CS/ ECE 252 Introduction to Computer Engineering

Homework 8 – Due at Lecture on Monday, May 4th

Instructions: You should do this homework in a group of **TWO** or **THREE** students from the SAME 252 section. You should hand in **ONE** copy of the homework. Front page of the answer sheet should contain

- **Name** and **UW ID** of the students in that group
- **Section number** (Lec 001 or Lec 002)
- Multiple pages should be **stapled**.

Warning: Most home works will use questions from your textbook, Patt and Patel's *Introduction to Computing Systems*, which we abbreviate *(ItCS)*

First contact for questions is TA Maheswaran Venkatachalam (kvmakes@cs.wisc.edu)
Solution of Problem 5 should be mailed to kvmakes@gmail.com with the subject CS252-HW_8

Problem 1

Output the following string on to the monitor with quotes.

"This is the first question of the last homework!"

Problem 2

Why is handshaking necessary in the case of asynchronous I/O?

Problem 3

Which is more efficient, interrupt-driven I/O or polling? Explain.

Problem 4

Give the differences between memory mapped I/O and Special I/O instructions

Problem 5

Here, we are going to design the "Connect Four" game. For this, we use a simulator called PennSim. The resources for PennSim are available in the course web page.

http://pages.cs.wisc.edu/~david/courses/cs252/Spring2009/includes/computing1.html

Make sure that you download the lc3os for PennSim only from here. Please don't download it from anywhere else.

http://pages.cs.wisc.edu/~david/courses/cs252/Spring2009/includes/handouts/lc3os.asm

Connect Four (also known as Plot Four, Find Four, Four in a Row, and Four in a Line) is a two-player game in which the players take turns in dropping alternating colored blocks into a vertically-suspended grid. The object of the game is to connect four singly-colored blocks in a row—vertically, horizontally, or diagonally—before your opponent can do likewise.

Here is the URL to play the game online

http://www.mathsisfun.com/games/connect4.html

You will be writing this game in assembly language, but using the keyboard to control which column the block will fall rather than the mouse. Be sure to try the online game so you have some idea of how it works.

You are provided with a assembly code that draws the board and fills the next appropriate block in the column in which the player wishes to drop the block. Your job is to write the modules mentioned below.

Subroutine Next_move

Input: R0 = current slider position (value is in [0-5])
R1 = Current player's number ((value is in {0, 1})
Output: R0 = new slider position (value is in [0-5])
Purpose: This routine is called by the main routine to determine the next player's move. The color of the slider determines whose turn it is, so call Print_Slider to print the slider with the current player's color. Call **Get_input** to read the player's keyboard input to update the slider position or drop a ball. If the player attempts to move the slider outside the legal range (i.e. move left when the slider is in position 0 or move right when the slider is in position 5), then ignore the move. If the slider position changes, call **Print_Slider** to display the new slider position. Repeat until Get_input indicates that the player has dropped the ball.

Subroutine Get_input

Input: Keyboard input only
Output: R0 = change in slider position or move (value is in {-1, 0, 1})
Purpose: Read one valid input character from the keyboard. Players indicate they want to move the slider left one position by typing the 'h' character, move the slider one position right by typing the 'j' character, or drop the ball by entering '\n' (the return key). All other inputs are not valid inputs and are ignored (i.e., Get_input discards them and does NOT return). The return values indicate: -1 ==> move slider left, 0 ==> drop ball, +1 ==> move slider right.

Subroutine Print_slider

Input: R0 = current slider position (value is in [0-5])
R1 = current player (value is in {0,1})

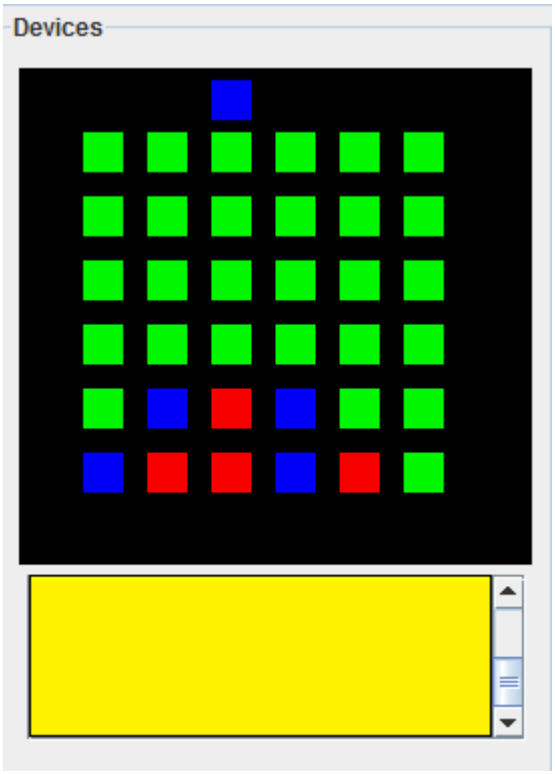
Ouptut: display output

Purpose: print the slider to indicate which column the ball will be dropped into. You have to store the colors for drawing the slider for each player in P0_COLOR (for player 0) and P1_COLOR (for player 1). If the input that you get in register R1 says it is player 0's turn, you should load the color from P0_COLOR. Similarly, if the input that you get says it is player 1's turn, you should load the color from P1_COLOR. You can get the color values from PennSim manual in the course web page. The location of the slider is determined by the current slider position. The slider should be 10 pixels high by 10 pixels wide. The top left pixel for each position is xC190, xC1A0, xC1B0, xC1C0, xC1D0, and xC1E0.
See the PennSim manual for details on how to use the video display.

