Quest 4: Game On

CS 2323

Computer Organization

(1000 xp)

Due: Monday, April 21, 11:59 PM

Objective:
Create a game using MIPS assembly language. This game should run in a MIPS simulator, such as SPIM or MARS. It should make use of ASCII characters to display the action. You may implement one of the games outlined below, or you may implement another game, but ask me first if it will be acceptable. You may work in groups of two.

Suggested Game 1: Hangman
The hangman game should include a list of words or phrases stored in ASCII values. A word should be selected from this list (in whatever order you see fit). The word or phrase should be presented to the user as blanks, along with a "gallows" to indicate the player's progress, and an initially empty list of guessed letters. The player should then be prompted to guess a letter. If the word does not contain that letter, or the player guesses a letter they have already guessed, a "body part" should be added to the gallows. If the word does contain the letter, all blanks in the word corresponding to that letter should be filled with that letter. Keep in mind that your program should accept both upper and lowercase letters. You should then add the letter to the list of guessed letters, and this list of letters should be displayed.

A player is victorious if they fill every blank with letters. A player loses if the entire body of the hangman is placed upon the gallows. The body parts should at least include head, torso, two arms, and two legs. Either way, the player should be asked if they want to play again. If they player answers yes, a new word should be selected and a new game begun. Otherwise, exit the program.

Optional Features (for additional xp):

- Select the word at random from the available words. (May require MARS)
- Create a multiplayer mode where a player secretly enters a word or phrase for the other player to guess. The input field should replace characters with asterisks, and respond to backspace.

Suggested Game 2: Connect Four
Create a multiplayer version of Connect Four. Connect four is played on a 7x6 grid, by two players: red and black (or X and O if you prefer). Players take turns dropping pieces into a column of the board. The piece falls to the lowest point in the column that is still available. A player wins if they manage to put four of their pieces consecutively in a straight line: vertically, horizontally, or diagonally. Your program should display the 7x6 grid, then ask a player which column they wish to play. The result of this move should be displayed, and then the program should determine if one of the players has won. If a player selects a full column, they forfeit their turn.

The game ends when one player wins or the board is filled without a winner, in which case there is a tie. Players should then be prompted as to whether they wish to play again, and if they answer in the affirmative, a new game should begin with a fresh board. Otherwise, the program should exit.

Optional Features (for additional xp):

- Implement a single-player version of the game with an AI that will play well.
- Allow players to specify the size of the board before the game begins (within an acceptable range… at least 4x4)

Notes on Grading
Programs will be granted experience based on the following:

Correctness: 700xp. The game must execute correctly and be playable.
Documentation: 200xp. The code must be sufficiently documented so that I don't have to decipher what a section of code is doing.
Presentation: 100xp. The game should be presented in as aesthetically pleasing a way as ASCII interface allows.
Bonus Points: Discretionary. Implementing optional features will grant bonus points. You may also come up with additional features for more points, but please ask me first if they will be allowed. Also, if you decide to do your own game, I will likely provide some suggested optional features.

Basic Strategies
You'll need to make use of the data section to store relevant data, and be able to manipulate the bytes therein. You'll need to make extensive use of the system calls for output and input, and it's probably a good idea to create procedures that display the game elements.

Notes on MARS
MARS is an enhanced MIPS simulator with a greater set of system calls and better support for the full MIPS instruction set. It should not be much more difficult to use than SPIM, but I do not require that it be used. MARS has system calls for: file I/O (also available in SPIM), system time, MIDI output, random number generation, and other things. If you choose to use this, let me know when you turn in your project.

It can be found at: `http://www.cs.missouristate.edu/MARS/`