# ЛАБОРАТОРНАЯ РАБОТА №5

# ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ОБРАБОТКИ СТРОКОВЫХ ДАННЫХ

Цель работы

Изучить основные команды языка ассемблера для обработки строковых данных и команды передачи управления, исследовать их воздействие на процесс ассемблирования и формирования листинга программы.

Исследовать особенности функционирования блоков 16-разрядного микропроцессора при выполнении команд обработки строк и передачи управления. Приобрести практические навыки программирования на языке ассемблера МП 8086 задач обработки линейных массивов.

Задачи

1. Повторить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Изучить команды строковых операций;
2. Изучить команды передачи управления в 16-разрядных процессорах и особенности оформления программ в ехе- и сом-форматах;
3. Составить программу, состоящую из следующих процедур обработки строк:
   1. Исследовать программу пересылки массивов, приведённую в приложении. Оптимизировать данную программу за счёт использования префикса повторения;
   2. Заполнить ячеек области памяти, начинающихся с адреса MAS1, рядом натуральных чисел. Здесь – последняя цифра номера зачётной книжки ();
   3. Переслать массив слов из области памяти, начиная с адреса MAS1, в область с начальным адресом MAS2;
   4. Найти в заданном массиве число, равное двум последним цифрам зачётной книжки (93), и определить его индекс;
   5. Вычислить сумму элементов массива MAS1, созданного п. 3.1;
4. Произвести отладку разработанных программ в пошаговом режиме и проследить за изменениями содержимого регистров;
5. Произвести ассемблирование программ и получить объектный и исполняемый модуль программ в eхе-формате и их листинг;
6. Рассчитать время выполнения программ;

Ход работы

Была исследована программа, представленная в приложении (Листинг 1). Она переносит данные из массива, начинающегося с ячейки NAME1 в массивы NAME2 и NAME3. В первом случае для этого используется метка B20 и оператор условного перехода JNZ, во втором – метка C20 и оператор цикла LOOP. Перенос данных осуществляется операцией MOV через посредника в виде регистра AX.

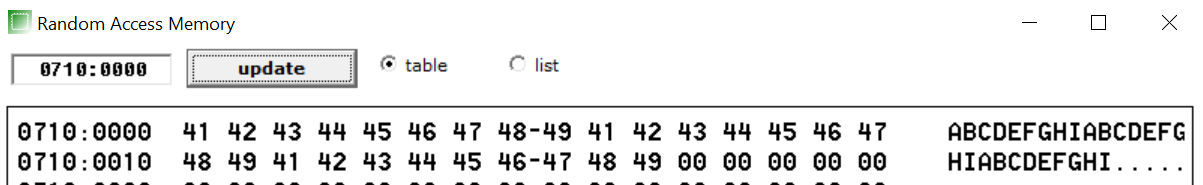


Рисунок 1 – Результат выполнения программы

Данные методы не эффективны, поскольку не используют цепочечные команды ассемблера для обработки массивов. Программа была переписана с использованием оператором MOVSB и RET (Листинг 2). Результат выполнения оказался тем же, а число операций существенно снизилось.

Затем была написана новая программа, осуществляющая заполнение и обработку классов в соответствии с текстом задания (Листинг 3). Поскольку при тестировании 130 элементов правильность результатов рассчитать трудно, для теста был выбран массив из 10 нечётных элементов начиная от 1, искомый – 9.

Изначально память под массивы была заполнена нулями (Рисунок 2).

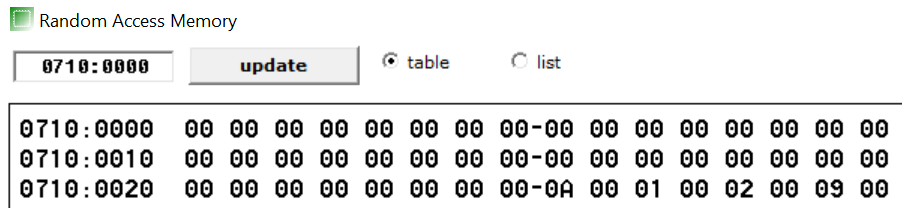


Рисунок 2 – Под массивы выделены ячейки 7100-7127, далее − переменные

Затем был заполнен первый массив вызовом процедуры fill\_array (Рисунок 3).

