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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Организация ЭВМ и систем»

ТЕМА: ИЗУЧЕНИЕ РЕЖИМОВ АДРЕСАЦИИ И ФОРМИРОВАНИЯ ИСПОЛНИТЕЛЬНОГО АДРЕСА.

Студент гр.0382 _____ Диденко Д.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить режимы адресации и формирования исполнительно адреса.

Задание.

Вариант 4.

Данные:

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

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.

Основные теоретические положения.

Все имеющиеся способы адресации можно условно разделить на три группы: регистровая, непосредственная и с указанием адреса памяти. При этом адрес памяти можно задавать по-разному: прямым указанием символического обозначения ячейки памяти, указанием регистра, в котором хранится требуемый адрес, или и того и другого. Таким образом, третья группа включает, в сущности, целый ряд способов адресации. Они обычно носят названия: прямая, базовая, индексная, базово-индексная, а также базовая,

индексная или базово-индексная со смещением.

1. Регистровая адресация

Операнд (байт или слово) находится в регистре. Способ применим ко всем программно-адресуемым регистрам процессора. Примеры:

push DS ;Сохранение DS в стеке

mov BP, SP ;Пересылка содержимого SP в BP

2. Непосредственная адресация

Операнд (байт или слово) может быть представлен в виде числа, адреса, кода ASCII, а также иметь символьное обозначение. Примеры:

mov AX, 4C00h ;Операнд - 16-ричное число

mov DX, offset mas ;Смещенке массива mas заносится в DX

mov DL, '!' ; Операнд - код ASCII символа '!'

num = 9; Число 9 получает обозначение num

mov CX, num ; Число, обозначенное num, загружается в CX

3. Прямая адресация памяти.

В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Пример:

mov DL, meml ;Содержимое байта памяти с символическим именем meml пересылается в DL

Если нужно обратиться к ячейке памяти с известным абсолютным адресом, то этот адрес можно непосредственно указать в качестве операнда. Предварительно необходимо настроить какой-либо сегментный регистр на начало того участка памяти, в котором находится искомая ячейка. Пример:

mov AX, 0 ; Настроим сегментный регистр ES на

mov ES,AX ;самое начало памяти (адрес 0)

mov AX, ES:[0] ;AX=содержимое слова с адресом 0000h:0000h

mov DX,ES:[2] /OX=содержимое слова с адресом 0000h:0002h

Заметим, что в этом случае сегментный регистр надо указывать обязательно. Все остальные способы адресации относятся к группе косвенной адресации памяти.

3.1. Базовая и индексная адресация памяти.

Относительный адрес ячейки памяти находится в регистре, обозначение которого заключается в квадратные скобки. При использовании регистров BX или BP адресацию называют базовой, при использовании регистров SI или DI - индексной. При адресации через регистры BX, SI или DI в качестве сегментного регистра подразумевается DS; при адресации через BP - регистр SS. Таким образом, косвенная адресация через регистр BP предназначена для работы со стеком. Однако при необходимости можно явно указать требуемый сегментный регистр. Еще раз отметим, что во всех базовых и индексных способах адресации операндом является содержимое ячейки памяти, адрес которой находится в том или ином регистре или вычисляется сложением содержимого двух регистров.

Примеры:

mov AL, [BX] ; Сегментный адрес предполагается в DS, смещение в BX

mov DL, ES:[BX] ; Сегментный адрес в ES, смещение в BX

mov DX, [BP] ; Сегментный адрес в SS, смещение в BP

mov AL, [DI] ; Сегментный адрес в DS, смещение в DI

3.2. Базовая и индексная адресации памяти со смещением

Относительный адрес операнда определяется суммой содержимого регистра (BX, BP, SI или DI) и указанного в команде числа, которое называют смещением.

Пример:

mas db 1,2,5,3,7,9,8,3,4 ;Массив символов

mov BX, 2 ; BX=индекс элемента в массиве

mov DL, mas[BX] ;B DL заносится третий элемент массива

В этом примере относительный адрес адресуемого элемента массива mas вычисляется как сумма содержимого BX (2) и значения символического обозначения mas, которое совпадает с относительным адресом начала массива mas. В результате в регистр DL будет загружен элемент массива mas с индексом

2, т.е. число 5. Предполагается, что базовый адрес сегмента, в который входит массив mas, загружен в DS. Такой же результат даст такая последовательность команд:

mov BX,offset mas ;BX=относительный адрес ячейки mas

mov DL,2[BX]

Здесь относительный адрес адресуемого элемента массива mas вычисляется как сумма содержимого регистра BX и дополнительного смещения, задаваемого константой 2. Последняя команда может быть записана в следующем виде:

mov DL, [BX+2]

mov DL, [BX]+2

Адресация с помощью регистров SI и DI осуществляется аналогично. При использовании регистра BP следует помнить, что в качестве сегментного регистра по умолчанию подразумевается регистр SS.

