# ECE/CS 252 Fall 2008 Homework 8 // Due on Dec 5, 2008 in lecture

**Problem 1:** What is the difference between:

(a) Memory mapped I/O and I/O mapped I/O
(b) Polling and interrupts
(c) Synchronous and Asynchronous interrupts
(d) Traps and Interrupts

**Problem 2:** Why is it important to check the ready bit of the KBSR before reading the KBDR?

**Problem 3:** Problem 9.17 on page 246 of ItCS.

**Problem 4: Breakout:**

By the end of homework 7, you should have the assembly code that creates the top border, left and right border, bottom border (the ideal paddle), the bricks and the ball that moves around and bounces off the walls and the bricks.

In this part, we will replace the bottom border (the perfect paddle) with the actual paddle that moves left or right. Here is what you need to do to kick things of in the right direction.

- Get the TA's starter implementation of breakout from the course website.
- Go through the TA's implementation of breakout and try to understand the code by reading the comments and identify locations where you should add your code.
- Get the latest LC3 OS from the course website and identify locations where you should add your code.

## Drawing the Paddle:

The first thing that you need to do is to replace the bottom border with the actual paddle. Some critical details about the paddle are as follows.

- The paddle should be of the same size as the bricks i.e. five 4x4 blocks.
- The color of the paddle must be RED so that the ball reacts to the paddle in the same way as it reacts to the wall.
- When the game starts, the paddle must be in the middle of the bottom row i.e. equidistant from the left and the right wall.
- You should use TRAP x40 to draw the paddle.
- Find the subroutine DRAW_PADDLE_SR and add the code for drawing the paddle to it.

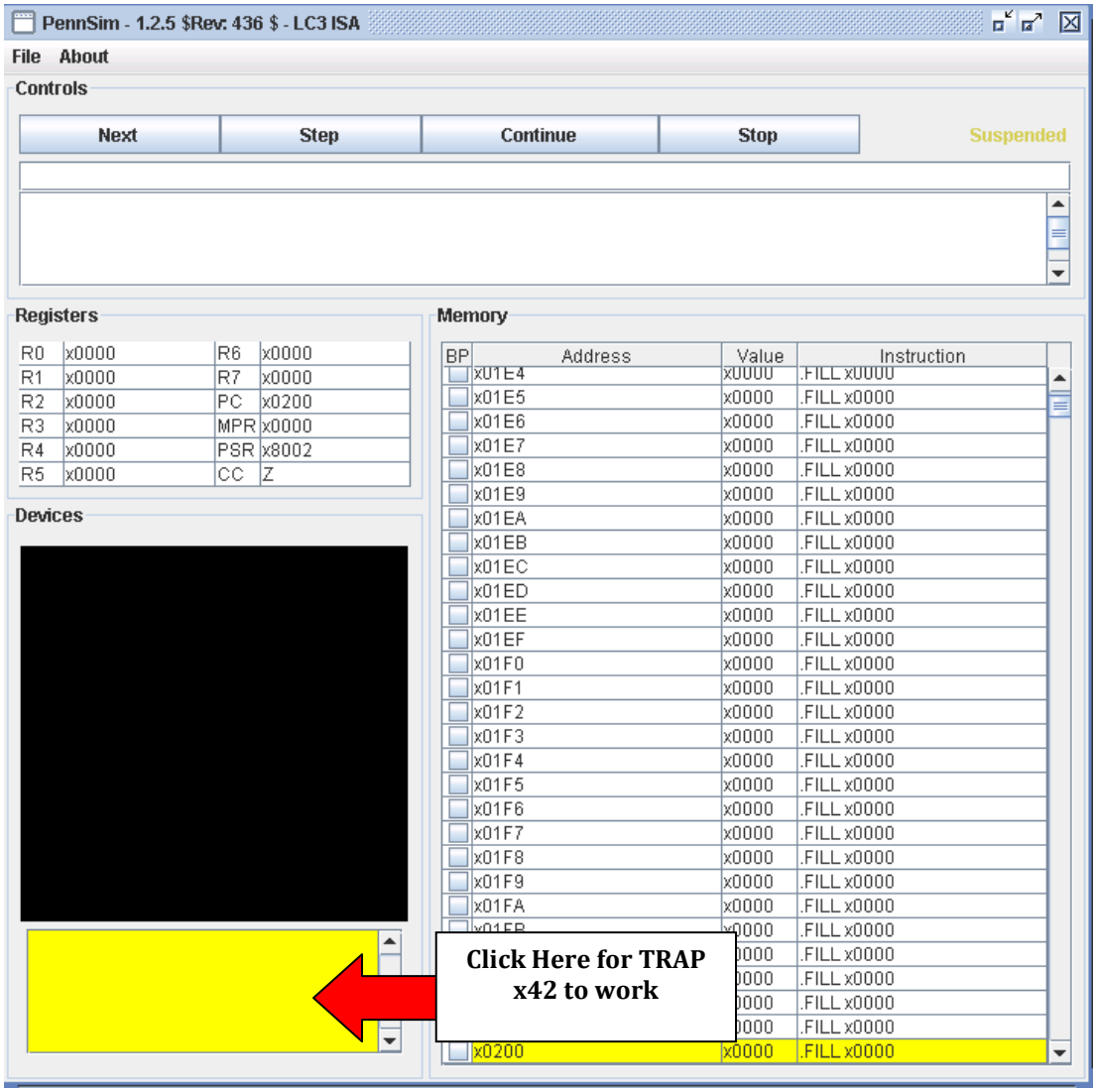
- Since the paddle is of the same color as the wall, you DO NOT need to add any extra code to make the ball bounce off the paddle (Treat the paddle as the wall).

