William Paterson University of New Jersey
Department of Computer Science
College of Science and Health
Course Outline

I. Course Title: CS 280 Computer and Assembler Language

3 credits

II. Course Description:

Number systems. Structure of digital computers and machine language. Data representation, instruction formats and addressing techniques. Symbolic coding, assembly systems and programming techniques. System stack, procedure calls, and program segmentation and linkage. Interrupts and I/O. Memory organization and addressing. Program translation and system programs: Assemblers, compilers, interpreters, preprocessors, linkers, loaders and debuggers.

III. Prerequisites: CS 230 with a grade of C- or better.

IV. Course Objectives:

To examine computer systems as machines and to show how the various structures and abstractions of computing are realized. This implies the following:

1. The study of the structure and capabilities of computers.
2. An understanding of data representation, instruction representation and execution, program translation, subprogram linkage and communication, and data structure implementation.
3. An understanding of various Input/Output operations, and the functions of system programs.
4. An understanding of the connections between hardware and assembly language, and those between high-level languages and assembly language.

V. Learning Outcomes :

After the completion of this course, a successful student will be able to do the following:

1) Describe the major classes of programming languages with their characteristics.
2) Perform conversions between binary, octal/hexadecimal, and decimal number systems and perform the basic arithmetic operations in these number systems.
3) Describe the assembly level machine organization including: the main memory, the CPU, the I/O systems, the system interconnection, the secondary storage, the

clock, and interrupts with their function.

4) Explain memory organization, memory segmentation and boundary alignments.

5) Represent an absolute address in selector/offset form and compute an absolute address from a selector/offset address.

6) Represent different types of data inside the computer: characters (in ASCII), integers (in two's complement and sign magnitude), decimal number (in BCD) and floating-point values (in binary). Perform addition and subtraction of two's complement binary integers, and recognize overflow conditions. Also perform conversion between 8-bit, 16-bit, and 32-bit two's complement binary integers.

7) Know how to use the services (routines) that the operating system (DOS, ROM-BIOS) provides to other programs to access the hardware.

8) Explain memory utilization under DOS, and the difference between .com programs and .exe programs.

9) Specify an instruction in assembly language, and in machine language, and be able to translate a machine language instruction into assembly language and vice versa.

10) Write and execute (trace) a Debug program to perform a two's complement binary integer arithmetic expression, and know how to interpret the result(s).

11) Use loop instructions to write and execute (trace) Debug programs with simple iteration executions, including the processing of arrays of two's complement binary integers. Arrays will be processed using both pointers and base-displacement addressing.

12) Use the compare and conditional/unconditional jump instructions to implement the high-level language structures if, if/else, and loop. Also write and execute a Debug program with one or more of these structures.

13) Write an assembly language program (program segments definition and symbolic addressing) and explain the assembly process of a source module.

14) Assemble, link, and execute an assembly language program (.exe program) in Debug environment (using the listing and the map files).

15) Write programs with calls to near/far procedures.

16) Use the stack to pass arguments to procedures.

17) Write and execute (in the Debug environment) assembly language programs with external procedures. Must also know how to link an assembly language program with a high-level language program.

18) Perform I/O operations using DOS services.

The course will also reinforce the following students learning outcomes of the university:

a) Effectively express themselves in written and oral form. Measure: exams, surveys, and projects.

b) Demonstrate ability to think critically. Measure: exams, surveys, and projects.

c) Locate and use information. Measure: projects.

d) Demonstrate ability to integrate knowledge and idea in a coherent and meaningful manner. Measure: exams, surveys, and projects.

e) Work effectively with others. Measure: projects.

VI. Course Contents:

1. Introduction
   - Major classes of programming languages
   S Number systems: binary, Hexadecimal/Octal number systems; conversion between decimal, binary, and hexadecimal/octal number systems; arithmetic operations on binary and hexadecimal/octal number systems.

2. Digital computer system hardware
   S Main Memory: addressing, boundary alignments, and organization; RAM, ROM.
   S CPU: CU, ALU and registers (8086 registers)
   S I/O System: I/O devices, controllers, communication between the CPU and controllers (memory-mapped I/O and I/O addressing).
   S Secondary storage: drives and controllers.
   S Clock
   S Interrupts: hardware and software interrupts, 8086 interrupts.
3. Data representation
   S Representation of characters (EBCDIC, ASCII, Extended ASCII)
   S Numeric data
      . binary integers: signed-magnitude, two=s complement (addition, subtraction, and overflow conditions)
      . Binary-coded decimal
      . Floating-point numbers
4. Software Environment
   S DOS and ROM-BIOS
   S Program files, batch files, and device drivers
   S Computer initialization
   S Input-Output interface: DOS and BIOS services, and interrupts instructions.
   S Memory utilization under DOS and DEBUG utility program.
5. Machine language instruction formats and assembly language statements.
6. Data movement and arithmetic instructions on binary integers.
7. Relative addressing and iteration
8. Comparing, conditional and unconditional jump instructions.
9. Arrays: Addressing and operations; multi-dimensional arrays.
10. Assembly language programming
    . General syntax of an assembly language statement