Course Description

Instructor. Prof. Richard Chang, chang@umbc.edu, 410-455-3093.
Office Hours: Tues & Thu 11:30am – 12:30pm, Wed 10:00am – 11:00am, ITE 326.

Course Web Page. http://umbc.edu/~chang/cs431

Time and Place. Tuesday & Thursday 10:00am – 11:15am, MP 102.

Textbook. *Compilers: Principles, Techniques & Tools* (second edition), Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman. Addison Wesley, 2007. (ISBN: 0-321-48681-1)

You will also need a C programming guide.

References:

- *Flex & Bison,* John Levine. O'Reily, 2009. (ISBN: 978-0-596-15597-1)
- *The Bison Manual,* Charles Donnelly and Richard Stallman. Free Software Foundation, 2003. (ISBN: 1-882-11423-X). This book is available online.

Prerequisites. The course prerequisites are CMSC313 Computer Organization and Assembly Language Programming, CMSC331 Principles of Programming Languages and CMSC341 Data Structures. You will need assembly language programming experience from CMSC313, an understanding of programming language design from CMSC331 and strong programming skills from CMSC341. In addition, CMSC441 Design and Analysis of Algorithms and CMSC451 Automata Theory and Formal Languages are very helpful, but not required.

Objectives. In this course, students will have the experience of designing and constructing a working compiler. Students will also gain an understanding of the algorithms used in compiler design.

Grading. Final grades will be based upon 5 homework assignments (20%) and 7 programming projects (80%). Each homework assignment contributes 4% to your final grade and each programming project contributes 10%, except for the final programming project which contributes 20%.

The final letter grades are based on the standard formula:

$$0 \le F < 60, \quad 60 \le D < 70, \quad 70 \le C < 80, \quad 80 \le B < 90, \quad 90 \le A \le 100$$

Depending upon the distribution of grades in the class, there may be adjustments in the students' favor, but under no circumstances will the letter grades be lower than in the standard formula. Grades will not be "curved" in the sense that the percentages of A's, B's and C's are not fixed.

Grades are given for work done *during* the semester; incomplete grades will only be given for medical illness or other such dire circumstances.

Quizzes, Tests and Exams. There are no quizzes, tests, mid-term or final exams in this class.

Programming Projects. Students will use the Unix tools `lex` and `yacc` to construct a compiler that translates a high-level programming language of their own design to Intel i386 assembly language. This semester-long project will be completed in 7 stages with due dates given below. All projects are due on Mondays at 12 noon.

- Project 1 (9/21) : a compiled calculator.
- Project 2 (10/5): variables, a symbol table and printing.
- Project 3 (10/12): printing string constants.
- Project 4 (10/26): Boolean values, `if` statements, `while` loops and `for` loops.
- Project 5 (11/9): procedure calls and parameter passing.
- Project 6 (11/23): function calls and local variables.
- Project 7 (12/14): additional features to be selected by the students.

Project grades will be based on design, correctness, completeness, efficiency, testing and documentation. Note that each programming project builds upon the previous one. So, it is very important for errors in a project to be corrected before proceeding to the next project. To facilitate this, there will be two types of deductions in the grade for a project: temporary and permanent. A temporary deduction will be rescinded if the error is corrected by the due date of the next project. Deductions for lack of documentation and lack of testing will always be permanent.

Programming Teams. Students may choose to work in a team of two students. This is entirely voluntary. The two students in a team will receive the same grade for the project. Please note that your teammate's performance will have an impact on your project grade — your teammate's failure to complete his/her share of the project is not an acceptable excuse. All programming teams must be formed by the due date of Project 2.

If your programming team does not work out, you can continue the rest of the semester working on the programming projects individually. However, your grades for previously submitted projects will remain the same. Furthermore, you will not be allowed to form another team (even with your previous teammate).

Late Pass. For one time during the semester, you can extend the due date of your project by one week without penalty. It is strongly suggested that you not use this "late pass" on Project 1. Each programming team is allowed only one late pass.

Attendance. Attendance is very important because a large portion of class time is allocated to discussions on the programming projects. These discussions will not be duplicated on the website or in the project descriptions and will contain information needed to complete the programming project.

Homework Policy. Assignments are due at the *beginning* of lecture— *this is to allow for timely grading and discussion of the homework solutions*.

Academic Integrity. Students are allowed to discuss homework problems and programming projects. Collaborators and reference materials must be acknowledged at the top of each

homework assignment. However, homework solutions must be written up *independently*. A student who is looking at someone else's solution or notes, whether in print or in electronic form, while writing up his or her own solution is considered to be cheating.

Students are allowed to discuss programming projects and receive debugging help from anyone. However, only the student and his/her programming partner (if any) are allowed to edit the programs *even for debugging purposes*. Thus, all of the source code for a programming project must be authored by the students submitting the project. Code reuse for standard data structures from reference sources will be allowed, but must be acknowledged and must be discussed with the instructor beforehand. All other appearances of other people's code in a submitted programming project will be considered cheating.

Cases of academic dishonesty will be dealt with severely.

The UMBC academic integrity policy is available at http://www.umbc.edu/provost/integrity_policy.html.

We will follow the textbook *Compilers: Principles, Techniques & Tools* (the purple dragon book) by Aho, Lam, Sethi & Ullman . The following schedule outlines the material to be covered during the semester and specifies the corresponding sections in the textbook.

Date	Topic	Reading	HW Due
Tue 09/01	Introduction	Chapter 1	
Thu 09/03	Overview of Compilation	Chapter 2	
Tue 09/08	Lex & Yacc		
Thu 09/10	Lex & Yacc		
Tue 09/15	Assembly Language Review		
Thu 09/17	Assembly Language Review		
Tue 09/22	Lexical Analysis	Chapter 3	
Thu 09/24	Lexical Analysis		
Tue 09/29	Lexical Analysis		
Thu 10/01	Lexical Analysis		
Tue 10/06	Syntax Analysis	Chapter 4	
Thu 10/08	Syntax Analysis		HW1
Tue 10/13	Syntax Analysis		
Thu 10/15	Syntax Analysis		
Tue 10/20	Syntax Analysis		
Thu 10/22	Syntax Analysis		HW2
Tue 10/27	Syntax-Directed Translation	Chapter 5	
Thu 10/29	Syntax-Directed Translation		
Tue 11/03	Syntax-Directed Translation		
Thu 11/05	Syntax-Directed Translation		HW3
Tue 11/10	Intermediate-Code Generation	Chapter 6	
Thu 11/12	Intermediate-Code Generation		
Tue 11/17	Run-Time Environments	Chapter 7	
Thu 11/19	Run-Time Environments		HW4
Tue 11/24	Run-Time Environments		
Thu 11/26	*Thanksgiving Break*		
Tue 12/01	Code Generation & Optimization		
Thu 12/03	Code Generation & Optimization	Chapter 8	
Tue 12/08	Code Generation & Optimization		
Thu 12/10	Code Generation & Optimization		HW5