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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Организация ЭВМ и систем»

Тема: Изучение организации ветвлений в программах на языке Assembler.

Студентка гр. 9382 _____ Балаева М.О.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2020

Цель работы.

Изучение организации ветвлений и управляющих структур на языке Assembler.

Постановка задачи.

Разработать на языке Ассемблера IBM PC программу, которая по заданным целым значениям a, b, i, k, размером 1 слово, вычисляет:

Значения $i_1 = f_1(a, b, i)$ и $i_2 = f_2(a, b, i)$,

Значения $res = f_3(i_1, i_2, k)$,

где функции f_1 и f_2определяются из таблицы 1, а f_3 - из таблицы 2 по цифрам шифра индивидуального задания.

Значения a, b, i, k являются исходными данными, которые должны быть выбраны самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть все возможные комбинации параметров a, b, i, k, позволяющие проверить различные маршруты выполнения программы.

Вариант 10:

1. $\{-(4i+3), при\ a > b\ 6i - 10, при\ a \le b$

2. $\{20 - 4i, при\ a > b\ -(6i - 6), при\ a \le b$

3. $\{|i_1| - i_2 \vee, при\ k < 0\ max(7, |i_2|), при\ k \ge 0$

Выполнение работы.

1. Используемые операции.

Для выполнения вычислений использовались операции:

a) **ADD** – для сложения двух данных и записи в регистр
b) **SUB** – для вычитания двух данных и записи в регистр
c) **NEG** – для получения противоположного значения данных в регистре
d) **SAL** – команда осуществляет сдвиг влево всех битов операнда.

Для выполнения условных переходов использовались следующие операции:

e) **CMP** - сравнение двух чисел
f) **JLE** – меньше или равно
g) **JL** – меньше
h) **JMP** – безусловный переход
i) **JE** – равно
j) **JGE** – больше или равно
k) **JG** – больше

2. Тестирование программы.

№ теста	Исходные данные	Ожидаемый результат	Полученный результат	Корректность работы программы
1	$a=0002_{16} = 2_{10}$ $b=0004_{16} = 4_{10}$ $i= 0005_{16} = 5_{10}$ $k=0000_{16} = 0_{10}$	$i1=0014_{16} = 20_{10}$ $i2=FFE8_{16} = -24_{10}$ $res=0018_{16} = 24_{10}$	$i1=0014_{16} = 20_{10}$ $i2=FFE8_{16} = -24_{10}$ $res=0018_{16} = 24_{10}$	+
2	$a=0002_{16} = 2_{10}$ $b=0004_{16} = 4_{10}$ $i= 0005_{16} = 5_{10}$ $k=0002_{16} = 2_{10}$	$i1=0014_{16} = 20_{10}$ $i2=FFE8_{16} = -24_{10}$ $res=0018_{16} = 24_{10}$	$i1=0014_{16} = 20_{10}$ $i2=FFE8_{16} = -24_{10}$ $res=0018_{16} = 24_{10}$	+
3	$a=0004_{16} = 4_{10}$ $b=0002_{16} = 2_{10}$ $i= 0005_{16} = 5_{10}$ $k=0000_{16} = 0_{10}$	$i1=FFE9_{16} = -23_{10}$ $i2=0000_{16} = 0_{10}$ $res=0007_{16} = 7_{10}$	$i1=FFE9_{16} = -23_{10}$ $i2=0000_{16} = 0_{10}$ $res=0007_{16} = 7_{10}$	+
4	$a=0004_{16} = 4_{10}$ $b=0002_{16} = 2_{10}$ $i= 0005_{16} = 5_{10}$ $k=FFE9_{16}=-23_{10}$	$i1=FFE9_{16} = -23_{10}$ $i2=0000_{16} = 0_{10}$ $res=0017_{16} = 23_{10}$	$i1=FFE9_{16} = -23_{10}$ $i2=0000_{16} = 0_{10}$ $res=0017_{16} = 23_{10}$	+

5	a=$FFFC_{16}$ = -4_{10} b=$FFFE_{16}$ = -2_{10} i= 0005_{16} = 5_{10} k=0001_{16}=1_{10}	i1=0014_{16} = 20_{10} i2=$FFE8_{16}$ = -24_{10} res=0018_{16} = 24_{10}	i1=0014_{16} = 20_{10} i2=$FFE8_{16}$ = -24_{10} res=0018_{16} = 24_{10}	+

Выводы.

