МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Организация ЭВМ и систем»
Тема: Представление и обработка целых чисел. Организация ветвящихся процессов
Вариант 8

Студент гр. 0382 _____ Кондратов Ю.А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

Обучение работе с ветвящимися процессами путём разработки программы, вычисляющей значение некоторых функций по заданным целочисленным значениям параметров, на языке Ассемблера.

Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

a) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

b) значения результирующей функции res = f3(i1,i2,k),

где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1,n2,n3), приведенным в табл.4.

Значения a, b, i, k являются исходными данными, которые должны выбираться студентом самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть всевозможные комбинации параметров a, b и k, позволяющие проверить различные маршруты выполнения программы, а также различные знаки параметров a и b.

Выполнение работы.

Функции для варианта 8 представлены на рисунке 1.

$$f2 = \begin{cases} -(4*i+3), & a>b \\ 6*i-10, & a\le b \end{cases} \quad f3 = \begin{cases} 7-4*i, & a>b \\ 8-6*i, & a\le b \end{cases} \quad f8 = \begin{cases} |i1|-|i2|, & k<0 \\ \max(4,|i2|-3), & k\ge 0 \end{cases}$$

Рисунок 1 –Функции для варианта 8

В процессе написания программы можно выделить следующие этапы:

1. Работа с сегментами.

Было создано три сегмента: DATA – сегмент данных, CODE – сегмент кода, AStack – сегмент стека. С помощью директивы ASSUME метки сегментов были записаны в соответствующие регистры. Также в сегменте кода была создана процедура Main и написаны инструкции необходимые для успешного завершения программы после возврата из функции. В сегменте данных

объявлены переменные var_a, var_b, var_i, var_k, var_i1, var_i2, var_res, хранящие значения соответствующих переменных из задания.

2. Написание функций f1, f2, f3.

Функции f1, f2, f3 написаны без использования PROC, поэтому в них используются условные и безусловные переходы. Используемые переходы представлены в таблице 1.

Таблица 1 – Используемые переходы

Команда	Использование в программе
JG	Условный переход, выполняется если SF = OF и ZF = 0. Используется в функциях f1 и f2 для обхода инструкций, которые выполняются при a <= b.
JMP	Безусловный переход, используется в функциях f1 и f2 при a > b для обхода инструкций, которые выполняются при a > b и перехода к записи ответа в var_i1 или var_i2, в функции f3 для обхода инструкций, выполняющихся при k < 0 и перехода к записи ответа в var_res.
JGE	Условный переход, выполняется при SF = OF. Используется в функции f3, при вычислении |i1| и |i2| для обхода инструкции по смене знака при i1 >= 0 или i2 >= 0.
JL	Условный переход, выполняется при SF != OF. Используется в функции f3 при вычислении max(4, |i2| - 3) для обхода инструкции по записи значения 4 в регистр, хранящий ответ, и для обхода инструкций, выполняющихся при k <= 0 и перехода к записи результата в var_res.

Файл диагностических сообщений, созданный при трансляции программы представлен в приложении Б. Исходный код программы см. в приложении А.

Тестирование.

Для проверки работоспособности программы разработаны тесты, представленные в таблице 2.

Таблица 2 – Тесты для проверки работоспособности программы.

Номер теста	var_a	var_b	var_i	var_k
1.	3	2	1	-1
2.	2	3	2	1
3.	2	3	3	1

Данные, используемые в тестах, записывались в соответствующие ячейки памяти в процессе отладки программы через отладчик AFDPRO.

Результаты тестирование представлены в таблице 3.

Таблица 3 – Результаты тестирования.

Номер теста из таблицы 2	var_i1	var_i2	var_res	Вердикт
1.	FFF9 (-7)	0003	0004	Тест пройден
2.	0002	FFFC (-4)	0004	Тест пройден
3.	0008	FFF6 (-10)	0007	Тест пройден

Выводы.