3.3. Базово-индексная адресация памяти

Относительный адрес операнда определяется суммой содержимого базового и индексного регистров.

Допускается использование следующих пар:

[BX][SI]

[BX][DI]

[BP][SI]

[BP][DI]

Если в качестве базового регистра выступает BX, то в качестве сегментного подразумевается DS (первые две команды); при использовании в качестве базового регистра BP сегментным регистром по умолчанию назначается SS (вторые две команды). При необходимости можно явно указать требуемый сегментный регистр.

Примеры:

mov BX, [BP][SI] ;В BX засылается слово из стека (сегментный адрес в SS), а смещение вычисляется как сумма содержимого BP и SI

mov BX, ES:[BP][SI] ;В BX засылается слово из сегмента, адрес которого находится в ES, а смещение вычисляется как сумма содержимого BP и SI

mov ES:[BX+DI],AX ;В ячейку памяти, сегментный адрес которой хранится в ES, а смещение равно сумме содержимого BX и DI, пересылается содержимое AX

3.4. Базово-индексная адресация памяти со смещением

Относительный адрес операнда определяется суммой трех величин: содержимого базового и индексного регистров, а также дополнительного смещения. Допускается использование тех же пар регистров, что и в базово-индексном способе; так же действующие правила определения сегментных регистров.

Примеры:

mov mas [BX] [SI] , 10 ;Число 10 пересылается в ячейку памяти,

; сегментный адрес которой хранится в DS, а

;смещение равно сумме содержимого

;BX и SI и смещения ячейки mas

mov AX,[BP+2+DI] ;В AX пересылается из стека слово,

;смещение которого равно сумме

;BP, DI и "добавки" 2

Значительная часть рассмотренных выше способов адресации служит для обращения к ячейкам памяти. Таким образом, один и тот же конечный результат можно получить с помощью различных способов адресации. Например, все три приведенные ниже команды:

mov DL,mas+3

mov DL,mas[BX] ;В BX заранее занесено число 3

mov DL,[SI][BX] ;В BX заранее занесено число 3, а в SI - смещение mas

приведут к загрузке в регистр DL четвертого элемента массива mas (если выполняются описанные в комментариях условия). Однако команды с

использованием различных способов адресации занимают различный объем памяти и выполняются за разное время. Так, первая из приведенных выше команд потребует для выполнения 15 машинных тактов, вторая - 18, а третья - 16. Разница невелика, однако при многократном выполнении команд в циклах суммарный эффект может быть значителен. С другой стороны, первые две команды занимают в памяти по 4 байта, а третья - только 2. Таким образом, тщательный выбор способов адресации позволяет в какой-то степени оптимизировать программы по времени выполнения или требуемой памяти, а иногда и по тому и по другому.

Ход выполнения лабораторной 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.

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

Выполнение работы.

1. lab2_err.asm(46): error A2052: Improper operand type.

Выражение: *mov mem3,[bx];*

Неподходящий тип операндов. Нельзя читать из памяти и писать в память одной командой.

2. lab2_err.asm(54): warning A4031: Operand types must match.

Выражение: *mov cx,vec2[di]*;

Несоответствие типов операндов. Размер элементов массива «vec2» 1 байт, а «cx» - 2 байта.

3. lab2_err.asm(58): warning A4031: Operand types must match.

Выражение:

Несоответствие типов операндов. Размер элементов матрицы «matr» 1 байт, а «cx» - 2 байта.

4. lab2_err.asm(59): error A2055: Illegal register value.

Выражение: *mov ax, matr[bx*4][di]*;

Незаконное использование регистра. Нельзя умножать 16-битные регистры.

5. lab2_err.asm(62): error A2046: Multiple base registers.

Выражение: *mov ax,matr[bp+bx];*

Слишком много базовых регистров. Нельзя использовать более одного базового регистра. Несоответствие типов операндов.Размер элементов матрицы ‘matr' 1 байт, а ‘ax' - 2 байта.

