Project Explanation

Chaoneng Quan

Overview

This project is named Multiple Choice Question Simulator(MCQS), MCQS is a console-based program in MIPS assembly language that simulating college student's typical scenario during final exams. Specifically, students have to guess the correct answer for a multiple-choice question. MCQS implemented three extra syscalls in MARS IDE, including random int range(syscall 42), read integer(syscall 5), and MIDI out synchronous(syscall 33).

Scenario

Joe is a computer science major student at University of Arizona. Final exam is tomorrow, but he did not study at all(of course) not to mention that he did not even understand the exam study guide. Therefore, he opens this Multiple-Choice Question Simulator to see how lucky he is, trying to get some comfort from playing simulation game, which does not help his grade at all.

Stages

Once the Run button is pressed on MARS IDE, playStartingSound() function will first be called, it will set the value of $v0 to 33, which is the service code for MIDI out synchronous. Then by changing the values of $a0-$a3 then syscall, a short famous tune will be played. Second, printStartingMessages() will be called, it will set the value of $v0 to 4, which is the service for printing strings, messages will be printed on console that shows game has started. Third, in the main() function body, $v0 is set to 42. That is the service code for generating random int within a range, and rage in stored in $a1, the generated random number is stored in $s0, the number of the user has tried(numTried) is default to 0 and stored in $s2. Fourth, promptAndCheckUserInput(generatedNumber) will be called until the function returns 0, which represents that user has successfully guessed the answer. This function returns 1 which represents that the user input is lower than the generated number. Or returns 2 represents that the user input is higher that the generated number. Fifth, if the function returns 0, guessedCorrectly() will be called that prints a message on console indicates that the user has won the game. Finally, printQuip(numTried) will be called then print a quip based the number of times user has tried on console. Then set $v0 to 10 to exit the program.

Code

The pseudo-code in the main() function is as below:

```
main(){
        playStartingSound();
        printStartingMessages();
        While(userInput != generatedNumber){
                if(promptAndCheckUserInput(generatedNumber) == 0)
                        guessedCorrectly();
                        printQuip(numTried);
```

```
                if(promptAndCheckUserInput(generatedNumber) == 1)
                        guessedTooLow();
                if(promptAndCheckUserInput(generatedNumber) == 2)
                        guessedTooHign();
                numTried++;
        }
    }
```

In the code, there are 7 private functions. playStartingSound(), printStartingMessage(), promptAndCheckuserInput(generatedNumber), printQuip(numTried), printGuessedCorrectly(), printGuessedTooHigh(), and printGuessedTooLow().

For the playStartingSound() method, there are four parameters for the syscall. $a0 represent the pitch, $a1 represents the duration, $a2 represents the instrument, $a3 represents the volume. For the promptAndCheckuserInput(generatedNumber) function, it takes in a parameter which holds the generated random value passed from main, then return a value represent if they are equal. For the printQuip(numTried) function, it takes in a parameter which represents how many times the user has tried until correctly guess the answer, then prints the quip according to that value. For printGuessedCorrectly(), printGuessedTooHigh(), and printGuessedTooLow() functions, $v0 is set to 4 and $a0 is holding the address of the string that will be printed on the console.