

CSL302: Compiler Design

Intermediate Code Generation

Vishwesh Jatala

Assistant Professor

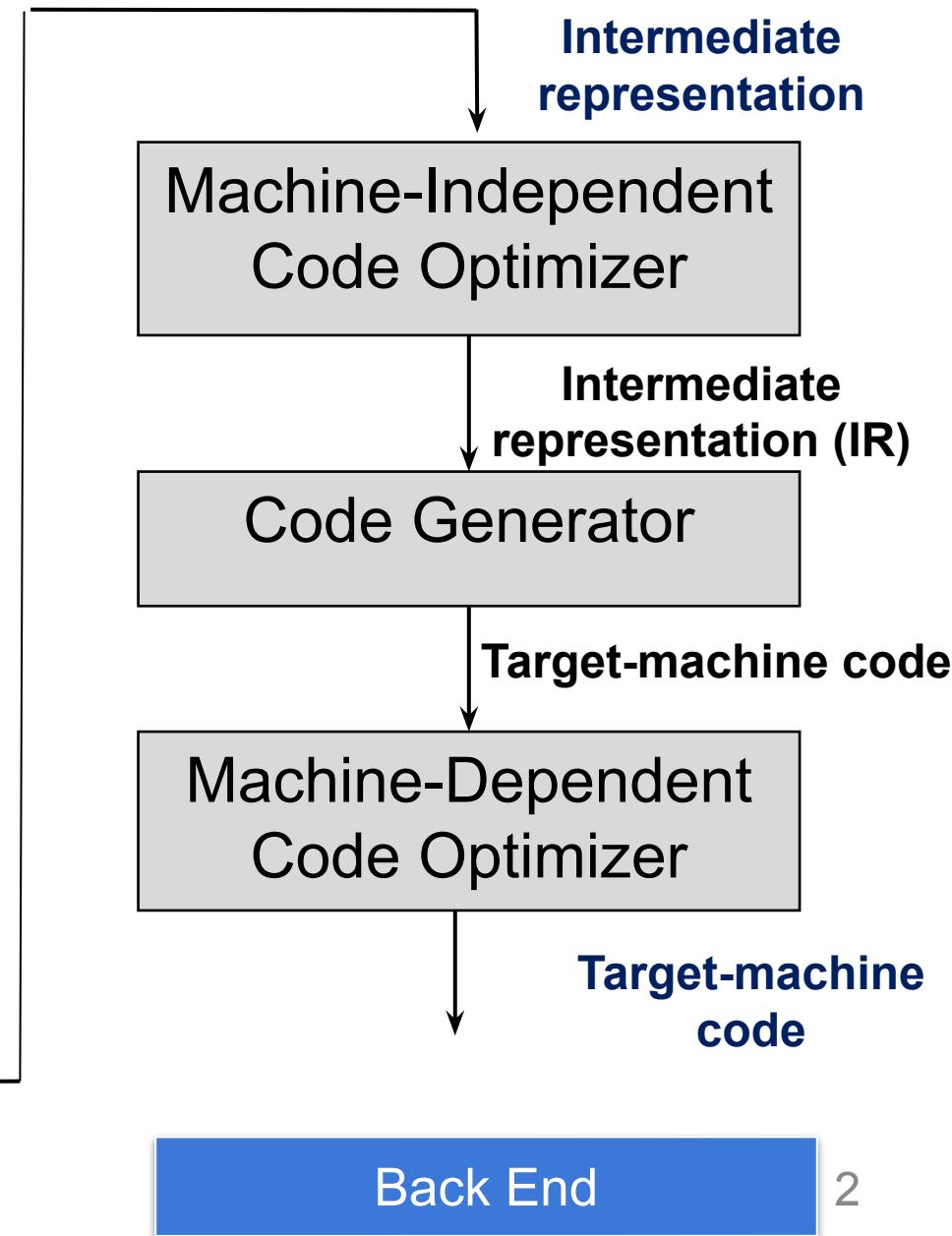
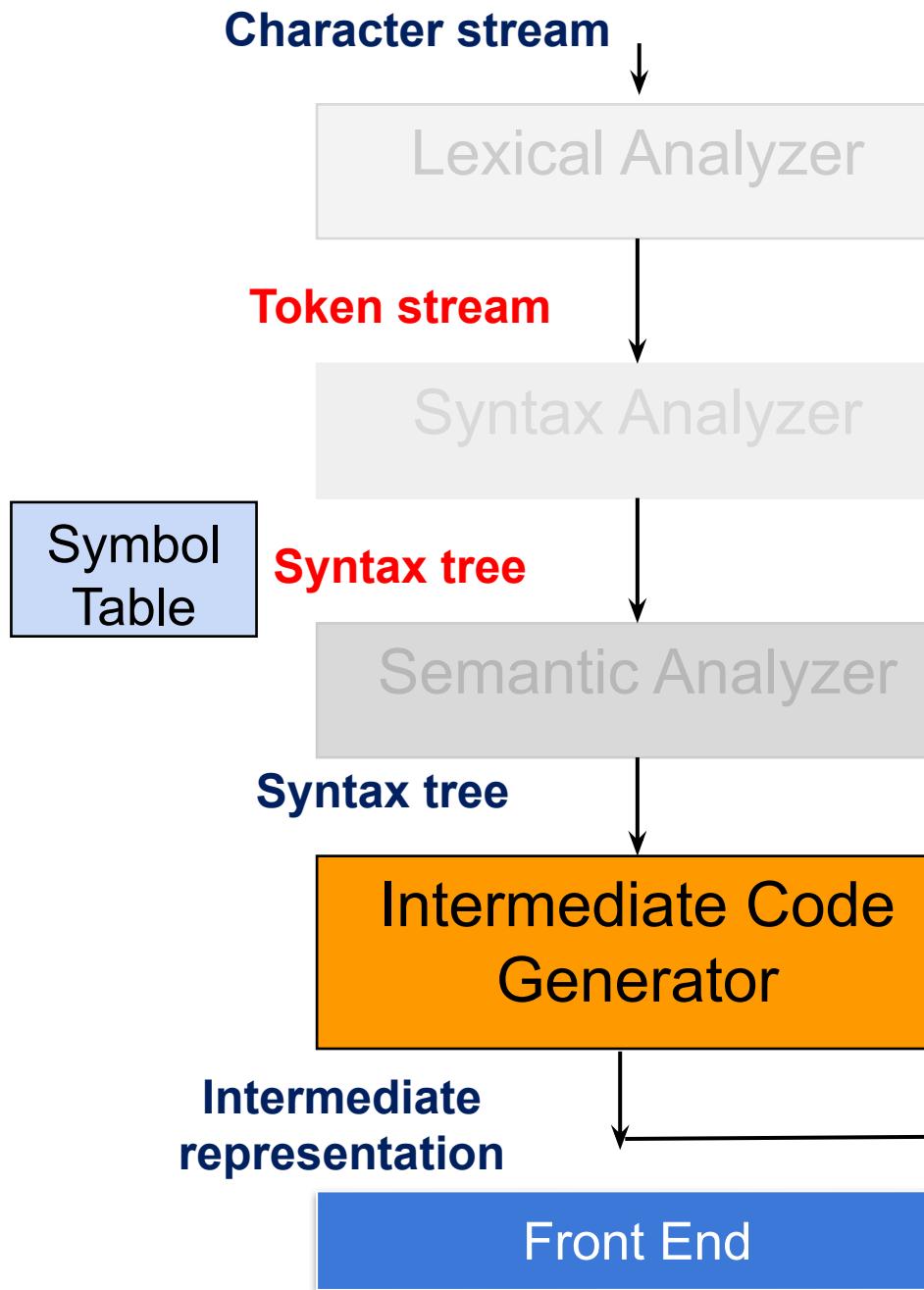
Department of CSE

