Midterm 1 Study Guide

Midterm will consist of multiple choice, true/false, short answer, and problems to work through.

Chapter 1 – Fundamentals of Computer Design

- Different goals for different classes of computers
- Discuss the power wall, ILP wall and memory wall and the implications of each
- Why has multicore technology become so important?
- In optimizations, why is it important to focus on the common case?
- Trends in technology
  - Integrated circuit logic technology - Moore's law
  - Performance trends – Bandwidth over Latency
  - Trends in power
    - Clock gating
    - Static power vs. dynamic power
- Dependability
  - Mean time to failure (MTTF)
    - Why is this misleading?
  - Mean time between failures (MTBF)
  - Module availability
  - Problem similar to the example on page 26
  - Problem similar to the example on page 27
  - Why is a single point of failure bad?
- Measuring, reporting and summarizing performance
  - Comparing two systems, showing speedup
    - Equations on page 28
  - Throughput
  - How do you define time for comparisons?
  - Benchmarks
    - Why are benchmarks important?
    - Why do benchmarks need multiple applications?
    - Why is it important to run the entire benchmark suite and not a subset?
    - Why is it necessary to create new benchmark suites?
- Quantitative principles of computer design
  - Amdahl's Law
    - Define
    - Use
    - Why is Amdahl's law fundamental to system design?
    - Problems similar to those on pages 40-41
  - Processor performance equation
    - Calculate CPU time
    - Calculate CPI
    - What are the components of CPI?
    - Problem similar to the one on page 43

Appendix A – Pipelining: Basic and Intermediate Concepts

- What is pipelining?
- What is parallelism?
- How does pipelining exploit parallelism?
- What are the advantages and disadvantages of a deeper pipeline?
- What is a RISC machine (and I don't just want the acronym)
  - In terms of types of instructions and structure of instructions
- Why is a RISC machine easy to pipeline and a CISC machine more difficult?
- What are the 5 pipeline stages? What happens in each stage?
- How can the register file be used in two pipeline stages?
- What is the purpose of pipeline registers? What information do they hold? Why are they essential to pipelining?
- The major hurdle of pipelining – pipeline hazards
  - What are the 3 pipeline hazards?

    - Why must a pipeline stall?
    - What is the difference between a data dependency and a data hazard?
        - Give code that shows a both data dependencies and data hazards and identify both
    - Show how pipeline stalls can effect the CPI
        - Equations on page A-12
        - Exercise on page A-13
    - Identify potential structural hazards in the standard 5 stage MIPS pipeline. How are these hazards avoided?
    - What mechanisms exist for minimizing stalls due to data hazards?
        - Show the flow of data as in figures A.7 and A.8
        - Which data hazards always results in a stall? Give an example in assembly. How can this stall be avoided.
    - Branch/control hazards?
        - What are they?
        - Why are they such a problem in pipelining
        - What mechanisms exist for reducing the effects of branch hazards? What are each of the following and discuss advantages/disadvantages
            - Stall then flush pipeline if necessary
            - Predicted not taken
            - predicted taken
            - Delayed branch
                - What is a branch delay slot? How is it filled (3 possibilities)?

Chapter 2 – Instruction-Level Parallelism and Its Exploitation

- What is ILP?
    - How does pipelining exploit ILP? Why is the pipeline essential to exploit ILP?
- Why is speculation imperative to exploiting more ILP?
- What is a data dependency?
- What is a name dependency?
    - Why are there name dependencies and how can we overcome them?
- What is a control dependency?
- Data hazards
    - What are the three types of hazards?
    - Give an example of each with assembly code
- Basic compiler techniques for exposing ILP?
    - Pipeline scheduling and loop unrolling
    - Example on page 76
    - Example on page 77
    - Example on page 78
    - Slides 17-22, you might have to do something similar
    - Why is loop unrolling hard? What things must be considered? What fundamental requirements in loop structure are necessary to fully exploit loop unrolling?
        - Register pressure
- Reducing branch costs with prediction
    - What is branch prediction? How does it affect CPI?
    - Compare and contrast static and dynamic branch prediction
    - What are branch prediction buffers?
        - Branch history table
            - Why can a simple branch history table using 1 bit be worse than just always predicting that a branch is taken?
        - Correlating branch predictors?
        - Tournament predictors?
        - Why is local and global information important?
- Calculate CPI given an instruction mix and branch prediction.
- Dynamic scheduling
    - What is dynamic scheduling?
    - What is the purpose of it?
        - What does it try to avoid?
        - What must it maintain?
        - Who is it better than static scheduling (compile time)
    - What is the limitation of a simple pipelining and how does dynamic scheduling attempt to overcome this?

