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. 0383 _____ Петровская Е.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучение представленя и работы с целыми числами, а также организации ветвящихся процессов на языке Ассемблера.

Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

- значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);
- 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 шифраиндивидуального 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 26:

```
      / 2 * (i + 1) - 4 ,if a > b
f1 = <
      \ 5 - 3 * (i + 1) ,if a <= b
```

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

```
      / min(i1, i2)     ,if k = 0
f3 = <
      \ max(i1, i2)     ,if k /= 0
```

Выполнение работы.

Считывание значений a, b, i, k во время отладки была использована внутренняя команда AFDPRO P {\W} adrr, value.

После инициализации программы начинается расчет f1 и f2, при написании которого для реализации умножения были использованы инструкции SHL и ADD, а для реализации ветвящихся процессов — CMP и ветвящиеся

переходы. После рассчета данных функций их значения сохраняются в i1 и i2 соответственно, и происходит расчет результирующей функции посредством тех же инструкций.

Таблица 1 – Результаты прогона программы hello1 под управлением отладчика

Исходные значения	Значение i1	Значение i2	Результирующая функция	Комметарий
a = 1 b = 1 i = 1 k = 1	FFFF = -1	9	9	Верно, т. к. a <= b => f1 = 5 - 3 - 3 = -1 f2 = 9 - 3 + 3 = 9 f3 = max(f1, f2) = 9
a = 1 b = 0 i = -1 k = 0	FFFC = -4	FFFE = -2	FFFC = -4	Верно, т. к. a > b => f1 = -2 + 2 - 4 = -4 f2 = 6 - 8 = -2 f3 = min(f1, f2) = -4
a = 10 b = -3 i = 0 k = 1	FFFE = -2	FFF8 = -8	FFFE = -2	Верно т. к. a > b => f1 = 2 - 4= -2 f2 = -8 f3 = max(f1, f2) = -2

Выводы.

