CMPSC 210
Principles of Computer Organization
Fall 2016
John Wenskovitch
http://cs.allegheny.edu/~jwenskovitch/teaching/CMPSC210

Lab 8 – Procedures
Due (via Bitbucket) Wednesday, 16 November 2016
30 points

Lab Goals

- Practice using procedures in MIPS
- Solve two programming challenges using MIPS

Assignment Details

We have spent several recent classes discussing the use of procedures in MIPS – what they are, how to use them, and what standards and conventions we have developed to organize the use of registers in a clean and consistent manner. In this lab, you will implement two programs that make use of procedures.

Note that you have two weeks to finish this lab, but start early! You can complete this lab with a partner, if you so choose.

Part 1: Guess My Number, Human Player (15 points)

Write a MIPS program that implements the "Guess My Number" game. The program should choose a number between 1 and 100 inclusive, and prompt the user to guess that number. After each guess, the system should print whether the guess was too high, too low, or correct. If correct, the program should print the number of guesses that the user made, and then prompt the user to play again. Sample run:

```
Welcome to this Guessing Game, human!
I'm thinking of a number between 1 and 100.
See if you can guess it!
Your guess: 50
Your guess was too high!
Your guess: 45
Your guess was too high!
Your guess: 40
Your guess was too low!
Your guess: 42
```

```
You got it in 4 tries!  My number was 42.

Would you like to play again (y/n)? y

Welcome to this Guessing Game, human!
I'm thinking of a number between 1 and 100.
See if you can guess it!
Your guess: 19
Your guess was too high!
Your guess: 7
You got it in 2 tries!  My number was 7.

Would you like to play again (y/n)? n
```

Your implementation of this game **MUST** use procedures. You are not limited to this list, but you must have a separate procedure for each of the following functionalities:

1. Welcome / Introduction messages
2. Get input from user (guess)
3. Determine and report high/low
4. Get input from user (play again)

You are welcome to create a main game loop procedure, or to leave that in the "main" portion of the code. The procedures that you implement **MUST** use the proper conventions that we have discussed, namely:

- Use the `$a` registers to pass values into each procedure (if necessary)
- Use the `$v` registers to return values from each procedure (if necessary)
- Use prologue and epilogue code blocks to save/restore important registers to/from memory before and after the procedure body

Part 2: Guess My Number, Computer Player (15 points)

In the previous section, you wrote a MIPS program that plays the "Guess My Number" game. Now, you will modify that program to switch roles:

- YOU will choose the number.
- The COMPUTER will attempt to guess the number.
- On each iteration, YOU must inform the computer whether its guess is too high, too low, or correct.

- The COMPUTER will repeat your feedback (useful for debugging).
- When guessing correctly, the COMPUTER will report how many guesses it took, and then ask if you want it to guess again.

When giving feedback to the computer about its guess, you should use the following:

- 1 = The computer's guess is too high.
- 0 = The computer's guess is correct.
- -1 = The computer's guess is too low.

Your input prompts and output labels should look similar to the ones in this sample run:

```
Welcome to this Guessing Game, human!
Please pick a number between 1 and 100.  Don't tell me what it is.
My first guess is 50.  How did I do?  1

Hmmm, my guess was too high.  My next guess is 25.  How did I do?  -1

OK, my guess was too low.  My next guess is 37.  How did I do?  1

Hmmm, my guess was too high.  My next guess is 31.  How did I do?  1

Hmmm, my guess was too high.  My next guess is 28.  How did I do?  0

Yay, I got it in 5 tries!  Your number was 28.
Would you like me to guess again (y/n)?  y

Welcome to this Guessing Game, human!
Please pick a number between 1 and 100.  Don't tell me what it is.
My first guess is 50.  How did I do?  1

Hmmm, my guess was too high.  My next guess is 25.  How did I do?  0

Yay, I got it in 2 tries!  Your number was 25.
Would you like me to guess again (y/n)?  n
```

There are two ways that you can have the computer guess your number. In the first case, the computer can simply try every number between 1 and 100 in order. This will give you partial credit. In the second case, you can keep track of the largest and smallest number that it could possibly be, and always guess the number in the middle. This will give you full credit.

Note that I do not expect your program to detect when you are lying to the computer about your number. If you want to feel superior to the machine, be my guest, but make sure that your program will respond to "real" input as well.

Your implementation of this game **MUST** use procedures. You are not limited to this list, but you must have a separate procedure for each of the following functionalities:

1. Welcome / Introduction messages
2. Compute a reasonable guess for the computer player
3. Get input from user (feedback on computer's guess)
4. Success message when the computer gets it right
5. Get input from user (play again)

You are welcome to create a main game loop procedure, or to leave that in the "main" portion of the code. The procedures that you implement **MUST** use the proper conventions that we have discussed, listed in the previous section.

Submission Details

For this assignment, your submission to your BitBucket repository should include the following:

1. A commented version of your Guess My Number, Human Player code.
2. A commented version of your Guess My Number, Computer Player code.

Before you turn in this assignment, you also must ensure that the course instructor has read access to your BitBucket repository that is named according to the convention `cs210f2016-<your user name>`.