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»

Тема: Представление и обработка целых чисел.
Организация ветвящихся процессов

Студент гр. 9383 _____ Чумак М.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить представление целых чисел, научиться их обрабатывать, познакомиться с организаций 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.

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 №23:

i1 = f1 = f5 = < / 20 - 4*i , при a>b
\ -(6*I - 6), при a<=b

/ 2*(i+1) -4 , при a>b

i2 = f2 = f6 = <

\ 5 - 3*(i+1), при a<=b

/ min (|i1 - i2|, 2), при k<0

i3 = f3 = f4 = <

\ max(-6, -i2), при k>=0

Выполнение работы.

В ходе работы была реализована программа на языке Ассемблер, которая по заданным целочисленным параметрам вычисляет значения функций. Исходные данные заносятся в программу до выполнения, а выходные данные отслеживаются через отладчик. Были реализованы следующие функции:

- f1_1, f1_2 — для нахождения значения f1 (если a>b, то выполняется f1_1, иначе f1_2);
- f2_1, f2_2 — для нахождения значений f2 (если a>b, то выполняется f2_1, иначе f2_2);
- f3, f3_1, f3_cmp_2, f3_res, f_end — для нахождения значений f3, где отдельно происходит сравнение с 2 (f3_cmp_2).

Следуя четвёртому пункту из замечаний, для минимизации длины кода, было решено упростить следующую функцию:

f6: 2*(i+1) - 4 → 2*i - 2; 5 - 3*(i+1) → 2 - 3*i

То есть после преобразований функции выглядят следующим образом:

/ 20 - 4*i , при a>b

i1 = f1 = f5 = <

\ -(6*I - 6), при a<=b

/ 2*i - 2 , при a>b

i2 = f2 = f6 = <

\ 2 - 3*i, при a<=b

/ min (|i1 - i2|, 2), при k<0

i3 = f3 = f4 = <

\ max(-6, -i2), при k>=0

Тестирование.

1) a = 1, b = 2, i = 2, k = -2 => f1 = -6, f2 = -4, f3 = 2

2) a = 1, b = 2, i = 2, k = 1 => f1 = -6, f2 = -4, f3 = 4

3) a = 2, b = 1, i = 2, k = -2 => f1 = 12, f2 = 2, f3 = 2

4) a = 2, b = 1, i = 2, k = 1 => f1 = 12, f2 = 2, f3 = -2

Выводы.

Изучено представление целых чисел, получены навыки работы с целыми числами и ветвящимися процессами.

Содержимое файла lb3.asm представлено в приложении А. Содержимое файла lb3.lst представлено в приложении Б.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Файл LAB3.ASM

```
AStack SEGMENT STACK
        DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
a       DW   2
b       DW   1
i       DW   2
k       DW   1
i1      DW   ?
i2      DW   ?
res     DW   ?
DATA ENDS

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

Main PROC FAR
        mov ax, DATA
        mov ds, ax
f1_1:
        mov ax, a
        cmp ax, b
        jg f1_2         ;если a > b, то переходим к f1_2
                        ;иначе a <= b, выполняем действия дальше
        mov ax, i
        shl ax, 1       ;ax = 2*i
        mov bx, ax      ;bx = 2*i
```

```
        shl ax, 1       ;ax = 4*i
        add ax, bx      ;ax = 6*i
        sub ax, 6       ;ax = 6*i-6
        neg ax                  ;ax = -(6*i-6)
        mov i1, ax
        jmp f2_1
f1_2:
        mov ax, i
        shl ax, 1       ;ax = 2*i
        shl ax, 1       ;ax = 4*i
        neg ax                  ;ax = -4*i
        add ax, 20      ;ax = -4*i+20, что идентично 20-4*i
        mov i1, ax
f2_1:
        mov ax, a
        cmp ax, b
        jg f2_2         ;если a > b, то переходим к f2_2
                        ;иначе a <= b, выполняем действия дальше
        mov ax, i
        mov bx, ax
        shl ax, 1       ;ax = 2*i
        add ax, bx      ;ax = 3*i
        neg ax                  ;ax = -(3*i)
        add ax, 2       ;ax = -(3*i)+2, что идентично 2-3*i
        mov i2, ax
        jmp f3
f2_2:
        mov ax, i
        shl ax, 1       ;ax = 2*i
        sub ax, 2       ;ax = 2*i-2
```

```
        mov i2, ax
f3:
        mov ax, k
        cmp k, 0
        jl f3_1         ;если k < 0, то переходим к f3_1
                        ;иначе k >= 0, выполняем действия дальше
        mov ax, i2      ;ax = i2
        neg ax                  ;ax = -i2
        cmp ax, -6
        jg f3_res       ;если ax > -6, то переходим к выводу -i2
        mov res, -6     ;иначе res = -6
        jmp f_end
f3_1:
        mov ax, i1      ;ax = i1
        sub ax, i2      ;ax = i1-i2
        cmp ax, 0
        jg f3_cmp_2         ;если ax > 0, то переходим к сравнению с 2
                        ;иначе идём берём модуль
        neg ax                  ;ax = -i1+i2
f3_cmp_2:
        cmp ax, 2
        jl f3_res       ;если ax < 2, то переходим к выводу ax
        mov res, 2
        jmp f_end
f3_res:
        mov res, ax
        jmp f_end
f_end:
        mov ah, 4ch
        int 21h
```

```
Main  ENDP
CODE        ENDS
        END Main
```

