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. 9383 _____ Лапина А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление целых чисел, научиться их обрабатывать, познакомиться с организаций ветвящихся процессов.

Текст задания.

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.

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

2) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

f1 = {-(4*i+3) , при a>b

{6*i-10, при a<=b

f2 = {7-4*i, при a>b

{8– 6*i, при a<=b

f3 = {|i1|-|i2|, при k<0

{max(4,|i2|-3), при k>=0

Ход работы.

В ходе работы была реализована программа на языке Ассемблер, которая по заданным целочисленным параметрам вычисляет значения функций. Исходные данные заносятся в программу до выполнения, а выходные данные отслеживаются через отладчик. Были реализованы следующие функции:

f1_1, f1_2 — для нахождения значения f1 (если a<=b, то выполняется f1_1, иначе f1_2);

f2_1, f2_2 — для нахождения значений f2 (если a<=b, то выполняется f2_1, иначе f2_2);

f3, f3_1, f_abs, f3_cmp_4, f3_res, f_abs_1, f_abs_2, f3_4 — для нахождения значений f3, так как используется несколько модульных значений, было необходимо предусмотреть все варианты и написать несколько функций.

Тестирование.

1) a = 1, b = 2, i = 2, k = -3 => f1 = 2, f2 = -4, f3 = -2
2) a = 1, b = 2, i = 2, k = 3 => f1 = 2, f2 = -4, f3 = 4
3) a = 2, b = 1, i = 2, k = -2 => f1 = -11, f2 = -1, f3 = 10
4) a = 2, b = 1, i = 2, k = 1 => f1 = -11, f2 = -1, f3 = 4

Выводы.

Изучено представление целых чисел, получены навыки работы с целыми числами и ветвящимися процессами.

Содержимое файла lb3.asm представлено в приложении А. Содержимое файла lb3.lst представлено в приложении Б.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

```
AStack SEGMENT STACK
     DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
a    DW   1
b    DW   2
i    DW   2
k    DW   -3
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_1:
     mov ax, a
     cmp ax, b
     jg f1_2         ;if a>b
                ;a<=b
     mov ax, i
     shl ax, 1  ;ax = 2*i
     mov bx, ax ;bx = 2*i
     shl ax, 1  ;ax = 4*i
     add ax, bx ;ax = 6*i
     sub ax, 10 ;ax = 6*i - 10
```

```
      mov i1, ax

      jmp f2_1
f1_2:
      mov ax, i
      shl ax, 1       ;ax = 2*i
      shl ax, 1       ;ax = 4*i
      add ax, 3       ;ax = 4*i+3
      neg ax             ;ax = -ax
      mov i1, ax

f2_1:
      mov ax, a
      cmp ax, b
      jg f2_2         ;if a>b
                 ;a<=b
      mov ax, i
      shl ax, 1  ;ax = 2*i
      mov bx, ax ;bx = 2*i
      shl ax, 1  ;ax = 4*i
      add ax, bx ;ax = 6*i
      neg ax          ;ax = -ax
         add ax, 8       ;ax = -6*i+8

      mov i2, ax
      jmp f3

f2_2:
      mov ax, i
      shl ax, 1  ;ax = 2*i
      mov bx, ax ;bx = 2*i
      shl ax, 1  ;ax = 4*i
      neg ax          ;ax = -ax
      add ax, 7  ;ax = -4*i + 7

      mov i2, ax

f3:
      mov ax, k
```

```
      cmp k, 0
      jl f3_1          ;if k < 0

      mov ax, i2
      cmp ax, 0  ;if ax < 0
      jl f_abs   ;then |ax|

      jmp f3_cmp_4

f3_1:
      mov ax, i1 ;ax = i1

      ;sub ax, i2     ;ax = i1 - i2
      cmp ax, 0  ;if ax < 0
        jl f_abs_1   ;then ax = |ax|

        mov res, ax
        mov ax, i2
           cmp ax, 0  ;if ax < 0
        jl f_abs_2   ;then ax = |ax|

        sub res, ax     ;res = |i1|-|i2|
      jmp f3_res
f_abs:
      neg ax           ;ax = -ax
        sub ax, 3       ;ax = |i2|-3
      jmp f3_cmp_4

f3_cmp_4:
      cmp ax, 4  ;if ax < 4
      jl f3_4          ;res = 4

      jmp f3_res

f3_res:
      mov res, ax     ;else res = ax

      jmp f_end
```

```
f_abs_1:
      neg ax           ;ax = |ax|
        mov res, ax
        mov ax, i2
           cmp ax, 0  ;if ax < 0
        jl f_abs_2    ;then ax = |ax|

        sub res, ax     ;res = |i1|-|i2|
      jmp f3_res

f_abs_2:
        neg ax           ;ax = |ax|
        sub res, ax     ;res = |i1|-|i2|
      jmp f3_res

f3_4:
      mov res, 4 ;res = 4
      jmp f_end

f_end:
      mov ah, 4ch
      int 21h

Main  ENDP
CODE  ENDS
      END Main
```

