

CSL302: Compiler Design

Intermediate Code Generation

Vishwesh Jatala

Assistant Professor

Department of CSE

Indian Institute of Technology Bhilai

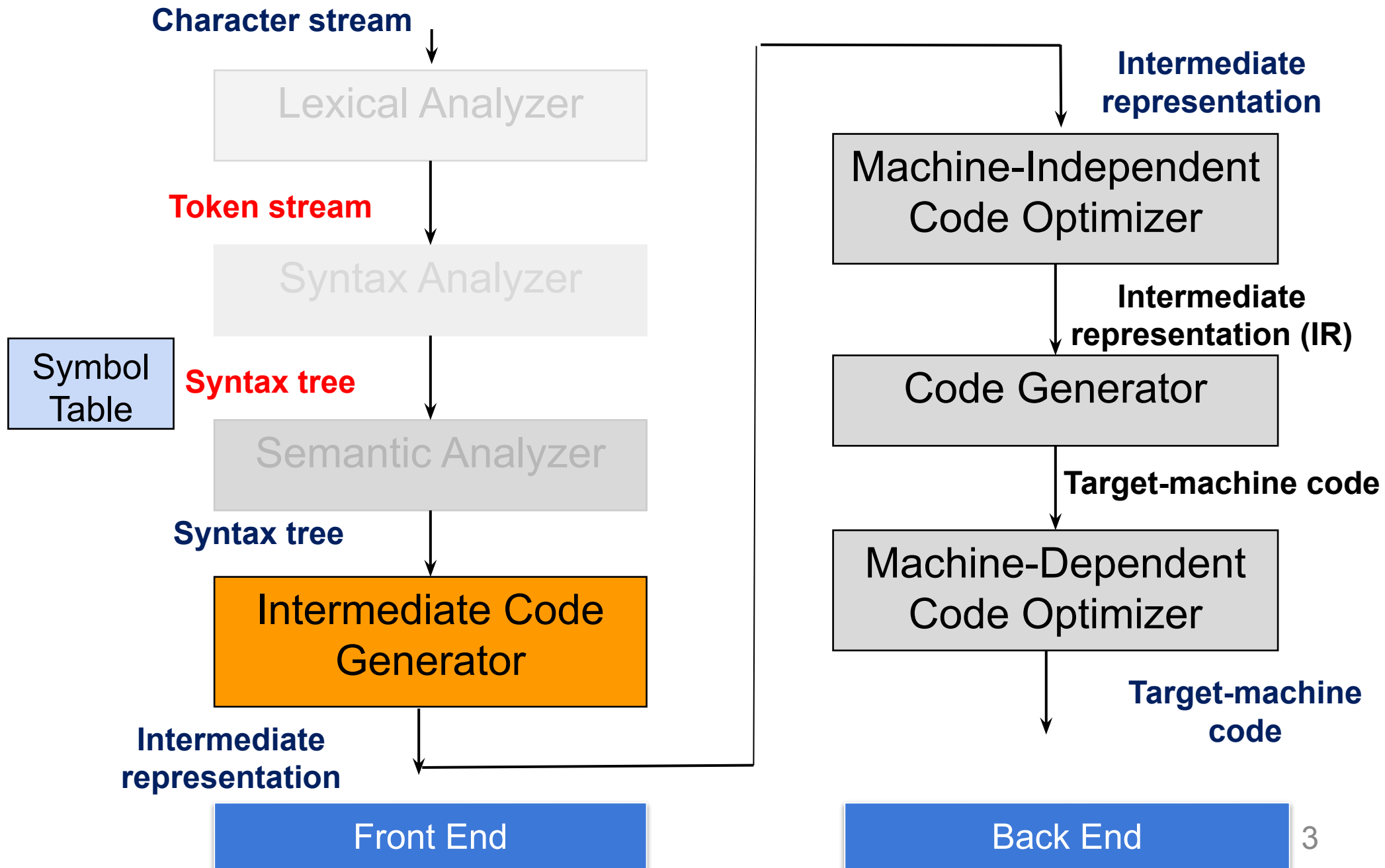
vishwesh@iitbhilai.ac.in



Acknowledgement

- References for today's slides
 - *Lecture notes of Prof. Amey Karkare (IIT Kanpur) and Late Prof. Sanjeev K Aggarwal (IIT Kanpur)*
 - *IIT Madras (Prof. Rupesh Nasre)*
 - *<http://www.cse.iitm.ac.in/~rupesh/teaching/compiler/aug15/schedule/4-sdt.pdf>*
 - *Course textbook*
 - *Stanford University:*
 - *<https://web.stanford.edu/class/archive/cs/cs143/cs143.1128/>*

Next...



