

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 #2

Reading

- Read [Section 2.2 of Paul Carter's PC Assembly Book](#)

Practice Exercise:

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

nasm -f elf prime.asm

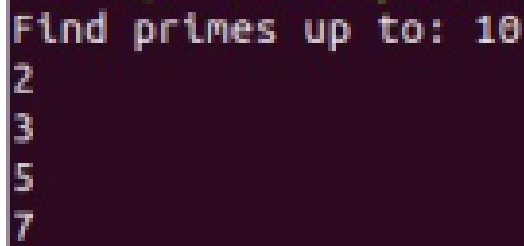
- Compile and link the assembly code with the C program (**driver.c**). In our machine, we will be using 32-bit registers thus we specify “-m32”.

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

- Execute the assembly code.

```
./prime
```

The code should show the following:

A terminal window with a dark background and light-colored text. The text reads: "Find primes up to: 10" on the first line, followed by "2", "3", "5", and "7" on subsequent lines, each on a new line.

```
Find primes up to: 10
2
3
5
7
```

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

What does the program do?

How does “cmp” and “je” instructions differ from each other?

How does “jnb” and “jnbe” instructions differ from each other?

Problem #2.

Waking up after a glorious dream, you became excited since tomorrow you will be celebrating the birthday of your best friend whom you haven't met in a long time. You went to the mall where you decided to buy a gift for him. Suddenly, he calls you. You were confused but decided to answer it anyway. "I won't be celebrating my birthday this year, so don't expect anything.", he said in total dismay. You check the calendar, and you were in total shock. "Tomorrow is the first of March! It's a leap year", you screamed thinking he won't be having his birthday in yet another 4 years because his birthday is on February 29.

- Write an assembly program that checks whether the year entered by the user is a leap year or not.

Note:

A leap year is a year exactly divisible by 4 except for the century years (or years ending in '00'). The century year is only a leap year when it is perfectly divisible by 400.

- Use control structures (*cmp, jmp, etc.*) to solve the problem.

- The output of your program should be something like this:

Output #1:

```
Enter a year: 1800
The year '1800' is NOT a leap year.
```

Output #2:

```
Enter a year: 2000
The year '2000' is a leap year.
```

Output #3:

```
Enter a year: 2018
The year '2018' is NOT a leap year.
```

Output #4:

```
Enter a year: 2020
The year '2020' is a leap year.
```

- Please use the TEST CASES below to verify if your code produces the correct output.

Year	Leap Year?
1600	✓
1700	✗
1800	✗
1904	✓
1906	✗
1908	✓
1950	✗
1963	✗
1977	✗
1984	✓
1999	✗
2000	✓
2003	✗
2004	✓
2008	✓
2012	✓
2013	✗
2018	✗
2019	✗
2020	✓

2021	×
2022	×
2024	✓
2100	×
2400	✓

- 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_lab2.asm* in camel case. For instance, if your surname is “Juan Dela Cruz”, submit it as follows:

DelaCruzJ_lab2.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: October 6, 2022, 11:59 PM

Rubric for Programming Exercises				
Program (50 pts)	Excellent	Good	Fair	Poor
<i>Program Execution</i>	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)
<i>Correct Output</i>	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)
<i>Design of Output</i>	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)
<i>Design of Logic</i>	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)
<i>Standards</i>	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)
<i>Documentation</i>	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)