МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

кафедра Информационные системы

09.03.02 Информационные системы и технологии (уровень бакалавриата)

Лабораторная работа №4

по дисциплине: «Инструментальные средства информационных систем»

на тему: «Исследование методов адресации и программирования арифметических и логических операций в 16-разрядных микропроцессорах»

Выполнил

студент 3 курса группы ИС/б-33-о

Генералов Николай Николаевич

Отметка о зачете\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись) (фамилия, инициалы)

г. Севастополь

2018 г.

# ЦЕЛЬ РАБОТЫ

Изучить основные директивы языка ассемблера, исследовать их воздействие на процесс ассемблирования и формирования листинга программы.

Исследовать особенности функционирования блоков 16-разрядного микропроцессора при выполнении арифметических и логических операций и при использовании различных способов адресации. Приобрести практические навыки программирования на языке ассемблера МП 8086 арифметических и логических операций с применением различных способов адресации.

# ПОСТАНОВКА ЗАДАЧИ

1. Изучить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Повторить команды пересылки данных, а также команды арифметических и логических операций.
2. Изучить методы адресации, используемые в 16-разрядных процессорах
3. Составит программу, состоящую из процедур обработки строк п. 2.4 – 2.10.
4. Заполнить 100+10i ячеек области памяти, начинающейся с адреса MAS рядом натуральных чисел. Здесь i – последняя цифра номера зачетной книжки.
5. Переслать массив слов из области памяти, начиная с адреса MAS1 в область с начальным адресом MAS2.
6. Найти в заданном массиве число, равное двум последним цифрам Вашей зачетной книжки и определить его индекс.
7. Переслать в память с адресом 2020:300 диагональные элементы матрицы размером 8×8. Значения элементов матрицы должны быть определены в сегменте данных программы.
8. Произвести отладку разработанных программ в пошаговом режиме и проследить за изменениями содержимого регистров
9. Произвести ассемблирование программы и получить объектный и исполняемый модуль программы в Ехе-формате и ее листинг.

# ХОД РАБОТЫ

Команды микропроцессора Intel 8086 реализуют разнообразные способы адресации, что упрощает организацию и использование сложных структур данных, а также расширяет возможности отдельных команд и повышает гибкость их применения.

1. Регистровая адресация. Операнд находится в одном из общих регистров МП, а в некоторых командах в одном из сегментных регистров.

Примеры:

MOV AX, SI ; <SI>→<AX>

ADD DI, BX ; <BX>+<DI>

AND CL, AX ; Ошибка, несоответствия размеров регистров

XOR AL, AH ; XOR <AL> и <AH>.

2. Непосредственная адресация. Операнды представляют собой константы длиной 8 или 16 бит, содержащиеся в командах. В МП нет команд непосредственной загрузки регистров.

SUB AL, 30H ; <AL>- 48 (30H = 48D)

MOV CL, 10 ; (10→<CL>)

AND AX, 0F000H ; Выделить старших 4 бита в AX

XOR DH, 1 ; Инвертировать младший бит в DH

CMP BL,40H ; Сравнить содержимое BL с числом 64.

3. Прямая адресация. Эффективный адрес берется из поля смещения команды:

MOV AX, GAMMA ; В Акк АХ загружено содержимое ячеек памяти

; с адресом, полученным суммированием

; <DS> сдвинутого на 4 разряда с адресом

; переменной GAMMA, определенном в ячейке программы.

ADD TEMP, BL ; <BL>+ <<DS>↑4+TEMP>

4. Косвенная регистровая. Эффективный адрес (ЕА) находится в одном из базовых или индексных регистров. В косвенной адресации могут использоваться только регистры ВХ, SI, DI. Косвенные регистровые операнды заключают в квадратные скобки.

ADD AX, [DI] ; К <AX> прибавляется содержимое ячейки памяти,

; адрес который находится в DI.

; исполнительный адрес: <DS>↑4+ <DI>.

