Review of Code Optimization

66.648 Compiler Design Lecture (04/21/97)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Review of Code optimization
- Examples
- Administration

What optimization is used

Before:	After:
a:=1	a:=1
if(x<y) goto L1	IF (x<y) goto l1
b:=2	b:=2
c:=a+b	c:=3
goto l2	goto L2
l1: b:=4	l1: b:=4
c:=b-1	c:=3
l2: d:=a+c	l1:d:=4

What optimization is used

Before:	After:
proc mult(a,b)	
t1:=a*B	proc main
return T1	...
end proc	a:=y
proc main	b:=z
…	t1:=a*b
x:= call mult(y,z)	x:=t1
.. Ed proc	end proc

What optimization is used

Before:	After:
a:=y	t1:=y*z
b:=z	x:=t1
t1:=a*b	
x:=t1	

What optimization is used

Before:	After:
debug:=false	debug:=false
t1:=not debug	t1:=true
if (t1) goto l1	l1:x:=y*z
…	
call print	
l1: x:=y*z	

What optimization is used

Before:	After:
debug:=false	l1:x:=y*z
t1:=not debug	
l1:x:=y*z	

What optimization is used

Before:if (x<y) goto l1	After:if(x<y) goto l1
t1:=4*I	t7:=4*i
t2:=t1+x	t2:=t7+x
goto l2	goto l2
l1:t3:=4*I	l1:t7:=4*i
t4:=t3+y	t4:=t7+x
l2: t5:=4*I	l2:t6:=t7+z
t6:=t5+z	

What optimization is used

Before:	After:
l1: sum:=sum+I	delta:=m/n
delta:=m/n	l1:sum:=sum+i
I:=I+delta	I:=I+delta
if (I<=n) goto l1	if(I<=n) goto l1

What optimization is used

Before:	After:
c:='s'	c.location is unchanged
l1: sum:=sum+I	sum.location= reg 2
I:=I+1	I.location=reg 3
if (I<=n) goto l1	n.location = reg 4
…	

Ordering of Optimizations

- Constant propagation
- common subexpression elimination
- global register allocation

Reasons

- Constant propagation and constant folding may change the register interference graph.
- Constant folding may reveal more common sub-expressions.
- Common sub-expression elimination will not introduce any constant folding.

How about the other optimizations order?

No single ordering can be guaranteed to be best in all cases. Also optimizations may need to be performed multiple times in a single compilation for increased efficiency.

Some Observations

- Inlining should be performed first because it can create additional opportunities for other optimizations, but no other optimization creates additional opportunities for inlining.
- Peephole optimizations should be performed last, because they are performed on target code, not on IL.
- Register allocation should be performed second last, because all optimizations can change the register interference graph.

Some Observations

- Copy propagation and common subexpression elimination should be done iteratively, till no further optimization of copy propagation and cse can be carried out.
- Unreachable code should be done first before dead code elimination.
- Constant progation and unreachable code and dead code elimination should be done iteratively till no such optimizations are possible.

Some Observations

- There is no ordering constraint between loop-invariant code motion and cse,We can choose code motion follow common subexpression elimination.

Resulting order

- Inline expansion
- constant propagation
- unreachable code elimination
- dead-code elimination
- common sub-expression elimination
- copy propagation
- loop invariant code motion
- Global register allocation and peephole optimization

Comments and Feedback

- Project 3 is due this Friday. Please start studying for the exam.