CS2053 - Computer Architecture

Department of Computer Science and Engineering - University of Moratuwa

Lab 8: Input & Output

Learning Outcomes:

After completing this lab, you will be able to:

- Have a clear understanding of the role of Input/Output operations in computer systems and how they enable communication between the CPU and external devices.
- Understand the characteristics of different approaches to handle IO, such as Programmed IO, Interrupt driven IO and DMA.
- Demonstrate the ability to interface with various input and output devices in a RISC-V based system.

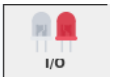
Introduction

Input & Output

Input and Output, often abbreviated as I/O, are the lifelines of computer systems, facilitating the flow of data between the digital realm and the physical world. While Input brings information into the system, Output allows the system to communicate its results. Together, they enable computers to interact with users, devices, and the environment, making them indispensable tools in modern life. Understanding the principles of I/O is fundamental to comprehending how computers bridge the gap between the virtual and tangible domains.

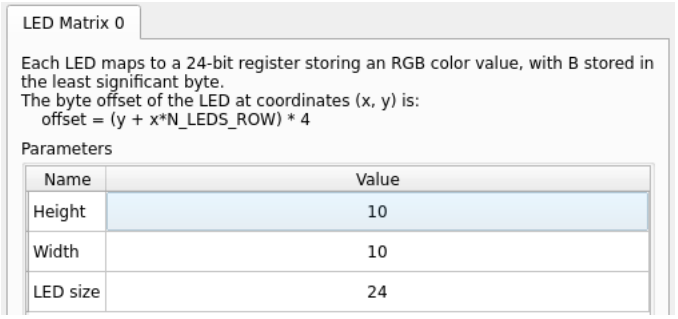
Example :

First load the LEDs.c example code. (File -> Load Example -> C -> leds.c).
Then create a LED matrix panel in the Ripes simulator.

- Open the I/O Devices by clicking on the 

icon on the left sidebar.
- Double click on the ‘LED Matrix’ option on the ‘Device’ section.

Adjust the grid size of the LED matrix with the following values.



LED_MATRIX_0_BASE (0xf0000000) is the base memory address where the output devices (LEDs in the matrix) are started to be mapped into the memory.

Change the processor architecture to ‘Single cycle processor’ to reduce instruction cycles, so that you can observe the changes in the LED panel quickly.
In addition, you can decrease Max. cache plot cycles from the settings menu, to increase the execution speed. (You can go as low as the value 1).

Compile and run the code. Observe the changes in the LED panel as each line gets executed. You can use fast execution if you just want to see the results.

Following code line will write the colour of the LED to the memory. Try to understand what is happening in this line of code underneath.

```
*(led_base + idx) = r << 16 | g << 8 | b;
```

Can you explain the behaviour of the LEDs with the memory mapped I/O theories you learnt in lectures?

Refer to this link to get more information about I/O in Ripes documentation.
http://ripes.me/Ripes/docs/mmio.html

Exercise 1 :

In Ripes you can simulate I/O read write operations with basic components such as LED matrices, Switches and Dial Pads. However IO techniques such as Interrupt driven I/O and DMA are not available in Ripes.

Therefore IO read will be done via programmed I/O(polling)

In this exercise we will simulate the working of a traffic light using Ripes. For this task create a LED matrix of height 3 and width 1, and a single switch in the I/O tab.

The assumed working principle: There is a switch to turn on and off the traffic light. When the switch is off all LEDs are black. When the switch is on, a separate counter is running and the traffic lights should indicate Stop, Get Ready and Go states repeatedly.

Then use the following code and complete the parts marked in bold.

Run the code and see how Memory-mapped IO works here.

Colour hints:
Green: 0xFF << 8
Orange: 0xFF << 8 | 0xFF << 16
Red: 0xFF << 16
Black: 0x0

```
#include "ripes_system.h"

unsigned* led_base = LED_MATRIX_0_BASE;
unsigned* switch_base = SWITCHES_0_BASE;

void main() {

   unsigned state = 0;
   unsigned count = 1;

   while (1) {

      if ([Check if switch 0 is toggled]) {

         if (count % 10 == 0){    //State Change
            if (state == 2){
               state = 0;
            }
            else{
               state++;
            }
         }

         if (state == 0){       //Go State
            [Complete Code]                  //Lower Led should be Green and others black

         }
         else if (state == 1){  //Get Ready State
            [Complete Code]                  //Middle Led should be Orange and others black
         }
         else{                  //Stop State
            [Complete Code]                  //Upper Led should be Red and others black
         }

         count++;         //Increment Count

      } else {              //Switch 0 is not toggled
         [Complete Code]                  //All black
      }
   }
}
```

Exercise 2 :

In the code below we will be simulating the movement of an LED through a LED matrix using the D-Pad. Copy and try the simulation accordingly.

```
#include "ripes_system.h"
#include <stdio.h>

#define W LED_MATRIX_0_WIDTH
#define H LED_MATRIX_0_HEIGHT
unsigned* led_current = LED_MATRIX_0_BASE;
unsigned* d_pad_right = D_PAD_0_RIGHT;
unsigned* d_pad_down = D_PAD_0_DOWN;
unsigned led_start = LED_MATRIX_0_BASE;

void main() {
    *(led_current) = 0xFF << 16;
   while (1) {
      if (*d_pad_right & 0x1==1){
         *(led_current)=0;
         *(led_current+1) = 0xFF << 16;
         led_current++;
      }else if(*d_pad_down & 0x1){
         *(led_current)=0;
         *(led_current+W) = 0xFF << 16;
         led_current = led_current+W;
      }
      // Implement the code for other 2 buttons of the D-Pad
   }
}
```

Observe how the LED moves with the button presses. Can you explain why there is a delay between the button presses and LED movement?
Try to implement the code for the other 2 buttons as well. Note that the boundary conditions are not handled in this code. So make sure to navigate the LED only within the matrix.

Additional Exercise:
So far we have dealt with programmed I/O only. Interrupt driven I/O is another technique of handling inputs and outputs. They are covered in the lecture as well. Refer to the lecture slides, as well as the following learning material to understand more about interrupt driven I/O.

https://courses.cs.washington.edu/courses/cse378/11wi/lectures/lec23.pdf

Will you be able to write an example code to implement interrupt driven I/O for any of the processors included in the Ripes simulator? Briefly describe.