

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: Трансляции, отладка и выполнение программ на языке
Ассемблера
Вариант №6

Студент гр. 1383

Богданов Е.М.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цели работы.

Изучить режимы адресации и формирование исполнительного адреса.

Задание.

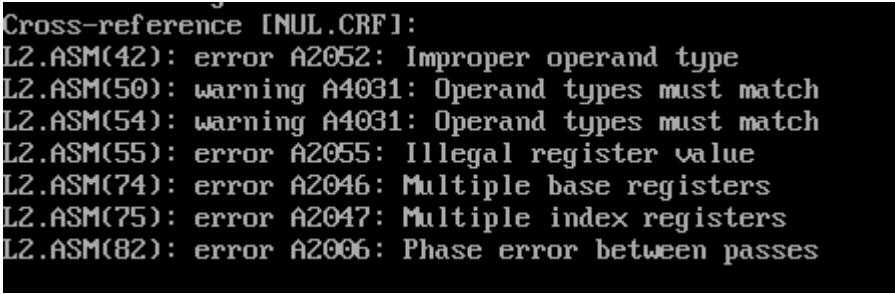
Лабораторная работа 2 предназначена для изучения режимов адресации, использует готовую программу `lr2_comp.asm` на Ассемблере, которая в автоматическом режиме выполняться не должна, так как не имеет самостоятельного функционального назначения, а только тестирует режимы адресации. Поэтому ее выполнение 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.

Выполнение работы.

1. Были изменены значения `vec1`, `vec2`, `matr` согласно варианту.
2. Была проведена попытка трансляции файла с получением ошибок (рис.1).

Рисунок 1 — Ошибки при первой попытке трансляции

A screenshot of a terminal window showing the output of an assembler. The text is as follows:

```
Cross-reference [NUL.CRF]:  
L2.ASM(42): error A2052: Improper operand type  
L2.ASM(50): warning A4031: Operand types must match  
L2.ASM(54): warning A4031: Operand types must match  
L2.ASM(55): error A2055: Illegal register value  
L2.ASM(74): error A2046: Multiple base registers  
L2.ASM(75): error A2047: Multiple index registers  
L2.ASM(82): error A2006: Phase error between passes
```

3. Были описаны предупреждения и ошибки(закомментированы):

`mov mem3, [bx];` — обратиться к области памяти можно только через регистр

`mov cx, vec2[di]` и `mov cx, matr[bx][di]` — регистры имеют размер 2 байта, тогда как `vec2[di]` и `matr[bx][di]`— 1 байт

mov ax,matr[bx*4][di] — регистр bx нельзя использовать для масштабирования

mov ax,matr[bp+bx]— базовый регистр должен быть один

mov ax,matr[bp+di+si] — индексный регистр должен быть один

4. Программа была протранслирована и был составлен протокол(табл.1 и рис. 2) :

Рисунок 2 — Начальное состояние регистров

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 SI
DX 0000	SP 0018	SS 1A05	FS 19F5	+6 0000	0 0 1 0

Таблица 1 — Протокол выполнение программы

Адрес команды	Символический код команды	16-ричный код команды	Регистры до выполнения	Регистры после выполнения
0000	PUSH DS	1E	SP=0018 STACK: +0 0000	SP=0016 STACK: +0 19F5
0001	SUB AX,AX	2B C0		
0003	PUSH AX	50	SP=0016 STACK: +0 19F5 +2 0000	SP=0014 STACK: +0 0000 +0 19F5
0004	MOV AX, 1A07	B8 07 1A	AX=0000	AX=1A07
0007	MOV DS, AX	8E D8	DS=19F5	DS=1A07
0009	MOV AX,01F4	B8 F4 01	AX= 1A07	AX=01F4
000C	MOV CX, AX	0B C8	CX=00B0	CX=01F4
000E	MOV BL,24	B3 24	BX=0000	BX=0024
0010	MOV BH, CE	B7 CE	BX=0024	BX=CE24
0012	MOV [0002],	C7 06 02 00		

	FFCE	CE FF		
0018	MOV BX, 0006	BB 06 00	BX=CE24	BX=0006
001B	MOV [0000], AX	A3 00 00		
001E	MOV AL, [BX]	8A 07	AX=01F4	AX=0112
0020	MOV AL, [BX+03]	8A 47 03	AX=0112	AX=010F
0023	MOV CX, [BX+03]	8B 4F 03	CX=01F4	CX=0B0F
0026	MOV DI, 0003	BF 02 00	DI=0000	DI=0002
0029	MOV AL, [000E+DI]	8A 85 0E 00	AX=010F	AX=01E2
002D	MOV BX, 0003	BB 03 00	BX=0006	BX=0003
0030	MOV AL, [0016+BX+DI]	8A 81 16 00	AX=01E2	AX=01FF
0034	MOV AX, 1A07	B8 07 1A	AX=01FF	AX=1A07
0037	MOV ES, AX	8E C0	ES=19F5	ES= 1A07
0039	MOV AX, ES: [BX]	26 8B 07	AX=1A07	AX=00FF
003C	MOV AX, 0000	B8 00 00	AX=00FF	AX=0000
003F	MOV ES, AX	8E C0	ES=1A07	ES=0000
0041	PUSH DS	1E	SP=0014 STACK: +0 0000 +2 19F5 +4 0000	SP=0012 STACK: +0 1A07 +2 0000 +4 19F5
0042	POP ES	07	SP=0012 ES=0000 STACK:	SP=0014 ES=1A07 STACK:

