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.0382 _____ Литягин С.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить представление и обработку целых чисел и организацию ветвящихся процессов.

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.

Вариант №11:

$$f2 = \begin{cases} -(4*i+3), & \text{при } a>b \\ 6*i-10, & \text{при } a<=b \end{cases} \qquad f6 = \begin{cases} 2*(i+1)-4, & \text{при } a>b \\ 5-3*(i+1), & \text{при } a<=b \end{cases} \qquad f5 = \begin{cases} \min(|i1|, 6), & \text{при } k=0 \\ |i1|+|i2|, & \text{при } k/=0 \end{cases}$$

Выполнение работы:

Создано три сегмента: AStack – сегмент стека, DATA – сегмент данных, CODE – сегмент кода. Используя директиву ASSUME, метки сегментов записаны в соответствующие регистры. В сегменте данных объявлены переменные a, b, I, k, i1, i2, res. В сегменте кода создана процедура Main, а также написаны инструкции для успешного завершения программы после операции ret.

Для выполнения работы использовались переходы во избежание использования процедур. Используемые переходы представлены в таблице 1.

Таблица 1 – Используемые переходы.

Переход	Описание
JMP	Безусловый переход к метке. Используется в функции f1 при a > b, чтобы избежать выполнения кода при a <= b; в функции f2 при a > b, чтобы избежать выполнения кода при a <= b; в функции f3 при k /= 0, чтобы избежать выполнения кода при k = 0, а также для возврата в f3 после взятия чисел по модулю
JLE	Условный переход. Используется в функциях f1и f2, чтобы при a <= b не выполнялась часть программы для a > b; в функции f3, для выбора в качестве ответа i1, если i1 <=6
JL	Условный переход. Используется в функции f3, чтобы брать i1 по модулю, если i1 < 0, или i2 по модулю, если i2 < 0
JE	Условный переход. Используется в функции f3, чтобы при k = 0 избежать выполнения кода при k /= 0

Исходный код программы см. в приложении А.

Тестирование:

Для проверки работоспособности были проведены тесты, представленные в таблице 2.

Таблица 2 – Тесты

Номер теста	a	b	i	k
1	5	-1	2	0
2	2	4	-3	0
3	2	4	-3	5

Результаты тестирования представлены в таблице 3.

Таблица 3 – Результаты тестирования

Номер теста	i1	i2	res	Оценка результата
1	000B (11)	0002 (2)	0006 (6)	Верно
2	001C (28)	000B (11)	0006 (6)	Верно
3	001C (28)	000B (11)	0027 (39)	Верно

Выводы.

В ходе работы были изучены представление и обработка целых чисел и организация ветвящихся процессов, а также разработана программа, производящая вычисления функций, согласно условиям.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.asm

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

DATA SEGMENT
a DW 2
b DW 4
i DW -3
k DW 5
i1 DW 0
i2 DW 0
res DW 0
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
f1:
      mov AX, i
      shl AX, 1 ; 2i
      shl AX, 1 ; 4i
      mov CX, a
      cmp CX, b
      jle f1_2 ; a<=b
f1_1:
      add AX, 3 ; 4i+3
      neg AX ; -(4i+3)
      jmp f1_res
f1_2:
      shl AX, 1 ; 8i
      mov BX, i
      shl BX, 1
      sub AX, BX ; 6i
      sub AX, 10 ; 6i - 10
f1_res:
      mov i1, AX
f2:
      mov AX, i
      add AX, 1 ; i+1
      mov CX, a
      cmp CX, b
      jle f2_2 ; a<=b
f2_1:
```

