

**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. 1381

Мамин Р.А.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2022

## 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:

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

б) 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.

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

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 для обхода инструкций, которые выполняются при $a \leq b$ .
JMP	Безусловный переход, используется 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 $i1 \geq 0$ или $i2 \geq 0$ .
JL	Условный переход, выполняется при $SF \neq OF$ . 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 $k \leq 0$ 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 тестирования 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 2
    var_b DW 3
    var_i DW 2
    var_k DW 1
    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 = 2i
    shl ax, 1 ;ax = 4i
    mov bx, var_a
    cmp bx, var_b
    jg f1_1

f1_2:
    mov bx, ax ;bx = 4i
    shr bx, 1 ;bx = 2i
    add ax, bx ;ax = 6i
    mov cx, ax ;cx = 6i
    sub ax, 0Ah ;ax = 6i - 10
    neg cx ;cx = -6i
    add cx, 8h ;cx = -6i + 8
    jmp f1_end

f1_1:
    neg ax
    sub ax, 3h ;ax = -4i - 3
    mov cx, ax ;cx = -4i -3
    add cx, 0Ah ;cx = -4i -3 + 10 = -4i + 7

f1_end:
    mov var_i1, ax ; i1 = f1(i)
    mov var_i2, cx ; 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

```

# ЛИСТИНГ

```
Microsoft (R) Macro Assembler Version 5.10
10:46:3
```

10/21/22

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 = 2i
000E    D1 E0                        shl ax, 1 ;ax = 4i
0010    8B 1E 0000 R                mov bx, var_a
0014    3B 1E 0002 R                cmp bx, var_b
0018    7F 13                        jg f1_1
001A                                f1_2:
001A    8B D8                        mov bx, ax ;bx = 4i
001C    D1 EB                        shr bx, 1 ;bx = 2i
001E    03 C3                        add ax, bx ;ax = 6i
0020    8B C8                        mov cx, ax ;cx = 6i
0022    2D 000A                      sub ax, 0Ah ;ax = 6i - 10
0025    F7 D9                        neg cx ;cx = -6i
0027    83 C1 08                      add cx, 8h ;cx = -6i + 8
002A    EB 0B 90                      jmp f1_end
002D                                f1_1:
002D    F7 D8                        neg ax
002F    2D 0003                      sub ax, 3h ;ax = -4i - 3
0032    8B C8                        mov cx, ax ;cx = -4i -3

```

[illegible]

```

0034 83 C1 0A          add cx, 0Ah ;cx = -4i -3 + 10 = -4i +
7
0037                f1_end:
0037 A3 0008 R          mov var_i1, ax ; i1 = f1(i)
003A 89 0E 000A R      mov var_i2, cx ; i2 = f2(i)
003E                f3:
003E 8B 1E 000A R      mov bx, var_i2
0042 83 FB 00          cmp bx, 0h
0045 7D 02             jge abs_i2
Microsoft (R) Macro Assembler Version 5.10          10/21/22
10:46:3
Page
1-2

```

```

0047 F7 DB             neg bx
0049                abs_i2: ; bx = |i2|
0049 A1 0006 R          mov ax, var_k
004C 3D 0000           cmp ax, 0h
004F 7C 10             jl f3_1
0051                f3_2:
0051 83 EB 03           sub bx, 3h ; bx = |i2| - 3
0054 83 FB 04           cmp bx, 4h
0057 7D 03             jge max
0059 BB 0004           mov bx, 4h
005C                max:
005C 8B C3             mov ax, bx
005E EB 0D 90          jmp f3_end
0061                f3_1:
0061 A1 0008 R          mov ax, var_i1
0064 3D 0000           cmp ax, 0h
0067 7D 02             jge abs_i1
0069 F7 D8             neg ax
006B                abs_i1: ; ax = |i1|
006B 2B C3             sub ax, bx
006D                f3_end:
006D A3 000C R          mov var_res, ax
0070 CB               ret
0071                Main ENDP
0071                CODE ENDS
0071                END Main
Microsoft (R) Macro Assembler Version 5.10          10/21/22
10:46:3

```

Symbols-1

#### Segments and Groups:

	N a m e	Length	Align	Combine Class
ASTACK	. . . . .	0040	PARA	STACK
CODE	. . . . .	0071	PARA	NONE
DATA	. . . . .	000E	PARA	NONE

#### Symbols:

N a m e	Type	Value	Attr
---------	------	-------	------



ABS_I1 . . . . .	L NEAR	006B	CODE	
ABS_I2 . . . . .	L NEAR	0049	CODE	
F1 . . . . .	L NEAR	0009	CODE	
F1_1 . . . . .	L NEAR	002D	CODE	
F1_2 . . . . .	L NEAR	001A	CODE	
F1_END . . . . .	L NEAR	0037	CODE	
F3 . . . . .	L NEAR	003E	CODE	
F3_1 . . . . .	L NEAR	0061	CODE	
F3_2 . . . . .	L NEAR	0051	CODE	
F3_END . . . . .	L NEAR	006D	CODE	
0071 MAIN . . . . .	F PROC	0000	CODE	Length =
MAX . . . . .	L NEAR	005C	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		

77 Source Lines  
77 Total Lines  
27 Symbols

48038 + 459222 Bytes symbol space free

0 Warning Errors  
0 Severe Errors