

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. 0383

Козлов Т.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

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.

Порядок выполнения работы.

1. Получить у преподавателя вариант набора значений исходных данных (массивов) `vec1`, `vec2` и `matr` из файла `lr2.dat`, приведенного в каталоге Задания и занести свои данные вместо значений, указанных в приведенной ниже программе.

2. Протранслировать программу с созданием файла диагностических сообщений; объяснить обнаруженные ошибки и закомментировать соответствующие операторы в тексте программы.

3. Снова протранслировать программу и скомпоновать загрузочный модуль.

4. Выполнить программу в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды. 6

5. Результаты прогона программы под управлением отладчика должны быть подписаны преподавателем и представлены в отчете. Пример используемой программы приведен ниже.

Вариант 4:

vec1 DB 12,11,10,9,5,6,7,8

vec2 DB -40,-50,40,50,-20,-30,20,30

matr DB 5,6,7,8,-8,-7,-6,-5,1,2,3,4,-4,-3,-2,-1

Ход работы.

1) При трансляции программы обнаружены следующие ошибки:

1. `mov mem3,[bx]` – error A2052: Improper operand type – инструкция `mov` неспособна перенести значение из одной ячейки памяти в другую. (для этого есть инструкция `movs` или можно использовать промежуточное значение)
2. `mov cx,vec2[di]` – warning A4031: Operand types must match – попытка положить данные из ячейки памяти с размером 1 байт в регистр с размером 2 байта.
3. `mov cx,matr[bx][di]` - warning A4031: Operand types must match – попытка положить данные из ячейки памяти с размером 1 байт в регистр с размером 2 байта.
4. `mov ax,matr[bx*4][di]` – error A2055: Illegal register value – недопустимое значение регистра.
5. `mov ax,matr[bp+bx]` – error A2046: Multiple base registers – недопустимо использовать несколько базовых регистров для адресации.
6. `mov ax,matr[bp+di+si]` – error 2047: Multiple index registers – недопустимо использовать несколько индексных регистров для адресации.

Строки, содержащие ошибки были закомментированы в файле LR2_FIX.asm

2) Запуск LR2_FIX.asm под отладчиком:

Начальное содержание сегментных регистров:

(CS) = 1A0A

(DS) = 19F5

(ES) = 19F5

(SS) = 1A05

Табл.1: Протокол выполнения IR2_FIX.asm

Адрес команды	Символический код команды	16- ричны й код команд ы	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	push DS	1E	(AX) = 0000 (BX) = 0000 (CX) = 0000 (DX) = 0000 (DI) = 0000 (SP) = 0018 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0000 Stack +0 0000	(AX) = 0000 (BX) = 0000 (CX) = 0000 (DX) = 0000 (DI) = 0000 (SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0001 Stack +0 19F5
0001	sub AX, AX	2BC0	(AX) = 0000 (BX) = 0000 (CX) = 0000 (DX) = 0000 (DI) = 0000 (SP) = 0016 (CS) = 1A0A	(AX) = 0000 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0016 (CS) = 1A0A

			(DS) = 19F5 (ES) = 19F5 (IP) = 0001 Stack +0 19F5	(DS) = 19F5 (ES) = 19F5 (IP) = 0003 Stack +0 19F5
0003	push AX	50	(AX) = 0000 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0003 Stack +0 19F5 Stack +2 19F5	(AX) = 0000 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0004 Stack +0 0000 Stack +2 19F5
0004	mov AX, 1A07	B8071 A	(AX) = 0000 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0004 Stack +0 0000 Stack +2 19F5	(AX) = 1A07 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0007 Stack +0 0000 Stack +2 19F5
0007	mov DS, AX	8ED8	(AX) = 1A07 (BX) = 0000 (CX) = 00B0	(AX) = 1A07 (BX) = 0000 (CX) = 00B0

			(DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (IP) = 0007 Stack +0 0000 Stack +2 19F5	(DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0009 Stack +0 0000 Stack +2 19F5
0009	mov AX, 01F4	B8F401	(AX) = 1A07 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0009 Stack +0 0000 Stack +2 19F5	(AX) = 01F4 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 000C Stack +0 0000 Stack +2 19F5
000C	mov CX AX	8BCB	(AX) = 01F4 (BX) = 0000 (CX) = 00B0 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5	(AX) = 01F4 (BX) = 0000 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5

