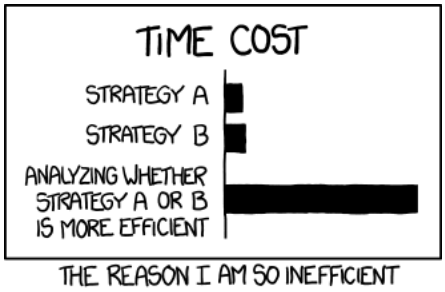
Lecture 14: Pipeline correctness

Friday, February 8, 2019 8:03 AM

Outline

- Finish up branch prediction
- Summary of hazards
- Exceptions



Branch prediction

Predict branches are always taken ↕ Static
always not taken

loops
backwards branches so always taken works very well

prevent stalling
Overcome or lessen impact of control hazards

better than always taken/not taken?
Special hint instructions (likely or unlike taken)
Choose dynamically
↳ History
look @ all recent instructions for a pattern
→ global history predictor
look @ the history for this particular branch
→ local history predictor

need some "memory" to hold history to make future predictions
↳ prediction table → set of registers/SRAM

bit if (x<x)
a = 12
same behavior else - -

beq if (a == 17)

Spectre
if (x < arr-size)
array[x]

Summary of hazards

Data Hazards
cause - data dependencies between instructions
solution + Forwarding is solution
negative effect - load to use → stall pipeline

Control hazards
cause - branches / jumps (control dependencies)
solution + branch prediction
negative effect - flush incorrect instructions

add
beq
lw
sub
control dependency

Exceptions → rare events

What happens when something goes wrong?

When something goes wrong → ask OS to deal

Kinds of exceptions?

- Memory access out of bounds
- div by 0
- don't have permission for instruction

- I/O → interrupts
- hardware faults
- software exceptions → syscall (ecall) ebreak
  ↳ gdb debugging breakpoint