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. 1382 _____ Исайкин Г. И.

Преподаватель _____ Ефремов М. А

Санкт-Петербург
2022

Цель работы.

Создать программу на Ассемблере с действиями над целыми числами и с ветвившимися процессами.

Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

- значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);
- значения результирующей функции res = f3(i1,i2,k)

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.

Таблица 1 — задание 8 варианта

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

Выполнение работы.

Программа состоит из трёх сегментов:

- Astack – блок стэка
- DATA – блок данных
- CODE — блок кода

Переменные a, b, i, k, i1, i2 находятся в блоке DATA. Основная логика находится в блоке кода, в процедуре Main. Программа сначала переносит значение i в регистр ax, после умножает на 2 (путём сдвига) и если:

- a>b то ещё раз умножает значение регистра на 2 и затем в переменную i1 перемещает значение ax и прибавляет к нему 3, после чего умножает на -1 путём побитовой инверсии, после чего из i2 (начальное значение i2 это 7) вычитает значение регистра ax, после чего программа «прыгает» к метки f3.
- a<=b то программа перемещается к метки f1_2, где к значению ax прибавляется i, значение ax ещё раз умножают на 2. В i1 перемещается значение ax, и из него вычитается 10. В i2 переносят значение 8 и вычитают из него значение ax.

Чтобы вычислить res, программа, если i2 меньше 0, то к его значение применяется инверсия. Сравнивает k с 0, и если:

- k<0, то делается инверсия i1, если оно меньше 0, и затем в res записывается i1 и вычитается i2, после программа прыгает в конец для завершения
- k>=0, то в ax записывается i2, вычитается 3 и затем сравнивается с res (начальное значение res – 3), и если ax меньше res, то программа прыгает в конец, иначе в res помещается значение ax, после чего работа завершается.

Таблица 2 — Тесты программы

№ теста	a	b	i	k	i1	i2	res
1	5	2	3	7	-15	-5	4 (верно)
	05 00	02 00	03 00	07 00	F1 FF	FB FF	04 00
2	3	7	5	4	20	-22	19 (верно)
	03 00	07 00	05 00	40 00	14 00	EA FF	13 00

3	-1	-5	-2	-1	5	15	-10 (верно)
	FF FF	FB FF	FE FF	FF FF	05 00	0F 00	F6 FF
4	0	0	56	0	326	-328	325 (верно)
	00 00	00 00	38 00	00 00	46 01	B8 FE	45 01

Выводы.

Были изучены базовые действия над целыми числами и ветвящиеся процессы на языке Ассемблер.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lr3.asm

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

DATA SEGMENT
    a DW 0
    b DW 0
    i DW 56
    k DW 0

    i1 DW 0
    i2 DW 7
    res DW 4
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack
    Main PROC FAR
        push ds
        mov ax, 0
        push ax
        mov ax, DATA
        mov ds, ax
        f1: ;2
        mov ax, i
        sal ax, 1
        mov cx, a
        cmp cx, b
        jle f1_2
        f1_1: ; - (4*i+3)
        sal ax, 1
        mov i1, ax
        add i1, 3
        neg i1
        f2_1: ;7 - 4*i
        sub i2, ax
        jmp f3
        f1_2: ;6*i -10
        add ax, i
        sal ax, 1
        mov i1, ax
        sub i1, 10
        f2_2: ; 8 -6*i
        mov i2, 8
        sub i2, ax
        f3: ;8
        cmp i2, 0
        jge f3_skeep_abs_i2
        neg i2
```

```
        f3_skeep_abs_i2:
        cmp k, 0
        jge f3_2
        f3_1: ; |i1| - |i2|
        cmp i1, 0
        jge f3_1_1
        neg i1
        f3_1_1:
        mov ax, i1
        mov res, ax
        mov ax, i2
        sub res, ax
        jmp fend
        f3_2: ;max(4,|i2|-3)
        mov ax, i2
        sub ax, 3
        cmp ax, res
        jl fend
        mov res, ax
        fend:
        ret
    Main ENDP
CODE ENDS
END Main
```

ДИАГНОСТИЧЕСКИЕ СООБЩЕНИЯ

Название файла: lr2.lst

