Input/Output (I/O, IO Notes)

- What is I/O?
- Waiting Loop
- Polling
- Exceptions, or Interrupts
  - Hardware interrupt
  - Software interrupt
  - Application: Hurricane detector
  - Application: Time sharing
Suppose $t0 = ff...ff0

lw $t1,0($t0) lw $t1,4($t0)
Waiting Loop

ff....ff0

Data_Reg

ff....ff4

Status_Reg

RDY bit (bit 0)
= 1 if Data_Reg has new byte
= 0 otherwise

# Loads $t1 with next key stroke
waiting_lp:
load $t1 with mem cell ff...ff4
$t1 = $t1 & 1
if $t1 = 0 then goto waiting_lp
load $t1 with mem cell ff...ff0

Wastes CPU cycles while waiting

Example

ff....ff0

Data_Reg

ff....ff4

Status_Reg

RDY bit (bit 0)
= 1 if Data_Reg has new byte
= 0 otherwise

# Loads $t1 with next key stroke
waiting_lp:
addi $t2,$0,-16 #$t2 = fff…ff0
lw $t1,4($t2)  # $t1 = Status_reg
andi $t1,$t1,1   # Mask = 00...0001
beq   $t1,$0,waiting_lp
lw     $t1,0($t2)  # $t1 = new key stroke
Polling

• What about multiple devices?
• Check each device periodically
  – Round robin polling -- everybody is repeatedly polled in some cyclic order

Exceptions

• Do your usual processing BUT when the keyboard is pressed then go get the keystroke. Then resume the usual processing
Exceptions

Keystroke, causes an interrupt

Get Keystroke

相似 to jal

Interrupt Handler

Exceptions

Review jal

1. $31 = PC
2. jump to subroutine

1. $31 = PC
2. jump to subroutine
Exceptions

1. Finish completing instruction
2. PC is stored (e.g., epc)
3. PC = specific address A

Hardware Exception

External hardware interrupts
- Level triggered interrupt (LTI)
- Edge triggered interrupt (ETI)

Internal hardware interrupts
- Overflow (OV)
- Incorrect memory access (IMA)
- Incorrect math result (IMR)
Exceptions

1. Finish completing instruction
2. PC is stored (e.g., epc)
3. PC = specific address A

Examples:

<table>
<thead>
<tr>
<th>Interrupt type</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTI</td>
<td>4</td>
</tr>
<tr>
<td>ETI</td>
<td>8</td>
</tr>
<tr>
<td>OV</td>
<td>12</td>
</tr>
<tr>
<td>IMA</td>
<td>16</td>
</tr>
<tr>
<td>IMR</td>
<td>20</td>
</tr>
</tbody>
</table>

Exceptions

Memory

<table>
<thead>
<tr>
<th>Addr</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>j Reset_Routine</td>
</tr>
<tr>
<td>4</td>
<td>j Handler_LTI</td>
</tr>
<tr>
<td>8</td>
<td>j Handler_ETI</td>
</tr>
<tr>
<td>12</td>
<td>j Handler_OV</td>
</tr>
<tr>
<td>16</td>
<td>j Handler_IMA</td>
</tr>
<tr>
<td>20</td>
<td>j Handler_IMR</td>
</tr>
</tbody>
</table>
**Hurricane Detector**

- **Processor**
- **External Interrupt**
- **Hurricane Detector**

**Memory**

```
4:
  j Hurricane

Hurricane:
  Save reg. values
  Print “Head for the hills!”
  Restore reg values
rfe
```

**Storage Area**

---

**Time Sharing**

- Processor wants to execute two programs P1 and P2 at the same time

**Memory**

```
Interrupt Handler
Storage
P1
P2
```

**CPU**

- Interrupt every 30 ms

```
Stored_$1
Stored_$2
.
.
Stored_$31
Stored_PC
```
Time Sharing

Initially:
Storage = init val of all regs for P2
Stored_PC = Start_P2
Jump to Start_P1

Interrupt Handler

Software Exception

- Special instructions, when executed then an interrupt occurs
- Graceful way of moving from user status to supervisor status
  - User status: instructions are executed but restrictions
  - Supervisor status: no restrictions
- Application: used in debuggers such as for breakpoints