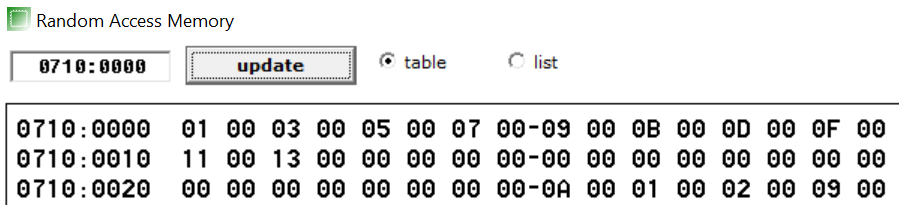


Рисунок 3 – Заполнение MAS1

Затем значения из MAS1 были скопированы в массив MAS2 процедурой copy\_array (Рисунок 4).

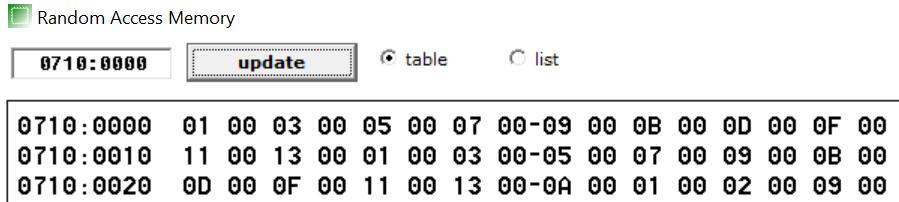


Рисунок 4 – Перенос значений из MAS1 в MAS2

Затем с помощью процедур search\_element и sum\_array были найдены индекс элемента 9 и сумма полученного массива. Элемент 9, как и ожидалось, расположен на 4 позиции (при нумерации с 0), а сумма составила 6416=10010.

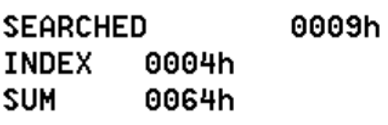


Рисунок 5 – Результаты вычислений (в переменных)

На основании данного теста был сделан вывод о правильной реализации строчных операций, условных переходов, циклов и процедур в написанной программе.

Код программы

Листинг 1 – Исходная программа из приложения

; multi-segment executable file template.

data segment

NAME1 DB 'ABCDEFGHI'

NAME2 DB 'JKLMNOPQR'

NAME3 DB 'STUVWXYZ\*'

ends

stack segment

dw 128 dup(0)

ends

code segment

start:

; set segment registers:

mov ax, data

mov ds, ax

mov es, ax

; add your code here

LEA SI, NAME1 ; Инициализация адресов

LEA DI, NAME2 ; NAME1 и NAME2

MOV CX, 09 ; Посчитать 9 символов

B20:

MOV AL, [SI] ; Взять из NAME1

MOV [DI], AL ; Поместить в NAME2

INC SI ; Следующий символ в NAME1

INC DI ; Следующая позиция в NAME2

DEC CX ; Уменьшить счетчик итераций

JNZ B20 ; Если счетчик > 0, то повторить

; RET ; Если счетчик = 0, то выйти

LEA SI, NAME2 ; Инициализация адресов

LEA DI, NAME3 ; NAME2 и NAME3

MOV CX, 09 ; Посчитать 9 символов

C20:

MOV AL, [SI] ; Взять из NAME2

MOV [DI], AL ; Поместить в NAME3

INC DI ; Следующий символ в NAME2

INC SI ; Следующая позиция в NAME3

LOOP C20 ; Уменьшить счетчик,

; если не равен нулю, то повторить

mov ax, 4c00h ; Выход из программы в операционную систему.

int 21h

ends

end start ; Установить точку входа и остановить ассемблер.

