**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МОЭВМ**

# ОТЧЕТ

**по лабораторной работе №2**

**по дисциплине «Организация ЭВМ и систем»**

**ТЕМА: «Изучение режимов адресации и формирования исполнительного адреса»**

Студент гр. 1381 _____ Возмитель В.Е.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022 г.

**Цель работы.**

Получить знания о представлении и обработке целых чисел. Изучить понятие ветвящихся процессов и их организацию. Разработать на языке Ассемблера программу, вычисляющую значения функций, в зависимости от заданных параметров.

**Задание.**

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. 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.

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.

Заданные функции:

$$f1 = \begin{cases} 15 - 2i, & \text{при a} > \text{b} \\ 3\text{i} + 4, & \text{при a} \leq \text{b} \end{cases}$$

$$f5 = \begin{cases} 20 - 4i, & \text{при a} > \text{b} \\ -(6\text{i} - 6), & \text{при a} \leq \text{b} \end{cases}$$

$$f4 = \begin{cases} \min(|\text{i1} - \text{i2}|, \ 2), & \text{при k} < 0 \\ \max(-6, \ -\text{i2}), & \text{при k} \geq 0 \end{cases}$$

**Выполнение работы.**

1. Определение модели памяти с помощью директивы. model. Модель памяти - small. Описание упрощенных директив сегментации:
   - .stack – для указания начала сегмента стека;
   - .data – для указания начала сегмента данных;
   - .code – для указания начала сегмента кода;
2. Выбор размера стека, и инициализация переменных i, a, b, k;
3. Сравнение переменных a и b с помощью команды cmp. Если a > b переход к метке second, иначе выполняются команды метки first.
4. Далее происходит вычисление значений i1и i2 заданных функций. Умножение реализовано с помощью логического сдвига влево и сложения. Значение i1 сохраняется в регистре ax. Значение i2 сохраняется в регистре cx.
5. При k ≥ 0 сравниваются значения i2 и -6. Из этих чисел выбирается большее и сохраняется в регистре cx. Иначе вычисляются модуль разности i1 и i2 и после он сравнивается с 2. Из этих чисел выбирается меньшее и сохраняется в регистре cx. В переменную res сохраняется конечное значение.

## Минимизация длины кода.

- $a \leq b$

  i1 = 3i + 4;  i2 = -6i + 6.

  Пусть j = 3i + 4, тогда

  i1 = j; i2 = -2j + 14.

- $a > b$

  i1 = -2i + 15;  i2 = -4i + 20.

  Пусть j = -2i + 15, тогда

  i1 = j;  i2 = 2j - 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 А. Код программы LAB3.ASM.

```
dosseg
.model small
.stack 100h
.data
i dw 1
a dw 6
b dw 5
k dw -1
res dw ?
.code
mov ax, @data
mov ds, ax
mov ax, a
cmp ax, b
jg second
first: ;if(a<=b)
      mov ax, i   ;i
      shl ax, 1   ;2i
      add ax, i   ;3i
      add ax, 4  ;3i+4      (1)

      mov cx, ax  ;3i+4
      neg cx           ;-3i-4
      shl cx, 1  ;-6i-8
      add cx, 14 ;-(6i-6)   (2)
      jmp final

second: ;if(a>b)
      mov ax, i   ;i
      shl ax, 1   ;2i
      neg ax      ;-2i
      add ax, 15  ;-2i+15        (1)

      mov cx, ax  ;-2i+15
      shl cx, 1  ;-4i+30
      sub cx, 10 ;-4i+20         (2)

final:
min:
      neg cx
      cmp k, 0
      jge max
      add cx, ax    ;-i2+i1
      abs:
      neg cx
      js abs          ;|-i2+i1|
      cmp cx, 2
      jge  result_min ;if(|i2 - i1|>=2)
      jmp result
```

```
max:
      cmp cx, -6
      jle result_max ;if(-i2<=-6)
      jmp result

result_min:
      mov cx, 2
      jmp result

result_max: mov cx, -6
result:
      mov res, cx
      mov ah, 4ch
      int 21h
      end
```

