T = ... | (T ... -> T)

E = ... | (E E ...) (App (E, List<E>)) fun

P = (program D ... E) main | (function-ref x)

global funs

D = (define (id [id : Type] ...) : Type Expr)

name input return body

(define (double [x : int]) : int (+ x x)) : (int -> int)

[double : int -> int,
  chicken : int -> bool]

x = value ids
f = fun ids

Before: $\Gamma \vdash E : T$ $[x \mapsto T]$

After: $\Gamma, \Sigma \vdash E : T$ $[x \mapsto T]$ $[f \mapsto T]$

$$\frac{\Gamma[x] = T}{\Gamma, \Sigma \vdash x : T} \qquad \frac{\Sigma(f) = T}{\Gamma, \Sigma \vdash f : T}$$

$$\frac{\Gamma, \Sigma \vdash e_0 : T_1 \ldots T_n \Rightarrow T_r \qquad \Gamma, \Sigma \vdash e_i : T_i}{\Gamma, \Sigma \vdash (e_0 \ldots e_n) : T_r}$$

$$\frac{\begin{array}{l} \Sigma_D = [f \mapsto (T_{fa} \ldots \Rightarrow T_{fr}) \ldots] \\ \emptyset, \Sigma_D \vdash e_m : T_m \\ [a \mapsto T_{fa}, \ldots], \Sigma_D \vdash e_f : T_{fr} \end{array}}{\vdash (\text{program } (\text{define } (f\ [a : T_a] \ldots) : T_{ff}\ e_f) \ldots\ e_m) : T_m}$$

```
(let ([f (if (> (read) 0)
             add1      sub1)])
  (f 27))
```

⇓

```
...
(function-ref add))
(fun-ref sub1)
```

(select-inst/arg normal (var x)) fun

(var x) leaq f(%rip), dest

read : callq _read_int

(f a b) fun : callq * % reg

15-2

$$(f \quad \underset{\text{rdi}}{a_0} \quad \underset{\text{rsi}}{a_1} \quad \underset{\text{rdx}}{a_2} \quad \underset{\text{rcx}}{a_3} \quad \underset{\text{r8}}{a_4} \quad \underset{\text{r9}}{a_5} \quad \underbrace{a_6 \quad a_7 \quad a_8}_{\text{stack}})$$

Caller	Callee	
8 (%rbp)		return address
0 (%rbp)		old %rbp
-8 (%rbp)		local var 1
...		
-8k (%rbp)		local var k — caller saves
8n-8 (%rsp)	8n+8 (%rbp)	argument n
...	...	...
0 (%rsp)	16 (%rbp)	argument 1
	8 (%rbp)	return address (caller)
	0 (%rbp)	caller's %rbp — callee saves
	-8 (%rbp)	callee's local 0
	...	...

fun:

```
pushq rbp
subq 8*var, rsp
movq rdi, a0
...
movq r9, a5
movq 8n+8(%rbp), r(6+n)
...
body
movq ans, rax
//restore (pops)
pop rsp
return
```

save callee-saves registers (pushq each one)

$(f \quad a_0 \quad a_1)$

eval f $\rightarrow A_f, I_f$

eval a_0 $\rightarrow A_0, I_0$

eval a_1 $\rightarrow A_1, I_1$

$I_f \ldots I_0 \ldots I_1 \ldots$

save caller saves pushing...

movq args into place * complicated

call

movq rax lhs

restore saves

15-3/

Everything that you used to do for the main expression, you must per-function:

- compute vars & types
- do register alloc (liveness)
- set aside local space
- NEW TASK: worry about the root stack

- save caller regs (if not a root)
  move args into place
  push all root regs onto root stack
  do call
  restore caller regs (if not a root) ← reads a root from the root stack