-1/

87

1 prog   1 fun (locals)   N labels   1 tail — now

1 prog   N fun (locals)   M labels   1 tail — eventually

---

select$_t$ (return a) := movq (select$_a$ a) %rax
jmp END

select$_t$ (seq s t) := select$_s$ s ++ select$_t$ t

select$_t$ (goto lab) := jmp lab

select$_t$ (goto-if (cmp l r) lab$_t$ lab$_f$) :=
cmpq (select$_a$ r) (select$_a$ l) ;
jmp cc lab$_t$; where cc = match cmp
jmp lab$_f$

match cmp	
=	=> e
<	=> lt
≤	=> le

select$_a$ (bool b) = if b then \$1 o.w. \$0

select$_e^{dst}$ (not a) = movq (select$_a$ a) dst ;
xorq \$1, dst

select$_e$ dst (cmp l r) = cmpq (select$_a$ r), (select$_a$ l);
set cc %al
movzbq %al, dst

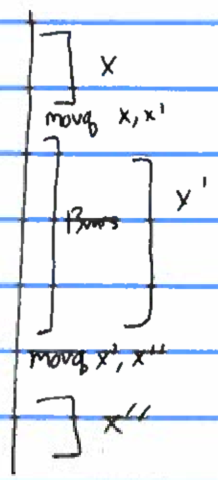
---

type p = can I type it? ($\emptyset \vdash p : T$?)
(p = (program : e))
└ yes ⟶ return $\emptyset$ (program : [type → T] e)
o.w ⟶ error

9-2

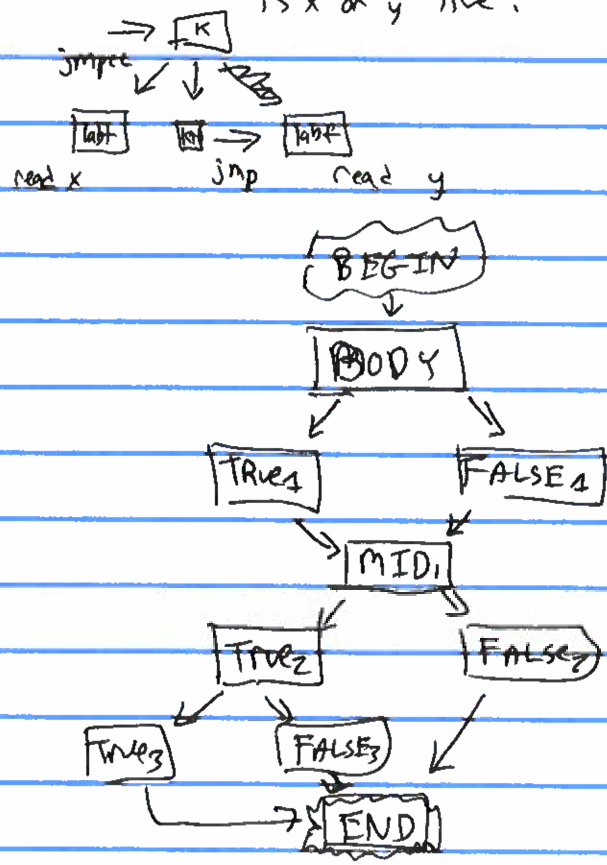
$$\text{live-after}_k = (\text{live-before}_{k+1} - W(k)) \cup R(k)$$

fun



range splitting

is x or y live?



$$\frac{\Gamma \vdash e_c : Bool \quad \Gamma \vdash e_t : T \quad \Gamma \vdash e_f : T}{\Gamma \vdash \text{if } e_c\ e_t\ e_f : T}$$

memoization

fib n =
 if n ≤ 1 then
 1
 o.w. (fib (n-1)) + (fib (n-2))

mfib n =
 if HISTORY[n] then ret H[n] o.w. H[n] = fib(n) ; ret H[n]

tfib n = History = new array (n-1) , fib (n)

live_p (xprogram ; l→b) = (live_e l→b ∅ BODY) = (MEM, la) ret (xprog ; MEM

live_e ALL MEM lab = if MEM[lab] then (MEM, MEM[lab])
 o.w. lab = END then (MEM, mt)
 o.w. (blk ; I) = ALL[lab]
 (MEM', live-after) = live_{is} ALL MEM I
 (MEM'[lab ↦ (blk ; [live ↦ live-after])), live-after)

-3

live$_{is}$ ALL MEM mt = (MEM, mt)

live$_{is}$ ALL MEM (f::r) =

(MEM', la) = (live$_{is}$ ALL MEM r)

case f with

(jmp lab) or (jmpcc lab) =>

(MEM'', la_{lab}) = (live$_e$ ALL MEM' lab)

(la_{lab0} :: _) = la_{lab} (la_0, la_r) = la

(MEM'', (la_{lab0} ∪ la_0) :: la_r

o.w. =>

(MEM', (la_0 − W(f)) ∪ R(f) :: la)

[x, y, z] = x :: y :: z :: mt l_1 ++ l_2

list, cons, empty, append

---

conflicts (cmpq, jmp, jmp-if) don't contrib (like retq

patch cmpq cannot have constant in 2nd pos

main callq BODY
movq %rax, %rdi
callq _print_int, if i[type] = S64 then _print_int o.w. _print_bool)
movq 0, %rax
retq

9-4/ conditional move

X.i = ... | cmovq cc s, d

```
    cmpq  l, r
    jmpcc labt
    jmp   labf
labt: movq  A, dst
    jmp   mid
labf: movq  B, dst
    jmp   mid
mid:
```

⇒

```
cmpq  l, r
movq  B, dst
cmovq cc A, dst
```

C.e = ... | if op l r t f

econe (let x := if (cmp l r) t f in b)

if already reg?