CSL302: Compiler Design

Intermediate Code Generation

Vishwesh Jatala

Assistant Professor

Department of CSE

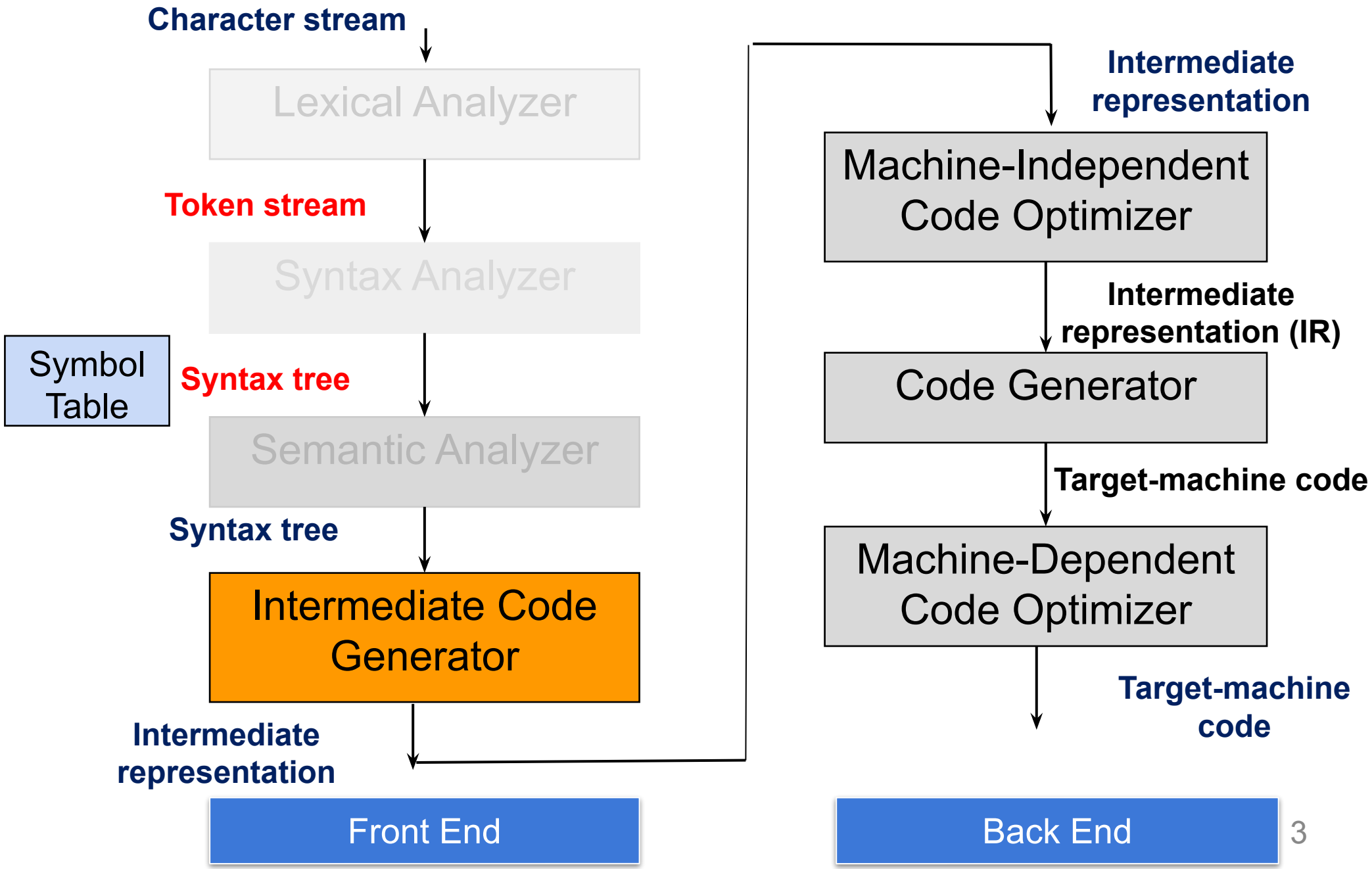
Indian Institute of Technology Bhilai

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…


Character stream
Lexical Analyzer
Token stream
Syntax Analyzer
Symbol Table
Syntax tree
Semantic Analyzer
Syntax tree
Intermediate Code Generator
Intermediate representation
Front End
Intermediate representation
Machine-Independent Code Optimizer
Intermediate representation (IR)
Code Generator
Target-machine code
Machine-Dependent Code Optimizer
Target-machine code
Back End

Recap

- Intermediate code generation
  - Expressions
    - Arithmetic
    - Boolean

Boolean Expressions

E →

| E relop E
| E or E
| E and E
| not E
| true
| false

Numerical representation

- relational expression a < b is equivalent to if a < b then 1 else 0

1.if a < b goto 4.
2.t = 0
3. goto 5
4. t = 1
5.

