Midterm 1 Study Guide

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
  o Integrated circuit logic technology - Moore’s law
  o Performance trends – Bandwidth over Latency
  o Trends in power
    ▪ Clock gating
    ▪ Static power vs. dynamic power
• Dependability
  o Mean time to failure (MTTF)
    ▪ Why is this misleading?
  o Mean time between failures (MTBF)
  o Module availability
  o Problem similar to the example on page 26
  o Problem similar to the example on page 27
  o Why is a single point of failure bad?
• Measuring, reporting and summarizing performance
  o Comparing two systems, showing speedup
    ▪ Equations on page 28
  o Throughput
  o How do you define time for comparisons?
  o 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
  o Amdahl’s Law
    ▪ Define
    ▪ Use
    ▪ Why is Amdahl’s law fundamental to system design?
    ▪ Problems similar to those on pages 40-41
  o 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)
  o 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
  o What are the 3 pipeline hazards?
  o Why must a pipeline stall?
  o 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
  o Show how pipeline stalls can effect the CPI
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)?

Appendix C
- The 36 terms on page C-2
- What are the 3 C’s in cache misses? What causes them and how can they be reduced/avoided?
- The 4 memory hierarchy questions on page C-6
- Calculate average memory access time as in the example on page C-15 and C-16, C-26, C-31, 295,
- 6 Basic cache optimizations – what are they and how do they improve cache performance? Do they always improve performance or does it depend on the benchmark?
  - larger block size to reduce miss rate
  - Larger caches to reduce miss rate
  - Higher associativity to reduce miss rate
  - Multilevel caches to reduce miss rate
  - Giving priority to read misses over writes to reduce miss penalty
  - Avoiding address translation during indexing of the cache to reduce hit time
- The difference and impacts of write-back versus write-through caches.
- Virtual memory
  - What is it?
  - What is its purpose?
  - How does it help a program? How does it hurt a program?
  - What does it mean to have a cache that is virtually indexed virtually tagged or virtually indexed physically tagged? What are the advantages and/or disadvantages of either way
  - What are page tables and what do they mean for virtual memory?
  - How can you speed up address translation?

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
  o What is branch prediction? How does it affect CPI?
  o Compare and contrast static and dynamic branch prediction
  o 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?
    ▪ How are branches addressed in the predictors? What is the aliasing problem? What problems can this lead to? How can this be overcome? Is it worth it?

• Dynamic scheduling
  o What is dynamic scheduling?
  o 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)
  o What is the limitation of a simple pipelining and how does dynamic scheduling attempt to overcome this?
  o 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?
  o What is out-of-order completion?
    ▪ What are the implications of it? (don’t forget exception handling)
    ▪ What are imprecise exceptions?
  o 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
        o 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?
  o 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
  o 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
  o Branch target buffers