MOV [SI], CL ; <<SI>> ←< CL>

5. Базовая адресация (база + смещение). ЕА определяется суммой значения смещения, указанного в команде и содержимого регистров ВХ или ВР.

К операциям в памяти можно адресовать, указывая прямой адрес, т.е. называя имя соответствующей области памяти, либо косвенные - через регистры-указатели, или индексные регистры. При прямой адресации 16-ричное смещение автоматически складывается с базовым адресом соответствующего сегмента. При косвенном обращении участвуют один или два из четырех регистров. Они указываются в квадратных скобках [ ] – признак косвенной адресации ВХ, ВР, SI, DI. Если указывается переменная, за ее именем следует выражение в квадратных скобках, которое задает базовые или индексные регистры.

В случае косвенной адресации может быть указан либо только базовый регистр, либо только индексный, либо оба регистра и может быть также указано 8 или 16-битовое смещение.

MOV AX, [BX] ; переслать слово из памяти в АХ. Слово находится в сегменте

; данных, адрес этого сегмента в регистре DS, смещение

; относительно этого адреса в регистре ВХ.

К операндам, находящимся в памяти, можно обратиться одним из четырех способов:

Указанием прямого 16-разрядного смещения

MOV REPORT, AL ;в байт памяти с именем REPORT пересылается содержимое AL

; ([DS]↑4+[REPORT])→AL

Использованием косвенного обращения через базовый регистр, содержимое которого суммируется с 8 или 16-разрядным смещением.

MOV ON[BX+2], AL

MOV BL, ON[BP]

Использованием косвенного обращения через индексный регистр, содержимое которого суммируется со смещением

MOV CL, ITEM[SI+1]

MOV ON[DI+1], CL

Использованием косвенного обращения через базовый и индексный регистры, содержимое которых суммируется со смещением

MOV AH, ITEM[BX+1] [SI+1]

MOV ON[BX+1][DI+1]

При определении ВР в качестве базового регистр обращение осуществляется к текущему сегменту стека SS (если нет префикса замены сегмента). Это делает базовую адресацию с регистром ВР очень удобным средством обращения к данных, находящимся в стеке. Обычно базовый регистр BREG указывает на начало структуры данных, а требуемый элемент адресуется с помощью смещения (расстояния) от базы.

Для обозначения базовой адресации используют два представления:

[BREG] DISP BREG – базовый регистр (ВХ или ВР)

[BREG+ DISP]

Пример записи команд с базовой адресацией:

MOV AX, [BP]10 ; Обе команды передают шестое слово массива, адресуемое ВР,

MOV AX, [BP+10] ; в Аккумулятор,

ADD [BX]TEMP,CX ; Прибавить <СХ> к слову ТЕМР в массиве, адресуемом ВХ.

6. Индексная адресация (смещенная база + индекс). Этот вид адресации называют адресацией с индексированием. Эффективный адрес вычисляется как сумма смещения, находящегося в команде, и содержимого индексного регистра DI или SI. Адресация удобна при доступе к элементам таблицы (массива), когда смещение указывает на начало таблицы (массива), а индекс – на элементы в таблице (массиве). По существу индексная и базовая адресация в МП К1810 аналогичны. Это объясняется тем, что базовые и индексные регистры имеют одинаковую длину. Индексная адресация обозначается в виде TABL[IREG]. Здесь TABL – 16-битовое смещение (адрес начала таблицы).

Примеры:

MOV ADRM [SI], AX ; Передать <АХ> в элемент массива с начальным

; адресом ADRM

ADD CX, MASS [DI] ; Прибавить к <СХ> элемент массива.

; MASS – смещение, указанное в команде.

Пример: Загрузить 3-й элемент массива в аккумулятор АL.

table DB 10, 20, 30, 40

MOV DI, 2 ; загрузить в индексный регистр номер выбираемого байта

; минус 1 (т.к. массив начинается с нулевого элемента)

MOV AL, TABLE [DI] ; загрузить 3-й байт таблицы в AL.