You are REQUIRED to follow the following parameter passing convention between your subroutines. Return address (R7) and return value (R0 and maybe R1) are caller saved. All other registers should be callee saved.

The output window shows the status of the game. It tells whether the game ended in a **It's a Tie** or **Player 0 wins!!** or **Player 1 wins!!** or **Not there, it's Full!**

This is how the grid and output window look like.

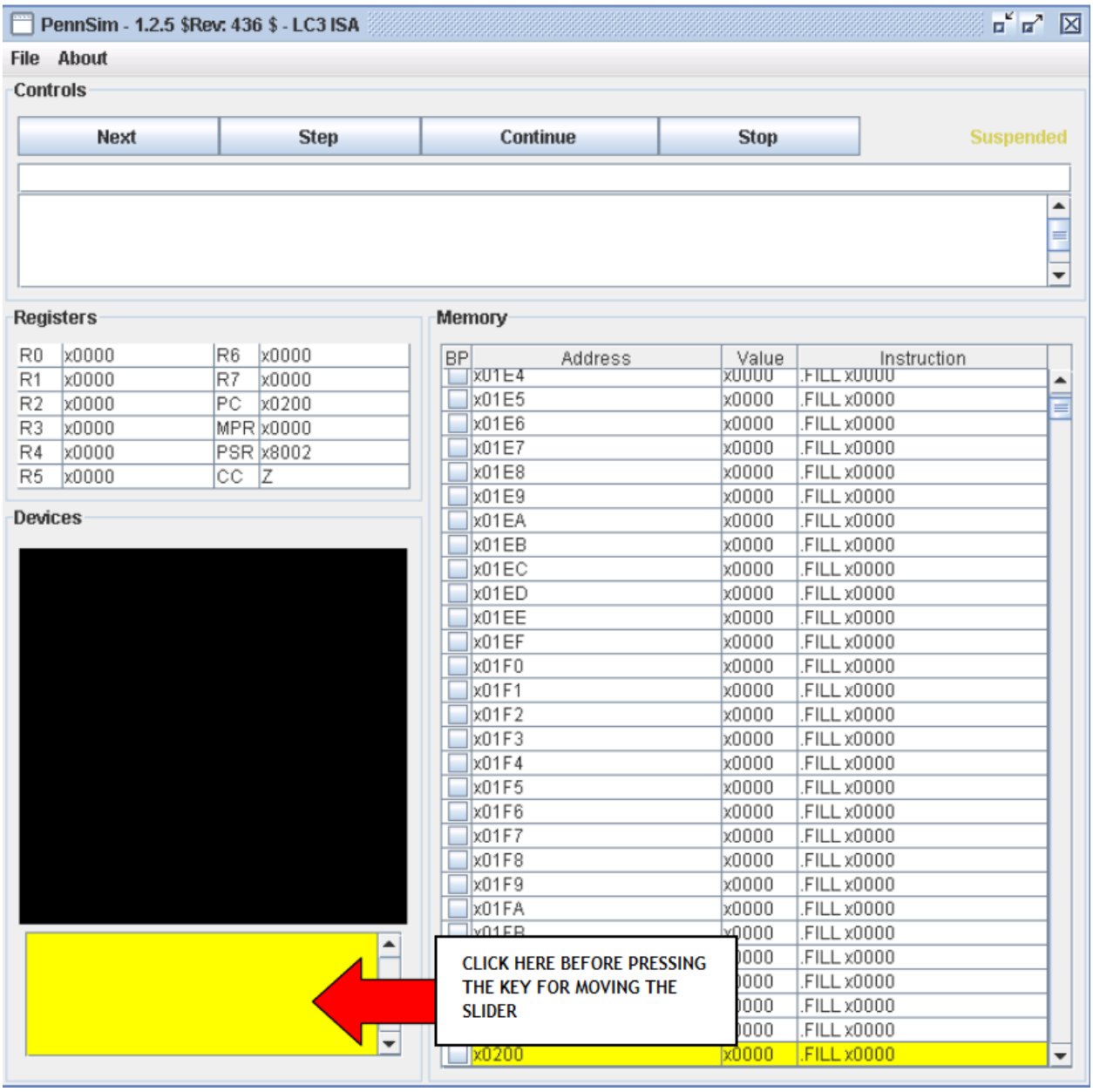


It's a tie: Tie occurs when the grid is completely filled with blocks (i.e. there are 36 blocks in the grid) and it doesn't contain 4 blocks of the same color connected in a row—vertically, horizontally, or diagonally.

