Additional Help

Since, I want you to make sure we are all on the same page, I have decided to give some more assistance. After this, you will be on your own in completing the assignment. Based on conversations with a small subset of the class, I am concerned in the direction in which some students are taking.

Remember the assignment is basically simulating an extended version of the first tutorial. In order for you to complete this, you will first think about the sequence of steps to print the fib sequence to standard output (I need to see the output, so here you would need to actually print to standard output).

I really did not expect anyone to be thinking of solving this recursively as that would not make any sense for simpleOS with the limited properties. With that said, let us consider two (2) versions of the fib sequence in Python code:

Version 1:

```
def fib1():
    a = 0
    b = 1
    n = 5
    counter = 0
    while (counter < n):
        print(a)
        a = a+b
        b = a-b
        counter = counter + 1
```

Version 2:

```
def fib2():
    a = 0
    b = 1
    while (a >= 0):
        print(a)
        a = a+b
        b = a-b
```

These are basic instructions that SimpleOS can follow. You should realize by now that you will need additional instructions.

You could create the following instructions: GOTO, COMP and HALT

The GOTO would just jump to a particular location that could be based on a comparison COMP. You then could use the HALT instruction to end your program.

NB: Please note that in Version 2 of the code above, you will need to manually intervene to end the program, and to make things easier for you, I will be quite fine with this implementation. Version 2 has less variables to keep track of.

Additionally, you can also use additional registers to assist you if you wish.

Please remember that you should not literally take the input from your machine standard input, but you must implement the sequence of statements that will do that. This is also similar to what we did with the first tutorial when we demonstrated loading from I/0 and storing to I/0. It is all about loading from standard input and storing to standard output.

Another important thing to remember, is that the array of integer for the memory, as shown in the actual helper code should be maintained. This means that although the instructions are in binary, these instructions should be converted to decimal when you are storing them. This also implies that you will need a mechanism to translate from binary to decimal and decimal to binary.

*** All the best ***