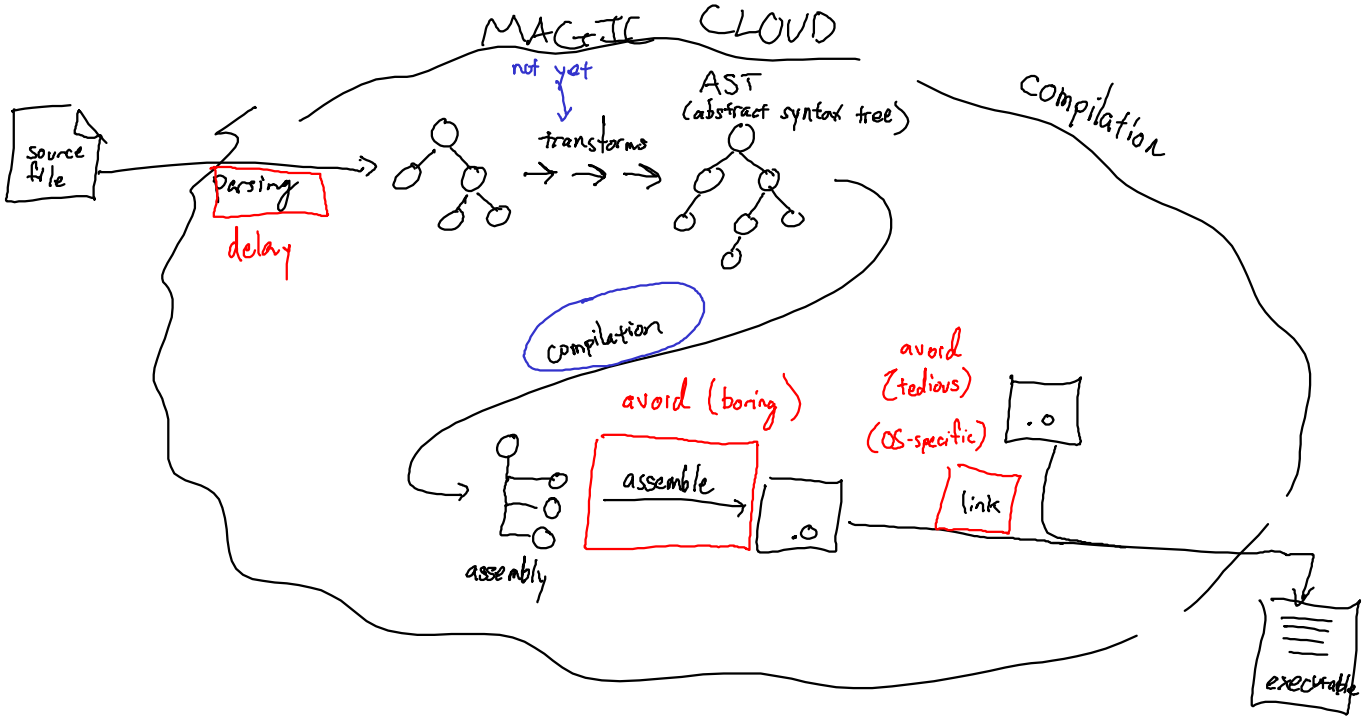

MAGIC CLOUD
not yet
AST
(abstract syntax tree)
transforms
source file
parsing
delay
compilation
avoid (boring)
avoid (tedious) (OS-specific)
assemble
.o
link
.o
assembly
executable
compilation

Auklet

1. Syntax

EBNF

<expr> ::= 0 | 1 | -1 | 2 | -2 | ...
| <expr> + <expr> | <expr> - <expr> | <expr> * <expr>
| after(<expr>) | before(<expr>)
| let <var> = <expr> in <expr> | <var>

	OCaml	English
	let x = 5;; ✓	I am standing here. ✓
Syntax error	let in let let;; ✗	am I I am I ✗
Semantic error	let x = "a" + 4;; ✗	More people have been to Russia than I have. ✗

2. Semantics

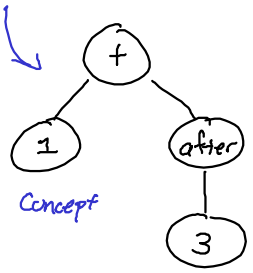
Expr		Result
1+2	⟹	3
5	⟹	5
let n=4 in n	⟹	4
let n=1+2 in n+3	⟹	6
after(5)	⟹	6
before(3)	⟹	2
after(x)	⇏	

```
type expr =
  | EInt of int
  | EPlus of expr * expr
  | EMinus of expr * expr
    ....
  | EAfter of expr
    ....
```

<expr> ::= 0 | 1 | -1 | 2 | -2 | ...
| <expr> + <expr> | <expr> - <expr> | <expr> * <expr>
| after(<expr>) | before(<expr>)
| let <var> = <expr> in <expr> | <var>

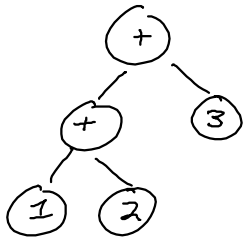
OCaml `1 + after(3)`

```
EPlus(EInt(1), EAfter(EInt(3)))
```



Concept

(1+2)+3



```
let rec compile_expression (e : expr) : instruction list =
  match e with
  | EInt(n) ->                                   mov rax, n
    [AsmMov(ArgRegister(RAX), ArgConstant n)]
  | EAfter(e') ->
    let instrs = compile_expression e' in
    instrs @                                     add rax, 1
      [AsmAdd(ArgRegister(RAX), ArgConstant 1)]
```

this function returns an instruction list which, when executed, will place in register RAX the value produced by evaluating the expression

AT&T	Intel
`mov $4, %rax`	`mov rax, 4`
`mov %rax, %rbx`	`mov rbx, rax`