10-1

uncover-locals

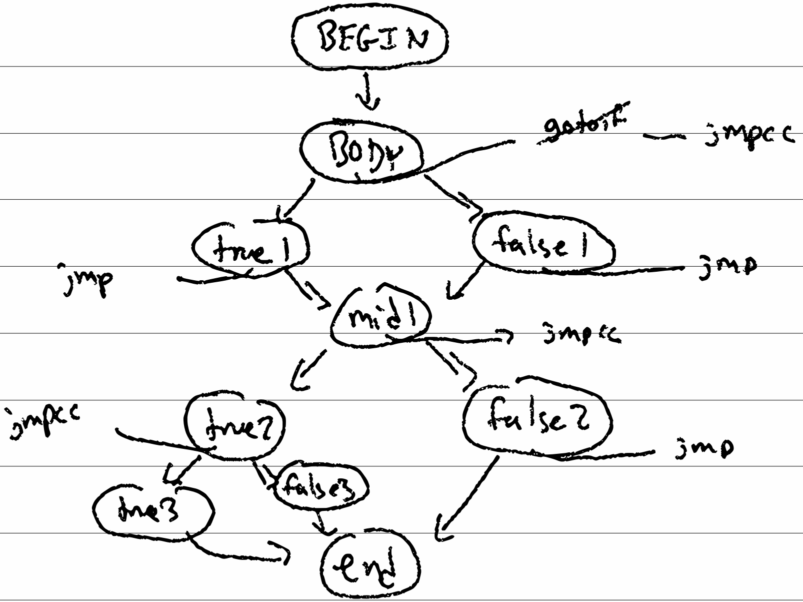
```
selectt   (return a)   =   movq  (selecta a)  %rax
                           jmp  END
selectt  (seq s t)   =   selects  s   ++   selectt  t
selectt  (goto lab)  =   jmp  lab
select  (goto-if  (cmp  aL  aR)   labt   labf)  =
      cmpq  (selecta aR),  (selecta aL)
      jmp cc   labt              cc =    = ~> e    < ~> l
      jmp      labf


selecta   (Bool  b)  =  if   b  then   $1    or    $0


selecte   dst   (not  a)  =   movq  (selecta  a)  dst
                              xorq   $1,  dst
selecte   dst  (cmp al  ar)  =
      cmpq  (select  ar),  (select  al)
      setcc   %al                      cc =   ≤ ~> le   > ~> g
      movzbq   %al,  dst
```

10-21

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



10-3 /

```
livep (program _ L) = after
   where (M', after) = live_ℓ L ∅ BODY

live_ℓ L M lab =
   if M[lab] then (M, M[lab])
   o.w. lab = END then (M, {%rax})
   o.w. (M'[lab ↦ after], after)
      where (M', after) = live_is L M L(lab)


live_is L M [] = (M, ∅)
live_is L M (f :: r) = live_i L M' f after
   where (M', after) = live_is L M r
```

10-4

```
live_i  L  M'  f  after =
  if f = (jmp lab) or (jmpcc lab) then
     (M'', after ∪ after')
     where (M'', after') = live_l L M' lab
  o.w.
     (M', (after \ W(f)) ∪ R(f))
```

---

cmpq, jmp, jmpcc don't have any interferences

(byte-reg r) is like r

xorq is like addq

patch ... cmpq can't have a constant in 2nd pos
(can't have 2 mem refs)

```
main :  callq BODY
        movq %rax, %rdi
        callq _print_int  <=>  if type is S64 ... print_int
                                  o.w.  print_bool
        movq $0, %rax
        retq
```

10-5)

X_i := .... | cmovq cc s, d

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

=>

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

⇑

Ce = ... | if op l r t f

↑↗ a ↗↗

econ$_e$ (let x := if (cmp l r) t f m))

↗ ↗

are already in rcoa-form