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.
Вариант №24

Студент гр. 8381 _____ Сергеев А.Д.

Преподаватель _____ Кирьянчиков А.В.

Санкт-Петербург
2019

Постановка задачи.

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),

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.

f1 = $\begin{cases} 7-4i, при\ a>b \\ 8-6i, при\ a \le b \end{cases}$

f2 = $\begin{cases} 7-4i, при\ a>b \\ 8-6i, при\ a \le b \end{cases}$

f3 = $\begin{cases} 5-4i, при\ a>b \\ 10-3i, при\ a \le b \end{cases}$

Оптимизации:

1) при a > b пусть k = 20 - 4i

   тогда в f1: i1 = k

   в f2: i2 = k - 15

2) при a <= b пусть k = 10 - 3i

   тогда в f1: i1 = k*2 - 14

   в f2: i2 = k

Замечания:

3) 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;
4) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;
5) при вычислении функций f1 и f2 нельзя использовать процедуры;

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

Тестирование программы.

№ теста	Исходные данные	Ожидаемый результат	Полученный результат	Корректность работы программы
1	$A = 0002_{16} = 2_{10}$ $B = 0001_{16} = 1_{10}$ $I = 0010_{16} = 16_{10}$ $K = 0000_{16} = 0_{10}$	$I1 = FFD4_{16} = -44_{10}$ $I2 = FFC5_{16} = -59_{10}$ $RES = 0006_{16} = 6_{10}$	$I1 = FFD4_{16} = -44_{10}$ $I2 = FFC5_{16} = -59_{10}$ $RES = 0006_{16} = 6_{10}$	+
2	$A = FFFF_{16} = -1_{10}$ $B = FFFE_{16} = -2_{10}$ $I = 000A_{16} = 10_{10}$ $K = 0001_{16} = 1_{10}$	$I1 = FFEC_{16} = -20_{10}$ $I2 = FFDD_{16} = -35_{10}$ $RES = 0037_{16} = 55_{10}$	$I1 = FFEC_{16} = -20_{10}$ $I2 = FFDD_{16} = -35_{10}$ $RES = 0037_{16} = 55_{10}$	+
3	$A = 0001_{16} = 1_{10}$ $B = 0002_{16} = 2_{10}$ $I = 0001_{16} = 1_{10}$ $K = 0000_{16} = 0_{10}$	$I1 = 0000_{16} = 0_{10}$ $I2 = 0007_{16} = 7_{10}$ $RES = 0000_{16} = 0_{10}$	$I1 = 0000_{16} = 0_{10}$ $I2 = 0007_{16} = 7_{10}$ $RES = 0000_{16} = 0_{10}$	+
4	$A = FFFE_{16} = -2_{10}$ $B = FFFF_{16} = -1_{10}$ $I = 0003_{16} = 3_{10}$ $K = 0001_{16} = 1_{10}$	$I1 = FFF4_{16} = -12_{10}$ $I2 = 0001_{16} = 1_{10}$ $RES = 000D_{16} = 13_{10}$	$I1 = FFF4_{16} = -12_{10}$ $I2 = 0001_{16} = 1_{10}$ $RES = 000D_{16} = 13_{10}$	+

Вывод.

В ходе выполнения данной лабораторной работы были получены сведения о реализации сравнения, меток и перехода по ним, а также изучены организации ветвлений в программах на языке Ассемблера.

Приложение 1. Исходный код программы.

```
DOSSEG
 .MODEL SMALL
 .STACK 4h
 .DATA
   A  DW 0
   B  DW 0
   I  DW 0
   K  DW 0
   R  DW 0
 .CODE

 DataLoading:
   mov ax, @Data ; Getting the pointer to actual app data
   mov ds, ax ; Applying pointer

   mov cx, I ; dx == I



 ; OCCUPATION: dx == I
 DecidingF1and2:
   mov ax, B ; ax == B
   neg ax ; ax == -B
   add ax, A ; ax == A - B
   cmp ax, 0 ; A - B ? 0
   jng F1and2var2 ; Goes here if A - B <= 0 -> A <= B

 F1and2var1: ; Goes here if A - B > 0 -> A > B
   sal cx, 1 ; cx == I << 1 == I * 2
   sal cx, 1 ; cx == I * 22 << 1 == I * 4
   neg cx ; cx == -I * 4

   add cx, 20 ; cx == 20 - I * 4 == i1
   mov dx, cx ; dx == cx == 20 - I * 4
   add dx, -15 ; dx == 5 - I * 4 == i2
   jmp DecidingF3

 F1and2var2:
   mov dx, cx ; cx == I
   sal dx, 1 ; dx == 2 * I
   add dx, cx ; dx == 3 * I
   neg dx ; dx == - 3 * I

   add dx, 10 ; dx == 10 - 3 * I == i2
   mov cx, dx ; cx == 10 - 3 * I
   sal cx, 1 ; cx == 10 - 3 * I << 1 = 20 - 6 * I
   add cx, -14 ; cx == 6 - 6 * I == i1
```

```
; OCCUPATION: cx == i1, dx == i2
DecidingF3:
  neg cx ; cx == -i1
  js DecidingF3 ; Goes here if -i1 < 0
              ; cx == |i1|

  mov ax, K ; ax == K
  cmp ax, 0 ; K ? 0
  jne F3var2 ; Goes here if K != 0

F3var1: ; Goes here if K == 0
  cmp cx, 6 ; |i1| ? 6
  jb PushingResults ; Goes here if |i1| < 6

  mov cx, 6 ; cx == 6 == res
  jmp PushingResults

F3var2:
  neg dx ; dx == -i2
  js F3var2 ; Goes here if -i2 < 0
           ; dx == |i2|

  add cx, dx ; cx == |i1| + |i2| == res



; OCCUPATION: cx == res
PushingResults:
  mov R, cx ; res goes to memory at "R" pointer

END
```

