МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

кафедра «Информационные системы»

Отчёт

по лабораторной работе №4

по дисциплине «Технические средства информационных систем»

Выполнил:

ст.гр. ИС/б-20-2-о

Филозоп А.Н.

Принял:

Минкин С.И.

Севастополь

2022 г.

4 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

Составит программу, состоящую из следующих процедур обработки строк:

– 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.

4.3. Ход выполнения работы

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
  MOV AX, 0
  as_loop:
    MOV DX, [BX]
    ADD AX, DX
        ADD BX, 2
    DEC CX
    JNZ as_loop
  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. 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 8_{10} .	Регистр AX содержит значение $0008_{16}=8_{10}$.
search_element	Искомый элемент 5, адрес начала строки и ё размер.	Регистр AX содержит -1_{10} .	Регистр AX содержит $FFFF_{16}=-1_{10}$.
array_sum	Начальный адрес строки и её размер.	Регистр AX содержит -50_{10} .	Регистр AX содержит $FFA6_{16}=-50_{10}$.

Все составленные тесты были пройдены, значит программа и процедуры в частности работают корректно.

Вывод

При выполнении данной лабораторной работы были получены навыки работы со строками, использования условных переходов и составления циклов и процедур. Также были повторно закреплены знания о способах адресации значений.

Приложение А

Ответы на контрольные вопросы

1. Различие между командой и директивой состоит в том, что команды порождают одну машинную команду, а директивы содержат управляющую информацию для ассемблера.

2. Для определения данных используются директивы DB (определение переменной размером 1 байт), DW (определение переменной размером 1 машинное слово, или 2 байта), DD (определение переменной размером 2 слова, или 4 байта). Для инициализации нескольких ячеек используется DUP.

3. Для оформления процедур используются директивы PROC (начало процедуры) и ENDP (окончание процедуры). После PROC допустимо указание NEAR (код процедуры находится в том же сегменте, что и точки вызова процедуры) или FAR (код процедуры находится в другом сегменте).

4. Используются следующие типы сегментов: DATA (сегмент с данными), CODE (сегмент с кодом), STACK (сегмент стека).

5. Используются следующие типы выравнивания: PARA (адрес сегмента XXX0, где X – любое шестнадцатеричное число), PAGE (XX00), WORD (XXXE) и BYTE (XXXX).

6. Директива ORG позволяет переопределить счётчик команд ассемблера. Используется при написании программ EXE, где требуется 100_{16} отступ на от начала исполняемого файла.

7. Директива END для процедур сообщает о завершении кода процедуры; для сегмента — приостановить инкремент счётчика адресов относительно сегмента.

8. В языке ассемблера строкой называется набор однотипных элементов (байтов, слов). Для обработки строк используются команды: MOVS (переслать элемент из SI в DI), LODS (загрузить элемент из SI в AL/AX), STOS (записать содержимое AL/AX в DI), CMPS (сравнить содержимое SI и DI), SCAS (сравнить AL/AX с содержимым SI).

9. Отличия SI от DI состоят в следующем: в командах обработки строк SI указывает на источник данных, а DI – на приёмник данных; SI обычно связан с сегментом данных DS, а DI – с дополнительным сегментом ES.

10. Для задания направления перемещения по строкам применяется флаг D, который устанавливается командой STD и сбрасывается командой CLD. При нулевом значении флага при каждом повторении происходит увеличение адресов источника и приемника, а при единичном – уменьшение.

11. Необходимость пересылки байта или слова ассемблер определяет при помощи постфикса команды. Таким образом, если указан постфикс W, то будет переслано слово, а если B – байт.

12. При использовании префикса REP количество повторений можно указать при помощи занесения в регистр CX количества необходимых повторений.

13. Существуют следующие модификации префикса REP: REPZ/REPE позволяют повторить операцию, пока флаг ZF не будет установлен в нуль или CX не будет равен 0; REPNZ/ REPNE позволяют повторить операцию, пока флаг ZF не будет установлен в единицу или CX не будет равен 1.

14. Особенность выполнения команд передачи управления в пределах одного сегмента и между сегментами состоит том, что в первом случае модифицируется только регистр IP и адрес перехода представлен одним словом, а во втором случае модифицируются регистры IP и CS и адрес перехода состоит из двух слов.

15. Существуют следующие команды условного перехода: JZ, JNZ, JG, JS и другие. Условие передачи управления определяется на основе состояния соответствующих флагов PSW.

16. Для управления циклами используются команды LOOP (уменьшение CX и переход по адресу при $CX \neq 0$), LOOPZ/LOOPE (то же, что и LOOP, но переходит при установке $ZF = 1$) и LOOPNZ/LOOPNE (то же, что и LOOP, но переходит при установке $ZF = 0$).

17. Суть защищённого режима работы процессора состоит в защите кода текущей программы или пространства других программ от непреднамеренного чте-

ния/записи. Защита осуществляется следующим образом: в сегментный регистр заносится не реальное значение сегмента, а селектор.

18. Дескриптор сегмента — структура данных, характеризующая размещение и длину используемого сегмента. Располагается в памяти ПЭВМ. Состоит из 4-х слов. Первое зарезервировано, третье содержит младшие разряды базы, четвёртое — размер сегмента. Второе слова содержит следующую информацию: биты «присутствие в основной памяти» P, «обращение к сегменту» A и «системный сегмент» S; уровень привилегий дескриптора DPL; тип сегмента; старшие разряды базы.

19. Виртуальная память – это способ организации основной памяти большой емкости с помощью внешней памяти. Она позволяет при составлении программы распоряжаться всем пространством адресов, называемых виртуальными. Для поддержки виртуальной памяти используется дескриптор сегмента следующим образом: бит P указывает при $P=1$, что данная страница находится в памяти и УУП осуществляет преобразование виртуального адреса в физический, иначе УУП передаёт ЦП сигнал отсутствия сегмента; в «Тип» указывается, разрешена ли запись в сегмент, тип сегмента.

20. Процессор второго поколения состоит из четырех блоков: адресного AU, шинного BU, исполнительного EU и командного IU, причем все блоки могут работать параллельно. Шинный блок осуществляет считывание памяти и портов ввода/вывода. Адресный блок вычисляет все адреса и формирует физические адреса. За счёт параллельной работы блоков возможно повышение производительности в 2-3 раза. Также процессор может работать в реальном и защищённом режимах.

21. …

22. Сегмент состояния задачи состоит из 22-х слов. В нём хранятся состояния всех регистров, начальные значения указателей стеков уровней привилегий 0, 1 и 2, селектор локальной таблицы дескрипторов данной задачи. Совокупность ячеек ПС, ФР и РОН называют блоком управления задачей.