			(IP) = 000C Stack +0 0000 Stack +2 19F5	(IP) = 000E Stack +0 0000 Stack +2 19F5
000E	mov BL, 24	B324	(AX) = 01F4 (BX) = 0000 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 000E Stack +0 0000 Stack +2 19F5	(AX) = 01F4 (BX) = 0024 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0010 Stack +0 0000 Stack +2 19F5
0010	Mov BH, CE	B7CE	(AX) = 01F4 (BX) = 0024 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0010 Stack +0 0000 Stack +2 19F5	(AX) = 01F4 (BX) = CE24 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0012 Stack +0 0000 Stack +2 19F5
0012	mov [0002], FFCE	C70602 00CEF F	(AX) = 01F4 (BX) = CE24 (CX) = 01F4 (DX) = 0000	(AX) = 01F4 (BX) = CE24 (CX) = 01F4 (DX) = 0000

			(DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0012 Stack +0 0000 Stack +2 19F5	(DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0018 Stack +0 0000 Stack +2 19F5
0018	mov BX, 0006	BB0600	(AX) = 01F4 (BX) = CE24 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0018 Stack +0 0000 Stack +2 19F5	(AX) = 01F4 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 001B Stack +0 0000 Stack +2 19F5
001B	mov [0000], AX	A30000	(AX) = 01F4 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 001B	(AX) = 01F4 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 001E

			Stack +0 0000 Stack +2 19F5	Stack +0 0000 Stack +2 19F5
001E	mov AL, [BX]	8A07	(AX) = 01F4 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 001E Stack +0 0000 Stack +2 19F5	(AX) = 010C (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0020 Stack +0 0000 Stack +2 19F5
0020	mov AL, [BX+03]	8A4703	(AX) = 010C (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0020 Stack +0 0000 Stack +2 19F5	(AX) = 0109 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0023 Stack +0 0000 Stack +2 19F5
0023	mov CX, [BX+03]	8B4F03	(AX) = 0109 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (DI) = 0000	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (DI) = 0000

			(SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0023 Stack +0 0000 Stack +2 19F5	(SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0026 Stack +0 0000 Stack +2 19F5
0026	mov DI, 0002	BF0200	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (DI) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0026 Stack +0 0000 Stack +2 19F5	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0029 Stack +0 0000 Stack +2 19F5
0029	mov AL, [000E+DI]	8A850 E00	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0029 Stack +0 0000	(AX) = 0128 (BX) = 0006 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 002D Stack +0 0000

			Stack +2 19F5	Stack +2 19F5
002D	mov BX, 0003		(AX) = 0128 (BX) = 0006 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 002D Stack +0 0000 Stack +2 19F5	(AX) = 0128 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0030 Stack +0 0000 Stack +2 19F5
0030	mov AL, [0016+BX+DI]		(AX) = 0128 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0030 Stack +0 0000 Stack +2 19F5	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0034 Stack +0 0000 Stack +2 19F5
0034	mov AX, 1A07		(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014	(AX) = 1A07 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014

			(CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0034 Stack +0 0000 Stack +2 19F5	(CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0037 Stack +0 0000 Stack +2 19F5
0037	mov ES, AX		(AX) = 1A07 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (IP) = 0037 Stack +0 0000 Stack +2 19F5	(AX) = 1A07 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0039 Stack +0 0000 Stack +2 19F5
0039	mov AX, ES: [BX]		(AX) = 1A07 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0039 Stack +0 0000 Stack +2 19F5	(AX) = 00FF (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 003C Stack +0 0000 Stack +2 19F5
003C	mov AX, 0000		(AX) = 00FF	(AX) = 0000

			(BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 003C Stack +0 0000 Stack +2 19F5	(BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 003F Stack +0 0000 Stack +2 19F5
003F	mov ES, AX		(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 003F Stack +0 0000 Stack +2 19F5	(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 0000 (IP) = 0041 Stack +0 0000 Stack +2 19F5
0041	push DS		(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A	(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0012 (CS) = 1A0A

