# My Notes

- Lexical analysis : Tokenisation
  Identifying "words". for each word, does it belong to the language? ~~f00~~ (~)

- Parsing: Syntax Analysis
  Grammatically correctly sentences
  prog $\in L(G_{python})$ ~~int for ( )~~

- Attribute Grammars : Semantic analysis

- IR
  - AST
  - 2/3 AC
  - SSA

- $$\begin{tikzcd} \text{Src} \arrow[r] & \text{AST} \arrow[r] & \text{Generic} \arrow[r] & \text{Gimple} \arrow[r] & \text{Tree opt} \arrow[d] \\ \text{RTL} \arrow[d] & \text{unSSA} \arrow[l] & \text{Tree SSA opt} \arrow[l] & & \text{SSA} \arrow[ll] \\ \text{RTL opt} \arrow[r] & \text{Code Gen} \arrow[r] & \text{Assembly.} & & \end{tikzcd}$$

(Register transfer Lang)

① Common Subexpr Elimination

$a = (x \times y) + z$
$b = (x \times y) + w$ } temp $= x \times y$
a = temp + z
b = temp + w

② Copy Propag^n

$a = x$
$b = a + y$ } $b = x + y$

③ Const Propagⁿ

a = 5
b = a+3 } b = 8

④ Dead Code Elimⁿ

a = 5    b = a+3
ret 10;    b = b×2;  } a = 5
ret 10;

⑤ Alias Analysis

*p = &x
*q = &x
*p = 10.
print(' ', *q) } *p = &x
*p = 10    q = p
print(  , *q)

⑥ Full redundancy elimⁿ

if (  ) x = a+b
else   x = a+b } x = a+b

⑦ Partial redundancy elimⁿ

if (    ) t = a+b
x = a+b

x = a+b
if (   ) t = x

— Loop removal

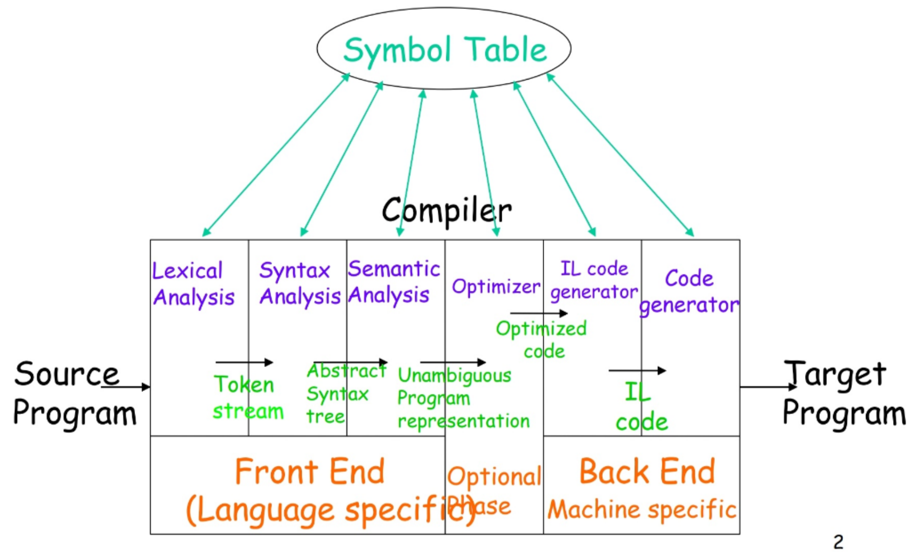
always terminates
is empty (has dead statements) or
directly computable (AP)

LIR to Assembly:

- Peephole Optimisⁿ
- Register Allocⁿ
- Code Generation

[illegible]

## Compiler structure



2

Cross Compiler: has impl lang $\neq$ target lang