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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Организация ЭВМ и систем»
Тема: Представление и обработка целых чисел. Организация ветвящихся процессов
Вариант 11

Студентка гр.1381 _____ Рымарь М.И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер. Научиться организовывать ветвящиеся процессы.

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 №11

/ -(4*i+3), при a>b f2 = < \ 6*i-10, при a<=b	/ 2*(i+1)-4, при a>b f6 = < \ 5-3*(i+1), при a<=b	/ min(|i1|, 6), при k=0 f5 = < \ |i1|+|i2|, при k/=0

Выполнение работы.

1. Были созданы три сегмента: сегмент стека (AStack), сегмент данных (DATA) и сегмент кода (CODE). Метки сегментов были записаны в соответствующие регистры с помощью директивы ASSUME (полное определение сегментов). Исходный код программы см. в приложении А.

2. В сегменте DATA были объявлены переменные a, b, i, k, i1, i2, res. В этом сегменте будут меняться некоторые переменные во время тестирования.

3. В сегменте CODE была создана процедура Main, в которой написаны инструкции для успешного завершения программы после операции ret. Для

выполнения задания использовались следующие переходы, чтобы избежать обращение к процедурам:

1). JMP (сокращение от JUMP) – команда безусловного перехода. Выполняет безусловный переход в указанное место. В процедуре Main используется в случае, когда a больше b, чтобы избежать выполнение кода в обратном случае. Также используется в f3_1 и f3_2, чтобы перейти к записи результата вычисления функции.

2). JLE (Jump Less Equal) – команда, выполняющая короткий переход, если первый операнд меньше второго операнда или равен ему при выполнении операции сравнения с помощью команды cmp. В процедуре Main используется в самом начале для перехода к метке AlessB, если a не больше b; также используется в f3_1 при условии k=0, то есть: если |i1| <= 6, то переход к метке min.

3). JGE (Jump Greater Equal) – команда, выполняющая короткий переход, если первый операнд больше второго операнда или равен ему при выполнении сравнения с помощью команды cmp. Используется в процедуре Main в двух случаях: ABSi1 и ABSi2, чтобы осуществить переход к f3, если i1>=0, или к метке f3_2, если i2>=0.

4). JNE (Jump Not Equal) – команда, выполняющая короткий переход, если первый операнд не равен второму операнду. Используется в f3, чтобы при k=0 избежать выполнение кода при k/=0.

Тестирование.

Чтобы проверить корректность работы программы, было проведено три теста.

1. Результаты работы программы при a=5; b=-1; i=2; k=0 представлены в табл.1.

i1	i2	res	Правильность результата
000B (11)	0002(2)	0006 (6)	Верно

Таблица 1 – Результаты первого теста

2. Результаты работы программы при a=2; b=4; i=-3; k=0 представлены в табл.2.

i1	i2	res	Правильность результата
001C (28)	000B(11)	0006 (6)	Верно

Таблица 2 – Результаты второго теста

2. Результаты работы программы при a=2; b=4; i=-3; k=5 представлены в табл.3.

i1	i2	res	Правильность результата
001C (28)	000B(11)	0027 (39)	Верно

Таблица 3 – Результаты третьего теста

Выводы.

В ходе выполнения лабораторной работы было изучено представление и обработка целых чисел, и организация ветвящихся процессов. Для выполнения задания была написана программа, которая вычисляет значения функций согласно заданным условиям.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *lab3.asm*

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

DATA SEGMENT
a DW 2
b DW 4
i DW -3
k DW 5
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

    mov AX, i
    ; i+1
    add AX, 1
    mov CX, a
    cmp CX, b
    jle AlessB

AmoreB:
    ; 2*i+2
    shl AX, 1
    ; 2*(i+1)-4 = 2i-2
    sub AX, 4
    mov i2, AX
    ; 4i-4
```

```
    shl AX, 1
    ; 4i+3
    add AX, 7
    ; -(4i+3)
    neg AX
    mov i1, AX
    jmp ABSi1

AlessB:
    mov BX, AX
    shl AX, 1
    shl AX, 1
    ; 3*(i+1)
    sub AX, BX
    ; -3*(i+1)
    neg AX
    ; 5-3*(i+1) = -3i+2
    add AX, 5
    mov i2, AX
    ; -6i+4
    shl AX, 1
    ; -6i+10
    add AX, 6
    ; 6i-10
    neg AX
    mov i1, AX
ABSi1:
    mov CX, i1
    cmp CX, 0
    jge f3
    neg i1
f3:
    mov CX, k
    cmp CX, 0
    jne ABSi2
f3_1:
    mov CX, i1
    cmp CX, 6
    jle min
    mov AX, 6
    jmp f3res
min:
    mov AX, i1
    jmp f3res
```

