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. 1303 _____ Ермакова В.М.

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:

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.

1.4.3

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

$$f4 = \begin{cases} -(6*i - 4), & \text{при } a>b \\ 3*(i+2), & \text{при } a<=b \end{cases}$$

$$f3 = \begin{cases} |i1 + i2|, & \text{при } k=0 \\ \min(i1,i2), & \text{при } k/=0 \end{cases}$$

Выполнение работы

1. Из таблицы получен вариант набора функций, которые необходимо реализовать, приведенного в каталоге Задания.

2. Программа протранслирована с различными значениями переменных, результат выполнения набора функций зафиксирован в таблице;

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) Inc – инкремент
4) Cmp – сравнение
5) Neg – смена знака

Команды побитового сдвига.

1) Sal - арифметический сдвиг влево

Команды передачи управления.

1) Jmp - безусловный переход
2) Int - вызов программного прерывания
3) Jge(jump greater equal) - выполняет короткий переход, если первый операнд больше второго операнда или равен ему при выполнении операции сравнения с помощью команды cmp
4) Jg(jump greater) - выполняет короткий переход, если первый операнд больше второго операнда при выполнении операции сравнения с помощью команды cmp.
5) Jne(jump negative equal) - выполняет короткий переход, если первый операнд не равен второму операнду при выполнении операции сравнения с помощью команды cmp.

Также были использованы метки (для примера B2), для перехода между некоторыми командами. Метка - это символьное имя, обозначающее ячейку памяти, которая содержит некоторую команду.

3. Программа выполнена в пошаговом режиме под управлением отладчика с фиксацией значений используемых переменных.

Таблица 1 – примеры тестовых случаев

№ теста	Тестируемый случай	Функции для данного случая	Данные	
			входные	выходные
1	a > b k = 0	f1 = 15 – 2*i f2 = -(6*i – 4) f3 = abs(f1 + f2)	a = 7, b = 3 k = 0 i = 2	f1 = 11 = 000B f2 = -8 = FFF8 f3 = 3 = 0003
2	a > b k != 0	f1 = 15 – 2*i f2 = -(6*i – 4) f3 = min(f1, f2)	a = 7, b = 3 k = 1 i = 3	f1 = 9 = 0009 f2 = -14 = FFF2 f3 = -14 = FFF2
3	a <= b k = 0	f1 = 3*i + 4 f2 = 3*(i + 2) f3 = abs(f1 + f2)	a = 5, b = 5 k = 0 i = 2	f1 = 10 = 000A f2 = 12 = 000C f3 = 22 = 0016
4	a <= b k != 0	f1 = 3*i + 4 f2 = 3*(i + 2) f3 = min(f1, f2)	a = 3, b = 5 k = 1 i = 3	f1 = 13 = 000D f2 = 15 = 000F f3 = 13 = 000D

Выводы

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: source.asm

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

AStack    SEGMENT  STACK
          DW 32 DUP(0)
AStack    ENDS

DATA      SEGMENT

i     DW    ?
a     DW    ?
b     DW    ?
k     DW    ?

i1    DW    ?          ;f1
i2    DW    ?          ;f4
res   DW    ?          ;f3

DATA      ENDS

CODE SEGMENT

Main      PROC  FAR
      mov   AX,DATA
      mov   DS,AX

;Вычисление f1 и f2
      mov ax,a   ;заносим значение a в ax
      mov cx,i   ;заносим i в cx
      cmp ax,b   ;Сравнение значений a и b
      jg A1      ;если a>b то на A1

                      ;если a<=b
      sal cx,1   ;умножение i на 2 cx = i*2
      add cx,i   ;cx = 2*i + i = 3*i
      mov ax,4   ;ax = 4
```

```
      add cx,ax  ;cx = 3*i + 4
      mov i1,cx  ;сохранение результата в f1

      add cx,ax  ;cx = 3*i + 4 + 2 = 3(i + 2)
      mov i2,cx  ;сохраняем рез-т в f2
      jmp A2           ;Пропускаем следующие шаги

A1:                    ;если a>b
      mov cx,i   ;восстановление значения i в cx
      sal cx,1   ;cx = i*2
      mov ax,15  ;ax = 15
      sub ax,cx  ;ax = ax - cx
      mov i1,ax  ;сохраняем результат в i1

      mov ax,cx  ;ax = 2*i
      sal cx,1   ;cx:=2*i*2
      add cx,ax  ;cx = 4*i + 2*i = 6*i
      mov ax,4   ;ax = 4
      sub ax,cx  ;ax = ax - cx = 4 - 6*i
      mov i2,ax  ;сохраняем результат в f2

;Вычисление f3
A2:
      mov ax,k

      cmp ax,0   ;сравниваем k и 0
      JNe B1           ;если k не равно 0 то перйти на B1

                       ;решение при к = 0
      mov dx,i1  ;dx = i1
      add dx,i2  ;dx = i1 + i2
      cmp dx,0
      JGe C1           ;если i1+ i2 >= 0 то перейти на C1

      neg dx           ;если i1 + i2 < 0 то меняем знак на
противоположный
      mov res,dx ;res = dx
      jmp B2
```

```
C1:
     mov res,dx
     jmp B2

B1:
                    ;если k не равно 0
     mov ax,i1
     mov bx,i2
     cmp ax,bx
     JGe C2         ;если i1 >= i2 то перейти на C2

     mov res,ax
     jmp B2

C2:
     mov res,bx ;если i1 >= i2

B2:
     int 20h

Main      ENDP
CODE      ENDS
          END Main
```