4-1)

uncover-locals : C → C (knows vars)

before info: ∅ after info: set of vars

{x, y, z, a, b, c}

uncover (program info [-main ⇒ +])

= (program info' [-main ⇒ +])

where info' = {x_0, ..., x_n}

if t = (seq (set! x_0 _) ... (set! x_n _) a)

4-2

select-instrs : C ⇒ X (has vars and breaks some rules)

select (program info [main ⇒ t])
= (program info [main ⇒ (block ∅ (select_t t))])

select_t (return a) = [(movq (select_a a) RAX); (jmp END)]
(seq s t) = select_s s ++ select_t t

select_s (set! x e) = select_e (select_a x) e

select_a (num n) = (num n) select_a (var v) = (var v)

select_e dst (Arg a) = [movq (select_a a), dst]
(Read) = [callq _read_int ; movq RAX dst]
(Neg a) = [movq (select_a a) dst ; negq dst]
Add a_L a_R = [movq (select_a a_R) dst; addq (select a_L) dst]

4-3/

uni: r → r opt → r rco → r econ → c uncover → c select → x assign → x patch → x

assign-homes : X (w/vars) ⇒ X (w/o vars)

assign (program (var {x1 ... xn}) [main ⇒ (block 0 is)])

let how-many-vars = n

stack-space = 8x(n or n+1) in

(program 0 [

Begin ⇒ [pushq RBP; movq RSP ⇒ RBP;
subq SS, RSP; jmp body]

end ⇒ [addq SS, RSP; popq RBP; retq]

body ⇒ [assign σ is) σ = [Xi ↦ %RBP(8xi)]

4-4/

assign σ [] = [] assign σ (i :is) = a σ i : assign σ is

assign σ (addq aL, aR) = addq (assign σ aL), (assign σ aR)

(negq a) = negq (assign σ a)

assign σ (num n) = (num n) (reg r) = (reg r)

(var v) = σ(v)

4-5/ patch : x → x

patch_is (cons i is) = patch i i ++ patch_is is

patch (addq R1(O1) R2(O2)) =
  [movq R1(O1), tmp-reg ;
    addq tmp, R2(O2)]
(movq R1(O1) R2(O2) =
  [movq R1(O1), tmp ;
    movq tmp, R2(O2)]
i = [i]

4-6 / runtime.c

```
int read_int() { int x; scanf("%d", &x); return x; }
void print_int(x) { printf("%d", x); return 0; }
```

main : X -> X

```
(program _ blks) = (program _ (blks ++
    [_main ↦ (block ∅
        [ callq BEGIN;
          movq RAX, RDI;
          callq _print_int
          retq ] ] }
```

4-7/

```
test-on-real-hardware : X -> Nums
    write (emit x) "x.s"
    exec "cc runtime.c x.s -o x.bin"
    exec "./x.bin" -> ans-str
    let ans = str->nums ans-str
    return ans
```

50!