Compiler Design

Preet Kanwal

Department of Computer Science & Engineering

Teaching Assistant : Sree Pranavi G

Compiler Design

Unit 3: L-Attributed SDD - Intermediate Code Generation

Preet Kanwal

Department of Computer Science & Engineering

Compiler Design

Lecture Overview

In this lecture, you will learn about -

- L–Attributed SDD to generate intermediate code for -
  - Expressions
  - Condition statement
  - If statement
  - If - else statement
  - While statement
  - Do - while statement
  - For statement
  - Boolean expressions

Compiler Design

Recap

- There are 2 kinds of attributes - Synthesized and Inherited.
- An SDD with only synthesized attributes is an S-attributed definition.
- An SDD is L-attributed if all its attributes are either -
  - Synthesized
  - Extended synthesized dependent on children as well as inherited attributes.
  - Inherited but dependent only on inherited attributes at parent and any siblings at left.
- Every S-attributed SDD is also L-attributed.

SDD to generate Intermediate Code - Arithmetic Expressions

Write the SDD to generate intermediate code.
The given example indicates the code and its corresponding intermediate code for an expression **a = b + - c** .

Input	Output
a = b + - c	*t1 = minus c* *t2 = b + t1* *a = t2*

SDD to generate Intermediate Code - Arithmetic Expressions

Assigning appropriate semantic rules to each production -

Production	Semantic Rule
S -> id = E;	*S.code = E.code || gen(id.lexval '=' E.addr)*
E -> E1 + E2	*E.addr = new Temp();* *E.code = E1.code || E2.code ||* *gen(E.addr '=' E1.addr '+' E2.addr)*
E -> -E1	*E.addr = new Temp();* *E.code = E1.code || gen(E.addr '=' 'minus' E1.addr)*
E -> (E1)	*E.addr = E1.addr* *E.code = E1.code*
E -> id	*E.addr = id.lexval* *E.code = ' '*

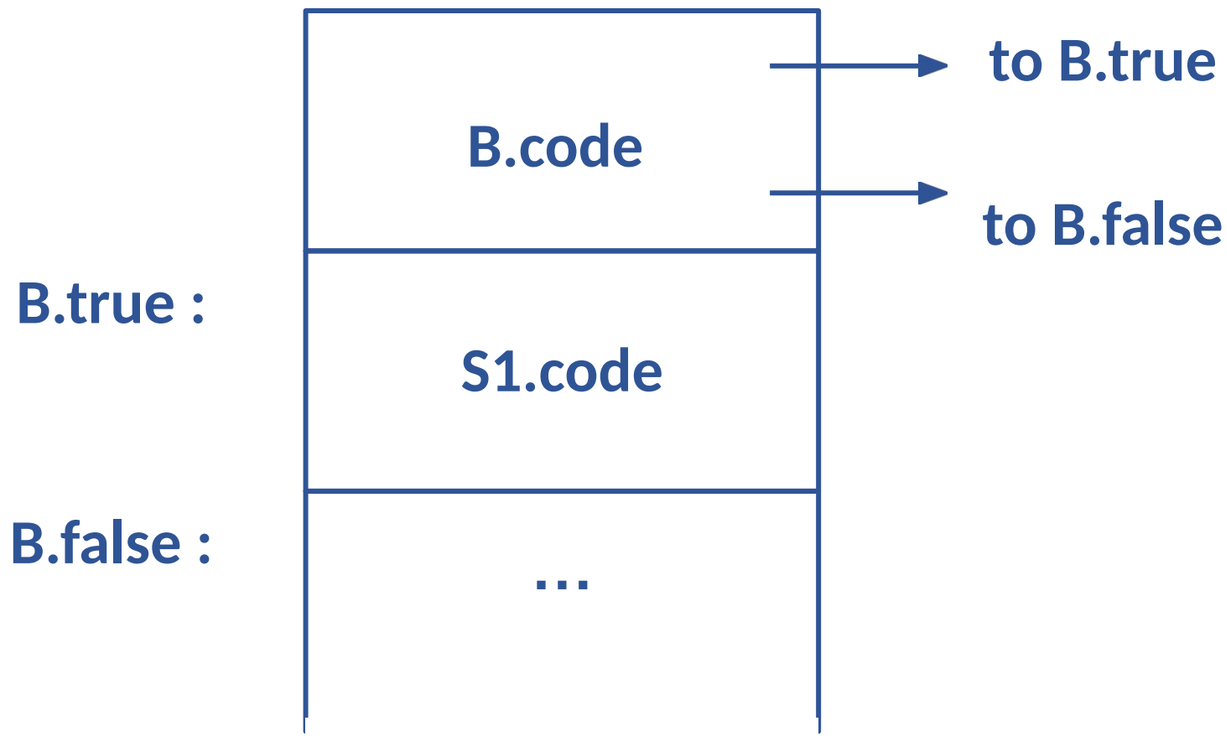
SDD to generate Intermediate Code - Conditions

