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		Луценко Д. А.
Преподаватель		Ефремов М. А.

Санкт-Петербург

2022

Цель работы.

Изучить представление и обработку целых чисел. Научиться организации ветвящихся процессов. Применить полученные знания на практике.

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);

Вариант 13:

```
        / - (4*i+3) , при a>b
f1 = <
        \ 6*i -10 , при a<=b

        / - (6*i+8) , при a>b
f2 = <
        \ 9 -3*(i-1), при a<=b

        / |i1 + i2|, при k=0
f3 = <
        \ min(i1,i2), при k/=0
```

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.

Выполнение работы.

Созданы три сегмента AStack, DATA, CODE - сегмент стека, сегмент кода и сегмент данных соответственно. В сегменте данных объявлены переменные a, b, i, k, i1, i2, res.

В сегменте кода находится процедура Main, в которой вычисляются значения данных в условии функций. Сначала в регистр ax записывается значение a. Далее с помощью функции cmp происходит сравнение значений переменных a и b. Команда jle проверяет условие a<=b, при еговыполнении производится переход по указанному адресу(к метке second). Там происходит расчёт значений i1 и i2. Расчёт делается при помощи shl(это логический сдвиг влево, что эквивалентно умножения на 2), так же используются sub(вычитания) и add(сложение) и ещё neg(меняет знак у числа). Если переход не был совершен, то начинается расчёт по метке first. В конце каждой ветке использована команда безусловного перехода(jmp) к метке final. В final мы при помощи сравнения(cmp) узнаём какой подсчёт результата нам нужен, если к = 0, то мы перейдём к метке res_1 и выполним

вычисления при этом если знак будет отрицательный это вызовет метку abs_1, и она поменяет знак. Если мы не перешли в res_1, то мы переходим в res_2 и в зависимости какое число меньше дальше вызываем либо min_i1, либо min_i2, которые записывают в ax либо i1, либо i2. И в конечном счете переходим к конечной метке ex, где в res записываем значение из ax.

Тестирование.

№	a	b	i	k	i1	i2	res
1	4	2	2	0	FFF5	FFEC	001F
2	4	2	-2	1	0005	0004	0004
3	1	2	-2	1	FFE6	12	FFE6

Выводы.

В ходе лабораторной работы были изучены представление и обработка целых чисел, организация ветвящихся процессов на языке Ассемблер.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.asm

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

DATA SEGMENT
      a DW 4
      b DW 2
      i DW 2
      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 ax, a
      cmp ax,b
      mov cx, i ;i
      shl cx, 1 ;2i
      shl cx, 1 ;4i
      jle second ;( a<=b)

first: ;(a > b)
      mov ax, cx
      add ax, 3 ;(4i + 3)
      neg ax
      mov i1, ax
      mov ax, cx
      add ax, i ;5i
      add ax, i ;6i
      add ax, 8 ;(6i + 8)
      neg ax
      mov i2, ax
      jmp final

second:
      mov ax, cx; 4i
```

```
      sub ax, i ; 3i
      sub ax, 12 ;3i - 12
      neg ax ;-3i + 12
      mov i2, ax
      neg ax; 3i - 12
      shl ax, 1; 6i - 24
      add ax, 14 ; 6i - 10
      mov i1, ax

      jmp final

abs_1:
      neg i1
      mov ax, i1
      jmp ex
final:
      cmp k,0
      jne res_2 ;k != 0
res_1: ;k = 0
      mov ax, i2
      add i1, ax
      cmp i1, -1
      jle abs_1 ;if(i1 <= -1)
      jmp ex
res_2:
      mov cx, i2
      cmp i1, cx
      jl min_i1
min_i2:
      mov ax, i2
      jmp ex
min_i1:
      mov ax, i1
ex:
      mov res, ax
      ret