ПРИЛОЖЕНИЕ В
ДИАГНОСТИЧЕСКОЕ СООБЩЕНИЕ

Файл LAB3.LST

```
#Microsoft (R) Macro Assembler Version 5.10             11/10/20 23:46:2
                                                        Page     1-1
0000                    AStack SEGMENT STACK
0000  0020[                     DW 32 DUP(?)
        ????
                    ]
0040                    AStack ENDS

0000                    DATA SEGMENT
0000  0002              a       DW   2
0002  0001              b       DW   1
0004  0002              i       DW   2
0006  0001              k       DW   1
0008  0000              i1      DW   ?
000A  0000              i2      DW   ?
000C  0000              res     DW   ?
000E                    DATA ENDS

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

0000                    Main PROC FAR
0000  B8 ---- R                mov ax, DATA
0003  8E D8                       mov ds, ax
0005                    f1_1:
0005  A1 0000 R                 mov ax, a
0008  3B 06 0002 R                cmp ax, b
```

```
000C  7F 16                     jg f1_2       ;РµСЃР»Рё  a  >  b,  C,Ps
РїРµСЂРµС
                        …PsPrPëPj Pє f1_2
                                    ;РёPSP°C‡Pµ a <= b, PIC
                        ‹РїPsP»PSCЏPµPj          PrPµP№CЃC,PIPёCЏI
PrP°P»CЊC€Pµ
000E  A1 0004 R             mov ax, i
0011  D1 E0                     shl ax, 1     ;ax = 2*i
0013  8B D8                     mov bx, ax    ;bx = 2*i
0015  D1 E0                     shl ax, 1     ;ax = 4*i
0017  03 C3                     add ax, bx    ;ax = 6*i
0019  2D 0006                   sub ax, 6     ;ax = 6*i-6
001C  F7 D8                     neg ax            ;ax = -(6*i-6)
001E  A3 0008 R             mov i1, ax
0021  EB 10 90                  jmp f2_1
0024                    f1_2:
0024  A1 0004 R             mov ax, i
0027  D1 E0                     shl ax, 1     ;ax = 2*i
0029  D1 E0                     shl ax, 1     ;ax = 4*i
002B  F7 D8                     neg ax            ;ax = -4*i
002D  05 0014                   add ax, 20    ;ax = -4*i+20, C‡C,Ps P
                        ёPrPµPSC,PёC‡PSPs 20-4*i
0030  A3 0008 R             mov i1, ax
0033                    f2_1:
0033  A1 0000 R             mov ax, a
0036  3B 06 0002 R              cmp ax, b
003A  7F 14                     jg f2_2       ;РµСЃР»Рё  a  >  b,  C,Ps
РїРµСЂРµС
                        …PsPrPëPj Pє f2_2
                                    ;РёPSP°C‡Pµ a <= b, PIC
```

```
003C  A1 0004 R                 mov ax, i
003F  8B D8                             mov bx, ax
0041  D1 E0                             shl ax, 1       ;ax = 2*i
0043  03 C3                             add ax, bx      ;ax = 3*i
0045  F7 D8                             neg ax                  ;ax = -(3*i)
0047  05 0002                           add ax, 2       ;ax = -(3*i)+2, C‡C,Ps
                        PëPrPµPSC,PëC‡PSPs 2-3*i
004A  A3 000A R                 mov i2, ax
004D  EB 0C 90                          jmp f3
0050                    f2_2:
0050  A1 0004 R                 mov ax, i
0053  D1 E0                             shl ax, 1       ;ax = 2*i
0055  2D 0002                           sub ax, 2       ;ax = 2*i-2
0058  A3 000A R                 mov i2, ax
005B                    f3:
005B  A1 0006 R                 mov ax, k
