# Master Course Syllabus

School of Engineering and Computer Science
Washington State University Vancouver

# CS 460

## Operating Systems and Computer Architecture

3 Semester Hours

### Catalog Description

Operating systems, computer architectures, and their interrelationships in micro, mini and large computer systems.

### Prerequisite Courses

CS 360 – Systems Programming

### Prerequisite Topics

- Proficiency with the C programming language
- Use of Unix environment for coding, compilation, debugging and testing
- Use of Unix operating system API, particularly low level I/O and file system access
- Concepts of Unix file system structure
- Assembly language programming

### Measured Course Outcomes

Students taking this course will:

1. Construct a bootstrap program of their own design, install it on a disk and execute it *(contributes to performance criteria* **B-2** *and* **E-2***)*.
2. Design and implement a simple operating system kernel capable of managing at least one process, which performs I/O and/or communicates with the user *(contributes to performance criteria* **B-2** *and* **K-4***)*.
3. Write a program which correctly sets up a processor's virtual memory tables then loads and executes a second process in a virtual memory environment *(contributes to performance criterion* **B-3***)*.
4. Contrast the advantages and disadvantages of RISC and CISC instruction set architectures *(contributes to performance criterion* **B-3** *and* **J-2***).*

### Required Textbooks

Andrew S. Tanenbaum, Operating Systems, Design and Implementation, Prentice Hall, 2nd Edition, 1997, ISBN 0-13-638677-6.

or

D. Irtegov, Operating Systems Fundamentals, Charles River Media, 2003, ISBN 1-58450-274-6

## Reference Material

Tom Shanley, Protected Mode Software Architecture, Addison-Wesley 1996, ISBN 0-201-55447-X.

James L. Antonakos, An Introduction to the Intel Family of Microprocessors, 3rd Edition, Prentice Hall, 1999, ISBN 0-13-893439-8.

## Major Topics Covered in the Course

1. Overview and history of computer operating systems
2. Operating system abstractions (processes, resources, APIs)
3. Instruction set architectures, CISC vs RISC
4. Program loading and execution
5. Memory management and virtual memory
6. Multi-tasking, scheduling and multi-processing
7. Synchronization
8. I/O architecture
9. Device management and drivers
10. File systems and disk I/O
11. Security issues and mechanisms

## Laboratory Projects

| Programming Project Area | Weeks |
|---|---|
| Boot loader program | 2 |
| Bare machine program loading and execution | 1 |
| Customized file system | 3 |
| Interrupt handler and/or device driver | 2 |
| Virtual memory management and privileged mode execution | 1 |
| Single or Multi-tasking program loading and execution | 4 |

## CSAB Category Content

| | FUNDAMENTAL | ADVANCED | | FUNDAMENTAL | ADVANCED |
|---|---|---|---|---|---|
| Data Structures | 0 | 0 | Computer Organization and Architecture | 0 | 2 |
| Algorithm & Software Design | 0 | 1 | Concepts of Programming Languages | 0 | 0 |

## Oral and Written Communications

There are no significant oral or written communications required in this course. Virtually all assignments consist of writing computer programs or solving problems.

## Social and Ethical Issues

This course contains no significant coverage of social and ethical issues.

## Theoretical Content

| Topic | Hours |
|---|---|
| Cache & virtual memory hit ratios, theoretical performance | 2 |

## Problem Analysis

Students are given 5-6 programming assignments. The requirements for these assignments are outlined by the instructor but modified by the students. It is the student's responsibility to apply lecture material and understand the implications of the requirements upon their program design.

## Solution Design

Students use resources describing the machine architecture in addition to lecture material to design their own solutions to each programming assignment. The 5-6 assignments involve considerable assembly language programming and require a detailed understanding of the hardware architecture as it relates to support of operating system features. The final assignment has each student designing, implementing and debugging a miniature operating system kernel which can load and execute small user programs. This final assignment usually requires ~2000 lines of assembly language code.

## CC2001

This course provides coverage of topics in the following areas (hours listed are minimums):

| | |
|---|---|
| OS1. Overview of operating systems [core] | 2 |
| OS2. Operating systems principles [core] | 2 |
| OS3. Concurrency [core] | 2 |
| OS4. Scheduling and dispatch [core] | 3 |
| OS5. Memory management [core] | 5 |
| OS6. Device management [elective] | 4 |
| OS7. Security and protection [elective] | 4 |
| OS8. File systems [elective] | 4 |

| | | |
|---|---|---|
| Course Coordinator: | Dick Lang | |
| Last Updated: | October 7, 2005 | (Approved) |
| Syllabus Version Number: | 1.2 | |