**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 №9**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2019

**Цель работы.**

Получить знания о представлении и обработке целых чисел. Изучить понятие ветвящихся процессов и их организацию. Разработать на языке Ассемблера программу, вычисляющую значения функций, в зависимости от заданных параметров.

**Задание.**

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k, вычисляет:

а) 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 fn1,fn2  определяется из табл.1, а функции fn3- из табл.2 по цифрам шифра индивидуального задания (n1.n2.n3).

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.

Заданные функции:

f1 =

f2 =

f3 =

**Теоретические сведения.**

* **SHL** приемник, счетчик; 8086, логический сдвиг влево.

Операция сдвига влево на 1 эквивалентна ум­ножению на 2. Так, число 0010b (2) после сдвига на 1 влево превращается в 0100b (4).

Команды SAL и SHL выполняют одну и ту же операцию (на самом деле это одна и та же команда) – на каждый шаг сдвига старший бит заносится в CF, **все** биты сдвигаются влево на одну позицию, и младший бит обнуляет­ся.

* **JMP** операнд; 8086, безусловный переход

JMP передает управление в другую точку программы. Операндом может быть непосред­ственный адрес для перехода, а также регистр или переменная, содержащая адрес.

* **СМР** приемник, источник; 8086, сравнение.

Сравнивает приемник и источник и устанавливает флаги. Сравнение осуществляется путем вычитания источника (число, регистр или пере­менная) из приемника (регистр или переменная; приемник и источник не могут быть переменными одновременно), причем результат вычита­ния никуда не записывается, единственным результатом работы этой команды оказывается изменение флагов CF, OF, SF, ZF, AF и PF.

**Jcc** операнд; 8086, переход по адресу, задаваемому операндом, если условие перехода сс истинно.

**JG** Переход, если больше (ZF = 0 и SF = OF)

**JGE**  Переход, если больше или равно (SF = OF)

**JLE**  Переход, если меньше или равно (ZF=1 или SF <> OF)

**JS**  Переход, если знак отрицательный (SF = 1)

Команды условного перехода (за исключением команды JCXZ) проверяют флаги, которые были установлены предыдущей командой. Условия для каждой мнемонической формы команды даны в круглых скобках выше после каждого описания. Если условие выполняется, то происходит переход по адресу, задаваемому операндом, иначе – выполняется команда, следующая за командой условного перехода.

**Ход работы.**

1. Определение модели памяти с помощью директивы .model. Модель памяти - small. Описание упрощенных директив сегментации:

* .stack – для указания начала сегмента стека;
* .data – для указания начала сегмента данных;
* .code – для указания начала сегмента кода;

1. Выбор размера стека, и инициализация переменных i, a, b, k;
2. Сравнение переменных a и b с помощью команды cmp. Если a > b переход к метке second, иначе выполняются команды метки first.
3. Далее происходит вычисление значений i1и i2 заданных функций. Умножение реализовано с помощью логического сдвига влево и сложения. Значение i1 сохраняется в регистре ax. Значение i2 сохраняется в регистре сx.
4. При k ≥ 0 вычисляется модуль i1, который сравнивается с 6. Из этих чисел выбирается большее и сохраняется в регистре ax. Иначе вычисляются модуль i1 и модуль i2 и складываются. Полученное значение сохраняется в регистре ax. В переменную res сохраняется конечное значение.

**Минимизация длины кода.**

* a ≤ b

i1 = 6i-10; i2 = 3i+6;

Пусть j = 3i+6, тогда

i1 = 2j-22; i2 = j;

* a > b

i1 = -4i-3; i2 = -6i+4;

Пусть j = -4i-3, тогда

i1 = j; i2 = j–2i+7;

**Вывод.**

В ходе выполнения данной лабораторной работы были получены сведения о реализации сравнения, меток и перехода по ним, а также изучены организации ветвлений в программах на языке Ассемблера.

**Приложение А. Код программы lr3.asm**

dosseg

.model small

.stack 100h

.data

i dw 5

a dw 15

b dw -3

k dw 9

res dw ?

