Final Exam Preparation

CS5513 Computer Architecture, Fall 2024

Wei Wang

Computer Science
University of Texas at San Antonio

Goals and Topics

- The goal is to help you systematically review the basic knowledge in Computer Architecture.
  - Only basic knowledge is tested.
  - No trick questions.
- Topics for this exam (comprehensive, all topics):
  - Introduction
  - ISA
  - Performance Metrics and Measurement
  - Computer Arithmetic
  - Basic CPU Implementation
  - Pipelining
  - Branch Prediction
  - Out-of-order Exuection: Scoreboarding, Tomasulo, Reorder Buffer
  - ILP Limitation and Multi-threading
  - Introduction to Cache and Cache Designs
  - GPU

Location, Time and Logistics

- Dec 7th, 2:00pm-04:00pm 2024, Saturday, in person.
- Closed-book, closed-notes, closed-everything
- Do not waste time – if you stuck on a problem, move forward and revisit the problem later.
- Do not leave questions blank.
- The exam have Conceptual questions and Problems.
  - The problems will be similar as those in the assignments, past exams and the example questions.

Materials to Review

- Slides, all questions are from slides
  - Make sure to always get the latest slides. I will update them to fix errors.
- Assignments and projects.
- Past exams.
- You can check out the textbooks, but it is not required.
  - There are some differences in the details between my slides and the textbooks. Please follow my slides in those cases.
- Problems that definitely will be in the exam:
  - ISA (like exam 1 and assignment 1)
  - Amdahl's Law (like exam 1 and assignment 2)
  - OoO scheduling algorithms (like exam 2 and assignment 4)
  - Cache (like Assignment 5)

Introduction

- The three topics in Computer Architecture
  - ISA, micro-arch and system architecture.
- The definitions of Moore's Law and Dennard Scaling.
- The impact of the failure of Dennard Scaling.
- Design metrics for computer architectures:
  - Performance, cost, availability and power dissipation
- The definition of Von Neumann architecture and Harvard Architecture.

Instruction Set Architecture

- Be able to read basic instructions with source and destination operands.
  - Know the basic syntax.
- Four source devices for operands, and their examples.
  - Stack, x86 FP instructions.
  - Accumulator, x86 multiplication/division instructions.
  - Register-Register, most RISC ISAs.
  - Register-Memory, most CISC ISAs.
- Know all addressing modes and their examples from the slides.
  - Be able to write and recognize an addressing mode.
  - Especially the example for addressing elements in two-dimensional arrays.
- Fixed-length and variable length ISAs.
  - Their definitions, advantages and disadvantages.

Instruction Set Architecture cont'd

- CISC and RISC
  - The full names of CISC and RISC.
  - Examples ISAs of CISC and RISC.
  - The features of CISC and RISC (slide 50 from the ISA lecture)
  - CISC vs. RISC
    - CISC has better programmability, smaller code sizes
    - RISC is easier to implement, thus having fewer transistors and better energy efficiency.
  - Modern processors mostly use RISC internally.
- SIMD instructions, definition and examples
  - Examples: MMX, SSE, AVX, 3D!Now

Computer Arithmetic

- Remember the binary representations of numbers 0 to 15.
- Two's complement encoding.
  - Be able to write two's complement encoding given a decimal number.
  - Why two's complement?
  - The problem is similar to the one from Assignment 2.
- Floating point encoding.
  - Be able to write 32-bit encoding given a binary real number.
  - The problem is similar to the one from Assignment 2.
- Know the four logic gates, and know that logic gates are used to construct functional units.

Performance Metrics

- Why use benchmarks and simulators?
- Know the definition of MIPS, MFLOPS, CPI and IPC.
- Problems, very similar to those in the assignment and midterm exam.
  - Be able to compute average, weighted average, geometric and harmonic mean given some execution times.
  - Amdahl's Law
    - Review the equations and examples in the slides for Amdahl's Law.
    - You should be able to compute the overall speedup given a percentage of enhance-able part and a speedup.
- Know the relationship of instructions per program, cycles per instruction (CPI) and cycles per second (frequency).

Basic CPU Implementation

- Know the five stages of basic RISC and know what does each stage do.
  - IF, ID, EX, MEM and WB.
  - In particular, ID and MEM each has two operations.
- Understand the multiplexer.
  - A multiplexer is a device that selects one of several inputs.
  - A multiplexer is controlled by the control signal.
  - Know where the multiplexers are used in CPU, e.g., the selection of source operands for the ALU.
- Clock signals: edges, read and write at different edges.
- The data paths for three types of instructions: ALU, memory and branch instructions.

Basic CPU Implementation cont'd

- The CPU components/functional units used in each stage of execution.
  - There are questions about these components and their connections.
- Multi-cycle CPU design.
  - The benefits of multi-cycle CPU design.
- Exceptions
  - The definition of exceptions and interrupts.
  - The Control (unit) is responsible for handling exceptions.

Pipelining

- Know the solutions to all types of hazards
  - Slide 37 of the pipelining slides has a summary of these solutions.
  - You also need to know whether a solution can properly solve the hazards or not. In particular,
    - Why only branch prediction is the only practical solution to control hazard? Why other solutions do not work well?
    - Does data bypassing/forwarding eliminate stalls? And why?
- The definition of superscalar CPUs.
- At least one problem about pipelining. The problem is similar to those in the assignment and midterm exam
  - In particular, you need to know how the pipeline works when stalling is the only solution to the hazards.

