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. 1381 _____ Смирнов Д. Ю.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблера. Научиться организовывать ветвящиеся процессы.

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 $i2 = f2(a, b, i)$

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.

Вариант 12 (2.7.4)

$$f1 = \begin{cases} -(4i + 3), & \text{при } a > b \\ 6i - 10, & \text{при } a \le b \end{cases}$$

$$f2 = \begin{cases} -(4i - 5), & \text{при } a > b \\ 10 - 3i, & \text{при } a \le b \end{cases}$$

$$f3 = \begin{cases} \min(|i1 - i2|, 2), & \text{при } k < 0 \\ \max(-6, -i2), & \text{при } k \ge 0 \end{cases}$$

Выполнение работы.

При выполнении данной лабораторной работы в программе было создано три сегмента: сегмент стека *AStack*, сегмент данных *DATA* и сегмент кода *CODE*. В сегменте данных были объявлены переменные *a*, *b*, *i*, *k*, *i1*, *i2* и *result*.

В сегменте *CODE* была написана процедура *Main*, в которой прописаны инструкции для завершения работы программы. Было использовано 8 меток:

- *enter_function* вычисляет *4i* в регистре ax, а так же сравнивает a и b. Если $a \le b$ то переходит к метке *function1_p2*, иначе вычисляет

$f1 = -(4i + 3)$ и $f2 = -(4i - 5)$ после чего переходит к метке function12_end.

- *function1_p2* вычисляет случаи, когда $a \leq b$. $\mathrm{f1} = 6\mathrm{i} - 10$ и $f2 = 10 - 3i$ после чего переходит к метке *function12_end*.
- *function12_end* перемещает рассчитанные в предыдущих метках значения f1 и f2 в i1 и i2 соответственно, после переходит к
- *function3* сравнивает k с 0. Если $k < 0$ переходит к метке *f3_k_l0*, иначе к метке *f3_k_ge0*.
- *f3_k_l0* вычисляется значение i1-i2, если оно больше или равно 0, то переход к метке *min*, иначе вычисляется –(i1-i2) и только после этого переход к *min*.
- *min* сравнивает $|i1 - i2|$ и 2, если первое меньше то переход к метке function3_end, иначе в регистр, который хранил значение выражения $|i1 - i2|$ записывается 2 и переход к *function3_end*.
- *f3_k_ge0* вычисляет -i2 и сравнивает с -2, если первый операнд больше, то переход на *function3_end*. Иначе в регистр, который хранил -i2 записываем -2 и переходим на *function3_end*.
- *function3_end* перемещает рассчитанное значение f3 в *result*, после чего инструкция *ret*, программа завершена.

Таблица 1 – Тестирование программы

№	Входные данные	Результат	Комментарий
1	a = 1 b = 2 I = 3 k = 4	i1=0008 (8) i2=0001 (1) result=FFFF (-1)	Верный результат
2	a = 2 b = 1 I = 1 k = 0	i1=FFF9 (-11) i2=0001 (1) result=FFFF (-1)	Верный результат
3	a = 1 b = 3 i = -2 k = -1	i1=FFEA (-22) i2=0010 (16) result=0002 (2)	Верный результат

Выводы.

Были изучены представление и обработка целых чисел. Получены знания об организации ветвящихся процессов на языке Ассемблера.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

Название файла: lab3.asm

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

DATA SEGMENT
      a DW 1
      b DW 3
      i DW -2
      k DW -1
      i1 DW ?
      i2 DW ?
      result DW ?
DATA ENDS

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

Main PROC FAR
      push DS
      sub AX, AX
      push AX
      mov AX, DATA
      mov DS, AX

      jmp enter_function

enter_function:
      mov AX, i

      shl AX, 1  ;\
                       ;     > 4i
      shl AX, 1  ;|

      mov CX, a ; CX = a
      cmp b, CX

      jge function1_p2 ; a <= b

      ; a > b
      mov CX, AX ; CX = 4i

      add AX, 3 ; AX = 4i + 3
      neg AX ; -(4i + 3) f1 end

      ; f2 a > b
      sub CX, 5 ; 4i - 5
      neg CX ; -(4i - 5) f2 end

      jmp function12_end

function1_p2:
```

```
      mov CX, i ; CX = i
      shl CX, 1 ; CX = 2i
      add AX, CX ; 4i + 2i
      sub AX, 10 ; 6i - 10 f1 end

      ; f2 a <= b
      neg CX ; CX = -2i
      sub CX, i ; CX = -3i
      add CX, 10 ; 10 - 3i f2 end

      jmp function12_end

function12_end:
      mov i1, AX
      mov i2, CX

function3:
      mov CX, k
      cmp CX, 0
      jge f3_k_ge0
      jmp f3_k_l0

f3_k_l0: ; k < 0
      mov AX, i1
      sub AX, i2 ; AX = i1-i2

      cmp AX, 0
      jge  min   ; \ abs
      neg AX ; | abs
      jmp min

min:
      cmp AX, 2 ; |i1-i2| <= 2
      jle function3_end
      mov AX, 2
      jmp function3_end

f3_k_ge0: ; k >= 0
      mov AX, i2
      neg AX ; AX = -i2
      cmp AX, -6
      jge function3_end
      mov AX, -6
      jmp function3_end

function3_end:
      mov result, AX
      ret

Main ENDP
CODE ENDS

END Main
```

ПРИЛОЖЕНИЕ Б

ФАЙЛ ЛИСТИНГА

```
Microsoft (R) Macro Assembler Version 5.10                  10/23/22
14:58:3
                                                            Page     1-1

