# Syllabus - CSCI 241, Fall 2017
# Computer Organization and Assembly Language Programming

**Instructor:** Zuoliu Ding
**Email:** zding@live.com
**Schedule:** 7:00 PM - 9:05 PM, Tuesday and Thursday
**Voice:** (909) 274-0012
**URL:** http://staffwww.fullcoll.edu/zding
**Classroom:** North Science - FC 611C

**CRN:** 13354

**Text:** ***"Assembly Language for x86 Processors"***, by Kip R. Irvine (7th Edition), Prentice-Hall, 2014.

**Course Prerequisite:** CSCI 133 or CSCI 223 or equivalent with grade "C" or better.

**Student Learning Outcomes:** Analyze the architecture and use the instruction set of one or more processing platforms (e.g., Intel), and to write computer programs using the native instructions for a target processor.

**Instructional objectives:**

A. Recognize, identify, and translate numbers between binary, hexadecimal, octal, and decimal representations.
B. Use effectively an interactive source code debugger to analyze and debug low-level code.
C. Describe and express in an assembly language program, data structures used to solve common engineering, mathematical, and business oriented problems.
D. Identify and discuss the machine organization of a typical single-processor computer system.
E. Name and define the elements of an assembly language program.
F. Employ assembly language directives and operators to effectively express code and data in an assembly language program.
G. Diagram and categorize the addressing modes of a complex instruction set microprocessor.
H. Use current software engineering techniques to design and implement programs.
I. Diagram and analyze the use of the hardware stack when designing procedures.
J. Employ structured techniques to construct branching and looping in assembly code.
K. Employ bitwise logical operators in assembly language programs.
L. Create and employ assembly language macros and object libraries.
M. Design programs that interact directly with hardware through both port I/O and memory mapped hardware.
N. Describe the interrupt process and the techniques used to write interrupt handlers.
O. Design programs that manipulate the file structures of the operating system directly.
P. Explain the organization of information on a floppy or hard disk.
Q. Describe and demonstrate the use of most instruction in the instruction set of the processor.
R. Write readable internal and external documentation for assembly language programs.
S. Describe and use binary coded decimal numbers in assembly language programs.
T. Formulate and employ structures to simplify the handling of aggregate data structures.
U. Design assembly language programs that use a modular approach to programming as well as use data abstraction ideas.
V. Employ conditional assembly to make assembly programs more flexible and maintainable.
W. Evaluate the efficiency of an assembly language program.

**Development environment:** Using Microsoft Visual Studio IDE in teaching and learning x86 Assembly Language on Windows. Most programs created here only can be run on Windows. To be familiar with Visual Studio IDE and Windows OS is strongly preferred.

**Attendance:** As a comprehensive, demanding course, your attendance is taken at each class session. A student who misses over four classes may be dropped. If you cannot attend class due to serious illness or other extraordinary circumstances, please contact me via email or other ways in advance. You may be asked to document your absences. If you wish to drop this course, it is entirely your responsibility to complete all the necessary paperwork. I'll be available for help at some time; and if you need to see me, please feel free to contact me to make an appointment. If I am more than fifteen minutes late for class, the entire class is excused for that day.

**General Policy:** This course will cover most of the chapters in the book, although some may receive more attention than others. A tentative Course Schedule of topics is available for your preparation and content preview is strongly recommended. ***Concept understanding, programming capability, and problem solving are highly emphasized*** throughout the whole course. The class work required consists of four categories. The weight of each category is estimated as follows and may be adjusted in teaching. **Final grading is not only based on the points, but from a comprehensive view of the student's performance**.

| Category | # of Times | Points | Percent |
|---|---|---|---|
| Examinations | 3 | 300 | 73% |
| Required Programming Exercises | About 20 | About 100 | 24% |

| Category | # of Times | Points | Percent |
|---|---|---|---|
| Section/Chapter Review Questions/Exercises | About 20 | N/A | N/A |
| Optional Programming Exercises (Extra Credits) | Variable | Added | Added |
| Lab Attendance | One hour per week | 12 | 3% |
| Total (Without Extra Credits) | - | 412+ | 100+% |