.code

mov ax, @data

mov ds, ax

mov ax, a

cmp ax, b

jg second

first: *;if(a<=b)*

mov cx, i *;i*

add cx, 2 *;i+2*

shl cx, 1 *;2(i+2)*

add cx, i *;2(i+2)+i*

add cx, 2 *;3(i+2)*

mov ax, cx *;3(i+2)*

shl ax, 1 *;6(i+2)*

sub ax, 22 *;6i-10*

jmp final

second: *;if(a>b)*

mov ax, i *;i*

shl ax, 1 *;2i*

shl ax, 1 *;4i*

add ax, 3 *;4i+3*

neg ax *;-(4i+3)*

mov cx, ax *;-(4i+3)*

sub cx, i *;-4i-3-i*

sub cx, i *;-5i-3-i*

add cx, 7 *;-(6i-4)*

final:

abs\_1: neg ax

js abs\_1 *;|i1|*

cmp k, 0

jge abs\_0

abs\_2: neg cx

js abs\_2 *;|i2|*

add ax, cx *;|i1|+|i2|*

jmp ex

abs\_0:

cmp ax, 6

jle mnr *;if(|i1|<=6)*

jmp ex

mnr: mov ax, 6

ex: mov res, ax

mov ah, 4ch

int 21h

end

**Приложение Б. Тестирование программы.**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Исходные данные** | **Ожидаемый**  **результат** | **Полученный**  **результат** |
| 1 | i = 000316 = 310  a = 000116 = 110  b = 000216 = 210  k = 000416 = 410 | i1 = 000816 = 810  i2 = 000F16 = 1510  res = 000816 = 810 | i1 = 000816 = 810  i2 = 000F16 = 1510  res = 000816 = 810 |
| 2 | i = 000216 = 210  a = 000516 = 510  b = 000016 = 010  k = FFFF16 = -110 | i1 = FFF516 = -1110  i2 = FFF816 = -810  res = 001316 = 1910 | i1 = FFF516 = -1110  i2 = FFF816 = -810  res = 001316 = 1910 |
| 3 | i = FFFE16 = -210  a = FFF816 = -810  b = 000716 = 710  k = FFFB16 = -510 | i1 = FFEA16 = -2210  i2 = 000016 = 010  res = 001616 = 2210 | i1 = FFEA16 = -2210  i2 = 000016 = 010  res = 001616 = 2210 |
| 4 | i = 000516 = 510  a = 000F16 = 1510  b = FFFD16 = -310  k = 000916 = 910 | i1 = FFE916 = -2310  i2 = FFE616 = -2610  res = 001716 = 2310 | i1 = FFE916 = -2310  i2 = FFE616 = -2610  res = 001716 = 2310 |
| 5 | i = 000A16 = 1010  a = 000A16 = 1010  b = 000A16 = 1010  k = 000016 = 010 | i1 = 003216 = 5010  i2 = 002416 = 3610  res = 003216 = 5010 | i1 = 003216 = 5010  i2 = 002416 = 3610  res = 003216 = 5010 |
| 6 | i = FFFF16 = -110  a = FFFB16 = -510  b = FFFA16 = -610  k = 000116 = 110 | i1 = 000116 = 110  i2 = 000A16 = 1010  res = 000616 = 610 | i1 = 000116 = 110  i2 = 000A16 = 1010  res = 000616 = 610 |

**Приложение В. Содержимое файла листинга**

Microsoft (R) Macro Assembler Version 5.10 9/28/19 00:17:50

Page 1-1

dosseg

.model small

.stack 100h

.data

0000 0005 i dw 5

0002 000F a dw 15

0004 FFFD b dw -3

0006 0009 k dw 9

0008 0000 res dw ?