Примеры записи базового индекса адресации:

MOV AX, [BX+2+DI] ; Операнды можно заключать в скобки

MOV AX, [DI+BX+2] ; в любом порядке, а сдвиг можно

MOV AX, [BX+2][DI] ; сочетать с любым из регистров

MOV AX, [BX][DI+2]

7. Базово-индексная адресация (по базе с индексированием). Эффективный адрес равен сумме содержимого базового регистра, индексного регистра и, возможно, смещения, указанного в команде. Этот способ целесообразно использовать при работе с двумерными таблицами. В этом случае базовый регистр содержит начальный адрес массива, а значения смещения и индексного регистра является смещением по строке и столбцу. В ассемблере МП 1810ВМ86 базово-индексная адресация представляется в виде: [BREG]АDR16[IREG].

Пример: загрузить в АХ 16-разрядный элемент таблицы, состоящей из 4-х столбцов и 3-х строк, находящийся в третьей строке на третьей позиции (3,3) [а22, если в таблице считать 0-й столбец и 0-ю строку].

TABLE DW 1024, 1048, 2048, 3600 ; Задание таблицы в начале ассемблерной

DW 4100, 5000, 600, 2000 ; программы

DW 80, 300, 4000, 5000 ;

VALUE DB 2 ; указание номера элемента в строке минус 1 (т.к. считается с 0).

……………………..

MOV BX, TABLE

MOV DI, 16

MOV BX, VALUE [BX][ DI]

8. Относительная адресация. Эффективный адрес вычисляется как сумма фиксированного смещения, находящегося в команде и текущего значения программного счетчика РС. При этом значение РС равно адресу байта, следующего за текущей командой. В МП К1810 относительная адресация применяется только в командах условных и безусловных переходов, вызова подпрограмм и управления итерациями (или циклами). Следует отметить, что программист в ассемблерных программах указывает не значение смещения, а абсолютный адрес перехода, т.е. метку команды, которой необходимо передать управление. Значение смещения выполняется автоматически программа - ассемблер.

9. Адресация цепочек. Для обращения к операциям цепочечных команд, используются индексные регистры. Регистр SI адресует первый байт (слово) цепочки источника, а регистр DI – первый байт (слово) цепочки получателя. В повторяющихся цепочных операциях МП автоматически изменяет содержимое регистров SI и DI.

10. Адресация портов ввода-вывода. Существует прямая и косвенная адресация портов. В прямой адресации номер порта представляет собой 8-битовый непосредственный операнд, находящийся во втором байте команды, что обеспечивает обращение к фиксированным портам 0-255.

При косвенной адресации номер порта находится в регистре DХ и имеет диапазон 0-65535. С помощью предварительной инициализации регистра DХ одна и та же команда может обращаться к любому порту в адресном пространстве ввода-вывода.

Примеры:

IN AL, 40H ; Ввод байта из порта номер 40Н

OUT DX, AX ; Вывод слова в порт с адресом, хранящемся в DХ

IN AX, DX ; Ввод слова из устройства, адрес которого хранится в регистре DХ.

Допустимые символы языка ассемблера состоят из прописных и строчных букв (латинских), цифр, специальных знаков +, -, \*, /, =, (), [], ’, ’’, ., ;, @, &, ?, <, >, % и символов: перевод строки ПС (ОАН), возврат каретки ВК (ОDН), табуляции (О9Н). Любой другой символ воспринимается как пробел. Наименьшей конструкцией модуля является идентификатор – последовательность букв и цифр (не более 31), начинающийся с буквы.

Директивы ассемблера формат:

{имя} директива {операнд(ы)} {; комментарий}

Имя директивы имеет другой смысл по сравнению с меткой и не заканчивается двоеточием. В ряде директив имя отсутствует.

Директивы используются для распределения памяти, связей между модулями, манипуляции с символами и т.д.

