

Low-Level Optimizations

CS 502

Lecture 15

12/04/08

Goals

- Devise optimizations specifically targeted to improve performance on specific architectures
 - Apply optimizations on low-level IR or assembly code.
 - Categories:
 - General-purpose: dead-code elimination
 - Control-flow:
 - Branch predication
 - loop optimizations
 - Peephole
-

Transformations

- ❑ dead-code and unreachable code elimination
 - ❑ straightening: create longer basic blocks by inlining branches with the code they branch to
 - ❑ if simplification: eliminate unused arms of a conditional
 - ❑ loop simplification: unrolling
 - ❑ loop inversion: replace a test at the beginning of a loop with a test at the end
 - ❑ loop-invariant code motion
 - ❑ branch simplification
 - ❑ tail merging: turn identical tails of basic blocks into a common jump to a single tail
 - ❑ conditional moves: remove simple if constructs by code sequences that contain no branches
 - ❑ instruction selection: replace combination of simple instructions into a single more complex instruction (that is presumably faster)
-

Unreachable and dead-code elimination

□ Dead-code elimination:

- eliminate executable code that has no effect on the computation
 - E.g., multiple stores of the same value to the same register with no intervening stores of different values

□ Unreachable code elimination:

- remove code that is not reachable along any control-flow path
-

Straightening

- Apply to pairs of basic blocks:
 - the first has no successor other than the second
 - the second has no predecessor other than the first:
 - second one either immediately follows the first, or
 - first must end with an unconditional branch to the second
 - For former case, nothing needs to be done; for second case, replace branch with the second block
-

Example

L1: ...

$a \leftarrow b + c$

goto L2

...

L6: ...

goto L4

L2: $b \leftarrow c + 2$

$a \leftarrow a + 1$

if ($c > 0$) goto L3

L5: ...

Block L1 ends with an unconditional branch to L2.

Assume L2 has no other predecessors

Example

L1: ...

$a \leftarrow b + c$

$b \leftarrow c + 2$

$a \leftarrow a + 1$

if ($c > 0$) goto L3

goto L5

...

L6: ...

goto L4

L5: ...

Merge block L2 into block L1.

Note that an unconditional branch to L5 must be inserted to reflect the fall-through between L2 and L5 in the original program

If simplification

- Apply to conditional tests in which one or both arms are empty:
 - May occur as a result of code hoisting
 - Cases:
 - If then branch empty, reverse sense of conditional, and eliminate the branch
 - Constant-value conditional tests
 - Common subexpressions in the conditional test and in a subsequent dependent if.
-

Example

L0: if $a > d$

 goto L2

 goto L5

L2: $b \leftarrow a$

$c \leftarrow b * 4$

 if $a > d$ or B

 goto L3

 goto L4

L3: $d \leftarrow b$

 goto L5

L4: $d \leftarrow c$

L5: $e \leftarrow a + d$

L0: if $a > d$

 goto L2

 goto L5

L2: $b \leftarrow a$

$c \leftarrow b * 4$

L3: $d \leftarrow b$

L5: $e \leftarrow a + d$

Loop simplifications

- A loop whose body is empty can be eliminated
 - Must ensure that iteration-control code has no side effects, or side-effects are simple and can be replaced by non-looping code
 - Example: Loop increments/decrements a variable by a constant, and checks against a final constant:
 - Compute final value at compile-time
 - Loop unrolling
 - Small number of known iterations
-

Example

L0: $s \leftarrow 0$

$i \leftarrow 0$

L2: if $i > 4$

goto L1

$i \leftarrow i + 1$

$s \leftarrow s + i$

goto L2

L1: ...

L0: $s \leftarrow 0$

$i \leftarrow 0$

$i \leftarrow i + 1$

$s \leftarrow s + i$

$i \leftarrow i + 1$

$s \leftarrow s + i$

$i \leftarrow i + 1$

$s \leftarrow s + i$

$i \leftarrow i + 1$

$s \leftarrow s + i$

....

Loop Inversion

- Transform a while loop into a repeat loop:
 - Moves the closing test from before the body of the loop to after it.
 - Requires that the loop be executed at least once, or need a test before to determine if it is entered
 - Advantage: Need only one branch to close the loop, rather than two (one to get back to the beginning, and one to perform the test)
-

Example

L0: $i \leftarrow 0$

L1: if $i < 100$

 goto L2

 goto L3

L2: $a[i] \leftarrow i + 1$

$i \leftarrow i + 1$

 goto L1

L3: ...

L0: $i \leftarrow 0$

L1: $a[i] \leftarrow i + 1$

$i \leftarrow i + 1$

 if $i < 100$

 goto L1

L3: ...

Unswitching

- Move loop invariant conditional branches out of loops
 - May increase code space, but reduces overall number of instructions executed
 - If conditional has no else branch, must supply one in the transformed code to set the final value of the loop-control variable
-

Example

L0: $i \leftarrow 1$

L1: if $i > 100$

 goto L3

L2: if $k = 2$ and $a[i] > 0$

$a[i] \leftarrow a[i] + 1$

 goto L1

L3: ...

L0: $i \leftarrow 1$

L1: if $k = 2$

 goto L2

 goto L3

L1: if $i > 100$

 goto L4

L2: if $a[i] > 0$

$a[i] \leftarrow a[i] + 1$

 goto L1

L3: $i \leftarrow 101$

L4: ...

Branch Optimizations

- Eliminate branch-to-branch transfers:
 - An unconditional branch to an unconditional branch can be replaced by a branch to the latter's target
 - A conditional branch to an unconditional branch can be replaced by a conditional branch to the latter's target
 - An unconditional branch to a conditional branch can be replaced by the conditional branch
 - A conditional branch to a conditional branch can be replaced by a conditional branch with the former's test and the latter's target
 - Latter condition must be true whenever the former is
 - Example:

if a = 0 goto L1

if a = 0 goto L2

....

L1: if a >= 0 goto L2

Tail Merging/Cross Jumping

- Search for basic blocks with common few last instructions
- Replace these instructions with a branch to a common block
- Algorithm: scan backwards through predecessors of blocks that have multiple predecessors looking for common instruction sequences

Example

...
r1 \leftarrow r2 + r3
r4 \leftarrow r3 shl 2
r2 \leftarrow r2 + 1
r2 \leftarrow r4 - r2
goto L1

...
r5 \leftarrow r4 - 6
r4 \leftarrow r3 shl 2
r2 \leftarrow r2 + 1
r2 \leftarrow r4 - r2

L1: ...

...
r1 \leftarrow r2 + r3
goto L2
...
r5 \leftarrow r4 - 6
L2: r4 \leftarrow r3 shl 2
 r2 \leftarrow r2 + 1
 r2 \leftarrow r4 - r2

L1: ...

Conditional Moves

- A conditional move copies a source to a target if and only if a specified condition is satisfied.
 - Found on many modern architectures (Pentium, Sparc, etc.)
 - Replace branching conditions
-

Example

if $a > b$

 goto L1

$c \leftarrow b$

 goto L2

L1: $c \leftarrow a$

L2: ...

$t1 \leftarrow a > b$

$c \leftarrow b$

$c \leftarrow (t1) a$

Code Patterns

if opd1 relop opd2

goto L1

r1 \leftarrow opd3

goto L2

L1: r1 \leftarrow opd4

L2: ...

r \leftarrow opd3

t \leftarrow opd1 relop opd2

r \leftarrow (t) opd4

r \leftarrow opd3

if opd1 relop opd2

goto L1

r \leftarrow opd4

L1:...

r \leftarrow opd3

t \leftarrow opd1 !relop opd2

reg \leftarrow (t) opd4