ПРИЛОЖЕНИЕ Б
ЛИСТИНГ И ИСПРАВЛЕННЫЙ КОД ПРОГРАММЫ

Название файла: lab2.lst

```
#Microsoft (R) Macro Assembler Version 5.10                    10/28/20
12:18:1
                                                                Page
1-1


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

0040                    AStack ENDS

0000                    DATA SEGMENT
0000  0001                   A     DW   1
0002  0002                   B     DW   2
0004  0002                   I     DW   2
0006  FFFD                   K     DW   -3
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_1:
0005  A1 0000 R               mov ax, a
0008  3B 06 0002 R            cmp ax, b
```

```
000C  7F 14                    JG F1_2          ;IF A>B
                                     ;A<=B
000E  A1 0004 R                MOV AX, I
0011  D1 E0                    SHL AX, 1   ;AX = 2*I
0013  8B D8                    MOV BX, AX  ;BX = 2*I
0015  D1 E0                    SHL AX, 1   ;AX = 4*I
0017  03 C3                    ADD AX, BX  ;AX = 6*I
0019  2D 000A                  SUB AX, 10 ;AX = 6*I - 10
001C  A3 0008 R                MOV I1, AX

001F  EB 10 90                 JMP F2_1
0022                 F1_2:
0022  A1 0004 R                MOV AX, I
0025  D1 E0                    SHL AX, 1        ;AX = 2*I
0027  D1 E0                    SHL AX, 1        ;AX = 4*I
0029  05 0003                  ADD AX, 3        ;AX = 4*I+3
002C  F7 D8                    NEG AX          ;AX = -AX
002E  A3 0008 R                MOV I1, AX

0031                 F2_1:
0031  A1 0000 R                MOV AX, A
0034  3B 06 0002 R             CMP AX, B
0038  7F 16                    JG F2_2          ;IF A>B
                                     ;A<=B
003A  A1 0004 R                MOV AX, I
003D  D1 E0                    SHL AX, 1   ;AX = 2*I
003F  8B D8                    MOV BX, AX  ;BX = 2*I
```

```
0041  D1 E0                    SHL AX, 1   ;AX = 4*I
0043  03 C3                    ADD AX, BX  ;AX = 6*I
0045  F7 D8                    NEG AX      ;AX = -AX
0047  05 0008                     ADD AX, 8        ;AX = -6*I+8

004A  A3 000A R                MOV I2, AX
004D  EB 12 90                 JMP F3

0050                 F2_2:
0050  A1 0004 R                MOV AX, I
0053  D1 E0                    SHL AX, 1   ;AX = 2*I
0055  8B D8                    MOV BX, AX  ;BX = 2*I
0057  D1 E0                    SHL AX, 1   ;AX = 4*I
0059  F7 D8                    NEG AX      ;AX = -AX
005B  05 0007                  ADD AX, 7   ;AX = -4*I + 7

005E  A3 000A R                MOV I2, AX

0061                 F3:
0061  A1 0006 R                MOV AX, K
0064  83 3E 0006 R 00               CMP K, 0
0069  7C 0B                    JL F3_1          ;IF K < 0

006B  A1 000A R                MOV AX, I2
006E  3D 0000                  CMP AX, 0   ;IF AX < 0
0071  7C 1D                    JL F_ABS    ;THEN |AX|

