Object-Oriented Intermediate Code Generation

CS 4447 / CS 9545 – Stephen M. Watt

***The* University *of* Western Ontario**

High-Level Intermediate Code (HCode)

- Suitable for first-stage optimizations
- Easy to in-line and transform functions.
- Types corresponding to a subset of C.
- Low-level control flow and explicit function calls.
- Expressions involving no function calls.

General Approach

- Visitor with state
- State keeps track of:
  - Code generated so far
  - Various counters for labels, etc
  - Mode: LHS, RHS, compile-time-constant RHS
- Can also keep track of
  - Externals used/defined
- Generate a sequence of HCode statements as a *side-effect* of the statement visit.

Framework

```
// HCodeGenerator implements CCodeVisitor
// and has methods to access state and to add
// new HCode statements and new temporaries.

HCodeGenerator hcgen = new HCodeGenerator(...);

// HCodeExprGenerator implements CCodeExprVisitor
// and has methods to visit expressions.
// Assignments, function calls, &&, ||, ?:, comma
  expressions
// add HCode temporaries and HCode assignments to them.

HCodeExprGenerator hcexprgen = new
  HCodeExprGenerator(hcgen);
Hcgen.setExprGenerator(hcexprgen);
```

If Statement Example

- Input:

```
if (a + b > 3) {
     x = a;
     y = f(b–3) + a;
}
else {
     y = a;
     x = f(b-3) + a;
}
```

- Output:

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
GOTO L5;
L4:
x = a;
t5 = call f(b-3);
y = t5 + a;
L5:
```

If Statement Example – Step By Step

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     = hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2
labels = L1, L2, L3
hcList =
stuff

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2
labels = L1, L2, L3
hcList =
stuff

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3

hcList =

stuff

```
t3 = a + b > 3;
```

```
hcExprVisitor returns the
expr t3
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
GOTO L5;
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
GOTO L5;
L4:
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLabel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
GOTO L5;
L4:
x = a;
t5 = call f(b-3);
y = t5 + a;
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLebel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
GOTO L5;
L4:
x = a;
t5 = call f(b-3);
y = t5 + a;
L5:
```

If Statement Example

```
// Pseudocode
visit(CCodeStatIf cc) {
 HCodeExpr  hex     =
hcExprVisitor.visit(cc._test);
 HCodeLabel thenLab = generateNewLabel();
 HCodeLabel endLab  = generateNewLabel();
 emit new HCodeStatIf(hex, thenLab);
 if (cc._optElstat != null) visit(cc._optElstat);
 emit new HCodeStatGoto(endLab);
 emit new HCodeStatLabel(thenLab);
 visit(cc._thstat);
 emit new HCodeStatLebel(endLab);
}
```

- **Input:**

```
if (a + b > 3) {
      x = a;
      y = f(b–3) + a;
}
else {
      y = a;
      x = f(b-3) + a;
}
```

- **Output:**

temps = t1, t2, t3

labels = L1, L2, L3, L4, L5

hcList =

stuff

```
t3 = a + b > 3;
IF (t3) L4;
y = a;
t4 = call f(b-3);
x = t4 + a;
GOTO L5;
L4:
x = a;
t5 = call f(b-3);
y = t5 + a;
L5:
```

Other Control Structures (on blackboard)

- For loops
  - Maintain a stack for “continue” and “break” labels.
  - If no “update”, assign “top” label to “continue” label variable.
  - While and do-while loops: same.

- Continue and Break statements
  - Goto corresponding label (stored in code generator state)

Other Control Structures (on blackboard)

- Switch: switch (e) s;

  LJ = generate new label for jump computation.

  TJ = generate new variable for switch expression.

  Cases = empty new array for (const, label) pairs

  LD = generate new “default” label.

  LB = generate new “break” label.

  emit assign TJ = genExpr(e);
  emit goto LJ
  genStat(s);
  emit goto LB;
  emit LJ:
  // Control logic
  for (const, label) in Cases do emit if t == const goto label
  if a default label was encountered, emit goto LD
  emit LB:

Other Control Structures (on blackboard)

- Switch alternatives
  - Linear sequence of ifs
  - Binary tree of ifs
  - Binary search on array of [Const, Label] pairs
  - Direct indexing using t-MinLabel into array of jump addresses
  - Combination of the above

Expressions (on blackboard)

- Simple tree generation for simple operators such as +, -, *, &, |
- Side effects of emitting code for more complex expressions

const	return Hconst
op a	return Hop(visit(a))
a op b	return Hop(visit(a), visit(b))
f(a,b,...)	emit temp = call visit(f) (visit(a), visit(b),...) return temp
a = expr	temp = visitLHS(a) emit temp = visit(expr) return asRhs(temp)
a && b	emit temp = 0 emit { if (visit(a)) if (visit(b)) temp = 1; } return temp
a || b	like a && b, but temp =1 and { if (!visit(a)) if (!visit(b)) temp = 0 }
a ? b : c	
a , b	

a+= expr a++ ++a

HCode Visitor – Wrap Up

- The HCode sequences for the “then” part and “else” part of the “if” are generated the same way.
- Labels and temporaries are generated as needed.
- Generation of code for expressions may either
  - just *return* an HCodeExpr, or
  - have the side effect of generating temporaries and assignments to them in the statement list, and then *returning* an expression involving the temporaries.