В ходе работы были изучены способы ветвления программы, условные и безусловные переходы, также была написана программа, вычисляющая значение функции по заданным целочисленным параметрам.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

main.asm:

```
AStack SEGMENT STACK
    DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
    var_a DW 0
    var_b DW 0
    var_i DW 0
    var_k DW 0
    var_i1 DW 0
    var_i2 DW 0
    var_res DW 0
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR
    push ds
    sub ax, ax
    push ax
    mov ax, DATA
    mov ds, ax
f1:
    mov ax, var_i
    shl ax, 1 ;ax = 4i
    shl ax, 1
    mov bx, var_a
    cmp bx, var_b
    jg f1_1
f1_2:
    mov bx, ax
    shr bx, 1 ;bx = 2i
    add ax, bx ;ax = 6i
    sub ax, 0Ah ;ax = 6i - 10
    jmp f1_end
f1_1:
    neg ax
    sub ax, 3h ;ax = -4i - 3
f1_end:
    mov var_i1, ax ; i1 = f1(i)
f2:
    mov ax, var_i
    shl ax, 1
    shl ax, 1 ;ax = 4i
    mov bx, var_a
    cmp bx, var_b
    jg f2_1
f2_2:
    mov bx, ax
    shr bx, 1 ;bx = 2i
    add ax, bx ;ax = 6i
```

```
    neg ax ;ax = -6i
    add ax, 8h
    jmp f2_end
f2_1:
    neg ax
    add ax, 7h
f2_end:
    mov var_i2, ax ;i2 = f2(i)
f3:
    mov bx, var_i2
    cmp bx, 0h
    jge abs_i2
    neg bx
abs_i2: ; bx = |i2|
    mov ax, var_k
    cmp ax, 0h
    jl f3_1
f3_2:
    sub bx, 3h ; bx = |i2| - 3
    cmp bx, 4h
    jge max
    mov bx, 4h
max:
    mov ax, bx
    jmp f3_end
f3_1:
    mov ax, var_i1
    cmp ax, 0h
    jge abs_i1
    neg ax
abs_i1: ; ax = |i1|
    sub ax, bx
f3_end:
    mov var_res, ax
    ret
Main ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ

```
Microsoft (R) Macro Assembler Version 5.10                  10/18/21
18:49:3
                                                             Page
1-1


0000                    AStack SEGMENT STACK
0000  0020[                      DW 32 DUP(?)
       ????
                    ]

0040                    AStack ENDS

0000                    DATA SEGMENT
0000  0000                       var_a DW 0
0002  0000                       var_b DW 0
0004  0000                       var_i DW 0
0006  0000                       var_k DW 0
0008  0000                       var_i1 DW 0
000A  0000                       var_i2 DW 0
000C  0000                       var_res DW 0
000E                    DATA ENDS

0000                    CODE SEGMENT
                            ASSUME CS:CODE, DS:DATA, SS:AStack

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
0009                    f1:
0009  A1 0004 R             mov ax, var_i
000C  D1 E0                      shl ax, 1 ;ax = 4i
000E  D1 E0                      shl ax, 1
0010  8B 1E 0000 R               mov bx, var_a
0014  3B 1E 0002 R               cmp bx, var_b
0018  7F 0C                      jg f1_1
001A                    f1_2:
001A  8B D8                      mov bx, ax
001C  D1 EB                      shr bx, 1 ;bx = 2i
001E  03 C3                      add ax, bx ;ax = 6i
0020  2D 000A                    sub ax, 0Ah ;ax = 6i - 10
0023  EB 06 90                   jmp f1_end
0026                    f1_1:
0026  F7 D8                      neg ax
0028  2D 0003                    sub ax, 3h ;ax = -4i - 3
002B                    f1_end:
002B  A3 0008 R             mov var_i1, ax ; i1 = f1(i)
002E                    f2:
002E  A1 0004 R             mov ax, var_i
0031  D1 E0                      shl ax, 1
0033  D1 E0                      shl ax, 1 ;ax = 4i
```

```
0035  8B 1E 0000 R            mov bx, var_a
0039  3B 1E 0002 R            cmp bx, var_b
003D  7F 0E                   jg f2_1
003F                  f2_2:
003F  8B D8                   mov bx, ax
0041  D1 EB                   shr bx, 1 ;bx = 2i
```



