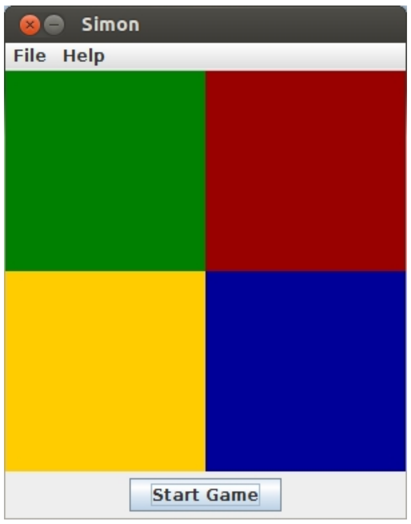
Project - Simon

CS 0447 — Computer Organization & Assembly Language

See CourseWeb for Due Date

The purpose of this project is for you to practice writing assembly language to interact with output/input hardware. The hardware for this project is a game called Simon (on left):



The Simon (Mars Tool) that we are going to use for this project is shown above (on right). This tool can be found in `simon.zip` located in the CourseWeb under this project. Extract all files to your `[..]/mars4_5/mars/tools` directory. If you extract all files to the right directory, when you run the MARS program, you should see "Simon (Register) V0.1" under the "Tools" menu.

Introduction to the Simon Game

Simon is a classic game from the 70'. See `http://en.wikipedia.org/wiki/Simon_(game)` for more detail. During the game play, the computer will play a sequence of tone and lit a button associated to each tone. Each round, a player has to play the sequence back by pressing a series of buttons. The length of the sequence starts at 1 at round 1 and increases by 1 for every round. At n^{th} round where $n \geq 2$, The computer will play a sequence of length n where the first $n-1$ tones are exactly the same as the sequence at round $n-1$ and the n^{th} tone is a new one (randomly pick a new one). The game is over when a player plays a wrong sequence.

Introduction to the Simon (Mars Tool)

The Simple Calculator (Mars Tool) consists of four colored buttons and one start game button. These buttons are connected directly to the register `$t9`. If the value stored in the register `$t9` is currently 0, when a button is pressed, the value of the register `$t9` will be changed according to the pressed button as shown below:

Button	Value of `$t9`
Blue	1
Yellow	2
Green	4
Red	8
Start Game	16

Note that if the content of the register `$t9` is not equal to 0, these buttons are disabled. Thus, it is the programmer responsible to change the content of the register `$t9` to 0 when the program is ready to accept the next input.

For this Simon (Mars Tool), the hardware is simply an input device. It is the program responsibility to tell the hardware to light a button and play sound, play starting sound, and play game over sound. A command can be sent to the Simon game by simply change the value of the register `$t8` from 0 to a specific value as shown below:

Value of `$t8`	Command
1	light the blue button and play its associated tone
2	light the yellow button and play its associated tone
4	light the green button and play its associated tone
8	light the red button and play its associated tone
15	play the game over tone, light all colored button three times, and enable the Start Game button
16	disable the Start Game button and play starting sound

Note that after the Simon game receives a command, it may take some time to process. After a received command has been processed, the Simon game will change the content of the register `$t8` to 0. Thus, it is the programmer responsibility to wait until the content of the register `$t8` is changed to 0 before sending the next command.

What to Do?

For this project, write a MIPS assembly program named `simon.asm` such that when the program is running, the Simon game will behave just like an actual Simon game hardware. Note that when a player plays a wrong sequence, the Simon game should play the game over tone, light all colored button three times, and enable the "Start Game" button for the next player without restarting the program.

Requirements

1. Since registers `$t8` and `$t9` are used for sending command and receiving button input, do not use these registers for any other purpose.
2. Your program must consist of at least two functions as follows:
   - `playSequence`: This function, when called, it will play a sequence of tones.

- `userPlay`: This function lets user play back the sequence. Note that this function should return a value to notify the caller whether user successfully play back the sequence or not.

3. You are allowed to have more functions.
4. Everything that a function needs should be passed as arguments.
5. Everything that a caller needs from a function must be passed as return values.
6. Every functions must follow the calling convention discussed in class.

Submission

The due date of this project is on the CourseWeb. Late submissions will not be accepted. You should submit the file `simon.asm` via CourseWeb.