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. 9382 _____ Герасев Г.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить арифметические команды ассемблера, разработать программу, вычисляющую необходимые переменные и углубить свои знания, в процессе написания программы.

Задание:

3 Вариант - 1.4.3

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.

/15-2*i , при a>b

f1 = <

\ 3*i+4 , при a<=b

/ -(6*i - 4) , при a>b

f4 = <

\ 3*(i+2) , при a<=b

/ |i1 + i2|, при k=0

f3 = <

\ min(i1,i2), при k/=0

Ход работы:

При разработке кода не было написано процедур для f1 и f2, были выбраны a = 4, b = 5, i = 3, k = 2, но также была рассмотрена программа с другими значениями этих переменных.

В ходе выполнения работы были использованы следующие команды арифметических операций:

Add – выполняет арифметическое сложение приемника и источника и помещает сумму в приемник.

Sub – вычитает источник из приемника и помещает разность в приемник.

Cmp - сравнивает приемник и источник и устанавливает флаги.

Neg - выполняет над числом, содержащимся в приемнике, операцию дополнения до двух.

В ходе выполнения работы были использованы следующие сдвиговые команды:

Sal – выполняет арифметический сдвиг влево.

В ходе выполнения работы были использованы следующие команды передачи ууправления:

Jg - переход, если больше (ZF = 0 и SF = OF).

Jz - переход, если 0 (ZF = 1).

Jl - переход, если меньше (SF <> OF).

Jmp - передает управление в другую точку программы.

Тестирование.

Вводные данные	Результат
a = 1 b = 1 i = 1	i1 = 7 (0007) i2 = 9 (0009) res = 7 (0007)

k = 1	
a = 5 b = 4 i = 3 k = 0	i1 = 9 (0009) i2 = -4 (FFF2) res = 5 (0005)
a = -5 b = -4 i = -3 k = -1	i1 = -5 (FFFB) i2 = -3 (FFFD) res = -5 (FFFB)

Выводы.

В результате выполнения лабораторной работы был разработан код для подсчитывания определенных выражений, оптимизировано умножение. Были улучшены навыки письма в ассемблере.

Приложение.

Текст файла MAIN.LST

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

0018                    AStack ENDS

0000                    DATA SEGMENT
0000  0001              a  DW 1
0002  0001              b  DW 1
0004  0001              i  DW 1
0006  0001              k  DW 1
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  8B 0E 0000 R       mov CX, DS:a
000D  8B 16 0002 R       mov DX, DS:b
0011  3B CA              cmp CX, DX
0013  7F 03              jg aGTb
0015  EB 25 90                  jmp aLEb

0018                     aGTb:
0018  A1 0004 R              mov AX, DS:i
001B  D1 E0                  sal AX, 1
001D  B9 000F                       mov CX, 15
0020  2B C8                  sub CX, AX
0022  89 0E 0008 R           mov DS:i1, CX

0026  A1 0004 R              mov AX, DS:i
0029  8B C8                  mov CX, AX
002B  D1 E1                  sal CX, 1
002D  03 C8                  add CX, AX
002F  D1 E1                  sal CX, 1

0031  B8 0004                       mov AX, 4
0034  2B C1                  sub AX, CX

0036  A3 000A R              mov DS:i2, AX
0039  EB 22 90                      jmp k_handle

003C                     aLEb:
003C  A1 0004 R                 mov AX, DS:i
003F  D1 E0                     sal AX, 1
```

```
0041  03 06 0004 R               add AX, DS:i
0045  05 0004                          add AX, 4
0048  A3 0008 R                  mov DS:i1, AX

004B  A1 0004 R                  mov AX, DS:i
004E  05 0002                          add AX, 2
0051  8B C8                      mov CX, AX
0053  D1 E0                      sal AX, 1
0055  03 C1                      add AX, CX
0057  A3 000A R                  mov DS:i2, AX
005A  EB 01 90                         jmp k_handle