Boolean Expression

E: $x < 100$

100: if $x < 100$ goto _
102: goto _

100: if $x < 100$ goto E.true
102: goto E.false

Syntax directed translation of boolean expressions

if E is of the form: $a < b$
then code is of the form:

if $a < b$ goto E.true
goto E.false

Syntax directed translation of boolean expressions

$E \rightarrow E_1 \text{ relop } E_2$
 $E.\text{code} = \text{gen}(\text{ if } E_1 \text{ relop } E_2 \text{ goto } E.\text{true}) \mid \mid$
 $\text{gen}(\text{goto } E.\text{false})$

Each Boolean expression E has two attributes, **true** and **false**. These attributes hold the label of the **target stmt** to jump to.

Control flow translation of boolean expression

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

$E_1.\text{true} := \text{newlabel}$

$E_1.\text{false} := E.\text{false}$

$E_2.\text{true} := E.\text{true}$

$E_2.\text{false} := E.\text{false}$

$E.\text{code} := E_1.\text{code} \parallel \text{gen}(E_1.\text{true}) \parallel E_2.\text{code}$

Control flow translation of boolean expression

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

$E_1.\text{true} := E.\text{true}$

$E_1.\text{false} := \text{newlabel}$

$E_2.\text{true} := E.\text{true}$

$E_2.\text{false} := E.\text{false}$

$E.\text{code} := E_1.\text{code} \parallel \text{gen}(E_1.\text{false}) \parallel E_2.\text{code}$

Example

Code for $a < b$ or $(c < d$ and $e < f)$

```
    if a < b goto Ltrue  
    goto L1
```

```
L1:  if c < d goto L2  
      goto Lfalse
```

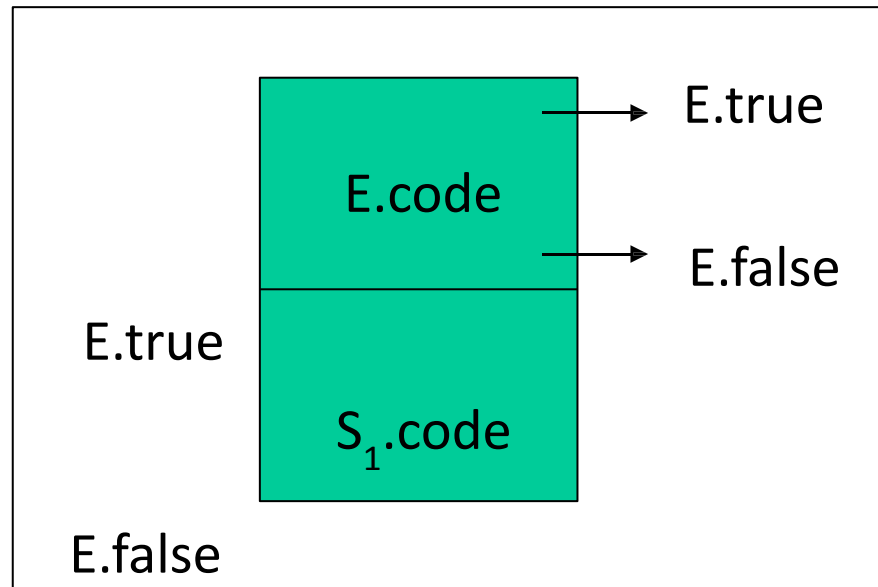
```
L2:  if e < f goto Ltrue  
      goto Lfalse
```

Ltrue:

Lfalse:

Control Flow Statements

- Flow of control
statements $S \rightarrow$ if E then
 S_1
| if E then S_1 else S_2
| while E do S