Приложение 2. Содержимое файла листинга.

```
#Microsoft (R) Macro Assembler Version 5.10                11/13/19 02:24:1

                                                           Page     1-1


     1                          DOSSEG
     2                           .MODEL SMALL
     3                           .STACK 4h
     4                           .DATA
```

```
 5 0000  0000                   A  DW 0
 6 0002  0000                   B  DW 0
 7 0004  0000                   I  DW 0
 8 0006  0000                   K  DW 0
 9 0008  0000                   R  DW 0
10                          .CODE
11
12 0000                        DataLoading:
13 0000  B8 ---- R               mov ax, @Data ; Getting the pointer
                               to actual app data
14 0003  8E D8                   mov ds, ax ; Applying pointer
15
16 0005  8B 0E 0004 R            mov cx, I ; dx == I
17
18
19
20                          ; OCCUPATION: dx == I
21 0009                        DecidingF1and2:
22 0009  A1 0002 R               mov ax, B ; ax == B
23 000C  F7 D8                   neg ax ; ax == -B
24 000E  03 06 0000 R            add ax, A ; ax == A - B
25 0012  3D 0000                 cmp ax, 0 ; A - B ? 0
26 0015  7E 11                   jng F1and2var2 ; Goes here if A - B
                               <= 0 -> A <= B
27
28 0017                        F1and2var1: ; Goes here if A - B > 0
                               -> A > B
```

```
29 0017  D1 E1                sal cx, 1 ; cx == I << 1 == I * 2
30 0019  D1 E1                sal cx, 1 ; cx == I * 22 << 1 == I
                            * 4
31 001B  F7 D9                neg cx ; cx == -I * 4
32
33 001D  83 C1 14                   add cx, 20 ; cx == 20 - I * 4 == i1
34 0020  8B D1                mov dx, cx ; dx == cx == 20 - I * 4
35 0022  83 C2 F1                   add dx, -15 ; dx == 5 - I * 4 == i2
36 0025  EB 13 90                   jmp DecidingF3
37
38 0028                      F1and2var2:
39 0028  8B D1                mov dx, cx ; cx == I
40 002A  D1 E2                sal dx, 1 ; dx == 2 * I
41 002C  03 D1                add dx, cx ; dx == 3 * I
42 002E  F7 DA                neg dx ; dx == - 3 * I
43
44 0030  83 C2 0A                   add dx, 10 ; dx == 10 - 3 * I == i2
45 0033  8B CA                mov cx, dx ; cx == 10 - 3 * I
46 0035  D1 E1                sal cx, 1 ; cx == 10 - 3 * I << 1 =
                            20 - 6 * I
47 0037  83 C1 F2                   add cx, -14 ; cx == 6 - 6 * I == i1
48
49
```

```
50
51                          ; OCCUPATION: cx == i1, dx == i2
52 003A                     DecidingF3:
53 003A  F7 D9                neg cx ; cx == -i1
54 003C  78 FC                js DecidingF3 ; Goes here if -i1 <
                            0
55                              ; cx == |i1|
56
57 003E  A1 0006 R            mov ax, K ; ax == K
58 0041  3D 0000              cmp ax, 0 ; K ? 0
59 0044  75 0B                jne F3var2 ; Goes here if K != 0
60
61 0046                     F3var1: ; Goes here if K == 0
62 0046  83 F9 06             cmp cx, 6 ; |i1| ? 6
63 0049  72 0C                jb PushingResults ; Goes here if |i
                            1| < 6
64
65 004B  B9 0006              mov cx, 6 ; cx == 6 == res
66 004E  EB 07 90                   jmp PushingResults
67
68 0051                     F3var2:
69 0051  F7 DA                neg dx ; dx == -i2
70 0053  78 FC                js F3var2 ; Goes here if -i2 < 0
```

```
71                                      ; dx == |i2|
72
73 0055  03 CA                          add cx, dx ; cx == |i1| + |i2| == r
                                       es
74
75
76
77                                  ; OCCUPATION: cx == res
78 0057                                  PushingResults:
79 0057  89 0E 0008 R                   mov R, cx ; res goes to memory at "
                                       R" pointer
80
81                                  END
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
DGROUP	GROUP			
_DATA	000A	WORD	PUBLIC	'DATA'
STACK	0004	PARA	STACK	'STACK'
_TEXT	005B	WORD	PUBLIC	'CODE'

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	_DATA
B	L WORD	0002	_DATA
DATALOADING	L NEAR	0000	_TEXT
DECIDINGF1AND2	L NEAR	0009	_TEXT
DECIDINGF3	L NEAR	003A	_TEXT
F1AND2VAR1	L NEAR	0017	_TEXT
F1AND2VAR2	L NEAR	0028	_TEXT

```
F3VAR1 . . . . . .     L NEAR       0046   _TEXT
F3VAR2 . . . . . .     L NEAR       0051   _TEXT

I . . . . . .  L WORD       0004   _DATA

K . . . . . .          L WORD       0006   _DATA

PUSHINGRESULTS . . . . . .     L NEAR       0057   _TEXT

R . . . . . .          L WORD       0008   _DATA

@CODE . . . . . .      TEXT  _TEXT
@CODESIZE . . . . . .          TEXT  0
@CPU . . . . . .       TEXT  0101h
@DATASIZE . . . . . .          TEXT  0
@FILENAME . . . . . .          TEXT  lab3
@VERSION . . . . . .   TEXT  510


    81 Source  Lines
    81 Total   Lines
    30 Symbols

 47440 + 459820 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```