CS 240
Fall 2019

Project 2
More Functions

Assigned: Monday, September 9, 2019
Due: 10:00 p.m., Sunday, September 15, 2019

There are five problems here. Do them, commenting neatly and copiously. Use `.8086` mode and 16-bit registers as usual. Make wise choices about data types and memory usage. Adhere to the calling convention for each function and make sure to not disturb the state of the machine (don't leave any footprints like changing register values or flags). Functions that leave footprints will be penalized signficantly.

1. **Polynomial**: Write the function `Polynomial` to compute $ax^2 + bx + c$. The value a is passed in `AX`, b in `BX`, c in `CX`, and x in `DX`. You should return the result in `AX`.

2. **Factorial**: Write the function `Factorial` that computes $x!$ (x factorial). x should be passed in `AX`. You should return the result in `AX`. `Factorial` should set the overflow flag on overflow and should clear it otherwise.

3. **Fibonacci**: Write the function `Fibonacci` that computes the i^{th} Fibonacci number in the sequence 1, 1, 2, 3, 5, 8, 13, ... The value of i is passed in `AX`. (The third Fibonacci number is 2.) You should return the result in `AX`.

4. **PrintString**: Write the function `PrintString` that works identically to the link library's `WriteString`. To write a single character, invoke the DOS function as described here. The the following sequence of assembly instructions will interrupt to DOS in order to write the character placed in `DL` to the standard output. (The example below outputs the letter `A`)

   ```
   mov dl, 41h ; Put the ASCII value of the character you want written into DL
   mov ah, 02h ; Set ah to the DOS code for write-character.
   int 21h     ; DOS!
   ```

5. **PrintHexDigit**: Write the function `PrintHexDigit` that works identically to the link library's `WriteHexDigit`. *Hint:* Use an array of characters with indirect addressing (section 4.4, pp. 117–123). Take care here. Only certain registers can be used as memory offsets; and be sure everything is initialized properly.

I will link test code to your functions, so do not define an entry point in your code. Your code should end with the assembly language statement:

```
END
```

without an entry point designation.

How to submit. Submit the following file: `funcs.asm` using the standard course submission procedure below.

1. At the DOS prompt, remove the flash drive.
2. Reboot the computer into Windows (or your operating system)
3. Reinsert the flash drive.
4. Transfer the files to `gemini.cs.hamilton.edu` (if you don't know how to do this, you'll need to research it).
5. Log in to `gemini.cs.hamilton.edu`
6. `[user@gemini ~]$ cs240`
7. `[user@gemini ~]$ submit`

Submit will not be open until 24 hours before the assignment is due. You may submit as many times as you want up to the deadline. Your final submission will be used in grading.