Compiling Techniques

Lecture 7: Coursework 1 - Intro
The Frontend

Scanner -> Tokenizer -> Parser -> Semantic Analyzer -> IR Generator

Lexer

Source

Errors
Coursework: A Python to RISC-V Compiler

**CW1 (30%)**
Parsing

**CW2 (30%)**
Semantic Analysis

**CW3 (40%)**
Code Generation

ChocoPy → AST → IR → RISC-V
# Coursework Schedule

<table>
<thead>
<tr>
<th>Week 1 (Jan 15)</th>
<th>Week 6 (Feb 26)</th>
<th>CW2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 2 (Jan 22)</td>
<td>Week 7 (Mar 4)</td>
<td></td>
</tr>
<tr>
<td>Week 3 (Jan 29)</td>
<td>Week 8 (Mar 11)</td>
<td></td>
</tr>
<tr>
<td>Week 4 (Feb 5)</td>
<td>Week 9 (Mar 18)</td>
<td>CW3</td>
</tr>
<tr>
<td>Week 5 (Feb 12)</td>
<td>Week 10 (Mar 25)</td>
<td></td>
</tr>
<tr>
<td>Learning Week</td>
<td>Week 11 (Apr 1)</td>
<td></td>
</tr>
</tbody>
</table>

**Deadlines:** **Friday noon**
Coursework

“Check out Learn” → “Compiling Techniques” → “Assessment”
A recursive-descent parser

CFG for function call

funcall ::= ID "(" arglist ")"
arglist ::= ID argrep | ε
argrep ::= "," ID argrep | ε

def parse_funcall():
    match(ID)
    match(LPAREN)
    parse_arglist()
    match(RPAREN)

def parse_arglist():
    if check(ID):
        match(ID)
        parse_argrep()

def parse_argrep():
    if check(COMMA):
        match(COMMA)
        match(ID)
        parse_argrep()
class Parser:

    def check(self, expected : TokenKind) -> bool:
        return self.lexer.peek().kind == expected

    def match(self, expected : TokenKind) -> Token:
        if self.check(expected):
            token = self.lexer.peek()
            self.lexer.consume()
            return token
        raise Exception(f"Error: token of kind ${expected} not found")
What is a token?

A token consists of a token class and other additional information.

Example: some token classes

- IDENTIFIER → foo, main, cnt, ...
- NUMBER → 0, −12, 1000, ...
- STRING_LITERAL → "Hello world!", "a", ...
- EQ → ==
- ASSIGN → =
- PLUS → +
- LPAR → (
- ...

class Token:
Kind: TokenKind
Value: Any = None