Common Subexpression Elimination

what is CSE?

- Goal: remove redundant computation, don't calculate the same expression multiple times

```
1: A = B * C
2: E = B * C
```

Keep the result of statement 1 in a temporary and reuse for statement 2

- Difficulty: how do we know when the same expression will produce the same result?

```
1: A = B * C
2: B = <new value>
3: E = B * C
```

Expression is no longer redundant, because computing it again will give a different result

- This becomes harder with pointers (how do we know when B is killed?)

key idea

- Idea: keep track of which expressions are "**available**" during execution of the program
- An expression is **available** if:
  - I am guaranteed to have computed it before (it is redundant)
  - If I recomputed the expression now, it will give the same result
- Issue: determining when an expression *becomes* available
  - This happens whenever an expression is computed
- Issue: determining when an expression is *no longer* available
  - This happens when one of its components is assigned to, or "killed."

why is CSE effective?

- Lots of redundancy in real code!
- Operations may not look redundant at the source code level, but result in redundancy at the 3AC level:

```
A[i] = A[i] + 1
```

```
T1 = &A
T2 = 4 * i
T3 = T1 + T2
T4 = *T3
T5 = &A
T6 = 4 * i
T7 = T5 + T6
T8 = T4 + 1
*T7 = T8
```

two varieties

- Local: within a single basic block
    - Easier problem to solve
- Global: within a single procedure or across the whole program
    - Intra- vs. inter-procedural
    - More powerful, but harder
    - Will come back to these sorts of “global” optimizations later

Maintaining available expressions

- For each 3AC operation in a basic block
  - Create name for expression (based on lexical representation)
  - If name not in available expression set, generate code, add it to set
    - Track register that holds result of and any variables used to compute expression
  - If name in available expression set, generate move instruction
  - If operation assigns to a variable, kill all dependent expressions