Main ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ

Название файла: lb3.lst

```
__Microsoft (R) Macro Assembler Version 5.10                 11/7/22
21:28:19
                                                             Page     1-1


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

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  0004                        a DW 4
 0002  0002                        b DW 2
 0004  0002                        i DW 2
 0006  0000                        k DW 0
 0008  0000                        i1 DW ?
 000A  0000                        i2 DW ?
 000C  0000                        res DW ?
 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  A1 0000 R            mov ax, a
 000C  3B 06 0002 R              cmp ax,b
 0010  8B 0E 0004 R              mov cx, i ;i
 0014  D1 E1                     shl cx, 1 ;2i
 0016  D1 E1                     shl cx, 1 ;4i
 0018  7E 1F                     jle second ;( a<=b)

 001A                   first: ;(a > b)
 001A  8B C1                     mov ax, cx
 001C  05 0003                   add ax, 3 ;(4i + 3)
 001F  F7 D8                     neg ax
 0021  A3 0008 R            mov i1, ax
 0024  8B C1                     mov ax, cx
```

```
 0026  03 06 0004 R              add ax, i ;5i
 002A  03 06 0004 R              add ax, i ;6i
 002E  05 0008                   add ax, 8 ;(6i + 8)
 0031  F7 D8                     neg ax
 0033  A3 000A R           mov i2, ax
 0036  EB 26 90                  jmp final

 0039                second:
 0039  8B C1                     mov ax, cx; 4i
 003B  2B 06 0004 R              sub ax, i ; 3i
 003F  2D 000C                   sub ax, 12 ;3i - 12
 0042  F7 D8                     neg ax ;-3i + 12
 0044  A3 000A R           mov i2, ax
__Microsoft (R) Macro Assembler Version 5.10                    11/7/22
21:28:19
                                                                Page     1-2


 0047  F7 D8                     neg ax; 3i - 12
 0049  D1 E0                     shl ax, 1; 6i - 24
 004B  05 000E                   add ax, 14 ; 6i - 10
 004E  A3 0008 R           mov i1, ax

 0051  EB 0B 90                  jmp final

 0054                abs_1:
 0054  F7 1E 0008 R              neg i1
 0058  A1 0008 R           mov ax, i1
 005B  EB 2C 90                  jmp ex
 005E                final:
 005E  83 3E 0006 R 00                cmp k,0
 0063  75 11                     jne res_2 ;k != 0
 0065                res_1: ;k = 0
 0065  A1 000A R           mov ax, i2
 0068  01 06 0008 R              add i1, ax
 006C  83 3E 0008 R FF                cmp i1, -1
 0071  7E E1                     jle abs_1 ;if(i1 <= -1)
 0073  EB 14 90                  jmp ex
 0076                res_2:
 0076  8B 0E 000A R              mov cx, i2
 007A  39 0E 0008 R              cmp i1, cx
 007E  7C 06                     jl min_i1
 0080                min_i2:
 0080  A1 000A R           mov ax, i2
 0083  EB 04 90                  jmp ex
 0086                min_i1:
 0086  A1 0008 R           mov ax, i1
 0089                ex:
 0089  A3 000C R           mov res, ax
 008C  CB                  ret
```

```
 008D                   Main ENDP
 008D                   CODE ENDS
                        END Main
__Microsoft (R) Macro Assembler Version 5.10                 11/7/22
21:28:19
                                                            Symbols-1


Segments and Groups:

                N a m e         Length     Align     Combine Class

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

Symbols:

                N a m e         Type   Value      Attr

A  . . . . . .                  L WORD     0000  DATA
ABS_1  . . . . . .              L NEAR     0054  CODE

B  . . . . . .                  L WORD     0002  DATA

EX . . . . . .                  L NEAR     0089  CODE

FINAL  . . . . . .              L NEAR     005E  CODE
FIRST  . . . . . .              L NEAR     001A  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 = 008D
MIN_I1 . . . . . .              L NEAR     0086  CODE
MIN_I2 . . . . . .              L NEAR     0080  CODE

RES  . . . . . .                L WORD     000C  DATA
RES_1  . . . . . .              L NEAR     0065  CODE
RES_2  . . . . . .              L NEAR     0076  CODE

SECOND . . . . . .              L NEAR     0039  CODE

@CPU . . . . . .                TEXT  0101h
@FILENAME  . . . . . .          TEXT  lb3
@VERSION . . . . . .            TEXT  510
```

```
   87 Source  Lines
   87 Total   Lines
   25 Symbols

47992 + 461315 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```