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.
Вариант 15.

Студент гр. 0382 _____ Санников В.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Разработать программу, которая по заданным целочисленным параметрам вычисляет значение некоторой функции по данным переменным и работает с ветвящимися процессами.

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.

Вариант №15:

/ 7 - 4*i , при a>b

f3 = <

\ 8 -6*i , при a<=b

/ 20 - 4*i , при a>b

f5 = <

\ -(6*I - 6), при a<=b

/ |i1 + i2|, при k=0

f3 = <

\ min(i1,i2), при k/=0

Ход работы:

Создается три сегмента: Astack – сегмент стека, DATA – сегмент данных и CODE – сегмент кода. С помощью директивы ASSUME, метки сегментов записаны в соответствующие регистры. В сегменте данных объявляются переменные: a, b, I, k, i1, i2, res. В сегменте кода создана процедура Main. В данной процедуре происходит основная вычислительная часть. Отработав, программа завершается после операции ret.

В программе были использованы переходы, чтобы избежать использование процедур. См таблицу 1.

Таблица 1 — Используемые переходы.

Переход	Описание
JMP	Безусловный переход к метке. Нужен в программе для избежания условия a<=b, также используется для перехода на метку завершения программы, когда значение найдено.
JG	Условный переход. Используется в программе для перехода на метку calc_1 при a > b
JGE	Условный переход. Используется для перехода на метку завершения программы при условии что ax >= 0
JLE	Также условный переход, необходимый для перехода на метку min_1 при условии, что ax <= cx
JE	Условный переход для проверки условия, что k = 0. Используется для перехода на метку calc3_1.

Исходный код программы см. в приложении А.

Файл диагностических сообщений см. в приложении Б.

Тестирование:

Для проверки работоспособности программы были проведены тесты, см. Таблицу 2.

Таблица 2 — Тесты

Номер теста	a	b	i	k
1	1	10	2	0
2	10	1	2	0
3	5	5	3	1

Результаты тестирования представлены в Таблице 3.

Таблица 3 — Результаты тестирования.

Номер теста	i1	i2	res	Оценка результата
1	FFFC (-4)	FAFF (-6)	000A (10)	Верно
2	FFFF (-1)	000C (12)	000B (11)	Корректно
3	FFF6 (-10)	FFF4 (-12)	FFF4 (-12)	Верно

Выводы.

В результате лабораторной работы было изучено представление, обработка целых чисел и организация ветвящихся процессов. Разработана программа, выполняющая вычисления функций по условию из задания.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.asm

```
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

DATA SEGMENT
     a DW 0
     b DW 0
     i DW 0
     k DW 0
     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
     sub AX, AX
     push AX
     mov AX, DATA
     mov DS, AX

f1_2:
     mov ax, i
     shl ax, 1 ;ax = 2i
     shl ax, 1 ;ax = 4i
     mov bx, a
     cmp bx, b
     jg calc_1

calc_2: ;if(a <= b)
     mov bx, ax ;bx = ax = 4i
     shr bx, 1 ;bx = 2i
     add ax, bx ;ax = 6i
     neg ax ;ax = -6i
     mov cx, ax ;cx = ax = -6i
     add cx, 6h ;cx = (6 - 6i)
     add ax, 8h ;ax = (8 - 6i)
     jmp res_f1_f2

calc_1: ;if(a > b)
     neg ax ;ax = -4i
     mov cx, ax ;cx = ax = -4i
     add ax, 7h ;ax = (7 - 4i)
```

```
      add cx, 14h ;cx = (20 - 4i)

res_f1_f2:
      mov i1, ax
      mov i2, cx

f3:
      mov bx, k
      cmp bx, 0h
      je calc3_1

calc3_2: ;if(k \= 0)
      cmp ax, cx
      jle min_1

min_2: ;if(ax > cx)
      mov res, cx
      jmp exit

min_1: ;if(ax <= cx)
      mov res, ax
      jmp exit

calc3_1: ;if(k == 0)
      add ax, cx
      mov res, ax
      cmp ax, 0h
      jge exit

abs:
      neg ax
      mov res, ax

exit:
      ret

Main ENDP
CODE ENDS
 END Main
```

