Dell may have shipped infected motherboards.

Dell is warning customers that its PowerEdge R410 rack-mount server may have shipped with spyware onboard in the embedded system management software (firmware). Good luck getting Norton McAfee to clean that.

http://www.theregister.co.uk/2010/07/21/dell_server_warning/

---

### Lecture 19

**CPU Design: The Single-Cycle II & Control**

**2010-07-22**

**Instructor Paul Pearce**

Dell may have shipped infected motherboards.

Good luck getting Norton McAfee to clean that.

http://www.theregister.co.uk/2010/07/21/dell_server_warning/

---

**Register-Register Timing: One complete cycle**

<table>
<thead>
<tr>
<th>Clk</th>
<th>PC</th>
<th>New Value</th>
<th>Old Value</th>
<th>Instruction Memory Access Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rs</td>
<td>Rr, Rd, etc</td>
<td>Old Value</td>
<td>Delay through Control Logic</td>
</tr>
<tr>
<td></td>
<td>ALU</td>
<td>Old Value</td>
<td>New Value</td>
<td>Register File Access Time</td>
</tr>
<tr>
<td></td>
<td>RegWrite</td>
<td>Old Value</td>
<td>New Value</td>
<td>Register Write Occurs Here</td>
</tr>
<tr>
<td></td>
<td>busA, B</td>
<td>Old Value</td>
<td>New Value</td>
<td>ALU Delay</td>
</tr>
<tr>
<td></td>
<td>busW</td>
<td>Old Value</td>
<td>New Value</td>
<td>RegFile</td>
</tr>
</tbody>
</table>

---

**3c: Logical Operations with Immediate**

• \( R[rt] = R[rs] \text{ op ZeroExt}[imm16] \)

**3c: Logical Operations with Immediate**

• \( R[rt] = R[rs] \text{ op ZeroExt}[imm16] \)

---

**How to Design a Processor: step-by-step**

1. Analyze instruction set architecture (ISA) \( \Rightarrow \) datapath requirements
   - meaning of each instruction is given by the register transfers
   - datapath must include storage element for ISA registers
   - datapath must support each register transfer

2. Select set of datapath components and establish clocking methodology

3. Assemble datapath meeting requirements

4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.

5. Assemble the control logic

Example: lw rt, rs, imm16

3d: Load Operations

- \( R[rt] = \text{Mem}[R[rs] + \text{SignExt}[\text{imm16}]] \)

Ex.: $lw \ rt, rs, \text{imm16}$

3e: Store Operations

- \( \text{Mem}[R[rs] + \text{SignExt}[\text{imm16}]] = R[rt] \)

Ex.: $sw \ rt, rs, \text{imm16}$

3f: The Branch Instruction

- \( \text{beq} \ rs, rt, \text{imm16} \)
  - \( \text{mem}[PC] \) Fetch the instruction from memory
  - \( \text{Zero} = R[rs] - R[rt] \) Calculate branch condition
  - If (Zero) Calculate the next instruction’s address
    - \( PC = PC + 4 + (\text{SignExt}(\text{imm16}) \times 4) \)
    - else
    - \( PC = PC + 4 \)

Datapath for Branch Operations

- \( \text{beq} \ rs, rt, \text{imm16} \)
- Datapath generates condition (zero)
Putting it All Together: A Single Cycle Datapath

An Abstract View of the Critical Path

Critical Path (Load Instruction) =
- Delay clock through PC (FFs) +
- Instruction Memory’s Access Time +
- Register File’s Access Time +
- ALU to Perform a 32-bit Add +
- Data Memory Access Time +
- Stable Time for Register File Write

( Assumes a fast controller)

Review: A Single Cycle Datapath

Recap: Meaning of the Control Signals

- nPC_sel: “+4” 0 ⇒ PC ← PC + 4
  “br” 1 ⇒ PC ← PC + 4 + (SignExt(imm16) , 00 )
- Later in lecture: higher-level connection between mux and branch condition

Administrivia

- HW 7 due Saturday
  - Specification update now online. Be sure to check it out. The changes are designed to make integration with project 2 easier. They are very minor.
- Project 2 online now!
- Reminder: Midterm regrades due Monday.
- Paul will be gone on Monday. Noah will be giving the lecture.
  - Paul will have OH on Monday (barring flight delays)
- Be prepared for a kinetic learning activity

Later in lecture: higher-level connection between

mux and branch condition
Recap: Meaning of the Control Signals

- ExtOp: zero, sign
- ALUsrc: 0 ⇒ reg; 1 ⇒ imm
- ALUctr: ADD, SUB, OR
- MemWr: 1 ⇒ write memory
- Mux: 0 ⇒ alu; 1 ⇒ mem
- RegDst: 0 ⇒ rt; 1 ⇒ rd
- RegWr: 1 ⇒ write register

RTL: The Add Instruction

```
<table>
<thead>
<tr>
<th></th>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**add rd, rs, rt**

- MEM[PC] Fetch the instruction from memory
- PC = PC + 4 Calculate the next instruction's address

Instruction Fetch Unit at the Beginning of Add

- Fetch the instruction from Instruction memory: Instruction = MEM[PC]
  - same for all instructions

Instruction Fetch Unit at the End of Add

- PC = PC + 4
  - This is the same for all instructions except: Branch and Jump

The Single Cycle Datapath during Add

```
R[rd] = R[rs] + R[rt]
```

Single Cycle Datapath during Or Immediate?

- R[rt] = R[rs] OR ZeroExt[Imm16]
The Single Cycle Datapath during Or Immediate?

- \( R[rt] = R[rs] \text{ OR} \text{ ZeroExt}[\text{imm16}] \)

The Single Cycle Datapath during Load?

- \( R[rt] = \text{Data Memory} (R[rs] + \text{SignExt}[\text{imm16}]) \)

The Single Cycle Datapath during Store?

- Data Memory (R[rs] + SignExt[imm16]) = R[rt]

The Single Cycle Datapath during Branch?

- If (R[rs] - R[rt] == 0) then Zero = 1; else Zero = 0
Peer Instruction

1) In the worst case, the biggest delay is the memory access time
2) With only changes to control, our datapath could write (something) to memory and registers in one cycle.

Summary: A Single Cycle Datapath

- We have everything! Now we just need to know how to BUILD CONTROL

° 5 steps to design a processor
  1. Analyze instruction set → datapath requirements
  2. Select set of datapath components & establish clock methodology
  3. Assemble datapath meeting the requirements
  4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  5. Assemble the control logic
   * Formulate Logic Equations
   * Design Circuits

Summary: Single-cycle Processor