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

Студентка гр. 9383 _____ Лихашва А.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Научиться представлять и обрабатывать целые числа, а также организовывать ветвящиеся процесса

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

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.

Вариант 9

/ - (4*i+3) , при a>b

f2 = <

\ 6*i -10 , при a<=b

/ -(6*i - 4) , при a>b

f4 = <

\ 3*(i+2) , при a<=b

/ |i1| + |i2|, при k<0

f7 = <

\ max(6, |i1|), при k>=0

Ход работы:

В сегменте данных объявлены переменные a, b, i, k, i1, i2, res. Данная программа по заданным целочисленным параметрам вычисляет значения функций. Исходные данные вносятся в программу до выполнения, а выходные данные отслеживаются через отладчик. В программе были реализованы следующие функции:

f1_1, f1_2 – для нахождения значения функции f1. Если a <= b, то выполняется f1_1, в ином случае выполняется f1_2.

f2_1, f2_2 – для нахождения функции значения f2. Если a <= b, то выполняется f2_1, в ином случае выполняется f2_2.

f3, f3_1, f3_abs, f3_cmp_6, f3_res, f_abs_1, f_abs_2, f3_6 – для нахождения значений функции f3. Используется несколько модульных значений, поэтому нужно предусмотреть все варианты и написать несколько функций.

Тестирование.

№	Входные данные	Выходные данные
1	a=1, b=2, i=3, k=4	i1=8, i2=15, res=8
2	a=1, b=2, i=3, k=3	i1=8, i2=15, res=8
3	a=4, b=3, i=2, k=1	i1=-11, i2=-8, res=6
4	a=2, b=1, i=2, k=1	i1=-11, i2=-8, res=6

Выводы.

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

Файл lab3.asm находится в приложении А.

Файл lab3.lst находится в приложении Б.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

DATA SEGMENT
a     DW   1
b     DW   2
i     DW   3
k     DW   4
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, 4       ;ax = -6*i+4

     mov i2, ax
     jmp f3

f2_2:
     mov ax, i
     shl ax, 1 ;ax = 2*i
     add ax, i ;ax = 3*i
     add ax, 6 ;ax = 3*i + 6
```

```
     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_6

f3_1:
     mov ax, i1     ;ax = i1

     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|

        add res, ax     ;res = |i1|+|i2|
     jmp f3_res
f_abs:
     neg ax         ;ax = -ax
     jmp f3_cmp_6

f3_cmp_6:
     cmp ax, 6 ;if ax < 6
     jl f3_6        ;res = 6
     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|

        add res, ax     ;res = |i1|+|i2|
     jmp f3_res

f_abs_2:
        neg ax          ;ax = |ax|
        add res, ax     ;res = |i1|+|i2|
     jmp f3_res

f3_6:
     mov res, 6     ;res = 6
     jmp f_end

f_end:
     mov ah, 4ch
     int 21h

Main ENDP
CODE ENDS
     END Main
```

ПРИЛОЖЕНИЕ Б
ЛИСТИНГ

```
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  0003              i    DW   3
0006  0004              k    DW   4
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 0004                 add ax, 4       ;ax = -6*i+4

004A  A3 000A R            mov i2, ax
004D  EB 10 90                  jmp f3
```

```
0050                    f2_2:
0050  A1 0004 R              mov ax, i
0053  D1 E0                  shl ax, 1 ;ax = 2*i
0055  03 06 0004 R           add ax, i ;ax = 3*i
0059  05 0006                add ax, 6 ;ax = 3*i + 6
005C  A3 000A R              mov i2, ax


005F                    f3:
005F  A1 0006 R              mov ax, k
0062  83 3E 0006 R 00             cmp k, 0
0067  7C 0B                  jl f3_1        ;if k < 0

0069  A1 000A R              mov ax, i2
006C  3D 0000                cmp ax, 0 ;if ax < 0
006F  7C 1D                  jl f_abs  ;then |ax|

0071  EB 20 90                    jmp f3_cmp_6

0074                    f3_1:
0074  A1 0008 R              mov ax, i1     ;ax = i1

0077  3D 0000                cmp ax, 0 ;if ax < 0
007A  7C 25                     jl f_abs_1  ;then ax = |ax|

007C  A3 000C R                 mov res, ax
007F  A1 000A R                 mov ax, i2
0082  3D 0000                     cmp ax, 0 ;if ax < 0
0085  7C 2D                     jl f_abs_2  ;then ax = |ax|

0087  01 06 000C R              add res, ax     ;res = |i1|+|i2|
008B  EB 0E 90                    jmp f3_res
008E                    f_abs:
008E  F7 D8                  neg ax         ;ax = -ax
```

```
0090  EB 01 90                  jmp f3_cmp_6

0093                    f3_cmp_6:
0093  3D 0006               cmp ax, 6 ;if ax < 6
0096  7C 24                 jl f3_6        ;res = 6
0098  EB 01 90                  jmp f3_res

009B                    f3_res:
009B  A3 000C R             mov res, ax    ;else res = ax
009E  EB 25 90                  jmp f_end



00A1                    f_abs_1:
00A1  F7 D8                 neg ax         ;ax = |ax|
00A3  A3 000C R                mov res, ax
00A6  A1 000A R                mov ax, i2
00A9  3D 0000                    cmp ax, 0 ;if ax < 0
00AC  7C 06                    jl f_abs_2  ;then ax = |ax|

00AE  01 06 000C R             add res, ax     ;res = |i1|+|i2|
00B2  EB E7                 jmp f3_res

00B4                    f_abs_2:
00B4  F7 D8                    neg ax          ;ax = |ax|
00B6  01 06 000C R             add res, ax     ;res = |i1|+|i2|
00BA  EB DF                 jmp f3_res

00BC                    f3_6:
00BC  C7 06 000C R 0006     mov res, 6     ;res = 6
00C2  EB 01 90                  jmp f_end

00C5                    f_end:
00C5  B4 4C                 mov ah, 4ch
00C7  CD 21                 int 21h
```

```
 00C9                    Main ENDP
 00C9                    CODE ENDS
                         END Main


Segments and Groups:

                N a m e           Length     Align    Combine
Class

ASTACK . . . . . . . . . . . . . .  0040 PARA STACK
CODE . . . . . . . . . . . . . . .  00C9 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    005F CODE
F3_1 . . . . . . . . . . . . . . .  L NEAR    0074 CODE
F3_6 . . . . . . . . . . . . . . .  L NEAR    00BC CODE
F3_CMP_6 . . . . . . . . . . . . .  L NEAR    0093 CODE
F3_RES . . . . . . . . . . . . . .  L NEAR    009B CODE
F_ABS  . . . . . . . . . . . . . .  L NEAR    008E CODE
F_ABS_1  . . . . . . . . . . . . .  L NEAR    00A1 CODE
F_ABS_2  . . . . . . . . . . . . .  L NEAR    00B4 CODE
```

```
F_END  . . . . . .  L NEAR    00C5 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      =
00C9

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

@CPU . . . . . .  TEXT  0101h
@FILENAME  . . . . . .  TEXT  lab3
@VERSION . . . . . .  TEXT  510


    132 Source  Lines
    132 Total   Lines
     29 Symbols

  48056 + 459204 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```