

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 гр. 1383

Кошкин Е.А.

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.

(41) `mov mem3, [bx]` — error: improper operand type.

Ошибка заключается в особенности команды `mov`. Нельзя осуществлять пересылку из одной области памяти в другую или из одной переменной в другую. Нужно использовать промежуточно регистр общего назначения.

(48) `mov cx, vec2[di]` — warning: operand types must match.

Операнды должны иметь одинаковую размерность.

(52) `mov cx, matr[bx][di]` — warning: operand types must match.

Аналогично.

(53) `mov ax, matr[bx*4][di]` — error: illegal register value.

В данной архитектуре нельзя масштабировать регистры.

(72) `mov ax, matr[bp + bx]` — error: multiple base registers.

Нельзя использовать больше одного базового регистра.

(73) `mov ax, matr[bp + di + si]` — error: multiple index registers.

Нельзя использовать больше одного индексного регистра.

2. Протокол выполнение программы.

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

Register	Value	Register	Value	Stack	Flags
AX	0000	SI	0000	CS	1A0A
BX	0000	DI	0000	IP	0000
CX	00B0	BP	0000	Stack	+0 0000
DX	0000	SP	001B	Flags	7202
		SS	1A05		+2 0000
		ES	19F5		+4 0000
		FS	19F5		OF DF IF SF ZF AF PF CF
					+6 0000

CMD >

Address	Disassembly	Comment
0000 1E	PUSH DS	
0001 2BC0	SUB AX, AX	
0003 50	PUSH AX	
0004 B8071A	MOV AX, 1A07	
0007 8ED8	MOV DS, AX	
0009 B8F401	MOV AX, 01F4	
000C 8BC8	MOV CX, AX	
000E B324	MOV BL, 24	

Address	Hex	ASCII
DS:0000	CD 20 FF 9F 00 EA F0 FE	= f.ñ = i.+....
DS:0010	18 01 10 01 18 01 92 01f.
DS:0020	FF FF FF FF FF FF FF FFδ. L.
DS:0030	A2 01 14 00 18 00 F5 19	ó.....J.
DS:0040	05 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ← 10 →

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 ax = 0000 sp = 0016 stack: +0 19F5 +2 0000 +4 0000 +6 0000	ip = 0004 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	ip = 000E

			ax = 01F4 cx = 00B0	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 = 0108
0020	mov al, [bx]+3	8A 47 03	ip = 0020 bx = 0006 ax = 0108	ip = 0023 bx = 0006 ax = 0105
0023	mov cx, 3[bx]	8B 4F 03	ip = 0023 bx = 0006 cx = 01F4	ip = 0026 bx = 0006 cx = 0105
0026	mov di, ind	BF 02 00	ip = 0026 di = 0000	ip = 0029 di = 0002
0029	mov al, ver2[di]	8A 85 0E 00	ip = 0029 ax = 0105	ip = 002D ax = 011E
002D	mov bx, 3	BB 03 00	ip = 002D bx = 0006	ip = 0030 bx = 0003
0030	mov al, matr[bx] [di]	8A 81 16 00	ip = 0030 ax = 011E	ip = 0034 ax = 0107
0034	mov ax, seg vec2	B8 07 1A	ip = 0034	ip = 0037

			ax = 0107	ax = 1A07
0037	mov es, ax	8E C0	ip = 0037 es = 19F5 ax = 1A07	ip = 0039 es = 1A07 ax = 1A07
0039	mov ax, es:[bx]	26 8B 07	ip = 0039 ax = 1A07	ip = 003C ax = 00FF
003C	mov ax, 0	B8 00 00	ip = 003C ax = 00FF	ip = 003F ax = 0000
003F	mov es, ax	8E C0	ip = 003F es = 1A07 ax = 0000	ip = 0041 es = 0000 ax = 0000
0041	push ds	1E	ip = 0041 sp = 0014 ds = 1A07 stack: +0 0000 +2 19F5 +4 0000 +6 0000	ip = 0042 sp = 0012 ds = 1A07 stack: +0 1A07 +2 0000 +4 19F5 +6 0000
0042	pop es	07	ip = 0042 sp = 0012 es = 0000 stack: +0 1A07 +2 0000 +4 19F5 +6 0000	ip = 0043 sp = 0014 es = 1A07 stack: +0 0000 +2 19F5 +4 0000 +6 0000
0043	mov cx, es:[bx-1]	26 8B 4F FF	ip = 0043 cx = 0105	ip = 0047 cx = FFCE

0047	xchg cx, ax	91	ip = 0047 ax = 0000 cx = FFCE	ip = 0048 ax = FFCE cx = 0000
0048	mov di, ind	BF 02 00	ip = 0048 di = 0002	ip = 004B di = 0002
004B	mov es:[bx+di], ax	26 89 01	ip = 004B es:[bx+di] = 0800 ax = FFCE	ip = 004E es:[bx+di] = FFCE ax = FFCE
004E	mov bp, sp	8B EC	ip = 004E bp = 0010 sp = 0014	ip = 0050 bp = 0014 sp = 0014
0050	push mem1	FF 36 00 00	ip = 0050 sp = 0014 stack: +0 0000 +2 19F5 +4 0000 +6 0000	ip = 0054 sp = 0012 stack: +0 01F4 +2 0000 +4 19F5 +6 0000
0054	push mem2	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 sp = 0010	ip = 005A bp = 0010 sp = 0010

005A	mov dx, [bp]+2	8B 56 02	ip = 005A dx = 01F4	ip = 005D dx = 01F4
005D	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.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.asm:

```
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 8,7,6,5,1,2,3,4
    vec2 DB -30,-40,30,40,-10,-20,10,20
    matr DB -1,-2,-3,-4,8,7,6,5,-5,-6,-7,-8,4,3,2,1
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

```

main.lst:

#Microsoft (R) Macro Assembler Version 5.10

10/9/22 14:26:25

Page 1-1

```

= 0024                                EOL EQU '$'
= 0002                                ind EQU 2
= 01F4                                n1 EQU 500
=-0032                                n2 EQU -50
                                     ; Стек программы
0000                                AStack SEGMENT STACK
0000 000C[                            DW 12 DUP(?)
    ????
                                     ]

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 08 07 06 05 01 02  vec1 DB 8,7,6,5,1,2,3,4
    03 04

```

000E	E2 D8 1E 28 F6 EC	vec2 DB -30,-40,30,40,-10,-20,10,20
	0A 14	
0016	FF FE FD FC 08 07	matr DB -1,-2,-3,-4,8,7,6,5,-5,-6,-7,-8,4,3
		,2,1
	06 05 FB FA F9 F8	
	04 03 02 01	
0026		DATA ENDS
		; Код программы
0000		CODE SEGMENT
		ASSUME CS:CODE, DS:DATA, SS:AStack
		; Головная процедура
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
		; ПРОВЕРКА РЕЖИМОВ АДРЕС
		АЦИИ 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 адресаци

я

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 базирован❖
❖ем 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 АДРЕ❖
❖АЦИИ С УЧЕТОМ СЕГМЕНТОВ
                        ; Переопределение сегме
нта
                        ; ----- 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 сегмент
	а стека
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

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		
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 main
@VERSION TEXT 510

82 Source Lines

82 Total Lines

19 Symbols

47810 + 459450 Bytes symbol space free

0 Warning Errors

0 Severe Errors