CSE443
Compilers

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

Intermediate Representation (IR): specification and generation

Figure 1.6, page 5 of text
IR: a motivating example
Boolean expressions can be evaluated
- to determine the flow of control
- for their value
Boolean expressions

Examples: \( \neg X \quad X \& Y \quad X \mid Y \)
Boolean expressions

Examples: \( \neg X \), \( X \land Y \), \( X \lor Y \)

We will do short-circuit evaluation
Boolean expressions

- Examples: !X  X & Y  X | Y
- We will do short-circuit evaluation
- For example:
  if (X | (Y & Z)) then { A } else { B }
  is translated as

  if X goto LA
  ifFalse Y goto LB
  ifFalse Z goto LB

LA:  A
  goto END
LB:  B
END: (next instruction)
Boolean expressions

A concrete exercise – how is this translated?

if (r < s | (r = s & 0 < s)) then { A } else { B }
Boolean expressions

- A concrete exercise - how is this translated?
- \( \text{if } (r < s \mid (r = s \& 0 < s)) \text{ then } \{ A \} \text{ else } \{ B \} \)

Here's a summary of the Intermediate Representation (IR) that we'll be using.

Three address code instructions (see 6.2.1, pages 364-5)

1. \( x = y \text{ op } z \)
2. \( x = \text{ op } y \) (treat i2r and r2i as unary ops)
3. \( x = y \)
4. goto \( L \)
5. if \( x \) goto \( L \) / ifFalse \( x \) goto \( L \)
6. if \( x \) relop \( y \) goto \( L \)
7. function calls:
   - param \( x \)
   - call \( p, n \)
   - \( y = \text{ call } p \)
   - return \( y \)
8. \( x = y[i] \) and \( x[i] = y \)
9. \( x = \& y, x = *y, *x = y \)

We'll start with these.

We'll spend significant time on function calls later.

We'll explore these as needed later on.

© 2020 Carl Alphonse - Reproduction of this material is prohibited without the author's consent
Boolean expressions

A concrete exercise - how is this translated?

if ( r < s | (r = s & 0 < s)) then { A } else { B }

Three address code instructions
(see 6.2.1, pages 364-5)

1. x = y op z
2. x = op y          (treat i2r and r2i as unary ops)
3. x = y
4. goto L
5. if x goto L / ifFalse x goto L
6. if x relop y goto L
7. function calls:
   - param x
   - call p, n
   - y = call p
   - return y
8. x = y[i] and x[i] = y
9. x = &y, x = *y, *x = y

Pause the recording at this point and try to come up with a suitable translation.

Once done, resume playback and check your work.

We'll start with these.

We'll spend significant time on function calls later.

We'll explore these as needed later on.
Boolean expressions

A concrete exercise - how is this translated?

if \((r < s \mid (r = s \& 0 < s))\) then \{ A \} else \{ B \}

This has the same form as our example:

if \((X \mid (Y \& Z))\) then \{ A \} else \{ B \}

is translated as

if X goto LA
ifFalse Y goto LB
ifFalse Z goto LB

LA: A
   goto END

LB: B

END: (next instruction)

Three address code instructions (see 6.2.1, pages 364-5)

1. \(x = y \ op \ z\)
2. \(x = \ op \ y\) (treat i2r and r2i as unary ops)
3. \(x = y\)
4. goto L
5. if x goto L / ifFalse x goto L
6. if x relop y goto L
7. function calls:
   - param x
   - call p, n
   - y = call p
   - return y
8. \(x = y[i] \ and \ x[i] = y\)
9. \(x = \& y, x = \*y, \*x = y\)

We'll start with these.

We'll spend significant time on function calls later.

We'll explore these as needed later on.
**Boolean expressions**

- A concrete exercise - how is this translated?

- if \((r < s | (r = s & 0 < s))\) then \{ A \} else \{ B \}

- if \((x | (y & z))\) then \{ A \} else \{ B \}

This has the same form as our example:

if \((X | (Y & Z))\) then \{ A \} else \{ B \}

is translated as

if X goto LA
ifFalse Y goto LB
ifFalse Z goto LB

LA: A
goto END

LB: B

END: (next instruction)

Three address code instructions (see 6.2.1, pages 364-5)

1. \(x = y \) op \(z\)  
2. \(x = \) op \(y\)  
3. \(x = y\)  
4. goto L  
5. if \(x\) goto L / ifFalse \(x\) goto L  
6. if \(x\) relop \(y\) goto L  
7. function calls:
   - param \(x\)
   - call \(p, n\)
   - \(y =\) call \(p\)
   - return \(y\)
8. \(x = y[i]\) and \(x[i] = y\)
9. \(x = \) dy, \(x = *y\), \(*x = y\)

© 2020 Carl Alphonse - Reproduction of this material is prohibited without the author's consent
Boolean expressions

A concrete exercise - how is this translated?

if ( r < s | (r = s & 0 < s)) then { A } else { B }

This has the same form as our example:

if ( r < s | (Y & Z)) then { A } else { B } is translated as

if r < s goto LA
ifFalse Y goto LB
ifFalse Z goto LB

LA:   A
     goto END
LB:   B
END: (next instruction)

Three address code instructions
(see 6.2.1, pages 364-5)

1. x = y op z
2. x = op y
3. x = y
4. goto L
5. if x goto L / ifFalse x goto L
6. if x relop y goto L
7. function calls:
   - param x
   - call p, n
   - y = call p
   - return y
8. x = y[i] and x[i] = y
9. x = d[y], x = *y, *x = y

We'll start with these.
We'll spend significant time on function calls later.
We'll explore these as needed later on.
Boolean expressions

- A concrete exercise - how is this translated?

