Instruction Selection via Peephole Optimization

Copyright 2010, Keith D. Cooper & Linda Torczon, all rights reserved.
Students enrolled in Comp 412 at Rice University have explicit permission to make copies of these materials for their personal use.
Faculty from other educational institutions may use these materials for nonprofit educational purposes, provided this copyright notice is preserved.

Definitions

Instruction selection
- Mapping *IR* into assembly code
- Assumes a fixed storage mapping
- Combining operations, using address modes

Instruction scheduling
- Reordering operations to hide latencies
- Assumes a fixed program *(set of operations)*
- Changes demand for registers

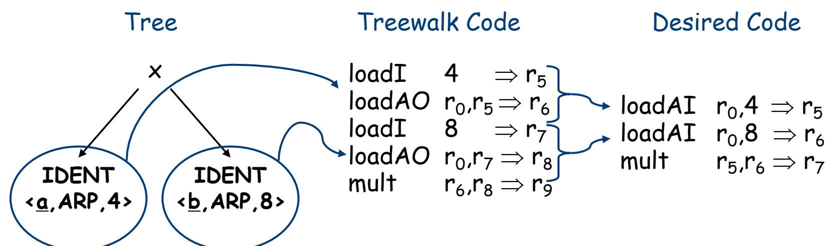
Register allocation
- Deciding which values will reside in registers
- Changes the storage mapping, may add false sharing
- Concerns about placement of data & memory operations

The Big Picture

In peephole optimization, we use a small window of instructions.
In particular, we look for pre-defined patterns.
We replace those patterns with another, more desirable pattern.
The pre-defined patterns are oftentimes known to the compiler designer.
Oftentimes, they are the result of a combination of procedures which emit code.
The code-generator does not have the context to optimize the code outright.
As such a second pass is necessary.

Example

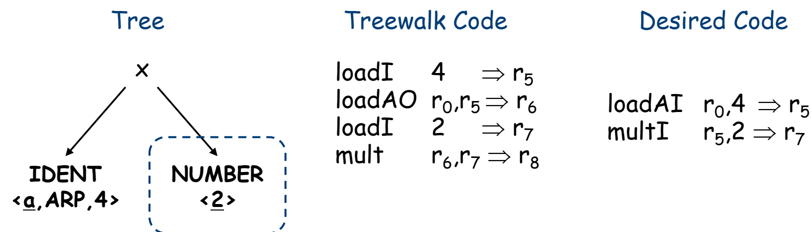
In this example, we replace the combination of **`loadI`** and **`loadAO`** by a single **`loadAI`**.
Naturally, if **`loadAI`** were not available on our target machine, we would not be able to perform this optimization.



Second Example

In this example, we can again combine the first two operations. We can also combine the second two operations, leading to the desired code.

Notice that `loadI` and `loadAO` combine to `loadAI` and `loadI` and `mult` combine to `multI`



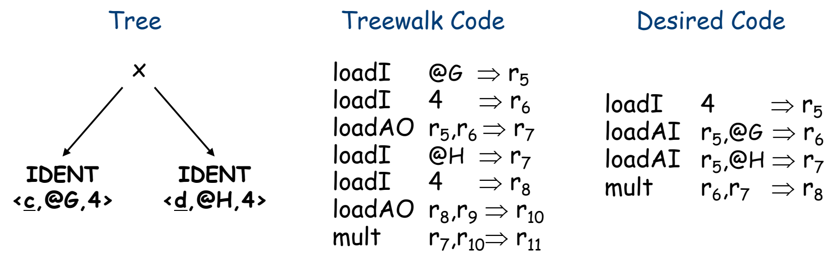
Second Example

Taking advantage of our vast algebraic knowledge, we may wish to produce the following code instead.

Desired Code		Even more desireable Code	
loadAI	$r_0,4 \Rightarrow r_5$	loadAI	$r_0,4 \Rightarrow r_5$
multI	$r_5,2 \Rightarrow r_7$	add	$r_5,r_5 \Rightarrow r_7$

Third Example

Consider the following code.
@ denotes the memory offset for its variable.



