CMPS 374/274 – Compiler Construction
Course Syllabus

Course Website
All course material (slides, assignments and homework) will be available on Moodle. Students must regularly check Moodle for new course announcements. Lecture notes will be posted within one day after the lecture.

Catalog Description
A course that covers syntax specifications of programming languages, parsing theory, top-down and bottom-up parsing, parser generators, syntax-directed code generation, symbol table organization and management, dynamic storage allocation, code optimization, dataflow analysis, and register allocation. Prerequisites: CMPS 255 – Computer Architecture, CMPS 258 – Programming Languages, and CMPS 257 – Theory of Computation. 3 credits. Biennially.

Textbook
- Available at AUB bookstore

Additional References

Course Topics
- Introduction to compilers (Chapters 1 and 2 – the dragon book)
- Lexical analysis and scanners (Chapter 3 – the dragon book)
- Syntax analysis and parsers (Chapter 4 – the dragon book)
- Syntax-Directed Translation (Chapter 5 – the dragon book)
- Semantic analysis (Chapter 6 – the dragon book + handouts)
- Intermediate code generation (Chapter 6 – the dragon book + handouts)
- Run-time environments (Chapter 7 – the dragon book + handouts)
- Intermediate code optimization (Chapters 8 and 9 – the dragon book + handouts) – if time permits
- Target code generation (Chapters 8 – the dragon book + handouts) – if time permits

Course Objectives
The objectives of this course are to educate the students in:
1. Expressing a programming language formally
2. The analysis and synthesis of a compiler
3. The different phases of a compiler
4. How to optimize the machine code generated by the compiler to make it faster and more efficient
5. The basic skills needed to design and implement a compiler of a given language
6. Utilizing compiler construction tools in the process of building a compiler

**Learning Outcomes**

At the end of the course, students will be able to:
1. Express the grammar of a programming language
2. Build lexical and syntax analyzers and use them in the construction of scanners and parsers
3. Perform the operations of semantic analysis
4. Build a code generator
5. Use different compiler optimization schemes in addition to efficient register allocation and garbage collection
6. Design and program a complete working compiler for a given language

**Evaluation**

A tentative breakdown of the course evaluation is given below. The percentages are subject to change as the course progresses.

- Written homework 5%
- Project & presentation 40%
- Midterm exam 20%
- Final exam 30%
- Class attendance & participation 5%

The instructor has the right to modify the grading criteria per his own judgment.

**Examinations**

Examinations may consist of multiple choice, True-False, and subjective questions. They will be closed book, closed notes. Calculators, phones and electrical equipment are not allowed. Cheating in exams will result in immediate zero grade. Makeup exams will only be given in extremely rare circumstances after presenting a solid and convincing case to the instructor within one week after the exam date. Missed exam will result in a zero grade for the particular exam.

**Project**

The programming project is a salient element of this course. It is preferable that students form groups of two, but a group of one is also acceptable. CMPS 274 students must only work with each other; the same applies for CMPS 374 students.

Students must form their groups and notify the TA within one week after the first class of their group members as well as the tool(s) chosen for the project (where applicable). Each group will be assigned a group number that must be used in each submission. Each group must select one of the members as delegate; he or she will be responsible for the submissions of his/her group. The project might include developing a compiler for a language of the students (or the instructor’s) choice, developing a translator for an existing language to produce additional useful code, enhancing some exiting compiler(s), or solving a problem related to an evolving research topic using compilers concepts. The project may thus be divided into multiple phases/checkpoints (typically three to five phases). For each phase, students must deliver a working code along with a report describing the design and implementation of their solution to the corresponding phase.

By the end of the project, students shall deliver a working and well-documented code along with a report combining all the phases of the project. Students must also conduct an in-class presentation to summarize their findings and explain the tool(s) used in the process of building their solution(s). Each phase will be graded on correctness, coding style, completeness of test cases, and quality of report.

Typically, each phase will comprise 9% of the grade, and the project presentation will comprise 4%.
Note: Students may use relevant code that they find online. This will not affect their grade. However, they must reference the source, and specify in the report which aspects were used; otherwise, it will be an act of plagiarism.

