1-1

compiler : program → ~~binary~~ ~~executable~~ ~~object~~ machine code + meta stuff

"gcc"

"javac"

program

~~CPP etc~~ CPP

gcc : C-program → x86-64 program

RTL → x86

→ arm

compiler — clang : CPP → C → LLVM → x86

javac : Java → JVM program

interpreter — Intel chip : x86-program → Answer / hardware

JVM : JVM prog → Answer / x86-program

C : functions

memory

local variables

strings

floats

ints

blocks

if, for etc

x86 : ints

floats

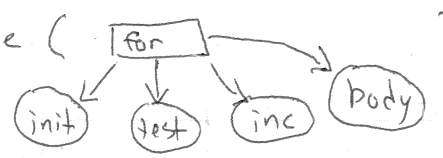
branches

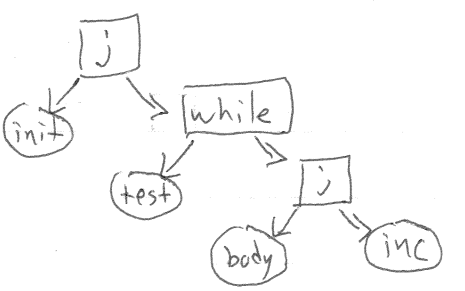
goto

memory

registers

~~C++~~ C → C - {for}

compile (

) = 

C → C - {-}

compile (X - Y) = X * (-1) * Y

local transformations

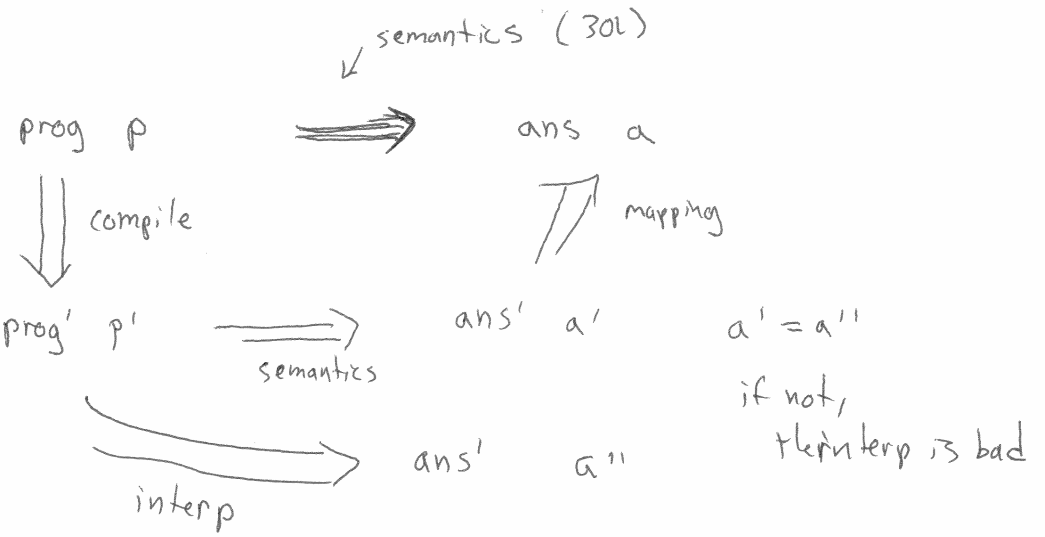
local variables vs registers

named places for values

infinite	finite
arbitary-sized	word-sized

compile("x = 29") = movq %r[redacted], $29

compiler (return x) = movq %rax, %r[redacted]
ret



Xavier Leroy : C to x86 (subset), Arm, RiscV
CompCert

Agrammar : e := int | (read) | (- e) | (+ e e)
| var | (let ([var e]) e)

Exp ⌜e⌝ := new Int(int) | new Read()
new Neg(⌜e⌝) | new Add(Exp, Exp)
new Var(string) | new Let(string, Exp, Exp)

p := (program e)