Hao Huynh

Professor Mark Lehr

CSC 11 – 48982

30 October 2015

Project 1: Paper-Rock-Scissors Game.

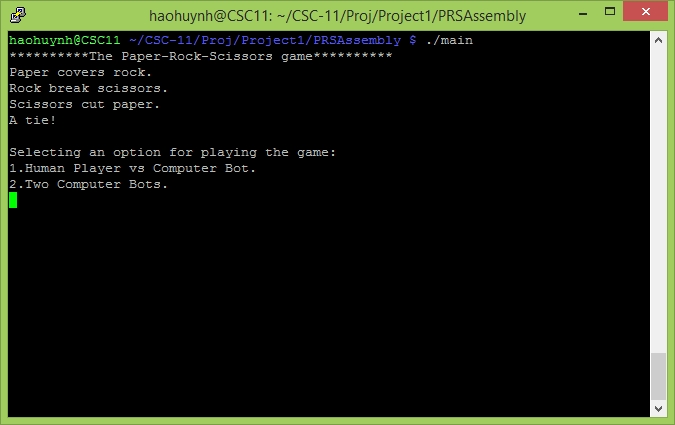
1. Introduction.

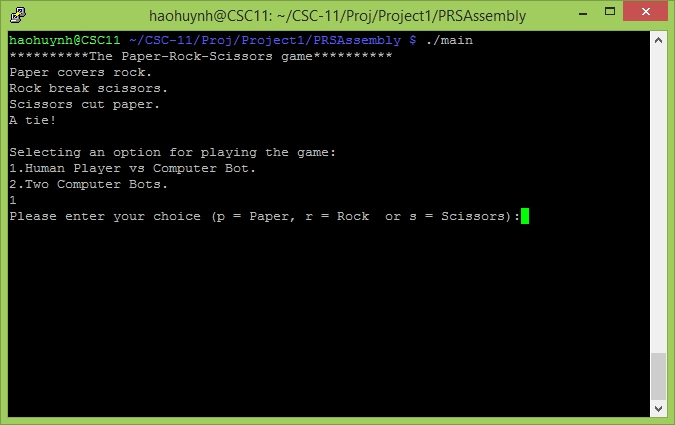
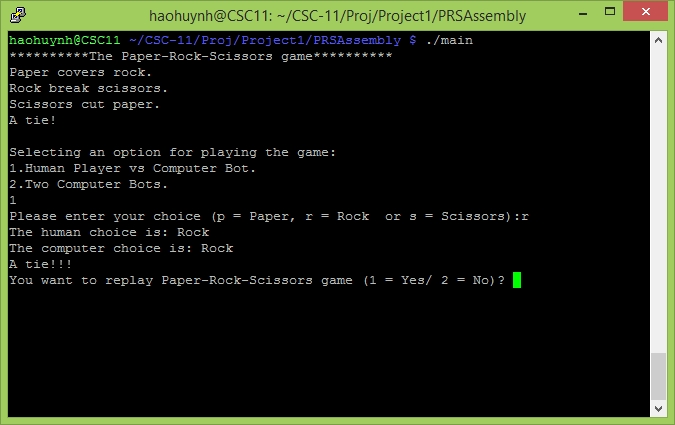
Paper-Rock-Scissors is a popular and simple game played between two players, and the players must form one of those three shape at the same time. In this game, the player can be either human or computer. Currently, the game supports two cases: human vs computer and computer bot vs computer bot. In addition, the winner is determined by the flowing rules:

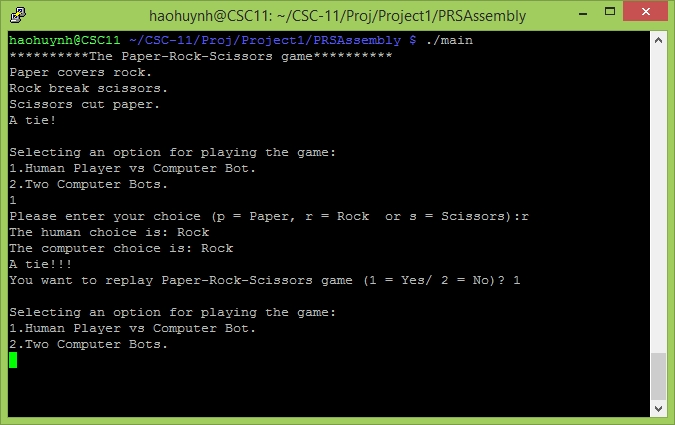
* Paper covers rock.
* Rock breaks scissors.
* Scissors cut paper.
* A tie (both of two players have the same shape).

