Course Description
CMPS 321

Computer Architecture

Catalog Description:
This course follows the Digital Logic Design course and focuses on the design of the CPU and computer system at the architectural (or functional) level: CPU instruction sets and functional units, data types, control unit design, interrupt handling and DMA, I/O support, memory hierarchy, virtual memory, and buses and bus timing. In contrast, the Digital Logic Design course is primarily concerned with implementation: that is, the combinatorial and sequential circuits which are the building blocks of the functional units.

Prerequisite:
CMPS 223

Units:
5

Instructor:
Marc Thomas

Goals:

(AR3) Instruction sets and assembly level machine organization: Understand the basic functional units (e.g. control unit, hardwiring vs. microcoding, ALU, registers, buses, interrupt architecture, basic fetch-execute cycle, etc.) in computer architecture.

(AR4) Memory system organization and architecture: Understand memory hierarchy (e.g. L1 caches, L2 caches, main memory, swap devices), fabrication issues, virtual memory systems and paging, compilations of multiple CPU’s (e.g. cache coherancy algorithms).

(AR5) Interfacing and communication: DMA, interrupt driven I/O, and interrupt driven I/O with DMA.

(AR6) Functional organization: Understand instruction set design (e.g. RISC and CISC), use of multiple functional units, static scheduling and pipelining.

(AR7) Multiprocessing and alternative architectures: Understand SMP systems and cache coherency.

Understand the design tradeoffs involved in computer architecture.

Understand how poor programming can result in very inefficient use of the hardware.

(Laboratory) Become proficient in writing, profiling, and analyzing programs which need to make intensive use of one or more of the following architectural features: the CPU, virtual memory, and I/O devices.

Texts:
(historical) John Hayes Computer Architecture and Organization
(historical) Levy and Eckhouse Computer Programming and Architecture: the VAX-11

Topics:
Introduction; Turing Machines; what constitutes a general purpose computer; design of a minimal hardwired CPU.
Some historical development of computing and some case studies; discussion of how fabrication has evolved; speed of operation and power consumption as they relate to VLSI design.

The basic fetch-execute cycle without interrupts; memory organization with stack, data, and code segments; how performance can be measured; REAL, VIRT, and PROF times.

Instruction set for a RISC (e.g. mips 2000) processor; instruction formats and special instructions (e.g. for support of spinlocks).

Memory design; discussion of the L2 cache and why it is needed; set-associative caches and cache performance; introduction to multi-threaded code.

Some discussion of operating system support and interface to the hardware, exceptions, signals, and system calls.

Algorithms for integer and floating point arithmetic.

Functional units and pipelining; the L1 cache; design of a 5-stage integer pipeline; hazards and forwarding; exception handling.

More on caches and virtual memory design; the TLB cache, memory error detection and correction.

More on I/O; buffering of I/O; I/O modes: polling, interrupt-driven I/O, interrupt-driven I/O with DMA.

**Laboratory:**

The laboratory session will parallel the lecture, illustrating the principles. We will often write programs which simulate various features of and considerations in computer architecture. I will provide a library of timing routines for profiling programs.

**Grading:**

Two midterms will be given, each worth 25%. I do not give make-up midterms; for an excused absence I count the other grades proportionately higher. One final exam, comprehensive but emphasizing the later material, will be given. It is mandatory and worth 25%. Homework and lab work are together worth the remaining 25%.