UW/CSE Hardware / Software Interface

Course Description

Draft of April 26, 2009

Structural place in the curriculum

- 4 credits (3 weekly lectures, 1 weekly section, no lab)
- Pre-requisites: 143
- Subsequent courses: The following courses would have this course as a pre-requisite: Hardware Design & Implementation, Operating Systems, Computer Architecture, Compilers, Embedded Systems, etc. The transition/overlap with Hardware Design & Implementation needs to be handled particularly carefully to avoid redundancy or gaps. It is not yet decided if Systems Programming should have this course as a pre-requisite; it probably should so that Systems Programming does not have to start C from scratch.
- Taken by: All students (CS and CE)
- Catalog description: To be determined

Course Overview / Goals

This course should develop students' sense of "what really happens" when software runs — and that this question can be answered at several levels of abstraction, including the hardware architecture level, the assembly level, the C programming level and the Java programming level. The core around which the course is built is C, assembly, and low-level data representation, but this is connected to higher levels (roughly how basic Java could be implemented), lower levels (the general structure of a processor), and the role of the operating system (but not how the O/S is implemented).

For (computer-science) students wanting to specialize at higher levels of abstraction, this could in the extreme be the only course they take that considers the "C level" and below. However, most will take a subset of Systems Programming, Hardware Design & Implementation, Operating Systems, Compilers, etc.

For students interested in hardware, embedded systems, computer engineering, computer architecture, etc., this course is the introductory course after which other courses will delve both deeper (into specific topics) and lower (into hardware implementation, circuit design, etc.).

Description of Possible Homework, Etc.

The course would probably have a mix of homeworks, projects (essentially larger homeworks), and exams. In other words, it would not have a lab component. Homeworks would involve C programming (though largely to understand the machine level of abstraction, not to write a non-trivial C application), assembly programming, and homework exercises (e.g., on topics like number representation or memory hierarchies).

Possible Textbook

The texts would almost surely be:

- Computer Organization and Design by Patterson and Hennessy
- A reference on C, probably instructor preference. Options include Harbison/Steele, Kochan, and Kernighan/Ritchie

These texts would not cover the week on Java to C or some of the material on the role of an operating system; auxiliary course notes should bridge this gap.

Approximate Topic List

Note that even more important than the topics at various levels of abstraction is the connection between them: Students should get an informal sense of how Java could be translated to C, C to assembly, and assembly to binary.

Weeks are approximate; they are particularly useful for identifying topics that are essential for "connecting the pieces" yet which do not command a large portion of the class (e.g., Java-to-C).
The order of presentation would depend on instructor preference. There are arguments for "top down" (they already know Java), for "bottom up" (understand what's going on below), or starting with C (useful for homeworks).

- Number representation: Twos complement, signed vs. unsigned, floating point (1 week)
- Assembly (2 weeks)
  - Memory vs. registers
  - Instruction format
  - Control structures in assembly (loops, procedure call)
- C (2 weeks)
  - Pointers, arrays, strings
  - memory management, malloc/free, stack vs. heap
  - structs
- Compilation, linking, libraries (code across multiple files) (< 1 week)
- The process model (what the O/S provides, not how it provides it) (1 week)
  - Virtualization and isolation (including virtual memory)
  - Components of a process state / notion of a context switch
  - System calls for accessing shared resources / communication channels
  - Asynchronous signals
- High-level machine architecture (1 week)
  - Register file
  - Instruction cycle
  - Caching / 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 (1 week)

Approximate Non-Topics
These are topics we feel are "close" to the course, but which there is likely not time for. If our time estimates are wrong or some topics above are deemed unimportant, they would be fine additions.

- Instruction-cycle details (fetch, decode, ...)
- Pipelining

Background / Correspondence to Old Curriculum
The material on assembly, machine language, and CPU organization is currently in 378. The material on number representation is currently in 370. This course would be many students' first exposure to C, where 303 currently has that distinction. However, competent C programming is covered in a separate course Systems Programming. The connection up to higher-level languages like Java is often lacking in our current curriculum. The role of an operating system is perhaps touched on in 378, but is otherwise currently seen only in 451.

Similar Courses Elsewhere
This course is perhaps most like Berkeley's 61C:

- http://www.eecs.berkeley.edu/Courses/Data/188.html
- http://inst.eecs.berkeley.edu/~cs61c/archives.html

Compared to 61C, this course (1) is a quarter instead of a semester, (2) does not go below high-level architecture issues (see hardware design/implementation for circuits, latches, etc.), (3) does connect higher up the software stack to a managed language like Java, (4) replaces some material on disks and virtual memory with material on the process model and the role of an operating system, (5) does not have a lab. (61C has a "check-off" lab where many weeks are spent writing C or assembly programs.)

Another similar course is Cornell's 3410:
http://www.cs.cornell.edu/courses/cs3410/2008fa/