CS465: Computer Systems Architecture

Final Review

Final Reminder

- Final time / location
  - May 11, 7:30-10:15am
  - Regular classroom
- Final will be close-book, close-notes, 4-way calculators allowed
  - MIPS ISA reference provided in exam
- Individual work
- To do well:
  - Go over the readings, solve exercises from homework, quizzes, classroom exercises and the textbook

Coverage

- Final exam is cumulative
  - Topics after midterm emphasized
  - Questions about earlier topics will be included and some are closely relevant
    - Performance calculation/comparison
    - Classic computer components
    - MIPS code reading/writing

Topics

- Before midterm
  - Introduction to computer systems
    - Performance
  - MIPS ISA
  - ALU implementation
  - Single-cycled processor (datapath basics)
- After midterm
  - Single-cycled processor (datapath, control)
  - Processor
    - Pipeline, ILP
  - Memory hierarchy
  - Multiprocessor

Reading List (Before Midterm)

- Chapter 1: Computer Abstractions and Technology
  - Sections 1.1- 1.3, 1.6- 1.11
  - Section 1.4: only need to know the components of a computer
  - Section 1.5: only need to know Moore's Law
  - Section 1.7: understand the issues but no need to remember the exact formula
- Chapter 2: Instruction Set Architecture
  - Sections 2.1-2.14, 2.19-2.20
- Chapter 3: Arithmetic for Computers
  - Sections 3.1 - 3.5, 3.9-3.10
- Chapter 4: The Processor
  - Sections 4.1-4.2
  - Section 4.1: only need to know the semantics of MIPS Lite, not the hardware details

Reading List (After Midterm)

- Chapter 4: The Processor
  - Sections 4.3-4.8,4.10, 4.14-4.15
  - Section 4.7: only need to know the high-level idea of the hazard checking, stall, and forwarding
  - Section 4.10: only need to know static multiple-issue and loop unrolling
- Chapter 5: Large and Fast: Exploiting Memory Hierarchy
  - Sections 5.1, 5.3- 5.4, 5.7-5.8, 5.16
- Chapter 6: Parallel Processors from Client to Cloud
  - Sections 6.1-6.3, 6.5,6.7, 5.10
  - Section 6.5/6.7: only need to know the basic models

Single-cycled CPU

- Single-cycle processor implementation based on MIPS Lite
  - Semantics/behavior of different instruction types
    - Trace datapath execution
- Control signals of different instruction types

Pipeline

- Pipeline
  - Basic 5 stages and datapath
  - Improved throughput, ideal CPI
- Hazards
  - Types
  - Detection and resolution
    - Forwarding, scheduling, branch prediction, stalling
  - Effect on performance (CPI w/ hazard stalls)
- Advanced ILP
  - Data dependences
  - Multiple issue and scheduling

Memory

- Memory hierarchy
  - Locality concept
- Cache
  - Block placement: direct-mapped, fully/set associative
  - Address mapping to block
    - Address division: index, offset, tag
  - Block replacement policy: LRU
  - Write policy
  - Performance effect
- Virtual memory
  - Main memory used as the cache of disk
  - Page table used for translation, TLB

Multiprocessor

- Multiprocessor architecture
  - Basic models and classifications
    - SIMD(vector), MIMD
    - SMP, message passing
- Cache coherence
  - Problem and solution
  - Snooping-based write invalidate protocol

Exercise

Single-cycled processor. Consider the following instruction mix:

R-type	Load	Store	Branch	Other I-type	Jump
20%	18%	25%	10%	24%	3%

- What fraction of all instructions use data memory?
- What fraction of all instructions use instruction memory?
- What fraction of all instructions use the sign-extend unit?
  - What does sign-extend unit do when its output is not needed?

Exercise

- Control signals of addi? Datapth support?
- Control signals of j? Datapath support?
- Can they operate correctly if ALUSrc is broken and stuck with 0?

Exercise

- Consider the following MIPS code:

I0: ADD $S4, $S1, $S0
I1: ADD $S4, $S4, $S3
I2: ADD $S4, $S4, $S3
I3: LW $S2, 100($S3)
I4: LW $S2, 0($S2)

- With no forwarding: insert **nop** in the given code sequence to indicate necessary stalls to run the sequence

-With full forwarding: draw diagram to show how the sequence is executed in the 5-stage pipeline(mark forwarding clearly)

- Can scheduling (reordering) help to reduce the number of cycles we need to execute the given sequence? How?

Exercise

We have a **2-way set associative** cache that uses 32-bit byte addresses with 4-bit offset field and 5-bit index field.

- The number of Tag bits =
- The size of each cache block in bytes =
- The number of blocks in **one set** =
- The number of **sets** in cache =
- The number of **blocks** in cache =

Exercise

- Suppose we want to achieve a 70X speedup from 100 processors.
- What fraction of the original program time can be sequential (i.e. cannot be parallelized) ?
  a. 10%
  b. 5%
  c. 1%
  d. <1%

Thank you and Good Luck!