Structure of Final

The final is comprehensive, but most coverage is on material since midterm

Pipelining

- Overall operation
- Hazards
- Interrupts/Exceptions

Caching

- Overall operation and design space

VM/Paging

The "style" will be like the MT

1

Reading Summary

Chapter 1: Skim

Chapter 2: 2.1-2.8, 2.10; 2.12-2.14 Skim

Chapter 3: 3.5

Chapter 4: 4.1-4.4; 4.5-4.9

Chapter 5: 5.1-5.8

Chapter 6: 6.3, 6.5-6.6

[Some material on multicore from Ch. 7; not responsible for any of it.]

2

Design of Pipelined MIPS

MIPS design:

- ❖ Know all components and their operation
- ❖ Know flow of logic -- which components are active when implementing a given operation
- ❖ Be able to specify control signals needed to accomplish specific instructions
- ❖ Be able to compare with 1-cycle, multi-cycle
- ❖ Know the issues in pipelined performance
  - Why is write back last for addi?
  - Why do we want to move branch decision logic earlier?
  - Give examples of problems from instructions being started before preceding instructions are complete

3

Caching

Know the various forms of caching: direct mapped, fully associative, k-way set associative

Operation

Policies: wt, wb, wa, allocate on load, lru, etc.

Describe importance size of blocks, associativity, size, etc. on performance

Know terms: index, tag, valid bit, dirty bit, etc.

4

VM/Paging

Explain why "the RAM is considered a fully associative cache for the disk"

Full associativity in an L1 cache requires a "parallel compare" of tags to find a line … don't pages also require it to find the page?

For 16K pages, how large is the virtual page no.?

Give advantages/disadvantages of large vs. small page sizes

Why do we bother with a TLB?

What stuff goes into a TLB entry?

How is that TLBs are fully associative?

5

TLB

Explain why it's called a "TLB"

6

Skills

In the pipelined datapath, show which wires are active for forwarding for a given code frag?

In schematic diagram of pipelined instructions show bubbles, stalls and forwarding

Compute a physical address given a virtual addr

Determine if an address is in a cache

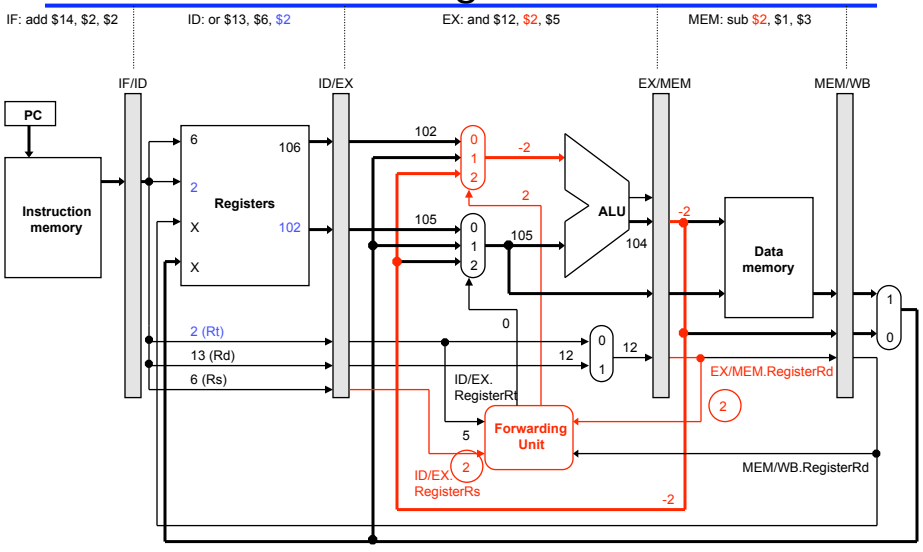
Determine if an page is in memory

Decide, given CPI and other data, which of two machines is faster

Revise assembly code to be hazard-free

7

Clock 4: forwarding $2 from EX/MEM



8