Struktur eines Compilers

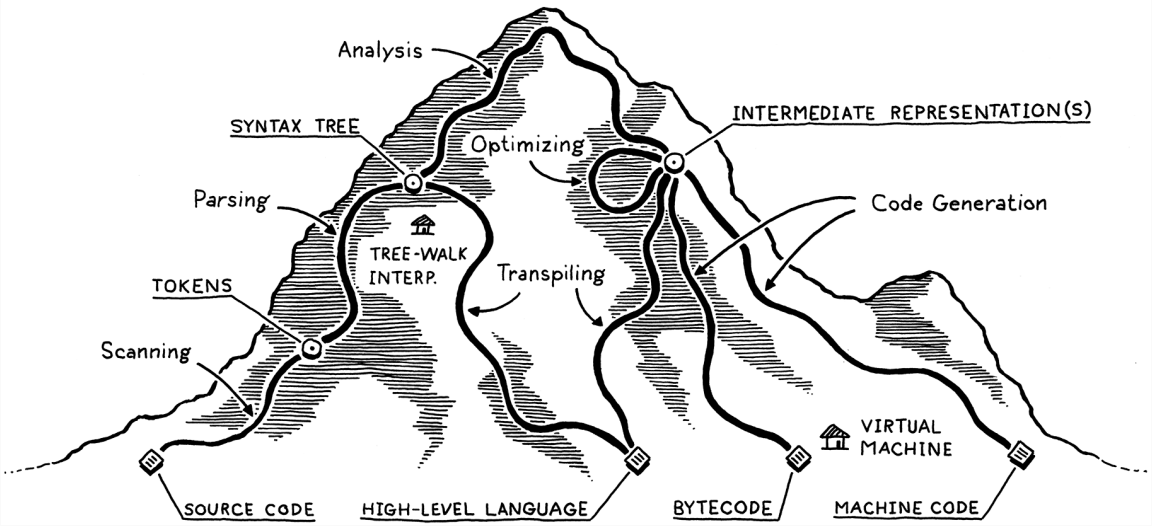
Carsten Gips (FH Bielefeld)

Sprachen verstehen, Texte transformieren

The cat runs quickly.

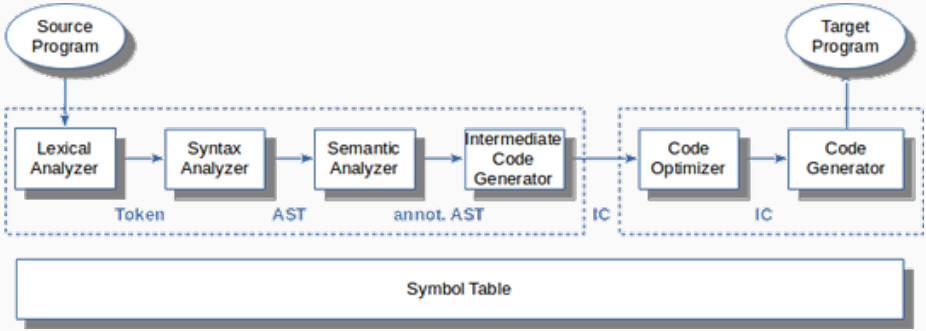
$=>$ Struktur? Bedeutung?

Compiler: Big Picture



Quelle: "A Map of the Territory (mountain.png)" by Bob Nystrom, licensed under MIT

Compiler: Prinzipieller Aufbau



Lexikalische Analyse: Wörter (“*Token*”) erkennen

```
sp = 100;
```

Lexikalische Analyse: Wörter (“*Token*”) erkennen

```
sp = 100;
```

```
<ID, sp>, <OP, =>, <INT, 100>, <SEM>
```

Syntaxanalyse: Sätze erkennen

```
<ID, sp>, <OP, =>, <INT, 100>, <SEM>
```

Syntaxanalyse: Sätze erkennen

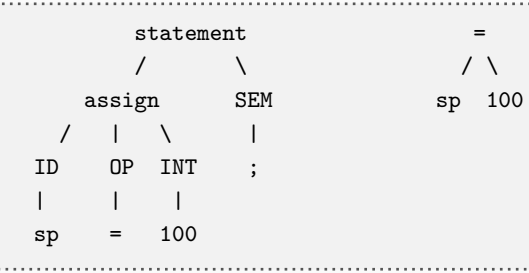
```
<ID, sp>, <OP, =>, <INT, 100>, <SEM>
```

```
statement : assign SEM ;
assign : ID OP INT ;
```

