

Discussion 08

Pipelining, Hazards

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

Announcements

Agenda

- Pipelining
- Hazards

Pipelining Basics

- Maximizes efficiency by using components of the datapath that are “sleeping” during each phase of processing
- **n**-stage pipeline means n instructions operate at the same time
- Maximizes throughput
 - Latency: how long it takes to finish 1 instruction
 - Throughput: number of operations finished per unit time
- Standard RISC-V 5 stage pipeline
 - **IF** , **ID** , **EX** , **MEM** , **WB**

How does this affect timing?

	Single cycle	n stage pipeline
Clock Cycle	Clock period of entire datapath	max(clock period of any stage)
Latency	Clock cycle	Clock cycle * n
Throughput (instruction / cycle)	1	1

Hazard Basics

- All hazards are caused by some kind of dependency
- Prevent future instructions from executing correctly or starting properly
- Types of hazards
 - Structural hazards
 - Data hazards
 - Control hazards
- Most common solution: stalls / NOP (bubble)

Structural Hazards

- The same piece of hardware is used by multiple stages
- Common situation:
 - Memory array (IF, Mem)
 - RegFile (ID, WB)
 - Hardware solution
 - Have the stages use separate components (IMEM + DMEM)
 - Adding multiple ports to RegFile to allow specialized access
 - Double pumping: 2 operations in the same cycle (e.g. write on rising edge and read on falling edge)
- Software solution
 - Stalling / NOP

Data Hazards

- Register value needs to be used before it's updated
- Common situation:
 - Register is computed in EX or MEM stage after it needs to be read
- Solution
 - Stalling / NOP
 - Forwarding

Control Hazards

- Caused by jumps / branches: we don't know which instruction to execute until EX stage
- Example

```
beq x0 a0 exit
addi a0 x0 1           # do we execute this line?
...
exit:
```

- Solution
 - Stalling / NOP
 - Branch prediction: guess whether a branch is “likely-taken” or “likely-not-taken” and flush half-executed instructions in pipeline if branch is not taken
- Don't worry too much about it

Thank you!