EE306 Introduction to Computing

Programming Assignment # 4

Prof. Nina K. Telang

All programming assignments are individual. You are allowed to seek help only from TAs and the instructor.

Purpose: To write the two-person game of Connect-4.

Connect-4 is a two-person game. The game involves the two players alternating turns. It consists of a grid with columns and rows. At his/her turn, a player drops a checker into the column of his/her choice. The game play continues until one of the players gets 4 checkers in a line. The line can be horizontal, vertical, or diagonal.

This assignment requires you to implement a Connect-4 game in LC-3. The game will interact with the two players via the console and the keyboard. You are given a starter code that will serve as a template. **DO NOT MODIFY THE SUBROUTINES AND VALUES THAT WE HAVE GIVEN YOU IN THE STARTER CODE.**

These instructions are organized as follows:

Rules: This section lists the rules of the game for the 6x6 grid which serves as our game board.

General Game description: In this section the mechanics of the game as implemented in this assignment are described.

Sample Input/Output: An example of an input/output for a game can be found here.

Starter Code: In this section the starter code provided is described, along with all subroutines that we have completed. The sections/subroutines that you need to complete for this assignment are also described here.

The file you will submit must be named **connect4.asm**.

1. Rules

In our variation of Connect 4, the game board consists of a 6 by 6 grid.

The rules are as follows:

- Each player takes turns dropping a checker into a column.
- A player cannot remove checkers from a column.

- A player cannot insert a checker into a column that is full (that is, a column where every row is occupied by a checker).
- The game ends when a player places four of one's own checkers next to each other vertically, horizontally, or diagonally.
- A tie occurs if every location on the grid is occupied but no player has four consecutive checkers.

2. General Game Description

At the beginning of the game the initial state of the game board must be displayed. The ASCII character "-" (ASCII code x002D) is used to represent an empty location in the grid, "O" (ASCII code x004F) to represent a checker of Player 1, and "X" (ASCII code x0058) to represent a checker of Player 2. To make the grid easier to read, a space must be inserted BETWEEN each column. Do not output a space after the 6th column. The initial state of the game board should look as follows:

```
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- - - - - -
```

Player 1 always goes first, and play alternates between Player 1 and Player 2. At the beginning of each turn you should output which player's turn it is, and prompt the player for her move. For Player 1 this should look as follows:

```
Player 1, choose a column:
```

To specify which column to drop a checker into, the player should input a single digit from 1 to 6. The number (from 1 to 6) specifies which column is selected, from left to right. For example, if the player wants to drop a checker into the 2nd column from the left, they would type 2 and then press enter. The program must make sure the player's input referred to a valid column. If the player's move is invalid, you should output an error message and prompt the same player for a move. For example, if it is Player 1's turn:

```
Player 1, choose a column: D
Invalid move. Try again.
Player 1, choose a column: 7
Invalid move. Try again.
Player 1, choose a column:
```

Note that you should output the same invalid move message if the player tries to play in a column which is already full.

Your program should keep prompting the player until a valid move is chosen. Be sure your program echoes the player's move to the screen as they type it. After you have echoed the player's move, you should output a newline character (ASCII code x000A) to move the cursor to the next line.

After a player has chosen a valid move, you should update the state of the game board to reflect the move, re-display the updated game board, and check for a winner. If there is no winner, you should continue with the next player's turn.

When a player has won or a tie has occurred, the game is over. At this point, your program should show the final board and then halt. For example, if Player 2 has four adjacent checkers, your program should output the following:

Player 2 Wins.

If a tie occurs, your program should output the following:

Tie Game.

3. Sample Input/Output

An example of the input/output for a game being played can be found below. To receive full credit for your program, your input/output format **MUST EXACTLY MATCH** the format in the example.

```
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- - - - - -
Player 1, choose a column: 1
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- - - - - -
O - - - - -
Player 2, choose a column: 2
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- - - - - -
O X - - - -
Player 1, choose a column: 2
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- O - - - -
O X - - - -
Player 2, choose a column: 3
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- O - - - -
O X X - - -
Player 1, choose a column: 3
- - - - - -
- - - - - -
- - - - - -
- - - - - -
- O O - - -
O X X - - -
Player 2, choose a column: 1
- - - - - -
- - - - - -
- - - - - -
- - - - - -
X O O - - -
O X X - - -
Player 1, choose a column: 4
- - - - - -
- - - - - -
- - - - - -
- - - - - -
X O O - - -
O X X O - -
Player 2, choose a column: 4
- - - - - -
- - - - - -
- - - - - -
- - - - - -
X O O X - -
O X X O - -
Player 1, choose a column: 3
- - - - - -
- - - - - -
- - - - - -
- - O - - -
X O O X - -
O X X O - -
Player 2, choose a column: 4
- - - - - -
- - - - - -
- - - - - -
- - O X - -
X O O X - -
O X X O - -
Player 1, choose a column: 4
- - - - - -
- - - - - -
- - - O - -
- - O X - -
X O O X - -
O X X O - -
Player 1 Wins.
----- Halting the processor -----
```

4. Starter Code:

The following is a description of the subroutines that the program needs. Some of these are provided for you in the starter code. Others you will need to complete.

- **DISPLAY_BOARD: You need to write this subroutine.**
  This subroutine displays the current status of the board. The initial state of the board, and an example of the state of board while being played are shown in previous sections.

- DISPLAY_TURN: This is provided to you.
  This subroutine displays the appropriate prompt: either "Player 1 choose a column: "or "Player 2 choose a column: ".

- GET_MOVE: This is provided to you.
  This subroutine gets a column from the user and also checks whether the move is valid or not by calling the CHECK_VALID subroutine.

- **CHECK_VALID: You need to write this subroutine.**
  This routine checks to see if the move is valid or not.
  Input: R0 has the ASCII value of the move.
  Output: R6 has a 0 if the move is invalid, and the decimal column value if valid.
  The description of valid and invalid moves is given in a previous section.

- UPDATE_BOARD: This is provided to you.
  This subroutine updates the game board with the last move.

- CHANGE_TURN: This is provided to you.
  This subroutine changes the turn by updating the turn from player 1 to player 2.

- CHECK_WINNER: This is provided to you.
  This subroutine checks if the last move resulted in a win. This subroutine calls other routines: CHECK_H, CHECK_V, CHECK_D which you will need to write.

- **CHECK_HORIZONTAL, CHECK_VERTICAL, CHECK_DIAGONAL, CHECK_D1, CHECK_D2. You need to write these subroutines.**
  These routines do the horizontal check, vertical check, and diagonal checks respectively. For each of these routines:
  Inputs: R6 is the column of the last move, R5 is the row of the last move.
  Outputs: R4 has 0 if not winning move, or 1 if winning move.

- UPDATE_STATE: This is provided to you.
  This subroutine updates the state of the game by checking the board to see if there is a win or not. If there is no win, then it updates the turn. If there is a win, it updates the winner.

- **GAME_OVER: You need to write this subroutine.**
  This routine checks the winner and outputs the proper message.