸĐŽ ĐŁÑ † ĐŸĐ³Ñ ‡ Đ°ĐœĐœÑ ±  
  
0000                          CODE SEGMENT  
                                ASSUME CS:CODE, DS:DATA, SS:AStack  
                                ; Đ≡ĐŸĐ»ĐŸĐ²ĐœĐ°Ñ ☒  
ĐŁÑ † ĐŸÑ Ų ĐμĐžÑ † Ñ † Đ°  
  
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  
                                ; Đ☐ĐĐ☐Đ◀Đŷ ĐĐ÷⁄Đ☼  
ĐĐŷ Đ⦿Đ★Đ☛Đ☐Đ☼Đ☐Đ☼Đ†‡ĐĐŷ ΔᵢΔ☼  
                                ŠĐ★Đ★ Đ☐Đ☼ Đ£ĐĐ☐Đ◀Đ☐Đŷ  
ΔᵢĐ☛Đŷ Δ©Đŷ Đ☐Đ★Đ☒  
                                ; ĐĐμĐ³ĐžÑ † Ñ † Ñ † ĐŸĐ²Đ°Ñ ☒  
Đ°ĐžÑ † ΔμÑ † Đ°Ñ Ų ĐžÑ ☒  
  
0009 B8 01F4                mov ax,n1  
  
000C 8B C8                  mov cx,ax  
  
000E B3 24                  mov bl,EOL  
  
0010 B7 CE                  mov bh,n2  
                                ; Đ☐Ñ † Ñ ☒ ĐœĐ°Ñ ☒  
Đ°ĐžÑ † ΔμÑ † Đ°Ñ Ų ĐžÑ ☒  
  
0012 C7 06 0002 R FFCE      mov mem2,n2  
  
0018 BB 0006 R              mov bx,OFFSET vec1
```

```

001B A3 0000 R          mov mem1,ax
                        ; D#DŸÑ ‡ D²DµDœDœD°Ñ ☒
D°DŽÑ ‡ DµÑ ‡ D°Ñ‘ŸDžÑ ☒
001E 8A 07              mov al,[bx]
                        ;error  mov mem3,[bx]
                        ; D$D°D·DžÑ ‡ DŸD²D°DœDœD°Ñ ☒
D°DŽÑ ‡ DµÑ ‡ D°Ñ‘ŸDžÑ ☒

```

0020 8A 47 03 mov al,[bx]+3

0023 8B 4F 03 mov cx,3[bx]

; D★DœDŽDμD°Ñ ħ DœD°Ñ ☒

D°DŽÑ ħ DμÑ ħ D°Ñ'ŸDžÑ ☒

0026 BF 0002 mov di,ind

0029 8A 85 000E R mov al,vec2[di]

002D 8B 8D 000E R mov

cx,vec2[di];D·D°DġDžÑ ħ DμÑ ħ Ñ ħ Ñ ☒ 140D ☒ (0

D ☒ - DġDŸ DžDœDŽDμD°Ñ ħ Ñ ħ D ☒ 14 - 20D²

16D¹Ñ ħ Ñ ħ -

Ñ ħ D»DμDŽÑ ħ Ñ ☒ Ñ ħ DžD¹)

LR2_COMP.ASM(49): warning A4031: Operand types must match

; D ☒ DŽÑ ħ DμÑ ħ D°Ñ'ŸDžÑ ☒ Ñ ħ

D±D°D·DžÑ ħ DŸD²D°DœDžDμD

Œ Dž

DžDœDŽDμD°Ñ ħ DžÑ ħ DŸD²D°DœDžDμDŒ

0031 BB 0003 mov bx,3

0034 8A 81 0016 R mov al,matr[bx][di];al = FD = -3 = matr[5]

0038 8B 89 0016 R mov cx,matr[bx][di];cx = FEFD FE = next =
m

atr[6] = -2

LR2_COMP.ASM(53): warning A4031: Operand types must match

;error mov ax,matr[bx*4][di] DœDμD»Ñ ☒ D·Ñ ☒

DŒ

D°Ñ ħ Ñ ħ D°D±DžÑ ħ DŸD²D°Ñ ħ Ñ ☒ D±Ñ ☒

; D ☒ D D ☒ D ☒ D ☒ D ☒ D ☒

D D \ D ☒ D★D ☒ D ☒ D ☒ D ☒ D ☒ D ☒ D ☒ D ☒ D ☒

[illegible]

0068	Main ENDP
0068	CODE ENDS
	END Main

Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA		STACK
CODE	0068	PARA		NONE
DATA	0026	PARA		NONE

Symbols:

N a m e	Type	Value	Attr
EOL	NUMBER	0024	
IND	NUMBER	0002	
MAIN	F PROC	0000	CODE Length = 0068
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 LR2_COMP

@VERSION TEXT 510

83 Source Lines

83 Total Lines

19 Symbols

47812 + 459448 Bytes symbol space free

2 Warning Errors

0 Severe Errors