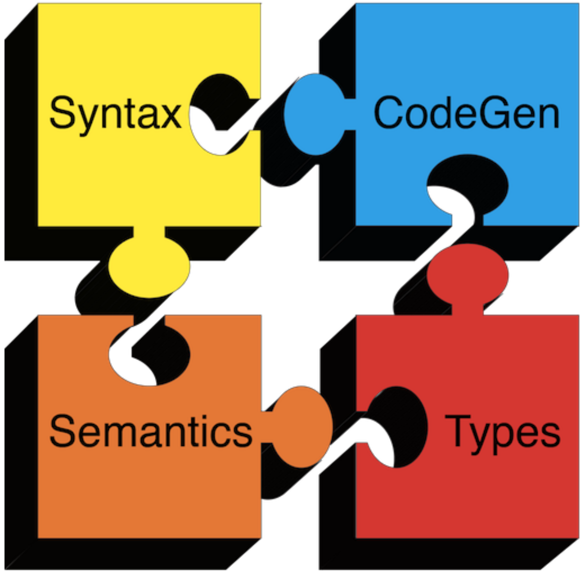

Compilers

Lexical Analysis

Alex Aiken

1. Lexical Analysis
2. Parsing
3. Semantic Analysis
4. Optimization
5. Code Generation

Lexical Analysis

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

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

Lexical Analysis

- Token Class (or Class)
  - In English:
  - In a programming language:

Lexical Analysis

- Token classes correspond to sets of strings.

- Identifier:
  - *strings of letters or digits, starting with a letter*
- Integer:
  - *a non-empty string of digits*
- Keyword:
  - *“else” or “if” or “begin” or …*
- Whitespace:
  - *a non-empty sequence of blanks, newlines, and tabs*

Lexical Analysis

- Classify program substrings according to role
- Communicate tokens to the parser

Lexical Analysis

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

Lexical Analysis

For the code fragment below,
choose the correct number of tokens in each class that appear in the code fragment

```
x = 0;\n\twhile (x < 10) {\n\tx++;\n}
```

- ○ W = 9; K = 1; I = 3; N = 2; O = 9
- ○ W = 11; K = 4; I = 0; N = 2; O = 9
- ○ W = 9; K = 4; I = 0; N = 3; O = 9
- ○ W = 11; K = 1; I = 3; N = 3; O = 9

W:	Whitespace
K:	Keyword
I:	Identifier
N:	Number
O:	Other Tokens: `{ } ( ) < ++ ; =`

Lexical Analysis

- An implementation must do two things:

1. Recognize substrings corresponding to tokens
    - The *lexemes*

2. Identify the token class of each lexeme