

CSL302: Compiler Design

Intermediate Code Generation

Vishwesh Jatala

Assistant Professor

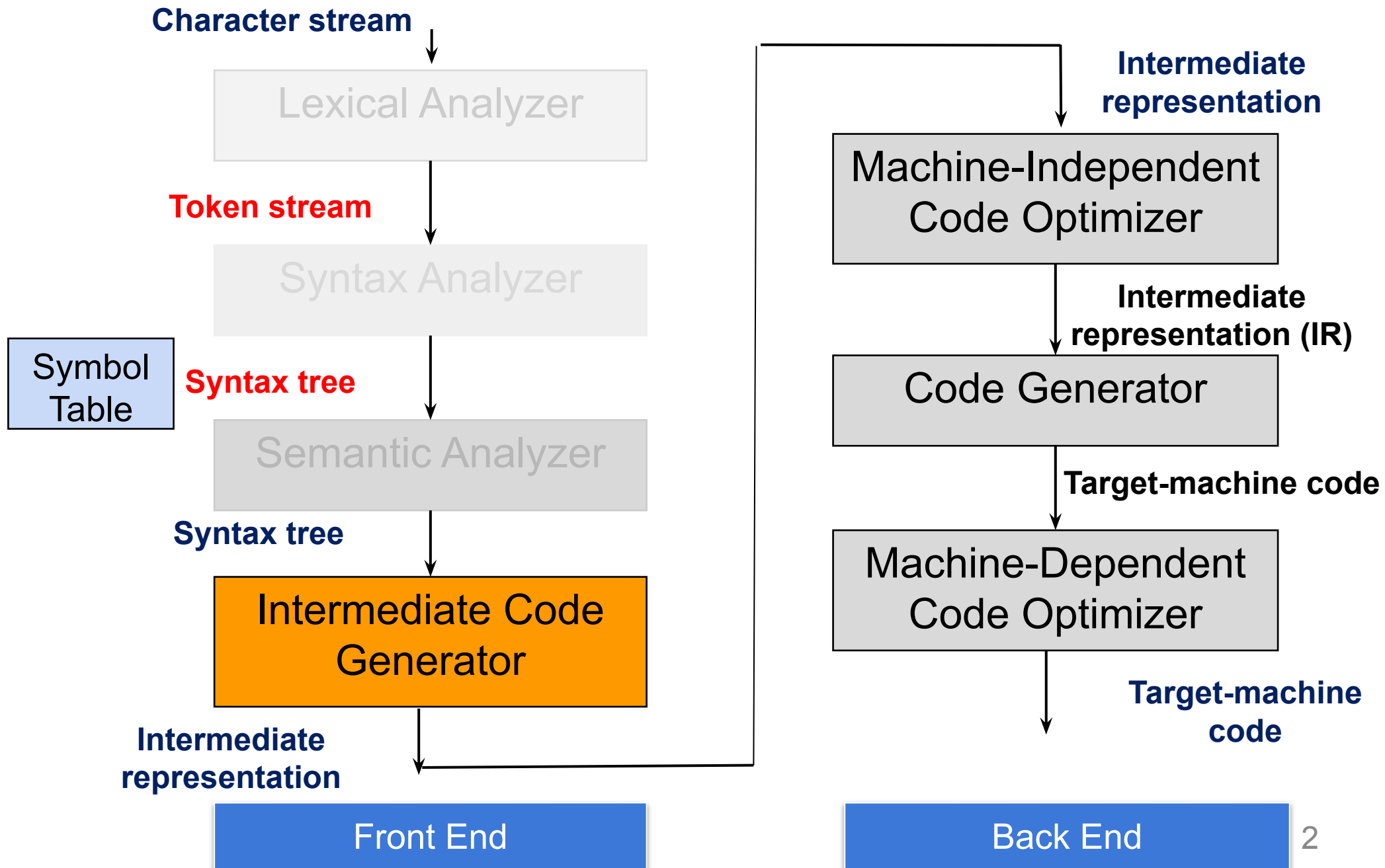
Department of CSE

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

$E.t = \{100\}$

$E.f = \{101\}$

$e < f$

```
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

$E \rightarrow id_1 \text{ relop } id_2$
 $E.\text{truelist} = \text{makelist}(\text{nextquad})$
 $E.\text{falselist} = \text{makelist}(\text{nextquad} + 1)$
 $\text{emit}(\text{if } id_1 \text{ relop } id_2 \text{ goto } \text{---})$
 $\text{emit}(\text{goto } \text{---})$

Boolean Expressions

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

- Need to know the starting address of code corresponding to E_2 .

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 .falselist, M .quad)
 E .truelist = merge(E_1 .truelist, E_2 .truelist)
 E .falselist = E_2 .falselist

