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

Студент гр. 0382 _____ Самулевич В.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить базовые операции над целыми числами, а также основные команды ветвления в языке Ассемблера.

Задание.

Вариант 23

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, соответствующие варианту 23, представлены на картинке:

$$f5 = \begin{cases} 20 - 4*i, & \text{при } a>b \\ -(6*I - 6), & \text{при } a<=b \end{cases} \quad f6 = \begin{cases} 2*(i+1) - 4, & \text{при } a>b \\ 5 - 3*(i+1), & \text{при } a<=b \end{cases} \quad f4 = \begin{cases} \min(|i1 - i2|, 2), & \text{при } k<0 \\ \max(-6, -i2), & \text{при } k>=0 \end{cases}$$

Рисунок 1.Функции для варианта 23

Для хранения входных данных, результата, а также временных значений, в сегменте DATA с помощью директивы DW были созданы переменные a, b, k, i, i1, i2, res ,ix2 ,ix3.

Для обеспечения ветвления программы использовались команды jmp, jg(переход если больше со знаком) , jl(переход если меньше со знаком) и

js(переход, если отрицательно).Для уменьшения объема кода были реализованы следующие модификации:

- Значения 2i и 3i вычисляются один раз в начале программы, после чего загружаются в память (в ячейки ix2 и ix3 соответственно). В процессе выполнения, при необходимости, эти значения загружаются обратно в AX.

- Переменные a и b сравниваются один раз в начале программы, после чего происходит переход на соответствующие результату выражения в f1 и f2.

Файл диагностических сообщений, созданный при трансляции программы, представлен в приложении B.Исходный код программы см. в приложении A.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	a=10 b= -15 k= -4 i=7	i1=FFF8 (-8) i2= 000C(12) res= 0002(2)	Ответ корректен
2.	a= 7 b= 4 k= 5 i= -5	i1=0028(40) i2=FFF4(-12) res=000C(12)	Ответ корректен
3.	a= -20 b= -14 k= -3 i= 9	i1=FFC4(-60) i2=FFE7(-25) res=2	Ответ корректен
4.	a= 5 b= 5 k=4 i=-4	i1=0012(18) i2=000E(14) res=FFFA(-6)	Ответ корректен

Выводы.

