Name Leonardo Roman

Lab 5 Exceptions and Interrupts

Computer Architecture

Purpose

Know the exception mechanism in mips assembly language and to be able to write interrupt handler for mips machine.

Assignment 1

Using the file "io.s" analyse the changes in the status and cause register while the program uses interrupts and polling. Initially cause and status registers have a value of 0 and 0x3000ff10 respectively. After running the program cause and status changes to 400 and 0x30000010 (at this instance the program is at polling state). Next change in registers is 400 and 0x3000ff11 and finally, once program is over, cause and status values are 0 and 0x3000ff11 respectively.

Analysis

While running the program, after entering label L1 line 48 in the file, the program gets into a polling state. The program polls until all data is entered and the flag signals the cpu to stop polling. The flag in this case in the counter $s0 which decrements as user enters the data. Once the counter gets decremented from 3 to 0, the program stops polling but then enters into interrupt state. Once in interrupt state, the counter resets to 3 again and serves the same purpose.

Observation

At the polling state, the program awaits for all data to be entered, this can be seen as the program loops L1 indefinitely while instruction "beq $t4,0,l1"(line 48) is true. While polling and the data is being entered the following happens:

User input Plus enter key	Cause Register	Status Register	EPC	Counter $s0	ASCII $t6 (Hex)
No input	0	3000ff10	0	3	0
“This”	400	30000010	0	2	54,68,69,73,a
“is”	400	30000010	0	1	69,73,a
“Computer”	400	3000ff11	4000bc	3	43,6f,6d,70,75,65,72,a
“Architecture”	400	3000ff11	4000bc	2	41,72,63,68,69,74,65,63,74,75,72,65,a
“Lab”	400	3000ff11	4000bc	1	4c,61,62,a
“5”	0	3000ff11	4000bc	0	35,a

Difference between polling and interrupts

- **Polling:** polling waits for a task to be done in order to receive any data. We can see this when the program is waiting for the counter $s0 to be decremented to zero in order to stop looping label “l1”. Polling constantly checks if the current task is done and once is done it gets all the data transfer by that particular task.
- **Interrupt:** interrupts are events cause from outside the processor (input). The program performs all instructions until it gets a “done” signal or an exception. We can see this in the io.s program when the counter $s0 gives the “done” signal as it reaches count zero.

How interrupts are enable

Interrupts are enabled whenever there is an exception that comes from outside the processor. The cause register will provide for the information about the level of pending interrupts and bit IPi becomes 1 if the interrupt has occurred at level i.

During polling interrupts are disabled. Polling simply checks/waits for all data to be collected before it gets transferred to the processor. Polling happens even if there has been no change in the connected device.

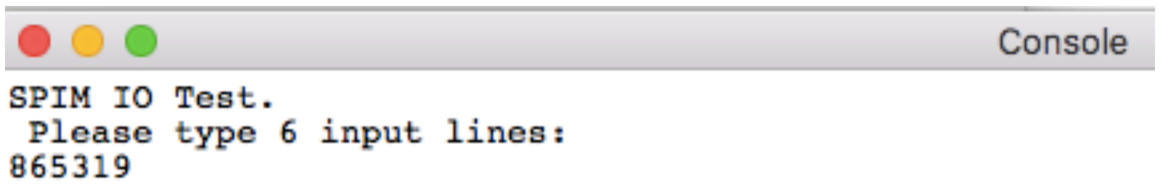
During polling the processor is in total control. If interrupts were enabled during polling, it might cause a loss of data. Remember that polling is simply a process that awaits for a task to be done and as soon the task is done the data gets transferred to its destination. If this polling gets interrupted then a particular task might not get completed properly and not all data gets transferred as desired.

Assignment 2

In this assignment io.s file is modified to print a random number from 1 to 9. In this modification, I got rid off the polling state and modified character write in line 110 in order to print a random number when an interrupt occurs(keystroke). A label RandomNumber which body calculates the random number is implemented. The calculation uses ascii value 48 (int number 0) and is incremented by one.Then using a branch condition the program check if 48 reaches 58 and if true then it loops back to RandomNumber. This RandomNumber loop gets interrupted by a single keystroke which will be the signal to print a random number from 0-9. Program stops after enter key gets pressed and it also would print a random number. The following picture demonstrates a simple set of letters entered including enter

Keystroke = {l,b,f,g,t,enter}

Output:



```
SPIM IO Test.
 Please type 6 input lines:
865319
```

.