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.

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить представление и обработку целых чисел, 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.

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

2) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

Вариант 2 — (1,3,2):

$$i1:\ f1 = \begin{cases} 15-2*i, & \text{при } a>b \\ 3*i+4, & \text{при } a<=b \end{cases}$$

i2: $f3 = \begin{cases} 7 - 4*i, & \text{при } a>b \\ 8 - 6*i, & \text{при } a<=b \end{cases}$

res: $f2 = \begin{cases} \max(i1, 10-i2), & \text{при } k<0 \\ |i1 - i2|, & \text{при } k>=0 \end{cases}$

Выполнение работы.

AStack — сегмент стека;

DATA — сегмент данных:

Объявлены переменные: *a, b, i, k,* с заданными вручную значениями. Так же объявлены: *i1, i2, res* (=0) — для хранения значений соответствующих функций.

CODE — сегмент кода:

Все необходимые операции прописаны в процедуре *Main.*

F_12:

Реализовано ветвление, зависящее от значений a, b. Сравнение происходит посредством *cmp,* с помощью команды (условного оператора) *jle* (<=) реализован переход к соответствующему случаю функций f1 и f2.

then_12:

Вычисляется значения *i1, i2* в случае: a>b. После выполнения всех арифметических операций посредством безусловного оператора *jmp* реализован переход к командам результирующей функции.

else_12:

Вычисляется значения *i1, i2* в случае: a<=b.

F_3:

Ветвление, зависящее от значения *k*. Реализовано посредством команды *jnl.* (>=).

then_3:

Вычисление res при k<0. При реализации также задействовано ветвление

при помощи *cmp* и *jnl*. (max_i1 – вспомогательный набор инструкций, при определении максимума).

else_3:

Вычисление res при k>=0. (pos – вспомогательный набор инструкций, в случае положительной разности под модулем)

Каждый рассмотренный случай результирующей функции заканчивается переходом в *stop*, для корректного окончания работы программы (при помощи команды *jmp).*

Исходный код программы см. в приложении А.

Файл листинга см. в приложении В.

Тестирование.

Тестирование реализовано в режиме отладчика.

Результаты тестирования представлены в таблице 1.

Таблица 1 – результаты тестирования.

№	Входные данные	Выходные данные	Комментарии
1	a = 5 b = 3 i = 1 k = 1	i1 = 13 i2 = 3 res = 10	Программа работает верно.
2	a = 3 b = 5 i = -2 k = 1	i1 = -2 i2 = 20 res = 22	Программа работает верно.
3	a = 8 b = 0	i1 = 19 i2 = 15	Программа работает верно.

	i = -2 k = -1	res = 19	
4	a = 0 b = 8 i = 2 k = -1	i1 = 10 i2 = -4 res = 14	Программа работает верно.
5	a = 7 b = 7 i = 0 k = 0	i1 = 4 i2 = 8 res = 4	Программа работает верно.

Выводы.

В ходе работы были изучены представление и обработка целых чисел, а также организация ветвящихся процессов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: 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 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
F_12:
     mov ax, i
     shl ax, 1   ;ax=2i
     mov bx, a
     cmp bx, b
     jle else_12

then_12:
     mov i1, 15
     sub i1, ax  ;15-2i

     mov i2, 7
     shl ax, 1   ;ax=4i
     sub i2, ax  ;7-4i
     jmp F_3

else_12:
     mov i1, 4
     add ax, i  ;ax=3i
     add i1, ax ;4+3i

     mov i2, 8
     shl ax, 1  ;ax=6i
```

```
     sub i2, ax ;8-6i

F_3:
     mov cx, k
     cmp cx, 0h
     jnl else_3

then_3:
     mov ax, i1  ;ax=i1
     mov bx, 10
     sub bx, i2  ;bx=10-i2
     cmp ax, bx
     jnl max_i1
     mov res, bx
     jmp stop

max_i1:
     mov res, ax
     jmp stop

else_3:
     mov ax, i1
     mov bx, i2
     sub ax, bx
     cmp ax, 0h
     jnl pos
     sub res, ax
     jmp stop

pos:
     add res, ax
     jmp stop

stop:
     ret

Main ENDP
CODE ENDS
     END Main
```

ПРИЛОЖЕНИЕ В

ФАЙЛ ЛИСТИНГА

Название файла: lb3.lst

