# MC97 General Information

### Max Hailperin; Spring, 1996

## 1 Outline

MC97 attempts to draw together the theory and practice of compiler construction. The early material on syntactic analysis and naive code generation, though theoretical in parts, will serve as the basis for compiler-writing projects. Meanwhile, the latter part of the course will press on to important contemporary topics in optimization: data-flow analysis, suppression of partial redundancies, register allocation and assignment, and instruction scheduling. These topics again appear theoretical, and because of time constraints they will not make their way into the projects. However, they are of great practical importance to modern high-performance computing systems, as we shall see.

## 2 Prerequisites

MC97 draws heavily on MC55, MC87, and MC48. Some notions from MC57 and MC36 also crop up. You're expected to be able to program. Since the compiler-building tools we have available are centered around the C programming language, it will be to your advantage to learn that language if you don't already know it.

## 3 Office Hours

I will be available in my office (OHS 303) on Wednesdays from 8:00 to 8:50, on Tuesdays and Thursdays from 11:30 to 12:20, on Mondays from 2:30 to 3:20, and by appointment. You may send me electronic mail at max@gac.edu or call me at extension 7466.

### World Wide Web

All course handouts will be available through my World Wide Web page, and some supplementary materials such as code to use as a starting point in assignments may be available there as well. The URL for this course is `http://www.gac.edu/~max/MC97`.

## 4 Texts

Our primary text will be the "red dragon book," i.e., *Compilers: Principles Techniques and Tools* by Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, Addison Wesley, 1986. We will also use some additional material: documentation, some of which will be on-line or in the lab area and some of which I'll hand out, and supplementary readings, which I will hand out.

## 5 Honor

Students are encouraged to discuss the course, including issues raised by the assignments. However, the solutions to homework and laboratory assignments should be individual original work unless otherwise specified. If an assignment makes you realize you don't understand the material, ask a fellow student a question designed to improve your understanding, *not* one designed to get the assignment done. To do otherwise is to cheat yourself out of understanding, as well as to be intolerably dishonorable.

Any substantive contribution to your solution by another person or taken from a publication should be properly acknowledged in writing. Failure to do so is plagiarism and will necessitate disciplinary action.

## 6 Late Assignments

All homework and lab assignments are due at the beginning of class on the day indicated. Late assignments will be penalized by the equivalent of one letter grade for each weekday late or fraction thereof. However, no late assignments will be accepted after graded assignments are handed back.

If you are too sick to complete an assignment on time, you will not be penalized. Simply write "late due to illness" at the top of the assignment, sign your name and hand it in. Other circumstances will be evaluated on a case-by-case basis.

## 7 Grade Changes

Please point out any arithmetic or clerical error I make in grading, and I will gladly fix it. You may also request reconsideration if I have been especially unjust.

## 8 Grading

I will provide you with a letter grade on each homework assignment and lab assignment, in addition to the mid-term and final grades, so that you may keep track of your performance. As a guideline, the homeworks will contribute approximately half of your final grade, with the labs contributing the other half. However, I reserve the right to subjectively adjust your final grade. Please see me if you have any question how you stand. Class participation is not graded; however, it allows you to find and repair the gaps in your understanding before doing the assignments, and thus can dramatically improve your grade.

## 9 Style Guidelines

All homework should be readily readable, and should not presuppose that I already know what you are trying to say. In particular:

- Use full English sentences where appropriate (namely almost everywhere, including in mathematical proofs or derivations).
- Word-process or type your homework if you can. In any case, make sure it is legible.

Programs should be easily read as well. Some points to remember are the following:

- Begin each program file with a comment block explaining the purpose of the program and identifying the author and date.

- In general, each procedure should be preceded by a brief comment specifying what the procedure does (*not* how it does it). This may include assumptions it makes about its inputs and guarantees it makes about its results. Occasionally the name of the procedure and of its arguments will adequately substitute for this. Global data structures may need to be similarly commented with information about purpose and invariant properties. Even type definitions may warrant an explanation if not self-explanatory.
- Use comments within procedures sparingly, making sure that they explain the code rather than echoing it. Comments should explain why the procedure does what it does, or why it works, or what a low-level operation means in higher-level terms.

