Pipeline Hazards

Kevin Walsh
CS 3410, Spring 2010
Computer Science
Cornell University

See: P&H Chapter 4.7
add r3, r1, r2
sub r5, r3, r4
lw r6, 4(r3)
or r5, r3, r5
sw r6, 12(r3)
Data Hazards

- register file reads occur in stage 2 (IF)
- register file writes occur in stage 5 (WB)
- next instructions may read values about to be written

How to detect? Logic in ID stage:

\[
\text{stall} = (\text{ID.rA} \neq 0 \&\& (\text{ID.rA} == \text{EX.rD} \mid\mid \\
\text{ID.rA} == \text{M.rD} \mid\mid \\
\text{ID.rA} == \text{WB.rD}))
\mid\mid \text{(same for rB)}
\]
Detecting Data Hazards

add r3, r1, r2
sub r5, r3, r5
or r6, r3, r4
add r6, r3, r8

PC

+4

IF/ID

ID/EX

EX/MEM

MEM/WB

detect hazard
What to do if data hazard detected?
<table>
<thead>
<tr>
<th>Clock cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>add r3, r1, r2</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>M</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub r5, r3, r5</td>
<td>IF</td>
<td>ID</td>
<td>ID</td>
<td>ID</td>
<td>ID</td>
<td>ID</td>
<td>EX</td>
<td></td>
</tr>
<tr>
<td>or r6, r3, r4</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>add r6, r3, r8</td>
<td>IF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IF</td>
</tr>
</tbody>
</table>
\[ \text{ADD} r_3, r_1, r_3, r_0 \]

\[ \text{SUB} r_3, r_1, r_3, r_0 \]
How to stall an instruction in ID stage

• prevent IF/ID pipeline register update
  – stalls the ID stage instruction

• convert ID stage instr into nop for later stages
  – innocuous “bubble” passes through pipeline

• prevent PC update
  – stalls the next (IF stage) instruction
Clock cycle

1  2  3  4  5  6  7  8

add r3, r1, r2

sub r5, r3, r5

or r6, r3, r4

add r6, r3, r8
<table>
<thead>
<tr>
<th>Clock cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>add r3, r1, r2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub r5, r3, r4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lw r6, 4(r3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or r5, r3, r5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sw r6, 12(r3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Forward correct value from?

1. ALU output: too late in cycle?
2. EX/MEM.D pipeline register (output from ALU)
3. WB data value (output from ALU or memory)
4. MEM output: too late in cycle, on critical path

to?

a) ID (just after register file)
   maybe pointless?
b) EX, just after ID/EX.A and ID/EX.B are read
c) MEM, just after EX/MEM.B is read: on critical path
inst mem

ADD

LW
add r4, r1, r2

Nop

sub r6, r4, r1

IF  ID  EX  M  WB

IF  ID  EX  M

IF  4ID  EX
WB to EX Bypass

- EX needs value being written by WB

Resolve:

Add bypass from WB final value to start of EX

Detect:
add r4, r1, r2
sub r6, r4, r1
MEM to EX Bypass

• EX needs ALU result that is still in MEM stage

Resolve:

Add a bypass from EX/MEM.D to start of EX

Detect:
add r1, r1, r2
SUB r1, r1, r3
OR r1, r4, r1

IF  ID  EX  M  WB
add r4, r1, r2
sub r6, r4, r1

IF D X M WB

IF

IF (1D) EX
Register File Bypass

- Reading a value that is currently being written

Detect:

\[(Ra == MEM/WB.Rd) \text{ or } (Rb == MEM/WB.Rd)\]

and (WB is writing a register)

Resolve:

Add a bypass around register file (WB to ID)

Better: (Hack) just negate register file clock

- writes happen at end of first half of each clock cycle
- reads happen during second half of each clock cycle
Find all hazards, and say how they are resolved:

- **add**: \( r3, r1, r2 \)
- **sub**: \( r3, r2, r1 \)
- **nand**: \( r4, r3, r1 \)
- **or**: \( r0, r3, r4 \)
- **xor**: \( r1, r4, r3 \)
- **sb**: \( r4, 1(r0) \)
Load Data Hazard

• Value not available until WB stage
• So: next instruction can’t proceed if hazard detected

Resolution:

• MIPS 2000/3000: one delay slot
  – ISA says results of loads are not available until one cycle later
  – Assembler inserts nop, or reorders to fill delay slot

• MIPS 4000 onwards: stall
  – But really, programmer/compiler reorders to avoid stalling in the load delay slot
add r3, r1, r2
nand r5, r3, r4
add r2, r6, r3
lw r6, 24(r3)
sw r6, 12(r2)

M -> x
w -> x
RF Bypass
w -> x

STALL 1 cycle
+ w x
Delay slot
Delay Slot(s)
- Modify ISA to match implementation

Stall
- Pause current and all subsequent instructions

Forward/Bypass
- Try to steal correct value from elsewhere in pipeline
- Otherwise, fall back to stalling or require a delay slot

Tradeoffs?
SUB

inst
mem

ADD

D

B

BEQ

data
mem

L: or r3, r2, r4

1C

10
beq r1, r2, L

14
add r3, r0, r3

19
sub r5, r4, r6

F

D

X

0

1

2

3

4

F

F

F

F

N0
Control Hazards

- instructions are fetched in stage 1 (IF)
- branch and jump decisions occur in stage 3 (EX)
- i.e. next PC is not known until 2 cycles after branch/jump

Delay Slot

- ISA says N instructions after branch/jump always executed
  - MIPS has 1 branch delay slot

Stall (+ Zap)

- prevent PC update
- clear IF/ID pipeline register
  - instruction just fetched might be wrong one, so convert to nop
- allow branch to continue into EX stage
Control Hazards

- instructions are fetched in stage 1 (IF)
- branch and jump decisions occur in stage 3 (EX)
- i.e. next PC not known until 2 cycles after branch/jump

Stall

Delay Slot

Speculative Execution

- Guess direction of the branch
  - Allow instructions to move through pipeline
  - Zap them later if wrong guess
- Useful for long pipelines
Loops

while (R3 = 0)

END

nop

T88:

END: BWE T88 dest
Data hazards

- register file reads occur in stage 2 (IF)
- register file writes occur in stage 5 (WB)
- next instructions may read values soon to be written

Control hazards

- branch instruction may change the PC in stage 3 (EX)
- next instructions have already started executing

Structural hazards

- resource contention
- so far: impossible because of ISA and pipeline design