			(DS) = 1A07 (ES) = 0000 (IP) = 0041 Stack +0 0000 Stack +2 19F5 Stack +4 0000	(DS) = 1A07 (ES) = 0000 (IP) = 0042 Stack +0 1A07 Stack +2 0000 Stack +4 19F5
0042	Pop ES		(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0012 (CS) = 1A0A (DS) = 1A07 (ES) = 0000 (IP) = 0042 Stack +0 1A07 Stack +2 0000 Stack +4 19F5	(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0043 Stack +0 0000 Stack +2 19F5 Stack +4 0000
0043	Mov CX, ES: [BX-01]		(AX) = 0000 (BX) = 0003 (CX) = 0509 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0043 Stack +0 0000	(AX) = 0000 (BX) = 0003 (CX) = FFCE (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0047 Stack +0 0000

			Stack +2 19F5 Stack +4 0000	Stack +2 19F5 Stack +4 0000
0047	Xchg AX, CX		(AX) = 0000 (BX) = 0003 (CX) = FFCE (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0047 Stack +0 0000 Stack +2 19F5 Stack +4 0000	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0048 Stack +0 0000 Stack +2 19F5 Stack +4 0000
0048	Mov DI, 0002		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 0048 Stack +0 0000 Stack +2 19F5 Stack +4 0000	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 004B Stack +0 0000 Stack +2 19F5 Stack +4 0000
004B	Mov ES:[BX + DI], AX		(AX) = FFCE (BX) = 0003 (CX) = 0000	(AX) = FFCE (BX) = 0003 (CX) = 0000

			(DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 004B Stack +0 0000 Stack +2 19F5 Stack +4 0000	(DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (IP) = 004E Stack +0 0000 Stack +2 19F5 Stack +4 0000
004E	Mov BP, SP		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0000 (IP) = 004E Stack +0 0000 Stack +2 19F5 Stack +4 0000	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0014 (IP) = 0050 Stack +0 0000 Stack +2 19F5 Stack +4 0000
0050	Push [0000]		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0014	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0012

			(CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0014 (IP) = 0050 Stack +0 0000 Stack +2 19F5 Stack +4 0000	(CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0014 (IP) = 0054 Stack +0 01F4 Stack +2 0000 Stack +4 19F5
0054	Push [0002]		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0012 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0014 (IP) = 0054 Stack +0 01F4 Stack +2 0000 Stack +4 19F5 Stack +6 19F5	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0014 (IP) = 0058 Stack +0 FFCE Stack +2 01F4 Stack +4 0000 Stack +6 19F5
0058	Mov BP, SP		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0010 (CS) = 1A0A	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0010 (CS) = 1A0A

			(DS) = 1A07 (ES) = 1A07 (BP) = 0014 (IP) = 0054 Stack +0 FFCE Stack +2 01F4 Stack +4 0000 Stack +6 19F5	(DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = 005A Stack +0 FFCE Stack +2 01F4 Stack +4 0000 Stack +6 19F5
005A	Mov DX, [BP+02]		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 0000 (DI) = 0002 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = 005A Stack +0 FFCE Stack +2 01F4 Stack +4 0000 Stack +6 19F5	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 01F4 (DI) = 0002 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = 005D Stack +0 FFCE Stack +2 01F4 Stack +4 0000 Stack +6 19F5
005D	Ret Far 0002		(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 01F4 (DI) = 0002 (SP) = 0010 (CS) = 1A0A	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 01F4 (DI) = 0002 (SP) = 0016 (CS) = 1A0A

			(DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = 005D Stack +0 FFCE Stack +2 01F4 Stack +4 0000 Stack +6 19F5	(DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = FFCE Stack +0 19F5 Stack +2 0000 Stack +4 0000 Stack +6 0000
FFCE			(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 01F4 (DI) = 0002 (SP) = 0016 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = FFCE Stack +0 19F5 Stack +2 0000 Stack +4 0000 Stack +6 0000	(AX) = FFCE (BX) = 0003 (CX) = 0000 (DX) = 01F4 (DI) = 0002 (SP) = 0016 (CS) = 1A0A (DS) = 1A07 (ES) = 1A07 (BP) = 0010 (IP) = FFCE Stack +0 19F5 Stack +2 0000 Stack +4 0000 Stack +6 0000 Программа не завершается

Компоненты программы см. в приложении Б.

Выводы.

