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. 9383 _____ Орлов Д.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление целых чисел, научиться их обрабатывать. Познакомиться с организацией ветвящихся процессов на Ассемблере.

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. из методички.

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.

Вариант 14:

```
        / 7 - 4*i , при a>b
f1 = <
        \ 8 -6*i , при a<=b

        / -(6*i - 4) , при a>b
f2 = <
        \ 3*(i+2) , при a<=b

        / max(i1,10-i2), при k<0
f3 = <
       \ |i1 - i2| , при k>=0
```

Ход работы.

В ходе работы была написана программа на языке Ассемблер, которая по заданным целочисленным параметрам вычисляет значения некоторых функций. Процесс выполнения программы ветвящийся и использует следующие команды Ассемблера:

- cmp – сравнение аргументов и установка флага ZF в соответствующее результату сравнения значение. 0, если аргументы равны и 1, если аргументы не равны.
- jle – условный переход по заданной метке при условии, что в предыдущем сравнении с использованием cmp первый аргумент меньше или равен второму.
- shl – побитовый сдвиг влево. Для целых чисел применение сдвига на 1 эквивалентно умножению значения на 2.
- add – арифметическое действие сложения целых чисел
- neg – арифметическое действие взятия противоположного целого числа
- sub – арифметическое действие вычитания целых чисел
- jmp – безусловный переход по заданной метке. Передача управления.

Исходные данные заносятся в программу до выполнения, а результат работы отслеживается через отладчик.

Тестирование.

Входные данные (a, b, i, k)	Результат вычислений (i1, i2, res)
1 1 1 1	0002=2 0009=9 0007 = 7
1 -1 4 -2	FFF7 = -9 FFEC = -20 001E = 30
-2 1 2 1	FFFC = -4 000C = 16 0010 = 16
-2 1 1 0	0002 = 2 0009 = 9 0007 = 7

Выводы.

Изучено представление целых чисел и разработана программа, выполняющая некоторые арифметические действия над целыми числами. Программа содержит ветвящиеся процессы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Текст файла lab3.asm

```
;       / 7 - 4*i , при a>b      (f1)
; f1 = <
;       \ 8 - 6*i , при a<=b       (f2)
;
;       / -(6*i - 4) , при a>b      (f11)
; f2 = <
;       \ 3*(i+2) , при a<=b      (f22)
;
;       / max(i1,10-i2), при k<0 (f3)
; f3 = <
;       \ |i1 - i2| , при k>=0       (absi)



AStack  SEGMENT STACK
   DW  32  DUP(?)
AStack  ENDS

DATA    SEGMENT
   A   DW  -2
   B   DW  1
   I   DW  4
   K   DW  0
   I1  DW  ?
   I2  DW  ?
   RES DW  ?
DATA    ENDS

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

Main    PROC FAR
   mov ax, DATA
   mov ds, ax

f1  :
   mov ax, A
   cmp ax, B   ; сравнивает A и B
         jle f11
   mov ax, 1    ; ax = 1
         sub ax, I         ; ax = 1 - i
         shl ax, 1         ; ax = 2 - 2i
         shl ax, 1         ; ax = 4 - 4i
         add ax, 3         ; ax = 7 - 4i
         mov I1,ax
         jmp f2

f2  :          ; A > B, так как мы переходим от f1
   add ax, I     ; ax = 7 - 4i + i = 7 - 3i
         sub ax, 5         ; ax = 7 - 3i - 5 = 2 - 3i
```

```
        shl ax, 1         ; ax = 2*(2 - 3i) = 4 - 6i = -(6i - 4)
        jmp f3


f11 :
        mov ax, 1         ; ax = 1
        sub ax, I         ; ax = 1 - i
   shl ax, 1      ;ax = 2*(1 - i) = 2 - 2i
        shl ax, 1         ;ax = 2*(2 - 2i) = 4 - 4i
        add ax, I         ;ax = 4 - 3i
        shl ax, 1         ;ax = 8 - 6i
        mov I1, ax
   jmp f22


f22 :
        neg ax     ;ax = -(8 - 6i) = 6i - 8
        shr ax, 1         ;ax = (6i - 8) / 2 = 3i - 4
        add ax, 10  ; ax = 3i - 4 + 10 = 3i + 6 = 3*(i + 2)
   jmp f3


f3  :
   cmp bx, K    ;max(i1, 10 - i2)
   jle absi
        neg ax            ; ax = -i2
        add ax, 10  ; ax = 10 - i2
        cmp I1, ax  ; i1 <= 10 - i2 -> ax --- max
        jle fin
        mov ax, I1
        jmp fin

absi :
   neg ax                 ;ax = -i2
        add ax, I1        ;ax = i1 - i2
        cmp bx, ax        ; i1 - i2 < 0 -> ax = -ax
        jle fin
        neg ax            ;ax = -ax
        jmp fin

fin :
   mov RES, ax
   mov  ah, 4ch
   int  21h

Main    ENDP
CODE    ENDS
END Main
```

