CS406: Compilers

Spring 2021

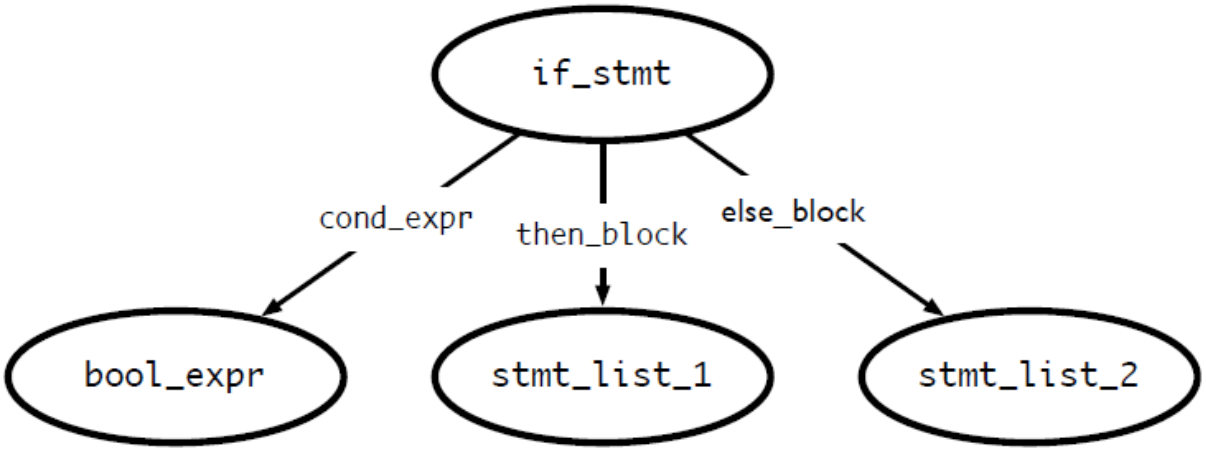
Week 9: IR code for if- statement, loops, switch, functions

If statements

```
if <bool_expr_1>
  <stmt_list_1>
else
  <stmt_list_2>
endif
```


Slide courtesy: Milind Kulkarni

If statements



Slide courtesy: Milind Kulkarni

Code-generation – if-statement

Program text	3AC
`INT a, b;`	

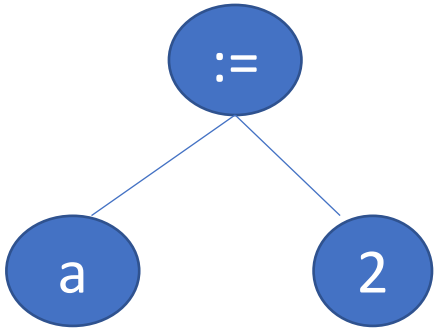
Code-generation – if-statement

Program text	3AC	
`INT a, b;`		Make entries in the symbol table

Code-generation – if-statement

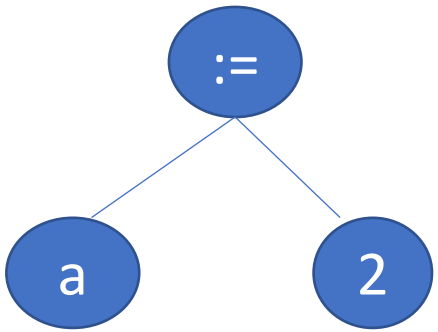
Program text	3AC
`INT a, b;` `a := 2;`	

Code-generation – if-statement

Program text	3AC
`INT a, b;` `a := 2;`	

1. “a” is left-child, type=l-val. No code generated. *Return an object containing identifier details after verifying that “a” is present in the symbol table.*

Code-generation – if-statement

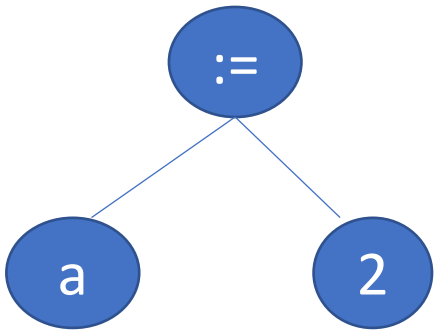
Program text	3AC
`INT a, b;` `a := 2;`	

1. “a” is left-child, type=l-val. No code generated. Pass up the identifier.
2. “2” is right-child, type=const. No code generated.

Code-generation – if-statement

Program text **3AC**

```
INT a, b;
a := 2;
```

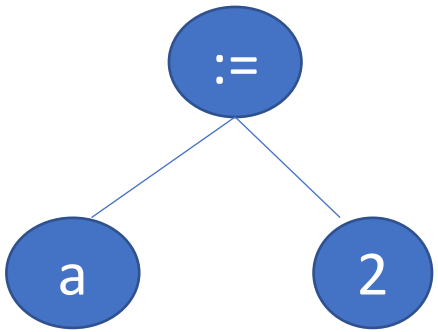


1. “a” is left-child, type=l-val. No code generated. Pass up the identifier.
2. “2” is right-child, type=const. No code generated.
3. Create a temporary T1 to store the result of the expression

Code-generation – if-statement

Program text **3AC**

```
INT a, b;
a := 2;
```



1. "a" is left-child, type=l-val. No code generated. Pass up the identifier.
2. "2" is right-child, type=const. No code generated.
3. Create a temporary T1 to store the result of the expression
   - Current node stores the op ':='. A call to `process_op` stores the RHS data in LHS

