CSUS

SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

Department of Computer Science

CSC 35

Spring 2021

Dr. Ghansah

**Lab #4: Division, Direct Console Output, and ASCII code**

**Purpose:** The purpose of this lab assignment is to familiarize you with the 1) DIV instruction used for implementing integer division, 2) method of converting a number to an ASCII character, and 3) mechanism to **output** ASCII character to the video screen using Irvine Libraries. It builds on the previous assignment.

**Introduction**:

Direct low level i/o (input/output) is easier to do with functions in Real Mode OS such as DOS/BIOS, Windows98, WindowsME, etc. Real mode allows memory and i/o ports to be accessed without restrictions. In this mode, INT 21h system calls can be used for outputting (writing) data on the screen or inputting (reading) from the keyboard at the low level. However, Protected mode OS such as Windows 10, Linux, etc hides most of the underlying low level features. Thus i/o can be done only at a high level. This is the reason why we are going to use Libraries to do i/o (ie. read and write). The Libraries are similar to high level language instructions for reading and writing except they are accessed at assembly language level. Because the operating system is protected, the job of the i/o library is to write the underlying code that requests the OS to perform input and output on behalf of the programmer in a safe manner. The author of the Textbook we are using has provided libraries we will simply call “Irvine Library”. We will use some of the i/o functions in this library in our assignments.

NOTE: In each assignment you will be told which library to use. You *cannot* use any other library the instructor has not specified in the assignment. You will be heavily penalized if you use any library the instructor has not specified for the corresponding assignment.

In the following assignment you are going to compute and print the number in variable ‘W’ on the screen in decimal. The number can only be printed on the screen as ASCII digit(s), in this case, ASCII decimal digits. Since the number W is in binary we have to 1) convert it to decimal digits, 2) convert each decimal digit to ASCII and then 3) print the ASCII digit. There is a general algorithm for doing this for multiple digits but we will save that for future assignments. This assignment is designed for only a single decimal digit output, for simplicity. Therefore, all that is needed is to convert the number to ASCII prior to printing it.

To covert a decimal digit to ASCII Add 30h (ie. 0x30) to it.

The following table shows high level language statements (in C-like notation) and the corresponding Intel assembly language expression for DIVISION and MODULUS.

|  |  |  |
| --- | --- | --- |
| High Level | Assembly | Meaning |
| c/2 | Recall that division can be described by the following formula.  Dividend/Divisor = Quotient + Remainder/Divisor.  The instruction for integer Division is  *DIV reg/mem*  Where the operand is the *divisor*.  For 8 bit operands the Dividend must be AX. The result of the division will be as follows. The quotient will be in AL and the remainder in AH. For 16 bit operands, the Dividend must be in DX:AX with the quotient in AX and the Remainder in DX. For 32 bit operands, the Dividend must be in EDX:EAX with the quotient in EAX and the Remainder will be placed in EDX. NOTE that the Remainder is the result of the MOD operation.  Here is an example for c/2.  *.code*  *Mov edx, 0*  *Mov eax, c*  *Mov ecx, 2*  *Div ecx* | Division. See section 7.3.4 of the Irvine 7th Ed text book  In the example shown, the result of the division (ie Quotient) will be in AX register and the remainder will be in DX register. |
| c%2 | Same instruction as above except the we look for the remainder of the division. which is also the same as the Modulus | c MOD 2 |

THE FOLLOWING PROCEDURE IS AN EXTENSION OF LAB 3. IT INCLUDES ADDITIONAL VARIABLES, DIVISION AND MODULUS COMPUTATIONS, AND OUTPUT TO THE SCREEN.

**Procedure:**

a) int loc, X, Y, W, loc1, loc2, loc3, sum;

int A=90;

X = 40

Y = 24

loc1 = Y \* 160 + X \* 2

loc2 = A\* 950

loc3 = loc2 – loc1

Y=3000

Y=Y-1

Sum = loc3/16 +Y + Y/4 + Y/200

W= sum % 7 + 3 ; sum **%** 7 = sum **MOD** 7

writeNum(W) ;display W on screen

**b) Output the number W to the video screen**

You can use Irvine Library Writechar to implement WriteNum in assembly. Writechar displays a single ASCII character to the screen. **It assumes that register AL contains the ASCII character to be printed on the screen.** Thus WriteNum(W) can be implemented as follows.

*1. Convert W to ASCII*

*2. Place the ASCII character in AL*

*3.Call writechar*

**c) Testing:** Be sure to test your program and make sure it works before you submit it to your lab instructor on CANVAS as specified below.

**d) Demonstration**: Demonstrate your program by providing screen shots showing input (X and Y) and corresponding output (W) values displayed on the screen. The lab instructor will also assemble and run the source you upload to CANVAS as specified below. The instructor will check whether your program produces input and corresponding output values on the screen.

**e) Submission:** Submit electronic copy of your program to CANVAS including a well d*ocumented program file(source code)* and output file (screen shots). **Filenames must be according to the format specified in the syllabus**