

$T = \dots \mid (T \dots \rightarrow T)$

$E = \dots \mid (E \ E \ \dots) \quad (\text{App}(\overset{\text{fn}}{\downarrow} E, \text{List}\langle E \rangle))$

$P = (\text{program} \quad \underbrace{D \dots}_{\text{global fns}} \quad \underbrace{E}_{\text{main}}) \mid (\text{function-ref } x)$

$D = (\text{define } (\underbrace{\text{id}}_{\text{name}} \quad \underbrace{[ \text{id} : \text{Type} ]}_{\text{input}} \dots) : \text{Type} \quad \underbrace{\text{Expr}}_{\text{body}})$

$(\text{define } (\text{double } [x : \text{int}]) : \text{int} \quad (+ \ x \ x)) : (\text{int} \rightarrow \text{int})$

$\left[ \begin{array}{l} \text{double} : \text{int} \rightarrow \text{int}, \\ \text{chicken} : \text{int} \rightarrow \text{bool} \end{array} \right]$

$x = \text{value ids}$   
 $f = \text{fun ids}$

$\rightarrow [x \mapsto T]$

$\rightarrow [x \mapsto T] \rightarrow [f \mapsto T]$

Before:  $\Gamma \vdash E : T$

After:  $\Gamma, \Sigma \vdash E : T$

$\Gamma[x] = T$

$\Sigma(f) = T$

$\Gamma, \Sigma \vdash x : T$

$\Gamma, \Sigma \vdash f : T$

$\Gamma, \Sigma \vdash e_0 : T_1 \dots T_n \rightarrow T_r$

$\Sigma_0 = [f \mapsto (T_{fa} \dots \rightarrow T_{fr}) \dots]$

$\Gamma, \Sigma \vdash e_i : T_i$

$\emptyset, \Sigma_0 \vdash e_m : T_m$

$\Gamma, \Sigma \vdash (e_0 \dots e_n) : T_r$

~~def~~  $[a \mapsto T_{fa}, \dots], \Sigma_0 \vdash e_i : T_{fr}$

$\vdash (\text{program } (\text{define } (f \ [a : T_{fa}] \dots) : T_{ff} \quad e_f) \dots e_m) : T_m$

$(\text{let } ([f \quad (\text{if } (> (\text{read}) \ 0) \text{ add } 1 \quad \text{sub } 1)]))$   
 $(f \ 27))$



$(\text{select -inst / arg} \quad \text{normal}(\text{var } x))$

$(\text{function-ref add})$   
 $(\text{fun-ref sub } 1)$

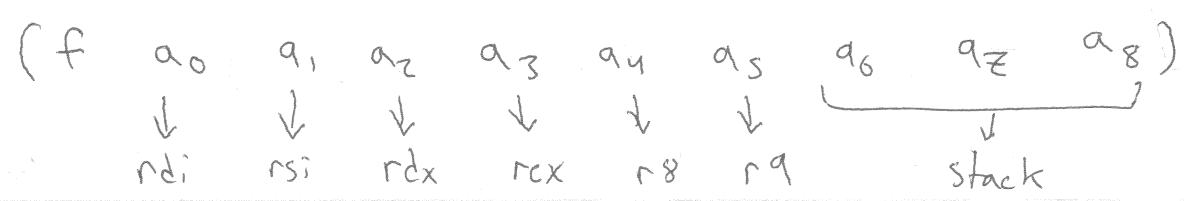
$(\text{var } x)$

$\text{leaq } f(\%rip), \text{dest}$

$(f \ a \ b)$

$\text{read} : \text{callq} - \text{read-int}$   
 $\text{fun} : \text{callq} * \% \text{reg}$

15-2/



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 a<sub>0</sub> a<sub>1</sub>)

eval f → Af, If

eval a<sub>0</sub> → A<sub>0</sub>, I<sub>0</sub>

eval a<sub>1</sub> → A<sub>1</sub>, I<sub>1</sub>

If ... I<sub>0</sub> ... I<sub>1</sub> ...

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 ← reads a root from ~~the~~ root stack  
(if not a root)

