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 _____ Корсунов А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер. Написать программу на основе изученного.

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. 1, а функции f3 - из табл.1. 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.

Имя функции	Функция
f1	15-2*i , при a>b
	3*i+4 , при a<=b
f2	-(4*i -5) , при a>b
	10 - 3*i , при a<=b
f3	|i1 - i2|, при k<0
	max(7, |i2|), при k>=0

Таблица 1

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.

Ход работы:

В ходе работы была написана программа, вычисляющая значение треух функций согласно целочисленным параметрам, которые записываются в сегменте данных.

В программе используется следующие операнды:

- mov – для переноса/присваивания значений
- cmp – для сравнения двух чисел (при его использовании изменяются флаги, которые используются для условных переходов)
- add – сумма чисел
- sub – разность чисел
- sal – побитовый сдвиг влево (для умножения чисел)
- neg – для изменения знака числа на противоположный
- jle – условный переход, срабатывающий, если левый аргумент выражения в cmp меньше или равен второму аргументу (срабатывает, при SF != OF и ZF == 1)
- jge – условный переход, срабатывающий, если левый аргумент выражения в cmp больше или равен второму аргументу (срабатывает, при SF == OF и ZF == 1)
- jg – условный переход, срабатывающий, если левый аргумент выражения в cmp больше второго аргумента (срабатывает при SF == OF и ZF == 0)
- jl – условный переход, срабатывающий, если левый аргумент выражения в cmp меньше второго аргумента (срабатывает при SF != OF и ZF == 0)
- jmp – безусловный переход (срабатывает в любом случае, делая переход по указанному адресу).

Примеры работы программы:

№ примера	a	b	i	k	i1	i2	res
1	2	1	2	-1	11	-3	14
2	2	4	3	-2	13	1	12
3	3	1	4	3	7	-11	11
4	3	5	-5	5	-11	25	25

Вывод:

Изучены представление и обработка целых чисел на языке Ассемблер. Написана программа на основе изученного.

Приложение А

main.asm:

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

DATA SEGMENT
a       DW 3
b       DW 5
i       DW -5
k       DW 5
i1      DW ?
i2      DW ?
res DW ?
DATA ENDS

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

Main PROC FAR
        mov ax, DATA
        mov ds, ax

fl_1:
        mov ax, a
        cmp ax, b
        jle fl_2
        mov ax, i
        sal ax, 1
        neg ax
```

```
        add ax, 15
        mov i1, ax
        jmp f2_1

f1_2:
        mov ax, i
        sal ax, 1
        add ax, i
        add ax, 4
        mov i1, ax
        jmp f2_2

f2_1:
        mov ax, i
        mov cl, 2
        sal ax, cl
        sub ax, 5
        neg ax
        mov i2, ax
        jmp f3_1

f2_2:
        mov ax, i
        sal ax, 1
        add ax, i
        neg ax
        add ax, 10
        mov i2, ax

f3_1:
```

```
        cmp k, 0
        jge f3_2
        mov ax, i1
        sub ax, i2
        cmp ax, 0
        jge f3_write
        neg ax
        jmp f3_write

f3_2:
        mov ax, i2
        cmp ax, 0
        jge f3_2_max
        neg ax

f3_2_max:
        cmp ax, 7
        jge f3_write
        mov ax, 7

f3_write:
        mov res, ax
        mov ah, 4ch
        int 21h

Main ENDP
CODE ENDS
        END Main
```

main.lst:

#Microsoft (R) Macro Assembler Version 5.10 11/11/20 18:28:1

Page 1-1

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

0018                    AStack ENDS

0000                    DATA SEGMENT
0000  0003              a       DW 3
0002  0005              b       DW 5
0004  FFFB                      i       DW -5
0006  0005              k       DW 5
0008  0000              i1      DW ?
000A  0000              i2      DW ?
000C  0000              res DW ?
000E                    DATA ENDS

0000                    CODE SEGMENT
                            ASSUME CS:CODE, SS:AStack, DS:DATA

0000                    Main PROC FAR
0000  B8 ---- R                mov ax, DATA
0003  8E D8                       mov ds, ax

0005                    fl_1:
```

```
0005  A1 0000 R                 mov ax, a
0008  3B 06 0002 R                    cmp ax, b
000C  7E 10                          jle f1_2
000E  A1 0004 R                 mov ax, i
0011  D1 E0                          sal ax, 1
0013  F7 D8                          neg ax
0015  05 000F                        add ax, 15
0018  A3 0008 R                 mov i1, ax
001B  EB 13 90                       jmp f2_1

001E                        f1_2:
001E  A1 0004 R                 mov ax, i
0021  D1 E0                          sal ax, 1
0023  03 06 0004 R                   add ax, i
0027  05 0004                        add ax, 4
002A  A3 0008 R                 mov i1, ax
002D  EB 13 90                       jmp f2_2

0030                        f2_1:
0030  A1 0004 R                 mov ax, i
0033  B1 02                          mov cl, 2
0035  D3 E0                          sal ax, cl
0037  2D 0005                        sub ax, 5
003A  F7 D8                          neg ax
003C  A3 000A R                 mov i2, ax
003F  EB 12 90                       jmp f3_1

0042                        f2_2:
0042  A1 0004 R                 mov ax, i
```

```
0045  D1 E0                    sal ax, 1
0047  03 06 0004 R             add ax, i
004B  F7 D8                    neg ax
004D  05 000A                  add ax, 10
0050  A3 000A R          mov i2, ax

0053                f3_1:
0053  83 3E 0006 R 00          cmp k, 0
0058  7D 11                    jge f3_2
005A  A1 0008 R           mov ax, i1
005D  2B 06 000A R             sub ax, i2
0061  3D 0000                  cmp ax, 0
0064  7D 17                    jge f3_write
0066  F7 D8                    neg ax
0068  EB 13 90                 jmp f3_write

006B                f3_2:
006B  A1 000A R           mov ax, i2
006E  3D 0000                  cmp ax, 0
0071  7D 02                    jge f3_2_max
0073  F7 D8                    neg ax

0075                f3_2_max:
0075  3D 0007                  cmp ax, 7
0078  7D 03                    jge f3_write
007A  B8 0007                  mov ax, 7
```

```
007D                    f3_write:
007D  A3 000C R                 mov res, ax
0080  B4 4C                     mov ah, 4ch
0082  CD 21                     int 21h

0084                    Main ENDP
0084                    CODE ENDS
                                END Main
#Microsoft (R) Macro Assembler Version 5.10          11/11/20 18:28:1
                                                     Symbols-1


Segments and Groups:

                N a m e         Length   AlignCombine Class

ASTACK . . . . . .              0018  PARA      STACK
CODE . . . . . .                0084  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     001E  CODE
F2_1 . . . . . .   L NEAR     0030  CODE
F2_2 . . . . . .   L NEAR     0042  CODE
F3_1 . . . . . .   L NEAR     0053  CODE
F3_2 . . . . . .   L NEAR     006B  CODE
F3_2_MAX . . . . . .       L NEAR     0075  CODE
F3_WRITE . . . . . .       L NEAR     007D  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 = 0084

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

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


    86 Source  Lines
    86 Total   Lines
    24 Symbols

  48016 + 463339 Bytes symbol space free

     0 Warning Errors
```

0 Severe Errors