Indian Institute of Technology Bhilai

vishwesh@iitbhilai.ac.in



Next...



Example ...

Code for

```
while a < b do  
    if c < d then x=y+z  
    else      x=y-z
```

L1: if a < b goto L2

goto Lnext

L2: if c < d goto L3

goto L4

L3: $t_1 = Y + Z$

$X = t_1$

goto L1

L4: $t_1 = Y - Z$

$X = t_1$

goto L1

Lnext:

BackPatching

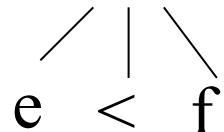
- Way to implement boolean expressions and flow of control statements in one pass
- We may not know the target labels
- leave them unspecified
- Back Patching is putting the address instead of labels when the proper label is determined

Generate code for $e < f$

Initialize nextquad to 100

E.t = {100}

E.f = {101}



```
100: if e < f goto -  
101 goto -
```

BackPatching

- **makelist(i)**: create a newlist containing only i, return a pointer to the list.
- **merge(p1,p2)**: merge lists pointed to by p1 and p2 and return a pointer to the concatenated list
- **backpatch(p,i)**: insert i as the target label for the statements in the list pointed to by p

$E \rightarrow id_1 \text{ relop } id_2$

E.trueclist = makelist(nextquad)

E.falselist = makelist(nextquad+ 1)

emit(if id₁ relop id₂ goto ---)

emit(goto ---)

Boolean Expressions

$$E \rightarrow E_1 \text{ or } E_2$$

- Need to know the starting address of code corresponding to E2.

Boolean Expressions

$E \rightarrow E_1 \text{ or } M E_2$

$M \rightarrow \epsilon$

- Insert a marker non terminal M into the grammar
to pick up index of next quadruple

$E \rightarrow E_1 \text{ or } M\ E_2$

backpatch($E_1.\text{falselist}$, $M.\text{quad}$)

$E.\text{truelist} = \text{merge}(E_1.\text{truelist}, E_2.\text{truelist})$

$E.\text{falselist} = E_2.\text{falselist}$

$M \rightarrow E$

$M.\text{quad} = \text{nextquad}$

$E \rightarrow E_1 \text{ and } M E_2$

backpatch($E_1.\text{truelist}$, $M.\text{quad}$)