.code

0000 B8 ---- R mov ax, @data

0003 8E D8 mov ds, ax

0005 A1 0002 R mov ax, a

0008 3B 06 0004 R cmp ax, b

000C 7F 1A jg second

000E first: ;if(a>b)

000E 8B 0E 0000 R mov cx, i ;i

0012 83 C1 02 add cx, 2 ;i+2

0015 D1 E1 shl cx, 1 ;2(i+2)

0017 03 0E 0000 R add cx, i ;2(i+2)+i

001B 83 C1 02 add cx, 2 ;3(i+2)

001E 8B C1 mov ax, cx ;3(i+2)

0020 D1 E0 shl ax, 1 ;6(i+2)

0022 2D 0016 sub ax, 22 ;6i-10

0025 EB 1A 90 jmp final

0028 second: ;if(a<=b)

0028 A1 0000 R mov ax, i ;i

002B D1 E0 shl ax, 1 ;2i

002D D1 E0 shl ax, 1 ;4i

002F 05 0003 add ax, 3 ;4i+3

0032 F7 D8 neg ax ;-(4i+3)

0034 8B C8 mov cx, ax ;-(4i+3)

0036 2B 0E 0000 R sub cx, i ;-4i-3-i

003A 2B 0E 0000 R sub cx, i ;-5i-3-i

003E 83 C1 07 add cx, 7 ;-(6i-4)

0041 final:

0041 83 3E 0006 R 00 cmp k, 0

0046 7D 0D jge abs\_0

0048 F7 D8 abs\_1: neg ax ;if(k<0)

004A 78 FC js abs\_1 ;|i1|

004C F7 D9 abs\_2: neg cx

004E 78 FC js abs\_2 ;|i2|

0050 03 C1 add ax, cx ;|i1|+|i2|

0052 EB 10 90 jmp ex

0055 F7 D8 abs\_0: neg ax ;if(k>=0)

0057 78 FC js abs\_0 ;|i1|

Microsoft (R) Macro Assembler Version 5.10 9/28/19 00:17:50

Page 1-2

0059 3D 0006 cmp ax, 6

005C 7E 03 jle mnr ;if(|i1|<=6)

005E EB 04 90 jmp ex

0061 B8 0006 mnr: mov ax, 6

0064 A3 0008 R ex: mov res, ax

0067 B4 4C mov ah, 4ch

0069 CD 21 int 21h

end

Microsoft (R) Macro Assembler Version 5.10 9/28/19 00:17:50

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

DGROUP . . . . . . . . . . . . . GROUP

\_DATA . . . . . . . . . . . . 000A WORD PUBLIC 'DATA'

STACK . . . . . . . . . . . . 0100 PARA STACK 'STACK'

\_TEXT . . . . . . . . . . . . . 006B WORD PUBLIC 'CODE'

Symbols:

N a m e Type Value Attr

A . . . . . . . . . . . . . . . L WORD 0002 \_DATA

ABS\_0 . . . . . . . . . . . . . L NEAR 0055 \_TEXT

ABS\_1 . . . . . . . . . . . . . L NEAR 0048 \_TEXT

ABS\_2 . . . . . . . . . . . . . L NEAR 004C \_TEXT

B . . . . . . . . . . . . . . . L WORD 0004 \_DATA

EX . . . . . . . . . . . . . . . L NEAR 0064 \_TEXT

FINAL . . . . . . . . . . . . . L NEAR 0041 \_TEXT

FIRST . . . . . . . . . . . . . L NEAR 000E \_TEXT

I . . . . . . . . . . . . . . . L WORD 0000 \_DATA

K . . . . . . . . . . . . . . . L WORD 0006 \_DATA

MNR . . . . . . . . . . . . . . L NEAR 0061 \_TEXT

RES . . . . . . . . . . . . . . L WORD 0008 \_DATA

SECOND . . . . . . . . . . . . . L NEAR 0028 \_TEXT

@CODE . . . . . . . . . . . . . TEXT \_TEXT

@CODESIZE . . . . . . . . . . . TEXT 0

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

@DATASIZE . . . . . . . . . . . TEXT 0

@FILENAME . . . . . . . . . . . TEXT lr3

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

Microsoft (R) Macro Assembler Version 5.10 9/28/19 00:17:50

Symbols-2

62 Source Lines

62 Total Lines

30 Symbols

48026 + 461281 Bytes symbol space free

0 Warning Errors

0 Severe Errors