Compilers

Peephole Optimization
• Optimizations can be directly applied to assembly code

• **Peephole optimization** is effective for improving assembly code
  – The “peephole” is a short sequence of (usually contiguous) instructions
  – The optimizer replaces the sequence with another equivalent one (but faster)
• Write peephole optimizations as replacement rules

\[ i_1, \ldots, i_n \rightarrow j_1, \ldots, j_m \]

where the rhs is the improved version of the lhs

• Example:

\[
\text{move } a, b, \text{ move } b, a \rightarrow \text{ move } a, b
\]

– Works if \text{move } b, a is not the target of a jump

• Another example

\[
\text{addiu } a, a, i, \text{ addiu } a, a, j \rightarrow \text{ addiu } a, a, i+j
\]
Many (but not all) of the basic block optimizations can be cast as peephole optimizations.

- Example: `addiu $a $b 0` → `move $a $b`
- Example: `move $a $a` →
- These two together eliminate `addiu $a $a 0`

As for local optimizations, peephole optimizations must be applied repeatedly for maximum effect.
• Many simple optimizations can still be applied on assembly language

• “Program optimization” is grossly misnamed
  – Code produced by “optimizers” is not optimal in any reasonable sense
  – “Program improvement” is a more appropriate term