Syntax directed translation of boolean expressions

E → E1 < E2

E.place := newtmp
emit(if E1.place < E2.place goto nextstat+3)
emit(E.place = 0)
emit(goto nextstat+2)
emit(E.place = 1)

"nextstat" is a global variable; a pointer to the statement to be emitted. emit also updates the nextstat as a side-effect.

Syntax directed translation of boolean expressions

$E \rightarrow E_1$ or E_2

E.place := newtmp

gen(E.place ':=' E_1.place 'or' E_2.place)

$E \rightarrow E_1$ and E_2

E.place:= newtmp

gen(E.place ':=' E_1.place 'and' E_2.place)

$E \rightarrow$ not E_1

E.place := newtmp

gen(E.place ':=' 'not' E_1.place)

Syntax directed translation of boolean expressions

E → true

E.place := newtmp

emit(E.place = '1')

E → false

E.place := newtmp

emit(E.place = '0')

Exercise

Generate TAC for

a < b or (c < d and e < f)

Operator	Meaning	Associativity
<	Relational less than	left-to-right
and	Logical AND	left-to-right
or	Logical OR	left-to-right

Precedence and Associativity Symbol. Top row as highest precedence.

Example:
Code for a < b or c < d and e < f

100: if a < b goto 103

101: $t_1 = 0$

102: goto 104

103: $t_1 = 1$

104:

if c < d goto 107

105: $t_2 = 0$

106: goto 108

107: $t_2 = 1$

108:

if e < f goto 111

109: $t_3 = 0$

110: goto 112

111: $t_3 = 1$

112:

$t_4 = t_2$ and t_3

113: $t_5 = t_1$ or t_4

Methods of translation

- Evaluate similar to arithmetic expressions
  - Normally use 1 for true and 0 for false

- Implement by flow of control using short circuiting
  - given expression E_1 or E_2
    if E_1 evaluates to true
    then E_1 or E_2 evaluates to true
    without evaluating E_2

Short Circuit Evaluation of boolean expressions

- Translate boolean expressions without:
  - generating code for storing the boolean result explicitly
  - evaluating the entire expression

- Flow of control

  statements $S \rightarrow$ if E then S_1

  | if E then S_1 else S_2

  | while E do S_1

Without Short Circuiting

E1 E2

if (x < 100 || x > 200) x =0 ;

100: if x < 100 goto 104

101: t_I = 0

102: goto 104

103: t_I = 1

104: if x> 200 goto 107

105: t2 = 0

106: goto 109

107: t2=1

108: t3=t1 || t2

109: if t3 goto 110

110: goto 112

111: x=0

112: ..

Short Circuiting

E1 E2

if (x < 100 || x > 200) x =0 ;

100: if x < 100 goto 104
101: t_1 = 0
102: goto 104
103: t_1 = 1
104: if x> 200 goto 107
105: t2 = 0
106: goto 109
107: t2=1
108: t3=t1 || t2
109: if t3 goto 110
110: goto 112
111: x=0
112: ..

100: if x< 100 goto 104
101: goto 102
102: if x> 200 goto 104
103: goto 105
104: x=0
105:

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

$E \longrightarrow E_1$ relop E_2

E.code = gen(if E_1 relop E_2 goto E.true) || gen(goto E.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

$E \rightarrow E_1$ and E_2

E_1.true := newlabel
E_1 false := E.false
E_2.true := E.true
E_2 false := E.false
E.code := E_1.code || gen(E_1.true) || E_2.code

Control flow translation of boolean expression

$E \rightarrow E_1$ or E_2

E_1.true := E.true
E_1.false := newlabel
E_2.true := E.true
E_2.false := E.false
E.code := E_1.code || gen(E_1.false) || E_2.code