**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: Изучение организации ветвлений в программах на языке Ассемблера**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8382 |  | Мирончик П.Д. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2019

**Цель работы.**

Изучение организации ветвлений в программах на языке Ассемблера.

**Формулировка задачи.**

Разработать на языке Ассемблера IBM PC программу, которая по заданным целым значениям *a*, *b*, *i*, *k*, размером 1 слово, вычисляет:

1. Значения и ,
2. Значения

где функции и определяются из таблицы 1, а - из таблицы 2 по цифрам шифра индивидуального задания.

Значения *a, b, i, k* являются исходными данными, которые должны быть выбраны самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть все возможные комбинации параметров *a, b, i, k*, позволяющие проверить различные маршруты выполнения программы.

**Ход работы**

В соответствии с вариантом были реализованы функции

f1 = a > b ? (4\*i-7) : (-(2\*i+6))

f2 = a > b ? (6\*i-4) : (3\*(i+1))

f3 = k < 0 ? min(i1,8) : max(8,i2)

Для выполнения вычислений использовались операции:

* ADD – для сложения двух данных и записи в регистр
* SUB – для вычитания двух данных и записи в регистр
* NEG – для получения противоположного значения данных в регистре
* SHL – для поразрядного сдвига содержимого регистра влево (данным способом реализовано умножение на 2).

Для выполнения условных переходов использовались следующие операции:

* CMP - сравнение двух чисел
* JL – меньше
* JMP – безусловный переход
* JG – больше

Тестирование программы:

|  |  |  |  |
| --- | --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Полученный результат** | **Корректность программы** |
| a=1  b=2  i=3  k=1 | i1=000816=8  i2=FFF416=-12  res=000C16=12 | i1=000816=8  i2=FFF416=-12  res=000C16=12 | + |
| a=2  b=1  i=4  k=1 | i1=FFED16=-19  i2=000416=4  res=000716=20 | i1=FFED16=-19  i2=000416=4  res=000716=20 | + |
| a=1  b=2  i=3  k=-1 | i1=000816=8  i2=FFF416=-12  res=000C16=12 | i1=000816=8  i2=FFF416=-12  res=000C16=12 | + |
| a=2  b=1  i=4  k=-1 | i1=FFED16=-19  i2=000416=4  res=000F16=15 | i1=FFED16=-19  i2=000416=4  res=000F16=15 | + |

**Выводы**

В ходе выполнения данной работы были изучены операторы ветвления языка Ассемблер, операторы сдвига и связанное с ними умножение чисел, получены навыки более детальной работы с отладчиком AFD.

**Приложение А. Код программы**

AStack SEGMENT STACK

DW 12 DUP(?)

AStack ENDS

; Данные программы

DATA SEGMENT

i1 DW 0

i2 DW 0

res DW ?

a DW 1

b DW 2

i DW 3

k DW 1

DATA ENDS

; Код программы

CODE SEGMENT

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

; Головная процедура

Main PROC FAR

; Push PSP address to exit

push ds

mov ax,0

push ax

; Set up DS

mov ax, DATA

mov ds, ax

mov ax,a

cmp ax,b

mov ax, i ; set up ax = i

jl a\_L\_b

a\_G\_b:

shl ax, 1

shl ax, 1

mov cx,ax ; cx = i \* 4

add ax, 3

neg ax

mov i1,ax

neg cx

add cx,20

mov i2,cx

jmp f\_res

a\_L\_b:

shl ax,1

mov cx,ax

add ax,cx

add ax,cx

mov cx,ax ; cx = 6\*i

sub ax,10

mov i1,ax

sub cx,6

neg cx

mov i2,cx

f\_res:

cmp cx,0 ; compute |i2|

jg i2\_G\_0

neg cx

i2\_G\_0:

mov ax,k ; comparing k > 0

cmp ax,0

jge k\_GE\_0

k\_L\_0:

mov ax,i1

cmp ax,0 ; модуль i1

jg i1\_G\_0

neg ax

i1\_G\_0: sub ax,cx

mov res,ax

jmp exit

k\_GE\_0:

cmp cx,7

jg i2\_G\_7

mov res,7 ; 7 >= i

jmp exit

i2\_G\_7: mov res,cx

exit:

mov ax,i1

mov ax,i2

mov ax,res

ret

Main ENDP

CODE ENDS

END Main

**Приложение B. Файл листинга**

Microsoft (R) Macro Assembler Version 5.10 10/9/19 12:48:12

Page 1-1

0000 AStack SEGMENT STACK

0000 000C[ DW 12 DUP(?)

????

]

0018 AStack ENDS

; Данные программы

0000 DATA SEGMENT

0000 0000 i1 DW 0

0002 0000 i2 DW 0

0004 0000 res DW ?

