

Introduction to Lab Maze Game

José Nelson Amaral

Maze Games



Two Screens

- Preparation Screen:
 - Reads the level of the game
- Game Screen:
 - Displays the maze
 - Player can move around by pressing using WASD.
 - Time remaining

Keyboard and Display MMIO Simulator

DISPLAY: Store to Transmitter Data 0xffff000c, cursor (60,0), area 95 x 30

Please enter 1,2 or 3 to choose the level and start the game

Font

☒ DAD

Fixed transmitter delay, select using slider

Delay length: 5 instruction executions

KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004

I

Tool Control

Disconnect from Program

Reset

Help

Close

Exceptions/Interruptions

- Enable interrupts for both the timer and the keyboard
- Create an exception handler

Enable Interrupts

- Keyboard:
 - Keyboard Control (0xFFFF0000): Bit 1 of must be 1 for the keyboard to be enabled
 - Must be re-enabled after every keyboard interrupt
 - Keyboard Data (x0xFFFF0004): Contains the ASCII character after a key is pressed
- Timer:
 - Timer (0x0xFFFF0018): Contains the current time
 - TimeCMP (0xFFFF0020): User-specified value. When matched by the timer an interrupt is generated
- Interruption Control:
 - ustatus register (CSR#0): bit 0 must be set to 1 for user interrupts to be allowed.
 - uie register (CSR#4): Bits 4 and 8 must be 1 in order to enable keyboard and timer interrupts.

Saving Registers

An interrupt handler must save all the registers that it uses.

- The label `iTrapData` designates a section of memory allocated for saving registers in the handler.
- Outside of the handler, `uscratch` (CSR #64) should contain the address of the `iTrapData` section.
- Use the `cssrw` instruction to swap a register with the `uscratch` and save all the required registers.

Exception Handler

The maze.s already contains the Handler Terminate section

The common.s file will already have the iTrapData section

Wall Structs

```
struct Wall {  
    int start_x  
    int start_y  
    int end_x  
    int end_y  
}
```

- Encodes the information regarding the walls in the maze
- Can be thought of as a C struct like the one given on the left.
- You will be given an array of these structs one for each wall in the maze.
- Each struct will be laid out in the memory as:

start_x	start_y	end_x	end_y
---------	---------	-------	-------

Example the following is an array with 2 walls:

15	4	25	4	1	6	1	14
----	---	----	---	---	---	---	----

Check the lab specification page for explanatory videos.

Game Loop