```
      shl AX, 1 ; 2*(i+1)
      sub AX, 4 ; 2*(i+1) - 4
      jmp f2_res
f2_2:
      mov BX, AX; (i+1)
      shl AX, 1 ; 2*(i+1)
      shl AX, 1 ; 4*(i+1)
      sub AX, BX ; 3*(i+1)
      neg AX ; -3*(i+1)
      add AX, 5 ; 5-3*(i+1)
f2_res:
      mov i2, AX
f3:
      mov CX, i2
      cmp CX, 0
      jl abs_i2 ; i2 < 0
      mov CX, i1
      cmp CX, 0
      jl abs_i1 ; i1 < 0
      mov CX, k
      cmp CX, 0
      je f3_1 ; k = 0
      jmp f3_2
abs_i1:
      mov AX, CX
      neg AX
      mov i1, AX ; |i1|
      jmp f3
abs_i2:
      mov AX, CX
      neg AX
      mov i2, AX ; |i2|
      jmp f3
f3_2:
      mov AX, i1
      add AX, i2 ; |i1|+|i2|
      jmp f3_end
f3_1:
      mov CX, i1
      sub CX, 6
      jle res1 ; i1 <= 6
      mov AX, 6 ; 6
      jmp f3_end
res1:
      mov AX, i1 ; i1
      jmp f3_end
f3_end:
      mov res, AX
      ret
Main ENDP
CODE ENDS
      END Main
```

ПРИЛОЖЕНИЕ Б
ЛИСТИНГИ

Название файла: lb2.lst

```
Microsoft (R) Macro Assembler Version 5.10              10/20/21 22:08:4
                                                            Page     1-1


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

0018                    AStack ENDS

0000                    DATA SEGMENT
0000  0002                    a DW 2
0002  0004                    b DW 4
0004  FFFD                    i DW -3
0006  0005                    k DW 5
0008  0000                    i1 DW 0
000A  0000                    i2 DW 0
000C  0000                    res DW 0
000E                    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                    f1:
0009  A1 0004 R               mov AX, i
000C  D1 E0                        shl AX, 1 ; 2i
000E  D1 E0                        shl AX, 1 ; 4i
0010  8B 0E 0000 R                mov CX, a
0014  3B 0E 0002 R                cmp CX, b
0018  7E 08                        jle f1_2 ; a<=b
001A                    f1_1:
001A  05 0003                      add AX, 3 ; 4i+3
001D  F7 D8                        neg AX ; -(4i+3)
001F  EB 0E 90                     jmp f1_res
0022                    f1_2:
0022  D1 E0                        shl AX, 1 ; 8i
0024  8B 1E 0004 R                mov BX, i
0028  D1 E3                        shl BX, 1
002A  2B C3                        sub AX, BX ; 6i
002C  2D 000A                      sub AX, 10 ; 6i - 10
002F                    f1_res:
002F  A3 0008 R               mov i1, AX
0032                    f2:
```

```
0032  A1 0004 R           mov AX, i
0035  05 0001                 add AX, 1 ; i+1
0038  8B 0E 0000 R           mov CX, a
003C  3B 0E 0002 R           cmp CX, b
0040  7E 08                  jle f2_2 ; a<=b
0042                  f2_1:
0042  D1 E0                  shl AX, 1 ; 2*(i+1)
0044  2D 0004                sub AX, 4 ; 2*(i+1) - 4
```



