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

отчет

по лабораторной работе №3

по дисциплине «Организация ЭВМ и систем»

Тема: Представление и обработка целых чисел. Организация ветвящихся процессов

Студент гр. 9382 _____ Рыжих Р.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Научиться представлять и обрабатывать целые числа, а также организовывать ветвящиеся процессы.

Основные теоретические положения.

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 функция f3 – из табл.3 по цифрам шифра индивидуального задания (n1, n2, n3) приведенным в таблице 4.

Значения a, b, i, k являются исходными данными, которые должны выбираться студентов 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

Вариант 17

```
        / 7 - 4*i ,  при  a>b
f3 = <
        \ 8 -6*i ,   при  a<=b

        / -(4*i -5) , при  a>b
f7 = <
        \  10 - 3*i , при  a<=b
        /  min(|i1|, 6), при  k=0
f5 = <
        \  |i1|+|i2|, при  k/=0
```

Ход работы:

В сегменте данных объявлены переменные a, b, i, k, i1, i2, res. Функции и ветвления реализованы через метки. Возвращаемые значения записываются в сегменте данных под соответствующей переменной или записываются в регистры. Для реализации ветвления использовалась команда CMP, она сравнивает два числа. В зависимости от результата сравнения, выполняется переход на ту или иную метку.

Исходный код программы.

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

DATA  SEGMENT                          ;SEG DATA
   a        DW 4
   b        DW 3
   i        DW 2
   k      DW  1
   i1     DW  ?
   i2     DW  ?
   res    DW  ?
DATA ENDS                                          ;ENDS DATA

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

Main    PROC  FAR
        push ds
        sub ax, ax
        push ax
        mov ax, DATA
        mov ds, ax
        mov ax, a
        cmp ax, b
jle f3
```

```
        mov ax, i ; f3_2
        shl ax, 1 ; i*4
        shl ax, 1 ; i*4
        neg ax
        add ax, 7
        mov i1, ax
        jmp f7_1
f3:
        mov ax, i ; f3_1
        shl ax, 1 ; i*4
        shl ax, 1 ;
        mov cx, i ;
        shl cx, 1 ; i*2
        add ax, cx; i*6
        neg ax
        add ax, 8 ;
        mov i1, ax;
        jmp f7_2
f7_1:
        mov ax, i ;
        shl ax, 1 ; i*4
        shl ax, 1 ;
        sub ax, 5 ;
        neg ax
        mov i2, ax
        jmp f5
f7_2:
        mov ax, i ;
        shl ax, 1 ; i*2
        add ax, i ; i*3
        neg ax    ; -i*3
        add ax, 10;
        mov i2, ax
        jmp f5
f5:
```

```
        mov ax, k
        cmp k, 0
jne f5_2        ; if k<>0
        mov ax, i1
        cmp ax, 6
jle i1_6
        mov res, 6
        jmp f_end
i1_6:
        mov ax, i1
        mov res, ax
        jmp f_end
f5_2:
        mov ax, i1
        cmp i1, 0
jl i1_neg ; i1<0
f_5_2_2:
        mov ax, i2
        cmp i2, 0
jl i2_neg ; i2<0
        jmp f_res
i1_neg:
        neg i1
        jmp f_5_2_2
i2_neg:
        neg i2
f_res:
        mov ax, i1
        add res, ax
        mov ax, i2
        add res, ax
f_end:
        mov ah, 4ch
        int 21h
```

```
Main      ENDP
CODE      ENDS
END Main                          ;ENDS CODE
```

Листинг программы.

```
Microsoft (R) Macro Assembler Version 5.10              11/25/20 05:03:2
                                                        Page     1-1


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

0018                        AStackENDS

0000                        DATA  SEGMENT                          ;SEG DA
                            TA
0000  0004                    a         DW 4
0002  0003                    b         DW 3
0004  0002                    i         DW 2
0006  0001                    k       DW  1
0008  0000                    i1      DW  ?
000A  0000                    i2      DW  ?
000C  0000                    res     DW  ?
000E                        DATA ENDS
                                                       ;ENDS DATA

0000                        CODE      SEGMENT
                                                ;SEG CODE
                            ASSUME  DS:DATA, CS:CODE, 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 0000 R                 mov ax, a
000C  3B 06 0002 R              cmp ax, b
0010  7E 12              jle f3
0012  A1 0004 R                 mov ax, i ; f3_2
0015  D1 E0                    shl ax, 1 ; i*4
0017  D1 E0                    shl ax, 1 ; i*4
0019  F7 D8                    neg ax
001B  05 0007                       add ax, 7
001E  A3 0008 R                 mov i1, ax
0021  EB 1B 90                     jmp f7_1
0024                      f3:
0024  A1 0004 R                 mov ax, i ; f3_1
0027  D1 E0                    shl ax, 1 ; i*4
0029  D1 E0                    shl ax, 1 ;
002B  8B 0E 0004 R              mov cx, i ;
002F  D1 E1                    shl cx, 1 ; i*2
0031  03 C1                    add ax, cx; i*6
0033  F7 D8                    neg ax
0035  05 0008                       add ax, 8 ;
0038  A3 0008 R                 mov i1, ax;
003B  EB 13 90                      jmp f7_2
003E                      f7_1:
003E  A1 0004 R                 mov ax, i ;
0041  D1 E0                    shl ax, 1 ; i*4
0043  D1 E0                    shl ax, 1 ;
Microsoft (R) Macro Assembler Version 5.10          11/25/20 05:03:2
                                                    Page     1-2


