Computer Architecture Experiment

Jiang Xiaohong

College of Computer Science & Engineering
Zhejiang University
Topics

- 0. Basic Knowledge
- 1. Warm up
- 2. Simple 5-stage of pipeline CPU Design
- 3. Pipelined CPU with stall
- 4. Pipelined CPU with forwarding
- 5. Pipelined CPU resolving control hazard and support execution 31 MIPS Instructions
Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution
Experiment Purpose

- Understand the principles of Pipelined CPU Bypass Unit
- Master the method of Pipelined Pipeline Forwarding Detection and Pipeline Forwards.
- Master the Condition In Which Pipeline Forwards.
- Master the Condition In Which Bypass Unit doesn’t Work and the Pipeline stalls.
- master methods of program verification of Pipelined CPU with forwarding
Experiment Task

- Design the **Bypass Unit** of Datapath of 5-stages Pipelined CPU
- **Modify** the CPU Controller
  - Conditions in Which Pipeline Forwards.
  - Conditions in Which Pipeline Stalls.
- **Verify the Pipeline CPU** with program and observe the execution of program
Data Hazard Stalls

- Minimizing Data Hazard Stalls by Forwarding: In most cases, the problem can be resolved by forwarding, also called bypassing, short-circuiting.

- Data Hazards Requiring Stalls: In some cases, data hazards can not be handled by bypassing.
Instruction Demo

1. **ADD $1,$2,$3**
   - Clock Cycle 1: IM
   - Clock Cycle 2: IM, ALU
   - Clock Cycle 3: ALU, DM
   - Clock Cycle 4: DM, Regs

2. **SUB $4,$1,$5**
   - Clock Cycle 1: IM
   - Clock Cycle 2: IM, ALU
   - Clock Cycle 3: ALU, DM
   - Clock Cycle 4: DM, Regs

3. **AND $6,$1,$7**
   - Clock Cycle 1: IM
   - Clock Cycle 2: IM, ALU
   - Clock Cycle 3: ALU, DM
   - Clock Cycle 4: DM, Regs

4. **OR $8,$1,$9**
   - Clock Cycle 1: IM
   - Clock Cycle 2: IM, ALU
   - Clock Cycle 3: ALU, DM
   - Clock Cycle 4: DM, Regs

5. **XOR $10,$1,$11**
   - Clock Cycle 1: IM
   - Clock Cycle 2: IM, ALU
   - Clock Cycle 3: ALU, DM
   - Clock Cycle 4: DM, Regs
Data Hazard Causes Stalls

ADD $1,$2,$3

SUB $4,$1,$5

AND $6,$1,$7

NOP

No hazard.

DOUBLE BUMP CAN DO!
Pipeline Forward to Avoid the Data hazard

1. ADD $1,$2,$3
2. SUB $4,$1,$5
3. AND $6,$1,$7
4. OR $8,$1,$9
5. XOR $10,$1,$11
Move the Forwarding path to ID stage
Move the forwarding control logic to ID stage
**Condition in Which Bypass Unit doesn't work**

ALU needs the value of R10 at the beginning of Clock Cycle 4, but at the end of Clock Cycle 4 DM outputs the R10.
Pipeline Stalls

CLOCK CYCLE 1
IM
LW R1,0(R2)

CLOCK CYCLE 2
Regs

CLOCK CYCLE 3
ALU

CLOCK CYCLE 4
DM

CLOCK CYCLE 5
Regs

CLOCK CYCLE 6

1. LW R1,0(R2)
2. SUB R4,R1,R6
3. AND R1,R6,R7
4. OR R8,R1,R9
Pipeline stalls at ID Stage

```
<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>IM</th>
<th>ALU</th>
<th>DM</th>
<th>Regs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LW</td>
<td></td>
<td></td>
<td>LW</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>IM</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>IM</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>IM</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>IM</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>IM</td>
</tr>
</tbody>
</table>
```

- LW R1,0(R2)
- SUB R4,R1,R6
- AND R1,R6,R7
- OR R8,R1,R9
Pipelined CPU Top Module

module top (input wire CCLK, BTN3, BTN2, input wire [3:0]SW, output wire LED, LCDE, LCDRS, LCDRW, output wire [3:0]LCDDAT);

assign pc [31:0] = if_npc[31:0];

if_stage x_if_stage(BTN3, rst, pc, mem_pc, mem_branch, id_wpcir, ...
IF_ins_type, IF_ins_number, ID_ins_type, ID_ins_number);

id_stage x_id_stage(BTN3, rst, if_inst, if_pc4, wb_destR,...,
EX_ins_type, EX_ins_number, id_FWA, id_FWB);

ex_stage x_ex_stage(BTN3, id_imm, id_inA, id_inB, id_wreg, ..
id_FWA, id_FWB, mem_aluR, wb_dest, ... , MEM_ins_number);

mem_stage x_mem_stage(BTN3, ex_destR, ex_inB, ex_aluR, ...
mem_aluR, ... , WB_ins_type, WB_ins_number);

wb_stage x_wb_stage(BTN3, mem_destR, mem_aluR, ...
wb_dest, ..., OUT_ins_type, OUT_ins_number);
Observation Info

■ Input
  – West Button: Step execute
  – South Button: Reset
  – 4 Slide Button: Register Index

■ Output
  – 0-7 Character of First line: Instruction Code
  – 8 of First line: Space
  – 9-10 of First line: Clock Count
  – 11 of First line: Space
  – 12-15 of First line: Register Content
  – Second line: “stage name”/number/type
  – stage name: 1-“f”, 2-“d”, 3-“e”, 4-“m”, 5-“w”
Test code

- lw r1, $21(r0) 8c010015
- add r2,r1,r1 00211020
- sub r3,r1,r2 00221822
- beq r1,r1,2 10210002
- andi r4,r2,0 30440000
- addi r5,r4,1 20850001
- ori r6,r3,0 34660000
- bne r6,r3,2 14c30002
- lw r7,$20(r0) 8c070014
- sw r7,$21(r0) Ac070015
- addi r8,r7,1 20e80001
- j 0 08000000
- andi r9,r8,1 31090001
Thanks!