CS 402 Introduction to Advanced Studies II

This is a second course in a two-course sequence that is designed to prepare students for graduate study in computer science. Topics in this course cover two undergraduate courses: cs350 (Computer Organization and Assembly Language Programming) and cs351 (Systems programming). The course provides Introduction to the internal architecture of computer systems-including micro-, mini-, and mainframe computer architectures. Focuses on the relationship among a computer's hardware, its native instruction set, and the implementation of high-level languages on that machine. In terms of systems programming, the course examines the components of sophisticated multilayer software systems, including device drivers, systems software, applications interfaces, and user interfaces. It explores the design and development of interrupt-driven and event-driven software. The programming languages used in this course are Assembly and C. This course does not apply toward master's or Ph.D. credit in Computer Science. Prerequisite: CS 401. 2-2-3

CS 350 Topics

1. Data representation: Binary, octal, hex, char, strings, floating-point, Boolean logic, bit operations. (5 hours)
2. Intermediate C: Representation of pointers, records, and arrays (C pointers, structures, and arrays). (3 hours)
3. Machine-level execution: Basic architecture, instruction cycle, instruction set architecture of a small machine. (3.5 hours)
4. Low-level programming: Assembler programming, subroutines, run-time stack. (3 hours)
5. Updated ISA and basic system-level structures: I/O, Interrupts, Caches, Instruction Pipeline, RISC/CISC architectures, processes, threads, multiprogramming, multiprocessing. (2.5 hours)
6. Basic hardware and computation: Switches, Transistors, Logic Gates, Combinatorial circuits. (3 hours)
7. Computation with memory: Storage Elements, Memory, Sequential Logic Circuits, maybe Finite State Machines. (2.5 hours)

CS 351 Topics

1. Advanced C Topics: pointers (with function pointers), memory management, array/structure memory layout. (3 hours)
2. Process abstractions implemented by the user/kernel: including virtual memory, logical control flow, exceptions (e.g., faults, traps, aborts, interrupts) and uniform I/O. (1.5 hours)
3. Process management and exceptional control flow: fork, wait, exec, kill, signal system calls, the UNIX process model, system call (trap) and signal facilities, and concurrency. Machine problem: implementing a shell. (4.5 hours)
4. The Memory Hierarchy: starting with the hardware cache and moving up to the OS virtual memory (segmentation and paging) facilities. Discussion motivated by the effect of system programming patterns on memory throughput and utilization. Machine problem: implementing a cache simulator. (4.5 hours)
5. Dynamic Memory Allocation and Garbage Collection (Explicit) DMA implementation strategies, metrics, and pros/cons. Basic garbage collection strategies (e.g., navigating a memory graph and tracing and ephemeral G.C.s). Machine problem: implementing malloc/free. (4.5 hours)
6. System-level I/O and basic IPC: A discussion of the UNIX I/O architecture, covering v-nodes,

open-file descriptions, per-process file tables and file descriptors. API: open, read, write, lseek, close, dup/dup2, pipe, and shared memory syscalls. (4.5 hours)

Total: 45 hours

Textbooks:

1. CS 350 topics: The Essentials of Computer Organization and Architecture, Linda Null and Julia Lobur, Jones and Bartlett Learning.
2. CS 351 topics: Randal Bryant and David O'Hallaron, Computer Systems: A Programmer's Perspective, 3rd Ed., Pearson, 2016. ISBN 013409266X.