Be sure your assignments are always stapled together and that your name is always on them.

## 10 Computer Logistics

You may use any computer facilities you want. However, my expectation is that the SGI workstations in OHS 326 and OHS 329 will be the normal choice. The assignments will presume you have access to a number of software tools that are already installed on the MCS SGIs. We'll be using the MIPS processor that the SGIs are based on as the target processor for your compliers. However, normally you'll be using a MIPS simulation program called SPIM rather than the real MIPS processor, and SPIM is in fact available on the NeXTs as well. (On the SGIs, the graphical user interface version of SPIM is called xspim while on the NeXTs it is NeXTspim.) So, given this and that similar development tools are available on the NeXTs (for example), it would be possible to work on machines other than the SGIs.

## 11 Accessibility

Please contact me immediately if you have special physical circumstances, e.g. impaired vision, which may affect the accessibility of the laboratory or other course components. I will do my best to facilitate necessary arrangements or resources.

## 12 Syllabus

In the reading column, section 0 means the material at the beginning of a chapter before the first section. If the reading is listed as up to a particular page, it means to the topic heading appearing on that page. If the next reading then just lists the same section number, that means to read the remainder of the section.

This is my best guess as to the rate at which we will cover material. However, don't be shocked if I have to pass out one or more revised syllabi.

| Date | Reading | Topic | Due |
|---|---|---|---|
| 2/5 | 1.0–2.2 | Introduction, MIPS ISA, syntax definition | |
| 2/7 | 2.3–2.4 | Syntax-directed translation | |
| 2/9 | 2.5–2.6 | Your first compiler | |
| 2/12 | 3.0–3.1, 3.3, 3.5 | Lexical analysis | HW 1 |
| 2/14 | | Lexical-analyzer generators | |
| 2/16 | | No class | |
| 2/19 | 4.0–4.2 | Context-free grammars | Lab 1 |
| 2/21 | 4.3 | Writing a grammar | |
| 2/23 | 4.4–p.188 | Top-down parsing | |
| 2/26 | 4.4 | More on predictive parsing | |
| 2/28 | 4.5, 4.7–p.220 | Bottom-up parsing | |
| 3/1 | 4.7–p.230 | SLR parser generation | Lab 2 |
| 3/4 | 4.7–p.240 | Canonical LR and LALR parser generation | |
| 3/6 | 4.8–p.251, 4.9 | Using ambiguous grammars; parser generators | |
| 3/8 | 5.0, 5.1, 5.3 | Syntax-directed definitions | |
| 3/11 | 5.4–p. 314 | L-attributed definitions | |
| 3/13 | 5.6, 5.2 | Bottom-up attribute evaluation | |
| 3/15 | 6.0–6.4 | Type checking | HW 2 |
| 3/18 | 7.0–7.2 | Run-time environments | |
| 3/20 | 7.3 | Storage-allocation strategies | Lab 3 |
| 3/22 | notes | Project run-time environment; modern R.T.E. issues | |
| 3/25 | 8.0–8.1 | Intermediate code | |
| 3/27 | 8.2, 8.3–p.481 | Declarations and simple assignments | |
| 3/29 | 8.3 | Array access; type conversions | |
| 4/10 | 8.4 | Control flow | |
| 4/12 | 9.0–9.3 | Code generation | Lab 4 |
| 4/15 | 9.4 | Basic blocks and flow graphs | |
| 4/17 | 10.0–10.2 | Optimization | |
| 4/19 | | Conferences | |
| 4/22 | | Conferences | |
| 4/24 | 10.4 | Loops in flow graphs | HW 3 |
| 4/26 | 10.5 | Data-flow analysis | |
| 4/29 | 10.6 | Iterative data-flow analysis | Lab 5 |
| 5/1 | | More data-flow analysis | |
| 5/3 | notes | Suppression of partial redundancy | |
| 5/6 | 10.11 | Data-flow analysis frameworks | |
| 5/8 | notes | Register allocation and assignment | HW 4 |
| 5/10 | notes | Instruction scheduling | |
| 5/13 | notes | More on instruction scheduling | HW 5 |
| 5/15 | | Synthesis | Lab 6 |