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: «Изучение организации ветвлений в программах на языке ассемблера.»

Студент гр. 2384 _____ Кузьминых Е.М.

Преподаватель _____ Морозов С.М.

Санкт-Петербург
2023

Цель работы.

Написать программу на языке Ассамблера, обрабатывающую значения, введенные пользователем, научиться работать с организацией ветвлений.

Задание.

Вариант №14

```
       / 7-4*i ,  при  a>b          /-(6*i-4), при  a>b
f3 = <                       f4 = <
       \ 8-6*i , при  a<=b          \ 3*(i+2), при  a<=b

                    / max(i1,10-i2), при  k<0
               f2 = <
                    \  |i1-i2|, при  k>=0
```

Разработать на языке Ассемблер iX86 программу, которая по заданным целым значениям a,b,i,k, размером 1 слово, вычисляет:

a) значения i1 = fn1(a,b,i) и i2 = fn2(a,b,i);

b) значения res= fn3(i1,i2,k), где вид функций fn1,fn2 определяется из табл. 1, а функции fn3 — из табл.2 по цифрам шифра индивидуального задания (n1.n2.n3).

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 все возможные комбинации параметров a,b и k, позволяющие проверить различные маршруты выполнения программы.

Выполнение работы.

В файле «lr3.asm» написана программа на языке ассемблера. Под сегмент стека выделено 12 слов. В сегменте данных создано 7 переменных, каждая из которых занимает 1 слово. В коде для реализации ветвления использованы: «cmp arg1, arg2» для сравнения, jle, jge и jmp для перемещения IP между строками.

Для минимизации количества операторов в коде, значения i1 и i2 вычислялись следующими способами:

1. Изначально в регистр dx было сохранено значение (-4i)
2. В случае, если (a>b), то значение переменной dx загружается в регистр ax с помощью инструкции mov ax, dx. Таким образом, ax становится равным -4i. К значению в регистре ax прибавляется 7 с помощью инструкции add ax, 7. Значение в регистре ax сохраняется в переменной i1 с помощью инструкции mov i1, ax. Теперь i1 равно -4i + 7. Значение переменной dx снова загружается в регистр ax с помощью инструкции mov ax, dx. Из значения в регистре ax вычитается значение переменной i с помощью инструкции sub ax, i, дважды. К значению в регистре ax прибавляется 4 с помощью инструкции add ax, 4. Теперь ax равно 4 - 6i. Значение в регистре ax сохраняется в переменной i2 с помощью инструкции mov i2, ax. Происходит переход к метке F1END с помощью инструкции jmp F1END.
3. Значение переменной dx загружается в регистр cx с помощью инструкции mov cx, dx. Таким образом, cx становится равным -4i. Значение в регистре cx инвертируется с помощью инструкции neg cx. Из значения в регистре cx вычитается значение переменной i с помощью инструкции sub cx, i. К значению в регистре cx прибавляется 6 с помощью инструкции add cx, 6. Значение в регистре cx сохраняется в переменной dx с помощью инструкции mov dx, cx. Значение в регистре cx также сохраняется в переменной i2 с помощью инструкции mov i2, cx. Значение в регистре dx инвертируется с помощью инструкции neg dx. Из значения в регистре dx вычитается значение переменной i с помощью инструкции sub dx, i, дважды. Из значения в регистре dx снова вычитается значение переменной i с помощью инструкции sub dx, i. К значению в регистре dx прибавляется 14 с помощью инструкции add dx,

14. Теперь dx равно -6i + 8.

4. Значение в регистре dx сохраняется в переменной i1 с помощью инструкции mov i1, dx. Теперь i1 равно -6i + 8.Сначала происходит загрузка значения переменной i1 в регистр ax с помощью инструкции mov ax, i1.Затем значение переменной i2 загружается в регистр bx с помощью инструкции mov bx, i2. Далее происходит сравнение значения переменной k с нулем с помощью инструкции cmp k, 0. Если значение k меньше или равно нулю (k <= 0), то происходит переход к метке F3E с помощью инструкции jge F3E. В метке F3E значение переменной bx инвертируется с помощью инструкции neg bx, затем к нему прибавляется 10 с помощью инструкции add bx, 10. Затем происходит сравнение значений переменных ax и bx с помощью инструкции cmp ax, bx. Если значение ax больше или равно bx (ax >= bx), то происходит переход к метке I1GE с помощью инструкции jge I1GE. В метке I1GE значение переменной ax копируется в переменную res с помощью инструкции mov res, ax. Если значение ax меньше bx (ax < bx), то происходит переход к метке I2MOD с помощью инструкции jmp I2MOD. В метке I2MOD значение переменной ax инвертируется с помощью инструкции neg ax, затем проверяется знак с помощью инструкции js I2MOD. Если значение ax отрицательно (знаковый флаг установлен), то происходит переход к метке I2MOD. В метке I2MOD значение переменной ax копируется в переменную res с помощью инструкции mov res, ax..

