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 _____ Пя С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить арифметические команды ассемблера, разработать программу, вычисляющую необходимые переменные и углубить свои знания, пиша программу.

Задание:

15 Вариант - 3.5.3

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.

/ 7 - 4*i , при a>b

f3 = <

\ 8 -6*i , при a<=b

/ 20 - 4*i , при a>b

f5 = <

\ -(6*I - 6), 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 = 4 b = 5 i = 3 k = 2	i1 = -10 (FFF6) i2 = -12 (FFF4) res = -12 (FFF4)
a = 5 b = 4 i = 3 k = 0	i1 = -5 (FFFB) i2 = 8 (0008) res = (0003)
a = -5 b = -4 i = -3 k = -1	i1 = 26 (001A) i2 = 24 (0018) res = 24 (0018)

Выводы.

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

Приложение.

Текст файла LR.LST


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

0018                    AStack ENDS
                        ; Данные программы
0000                    DATA SEGMENT
0000  0004              a  DW 4
0002  0005              b  DW 5
0004  0003              i  DW 3
0006  0002              k  DW 2
0008  0000              i1  DW 0
000A  0000              i2 DW 0
000C  0000              res DW 0
000E                    DATA ENDS
                        ; Код программы
0000                    CODE SEGMENT
                         ASSUME CS:CODE, DS:DATA, SS:AStack
                        ; Головная процедура
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 4B              jg fst_case
0015  A1 0004 R          mov AX, DS:i
0018  D1 E0              sal AX, 1
001A  D1 E0              sal AX, 1
001C  03 06 0004 R       add AX, DS:i
0020  03 06 0004 R       add AX, DS:i
0024  B9 0008                   mov CX, 8
0027  2B C8              sub CX, AX
0029  89 0E 0008 R       mov DS:i1, CX
002D  2D 0006                   sub AX, 6
0030  F7 D8              neg AX
0032  A3 000A R          mov DS:i2, AX
0035  83 3E 0006 R 00           cmp DS:k, 0
003A  74 10              jz thrd_case
003C  8B 0E 0008 R       mov CX, DS:i1
0040  3B C8              cmp CX, AX
0042  7C 03              jl min_case
0044  EB 3A 90                 jmp end_case

0047                     min_case:
0047  8B C1                 mov AX, CX
0049  EB 35 90                     jmp end_case

004C                     thrd_case:
004C  A1 0008 R             mov AX, DS:i1
```

```
 004F  03 06 000A R             add AX, DS:i2
 0053  3D 0000                         cmp AX, 0
 0056  7C 03                    jl n_case
 0058  EB 26 90                         jmp end_case

 005B                         n_case:
 005B  F7 D8                    neg AX
 005D  EB 21 90                         jmp end_case

 0060                         fst_case:
 0060  A1 0004 R                mov AX, DS:i
 0063  D1 E0                    sal AX, 1
 0065  D1 E0                    sal AX, 1
 0067  B9 0007                         mov CX, 7
 006A  2B C8                    sub CX, AX
 006C  89 0E 0008 R             mov DS:i1, CX
 0070  B9 0014                         mov CX, 20
 0073  2B C8                    sub CX, AX
 0075  89 0E 000A R             mov DS:i2, CX
 0079  83 3E 0006 R 00                  cmp DS:k, 0
 007E  74 CC                    jz thrd_case

 0080                         end_case:
 0080  A3 000C R                mov DS:res, AX
 0083  CB                       ret
 0084                        Main ENDP
 0084                        CODE ENDS
                              END Main
__Microsoft (R) Macro Assembler Version 5.10                 10/28/20 21:50:0
                                                             Symbols-1


Segments and Groups:

                N a m e           Length  Align Combine Class

ASTACK . . . . . .              0018   PARA   STACK
CODE . . . . . .                0084   PARA   NONE
DATA . . . . . .                000E   PARA   NONE

Symbols:

                N a m e           Type    Value  Attr

A  . . . . . .                  L WORD 0000   DATA

B  . . . . . .                  L WORD 0002   DATA

END_CASE . . . . . .            L NEAR 0080   CODE

FST_CASE . . . . . .            L NEAR 0060   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 = 0084
MIN_CASE . . . . . .            L NEAR 0047   CODE

N_CASE . . . . . .              L NEAR 005B   CODE

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

```
THRD_CASE  . . . . . .          L NEAR 004C   CODE

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


     79 Source  Lines
     79 Total   Lines
     21 Symbols

  47992 + 461315 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

Текст файла LR.ASM

```
AStack SEGMENT STACK
 DW 12 DUP(?)
AStack ENDS
; Данные программы
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
; Код программы
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, SS:AStack
; Головная процедура
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 fst_case
 mov AX, DS:i
 sal AX, 1
 sal AX, 1
 add AX, DS:i
 add AX, DS:i
 mov CX, 8
 sub CX, AX
 mov DS:i1, CX
 sub AX, 6
 neg AX
 mov DS:i2, AX
 cmp DS:k, 0
 jz thrd_case
 mov CX, DS:i1
 cmp CX, AX
 jl min_case
 jmp end_case

 min_case:
     mov AX, CX
     jmp end_case
```

```
 thrd_case:
     mov AX, DS:i1
     add AX, DS:i2
     cmp AX, 0
     jl n_case
     jmp end_case

 n_case:
     neg AX
     jmp end_case

 fst_case:
     mov AX, DS:i
     sal AX, 1
     sal AX, 1
     mov CX, 7
     sub CX, AX
     mov DS:i1, CX
     mov CX, 20
     sub CX, AX
     mov DS:i2, CX
     cmp DS:k, 0
     jz thrd_case

 end_case:
     mov DS:res, AX
     ret
Main ENDP
CODE ENDS
 END Main
```