Detailed Syllabus for TCSS 371 Machine Organization
in the New Curriculum
(revised September 2009)

Catalog Description:
Develops the hardware basis for computing systems, and the relationship between hardware and software. Covers number representations, digital logic, machine organization, instruction set architecture and assembly language. Includes an introduction to high-level languages and the translation of such languages into machine instructions. Prerequisite: TCSS 143.

Purpose:
TCSS371 provides an introduction to computer hardware concepts along with an understanding of the relationship between hardware and software. An early understanding of these concepts will help the student in more advanced programming courses. This course is the prerequisite for Computer Architecture in which various concepts of machine and software optimization are learned.

Educational Objectives
- Knowledge of fundamental computer organization (CPU, Memory and I/O) and Instruction Set Architecture.
- Knowledge of digital logic related to implementation of the above.
- Understanding of the basic machine cycle and computational operations.
- Ability to write small programs and function modules in assembly language.
- Ability to write small to medium programs in C.

Educational Outcomes
Upon successful completion of this course, a student should be able to:
- explain how data and information is stored in a binary computer,
- generate simple circuits from Boolean expressions and truth tables,
- explain the types of instructions that make up a typical instruction set,
- give examples of the various addressing modes included in an instruction set,
- describe the instruction execution cycle in detail,
- explain the roles of various components of a computer system and how data flows through that system,
- write simple assembly language programs and function modules,
- explain how external I/O devices are accessed,
- explain how interrupts are used in a computer,
- write an assembly language program that utilizes a stack,
- write a simple, multi-function program in C that uses pointers and that uses the standard library functions for I/O
- explain how a high-level imperative language is mapped to the assembly level

Topics covered (not necessarily in the order given)
1. Machine representation of numbers and non-numeric data
    a. Binary representations of fixed precision whole numbers

    b. Binary arithmetic
    c. ASCII, Unicode – representation of characters
    d. General data types
2. Digital logic
    a. Gates
    b. Combinational circuits
        i. Digital logic
        ii. Boolean algebra applied to circuits
        iii. Representative circuits (multiplexers, decoders, adders, etc.)
        iv. Arithmetic-logic unit
3. Computer organization
    a. CPU internal (registers, bus, ALU, flags, etc.)
    b. Main memory and memory maps
    c. Busses
    d. Simple I/O & Interrupts
    e. External storage
4. Instruction Set Architecture
    a. ISA of a typical 8/16 bit CPU
        i. data movement instructions
        ii. arithmetic/logic instructions
        iii. flow of control instructions (unconditional and conditional branches)
        iv. I/O (separate I/O space vs. memory mapped I/O)
        v. control instructions
        vi. introduction to interrupts and traps
        vii. addressing modes
    b. machine cycle (fetch, decode, execute)
        i. simulation of a simple machine
    c. examples of numeric and non-numeric computations
5. Assembly language and machine translation [NOTE: this subject may be introduced earlier in the quarter in order to allow time for students to write actual assembly language programs.]
    a. introduction to assembly process
    b. the assembly development tools
    c. machine translation from assembly statements into machine code
    d. modular assembly, linking and creation of executable programs
6. Mapping a high-level imperative language to the assembly level (C will be used as the imperative language):
    a. conditionals and loops
    b. function calls
    c. primitive data
    d. non-primitive data: pointers, arrays, structs
    e. memory management
    f. inline assembly instructions
7. High-level language access to hardware
    a. Using standard library functions to perform I/O