			+0 1A07 +2 0000 +4 19F5	+0 0000 +2 19F5 +4 0000
0043	MOV CX, ES:[BX-01]	26 8B 4F FF	CX=0B0F	CX=FFCE
0047	XCHG AX, CX	91	AX=0000 CX=FFCE	AX=FFCE CX=000
0048	MOV DI, 0002	BF 02 00		
004B	MOV ES:[BX+DI], AX	26 89 01		
004E	MOV BP,SP	8B EC	BP=0000	BP=0014
0050	PUSH [0000]	FF 36 00 00	SP=0014 STACK: +0 0000 +2 19F5 +4 0000	SP=0012 STACK: +0 01F4 +2 0000 +4 19F5
0054	PUSH [0002]	FF 36 02 00	SP=0012 STACK: +0 01F4 +2 0000 +4 19F5 +6 0000	SP=0010 STACK: +0 FFCE +2 01F4 +4 0000 +6 19F5
0058	MOV BP, SP	8B EC	BP=0014	BP=0010
005A	MOV DX, [BP+02]	8B 56 02	DX=0000	DX=01F4
005D	RET FAR 0002	CA 02 00	SP=0010 CS=1A0A +0 FFCE	SP=0016 CS=01F4 +0 19F5

			+2 01F4	+2 0000
			+4 0000	+4 0000
			+6 19F5	+6 0000

Выводы.

Изучены режимы адресации на языке Ассемблера.

Составлен протокол работы программы, найдены и объяснены ошибки в исходном коде

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
; 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 18,17,16,15,11,12,13,14
vec2 DB 30,40,-30,-40,10,20,-10,-20
matr DB -4,-3,1,2,-2,-1,3,4,5,6,7,8,-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]
```

```

    ;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

```


ПРИЛОЖЕНИЕ Б

ЛИСТИНГ

Microsoft (R) Macro Assembler Version 5.10

10/22/22 17:40:2

Page 1-1

```

; Программа изучения режи
; ов адресации процессора I
ntelX86
= 0024          EOL EQU '$'
= 0002          ind EQU 2
= 01F4          n1 EQU 500
=-0032         n2 EQU -50

; Стек программы
AStack SEGMENT STACK
0000          DW 12 DUP(?)
0000 000C[
        ????
]

0018          AStack ENDS
; Данные программы
0000          DATA SEGMENT
; Директивы описания данн
; x
0000 0000      mem1 DW 0
0002 0000      mem2 DW 0
0004 0000      mem3 DW 0
0006 12 11 10 0F 0B 0C      vec1 DB 18,17,16,15,11,12,13,14
        0D 0E
000E 1E 28 E2 D8 0A 14      vec2 DB 30,40,-30,-40,10,20,-10,-20
        F6 EC
0016 FC FD 01 02 FE FF      matr DB -4,-3,1,2,-2,-1,3,4,5,6,7,8,-8,-7,-6,-
        5
        03 04 05 06 07 08
        F8 F9 FA FB
0026          DATA ENDS
; Код программы
0000          CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:AStack
; Головная процедура
Main PROC FAR
0000          push DS
0001 1E         sub AX,AX
0003 2B C0      push AX
0004 50         mov AX,DATA
0004 B8 ---- R  mov DS,AX
0007 8E D8      mov DS,AX
; ПРОВЕРКА РЕЖИМОВ АДРЕСА
; ИИ НА УРОВНЕ СМЕЩЕНИЙ
; Регистровая адресация
0009 B8 01F4    mov ax,n1
000C 8B C8      mov cx,ax
000E B3 24      mov bl,EOL
0010 B7 CE      mov bh,n2
; Прямая адресация
0012 C7 06 0002 R FFCE    mov mem2,n2
0018 BB 0006 R  mov bx,OFFSET vec1
001B A3 0000 R  mov mem1,ax
; Косвенная адресация
001E 8A 07      mov al,[bx]
;mov mem3,[bx];
```

```

; Базированная адресация

0020 8A 47 03          mov al,[bx]+3
0023 8B 4F 03          mov cx,3[bx]
; Индексная адресация
0026 BF 0002          mov di,ind
0029 8A 85 000E R      mov al,vec2[di]
;mov cx,vec2[di]
; Адресация с базирование?
? и индексированием
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]
; ПРОВЕРКА РЕЖИМОВ АДРЕСА?
?ИИ С УЧЕТОМ СЕГМЕНТОВ
; Переопределение сегмент
a
; ----- вариант 1
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
; ----- вариант 2
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
; ----- вариант 3
0048 BF 0002          mov di,ind
004B 26: 89 01      mov es:[bx+di],ax
; ----- вариант 4
004E 8B EC          mov bp,sp
;mov ax,matr[bp+bx]
;mov ax,matr[bp+di+si]
; Использование сегмента ?
?тека
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 CA 0002          ret 2
0060                Main ENDP
0060                CODE ENDS
                END Main

```

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	
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
N1	NUMBER	01F4	
N2	NUMBER	-0032	
VEC1	L BYTE	0006	DATA
VEC2	L BYTE	000E	DATA
@CPU	TEXT	0101h	
@FILENAME	TEXT	L2	
@VERSION	TEXT	510	

84 Source Lines
84 Total Lines
19 Symbols

47842 + 459418 Bytes symbol space free

0 Warning Errors
0 Severe Errors