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. 1381 _____ Таргонский М. А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2022

Цель работы.

Изучить реализацию ветвления на языке Ассемблера и реализовать программу, содержащую ветвление.

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.

Выполнение работы.

В ходе выполнения лабораторной работы была написана программа на языке Ассемблера. Вначале мы создаем три сегмента: AStack – сегмент стека, DATA – сегмент данных и CODE – сегмент кода. В сегменте данных были объявлены переменные: a, b, i, k – заполненные целыми числами на выбор, а также res – переменная для хранения результатов вычислений.

Сравнение переменных a и b с помощью команды cmp. Если a > b переход к метке calc_1, иначе выполняются команды метки calc_2.

Далее происходит вычисление значений i1и i2 заданных функций. Умножение реализовано с помощью логического сдвига влево и сложения. Значение i1 сохраняется в регистре ax. Значение i2 сохраняется в регистре cx.

При k ≥ 0 сравниваются значения i2 и -6. Из этих чисел выбирается большее и сохраняется в регистре cx. Иначе вычисляются модуль разности i1 и i2 и после он сравнивается с 2. Из этих чисел выбирается меньшее и сохраняется в регистре cx. В переменную res сохраняется конечное значение.

Таблица 1. Тестирование работы программы

Значение констант	Ожидаемый результат	Полученный результат
a = 1 b = 2 i = 3 k = 4	i1 = -12 i2 = -7 f3 = 7	i1 = -12 i2 = -7 f3 = 7
a = 4 b = 2 i = 5 k = -3	i1 = 0 i2 = 8 f3 = 2	i1 = 0 i2 = 8 f3 = 2
a = 2 b = 4 i = 5 k = -3	i1 = -24 i2 = -13 f3 = 2	i1 = -24 i2 = -13 f3 = 2

Выводы.

В ходе выполнения лабораторной работы была написана программа на языке Ассемблера, содержащая ветвления. Реализованная программа была отлажена с разными входными значениями, полученный результат был

сравнен с ожидаемым и представлен в таблице 1. Ожидаемый и полученный результат во всех случаях совпадают.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

ФАЙЛ LAB3.ASM

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

DATA SEGMENT
        a DW 1
        b DW 2
        i DW 3
        k DW 4
        ; 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

start_calc:
```

```
        mov AX, i ; AX = i
        shl AX, 1 ; AX = 2i
        mov CX, b ; CX = b

        cmp a, CX

        jg calc_1 ; jump if a > b

calc_2: ; a <= b
    add AX, i ; AX = 3i
    mov CX, AX ; CX = 3i
        shl AX, 1 ; AX = 6i
    neg AX ; AX = -6i
        add AX, 6 ; AX = -6i+6
        ; mov i1, AX ; i1 = -6i+6

        neg CX        ; CX = -3i
        add CX, 2 ; CX = 2 - 3i
        ; mov i2, CX ; i2 = 2 - 3i

        jmp res_calc

calc_1: ; a > b
    mov CX, AX ; CX = 2i
        sub CX, 2 ; CX = 2i - 2
        ; mov i2, CX ; i2 = 2i - 2

    shl AX, 1 ; AX = 4i
    neg AX ; AX = -4i
```

```
    add AX, 20 ; AX = -4i + 20
        ; mov i1, AX ; i1 = -4i + 20

res_calc:
min:
        neg CX
        cmp k, 0
        jge max                 ;if(k>=0)
        add CX, AX;-i2+i1
        abs:
        neg CX
        js abs      ;|-i2+i1|
        cmp CX, 2
        jge     result_min ;if(|i2 - i1|>=2)
        jmp result

max:
        cmp CX, -6
        jle result_max ;if(-i2<=-6)
        jmp result

result_min:
        mov CX, 2
        jmp result

result_max: mov CX, -6
result:
        mov res, CX
```

```
MAIN ENDP
CODE ENDS

END Main
```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ ПРОГРАММЫ

ФАЙЛ LAB3.LST


```
0000                    AStack SEGMENT STACK
0000  000C[                    DW 12 DUP(?)
        ????
                    ]

0018                    AStack ENDS

0000                    DATA SEGMENT
0000  0001                  a DW 1
0002  0002                  b DW 2
0004  0003                  i DW 3
0006  0004                  k DW 4
                            ; i1 DW ?
                            ; i2 DW ?
0008  0000                  res DW ?
000A                    DATA ENDS

