Intermediate Code Generation

Chapter 7

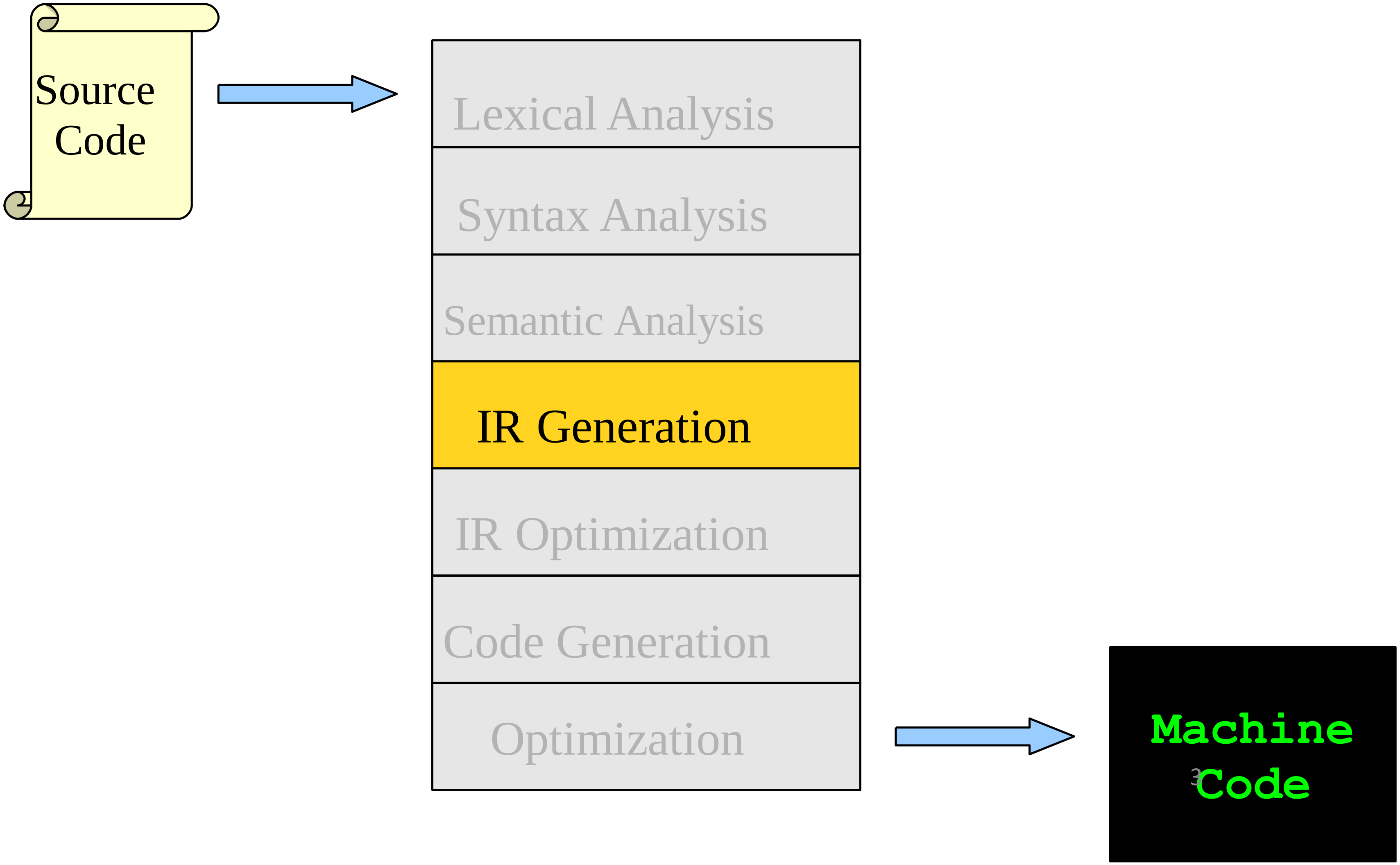
Overview of Code Generation

- The task of code generation is to generate executable code for a target machine that is a faithful representation of the semantics of the source code
- Code generation is typically broken into several steps

1) **Intermediate code generation**
2) Generate some form of assembly code
3) Optimization: To improve the speed and size of the target code

- We will talk about general techniques of code generation rather than present a detailed description for a particular target machine

Where We Are



Source Code

Lexical Analysis

Syntax Analysis

Semantic Analysis

IR Generation

IR Optimization

Code Generation

Optimization

Machine Code

Outline

- Intermediate code generation
  - Intermediate Code for Code Generation
  - Basic Code Generation Techniques
  - Code Generation of Control Statements and Logical Expressions

1 Intermediate Code for Code Generation

- Intermediate Representation (IR)
  - A data structure that represents the source program during translation is called an IR
  - For example: abstract syntax tree
- The need for intermediate code

  Abstract syntax tree does not resemble target code, particularly in its representation of control flow constructs
- Intermediate code

  Representation of the syntax tree in sequential form that more closely resembles target code

Three-Address Code

- Popular forms of intermediate code:
  - Three-address code
- The most basic instruction of three address code has the general form x=y op z which represents the evaluation of expressions
  - x,y,z are names, constants or compiler-generated temporary names
  - op stands for any arithmetic or logical operator, such as + , 'and'
  - "Three-address code" comes from this form of instruction, in general each of x,y and z represents an address in memory

Example: Computation of an expression is represented in three-address code

2*a+(b-3)

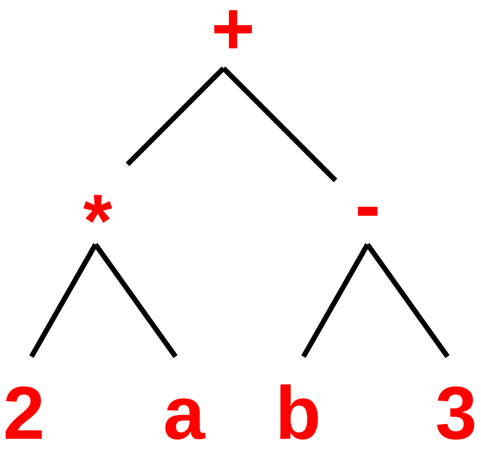
the corresponding

three-address code:

t1 = 2*a

t2 = b-3

t3 = t1+t2



where t1,t2,t3 are names for temporaries, they correspond to the interior nodes of the syntax tree and represent their computed values

Other instructions of three-address code

- Instructions of Three-address code for each construction of a standard programming language

1. Assignment statement has the form "x=y op z", where op is a binary operation
2. Assignment statement has the form "x=op y", where op is a unary operation
3. Copy statement has the form "x=y" where the value of y is assigned to x

4. The unconditional jump “goto L”
5. Conditional jumps ,such as “if B goto L” , “if_false B goto L”
6. Statement “Label L” represents the position of the jump address
7. “read x”
8. “write x”
9. Statement “halt” serves to mark the end of the code

Example

```
read x;
if 0<x then
        fact:=1;
        repeat
           fact:=fact*x;
           x:=x-1;
        until x=0;
        write fact
end
```

Three-address code for it

```
read x
_t1=0<x
if _false _t1 goto L1
fact=1
label L2
_t2=fact*x
fact= _t2
_t3=x-1
x=_t3
_t4=x==0
if_false _t4 goto L2
write fact
Label L1
halt
```