# 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 основной памяти.**

Студент гр. 3388 Потоцкий С.С.

Преподаватель Фирсов М.А.

Санкт-Петербург 2024

# Цель работы.

Ознакомиться с режимами адресации основной памяти.

# Задание.

**Часть 1.**

1. Получить у преподавателя вариант выбора значений исходных данных (массивов) vec1, vec2 и matr из файла lr2+.dat и занести свои данные вместо значений, указанных в приведённой для образца программе.

2. Протранслировать программу с созданием файла диагностических сообщений и объяснить обнаруженные ошибки (error) и предупреждения (warning). Листинг привести в приложении к отчёту. Закомментировать операторы с ошибками в тексте программы, а операторы с предупреждениями оставить без изменения. Объяснения ошибок и предупреждений должны быть приведены в отчёте по лабораторной работе.

3. Снова протранслировать программу и скомпоновать загрузочный модуль. Учесть, что программа учебная и может выполняться только под отладчиком. В автоматическом режиме она выполняться не должна.

4. Выполнить программу в пошаговом режиме под управлением отладчика с фиксацией содержимого **используемых** регистров и ячеек памяти до и после выполнения каждой команды. Разобраться в используемых режимах адресации и получаемых результатах. Результаты прогона программы под управлением отладчика должны быть представлены в отчёте по лабораторной работе в табличном виде, аналогичном указанному в лаб.работе №1.

Вариант 16:

і vec1 і 19,18,17,16,12,13,14,15

16 і vec2 і 30,60,-30,-60,10,40,-10,-40

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

# Основные теоретические положения.

Процессоры Intel x86 используют несколько режимов адресации для указания местоположения операндов, обеспечивая гибкость работы с памятью и регистрами. Всего существует 8 режимов адресации:

1. **Регистровая адресация**: Операнд находится в регистре общего назначения или сегментном регистре. В команде на языке ассемблера указывается имя соответствующего регистра. Это самый быстрый и простой способ доступа к операндам.
2. **Непосредственная адресация**: Операнд задается прямо в коде команды. Например, в арифметических командах или командах пересылки один из операндов может быть указан непосредственно, что удобно для работы с константами.
3. **Прямая адресация**: Операнд находится в памяти, и его адрес известен заранее. Доступ осуществляется напрямую через указание этого адреса или имени переменной. Обычно используется в программах с директивами определения данных для обращения к статическим переменным.
4. **Косвенная адресация**: Вместо явного указания адреса операнда в памяти, он хранится в регистре. Например, адрес может быть записан в EAX, EBX, ECX и других регистрах, и доступ к памяти осуществляется через их содержимое. DS и SS используются для сегментации.
5. **Базовая и индексная адресация**: Доступ к элементам структур данных или массивов осуществляется через регистр, который содержит базовый адрес. Например, в регистрах хранится адрес начала массива, и смещение от этого адреса используется для доступа к отдельным элементам.
6. **Индексная адресация с масштабированием**: Этот метод адресации позволяет работать с массивами слов, двойных слов или учетверенных слов. Регистр, хранящий индекс элемента массива, может быть масштабирован (умножен на 2, 4 или 8), что соответствует размеру элемента массива.
7. **Адресация по базе с индексированием и масштабированием**: Самый универсальный метод адресации, который объединяет базовую и индексную адресацию с возможностью масштабирования индекса. Подходит для работы с любыми типами данных, включая сложные структуры.

# Выполнение работы Часть 1.

1. Изменим содержимое файла lr2\_comp.asm в соответствии с полученным вариантом (см. рис. 1).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1

1. Протранслируем программу lr2\_comp.asm с созданием файла диагностических сообщений при помощи команды masm. В результате трансляции получаем 5 ошибок и 2 предупреждения (см. рис. 2.)

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2

После трансляции были получены ошибки, в которых сначала приведено название файла, в котором была обнаружена ошибка, а в скобках указана строчка программы. Конкретно о каждой ошибке:

Error A2052: Improper operand type – неправильный тип операнда, возникает в строчке mov mem3,[bx] в этой строчке происходит попытка переместить значение из памяти (адресуемой через регистр bx) в память (mem3). Команда mov не поддерживает операцию перемещения данных из одного участка памяти в другой напрямую. Один из операндов должен быть регистром.

Error A2055: Illegal register value – недопустимое значение регистра, возникает в строчке mov ax, matr[bx\*4][di] возникает про попытке умножения регистра bx на 4, но данный регистр невозможно использовать как индекс с масштабированием, что вызывает ошибку.

