Spring 2017
EE 3613: Computer Organization
Chapter 4: Pipelining - 2

Avinash Kodi
Department of Electrical Engineering & Computer Science
Ohio University, Athens, Ohio 45701
E-mail: avinashk@eecs.ohio.edu
Website: http://oucsace.cs.ohiou.edu/~avinashk/ee461a.htm
Acknowledgement: Mary J. Irwin, PSU; Srinivasan Ramasubramanian, UofA,

Course Administration

• Homework 4 due Monday March 27

• Homework 5 posted, due on April 5

• Exam – 2 is on April 7 in-class
  ▪ Topics are Sequential Logic (Moore and Mealy machines), Datapath and Control and Pipelining
Review: MIPS Pipeline Data and Control Paths

Control Settings

<table>
<thead>
<tr>
<th></th>
<th>EX Stage</th>
<th>MEM Stage</th>
<th>WB Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RegDst</td>
<td>ALU Op1</td>
<td>ALU Op0</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>lw</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sw</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>beq</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Review: One Way to “Fix” a Data Hazard

- **Inst. Order**: add $1, stall, sub $4, $1, $5, and $6, $7, $1
- **Fix data hazard by waiting** – stall
- **but impacts CPI**

Review: Another Way to “Fix” a Data Hazard

- **Inst. Order**: add $1, sub $4, $1, $5, and $6, $7, $1 or $8, $1, $1, sw $4, 4($1)
- **Fix data hazards by forwarding** results as soon as they are available to where they are needed
Data Forwarding (aka Bypassing)

- Take the result from the earliest point that it exists in any of the pipeline state registers and forward it to the functional units (e.g., the ALU) that needs it in that cycle
- For ALU functional unit: the inputs can come from any pipeline register rather than just from ID/EX by
  - adding multiplexors to the inputs of the ALU
  - connecting the Rd write data in EX/MEM or MEM/WB to either (or both) of the EX’s stage Rs and Rt ALU mux inputs
  - adding the proper control hardware to control the new muxes
- Other functional units may need similar forwarding logic (e.g., the DM)
- With forwarding can achieve a CPI of 1 even in the presence of data dependencies

Data Forwarding Control Conditions

1. EX/MEM hazard:
   if (EX/MEM.RegWrite and (EX/MEM.RegisterRd != 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
     ForwardA = 10
   if (EX/MEM.RegWrite and (EX/MEM.RegisterRd != 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
     ForwardB = 10
   
   Forwards the result from the previous instr. to either input of the ALU

2. MEM/WB hazard:
   if (MEM/WB.RegWrite and (MEM/WB.RegisterRd != 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
     ForwardA = 01
   if (MEM/WB.RegWrite and (MEM/WB.RegisterRd != 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
     ForwardB = 01
   
   Forwards the result from the second previous instr. to either input of the ALU
Forwarding Illustration

Instr. Order

<table>
<thead>
<tr>
<th>Instruction</th>
<th>IM</th>
<th>Reg</th>
<th>DM</th>
<th>Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>add $1,</td>
<td>IM</td>
<td>Reg</td>
<td>DM</td>
<td>Reg</td>
</tr>
<tr>
<td>sub $4,$1,$5</td>
<td>IM</td>
<td>Reg</td>
<td>DM</td>
<td>Reg</td>
</tr>
<tr>
<td>and $6,$7,$1</td>
<td>IM</td>
<td>Reg</td>
<td>DM</td>
<td>Reg</td>
</tr>
</tbody>
</table>

EX/MEM hazard forwarding  MEM/WB hazard forwarding

Yet Another Complication!

- Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction – which should be forwarded?

Instr. Order

<table>
<thead>
<tr>
<th>Instruction</th>
<th>IM</th>
<th>Reg</th>
<th>DM</th>
<th>Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>add $1,$1,$2</td>
<td>IM</td>
<td>Reg</td>
<td>DM</td>
<td>Reg</td>
</tr>
<tr>
<td>add $1,$1,$3</td>
<td>IM</td>
<td>Reg</td>
<td>DM</td>
<td>Reg</td>
</tr>
<tr>
<td>add $1,$1,$4</td>
<td>IM</td>
<td>Reg</td>
<td>DM</td>
<td>Reg</td>
</tr>
</tbody>
</table>
Yet Another Complication!

- Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction – which should be forwarded?

Corrected Data Forwarding Control Conditions

2. MEM/WB hazard:
   if (MEM/WB.RegWrite
   and (MEM/WB.RegisterRd != 0)
   and (EX/MEM.RegisterRd != ID/EX.RegisterRs)
   and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
   ForwardA = 01

   if (MEM/WB.RegWrite
   and (MEM/WB.RegisterRd != 0)
   and (EX/MEM.RegisterRd != ID/EX.RegisterRt)
   and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
   ForwardB = 01
Memory-to-Memory Copies

- For loads immediately followed by stores (memory-to-memory copies) can avoid a stall by adding forwarding hardware from the MEM/WB register to the data memory input.
  - Would need to add a Forward Unit and a mux to the memory access stage

Forwarding with Load-use Data Hazards
Forwarding with Load-use Data Hazards

Load-use Hazard Detection Unit

- Need a Hazard detection Unit in the ID stage that inserts a stall between the load and its use

2. ID Hazard Detection
   if (ID/EX.MemRead
   and ((ID/EX.RegisterRt = IF/ID.RegisterRs)
   or (ID/EX.RegisterRt = IF/ID.RegisterRt)))
   stall the pipeline

- The first line tests to see if the instruction now in the EX stage is a lw; the next two lines check to see if the destination register of the lw matches either source register of the instruction in the ID stage (the load-use instruction)
- After this one cycle stall, the forwarding logic can handle the remaining data hazards
Stall Hardware

- Along with the Hazard Unit, we have to implement the stall
- Prevent the instructions in the IF and ID stages from progressing down the pipeline – done by preventing the PC register and the IF/ID pipeline register from changing
  - Hazard detection Unit controls the writing of the PC (PC.write) and IF/ID (IF/ID.write) registers
- Insert a “bubble” between the lw instruction (in the EX stage) and the load-use instruction (in the ID stage) (i.e., insert a noop in the execution stream)
  - Set the control bits in the EX, MEM, and WB control fields of the ID/EX pipeline register to 0 (noop). The Hazard Unit controls the mux that chooses between the real control values and the 0's.
- Let the lw instruction and the instructions after it in the pipeline (before it in the code) proceed normally down the pipeline
Adding the Hazard Hardware

Instruction Memory
Read Address

Register File
Read Addr 1
Write Addr
Read Data 1
Write Data

Add

Hazard Unit
Control

ID/EX

ID/EX.MemRead

EX/MEM

MEM/WB

Branch

Forward Unit

ALU
Shift left 2
Add

Data Memory
Address Read Data

ALU cntrl

Forward

ID/EX,RegisterRs