

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 #5

### Reading

- Read [Section 4.6 until 4.8 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**

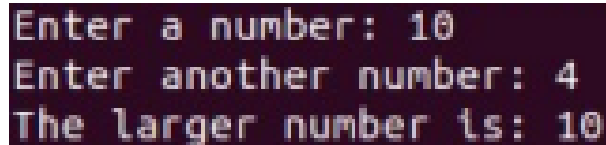
- Compile and link the assembly code with the C program (**driver.c**). In our machine, we will be using 32-bit registers thus we specify “-m32”.

```
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 shows 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 #5.

*Person 1:* What bitwise operation did the pirate say to his comrades?

*Person 2:* --

*Person 1:* ORRRR!!!

*Person 2:* I have a better one!

*Person 1:* Oh! Do tell!

*Person 2:* What bitwise operations form a song?

*Person 1:* What?

*Person 2:* We're XOR AND! Flyin! There's not a star in heaven that we can't reach.

*Person 1:* --

*Person 2:* Now, we're even! --

- Write an assembly program that implements the Bitwise OR, AND, and XOR operations.

*Note:*

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(A   B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001

- Use Bitwise assembly operations to solve the problem.
- The output of your program should be something like this:

```
Enter a number: 12
Enter another number: 25
12 & 25 is 8
12 | 25 is 29
12 ^ 25 is 21
```

- 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\_lab5.asm* in camel case. For instance, if your surname is “Juan Dela Cruz”, submit it as follows:

*DelaCruzJ\_lab5.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**

Rubric for Programming Exercises				
Program (50 pts)	Excellent	Good	Fair	Poor
<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)