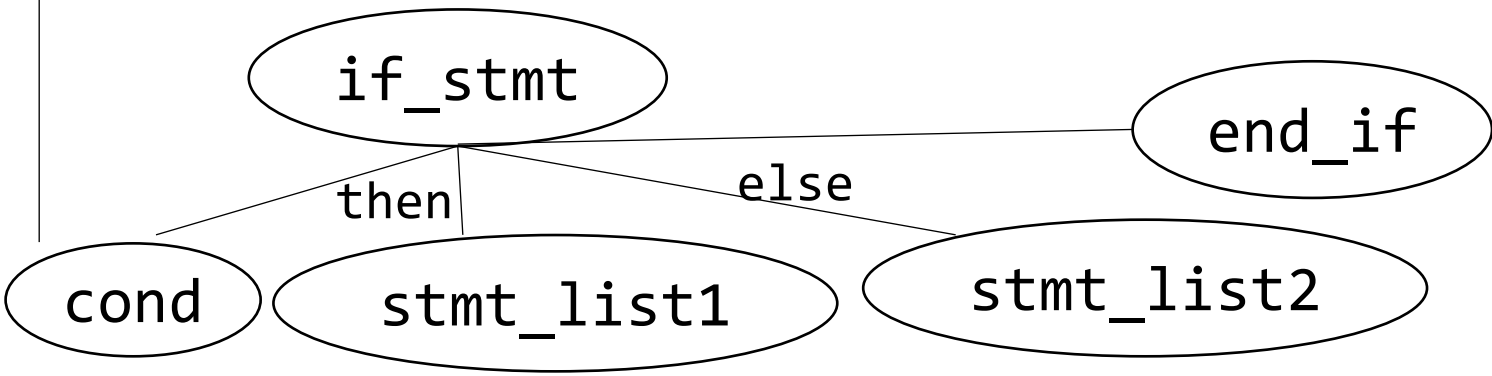
Code-generation – if-statement

Program text	3AC
`INT a, b;`	`STOREI 2 T1`
`a := 2;`	`STOREI T1 a`

Code-generation – if-statement

Program text	3AC

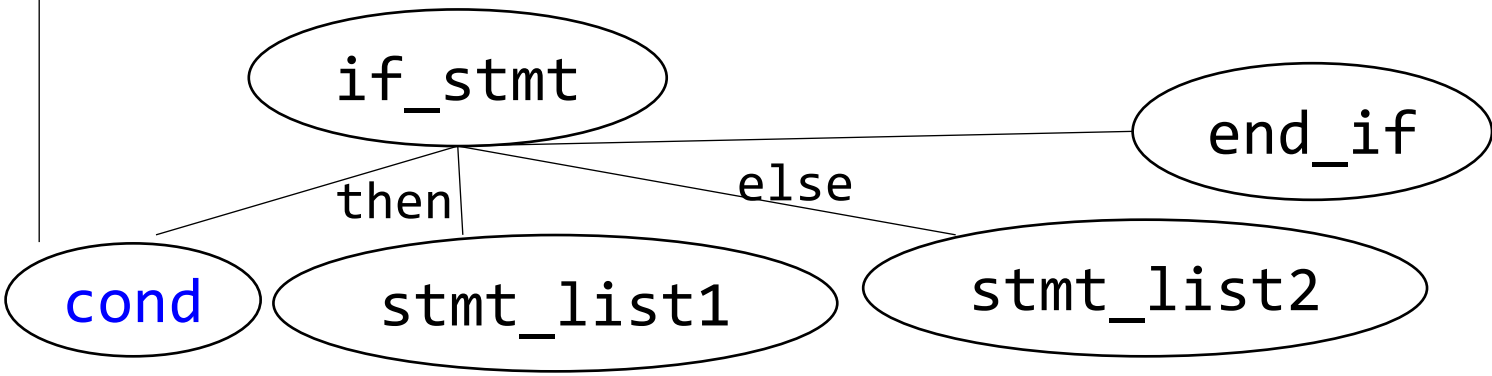
INT a, b;
a := 2;
IF (a = 1)
  b := 1;
ELSE
  b := 2;
ENDIF
	STOREI 2 T1
	STOREI T1 a



Code-generation – if-statement

Program text	3AC

INT a, b;
a := 2;
IF (a = 1)
  b := 1;
ELSE
  b := 2;
ENDIF
	STOREI 2 T1
	STOREI T1 a

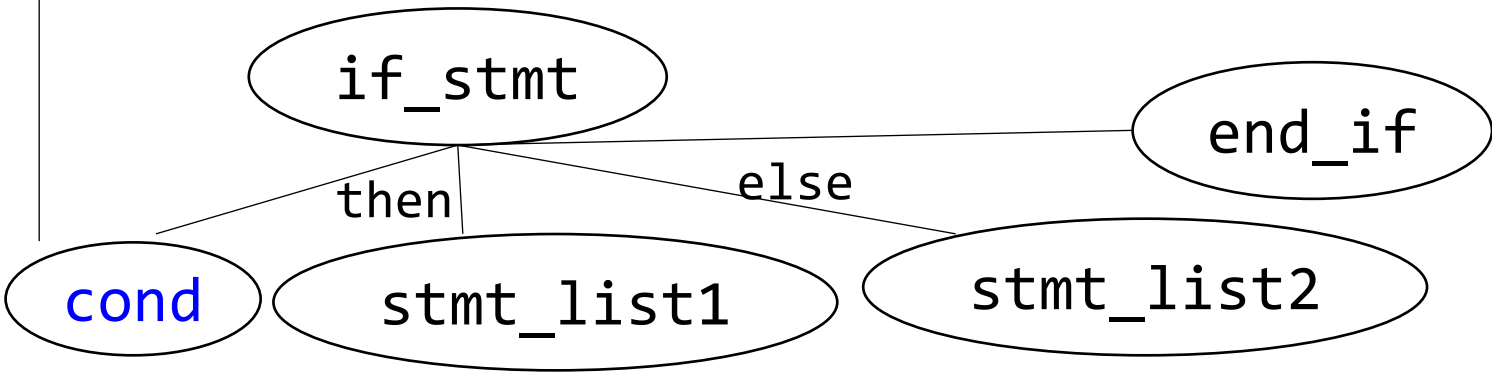


1. Generate code for cond

Code-generation – if-statement

Program text	3AC

INT a, b;
a := 2;
IF (a = 1)
  b := 1;
ELSE
  b := 2;
ENDIF
	STOREI 2 T1
	STOREI T1 a