Переменная – это единица данных, имеющая имя. Она имеет три атрибута: сегмент, смещение и тип. Сегмент SEG определяет сегмент, содержащий переменную. Смещение OFFSET, расстояние от начала сегмента до переменной, тип – число байтов переменной (1,2 или 4).

Метка, представляющая имя ячейки памяти, имеет атрибутами сегмент, смещение, расстояние. Константа отличается от переменной и метки тем, что она определяет только число. Символьные цепочки заключаются в апострофы и обычно имеют длину до 255 знаков.

Для определения и инициализации данных предназначены директивы:

DB - определить байт (Define Byte)

DW - определить слово (Define Word)

DD - определить двойное слово (Define Double Word).

В ассемблере вводится понятие логического сегмента, под которым понимается часть программы, которая может включать сегменты для машинного кода, данных и стека. Каждый логический сегмент должен начинаться с директивы SEGMENT и заканчиваться директивой ENDS. Логическому сегменту присваивается имя, данное программистом, и список параметров (атрибутов), которые не обязательны, но необходимы в случае программы, включающие несколько модулей.

Параметры директивы SEGMENT.

Выравнивание. Определяет границу начала сегмента. Обычное значение - PARA, по которому сегмент устанавливается на границу параграфа. В этом случае адрес кратен 16. При отсутствии этого операнда ассемблер по умолчанию принимает PARA; адрес сегмента ХХХ0. Бывает: PAGE=ХХ00; WORD=ХХХЕ (четная граница); BYTE=ХХХХ – любая шестнадцатеричная цифра.

Объединение. Определяет, объединяется ли данный сегмент с другими сегментами в процессе компоновки после ассемблирования. Возможны следующие типы объединений:

STACK, COMMON (общий), PUBLIC (общедоступный), AT- и MEMORY.

Все PUBLIC - сегменты, имеющие одинаковое имя или класс, загружаются компоновщиком в смежные области. Все такие сегменты имеют один общий базовый адрес.

Для сегментов COMMON с одинаковыми именами и классами компоновщик устанавливает один общий базовый адрес. При выполнении происходит наложение одного сегмента на другой. Размер общей области определяется самым длинным сегментом.

1. ТЕКСТ ПРОГРАММЫ

page 60,132

TITLE lab4\_exe.asm

STACKSG SEGMENT PARA 'stack'

;DB 16 DUP('STACKSG')

STACKSG ENDS

DATASG SEGMENT PARA 'data'

MY\_LATEST\_NUMS DW 055h

MAS\_SIZE DW 0B4h

MAS 0B4h DUP(?)

MAS1\_2\_SIZE DB 5

MAS1 5 DUP(1,2,3,4,5)

MAS2 25 DUP(?)

;MATRIX\_CALC\_FORMULA: C=(A-i)+B

A DB 10

B DB 5

I DB 5

MATRIX DW 1,2,3,4,5,6,7,8

DW 9,10,11,12,13,14,15,16,17

DW 21,22,23,24,25,26,27,28

DW 39,310,311,312,313,314,315,316,317

DW 31,32,33,34,35,36,37,38

DW 29,210,211,212,213,214,215,216,217

DW 41,42,43,44,45,46,47,48

DW 94,104,114,124,134,144,154,164,174

DATASG ENDS

CODESG SEGMENT PARA 'code'

;INIT MAS ARRAY

INIT\_THE\_MASS PROC

MOV AX, 00h

MOV CX, 0B4h

LEA SI, [MAS + 100h]

cycle:

INC AL

MOV [SI], AL

INC SI

DEC CX

JNZ cycle

RET

INIT\_THE\_MASS ENDP

;COPY ARRAYS

COPY\_MAS1\_TO\_MAS2 PROC

LEA SI, [MAS1 + 100h]

LEA DI, [MAS2 + 100h]

MOV CX, 25

REP MOVSB

RET

COPY\_MAS1\_TO\_MAS2 ENDP

;FIND MY\_LATEST\_NUMS VAL IN ARRAY

FIND\_MY\_NUMS\_IN\_MAS PROC

LEA SI, [MAS + 100h]