```
0047  EB 0E 90               jmp f2_res
004A                  f2_2:
004A  8B D8                  mov BX, AX; (i+1)
004C  D1 E0                  shl AX, 1 ; 2*(i+1)
004E  D1 E0                  shl AX, 1 ; 4*(i+1)
0050  2B C3                  sub AX, BX ; 3*(i+1)
0052  F7 D8                  neg AX ; -3*(i+1)
0054  05 0005                add AX, 5 ; 5-3*(i+1)
0057                  f2_res:
0057  A3 000A R           mov i2, AX
005A                  f3:
005A  8B 0E 000A R           mov CX, i2
005E  83 F9 00               cmp CX, 0
0061  7C 1E                  jl abs_i2 ; i2 < 0
0063  8B 0E 0008 R           mov CX, i1
0067  83 F9 00               cmp CX, 0
006A  7C 0C                  jl abs_i1 ; i1 < 0
006C  8B 0E 0006 R           mov CX, k
0070  83 F9 00               cmp CX, 0
0073  74 1F                  je f3_1 ; k = 0
0075  EB 13 90               jmp f3_2
0078                  abs_i1:
0078  8B C1                  mov AX, CX
007A  F7 D8                  neg AX
007C  A3 0008 R           mov i1, AX ; |i1|
007F  EB D9                  jmp f3
0081                  abs_i2:
0081  8B C1                  mov AX, CX
0083  F7 D8                  neg AX
0085  A3 000A R           mov i2, AX ; |i2|
0088  EB D0                  jmp f3
008A                  f3_2:
008A  A1 0008 R           mov AX, i1
008D  03 06 000A R           add AX, i2 ; |i1|+|i2|
0091  EB 16 90               jmp f3_end
0094                  f3_1:
0094  8B 0E 0008 R           mov CX, i1
0098  83 E9 06               sub CX, 6
009B  7E 06                  jle res1 ; i1 <= 6
009D  B8 0006                mov AX, 6 ; 6
00A0  EB 07 90               jmp f3_end
00A3                  res1:
00A3  A1 0008 R           mov AX, i1 ; i1
00A6  EB 01 90               jmp f3_end
```

```
 00A9                   f3_end:
 00A9  A3 000C R              mov res, AX
 00AC  CB                     ret
 00AD                   Main ENDP
 00AD                   CODE ENDS
                              END Main

Microsoft (R) Macro Assembler Version 5.10              10/20/21 22:08:4

Symbols-1


      Segments and Groups:

                N a m e         Length     Align    Combine Class

      ASTACK . . . . . .        0018  PARA  STACK
      CODE . . . . . .          00AD  PARA  NONE
      DATA . . . . . .          000E  PARA  NONE

      Symbols:

                N a m e         Type   Value     Attr

      A  . . . . . .            L WORD     0000  DATA
      ABS_I1 . . . . . .        L NEAR     0078  CODE
      ABS_I2 . . . . . .        L NEAR     0081  CODE

      B  . . . . . .            L WORD     0002  DATA

      F1 . . . . . .            L NEAR     0009  CODE
      F1_1 . . . . . .          L NEAR     001A  CODE
      F1_2 . . . . . .          L NEAR     0022  CODE
      F1_RES . . . . . .        L NEAR     002F  CODE
      F2 . . . . . .            L NEAR     0032  CODE
      F2_1 . . . . . .          L NEAR     0042  CODE
      F2_2 . . . . . .          L NEAR     004A  CODE
      F2_RES . . . . . .        L NEAR     0057  CODE
      F3 . . . . . .            L NEAR     005A  CODE
      F3_1 . . . . . .          L NEAR     0094  CODE
      F3_2 . . . . . .          L NEAR     008A  CODE
      F3_END . . . . . .        L NEAR     00A9  CODE

      I  . . . . . .            L WORD     0004  DATA
      I1 . . . . . .            L WORD     0008  DATA
      I2 . . . . . .            L WORD     000A  DATA

      K  . . . . . .            L WORD     0006  DATA

      MAIN . . . . . .          F PROC     0000  CODE  Length =
00AD

      RES  . . . . . .          L WORD     000C  DATA
      RES1 . . . . . .          L NEAR     00A3  CODE

      @CPU . . . . . .          TEXT  0101h
```

```
     @FILENAME  . . . . . .     TEXT  lb3
     @VERSION . . . . . .       TEXT  510

□Microsoft (R) Macro Assembler Version 5.10              10/20/21 22:08:4


Symbols-2

          101 Source  Lines
          101 Total   Lines
           31 Symbols

        48030 + 459230 Bytes symbol space free

            0 Warning Errors
            0 Severe  Errors
```