1. Generate code for cond

```
STOREI 1 T2
NE a T2 label1
```

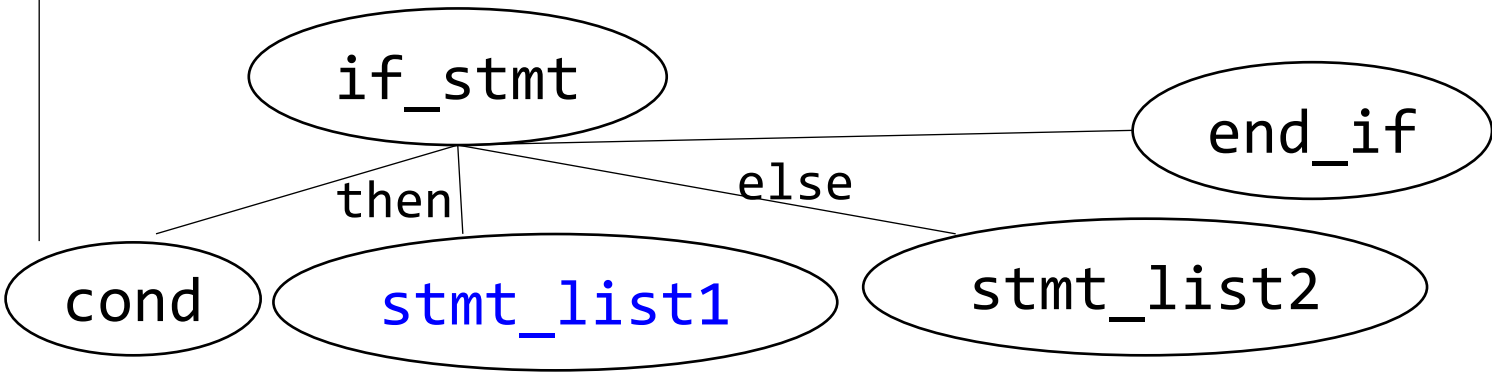
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```

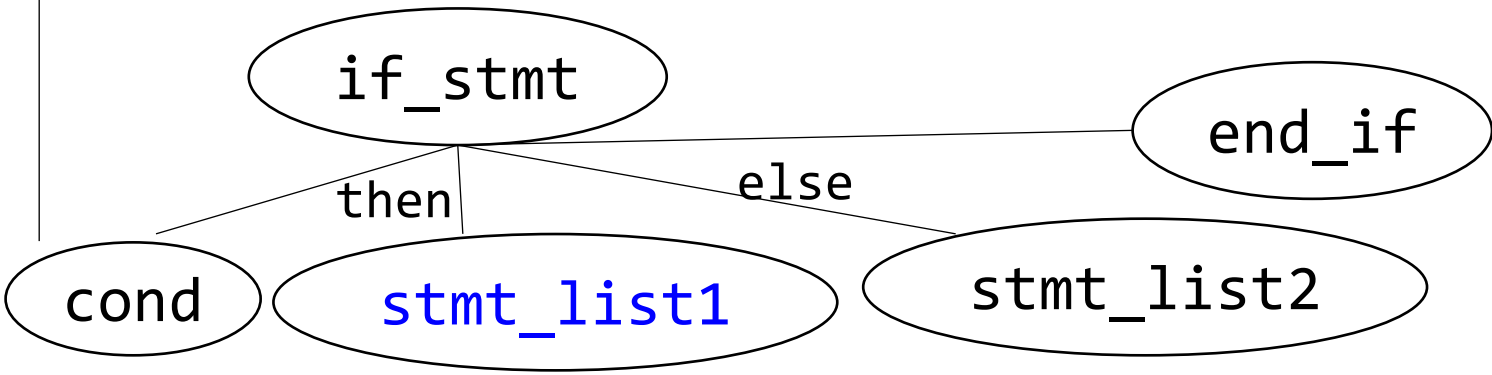


2. Generate code for stmt_list1

Code-generation – if-statement

Program text	3AC

INT a, b;
a := 2;
IF (a = 1)
  b := 1;
ELSE
  b := 2;
ENDIF
	STOREI 2 T1
	STOREI T1 a



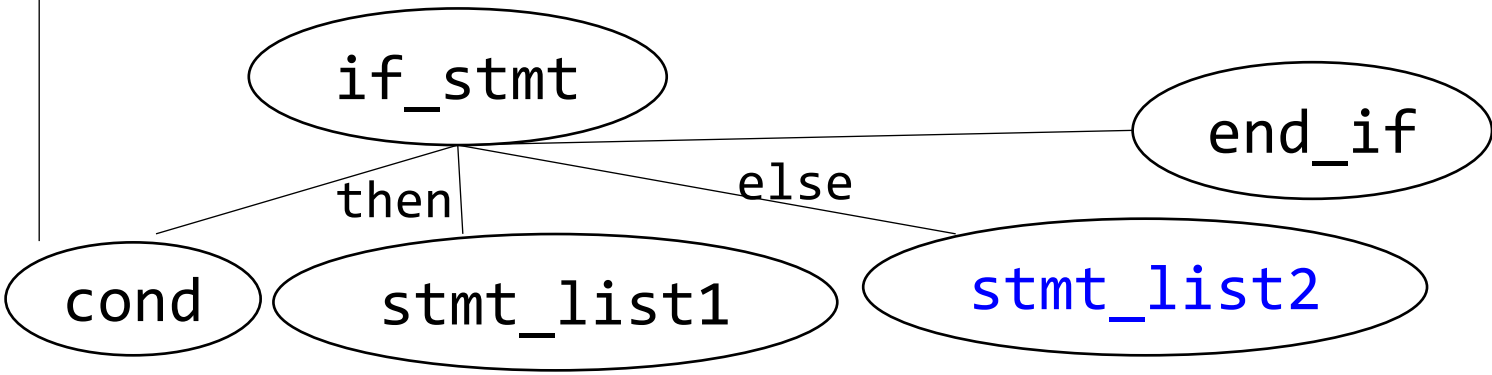
2. Generate code for stmt_list1

```
STOREI 1 T3
STOREI T3 b
JUMP label2
```

Code-generation – if-statement

Program text	3AC

INT a, b;
a := 2;
IF (a = 1)
  b := 1;
ELSE
  b := 2;
ENDIF
	STOREI 2 T1
	STOREI T1 a



3. Generate code for stmt_list2

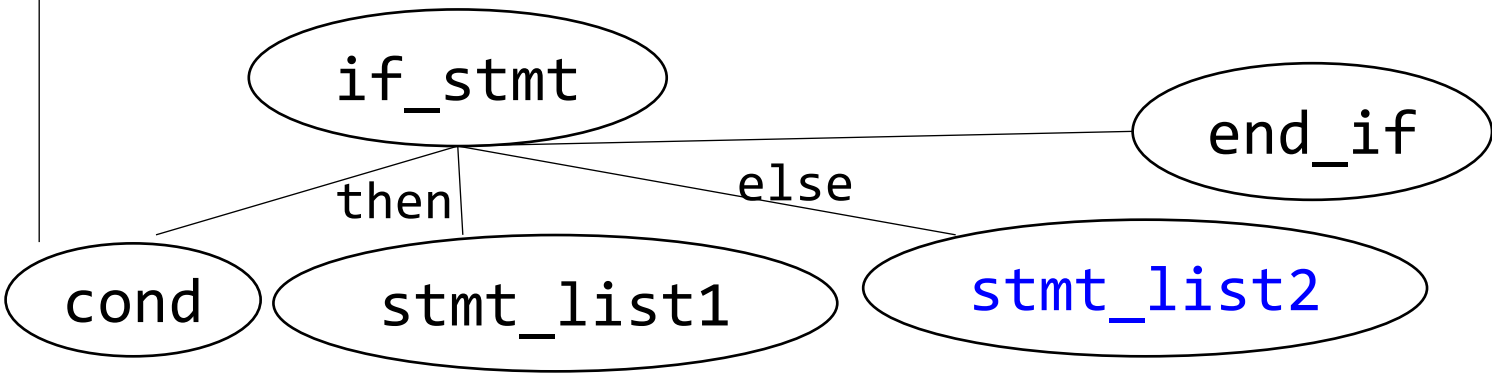
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



