Compilers
G52CMP

January 2002

Contact details:

- Roland Backhouse,
  room B30, rcb@cs.nott.ac.uk, x 14212.

Please keep email traffic to a minimum. Come to see me if you have a query. I will always be available immediately after each lecture. You can speak to me then or make an appointment to see me later.

Aims of Course

Aims:

> To capture the interplay of theoretical ideas with practical programming problems.

Setting:

> Compilers: how do they work, how are they constructed?

Specifics:

- Understanding of compiler development.
- Experience of studying a large program.

Assessment

- 25% coursework.
- 75% 2-hour examination.

Coursework: Obtainable from

`http://www.cs.nott.ac.uk/~rcb/G51CMP`

Examination: 3 questions out of 5.

NB. If you are in the unfortunate situation of resitting the examination then any coursework you have done does not count and you will be required to answer 4 out of 5 questions.

Books

- *Programming Language Processors in Java. Compilers and Interpreters*. David A. Watt and Deryck F. Brown Pearson Education Limited 2000.

This is the main reference for the course and is a compulsory purchase.

The book's main advantage is that it uses Java and is clearly written. It is weak on linking the theory with practice and I will be supplementing the material in the textbook with lectures on the theoretical basis. There are several errors in the book. Consult the authors' web pages for corrections.

- *Compilers: Principles, Techniques, and Tools*. Alfred Aho, Ravi Sethi, and Jeffrey Ullman. Addison Wesley, 1986. (The 'Dragon book' – a classic.)

This is a well-written textbook that is well worth buying. It is very thorough and does a good job of demonstrating the relevance of the theory to the practice of constructing compilers. A disadvantage of this book is that uses C. However, reading C should not be a problem since the syntax of Java is based on C. The book includes a lot of material that I will not have time to cover. If you want a book that will last you many years rather than just for one semester then buy this one.

Handouts

- There will be no handouts during lectures. You should come prepared to take notes during the lecture.

- Lecture material: Photocopies of diagrams and other material from Watt and Brown's book (reproduced with the permission of the authors) can be purchased from the secretaries' office (room B33).

Coursework – MiniTriangle Type Checker

- Medium-scale programming.
- Visitor Pattern (OO "design pattern")
- Abstract Syntax Trees

Part 1 Familiarization.

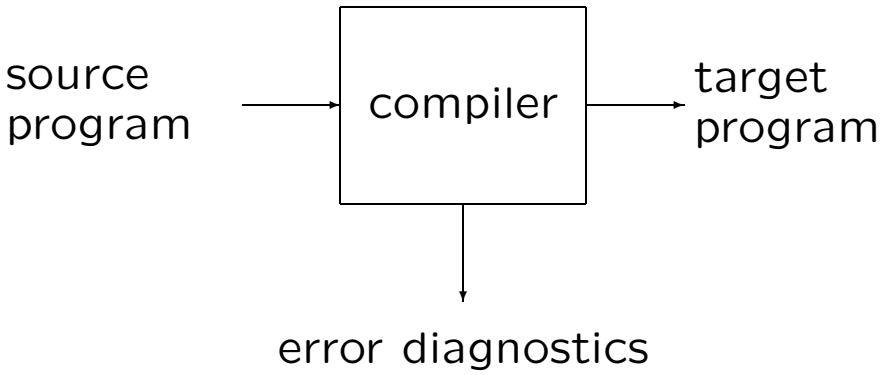
Part 2 Implementation.

Lecture 1

Basic Overview

What is a compiler?

Compilers are program translators:



Source programs: Many possible source languages, from traditional, to application specific languages.

Target programs: Another programming language, often the machine language of a particular computer system.

Error diagnostics: Essential for program development.

Where are compilers used?

- **implementation of programming languages**
  C, C++, Java, Lisp, Prolog, SML, Haskell, Ada, Fortran, ...

- **document processing**
  LaTeX → DVI/HTML/PDF, DVI → PostScript, Word documents → ...

- **natural language processing**
  nl → database query language → database commands

- **hardware design**
  silicon compilers, CAD data → machine operations

- **report generation**
  CAD data → list of parts, web-server access logs → access statistics, ...

- **all kinds of input/output translations**
  various UNIX text filters, ...

- ...

Not every program is a compiler, but many applications make use of compiler technology.

Compilation can be a complex process

A simple C (?) program:

```
main(){int i,tot,N=10;for/* This is a comment. I love
comments */(i=0;i<N;i++) tot=tot+   i;    printf("Total =
%d\n", tot);}
```

SPARC assembler code generated for it by gcc:

```
    .file    "for.c"
gcc2_compiled.:
.section    ".rodata"
    .align 8
.LLC0:
    .asciz  "Total = \n%d\n"
.section    ".text"
    .align 4
    .global  main
    .type    main,#function
    .proc    04
main:
    !#PROLOGUE# 0
    save %sp,-128,%sp
    !#PROLOGUE# 1
    mov 10,%o0
    st %o0,[%fp-28]
    st %g0,[%fp-20]
.LL2:
    ld [%fp-20],%o0
    ld [%fp-28],%o1
    cmp %o0,%o1
    bl .LL5
    nop
    b .LL3
    nop
.LL5:
    ld [%fp-24],%o0
    ld [%fp-20],%o1
    add %o0,%o1,%o0
    st %o0,[%fp-24]
.LL4:
    ld [%fp-20],%o1
    add %o1,1,%o0
    mov %o0,%o1
    st %o1,[%fp-20]
    b .LL2
    nop
.LL3:
    sethi %hi(.LLC0),%o1
    or %o1,%lo(.LLC0),%o0
    ld [%fp-24],%o1
    call printf,0
    nop
.LL1:
    ret
    restore
.LLfe1:
    .size     main,.LLfe1-main
    .ident    "GCC: (GNU) 2.7.2"
```

Inside the compiler

Traditionally, compilation is broken down into several steps or *passes*:

- **Lexical analysis**: Translate the input program, entered as a sequence of characters, into a sequence of words or symbols (tokens). For example, the keyword `for` should be treated as a single entity, not as a 3 character string.