В ходе выполнения работы были изучены способы взаимодействия с массивами, режимы адресации и формирования исполнительного адреса

ПРИЛОЖЕНИЕ А

Тексты компонентов программы LR2.exe

LR2.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 12,11,10,9,5,6,7,8

vec2 DB -40,-50,40,50,-20,-30,20,30

matr DB 5,6,7,8,-8,-7,-6,-5,1,2,3,4,-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]
```

```
push mem1
push mem2
mov bp,sp
mov dx,[bp]+2
ret 2
```

```
Main    ENDP
CODE    ENDS
END Main
```

LR2.1st

Microsoft (R) Macro Assembler Version 5.10

9/29/21 00:44:11

Page 1-1

$$; \text{P}\overline{\text{U}}\text{C}\overline{\text{T}}\text{P}\text{sPiC}\overline{\text{T}}\text{P}^{\circ}\text{PjPjP}^{\circ} \text{P}\overline{\text{e}}\text{P}\cdot\text{C}\acute{\text{r}}\text{C}\ddagger\text{P}_{\mu}\text{PSP}\overline{\text{e}}\text{C}\overline{\text{U}}$$
CTbP_μP¶PëP

jPsPI P°PrCтPμCΓP°C†PëPë

$$\text{P}^{\text{i}}\text{C}\bar{\text{b}}\text{P}_\text{s}\text{C}^{\dagger}\text{P}_\mu\text{C}\bar{\Gamma}\text{C}\bar{\Gamma}\text{P}_\text{s}\text{C}\bar{\text{b}}\text{P}^{\circ}\text{I}$$

ntelX86

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

$$; P\check{Y}C, P_{\mu}P_{\epsilon} \quad P\ddot{i}C\check{T}P_sP\ddot{i}C\check{T}P^{\circ}P_jP_jC\langle$$

```

0000                                     AStack SEGMENT STACK
0000 000C[                               DW 12 DUP(?)
      ????
```

```

0018          AStack ENDS

;P”P°PSPSC<Pμ PïCṪPsPiCṪP°PjPjC<

0000          DATA    SEGMENT

;P”PëCṪPμPeC,PëPIC< PsPiPëCÍP°PSPëCİ

PrP°PSPSC<

C...

0000 0000          mem1    DW    0
0002 0000          mem2    DW    0
0004 0000          mem3    DW    0
0006 0C 0B 0A 09 05 06      vec1    DB    12,11,10,9,5,6,7,8
07 08
000E D8 CE 28 32 EC E2      vec2    DB    -40,-50,40,50,-20,-30,20,30
14 1E
0016 05 06 07 08 F8 F9matr    DB    5,6,7,8,-8,-7,-6,-5,1,2,3,4,-4,
-3,-2,-1
FA FB 01 02 03 04
FC FD FE FF

0026          DATA    ENDS

; PљPsPr PïCṪPsPiCṪP°PjPjC<

0000          CODE     SEGMENT

          ASSUME CS:CODE, DS:DATA, SS:AStack

; P“PsP»PsPIPSP°Cİ PïCṪPsC†PμPrCíCṪP°

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

; P_цP P_hP'P•P P_лP_h P P•P–P★P_нP_hP'

P_hP''P P•P_ŸP_h

P_l'P★P★P_кP_h PJP P_hP'P_кP•

P_ŸP_нP•P©P•P_кP★P™

; P P_μPiP_ëC_ГC,C_нP_sPIP°C_Ц

P°PrC_нP_μC_ГP°C†P_ëC_Ц

0009 B8 01F4	mov ax,n1
000C 8B C8	mov cx,ax
000E B3 24	mov bl,EOL
0010 B7 CE	mov bh,n2

; P_цC_нC_ЦPjP°C_Ц P°PrC_нP_μC_ГP°C†P_ëC_Ц

0012 C7 06 0002 R FFCE	mov mem2,n2
------------------------	-------------

0018 BB 0006 R mov bx,OFFSET vec1

001B A3 0000 R mov mem1,ax

; P_лP_sC_ГPIP_μPSPSP°C_Ц P°PrC_ТP_μC_ГP°C†P_ëC_Ц

001E 8A 07 mov al,[bx]

mov mem3,[bx]

