CSCI 4717 Test One Study Guide

Material Covered:

Basically, any material covered in the lectures and labs is fair game for the test. As far as the lectures are concerned, the test covers the material we covered up to and including the pipelining PowerPoint lecture that covered on Thursday, September 20th. We only got to slide 24 titled, "In-Class Exercise," Below I've tried to identify the specific areas you should ***concentrate*** on. This doesn't mean that other stuff won't be seen - just that this stuff will definitely be seen.

- What constitutes computer architecture and how does it differ from computer organization.
- What is important when measuring performance, e.g., is performance always about GigaHertz?
- Don't worry about programming the old, historic machines even though I gave you the instruction sets for them. What I need you to understand is what we learned from them. For example,
    - What is the method of differences?
    - What are the differences between the von Neumann and Harvard architectures?
    - What was important about what von Neumann proposed with his IAS machine?
    - What were the important achievements of the Manchester "Baby"?
    - What were the difficulties surrounding programming of these old machines?
    - What features would you most want to add to them to make them more usable?
- You should understand the mechanics, benefits, and drawbacks of the different addressing modes. In addition, you should understand the operation, benefits, and drawbacks of 3-, 2-, 1-, and 0-address instructions. For example, how do the differences affect the compiler writer? You will not be asked to derive the RPN representation of a mathematical function, but if given the RPN representation, you should be able to use it to create a 0-address instruction program. Oh, and you need to understand why we have data structure alignment.
- Please review the operation of the stack. I know that the operation of a stack should be obvious, but knowing about pre/post increment/decrement with respect to writing MIPS or ARM code is important. There may be a time when you have to write code on an assembly language machine that requires hand-coding of the stack.
- I need for you to understand the issues surrounding instruction set design and addressing modes. This includes instruction set size, variable versus fixed opcode lengths, number of operands (addresses) used by the instructions, and the application of all the different addressing modes. Be able to discuss the "Execution of a Machine Instruction" figure on slide 3 of the Instruction Set Design lecture slides.
- Unfortunately, the cache slides are rather brief, so you may need to rely on the reading and your notes. There is, however, a worksheet that you can use to study. Please understand
    - the basic operation of a cache;
    - the partitioning of memory into blocks;
    - each of the three cache mapping algorithms and their operation;
    - the differences between how caches handle data and instructions; and
    - issues when writing to the cache and the complexity of multiple processors with caches. (We will study the details of how we handle multiple processors later in the semester.)
- As far as pipelining is concerned, you need to be able to compute the ideal speed-up of

a k-stage pipeline, the factors limiting the realization of an ideal speed-up, and how to compute the number of cycles required to execute a block of code based on the code's characteristics (slide 24). We did not get to CISC pipelines versus RISC pipelines and branch prediction algorithms, so that will be available for the next test!

- I won't be asking you to write any assembly language code, but I might use assembly language as examples of architectural features that you'll need to understand. I will provide instruction syntax and descriptions for any code I use.

Developed by David Tarnoff for CSCI 4717 -- Computer Architecture at ETSU