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

Студент гр. 0382 _____ Азаров М.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Разобрать и научится использовать механизм ветвления в программах на языке Ассемблер. Разработать программу на основе полученных знаний.

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.

Выполнение работы.

Функции выбранные в соответствии с вариантом:

$$f1 = \begin{cases} 15-2*i, & \text{при } a>b \\ 3*i+4, & \text{при } a<=b \end{cases}$$

Рисунок 1: функция f1

$$f2 = \begin{cases} -(4*i+3), & \text{при } a>b \\ 6*i-10, & \text{при } a<=b \end{cases}$$

Рисунок 2: функция f2

$$f1 = \begin{cases} \min(i1,i2), & \text{при } k=0 \\ \max(i1,i2), & \text{при } k/=0 \end{cases}$$

Рисунок 3: функция f3

В процессе выполнения задания была разработана программ, которая состоит из несколько частей:

1. Описание сегментов программы. В их число входят сегмент стека , сегмент данных в котором была выделена память для переменных ***a, b, i, k, i1, i2, res***.

2. Потом идет сегмент кода, в котором прописана сама программа.

3. Прописываются необходимые вещи для нормальной работы любой программы , такие как сохранение адреса начала PSP в стеке, загрузка сегментного регистра данных и т.д.

4. Затем анализируются значения ***a*** и ***b.*** Если ***a>b*** то выполняется блок программы ответственный за функции ***f1*** и ***f2*** для ***a>b***. Иначе выполняется блок для функций ***f1*** и ***f2*** при ***a<=b.*** Все это происходит в блоке начиная с метки ***f12*** до метки ***end_f12***.

5. В блоке с меткой ***regulate*** происходит определение максимального и минимального значения переменных ***i1, i2***. Максимальное значение записывается в AX, минимальное BX.

6. В блоке с меткой ***f3*** анализируется значение ***k*** и выполняется функция ***f3*** в соответствии со значением ***k***.

7. Выполняет выход из программы

Тестирование.

Результаты тестирования представлены в табл. 1. Тестирование проводилось в отладчике **AFDPRO**.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	a = 5; b = -2; i = 2; k = 0	i1 = 0B (11); i2 = FFF5 (-11); res = FFF5(-11)	Программа работает корректно
2.	a = 5; b = 2; i = -2; k = 1	i1 = 13h (19); i2 = 5; res =13h (19)	Программа работает корректно
3.	a = -5; b = 2; i = -2; k = 1	i1 = FFFE (-2); i2 = FFEA (-22); res = FFFE(-2)	Программа работает корректно

Выводы.

Был изучен механизм ветвления в программах на языке Ассемблер.

Разработана программа, выполняющая поставленную задачу , а именно 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),

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.asm

```
AStack  SEGMENT  STACK                ;из-за Атрибут комбинирования
сегментов STACK - загрузка  AStack в регистр
         DW 12 DUP('!')                  ;SS      будет      выполнена
автоматически до начала выполнения программы
AStack ENDS

DATA SEGMENT
    a Dw -5
    b Dw 2
    i Dw -2
    k Dw 1
    i1 Dw 0
    i2 Dw 0
    res Dw 0
DATA ENDS

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

Main PROC FAR
    push  DS       ;\  Сохранение адреса начала PSP в стеке
    sub   AX,AX    ; > для последующего восстановления по
    push  AX       ;/  команде ret, завершающей процедуру.
    mov   AX,DATA             ; Загрузка сегментного
    mov   DS,AX               ; регистра данных.



f12:
;if a > b
    mov   ax, i ;ax = i
     shl   ax, 1 ;ax = 2i

     mov   bx, a
     cmp   bx, b
     JLE else_f12 ;если a>=b скачок

then_f12:
     mov i1, 15
     sub  i1, ax

     mov i2, -3
     shl   ax, 1 ;ax = 4i
     sub i2, ax
     jmp end_f2

else_f12:
     add   ax, i ;ax = 3i
     mov   i1, ax
     add   i1, 4
```

```
        mov   i2, -10
        shl   ax, 1 ;ax = 6i
        add   i2, ax
    end_f2:



    regulate:
    ;if i1 >= i2
        mov   ax, i1
        mov   bx, i2
        cmp   ax, bx
        JL    swap ;  если i1<i2 скачок
        JGE   end_reg ;если i1>=i2 скачок
    swap:
        mov   bx, i1
        mov   ax, i2
    end_reg:
    ;теперь ax>bx


    f3:
    ;if k==0
        cmp   k, 0h
        JNE else_f3 ; если k!=0 скачок

    then_f3:
        mov res, bx
        jmp end_f3

    else_f3:
        mov res, ax
    end_f3:

        ret

    Main ENDP
    CODE ENDS
        END Main
```