- if \((r < s \lor (r = s \land 0 < s))\) then \{A\} else \{B\}

- if \((x \lor (y \land z))\) then \{A\} else \{B\}

This has the same form as our example:

if \((r < s \lor (r = s \land Z))\) then \{A\} else \{B\}

is translated as

if \(r < s\) goto LA
ifFalse
\(r = s\) goto LB
ifFalse Z goto LB

LA:   A
goto END

LB:   B

END: (next instruction)

Three address code instructions (see 6.2.1, pages 364-5)

1. \(x = y\) op \(z\)
2. \(x =\) op \(y\)  \hspace{1cm} \text{(treat i2r and r2i as unary ops)}
3. \(x = y\)
4. goto L
5. if \(x\) goto L / ifFalse x goto L
6. if \(x\) relop \(y\) goto L
7. function calls:
   - param \(x\)
   - call \(p, n\)
   - \(y = \) call \(p\)
   - return \(y\)
8. \(x = y[i]\) and \(x[i] = y\)
9. \(x = \delta y, x = \ast y, \ast x = y\)

We'll start with these.

We'll spend significant time on function calls later.

We'll explore these as needed later on.
Boolean expressions

- A concrete exercise - how is this translated?

- if \((r < s \mid (r = s \& 0 < s))\) then \{ A \} else \{ B \}
- if \((x \mid (y \& z))\) then \{ A \} else \{ B \}

This has the same form as our example:

\[
\text{if } (r < s \mid (r = s \& 0 < s)) \text{ then } \{ A \} \text{ else } \{ B \}
\]

is translated as

\[
\begin{align*}
\text{if } r < s & \text{ goto LA} \\
\text{ifFalse } r = s & \text{ goto LB} \\
\text{ifFalse } 0 < s & \text{ goto LB}
\end{align*}
\]

LA: A

\[
\text{goto END}
\]

LB: B

END: (next instruction)

Three address code instructions (see 6.2.1, pages 364-5)

1. \(x = y \text{ op } z\)  
2. \(x = \text{ op } y\)  
   (treat \text{i2r} and \text{r2i} as unary ops)
3. \(x = y\)
4. goto L
5. if \(x\) goto L / ifFalse \(x\) goto L
6. if \(x\) relop y goto L
7. function calls:
   - param \(x\)
   - call \(p, n\)
   - \(y = \text{ call } p\)
   - return \(y\)
8. \(x = y[i] \text{ and } x[i] = y\)
9. \(x = \Delta y, x = *y, *x = y\)

We'll start with these.

We'll spend significant time on function calls later.

We'll explore these as needed later on.
IR: the general case
Let's generalize from the previous concrete example to one with an arbitrary Boolean expression $B$.

We assume that IR instructions are placed into an array.
Flow-of-Control (6.3.3)

if ( B ) then S1 else S2

B.true =NewLabel()
B.false =NewLabel()
S.next = S1.next = S2.next
S.code = B.code || label(B.true) || S1.code || gen('goto' S.next) || label(B.false) || S2.code
Flow-of-Control (6.3.3)

$S \rightarrow \text{if ( } B \text{ ) then } S1$

$B.true = \text{newlabel()}$
$B.false = S.next = S1.next$
$S.code = B.code \| label(B.true) \| S1.code$

ifTrue:
goto LS1
ifFalse:
goto END
Flow-of-Control (6.3.3)

while ( B ) then S1

begin = newlabel()
B.true = newlabel()
B.false = S.next
S1.next = begin
S.code = label(begin) || B.code || label(B.true) || S1.code || gen('goto' begin)

BEGIN

B.code

ifTrue:
goto LS1
ifFalse:
goto END
LS1

S1.code

goto BEGIN

END
Boolean expressions: value or control flow?
6.6.6 Boolean values and jumping code

"S \to \text{id} = E; | \text{if (E)} S | \text{while (E)} S | S S

Nonterminal $E$ governs the flow on control in $S \to \text{while (E)} S1$. The same nonterminal $E$ denotes a value in $S \to \text{id} = E; [...]"
6.6.6 Boolean values and jumping code

“Suppose that attribute E.n denotes the syntax-tree node for an expression E and that nodes are objects. Let method `jump` generate jumping code at an expression node, and let method `rvalue` generate code to compute the value of the node into a temporary.”

[p. 408]
Value of Boolean expression

"When $E$ appears in $S \rightarrow \text{while} (E) \ S_1$, method \textit{jump} is called at node $E.n$
[...]
When $E$ appears in $S \rightarrow id = E;$, method \textit{rvalue} is called at node $E.n$" [p. 408]
"If E has the form E1 + E2, the method call E.n.rvalue() generates code as discussed in section 6.4." [p. 408]

"E-> E1 + E2
   E.addr = new Temp()
   E.code = E1.code || E2.code || gen(E.addr '=' E1.addr '+' E2.addr)"

"If E has the form E1 && E2 we first generate jumping code for E and then assign true or false to a new temporary t at the true and false exits, respectively, from the jumping code." [p. 408]

Translation of: x = a<b && c<d
   ifFalse a < b goto L1
   ifFalse c < d goto L1
   t = true
   goto L2
L1:  t = false
L2:  x = t