```
0043  03 C3                   add ax, bx ;ax = 6i
0045  F7 D8                   neg ax ;ax = -6i
0047  05 0008                 add ax, 8h
004A  EB 06 90                jmp f2_end
004D                  f2_1:
004D  F7 D8                   neg ax
004F  05 0007                 add ax, 7h
0052                  f2_end:
0052  A3 000A R           mov var_i2, ax ;i2 = f2(i)
0055                  f3:
0055  8B 1E 000A R            mov bx, var_i2
0059  83 FB 00                cmp bx, 0h
005C  7D 02                   jge abs_i2
005E  F7 DB                   neg bx
0060                  abs_i2: ; bx = |i2|
0060  A1 0006 R           mov ax, var_k
0063  3D 0000                 cmp ax, 0h
0066  7C 10                   jl f3_1
0068                  f3_2:
0068  83 EB 03                sub bx, 3h ; bx = |i2| - 3
006B  83 FB 04                cmp bx, 4h
006E  7D 03                   jge max
0070  BB 0004                 mov bx, 4h
0073                  max:
0073  8B C3                   mov ax, bx
0075  EB 0D 90                jmp f3_end
0078                  f3_1:
0078  A1 0008 R           mov ax, var_i1
007B  3D 0000                 cmp ax, 0h
007E  7D 02                   jge abs_i1
0080  F7 D8                   neg ax
0082                  abs_i1: ; ax = |i1|
0082  2B C3                   sub ax, bx
0084                  f3_end:
0084  A3 000C R           mov var_res, ax
0087  CB                      ret
0088                  Main ENDP
0088                  CODE ENDS
                      END Main
```

```
      Segments and Groups:

                N a m e         Length     Align     Combine Class

      ASTACK . . . . . .      0040  PARA  STACK
      CODE . . . . . .        0088  PARA  NONE
      DATA . . . . . .        000E  PARA  NONE

      Symbols:

                N a m e         Type  Value     Attr

      ABS_I1 . . . . . .      L NEAR    0082  CODE
      ABS_I2 . . . . . .      L NEAR    0060  CODE

      F1 . . . . . .          L NEAR    0009  CODE
      F1_1 . . . . . .        L NEAR    0026  CODE
      F1_2 . . . . . .        L NEAR    001A  CODE
      F1_END . . . . . .      L NEAR    002B  CODE
      F2 . . . . . .          L NEAR    002E  CODE
      F2_1 . . . . . .        L NEAR    004D  CODE
      F2_2 . . . . . .        L NEAR    003F  CODE
      F2_END . . . . . .      L NEAR    0052  CODE
      F3 . . . . . .          L NEAR    0055  CODE
      F3_1 . . . . . .        L NEAR    0078  CODE
      F3_2 . . . . . .        L NEAR    0068  CODE
      F3_END . . . . . .      L NEAR    0084  CODE

      MAIN . . . . . .        F PROC    0000  CODE  Length  =
0088
      MAX  . . . . . .        L NEAR    0073  CODE

      VAR_A  . . . . . .      L WORD    0000  DATA
      VAR_B  . . . . . .      L WORD    0002  DATA
      VAR_I  . . . . . .      L WORD    0004  DATA
      VAR_I1 . . . . . .      L WORD    0008  DATA
      VAR_I2 . . . . . .      L WORD    000A  DATA
      VAR_K  . . . . . .      L WORD    0006  DATA
      VAR_RES  . . . . . .    L WORD    000C  DATA

      @CPU . . . . . .        TEXT  0101h
      @FILENAME  . . . . . .  TEXT  MAIN
      @VERSION . . . . . .    TEXT  510

Microsoft (R) Macro Assembler Version 5.10                    10/18/21
18:49:3


Symbols-2
          90 Source  Lines
          90 Total   Lines
          31 Symbols

       48054 + 459206 Bytes symbol space free

           0 Warning Errors
           0 Severe  Errors
```