## Приложение Б. Тестирование.

| № | Исходные данные | Ожидаемый результат | Полученный результат |
|---|---|---|---|
| 1 | $i = 0003_{16} = 3_{10}$<br>$a = 0001_{16} = 17_{10}$<br>$b = 0002_{16} = 15_{10}$<br>$k = 0004_{16} = -6_{10}$ | $i1 = 0009_{16} = 9_{10}$<br>$i2 = 0008_{16} = 8_{10}$<br>$res = 0001_{16} = 1_{10}$ | $i1 = 0009_{16} = 9_{10}$<br>$i2 = 0008_{16} = 8_{10}$<br>$res = 0001_{16} = 1_{10}$ |
| 2 | $i = 0002_{16} = 10_{10}$<br>$a = 0005_{16} = -17_{10}$<br>$b = 0000_{16} = -15_{10}$<br>$k = FFFF_{16} = 5_{10}$ | $i1 = 0022_{16} = 34_{10}$<br>$i2 = FFCA_{16} = -54_{10}$<br>$res = 0036_{16} = 54_{10}$ | $i1 = 0022_{16} = 34_{10}$<br>$i2 = FFCA_{16} = -54_{10}$<br>$res = 0036_{16} = 54_{10}$ |
| 3 | $i = FFFB_{16} = -5_{10}$<br>$a = 0001_{16} = 1_{10}$<br>$b = 0001_{16} = 1_{10}$<br>$k = 0000_{16} = 0_{10}$ | $i1 = FFF5_{16} = -11_{10}$<br>$i2 = 0024_{16} = 36_{10}$<br>$res = FFFA_{16} = -6_{10}$ | $i1 = FFF5_{16} = -11_{10}$<br>$i2 = 0024_{16} = 36_{10}$<br>$res = FFFA_{16} = -6_{10}$ |
| 4 | $i = 0000_{16} = 0_{10}$<br>$a = FFFB_{16} = -5_{10}$<br>$b = FFFD_{16} = -15_{10}$<br>$k = FFFF_{16} = -1_{10}$ | $i1 = 000F_{16} = 15_{10}$<br>$i2 = 0014_{16} = 20_{10}$<br>$res = 0002_{16} = 2_{10}$ | $i1 = 000F_{16} = 15_{10}$<br>$i2 = 0014_{16} = 20_{10}$<br>$res = 0002_{16} = 2_{10}$ |
| 5 | $i = FFFF_{16} = -1_{10}$<br>$a = 000A_{16} = 10_{10}$<br>$b = 000A_{16} = 10_{10}$<br>$k = 000A_{16} = 10_{10}$ | $i1 = 0001_{16} = 1_{10}$<br>$i2 = 000C_{16} = 12_{10}$<br>$res = FFFA_{16} = -6_{10}$ | $i1 = 0001_{16} = 1_{10}$<br>$i2 = 000C_{16} = 12_{10}$<br>$res = FFFA_{16} = -6_{10}$ |
| 6 | $i = 0001_{16} = 1_{10}$<br>$a = 0006_{16} = 6_{10}$<br>$b = 0005_{16} = 5_{10}$<br>$k = FFFF_{16} = -1_{10}$ | $i1 = 000D_{16} = 13_{10}$<br>$i2 = 0010_{16} = 16_{10}$<br>$res = 0002_{16} = 2_{10}$ | $i1 = 000C_{16} = 13_{10}$<br>$i2 = 0010_{16} = 16_{10}$<br>$res = 0002_{16} = 2_{10}$ |

