# Partial Redundancy Elimination

This lecture is primarily based on Konstantinos Sagonas set of slides (Advanced Compiler Techniques, (2AD518) at Uppsala University, January-February 2004). Used with kind permission. (In turn based on Keith Cooper's slides)



## Common-Subexpression Elimination

An occurrence of an expression in a program is a common subexpression if there is another occurrence of the expression whose evaluation always precedes this one in execution order and if the operands of the expression remain unchanged between the two evaluations.

Local Common Subexpression Elimination (CSE) keeps track of the set of *available expressions* within a basic block and replaces instances of them by references to new temporaries that keep their value.

```
...
a=(x+y)+z;
b=a-1;
c=x+y;
...
```

Before CSE

```
...
t=x+y;
a=t+z;
b=a-1;
c=t;
...
```

After CSE





## Available Expressions

- An expression x+y is available at a program point p if
  - every path from the initial node to p evaluates x+y before reaching p,
  - and there are no assignments to x or y after the evaluation but before p.
- Available Expression information can be used to do global (across basic blocks) CSE.
- If an expression is available at the point of its use, there is no need to re-evaluate it.





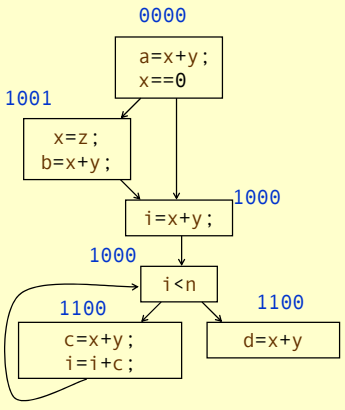
## Computing Available Expressions

- Represent sets of expressions using bit vectors
- Each expression corresponds to a bit
- Run dataflow algorithm similar to reaching definitions
- Notice that:
  - A definition reaches a basic block if it comes from ANY predecessor in CFG.
  - An expression is available at a basic block only if it is available from ALL block's predecessors in the CFG.





Expressions
1: x+y
2: i<n
3: i+c
4: x==0



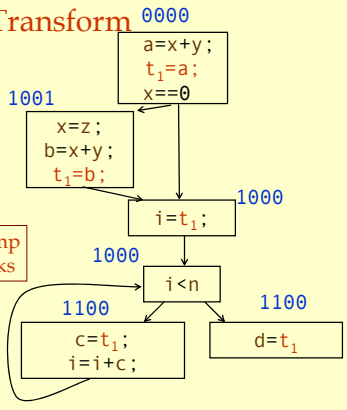





## Global CSE Transform

Expressions
1: x+y
2: i<n
3: i+c
4: x==0

Must use same temp for CSE in all blocks



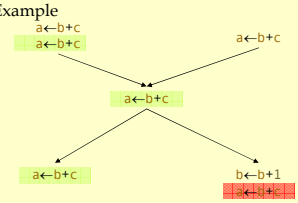
## Redundant Expressions

Redundant Expressions

An expression is redundant at a point *p* if, on every path to *p*

1. It is evaluated before reaching *p*, and
2. None of its constituent values is redefined before *p*

Example




**Some occurrences of b+c are redundant**

**Not all occurrences of b+c are redundant!**

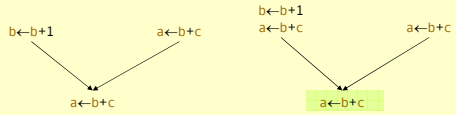


## Partially Redundant Expressions

Partially Redundant Expressions

An expression is partially redundant at *p* if it is redundant along some, but not all, paths reaching *p*.

Example



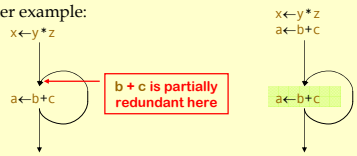

**Inserting a copy of "a←b+c" after the definition of b can make it redundant.**



## Loop Invariant Expressions

Partially Redundant Expressions

Another example:




Loop invariant expressions are partially redundant.

- Partial redundancy elimination performs code motion.
- Major part of the work is figuring out where to insert operations.