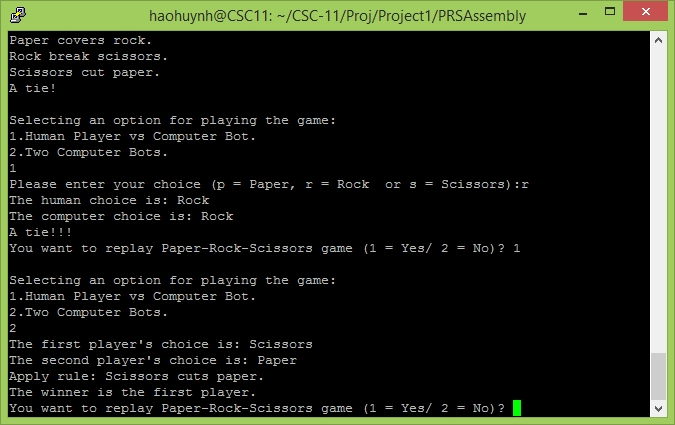
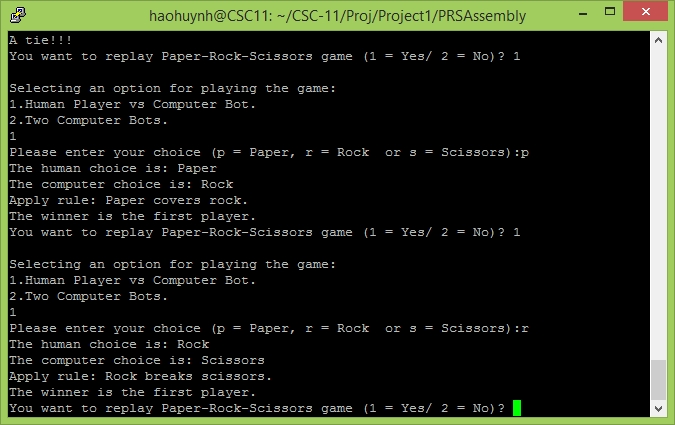
I decided to implement this game because the C++ code of its program is actually not complex and long, so I could finish it within a limited period of time. However, I still could demonstrate techniques such as calling external C functions as well as internal functions from an Assembly program, applying basic computational mathematics, and using Assembly constructs including if, if-else, switch, while, do-while, and for.

1. Summary.
   1. The program statistics:
      1. The number lines of code (LOC) are about 600 lines.
      2. There are 30 variables.
      3. There are 7 methods.
   2. Miscellaneous: This program took me approximately 30 hours for coding and fixing errors. Moreover, I had learned about the standard input stream buffer and the way to scan a character right after reading a number.
2. Description.
   1. The program problems:
      1. Read a character from keyboard buffer right after scanning a number.
      2. Generate random numbers.
      3. Find modulo of large numbers occupying a word.
      4. Translate the game rules from the C++ code into the Assembly language.
   2. The program solutions:
      1. Use the character scan pattern as “ %c” (the space before %c solves the new line input problem).
      2. Take advantage of C functions (i.e. time, rand, and srand).
      3. Shift the numerator (a large number) 1 bit to the right. (this is just a temporary solution)
      4. Divide C++ code into small, logical pieces (i.e. constructs), then translate those pieces into Assembly.
   3. Sample Input/Output:
      1. The Paper-Rock-Scissors game menu: (The player choose an option by its index)



* + 1. Human Player vs Computer Bot:
    2. The human player chooses Rock and has a tie game:
    3. The player replays the game:



* + 1. Computer Bot vs Computer Bot:
    2. Another game between Human Player and Computer Bot:
  1. Flowcharts:
  2. Concepts:

|  |  |
| --- | --- |
| Concept | Location |
| 1. External C functions | * All files (exclude getMod.s) |
| 1. Internal functions | * main.s * processHumanVsComputer.s * processComputerBots. * getRandomPRS.s * findTheWinner.s |
| 1. Constructors:  * If * If-else * Switch * While * Do-while | * main.s * processHumanVsComputer.s * displayPRS.s * getMod.s * isTheFormerWin.s * findTheWinner.s |
| 1. Stack | * processComputerBots |

1. Program.
   1. [main.s](PRSAssembly/main.s)
   2. [displayPRS.s](PRSAssembly/displayPRS.s)
   3. [processHumanVsComputer.s](PRSAssembly/processHumanVsComputer.s)
   4. [processComputerBots](PRSAssembly/processComputerBots.s).
   5. [getRandomPRS.s](PRSAssembly/getRandomPRS.s)
   6. [getMod.s](PRSAssembly/getMod.s)
   7. [isTheFormerWin.s](PRSAssembly/isTheFormerWin.s)
   8. [findTheWinner.s](PRSAssembly/findTheWinner.s)