Production	Semantic Rule
B -> id1 rel id2	*B.code = gen('if' id1.lexval rel.op id2.lexval 'goto' B.true) || gen('goto' B.false)*
rel -> >	*rel.op = ">"*
rel -> <	*rel.op = "<"*
rel -> >=	*rel.op = ">="*
rel -> <=	*rel.op = "<="*

SDD to generate Intermediate Code - If Statement

Production	Semantic Rule
S -> if (B) S1	*{ B.true = new Label();* *B.false = S.next;* *S1.next = S.next;* *S.code = B.code || label(B.true) || S1.code; }*

SDD to generate Intermediate Code - If statement



SDD to generate Intermediate Code

- **If Else**

  S -> if (C) S1 else S2

  {C.true = new label();

  C.false = new label();

  S1.next = S.next;

  S2.next = S.next;

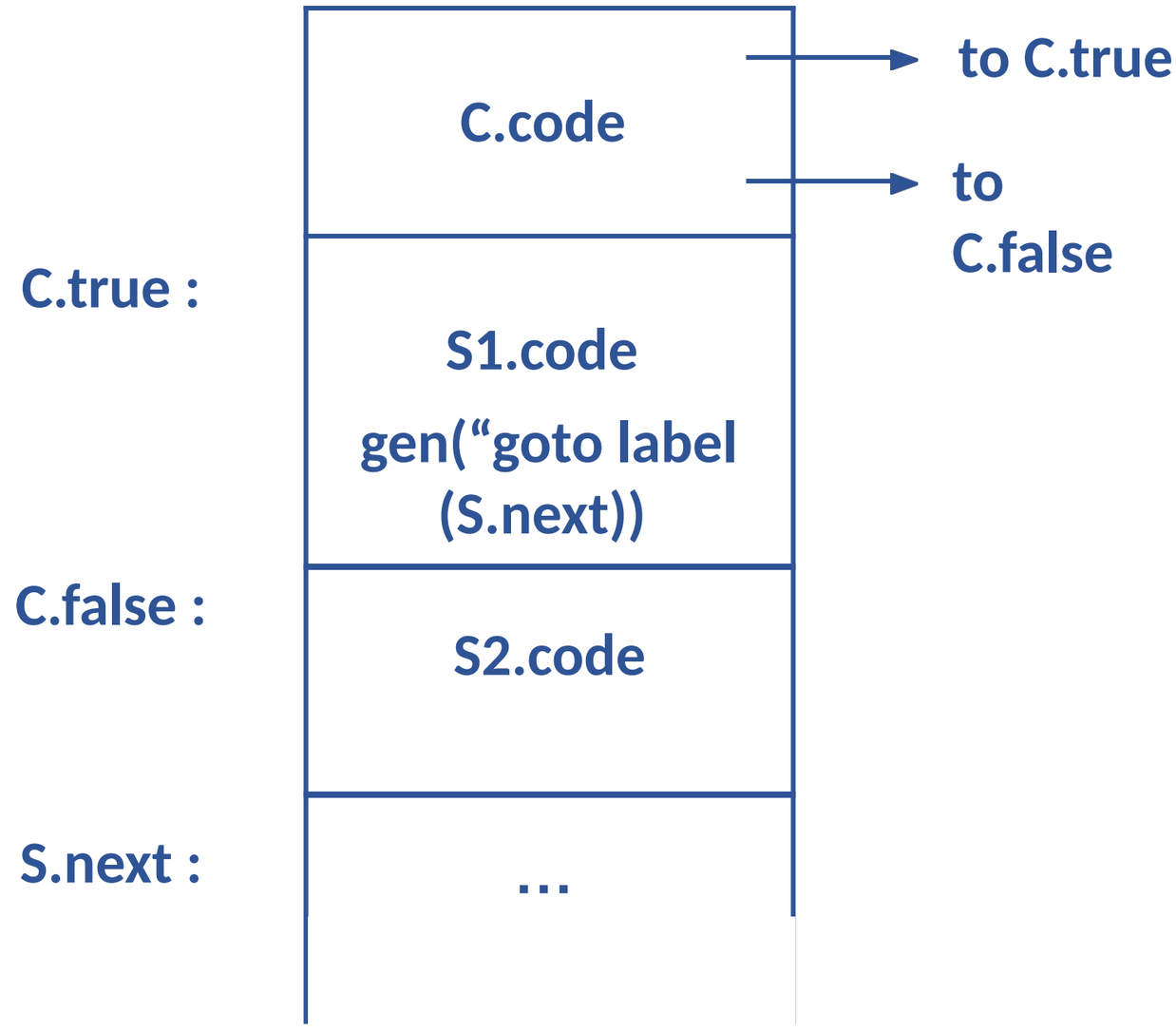
  S.code = C.code || label(C.true) || S1.code || gen("goto" label(S.next))

