1. Introduction

Why studying compilers construction

Andrea Polini

Formal Languages and Compilers
Master in Computer Science
University of Camerino

October 14th, 2014

ToC

ToC

Teacher and Course

- Andrea Polini
  - e-mail: andrea.polini@unicam.it
  - web: http://didattica.cs.unicam.it/doku.php?id=flc:flc2014
- Formal Languages and Compilers
  - lessons:
    - FLC: Tuesday 3pm to 5pm
    - FLC: Thursday 3pm to 5pm
  - web: http://didattica.cs.unicam.it/
- Exam dates:
  - February 3rd and 24th, 2015
  - June 9th and July 7th, 2015
  - September 8th and 29th, 2015
  - February 2nd and 23th, 2016

Course Objective

- At the end of the course you should be able to understand the basic issues related to compilers construction.
- At the same time you should have developed skills that will permit you to develop a compiler for a new simple language.

Study material

- **Reference book**:
  - Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman
    Compilers – Principles, Techniques and Tools, 2nd Ed.
    Addison-Wesley, 2007.
- **Further references provided by the teacher**

Final Exam!!!

- Individual project
- Written paper

ToC

Compilers vs. Interpreters

Two approaches to permit the execution of a program, written using an high level language, on a physical machine:

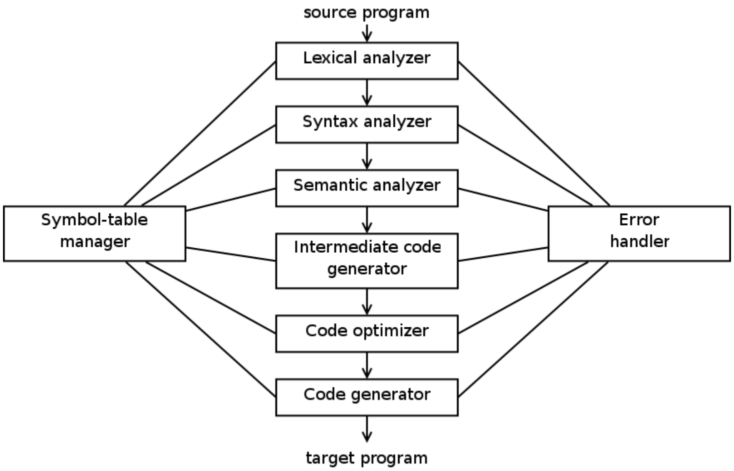
- Compilers: use of a program that can read a program in one language (source) and translate it into an equivalent program in another language (target)
- Interpreters: use of a program that takes in input the program and data and run the program on the data without the need to make an explicit translation into the machine code

Birth

- 1954 – IBM develops the 704 (software cost $>$ hardware cost)
- 1954 - 1957 – FORTRAN I (FORmula TRANslating system) is developed (In 1958 50% of code is written in FORTRAN)
    - The definition of the first compiler led to an enourmous body of theoretical work

Compiler constuction is a complex engineering activity (practice) which need to be based on well defined theoretical background (theory)

Structure of a Compiler



Lexical analysis

After having defined the alphabet to be used, the first things to do is to recognize words

This is a sentence

The lexical analysis divided the program text into words or “tokens”

```
position = initial + rate * 60
```

Lexical analysis

After having defined the alphabet to be used, the first things to do is to recognize words

This is a sentence

The lexical analysis divided the program text into words or “tokens”

```
position = initial + rate * 60
```

Syntax Analysis

After having understood the words we need to understand the sentence structure. Not so much different from the syntax of what we do for understanding natural language

This line includes a long sentence

Semantic Analysis

One the structure of the sentence is clear we need to understand the meaning:

- Humans can manage quite well this activity, the same is not so true for machines

Examples:

- Jack said Jerry left his assignment at home
- Jack said Jerry left his assignment at home?

- Compilers perform many semantic checks besides variable bindings.

Code Optimization

Not so important for natural language! It is now the most complex and effort prone activity in the construction of modern compilers

- The compilers modify the program so that it
  - runs faster
  - uses less memory
  - uses less power
  - makes less database accesses
  - uses less bandwidth
  - . . .

Code generation

Permits to produce assembly code to be run on the target machine

Proportions of the various phases changed from the pioneering era

Code generation

Permits to produce assembly code to be run on the target machine

Proportions of the various phases
changed from the pioneering era

Interesting Questions?

- Why are there so many programming languages?
  - Application domains have distinctive needs (scientific computing, business applications, system programming, etc.)
- Why are there new programming languages?
  - Cost of training programmers is the dominant cost for a programming language
  - productivity $>$ training cost?
- What is a good programming language?
  - Is it the one programmers use?

Interesting Questions?

- Why are there so many programming languages?
  - Application domains have distinctive needs (scientific computing, business applications, system programming, etc.)
- Why are there new programming languages?
  - Cost of training programmers is the dominant cost for a programming language
  - productivity $>$ training cost?
- What is a good programming language?
  - Is it the one programmers use?

Interesting Questions?

- Why are there so many programming languages?
  - Application domains have distinctive needs (scientific computing, business applications, system programming, etc.)
- Why are there new programming languages?
  - Cost of training programmers is the dominant cost for a programming language
  - productivity $>$ training cost?
- What is a good programming language?
  - Is it the one programmers use?