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. 0382 _____ Довченко М.К.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Целью данной работы является изучение представления и обработки целых числе, организации ветвящихся процессов.

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.

Выполнение работы.

Вариант 22:

$$f4 = \begin{cases} -(6*i - 4), & \text{при } a>b \\ 3*(i+2), & \text{при } a<=b \end{cases} \quad f8 = \begin{cases} -(6*i+8), & \text{при } a>b \\ 9 -3*(i-1), & \text{при } a<=b \end{cases} \quad f3 = \begin{cases} |i1 + i2|, & \text{при } k=0 \\ \min(i1,i2), & \text{при } k/=0 \end{cases}$$

При выполнении работы были использованы некоторые следующие команды:

cmp - сравнивает 2 операнда

jmp - перемещает в какую то часть кода, также существуют другие условные jump функции, для их применения необходимо сначала использовать функцию cmp (например jle, jne, jl)

neg - меняет знак значения какого либо операнда

shl - перемещает все биты значения операнда влево

Для выполнения программы также используются метки, для того чтобы можно было пользоваться jump командами.

Исходный программый код смотреть в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	a = 4 b = 3	i1 = -8 i2 = -20	Программа работает корректно

	i = 2 k = 1	res = -20	
2.	a = 3 b = 4 i = 5 k = 0	i1 = 21 i2 = -3 res = 18	Программа работает корректно
3.	a = 25 b = -30 i = 16 k = 1	i1 = -92 i2 = -104 res = -104	Программа работает корректно

Выводы.

При выполнении данной лабораторной работы были изучены принципы представления и обработки целых чисел, организации ветвящихся процессов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src.asm

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

DATA SEGMENT
    a   DW 4
    b   DW 3
    i   DW 2
    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 AX, a
    cmp AX, b
    jle f2

    f1:
    mov AX, i
    shl AX, 1  ; умножаем i на 2 (2i)
    mov BX, AX ; сохраняем значение 2i в BX
    shl AX, 1  ; умножаем 2i на 2 (4i)
    add AX, BX ; получаем 6i в регистре AX
    neg AX      ; вносим в скобку
    mov i1, AX  ; сохраняем значение -6i в i1
```

```
add i1, 4   ; отнимаем 4 от 6i
mov i2, AX  ; сохраняем значение -6i в i2
sub i2, 8   ; прибавляем 8 к 6i
jmp f3

f2:
mov AX, i
shl AX, 1   ; умножаем i на 2 (2i)
mov BX, AX  ; сохраняем значение 2i в BX
add AX, BX  ; получаем 3i в регистре AX
mov i1, AX  ; сохраняем значение 3i в i1
mov i2, AX  ; сохраняем значение 3i в i2
add i1, 6
sub i2, 12
neg i2

f3:
mov AX, k
cmp AX, 0   ; сравниваем K с нулем
jne f3_2

f3_1:
mov AX, i1
add AX, i2
cmp AX, 0
jl negative
jmp result

negative:
neg AX
jmp result

f3_2:
mov BX, i1
cmp BX, i2
jle min_i1
mov AX, i2
jmp result

min_i1:
```

```
        mov AX, i1

        result:
        mov res, AX
        ret

    Main ENDP
    CODE ENDS
        END Main
```

Название файла: src.lst

```
#Microsoft (R) Macro Assembler Version 5.10                    11/29/21
18:47:4
                                                               Page     1-1


 = 0024                     EOL EQU '$'
 0000                   AStack SEGMENT STACK
 0000  000C[                       DW 12 DUP(?)
        ????
                    ]

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  0004                        a   DW 4
 0002  0003                        b   DW 3
 0004  0002                        i   DW 2
 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  A1 0000 R         mov AX, a
 000C  3B 06 0002 R             cmp AX, b
 0010  7E 20                    jle f2

