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(ка) гр. 1381 _____ Денисова О.К.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы

Изучить реализацию ветвления на языке Ассемблера и реализовать программу, содержащую ветвление.

Общая формулировка задачи

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.

Выполнение работы

В ходе выполнения лабораторной работы была написана программа на языке Ассемблера. Вначале мы создаем три сегмента: AStack – сегмент стека, DATA – сегмент данных и CODE – сегмент кода.

В сегменте данных были объявлены переменные: a, b, i, k – заполненные целыми числами на выбор, а также i1, i2. result – переменные для хранения результатов вычислений.

В сегменте кода в процедуре Main было написано 9 меток:

1) start_calc : в AX устанавливается значение i, затем с помощью shl в AX значение становится 2i. Это делается затем, что мы потом будем несколько раз использовать это значение, потому устанавливаем его сразу. В CX устанавливается значение b. С

помощью cmp сравниваются a и CX (a и b). Если a > b, переходим на метку calc_1

2) calc_2: выполняется, если a <= b. В AX и CX записывается 3i, к AX прибавляется 4, таким образом мы находим значение функции f1 при a <= b. Результат заносится в i1.

Затем в CX меняется знак с помощью neg, и добавляется 10. Таким образом имеем в CX результат функции f2 при a <= b. Результат записывается в i2.

После этого переходим в res_calc

3) calc_1: выполняется, если a > b. В CX кладем 15 и вычитаем значение, хранящееся в AX (2i). Таким образом в CX получаем значение функции f1 при a > b.

Затем с помощью shl увеличиваем значение в AX в 2 раза (делаем 1 сдвиг влево, получается, что в AX теперь 4i). Вычитаем 5 из AX, меняем знак с помощью neg, таким образом получаем в AX значение функции f2 при a > b и полученный результат кладем в i2.

4) res_calc: здесь кладем k в AX и сравниваем с 0. Если k < 0, переходим к res_1. Иначе кладем i2 в AX и сравниваем с 0. Если AX >= 0, переходим к mod_ok, иначе меняем знак AX с помощью neg. Таким образом реализуется взятие по модулю i2.

5) mod_ok: сравниваем AX с 7. Если AX >= 7, тогда переходим к ok_1, иначе кладем в AX 7.

6) ok_1: кладем значение AX в result, переходим к finish. Здесь завершается вычисление итоговой функции f3 при k >= 0.

7) res_1: кладем в AX i1, затем вычитаем из него i2 с помощью sub. Сравниваем AX с 0: если AX >= 0, переходим к mod_ok_1, иначе меняем знак AX с помощью neg.

8) mod_ok_1: кладем получившийся в AX результат в result. Здесь заканчивается вычисление итоговой функции f3 при k < 0.

9) finish: происходит ret.

После выполнения программа завершается.

Значения констант	Ожидаемый результат	Полученный результат
a = 4 b = 2 i = 5 k = -3	i1 = 5 (0005) i2 = -15 (FFF1) f3 = 20 (0014)	i1 = 5 (0005) i2 = -15 (FFF1) f3 = 20 (0014)
a = 2 b = 4 i = 5 k = -3	i1 = 19 (0013) i2 = -5 (FFFB) f3 = 24 (0018)	i1 = 19 (0013) i2 = -5 (FFFB) f3 = 24 (0018)
a = 2 b = 4 i = 5 k = 3	i1 = 19 (0013) i2 = -5 (FFFB) f3 = 7 (0007)	i1 = 19 (0013) i2 = -5 (FFFB) f3 = 7 (0007)

Таблица 1. Тестирование работы программы

Выводы

В ходе выполнения лабораторной работы была написана программа на языке Ассемблера, содержащая ветвления. Реализованная программа была отлажена с разными входными значениями, полученный результат был сравнен с ожидаемым и представлен в таблице 1. Ожидаемый и полученный результат во всех случаях совпадают.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ
ФАЙЛ LAB3.ASM

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

DATA SEGMENT
     a DW 2
     b DW 4
     i DW 5
     k DW 3
     i1 DW ?
     i2 DW ?
     result DW ?
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

start_calc:
     mov AX, i ; AX = i
     shl AX, 1 ; AX = 2i
     mov CX, b ; CX = b

     cmp a, CX

     jg calc_1 ; jump if a > b

calc_2: ; a <= b
     add AX, i ; AX = 3i
     mov CX, AX ; CX = 3i
     add AX, 4 ; AX = 3i + 4
     mov i1, AX ; i1 = 3i + 4

     neg CX    ; CX = -3i
     add CX, 10 ; CX = 10 - 3i
     mov i2, CX ; i2 = 10 - 3i

     jmp res_calc

calc_1: ; a > b
     mov CX, 15 ; CX = 15
     sub CX, AX ; CX = 15 - 2i
     mov i1, CX ; i1 = 15 - 2i

     shl AX, 1 ; AX = 4i
     sub AX, 5 ; AX = 4i - 5
```

```
     neg AX ; AX = - (4i - 5)
     mov i2, AX ; i2 = - (4i - 5)