```
ABSi2:
    mov CX, i2
    cmp CX, 0
    jge f3_2
    neg i2
f3_2:
    mov AX, i1
    add AX, i2
    jmp f3res
f3res:
    mov res, AX
    ret
Main ENDP
CODE ENDS
    END Main
```

Название файла: *lab2.lst*

```
Microsoft (R) Macro Assembler Version 5.10                  10/20/22
15:46:2
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  FFFD                                           i DW -3
0006  0005                                           k DW 5
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  A1 0004 R                                     mov AX, i
; i+1
```

```
000C  05 0001                                     add AX, 1
000F  8B 0E 0000 R                              mov CX, a
0013  3B 0E 0002 R                              cmp CX, b
0017  7E 15                                       jle AlessB

0019                                               AmoreB:
; 2*i+2
0019  D1 E0                                       shl AX, 1
; 2*(i+1)-4 = 2i-2
001B  2D 0004                                     sub AX, 4
001E  A3 000A R                                 mov i2, AX
; 4i-4
0021  D1 E0                                       shl AX, 1
; 4i+3
0023  05 0007                                     add AX, 7
; -(4i+3)
0026  F7 D8                                       neg AX
0028  A3 0008 R                                 mov i1, AX
002B  EB 1B 90                                    jmp ABSi1

002E                                               AlessB:
002E  8B D8                                       mov BX, AX
0030  D1 E0                                       shl AX, 1
0032  D1 E0                                       shl AX, 1
; 3*(i+1)

Microsoft (R) Macro Assembler Version 5.10                  10/20/22
15:46:2
Page     1-2


0034  2B C3                                       sub AX, BX
; -3*(i+1)
0036  F7 D8                                       neg AX
; 5-3*(i+1) = -3i+2
0038  05 0005                                     add AX, 5
003B  A3 000A R                                 mov i2, AX
; -6i+4
003E  D1 E0                                       shl AX, 1
; -6i+10
0040  05 0006                                     add AX, 6
; 6i-10
0043  F7 D8                                       neg AX
0045  A3 0008 R                                 mov i1, AX
0048                                               ABSi1:
0048  8B 0E 0008 R                              mov CX, i1
004C  83 F9 00                                    cmp CX, 0
004F  7D 04                                       jge f3
0051  F7 1E 0008 R                              neg i1
0055                                               f3:
0055  8B 0E 0006 R                              mov CX, k
0059  83 F9 00                                    cmp CX, 0
005C  75 15                                       jne ABSi2
005E                                               f3_1:
005E  8B 0E 0008 R                              mov CX, i1
0062  83 F9 06                                    cmp CX, 6
0065  7E 06                                       jle min
```

```
0067  B8 0006                                       mov AX, 6
006A  EB 1E 90                                      jmp f3res
006D                                                 min:
006D  A1 0008 R                                   mov AX, i1
0070  EB 18 90                                      jmp f3res
0073                                                 ABSi2:
0073  8B 0E 000A R                                mov CX, i2
0077  83 F9 00                                      cmp CX, 0
007A  7D 04                                         jge f3_2
007C  F7 1E 000A R                                neg i2
0080                                                  f3_2:
0080  A1 0008 R                                   mov AX, i1
0083  03 06 000A R                                add AX, i2
0087  EB 01 90                                      jmp f3res
008A                                                 f3res:
008A  A3 000C R                                   mov res, AX
008D  CB                                            ret
008E                                                 Main ENDP
008E                                                 CODE ENDS
END Main

Microsoft (R) Macro Assembler Version 5.10                  10/20/22
15:46:2
Symbols-1


Segments and Groups:

N a m e                                 Length   Align     Combine Class

ASTACK . . . . . .                      0018    PARA  STACK
CODE . . . . . .                        008E    PARA  NONE
DATA . . . . . .                        000E    PARA  NONE

Symbols:

N a m e                                 Type     Value      Attr

A  . . . . . .                          L WORD  0000  DATA
ABSI1  . . . . . .                      L NEAR  0048  CODE
ABSI2  . . . . . .                      L NEAR  0073  CODE
ALESSB . . . . . .                      L NEAR  002E  CODE
AMOREB . . . . . .                      L NEAR  0019  CODE

B  . . . . . .                          L WORD  0002  DATA

F3 . . . . . .                          L NEAR  0055  CODE
F3RES  . . . . . .                      L NEAR  008A  CODE
F3_1 . . . . . .                        L NEAR  005E  CODE
F3_2 . . . . . .                        L NEAR  0080  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 = 008E
MIN  . . . . . .          L NEAR  006D  CODE

RES  . . . . . .          L WORD  000C  DATA

@CPU . . . . . .          TEXT  0101h
@FILENAME  . . . . . .    TEXT  lab3
@VERSION . . . . . .      TEXT  510


97 Source  Lines
97 Total   Lines
25 Symbols

48014 + 461293 Bytes symbol space free

0 Warning Errors
0 Severe  Errors
```