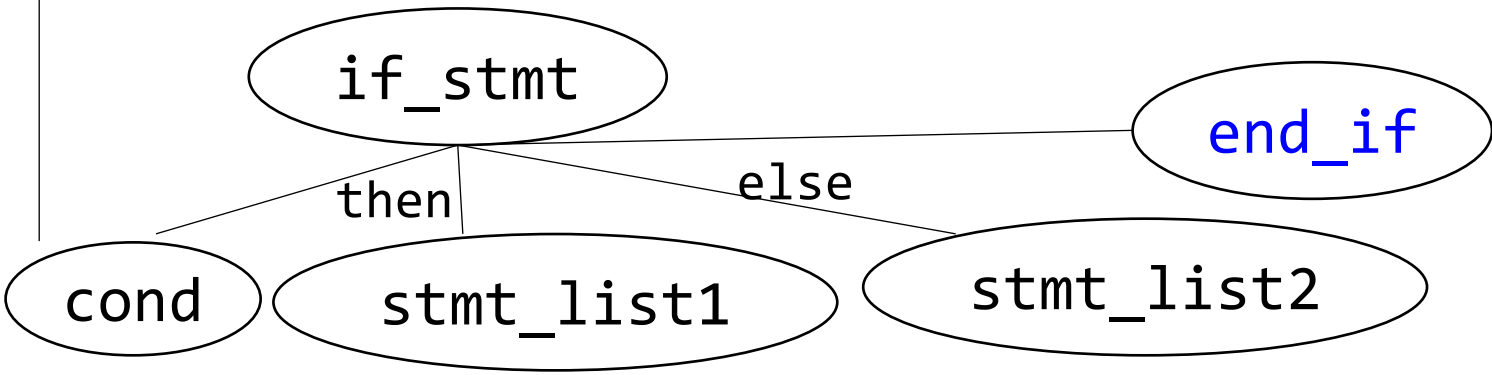
3. Generate code for stmt_list2

```
LABEL label1
STOREI 2 T4
STOREI T4 b
JUMP label2
```

Code-generation – if-statement

Program text	3AC

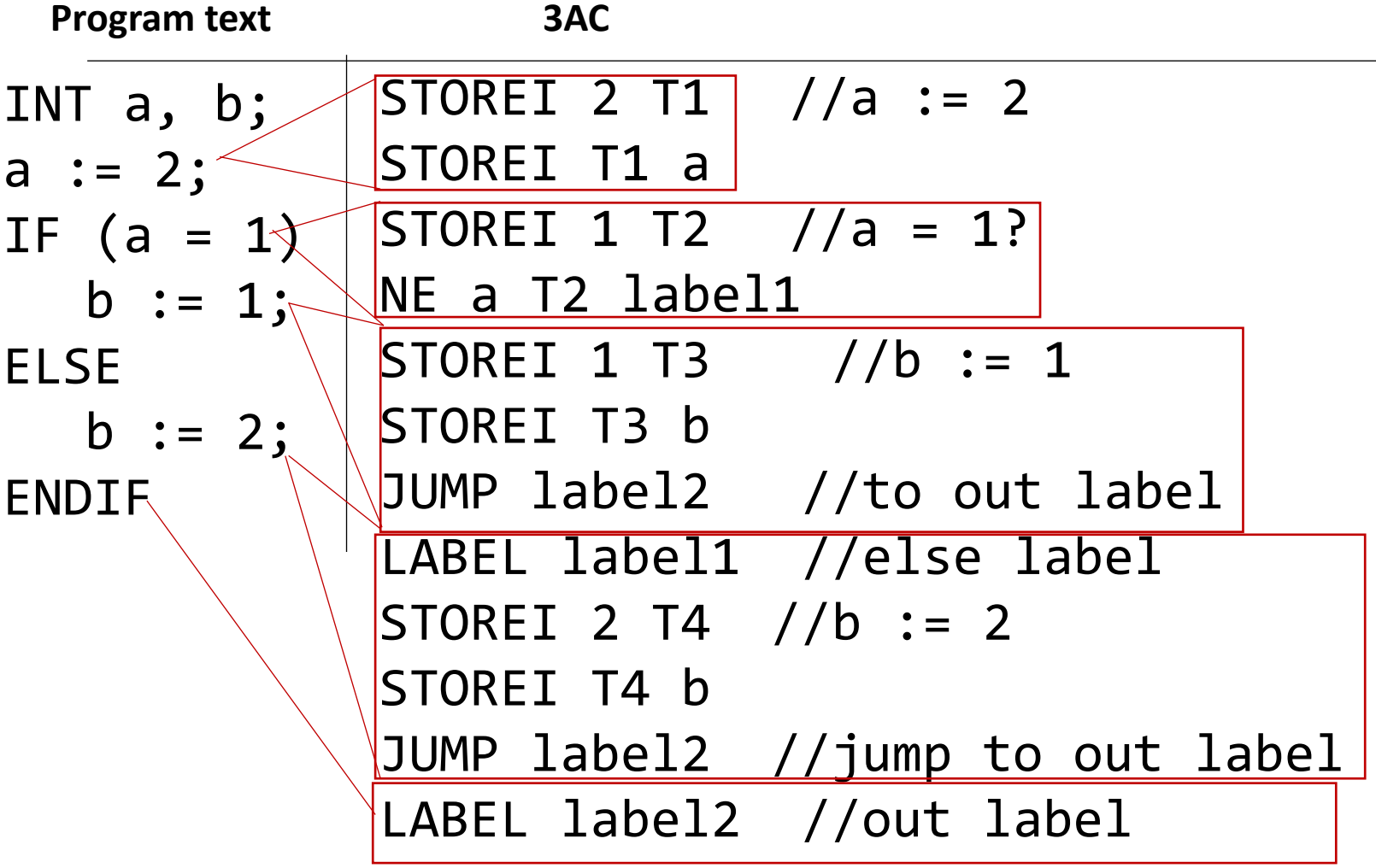
INT a, b;
a := 2;
IF (a = 1)
  b := 1;
ELSE
  b := 2;
ENDIF
	STOREI 2 T1
	STOREI T1 a



4. Generate code for end_if

LABEL label2

Code-generation – if-statement



Jumps and Labels?

