Introduction

The Structure of a Compiler

Linguaggi e Compilatori
Modulo Compilatori

Alessandra Di Pierro
alessandra.dipierro@univr.it

Text Book and ▸ Course Info

What is a compiler?

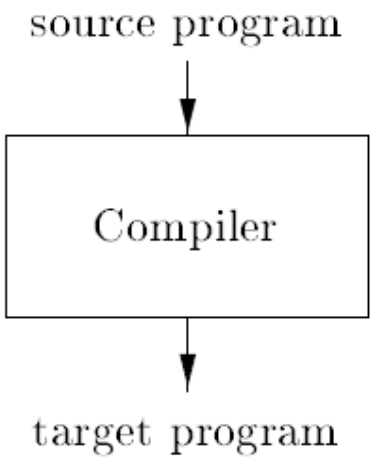


Figure 1.1: A compiler

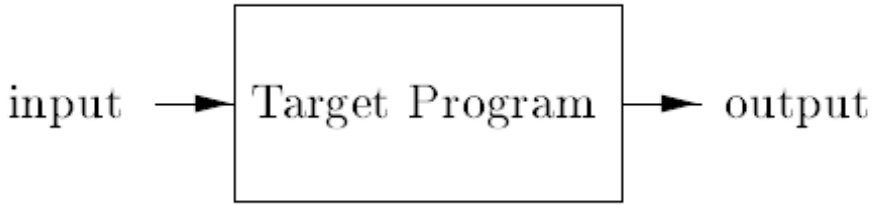


Figure 1.2: Running the target program

