

UNIVERSITY OF THE PHILIPPINES VISAYAS  
COLLEGE OF ARTS AND SCIENCES  
DIVISION OF PHYSICAL SCIENCES AND MATHEMATICS

CMSC 131  
Introduction to Computer Organization and Machine Level Computing  
A.Y. 2022 - 2023

Assignment Guide

Prepared by:

Jayvee B. Castañeda  
Instructor

*ACADEMIC INTEGRITY*

*As a student of the University of the Philippines, I pledge to act ethically and uphold the value of honor and excellence. I understand that suspected misconduct on given assignments/examinations will be reported to the appropriate office and if established, will result in disciplinary action in accordance with University rules, policies and procedures. I may work with others only to the extent allowed by the Instructor.*

## Laboratory Exercise #4

### Reading

- Read [Chapter 3 of Paul Carter's PC Assembly Book](#)

### Practice Exercise:

- Assemble the assembly code (**max.asm**). This will create an object file (**max.o**) for math.asm.

**nasm -f elf max.asm**

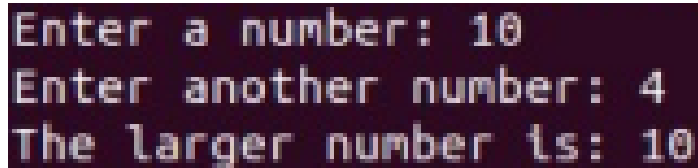
- Compile and link the assembly code with the C program (**driver.c**). In our machine, we will be using 32-bit registers thus we specify “-m32”.

```
gcc -m32 -o max driver.c max.o asm_io.o
```

- Execute the assembly code.

```
./max
```

The code should show the following:

A terminal window with a dark background and light-colored text. It displays three lines of output: "Enter a number: 10", "Enter another number: 4", and "The larger number is: 10".

```
Enter a number: 10
Enter another number: 4
The larger number is: 10
```

- Analyze the assembly code (max.asm). Reflective questions:

*What does the program do?*

*How do the “OR”, “AND”, “XOR”, and “NOT” instructions differ from each other?*

## Problem #4.

*Slide to the left...*

*Slide to the right...*

*Criss-cross!*

*Criss-cross!!!*

A student from UPV likes to enroll in the BS in Computer Science program, however, he needs to SHIFT to do so since he is currently a BS Accountancy student. He can either shift to the left or to the right depending on his situation. Help him weigh his options on which one is more suitable for him.

- Write an assembly program that implements the left shift and right shift operations.

**Note:**

**<< (Left shift):** Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift. Or in other words left shifting an integer “x” with an integer “y” ( $x \ll y$ ) is equivalent to multiplying x with  $2^y$  (2 raised to the power of y).

**>> (Right shift):** Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift. Similarly, right shifting ( $x \gg y$ ) is equivalent to dividing x with  $2^y$ .

- Design a high-level programming logic for the problem above and translate it to assembly language.
- Use bit operations (*shifts and bitwise operations*) to solve the problem.
- The output of your program should be something like this:

```
Enter a number: 5
Enter the number of places to shift: 1
5 << 1 is 10
5 >> 1 is 2
```

- A good programming practice is to *write comments on important line of codes* for readability and documentation.
- Save your program in a file called *SurnameFirstLetterOfFirstName\_lab4.asm* in camel case. For instance, if your surname is “Juan Dela Cruz”, submit it as follows:

*DelaCruzJ\_lab4.asm*

- Take a screen recording of your working code and make sure to **record a video explaining each line of your code** as well as showing the correct output of your code. Use screen recorder application in Ubuntu (<https://itsfoss.com/best-linux-screen-recorders/>) or Windows (<https://atomisystems.com/screencasting/record-screen-windows-10/>)

### Submission Requirements:

1. Program Code (‘.asm’ file)
2. Screen Recorded Defense Video

**DEADLINE: November 3, 2022, 11:59 PM**

<b>Rubric for Programming Exercises</b>				
<b>Program (50 pts)</b>	<b>Excellent</b>	<b>Good</b>	<b>Fair</b>	<b>Poor</b>
<b><i>Program Execution</i></b>	Program executes correctly with no syntax or runtime errors (9-10)	Program executes with minor (easily fixed) error (4-8)	Program executes with a major (not easily fixed) error (2-3)	Program does not execute (0-1)
<b><i>Correct Output</i></b>	Program displays correct output with no errors (9- 10)	Output has minor errors (6-8)	Output has multiple errors (3-5)	Output is incorrect (0- 2)
<b><i>Design of Output</i></b>	Program displays more than expected (7-8)	Program displays minimally expected output (5-6)	Program does not display the required output (3-4)	Output is poorly designed (0-2)
<b><i>Design of Logic</i></b>	Program is logically well-designed (9-10)	Program has slight logic errors that do not significantly affect the results (6-8)	Program has significant logic errors (3-5)	Program is incorrect (0-2)
<b><i>Standards</i></b>	Program is stylistically well designed (6-7)	Few inappropriate design choices (i.e., poor variable names, improper indentation) (4-5)	Several inappropriate design choices (i.e., poor variable names, improper indentation) (2-3)	Program is poorly written (0-1)
<b><i>Documentation</i></b>	Program is well documented (5)	Missing one required comment (4)	Missing two or more required comments (2- 3)	Most or all documentation missing (0-1)