Branch Prediction

- The basic two-bit saturate counter for branch prediction.
  - You need to memorize the state machine.
- Know the implementation of Branch Prediction Buffer and Branch Target Buffer.
  - What components do these two buffers have?
  - How does a branch locates its entries in these two buffers?
- Correlating branch prediction
  - Why does correcting branch prediction work for some branches?
  - Why does correlating branch prediction not work form some branches?
  - The implementation of correlating branch prediction with Global Branch History Register (GBHR) and two-bit saturate counters.
- There will be one problem about branch prediction, similar to the assignment and the one in the midterm exam.

OoO: Scoreboarding, Tomasulo's Algorithm and Reorder Buffer

- Be able to carry out the algrithms for OoO scheduling
- Why can OoO improve performance?
- What are the differences of scoreboards (SB), reservation station (RS) and reorder buffer (ROB)?
  - Scoreboard is a buffer for storing the status of the instructions
  - Reservation station is a buffer for holding the status and the source operands of the instructions.
  - Reorder buffer is a buffer for holding the status and the results of the instructions.
  - RS replaces SB, but ROB cannot replace RS.
- Common Data Bus (CDB)
  - What is the use of CDB? (sending results and source FU)
  - What is difference between CDB and the buses in pipelining? (CDB is a broadcasting bus)

ILP Limitation and Multithreading

- If we have unlimited resources in the CPU, what limits ILP?
  - RAW hazards
- What are the practical limitation on ILP?
  - Renaming register count, ROB size (instr window size), Branch prediction accuracy, memory aliasing accuracy, memory latency, FU count and latency
- Why Multithreading (MT) is adopted?
  - Most techniques to increase ILP failed
  - Many application naturally has TLP in them.
  - TLP is cheaper to exploit than ILP at the moment.
- Know the characteristics of the three types of MT: fine-grained, coarse-grained and SMT.

Introduction to Cache

- Know the basic types of memory devices: registers, L1 cache, L2 cache, L3 cache, SSD and HDD.
  - In particular, their relative speeds and maximum size.
- Why do caches improve performance?
  - Temporal and spatial locality.
- Basic terminologies: cache hit, cache miss, hit rate, miss rate and block/cache-lines.
- Miss penalty = access time + transfer time.
- Four policies about caches: cache line identification policy, cache line placement policy, cache line replacement policy and write strategy.
- A cache frame includes data, tag and state bits.

Cache Designs

- Cache Placement Policies
  - Understand the designs of direct-mapped, fully-associative and set-associative caches
    - A direct-mapped cache is a set-associative cache with **One Way**.
    - A fully-associative cache is a set-associative cache with **One Set**.
  - Know the pros and cons of these three types of caches.
    - Which one is the fastest/slowest?
    - Which one has the lowest/highest cache miss rates?
  - Given a cache configuration, you should be able to compute the numbers of ways and sets, and can determine the mapping of an memory address to a particular set.

Cache Designs cont'd

- For set-associative cache, how to determine the best number of ways?
  - Using benchmarks and simulators (you did this in the project).
  - Be able to compute the AMATs like those in the table of the slide 53 of the "Cache Designs" lecture.
    - Also see the example problem.
- Know the definition of 3 C's: Compulsory miss, Capacity miss and Conflict miss.
  - Compulsory misses are determined by program behaviors and are generally not affected by cache configurations.
  - Larger caches have fewer capacity misses.
  - More ways reduce conflict misses.

Cache Designs cont'd

- Cache Replacement Policy
  - Know the LRU algorithm. Given a sequence of memory accesses and a cache, you should be able to compute the number of cache misses.
    - Check out the example in the lecture slides.
    - The problem will be similar to Assignment 5.
  - Know that LRU algorithms are implemented with approximations in the hardware.
    - No need to remember the clock algorithm.

Cache Designs cont'd

- Write Policy
  - Write-back + write-allocate
    - Pros: fewer DRAM writes; faster writes; less usage of DRAM bandwidth and power;
    - Cons: cache and DRAM inconsistent; evictions may be longer; may increase cache miss rates (streaming writes pollute caches)
  - Write-through + no-write-allocate
    - Pros: cache and DRAM are consistent (good for I/O memory); easier to implementation; does not pollute cache.
    - Cons: Slow writes; need more DRAM bandwidth and power.
    - Write buffers are used to improve write speed and reduce DRAM bandwidth usage.
    - Write buffers can also serve read requests.
    - Write buffer sizes and flush rates must be properly determined.

GPU

- GPU will be on the exam, because we need to know why GPUs are used for machine learning.
- CPUs are designed for general purpose applications.
  - These applications have low parallelism, but good data locality.
  - So CPU designs use complex control logics to discover instruction parallelism and use large caches to exploit data locality.
- GPUs are designed for graphics applications and applications with heavy SIMD operations.
  - These applications have high parallelism, but poor data locality.
  - So GPU designs use simple controls logic with large number of Floating-point functional units to exploit the parallelism in SIMD.
  - GPUs also have smaller caches, but more memory channels to improve DRAM performance.