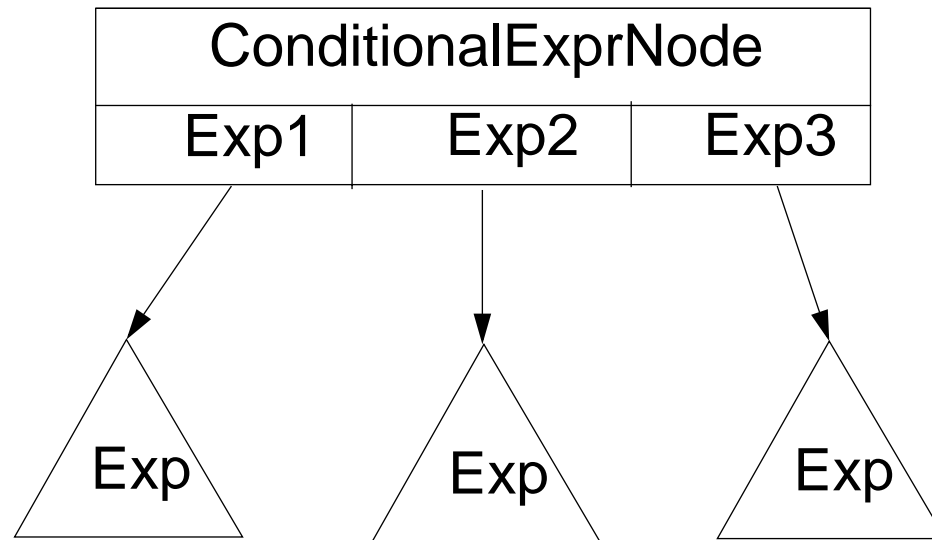


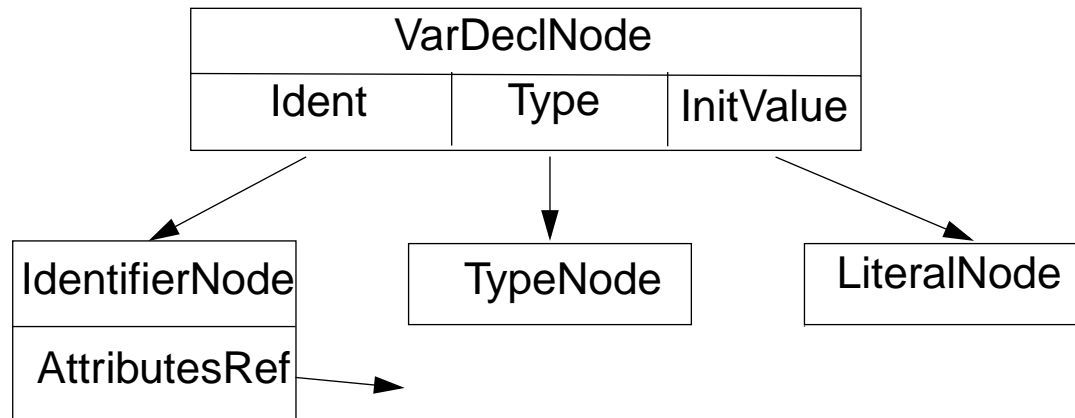
Simple Code Generation

"Translation"



1. Evaluate `Exp1`.
2. If `Exp1` is zero go to step 5.
3. Evaluate `ans ← Exp2`.
4. Go to step 6.
5. Evaluate `ans ← Exp3`.
6. Use the value computed in `ans` as the result of the conditional expression.

Variable Declarations



`VarDeclNode.CodeGen ()`

1. if `Ident.Address.AccessMode = Global`
2. then `Ident.Address.Label ← CreateUniqueLabel()`
3. `GenGlobalDecl(Ident.Address.Label, Type, InitValue)`
4. else `Ident.Address.Offset ← GenLocalDecl(Type)`
 if `InitValue ≠ null`
5. then • *Generate code to store InitValue in Ident*

Code Generation Support Routines

GetTemp(Type)

FreeTemp(Address1, Address2, ...)