```
Microsoft (R) Macro Assembler Version 5.10                  11/3/21
20:28:28
                                                             Page     1-1


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

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  0000                        a   DW 0
 0002  0000                        b   DW 0
 0004  0000                        i   DW 0
 0006  0000                        k   DW 0
 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                   F_12:
 0009  A1 0004 R                mov ax, i
 000C  D1 E0                         shl ax, 1   ;ax=2i
 000E  8B 1E 0000 R                  mov bx, a
 0012  3B 1E 0002 R                  cmp bx, b
 0016  7E 19                         jle else_12

 0018                   then_12:
 0018  C7 06 0008 R 000F             mov i1, 15
 001E  29 06 0008 R                  sub i1, ax  ;15-2i

 0022  C7 06 000A R 0007             mov i2, 7
 0028  D1 E0                         shl ax, 1   ;ax=4i
 002A  29 06 000A R                  sub i2, ax  ;7-4i
 002E  EB 1B 90                      jmp F_3

 0031                   else_12:
```

```
 0031  C7 06 0008 R 0004        mov i1, 4
 0037  03 06 0004 R             add ax, i  ;ax=3i
 003B  01 06 0008 R             add i1, ax ;4+3i

 003F  C7 06 000A R 0008        mov i2, 8
 0045  D1 E0                    shl ax, 1  ;ax=6i
 0047  29 06 000A R             sub i2, ax ;8-6i

 004B                F_3:
 004B  8B 0E 0006 R             mov cx, k
```



```
 004F  83 F9 00                 cmp cx, 0h
 0052  7D 1B                    jnl else_3

 0054                then_3:
 0054  A1 0008 R           mov ax, i1  ;ax=i1

 0057  BB 000A                  mov bx, 10
 005A  2B 1E 000A R             sub bx, i2  ;bx=10-i2
 005E  3B C3                    cmp ax, bx
 0060  7D 07                    jnl max_i1
 0062  89 1E 000C R             mov res, bx
 0066  EB 23 90                 jmp stop

 0069                max_i1:
 0069  A3 000C R           mov res, ax
 006C  EB 1D 90                 jmp stop

 006F                else_3:
 006F  A1 0008 R           mov ax, i1
 0072  8B 1E 000A R              mov bx, i2
 0076  2B C3                     sub ax, bx
 0078  3D 0000                   cmp ax, 0h
 007B  7D 07                     jnl pos
 007D  29 06 000C R              sub res, ax
 0081  EB 08 90                  jmp stop

 0084                pos:
 0084  01 06 000C R              add res, ax
 0088  EB 01 90                  jmp stop

 008B                stop:
 008B  CB                        ret


 008C                Main ENDP
 008C                CODE ENDS
                          END Main
```

```
Segments and Groups:

                N a m e         Length      Align     Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                008C  PARA  NONE
DATA . . . . . .                000E  PARA  NONE

Symbols:

                N a m e         Type   Value      Attr

A  . . . . . .                  L WORD     0000  DATA

B  . . . . . .                  L WORD     0002  DATA

ELSE_12  . . . . . .            L NEAR     0031  CODE
ELSE_3 . . . . . .              L NEAR     006F  CODE

F_12 . . . . . .                L NEAR     0009  CODE
F_3  . . . . . .                L NEAR     004B  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 = 008C
MAX_I1 . . . . . .              L NEAR     0069  CODE

POS  . . . . . .                L NEAR     0084  CODE

RES  . . . . . .                L WORD     000C  DATA

STOP . . . . . .                L NEAR     008B  CODE

THEN_12  . . . . . .            L NEAR     0018  CODE
THEN_3 . . . . . .              L NEAR     0054  CODE

@CPU . . . . . .                TEXT  0101h
@FILENAME  . . . . . .          TEXT  lb
@VERSION . . . . . .            TEXT  510

Microsoft (R) Macro Assembler Version 5.10                  11/3/21
20:28:28

                                                             Symbols-2


     87 Source  Lines
     87 Total   Lines
     25 Symbols

  48072 + 461235 Bytes symbol space free
```

```
0 Warning Errors
0 Severe  Errors
```