Interpreter

Another kind of language processing

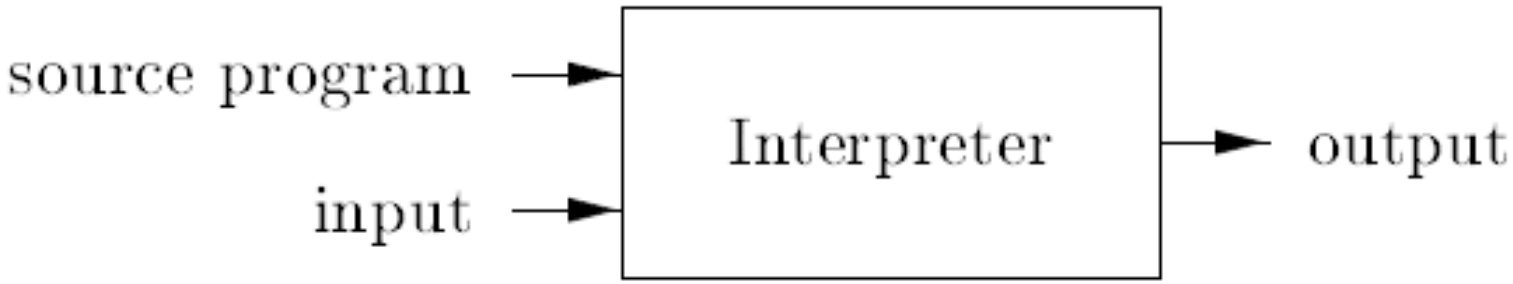


Figure 1.3: An interpreter

Hybrid Approaches

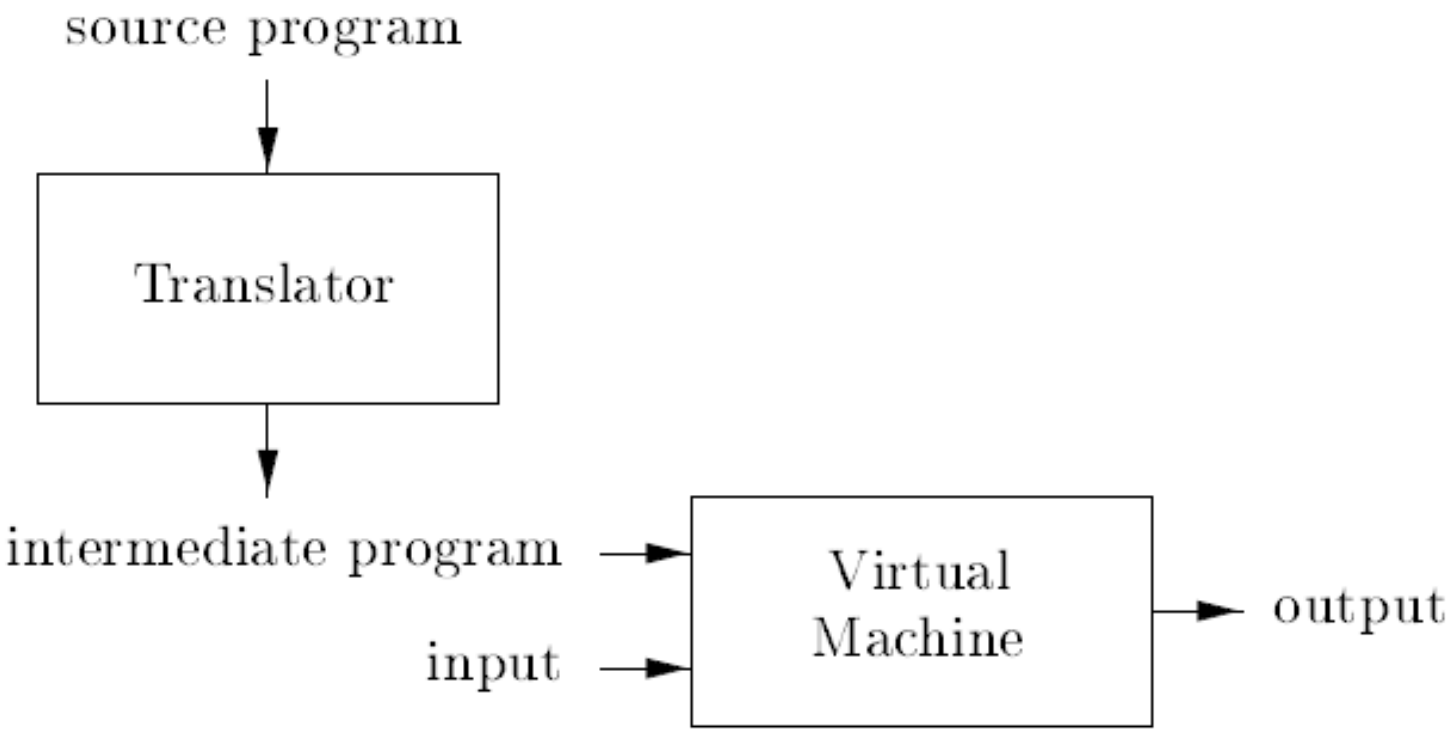


Figure 1.4: A hybrid compiler

Producing a machine code

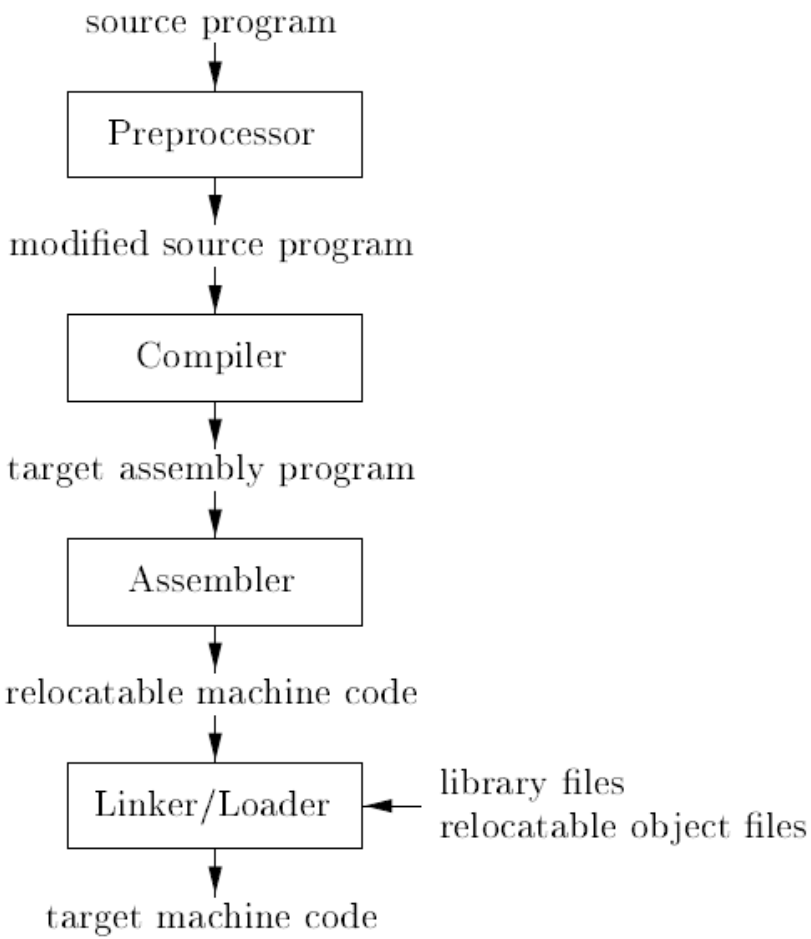


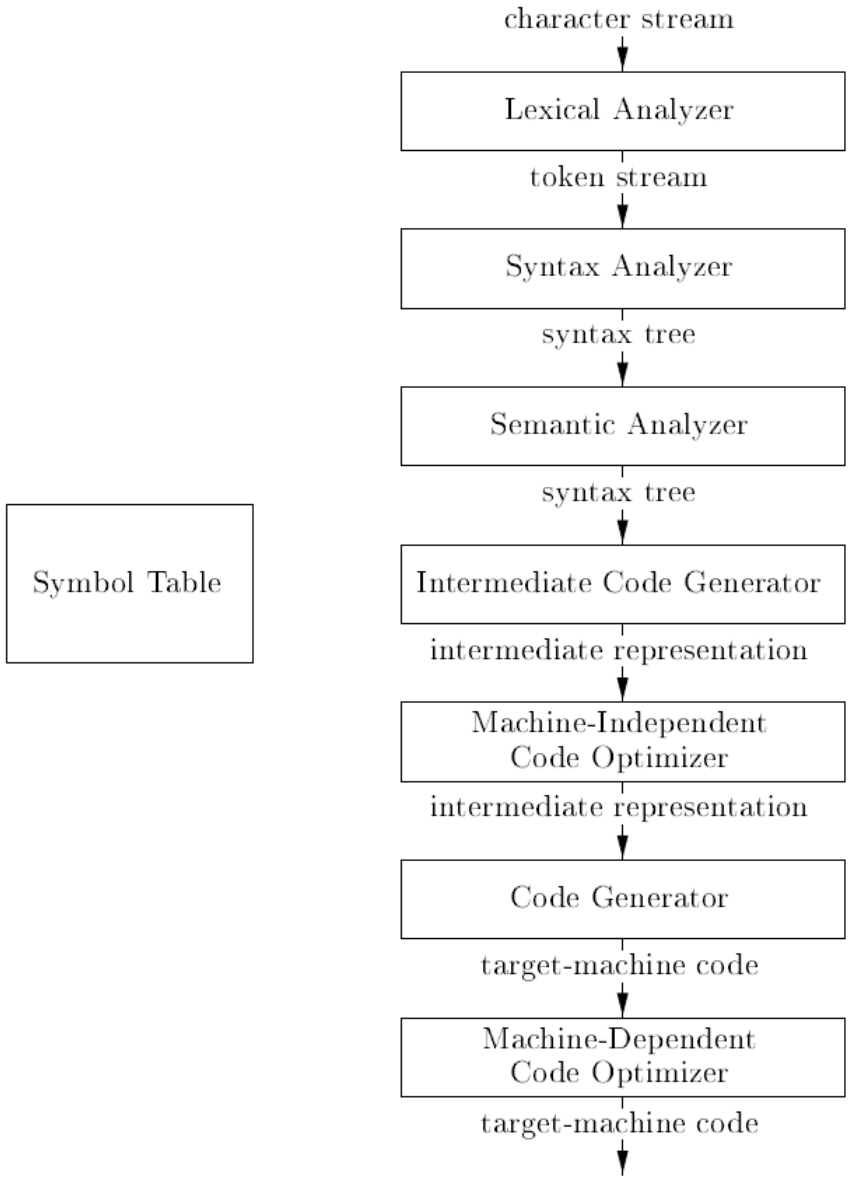
Figure 1.5: A language-processing system

Phases of a Compiler

- Analysis or *front-end*
- Synthesis or *back-end*

The symbol table stores information about the entire source program.
Maps variables into attributes, i.e. type, name, dimension, address, etc.
This information helps us detecting inconsistencies and misuses during type checking.

