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 prime.asm.

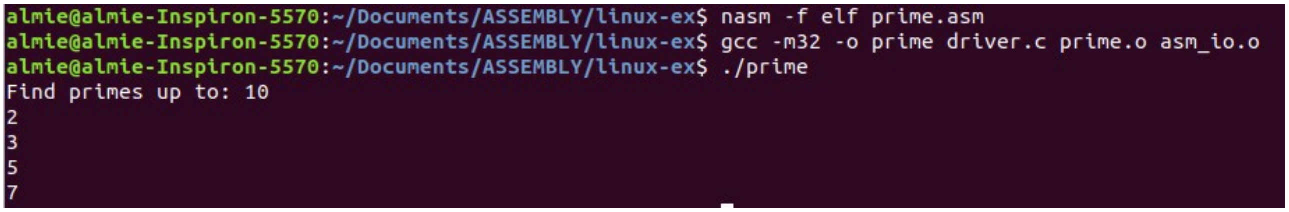
```
  nasm -f elf prime.asm
```

- 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
```

- Execute the assembly code.

```
  ./prime
```



```
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ nasm -f elf prime.asm
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ gcc -m32 -o prime driver.c prime.o asm_io.o
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ ./prime
Find primes up to: 10
2
3
5
7
```

- Analyze the sample code (prime.asm). Reflective questions:

```
  What does the prime.asm do? How does “cmp” and “je”
instructions differ from each other?  How does “jnb” and “jnbe”
instructions differ from each other?
```

Problem #2.

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

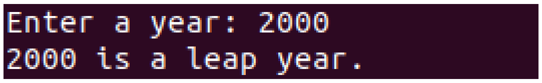
A leap year is exactly divisible by 4 except for century years (years ending with 00). The century year is a leap year only if it is perfectly divisible by 400.

For example,

- 1999 is not a leap year
- 2000 is a leap year
- 2004 is a leap year

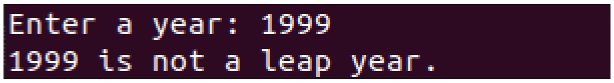
- Use control structures (*like cmp, je, etc*.) to solve the problem.
- The output of your program is something like this:

Output #1:



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

Output #2:



```
Enter a year: 1999
1999 is not a leap year.
```

- A good programming practice is to write comments on important line of codes for readability and documentation.
- Save your program in a file called surname_lab2.asm. For instance if your surname is “Dela Cruz”, submit it as follows:

```
delacruz_lab2.asm
```

Note: Take a screen record 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/)

Deadline :____

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