Листинг 2 – Оптимизированная программа из приложения

; multi–segment executable file template.

data segment

NAME1 DB 'ABCDEFGHI'

NAME2 DB 'JKLMNOPQR'

NAME3 DB 'STUVWXYZ\*'

ends

stack segment

dw 128 dup(0)

ends

code segment

start:

; set segment registers:

mov ax, data

mov ds, ax

mov es, ax

; add your code here

LEA SI, NAME1 ; Èíèöèàëèçàöèÿ àäðåñîâ

LEA DI, NAME2 ; NAME1 è NAME2

MOV CX, 09 ; Ïåðåñëàòü 9 ñèìâîëîâ

REP MOVSB

LEA SI, NAME2 ; Èíèöèàëèçàöèÿ àäðåñîâ

LEA DI, NAME3 ; NAME2 è NAME3

MOV CX, 09 ; ïåðåñëàòü 9 ñèìâîëîâ

REP MOVSB

mov ax, 4c00h ; exit to operating system.

int 21h

ends

end start ; set entry point and stop the assembler.

Листинг 3 – Программа работы с массивами

data segment

MAS1 DW 10 DUP(0) ; первый массив

MAS2 DW 10 DUP(0) ; второй массив

size DW 10 ; размер массивов

first DW 1 ; первый элемент MAS1

step DW 2 ; шаг заполнения MAS1

searched DW 9 ; искомый элемент

index DW ? ; индекс искомого элемента

sum DW ? ; сумма элементов массива

ends

stack segment

dw 128 dup(0)

ends

code segment

start:

; инициализация адресных регистров

mov ax, data

mov ds, ax

mov es, ax

; КОД ПРОГРАММЫ

CALL fill\_array

;CALL copy\_array

;CALL search\_element

CALL sum\_array

; ФИНАЛОЧКА

MOV AX, 4C00h

INT 21h

; Заполнение MAS1

fill\_array PROC

MOV CX, size ; счётчик длины массива

MOV DX, first ; элемент заполнения

LEA BX, MAS1 ; адрес элемента массива

fill\_loop:

MOV [BX], DX

ADD DX, step

ADD BX, 2

LOOP fill\_loop

RET

fill\_array ENDP

; Копирование MAS1 в MAS2

copy\_array PROC

LEA SI, MAS1 ; адрес-источник

LEA DI, MAS2 ; адреc-приёмник

MOV CX, size ; счётчик длины массива

CLD ; сброс флага D

REP MOVSW ; перенос массива

RET

copy\_array ENDP

; Поиск индекса

search\_element PROC

LEA DI, MAS1 ; адрес ячейки массива

LEA SI, [searched + 2] ; адрес искомого элемента

MOV index, -1 ; значение индекса

MOV CX, size ; счётчик длины массива

search\_loop:

SUB SI, 2

INC index

CMPSW

JZ search\_end ; элемент найден

LOOP search\_loop

MOV index, -1 ; элемент не найден

search\_end:

RET

search\_element ENDP

; Сумма MAS1

sum\_array PROC

MOV sum, 0 ; сумма элементов

LEA SI, MAS1 ; адрес ячейки массива

MOV CX, size ; счётчик длины массива

sum\_loop:

MOV AX, [SI]

ADD [sum], AX

ADD SI, 2

LOOP sum\_loop

RET

sum\_array ENDP

ends

end start

Вывод

В ходе работы были получены навыки работы со строками, условными переходами, циклами и процедурами на ассемблере 16-разрядного микропроцессора. Полученные навыки позволяют обрабатывать линейные массивы с помощью ассемблерного кода.