005E  83 3E 0006 R 00                   cmp k, 0
0063  7C 13                             jl f3_1         ;PµCЃP»Pë  k  <  0,  C,Ps
PïPµCЂPµC
                        …PsPrPëPj Pє f3_1
                                        ;PëPSP°C‡Pµ k >= 0, PIC
                        ‹PïPsP»PSCЏPµPj                 PrPµP№CЃC,PIPëCЏI
PrP°P»CЊC€Pµ
0065  A1 000A R                 mov ax, i2   ;ax = i2
0068  F7 D8                             neg ax                  ;ax = -i2
006A  3D FFFA                           cmp ax, -6
006D  7F 25                             jg f3_res       ;PµCЃP»Pë ax > -6, C,Ps
                        PïPµCЂPµC…PsPrPëPj Pє PIC‹PIPsPrCѓ -i2
006F  C7 06 000C R FFFA                 mov res, -6  ;PëPSP°C‡Pµ res = -6
0075  EB 23 90                          jmp f_end
```

```
0078                    f3_1:
0078  A1 0008 R                 mov ax, i1   ;ax = i1
007B  2B 06 000A R                    sub ax, i2     ;ax = i1-i2
007F  3D 0000                         cmp ax, 0
0082  7F 02                           jg f3_cmp_2        ;PµCЃP»Pë   ax   >   0,
C,Ps
                        PïPµCЂPµC…PsPrPëPj Pє CЃCЂP°PIPSPµPSPëCЋ
CЃ 2
                                          ;PëPSP°C‡Pµ PëPrC‘Pj P±
                        PµCЂC‘Pj PjPsPrCѓP»CЊ
0084  F7 D8                           neg ax             ;ax = -i1+i2
0086                    f3_cmp_2:
0086  3D 0002                         cmp ax, 2
0089  7C 09                           jl f3_res      ;PµCЃP»Pë ax < 2, C,Ps
                        PïPµCЂPµC…PsPrPëPj Pє PIC‹PIPsPrCѓ ax
008B  C7 06 000C R 0002               mov res, 2
0091  EB 07 90                       jmp f_end
0094                    f3_res:
0094  A3 000C R                 mov res, ax
0097  EB 01 90                       jmp f_end
009A                    f_end:
009A  B4 4C                           mov ah, 4ch
009C  CD 21                           int 21h
009E                    Main  ENDP
009E                    CODE        ENDS
                              END Main
                                    Symbols-1
```

Segments and Groups:

```
                N a m e         Length         Align   Combine Class

ASTACK . . . . . .              0040  PARA      STACK
CODE . . . . . .                009E  PARA      NONE
DATA . . . . . .                000E  PARA      NONE

Symbols:

                N a m e         Type    Value Attr

A . . . . . .                   L WORD    0000  DATA

B . . . . . .                   L WORD    0002  DATA

F1_1 . . . . . .                L NEAR    0005  CODE
F1_2 . . . . . .                L NEAR    0024  CODE
F2_1 . . . . . .                L NEAR    0033  CODE
F2_2 . . . . . .                L NEAR    0050  CODE
F3 . . . . . .                  L NEAR    005B  CODE
F3_1 . . . . . .                L NEAR    0078  CODE
F3_CMP_2 . . . . . .            L NEAR    0086  CODE
F3_RES . . . . . .              L NEAR    0094  CODE
F_END . . . . . .               L NEAR    009A  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 = 009E

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

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


    91 Source  Lines
    91 Total   Lines
    25 Symbols

 47962 + 459298 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```