- Who will generate labels?
- When will the labels be generated?
- To what addresses will the labels be associated with?

How are targets of jumps decided?

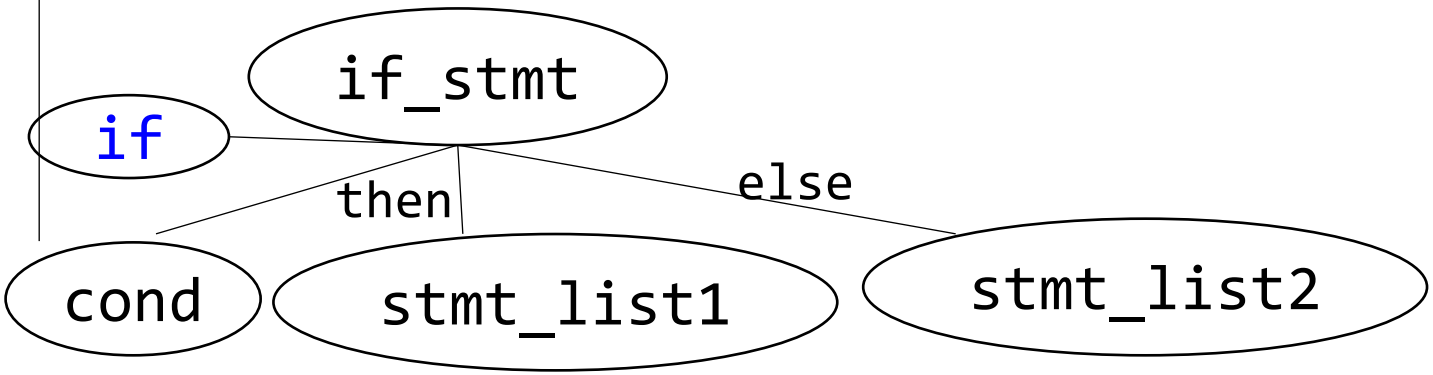
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



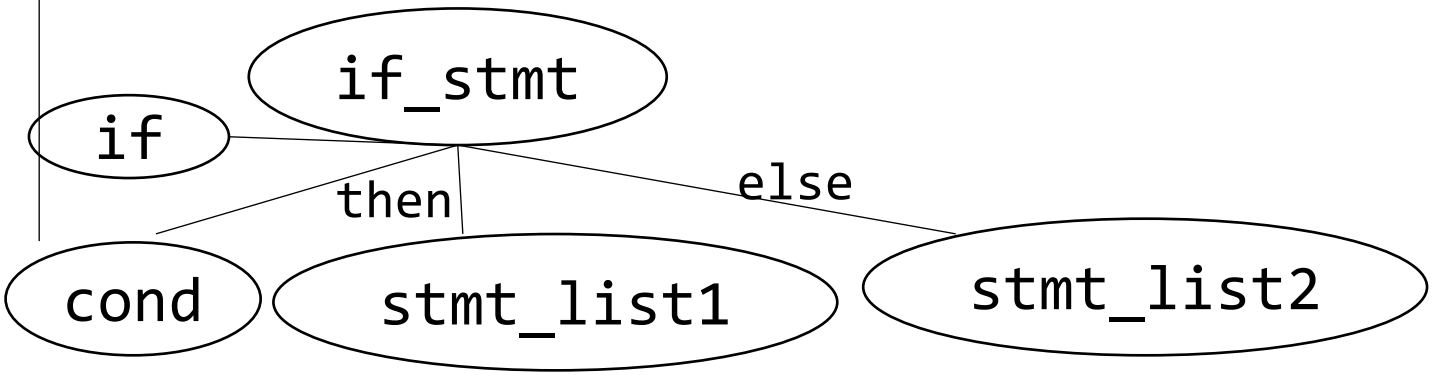
1 Generate out label and store it in semantic record of if_stmt (label2)

Code-generation – if-statement

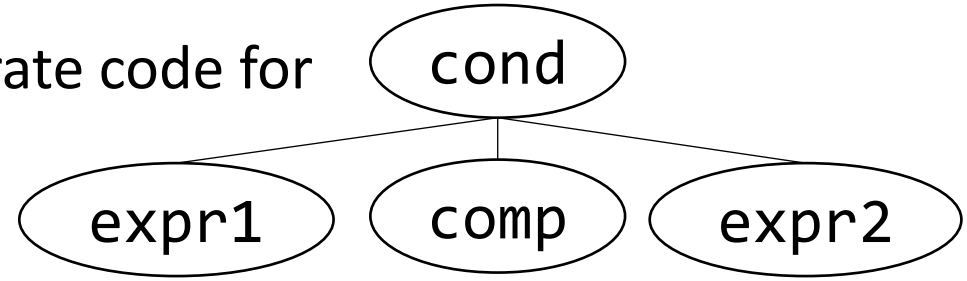
Program text | **3AC**

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

```
STOREI 2 T1
STOREI T1 a
```



