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

Студентка гр. 0383 _____ Куртова К. 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),

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 12:

$$f2 = \begin{cases} -(4i+3), a > b \\ 6i - 10, a \leq b \end{cases}$$

$$f7 = \begin{cases} -(4i-5), a > b \\ 10 - 3i, a \leq b \end{cases}$$

$$f4 = \begin{cases} \min(|i_1 - i_2|, 2), k < 0 \\ \max(-6, -i_2), k \geq 0 \end{cases}$$

Ход работы.

Для программы прописано три сегмента — сегмент стека, сегмент данных, в котором хранится информация о переменных a, b, I, k, i_1, i_2, res, для каждого из которых выделено слово (word) в памяти. Описан сегмент кода, в котором находится головная процедура Main, в которой и производятся основные вычисления.

В ходе выполнения работы были использованы условные (jmp) и безусловные (jle, lge, jl) переходы. Вычисление функций было выполнено без использования отдельных процедур, только с использованием переходов, которые переходят к указанным лейблам. Код программы представлен в приложении А, листинг программы представлен в приложении Б.

Тестирование программы.

В таблице 1 представлен результат тестирования программы.

Таблица 1 — Результат тестирования программы

Входные данные	Результирующие данные	Проверка
a = 5 b = 3 i = 2 k = -1	i1 = FFF5 = -11 i2 = FFFD = -3 res = 0002 = 2	Верно. i1 = -(8+3) = -11 i2 = -(8-5) = -3 |-11 + 3| > 2 => res = 2
a = 3 b = 5 i = 4 k = -1	i1 = 000E = 14 i2 = i2 = FFFE = -2 res = 0002 = 2	Верно. i1 = 24 – 10 = 14 i2 = 10 – 12 = -2 |14 + 2| > 2 => res = 2
a = 3 b = 5 i = 4 k = 1	i1 = 000E = 14 i2 = i2 = FFFE = -2 res = 0002 = 2	Верно. i1 = 24 – 10 = 14 i2 = 10 – 12 = -2 -6 < 2 => res = 2
a = 3 b = -5 i = -2 k = 1	i1 = 0005 = 5 i2 = 000D = 13 res = FFFA = -6	Верно. i1 = -(-8 + 3) = 5 i2 = -(-8 -5) = 13 -6 > -13 => res = -6

Выводы.

В ходе лабораторной работы были изучены представление и обработка целых чисел. Также были рассмотрены условные и безусловные переходы, и с их помощью была разработана программа, которая по заданным значениям переменных вычисляет значение нескольких функций.

ПРИЛОЖЕНИЕ А

Текст разработанной программы lab3.txt

```
EOL EQU '$

;----------Стек программы----------
AStack SEGMENT STACK
DW 32 DUP(?)
AStack ENDS

;----------Данные программы--------
DATA SEGMENT
buffer DB 128 DUP(?)
a DW 3
b DW -5
i DW -2
k DW 1
i1 DW ?
i2 DW ?
res DW ?
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 ax, i
        mov bx, a
        cmp bx, b    ; Сравниваем a и b
        jle case2    ; Если a <= b, перейти к case2
case1:
        ; Вычисление f2
        shl ax, 1
        shl ax, 1    ; ax = 4i
        neg ax               ; ax = -4i
        sub ax, 3    ; ax = f2(i) = -4i - 3
        mov i1, ax   ; i1 = -4i-3
```

```
        ; Вычисление f7
        add ax, 8      ; ax = f7(i) = -4i + 5
        mov i2, ax     ; i2 = -4i + 5
        jmp f4_
case2:
        ; Вычисление f7
        mov bx, ax   ; bx = i
        shl ax, 1      ; ax = 2i
        add ax, bx     ; ax = 3i
        mov bx, 10
        sub bx, ax     ; bx = 10 - 3i
        mov i2, bx
        ; Вычисление f2
        shl bx, 1      ; bx = 20 - 6i
        neg bx                 ; bx = 6i - 20
        add bx, 10     ; bx = 6i - 10
        mov i1, bx

f4_:
        mov bx, k
        cmp bx, 0
        jge f4_case2           ; Если k >= 0, перейти к f3_case2

f4_case1:
        mov ax, i1
        sub ax, i2     ; i1-i2
        cmp ax, 0
        jl abs         ; Если i1 - i2 < 0, найдём модуль выражения
        jmp min                ; Больше 0, перейти к поиску минимального числа в
паре
abs:
        neg ax
min:
        mov bx, 2
        cmp ax, 2
        jle absmin     ; Если |i1 - i2| <= 2, перейти к absmin
twomin:
        mov cx, 2
        jmp f4_end
absmin:
        mov cx, ax
        jmp f4_end

f4_case2:
```

```
        mov ax, i2
        neg ax
        cmp ax, -6
        jge i2max     ; Если -i2 >= -6, перейти к i2max
neg6max:
        mov cx, -6
        jmp f4_end
i2max:
        mov cx, ax
f4_end:
        mov res, cx
        ret

Main ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ Б

Файл листинга программы lab3.lst


