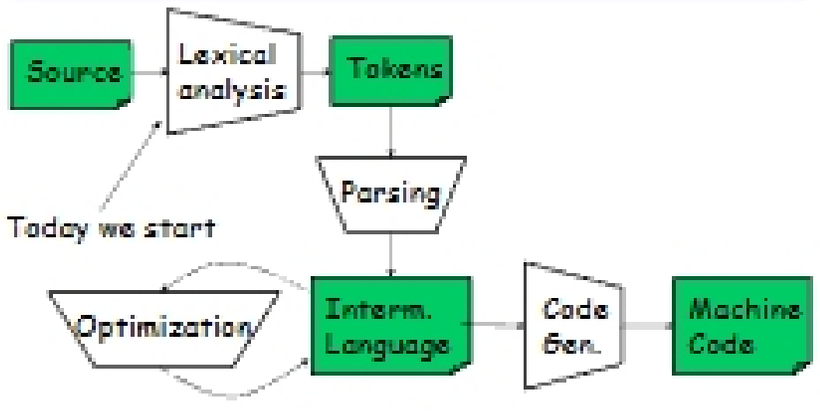
Lexical Analysis

The Structure of a Compiler



Lexical Analysis

- What do we want to do? Example:

```
if (i == j)
    z = 0;
else
    z = 1;
```

- The input is just a sequence of characters:

```
\tif (i == j)\n\t\tz = 0;\n\telse\n\t\tz = 1;
```

- Goal: Partition input string into substrings
  - And classify them according to their role

What's a Token?

- Output of lexical analysis is a stream of tokens
- A token is a syntactic category
  - In English:
    noun, verb, adjective, ..
  - In a programming language:
    Identifier, (Keyword) Integer, Float, IF, THEN ..
- Parser relies on the token distinctions:
  - E.g., identifiers are treated differently than keywords

Tokens

- A token corresponds to a string or a set of strings.
- Identifier: *strings of letters or digits, starting with a letter*
- Integer: *a non-empty string of digits*
- (Keyword) IF: string if
- Whitespace: *a non-empty sequence of blanks, newlines, and tabs*
- OpenPar: *a left-parenthesis*

Lexical Analyzer: Implementation

- An implementation must do two things:

1. Recognize substrings corresponding to tokens
2. Return the value or lexeme of the token
   - The lexeme is the substring

Example

- When reading the following:

  \tif (i == j)\n\t\tz = 0;\n\telse\n\t\tz = 1;

- Token-lexeme pairs returned by the lexer:
  - (Whitespace, "\t")
  - ((Keyword) IF, "if")
  - (LeftPar, "(")
  - (Identifier, "i")
  - (Relation, "==")
  - (Identifier, "j")
  - ...

Lexical Analyzer: Implementation

- The lexer usually discards "uninteresting" tokens that don't contribute to parsing.
- Examples: Whitespace, Comments
- Question: What happens if we remove all whitespace and all comments prior to lexing?

Lookahead.

- Two important points:
  1. The goal is to partition the string. This is implemented by reading left-to-right, recognizing one token at a time
  2. "Lookahead" may be required to decide where one token ends and the next token begins
  - Even our simple example has lookahead issues

    i vs. if

    = vs. ==