2 Generate code for



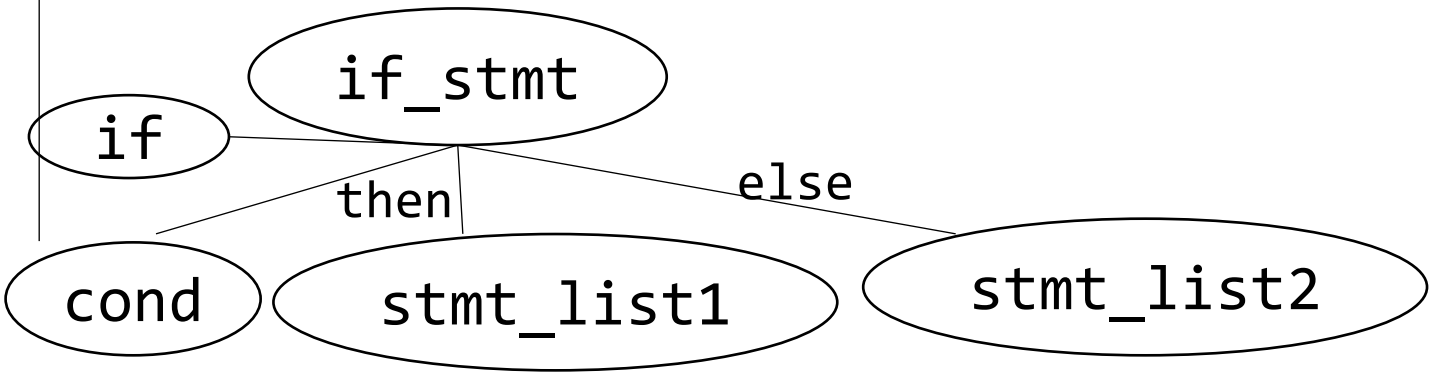

```
STOREI 1 T2
```

Code-generation – if-statement

Program text | **3AC**

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

```
STOREI 2 T1
STOREI T1 a
```



2. Store the result of calling `process_op`, where op is "=", in the node `cond` (`bool_expr1=false`)

```
STOREI 1 T2
```

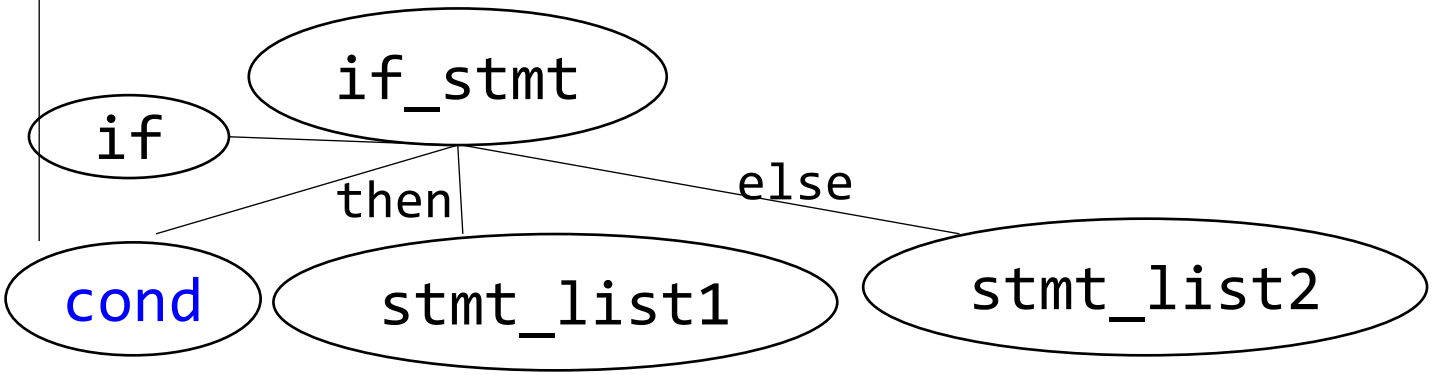
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



2. Cond has been matched. a) Generate label for else part (`label1`) b) generate statement: `JUMP0 bool_expr1 label1`

The generated statement conditionally jumps to the else part if cond is false.

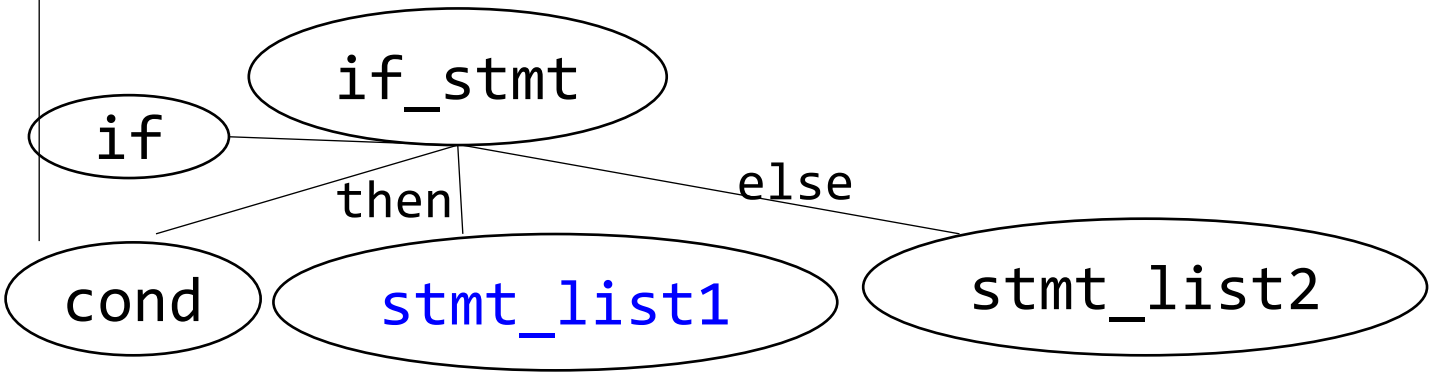
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



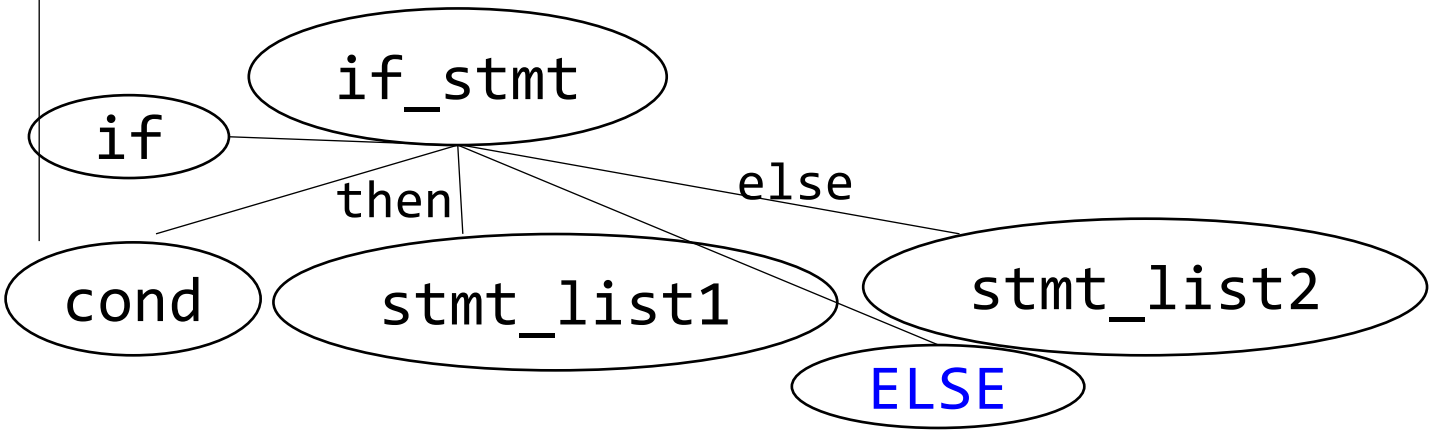
3. Generate code for `stmt_list1`

(`STOREI 1 T3`
`STOREI T3 b`)