Error A2046: Multiple base registers – несколько базовых регистров, возникает в строчке mov ax, matr[bp+bx], потому что в одной инструкции используется два базовых регистра, что недопустимо.

Error A2047: Multiple index registers – несколько индексных регистров, возникает в строчке mov ax, m, потому что в одной инструкции используется два индексных регистра di и si, что недопустимо.

Error A2006: Phase error between passes – фазовая ошибка между проходами, возникает, если в коде есть сложные адресации или неправильно использованные регистры, что затрудняет сборку программы на нескольких этапах. Пропадает после исправления четырех предыдущих ошибок.

**Предупреждения:**

Warning A4031: Operand types must match – типы операндов должны совпадать возникает в строчках mov cx, matr[bx][di] и mov cx, vec2[di] возникают из-за того, что типы операднов не соответствуют друг другу, а значит и количество памяти используемых для каждого из них не совпадают.

1. Протранслируем программу снова, после исправления ошибок, а также скомпонуем загрузочный модуль, заметим, что программа не может выполниться в автоматическом режиме (см. рис. 3).

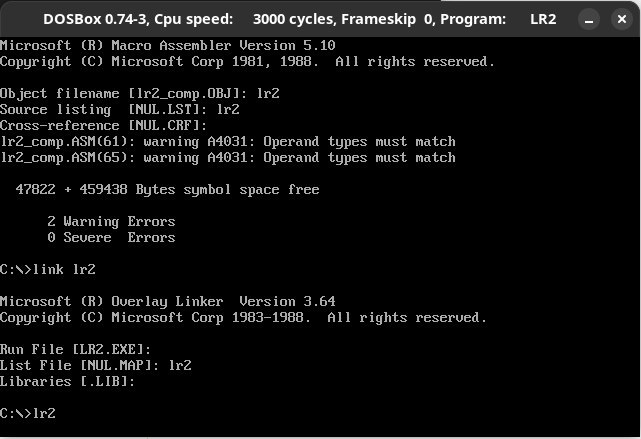


Рисунок 3

Выполним программу в ручном режиме (см. рис. 4)

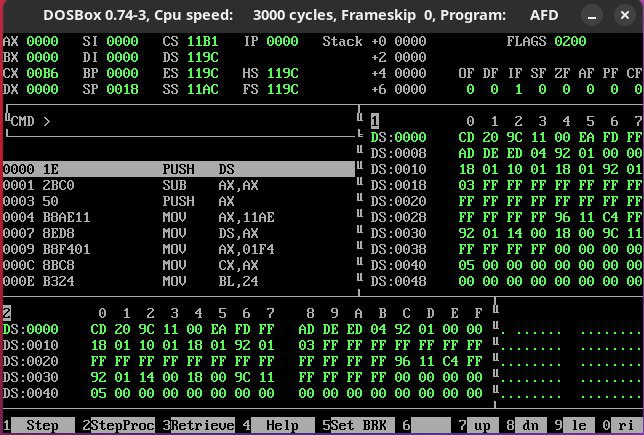


Рисунок 4

1. Пошагово выполним программу и зафиксируем измнения в регистрах и стеке после каждого шага.

