

-1)

select : C \rightarrow X (w/ vars and break some x86 rules)

select_p (program : [BODY \rightarrow +]) = program : [BODY \rightarrow (block \emptyset (select₊ +))]

select₊ (return a) := [movq (select_a a) RAX ;
(jmp END)]

select₊ (seg s +) := select_s s ++ select₊ + c x
↓ ↓
select_a (num n) = (num)
(var x) = (var x)

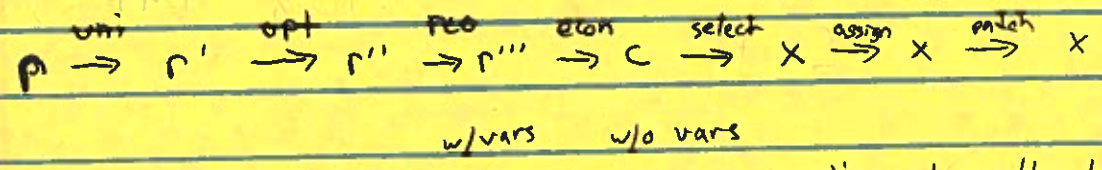
select_s (set! x e) := ~~movq (f e) x~~
select_e x e
↑
(select_a x)

select_e dst a := [movq (select_a a) dst]

select_e dst (read) := [callq read-int ; movq RAX dst]

(- a) := [movq (select_a a) dst ; negq dst]

(+ a_L a_R) := [movq (select_a a_R) dst ; addq (select_a a_L) dst]



assign-holmes : X \rightarrow X "register allocation" must be div by 16

assign (program ip [BODY \rightarrow (block ib ~~IS~~)]) VC = ~~8~~ 8 * (n or n+1)
local-vars = (x₁ ... x_n)

(program (ip / local-vars) [
BEGIN \rightarrow (block \emptyset [pushq RBP ; movq RSP RBP ;
subq VC, RSP ; jmp BODY])
END \rightarrow (block \emptyset [addq VC, RSP ; popq RBP ;
retq])

BODY \rightarrow (block ib (assign σ IS))]

$\sigma = [x_1 \rightarrow \%RSP(8 \times 1) , \dots , x_n \rightarrow \%RSP(8 \times n)]$

4-2/ assigns σ $[i] = [i]$

$(i:is) = (\text{assign } \sigma, i) : (\text{assign } \sigma \text{ is})$

$\text{assign } \sigma (\text{addg } a_1, a_2) = \text{addg } (\text{assign } \sigma a_1) (\text{assign } \sigma a_2)$

$(\text{addg } a_2) = \text{addg } (\text{assign } \sigma a_2)$

$(\text{movg } a_1, a_2) = \text{movg } (a_1) (a_2)$

$\text{jump LAB} = \text{jump LAB}$

$(\text{callg LAB}) = \text{callg LAB}$

$\text{assign } \sigma (\text{num } n) = (\text{num } n)$

$(\text{var } x) = \sigma(x)$

patch recurs like assign

$\text{TMP} = \text{RAX}$

$\text{patch } (\text{addg } R_1(O_1), R_2(O_2)) = [\text{movg } R_1(O_1), \text{TMP-REG} ; \text{addg } \text{TMP-REG}, R_2(O_2)]$

$(\text{movg } R_1(O_1), R_2(O_2)) = [\text{movg } R_1(O_1), \text{TMP} ; \text{movg } \text{TMP}, R_2(O_2)]$

$i = [i]$

$\text{patch } (i:is) = \text{patch } i ++ \text{patch } is$

runtime.c int64_t

$\text{int read-int } () \{ \text{int } x; \text{scanf}("%d", \&x); \text{return } x; \}$

$\text{int print-int } (\text{int } x) \{ \text{printf}("%d", x); \text{return } 0; \}$

main : $X \rightarrow X$

ans is in RAX
ans is printed
& can be called

main (program ; blks)

$= (\text{program } i \text{ blks } +$

.. print x to x.s ..

[main \rightarrow (block σ

\$ cc runtime.c x.s -o x.bin

./x.bin

[callg BEGIN ;
movg RAX, RDI ;
callg -print-int ;
retg]])

... read output ... turn into number ...

... compare w/ expected ...