МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Архитектура ЭВМ и систем»
Тема: Представление и обработка целых чисел. Организация ветвящихся процессов.

Студент гр. 9383 _____ Моисейченко К.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер. Научиться строить программы с условными переходами.

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.

Исходные данные.

Вариант 11

$$f1 = \begin{cases} -(4 * i + 3), & \text{при } a > b \\ 6 * i - 10, & \text{при } a <= b \end{cases}$$

$$f2 = \begin{cases} 2 * (i + 1) - 4, & \text{при } a > b \\ 5 – 3 * (i + 1), & \text{при } a <= b \end{cases}$$

$$f5 = \begin{cases} \min(|i1|, 6), & \text{при } k = 0 \\ |i1| + |i2|, & \text{при } k != 0 \end{cases}$$

Ход работы.

Была разработана программа, которая вычисляет значение функции по заданным целочисленным параметрам.

Исходные и выходные данные записываются в сегмент данных. Правильность записи была проверена с помощью отладчика.

Для подсчета значений функции были использованы следующие операнды:

add – для суммирования

sub – для вычитания

shl – для логического сдвига влево, что равнозначно умножению на два

Результаты записывались по заранее заданным адресам переменных i1, i2 и res.

Для реализации условных переходов были использованы следующие операнды:

cmp – для сравнения двух чисел. При использовании данного операнда его результат записывается с помощью выставления соответствующих флагов.
jg – условный переход, срабатывающий, если левый аргумент выражения в cmp был больше второго.
jl – условный переход, срабатывающий, если левый аргумент выражения в cmp был меньше второго.
jmp – безусловный переход. Используется, если для перехода к следующему адресу не нужно делать дополнительных проверок.
Исходный код и листинг программы представлены в приложении А.

Тестирование.

1. a = 1, b = 2, i = 3, k = 4 => i1 = 8, i2 = -7, res = 15
2. a = 2, b = 1, i = 3, k = 4 => i1 = -15, i2 = 4, res = 19
3. a = 1, b = 2, i = 3, k = 0 => i1 = 8, i2 = -7, res = 6
4. a = 2, b = 1, i = -1, k = 0 => i1 = 1, i2 = 5, res = 1

Выводы.

Было изучено представление и обработка целых чисел на языке Ассемблер. Была построена программа с условными переходами, которая считает значения функций с заданными целочисленными параметрами.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lr3.asm:

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

DATA SEGMENT
a     DW    1
b     DW    2
i     DW    3
k     DW    4
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

f2:
mov ax, a
cmp ax, b
jg f2_1           ; a > b
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, 10       ; ax = 6*i - 10
mov i1, ax

neg ax             ; ax = -6*i + 10
```

```
shr ax, 1           ; ax = -3*i + 5
sub ax, 3           ; ax = 5 -3*(i + 1)
mov i2, ax
jmp f5

f2_1:
mov ax, i
shl ax, 1           ; ax = 2*i
shl ax, 1           ; ax = 4*i
add ax, 3           ; ax = 4*i + 3
neg ax              ; ax = -(4*i + 3)
mov i1, ax

neg ax              ; ax = 4*i + 3
sub ax, 7           ; ax = 4*i - 4
shr ax, 1           ; ax = 2*(i + 1) - 4
mov i2, ax
jmp f5

f5:
mov bx, i1
cmp bx, 0
jl f5_neg
jmp f5_1

f5_neg:
neg bx
jmp f5_1

f5_1:
mov ax, k
cmp ax, 0
je f5_cmp_6         ;k = 0
jmp f5_sum          ;k != 0

f5_cmp_6:
cmp bx, 6
jl res_i1
```

```
    jmp res_6

    res_i1:
    mov res, bx
    jmp f_end

    res_6:
    mov res, 6
    jmp f_end

    f5_sum:
    mov cx, i2
    cmp cx, 0
    jl f5_neg_sum
    jmp f5_res_sum

    f5_neg_sum:
    neg cx
    jmp f5_res_sum

    f5_res_sum:
    mov ax, bx
    add ax, cx
    mov res, ax
    jmp f_end

    f_end:
    mov ah, 4ch
    int 21h

    Main  ENDP
    CODE  ENDS
    END Main
```

Файл lr3.lst:

```
    Microsoft (R) Macro Assembler Version 5.10                  12/17/20
11:05:2
```

```
0000                    AStack SEGMENT STACK
0000  0020[                       DW 32 DUP(?)
   ????
 ]

0040                    AStack ENDS

0000                    DATA SEGMENT
0000  0001              a     DW    1
0002  0002              b     DW    2
0004  0003              i     DW    3
0006  0004              k     DW    4
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                    f2:
0005  A1 0000 R                 mov ax, a
0008  3B 06 0002 R                  cmp ax, b
000C  7F 1E                         jg f2_1           ; a > b
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 000A                       sub ax, 10       ; ax = 6*i - 10
001C  A3 0008 R                 mov i1, ax