6. lab2_err.asm(63): error A2047: Multiple index registers.

Выражение: *mov ax,matr[bp+di+si];*

Слишком много индексных регистров, нельзя использовать более одного индексного регистра. Слишком много регистров, нельзя использовать более двух регистров. Несоответствие типов операндов, размер элементов матрицы ‘matr' 1 байт, а ‘ax' - 2 байта.

Пояснения корректных методов адресации представлено в табл. 1.

Таблица 1 – Пояснения корректных методов адресации.

Команда	Краткое пояснение
mov ax, n1	поместить число n1 в AX
mov cx, ax	поместить значение AX в CX
mov bl, EOL	поместить символ EOL в BL
mov bh, n2	поместить число n2 в BH
mov mem2, n2	поместить n2 в память по адресу mem2
mov bx, OFFSET vec2	поместить смещение массива vec2 в BX

mov meml, ax	поместить значение AX в память по адресу meml
mov al, [bx]	поместить байт по адресу BX в AL
mov al, [bx]+3	поместить байт по адресу BX+3 в AL
mov ex, 3[bx]	поместить слово по адресу BX+3 в CX
mov di, ind	поместить число ind в DI
mov al, vec2[di]	поместить байт по адресу vec2+di в al
mov bx, 3	поместить число 3 в BX
mov al, matr[bx][di]	поместить байт по адресу matr+bx+di в al
mov ax, SEG vec2	поместить сегмент, в котором vec2 в AX
mov es, ax	поместить значение AX в сегментный регистр ES
mov ax, es:[bx]	поместить слово по адресу ES:[BX] в AX
mov ax, 0	поместить число 0 в AX
mov es, ax	поместить значение AX в сегментный регистр ES
push ds	поместить значение DS в стек
pop es	извлечь значение из стека и поместить его в ES
mov ex, es:[bx-l]	поместить слово по адресу ES:[BX-1] в CX
xchg ex, ax	поместить значение AX в CX, а CX в AX
mov di, ind	поместить число ind в DI
mov es:[bx + di], ax	поместить значение AX в слово по адресу ES:[BX+DI]
mov bp, sp	поместить значение SP в BP
mov dx, [bp]+2	поместить в DX слово по адресу BP+2
Ret 2	поместить слово по адресу SS:[SP] в IP, SS:[SP+2] в CS и увеличить SP на б

Результаты хода программы lab2.exe представлены в табл. 2.

Таблица 2 – Ход программы lab2.exe.

№ п/п	Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
				До выполнения	После выполнения
1.	0000	push DS	1E	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000

				(SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 0000 +4 0000 +6 0000 (Flags) =	(SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 19F5 +2 0000 +4 0000 +6 0000 (Flags) = 7200
2.	0001	sub AX,AX	2BC0	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 19F5 +2 0000 +4 0000 +6 0000 (Flags) = 7200	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 19F5 +2 0000 +4 0000 +6 0000 (Flags) = 7244
3.	0003	push AX	50	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000

				(BP) = 0000 (SP) = 0016 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 19F5 +2 0000 +4 0000 +6 0000 (Flags) = 7244	(BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
4.	0004	mov AX,DATA	B8071A	(AX) = 0000 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 1A07 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
5.	0007	mov DS,AX	8ED8	(AX) = 1A07 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000	(AX) = 1A07 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000

				(DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 19F5 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
6.	0009	mov ax,n1	B8F401	(AX) = 1A07 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 1AF4 (BX) = 0000 (CX) = 0096 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
7.	000C	mov cx,ax	8BC8	(AX) = 1AF4 (BX) = 0000 (CX) = 0096 (DX) = 0000	(AX) = 1AF4 (BX) = 0000 (CX) = 01F4 (DX) = 0000

				(SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
8.	000E	mov bl,EOL	B324	(AX) = 1AF4 (BX) = 0000 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 1AF4 (BX) = 0024 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
9.	0010	mov bh,n2	B7CE	(AX) = 1AF4 (BX) = 0024 (CX) = 01F4	(AX) = 1AF4 (BX) = CE24 (CX) = 01F4

				(DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
10.	0012	mov mem2,n2	C7060200CEFF	(AX) = 1AF4 (BX) = CE24 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 1AF4 (BX) = CE24 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
11.	0018	mov bx,OFFSET vec1	BB0600	(AX) = 1AF4 (BX) = CE24	(AX) = 1AF4 (BX) = 0006