Таблица 1 – Протокол пошагового исполнения lr2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Код команды | 16-ый код команды | Содержание регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0000 | PUSH DS | 1E | (sp) = 0018  (ip) = 0000  (ds) = 119C  Stack:  (+0) 0000 | \*(sp) = 0016  \*(ip) = 0001  (ds) = 119C  Stack:  \*(+0) 119C |
| 0001 | SUB AX, AX | 2BC0 | (ax) = 0000  (ip) = 0001 | (ax) = 0000  \*(ip) = 0003 |
| 0003 | PUSH AX | 50 | (sp) = 0016  (ip) = 0003  (ax) = 0000  Stack:  (+0) 119C  (+2) 0000 | \*(sp) = 0014  \*(ip) = 0004  (ax) = 0000  Stack:  \*(+0) 0000  \*(+2) 119C |
| 0004 | MOV AX, 11AE | B8AE11 | (ax) = 0000  (ip) = 0004 | \*(ax) = 11AE  \*(ip) = 0007 |
| 0007 | MOV DS, AX | 8ED8 | (ds) = 119C  (ip) = 0007 | \*(ds) = 11AE  \*(ip) = 0009 |
| 0009 | MOV AX, 01F4 | B8F401 | (ax) = 11AE  (ip) = 0009 | \*(ax) = 01F4  \*(ip) = 000C |
| 000С | MOV CX, AX | 8BC8 | (cx) = 00B6  (ip) = 000C | \*(cx) = 01F4  \*(ip) = 000E |
| 000E | MOV BL, 24 | B324 | (bx) = 0000  (ip) = 000E | \*(bx) = 0024  \*(ip) = 0010 |
| 0010 | MOV BH, CE | B7CE | (ip) = 0010  (bx) = 0024 | \*(ip) = 0012  \*(bx) = CE24 |
| 0012 | MOV [0002], FFCE | C7060200CEFF | (ip) = 0012 | \*(ip) = 0018 |
| 0018 | MOV BX, 0006 | BB0600 | (bx) = CE24  (ip) = 0018 | \*(bx) = 0006  \*(ip) = 001B |
| 001B | MOV [0000], AX | A30000 | (ip) = 001B | (ip) = 001E |
| 001E | MOV AL, [BX] | 8A07 | (ax) = 01F4  (ip) = 001E | \*(ax) = 0113  \*(ip) = 0020 |
| 0020 | MOV AL, [BX+03] | 8A4703 | (ax) = 0113  (ip) = 0020 | \*(ax) = 0110  \*(ip) = 0023 |
| 0023 | MOV CX, [BX+03] | 8B4F03 | (cx) = 01F4  (ip) = 0023 | \*(cx) = 0C10  \*(ip) = 0026 |
| 0026 | MOV DI, 0002 | BF0200 | (di) = 0000  (ip) = 00026 | \*(di) = 0002  \*(ip) = 0029 |
| 0029 | MOV AL, [DI+000E] | BF0200 | (ax) = 0110  (ip) = 0029 | \*(ax) = 01E2  \*(ip) = 002D |
| 002D | MOV CX, [DI+000E] | 8B8DOE00 | (ip) = 002D  (cx) = 0C10 | \*(ip) = 0031  \*(cx) = C4E2 |
| 0031 | MOV BX, 0003 | BB0300 | (bx) = 0006  (ip) = 0031 | \*(bx) = 0003  \*(ip) = 0034 |
| 0034 | MOV AL, [BX+DI+0016] | 8A811600 | (ax) = 01E2  (ip) = 0034 | \*(ax) = 0101  \*(ip) = 0038 |
| 0038 | MOV CX, [BX+DI+0016] | 8B891600 | (cx) = C4E2  (ip) = 0038 | \*(cx) = FD01  \*(ip) = 003C |
| 003C | MOV AX, 11AE | B8AE11 | (ax) = 0101  (ip) = 003C | \*(ax) = 11AE  \*(ip) = 003F |
| 003F | MOV ES, AX | 8EC0 | (es) = 119C  (ip) = 003F | \*(es) = 11AE  \*(ip) = 0041 |
| 0041 | MOV AX, ES:[BX] | 268B07 | (ax) = 11AE  (ip) = 0041 | \*(ax) = 00FF  \*(ip) = 0044 |
| 0044 | MOV AX, 0000 | B80000 | (ax) = 00FF  (ip) = 0044 | \*(ax) = 0000  \*(ip) = 0047 |
| 0047 | MOV ES, AX | 8EC0 | (es) = 11AE  (ip) = 0047 | \*(es) = 0000  \*(ip) = 0049 |
| 0049 | PUSH DS | 1E | (ip) = 0049  Stack:  (+0) 0000  (+2) 119C  (+4) 0000 | \*(ip) = 004A  Stack:  \*(+0) 11AE  \*(+2) 0000  \*(+4) 119C |
| 004A | POP ES | 07 | (sp) = 0012  (es) = 0000  (ip) = 004A  Stack:  (+0) 11AE  (+2) 0000  (+4) 119C | \*(sp) = 0014  \*(es) = 11AE  \*(ip) = 004B  Stack:  \*(+0) 0000  \*(+2) 119C  \*(+4) 0000 |
| 004B | MOV CX, ES:[BX-01] | 268B4FFF | (cx) = FD01  (ip) = 004B | \*(cx) = FFCE  \*(ip) = 004F |
| 004F | XCHG AX, CX | 91 | (ax) = 0000  (cx) = FFCE  (ip) = 004F | \*(ax) = FFCE  \*(cx) = 0000  \*(ip) = 0050 |
| 0050 | MOV DI, 0002 | BF0200 | (di) = 0002  (ip) = 0050 | (di) = 0002  \*(ip) = 0053 |
| 0053 | MOV ES;[BX+DI], AX | 268901 | (es) = 11AE  (ip) = 0053 | (es) = 11AE  \*(ip) = 0056 |
| 0056 | MOV BP, SP | 8BEC | (bp) = 0000  (ip) = 0056 | \*(bp) = 0014  \*(ip) = 0058 |
| 0058 | PUSH [0000] | FF360000 | (sp) = 0014  (ip) = 0058  Stack:  (+0) 0000  (+2) 119C  (+4) 0000 | \*(sp) = 0012  \*(ip) = 005C  Stack:  \*(+0) 01F4  \*(+2) 0000  \*(+4) 119C |
| 005C | PUSH [0002] | FF360200 | (sp) = 0012  (ip) = 005C  Stack:  (+0) 01F4  (+2) 0000  (+4) 119C  (+6) 0000 | \*(sp) = 0010  \*(ip) = 0060  Stack:  \*(+0) FFCE  \*(+2) 01F4  \*(+4) 0000  \*(+6) 119C |
| 0060 | MOV BP, SP | 8BEC | (bp) = 0014  (ip) = 0060 | \*(bp) = 0010  \*(ip) = 0062 |
| 0062 | MOV DX, [BP+02] | 8B5602 | (dx) = 0000  (ip) = 0062 | \*(dx) = 01F4  \*(ip) = 0065 |
| 0065 | RET Far | CB | (sp) = 0010  (cs) = 11B1  (ip) = 0065  Stack:  (+0) FFCE  (+2) 01F4  (+4) 0000  (+6) 119C | \*(sp) = 0014  \*(cs) = 01F4  \*(ip) = FFCE  Stack:  \*(+0) 0000  \*(+2) 119C  \*(+4) 0000  \*(+6) 0000 |

