**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:

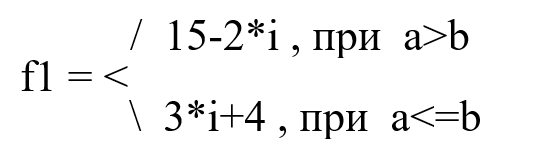
а) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

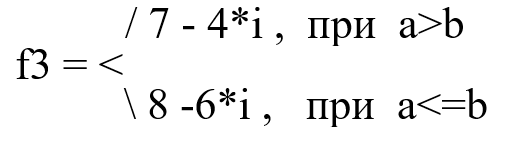
b) значения результирующей функции res = f3(i1, i2, k),

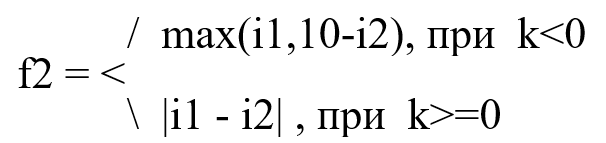
где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1, n2, n3), приведенным в табл.4.

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.

2) add - сложение

3) sub - вычитание

4) cmp – сравнение

5) neg – смена знака

Команды побитового сдвига.

1) sal - арифметический сдвиг влево команды передачи управления.

2) jmp – команда безусловного перехода

3) Int - вызов программного прерывания

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

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

Преобразования:

1. Перед ветвлением в регистры были записаны значения: i, 2\*i, 3\*i.
2. Выражение 8-6\*i можно получить из 3\*i+4:

2\*i+3\*i+4 = 5\*i+4 shl (3\*i +4) = 6\*i +8

5\*i+4+i = 6\*i+4 neg(6\*i +8) = -6\*i -8

-(6\*i+4) = -6\*i-4 add(-6\*i -8) +18 = 8-6\*i

-6\*i-4+12 = 8-6\*i

1. Выражение 7-4\*i можно получить из 15-2\*i:

-(2i) = -2i Улучшите сами: надо 2 шага

-2\*i+15-2\*i = 15-4\*i

15-4\*i-8 = 7-4\*i

С учетом этого исправьте программу.

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 45

b DW 345

i DW 1

k DW -10

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

mov CX,i ; cx = i

mov BX,CX

sal BX,1 ; bx = 2\*i

mov DX,BX

add DX,i add DX,CX; dx = 3\*I ; больше используйте регистры

mov AX,a ; ax = a

cmp AX,b ; сравнение a и b

jg AGreaterB ; a>b

;f1

add DX,4 ; dx = 3\*i+4

;f3

add BX,DX ; bx = 5\*i+4

add BX,CX ; bx = 6\*i+4

neg BX ; bx = -6\*i-4

add BX,12 ; bx = 8-6\*i

jmp F2 ; выполнение безусловного перехода

AGreaterB:

;f1

neg DX ; dx = -3\*i

add DX,15; dx = 15-3\*i

add DX,CX ; dx = 15-2\*i

;f3

neg BX ; bx = -2\*i

add BX,DX ; bx = 15-4\*i

sub BX,8 ; bx = 7-4\*i

;f2

F2:

mov i1,DX

mov i2,BX

mov AX,k

cmp AX,0

jl KLESSZERO ; короткий переход, если k < 0 при выполнении операции cmp

sub DX,BX ; dx = i1-i2

js Module\_is\_negative ; i1-i2<0

mov res,DX

jmp QUIT ; выполнение безусловного перехода

Module\_is\_negative:

neg DX

mov res,DX

Я же писал: вместо всего этого модуль вычисляется двумя командами: L: neg dx ; js L

Затем mov res, dx

jmp QUIT

KLESSZERO:

neg BX

add BX,10 ; bx = 10-i2

cmp DX,BX

jg i1\_GREATER\_i2 ; короткий переход, если i1>i2 при выполнении операции cmp

mov res,BX ; res = 10-i2

jmp QUIT ; выполнение безусловного перехода

i1\_GREATER\_i2:

mov res,DX

QUIT:

int 20 ; завершение программы

Main ENDP

CODE ENDS

END Main