Текст файла lab3.lst

```
Microsoft (R) Macro Assembler Version 5.10                12/24/20 11:30:3
                                                             Page     1-1


                        ;       / 7 - 4*i , при a>b      (f1)
                        ; f1 = <
                        ;       \ 8 - 6*i , при a<=b (f2)
                        ;
```

```
                        ;       / -(6*i - 4) , при a>b          (f11)
                        ; f2 = <
                        ;       \ 3*(i+2) , при a<=b            (f22)
                        ;
                        ;       / max(i1,10-i2), при k<0 (f3)
                        ; f3 = <
                        ;       \ |i1 - i2| , при k>=0          (absi)



0000                            AStack  SEGMENT STACK
0000  0020[                       DW  32  DUP(?)
   ????
                    ]

0040                            AStack  ENDS

0000                            DATA    SEGMENT
0000  FFFE                        A   DW  -2
0002  0001                        B   DW  1
0004  0004                        I   DW  4
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  B8 ---- R               mov ax, DATA
0003  8E D8                       mov ds, ax

0005                            f1  :
0005  A1 0000 R               mov ax, A
0008  3B 06 0002 R                cmp ax, B   ; сравнивает A и B
000C  7E 20                             jle f11
000E  B8 0001                     mov ax, 1     ; ax = 1
0011  2B 06 0004 R                      sub ax, I          ; ax = 1 - i
0015  D1 E0                             shl ax, 1          ; ax = 2 - 2i
0017  D1 E0                             shl ax, 1          ; ax = 4 - 4i
0019  05 0003                           add ax, 3          ; ax = 7 - 4i
001C  A3 0008 R                         mov I1,ax
001F  EB 01 90                          jmp f2

0022                            f2  :         ; A > B, так как мы пЙ
```

```
                        µреходим от f1
0022  03 06 0004 R              add ax, I       ; ax = 7 - 4i + i = 7 - 3i
0026  2D 0005                       sub ax, 5         ; ax = 7 - 3i - 5 = 2 -
Microsoft (R) Macro Assembler Version 5.10              12/24/20 11:30:3
                                    Page     1-2


                        3i
0029  D1 E0                         shl ax, 1         ; ax = 2*(2 - 3i) = 4 -
                        6i = -(6i - 4)
002B  EB 22 90                      jmp f3


002E                        f11 :
002E  B8 0001                       mov ax, 1         ; ax = 1
0031  2B 06 0004 R                  sub ax, I         ; ax = 1 - i
0035  D1 E0                       shl ax, 1       ;ax = 2*(1 - i) = 2 - 2i
0037  D1 E0                         shl ax, 1         ;ax = 2*(2 - 2i) = 4 -
                        4i
0039  03 06 0004 R                  add ax, I         ;ax = 4 - 3i
003D  D1 E0                         shl ax, 1         ;ax = 8 - 6i
003F  A3 0008 R                 mov I1, ax
0042  EB 01 90                    jmp f22


0045                        f22 :
0045  F7 D8                         neg ax      ;ax = -(8 - 6i) = 6i - 8
0047  D1 E8                         shr ax, 1         ;ax = (6i - 8) / 2 = 3i
                        - 4
0049  05 000A                       add ax, 10  ; ax = 3i - 4 + 10 = 3i + 6
                        = 3*(i + 2)
004C  EB 01 90                    jmp f3


004F                        f3  :
004F  3B 1E 0006 R                cmp bx, K     ;max(i1, 10 - i2)
0053  7E 11                       jle absi
0055  F7 D8                         neg ax            ; ax = -i2
0057  05 000A                       add ax, 10  ; ax = 10 - i2
005A  39 06 0008 R                  cmp I1, ax  ; i1 <= 10 - i2 -> ax --- m
                        ax
005E  7E 15                         jle fin
0060  A1 0008 R                 mov ax, I1
0063  EB 10 90                      jmp fin

0066                        absi :
0066  F7 D8                       neg ax                ;ax = -i2
```

```
0068  03 06 0008 R                    add ax, I1        ;ax = i1 - i2
006C  3B D8                           cmp bx, ax        ; i1 - i2 < 0 -> ax = -
                        ax
006E  7E 05                           jle fin
0070  F7 D8                           neg ax            ;ax = -ax
0072  EB 01 90                        jmp fin

0075                          fin :
0075  A3 000C R                   mov RES, ax
0078  B4 4C                       mov  ah, 4ch
007A  CD 21                       int  21h

007C                          Main    ENDP
007C                          CODE    ENDS
Microsoft (R) Macro Assembler Version 5.10              12/24/20 11:30:3
                                        Page     1-3


                        END Main
Microsoft (R) Macro Assembler Version 5.10              12/24/20 11:30:3
                                        Symbols-1


        Segments and Groups:

                N a m e         Length   Align   Combine Class

        ASTACK . . . . . .              0040    PARA  STACK
        CODE . . . . . .  007C    PARA  NONE
        DATA . . . . . .  000E    PARA  NONE

        Symbols:

                N a m e         Type     Value    Attr

        A . . . . . .     L WORD        0000     DATA
        ABSI . . . . . .  L NEAR        0066     CODE

        B . . . . . .     L WORD        0002     DATA

        F1 . . . . . .    L NEAR        0005     CODE
        F11 . . . . . .   L NEAR        002E     CODE
        F2 . . . . . .    L NEAR        0022     CODE
        F22 . . . . . .   L NEAR        0045     CODE
        F3 . . . . . .    L NEAR        004F     CODE
        FIN . . . . . .   L NEAR        0075     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 = 007C

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

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


    98 Source  Lines
    98 Total   Lines
    23 Symbols

  48018 + 461289 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```