LR2.asm(47): error A2052: Improper operand type

; P°P°P·P_ëC_ТP_sPIP°PSPSP°C_Ц

P°PrC_ТP_μC_ГP°C†P_ëC_Ц

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

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

; P★PSPPrP_μP_ëC_ГPSP°C_Ц P°PrC_ТP_μC_ГP°C†P_ëC_Ц

0026 BF 0002 mov di,ind

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

002D 8B 8D 000E R mov cx,vec2[di]

LR2.asm(54): warning A4031: Operand types must match

; P_ђPrC_ТP_μC_ГP°C†P_ëC_Ц C_Г

P±P°P·P_ëC_ТP_sPIP°PSP_ëP_μ

P_j P_ë P_ëPSPPrP_μP_ëC_ГP_ëC_ТP_sPIP°PSP_ëP_μP_j

0031 BB 0003 mov bx,3

0034 8A 81 0016 R mov al,matr[bx][di]

0038 8B 89 0016 R mov cx,matr[bx][di]

LR2.asm(58): warning A4031: Operand types must match

003C 8B 85 0022 R mov ax,matr[bx*4][di]

LR2.asm(59): error A2055: Illegal register value

; P_uP P_hP'P•P P_лP_h P P•P–P★P_нP_hP'

P_hP''P P•P_ŸP_h

P_iP★P★P_ŸPJP_§P•P_ŸP_hP_нP_ŸP•P“P_нP•P_кP_ŸP_hP'

; P_uP_μC_тP_μP_sP_iC_тP_μP_rP_μP»P_μPSP_ëP_μ

C_тP_μP_iP_jP_μPSC

,P°

; ----- PIP°C_тP_ëP°PSC, 1

0040 B8 ---- R mov ax, SEG vec2

0043 8E C0 mov es, ax

0045 26: 8B 07 mov ax, es:[bx]

0048 B8 0000 mov ax, 0

; ----- PIP°C_тP_ëP°PSC, 2

004B 8E C0 mov es, ax

004D 1E push ds

004E 07 pop es

004F 26: 8B 4F FF mov cx, es:[bx-1]

0053 91 xchg cx,ax

; ----- PIP°C_тP_ëP°PSC, 3

0054 BF 0002 mov di,ind

0057 26: 89 01 mov es:[bx+di],ax

; ----- PIP°C_тP_ëP°PSC, 4

005A 8B EC mov bp,sp

005C 3E: 8B 86 0016 R mov ax,matr[bp+bx]

LR2.asm(79): error A2046: Multiple base registers

0061 3E: 8B 83 0016 R mov ax,matr[bp+di+si]

LR2.asm(80): error A2047: Multiple index registers

; P★C_тP_iP_sP»C_тP•P_sPIP°PSP_ëP_μ

C_тP_μP_iP_jP_μPSC,P°

C_тC,P_μP_ëP°

0066 FF 36 0000 R push mem1

```

006A FF 36 0002 R          push mem2
006E 8B EC                mov bp,sp
0070 8B 56 02             mov dx,[bp]+2
0073 CA 0002             ret 2
0076                      Main  ENDP
LR2.asm(87): error A2006: Phase error between passes
0076                      CODE  ENDS
                      END Main

```

Microsoft (R) Macro Assembler Version 5.10 9/29/21 00:44:11
Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA		STACK
CODE	0076	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 = 0076
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
@VERSION ..... TEXT 510

```

89 Source Lines

89 Total Lines

19 Symbols

47836 + 459424 Bytes symbol space free

2 Warning Errors

5 Severe Errors

ПРИЛОЖЕНИЕ Б

Тексты компонентов программы LR2_FIX.exe

LR2_FIX.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 12,11,10,9,5,6,7,8
vec2 DB -40,-50,40,50,-20,-30,20,30
matr DB 5,6,7,8,-8,-7,-6,-5,1,2,3,4,-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

```



```
; PµCḐPsPiCḐP°PjPjP° PëP·CřC‡PµPSPëCḐ CḐPµP¶PëP
jPsPI P°PrCḐPµCḐP°C‡PëPë PiCḐPsC‡PµCḐCḐPsCḐP° I
ntelX86
```

