Лабораторная работа №4

«Исследование методов программирования обработки строковых данных»

4.1 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.

4.2 Постановка задачи

Вариант – 8

Требуется повторить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Изучить команды строковых операций. Изучить команды передачи управления в 16–разрядных процессорах и особенности оформления программ в ехе– и сом–форматах.

Составить программу, состоящую из следующих процедур обработки строк:

Исследовать программу пересылки массивов, приведенную в приложении. Оптимизировать данную программу за счет использования префикса повторения.

Заполнить 100+10i ячеек области памяти, начинающейся с адреса MAS1 рядом натуральных чисел.

Переслать массив слов из области памяти, начиная с адреса MAS1 в область с начальным адресом MAS2.

Найти в заданном массиве число, равное двум последним цифрам Вашей зачетной книжки и определить его индекс. Вычислить сумму элементов массива MAS1, созданном.

4.3 Ход работы

Были изучены команды строковых операций. Изучены команды передачи управления в 16–разрядных процессорах.

Была составлена программа согласно заданию. В листинге 1 представлен код составленной программы.

Листинг 1 – Программа обработки строк

ORG 100h

\_init PROC

CALL main

RET

\_init ENDP

main PROC

MOV AX, [main\_array\_size]

LEA BX, main\_array

MOV CX, -50

MOV DX, 10

CALL generate\_array

LEA AX, main\_array

LEA BX, main\_second\_array

MOV CX, [main\_array\_size]

CALL copy\_array

MOV AX, 30

LEA BX, main\_second\_array

MOV CX, [main\_array\_size]

CALL search\_element

MOV AX, 5

LEA BX, main\_second\_array

MOV CX, [main\_array\_size]

CALL search\_element

MOV AX, 0

LEA BX, main\_array

MOV CX, [main\_array\_size]

CALL array\_sum

RET

main ENDP

; Register arguments

; AX -> array size

; BX -> array address

; CX -> initialize value

; DX -> step

generate\_array PROC

ga\_loop:

MOV [BX], CX

ADD CX, DX

INC BX

INC BX

DEC AX

JNZ ga\_loop

RET

generate\_array ENDP

; Register arguments

; AX -> "from" array

; BX -> "to" array

; CX -> array size

copy\_array PROC

PUSH SI

PUSH DI

CLD

MOV SI, AX

MOV DI, BX

REP MOVSW

POP DI

POP SI

RET

copy\_array ENDP

; Register arguments

; AX -> value

; BX -> array address

; CX -> array size

; RETURN

; AX -> index. -1 if element is not found.

search\_element PROC

MOV \_se\_element, AX

MOV \_se\_array\_start, BX

se\_loop:

MOV AX, [\_se\_element]

XOR AX, [BX]

JZ se\_loop\_found

INC BX

INC BX

DEC CX

JNZ se\_loop

MOV AX, -1

JMP se\_loop\_end

se\_loop\_found:

MOV AX, BX

SUB AX, \_se\_array\_start

SHR AX, 1

se\_loop\_end:

RET

search\_element ENDP

; Register arguments

; AX -> None

; BX -> array address

; CX -> array size

; RETURN

; AX -> sum.

array\_sum PROC

PUSH DX

MOV AX, 0

as\_loop:

MOV DX, [BX]

ADD AX, DX

ADD BX, 2

DEC CX

JNZ as\_loop

as\_loop\_end:

POP DX

RET

array\_sum ENDP

main\_array DW 10 DUP(0)

main\_second\_array DW 10 DUP(0)

main\_array\_size DW 10

\_se\_element DW ?

\_se\_array\_start DW ?

END \_init

Программа была запущена в эмуляторе. Было проведено тестирование составленной программы. Для удобства тестовые данные и результаты тестирования представлены в таблице 1. Таблица 1 содержит название вызываемой процедуры, входные данные, ожидаемый и полученный результат выполнения процедуры.

Таблица 1 – Тестирование разработанных процедур

|  |  |  |  |
| --- | --- | --- | --- |
| Название процедуры | Входные данные | Ожидаемый результат выполнения процедуры | Результат выполнения процедуры |
| generate\_array | Адрес начала строки из слов, размер строки, начало -50, шаг 10. | Строка из элементов: {-50, -40, -30, -20, -10, 0, 10, 20, 30, 40}. | Строка из элементов: {-50, -40, -30, -20, -10, 0, 10, 20, 30, 40}. |
| copy\_array | Адрес источника, адрес приёмника и размер переданных строк. | Строка-приёмник содержит в точности те же элементы, что и строка-источник | Строка-приёмник содержит в точности те же элементы, что и строка-источник. |
| search\_element | Искомый элемент 30, адрес начала строки, размер строки | Регистр AX содержит значение 810. | Регистр AX содержит  значение 000816=810. |
| search\_element | Искомый элемент 5, адрес начала строки и её размер. | Регистр AX содержит -110. | Регистр AX содержит FFFF16=-110. |
| array\_sum | Начальный адрес строки и её размер. | Регистр AX содержит  −5010. | Регистр AX содержит FFA616=-5010. |

Все составленные тесты были пройдены, значит программа и процедуры в частности работают корректно. Результаты тестирования полностью соответствуют ожиданиям.

Выводы

При выполнении данной работы были получены навыки работы со строками, использования условных переходов и составления циклов и процедур. Также были повторно закреплены знания о способах адресации значений. Изучены основные команды языка ассемблера для обработки строковых данных и команды передачи управления, исследованы их воздействие на процесс ассемблирования и формирования листинга программы. Исследованы особенности функционирования блоков 16–разрядного микропроцессора при выполнении команд обработки строк и передачи управления. Приобретены практические навыки программирования на языке ассемблера МП 8086 задач обработки линейных массивов.