Lec 11 SSA

SSA

- Every var / temp assigned at most once
- Like functional prog. & math
- Makes optimisation easier
  - ex `i ← 0` …] if SSA, i not changed here
    `if (i > 0) then cont else done` ⇝ `goto done`

Transforming basic block

Recall basic block : - starts with label
- ends with jump
- no jump / label in middle

Use generation counter.

Consider

```
dist (x0, y0) :
  x1 ← x0 * x0
  y1 ← y0 * y0          ← Works fine
  t0 ← x1 + y1
  t1 ← isqrt (t0)
  return t1
```

```
pow(b0, e0) :
  r0 ← 1
  goto loop
loop:
  if (e0 > 0) then done else body
body:                                  Problem: at next iter we use wrong r and e
  r1 ← r0 * b0
  e1 ← e0 - 1
  goto loop
done:
  return r0
```

Idea : use parameters in each block

```
pow (b0, e0)
  r0 ← 1
  goto loop (b0, e0, r0)
loop (b1, e1, r1)
  if (e1 > 0) then done (b1, e1, r1) else body (b1, e1, r1)
body (b2, e2, r2)
  r3 ← r2 * b2
  e3 ← e2 - 1
  goto loop (b3, e3, r3)
done (b4, e4, r4)
  return r4
```

Minimal SSA : each var / tmp only used in one basic block
⇝ Optimisation can operate on individual basic block

SSA & FP

- SSA progs can be viewed as functional prog
  - only tail calls
  - no HOFs
  - no shadowing

Above :

```
let pow (b0, e0) = let r0 = 1 in loop (r0, b0, e0)
let loop (b1, e1, r1) = if e > 0 then done (b1, e1, r1)
                        else body (b1, e1, r1)
let body (b2, e2, r2) = let r3 = r2 * r2 in
                        let e3 = e2 - 1 in
                        loop (r3, e3, b2)
let done (b4, e4, r4) = r4
```

Steps

1. Convert to minimal SSA
2. Determine & remove unnecessary parameters
   - Blocks called only at one place ⇝
   - Params that are not live
   - Params whose number's not changed within block, just passed through from arg to goto, viz. not updated ⇝

```
pow (b0, e0)
  r0 ← 1
  goto loop (b0, e0, r0)
loop (b1, e1, r1)
  if (e1 > 0) then done (b1, e1, r1) else body (b1, e1, r1)
body (b2, e2, r2)
  r3 ← r2 * b0
  e3 ← e2 - 1
  goto loop (b2, e3, r3)
done (b3, e4, r4)
  return r1
```

In general, x_i in label $l(\ldots, x_i, \ldots)$ can be removed if $\exists k$ s.t. all jumps to l have form goto $l(\ldots, x_k, \ldots)$ or goto $l(\ldots, x_i, \ldots)$. Then replace x_i with x_k

3. Code gen — move things into args before jumping ⇝

```
pow :
  r0 ← 1
  e1 ← e0
  r1 ← r0
  goto loop
```

ϕ function

- organise call / jumps differently
- for each var, list all possible temps

loop block :

$$e_1 = \begin{cases} e_0 & \text{if from pow} \\ e_3 & \text{if from body} \end{cases}$$

⇝

```
loop :
  e1 = ϕ (e0, e3)
  r1 = ϕ (r0, r3)      ← useful for liveness & SSA minimisation
  if (e1 > 0) then done else body
```