CS2200 Introduction to Systems and Networking

Purpose & Outcomes

Purpose

Provide a broad exposure to computer system structure and networking including software abstractions in operating systems for orchestrating the usage of the computing resources:

- Organization of the processor
- Memory hierarchy
- Storage devices
- Parallel processors
- Networking hardware
- Software abstractions in the operating systems for orchestrating their usage
- Networking protocols to connect the computer system to its environment

Outcomes

- (Competency Knowledge) Understand the difference between RISC and CISC architectures. Be able to identify the strengths and weaknesses of each paradigm.
- (Competency Knowledge) Understand and be able explain runtime system concepts such as procedure calls and register saving. Be able to write recursive subroutines in assembly.
- (Competency Application) Understand how a processor is controlled. Given a datapath and an instruction set be able to write the finite state machine steps in a high-level meta language.
- (Competency Knowledge) Understand and be able to explain (at a high level) hardware modifications required to implement an interrupt system and to understand the basic concepts required to write an interrupt handler (in assembly language).
- (Competency Knowledge) Understand the basic principles of pipelining:
- Pipelining registers
- Potential performance improvements with pipelining
- Pipelining Hazards: Structural, Data and Control
- (Competency Knowledge) Understand basic concepts of processor scheduling: Process vs program, PCB, scheduling algorithms (Round Robin, Shortest Job First, First Come First Served, Priority, Multilevel Queues), types of scheduler (short, medium and long term) and context switching.
- (Competency Comprehension) Given a set of processes with appropriate parameters show scheduling behavior under different scheduling algorithms.
- (Competency Application) Be able to calculate the proper size required for pipeline register and speedups with pipelining.
- (Competency Application) Be able to solve basic word problems involving Amdahl's Law.

- (Competency Knowledge) Be able to identify and explain how to avoid or minimize the effect of the different types of pipelining hazards.
- (Competency Knowledge) Understand the drivers of memory cache designs: Temporal locality, spatial locality and working set. Be able to match the design with the motivator.
- (Competency Knowledge) Understand the basic operation of virtual memory and typical components: Page table, virtual pages, physical frames, TLB, page/frame offset, page replacement algorithms (LRU, Random, FCFS, and Optimum). Be able to describe the basic operation and identify the necessary subsystems.
- (Competency Knowledge) Understand the basic design of typical caches including indexes, tags, dirty and valid bits as well as multi-word blocks, set-associative and ully associative caches. Given selected design parameters (i.e. word size, memory available for data
- (Competency Knowledge) Understand basic concepts of parallel processing: UMA (SMP) vs NUMA configurations, multiprocessor cache coherency, network interconnection schemes, threads, mutex, condition variables.
- (Accomplishment Application) Be able to write multi-threaded programs using the pthreads package. An example would be a multithreaded producer consumer application.
- (Competency Knowledge) Understand basic networking concepts: Ethernet (CSMA/CD), Token Ring, Payload vs. header and trailer, checksums, bandwidth, effective bandwidth, latency, MAC addresses, Network (IP) addresses, protocol stacks, TCP/IP, routing, hubs/repeaters, bridges, VLANS, routers.
- (Competency Knowledge) Understand fundamentals of I/O devices such as polling versus interrupts, memory mapped I/O, device registers (data, control and status), disk memory concepts (sectors, tracks, platters, cylinders, seek time, rotational latency), disk scheduling algorithms (FCFS, SSTF, scan, c-scan, look, c-look)
- (Accomplishment Synthesis) Write and debug medium sized C programs that simulate various of the above subsystems (interrupt enabled processor, virtual memory, multi-threaded operating system schedulers, reliable transport layer protocol which will be examples of operating-system-like coding.

15 week Semester Lecture schedule (with reference to textbook):

- (Arch) Processor design and implementation – 3 weeks (Ch 2-5)
- (OS) Processor scheduling – 1 week (Ch 6)
- (OS/Arch) Memory management including memory hierarchy – 3 weeks (Ch 7-9)
- (OS/Arch) Multiprocessor and multithreaded programming – 2 weeks (Ch 12)
- (Arch/OS) I/O: Interfacing I/O devices, interrupts, handlers, drivers, disk scheduling – 1 week (Ch 10)
- (OS) File systems – 2 weeks (Ch 11)
- (OS/Arch) Networking – 2 weeks (Ch 13)
- (Misc) tests, review – 1 week

Project schedule (with assistance in recitation meetings – 2 hours of scheduled time every week):

(Arch) Processor implementation
(Arch) Interrupt implementation and interrupt handler
(OS) Virtual Memory management
(OS) Multithreaded processor scheduler
(OS/networking) Reliable transport protocol