    - What is the difference between in-order execution and out-of-order execution?
        - How does out-or-order execution introduce WAR and WAW hazards? Give examples in assembly code
        - What mechanism exists for dealing with these hazards?
    - What is out-of-order completion?
        - What are the implications of it? (don't forget exception handling)
        - What are imprecise exceptions?
    - Tomasulo's algorithm
        - Why was it developed?
        - What are the goals?
        - What are the two major advantages? What are the drawbacks?
        - Draw a block diagram
        - When is the register file accessed during execution? When is it not accessed?
        - What is the purpose of the following components in Tomasulo's algorithm?
            - Reservation stations
                - What are the fields within the reservation station and how are they used
            - Common data bus
        - Discuss the workings of Tomasulo's algorithm
        - How does Tomasulo's algorithm effective unroll loops dynamically?
        - Show the state of the system after a few instructions have executed
            - Similar to the example given in the slides
            - Review examples in section 2.5
        - How do reservation stations affect RAW, WAR, and WAW hazards?
        - How do the reservation stations assist in register renaming?
        - What does the reorder buffer add to Tomasulo's algorithm?
    - Hardware based speculation
        - Three key ideas:
            - Dynamic branch prediction
            - Control flow speculation
            - Dynamic scheduling
        - What is in-order commit?
            - How is this important for speculation
            - How does the reorder buffer assist in in-order commit?
        - What is the reorder buffer?
            - Why is it important?
            - When using a reorder buffer, when are the results reflected in the register file?
        - Show hardware structure of system with reorder buffer
        - Trace an example using a reorder buffer
            - Similar to the example in the slides
    - Reducing the CPI to less than one
        - What key mechanisms are required to reduce the CPI to less than one?
        - What is a VLIW processor?
            - What are the advantages and disadvantages to VLIW
    - Branch target buffers

Chapter 3 – Limits on Instruction-Level Parallelism

- Discuss key issues that limit the amount of ILP we can achieve?
- Define TLP
- Define multithreading
- Multithreading: Using ILP Support to Exploit TLP
    - What is the key idea in using ILP to exploit TLP? What is the concept of reuse?
    - Compare and contrast fine grained and coarse grained TLP
        - Discuss advantages and disadvantages to each
    - What is simultaneous multithreading?
        - How is it different or the same as multi-processing
        - How is it different than multithreading

Chapter 4 – Multiprocessors and Thread-Level Parallelism

- Taxonomy of parallel architectures
    - What are SISD, SIMD, MISD, and MIMD?
- Amdahls law and speedup equations

    - Give a percentage of a program that is parallelizable, calculate the speedup obtained using different numbers of CPUs
- Uniform memory access vs. non-uniform memory access
- Centralized shared memory model vs distributed memory model
    - Advantages and disadvantages
- What is cache coherency?
    - Why does this problem exist?
- Private data vs. shared data
- Cache coherency schemes provide migration and replication of shared data items. What is migration and replication?
- What are the two cache coherency protocols that we discussed? How are they similar and how are they different? What are the advantages and disadvantages of each, if any?
- Discuss the basic idea of the snooping protocol and directory based protocols
    - How do the work
    - What information is stored for each cache block?
    - Know what all the states are and how the transitions work between each state (all of the state diagrams in the lecture slides)
    - In either protocol, how does a processor see the state of memory?
- What is false and true sharing and how are they similar and the same. Include discussion of block size
- What is an atomic operation and why is it necessary for sharing data?