**CMPS 293 – Introduction to Assembly Language**

**Last Updated:** 8/26/2008

**Current Course Description:** Credit 3 hours. Prerequisite: Computer Science 280. Fundamentals of assembly language programming. Topics include machine representation of data, fix point, floating point, and decimal arithmetic, macros, address modification, bit manipulation, and subroutine linkage.

**Minimum Topics:**
- Basic concepts: virtual machine; data presentation; Boolean operation
- IA-32 Processor Architecture: basic Microcomputer design; instruction execution cycle; memory management; components of IA-32 microcomputer; input-output system
- Assembly language fundamentals: basic elements of assembly language; assembling, linking, and running programs; defining data; symbolic constants
- Data transfer, addressing, and arithmetic: data transfer instructions; addition and subtraction, data-related operators and directives; indirect addressing; JMP and Loop instructions; shift and rotation instructions
- Procedures: linking to an external library; stack operations; defining and using procedures
- Conditional processing: Boolean and comparison instructions; conditional jumps; conditional loop instructions; conditional structures
- Macros

**Learning Objectives:** Students will be able to:
- Demonstrate understanding of the relationship between programming statements and the fundamental capabilities of a central processor
- Demonstrate understanding of the challenges faced by compilers for high-level languages
- Demonstrate ability to distinguish areas where assembly language is most useful, and conversely, where it is least useful
- Demonstrate understanding of some basic principles of computer organization and architecture
- Demonstrate understanding of how data is represented, stored, and managed (including addressing) at a very basic level
- Demonstrate ability to use an assembly language to create working programs on an actual computer

**Relevance to Program Learning Outcomes and Evaluation:**

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline
   **Justification:** Course requires application of computing on binary numbers and to solving equations on assembly language, etc.
   **Measured by:** assignments/programs, tests, and quizzes throughout semester

b. An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs
   **Justification:** Course requires analysis of problems and subsequent development of computational procedures to solve the problems.
   **Measured by:** assignments/programs throughout semester

c. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
Justification: Course requires analysis and development of computational processes to solve specific problems and meet well specified computational targets.

Measured by: assignments throughout semester and tests

i. An ability to use current techniques, skills, and tools necessary for computing practice

Justification: Course provides critical understanding about computation at the machine level and its relation to many general purpose problem formulations which are necessary for computing practice

Measured by: several assignments/programs and tests throughout semester

Units Covered:
DS2 Basic logic (2/10)
PF1 Fundamental programming constructs (9/9)
PF2 Algorithms and problem-solving (2/6)
PF3 Fundamental data structures (1/14)
AR1 Digital logic and digital systems (1/6)
AR2 Machine level representation of data (3/3)
AR3 Assembly level machine organization (6/9)
AR4 Memory system Organization and architectures (2/5)
AR5 Interfacing and communication (1/3)
PL3 Introduction to language translation (1/2)
PL4 Declarations and types (1/3)
SE1 Software design (2/8)
SE3 Software tools and environments (1/3)