Agenda

- Lab Administrivia
- Lab 2
  - Working set
- Problem Solving
  - working set
  - caches
  - write buffers
  - optimizing software
- Lab 1, Question 2.5
  - Iron Law
  - power vs energy
- Lab 2 Scripting

Friday, February 25, 2011
Power 4

- 2001
- dual core
- 170M transistors
- “sun blocking” 400mm²
- 130nm–180nm
- 12 stages
- 8 execution units
  - 2 integer units
  - 2 floating units
  - 2 LD/ST units
  - 1 branch unit
  - 1 cond–eval
- “up to 200 instructions in flight”
Lab Administrivia

• Lab 1
  – Will be returned on Monday

• Lab 2
  – each student turns in a single, complete report
    • (group work encouraged, but turn in your own write–up)
  – turn in a lab report
    • “sanitize” your data
    • make graphs, tables
    • explain what you did
    • explain why you got what you got
Lab Administrivia

• Lab 2
  – Due this Wednesday!
Working Set

Benchmark X (log-linear)

Benchmark X (log-log)
Working Set

Benchmark X (log-linear)

Data Miss Rate

Cache Size (KB)

Benchmark X (log-log)

Data Miss Rate

Cache Size (KB)
Section Handout

- Form groups of 4
- Working Set Sizes
- Caches
• Write allocate
  • allocate space in the cache on a write-miss

• write-through
  • writes are immediately sent out to DRAM

• write-back
  • if it’s in the cache, perform write to the cache
  • mark it “dirty”
  • on eviction, write-back to the DRAM
Section Handout

- Form groups of 4
- Working Set Sizes
- Caches
- Write buffers
- Optimizing Software
- Power Vs Energy
Power vs Energy (from Lab 1)

- 5 stage processor
- 4 stage processor
  - combined ALU+MEM stage (in parallel, only run one)
  - MEM can’t handle address offsets

- Which is more power-efficient?
- Which is more energy-efficient?
• 5 stage processor
• 4 stage processor (parallel ALU+MEM)

• Iron Law:
  – instructions per program
    • increases for the 4–stage
  – seconds per cycle
    • ~stays the same!!!
  – cycles per instruction
    • decreases for the 4–stage
    • no load–use delay!
Power vs Energy (from Lab 1)

• 5 stage processor
• 4 stage processor (parallel ALU+MEM)

• Overall:
  – 4-stage will probably take more cycles to execute a program
Power vs Energy (from Lab 1)

• 5 stage processor
• 4 stage processor
  – combined ALU+MEM stage (in parallel, only run one)
  – MEM can’t handle address offsets

• Which is more power-efficient?
  – 4 stage (less area, less bypasses, less register state)
  – fewer transistors => less energy burnt per unit time (less power)
Power vs Energy (from Lab 1)

• Which is more energy-efficient?
  – Assume:
    • clock rates the same (seconds/cycle)
    • 4 stage burns 10% less power (10% smaller area)
    • 4 stage executes 30% more instructions for a given program
    • 5 stage stalls 5% of the time due to load-use delays
  
  – Answer:
    • the 5-stage is more energy efficient!
    • it burns more power, but it finishes faster, so it uses less energy overall!
    • (this is a so called “race to halt” scenario)
Power vs Energy (from Lab 1)

• more power efficient?
  – 4–stage
• more energy–efficient?
  – 5–stage

• Lab 1 Impact:
  – some students tried to justify the 4–stage for “energy reasons” and “despite more instructions” it would be “better”.
  – as we’ve shown, this is incorrect! The 4–stage is both slower and less energy–efficient!
Power vs Energy (from Lab 1)

• Final Caveats
  – assumed we’re measuring energy over a single program
  – assumed once finished, can go to a lower energy state
    • example:
      – cellphones can load a webpage, then go back to lower energy state
    • counter–example:
      – servers are always at ~20% load, so there is always work to do! No sleep for them!
      – thus power–efficiency ~ energy–efficiency (because the time interval is immaterial)
• Scripting *highly* recommended
  – Useful for removing tedium of labs
  – Helpful for talking to unresponsive target UI
• simics> commands can be put into a single .simics textfile
• invoke script:
  – host$ ./simics -c checkpoints/after_boot.conf scripts/example_script.simics
• can type commands on the target’s prompt
  – simics> con0.input “echo Hello”
This script:
- mounts host file system
- copies files over
- loads g-cache
- changes the size of the g-cache
- warms up a benchmark
- collects results
- prints them to the screen
• 2 levels of scripting can relieve even more burden!
• Great power
  – (great responsibility)
  – don’t hose the machine with jobs!
Questions?