**Вывод.**

В ходе выполнения работы было произведено ознакомление с режимами адресации основной памяти на языке Ассемблера, при помощи отладчика afd.

**ПРИЛОЖЕНИЕ A**

1. **LR2COMP.LST (до исправления)**

Microsoft (R) Macro Assembler Version 5.10 9/23/24 16:37:20

Page 1-1

; Учебная программа лабораторной работы №2 по

дисциплине "Организация ЭВМ и С";

;

= 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 13 12 11 10 0C 0D vec1 DB 19,18,17,16,12,13,14,15

0E 0F

000E 1E 3C E2 C4 0A 28 vec2 DB 30,60,-30,-60,10,40,-10,-40

F6 D8

0016 04 03 FF FE 02 01 matr DB 4,3,-1,-2,2,1,-3,-4,-5,-6,-7,-8

,8,7,6,5

FD FC FB FA F9 F8

08 07 06 05

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

Microsoft (R) Macro Assembler Version 5.10 9/23/24 16:37:20

Page 1-2

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]

lr2\_comp.ASM(54): error A2052: Improper operand type

; Базированная адресация

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

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

; Индексированная адресация

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\_comp.ASM(61): warning A4031: Operand types must match

; Адресация с базированием и индексированием

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\_comp.ASM(65): warning A4031: Operand types must match

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

lr2\_comp.ASM(66): error A2055: Illegal register value

; ПРОВЕРКА АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмента

; ------ вариант 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

; ------ вариант 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

; ------ вариант 3

0054 BF 0002 mov di,ind

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

; ------ вариант 4

005A 8B EC mov bp,sp

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

lr2\_comp.ASM(86): error A2046: Multiple base registers

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

lr2\_comp.ASM(87): error A2047: Multiple index registers

; Использование сегмента стека

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 CB ret

0074 Main ENDP

lr2\_comp.ASM(94): error A2006: Phase error between passes

0074 CODE ENDS

END Main

Microsoft (R) Macro Assembler Version 5.10 9/23/24 16:37:20

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 0074 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 = 0074

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

96 Source Lines

96 Total Lines

19 Symbols

47806 + 459454 Bytes symbol space free

2 Warning Errors

5 Severe Errors

**3. lr2\_comp.asm**

; Учебная программа лабораторной работы №2 по дисциплине "Организация ЭВМ и С";

;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 19,18,17,16,12,13,14,15

vec2 DB 30,60,-30,-60,10,40,-10,-40

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]

; Индексированная 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 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

Main ENDP

CODE ENDS

END Main

**4. LR2.MAP**

Start Stop Length Name Class

00000H 00017H 00018H ASTACK

00020H 00045H 00026H DATA

00050H 000B5H 00066H CODE

Program entry point at 0005:0000