Compilation process



Compilation process

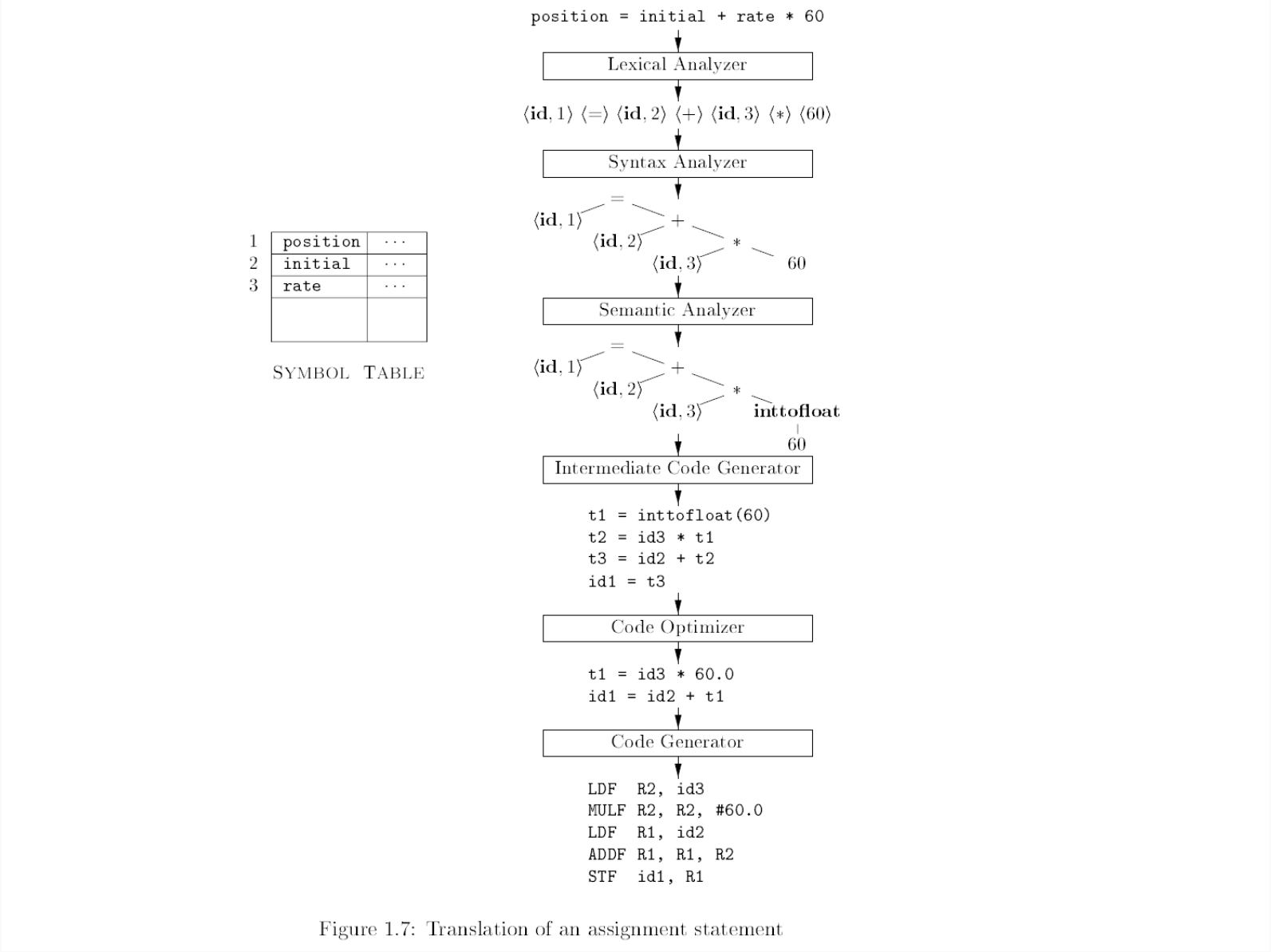


Figure 1.7: Translation of an assignment statement

Languages and Compilers

The design of programming languages is strongly related to the design of compilers.
Adding new language features places new demand to compilers writers

- **1940's** programs are sequences of 0's and 1's (first electronic computers)
- **Early 1950's** Assembly
- **Late 1950's** Fortran, Cobol, Lisp
- **1970's** C Language
- **1990's** C++, Java

Since 1940's computer architectures has evolved as well!

High-level Programming Languages

They define programming abstractions.

Compilers must translate programs to the target language.

Easier to write programs but the generated target programs run more slowly.

Need for optimisation

Example: the `register` keyword in C.

A Simple Example

Objective: to translate programs such as the following simple one:

```
{
      int i; int j; float[100] a; float v; float x;

      while ( true ) {
            do i = i+1; while ( a[i] < v );
            do j = j-1; while ( a[j] > v );
            if ( i >= j ) break;
            x = a[i]; a[i] = a[j]; a[j] = x;
      }
}
```

Figure 2.1: A code fragment to be translated

A Simple Example (ctd.)

The compiler front end (▸ source code) translates the program into the form:

```
 1:  i = i + 1
 2:  t1 = a [ i ]
 3:  if t1 < v goto 1
 4:  j = j - 1
 5:  t2 = a [ j ]
 6:  if t2 > v goto 4
 7:  ifFalse i >= j goto 9
 8:  goto 14
 9:  x = a [ i ]
10:  t3 = a [ j ]
11:  a [ i ] = t3
12:  a [ j ] = x
13:  goto 1
14:
```

Figure 2.2: Simplified intermediate code for the program fragment in Fig. 2.1