ПРИЛОЖЕНИЕ Б

ФАЙЛЫ ДИАГНОСТИЧЕСКИХ СООБЩЕНИЙ

Название файла: main.lst

```
#Microsoft (R) Macro Assembler Version 5.10
11/10/21 23:26:3

Page     1-1


 0000                   AStack SEGMENT STACK
 0000  000C[                     DW 12 DUP(?)
        ????
                ]

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  0000                      a DW 0
 0002  0000                      b DW 0
 0004  0000                      i DW 0
 0006  0000                      k DW 0
 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
 0001  2B C0                         sub AX, AX
 0003  50                       push AX
 0004  B8 ---- R                mov AX, DATA
 0007  8E D8                         mov DS, AX

 0009                   f1_2:
 0009  A1 0004 R                mov ax, i
 000C  D1 E0                         shl ax, 1 ;ax = 2i
 000E  D1 E0                         shl ax, 1 ;ax = 4i
 0010  8B 1E 0000 R                  mov bx, a
 0014  3B 1E 0002 R                  cmp bx, b
 0018  7F 13                         jg calc_1

 001A                   calc_2: ;if(a <= b)
 001A  8B D8                         mov bx, ax ;bx = ax = 4i
 001C  D1 EB                         shr bx, 1 ;bx = 2i
 001E  03 C3                         add ax, bx ;ax = 6i
 0020  F7 D8                         neg ax ;ax = -6i
 0022  8B C8                         mov cx, ax ;cx = ax = -6i
 0024  83 C1 06                      add cx, 6h ;cx = (6 - 6i)
```

```
 0027  05 0008                  add ax, 8h ;ax = (8 - 6i)
 002A  EB 0B 90                  jmp res_f1_f2

 002D                 calc_1: ;if(a > b)
 002D  F7 D8                     neg ax ;ax = -4i
 002F  8B C8                     mov cx, ax ;cx = ax = -4i
 0031  05 0007                   add ax, 7h ;ax = (7 - 4i)
 0034  83 C1 14                  add cx, 14h ;cx = (20 - 4i)

 0037                 res_f1_f2:
 0037  A3 0008 R            mov i1, ax
#Microsoft (R) Macro Assembler Version 5.10
11/10/21 23:26:3

Page     1-2


 003A  89 0E 000A R              mov i2, cx

 003E                 f3:
 003E  8B 1E 0006 R              mov bx, k
 0042  83 FB 00                  cmp bx, 0h
 0045  74 11                     je calc3_1

 0047                 calc3_2: ;if(k \= 0)
 0047  3B C1                     cmp ax, cx
 0049  7E 07                     jle min_1

 004B                 min_2: ;if(ax > cx)
 004B  89 0E 000C R              mov res, cx
 004F  EB 16 90                  jmp exit

 0052                 min_1: ;if(ax <= cx)
 0052  A3 000C R            mov res, ax
 0055  EB 10 90                  jmp exit

 0058                 calc3_1: ;if(k == 0)
 0058  03 C1                     add ax, cx
 005A  A3 000C R            mov res, ax
 005D  3D 0000                   cmp ax, 0h
 0060  7D 05                     jge exit

 0062                 abs:
 0062  F7 D8                     neg ax
 0064  A3 000C R            mov res, ax

 0067                 exit:
 0067  CB                   ret

 0068                 Main ENDP
 0068                 CODE ENDS
                       END Main
#Microsoft (R) Macro Assembler Version 5.10
11/10/21 23:26:3
```

```
Symbols-1


Segments and Groups:

                N a m e         Length     Align     Combine
Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                0068  PARA  NONE
DATA . . . . . .                000E  PARA  NONE

Symbols:

                N a m e         Type  Value     Attr

A  . . . . . .                  L WORD     0000  DATA
ABS  . . . . . .                L NEAR     0062  CODE

B  . . . . . .                  L WORD     0002  DATA

CALC3_1  . . . . . .            L NEAR     0058  CODE
CALC3_2  . . . . . .            L NEAR     0047  CODE
CALC_1 . . . . . .              L NEAR     002D  CODE
CALC_2 . . . . . .              L NEAR     001A  CODE

EXIT . . . . . .                L NEAR     0067  CODE

F1_2 . . . . . .                L NEAR     0009  CODE
F3 . . . . . .                  L NEAR     003E  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 = 0068
MIN_1  . . . . . .              L NEAR     0052  CODE
MIN_2  . . . . . .              L NEAR     004B  CODE

RES  . . . . . .                L WORD     000C  DATA
RES_F1_F2  . . . . . .          L NEAR     0037  CODE

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

#Microsoft (R) Macro Assembler Version 5.10
11/10/21 23:26:3


Symbols-2
```

```
   86 Source  Lines
   86 Total   Lines
   27 Symbols

48042 + 461265 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```