Guidelines for Project Reports
Students must organize the report of each phase such that the final report will be a coherent report “stitching” the reports of all the phases together. Reports must be in PDF and must be formatted using the IEEE Templates for Transactions format (using LATEX ONLY):

http://www.ieee.org/publications_standards/publications/authors/author_templates.html

Reports not formatted using LATEX will not be corrected, which will result in a zero grade on the report.

Details on what to include in the report will be specified in the given description. However, each report must at least include the following elements:

1. Abstract
2. Introduction
3. An overview of the tool(s) used
   a. Advantages of the tool(s)
   b. Disadvantages of the tool(s)
4. Design of the corresponding phase
   a. The errors handled in each phase
   b. Assumptions made (if any)
   c. Completed and missing (unfulfilled) requirements
5. Tasks handled by each group member
6. Personal Comments
7. Conclusions

Guidelines for Project Presentations
For project presentations, at least the following elements must be included:

8. A brief overview of the tool(s) used
9. Design of compiler
   a. Design of every phase
   b. The errors handled in each phase
   c. Assumptions made (if any)
   d. Any additional tool(s) used
10. Tasks handled by each group member
11. Short demo

Homework
Written homework consist of problems from the book, made up problems, or readings from literature. Written homework MUST be done and submitted individually.

Note on Submission and Academic Integrity
Students are encouraged to talk to each other, to the TA, to the instructor, or to anyone else about any of the assignments. Any assistance, though, must be limited to discussion of the problem and outlining general approaches to a solution. Each group must write and submit its own solution. Submission guidelines will be provided in each assignment’s handout. Consulting another student or group’s solution is prohibited, and submitted assignments and homework may not be copied from any source. These and any other form of collaboration on assignments constitute cheating. Detected cheating will not be tolerated; the parties involved in cheating will get the mark 0. Repeated cases will be reported to the Disciplinary Committee for further action, which may result in a Dean’s warning or expulsion from the university.
Late submissions will get -10% for every day late and will not be accepted after 2 days of due date.

**Attendance:**
Attendance may be taken. Students are expected to attend all lectures. For each lecture missed, the student will lose 1%. Students are entitled for only 5 missed lectures (i.e., 5%). Lectures are in sequence; if a lecture is in missed, chances are the subsequent lecture(s) won’t be understood. Catching up on the sequence of the lectures is the student’s responsibility and should be done in his/her own time and NOT at the expense of the instructor’s or TA’s times.

**Re-grading Policy:**
If the student thinks the instructor or TA made a mistake in grading, he/she should return the assignment or homework with a note explaining his/her concern to the TA no later than one week after the day the assignment is returned.

**Note for Special Needs Students**
AUB strives to make learning experiences as accessible as possible. If you anticipate or experience academic barriers due to a disability (including mental health, chronic or temporary medical conditions), please inform the course instructor immediately so that we can privately discuss options. In order to help establish reasonable accommodations and facilitate a smooth accommodations process, you are encouraged to contact the Accessible Education Office: accessibility@aub.edu.lb; +961-1-350000, x3246; West Hall, 314.

**Nondiscrimination**
AUB is committed to facilitating a campus free of all forms of discrimination including sex/gender-based harassment prohibited by Title IX. The University’s non-discrimination policy applies to, and protects, all students, faculty, and staff. If you think you have experienced discrimination or harassment, including sexual misconduct, we encourage you to tell someone promptly. If you speak to a faculty or staff member about an issue such as harassment, sexual violence, or discrimination, the information will be kept as private as possible, however, faculty and designated staff are required to bring it to the attention of the University’s Title IX Coordinator. Faculty can refer you to fully confidential resources, and you can find information and contacts at www.aub.edu.lb/titleix. To report an incident, contact the University's Title IX Coordinator Trudi Hodges at 01-350000 ext. 2514, or titleix@aub.edu.lb. An anonymous report may be submitted online via EthicsPoint at www.aub.ethicspoint.com