  || label(C.false) || S2.code;}

SDD to generate Intermediate Code

- If Else

S -> if (C) S1 else S2



SDD to generate Intermediate Code

- **While**

  S -> while (C) S1

  {begin = new label();

  C.true = new label();

  C.false = S.next;

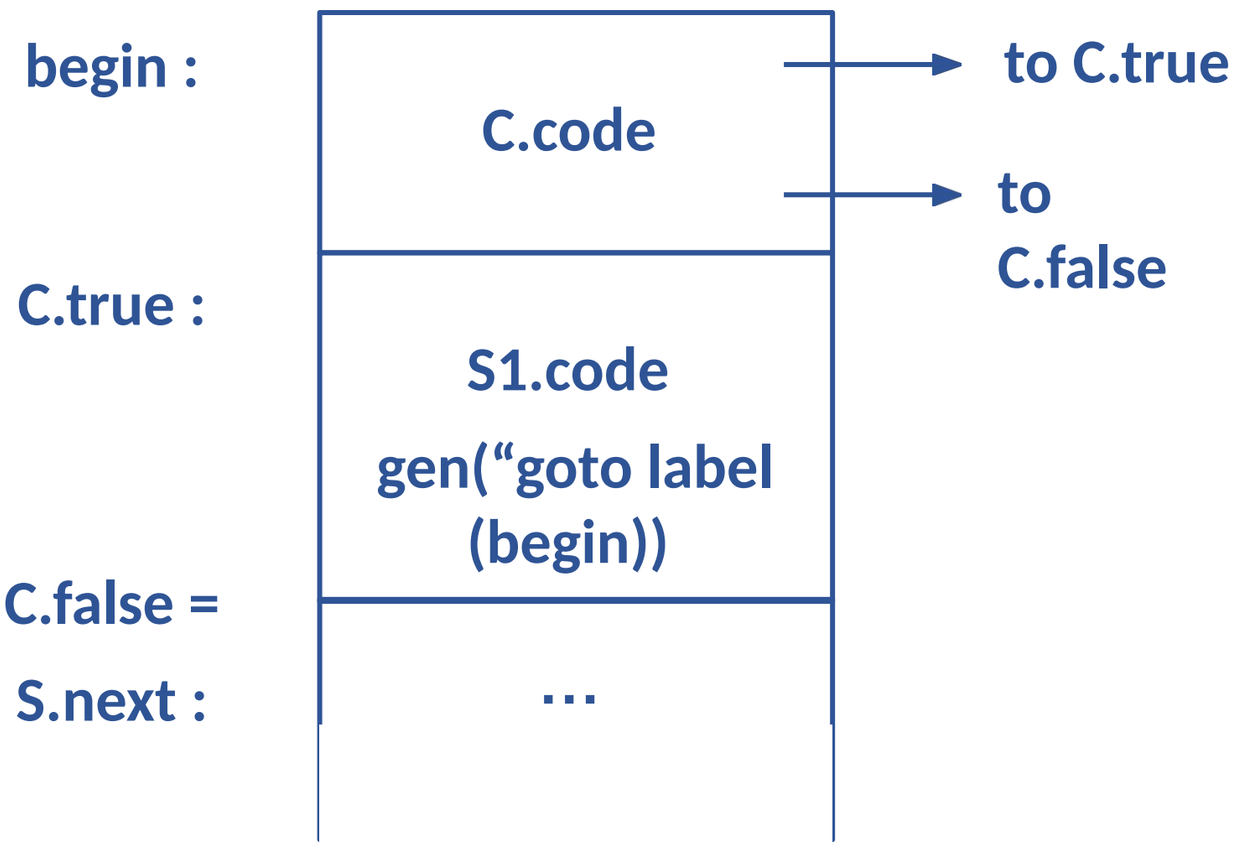
  S1.next = begin;