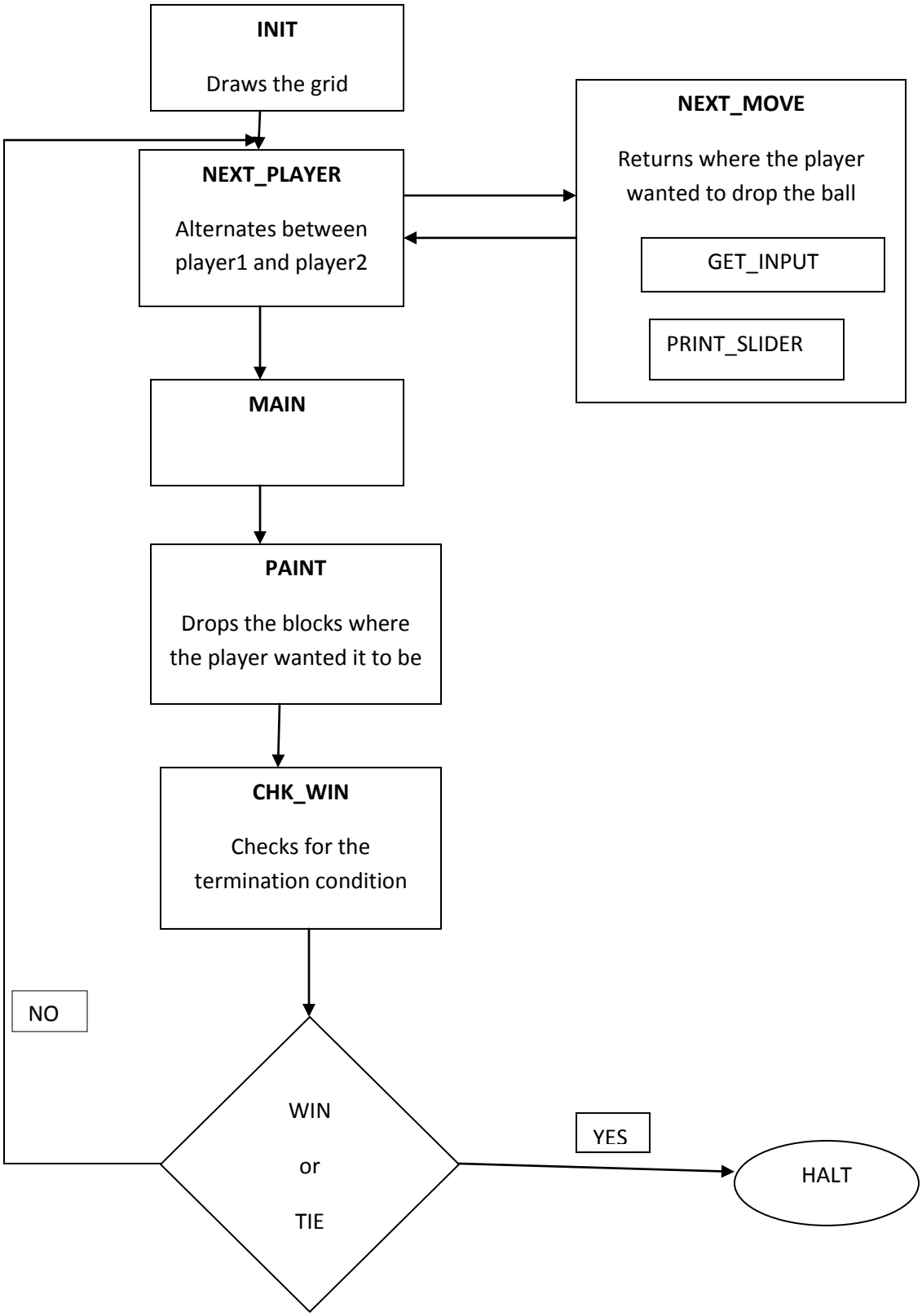
Player 1 or Player 0 Wins: When the grid contains 4 blocks of the same color connected in a row—vertically, horizontally, or diagonally.

Not there, it's full: This gets displayed when you try to place the block in a column which already contains 6 blocks.

Important Note: *In order for the Keyboard input to work, make sure that you click in the box/window right below the drawing area after to press 'continue'. Refer to the figure below.*



Find below the flow of the code


INIT
Draws the grid
NEXT_MOVE
Returns where the player wanted to drop the ball
GET_INPUT
PRINT_SLIDER
NEXT_PLAYER
Alternates between player1 and player2
MAIN
PAINT
Drops the blocks where the player wanted it to be
CHK_WIN
Checks for the termination condition
NO
WIN
or
TIE
YES
HALT