Miami Dade College

Course Description

CET3126C | Computer Architecture | 4.00 credits

This course is intended for upper division students majoring in Electronics Engineering Technology as well as Information Systems Technology. This course introduces the study of advanced microprocessor design. Students will learn the basic organization of computer systems including instruction-set architecture, execution pipeline, memory hierarchy, virtual memory, and I/O subsystems. Students also learn advanced processor microarchitecture issues such as dynamic instruction scheduling, branch prediction, lock-up free caches, instruction-level parallelism, multiple instruction fetch/ issuing, speculative execution, etc. to improve computer processor performance. Students will experimentally verify microarchitecture designs using industry standard microarchitecture simulators.

Course Competencies:

Competency 1: The student will demonstrate an understanding of the history, key terminology, and current industry trends in computer architecture by:

1. Describing the history and development of computational devices.
2. Identifying the different classes of computing applications and their characteristics (i.e. desktop computers, servers, and embedded computers).
3. Listing the classic components of a computational system (i.e. input, output, memory, data path, control) and understanding their functionality.
4. Discussing the implications of the evolution of transistor sizes and the impending Power Wall.
5. Discussing the significance of switching from uniprocessors to multiprocessors.
6. Identifying the steps of the manufacturing process of integrated circuits.
7. Calculating performance measures of a system using standard industry metrics such as Cycles Per Instruction (CPI), Instructions Per Cycle (ICP), and Speedup.

Competency 2: The student will demonstrate an understanding of how data is represented within a computer system by:

1. Identifying and performing operations as implemented in hardware.
2. Converting between decimal, binary, and hexadecimal representations.
3. Converting between signed and unsigned numbers.
4. Converting between decimal and floating-point rotation.
5. Identifying and performing logical operations as implemented in hardware.
6. Distinguishing between Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC) instruction sets and identifying the advantages and disadvantages of each.
7. Identifying the different ISA addressing modes of an instruction set architecture (e.g. MIPS).
8. Converting between binary, assembly code, and higher-level instruction.
9. Comparing various programming languages including compiled and interpreted languages.
10. Comparing the implications of 32-bit instruction set architecture (ISA) versus 64-bit ISAs.
11. Enumerating the major ISAs prevalent in the industry.

Competency 3: The student will demonstrate an understanding of the processor (CPU) subsystem by:

1. Identifying the major components of a CPU (e.g. instruction/data cache, registers, ALU).
2. Describing the stages of the common five stage pipeline (i.e. instruction Fetch, Instruction Decode, Execute, Memory, Write Back).
3. Identifying data paths for various operations including arithmetic operations of a microprocessor.
4. Identifying control paths for various operations of a microprocessor.
5. Describing the implementation of pipelining data paths and controllers.
6. Identifying data hazards including Read after Write (RAW), Write after Write (WAW), Write after Read (WAR), and deciding between forwarding or stating the pipelined data.

7. Identifying and managing control hazards.
8. Calculating the theoretical speed-up from pipelining.
9. Identifying the common forms of branch prediction (e.g. static, two-bit, bimodal, etc.).
10. Determining the success rate of a given branch predictor for a sequence of code.
11. Discussing advanced instruction level parallelism concepts such as state multiple issue, dynamic multiple issue (i.e. superscalar), speculation, out of order execution, and register renaming.

Competency 4: The student will demonstrate an understanding of the memory subsystem and hierarchy by:

1. Discussing the need for multilevel hierarchical memory design including registers, cache, memory, and secondary storage.
2. Identifying the basics of caches including direct- mapping, set-associative, fully associativity, and common replacement policies.
3. Designing caches of various sizes and associativity.
4. Tracing the hits and miss rate of a cache given a cache scheme and a pattern of memory addresses.
5. Calculating the hits and miss rate of a cache given a cache scheme and a pattern of memory accesses.
6. Describing the various methods of cache coherence.
7. Discussing the performance and security implications of virtual memory.
8. Distinguishing between virtual and physical memory addresses for given physical memory configurations.
9. Discussing the Translation Lookaside Buffer (TLB) and the processes by which the hardware translates a virtual memory address to physical memory address.
10. Calculating slowdown due to memory speeds.

Competency 5: The student will demonstrate an understanding of the various types of storage and the Input/output (I/O) subsystem by:

1. Identifying the main aspects of storage (i.e. dependability, reliability, and availability.
2. Distinguishing between persistent versus non- persistent storage memory (memory vs. I/O)
3. Calculating disk capacity given sectors, heads, etc.
4. Identifying the various RAID configurations.
5. Describing the process by which the processor, memory, and I/O devices are connected.
6. Determining I/O performance measures.

Competency 6: The student will demonstrate an understanding of the issues associated with multi- processors and benchmarking by:

1. Discussing the difficulties faced in creating parallel processing programs.
2. Describing the implication of shared memory multiprocessors and multicore devices.
3. Distinguishing clusters and other message-passing multiprocessors.
4. Describing the benefit of hardware multithreading.
5. Calculating the theoretical speedup from N- processing cores versus 1 processing core.
6. Identifying specialized multi-processing architecture (i.e. SISD, MIMD, SIMD, SPMD, and Vectors).
7. Articulating the need for benchmarking and benchmark suites.
8. Describing the relevance, benefits, and limitations of benchmarking.
9. Identifying popular benchmarking suites available for use.

Competency 7: The student will demonstrate practical skills related to computer architecture research by:

1. Determining the “instruction mix” of industry standard benchmarks through the use of profiling tools.
2. Experimentally verifying the performance of cache modifications such as replacement policy, size, and associativity using industry standard benchmarks (e.g. SPEC) and simulation software (e.g. SimpleScalar, M5, etc.)
3. Experimentally verifying the performance of branch prediction modifications such as different schemes (e.g. perfect, always taken, always not taken, bi-modal, etc.) sizes using industry standard benchmarks (e.g. SPEC) and simulation software (e.g. Simple Scalar, M5, etc.)

4. Presenting a research paper from a recent industry conference (e.g. ASPLOS, ISCA, etc.) or industry journal (e.g. IEEE transactions, SIGARCH, SIGGRAPH, etc.).