What Is Optimization?

• The process of automated translation of a program will invariably introduced **inefficiencies**. Our goal in **optimization** is to remove as many of these inefficiencies as possible.

• Optimization can be **local** (optimizing basic blocks within a program) or **global** (across the entire program).

• Even after optimizing intermediate code, it may be necessary to optimize the final object code because of **inefficiencies introduced in final code generation**.
A Sample Program in JASON

PROGRAM MySample;
INTEGER x, y;
BEGIN
SET x := 12;
SET y := 3;
WHILE y ! 0 DO
SET x := x + y;
SET y := y - 1
ENDWHILE;
END.

Basic Blocks

- A basic block is a sequence of instruction that will be performed in sequence, always going from the beginning of the block to the end of the block without jumping out of the block.
- There may be more than one basic block that transfers control to a given block and there may be more than one basic block to which we will transfer control as we leave a given block.
The Basic Blocks Of Our Sample Program

X := 12
Y := 3

!_1:
if Y != 0 goto !_2

X := X + Y
Y := Y - 1
goto !_1

!_2:

Flow Graphs

X := 12
Y := 3

!_1:
if Y != 0 goto !_2

X := X + Y
Y := Y - 1
goto !_1

!_2:
Principle Optimizations On Basic Blocks

- There are several different optimizations that we can (and will) perform on basic blocks. They include:
  - Common Sub-expression Elimination
  - Copy propagation
  - Dead-Code Elimination
  - Arithmetic Transformation

Common Subexpression Elimination

\[
\begin{align*}
b &:= 4-2 \\
$$_1 := b / 2$$ \\
$$_2 := a * $$$_1$$ \\
$$_3 := $$$_2 * b$$ \\
$$_4 := $$$_3 + c$$ \\
$$_5 := $$$_2 * b$$ \\
$$_6 := $$$_5 + c$$ \\
d &:= $$$_4 * $$$_6$$
\end{align*}
\]
Common Subexpression Elimination

\[ b := 4 - 2 \]
\[ _$1 := b / 2 \]
\[ _$2 := a * _$1 \]
\[ _$3 := _$2 * b \]
\[ b := _$3 + c \]
\[ _$5 := _$2 * b \]
\[ _$6 := _$5 + c \]
\[ d := _$4 * _$6 \]

We cannot use subexpression elimination here because \( b \)'s value was changed

Copy Propagation

\[ b := 4 - 2 \]
\[ _$1 := b / 2 \]
\[ _$2 := a * _$1 \]
\[ _$3 := _$2 * b \]
\[ _$4 := _$3 + c \]
\[ _$5 := _$3 \]
\[ _$6 := _$5 + c \]
\[ d := _$4 * _$6 \]

\[ b := 4 - 2 \]
\[ _$1 := b / 2 \]
\[ _$2 := a * _$1 \]
\[ _$3 := _$2 * b \]
\[ _$4 := _$3 + c \]
\[ _$5 := _$3 \]
\[ _$6 := _$3 + c \]
\[ d := _$4 * _$6 \]
Subexpression After Copy Propagation

\[
b := 4-2 \\
_1 := b / 2 \\
_2 := a \times _1 \\
_3 := _2 \times b \\
_4 := _3 + c \\
_5 := _3 \\
_6 := _3 + c \\
d := _4 \times _6
\]

Copy Propagation After Subexpression

\[
b := 4-2 \\
_1 := b / 2 \\
_2 := a \times _1 \\
_3 := _2 \times b \\
_4 := _3 + c \\
_5 := _3 \\
_6 := _4 \\
d := _4 \times _6
\]
Dead-Code Elimination

\[ b := 4 - 2 \]
\[ \$_1 := b / 2 \]
\[ \$_2 := a \times \$_1 \]
\[ \$_3 := \$_2 \times b \]
\[ \$_4 := \$_3 + c \]
\[ \$_5 := \$_3 \]
\[ \$_6 := \$_4 \]
\[ d := \$_4 \times \$_4 \]