0073  EB 23 90                 JMP F3_CMP_4

0076                 F3_1:
0076  A1 0008 R                MOV AX, I1  ;AX = I1
```

```
                        ;SUB AX, I2 ;AX = I1 - I2
0079  3D 0000                CMP AX, 0   ;IF AX < 0
007C  7C 28                    JL F_ABS_1     ;THEN AX = |AX|

007E  A3 000C R                MOV RES, AX
0081  A1 000A R                MOV AX, I2
0084  3D 0000                    CMP AX, 0   ;IF AX < 0
0087  7C 30                    JL F_ABS_2     ;THEN AX = |AX|

0089  29 06 000C R             SUB RES, AX      ;RES = |I1|-|I2|
008D  EB 11 90              JMP F3_RES
0090              F_ABS:
0090  F7 D8                  NEG AX      ;AX = -AX
0092  2D 0003                  SUB AX, 3        ;AX = |I2|-3
0095  EB 01 90              JMP F3_CMP_4

0098              F3_CMP_4:
0098  3D 0004               CMP AX, 4   ;IF AX < 4
009B  7C 24                 JL F3_4          ;RES = 4

009D  EB 01 90              JMP F3_RES
```

```
00A0                    F3_RES:
00A0  A3 000C R                 MOV RES, AX ;ELSE RES = AX

00A3  EB 25 90                  JMP F_END

00A6                    F_ABS_1:
00A6  F7 D8                     NEG AX      ;AX = |AX|
00A8  A3 000C R                    MOV RES, AX
00AB  A1 000A R                    MOV AX, I2
00AE  3D 0000                         CMP AX, 0   ;IF AX < 0
00B1  7C 06                        JL F_ABS_2     ;THEN AX = |AX|

00B3  29 06 000C R                 SUB RES, AX      ;RES = |I1|-|I2|
00B7  EB E7                     JMP F3_RES

00B9                    F_ABS_2:
00B9  F7 D8                        NEG AX           ;AX = |AX|
00BB  29 06 000C R                 SUB RES, AX      ;RES = |I1|-|I2|
00BF  EB DF                     JMP F3_RES

00C1                    F3_4:
00C1  C7 06 000C R 0004         MOV RES, 4  ;RES = 4
00C7  EB 01 90                  JMP F_END

00CA                    F_END:
00CA  B4 4C                     MOV AH, 4CH
00CC  CD 21                     INT 21H

00CE                    MAIN  ENDP
00CE                    CODE  ENDS
                              END MAIN
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0040	PARA	STACK	
CODE	00CE	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
F1_1	L NEAR	0005	CODE
F1_2	L NEAR	0022	CODE
F2_1	L NEAR	0031	CODE
F2_2	L NEAR	0050	CODE
F3	L NEAR	0061	CODE
F3_1	L NEAR	0076	CODE
F3_4	L NEAR	00C1	CODE
F3_CMP_4	L NEAR	0098	CODE
F3_RES	L NEAR	00A0	CODE
F_ABS	L NEAR	0090	CODE
F_ABS_1	L NEAR	00A6	CODE
F_ABS_2	L NEAR	00B9	CODE
F_END	L NEAR	00CA	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  =
00CE

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

     @CPU . . . . . .  TEXT  0101H
     @FILENAME  . . . . . .  TEXT  LB3
     @VERSION . . . . . .  TEXT  510
```

136 SOURCE LINES
136 TOTAL LINES
29 SYMBOLS

48012 + 457198 BYTES SYMBOL SPACE FREE

0 WARNING ERRORS
0 SEVERE ERRORS