   В результате в DosBox была протранслирована программа, значения для работы передавались во время отладки, во время нажатия клавиши f8.

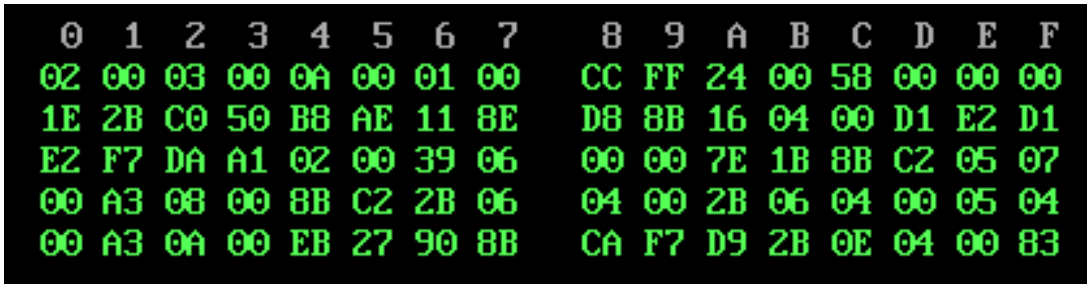

Рисунок 1 – введенные данные и результат работы программы

Тестирование.

a	b	i	k	I1	I2	res
0002	0003	000A	0001	FFCC	0024	58 (88 в десятичной)
0005	0006	000D	0001	FFBA	2D	73(115 в десятеричной)
0002	0003	000A	0000	FFCC	0024	58 (88 в десятичной)
0008	0008	000B	FFFF	FFC6	0027	FFE3 (-27 в десятичной)
0002	0003	000A	FFFF	FFCC	0024	FFE6 (-24 в десятичной)

Выводы.

В результате выполнения лабораторной работы была изучена организация ветвлений, была написана программа, обрабатывающая введенные значения и вычисляющая конечный результат в зависимости от этих значений.

ПРИЛОЖЕНИЕ А

Файл lb3.asm

```
AStack SEGMENT STACK
      DW 12 DUP(?)
AStack ENDS

DATA SEGMENT
      a DW 0
      b DW 0
      i DW 0
      k DW 0
      i1 DW ?
      i2 DW ?
      res DW ?
DATA ENDS


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

Main PROC FAR
      push DS
      sub AX, AX
      push AX
      mov AX, DATA
      mov DS, AX

      mov dx, i
      shl dx, 1
      shl dx, 1
      neg dx ; dx = -4i


      mov ax, b
      cmp a, ax
      jle F1LE

      ; случай a > b
      mov ax, dx ; ax = -4i
      add ax, 7
      mov i1, ax ;i1 = -4i + 7
      mov ax, dx ; ax = -4i
      sub ax, i
      sub ax, i
      add ax, 4 ; ax = 4 - 6i
      mov i2, ax ; сохраняем i2
      jmp F1END

F1LE: ; a <= b
```

```
      mov cx, dx ; cx = -4i
      neg cx;
      sub cx,i;
      add cx, 6; cx = 3i+6
      mov dx, cx
      mov i2, cx
      neg dx ; dx = -3i - 6
      sub dx, i
      sub dx, i
      sub dx, i
      add dx, 14; dx = -6i + 8
      mov i1, dx

F1END:
      mov ax, i1
      mov bx, i2
      cmp k, 0 ; если k <= 0
      jge F3E
      neg bx
      add bx,10
      cmp ax, bx
      jge I1GE
      mov res, bx
      jmp FIN

I1GE:
      mov res, ax
      jmp FIN

F3E:
      sub ax, bx
I2MOD:
      neg ax
      js I2MOD
      mov res, ax

FIN:
      ret
Main ENDP
CODE ENDS
      END Main
```