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»

Студент гр. 1303 _____ Кропотов Н.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров вычисляется 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.

$$f3 = \begin{cases} 7 - 4*i, & \text{при } a>b \\ 8 - 6*i, & \text{при } a<=b \end{cases} \qquad f5 = \begin{cases} 20 - 4*i, & \text{при } a>b \\ -(6*I - 6), & \text{при } a<=b \end{cases} \qquad f3 = \begin{cases} |i1 + i2|, & \text{при } k=0 \\ \min(i1,i2), & \text{при } k/=0 \end{cases}$$

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

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. Cmp – сравнение;
4. 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.

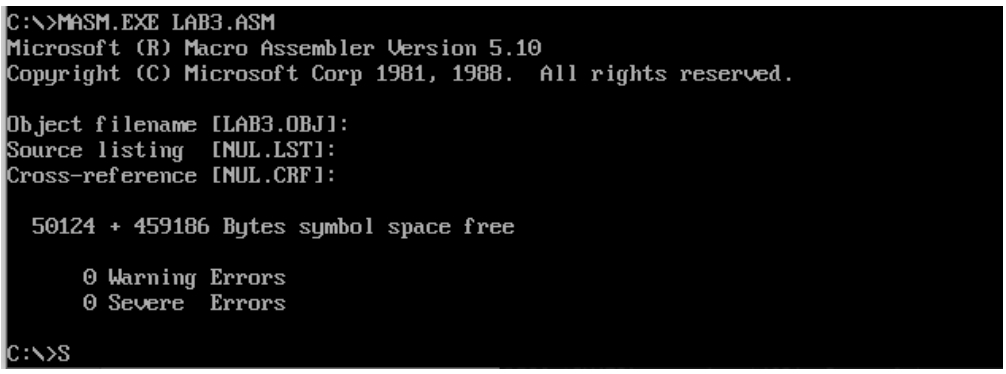

Рис. 1 – Трансляция программы с созданием файла диагностических сообщений

Вывод.

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

ПРИЛОЖЕНИЕ А

Тексты исходных файлов программ lab3.asm.

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

AStack    SEGMENT  STACK
          DW 32 DUP(0)
AStack    ENDS

DATA      SEGMENT

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

i1    DW    ?          ;f3
i2    DW    ?          ;f5
res   DW    ?          ;f3

DATA      ENDS

CODE SEGMENT

      ;f3 = 7-4i, a>b
      ;       8-6i, a<=b
      ;f5 = 20-4i, a>b
      ;       -(6i-6), a<=b
      ;f3 = |i1+i2|, k=0
      ;       min(i1,i2), k/=0

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 PART1   ;если a>b, то на PART1

                       ;если a<=b
      sal cx,1   ;умножение i на 2 => cx = i*2
      add cx,i   ;cx = 2*i + i = 3*i
      sal cx,1   ;умножение 3i на 2 => cx = i*6
      neg cx     ;cx = -6*i
      add cx,8   ;cx = -6*i + 8
      mov i1,cx  ;сохранение результата в i1

      sub cx,2   ;cx = -6*i + 8 - 2 = -6*i + 6
      mov i2,cx  ;сохраняем результат в i2
      jmp PART2  ;пропускаем следующие шаги
PART1:           ;если a>b
      mov cl, 2
      mov dx,i   ;восстановление значения i в dx
      sal dx,cl  ;dx = i*4
      mov ax,7   ;ax = 7
      sub ax,dx  ;ax = ax - dx = 7 - 4i
      mov i1,ax  ;сохраняем результат в i1

      mov ax,20  ;ax = 20
      sub ax,dx  ;ax = ax - dx = 20 - 4*i
      mov i2,ax  ;сохраняем результат в i2
;Вычисление f3
PART2:
      mov ax,k
      cmp ax,0   ;сравниваем k и 0
      JNe PART4  ;если k не равно 0 то перйти на PART4

                 ;решение при к = 0
      mov dx,i1  ;dx = i1
      add dx,i2  ;dx = i1 + i2
      cmp dx,0
```

```
      JGe PART3  ;если i1+ i2 >= 0 то перейти на PART3

      neg dx     ;если i1 + i2 < 0 то меняем знак на противоположный
      mov res,dx ;res = dx
      jmp ENDPART
PART3:
      mov res,dx
      jmp ENDPART
PART4:
      mov ax,i1  ;если k не равно 0
      mov bx,i2
      cmp ax,bx
      JGe PART5  ;если i1 >= i2 то перейти на PART5

      mov res,ax
      jmp ENDPART

PART5:
      mov res,bx ;если i1 >= i2

ENDPART:
      int 20h

Main      ENDP
CODE      ENDS
          END Main
```