## Moving the Paddle:

The next step is to move the paddle based on the user input. For this purpose you will implement **TRAP x42**.
To implement **TRAP x42**,

- Open the lc3os_hw8.asm file.
- Search for the GET_EVENT subroutine.
- Implement the code for getting user event.
- Follow the comments provided in the GET_EVENT function to implement the code.
- The key pressed by the user is stored in register R5.
- If the user does not press any key, R5 should contain '-1'.

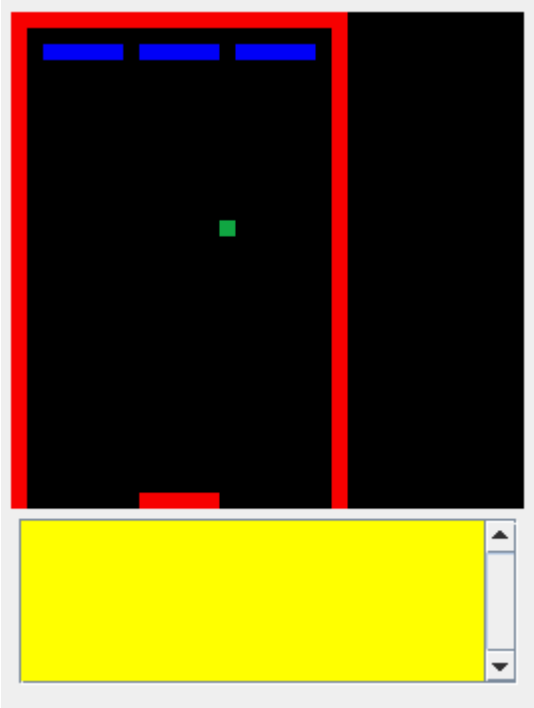
**Important Note:** *In order for the TRAP x42 to work, make sure that you click in the box/window right below the drawing area after to press 'continue'. Refer to the figure below.*



In order to implement the functionality to move the slider, you need to add code to PADDLE_NEXT_LOC_SR subroutine.

- Get the key pressed by the user by calling **TRAP x42.**
- If the user presses 'a' the slider should move left 4 pixels i.e. one 4x4 block.
- If the user presses 'd' the slider should move right 4 pixels i.e. one 4x4 block.
- If the user presses 'q' the game should HALT.
- If the user presses any other key the slider should not move.

The figure below shows how your game should look like.



**Console Input and Display:**

Once all the bricks are destroyed, you should display "**You Win!!!**" on the console. For this purpose you will use **TRAP x22**. **TRAP x22** is already implemented for you in lc3os.asm.
**TRAP x22** writes a null-terminated string of characters to the console starting from the address stored in R0.
In order to display a message, you need to store it in the memory first. You can use the following format.
**YOU_WIN: .STRINGZ "You Win!!!"**

In your program, you will store the address YOU_WIN in R0 and then call TRAP x22 to display the message on the console.
If the ball misses the paddle and falls on the floor, you should display "**You Lose!!!**".

The game ends when:

- All the bricks are destroyed.
- Ball misses the paddle and falls to the floor.

Once the game ends, you should display "**Play Again?**" and get the user input using **TRAP x20**. **TRAP x20** is also implemented in lc3os.asm.
**TRAP x20** reads a single character of input from keyboard device into R0.

- If the user presses 'y' then restart the game.
- If the user presses 'n' then HALT.

**Turn In:**
You need to turn in a printout of your completed source code and a screenshot of your Breakout board which should look similar to the picture above. You will also place your .asm files for **both the game and the OS** in the HW8 dropbox at learn@UW by **5:00pm on the due date**.

**Demos:**
We will be having group demos on Saturday, 6th December. Following are the details about the group demos.

- Demos will be graded.
- Sign up for the group demos. (Details about signing up will be sent later)
- Each of the group members must be present at the demo.
- You will be asked questions and you will have to explain the functionality of your code (This is why commenting your code is important!!!).
- Each of the group members must know what different pieces of code are doing.