$S \rightarrow \text{if } E \text{ then } S_1$

$E.\text{true} = \text{newlabel}$

$E.\text{false} = S.\text{next}$

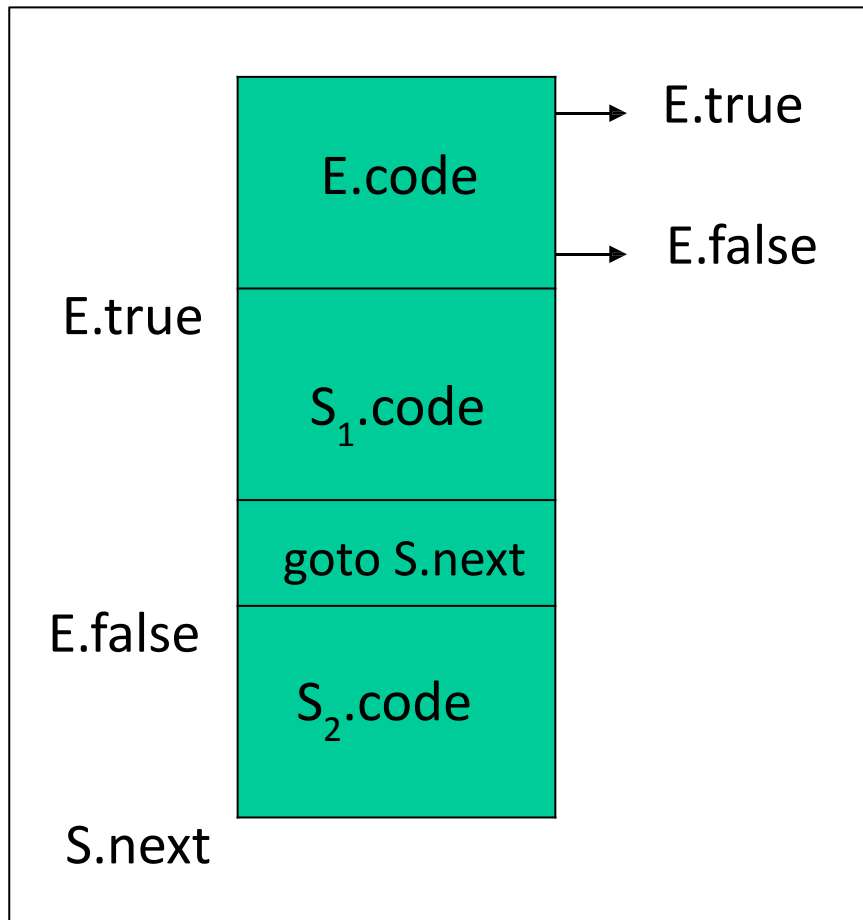
$S_1.\text{next} = S.\text{next}$

$S.\text{code} = E.\text{code} \parallel$

$\text{gen}(E.\text{true} ':') \parallel$

$S_1.\text{code}$

.next means waht ??



$S \rightarrow \text{if } E \text{ then } S_1 \text{ else } S_2$
E.true = newlabel
E.false = newlabel
 $S_1.\text{next} = S.\text{next}$
 $S_2.\text{next} = S.\text{next}$
 $S.\text{code} = E.\text{code} \parallel$

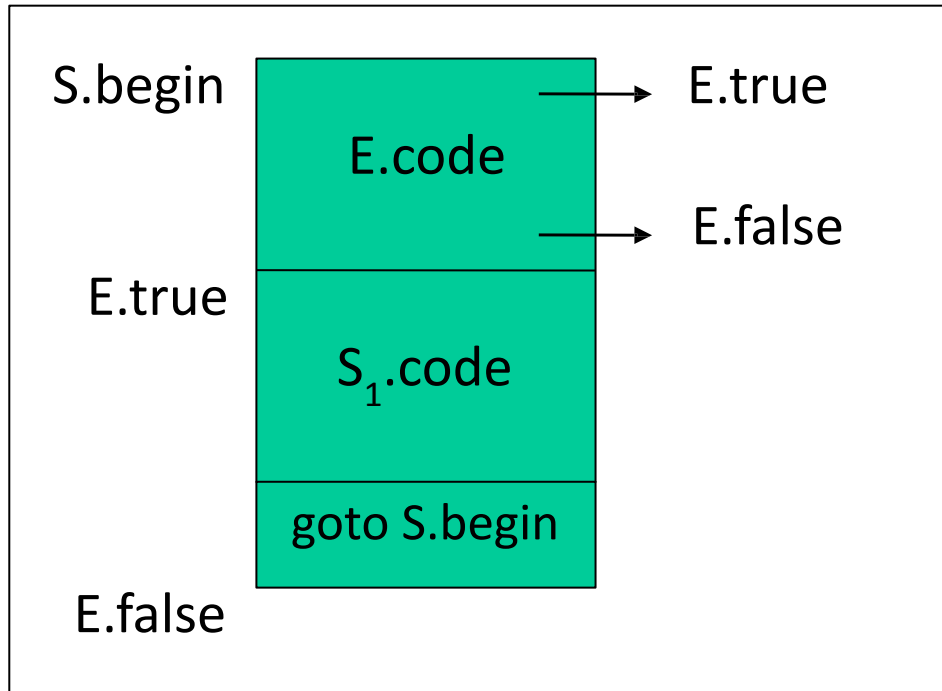
gen(E.true ':') ||
S₁.code ||
gen(goto S.next) ||
gen(E.false ':') ||
S₂.code

Example ...

Code for

```
if c < d then x = y + z  
else    x = y - z
```

```
L1:    if c < d goto L2  
        goto L3  
L2:    t1 = Y + Z  
        X = t1  
        goto Lnext  
L3:    t2 = Y - Z  
        X = t2  
Lnext:
```



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

```
S.begin = newlabel
E.true = newlabel
E.false = S.next
S1.next = S.begin
S.code = gen(S.begin ':') ||
        E.code ||
        gen(E.true ':') ||
        S1.code ||
        gen(goto S.begin)
```