```
= '$                    EOL EQU '$

                        ;----------PЎC,PµPє PïCЂPsPiCЂP°PjPjC‹---------
                        -
0000                    AStack SEGMENT STACK
0000  0020[                     DW 32 DUP(?)
        ????
                    ]

0040                    AStack ENDS

                        ;----------P”P°PSPSC‹Pµ PïCЂPsPiCЂP°PjPjC‹-----
                        ---
0000                    DATA SEGMENT
0000  0080[                     buffer DB 128 DUP(?)
        ??
                    ]

0080  0003              a DW 3
0082  FFFB                      b DW -5
0084  FFFE              i DW -2
0086  0001              k DW 1
0088  0000              i1 DW ?
008A  0000              i2 DW ?
008C  0000              res DW ?
008E                    DATA ENDS
```

```
                        ;----------PЛьPsPr PїCЂPsPiCЂP°PjPjC‹----------
0000                    CODE SEGMENT
                        ASSUME CS:CODE, DS:DATA, SS:AStack

                        ;--------P“PsP»PsPIPSP°CЦ PїCЂPsC†PµPrCґCЂP°-
--
                        -----
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 0084 R                  mov ax, i
000C  8B 1E 0080 R                    mov bx, a
0010  3B 1E 0082 R                    cmp bx, b    ; PЎCЂP°PIPSPëPIP°PµPj
                        a Pë b
0014  7E 15                           jle case2    ; P•CЃP»Pë a <= b, PїPµ
                        CЂPµP№C,Pë Pє case2
0016                    case1:
                              ; P’C‹C‡PëCЃP»PµPSPëPµ f2
0016  D1 E0                           shl ax, 1
0018  D1 E0                           shl ax, 1    ; ax = 4i
001A  F7 D8                           neg ax              ; ax = -4i
001C  2D 0003                         sub ax, 3    ; ax = f2(i) = -4i - 3
001F  A3 0088 R                  mov i1, ax   ; i1 = -4i-3
                              ; P’C‹C‡PëCЃP»PµPSPëPµ f7
 Microsoft (R) Macro Assembler Version 5.10                11/4/21 16:31:48
                                       Page     1-2


0022  05 0008                         add ax, 8    ; ax = f7(i) = -4i + 5
0025  A3 008A R                  mov i2, ax   ; i2 = -4i + 5
```

```
0028  EB 1B 90                  jmp f4_
002B                    case2:
                        ; P'C‹C‡PëCЃP»PµPSPëPµ f7
002B  8B D8                     mov bx, ax  ; bx = i
002D  D1 E0                     shl ax, 1   ; ax = 2i
002F  03 C3                     add ax, bx  ; ax = 3i
0031  BB 000A                   mov bx, 10
0034  2B D8                     sub bx, ax  ; bx = 10 - 3i
0036  89 1E 008A R              mov i2, bx
                        ; P'C‹C‡PëCЃP»PµPSPëPµ f2
003A  D1 E3                     shl bx, 1   ; bx = 20 - 6i
003C  F7 DB                     neg bx              ; bx = 6i - 20
003E  83 C3 0A                  add bx, 10  ; bx = 6i - 10
0041  89 1E 0088 R              mov i1, bx

0045                    f4_:
0045  8B 1E 0086 R              mov bx, k
0049  83 FB 00                  cmp bx, 0
004C  7D 24                     jge f4_case2        ;  P•CЃP»Pë  k  >=  0,
PїPµ
                        CЂPµP№C,Pë Pє f3_case2