No references to \$_5 after defining its value

Arithmetic Transformations

- We can use the laws of algebra to replace expressions that either do not need to be calculated or can be calculated more quickly by other means.
- These algebraic transformations include:
  - Constant Folding
  - Algebraic Simplification
  - Reduction In Strength
Constant Folding

\[
\begin{align*}
    b & := 4 - 2 \\
    \$_1 & := b / 2 \\
    \$_2 & := a * \$_1 \\
    \$_3 & := \$_2 * b \\
    \$_4 & := \$_3 + c \\
    \$_6 & := \$_4 \\
    d & := \$_4 * \$_4 \\
\end{align*}
\]

\[
\begin{align*}
    b & := 2 \\
    \$_1 & := b / 2 \\
    \$_2 & := a * \$_1 \\
    \$_3 & := \$_2 * b \\
    \$_4 & := \$_3 + c \\
    d & := \$_4 * \$_4 \\
\end{align*}
\]

Copy Propagation & Dead-Code Elimination After Constant Folding

\[
\begin{align*}
    b & := 2 \\
    \$_1 & := b / 2 \\
    \$_2 & := a * \$_1 \\
    \$_3 & := \$_2 * b \\
    \$_4 & := \$_3 + c \\
    d & := \$_4 * \$_4 \\
\end{align*}
\]
More Constant Folding

$$_1 := 2 / 2$$
$$_2 := a * $$$_1$$
$$_3 := $$$_2 * 2$$
$$_4 := $$$_3 + c$$
$$_6 := $$$_4$$
d := $$$_4 * $$$_4$$

$$_1 := 1$$
$$_2 := a * $$$_1$$
$$_3 := $$$_2 * 2$$
$$_4 := $$$_3 + c$$
d := $$$_4 * $$$_4$$

More Copy Propagation & Dead-Code Elimination

$$_1 := 1$$
$$\{$$
$$$_2 := a * $$$_1$$
$$$_3 := $$$_2 * 2$$
$$$_4 := $$$_3 + c$$
$$$_6 := $$$_4$$
d := $$$_4 * $$$_4$$
$$\}$$

$$\{$$
$$_2 := a * 1$$
$$_3 := $$$_2 * 2$$
$$$_4 := $$$_3 + c$$
d := $$$_4 * $$$_4$$
$$\}$$
Algebraic Simplification

- We can simplify our expressions by using algebraic identities:
  \[ x + 0 = 0 + x = x \]
  \[ x - 0 = x \]
  \[ x \cdot 1 = 1 \cdot x = x \]
  \[ x / 1 = x \]

Applying Algebraic Simplification

\[
\begin{align*}
$_2 & := a \times 1 \\
$_3 & := $_2 \times 2 \\
$_4 & := $_3 + c \\
d & := $_4 \times $_4
\end{align*}
\]
After Copy Propagation & Dead-Code Elimination

\[\begin{align*}
$_2 & := a \\
$_3 & :=$_2 \times 2 \\
$_4 & :=$_3 + c \\
d & :=$_4 \times$_4
\end{align*}\]

After Copy Propagation & Dead-Code Elimination

\[\begin{align*}
$_2 & := a \\
$_3 & :=$_2 \times 2 \\
$_4 & :=$_3 + c \\
d & :=$_4 \times$_4
\end{align*}\]
Reduction In Strength

- We can replace multiplication and division (or exponentiation) with addition and subtraction (or multiplication) which can usually be done much more quickly.
- We can use the identities:
  - $x^2 = x \cdot x$
  - $2 \cdot x = x + x$
- We can also use shifts to replace multiplication and division by powers of 2

Applying Reduction In Strength

\[\begin{align*}
_3 & := a \cdot 2 \\
_4 & := _3 + c \\
d & := _4 \cdot _4
\end{align*}\]

\[\begin{align*}
_3 & := a + a \\
_4 & := _3 + c \\
d & := _4 \cdot _4
\end{align*}\]
Our End Result

\[
\begin{align*}
  b & := 4 - 2 \\
  _3 & := a + a \\
  _4 & := _3 + c \\
  d & := _4 \times _4 \\
  _1 & := b / 2 \\
  _2 & := a \times _1 \\
  _3 & := _2 \times b \\
  _4 & := _3 + c \\
  _5 & := _2 \times b \\
  _6 & := _5 + c \\
  d & := _4 \times _6
\end{align*}
\]