005D                        k_handle:
005D  83 3E 0006 R 00                  cmp DS:k, 0
0062  74 03                    jz kEQzero
0064  EB 15 90                        jmp kNotEQzero

0067                        kEQzero:
0067  A1 0008 R                 mov AX, DS:i1
006A  03 06 000A R              add AX, DS:i2
006E  3D 0000                          cmp AX, 0
0071  7C 03                     jl neg_case
0073  EB 16 90                        jmp end_case

0076                        neg_case:
0076  F7 D8                     neg AX
0078  EB 11 90                        jmp end_case

007B                        kNotEQzero:
007B  A1 0008 R                    mov AX, DS:i1
007E  8B 0E 000A R                 mov CX, DS:i2
0082  3B C1                        cmp AX, CX
0084  7C 05                        jl end_case
0086  8B C8                        mov CX, AX
0088  EB 01 90                        jmp end_case

008B                        end_case:
008B  A3 000C R                 mov DS:res, AX
008E  CB                        ret

008F                       Main ENDP
008F                       CODE ENDS
                            END Main
```

```
#Microsoft (R) Macro Assembler Version 5.10                 11/14/20 04:27:4
                                                            Symbols-1


Segments and Groups:

                N a m e                 Length  Align Combine Class

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

Symbols:

                N a m e                 Type    Value  Attr

A  . . . . . . . . . . . . . . .        L WORD 0000   DATA
AGTB . . . . . . . . . . . . . .        L NEAR 0018   CODE
ALEB . . . . . . . . . . . . . .        L NEAR 003C   CODE

B  . . . . . . . . . . . . . . .        L WORD 0002   DATA

END_CASE . . . . . . . . . . . .        L NEAR 008B   CODE

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

K  . . . . . . . . . . . . . . .        L WORD 0006   DATA
KEQZERO  . . . . . . . . . . . .        L NEAR 0067   CODE
KNOTEQZERO . . . . . . . . . . .        L NEAR 007B   CODE
K_HANDLE . . . . . . . . . . . .        L NEAR 005D   CODE

MAIN . . . . . . . . . . . . . .        F PROC 0000   CODE   Length = 008F

NEG_CASE . . . . . . . . . . . .        L NEAR 0076   CODE

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

@CPU . . . . . . . . . . . . . .        TEXT  0101h
@FILENAME  . . . . . . . . . . .        TEXT  MAIN_1
@VERSION . . . . . . . . . . . .        TEXT  510


     94 Source  Lines
     94 Total   Lines
     23 Symbols

  47952 + 461355 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

Текст файла MAIN.ASM

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

DATA SEGMENT
a  DW -5
b  DW -4
i  DW -3
k  DW -1
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
 mov CX, DS:a
 mov DX, DS:b
 cmp CX, DX
 jg aGTb
 jmp aLEb

 aGTb:
     mov AX, DS:i
     sal AX, 1
     mov CX, 15
     sub CX, AX
     mov DS:i1, CX

     mov AX, DS:i
     mov CX, AX
     sal CX, 1
     add CX, AX
     sal CX, 1

     mov AX, 4
     sub AX, CX

     mov DS:i2, AX
     jmp k_handle

 aLEb:
        mov AX, DS:i
        sal AX, 1
        add AX, DS:i
        add AX, 4
        mov DS:i1, AX

        mov AX, DS:i
        add AX, 2
        mov CX, AX
        sal AX, 1
        add AX, CX
        mov DS:i2, AX
        jmp k_handle

 k_handle:
        cmp DS:k, 0
     jz kEQzero
     jmp kNotEQzero

 kEQzero:
     mov AX, DS:i1
     add AX, DS:i2
     cmp AX, 0
     jl neg_case
     jmp end_case

 neg_case:
```

```
    neg AX
    jmp end_case

 kNotEQzero:
       mov AX, DS:i1
       mov CX, DS:i2
       cmp AX, CX
       jl end_case
       mov CX, AX
       jmp end_case

 end_case:
    mov DS:res, AX
    ret

Main ENDP
CODE ENDS
 END Main
```