MOV AX, 00h

MOV BX, 00h

MOV CX, 0b4h

MOV DX, 055h

find\_cycle:

LODSB

CMP AX, DX

JZ find\_result

INC BX

loop find\_cycle

find\_result:

RET

FIND\_MY\_NUMS\_IN\_MAS ENDP

;MATRIX

FIND\_DIAGONAL\_VAL\_IN\_MATRIX PROC

ASSUME ES:02020h

LEA DI, ES:300h

LEA SI, [MATRIX + 100h]

MOV DX, 08h

MOV CX, 08h

cycle1:

MOVSW

ADD SI, DX

loop cycle1

RET

FIND\_DIAGONAL\_VAL\_IN\_MATRIX ENDP

;MAIN\_PROG

BEGIN PROC FAR

ASSUME SS: STACKSG, CS:CODESG, DS:DATASG, ES:NOTHING

PUSH DS

CALL INIT\_THE\_MASS

CALL COPY\_MAS1\_TO\_MAS2

CALL FIND\_MY\_NUMS\_IN\_MAS

CALL FIND\_DIAGONAL\_VAL\_IN\_MATRIX

BEGIN ENDP

CODESG ENDS

END BEGIN

1. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

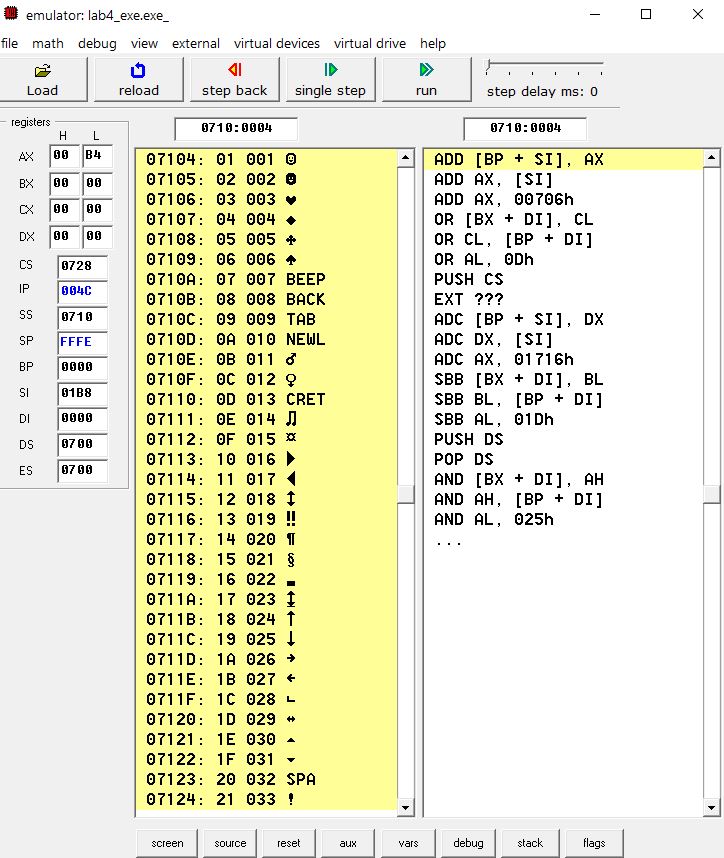


Рисунок 2 – Инициализация данных массива согласно п.2.4

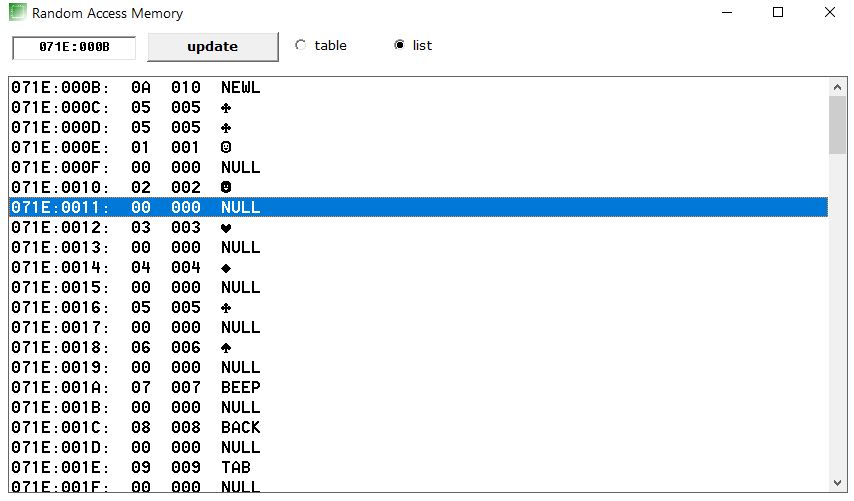


Рисунок 2 – Инициализация данных массива MAS1 согласно п.2.5

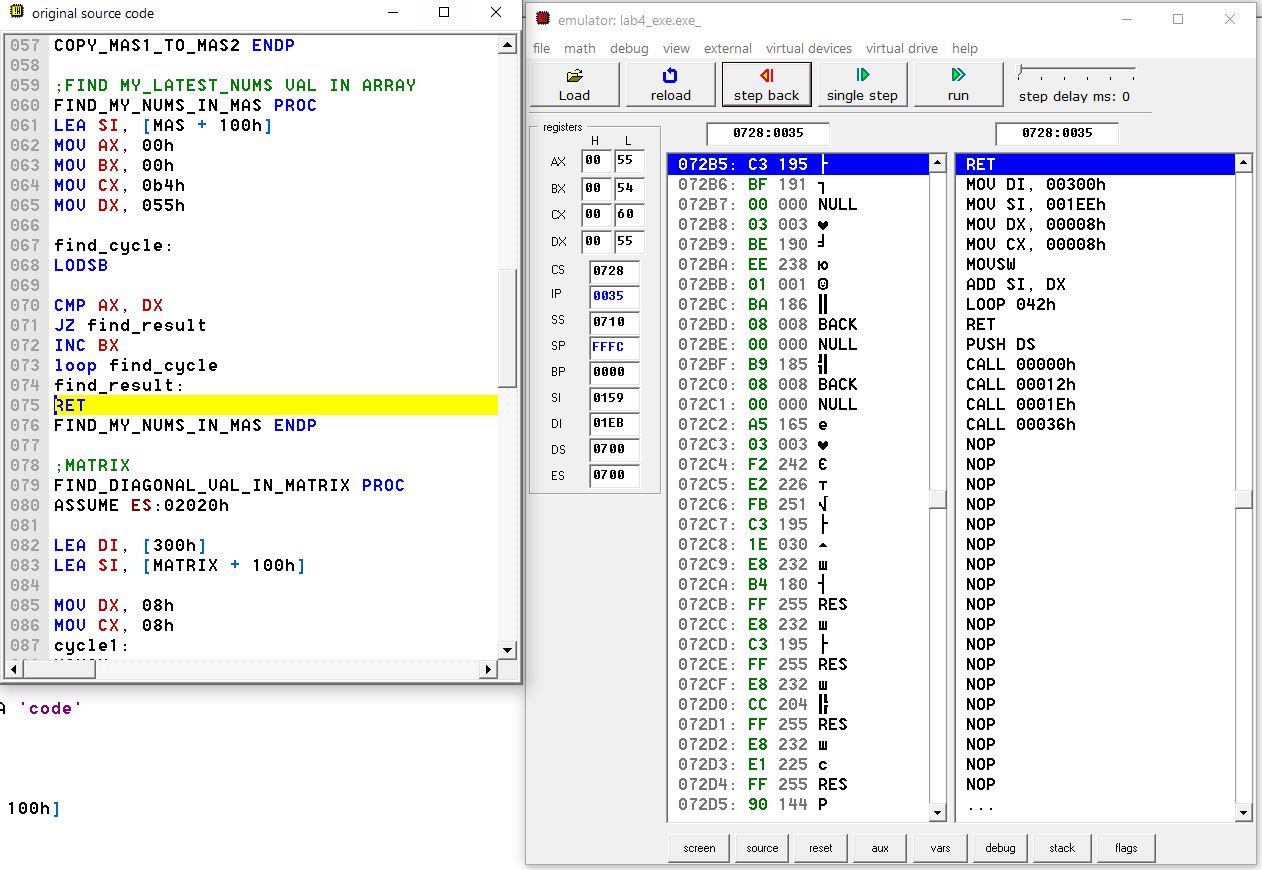


Рисунок 3 – Результат выполнения подпрограммы поиска числа п.2.6

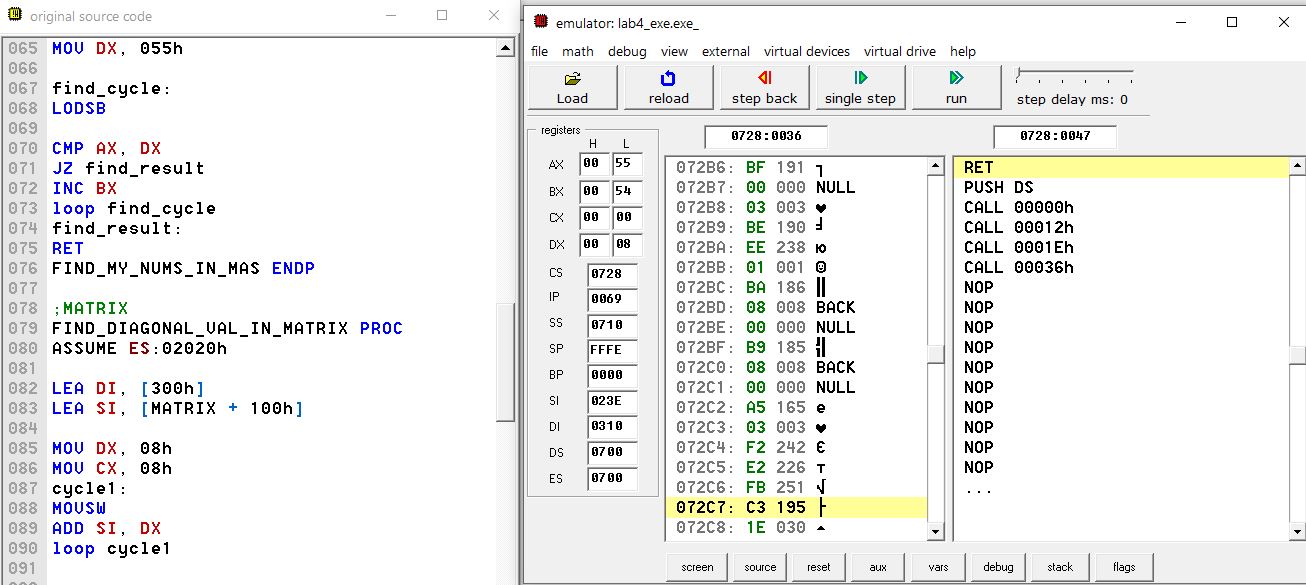


Рисунок 4 – Результат выполнения подпрограммы записи диагональных элементов матрицы

ВЫВОД

В ходе выполнения лабораторной работы были изучены основные директивы языка ассемблера, исследовано их воздействие. Были приобретены практические навыки программирования на языке ассемблера МП 8086 арифметических и логических операций с применением различных способов адресации. Для закрепления практических навыков была написана программа на языке ассемблера МП 8086 в которой выполняется пересылка данных из одной области памяти в другую, поиск значения в массиве данных и поиск диагональных элементов в матрице.