				(CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
12.	001B	mov mem1,ax	A30000	(AX) = 1AF4 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 01F4 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
13.	001E	mov al,[bx]	8A4703	(AX) = 01F4	(AX) = 010C

				(BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
14.	0020	mov al,[bx]+3	8A4703	(AX) = 010C (BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 0109 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244

15.	0023	mov cx,3[bx]	8B4F03	(AX) = 0109 (BX) = 0006 (CX) = 01F4 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
16.	0026	mov di,ind	BF0200	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0000 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000

				(Flags) = 7244	(Flags) = 7244
17.	0029	mov al,vec2[di]	8A850E00	(AX) = 0109 (BX) = 0006 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 0128 (BX) = 0006 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
18.	002D	mov bx,3	BB0300	(AX) = 0128 (BX) = 0006 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000	(AX) = 0128 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000

				+6 0000 (Flags) = 7244	+6 0000 (Flags) = 7244
19.	0030	mov al,matr[bx][di]	8A811600	(AX) = 0128 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244
20.	0034	mov bp,sp	8BEC	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0000 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0014 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5

				+4 0000 +6 0000 (Flags) = 7244	+4 0000 +6 0000 (Flags) = 7244
21.	0036	push mem1	AA360000	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0014 (SP) = 0014 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 0000 +2 19F5 +4 0000 +6 0000 (Flags) = 7244	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0014 (SP) = 0012 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 01A4 +2 0000 +4 19F5 +6 0000 (Flags) = 7244
22.	003A	push mem2	FF360200	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0014 (SP) = 0012 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 01A4	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0014 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 FFCE

				+2 0000 +4 19F5 +6 0000 (Flags) = 7244	+2 01A4 +4 0000 +6 19F5 (Flags) = 7244
23.	003E	mov bp,sp	8BEC	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0014 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 FFCE +2 01A4 +4 0000 +6 19F5 (Flags) = 7244	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0010 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 FFCE +2 01A4 +4 0000 +6 19F5 (Flags) = 7244
24.	0040	mov dx,[bp]+2	8B5602	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 0000 (SI) = 0000 (DI) = 0002 (BP) = 0010 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack:	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 01F4 (SI) = 0000 (DI) = 0002 (BP) = 0010 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack:

				+0 FFCE +2 01A4 +4 0000 +6 19F5 (Flags) = 7244	+0 FFCE +2 01A4 +4 0000 +6 19F5 (Flags) = 7244
25.	0043	ret 2	CA0200	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 01F4 (SI) = 0000 (DI) = 0002 (BP) = 0010 (SP) = 0010 (CS) = 1A0A (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 FFCE +2 01A4 +4 0000 +6 19F5 (Flags) = 7244	(AX) = 01F9 (BX) = 0003 (CX) = 0509 (DX) = 01F4 (SI) = 0000 (DI) = 0002 (BP) = 0010 (SP) = 0016 (CS) = 01F4 (DS) = 1A07 (ES) = 19F5 (SS) = 1A05 Stack: +0 1975 +2 0000 +4 0000 +6 0000 (Flags) = 7244

Выводы.

Изучены режимы адресации и формирования исполнительно адреса.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ 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 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]

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
```

Название файла: LAB2.LST

```
                        ; Программа изучения режимов адресации процессо
                        ра IntelX86
 = 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 данных
 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

                        ; 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 АДРЕСАЦИИ 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 базированием 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]

0034  8B EC             mov bp,sp

                                            ;mov ax
                    ,matr[bp+bx]

                                            ;mov ax
                    ,matr[bp+di+si]
                    ; Использование сегмента стека
0036  FF 36 0000 R      push mem1
003A  FF 36 0002 R      push mem2
003E  8B EC             mov bp,sp
0040  8B 56 02          mov dx,[bp]+2
0043  CA 0002           ret 2
```

```
 0046                   Main ENDP
 0046                   CODE ENDS
                         END Main

Microsoft (R) Macro Assembler Version 5.10                  10/2/21
16:06:06
                                                            Symbols-1


Segments and Groups:

                N a m e         Length      Align     Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                0046  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    =
0046
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


     72 Source  Lines
     72 Total   Lines
     19 Symbols

  47904 + 461403 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```