Laboratory Exercise #7

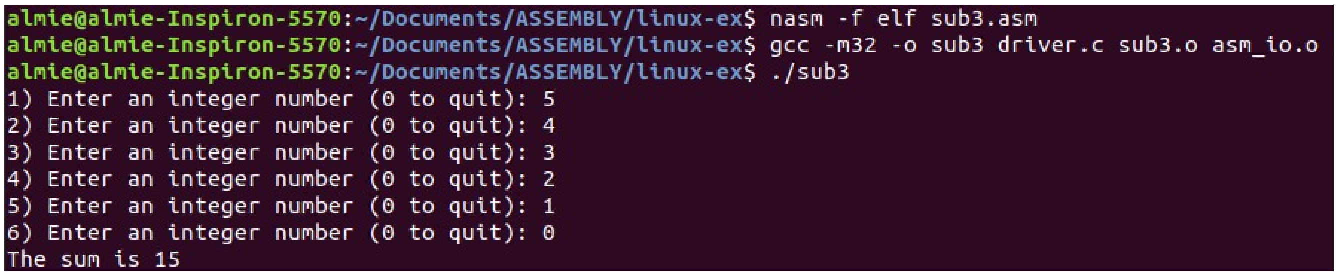
Reading

- Read Section 4.6 of Paul Carter's PC Assembly Book

Practice Exercise:

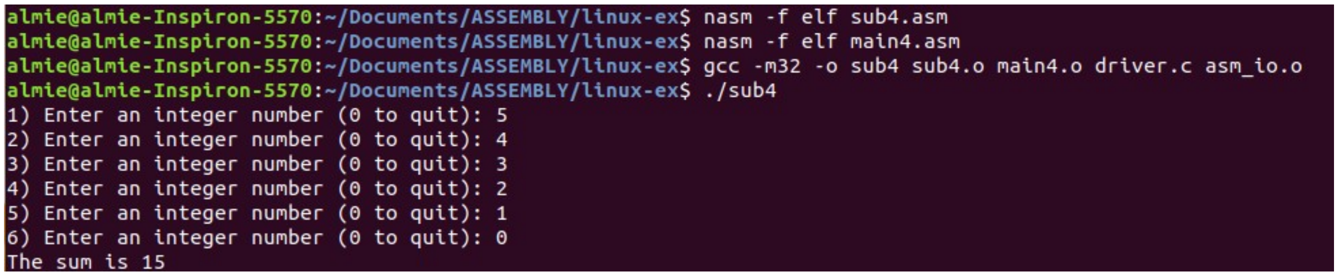
- Compare and contrast “sub3.asm” and “sub4.asm”.

sub3.asm



```
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ nasm -f elf sub3.asm
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ gcc -m32 -o sub3 driver.c sub3.o asm_io.o
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ ./sub3
1) Enter an integer number (0 to quit): 5
2) Enter an integer number (0 to quit): 4
3) Enter an integer number (0 to quit): 3
4) Enter an integer number (0 to quit): 2
5) Enter an integer number (0 to quit): 1
6) Enter an integer number (0 to quit): 0
The sum is 15
```

sub4.asm



```
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ nasm -f elf sub4.asm
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ nasm -f elf main4.asm
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ gcc -m32 -o sub4 sub4.o main4.o driver.c asm_io.o
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ ./sub4
1) Enter an integer number (0 to quit): 5
2) Enter an integer number (0 to quit): 4
3) Enter an integer number (0 to quit): 3
4) Enter an integer number (0 to quit): 2
5) Enter an integer number (0 to quit): 1
6) Enter an integer number (0 to quit): 0
The sum is 15
```

- Explore **main4.asm.** Reflective questions:

```
   What is the purpose of main4.asm? What is the purpose of
sub4.asm? Explain the relationship between main4.asm and sub4.asm.
```

- Analyze the sample codes (sub3.asm and sub4.asm). Reflective questions:

```
   What is the name of the subprogram in the 2 assembly programs?
What is the major difference between the 2 assembly programs? What
are multi-module programs?
```

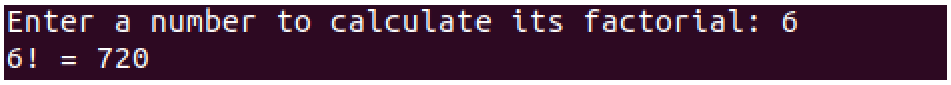
Problem #7.

- Write an assembly program that computes the factorial of a number.
- Below is the code snippet in high level language (C language). Translate it into assembly language program using subprogram (name your subprogram as **factorial**). However, implement a multi-module program for this problem (just like in the example “**sub4.asm**” and “**main4.asm**”) where you have “**factorial.asm**” that consists of subprograms **get_int** and **factorial**. “**main.asm**” consists of external subprograms **get_int** and **factorial**.

```c
int main(){

   int n;

   printf("Enter a number to calculate its factorial:");
   scanf("%d", &n);

   printf("%d! = %d\n", n, factorial(n));

   return 0;
}

factorial(int n){
   int c;
   long r = 1;

   for (c = 1; c <= n; c++){

       r = r * c;

   }

   return r;
}
```

- Run two programs “factorial.asm” and “main.asm” implementing multi-module programs.

- The output of your program is something like this:



```
Enter a number to calculate its factorial: 6
6! = 720
```

- A good programming practice is to write comments on important line of codes for readability and documentation.
-

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)