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 _____ Живаев М.А

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

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.

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.

Вариант 2:

$$i1 = f1 = \begin{cases} 15 - 2*i, & \text{при } a>b \\ 3*i + 4, & \text{при } a<=b \end{cases}$$

$$i2 = f3 = \begin{cases} 7 - 4*i, & \text{при } a>b \\ 8 - 6*i, & \text{при } a<=b \end{cases}$$

$$res = f2 = \begin{cases} \max(i1, 10 - i2), & \text{при } k<0 \\ |i1 - i2|, & \text{при } k>=0 \end{cases}$$

Выполнение работы.

Числа для работы программы вводятся сразу в asm файл. Для реализации алгоритмов использовались команда сравнения cmp и различные условные переходы. Для функций f1 и f3 сначала выполняются действия для случая, где a>b, иначе для a<=b. Чтобы определить какие действия выполнять, используется команда cmp для сравнения чисел a и b. А с помощью команды jle (в зависимости от результата) программа переходит к блоку, где рассчитываются соответствующие значения f1 и f3. Для операций умножения использовался битовый сдвиг влево(команда shl) и сложение (команда add).

Результаты тестирования представлены в табл. 1.

Тексты исходных файлов программ см. в приложении А.

Тексты файлов диагностических сообщений см. в приложении Б.

Таблица 1. Проверка работы программы.

№	Входные данные	Значение i1	Значение i2	Значение res
1	a = 10, b = 5, i = 5, k = 4	5	-13	18
2	a = 5, b = 10, i = 5, k = 4	19	-22	41
3	a = 1, b = 2, i = 3, k = -1	13	-10	20

Выводы.

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

ПРИЛОЖЕНИЕ А

ТЕКСТЫ ИСХОДНЫХ ФАЙЛОВ ПРОГРАММ

Название файла: **lab3.asm**

```
; Стек  программы
AStack SEGMENT  STACK
           DW 12 DUP(?)
AStack  ENDS
;Данные программы
DATA SEGMENT
    ;Директивы описания данных
    a      DW 10
    b      DW 5
    i      DW 5
    k      DW 4
    i1     DW 0
    i2     DW 0
    res    DW 0
    temp   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    bx, b
```

```
cmp     a, bx
jle     less_equal

; a > b
; f1
mov     ax, i
shl     ax, 1
mov     cx, 15
sub     cx, ax
mov     i1, cx

; f3
shl     ax, 1
mov     cx, 7
sub     cx, ax
mov     i2, cx

jmp     fin

; a <= b
less_equal:
; f1
mov     cx, i
shl     cx, 1
add     cx, i
mov     temp, cx ; temp = 3i
add     cx, 4
mov     i1, cx

; f3
mov     ax, temp
shl     ax, 1
mov     cx, 8
sub     cx, ax
```

```
mov    i2, cx

; f2
fin:
mov    bx, k
mov    cx, i1
mov    ax, i2

cmp    bx, 0
jl     negk

sub    cx, ax

cmp    cx, 0
jge    abs_skip
neg    cx
abs_skip:

mov    res, cx
jmp    result

negk:
mov    dx, 10
sub    dx, ax

cmp    dx, cx
jg     max
mov    res, cx
jmp    result

max:
mov    res, dx
```

```
    result:
    ret
Main ENDP
CODE      ENDS
END Main
```

ПРИЛОЖЕНИЕ Б

ТЕКСТЫ ФАЙЛОВ ДИАГНОСТИЧЕСКИХ СООБЩЕНИЙ

Название файла: **lab3.lst**

```
Microsoft (R) Macro Assembler Version 5.10                  11/21/21
18:13:4
                                                             Page
1-1


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

 0018                 AStack  ENDS
                ;Данные программы
 0000                 DATA SEGMENT
                    ;Директивы описания дан
                ных
 0000  000A             a     DW 10
 0002  0005             b     DW 5
 0004  0005             i     DW 5
 0006  0004             k     DW 4
 0008  0000             i1    DW 0
 000A  0000             i2    DW 0
 000C  0000             res   DW 0
 000E  0000             temp  DW 0

 0010                 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 1E 0002 R              mov    bx, b
 000D  39 1E 0000 R              cmp    a, bx
 0011  7E 1C               jle    less_equal

                    ; a > b
                    ; f1
 0013  A1 0004 R           mov    ax, i
 0016  D1 E0               shl    ax, 1
 0018  B9 000F                   mov    cx, 15
 001B  2B C8               sub    cx, ax
 001D  89 0E 0008 R              mov    i1, cx

                    ; f3
 0021  D1 E0               shl    ax, 1
 0023  B9 0007                   mov    cx, 7
 0026  2B C8               sub    cx, ax
 0028  89 0E 000A R              mov    i2, cx

 002C  EB 24 90                  jmp    fin
```



```
                    ; a <= b
```

```
002F                    less_equal:
                    ; f1
002F  8B 0E 0004 R              mov    cx, i
0033  D1 E1                shl    cx, 1
0035  03 0E 0004 R              add    cx, i
0039  89 0E 000E R              mov    temp, cx ; temp = 3i
003D  83 C1 04                  add    cx, 4
0040  89 0E 0008 R              mov    i1, cx

                    ; f3
0044  A1 000E R            mov    ax, temp
0047  D1 E0                shl    ax, 1
0049  B9 0008                   mov    cx, 8
004C  2B C8                sub    cx, ax
004E  89 0E 000A R              mov    i2, cx

                    ; f2
0052                    fin:
0052  8B 1E 0006 R              mov    bx, k
0056  8B 0E 0008 R              mov    cx, i1
005A  A1 000A R            mov    ax, i2

005D  83 FB 00                  cmp    bx, 0
0060  7C 10                jl     negk

0062  2B C8                sub    cx, ax

0064  83 F9 00                  cmp    cx, 0
0067  7D 02                jge    abs_skip
0069  F7 D9                neg    cx
006B                    abs_skip:

006B  89 0E 000C R              mov    res, cx
006F  EB 15 90                  jmp    result
```

```
0072                    negk:
0072  BA 000A                   mov    dx, 10
0075  2B D0             sub    dx, ax

0077  3B D1             cmp    dx, cx
0079  7F 07             jg     max
007B  89 0E 000C R             mov    res, cx
007F  EB 05 90                 jmp    result

0082                    max:
0082  89 16 000C R             mov    res, dx



0086                    result:
0086  CB                ret
0087                Main ENDP
0087                CODE      ENDS
                END Main
```


Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0087	PARA	NONE	
DATA	0010	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr

```
A  . . . . . .  L WORD   0000  DATA
ABS_SKIP . . . . . .  L NEAR   006B  CODE

B  . . . . . .  L WORD   0002  DATA

FIN  . . . . . .  L NEAR   0052  CODE

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

K  . . . . . .  L WORD   0006  DATA

LESS_EQUAL . . . . . .  L NEAR   002F  CODE

MAIN . . . . . .  F PROC   0000  CODE  Length =
0087
MAX  . . . . . .  L NEAR   0082  CODE

NEGK . . . . . .  L NEAR   0072  CODE

RES  . . . . . .  L WORD   000C  DATA
RESULT . . . . . .  L NEAR   0086  CODE

TEMP . . . . . .  L WORD   000E  DATA

@CPU . . . . . .  TEXT  0101h
@FILENAME  . . . . . .  TEXT  LAB3
@VERSION . . . . . .  TEXT  510


    104 Source  Lines
    104 Total   Lines
     23 Symbols
```

```
48056 + 461251 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```