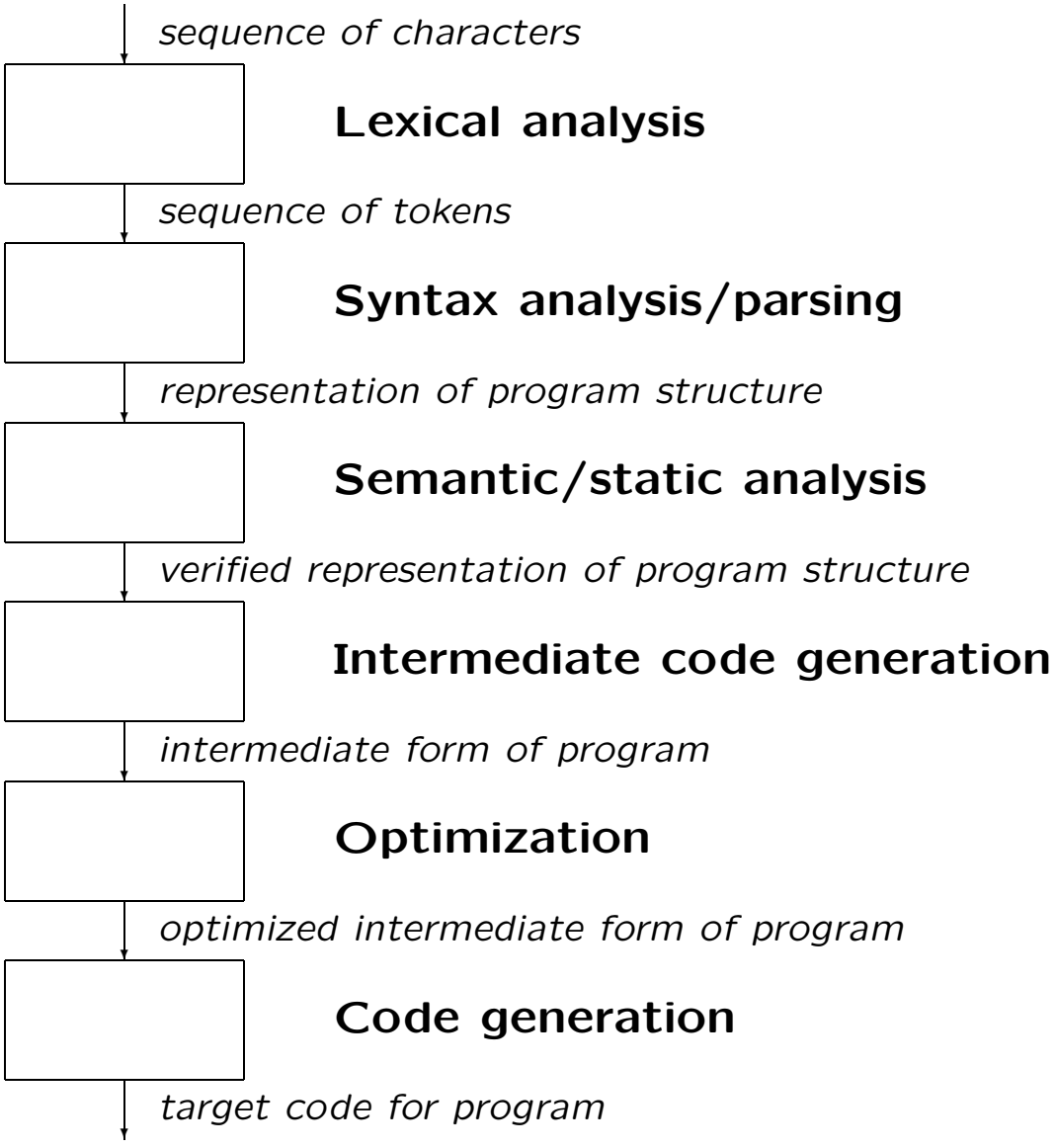
- **Syntax analysis/parsing**: Determine the structure of the program, for example, identify the components of each statement and expression and check for syntax errors.

- **Static/Semantic analysis**: Check that the program is reasonable, for example, that it does not include references to undefined variables.

- **Code generation**: Output the appropriate sequence of target language instructions.

This leads to a compilation pipeline with various intermediate data formats. Parts closer to the **front-end** of this pipeline depend more heavily on the source language whereas parts closer to the **back-end** depend more heavily on the target language.

Inside the compiler, continued



Inside the compiler, A Simple Example

See Aho, Sethi and Ullman, p13.

Summary and Homework

Summary: This lecture has been about:

- Aims of the course.
- What a compiler is.
- Applications of compiler technology.
- Overall structure of compilers.

Homework:

- Read chapter 1 of Watt and Brown.