**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.0382 _____ Диденко Д.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

**Цель работы.**

Изучить условные переходы и арифметические операции на ассемблере.

**Задание.**

Вариант 4.

Функции:

f1:

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

f2:

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

f3:

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

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.

**Основные теоретические положения.**

Есть 2 типа выполнения условий в ассемблере:

Прыжок без условия (или «безусловный прыжок») — выполняется инструкцией JMP. Выполнение данной инструкции часто включает в себя

передачу управления в адрес инструкции, которая не следует за выполняемой в настоящее время инструкцией. Результатом передачи управления может быть выполнение нового набора инструкций или повторное выполнение текущих инструкций.

Прыжок с условием (или «условный прыжок») — выполняется с помощью инструкций типа J<условие> и зависит от самого условия. Условные инструкции, изменяя значение смещения в регистре IP, передают управление, прерывая последовательный поток выполнения кода.

Инструкция CMP

Инструкция CMP (от англ. «COMPARE») сравнивает два операнда. Фактически, она выполняет операцию вычитания между двумя операндами для проверки того, равны ли эти операнды или нет. Используется вместе с инструкцией условного прыжка.

Синтаксис инструкции CMP:

*CMP <назначение>, <источник>.*

Инструкция CMP сравнивает два числовых поля. Операнд назначения может находиться либо в регистре, либо в памяти. Исходным операндом (источник) могут быть константы, регистры или память.

Прыжок без условия

Безусловный прыжок выполняется инструкцией JMP, которая включает в себя имя метки, куда следует перебросить точку выполнения программы:

JMP *<label>*

Прыжок с условием

Если при выполнении операции условного прыжка выполняется обозначенное условие, то точка выполнения программы переносится в указанную инструкцию. Существует множество инструкций условного прыжка, в зависимости от условия и данных.

Команды условного перехода, использующиеся после команд сравнения операндов со знаком представлены на рис.1.

Рис.1.

| *Мнемоника* | *Описание* | *Состояние флагов* |
|---|---|---|
| JG | Переход, если больше (*левоп* > *правоп*) | SF = OF и ZF = 0 |
| JNLE | Переход, если не меньше или равно (синоним команды JG) | SF = OF и ZF= 0 |
| JGE | Переход, если больше или равно (*левоп* >= *правоп*) | SF = 0 или ZF = 1 |
| JNL | Переход, если не меньше (синоним команды JGE)[4] | SF = 0 или ZF = 1 |
| JL | Переход, если меньше (*левоп* < *правоп*) | SF ≠ OF и ZF = 0 |
| JNGE | Переход, если не больше или равно (синоним команды JL) | SF ≠ OF и ZF = 0 |
| JLE | Переход, если меньше или равно (*левоп* <= *правоп*) | SF ≠ OF или ZF = 1 |
| JNG | Переход, если не больше (синоним команды JBE) | SF ≠ OF или ZF = 1 |

Команды условного перехода, использующиеся после команд сравнения беззнаковых операндов представлены на рис.2.

Рис.2.

| *Мнемоника* | *Описание* | *Состояние флагов* |
|---|---|---|
| JA | Переход, если выше[1] (*левоп* > *правоп*) | CF = 0 и ZF = 0 |
| JNBE | Переход, если не ниже или равно (синоним команды JA) | CF = 0 и ZF = 0 |
| JAE | Переход, если выше или равно (*левоп* >= *правоп*) | CF = 0 или ZF = 1 |
| JNB | Переход, если не ниже (синоним команды JAE)[2] | CF = 0 или ZF = 1 |
| JB | Переход, если ниже (*левоп* < *правоп*) | CF = 1 и ZF = 0 |
| JNAE | Переход, если не выше или равно (синоним команды JB) | CF = 1 и ZF = 0 |
| JBE | Переход, если ниже или равно (*левоп* <= *правоп*) | CF = 1 или ZF = 1 |
| JNA | Переход, если не выше (синоним команды JBE)[3] | CF = 1 или ZF = 1 |

**Выполнение работы.**

Результаты тестирования программы lab3.exe представлены в табл. 1.

Таблица 1 – Тестирование программы lab3.exe.

| № п/п | Входные данные | Вывод | Результат |
|---|---|---|---|
| 1. | a = 6<br>b = 5<br>i = 2<br>k = -1 | res = 1 | Программа работает верно |
| 2. | a = 6<br>b = 5<br>i = 2<br>k = 2 | res = -6 | Программа работает верно |
| 3. | a = 5<br>b = 5<br>i = 2 | res = 2 | Программа работает верно |

| | | | |
|---|---|---|---|
| | k = -1 | | |
| 4. | a = 5<br>b = 5<br>i = 2<br>k = 2 | res = 6 | Программа работает верно |

**Выводы.**

Изучены условные переходы и арифметические операции на ассемблере.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