$E.\text{truelist} = E_2.\text{truelist}$

$E.\text{falselist} = \text{merge}(E_1.\text{falselist}, E_2.\text{falselist})$

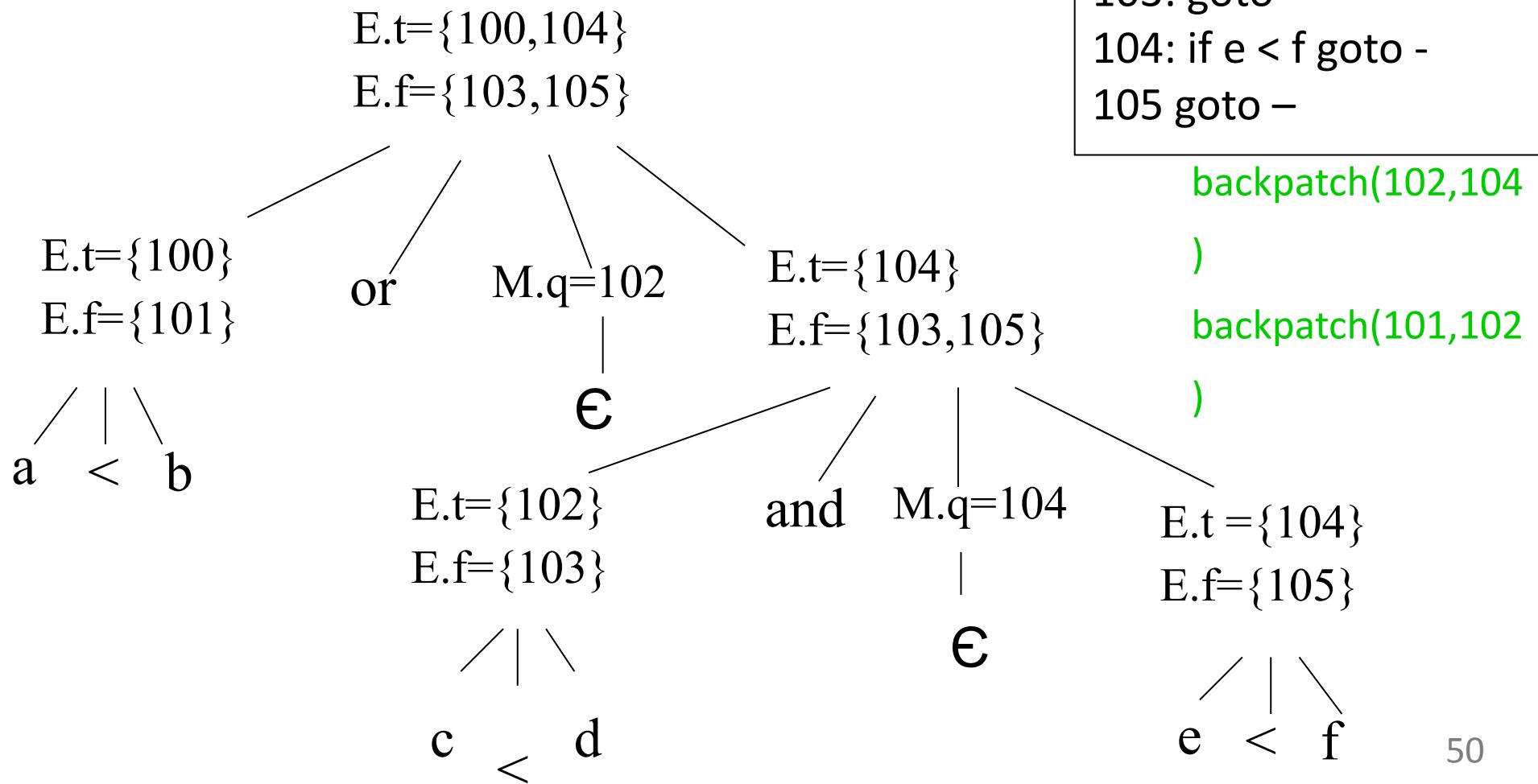
$E \rightarrow \text{not } E_1$

$E.\text{truelist} = E_1.\text{falselist}$

$E.\text{falselist} = E_1.\text{truelist}$

Generate code for a < b or c < d and e < f

Initialize nextquad to 100



```

100: if a < b goto -
101: goto - 102
102: if c < d goto - 104
103: goto -
104: if e < f goto -
105 goto -
  
```

backpatch(102,104)

)

backpatch(101,102)

)

Flow of Control

Statements

$S \rightarrow \text{if } E \text{ then } S_1$
| if E then S_1 else S_2
| while E do S_1
| begin L end
| A

$L \rightarrow L ; S$
| S

S : Statement

A : Assignment

L : Statement list

Scheme to implement translation

$S \rightarrow \text{if } E \text{ then } M S_1$
backpatch(E.truelist, M.quad)
 $S.\text{nextlist} = \text{merge}(E.\text{falselist}, S_1.\text{nextlist})$

$S \rightarrow \text{if } E \text{ then } M_1 S_1 \text{ N else } M_2 S_2$
backpatch(E.truelist, $M_1.\text{quad}$)
backpatch(E.falselist, $M_2.\text{quad}$)
 $S.\text{next} = \text{merge}(S_1.\text{nextlist},$
 $\quad \quad \quad N.\text{nextlist}, S_2.\text{nextlist})$

Scheme to implement translation

$S \rightarrow \text{while } M_1 E \text{ do } M_2 S_1$

backpatch($S_1.\text{nextlist}$, $M_1.\text{quad}$)

backpatch($E.\text{truelist}$, $M_2.\text{quad}$)

$S.\text{nextlist} = E.\text{falseclist}$

emit(goto $M_1.\text{quad}$)

Exercise

Write semantic rules for the FOR statement using backpatch

$$S \rightarrow \text{for}(E_1; E_2; E_3) S_1$$

Reading Exercise

- Intermediate Code for switch statements
 - Section 6.8