Subject Introduction

COMP3131/9102: Programming Languages and Compilers
Semester 2, 2001

Objectives

This subject focusses on compilation techniques needed to translate programs in a standard programming language into executable code on microprocessor architectures. By writing a compiler in Java for a variant of C, you are expected to gain a good understanding of important fundamental principles in programming languages and develop further your programming and software engineering skills.

This project can be challenging but fun. It is probably the first time that you write a compiler and run a program that you compiled!

Pre-requisites

Students must have successfully completed COMP2011 Data Organisation before attempting this subject. Therefore, this subject makes some basic assumptions about your knowledge of Java, data structures and algorithms. You are expected to program efficiently.

Web Pages

The subject web pages will provide links to all material related to this subject. The pages can be accessed via:

http://www.cse.unsw.edu.au/~cs3131/

or

http://www.cse.unsw.edu.au/~cs9102/

You should become familiar with the structure and content of the subject web pages early on, and consult them frequently. In particular, all important announcements will be posted to the Subject Noticeboard on the opening page. Urgent information will also be sent to you by email via the class mailing-lists:

comp3131-list and comp9102-list

It is your responsibility to check this home page and your email regularly.

Staff

Name	Room	Phone	Email	Role
Jingling Xue	K17-501K	9385 4889	jxue	Lecturer-in-charge
Patryk Zadarnowski	K17-501/15	9385 4056	patrykz@cse.unsw.edu.au	Subject Admin

The Handbook Entry

```
Staff Contact: Dr J. Xue
UC 6 S2 L3 T1
Prerequisites: COMP2011
Note/s: Excluded COMP9102.

Covers the fundamental principles in programming languages and
implementation techniques for compilers (emphasis on compiler
front ends). Course contents selected from: program syntax and semantics,
formal translation of programming languages, finite-state recognisers
and regular expressions, context-free parsing techniques such as
LL(k) and LR(k), attribute grammars, syntax-directed translation,
type checking, code optimisation and code generation.

Lab: implementation of a compiler in a modern programming language
for a small programming language.
```

Syllabus

The following is a summary of the topics that will be covered in this subject:

1. Lexical analysis
   - crafting a scanner by hand
   - regular expressions, NFA and DFA
   - scanner generator (e.g.,, lex and JLex)
2. Context-free grammars
   - parsing
   - derivations
   - parse trees
   - abstract syntax trees (ASTs)
3. Syntactic analysis

    - recursive-descent parsing and LL(k) parsing
    - bottom-up parsing and LR(k) parsing
    - parser generators (e.g., yacc, JavaCC and JavaCUP)
4. Semantic analysis
    - attribute grammars
    - symbol table
    - identification (i.e., binding)
    - type checking
5. Code generation
    - syntax-directed translation
    - Jasmin assembly language
    - Java Virtual Machine (JVM)
    - Java byte code generation

Reading Material

The following book is the recommended text for this subject:

- Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman, *Compilers: Principles, Techniques and Tools*, Addison-Wesley, 1986.
  This is the Bible on compiler construction. It is commonly known as the Dragon Book from the picture on its cover. For the professional, this is definitely the ultimate reference. For the uninitiated, some chapters can be tough to read. However, not all the topics in the text will be given the same amount of attention, with preference tending towards the topics needed to support the assignment tasks of implementing our C00 compiler.

  The lecture transparencies and recommended on-line resources over the semester should provide sufficient supplement materials for you to study this subject well.

Other useful reference books include the following:

- Afred V. Aho and Jeffrey D. Ullman, *Theory of Parsing, Translation and Compiling*, Addison-Wesley, 1986.
- Andew W. Appel, *Modern Compiler Implementation in Java*, Cambridge University Press, 1997.
- C. N. Fisher and R. J. LeBlanc, Jr., *Crafting a Compiler with C*, Benjamin/Cummings, 1991.
- Thomas. W. Parsons, *Introduction to Compiler Construction*, W. H. Freeman and Company, 1992.
- N. Wirth, *Compiler Construction*, Addison-Wesley, 1996.

Lecture Timetable

Each week there will be one three-hour lecture on Monday (6pm -- 9pm, Smith Theatre) during which new material will be presented.

Assignments

Every student will do a compiler project that will produce a compiler that translates a variant of C called *VC* into Java bytecode.

The project must be coded in Java.

The project is broken up into five individual assignments:

1. scanner: read characters and produces tokens
2. recogniser: read tokens and parse the program only for syntactic correctness
3. parser: extend the parser to build an abstract syntax tree (AST)
4. static semantics: check semantics at compile-time
5. code generator -- generate Java bytecode

The completed compiler will have approximately 4,500 lines of code. We will supply many classes to simplify the coding, so you will have to write about between 2,000 and 2,500 lines of code.

By the end of the semester you will have implemented a compiler that will produce code that can be executed.

You will implement each assignment component by building on the previous ones that you have completed. Therefore, you are strongly advised to complete all assignments and complete them on time.

Assignments are to be completed individually and without assistance from others except the staff in this subject. Of course, all legal collaborations and discussions among students are encouraged. Plagiarism will not be tolerated; assignment submissions will be checked for similarity and anyone found guilty of submitting assignment work that they did not do themselves will be awarded a mark of zero for the subject.

This year, there will be a bonus project on writing a compiler for an extended language of VC. Details can be found in its specification .

FInal Exam

There will be a closed-book written exam for this subject in the November exam period. The exam will consist of mostly questions on the theoetical aspect of the subject. For example, the topics that typically appear in the exam include regular expressions, NFA, DFA, context-free grammars.

Assessment

The subject assessment includes components from the compiler project and a final closed-book exam.

The compiler project is marked out of 100 in the following proportions:

Component	Value
Assignment 1	14/100
Assignment 2	14/100
Assignment 3	16/100
Assignment 4	27/100
Assignment 5	29/100

The final exam is also marked out of 100.

Your final mark in this subject will be based on a combination of the above components.

```
prog = mark for all programming assignments (out of 100)
exam  = mark for the final exam (out of 100)
bonus = mark for the bonus programming assignment (out of 5)

final = min((2 x prog x exam) / (prog + exam) + bonus, 100)
```

I hope you enjoy this subject!

Jingling Xue
Last updated 07/23/2002 09:56:00