**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

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

Тема «**Представление и обработка целых чисел. Организация ветвящихся процессов**»

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1303 |  | Сырцева Д.Д |
| Преподаватель |  | Ефремов М.А |

Санкт-Петербург

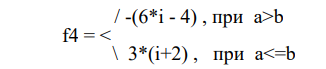
2022

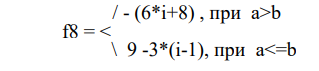
## Цель работы.

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.

f1:

f2:

f3:

## Выполнение работы. Протокол работы на компьютере.

**Ход работы:**

1. Получен вариант набора функций из таблиц 2 и 3, которые необходимо реализовать.
2. Реализованная программа протранлирована с различными исходными данными(рис1). Результат выполнения работы зафиксирован в таблице 1.
3. Используемые команды:

* MOV  Пересылка данных
* PUSH Засылка операнда в стек
* NEG Получение дополнительного кода (изменение знака)
* SUB Вычитание
* CMP Сравнение двух операндов
* ADD Сложение
* SAL Сдвиг влево (умножение на 2)
* JMP Команда безусловного перехода
* **JG** выполняет короткий переход, если первый операнд больше второго операнда при выполнении операции сравнения с помощью команды CMP
* JNE Переход если не равно
* JGE Переход если больше или равно
* JG Переход если больше
* INT Вызов программного прерывания

**Трансляция программы:**

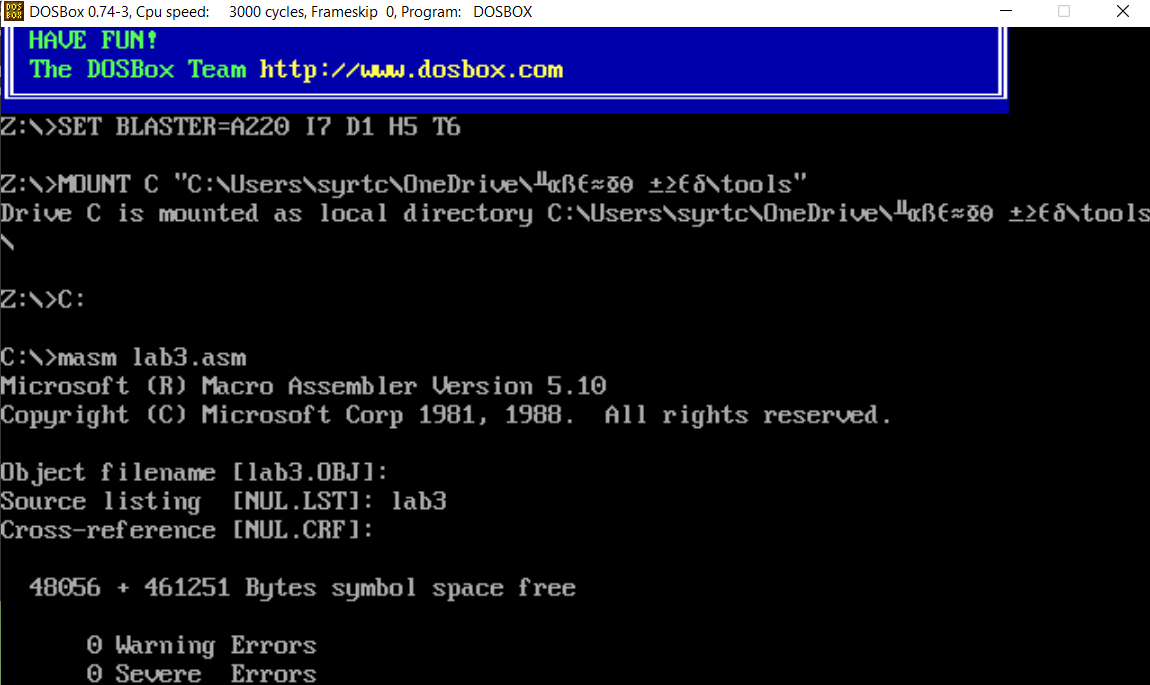


Рисунок1

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Тестируемый случай и используемые функции | Входные данные | Выходные данные |
| 1 | a<b, k=0  f1:  f2:  f3: | a= -5  b=5  i=0  k=0 | i1 = 0006  i2 = 000C  res = 0012 |
| 2 | a>b, k!=0  f1:  f2:  f3: | a=2  b=0  i=-5  k=5 | i1 = 0022  i2 = 0016  res = 0016 |
| 3 | a>b, k=0  f1:  f2:  f3: | a=8  b=1  i=7  k=0 | i1 = FFDA  i2 = FFCE  res = 0058 |

## Вывод.

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

## ПРИЛОЖЕНИЕ А

## Текст исходного файла программы lab3.

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

AStack SEGMENT STACK

DW 12 DUP(0)

AStack ENDS

DATA SEGMENT

i DW 0

a DW 0

b DW 0

k DW 0

i1 DW 0

i2 DW 0

res DW 0

DATA ENDS

CODE SEGMENT

Main PROC FAR

push ds

sub ax,ax

push ax

mov ax,DATA

mov ds,ax

mov ax,a ;а -> ах

mov cx,i ;i -> cx

cmp ax,b ;Сравнение значений a и b

jg STEP1 ;при a>b переход на STEP1

;if a<=b

add cx, i ;i+i = 2i

add cx, i ;2i+i = 3i

add cx,6 ;3i+6

mov i1,cx ;перемещаем результат в i1

neg cx ;-(3i+6) = -3i-6

add cx,18 ;-3i-6+18 = -3i+12

mov i2,cx ;перемещаем результат в i2

jmp STEP2 ;переход на STEP2

STEP1:

;if a>b

mov cx, i ;cx = i

add cx, i ;i+i = 2i

add cx, i ;2i+i = 3i

sal cx, 1 ;3i<<1 = 3i\*2 = 6i

sub cx, 4 ;6i-4

neg cx ;-(6i-4)

mov i1,cx ;перемещаем результат в i1

sub cx, 12 ;-6i+4-12 = -6i-8 = -(6i+8)

mov i2,cx ;перемещаем результат в i2

;Вычисление f3

STEP2:

mov ax,k

cmp ax,0 ;сравнение к и 0

JNe STEP3 ;если k не равно 0 то переход на STEP3

;к = 0

mov dx,i1 ;dx = i1

add dx,i2 ;dx = i1 + i2

cmp dx,0 ;сравнение i1+i2 и 0

JGe STEP5 ;если i1+ i2 >= 0 то перейти на STEP5

neg dx ;если i1 + i2 < 0 то меняем знак на противоположный

mov res,dx ;res = dx

jmp STEP6

STEP5:

mov res,dx

jmp STEP6

STEP3:

mov ax,i1 ;если k не равно 0

mov bx,i2

cmp ax,bx ;сравнение i1 и i2

JGe STEP4 ;если i1 >= i2 то перейти на STEP4

mov res,ax

jmp STEP6

STEP4:

mov res,bx ;если i1 >= i2, то в res перемещаем значение i2

STEP6:

int 20h

Main ENDP

CODE ENDS

END Main