## Приложение В. Содержимое файла листинга.

```
dosseg
                        .model small
                        .stack 100h
                        .data
0000  0001                   i dw 1
0002  0006                   a dw 6
0004  0005                   b dw 5
0006  FFFF                   k dw -1
0008  0000                   res dw ?
                        .code
0000  B8 ---- R         mov ax, @data
0003  8E D8                  mov ds, ax
0005  A1 0002 R         mov ax, a
0008  3B 06 0004 R           cmp ax, b
000C  7F 18                  jg second
000E                    first: ;if(a<=b)
000E  A1 0000 R              mov ax, i   ;i
0011  D1 E0                       shl ax, 1   ;2i
0013  03 06 0000 R                add ax, i   ;3i
0017  05 0004                     add ax, 4  ;3i+4      (1)


001A  8B C8                       mov cx, ax  ;3i+4
001C  F7 D9                       neg cx          ;-3i-4
001E  D1 E1                       shl cx, 1  ;-6i-8
0020  83 C1 0E                    add cx, 14 ;-(6i-6)   (2)
0023  EB 12 90                    jmp final

0026                    second: ;if(a>b)
0026  A1 0000 R              mov ax, i   ;i
0029  D1 E0                       shl ax, 1   ;2i
002B  F7 D8                       neg ax      ;-2i
002D  05 000F                     add ax, 15  ;-2i+15        (1)

0030  8B C8                       mov cx, ax  ;-2i+15
0032  D1 E1                       shl cx, 1  ;-4i+30
0034  83 E9 0A                    sub cx, 10 ;-4i+20         (2)

0037                    final:
0037                    min:
0037  F7 D9                       neg cx
0039  83 3E 0006 R 00                 cmp k, 0
003E  7D 0E                       jge max         ;if(k>=0)
0040  03 C8                       add cx, ax ;-i2+i1
0042                         abs:
0042  F7 D9                       neg cx
0044  78 FC                       js abs      ;|-i2+i1|
0046  83 F9 02                    cmp cx, 2
```

```
 0049  7D 0B                    jge  result_min ;if(|i2 - i1|>=2)
 004B  EB 12 90                 jmp result

 004E                max:
 004E  83 F9 FA                 cmp cx, -6
 0051  7E 09                    jle result_max ;if(-i2<=-6)
Microsoft (R) Macro Assembler Version 5.10                  10/30/22
14:13:2
                                                             Page
1-2


 0053  EB 0A 90                 jmp result

 0056                result_min:
 0056  B9 0002                  mov cx, 2
 0059  EB 04 90                 jmp result

 005C  B9 FFFA            result_max: mov cx, -6
 005F                result:
 005F  89 0E 0008 R             mov res, cx
 0063  B4 4C                    mov ah, 4ch
 0065  CD 21                    int 21h
                           end
Microsoft (R) Macro Assembler Version 5.10                  10/30/22
14:13:2
                                                             Symbols-1


Segments and Groups:

                N a m e         Length     Align     Combine Class

DGROUP . . . . . . . . . . . .      GROUP
  _DATA  . . . . . . . . . . .      000A  WORD  PUBLIC     'DATA'
  STACK  . . . . . . . . . . .      0100  PARA  STACK'STACK'
_TEXT  . . . . . . . . . . . .      0067  WORD  PUBLIC     'CODE'

Symbols:

                N a m e         Type   Value      Attr

A  . . . . . . . . . . . . . .      L WORD     0002 _DATA
ABS  . . . . . . . . . . . . .      L NEAR     0042 _TEXT

B  . . . . . . . . . . . . . .      L WORD     0004 _DATA

FINAL  . . . . . . . . . . . .      L NEAR     0037 _TEXT
FIRST  . . . . . . . . . . . .      L NEAR     000E _TEXT
```

```
I  . . . . . .  L WORD    0000 _DATA

K  . . . . . .  L WORD    0006 _DATA

MAX  . . . . . .  L NEAR    004E _TEXT
MIN  . . . . . .  L NEAR    0037 _TEXT

RES  . . . . . .  L WORD    0008 _DATA
RESULT . . . . . .  L NEAR    005F _TEXT
RESULT_MAX . . . . . .  L NEAR    005C _TEXT
RESULT_MIN . . . . . .  L NEAR    0056 _TEXT

SECOND . . . . . .  L NEAR    0026 _TEXT

@CODE  . . . . . .  TEXT  _TEXT
@CODESIZE  . . . . . .  TEXT  0
@CPU . . . . . .  TEXT  0101h
@DATASIZE  . . . . . .  TEXT  0
@FILENAME  . . . . . .  TEXT  LAB3
@VERSION . . . . . .  TEXT  510
```



```
     65 Source  Lines
     65 Total   Lines
     31 Symbols

  47952 + 459308 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```