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:

\[
\text{stall} = (\text{ID.rA} \neq 0 \&\& (\text{ID.rA} == \text{EX.rD} \text{ | |} \text{ID.rA} == \text{M.rD} \text{ | |} \text{ID.rA} == \text{WB.rD})) \text{ | | (same for rB)}
\]
add r3, r1, r2
sub r5, r3, r5
or r6, r3, r4
add r6, r3, r8
What to do if data hazard detected?
add r3, r1, r2
sub r5, r3, r5
or r6, r3, r4
add r6, r3, r8
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
add r3, r1, r2
sub r5, r3, r5
or r6, r3, r4
add r6, r3, r8
Forward correct value from?  to?

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

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
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 r4, r1, r2

nop

nop

sub r6, r4, r1
Register File Bypass

- Reading a value that is currently being written

Detect:

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

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 \quad r3, \quad r1, \quad r2
sub \quad r3, \quad r2, \quad r1
nand \quad r4, \quad r3, \quad r1
or \quad r0, \quad r3, \quad r4
xor \quad r1, \quad r4, \quad r3
sb \quad r4, \quad 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)
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?
beq r1, r2, L
add r3, r0, r3
sub r5, r4, r6
L: or r3, r2, r4
beq r1, r2, L
add r3, r0, r3
sub r5, r4, r6
L: or r3, r2, r4
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
beq r1, r2, L
ori r2, r0, 1
L: or r3, r1, r4
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
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