Peephole Matching

Modern peephole instruction selectors break problem into three tasks:

Peephole Matching

Expander

- Turns IR code into a low-level IR (LLIR) such as Register Transfer Language (RTL)
- Operation-by-operation, template-driven rewriting
- Significant, albeit constant, expansion of size

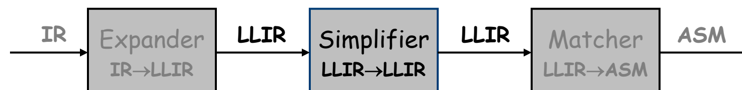
Peephole Matching

Simplifier

Looks at LLIR through window and rewrites it

Uses forward substitution, algebraic simplification, local constant propagation, and dead-effect elimination

Performs local optimization within window



This is the heart of the peephole system

Peephole Matching

Matcher

- Compares simplified LLIR against a library of patterns
- Picks low-cost pattern that captures effects
- Generates the assembly code output

Example : w := x – 2 * y

Original IR Code

OP	Arg_1	Arg_2	Result
mult	2	y	t_1
sub	x	t_1	w

Expander

Original IR Code

OP	Arg_1	Arg_2	Result
mult	2	y	t_1
sub	x	t_1	w

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

This version of the example assumes that the addresses of x, y, and w are all stored in the AR. The address of the AR is stored in r_0

Simplifier

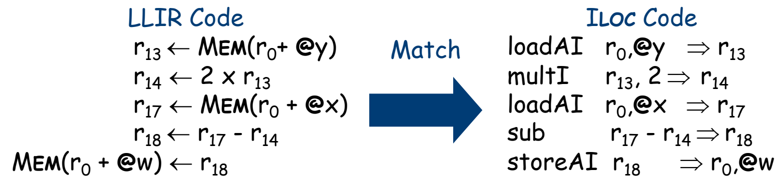
LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

LLIR Code

$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow 2 \times r_{13}$
$r_{17} \leftarrow MEM(r_0 + @x)$
$r_{18} \leftarrow r_{17} - r_{14}$
$MEM(r_0 + @w) \leftarrow r_{18}$

Matcher



Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow$ @y
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow$ MEM(r_{12})
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow$ @x
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow$ MEM(r_{16})
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow$ @w
$r_{20} \leftarrow r_0 + r_{19}$
MEM(r_{20}) $\leftarrow r_{18}$

Window:

$r_{10} \leftarrow 2$
$r_{11} \leftarrow$ @y
$r_{12} \leftarrow r_0 + r_{11}$

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

Window:
$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$

→

$r_{10} \leftarrow 2$
$r_{12} \leftarrow r_0 + @y$
$r_{13} \leftarrow MEM(r_{12})$

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

Window:
$r_{10} \leftarrow 2$
$r_{12} \leftarrow r_0 + @y$
$r_{13} \leftarrow MEM(r_{12})$

→

$r_{10} \leftarrow 2$
$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow r_{10} \times r_{13}$

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

$r_{10} \leftarrow 2$
$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow r_{10} \times r_{13}$

→

$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow 2 \times r_{13}$
$r_{15} \leftarrow @x$

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow 2 \times r_{13}$
$r_{15} \leftarrow @x$

→

$r_{14} \leftarrow 2 \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$

Optimized Code
$r_{13} \leftarrow MEM(r_0 + @y)$

Simplifier emits ops when they roll out of the window.

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

$r_{14} \leftarrow 2 \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$

$r_{14} \leftarrow 2 \times r_{13}$
$r_{16} \leftarrow r_0 + @x$
$r_{17} \leftarrow MEM(r_{16})$

