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 Ассемблера

Студент гр. 1383 _____ Кошкин Е.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Получение организации ветвящихся процессов 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.

Вариант 12 (f2, f7, f4).

Упрощение выражений:

6i - 10 = 2* (2i + i) - 10

10 - 3i = - (2i + i - 10)

Так как 2i используется везде, заранее вычисляется. Далее сравниваются a и b и взависимости от результата выполняется переход по метке. В итоге в cx результат f2, в dx результат (-f7), так как в f4 используется -f7. Далее выполняются сравнение к и аналогичные операции.

Таблица 1. Тестирование

№	a	b	i	k	f2	-f7	f4
1	2	2	3	0	8	-1	-1
2	-1	0	-1	-2	-16	-13	2
3	-3	-1	2	-1	2	-4	2
4	-6	-2	1	1	-4	-7	-6
5	3	2	1	0	-7	-1	-1
6	4	0	-1	-6	1	-9	2
7	6	3	2	3	-11	3	3

Выводы.

Получены навыки организации ветвящихся процессов на языке Ассемблера.

Разработана программа, вычисляющая значения функций при определенных параметрах.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.asm:

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

DATA SEGMENT
	a DW 0
	b DW 0
	i DW 0
	k DW 0
	res DW 0
DATA ENDS

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

Main PROC FAR
	push ds ; initialization
	sub ax, ax
	push ax
	mov ax, DATA
	mov ds, ax

	mov cx, i ;cx = i
	shl cx, 1 ;cx = 2i
	mov ax, a
	cmp ax, b
	jle lessb ;if a <= b

	shl cx, 1 ;cx = 4i
	mov dx, cx ;dx = 4i
	sub dx, 5 ;dx = 4i - 5
	add cx, 3 ;cx = 4i + 3
	neg cx ;cx = -(4i + 3)
	jmp step1

lessb:
	add cx, i ;cx = 3i
```

```
        mov dx, cx ;dx = cx
        shl cx, 1 ;cx = 6i
        sub cx, 10 ;cx = 6i - 10
        sub dx, 10 ;dx = 3i - 10

step1: ;cx = f1, dx = -f2
        mov ax, k
        cmp ax, 0
        jl less0 ;if k < 0
        mov ax, -6
        cmp dx, ax
        jg first ;if dx > ax
        jmp second

less0:
        add dx, cx
        cmp dx, 0
        jge module ;if dx >= 0
        neg dx

module:
        mov ax, 2
        cmp dx, ax
        jg second ;if dx > ax

first:
        mov res, dx
        ret

second:
        mov res, ax
        ret
Main ENDP
CODE ENDS
        END Main
```

main.lst:

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

0018                    AStack ENDS

0000                    DATA SEGMENT
0000  0000                 a DW 0
0002  0000                 b DW 0
0004  0000                 i DW 0
0006  0000                 k DW 0
0008  0000                 res DW 0
000A                    DATA ENDS

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

0000                    Main PROC FAR
0000  1E                  push ds ; initialization
0001  2B C0                    sub ax, ax
0003  50                   push ax
0004  B8 ---- R            mov ax, DATA
0007  8E D8                    mov ds, ax
```

```
0009  8B 0E 0004 R              mov cx, i   ;cx = i
000D  D1 E1                     shl cx, 1   ;cx = 2i
000F  A1 0000 R           mov ax, a
0012  3B 06 0002 R              cmp ax, b
0016  7E 0F                     jle lessb ;if a <= b

0018  D1 E1                     shl cx, 1   ;cx = 4i
001A  8B D1                     mov dx, cx  ;dx = 4i
001C  83 EA 05                  sub dx, 5   ;dx = 4i - 5
001F  83 C1 03                  add cx, 3   ;cx = 4i + 3
0022  F7 D9                     neg cx      ;cx = -(4i + 3)
0024  EB 0F 90                  jmp step1

0027                      lessb:
0027  03 0E 0004 R              add cx, i   ;cx = 3i
002B  8B D1                     mov dx, cx  ;dx = cx
002D  D1 E1                     shl cx, 1   ;cx = 6i
002F  83 E9 0A                  sub cx, 10  ;cx = 6i - 10
0032  83 EA 0A                  sub dx, 10  ;dx = 3i - 10

0035                      step1: ;cx = f1, dx = -f2
0035  A1 0006 R           mov ax, k
0038  3D 0000                   cmp ax, 0
003B  7C 0A                     jl less0 ;if k < 0
003D  B8 FFFA                   mov ax, -6
0040  3B D0                     cmp dx, ax
0042  7F 13                     jg first ;if dx > ax
0044  EB 16 90                  jmp second
```

```
0047                    less0:
0047  03 D1                     add dx, cx
0049  83 FA 00                  cmp dx, 0
004C  7D 02                     jge module ;if dx >= 0
004E  F7 DA                     neg dx

0050                    module:
0050  B8 0002                   mov ax, 2
0053  3B D0                     cmp dx, ax
0055  7F 05                     jg second ;if dx > ax

0057                    first:
0057  89 16 0008 R              mov res, dx
005B  CB                    ret

005C                    second:
005C  A3 0008 R             mov res, ax
005F  CB                    ret
0060                    Main ENDP
0060                    CODE ENDS
                            END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0060	PARA	NONE	
DATA	000A	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
FIRST	L NEAR	0057	CODE
I	L WORD	0004	DATA
K	L WORD	0006	DATA
LESS0	L NEAR	0047	CODE
LESSB	L NEAR	0027	CODE

```
MAIN . . . . . .          F PROC      0000  CODE        Length      =
0060

MODULE . . . . . .        L NEAR      0050  CODE

RES  . . . . . .    L WORD     0008  DATA

SECOND . . . . . .        L NEAR      005C  CODE
STEP1  . . . . . .  L NEAR      0035  CODE

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


    72 Source  Lines
    72 Total   Lines
    20 Symbols

  47964 + 461343 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```