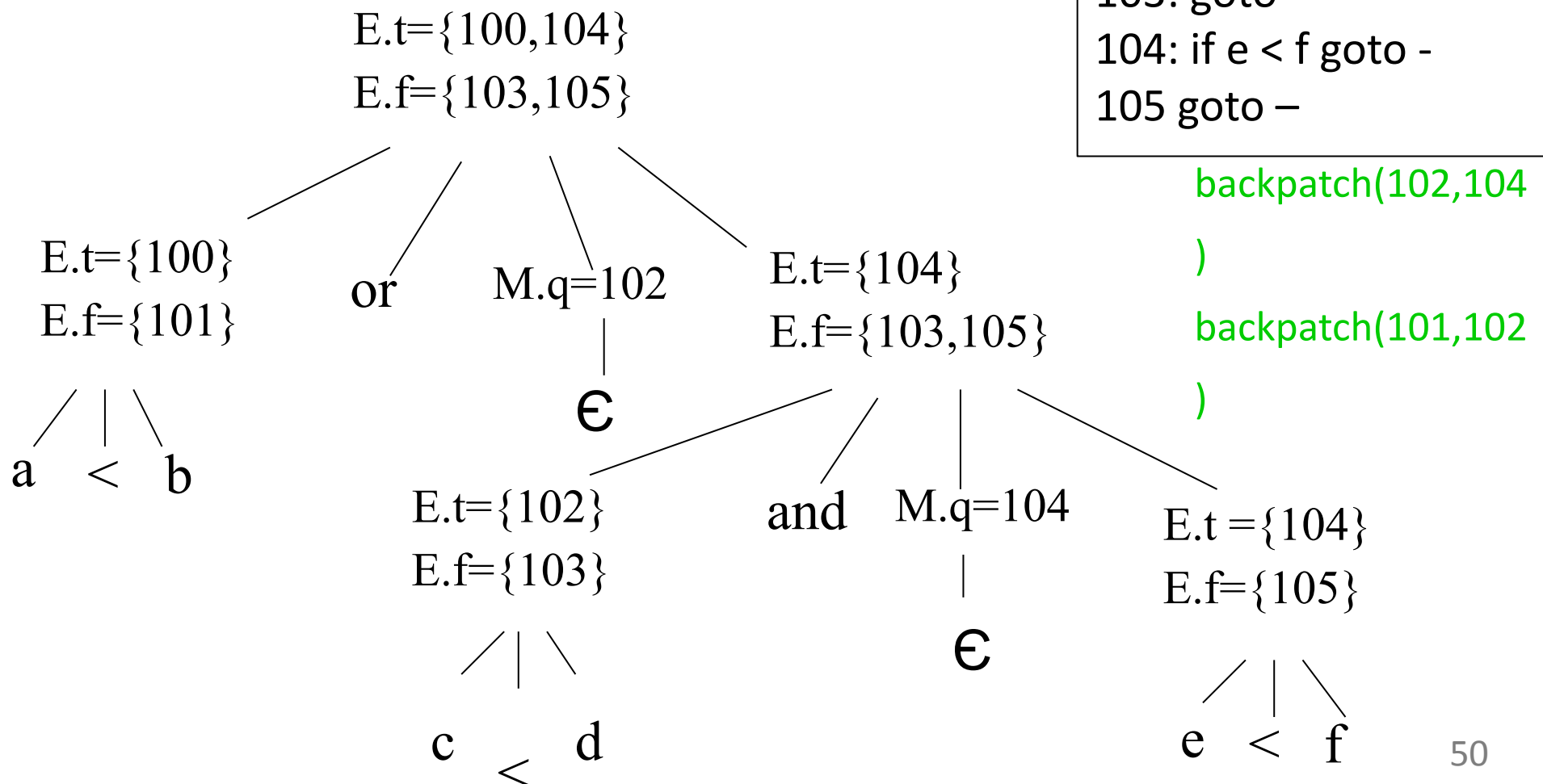
$M \rightarrow \epsilon$
 M .quad = nextquad

$E \rightarrow E_1 \text{ and } M E_2$
 backpatch(E_1 .truelist, M .quad)
 E .truelist = E_2 .truelist
 E .falselist = merge(E_1 .falselist, E_2 .falselist)

$E \rightarrow \text{not } E_1$
 E .truelist = E_1 .falselist
 E .falselist = E_1 .truelist

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

Initialize nextquad to 100



Flow of Control

Statements

$S \rightarrow$ if E 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

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

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

Scheme to implement translation

```
S → while M1 E do M2 S1  
    backpatch(S1.nextlist, M1.quad)  
    backpatch(E.truelist, M2.quad)  
    S.nextlist = E.falselist  
    emit(goto M1.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