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. 0383 _____ Зенин П.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Познакомиться с организацией ветвлений на языке Ассемблера.

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.

Вариант 3:

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

$$f2 = \begin{cases} -(6*i-4), & \text{при } a>b \\ 3(i+2), & \text{при } a<=b \end{cases}$$

/ |i1+i2|, при k=0

f3=<

\ min(i1, i2), при k/=0

Выполнение работы.

Была создана функция Read, конвертирующая входную строку в целые числа. Ввод строки осуществляется при помощи прерывания int 21h, ah=09h. Для каждого числа необходимо вводить цифры с разных строк. Затем начинается процесс вычислений функций под номерами 1,4 и 3. Так как условия для функций 1 и 4 — одинаковы, то выполнить сравнение между числами a и b можно один раз. Сравнение производится при помощи последовательный команд: cmp ax, bx, и jle. Первая команда выстраивает флаги процессора, а вторая выполняет переход в соответствии с флагами. Арифметические операции сложение и вычитание выполнялись при помощи команд add и sub соответственно. Умножение производилось при помощи битового сдвига влево и дополнительного сложения (если множитель не равен степени двойки). Модуль числа вычислялся следующим образом: сначала определялся знак числа: если знак меньше нуля (число отрицательное), то, при помощи команды neg выполнялось изменение знака числа на противоположный. Если число было больше, то оно оставалось без изменений. Нахождение минимального числа производилось при помощи сравнения этих чисел.

Результаты тестирования программы см в таблице 1.

Код и листинг программы см. в приложении А.

Таблица 1. – Результаты тестирования программы

		a	b	i	k	f1	f2	f3	test
a>b	k=0	5	3	2	0	11	-8	3	Верно
a<=b	k=0	3	5	2	0	10	12	22	Верно
a>b	k/=0	5	3	2	1	11	-8	-8	Верно
a<=b	k/=0	3	5	2	1	10	12	10	Верно
a>b	k=0	5	3	10	0	-5	-56	61	Верно

Выводы.

В этой работе были изучена работа с ветвлениями и обработка пользовательского ввода на языке Ассемблера. Изучены команды сравнения и битового сдвига.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ И ДРУГИЕ ФАЙЛЫ

Название файла: LAB3.ASM

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


DATA SEGMENT
    INPUT DB 10, 10 DUP ('$')
    SIGN DB 0
    LEN DB 0
    TEMP DB 0
    A DW 0
    B DW 0
    I DW 0
    K DW 0
    RES1 DW 0
    RES2 DW 0
    RES3 DW 0
DATA ENDS


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


     READ PROC NEAR

        MOV SIGN, 0             ;SIGN VALUE INITIALIZER
        MOV LEN, 0              ;NUMBER LENGTH VALUE INITIALIZER
        MOV DI, 2               ;NUMBER OF THE FIRST ELEMENT IN INPUT,
ITTERATOR
        MOV BH, 0               ;INIT
        MOV CX, 0
        MOV AX, 0
```

```
        MOV DX, OFFSET INPUT    ;WRTITING OFFSET TO INPUT STRING INTO DX
        MOV AH, 0AH             ;READING USER INPUT AND WRITING IT TO "INPUT"
STRING
        INT 21H                 ;INTERRUPTION

        CMP INPUT[DI], 2DH      ;COMPARING FIRST SYMBOL OF INPUT WITH DASH
SYMBOL

        MOV DX, 0               ;INIT

        JNZ COUNTING             ;CHECKING A==B STATEMENT
        MOV SIGN, 1             ;IF FIRST SYBMOL IS DASH -> SIGN CHANGES TO 1
        ADD DI, 1               ;PROCEED TO NEXT SYBOL IN INPUT
        JMP COUNTING            ;JUMPING TO COUNTING PART

        COUNTING:
            CMP INPUT[DI], 24H
            JZ PROCESSING
            ADD DI, 1
            JMP COUNTING

        PROCESSING:
            MOV LEN, DI
            MOV DI, 2

            CMP SIGN, 1
            JNZ SKIP
            ADD DI, 1
            SKIP:

            MOV BH, LEN
            SUB BH, 4
            SUB BH, SIGN        ;FINDING AMOUNT OF DIGITS OF THE ABSOLUTE
NUMBER
            JZ FINALCALC