В ходе выполнения лабораторной работы была изучена работа с целыми числами и условными переходами на языке Ассемблера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

```
     ;var____6,8,1
     ;Data input and processing through debugger
     ;Instead of multiply in f1 and f2 use ariphmetic slides and
possibly ADD
     ;No procedurs are to be used in f1 and f2
     ;Code size should be minimal - change original functons if needed

     ;integers a, b, i and k are typed in by user while debugging. Use
different variations
     ; find i1 = f1(a,b,i) and i2 = f2(a,b,i)
     ; find resulting function res = f3(i1, i2, k)

     ;       / 2 * (i + 1) - 4 ,if a > b        => 2*i -2
     ; f1 = <
     ;       \ 5 - 3 * (i + 1) ,if a <= b       => 2 - 3*i
     ;
     ;       / -(6 * i + 8)    ,if a > b        => - 8 - 6*i
     ; f2 = <
     ;       \ 9 - 3 * (i - 1) ,if a <= b       => 12 - 3*i
     ;
     ;     / min(i1, i2)     ,if k = 0
     ; f3 = <
     ;     \ max(i1, i2)     ,if (k = 0) = false

     ;               PROG:


     AStack SEGMENT STACK
          DW 12 DUP(?) ;DUP - declares an array of 12 unitialized(=?)
WORDS(DW=define words) = 2 bytes
                          ;thus DW 12 DUP(?) is a declaration of
(12*2=)24 bytes and they`re nor initialized
     AStack ENDS


     DATA SEGMENT
     a    DW   0
     b    DW   0
     i    DW   0
     k    DW   0
     i1   DW   0
     i2   DW   0
     DATA ENDS

     CODE SEGMENT
          ASSUME CS:CODE, DS:DATA ,SS:AStack

     Main PROC FAR
```

```
     PUSH DS              ;write ds into stack
     SUB AX, AX           ;write a 0
     PUSH AX              ;write  ax  into  stack   => Basically,  stack
initialization
     MOV AX, DATA              ;move adress of
     MOV DS, AX           ;DATA into DS

                    ;f1 & f2
                              ;multiply only by shifting so use SHL or
SHR
                              ;shift left 1 => x2
                              ;shift left 2 => x4
                              ;shift left 3 => x8
          MOV CX, i
          ADD CX, CX          ;in  f1  i  will  be  multiplied  by  2  or  3
either way
          MOV BX, b
          CMP a, BX           ;if a <= b jump to f1sec
          JLE f1sec
                              ;we go here if a > b
          MOV AX, CX          ;AX = 2i, f2: if a > b => - 8 - 6*i
          ADD CX, -2          ;CX = 2*i -2, f1 done
          ADD AX, i           ;AX = 3i
          SHL AX, 1           ;AX = 6i
          NEG AX
          ADD AX, -8          ;AX = - 8 - 6i, f2 done

          JMP f12save

          f1sec:                   ;f2 = 12 - 3*i
               ADD CX, i      ;now CX = 3i
               NEG CX              ;now CX = -3i
               MOV AX, CX     ;AX = -3i
               ADD AX, 12
               ADD CX, 2
               JMP f12save
          f12save:
               MOV i1, CX
               MOV i2, AX     ;save f1 & f2 results into i1 & i2

                    ;f3 and resulting function
          MOV CX, i1
          CMP CX, i2      ;check if i1 <= i2, if true -> jump
          JLE other
          MOV CX, i2 ;i2 is min in CX
          MOV AX, i1 ;i1 is max in AX
               other:
               ;MOV CX, i1
               ;MOV AX, i2

          MOV BX, k
          CMP BX, 0
          JZ fin     ;JumpZero check if ZF = 0, if k /= 0 continue here,
otherwise jump
          MOV CX, AX      ;get max into CX
```

```
        jmp fin    ; CX has min by default, if k == 0 then res = min
                   ;CX NOW HAS RESULT

        fin: RET        ;return to DOS
    Main  ENDP
    CODE ENDS
    END Main
```

