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
i2_2:
        mov ax, a
        cmp ax, b
        jg i2_1         ;если a > b, то переходим к i2_1
                        ;иначе a <= b, выполняем действия дальше
        mov ax, i       ;ax = i
        mov bx, ax      ;bx = i, ax = i
        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 ax = 2-3*i
        mov i2, ax
i1_2:
        shl ax, 1     ;ax = 4-6*i
        add ax, 2     ;ax = 6-6*i, что идентично ax = -(6*i-6)
        mov i1, ax
        jmp i3
i2_1:
        mov ax, i
        shl ax, 1     ;ax = 2*i
        sub ax, 2     ;ax = 2*i-2
        mov i2, ax
i1_1:
        shl ax, 1     ;ax = 4*i-4
        neg ax              ;ax = 4-4*i
        add ax, 16    ;ax = 20-4*i
        mov i1, ax
i3:
        mov ax, k
        cmp k, 0
        jl i3_1       ;если k < 0, то переходим к i3_1
                      ;иначе k >= 0, выполняем действия дальше
        mov ax, i2    ;ax = i2
        neg ax              ;ax = -i2
        cmp ax, -6
        jg i3_res     ;если ax > -6, то переходим к выводу -i2
        mov res, -6   ;иначе res = -6
        jmp f_end
```

```
i3_1:
        mov ax, i1    ;ax = i1
        sub ax, i2    ;ax = i1-i2
        cmp ax, 0
        jg i3_cmp_2;если ax > 0, то переходим к сравнению с 2
                    ;иначе берём модуль
        neg ax              ;ax = -i1+i2
i3_cmp_2:
        cmp ax, 2
        jl i3_res     ;если ax < 2, то переходим к выводу ax
        mov res, 2
        jmp f_end
i3_res:
        mov res, ax
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              11/12/20 10:39:1
                                                        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                    i2_2:
0005  A1 0000 R              mov ax, a
0008  3B 06 0002 R                cmp ax, b
```

```
000C  7F 1C                   jg i2_1       ;РµСЃР»Рё a > b, C,Ps РїРµСЂРµС
                              …PsPrPëPj Pє i2_1
                                            ;РëPSP°C‡Pµ a <= b, PIC ‹РїРsР»PSCЏPµPj PrPµP№CЃC,PIPëCЏ PrР°Р»СЊС€Pµ
000E  A1 0004 R               mov ax, i     ;ax = i
0011  8B D8                   mov bx, ax    ;bx = i, ax = i
0013  D1 E0                   shl ax, 1     ;ax = 2*i
0015  03 C3                   add ax, bx    ;ax = 3*i
0017  F7 D8                   neg ax        ;ax = (-3)*i
0019  05 0002                 add ax, 2     ;ax = (-3)*i+2, C‡C,Ps PëPrPµPSC,PëC‡PSPs ax = 2-3*i
001C  A3 000A R               mov i2, ax
001F                          i1_2:
001F  D1 E0                   shl ax, 1     ;ax = 4-6*i
0021  05 0002                 add ax, 2     ;ax = 6-6*i, C‡C,Ps PëP rPµPSC,PëC‡PSPs ax = -(6*i-6)
0024  A3 0008 R               mov i1, ax
0027  EB 16 90                jmp i3
002A                          i2_1:
002A  A1 0004 R               mov ax, i
002D  D1 E0                   shl ax, 1     ;ax = 2*i
002F  2D 0002                 sub ax, 2     ;ax = 2*i-2
0032  A3 000A R               mov i2, ax
0035                          i1_1:
0035  D1 E0                   shl ax, 1     ;ax = 4*i-4
0037  F7 D8                   neg ax        ;ax = 4-4*i
0039  05 0010                 add ax, 16    ;ax = 20-4*i
003C  A3 0008 R               mov i1, ax
```

```
003F                    i3:
003F  A1 0006 R                 mov ax, k
0042  83 3E 0006 R 00                   cmp k, 0
0047  7C 13                             jl i3_1         ;РµСЃР»Рё  k  <  0,  C,Ps
РїРµСЂРµС
                        …PsPrPëPj Pє i3_1
                                        ;РëPSP°C‡Pµ k >= 0, PIC
                        ‹РїРsР»PSCЏРµРj           PrPµP№CЃC,PIPëCЏ
PrР°Р»СЊС€Рµ
003F  A1 0006 R                 mov ax, k       ;ax = i2
004C  F7 D8                             neg ax          ;ax = -i2
004E  3D FFFA                           cmp ax, -6
0051  7F 25                             jg i3_res       ;РµСЃР»Рё ax > -6, C,Ps
                        РїРµСЂРµС…PsPrPëPj Pє PIC‹PIPsPrCѓ -i2
0053  C7 06 000C R FFFA                 mov res, -6  ;РëPSP°C‡Pµ res = -6
0059  EB 20 90                          jmp f_end
005C                    i3_1:
005C  A1 0008 R                 mov ax, i1      ;ax = i1
005F  2B 06 000A R                      sub ax, i2      ;ax = i1-i2
0063  3D 0000                           cmp ax, 0
0066  7F 02                             jg i3_cmp_2;РµСЃР»Рё ax > 0, C,Ps
                        РїРµСЂРµС…PsPrPëPj Pє СЃСЂР°РІРSPµPSPëCЋ
СЃ 2
                                        ;РëPSP°C‡Pµ Р±РµСЂС‘Pj
                        PjPsPrCѓР»СЊ
0068  F7 D8                             neg ax          ;ax = -i1+i2
```

```
006A                    i3_cmp_2:
006A  3D 0002                   cmp ax, 2
006D  7C 09                     jl i3_res       ;PµCЃP»Pë ax < 2, C,Ps
                        PïPµCЋPµC…PsPrPëPj Pє PIC‹PIPsPrCґ ax
006F  C7 06 000C R 0002         mov res, 2
0075  EB 04 90                  jmp f_end
0078                    i3_res:
0078  A3 000C R               mov res, ax
007B                    f_end:
007B  B4 4C                     mov ah, 4ch
007D  CD 21                     int 21h
007F                    Main  ENDP
007F                    CODE        ENDS
                              END Main
Microsoft (R) Macro Assembler Version 5.10          11/12/20 10:39:1
                                    Symbols-1
Segments and Groups:

            N a m e         Length         AlignCombine Class

ASTACK . . . . . .          0040  PARA        STACK
CODE . . . . . .            007F  PARA        NONE
DATA . . . . . .            000E  PARA        NONE

Symbols:

            N a m e         Type    Value Attr

A  . . . . . .              L WORD    0000  DATA
```

```
B . . . . . .      L WORD    0002  DATA

F_END . . . . . .        L NEAR    007B  CODE

I . . . . . .      L WORD    0004  DATA
I1 . . . . . .     L WORD    0008  DATA
I1_1 . . . . . .   L NEAR    0035  CODE
I1_2 . . . . . .   L NEAR    001F  CODE
I2 . . . . . .     L WORD    000A  DATA
I2_1 . . . . . .   L NEAR    002A  CODE
I2_2 . . . . . .   L NEAR    0005  CODE
I3 . . . . . .     L NEAR    003F  CODE
I3_1 . . . . . .   L NEAR    005C  CODE
I3_CMP_2 . . . . . .       L NEAR    006A  CODE
I3_RES . . . . . .         L NEAR    0078  CODE

K . . . . . .      L WORD    0006  DATA

MAIN . . . . . .         F PROC    0000  CODE        Length = 007F

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

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


    78 Source  Lines
    78 Total   Lines
    25 Symbols
```

47962 + 459298 Bytes symbol space free

0 Warning Errors

0 Severe Errors