National University of Computer and Emerging Sciences

Lab Manual 5

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE

Course Instructor	Dr Asma, Miss Aleena
Lab Instructor(s)	Maham Saleem Sidra
Section	3F,3E
Semester	Fall 2021

Department of Computer Science
FAST-NU, Lahore, Pakistan

DIV Instruction
The division used in the process is integer division and not floating-point division. Integer division gives an integer quotient and an integer remainder.

There are two forms of the DIV instruction. The first form divides a 32bit number in DX:AX by its 16bit operand and stores the 16bit quotient in AX and the 16bit remainder in DX.

The second form divides a 16bit number in AX by its 8bit operand and stores the 8bit quotient in AL and the 8bit remainder in AH. For example, "DIV BL" has an 8bit operand, so the implied dividend is 16bit and is stored in the AX register and "DIV BX" has a 16bit operand, so the implied dividend is 32bit and is therefore stored in the concatenation of the DX and AX registers. The higher word is stored in DX and the lower word in AX.

If a large number is divided by a very small number, it is possible that the quotient is larger than the space provided for it in the implied destination. In this case an interrupt is automatically generated and the program is usually terminated as a result. This is called a divide overflow error; just like the calculator shows an –E– when the result cannot be displayed. This interrupt will be discussed later in the discussion of interrupts. DIV (divide) performs an unsigned division of the accumulator (and its extension) by the source operand. If the source operand is a byte, it is divided into the two-byte dividend assumed to be in registers AL and AH. The byte quotient is returned in AL, and the byte remainder is returned in AH. If the source operand is a word, it is divided into the two-word dividend in registers AX and DX. The word quotient is returned in AX, and the word remainder is returned in DX. If the quotient exceeds the capacity of its destination register (FF for byte source, FFFF for word source), as when division by zero is attempted, a type 0 interrupt is generated, and the quotient and remainder are undefined.

MUL Instruction
MUL (multiply) performs an unsigned multiplication of the source operand and the accumulator. If the source operand is a byte, then it is multiplied by register AL and the double-length result is returned in AH and AL. If the source operand is a word, then it is multiplied by register AX, and the double-length result is returned in registers DX and AX.

Problem 1: Write an assembly program to calculate factorial of a number. (Use MUL instruction and a loop)

Problem 2: Write an assembly program that traverses the array and divide each element by some number. The array does not have same size element. Some of them are 2-byte numbers (16 bits) and some of them are 4-byte numbers (32 bits). Before each element there is a flag; if flag is 0 it means that the number is a 2-byte number. If the flag is 1, it means that the number is a 4-byte number.
For example:
Numbers: dw 1, 0xAFFE,0x1230, 0, 0x1234
In the above example, there are two numbers which are 0x1230AFFE and 0x1234. 0x1230AFFE has been stored in 4 bytes whereas as 0x1230 has been stored in 2 bytes.

Problem 3: Write a function **PrintRectanlge** that prints a rectangle having its TopLeft and BottomRight corners at (top, left) and (bottom, right) coordinates respectively where top, left, bottom and right are parameters passed by caller. Also pass attribute by caller to print colored rectangle. Following is a red rectangle with TopLeft = (2, 10) and BottomRight = (20, 60).

Problem 4: Call above function PrintRectangle and print 3 rectangles of different sizes and colors.

HINT: the Loop should have 3 steps,

Loop1:

print * at new location

delay

clear screen

jmp loop1

code for delay is

```
delay:
push cx
mov cx, 3 ; change the values  to increase delay time
delay_loop1:
push cx
mov cx, 0xFFFF
delay_loop2:
loop delay_loop2
pop cx
loop delay_loop1
pop cx
ret
```