0045  2D 0005                       sub ax, 5 ;
0048  F7 D8                    neg ax
004A  A3 000A R                 mov i2, ax
004D  EB 15 90                      jmp f5
```

```
0050                                    f7_2:
0050  A1 0004 R                                 mov ax, i ;
0053  D1 E0                                      shl ax, 1 ; i*2
0055  03 06 0004 R                              add ax, i ; i*3
0059  F7 D8                                      neg ax    ; -i*3
005B  05 000A                                           add ax, 10;
005E  A3 000A R                                 mov i2, ax
0061  EB 01 90                                           jmp f5
0064                                    f5:
0064  A1 0006 R                                 mov ax, k
0067  83 3E 0006 R 00                                   cmp k, 0
006C  75 1A                             jne f5_2          ; if k<>0
006E  A1 0008 R                                 mov ax, i1
0071  3D 0006                                           cmp ax, 6
0074  7E 09                             jle i1_6
0076  C7 06 000C R 0006                         mov res, 6
007C  EB 39 90                                           jmp f_end
007F                                    i1_6:
007F  A1 0008 R                                 mov ax, i1
0082  A3 000C R                                 mov res, ax
0085  EB 30 90                                           jmp f_end
0088                                    f5_2:
0088  A1 0008 R                                 mov ax, i1
008B  83 3E 0008 R 00                                   cmp i1, 0
0090  7C 0D                             jl i1_neg ; i1<0
0092                                    f_5_2_2:
0092  A1 000A R                                 mov ax, i2
0095  83 3E 000A R 00                                   cmp i2, 0
009A  7C 09                             jl i2_neg ; i2<0
009C  EB 0B 90                                           jmp f_res
009F                                    i1_neg:
009F  F7 1E 0008 R                              neg i1
00A3  EB ED                                     jmp f_5_2_2
00A5                                    i2_neg:
00A5  F7 1E 000A R                              neg i2
```

```
00A9                    f_res:
00A9  A1 0008 R                 mov ax, i1
00AC  01 06 000C R              add res, ax
00B0  A1 000A R                 mov ax, i2
00B3  01 06 000C R              add res, ax
00B7                    f_end:
00B7  B4 4C                     mov ah, 4ch
00B9  CD 21                     int 21h

00BB                    Main    ENDP
00BB                    CODE    ENDS
                        END Main                        ;ENDS
                        CODE
Microsoft (R) Macro Assembler Version 5.10              11/25/20 05:03:2
                                Symbols-1


Segments and Groups:

                N a m e         Length  Align  Combine Class

ASTACK . . . . . .      0018    PARA  STACK
CODE . . . . . .        00BB    PARA  NONE
DATA . . . . . .        000E    PARA  NONE

Symbols:

                N a m e         Type    Value   Attr

A  . . . . . .          L WORD      0000    DATA

B  . . . . . .          L WORD      0002    DATA

F3 . . . . . .          L NEAR      0024    CODE
F5 . . . . . .          L NEAR      0064    CODE
```

```
F5_2 . . . . . .        L NEAR      0088   CODE
F7_1 . . . . . .        L NEAR      003E   CODE
F7_2 . . . . . .        L NEAR      0050   CODE
F_5_2_2 . . . . . .     L NEAR      0092   CODE
F_END . . . . . .       L NEAR      00B7   CODE
F_RES . . . . . .       L NEAR      00A9   CODE

I . . . . . .  L WORD    0004   DATA
I1 . . . . . .  L WORD   0008   DATA
I1_6 . . . . . .        L NEAR      007F   CODE
I1_NEG . . . . . .      L NEAR      009F   CODE
I2 . . . . . .  L WORD   000A   DATA
I2_NEG . . . . . .      L NEAR      00A5   CODE

K . . . . . .           L WORD      0006   DATA

MAIN . . . . . .        F PROC      0000   CODE Length = 00BB

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

@CPU . . . . . .        TEXT  0101h
@FILENAME . . . . . .         TEXT  LAB3
@VERSION . . . . . .    TEXT  510


    99 Source  Lines
    99 Total   Lines
    27 Symbols

  48040 + 461267 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```

Тестирование.

№	Входные данные	Выходные данные	Правильный результат
1	**a=1, b=1, i=1, k=1**	**i1=2, i2=7, res=9**	**i1=2, i2=7, res=9**
2	**a=1, b=2, i=3, k=-4**	**i1=-10, i2=1 res=11**	**i1=-10, i2=1 res=11**
3	**a=4, b=3, i=2, k=1**	**i1=-1, i2=-3, res=4**	**i1=-1, i2=-3, res=4**
4	**a=2, b=1, i=1, k=0**	**i1=-1, i2=-3, res=-1**	**i1=-1, i2=-3, res=-1**
5	**a=1, b=1, i=1, k=0**	**i1=2, i2=7, res=2**	**i1=2, i2=7, res=2**
6	**a=1, b=1, i=0, k=0**	**i1=8, i2=10, res=6**	**i1=8, i2=10, res=6**

Выводы.

Были изучены режимы адресации, определены ошибки в программе, и было дано объяснение ошибкам.