CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
Phases of a compiler

Figure 1.6, page 5 of text

Optimizations
Algebraic Identities [p. 536]

\[ x + 0 = 0 + x = x \]
\[ x \times 1 = 1 \times x = x \]
\[ x - 0 = x \]
\[ x \div 1 = x \]
Algebraic Identities [p. 536]

- $x^2 = x \times x$
- $2 \times x = x + x$
- $x / 2 = x \times 0.5$

Can also use left and right for integers
(But see https://en.wikipedia.org/wiki/Arithmetic_shift)
Algebraic Identities [p. 536]

Constant folding

"...evaluate constant expressions at compile time and replace the constant expressions by their values."
See footnote 2:

“Arithmetic expressions should be evaluated the same way at compile time as they are at run time. K. Thompson has suggested an elegant solution to constant folding: compile the constant expression, execute the target code on the spot, and replace the expression with the result. Thus, the compiler does not need to contain an interpreter.”
Peephole optimization
[p 549]

"The peephole is a small, sliding window on a program." [p. 549]

"In general, repeated passes over the target code are necessary to get the maximum benefit." [p. 550]
Peepholo optimization: redundant LD/ST

LD RO, a
ST a, RO

If the ST instruction has a label, cannot remove it. (If instructions are in the same block we’re OK.)
Peephole optimization: unreachable code

if \( E=K \) goto L1

goto L2

L1: ...

... ...

L2: ...

... ...

Suppose \( K \) is a constant.
if E=K goto L1
  goto L2
L1: ...do something...
...
L2: ...do something...
...

Peephole optimization: unreachable code

Eliminate jumps over jumps
Peephole optimization: unreachable code

if $E=K$ goto L1
  goto L2
L1: ...
  ...
L2: ...
...

if $E!=K$ goto L2
  L1: ...
    ...
L2: ...
  ...

Eliminate jumps over jumps
Peephole optimization: unreachable code

If there are no jumps to L1, we can remove label.
Peephole optimization: unreachable code

If E is set to a constant value other than K, then...

```plaintext
if E=K goto L1
  goto L2
L1: ...
  ...
L2: ...
  ...
if E!=K goto L2
  ...
  ...
  L2: ...
  ...
```
Peephole optimization: unreachable code

if E=K goto L1
    goto L2
L1: ...
    ...
L2: ...
    ...

if true goto L2
    ...
    ...
    ...

...conditional jump becomes unconditional...
Peephole optimization: unreachable code

if $E=K$ goto L1
goto L2
L1: ...
...
L2: ...
...

...and the unreachable code can be removed.
Peephole optimization: flow-of-control

goto L1
...
L1: goto L2
...
L2:
Peephole optimization: flow-of-control

```
goto L1
...
L1: goto L2
...
L2:
```

```
goto L2
...
L1: goto L2
...
L2:
```
Peephole optimization: flow-of-control

If there are no jumps to L1, and L1 is preceded by an unconditional jump...
Peephole optimization: flow-of-control

goto L1
...
L1: goto L2
...
L2:

goto L2
...
L2:

...then we can eliminate the statement labelled L1
Peephole optimization: flow-of-control

if $a < b$ goto L1
...
L1: goto L2
...
L2:

if $a < b$ goto L2
...
L2:

...similar arguments can be made for conditional jumps.
Other optimizations
(more to come)