 0012                    f1:
 0012  A1 0004 R         mov AX, i
 0015  D1 E0                     shl AX, 1   ; CґPjPSPsP¶P°PµPj i PSP° 2
(2i)
 0017  8B D8                         mov   BX,   AX   ;   CЃPsC…CЂP°PSCЏPµPj
P·PSP°C‡PµP
                    SPёPµ 2i PI BX
 0019  D1 E0                     shl AX, 1   ; CґPjPSPsP¶P°PµPj 2i PSP° 2
(4i
                    )
 001B  03 C3                      add  AX,  BX  ;  PїPsP»CґC‡P°PµPj  6i  PI
CЂPµPiP
                    ёCЃC,CЂPµ AX
 001D  F7 D8                     neg AX      ; PIPSPsCЃPёPj PI CЃPєPsP±PєCґ
 001F  A3 0008 R         mov i1, AX  ; CЃPsC…CЂP°PSCЏPµPj P·PSP°C‡Pµ
                    PSPёPµ -6i PI i1
 0022  83 06 0008 R 04               add i1, 4    ; PsC,PSPёPjP°PµPj 4
PsC, 6i
 0027  A3 000A R         mov i2, AX  ; CЃPsC…CЂP°PSCЏPµPj P·PSP°C‡Pµ
                    PSPёPµ -6i PI i2
 002A  83 2E 000A R 08               sub i2, 8   ; PїCЂPёP±P°PIP»CЏPµPj 8
Pє 6i
 002F  EB 1E 90                  jmp f3

 0032                    f2:
 0032  A1 0004 R         mov AX, i
 0035  D1 E0                     shl AX, 1    ; CґPjPSPsP¶P°PµPj i PSP° 2
(2i
                    )
```

```
 0037  8B D8                      mov  BX,  AX    ;  CЃPsC…CЂP°PSCЏPµPj
P·PSP°C‡Pµ
                     PSPёPµ 2i PI BX
 0039  03 C3                    add  AX,  BX    ;  PїPsP»CѓC‡P°PµPj  3i  PI
CЂPµPi
                     PёCЃC,CЂPµ AX
 003B  A3 0008 R          mov i1, AX  ; CЃPsC…CЂP°PSCЏPµPj P·PSP°C‡Pµ
                     PSPёPµ 3i PI i1
 003E  A3 000A R          mov i2, AX  ; CЃPsC…CЂP°PSCЏPµPj P·PSP°C‡Pµ
                     PSPёPµ 3i PI i2
 0041  83 06 0008 R 06              add i1, 6
 0046  83 2E 000A R 0C              sub i2, 12
 004B  F7 1E 000A R            neg i2

 004F                       f3:
 004F  A1 0006 R            mov AX, k
 0052  3D 0000                   cmp  AX,  0    ;  CЃCЂP°PIPSPёPIP°PµPj  K  CЃ
PSC
                     ѓP»PµPj
 0055  75 14                     jne f3_2

 0057                       f3_1:
 0057  A1 0008 R            mov AX, i1
 005A  03 06 000A R               add AX, i2
 005E  3D 0000                    cmp AX, 0
 0061  7C 03                      jl negative
 0063  EB 19 90                   jmp result

 0066                       negative:
 0066  F7 D8                      neg AX
 0068  EB 14 90                   jmp result

 006B                       f3_2:
 006B  8B 1E 0008 R               mov BX, i1
 006F  3B 1E 000A R               cmp BX, i2
```

```
0073  7E 06                 jle min_i1
0075  A1 000A R          mov AX, i2
0078  EB 04 90               jmp result

007B                     min_i1:
007B  A1 0008 R          mov AX, i1

007E                     result:
007E  A3 000C R          mov res, AX
0081  CB                 ret

0082                 Main ENDP
0082                 CODE ENDS
                         END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0082	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr	
A	L WORD	0000	DATA	
B	L WORD	0002	DATA	
EOL	NUMBER	0024		
F1	L NEAR	0012	CODE	
F2	L NEAR	0032	CODE	
F3	L NEAR	004F	CODE	
F3_1	L NEAR	0057	CODE	
F3_2	L NEAR	006B	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 = 0082
MIN_I1	L NEAR	007B	CODE	
NEGATIVE	L NEAR	0066	CODE	

```
RES  . . . . . .     L WORD     000C  DATA
RESULT . . . . . .     L NEAR     007E  CODE

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


     86 Source  Lines
     86 Total   Lines
     25 Symbols

  48070 + 461237 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```