

# Review For Final

## CS4760

---

Note that the final is NOT cumulative. While it may include a few concepts from the material before the midterm, the questions will focus mainly on the following material:

### 1.0 Instruction-Level Parallelism

- **Dynamic Scheduling:**
- **Scoreboarding**
- Understand how it differs from normal pipeline execution--what pipeline stages are being changed/replaced?
- What are the phases of execution in scoreboarding?
- What do the data structures look like?
- Understand the algorithm and be able to do an example
- What hazards can occur, and where are they checked and prevented?
  
- **Tomasulo's Algorithm**
- Understand main differences between Tomasulo's and scoreboarding
- What are the main hardware features of Tomasulo's algorithm?
- What hazards are encountered, and which are not? Why?
- What are the phases of execution in Tomasulo's?
- What do the data structures look like?
- Understand algorithm and be able to do an example
  
- **Dynamic Hardware Branch Prediction**
- Understand the following techniques
- In what phase of the pipeline are they used?
- Is this an effective technique for DLX?
- What elements are stored in the data structures?
- **1. Branch Prediction Buffer**
- Understand 1-bit vs. 2-bit vs. n-bit predictors -- how many bits to be effective?
- How many entries in the buffer are needed?
- **2. Branch Target Buffers**
- What elements are stored in the branch target buffer?

- **Loop Unrolling**
- Understand the technique and be able to do a simple example
  
- **Superscalar and VLIW**
- Characteristics of superscalar and VLIW
- Limitations of multiple-issue processors
- Study examples of (scoreboard + Tomasulo) and (VLIW + loop unrolling)
  
- **Hardware support for extracting more parallelism**
- Understand conditional or predicated instructions
  
- **Hardware Speculation**
- \*\*\*\* Review your code for project 2 \*\*\*\* : be able to answer questions on your logic
- Study examples of Tomasulo's algorithm in combination with speculative execution
  
- **Software pipelining**
- Understand how this works (conceptually)
- Understand the different problems addressed by this technique and loop unrolling
  
- **Caches**
- Four questions characterize memory hierarchies
- Where can a block be placed in the upper level (Block placement)?
- How is a block found if it is in the upper level (Block identification)?
- Which block should be replaced on a miss (Block replacement)?
- What happens on a write (Write strategy)?
  
- Understand associativity: fully associative, direct-mapped and set-associative caches
- Understand the terminology (n-way set-associative means a set of size n)
- Understand the parts of an address and how they are used types of caches to decide whether there is a hit or miss in the cache
- Understand replacements policies, which works best
- Understand copy-back and write through caches
- What is the difference between write-allocate and write around?
- How does a write buffer improve performance?

- Classifying misses: what are compulsory, capacity and conflict misses?
- Which arise/don't arise caches of various associativities?
  
- Understand the improvements we discussed to various parts of the average memory access time
- $AMAT = HT + MR * MP$
  
- Reducing miss rates with
  1. Larger block sizes: why does this help?
  2. Higher associativity: what kinds of misses does this eliminate? What cost may there be for increasing associativity? What is the point of diminishing returns?
  3. What is a victim cache? How effective?
  4. (Don't worry about pseudo-associativity!)
  5. Hardware prefetching: understand stream buffers
  6. Software prefetching: understand register vs. cache prefetches
  7. (Don't worry about compiler optimizations)
  
- **Reducing Miss Penalty**
  1. What are the ways we give reads priority?
  2. Be able to explain sub-blocks and what is the difference between them and just smaller blocks
  3. How do early restart and critical word first work?
  4. Explain in general terms a non-blocking cache
  5. Second-level and multiple-level caches: be able to indicate what are the different associativity levels and sizes expected in first versus second-level caches
  
- **Reducing Hit times**
  1. Direct-mapped caches: why are these faster?
  2. Virtually-addressed caches: why do these have faster hit times  
Be able to explain the difference between accessign a physically-addressed and virtually-addressed cache
  3. Pipelined writes
  
- Be able to calculate the CPI when memory accesses are taken into account

- **Memory systems**
- What are the basic differences between DRAM and SRAM?
- What are RAS/CAS and how are they used?
- What are the basic types of memory chips?
- Understand interleaved and independent memory banks
  
- **Virtual Memory**
- Understand virtual memory and the movement of pages between memory and disks
- What are the advantages/disadvantages of paging vs. segments? of large vs. small pages?
- What is the associativity, block replacement strategy, write policy of virtual memory?
- Understand the TLB and page tables
- Understand how the data structures are accessed differently depending on whether the cache is physically or virtually addressed
- Does a miss in the TLB tell you whether the data is in cache? In memory? Why?
- Understand how translation occurs in virtually-addressed vs. physically-addressed caches
  
- **I/O**
- Understand how disks work (seek, rotation, data transfer)
- Understand: striping for high performance, redundancy for high reliability
- Understand the differences between RAID Levels 1, 3, 4, and 5
  
- **Multiprocessors**
- Understand the difference between centrally shared memory and physically-distributed shared memory multiprocessors
- For centrally shared multiprocessors, understand write-invalidate snoopy cache coherence protocols
  
- **Project 3**
- Review fact sheets for all the projects, looking for general technology trends  
For example:
  1. Approximate clock rates, transistor sizes, cache sizes of current and future high performance processors
  2. Approximate processor counts, configuration and performance of large multiprocessors
  3. Capacity and bandwidth of I/O technologies such as RAMBUS and DVD.