004E                    f4_case1:
004E  A1 0088 R                 mov ax, i1
0051  2B 06 008A R              sub ax, i2  ; i1-i2
0055  3D 0000                   cmp ax, 0
0058  7C 03                     jl abs      ; P•CЃP»Pë i1 - i2 < 0,
                         PSP°P№PrC'Pj                   PjPsPrCґP»CЊ
PIC‹CЂP°P¶PµPSPëCЏ
005A  EB 03 90                  jmp min             ;  P'PsP»CЊC€Pµ  0,
PїPµC
                        ЂPµP№C,Pë          Pє          PїPsPëCЃPєCґ
PjPëPSPëPjP°P»CЊPSPsP
                        iPs C‡PëCЃP»P° PI PїP°CЂPµ
```

```
005D                    abs:
005D  F7 D8                             neg ax
005F                    min:
005F  BB 0002                           mov bx, 2
0062  3D 0002                           cmp ax, 2
0065  7E 06                             jle absmin    ; P•CЃP»Pë |i1 - i2| <=
                         2, PїPµCЂPµP№C,Pë Pє absmin
0067                    twomin:
0067  B9 0002                           mov cx, 2
006A  EB 18 90                          jmp f4_end
006D                    absmin:
006D  8B C8                             mov cx, ax
006F  EB 13 90                          jmp f4_end

0072                    f4_case2:
0072  A1 008A R                 mov ax, i2
0075  F7 D8                             neg ax
0077  3D FFFA                           cmp ax, -6
007A  7D 06                             jge i2max     ; P•CЃP»Pë -i2 >= -6, P
                        їPµCЂPµP№C,Pë Pє i2max
007C                    neg6max:
007C  B9 FFFA                           mov cx, -6
 Microsoft (R) Macro Assembler Version 5.10                 11/4/21 16:31:48
                                        Page     1-3


007F  EB 03 90                          jmp f4_end
0082                    i2max:
0082  8B C8                             mov cx, ax
0084                    f4_end:
0084  89 0E 008C R                      mov res, cx
0088  CB                        ret

0089                    Main ENDP
```

```
0089                    CODE ENDS
                        END Main
Microsoft (R) Macro Assembler Version 5.10          11/4/21 16:31:48
                                                    Symbols-1


Segments and Groups:

                N a m e         Length   Align   Combine Class

ASTACK . . . . . . . . . . . .  0040    PARA    STACK
CODE . . . . . . . . . . . . .  0089    PARA    NONE
DATA . . . . . . . . . . . . .  008E    PARA    NONE

Symbols:

                N a m e         Type     Value   Attr

A  . . . . . . . . . . . . . .  L WORD   0080    DATA
ABS  . . . . . . . . . . . . .  L NEAR   005D    CODE
ABSMIN . . . . . . . . . . . .  L NEAR   006D    CODE

B  . . . . . . . . . . . . . .  L WORD   0082    DATA
BUFFER . . . . . . . . . . . .  L BYTE   0000    DATA    Length = 0080

CASE1  . . . . . . . . . . . .  L NEAR   0016    CODE
CASE2  . . . . . . . . . . . .  L NEAR   002B    CODE

EOL  . . . . . . . . . . . . .  TEXT  '$

F4_  . . . . . . . . . . . . .  L NEAR   0045    CODE
F4_CASE1 . . . . . . . . . . .  L NEAR   004E    CODE
F4_CASE2 . . . . . . . . . . .  L NEAR   0072    CODE
F4_END . . . . . . . . . . . .  L NEAR   0084    CODE
```

```
I . . . . . .      L WORD    0084  DATA
I1 . . . . . .     L WORD    0088  DATA
I2 . . . . . .     L WORD    008A  DATA
I2MAX . . . . . .        L NEAR    0082  CODE

K . . . . . .      L WORD    0086  DATA

MAIN . . . . . .          F PROC    0000  CODE        Length = 0089
MIN . . . . . .    L NEAR    005F  CODE

NEG6MAX . . . . . .       L NEAR    007C  CODE

RES . . . . . .    L WORD    008C  DATA

TWOMIN . . . . . .        L NEAR    0067  CODE

@CPU . . . . . .          TEXT  0101h
@FILENAME . . . . . .     TEXT  lab3_fix
@VERSION . . . . . .      TEXT  510
  Microsoft (R) Macro Assembler Version 5.10                11/4/21 16:31:48

                                                            Symbols-2
   101 Source  Lines
   101 Total   Lines
    30 Symbols

 47942 + 459318 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```