            JMP ITTER

        ITTER:
            MOV TEMP, BH
```

```
    MOV AX, 0
    MOV DX, 0
    MOV AL, INPUT[DI]
    SUB AL, 30H
POWER:
    MOV DX, AX
MULT10:
    SHL AX, 1
    SHL AX, 1
    SHL AX, 1
    ADD AX, DX
    ADD AX, DX
    SUB TEMP, 1
    CMP TEMP, 0
    JNZ POWER
ADD CX, AX
ADD DI, 1
SUB BH, 1
CMP BH, 0
JNZ ITTER

FINALCALC:
MOV AX, 0
MOV AL, INPUT[DI]
SUB AL, 30H
ADD CX, AX

CMP SIGN, 1
JNZ SKIP2
NEG CX
SKIP2:

MOV DI, 2           ;INPUT BUFFER CLEAR
MOV DX, 10
CLEAR:
MOV INPUT[DI], 24H
ADD DI, 1
SUB DX, 1
JNZ CLEAR
```

```
    RET
READ ENDP


MAIN PROC FAR
    PUSH  DS
    SUB   AX,AX
    PUSH  AX
    MOV   AX,DATA
    MOV   DS,AX
    MOV   CX, 0

    CALL READ
    MOV A, CX
    CALL READ
    MOV B, CX
    CALL READ
    MOV I, CX
    CALL READ
    MOV K, CX




    MOV AX, A
    MOV BX, B
    CMP AX, BX
    JLE LESS


    MOV AX, 15      ;F1.1
    SUB AX, I       ;
    SUB AX, I       ;
    MOV RES1, AX    ;F1.1

    MOV AX, I       ;F4.1
    SHL AX, 1       ;
    SHL AX, 1       ;
    ADD AX, I       ;
    ADD AX, I       ;
```

```
SUB AX, 4       ;
NEG AX          ;
MOV RES2, AX    ;F4.1

JMP F3

LESS:

MOV AX, I       ;F1.2
SHL AX, 1       ;
ADD AX, I       ;
ADD AX, 4       ;
MOV RES1, AX    ;F1.2

MOV AX, I       ;F4.2
ADD AX, 2       ;
MOV DX, AX      ;
SHL AX, 1       ;
ADD AX, DX      ;
MOV RES2, AX    ;F4.2


F3:
MOV CX, K
CMP CX, 0
JNZ F32

MOV CX, RES1
ADD CX, RES2
CMP CX, 0
JNL FINALE

NEG CX
JMP FINALE

F32:
MOV AX, RES1
MOV BX, RES2
CMP AX, BX
JLE F321
```

```
        MOV CX, RES2
        JMP FINALE

        F321:
        MOV CX, RES1


        FINALE:
        MOV AH,4CH
        INT 21H

    MAIN ENDP
    CODE ENDS
END MAIN
```

Файл LAB3.LST

```
#Microsoft (R) Macro Assembler Version 5.10                 11/18/21 07:31:2
                                                            Page     1-1


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

 0018                   AStack ENDS


 0000                   DATA SEGMENT
 0000  0A                   input DB 10, 10 DUP ('$')
       000A[
         24
                    ]

 000B  00                   sign DB 0
 000C  00                   len DB 0
 000D  00                   temp DB 0
```

```
000E  0000                a DW 0
0010  0000                b DW 0
0012  0000                i DW 0
0014  0000                k DW 0
0016  0000                res1 DW 0
0018  0000                res2 DW 0
001A  0000                res3 DW 0
001C                  DATA ENDS


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


0000                       Read PROC NEAR

0000  C6 06 000B R 00              mov sign, 0             ;sign value ini
                      tializer
0005  C6 06 000C R 00              mov len, 0              ;number length
                      value initializer
000A  BF 0002                      mov di, 2               ;number of the
                      first element in input, itterator
000D  B7 00                        mov bh, 0               ;init
000F  B9 0000                      mov cx, 0
0012  B8 0000                      mov ax, 0
0015  BA 0000 R                 mov dx, OFFSET input    ;wrtiting offse
                      t to input string into dx
0018  B4 0A                        mov ah, 0Ah             ;reading user i
                      nput and writing it to "input" string
001A  CD 21                        int 21h                 ;interruption

001C  80 BD 0000 R 2D              cmp input[di], 2Dh      ;comparing firs
                      t symbol of input with dash symbol

0021  BA 0000                      mov dx, 0               ;init

0024  75 0B                        jnz counting             ;checking a==b
#Microsoft (R) Macro Assembler Version 5.10                 11/18/21 07:31:2
                                                            Page     1-2
```