0000                    CODE SEGMENT
                            ASSUME CS:CODE, DS:DATA, SS:AStack

0000                    Main PROC FAR
0000  1E                     push DS
```

```
0001  2B C0                         sub AX, AX
0003  50                       push AX
0004  B8 ---- R                mov AX, DATA
0007  8E D8                         mov DS, AX

0009                        start_calc:
0009  A1 0004 R                 mov AX, i ; AX = i
000C  D1 E0                         shl AX, 1 ; AX = 2i
000E  8B 0E 0002 R                  mov CX, b ; CX = b

0012  39 0E 0000 R                  cmp a, CX

0016  7F 15                         jg calc_1 ; jump if a > b

0018                        calc_2: ; a <= b
0018  03 06 0004 R              add AX, i ; AX = 3i
001C  8B C8                     mov CX, AX ; CX = 3i
001E  D1 E0                         shl AX, 1 ; AX = 6i
0020  F7 D8                     neg AX ; AX = -6i
0022  05 0006                       add AX, 6 ; AX = -6i+6
                               ; mov i1, AX ; i1 = -6i+6

0025  F7 D9                         neg CX       ; CX = -3i
0027  83 C1 02                      add CX, 2 ; CX = 2 - 3i
                               ; mov i2, CX ; i2 = 2 - 3i

002A  EB 0D 90                      jmp res_calc

002D                        calc_1: ; a > b
```

```
002D  8B C8                     mov CX, AX ; CX = 2i
002F  83 E9 02                      sub CX, 2 ; CX = 2i - 2
                                ; mov i2, CX ; i2 = 2i - 2
#Microsoft (R) Macro Assembler Version 5.10              11/7/22 14:46:27
                                                         Page     1-2


0032  D1 E0                     shl AX, 1 ; AX = 4i
0034  F7 D8                     neg AX ; AX = -4i
0036  05 0014                   add AX, 20 ; AX = -4i + 20
                                ; mov i1, AX ; i1 = -4i + 20

0039                    res_calc:
0039                    min:
0039  F7 D9                          neg CX
003B  83 3E 0006 R 00                cmp k, 0
0040  7D 0E                          jge max               ;if(k>=0)
0042  03 C8                          add CX, AX;-i2+i1
0044                              abs:
0044  F7 D9                          neg CX
0046  78 FC                          js abs      ;|-i2+i1|
0048  83 F9 02                       cmp CX, 2
004B  7D 0B                          jge     result_min ;if(|i2 - i1|>=2)
004D  EB 12 90                       jmp result

0050                    max:
0050  83 F9 FA                       cmp CX, -6
0053  7E 09                          jle result_max ;if(-i2<=-6)
```

```
0055  EB 0A 90                  jmp result

0058                    result_min:
0058  B9 0002                   mov CX, 2
005B  EB 04 90                  jmp result

005E  B9 FFFA           result_max: mov CX, -6
0061                    result:
0061  89 0E 0008 R              mov res, CX

0065                    MAIN ENDP
0065                    CODE ENDS

                        END Main
#Microsoft (R) Macro Assembler Version 5.10          11/7/22 14:46:27
                                                     Symbols-1


Segments and Groups:

                N a m e         Length   AlignCombine Class

ASTACK . . . . . .  .           0018  PARA      STACK
CODE . . . . . .  .             0065  PARA      NONE
DATA . . . . . .  .             000A  PARA      NONE

Symbols:

                N a m e         Type   Value Attr
```

```
A . . . . . .  L WORD   0000  DATA
ABS . . . . . .  L NEAR   0044  CODE

B . . . . . .  L WORD   0002  DATA

CALC_1 . . . . . .  L NEAR   002D  CODE
CALC_2 . . . . . .  L NEAR   0018  CODE

I . . . . . .  L WORD   0004  DATA

K . . . . . .  L WORD   0006  DATA

MAIN . . . . . .  F PROC   0000  CODE      Length = 0065
MAX . . . . . .  L NEAR   0050  CODE
MIN . . . . . .  L NEAR   0039  CODE

RES . . . . . .  L WORD   0008  DATA
RESULT . . . . . .  L NEAR   0061  CODE
RESULT_MAX . . . . . .  L NEAR   005E  CODE
RESULT_MIN . . . . . .  L NEAR   0058  CODE
RES_CALC . . . . . .  L NEAR   0039  CODE

START_CALC . . . . . .  L NEAR   0009  CODE

@CPU . . . . . .  TEXT  0101h
@FILENAME . . . . . .  TEXT  LAB3
@VERSION . . . . . .  TEXT  510
```

87 Source Lines

87 Total   Lines

24 Symbols

47978 + 461329 Bytes symbol space free

0 Warning Errors

0 Severe Errors