**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.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Чернякова В.А. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2022

## Цель работы.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров вычисляет значения функций.

## Задание.

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

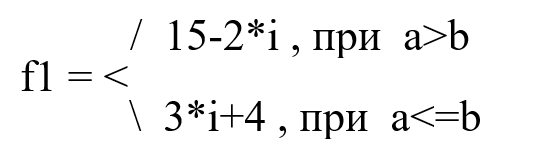
а) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

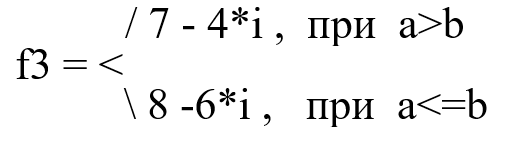
b) значения результирующей функции res = f3(i1, i2, k),

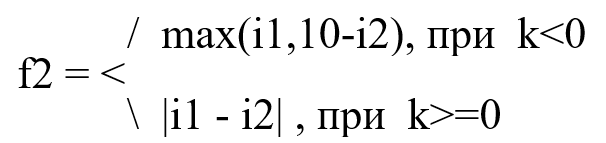
где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1, n2, n3), приведенным в табл.4.

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 2







## Ход работы.

Были реализованы функции из каталога заданий, соответствующие 2 варианту. Реализованная программа протранслирована с различными тестовыми данными.

Для выполнения данного задания были использованы такие команды общего назначения как:

Команды передачи данных.

1) mov – присваивание

Двоичные арифметические команды.

1) add - сложение

2) sub - вычитание

3) cmp – сравнение

4) neg – смена знака

Команды побитового сдвига.

1) sal - арифметический сдвиг влево команды передачи управления.

1) jmp – команда безусловного перехода

2) Int - вызов программного прерывания

3) jg(jump greater) - выполняет короткий переход, если первый операнд больше второго операнда при выполнении операции сравнения с помощью команды cmp.

4) jl(jump less) - выполняет короткий переход, если первый операнд меньше второго операнда при выполнении операции сравнения с помощью команды cmp.

Для реализации ветвления в программе использовались метки. Метка - это символьное имя, обозначающее ячейку памяти, которая содержит некоторую команду.

Код написанной программы, отвечающей требованиям лабораторной работы согласно варианту, представлена в приложении. Программа протестирована с различными значениями, результат выполнения тестирования записан в таблице 1.

Таблица 1 – результат тестирования программы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ теста** | **Тестируемый**  **случай** | **Функции для данного случая** | **Данные** | |
| **Входные** | **Выходные** |
| 1 | a > b  k < 0 | f1 = 15-2\*i  f3 = 7-4\*i  f2 = max(i1,i2) | a = 56  b = 1  k = -15  i = 1 | f1 = 13  f3 = 3  f2 = 13 |
| 2 | a > b  k > 0 | f1 = 15-2\*i  f3 = 7-4\*i  f2 = abs(i1-i2) | a = -10  b = -46  k = 56  i = 2 | f1 = 11  f3 = -1  f2 = 12 |
| 3 | a > b  k = 0 | f1 = 15-2\*i  f3 = 7-4\*i  f2 = abs(i1-i2) | a = 23  b = -9  k = 0  i = 3 | f1 = 9  f3 = -5  f2 = 14 |
| 4 | a <= b  k < 0 | f1 = 3\*i+4  f3 = 8-6\*i  f2 = max(i1,i2) | a = 45  b = 345  k = -10  i = 1 | f1 = 7  f3 = 2  f2 = 7 |
| 5 | a <= b  k > 0 | f1 = 3\*i+4  f3 = 8-6\*i  f2 = abs(i1-i2) | a = -7  b = 34  k = 7  i = 3 | f1 = 13  f3 = -10  f2 = 23 |
| 6 | a <= b  k = 0 | f1 = 3\*i+4  f3 = 8-6\*i  f2 = abs(i1-i2) | a = 34  b = 34  k = 0  i = 2 | f1 = 10  f3 = -4  f2 = 14 |

## Вывод.

В ходе выполнения лабораторной работы были получены навыки разработки программы с заданными целочисленными значениями на языке программирования Ассемблер

# ПРИЛОЖЕНИЕ.

**Lb3.ASM**

Исходный код программы.

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

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 ;f1

i2 DW 0 ;f3

res DW 0 ;f2

DATA ENDS

CODE SEGMENT

Main PROC **FAR**

**push** **DS**

**sub** **AX,AX**

**push** **AX**

**mov** **AX,** DATA

**mov** **DS,** **AX**

;вычисление f1 и f3

**mov** **AX,**a ; ax = a

**mov** **CX,**i ; cx = i

**cmp** **AX,**b ; сравнение двух операндов a и b

**jg** AGreaterB ; короткий переход, если первый операнд больше первого при выполнении операции cmp

;f1

**sal** **CX,**1 ; i<<1 = 2\*i

**add** **CX,**i ; i\*2+i = 3\*i

**mov** i1**,CX** ; запись результата функции f1

**add** i1**,**4 ; 3\*i+4

;f3

**sal** **CX,**1 ; i<<1 = 6\*i

**mov** i2**,**8 ; запись числа 8 в результат работы функции f3 для дальнейших вычислений

**sub** i2**,CX** ; 8-6\*i

**jmp** F2 ; выполнение безусловного перехода

AGreaterB**:**

;f1

**sal** **CX,**1 ; i<<1 = 2\*i

**mov** i1**,**15 ; запись числа 15 в результат работы функции f1 для дальнейших вычислений

**sub** i1**,CX** ; 15-2\*i

;f3

**sal** **CX,**1 ; i<<1 = 4\*i

**mov** i2**,**7 ; запись числа 7 в результат работы функции f3 для дальнейших вычислений

**sub** i2**,CX** ; 7-4\*i

;f2

F2**:**

**cmp** k**,**0 ; сравнение двух операндов k и 0

**jl** KLESSZERO ; короткий переход, если первый операнд меньше второго при выполнении операции cmp

;далее выполняется, если k >= 0

**mov** **AX,**i1 ; запись значения i1 для дальнейших вычислений

**sub** **AX,**i2 ; i1-i2

**mov** res**,AX** ; res = i1 - i2

**cmp** res**,**0 ; сравнение двух операндов res и 0

**jl** Module\_is\_negative ; короткий переход, если первый операнд меньше второго при выполнении операции cmp

**jmp** QUIT ; выполнение безусловного перехода

Module\_is\_negative**:**

;необходимо вывести модуль выражения i1-i2, знак которого "-" => смена знака

**neg** res ; изменение знака числа res

**jmp** QUIT ; выполнение безусловного перехода

KLESSZERO**:**

; выполняется, когда k < 0

**mov** **AX,**10 ; запись 10 для дальнейших вычислений

**sub** **AX,**i2 ; 10 - i2

**cmp** i1**,AX** ; сравнение i1 и 10 - i2

**jg** i1\_GREATER\_i2 ; короткий переход, если первый операнд больше первого при выполнении операции cmp

**mov** res**,AX** ; res = 10 -i2

**jmp** QUIT ; выполнение безусловного перехода

i1\_GREATER\_i2**:**

;при сравнении i1 и 10 - i2 первое оказалось больше

**mov** **AX,**i1 ; запись i1 для вывода результата

**mov** res**,AX** ; res = i1

QUIT**:**

**int** 20 ; завершение программы

Main ENDP

CODE ENDS

END Main