001F  F7 D8                         neg ax              ; ax = -6*i +
10
0021  D1 E8                         shr ax, 1           ; ax = -3*i +
5
0023  2D 0003                       sub ax, 3          ; ax = 5 -3*(i
+ 1)
0026  A3 000A R                 mov i2, ax
0029  EB 1D 90                     jmp f5

002C                    f2_1:
002C  A1 0004 R                 mov ax, i
002F  D1 E0                         shl ax, 1          ; ax = 2*i
0031  D1 E0                         shl ax, 1          ; ax = 4*i
0033  05 0003                       add ax, 3          ; ax = 4*i + 3
0036  F7 D8                         neg ax             ; ax = -(4*i +
3)
0038  A3 0008 R                 mov i1, ax
```

```
003B  F7 D8                     neg ax              ; ax = 4*i + 3
003D  2D 0007                   sub ax, 7           ; ax = 4*i - 4
0040  D1 E8                     shr ax, 1            ; ax = 2*(i +
1) - 4
0042  A3 000A R           mov i2, ax
Microsoft (R) Macro Assembler Version 5.10                12/17/20
11:05:2
                                                             Page
1-2


0045  EB 01 90                  jmp f5

0048                    f5:
0048  8B 1E 0008 R              mov bx, i1
004C  83 FB 00                  cmp bx, 0
004F  7C 03                     jl f5_neg
0051  EB 06 90                  jmp f5_1

0054                    f5_neg:
0054  F7 DB                     neg bx
0056  EB 01 90                  jmp f5_1

0059                    f5_1:
0059  A1 0006 R                 mov ax, k
005C  3D 0000                   cmp ax, 0
005F  74 03                     je f5_cmp_6         ;k = 0
0061  EB 19 90                  jmp f5_sum          ;k != 0

0064                    f5_cmp_6:
0064  83 FB 06                  cmp bx, 6
0067  7C 03                     jl res_i1
0069  EB 08 90                  jmp res_6

006C                    res_i1:
006C  89 1E 000C R              mov res, bx
0070  EB 25 90                  jmp f_end

0073                    res_6:
0073  C7 06 000C R 0006         mov res, 6
0079  EB 1C 90                  jmp f_end

007C                    f5_sum:
007C  8B 0E 000A R              mov cx, i2
0080  83 F9 00                  cmp cx, 0
0083  7C 03                     jl f5_neg_sum
0085  EB 06 90                  jmp f5_res_sum

0088                    f5_neg_sum:
0088  F7 D9                     neg cx
008A  EB 01 90                  jmp f5_res_sum

008D                    f5_res_sum:
008D  8B C3                     mov ax, bx
008F  03 C1                     add ax, cx
0091  A3 000C R                 mov res, ax
```

```
      0094  EB 01 90                  jmp f_end

      0097                    f_end:
      0097  B4 4C                     mov ah, 4ch
      0099  CD 21                     int 21h

      009B                    Main  ENDP
      009B                    CODE  ENDS
      END Main
      Microsoft (R) Macro Assembler Version 5.10                  12/17/20
11:05:2

Symbols-1


      Segments and Groups:

                      N a m e         Length      Align     Combine
Class

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

      Symbols:

                      N a m e         Type   Value      Attr

      A  . . . . . .                  L WORD     0000  DATA

      B  . . . . . .                  L WORD     0002  DATA

      F2 . . . . . .                  L NEAR     0005  CODE
      F2_1 . . . . . .                L NEAR     002C  CODE
      F5 . . . . . .                  L NEAR     0048  CODE
      F5_1 . . . . . .                L NEAR     0059  CODE
      F5_CMP_6 . . . . . .            L NEAR     0064  CODE
      F5_NEG . . . . . .              L NEAR     0054  CODE
      F5_NEG_SUM . . . . . .          L NEAR     0088  CODE
      F5_RES_SUM . . . . . .          L NEAR     008D  CODE
      F5_SUM . . . . . .              L NEAR     007C  CODE
      F_END  . . . . . .              L NEAR     0097  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
= 009B

      RES  . . . . . .                L WORD     000C  DATA
      RES_6  . . . . . .              L NEAR     0073  CODE
      RES_I1 . . . . . .              L NEAR     006C  CODE
```

```
     @CPU . . . . . .     TEXT  0101h
     @FILENAME  . . . . . .     TEXT  lr3
     @VERSION . . . . . .     TEXT  510

     Microsoft (R) Macro Assembler Version 5.10                  12/17/20
11:05:2


Symbols-2




          105 Source  Lines
          105 Total   Lines
           28 Symbols

        48068 + 459192 Bytes symbol space free

            0 Warning Errors
            0 Severe  Errors
```