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

```
; PŸC,PµPe PiCḐPsPiCḐP°PjPjC<
```

```
0000          AStack SEGMENT STACK
0000 000C[          DW 12 DUP(?)
          ????
```

```
]
```

```
0018          AStack ENDS
```

```
;P”P°PSPSC<Pµ PiCḐPsPiCḐP°PjPjC<
```

```
0000          DATA SEGMENT
```

```
;P”PëCḐPµPeC,PëPIC< PsPiPëCḐP°PSPëCḐ PrP°PSPSC<
C...
```

```
0000 0000          mem1 DW 0
0002 0000          mem2 DW 0
0004 0000          mem3 DW 0
```

```

0006 0C 0B 0A 09 05 06      vec1  DB  12,11,10,9,5,6,7,8
                                07 08
000E D8 CE 28 32 EC E2      vec2  DB  -40,-50,40,50,-20,-30,20,30
                                14 1E
0016 05 06 07 08 F8 F9      matr  DB  5,6,7,8,-8,-7,-6,-5,1,2,3,4,-4,
                                -3,-2,-1
                                FA FB 01 02 03 04
                                FC FD FE FF

```

```

0026                                DATA  ENDS

```

```

; PлPsPr PiCтPsPiCтP°PjPjCг

```

```

0000                                CODE  SEGMENT
                                ASSUME CS:CODE, DS:DATA, SS:AStack

```

```

; P“PsP»PsPIPSP°Cл PiCтPsC†PμPrCгCтP°

```

```

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

```

```

; PлP PhP’P•P PлPh P P•P–P ★ PлPhP’ PhP”P P•PŸPh
PгP ★ P ★ PкPh PJP PhP’PкP• PŸPлP•P©P•PкP ★ P™
; P PμPiPëCгCтPsPIP°Cл P°PrCтPμCгP°C†PëCл

```

```

0009 B8 01F4                        mov  ax,n1
000C 8B C8                          mov  cx,ax
000E B3 24                          mov  bl,EOL
0010 B7 CE                          mov  bh,n2

```

; P_uC₇C₀P_jP^oC₁ P^oP_rC₇P_μC₁ΓP^oC₁†P_ëC₁

0012 C7 06 0002 R FFCE mov mem2,n2

```

0018 BB 0006 R      mov bx,OFFSET vec1
001B A3 0000 R      mov mem1,ax
                   ; PЛPсГПIPμPSPSP°CЦ P°PrCТPμГП°C†PөCЦ
001E 8A 07          mov al,[bx]
                   ;mov mem3,[bx]
                   ; P°P°P·PөCТPсPIP°PSPSP°CЦ P°PrCТPμГП°C†PөCЦ
0020 8A 47 03       mov al,[bx]+3
0023 8B 4F 03       mov cx,3[bx]
                   ; P★PSPPrPμPөГПPSP°CЦ P°PrCТPμГП°C†PөCЦ
0026 BF 0002        mov di,ind
0029 8A 85 000E R    mov al,vec2[di]
                   ;mov cx,vec2[di]
                   ; PђPrCТPμГП°C†PөCЦ ГП P±P°P·PөCТPсPIP°PSPөPμ
                   Pј Pө PөPSPPrPμPөГПPөCТPсPIP°PSPөPμPј
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]
                   ; PцP PђP'P·P PЛPђ P P·P–P★PђPђP' PђP''P P·PЎPђ
                   PіP★P★ PЎ PJP§P·PђPђPђ PЎP·P°PђP·PкPђPђP'
                   ; PцPμCТPμPсPіCТPμPrPμP»PμPSPөPμ
CГPμPіPјPμPSC
                   ,P°
                   ; ----- PIP°CТPөP°PSC, 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
		; ----- PIP°CṪPḡP°PSC, 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
		; ----- PIP°CṪPḡP°PSC, 3
0048	BF 0002	mov di,ind
004B	26: 89 01	mov es:[bx+di],ax
		; ----- PIP°CṪPḡP°PSC, 4
004E	8B EC	mov bp,sp
		;mov ax,matr[bp+bx]
		;mov ax,matr[bp+di+si]
		; P★CṪPḡPṡP»CḤP·PsPIP°PSPḡPμ CṪPμPiPjPμPSC,P°
		CṪC,PμPeP°
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 LR2_FIX
@VERSION TEXT 510

89 Source Lines

89 Total Lines

19 Symbols

47816 + 459444 Bytes symbol space free

0 Warning Errors

0 Severe Errors