Pipeline

Krerk Piromsopa, Ph.D.
Department of Computer Engineering
Chulalongkorn University

Stall and Performance

- If CPI = 1, 30% branch,
  Stall 3 cycles => new CPI = ?

Software Scheduling

- Try producing fast code for
- a = b + c;
- d = e – f;
- assuming a, b, c, d ,e, and f in memory.
  Slow code:

```
LW   Rb,b
LW   Rc,c
ADD  Ra,Rb,Rc
SW   a,Ra
LW   Re,e
LW   Rf,f
SUB  Rd,Re,Rf
SW   d,Rd
```

Fast code:

```
LW  Rb,b
LW  Rc,c
LW  Re,e
ADD Ra,Rb,Rc
LW  Rf,f
SW  a,Ra
SUB Rd,Re,Rf
SW  d,Rd
```

Branch and Pipeline

- Stall until branch direction is clear
- Predict Branch Not Taken
  - Execute successor instructions in sequence
  - "Squash" instructions in pipeline if branch actually taken
  - Advantage of late pipeline state update
  - 47% MIPS branches not taken on average
  - PC+4 already calculated, so use it to get next instruction
- Predict Branch Taken
  - 53% MIPS branches taken on average
  - But haven't calculated branch target address in MIPS
  - MIPS still incurs 1 cycle branch penalty
  - Other machines: branch target known before outcome

Branch and Pipeline

- Delayed Branch
  - Define branch to take place AFTER a following instruction
    -
```
            branch instruction
            sequential successor1
            sequential successor2
            ........
            sequential successorn
          branch target if taken
```
- 1 slot delay allows proper decision and branch target address in 5 stage pipeline
- MIPS uses this

Branch &Pipeline

Assume 4% unconditional branch, 6% conditional branch- untaken, 10% conditional branch-taken

Scheduling scheme	*Branch penalty*	*CPI*	*speedup v. unpipelined*	*speedup v. stall*
Stall pipeline	3	1.60	3.1	1.0
Predict taken	1	1.20	4.2	1.33
Predict not taken	1	1.14	4.4	1.40
Delayed branch	0.5	1.10	4.5	1.45

$$\text{Pipeline speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}$$

Pipeline Performance

- Hazards limit performance on computers:
  - Structural: need more HW resources
  - Data (RAW,WAR,WAW): need forwarding, compiler scheduling
  - Control: delayed branch, prediction
- Exceptions, Interrupts add complexity

Pipeline Performance

- Speed Up Pipeline Depth; if ideal CPI is 1, then:

$$\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}}$$