* Note that attendance does not relate to the grade you are getting, so grading standards are based on the percentage not on a curve:

| A: 90%+ | B: 80-89% | C: 70-79% | D: 60-69% | F: Below 60% |
|---|---|---|---|---|

**Examinations:** There will be two mid-term tests and one final. Each of these exams is worth 100 points. There will be no make-ups. Midterm 1 will cover Chapters 1 - 5; Midterm 2 will cover Chapters 6 - 9. The final will cover Chapters 10 - 13. Parts of the Chapter 14, 15, 16, or 17 may be considered as selective. The format of these exams will consist of multiple choice, fill-in-the-blanks, code tracing, diagramming, and function designing and implementations.

**Programming Exercises:** There will be small problems to solve in programming for each class session (a two-hour lesson), which are mainly selected from the Programming Exercises in the text. They are worth about 5 points each. I may provide starter kits downloadable from Assignments for some projects. You can create your own design to solve the problem without caring starter kits, as long as you follow assignment requirements. You must ***submit your assignment for both code and documented explanations required***. Submitting your programming work in Email is highly preferred. The assignments will be graded on organization, correctness, and level of professional quality. **You may be required to answer questions regarding your code to make sure that you understand the problem and you do the programming yourself. Copying others' program is considered as cheating with zero point received.**

**Section/Chapter Review Questions/Exercises:** These are self-test and no turn-in required but also important homework. The questions are selected from the Section or Chapter Reviews in textbook. **You can find the answers available in the textbook and class site to verify by yourself**. The Review Questions/Exercises help you understand concepts in teaching materials and in tests. ***A strong recommendation is that you do all of them after the class and make sure your answers are correct***. You are free to bring any problem to the class to ask, if necessary.

**Extra Credits:** We have optional programming exercises nearly for each chapter, which can be additional problems to consider with bonus points. You are encouraged to solve these problems to obtain extra credits, as long as you have potential to face more challenges. There might be essays assigned to write for designated topics that will evaluate your ability in analysis and comprehension of concepts. The seminars or quizzes might be scheduled according to teaching pace and content. ***You are always encouraged to present your own opinion, point of view, and code description in your assignments, as well as in classroom discussions***.

**Assignment Turn-in Policy:**

1. Any assignment including Programming and Extra Exercises, will be expected within one week. During this period, the maximum score is full points.
2. A grace period of the second week turn-in: When submitting in the second week, your maximum score is 70% of the full points.
3. Not acceptable after the second week.

**Final Exam:** 12/14/2017, 7:00 PM - 9:05 PM, FC 611C

**Laboratory Hours:** As part of this course, students are required to spend one hour per week outside of class time engaged in instructional activities related to the course. You must use the Computer Science Lab to meet this requirement. Please contact CS Lab for actual open hours. If you miss a week, you may make it up the following week. Totally in the semester, 12 lab hours are expected. See Lab policies and schedules.

**College Policies:** Please read College Policies (http://math.fullcoll.edu/downloads/division-policies-10-08.pdf) carefully about Academic Honesty, Americans with Disabilities Act (ADA) Statement, Emergency Response Statement, Student behavior/conduct, Grievance/Grade Appeal Procedure, etc.

**Information and Resources:** You can click the hot links on the left pane to know more about the class and your progress.

- **Course Schedule:** The tentative course schedule planning on weekly coverage.
- **Assignments:** Explanations of teaching contents and homework updated after each lesson.
- **Textbook Site:** The textbook sample source code, information and help.
- **Reference Links:** The online resource, useful and helpful documentation with technical details.
- **Score Board:** Check your scores of assignments and tests by the Student ID/Password.
- **Evaluate Me:** Put your feedback to evaluate my lecture and class, send comments if any.