Optimized Code
$r_{13} \leftarrow MEM(r_0 + @y)$

Steps of the Simplifier *3-operation window*

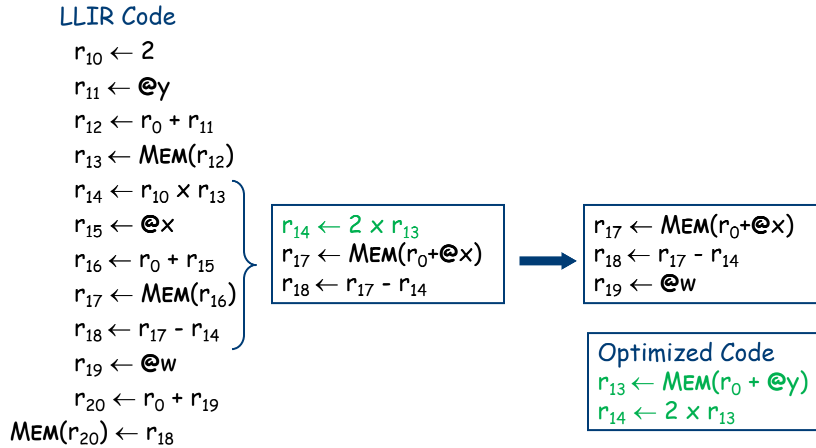
LLIR Code

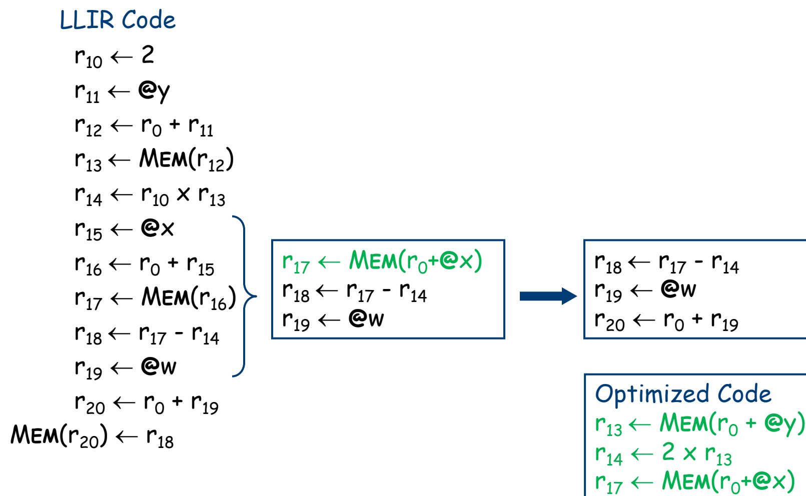
$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

$r_{14} \leftarrow 2 \times r_{13}$
$r_{16} \leftarrow r_0 + @x$
$r_{17} \leftarrow MEM(r_{16})$

$r_{14} \leftarrow 2 \times r_{13}$
$r_{17} \leftarrow MEM(r_0 + @x)$
$r_{18} \leftarrow r_{17} - r_{14}$

Optimized Code
$r_{13} \leftarrow MEM(r_0 + @y)$


Steps of the Simplifier 3-operation window
LLIR Code
r10 ← 2
r11 ← @y
r12 ← r0 + r11
r13 ← MEM(r12)
r14 ← r10 x r13
r15 ← @x
r16 ← r0 + r15
r17 ← MEM(r16)
r18 ← r17 - r14
r19 ← @w
r20 ← r0 + r19
MEM(r20) ← r18
r14 ← 2 x r13
r17 ← MEM(r0+@x)
r18 ← r17 - r14
r17 ← MEM(r0+@x)
r18 ← r17 - r14
r19 ← @w
Optimized Code
r13 ← MEM(r0 + @y)
r14 ← 2 x r13


Steps of the Simplifier 3-operation window
LLIR Code
r10 ← 2
r11 ← @y
r12 ← r0 + r11
r13 ← MEM(r12)
r14 ← r10 x r13
r15 ← @x
r16 ← r0 + r15
r17 ← MEM(r16)
r18 ← r17 - r14
r19 ← @w
r20 ← r0 + r19
MEM(r20) ← r18
r17 ← MEM(r0+@x)
r18 ← r17 - r14
r19 ← @w
r18 ← r17 - r14
r19 ← @w
r20 ← r0 + r19
Optimized Code
r13 ← MEM(r0 + @y)
r14 ← 2 x r13
r17 ← MEM(r0+@x)

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

Optimized Code
$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow 2 \times r_{13}$
$r_{17} \leftarrow MEM(r_0 + @x)$

$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$

→

$r_{18} \leftarrow r_{17} - r_{14}$
$r_{20} \leftarrow r_0 + @w$
$MEM(r_{20}) \leftarrow r_{18}$

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

Optimized Code
$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow 2 \times r_{13}$
$r_{17} \leftarrow MEM(r_0 + @x)$

$r_{18} \leftarrow r_{17} - r_{14}$
$r_{20} \leftarrow r_0 + @w$
$MEM(r_{20}) \leftarrow r_{18}$

→

$r_{18} \leftarrow r_{17} - r_{14}$
$MEM(r_0 + @w) \leftarrow r_{18}$

Steps of the Simplifier *3-operation window*

LLIR Code

$r_{10} \leftarrow 2$
$r_{11} \leftarrow @y$
$r_{12} \leftarrow r_0 + r_{11}$
$r_{13} \leftarrow MEM(r_{12})$
$r_{14} \leftarrow r_{10} \times r_{13}$
$r_{15} \leftarrow @x$
$r_{16} \leftarrow r_0 + r_{15}$
$r_{17} \leftarrow MEM(r_{16})$
$r_{18} \leftarrow r_{17} - r_{14}$
$r_{19} \leftarrow @w$
$r_{20} \leftarrow r_0 + r_{19}$
$MEM(r_{20}) \leftarrow r_{18}$

$r_{18} \leftarrow r_{17} - r_{14}$
$MEM(r_0 + @w) \leftarrow r_{18}$

Optimized Code

$r_{13} \leftarrow MEM(r_0 + @y)$
$r_{14} \leftarrow 2 \times r_{13}$
$r_{17} \leftarrow MEM(r_0 + @x)$
$r_{18} \leftarrow r_{17} - r_{14}$
$MEM(r_0 + @w) \leftarrow r_{18}$