0006 0001 a DW 1

0008 0002 b DW 2

000A 0003 i DW 3

000C 0001 k DW 1

000E DATA ENDS

; Код программы

0000 CODE SEGMENT

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

; Головная процедура

0000 Main PROC FAR

; Push PSP address to exit

0000 1E push ds

0001 B8 0000 mov ax,0

0004 50 push ax

; Set up DS

0005 B8 ---- R mov ax, DATA

0008 8E D8 mov ds, ax

000A A1 0006 R mov ax,a

000D 3B 06 0008 R cmp ax,b

0011 A1 000A R mov ax, i ; set up ax = i

0014 7C 1A jl a\_L\_b

0016 a\_G\_b:

0016 D1 E0 shl ax, 1

0018 D1 E0 shl ax, 1

001A 8B C8 mov cx,ax ; cx = i \* 4

001C 05 0003 add ax, 3

001F F7 D8 neg ax

0021 A3 0000 R mov i1,ax

0024 F7 D9 neg cx

0026 83 C1 14 add cx,20

Microsoft (R) Macro Assembler Version 5.10 10/9/19 12:48:12

Page 1-2

0029 89 0E 0002 R mov i2,cx

002D EB 1A 90 jmp f\_res

0030 a\_L\_b:

0030 D1 E0 shl ax,1

0032 8B C8 mov cx,ax

0034 03 C1 add ax,cx

0036 03 C1 add ax,cx

0038 8B C8 mov cx,ax ; cx = 6\*i

003A 2D 000A sub ax,10

003D A3 0000 R mov i1,ax

0040 83 E9 06 sub cx,6

0043 F7 D9 neg cx

0045 89 0E 0002 R mov i2,cx

0049 f\_res:

0049 83 F9 00 cmp cx,0 ; compute |i2|

004C 7F 02 jg i2\_G\_0

004E F7 D9 neg cx

0050 i2\_G\_0:

0050 A1 000C R mov ax,k ; comparing k > 0

0053 3D 0000 cmp ax,0

0056 7D 12 jge k\_GE\_0

0058 k\_L\_0:

0058 A1 0000 R mov ax,i1

005B 3D 0000 cmp ax,0 ; модуль i1

005E 7F 02 jg i1\_G\_0

0060 F7 D8 neg ax

0062 2B C1 i1\_G\_0: sub ax,cx

0064 A3 0004 R mov res,ax

0067 EB 13 90 jmp exit

006A k\_GE\_0:

006A 83 F9 07 cmp cx,7

006D 7F 09 jg i2\_G\_7

006F C7 06 0004 R 0007 mov res,7 ; 7 >= i

0075 EB 05 90 jmp exit

0078 89 0E 0004 R i2\_G\_7: mov res,cx

007C exit:

007C A1 0000 R mov ax,i1

007F A1 0002 R mov ax,i2

0082 A1 0004 R mov ax,res

0085 CB ret

0086 Main ENDP

0086 CODE ENDS

END Main

Microsoft (R) Macro Assembler Version 5.10 10/9/19 12:48:12

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 0086 PARA NONE

DATA . . . . . . . . . . . . . . 000E PARA NONE

Symbols:

N a m e Type Value Attr

A . . . . . . . . . . . . . . . L WORD 0006 DATA

A\_G\_B . . . . . . . . . . . . . L NEAR 0016 CODE

A\_L\_B . . . . . . . . . . . . . L NEAR 0030 CODE

B . . . . . . . . . . . . . . . L WORD 0008 DATA

EXIT . . . . . . . . . . . . . . L NEAR 007C CODE

F\_RES . . . . . . . . . . . . . L NEAR 0049 CODE

I . . . . . . . . . . . . . . . L WORD 000A DATA

I1 . . . . . . . . . . . . . . . L WORD 0000 DATA

I1\_G\_0 . . . . . . . . . . . . . L NEAR 0062 CODE

I2 . . . . . . . . . . . . . . . L WORD 0002 DATA

I2\_G\_0 . . . . . . . . . . . . . L NEAR 0050 CODE

I2\_G\_7 . . . . . . . . . . . . . L NEAR 0078 CODE

K . . . . . . . . . . . . . . . L WORD 000C DATA

K\_GE\_0 . . . . . . . . . . . . . L NEAR 006A CODE

K\_L\_0 . . . . . . . . . . . . . L NEAR 0058 CODE

MAIN . . . . . . . . . . . . . . F PROC 0000 CODE Length = 0086

RES . . . . . . . . . . . . . . L WORD 0004 DATA

@CPU . . . . . . . . . . . . . . TEXT 0101h

@FILENAME . . . . . . . . . . . TEXT main

@VERSION . . . . . . . . . . . . TEXT 510

102 Source Lines

102 Total Lines

25 Symbols

48056 + 461251 Bytes symbol space free

0 Warning Errors

0 Severe Errors