Code-generation – if-statement

Program text | **3AC**

```
INT a, b;          STOREI 2 T1
a := 2;            STOREI T1 a
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```



4. Generate unconditional jump to out label (label2). `Label2` can be obtained from the semantic record of `if` (slide 26)

```
JUMP label2
```

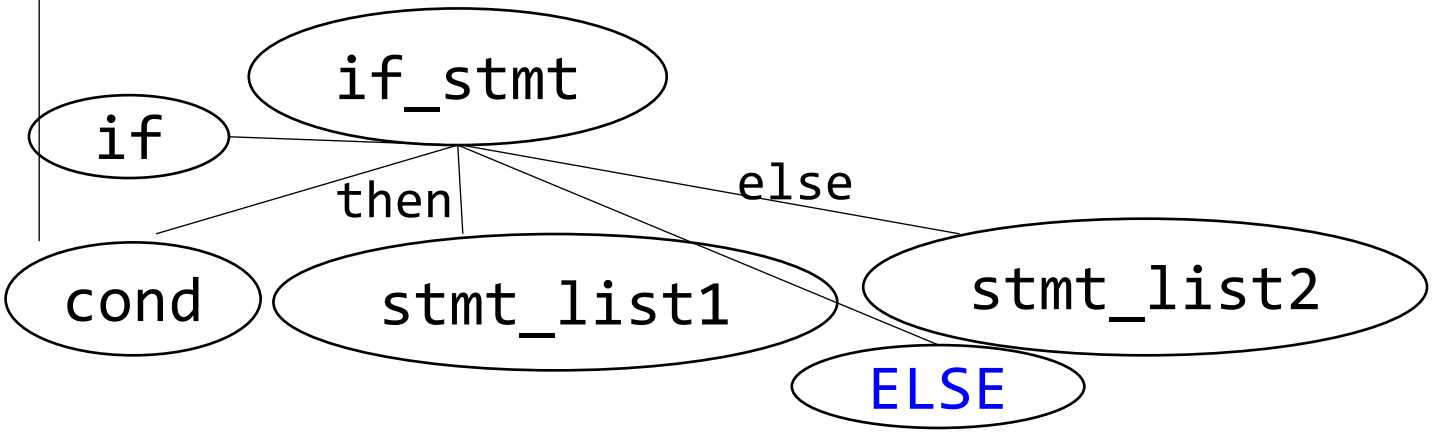
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



4. Associate else part label (`label1`) with address of next instruction i.e. generate a statement: `LABEL label1`
`Label1` can be obtained from semantic record of `if` updated by `cond (slide 29)`

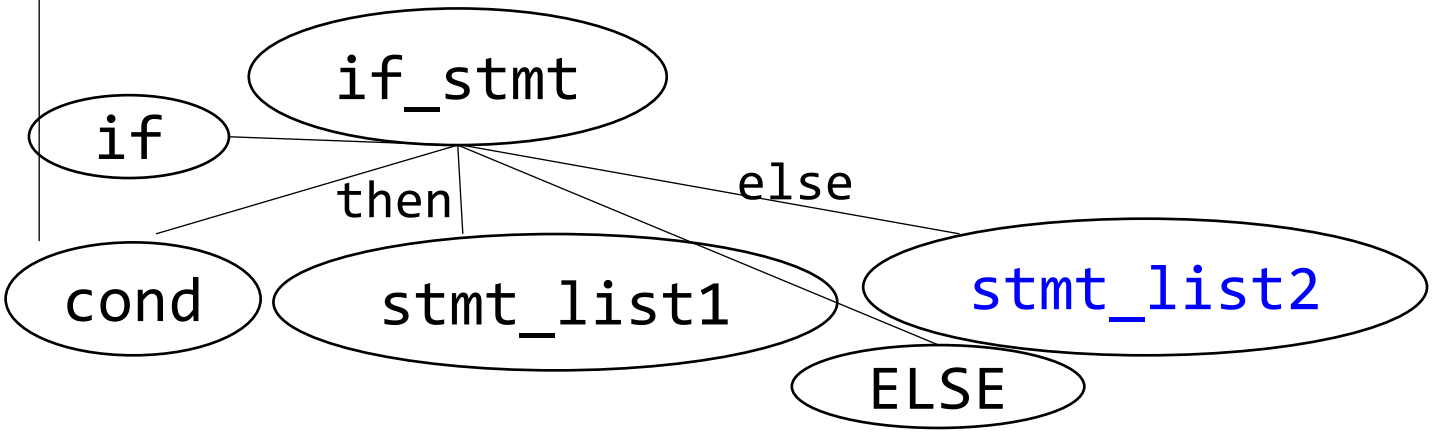
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



5. Generate code for `stmt_list2`

