CS 251 – Computer Organization and Assembly Language
Fall 2017

Instructor: Justin Miller
Class: Halsey 237 (in classroom)
Email: millerju@uwosh.edu

Class information: 5:00 PM to 8:00 PM Mondays
Office hours: 8:00 PM to 9:00 PM Mondays (in classroom)

Prerequisites: CS 221 with a grade of C or better.

Course website: http://www.uwosh.edu/d2l. You should check d2l on a regular basis as it will contain any lecture notes, handouts, assignments, announcements, and grades. I'll do my best to let you know when something new and important comes up, but it is your responsibility to check the web site frequently for information that you might not get otherwise.

Required textbook: NONE

Note: If you have special needs, please come and talk to me at the end of the first class.

Course description: This course aims to give students an overview of processor and memory hardware, and to teach them how high-level language programs map into Assembly Language. Students will learn how computer hardware supports the instruction set architecture. Students will be able to analyze why programs behave the way they do. Students will learn how to implement pointers and references in machine language.

Course grade: Your final course grade will be based on the following components:

Component	Weight
Exams (2)	60%
Assignments (10)	20%
Readings (10)	10%
Quizes (10)	10%

Grading will be on a plus/minus system. Grading *may* be done on a curve depending on the overall performance of the class. If no curve is used, your grade will be computed based on the following:

Your letter grade for the course will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
>=92	A	72-78	C
90-92	A-	70-72	C-

88-90	B+	68-70	D+
82-88	B	62-68	D
80-82	B-	60-62	D-
78-80	C+	<60	F

Exams: Exam material will come from the lecture notes, quizzes and assignments. There will be more information about each exam as it approaches.

If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do both of the following, which are then subject to my approval:

1. Make arrangements prior to the scheduled exam (for last minute emergencies, call my cell or leave a message at the Computer Science office, 920-424-2068). **No after-the-fact notifications will be accepted.**
2. Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

Assignments: All assignments must be submitted electronically via d2l. It is your responsibility to ensure that your assignment was submitted correctly. You must double check to ensure your assignment was uploaded correctly.

No late submissions will be accepted.

Academic Dishonesty: Academic dishonesty of any kind will not be tolerated. All assignments, quizzes and exams are to be completed individually. While discussion of ideas and problems with fellow students is encouraged, all projects must be done individually. In certain circumstances, code fragments from the instructor may be provided to eliminate tedious coding or to provide a common framework for all students. All other code must be original. Online resources may be used to help you understand the material, but you may not copy online code nor can you "borrow" code from other students, past or present. For group assignments, each group must submit original work.

Any suspected academic dishonesty will be dealt with on a case-by-case basis. Any clarification of what does or does not constitute academic dishonesty must take place *before* you turn in questionable work. For clarification on what constitutes academic dishonesty, contact me or consult the printed policy in the UWO Student Discipline Code, Chapter UWS 14.

Topic	Coverage	Learning outcomes, i.e., the student is expected to:
Data Representation	i. Character ii. Integer iii. IEEE 754 Floating Point Standard	a. Express characters and integers in binary, hexadecimal, signed and unsigned representations. b. Determine when overflows occur in signed or unsigned additions and subtractions of integers. c. Write normalized and de-normalized floating point numbers in single and

		double precision using IEEE 754 Floating Point Standard. d. Analyze the IEEE 754 Floating Point Standard and determine what integers cannot be represented exactly by the Floating Point Standard
Instruction Set Architecture of a RISC computer	i. CPU, memory and the system bus ii. Program memory layout iii. Assembly language instructions iv. Assembly language implementation of high-level language control structures v. Assembly language implementation of variables and expressions vi. Assembly language implementation of one- and two-dimensional arrays vii. Pseudo-instructions and how an assembler works	a. Organize the memory layout of global integers and characters based on Little/Big-Endian conventions b. Edit an assembly language program, assemble the program and print its output on the Linux console c. Design assembly language program given high-level source code d. Implement assembly language programs that read in integers from the console, process the input and print the results on the console e. Implement high-level language control structures in assembly language f. Implement one- and two-dimensional arrays and control structures in assembly language (do-while, if-else, and for loop)
Implementation of functions	i. System stack, stack frames and local variables ii. Return address, register conventions iii. Parameter-passing methods: call by value, call by reference	a. write nested function calls using stack frames and local variables b. write an assembly language program with recursive functions c. write an assembly language function to perform numerical analysis, e.g., finding the square root of a double-precision floating point number
Computer architecture below the instruction-level	i. Fetch-execute cycle ii. Instruction encoding	a. encode assembly language instructions into machine

		language instructions b. read, understand, and debug machine language programs c. design and implement machine language programs