PSet 1

• pick up your homework
• (mostly) effort grade
  – 1 point per question
  – 5 points total
Agenda

• Some Questions from Last Week
• Quiz Prep/Pset #1 Review

Friday, February 11, 2011
Fully Bypassed Datapath

Is there still a need for the stall signal?

\[
\text{stall} = (rs_D=ws_E). (\text{opcode}_E=\text{LW}_E). (ws_E\neq 0). \text{re1}_D \\
+ (rt_D=ws_E). (\text{opcode}_E=\text{LW}_E). (ws_E\neq 0). \text{re2}_D
\]
When are branches resolved?

• Execute Stage?
  – two cycle penalty

• Decode Stage?
  – one cycle penalty
  – extra hardware

• Answer?
  – we will tell you!
MIPS (from wikipedia)

http://en.wikipedia.org/wiki/MIPS_architecture

Friday, February 11, 2011
Quiz 1

• This monday! (Feb 14)
• completely closed book (only pencils)
• Will look just like the pset
Quiz 1

• Possible topics
  – pipelining
    • bypassing/interlocking
    • precise exceptions
    • branches
  – microcoding
  – Iron Law
  – ISA design

• How best to study?
  – fully understand pset #1
  – go over past tests
  – go over past psets
Assume that compiler technology is poor, and therefore your users are far more apt to write all of their code in assembly. A ______ ISA would be most appreciated...

- CISC
- (arguably RISC)
• You desire to make compilers better at targeting your yet-to-be-designed machine....

• RISC
• Logic is super fast compared to instruction fetch (10x)...

• CISC
• Starting with a clean slate in 2011 (area/logic/memory is cheap)...

• RISC
  – RISC designed to be easily exploited by pipelined architectures
  – perfect for the mobile space
    • smaller design
    • cheaper to implement
    • mobile space allows clean break from binary compatibility

Problem Set 1 Review – P5

- A: Delay slot?
- B: Adding complex inst?
- C: reducing registers?
- D: faster memory?
- E: Adding 16b versions of instructions?
- F: Microcoded→RISC uops?
Administrivia Aside

Newsgroup
ucb.class.cs152
news.csua.berkeley.edu

Can use EECS web UI:
http://inst.eecs.berkeley.edu/webnews
Note:
#1 - only drives bus if ChipEnable=1, Write=0 (read mode)
#2 - Write/Read signal is “don’t care” if ChipEnable=0
Microcoded MIPS

Notice A and B are registered
- You *cannot* push to A/B and perform ALU Op with them in the same cycle
• For microcode problems, key is to get the pseudocode right
  – Control signals follow readily from pseudocode

• Sanity checks:
  – Only one device may drive the bus
  – The bus probably should be driven every cycle
  – Don’t read from a register whose write-enable was a don’t-care
• don’t cares
  – If you won’t read A/B/MA registers again, their write-enables should be don’t-cares
  – If enMem is off, Mem Wr is a don’t-care
  – If enReg is off, Reg Wr is a don’t-care
Problem Set 1 Review – P2

- P2A: $M[rd] \leftarrow M[rs] + M[rt]$
  - $MA \leftarrow R[rs]$
  - $A \leftarrow \text{Mem}$
  - $MA \leftarrow R[rd]$
  - $B \leftarrow \text{Mem}$
  - $MA \leftarrow R[rd]$
  - $\text{Mem} \leftarrow \text{ALU} (A+B); \ u\text{BR}=J$

- Note efficiency: 9 cycles vs. 18 for \text{ld,ld,add,stu}
• P2B (movn):
  – A ← R[rt]
  – uBr if A==0 (Fetch0); A← R[rs]
  – R[rd] ← A; uBR=J
Problem Set 1 Review – P3

• 3.A: only helps with ALU ops two away
• 3.B: M1→D
• 3.C: problem: writing back before we know exceptions didn’t occur in earlier instructions
• 3.D: stall ALU in Decode if inst in Exe is Lw/Sw
• 3.E: read current rd when in Decode, can restore on exception
Problem Set 1 Review – P1

• Skipping 1.A–1.C (solutions online)

• 1.D: What did you notice about relative code size for CISC, RISC, and stack machines?

• 1.E: What optimization strategies proved effective? Did anyone beat my solution (7 instructions)?