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

/ - (4 * i + 3), при a > b

f1 = <

\ 6 * i - 10, при a <= b

/ 2 * (i + 1) - 4, при a > b

f2 = <

\ 5 – 3 * (i + 1), при a <= b

/ min(|i1|, 6), при k = 0

f5 = <

\ |i1| + |i2|, при k != 0

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

    mov ax, i
```

```
    add ax, 1           ; ax = i + 1
    mov bx, ax          ; bx = i + 1
    shl ax, 1           ; ax = 2*(i + 1)
    add ax, bx          ; ax = 3*(i + 1)
    mov bx, 5
    sub bx, ax          ; bx = 5 - 3*(1 + i)
    mov i2, bx
    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

  mov ax, i
  add ax, 1           ; ax = i + 1
  shl ax, 1           ; ax = 2*(i + 1)
    sub ax, 4           ; 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
03:00:4
                                                                      Page
1-1


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 29                     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  A1 0004 R                 mov ax, i
0022  05 0001                     add ax, 1           ; ax = i + 1
0025  8B D8                     mov bx, ax          ; bx = i + 1
0027  D1 E0                     shl ax, 1           ; ax = 2*(i + 1)
0029  03 C3                     add ax, bx          ; ax = 3*(i + 1)
002B  BB 0005                     mov bx, 5
002E  2B D8                     sub bx, ax          ; bx = 5 - 3*(1 + i)
0030  89 1E 000A R                mov i2, bx
0034  EB 21 90                    jmp f5

0037                    f2_1:
0037  A1 0004 R                 mov ax, i
```

```
003A  D1 E0                shl ax, 1          ; ax = 2*i
003C  D1 E0                shl ax, 1          ; ax = 4*i
003E  05 0003                add ax, 3          ; ax = 4*i + 3
0041  F7 D8                neg ax             ; ax = -(4*i + 3)
0043  A3 0008 R            mov i1, ax

Microsoft (R) Macro Assembler Version 5.10                12/17/20
03:00:4
                                                             Page
1-2


0046  A1 0004 R          mov ax, i
0049  05 0001              add ax, 1          ; ax = i + 1
004C  D1 E0              shl ax, 1          ; ax = 2*(i + 1)
004E  2D 0004                sub ax, 4          ; ax = 2*(i + 1) -
4
0051  A3 000A R          mov i2, ax
0054  EB 01 90             jmp f5

0057                  f5:
0057  8B 1E 0008 R           mov bx, i1
005B  83 FB 00                cmp bx, 0
005E  7C 03                 jl f5_neg
0060  EB 06 90                jmp f5_1

0063                  f5_neg:
0063  F7 DB                neg bx
0065  EB 01 90              jmp f5_1

0068                  f5_1:
0068  A1 0006 R             mov ax, k
006B  3D 0000                 cmp ax, 0
006E  74 03                 je f5_cmp_6        ;k = 0
0070  EB 19 90                jmp f5_sum         ;k != 0

0073                  f5_cmp_6:
0073  83 FB 06                cmp bx, 6
0076  7C 03                jl res_i1
0078  EB 08 90                jmp res_6

007B                  res_i1:
007B  89 1E 000C R            mov res, bx
007F  EB 25 90                jmp f_end

0082                  res_6:
0082  C7 06 000C R 0006       mov res, 6
0088  EB 1C 90                jmp f_end

008B                  f5_sum:
008B  8B 0E 000A R            mov cx, i2
008F  83 F9 00                cmp cx, 0
0092  7C 03                jl f5_neg_sum
0094  EB 06 90                jmp f5_res_sum

0097                  f5_neg_sum:
0097  F7 D9                neg cx
```

```
0099  EB 01 90              jmp f5_res_sum

009C                  f5_res_sum:
009C  8B C3                 mov ax, bx
009E  03 C1                 add ax, cx
00A0  A3 000C R             mov res, ax
00A3  EB 01 90                 jmp f_end

00A6                  f_end:
00A6  B4 4C                    mov ah, 4ch
Microsoft (R) Macro Assembler Version 5.10                  12/17/20
03:00:4
                                                              Page
1-3


00A8  CD 21                   int 21h

00AA                  Main    ENDP
00AA                  CODE    ENDS
                      END Main
Microsoft (R) Macro Assembler Version 5.10                  12/17/20
03:00:4

Symbols-1


      Segments and Groups:

                N a m e                 Length     Align    Combine
Class

      ASTACK . . . . . .              0040    PARA    STACK
      CODE . . . . . .                00AA    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    0037    CODE
      F5 . . . . . .                  L NEAR    0057    CODE
      F5_1 . . . . . .                L NEAR    0068    CODE
      F5_CMP_6 . . . . . .            L NEAR    0073    CODE
      F5_NEG . . . . . .              L NEAR    0063    CODE
      F5_NEG_SUM . . . . . .          L NEAR    0097    CODE
      F5_RES_SUM . . . . . .          L NEAR    009C    CODE
      F5_SUM . . . . . .              L NEAR    008B    CODE
      F_END  . . . . . .              L NEAR    00A6    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 = 00AA

     RES  . . . . . .      L WORD    000C    DATA
     RES_6  . . . . . .      L NEAR    0082    CODE
     RES_I1 . . . . . .      L NEAR    007B    CODE

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

     Microsoft (R) Macro Assembler Version 5.10                12/17/20
03:00:4


Symbols-2




          110 Source  Lines
          110 Total   Lines
           28 Symbols

        48068 + 459192 Bytes symbol space free

            0 Warning Errors
            0 Severe  Errors
```