CSC 271 Introduction to Computer Systems 4 cr.

Catalog description:
The important concepts of a complete large computer system are introduced, including the machine and assembly language, the low level use of a standard environment and its implementation language. They are firmly interlinked by studying the methodologies of low level program control and debugging of assembler and implementation language code under the standard environment, the assembly code generated by compilers, and the overall environment. Three lecture hours and three hours of scheduled laboratory per week, plus extensive laboratory work outside of class.

Prerequisite: CSC 260 with grade of C+ or higher.

Goals:
The goals of this course are to provide a technical bridge between the programming and problem solving goals of CSC 201, 202 and 260, which emphasize problem solving skills, and the explicit knowledge of the environment in which real problems are solved.

This provides a basis for advanced courses, which more and more will use standard environments because of the wide availability of course materials for them. If taken concurrently with CSC 260, this course allows a simultaneous glimpse of efficiency in the compiled versions of the Data Structures algorithms. It also provides a concrete preparation for CSC 280.

CG01: understand the basic features of a system such as Unix or Linux;
CG02: understand the methodology used to create a program on such a system;
CG03: be familiar with the assembly language of a high performance RISC processor such as SPARC or MIPS;
CG04: understand how the efficiency of high level languages relates to an assembly language;
CG05: be familiar with a language used for system programming or scripting such as C or Perl.

Objectives:
Upon successful completion of the course, a student will have:

CO01: used the operating system presented as the base for several programs successfully designed, implemented and debugged;
CO02: designed, implemented and debugged several programs in assembler;
CO03: examined the code produced by a compiler or performance of an interpreter;
CO04: written at least one program in a scripting language.

<table>
<thead>
<tr>
<th>PO1</th>
<th>PO2</th>
<th>PO3</th>
<th>PO4</th>
<th>PO5</th>
<th>PO6</th>
<th>PO7</th>
<th>PO8</th>
<th>PO9</th>
<th>PO10</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Topics:
- Introduction to computer organization
- Review of number systems and encoding
- Processor programming model
- Mechanics of programming under a standard operating system
- Processor control flow
- Linkage conventions
- Delay slots and performance issues
- Arithmetic instructions
- Bits and bytes
- Implementation language and compiler usage of the instruction set
- Standard libraries
- Standard preprocessors
- Basic program management and structure

AR4(1.5) AR2(1.5) AR4(1.5) OS2(3) AR3(2) AR3(3) AR3(1) AR8(1) (not core) AR3(2) AR2(1.5), AR3(1) PL3(2) PL10(3), AR8(2), OS11(3), OS12(9) OS8(2) (not core)

Typical assignments: a small binary program hand compiled from assembler, several small assembler programs, several small implementation language programs including a sparse matrix abstract data type, benchmarking of the implementation language, and hand-tuned assembler versions of an enumerative puzzle solver.
All programs must conform to departmental guidelines for program design and implementation, and all laboratory reports must conform to the written guidelines supplied by the instructor. Regardless of numeric average or grades on individual assignments or examinations, a student will not be eligible for a passing grade in the course unless he or she has submitted a lab report for every programming assignment within the time-frame specified by the instructor.

The course grade will be determined using the following approximate weights: lab reports - 40%, final exam - 25%, other tests and homework - 35%.

<table>
<thead>
<tr>
<th></th>
<th>Lab Assignments</th>
<th>Programming Assignments</th>
<th>Exams</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO01</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CO02</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO03</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO04</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Bibliography:
Kane & Heinrich. *MIPS RISC Architecture* (Prentice Hall, 1992)
Ramteke, Timothy: *Unix by Experimentation* (Prentice Hall, 1999)