Computer Organization and Assembly Language
CS251 - Spring 2013
Credits: 3 hours

Instructor: Justin Miller **Office:** Halsey 221
Email: millerju@uwosh.edu
Office Hours: **Tuesdays 7:30-8:30 pm in HS220**

Class Meetings: **Tuesdays 4:30-7:30 pm in HS208**

Prerequisites: A grade of C or better in CS221

Class Web Page: http://csf11.acs.uwosh.edu/moodle/

Textbook: No required textbook

Suggested Reference:
Intro to Computing Systems: From Bits and Gates to C and Beyond (Second Edition) by Yale N. Patt and Sanjay J. Patel

Tests: Exam #1: February 26th
Exam #2: April 2nd
Exam #3: May 7th

If you have special needs, please come and talk to me as soon as possible so I can accommodate your needs right away.

Topic Coverage and Learning Outcomes:

This course aims to give students an overview of processor and memory hardware, and to teach them how high-level language programs map onto some Reduced Instruction Set Architecture or RISC computer. Students will learn how computer hardware supports the instruction set architecture. Students will be able to analyze why programs behave the way they do and how inefficiencies arise. Students will learn how to implement pointers and references in machine language.

1. Data Representation
i. Character
ii. Integer
iii. IEEE 754 Floating Point Standard

The student is expected to:
(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 denormalized floating point numbers in single and double precision using the 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

2. 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

The student is expected to:
(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 console, process the input and print 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)

3. 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

The student is expected to:
(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

4. Computer architecture below the instruction level

i. Fetch-execute cycle
ii. Instruction encoding

The student is expected to:
(a) encode assembly language instructions into machine language instructions
(b) read, understand, and debug machine language programs
(c) design and implement machine language programs

Course Grading Policy

Your final grade for this course will be based on three components: in class quizzes, homework (programming or written) assignments and exams. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Quizzes (all equally weighted)	10%
Homework assignments (all equally weighted)	30%
Exams (all equally weighted)	60%

Finally, your letter grade for the course will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
≥ 92	A	≥ 72	C
≥ 90	A-	≥ 70	C-
≥ 88	B+	≥ 68	D+
≥ 82	B	≥ 62	D
≥ 80	B-	≥ 60	D-
≥ 78	C+	<60	F

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me *as soon as possible*. Do not wait until the end of the semester to bring up grading issues. Also, I will *not* be available to discuss grades after the end of the final week.

Attendance and Participation

You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me outside of class to discuss any questions you may have, and to have completed the assignments on time. **It is hard to imagine how a student could do well in this course while missing classes or attending them unprepared.** On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage you to ask any question** or raise any issue you have with the course either during class or in my office hours. I will also gladly meet with you by appointment. Send me email to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available just before an exam or the due date for an assignment since you may not be the only one needing help at the last minute.

Late Submissions

I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each one of them will come with a deadline (day and time) after which any submission will be considered late. The late-submission policy works as follows:

Turned in	Penalty
On the due date but after the deadline	10%
One day after the due date	20%
Two days after the due date	40%
Three days after the due date	60%
Four days after the due date	80%

More than four days after the due date	100%

Note that assignments that are more than four days late receive no points. Weekend days and holidays count as "regular days" when computing late penalties. Each (late) day starts precisely at midnight. Extensions on assignments may be granted at the discretion of the instructor if you provide a valid justification (in the form of a written excuse from a medical doctor or the Dean of Students Office) before the due date. Late submissions can easily be avoided by starting to work on the assignment right away and asking for help early if you get stuck.

If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give the instructor a valid justification (see above), ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester. If you miss a quiz, you **may** be able to take a make-up quiz, provided you give the instructor a valid justification (in writing) for your absence.

Collaborating versus Cheating

While it is acceptable to discuss the problem statement, premises, goals, constraints, etc., of the assignments with others, you must submit your OWN work EXCLUSIVELY. You may not "borrow" any piece of code or design of any length from anybody else, unless you can live with a zero and the other potential academic sanctions of cheating (see the UWO Student Discipline Code - Chapter UWS 14).

In conclusion, remember that computer science classes require a lot of work in addition to active participation in class. It takes considerable practice to develop the technical and analytical skills targeted by this course. You will need to spend **at least (and typically much more than) three hours of effort outside of class for each in-class hour**. Having said this, I expect every hardworking student to do well in this course.

Have fun this semester and good luck!