Название файла: list.lst

```
    #Microsoft     (R)     Macro     Assembler     Version     5.10
11/1/21 22:23:31
                                                                Page
1-1


     0000                AStack SEGMENT STACK          ;из-за Атрибут
ком
                         бинирования   сегментов   STACK   -   загрузка
AStack
```

```
                        в регистр
      0000  000C[                DW 12 DUP('!')                ;SS
бу
                        дет выполнена автоматически до начала
выполнени
                        я программы
             0021
                    ]

      0018              AStack ENDS

      0000              DATA SEGMENT
      0000  0005                a Dw 5
      0002  0002                b Dw 2
      0004  FFFE                i Dw -2
      0006  0001                k Dw 1
      0008  0000                i1 Dw 0
      000A  0000                i2 Dw 0
      000C  0000                res Dw 0
      000E              DATA ENDS

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

      0000              Main PROC FAR
      0000  1E                  push  DS          ;\  Сохранение адреса
начала
                         PSP в стеке
      0001  2B C0                     sub     AX,AX       ;  >  для
последующего восстан
                        овления по
      0003  50                   push   AX          ;/   команде ret,
завершающей
                         процедуру.
      0004  B8 ---- R           mov    AX,DATA              ; Загрузка
сегмен
                        тного
      0007  8E D8                     mov    DS,AX                  ;
регистра данных
                        .



      0009              f12:
                        ;if a > b
      0009  A1 0004 R           mov   ax, i ;ax = i
      000C  D1 E0                    shl   ax, 1 ;ax = 2i

      000E  8B 1E 0000 R             mov   bx, a
      0012  3B 1E 0002 R             cmp   bx, b
      0016  7E 19                    JLE else_f12 ;если a>=b скачок

      0018              then_f12:
      0018  C7 06 0008 R 000F        mov i1, 15
      001E  29 06 0008 R             sub  i1, ax
```

```
0022  C7 06 000A R FFFD        mov i2, -3
0028  D1 E0                    shl   ax, 1 ;ax = 4i
002A  29 06 000A R             sub i2, ax

     #Microsoft      (R)      Macro      Assembler      Version      5.10
11/1/21 22:23:31                                                      Page
1-2


002E  EB 19 90                 jmp end_f2

0031                 else_f12:
0031  03 06 0004 R             add   ax, i ;ax = 3i
0035  A3 0008 R           mov   i1, ax
0038  83 06 0008 R 04               add   i1, 4

003D  C7 06 000A R FFF6        mov   i2, -10
0043  D1 E0                    shl   ax, 1 ;ax = 6i
0045  01 06 000A R             add   i2, ax
0049                 end_f2:



0049                 regulate:
                     ;if i1 >= i2
0049  A1 0008 R           mov   ax, i1
004C  8B 1E 000A R             mov   bx, i2
0050  3B C3                    cmp   ax, bx
0052  7C 02                    JL    swap ;  если i1<i2 скачок
0054  7D 07                    JGE   end_reg ;если i1>=i2 скачок
0056                 swap:
0056  8B 1E 0008 R             mov   bx, i1
005A  A1 000A R           mov   ax, i2
005D                 end_reg:
                     ;теперь ax>bx


005D                 f3:
                     ;if k==0
005D  83 3E 0006 R 00               cmp   k, 0h
0062  75 07                    JNE else_f3 ; если k!=0 скачок

0064                 then_f3:
0064  89 1E 000C R             mov res, bx
0068  EB 04 90                 jmp end_f3

006B                 else_f3:
006B  A3 000C R           mov res, ax
006E                 end_f3:

006E  CB                       ret

006F                 Main ENDP
```

```
006F                    CODE ENDS
                            END Main

     #Microsoft      (R)      Macro      Assembler      Version      5.10
11/1/21 22:23:31
                                                                      Symb
ols-1


     Segments and Groups:

                    N a m e         Length      Align     Combine
Class

     ASTACK . . . . . .              0018  PARA  STACK
     CODE . . . . . .                006F  PARA  NONE
     DATA . . . . . .                000E  PARA  NONE

     Symbols:

                    N a m e         Type   Value      Attr

     A  . . . . . .                  L WORD     0000  DATA

     B  . . . . . .                  L WORD     0002  DATA

     ELSE_F12 . . . . . .            L NEAR     0031  CODE
     ELSE_F3  . . . . . .            L NEAR     006B  CODE
     END_F2 . . . . . .              L NEAR     0049  CODE
     END_F3 . . . . . .              L NEAR     006E  CODE
     END_REG  . . . . . .            L NEAR     005D  CODE

     F12  . . . . . .                L NEAR     0009  CODE
     F3 . . . . . .                  L NEAR     005D  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
= 006F

     REGULATE . . . . . .            L NEAR     0049  CODE
     RES  . . . . . .                L WORD     000C  DATA

     SWAP . . . . . .                L NEAR     0056  CODE

     THEN_F12 . . . . . .            L NEAR     0018  CODE
     THEN_F3  . . . . . .            L NEAR     0064  CODE

     @CPU . . . . . .                TEXT  0101h
     @FILENAME  . . . . . .          TEXT  LB3
     @VERSION . . . . . .            TEXT  510
```

```
          88 Source  Lines
          88 Total   Lines
          27 Symbols

       47986 + 461321 Bytes symbol space free

           0 Warning Errors
           0 Severe  Errors
```