```
                        statement
0026  C6 06 000B R 01               mov sign, 1             ;if first sybmo
                      l is dash -> sign changes to 1
002B  83 C7 01                      add di, 1               ;proceed to nex
                      t sybol in input
002E  EB 01 90                      jmp counting            ;jumping to cou
                      nting part

0031                          counting:
0031  80 BD 0000 R 24                     cmp input[di], 24h
0036  74 05                               jz processing
0038  83 C7 01                            add di, 1
003B  EB F4                               jmp counting

003D                          processing:
003D  89 3E 000C R                        mov len, di
LAB3.ASM(53): warning A4031: Operand types must match
0041  BF 0002                             mov di, 2

0044  80 3E 000B R 01                     cmp sign, 1
0049  75 03                               jnz skip
004B  83 C7 01                            add di, 1
004E                                skip:

004E  8A 3E 000C R                        mov bh, len
0052  80 EF 04                            sub bh, 4
0055  2A 3E 000B R                        sub bh, sign        ;finding amount
                      of digits of the absolute number
0059  74 38                               jz finalcalc

005B  EB 01 90                            jmp itter

005E                          itter:
005E  88 3E 000D R                        mov temp, bh
0062  B8 0000                             mov ax, 0
0065  BA 0000                             mov dx, 0
0068  8A 85 0000 R                        mov al, input[di]
006C  2C 30                               sub al, 30h
006E                          power:
```

```
006E  8B D0                           mov dx, ax
0070                            mult10:
0070  D1 E0                           SHL ax, 1
0072  D1 E0                           SHL ax, 1
0074  D1 E0                           SHL ax, 1
0076  03 C2                           add ax, dx
0078  03 C2                           add ax, dx
007A  80 2E 000D R 01                 sub temp, 1
007F  80 3E 000D R 00                 cmp temp, 0
0084  75 E8                           jnz power
0086  03 C8                       add cx, ax
0088  83 C7 01                    add di, 1
008B  80 EF 01                    sub bh, 1
008E  80 FF 00                    cmp bh, 0
0091  75 CB                       jnz itter
```



```
0093                            finalcalc:
0093  B8 0000                     mov ax, 0
0096  8A 85 0000 R                mov al, input[di]
009A  2C 30                       sub al, 30h
009C  03 C8                       add cx, ax

009E  80 3E 000B R 01             cmp sign, 1
00A3  75 02                       jnz skip2
00A5  F7 D9                       neg cx
00A7                            skip2:

00A7  BF 0002                     mov di, 2           ;input buffer clear
00AA  BA 000A                     mov dx, 10
00AD                            clear:
00AD  C6 85 0000 R 24             mov input[di], 24h
00B2  83 C7 01                    add di, 1
00B5  83 EA 01                    sub dx, 1
00B8  75 F3                       jnz clear

00BA  C3                        ret
```

```
00BB                        Read ENDP


00BB                        Main PROC FAR
00BB  1E                        push  DS
00BC  2B C0                           sub   AX,AX
00BE  50                        push  AX
00BF  B8 ---- R                 mov   AX,DATA
00C2  8E D8                           mov   DS,AX
00C4  B9 0000                         mov   CX, 0

00C7  E8 0000 R                 Call Read
00CA  89 0E 000E R                    mov a, cx
00CE  E8 0000 R                 Call Read
00D1  89 0E 0010 R                    mov b, cx
00D5  E8 0000 R                 Call Read
00D8  89 0E 0012 R                    mov i, cx
00DC  E8 0000 R                 Call Read
00DF  89 0E 0014 R                    mov k, cx




00E3  A1 000E R                 mov ax, a
00E6  8B 1E 0010 R                    mov bx, b
00EA  3B C3                           cmp ax, bx
00EC  7E 28                           jle less


00EE  B8 000F                         mov ax, 15      ;f1.1
00F1  2B 06 0012 R                    sub ax, i       ;
00F5  2B 06 0012 R                    sub ax, i       ;
00F9  A3 0016 R                 mov res1, ax    ;f1.1

#Microsoft (R) Macro Assembler Version 5.10                 11/18/21 07:31:2
                                                            Page     1-4


00FC  A1 0012 R                 mov ax, i       ;f4.1
00FF  D1 E0                           shl ax, 1       ;
```

```
0101  D1 E0                         shl ax, 1       ;
0103  03 06 0012 R                   add ax, i       ;
0107  03 06 0012 R                   add ax, i       ;
010B  2D 0004                        sub ax, 4       ;
010E  F7 D8                          neg ax          ;
0110  A3 0018 R                 mov res2, ax    ;f4.1