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

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  0001                        a DW 1
 0002  0003                        b DW 3
 0004  FFFE                        i DW -2
 0006  FFFF                        k DW -1
 0008  0000                        i1 DW ?
 000A  0000                        i2 DW ?
 000C  0000                        result DW ?
 000E                   DATA ENDS

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

 0000                   Main PROC FAR
 0000  1E                   push DS
 0001  2B C0                     sub AX, AX
 0003  50                   push AX
 0004  B8 ---- R            mov AX, DATA
 0007  8E D8                     mov DS, AX

 0009  EB 01 90                  jmp enter_function

 000C                   enter_function:
 000C  A1 0004 R            mov AX, i

 000F  D1 E0                     shl AX, 1  ;\
                                            ;    > 4i
 0011  D1 E0                     shl AX, 1  ;|

 0013  8B 0E 0000 R              mov CX, a ; CX = a
 0017  39 0E 0002 R              cmp b, CX

 001B  7D 0F                     jge function1_p2 ; a <= b

                            ; a > b
 001D  8B C8                     mov CX, AX ; CX = 4i

 001F  05 0003                   add AX, 3 ; AX = 4i + 3
 0022  F7 D8                     neg AX ; -(4i + 3) f1 end

                            ; f2 a > b
 0024  83 E9 05                  sub CX, 5 ; 4i - 5
 0027  F7 D9                     neg CX ; -(4i - 5) f2 end
```

```
 0029  EB 18 90                jmp function12_end

 002C                  function1_p2:
Microsoft (R) Macro Assembler Version 5.10                     10/23/22
14:58:3
                                                               Page     1-2


 002C  8B 0E 0004 R             mov CX, i ; CX = i
 0030  D1 E1                    shl CX, 1 ; CX = 2i
 0032  03 C1                    add AX, CX ; 4i + 2i
 0034  2D 000A                  sub AX, 10 ; 6i - 10 f1 end

                              ; f2 a <= b
 0037  F7 D9                    neg CX ; CX = -2i
 0039  2B 0E 0004 R             sub CX, i ; CX = -3i
 003D  83 C1 0A                 add CX, 10 ; 10 - 3i f2 end

 0040  EB 01 90                 jmp function12_end

 0043                  function12_end:
 0043  A3 0008 R           mov i1, AX
 0046  89 0E 000A R             mov i2, CX

 004A                  function3:
 004A  8B 0E 0006 R             mov CX, k
 004E  83 F9 00                 cmp CX, 0
 0051  7D 1F                    jge f3_k_ge0
 0053  EB 01 90                 jmp f3_k_l0

 0056                  f3_k_l0: ; k < 0
 0056  A1 0008 R           mov AX, i1
 0059  2B 06 000A R             sub AX, i2 ; AX = i1-i2

 005D  3D 0000                  cmp AX, 0
 0060  7D 05                    jge  min   ; \ abs
 0062  F7 D8                    neg AX ; | abs
 0064  EB 01 90                 jmp min

 0067                  min:
 0067  3D 0002                  cmp AX, 2 ; |i1-i2| <= 2
 006A  7E 16                    jle function3_end
 006C  B8 0002                  mov AX, 2
 006F  EB 11 90                 jmp function3_end

 0072                  f3_k_ge0: ; k >= 0
 0072  A1 000A R           mov AX, i2
 0075  F7 D8                    neg AX ; AX = -i2
 0077  3D FFFA                  cmp AX, -6
 007A  7D 06                    jge function3_end
 007C  B8 FFFA                  mov AX, -6
 007F  EB 01 90                 jmp function3_end

 0082                  function3_end:
 0082  A3 000C R           mov result, AX
 0085  CB                       ret
```

```
 0086                   Main ENDP
 0086                   CODE ENDS

                        END Main
Microsoft (R) Macro Assembler Version 5.10                      10/23/22
14:58:3
                                                              Symbols-1

Segments and Groups:

                N a m e         Length     Align     Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                0086  PARA  NONE
DATA . . . . . .                000E  PARA  NONE

Symbols:

                N a m e         Type   Value     Attr

A  . . . . . .                  L WORD     0000  DATA

B  . . . . . .                  L WORD     0002  DATA

ENTER_FUNCTION . . . . . .      L NEAR     000C  CODE

F3_K_GE0 . . . . . .            L NEAR     0072  CODE
F3_K_L0  . . . . . .            L NEAR     0056  CODE
FUNCTION12_END . . . . . .      L NEAR     0043  CODE
FUNCTION1_P2 . . . . . .        L NEAR     002C  CODE
FUNCTION3  . . . . . .          L NEAR     004A  CODE
FUNCTION3_END  . . . . . .      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 = 0086
MIN  . . . . . .                L NEAR     0067  CODE

RESULT . . . . . .              L WORD     000C  DATA

@CPU . . . . . .                TEXT  0101h
@FILENAME  . . . . . .          TEXT  LAB3
@VERSION . . . . . .            TEXT  510


    104 Source  Lines
    104 Total   Lines
     24 Symbols

  47976 + 461331 Bytes symbol space free
      0 Warning Errors
      0 Severe  Errors
```