Syntaxanalyse: Sätze erkennen

```
<ID, sp>, <OP, =>, <INT, 100>, <SEM>
```

```
statement : assign SEM ;
assign : ID OP INT ;
```



Vorschau: Parser implementieren

```
stat : assign | ifstat | ... ;
assign : ID '=' expr ';' ;
```

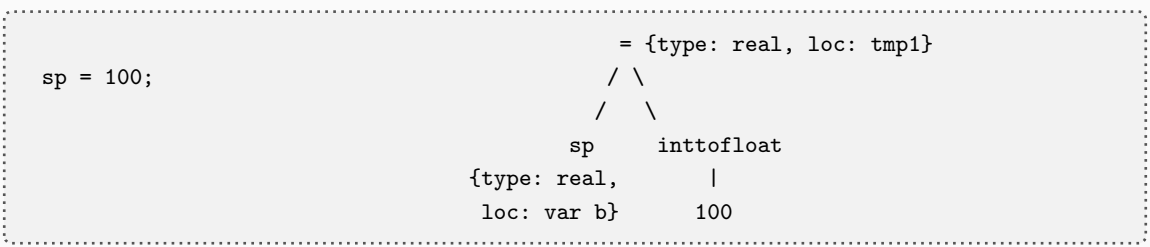
```
void stat() {
    switch (<<current token>>) {
        case ID : assign(); break;
        case IF : ifstat(); break;
        ...
        default : <<raise exception>>
    }
}
void assign() {
    match(ID);
    match('=');
    expr();
    match(';');
}
```

Semantische Analyse: Bedeutung erkennen

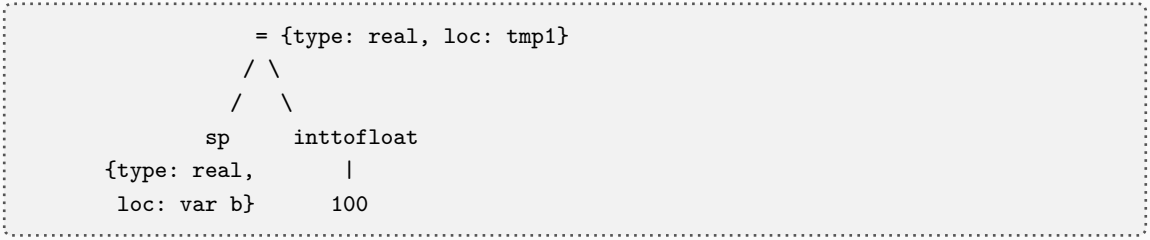
```
{
    int x = 42;
    {
        int x = 7;
        x += 3;     // ???
    }
}
```

Semantische Analyse: Bedeutung erkennen

```
{
    int x = 42;
    {
        int x = 7;
        x += 3;     // ???
    }
}
```



Zwischencode generieren



=> `t1 = inttofloat(100)`

Code optimieren

`t1 = inttofloat(100)` $\Rightarrow$ `t1 = 100.0`

`x = y*0;` $\Rightarrow$ `x = 0;`

Code generieren

- Maschinencode:

```
STD  t1, 100.0
```

- Andere Sprache:
  - Bytecode
  - C
  - ...

Probleme

```
5*4+3
```

AST?

Probleme

```
5*4+3
```

AST?

```
stat : expr ';'
     | ID '(' ')' ';'
     ;
expr : ID '(' ')'
     | INT
     ;
```

Wrap-Up

- Compiler übersetzen Text in ein anderes Format

- Typische Phasen:
  1. Lexikalische Analyse
  2. Syntaxanalyse
  3. Semantische Analyse
  4. Generierung von Zwischencode
  5. Optimierung des (Zwischen-) Codes
  6. Codegenerierung

LICENSE

Exceptions

- Figure “A Map of the Territory (mountain.png)” (https://github.com/munificent/craftinginterpreters/blob/master/site/image/a-map-of-the-territory/mountain.png), by Bob Nystrom, licensed under MIT