**CSE 351 The Hardware/Software Interface**

## Credits
4.0 (3 hrs lecture, 1 hr section)

## Lead Instructor
Luis Ceze

## Textbook
*Computer Systems*, Bryant & O'Hallaron
*C Programming Language*, Kernighan & Ritchie

## Course Description
Examines key computational abstraction levels below modern high-level languages; number representation, assembly language, introduction to C, memory management, the operating-system process model, high-level machine architecture including the memory hierarchy, and how high-level languages are implemented.

## Prerequisites
CSE 143.

## CE Major Status
Required

## Course Objectives
At the end of this course, students should:

- understand the multi-step process by which a high-level program becomes a stream of instructions executed by a processor;
- know what a pointer is and how to use it in manipulating complex data structures;
- be facile enough with assembly programming (X86) to write simple pieces of code and understand how it maps to high-level languages (and vice-versa);
- understand the basic organization and parameters of memory hierarchy and its importance for system performance;
- be able to explain the role of an operating system;
- know how Java fundamentally differs from C;
- grasp what parallelism is and why it is important at the system level; and
- be more effective programmers (more efficient at finding bugs, improved intuition about system performance).

## ABET Outcomes

(6) an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions
(7) an ability to acquire and apply new knowledge as needed, using appropriate learning strategies

## Course Topics

Number representation: Two's complement, signed vs. unsigned, floating point (1 week)
Assembly (2 weeks)
    Memory vs. registers
Instruction format
Control structures in assembly (loops, procedure calls)
C (2 weeks)
    Pointers, arrays, strings
Memory management, malloc/free, stack vs. heap
structs
Compilation, linking, libraries (code across multiple files) (0.5 weeks)
The process model (what the operating system provides, not how it provides it) (1 week)
    Virtualization and isolation (including virtual memory)
Components of a process state and notion of a context switch
System calls for accessing shared resources and communication channels
Asynchronous signals
High-level machine architecture (2 weeks)
    Register file
Instruction cycle
Caching and the memory hierarchy
The Java-to-C connection (1 week)
    Representing an object as pointer to struct with pointer to method-table; performing a method call
Constructors as malloc-then-initialize
Garbage collection via reachability from the stack
Java array-bounds-checking via array-size fields
Parallelism/multicore/pthreads (0.5 weeks)