

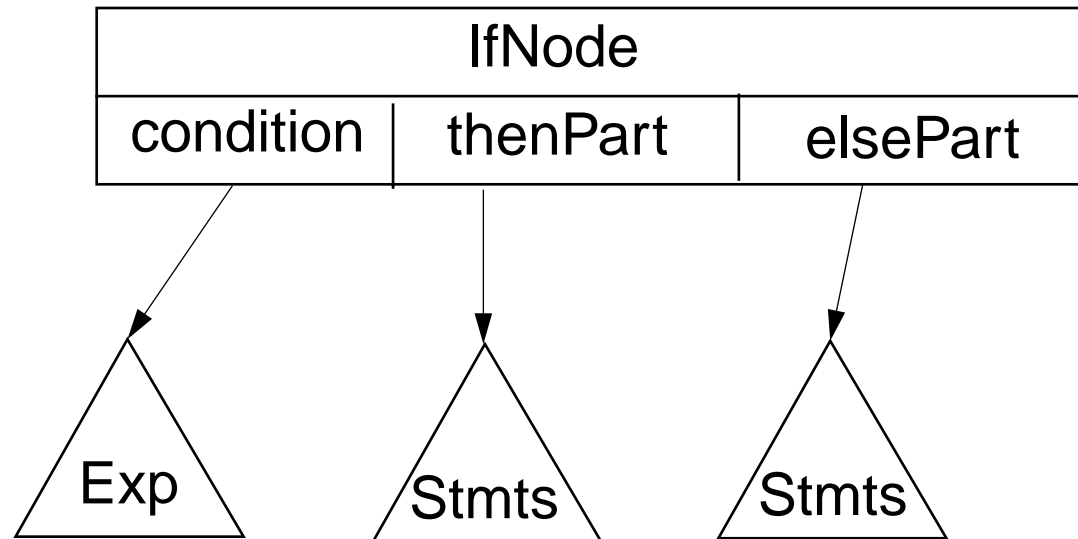
Code Generation for Control Structures

If Statements

Code Template:

1. If condition is false go to 4.
2. Execute statements in thenPart.
3. Go to 5.
4. Execute statements in elsePart.
5. Statements following the if statement.

AST Node Structure:



Code Generation:

```
IfNode.CodeGen ( )
1. ElseLabel ← CreateUniqueLabel()
2. condition.Result ← AddressNode(JumpIfFalse, ElseLabel)
3. condition.CodeGen()
4. thenPart.CodeGen()
5. OutLabel ← CreateUniqueLabel()
6. GenGoTo(OutLabel)
7. GenLabel(ElseLabel)
8. elsePart.CodeGen()
9. GenLabel(OutLabel)
```

Example:

Consider `if (b) a=1; else a=2;`

Assume `a` and `b` are local variables with indices of 1 and 2. We know `b` is a boolean, limited to either 1 (true) or 0 (false). The code we generate is:

```
iload    2 ; Push local #2 (b) onto the stack
ifeq     L1; Goto L1 if b is 0 (false)
iconst_1 ; Push 1
istore   1 ; Store stack top into local #1 (a)
goto     L2; Skip around else statements
L1:iconst_2 ; Push 2
    istore   1 ; Store stack top into local #1 (a)
L2:
```

Special case -- no else part:

```
if (b) a=1;
```

will result in code with an extra jump:

```
iload    2 ; Push local #2 (B) onto the stack
ifeq     L1; Goto L1 if B is 0 (false)
iconst_1 ; Push 1
istore   1 ; Store stack top into local #1 (a)
goto     L2; Skip around else statements
```

L1:

L2:

Improved Code Generation:

```
fNode.CodeGen( )
```

1. ElseLabel ← CreateUniqueLabel()
2. condition.Result ← AddressNode(JumpIfFalse, ElseLabel)
3. condition.CodeGen()
4. thenPart.CodeGen()
5. if elsePart = null
6. then GenLabel(ElseLabel)
7. else OutLabel ← CreateUniqueLabel()
8. GenGoTo(OutLabel)
9. GenLabel(ElseLabel)
10. elsePart.CodeGen()
11. GenLabel(OutLabel)

While Loops

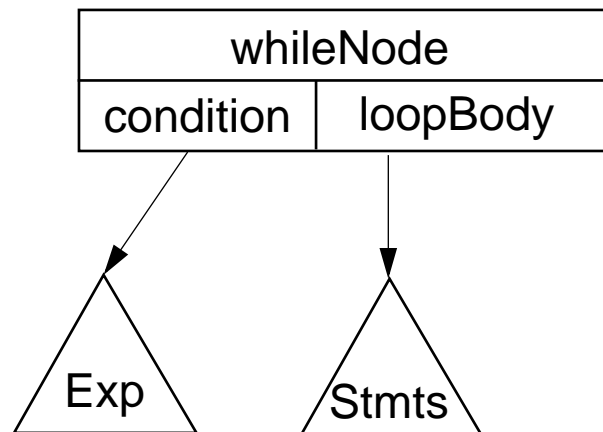
Code Template:

1. If condition is false go to 4.
2. Execute statements in loopBody.
3. Go to 1.
4. Statements following the while loop.

Better version:

1. Go to 3.
2. Execute statements in loopBody.
3. If condition is true go to 2.

AST Node Structure:



Code Generation:

- ```
WhileNode.CodeGen()
```
1. ConditionLabel ← getContinueLabel()
  2. GenGoTo(ConditionLabel)
  3. TopLabel ← CreateUniqueLabel()
  4. GenLabel(TopLabel)
  5. loopBody.CodeGen()
  6. GenLabel(ConditionLabel)
  7. condition.Result ← AddressNode(JumpIfTrue, TopLabel)
  8. condition.CodeGen()

## Example:

In this while loop, I is an integer local with an index of 2 and L1 is chosen as the ConditionLabel

```
while (I >= 0) { I--; }
```

The JVM code generated is:

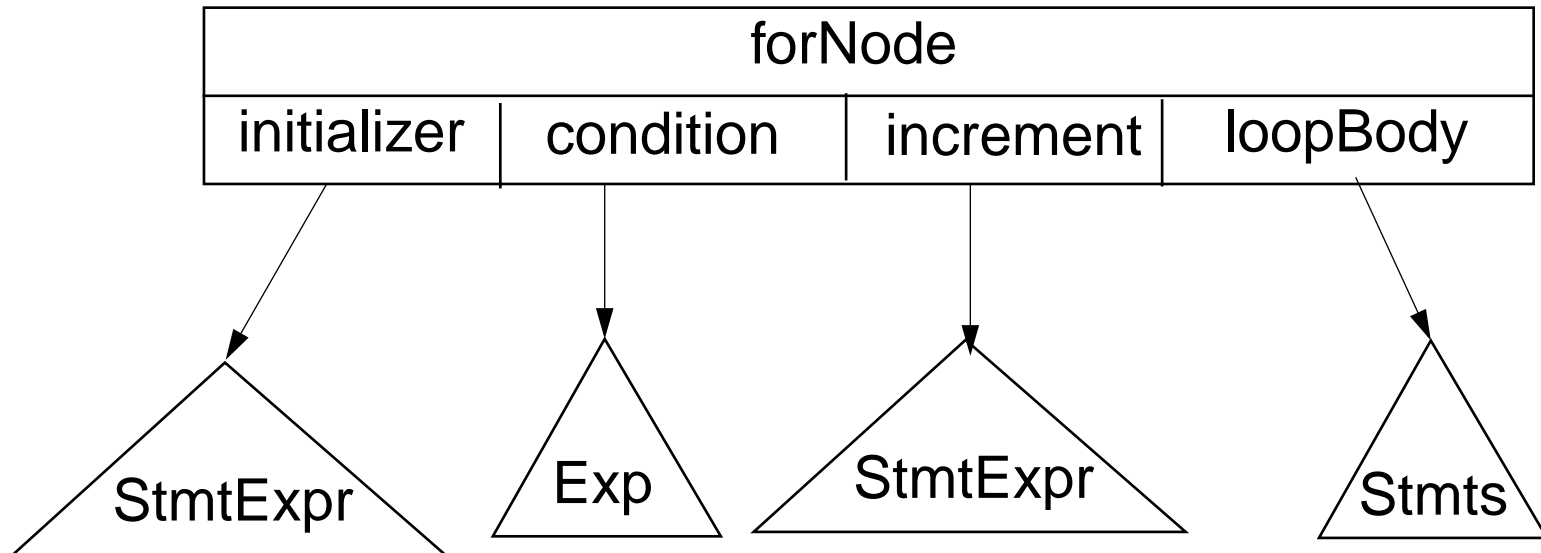
```
goto L1; Skip around loop body
L2:
 iload 2 ; Push local #2 (I) onto the stack
 iconst_1 ; Push 1
 isub ; Compute I-1
 istore 2 ; Store I-1 into local #2 (I)
L1:
 iload 2 ; Push local #2 (I) onto the stack
 ifge L2; Goto L2 if I is >= 0
```

# C-style For Loops

## Code Template:

1. Execute initializer code.
2. Go to 5.
3. Execute statements in loopBody.
4. Execute increment code.
5. If condition is true go to 3.

## AST Node Structure



## Code Generation:

- ```
ForNode.CodeGen ( )
1. initializer.CodeGen()
2. SkipLabel ← CreateUniqueLabel()
3. GenGoTo(SkipLabel)
4. TopLabel ← CreateUniqueLabel()
5. GenLabel(TopLabel)
6. loopBody.CodeGen()
7. ContinueLabel ← getContinueLabel()
8. GenLabel(ContinueLabel)
9. increment.CodeGen()
10. GenLabel(SkipLabel)
11. if condition = null
12. then GenGoTo(TopLabel)
13. else condition.Result ← AddressNode(JumpIfTrue, TopLabel)
14.      condition.CodeGen()
```

Special Case Example:

The special case of

```
for ( ; ; ) { }
```

generates:

```
        goto    L1
L1: L2: L3:
        goto    L2
```

Example:

```
for (i=100;i!=0;i--) {  
    j = i;  
}
```

If *i* and *j* are locals with variable indices of 1 and 2, the JVM code we generate is

```
ipush  100; Push 100 onto the stack  
istore 1 ; Store 100 into local #1 (i)  
goto   L1; skip around loop body and increment  
L2:  
  iload 1 ; Push local #1 (i) onto the stack  
  istore 2 ; Store i into local #2 (j)  
L3:      ; Target label for continue statements  
  iload 1 ; Push local #1 (i) onto the stack  
  iconst_1 ; Push 1  
  isub      ; Compute i-1  
  istore 1 ; Store i-1 into local #1 (i)  
L1:  
  iload 1 ; Push local #1 (i) onto the stack  
  ifne  L2; Goto L2 if i is != 0
```