**Название файла:** lab3.asm

```
; Стек программы

AStack SEGMENT STACK
 DW 12 DUP(?)
AStack ENDS


DATA SEGMENT
a DW 0
b DW 0
i DW 0
k DW 0
i1 DW 0
i2 DW 0
res DW 0
tst DW 6
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

 ;Entering data
 mov a,5
 mov b,5
 mov i,2
 mov k,-1

 mov AX, i
 shl AX,1; = 2i
```

```
 mov BX, a
 cmp BX, b

 JLE f1_under

f1_over:
 mov i1, 15
 sub i1, AX; = 15 - 2i
 mov AX, i1

 shl AX,1; = 30 - 4i
 sub AX,10; = 20 - 4i
 mov i2,AX

 JMP f3_choice

f1_under:
 add AX, i; = 3i
 add AX, 4 ; = 3i + 4
 mov i1, AX

 shl AX,1
 sub AX,14
 neg AX
 mov i2,AX

f3_choice:
 mov AX, k
 cmp AX, 0
 JGE f3_over

f3_under:
 mov AX,i1
 sub AX,i2; = i1 - i2
 cmp AX, 0
 JGE positive
 NEG AX; abs AX
 positive:
 cmp AX, 2
 JB bigger
```

```
 mov res, 2
 ret

f3_over:
 mov AX, i2
 NEG AX; -i2
 cmp AX, -6
 JGE bigger
 mov res, -6
 ret

bigger:
 mov res, AX
 ret

Main ENDP
CODE ENDS
 END Main
```

**Название файла:** LAB3.LST

```
'Microsoft (R) Macro Assembler Version 5.10                    11/7/21
14:26:07
                                                               Page
1-1


                        ; Стек программы

 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  0000                    i DW 0
 0006  0000                    k DW 0
```

```
0008  0000                i1 DW 0
000A  0000                i2 DW 0
000C  0000                res DW 0
000E  0006                tst DW 6
0010                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

                     ;Entering data
0009  C7 06 0000 R 0005    mov a,5
000F  C7 06 0002 R 0005    mov b,5
0015  C7 06 0004 R 0002    mov i,2
001B  C7 06 0006 R FFFF    mov k,-1

0021  A1 0004 R      mov AX, i
0024  D1 E0                shl AX,1; = 2i
0026  8B 1E 0000 R         mov BX, a
002A  3B 1E 0002 R         cmp BX, b

002E  7E 18                JLE f1_under

0030                f1_over:
0030  C7 06 0008 R 000F    mov i1, 15
0036  29 06 0008 R         sub i1, AX; = 15 - 2i
003A  A1 0008 R      mov AX, i1

003D  D1 E0                shl AX,1; = 30 - 4i
003F  2D 000A              sub AX,10; = 20 - 4i
0042  A3 000A R      mov i2,AX

0045  EB 15 90             JMP f3_choice
```

```
0048                    f1_under:
0048  03 06 0004 R              add AX, i; = 3i
004C  05 0004                   add AX, 4 ; = 3i + 4
004F  A3 0008 R          mov i1, AX

0052  D1 E0                     shl AX,1
0054  2D 000E                   sub AX,14
0057  F7 D8                     neg AX
0059  A3 000A R          mov i2,AX

005C                    f3_choice:
005C  A1 0006 R          mov AX, k
005F  3D 0000                   cmp AX, 0
0062  7D 1A                     JGE f3_over

0064                    f3_under:
0064  A1 0008 R          mov AX,i1
0067  2B 06 000A R              sub AX,i2; = i1 - i2
006B  3D 0000                   cmp AX, 0
006E  7D 02                     JGE positive
0070  F7 D8                     NEG AX; abs AX
0072                     positive:
0072  3D 0002                   cmp AX, 2
0075  72 18                     JB bigger
0077  C7 06 000C R 0002         mov res, 2
007D  CB                  ret

007E                    f3_over:
007E  A1 000A R          mov AX, i2
0081  F7 D8                     NEG AX; -i2
0083  3D FFFA                   cmp AX, -6
0086  7D 07                     JGE bigger
0088  C7 06 000C R FFFA         mov res, -6
008E  CB                  ret
```

```
 008F                   bigger:
 008F  A3 000C R         mov res, AX
 0092  CB                ret

 0093                   Main ENDP
 0093                   CODE ENDS
                         END Main
'Microsoft (R) Macro Assembler Version 5.10                 11/7/21
14:26:07
                                                            Symbols-1


Segments and Groups:

                N a m e         Length     Align     Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                0093  PARA  NONE
DATA . . . . . .                0010  PARA  NONE

Symbols:

                N a m e         Type   Value      Attr

A  . . . . . .                  L WORD     0000  DATA

B  . . . . . .                  L WORD     0002  DATA
BIGGER . . . . . .              L NEAR     008F  CODE

F1_OVER  . . . . . .            L NEAR     0030  CODE
F1_UNDER . . . . . .            L NEAR     0048  CODE
F3_CHOICE  . . . . . .          L NEAR     005C  CODE
F3_OVER  . . . . . .            L NEAR     007E  CODE
F3_UNDER . . . . . .            L NEAR     0064  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    =
0093

POSITIVE . . . . . .    L NEAR     0072  CODE

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

TST  . . . . . .    L WORD     000E  DATA

@CPU . . . . . .    TEXT  0101h
@FILENAME  . . . . . .    TEXT  lab3
@VERSION . . . . . .    TEXT  510


     93 Source  Lines
     93 Total   Lines
     24 Symbols

  47978 + 461329 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```