res_calc:
     mov AX, k ; AX = k
     cmp AX, 0

     jl res_1 ; jump if k < 0

     ; k >= 0
     mov AX, i2 ; AX = i2
     cmp AX, 0
     jge mod_ok ; AX = abs(i2)

     neg AX

mod_ok:
     cmp AX, 7
     jge ok_1 ; jump if abs(i2) >= 7

     mov AX, 7
ok_1:
     mov result, AX ; result = AX
     jmp finish
res_1:
     mov AX, i1 ; AX = i1
     sub AX, i2 ; AX = i1 - i2
     cmp AX, 0
     jge mod_ok_1 ; AX = abs(i1 - i2)

     neg AX
mod_ok_1:
     mov result, AX

finish:
     ret

MAIN ENDP
CODE ENDS

END Main
```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ ПРОГРАММЫ

ФАЙЛ LAB3.LST

```
     Microsoft      (R)      Macro      Assembler      Version      5.10
10/29/22 22:14:2
                                                                    Page
1-1


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

       0018                 AStack ENDS

       0000                 DATA SEGMENT
       0000  0002                    a DW 2
       0002  0004                    b DW 4
       0004  0005                    i DW 5
       0006  0003                    k DW 3
       0008  0000                    i1 DW ?
       000A  0000                    i2 DW ?
       000C  0000                    result DW ?
       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                 start_calc:
       0009  A1 0004 R                    mov AX, i ; AX = i
       000C  D1 E0                        shl AX, 1 ; AX = 2i
       000E  8B 0E 0002 R                 mov CX, b ; CX = b

       0012  39 0E 0000 R                 cmp a, CX

       0016  7F 18                        jg calc_1 ; jump if a > b

       0018                 calc_2: ; a <= b
       0018  03 06 0004 R                 add AX, i ; AX = 3i
       001C  8B C8                        mov CX, AX ; CX = 3i
       001E  05 0004                      add AX, 4 ; AX = 3i + 4
       0021  A3 0008 R                    mov i1, AX ; i1 = 3i + 4

       0024  F7 D9                        neg CX    ; CX = -3i
       0026  83 C1 0A                     add CX, 10 ; CX = 10 - 3i
       0029  89 0E 000A R                 mov i2, CX ; i2 = 10 - 3i
```

```
002D  EB 14 90                jmp res_calc

0030                  calc_1: ; a > b
0030  B9 000F                 mov CX, 15 ; CX = 15
0033  2B C8                   sub CX, AX ; CX = 15 - 2i
0035  89 0E 0008 R            mov i1, CX ; i1 = 15 - 2i

0039  D1 E0                   shl AX, 1 ; AX = 4i
```



```
003B  2D 0005                 sub AX, 5 ; AX = 4i - 5
003E  F7 D8                   neg AX ; AX = - (4i - 5)
0040  A3 000A R               mov i2, AX ; i2 = - (4i - 5)

0043                  res_calc:
0043  A1 0006 R               mov AX, k ; AX = k
0046  3D 0000                 cmp AX, 0

0049  7C 18                   jl res_1 ; jump if k < 0

                            ; k >= 0
004B  A1 000A R               mov AX, i2 ; AX = i2
004E  3D 0000                 cmp AX, 0
0051  7D 02                   jge mod_ok ; AX = abs(i2)

0053  F7 D8                   neg AX

0055                  mod_ok:
0055  3D 0007                 cmp AX, 7
0058  7D 03                   jge ok_1 ; jump if abs(i2) >= 7

005A  B8 0007                 mov AX, 7
005D                  ok_1:
005D  A3 000C R               mov result, AX ; result = AX
0060  EB 12 90                jmp finish
0063                  res_1:
0063  A1 0008 R               mov AX, i1 ; AX = i1
0066  2B 06 000A R            sub AX, i2 ; AX = i1 - i2
006A  3D 0000                 cmp AX, 0
006D  7D 02                   jge mod_ok_1 ; AX = abs(i1 - i2)

006F  F7 D8                   neg AX
0071                  mod_ok_1:
0071  A3 000C R               mov result, AX

0074                  finish:
0074  CB                      ret

0075                  MAIN ENDP
0075                  CODE ENDS

                      END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0075	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
CALC_1	L NEAR	0030	CODE
CALC_2	L NEAR	0018	CODE
FINISH	L NEAR	0074	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 = 0075
MOD_OK	L NEAR	0055	CODE
MOD_OK_1	L NEAR	0071	CODE
OK_1	L NEAR	005D	CODE
RESULT	L WORD	000C	DATA
RES_1	L NEAR	0063	CODE
RES_CALC	L NEAR	0043	CODE
START_CALC	L NEAR	0009	CODE
@CPU	TEXT	0101h	
@FILENAME	TEXT	lab3	
@VERSION	TEXT	510	

```
          93 Source  Lines
          93 Total   Lines
          25 Symbols

       48034 + 446649 Bytes symbol space free

           0 Warning Errors
           0 Severe  Errors
```