GenLoadGlobal(Result, Type, Label)

GenLoadLocal(Result, Type, Offset),

GenLoadLiteral(Result, Type, Value).

IdentifierNode.CodeGen ()

1. if Address.AccessMode = Global
2. then Result \leftarrow GenLoadGlobal(Result, Type, Address.Label)
3. elsif Address.AccessMode = Local
4. then Result \leftarrow GenLoadLocal(Result, Type, Address.Offset)
5. elsif Address.AccessMode = Literal
6. then if Result \neq null
7. then Result \leftarrow GenLoadLiteral(Result, Type, Address.Value)
8. else Result \leftarrow AddressNode(Literal, Address.Value)

CodeGen Example

PlusNode.CodeGen()

1. LeftOperand.CodeGen()
2. RightOperand.CodeGen()
3. Result \leftarrow GenAdd(Result, Type,
LeftOperand.Result, RightOperand.Result)
4. FreeTemp(LeftOperand.Result, RightOperand.Result)

Following is JVM code for $A+B+3$ where A is a global integer variable with a label of L1 in class C and B is a local integer variable assigned a local index (an offset within the frame) of 4:

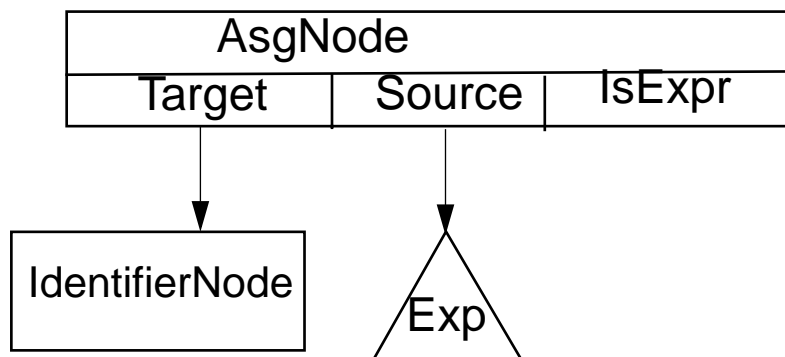
JVM Code	Comments	Generated By
<code>getstatic C/L1 I</code>	Push static integer field onto stack	GenLoadGlobal
<code>iload 4</code>	Push integer local 4 onto stack	GenLoadLocal
<code>iadd</code>	Add top two stack locations	GenAdd
<code>iconst_3</code>	Push integer literal 3 onto stack	GenLoadLiteral
<code>iadd</code>	Add top two stack locations	GenAdd

CodeGen for Conditionals

Code for $A < B$ where A and B are local integer variables with frame offsets 2 and 3:

```
    iload 2      ; Push local #2 (A) onto the stack
    iload 3      ; Push local #3 (B) onto the stack
    if_icmplt L1 ; Goto L1 if A < B
    iconst_0     ; Push 0 (false) onto the stack
    goto L2      ; Skip around next instruction
L1:  iconst_1     ; Push 1 (true) onto the stack
L2:
```

Assignment



AsgNode.CodeGen()

1. Source.CodeGen()
2. if Source.Result.AccessMode \in {JumpIfTrue, JumpIfFalse}
3. then Result \leftarrow ConvertFromJumpCode (Result,
Source.Result.AccessMode, Source.Result.Label)
4. else Result \leftarrow Source.Result
5. if Target.Address.AccessMode = Global
6. then GenStoreGlobal(Result, Type, Target.Address.Label, IsExpr)
7. else GenStoreLocal(Result, Type, Target.Address.Offset, IsExpr)

Assignment Example

Consider $A = B = C + 1$ where A , B and C are local integers with frame indices of 1, 2 and 3 respectively. First, $C + 1$ is evaluated. Since $B = C + 1$ is used as an expression, the value of $C + 1$ is duplicated (and thereby preserved) so that it can also be assigned to A . The JVM code generated is:

```
iload 3 ; Push local #3 (C) onto the stack
iconst_1 ; Push 1 onto the stack
iadd ; Compute C + 1
dup ; Duplicate C + 1 for second store
istore 2 ; Store top of stack into local #2 (B)
istore 1 ; Store top of stack into local #1 (A)
```