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 #0

Reading

- Read Section 1 .3 and 1.4 of Paul Carter's PC Assembly Book

Practice Exercise:

- Make sure to download and install Nasm (https://www.nasm.us/xdoc/2.12/html/nasmdoc1.html)
- Once installed, download the sample Linux assembly code here: (http://pacman128.github.io/pcasm/)
- Extract the Linux example code folder and go to the directory of the said folder.
- If first time to run Nasm, assemble and run the command (See **asm_io.asm** for the command on different operating system):

```
nasm -f elf -d ELF_TYPE asm_io.asm
```

- Then assemble the first assembly code (**first.asm)**. This will create an object file (**first.o**) for first.asm.

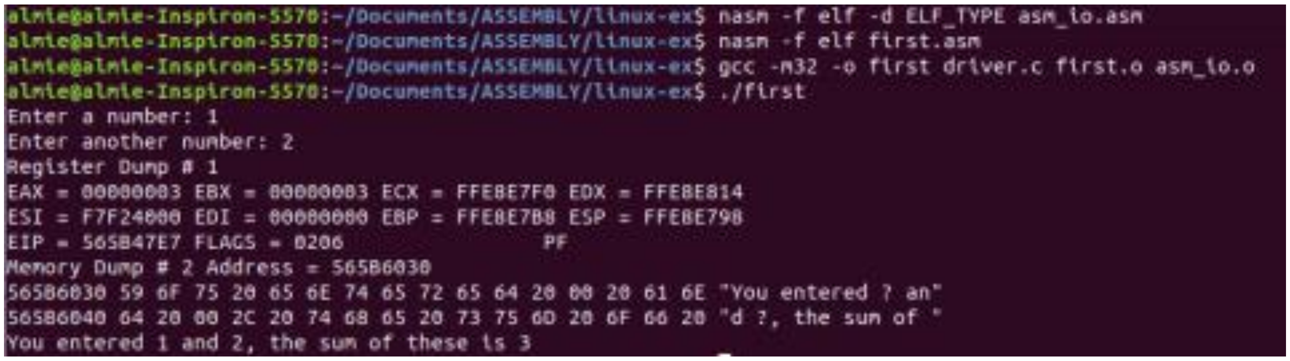
nasm -f elf first.asm

- 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 first driver.c first.o asm_io

- Execute the assembly code.

  ./first



```
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ nasm -f elf -d ELF_TYPE asm_io.asm
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ nasm -f elf first.asm
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ gcc -m32 -o first driver.c first.o asm_io.o
almie@almie-Inspiron-5570:~/Documents/ASSEMBLY/linux-ex$ ./first
Enter a number: 1
Enter another number: 2
Register Dump # 1
EAX = 00000003 EBX = 00000003 ECX = FFE8E7F0 EDX = FFE8E814
ESI = F7F24000 EDI = 00000000 EBP = FFE8E7B8 ESP = FFE8E798
EIP = 565B47E7 FLAGS = 0206                PF
Memory Dump # 2 Address = 565B6030
565B6030 59 6F 75 20 65 6E 74 65 72 65 64 20 00 20 61 6E "You entered ? an"
565B6040 64 20 00 2C 20 74 68 65 20 73 75 6D 20 6F 66 20 "d ?, the sum of "
You entered 1 and 2, the sum of these is 3
```

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

  What does the program do?

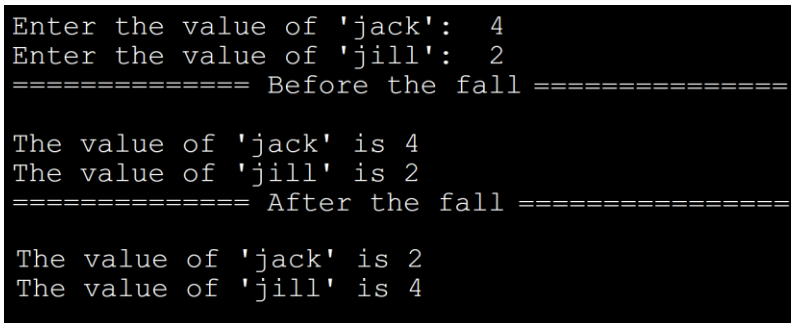
  What does it mean to compile an assembly program?

  How does the assembly program get executed?

Problem #0.

Jack and Jill are each carrying a number of pails while climbing up the hill to fetch water for their home. While climbing, Jack fell down and broke his crown causing the pails to fall on the opposite side of the hill. Jill, shocked by what she saw came tumbling in the direction of Jack's pails which also caused her to lose her pails as they fell down as well in Jack's direction. After the accident, Jack and Jill got up and were forced to bring the other's number of pails back up the hill.

- Write an assembly program that creates two variables **jack** and **jill**, representing the number of pails they each have at the start.
- Have the program get inputs from the user and store it in variables jack and jill. Take note that values of jack and jill should be integer values from the user, respectively.
- The program should print the two values of the variables, identifying the number of pails before the fall.
- Swap the corresponding values of **jack** and **jill**, to represent the falling of pails in the other's direction.
- Print the two values (**jack** and **jill**) again, identifying them.



```
Enter the value of 'jack':  4
Enter the value of 'jill':  2
============== Before the fall ===============

The value of 'jack' is 4
The value of 'jill' is 2
============== After the fall ================

 The value of 'jack' is 2
 The value of 'jill' is 4
```

- 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_lab0.asm*** in camel case. For instance, if your surname is "Juan Dela Cruz", submit it as follows:

DelaCruzJ_lab0.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: September 21, 2022, 11:59 PM

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)