1. **Number and title of course:** CS 152, Computer Architecture and Engineering

2. **Catalog description:** (5 units) Three hours of lecture, one hour of discussion per week and 3 hours of lab per week. This course provides a comprehensive overview of the instruction set architecture and microarchitectural design of modern computers. Microprogramming; pipelining and hazards; caches and memory hierarchies; virtual memory; speculative and out-of-order execution; VLIW processors; vector processors; multithreaded processors; multiprocessor issues including synchronization and memory models; and multiprocessor cache coherency protocols. The course emphasizes the interaction of hardware and software, and the impact of technology advancements and application requirements on computer architecture. Architectural design issues are explored in a sequence of simulation-based lab exercises, including open-ended design problems.

3. **Prerequisites:** CS61C

4. **Textbooks and/or other required/recommended material:**

5. **Course objectives:** CS152 is an elective lab-centric course that is intended to provide a foundation for students interested in performance programming, compilers, and operating systems, as well as computer architecture and engineering. Students should develop an understanding of how software interacts with hardware, and how trends in technology, applications, and economics drive continuing changes in the field. The course covers different forms of parallelism found in applications (instruction-level, data-level, thread-level, gate-level) and how these can be exploited with various architectural features. The course also explores techniques that take advantage of the predictability in real applications to accelerate execution or simplify hardware.

   The course requires students to complete a series of lab assignments using a variety of computer architectures, and provides hands-on experience with the interaction of software and hardware, for a variety of machine designs. Extensive use will be made of the Virtutech Simics full-system machine simulator. Each lab includes a directed component to guide students in learning certain concepts, plus an open-ended assignment to allow students to show their creativity.

6. **Topics covered:**
   - ISA design
   - Introduction to Simics simulator
   - Simple machine implementations
   - From CISC to RISC
   - Pipelining
   - Memory hierarchy
   - Address translation and protection
   - Virtual memory
   - Virtual memory and caches
   - Complex pipelines
• Out-of-order issue; Register renaming; Branch prediction
• Advanced out-of-order superscalars
• Out-of-order memory accesses
• VLIW machines
• Multithreading
• Synchronization and sequential consistency
• Snoopy caches
• Directory protocols
• Virtual machines

7. **Class/laboratory schedule:** Three hours of lecture and one hour of discussion per week and labs.

8. **Contribution of course meeting the professional component:** This class contributes to computer science and engineering topics. It is approximately 50% science and 50% hands-on laboratory experience. The lab assignments contain both an analysis component, where students learn how various architectural features interact with software behavior, and an open-ended design component, where students are encouraged to explore the effectiveness of their own architectural design ideas.

9. **Relationship of course to program outcomes:**
   - #1: An ability to run architectural experiments by configuring architectural simulators, running application workloads and interpreting simulation results.
   - #2: An ability to design architecture components to meet specifications in terms of cost/performance constraints during the directed portions of the labs.
   - #3: An ability to work cooperatively in a team while performing lab exercises and open-ended design space explorations.
   - #4: An ability to identify, formulate, and solve engineering design problems in the open-ended portions of the lab exercises.
   - #6 An ability to communicate results of lab design-space explorations using lab reports.
   - #10 An in-depth ability to trade off hardware versus software components in the design of a computing system.

10. **Prepared by:** Professor Krste Asanović (Spring/2009)