Были изучены основные арифметические команды и команды ветвления в языке ассемблера, а также написана программа, использующая их.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

```
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

DATA SEGMENT

a DW 5
b DW 5
i DW -4
k DW 4
i1 DW 0
i2 DW 0
res DW 0
ix2  DW 0
ix3  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

    mov AX,i
    shl AX,1
    mov ix2,AX
    add AX,i
    mov ix3,AX
    mov CX,a
    cmp CX,b
    jg first_path ;a > b

second_path:

f1_2: ;i1=-6-6i

    shl AX,1
    neg AX
    sub AX,6
    mov i1,AX


f2_2: ;i2=-3i+2

    mov AX,ix3
```

```
    neg AX
    add AX,2
    mov i2,AX
    jmp f3


first_path:

f1_1: ;i1=20-4i

    mov AX,ix2
    shl AX,1
    neg AX
    add AX,20
    mov i1,AX

f2_1: ;i2=2i-2

    mov AX,ix2
    sub AX,2
    mov i2,AX


f3:
    neg AX
    mov CX,k
    cmp CX,0
    jl f3_1 ; K<0

f3_2: ;res=max(-6,-i2)

    mov DX,-6
    cmp AX,DX
    jg f3_res_1 ; -i2 > -6

f3_res_2:
    mov res,DX
    jmp stop

f3_res_1:
    mov res,AX
    jmp stop

f3_1: ;res=min (| i1-i2|,2)

    add AX,i1
module:
    neg AX
    js module
    mov DX,2
    cmp AX,DX
    jl f3_res_1
    jmp f3_res_2

stop:

    ret
Main ENDP
```

```
CODE ENDS
    END Main
```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ

```
Microsoft (R) Macro Assembler Version 5.10                  11/3/21 19:09:02
                                                             Page     1-1


 0000                   AStack SEGMENT STACK
 0000  000C[                       DW 12 DUP(?)
         ????
                    ]

 0018                   AStack ENDS

 0000                   DATA SEGMENT

 0000  0005            a DW 5
 0002  0005            b DW 5
 0004  FFFC            i DW -4
 0006  0004            k DW 4
 0008  0000            i1 DW 0
 000A  0000            i2 DW 0
 000C  0000            res DW 0
 000E  0000            ix2  DW 0
 0010  0000            ix3  DW 0

 0012                   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  A1 0004 R           mov AX,i
 000C  D1 E0                     shl AX,1
 000E  A3 000E R           mov ix2,AX
 0011  03 06 0004 R              add AX,i
 0015  A3 0010 R           mov ix3,AX
 0018  8B 0E 0000 R              mov CX,a
 001C  3B 0E 0002 R              cmp CX,b
 0020  7F 18                     jg first_path ;a > b

 0022                   second_path:

 0022                   f1_2: ;i1=-6-6i

 0022  D1 E0                     shl AX,1
 0024  F7 D8                     neg AX
 0026  2D 0006                   sub AX,6
 0029  A3 0008 R           mov i1,AX
```

```
002C                    f2_2: ;i2=-3i+2

002C  A1 0010 R              mov AX,ix3
002F  F7 D8                       neg AX
```



```
0031  05 0002                     add AX,2
0034  A3 000A R              mov i2,AX
0037  EB 17 90                    jmp f3


003A                    first_path:

003A                    f1_1: ;i1=20-4i

003A  A1 000E R              mov AX,ix2
003D  D1 E0                       shl AX,1
003F  F7 D8                       neg AX
0041  05 0014                     add AX,20
0044  A3 0008 R              mov i1,AX

0047                    f2_1: ;i2=2i-2

0047  A1 000E R              mov AX,ix2
004A  2D 0002                     sub AX,2
004D  A3 000A R              mov i2,AX


0050                    f3:
0050  F7 D8                       neg AX
0052  8B 0E 0006 R               mov CX,k
0056  83 F9 00                    cmp CX,0
0059  7C 14                       jl f3_1 ; K<0

005B                    f3_2: ;res=max(-6,-i2)

005B  BA FFFA                     mov DX,-6
005E  3B C2                       cmp AX,DX
0060  7F 07                       jg f3_res_1 ; -i2 > -6

0062                    f3_res_2:
0062  89 16 000C R                mov res,DX
0066  EB 18 90                    jmp stop

0069                    f3_res_1:
0069  A3 000C R              mov res,AX
006C  EB 12 90                    jmp stop

006F                    f3_1: ;res=min (| i1-i2|,2)

006F  03 06 0008 R                add AX,i1
0073                    module:
0073  F7 D8                       neg AX
0075  78 FC                       js module
0077  BA 0002                     mov DX,2
```

```
 007A  3B C2                    cmp AX,DX
 007C  7C EB                    jl f3_res_1
 007E  EB E2                    jmp f3_res_2

 0080                  stop:

Microsoft (R) Macro Assembler Version 5.10                  11/3/21 19:09:02
                                                             Page     1-3


 0080  CB                       ret
 0081                  Main ENDP
 0081                  CODE ENDS
                           END Main


Microsoft (R) Macro Assembler Version 5.10                    11/3/21
19:09:02
                                                              Symbols-1


Segments and Groups:

                N a m e         Length     Align    Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                0081  PARA  NONE
DATA . . . . . .                0012  PARA  NONE

Symbols:

                N a m e         Type   Value      Attr

A  . . . . . .                  L WORD     0000  DATA

B  . . . . . .                  L WORD     0002  DATA

F1_1 . . . . . .                L NEAR     003A  CODE
F1_2 . . . . . .                L NEAR     0022  CODE
F2_1 . . . . . .                L NEAR     0047  CODE
F2_2 . . . . . .                L NEAR     002C  CODE
F3 . . . . . .                  L NEAR     0050  CODE
F3_1 . . . . . .                L NEAR     006F  CODE
F3_2 . . . . . .                L NEAR     005B  CODE
F3_RES_1 . . . . . .            L NEAR     0069  CODE
F3_RES_2 . . . . . .            L NEAR     0062  CODE
FIRST_PATH . . . . . .          L NEAR     003A  CODE

I  . . . . . .                  L WORD     0004  DATA
I1 . . . . . .                  L WORD     0008  DATA
I2 . . . . . .                  L WORD     000A  DATA
IX2  . . . . . .                L WORD     000E  DATA
IX3  . . . . . .                L WORD     0010  DATA

K  . . . . . .                  L WORD     0006  DATA

MAIN . . . . . .                F PROC     0000  CODE  Length = 0081
```

```
MODULE . . . . . .      L NEAR     0073  CODE

RES  . . . . . .        L WORD     000C  DATA

SECOND_PATH  . . . . . .      L NEAR     0022  CODE
STOP . . . . . .        L NEAR     0080  CODE

@CPU . . . . . .        TEXT  0101h
@FILENAME  . . . . . .        TEXT  lab3
@VERSION . . . . . .        TEXT  510


Microsoft (R) Macro Assembler Version 5.10                  11/3/21
19:09:02
                                                             Symbols-2


    110 Source  Lines
    110 Total   Lines
     31 Symbols

  47976 + 459284 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```