В ходе данной лабораторной работы была изучена организация ветвления, а также операция сравнения, реализация меток и переход по данным меткам на языке Ассемблера. В ходе разработки программы была применена минимизация кода, результаты вычислений контролировались в режиме отладки.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

DATA SEGMENT
        A       DW 0
        B       DW 0
        I       DW 0
        K       DW 0
        I1      DW ?
        I2      DW ?
        RES DW ?
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   BX,I
        sal   BX,1h
        sal   BX,1  ;4i is needed in both parts
        mov   AX,A
        cmp   AX,B
        jle cond_b

        add BX,3        ;4i + 3
        neg BX
        mov I1,BX
        add BX,23
        mov I2,BX
        jmp f3

cond_b:
```

```
        add BX,I
        add BX,I
        sub BX,10  ;6i - 10
        mov I1,BX
        neg BX
        sub BX, 4; BX-4
        mov I2,BX
f3:
        cmp BX,0; i2 is there and we need abs value in both parts
        jge skip_abs_1
        neg BX
skip_abs_1:
        mov AX,K
        cmp AX,0
        jge f3_b
        mov AX,I1
        cmp AX, 0
        jge skip_abs_2
        neg AX
skip_abs_2:
        sub AX,BX
        jmp f3_end
f3_b:
        cmp BX,7
        jge max_b
        mov AX, 7
        jmp f3_end
max_b:
        mov AX,BX
f3_end:
        mov RES,AX
   ret
Main ENDP
        CODE     ENDS
        END Main
```

ПРИЛОЖЕНИЕ Б
ФАЙЛ ЛИСТИНГА

Page 1-1

```
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                I1      DW ?
000A  0000                I2      DW ?
000C  0000                RES DW ?
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  8B 1E 0004 R      mov   BX,I
000D  D1 E3             sal   BX,1
000F  D1 E3             sal   BX,1  ;4i is needed in both parts
0011  A1 0000 R         mov   AX,A
0014  3B 06 0002 R      cmp   AX,B
0018  7E 13             jle cond_b
```

```
001A  83 C3 03                        add BX,3        ;4i + 3
001D  F7 DB                     neg BX
001F  89 1E 0008 R              mov I1,BX
0023  83 C3 17                        add BX,23
0026  89 1E 000A R              mov I2,BX
002A  EB 19 90                        jmp f3

002D                            cond_b:
002D  03 1E 0004 R              add BX,I
0031  03 1E 0004 R              add BX,I
0035  83 EB 0A                        sub BX,10  ;6i - 10
0038  89 1E 0008 R              mov I1,BX
003C  F7 DB                     neg BX
003E  83 EB 04                        sub BX, 4; BX-4
0041  89 1E 000A R              mov I2,BX
0045                            f3:
0045  83 FB 00                        cmp BX,0; i2 is there and we need abs v
                                alue in both parts
0048  7D 02                     jge skip_abs_1
004A  F7 DB                     neg BX
004C                            skip_abs_1:



004C  A1 0006 R                 mov AX,K
004F  3D 0000                         cmp AX,0
0052  7D 0F                     jge f3_b
0054  A1 0008 R                 mov AX,I1
0057  3D 0000                         cmp AX, 0
005A  7D 02                     jge skip_abs_2
005C  F7 D8                     neg AX
005E                            skip_abs_2:
005E  2B C3                     sub AX,BX
0060  EB 0E 90                        jmp f3_end
0063                            f3_b:
0063  83 FB 07                        cmp BX,7
0066  7D 06                     jge max_b
0068  B8 0007                         mov AX, 7
006B  EB 03 90                         jmp f3_end
```

```
006E                    max_b:
006E  8B C3                     mov AX,BX
0070                    f3_end:
0070  A3 000C R                 mov RES,AX
0073  CB                    ret
0074                    Main ENDP
0074                            CODE    ENDS
                                END Main

                                Symbols-1


Segments and Groups:

                N a m e         Length  Align   Combine Class

ASTACK . . . . . .  .           0018    PARA    STACK
CODE . . . . . .  .             0074    PARA    NONE
DATA . . . . . .  .             000E    PARA    NONE

Symbols:

                N a m e         Type     Value   Attr

A  . . . . . .  .               L WORD          0000    DATA

B  . . . . . .  .               L WORD          0002    DATA

COND_B . . . . . .  .           L NEAR          002D    CODE

F3 . . . . . .  .               L NEAR          0045    CODE
F3_B . . . . . .  .             L NEAR          0063    CODE
F3_END . . . . . .  .           L NEAR          0070    CODE

I  . . . . . .  .               L WORD          0004    DATA
I1 . . . . . .  .               L WORD          0008    DATA
I2 . . . . . .  .               L WORD          000A    DATA

K  . . . . . .  .               L WORD          0006    DATA

MAIN . . . . . .  .             F PROC          0000    CODE    Length = 0074
```

```
MAX_B . . . . . .     L NEAR        006E   CODE

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

SKIP_ABS_1 . . . . . .  L NEAR      004C   CODE
SKIP_ABS_2 . . . . . .  L NEAR      005E   CODE

@CPU . . . . . .      TEXT  0101h
@FILENAME . . . . . .   TEXT  ANN_3
@VERSION . . . . . .    TEXT  510


    73 Source  Lines
    73 Total   Lines
    23 Symbols

 48058 + 461249 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```

ПРИЛОЖЕНИЕ В

ФАЙЛ КАРТ ПАМЯТИ

```
Start  Stop   Length Name               Class
00000H 00017H 00018H ASTACK
00020H 0002DH 0000EH DATA
00030H 000A3H 00074H CODE

Program entry point at 0003:0000
```