```
Microsoft (R) Macro Assembler Version 5.10                  11/7/22
20:24:33
                                                             Page
1-1


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

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  0000                       a DW 0
 0002  0000                       b DW 0
 0004  0038                       i DW 56
 0006  0000                       k DW 0

 0008  0000                       i1 DW 0
 000A  0007                       i2 DW 7
 000C  0004                       res DW 4
 000E                   DATA ENDS

 0000                   CODE SEGMENT
                            ASSUME CS:CODE, DS:DATA, SS:AStack
 0000                       Main PROC FAR
 0000  1E                       push ds
 0001  B8 0000                       mov ax, 0
 0004  50                       push ax
 0005  B8 ---- R                mov ax, DATA
 0008  8E D8                         mov ds, ax
 000A                           f1: ;2
 000A  A1 0004 R                mov ax, i
 000D  D1 E0                         sal ax, 1
 000F  8B 0E 0000 R                  mov cx, a
 0013  3B 0E 0002 R                  cmp cx, b
 0017  7E 15                         jle f1_2
 0019                           f1_1: ; - (4*i+3)
 0019  D1 E0                         sal ax, 1
 001B  A3 0008 R                mov i1, ax
 001E  83 06 0008 R 03                    add i1, 3
 0023  F7 1E 0008 R                  neg i1
 0027                           f2_1: ;7 - 4*i
 0027  29 06 000A R                  sub i2, ax
 002B  EB 19 90                      jmp f3
 002E                           f1_2: ;6*i -10
 002E  03 06 0004 R                  add ax, i
 0032  D1 E0                         sal ax, 1
 0034  A3 0008 R                mov i1, ax
 0037  83 2E 0008 R 0A                    sub i1, 10
 003C                           f2_2: ; 8 -6*i
```

```
 003C  C7 06 000A R 0008           mov i2, 8
 0042  29 06 000A R                sub i2, ax
 0046                         f3: ;8
 0046  83 3E 000A R 00                  cmp i2, 0
 004B  7D 04                       jge f3_skeep_abs_i2
 004D  F7 1E 000A R                neg i2
 0051                         f3_skeep_abs_i2:
 0051  83 3E 0006 R 00                  cmp k, 0
```



```
 0056  7D 1B                       jge f3_2
 0058                         f3_1: ; |i1| - |i2|
 0058  83 3E 0008 R 00                  cmp i1, 0
 005D  7D 04                       jge f3_1_1
 005F  F7 1E 0008 R                neg i1
 0063                         f3_1_1:
 0063  A1 0008 R                   mov ax, i1
 0066  A3 000C R                   mov res, ax
 0069  A1 000A R                   mov ax, i2
 006C  29 06 000C R                sub res, ax
 0070  EB 10 90                    jmp fend
 0073                         f3_2: ;max(4,|i2|-3)
 0073  A1 000A R                   mov ax, i2
 0076  2D 0003                     sub ax, 3
 0079  3B 06 000C R                cmp ax, res
 007D  7C 03                       jl fend
 007F  A3 000C R                   mov res, ax
 0082                         fend:
 0082  CB                         ret
 0083                     Main ENDP
 0083                 CODE ENDS
                      END Main
```


Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0018	PARA	STACK
CODE	0083	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 . . . . . .     L NEAR     000A  CODE
F1_1 . . . . . .   L NEAR     0019  CODE
F1_2 . . . . . .   L NEAR     002E  CODE
F2_1 . . . . . .   L NEAR     0027  CODE
F2_2 . . . . . .   L NEAR     003C  CODE
F3 . . . . . .     L NEAR     0046  CODE
F3_1 . . . . . .   L NEAR     0058  CODE
F3_1_1 . . . . . .  L NEAR     0063  CODE
F3_2 . . . . . .   L NEAR     0073  CODE
F3_SKEEP_ABS_I2  . . . . . .  L NEAR  0051  CODE
FEND . . . . . .   L NEAR     0082  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 =
0083

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

@CPU . . . . . .   TEXT  0101h
@FILENAME  . . . . . .  TEXT  LR3
@VERSION . . . . . .  TEXT  510


     73 Source  Lines
     73 Total   Lines
     27 Symbols

  48070 + 461237 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```