6.6. Profitable inserting of an expression

Inserting a copy of an expression $E$ is profitable if this step reduces the execution time.

Issues the compiler should consider

- Inserting a copy (potentially) increases the size of the target block (where $E$ is inserted)
- May (or may not) take more cycles to execute
- 2nd order effects...
  - Increased cache miss rate
  - Potential for conflict misses (or may not)
- inserting a copy of E may introduce
  increase register pressure

- value computed by E is reused —
  may have to be stored in register

Difficult decision for compiler: inserting and not
inserting ...
Idea:
Compiler considers execution frequency of basic blocks

... the compiler needs information about execution frequency!!

JIT compiling works well...

Section 7
6.7 Transformations

Assume $I$ nodes inserted, down safety known, information about execution frequency.

$I$ node with 1 operand
- decision made: $E$ is to be inserted
  - Insert $E$ into predecessor block that leads to $I$ operand

$E$ is now fully redundant
- either computed earlier or in block 5
* Insert an assignment of E to some temporary variable (t_E)
  - hold value of E
  - save re-computation

* Replace computation of E in other block(s) with a read operation (use t_E)

* Remove the computation of E in blocks that have been transformed.
1) convert to SSA, insert \( \phi \) nodes, versions
2) insert \( \Phi \) nodes
   for \( E : x + y \) versions
down safe:
\( \Phi \) in \( B_2 \) is down safe
\( \bot \) of \( \Phi \) in \( B_2 \) is a legal candidate
   assume insertion of \( E \) into \( B_2 \) is profitable
Insert copy of \( E \) into \( B_2 \)
insert temporary version, assign expression
use \( t_i \) : remove \( E_i \) from program