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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Организация ЭВМ и систем»
Тема: Представление и обработка целых чисел. Организация ветвящихся процессов

Студент гр. 1303 _____ Карагезов С.Ю.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить условные переходы и арифметические операции на ассемблере.

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.

Ход работы.

Вариант 11

Функции:

f1: / -(4*i+3), при a>b

f2 = <

\ 6*i-10, при a<=b

f2: / 2*(i+1) -4, при a>b

f6 = <

\ 5 – 3*(i+1), при a<=b

f3: / min(i1, 6), при k = 0

f5 = <

\ |i1| + |i2|, при k /= 0

Выполнение работы.

1. Из таблицы получен вариант набора функций, которые необходимо реализовать, вариант представлен в разделе Задание.

2. Программа протранслирована с различными значениями переменных.

3. Программа выполнена в пошаговом режиме под управлением отладчика.

В сегменте данных заданы метки для переменных a, b, i, k, i1, i2, res, использующихся в программе. Их значения изначально равны нулю, при необходимости их можно поменять в коде.

Реализация ветвления осуществлена при помощи расстановок меток в исходном коде с условными и безусловными переходами по этим меткам.

Поведение ветвления зависит от значений a и b: изначально происходит сравнение этих двух переменных, исходя из результата которого происходит переход к соответствующей метке.

```
i a b k    i1 i2 res
1 2 3 4 -> -4 -1  5
1 3 2 4 -> -7 0   7
1 2 3 0 -> -4 -1  4
1 3 2 0 -> -7 0   6
```

Рис. 1 Проведены четыре теста с разными входными данными.

Вывод.

Изучены условные переходы и арифметические операции на ассемблере.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

```
assume ss:my_stack, cs:my_code, ds:my_data

my_stack segment stack
  dw 12 dup('?')
my_stack ends

my_data segment

  i dw 0
  a dw 0
  b dw 0
  k dw 0

  i1 dw 0
  i2 dw 0
  res dw 0

my_data ends

my_code segment

main proc far
  push ds
  xor ax, ax
  push ax
  mov ax, my_data
  mov ds, ax

; f1 & f2
  mov ax, a
  cmp ax, b
  jg greater

less_or_equal:
  mov ax, i
  sal ax, 1  ; 2 * i
  add ax, i  ; 3 * i
  push ax
  sub ax, 2  ; 3 * i - 2
  neg ax     ; 2 - 3 * i
  mov i2, ax

  pop ax     ; 3 * i
  sal ax, 1  ; 6 * i
  sub ax, 10 ; 6 * i - 10
  mov i1, ax
  jmp end_f1_f2

greater:
```

```
  mov ax, i
  sal ax, 1  ; 2 * i
  sub ax, 2  ; 2 * i - 2
  mov i2, ax

  add ax, 2  ; 2 * i
  sal ax, 1  ; 4 * i
  add ax, 3  ; 4 * i + 3
  neg ax     ; -(4 * i + 3)
  mov i1, ax

end_f1_f2:
; f3
  mov ax, k
  cmp ax, 0
  je equal_zero

not_equal_zero:
  mov ax, i1
  cmp ax, 0
  jge greater_zero_1
  neg ax
greater_zero_1:
  mov bx, i2
  cmp bx, 0
  jge greater_zero_2
  neg bx
greater_zero_2:
  add ax, bx
  jmp end_f3

equal_zero:
  mov ax, i1
  cmp ax, 0
  jge greater_zero_3
  neg ax
greater_zero_3:
  cmp ax, 6
  jle end_f3
  mov ax, 6

end_f3:
  mov res, ax

  mov ax, i1
  mov bx, i2
  mov cx, res

  ret
main endp

my_code ends

end main
```