Название файла: lab3.lst

```
    #Microsoft      (R)      Macro      Assembler      Version      5.10
11/25/21 08:41:0
                                                                    Page
1-1


                        ;var____6,8,1
                        ;Data input and processing through debugger
                        ;Instead   of   multiply   in   f1   and   f2   use
ariphmeti
                        c slides and possibly ADD
                        ;No procedurs are to be used in f1 and f2
                        ;Code size should be minimal - change original
                        functons if needed

                        ;integers a, b, i and k are typed in by user
wh
                        ile debugging. Use different variations
                        ; find i1 = f1(a,b,i) and i2 = f2(a,b,i)
                        ; find resulting function res = f3(i1, i2, k)

                        ;       / 2 * (i + 1) - 4 ,if a > b
                             => 2*i -2
                        ; f1 = <
                        ;       \ 5 - 3 * (i + 1) ,if a <= b
                             => 2 - 3*i
                        ;
                        ;       / -(6 * i + 8)    ,if a > b
                             => - 8 - 6*i
                        ; f2 = <
                        ;       \ 9 - 3 * (i - 1) ,if a <= b
```

```
                              => 12 - 3*i
                         ;
                         ;    / min(i1, i2)     ,if k = 0
                         ; f3 = <
                         ;    \ max(i1, i2)     ,if (k = 0) = false

                         ;               PROG:


      0000               AStack SEGMENT STACK
      0000  000C[                 DW  12  DUP(?)  ;DUP - declares an
array o
                         f 12 unitialized(=?) WORDS(DW=define words) =
2
                          bytes
            ????
                    ]

                                                 ;thus DW 12 DU
                         P(?) is a declaration of (12*2=)24 bytes and
th
                         ey`re nor initialized
      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               DATA  ENDS
```

```
0000                    CODE  SEGMENT
                              ASSUME CS:CODE, DS:DATA ,SS:AStack

0000                    Main  PROC FAR

0000  1E                PUSH DS                 ;write ds into
                        stack
0001  2B C0                   SUB AX, AX              ;write a 0
0003  50                PUSH AX                 ;write ax into
                        stack  => Basically, stack initialization
0004  B8 ---- R         MOV AX, DATA                ;move adress of
0007  8E D8                   MOV DS, AX              ;DATA into DS

                                              ;f1 & f2
                                                    ;multip
                        ly only by shifting so use SHL or SHR
                                                    ;shift
                        left 1 => x2
                                                    ;shift
                        left 2 => x4
                                                    ;shift
                        left 3 => x8
0009  8B 0E 0004 R                 MOV CX, i
000D  03 C9                        ADD CX, CX           ;in f1
                        i will be multiplied by 2 or 3 either way
000F  8B 1E 0002 R                 MOV BX, b
0013  39 1E 0000 R                 CMP a, BX            ;if a <
                        = b jump to f1sec
0017  7E 13                        JLE f1sec
                                                    ;we go
                        here if a > b
```

```
0019  8B C1                    MOV AX, CX          ;AX = 2
                    i, f2: if a > b => - 8 - 6*i
001B  83 C1 FE                 ADD CX, -2          ;CX = 2
                    *i -2, f1 done
001E  03 06 0004 R             ADD AX, i           ;AX = 3
                    i
0022  D1 E0                    SHL AX, 1           ;AX = 6
                    i
0024  F7 D8                    NEG AX
0026  05 FFF8                 ADD AX, -8          ;AX = -
                     8 - 6i, f2 done

0029  EB 12 90                 JMP f12save

002C                        f1sec:                 ;f2 = 1
                    2 - 3*i
002C  03 0E 0004 R                 ADD CX, i       ;now CX
                     = 3i
0030  F7 D9                        NEG CX              ;now CX
                     = -3i
0032  8B C1                        MOV AX, CX      ;AX = -
                    3i
0034  05 000C                      ADD AX, 12
```

```
0037  83 C1 02                    ADD CX, 2
003A  EB 01 90                    JMP f12save
003D                        f12save:
003D  89 0E 0008 R                MOV i1, CX
0041  A3 000A R             MOV i2, AX      ;save f
                        1 & f2 results into i1 & i2

                                  ;f3 and resulting funct
                        ion
0044  8B 0E 0008 R          MOV CX, i1
0048  3B 0E 000A R          CMP CX, i2      ;check if i1 <= i2,
if
                        true -> jump
004C  7E 07                       JLE other
004E  8B 0E 000A R          MOV CX, i2 ;i2 is min in CX
0052  A1 0008 R           MOV AX, i1 ;i1 is max in AX
0055                              other:
                                  ;MOV CX, i1
                                  ;MOV AX, i2

0055  8B 1E 0006 R          MOV BX, k
0059  83 FB 00               CMP BX, 0
005C  74 05                  JZ fin     ;JumpZero check if ZF =
0, if k
                         /= 0 continue here, otherwise jump
005E  8B C8                       MOV CX, AX      ;get max into CX

0060  EB 01 90               jmp fin    ; CX has min by default,
if k =
                        = 0 then res = min
                                  ;CX NOW HAS RESULT
```

```
0063  CB                fin: RET        ;return to DOS
0064                Main  ENDP
0064                CODE ENDS
                    END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0064	PARA	NONE	
DATA	000C	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
F12SAVE	L NEAR	003D	CODE
F1SEC	L NEAR	002C	CODE
FIN	L NEAR	0063	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 = 0064
OTHER	L NEAR	0055	CODE

```
@CPU . . . . . .     TEXT  0101h
@FILENAME  . . . . . .     TEXT  lab3
@VERSION . . . . . .     TEXT  510


    104 Source  Lines
    104 Total   Lines
     19 Symbols

  47966 + 459291 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```