```
(STOREI 2 T4
STOREI T4 b)
```

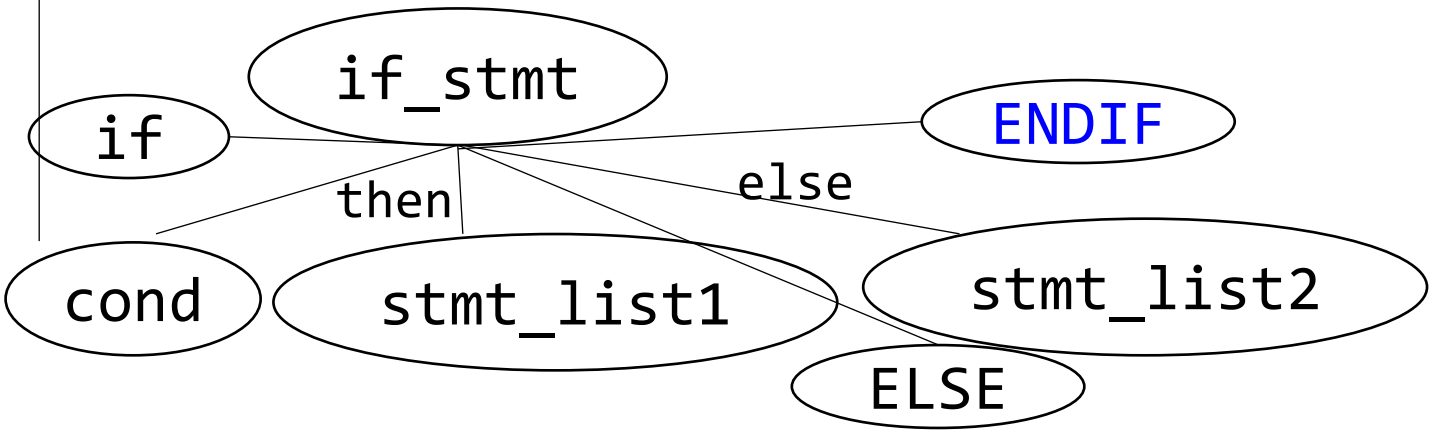
Code-generation – if-statement

Program text

```
INT a, b;
a := 2;
IF (a = 1)
   b := 1;
ELSE
   b := 2;
ENDIF
```

3AC

```
STOREI 2 T1
STOREI T1 a
```



5. Associate out label (`label2`) with address of next instruction
   i.e. generate a statement: `LABEL label2`

Observations

- We added tokens IF, ELSE, ENDIF to AST
- Generated code is equivalent but not exact
  - e.g. “NE a T2 label1” is replaced with an equivalent “JUMP0 bool_expr label1”
- Done in one pass

Will this approach work when generating machine code directly?

Generating code for ifs

```
if <bool_expr_1>
  <stmt_list_1>
else
  <stmt_list_2>
endif
```

```
  <code for bool_expr_1>
  j<!op> ELSE_1
  <code for stmt_list_1>
  jmp OUT_1
ELSE_1:
  <code for stmt_list_2>
OUT_1:
```


Slide courtesy: Milind Kulkarni

Notes on code generation

- The <op> in j<!op> is dependent on the type of comparison you are doing in <bool_expr>
- When you generate JUMP instructions, you should also generate the appropriate LABELs
- Remember: labels have to be unique!

Slide courtesy: Milind Kulkarni

do-while

- `do{S}while(B);` `//S is executed at least once and again and again and again... while B remains true`

do-while

- do{S}while(B); //S is executed at least once and again and again and again... while B remains true

```
LOOP:
   <stmt_list>
   <bool_expr>
   j<!op> OUT
   jmp LOOP
OUT:
```

repeat-until

- `repeat{S}until(B);` `//S is executed at least once and again and again and again... while B remains false`

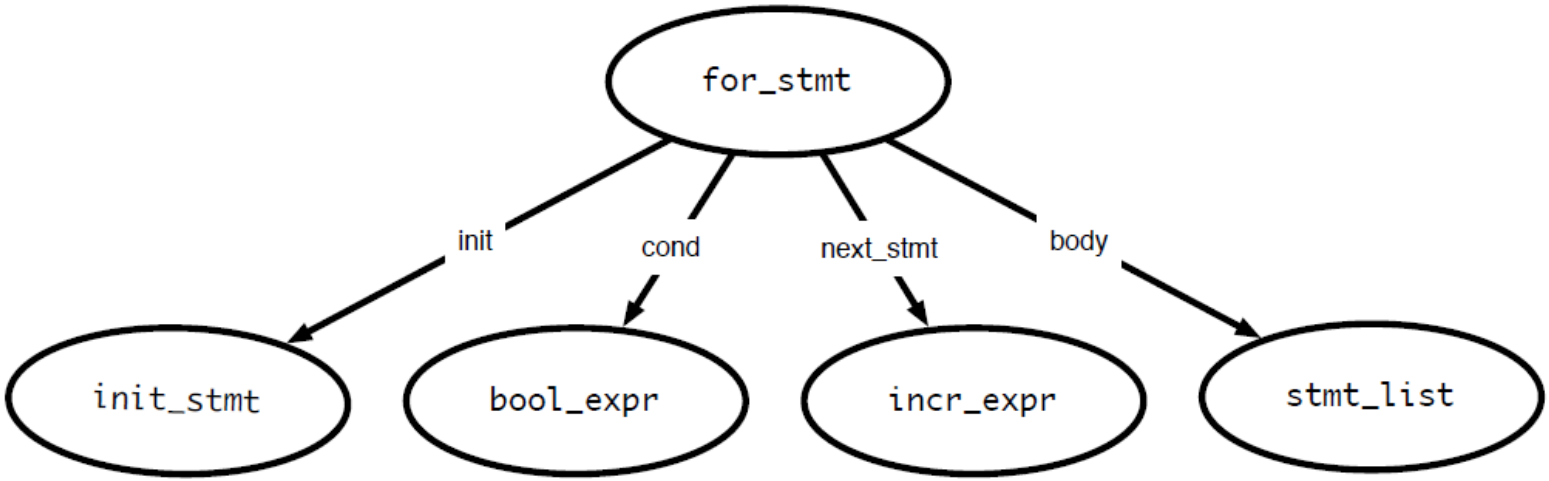
repeat-until

- `repeat{S}until(B);` `//S is executed at least once and again and again and again... while B remains false`

```
LOOP:
   <stmt_list>
   <bool_expr>
   j<!op> LOOP
OUT:
```

For loops

```
for (<init_stmt>;<bool_expr>;<incr_stmt>)
  <stmt_list>
end
```



Slide courtesy: Milind Kulkarni

Generating code: for loops

```
for (<init_stmt>;<bool_expr>;<incr_stmt>)
  <stmt_list>
end
```

```
  <init_stmt>
LOOP:
  <bool_expr>
  j<!op> OUT
  <stmt_list>
INCR:
  <incr_stmt>
  jmp LOOP
OUT:
```

- Execute init_stmt first
- Jump out of loop if bool_expr is false
- Execute incr_stmt after block, jump back to top of loop
- Question: Why do we have the INCR label?

Slide courtesy: Milind Kulkarni