0113  EB 1F 90                      jmp f3

0116                            less:

0116  A1 0012 R                 mov ax, i       ;f1.2
0119  D1 E0                          SHL ax, 1       ;
011B  03 06 0012 R                   add ax, i       ;
011F  05 0004                        add ax, 4       ;
0122  A3 0016 R                 mov res1, ax    ;f1.2

0125  A1 0012 R                 mov ax, i       ;f4.2
0128  05 0002                        add ax, 2       ;
012B  8B D0                          mov dx, ax      ;
012D  D1 E0                          shl ax, 1       ;
012F  03 C2                          add ax, dx      ;
0131  A3 0018 R                 mov res2, ax    ;f4.2


0134                            f3:
0134  8B 0E 0014 R                   mov cx, k
0138  83 F9 00                       cmp cx, 0
013B  75 12                          jnz f32

013D  8B 0E 0016 R                   mov cx, res1
0141  03 0E 0018 R                   add cx, res2
0145  83 F9 00                       cmp cx, 0
0148  7D 1B                          jnl finale

014A  F7 D9                          neg cx
014C  EB 17 90                       jmp finale

014F                            f32:
014F  A1 0016 R                 mov ax, res1
```

```
 0152  8B 1E 0018 R                    mov bx, res2
 0156  3B C3                           cmp ax, bx
 0158  7E 07                           jle f321
 015A  8B 0E 0018 R                    mov cx, res2
 015E  EB 05 90                        jmp finale

 0161                            f321:
 0161  8B 0E 0016 R                    mov cx, res1


 0165                            finale:
 0165  B4 4C                           mov ah,4Ch
#Microsoft (R) Macro Assembler Version 5.10                 11/18/21 07:31:2
                                                            Page     1-5


 0167  CD 21                           int 21h

 0169                        Main ENDP
 0169                        CODE ENDS
                         END  Main#Microsoft  (R)  Macro  Assembler  Version  5.10
11/18/21 07:31:2
                                                            Symbols-1


Segments and Groups:

                N a m e             Length       Align      Combine Class

ASTACK . . . . . . . . . . . . .    0018  PARA  STACK
CODE . . . . . . . . . . . . . .    0169  PARA  NONE
DATA . . . . . . . . . . . . . .    001C  PARA  NONE

Symbols:

                N a m e             Type   Value       Attr

A  . . . . . . . . . . . . . . .    L WORD      000E  DATA

B  . . . . . . . . . . . . . . .    L WORD      0010  DATA
```

```
CLEAR . . . . . L NEAR      00AD  CODE
COUNTING . . . . . L NEAR      0031  CODE

F3 . . . . . L NEAR      0134  CODE
F32 . . . . . L NEAR      014F  CODE
F321 . . . . . L NEAR      0161  CODE
FINALCALC . . . . . L NEAR      0093  CODE
FINALE . . . . . L NEAR      0165  CODE

I . . . . . L WORD      0012  DATA
INPUT . . . . . L BYTE      0000  DATA
ITTER . . . . . L NEAR      005E  CODE

K . . . . . L WORD      0014  DATA

LEN . . . . . L BYTE      000C  DATA
LESS . . . . . L NEAR      0116  CODE

MAIN . . . . . F PROC      00BB  CODE  Length = 00AE
MULT10 . . . . . L NEAR      0070  CODE

POWER . . . . . L NEAR      006E  CODE
PROCESSING . . . . . L NEAR      003D  CODE

READ . . . . . N PROC      0000  CODE  Length = 00BB
RES1 . . . . . L WORD      0016  DATA
RES2 . . . . . L WORD      0018  DATA
RES3 . . . . . L WORD      001A  DATA

SIGN . . . . . L BYTE      000B  DATA
SKIP . . . . . L NEAR      004E  CODE
SKIP2 . . . . . L NEAR      00A7  CODE

TEMP . . . . . L BYTE      000D  DATA

@CPU . . . . . TEXT  0101h
@FILENAME . . . . . TEXT  LAB3
@VERSION . . . . . TEXT   510#Microsoft (R) Macro Assembler
Version 5.10                11/18/21 07:31:2
```

```
  203 Source  Lines
  203 Total   Lines
   35 Symbols

47886 + 451179 Bytes symbol space free

    1 Warning Errors
    0 Severe  Errors
```