  S.code = label(begin) || C.code || label(C.true) || S1.code || gen("goto" label(begin));}

SDD to generate Intermediate Code

- While

S -> while (C) S1



SDD to generate Intermediate Code

- **Do - While**

  S -> do (S1) while (C)

  {C.true = new label();

  C.false = S.next;

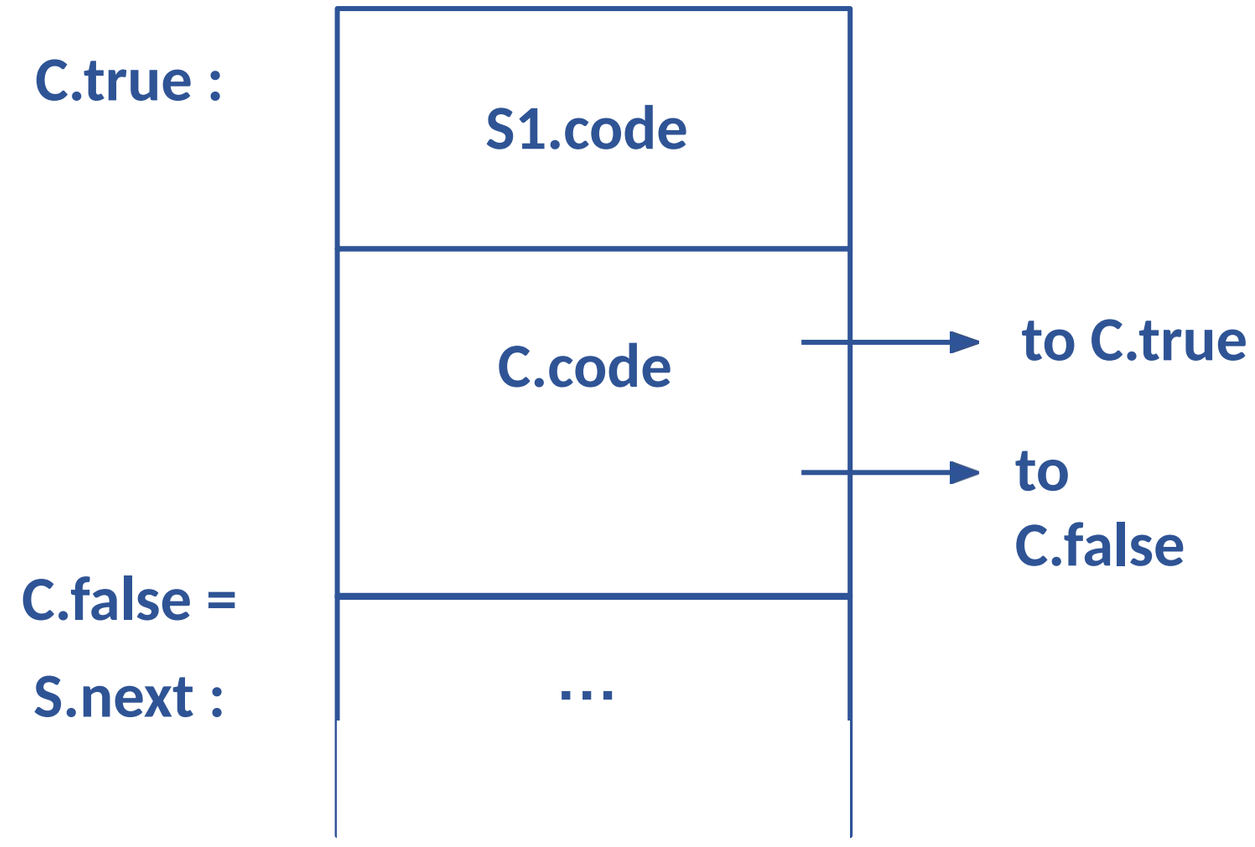
  S1.next = NULL;

  S.code = label(C.true) || S1.code ||C.code; }

SDD to generate Intermediate Code

- Do - While

S -> do (S1) while (C)



SDD to generate Intermediate Code

- **For**

  S -> for (S1 ; C; S2) S3

  {C.true = new label();

  C.false = S.next;

  S1.next = new label();

  S2.next = S1.next;

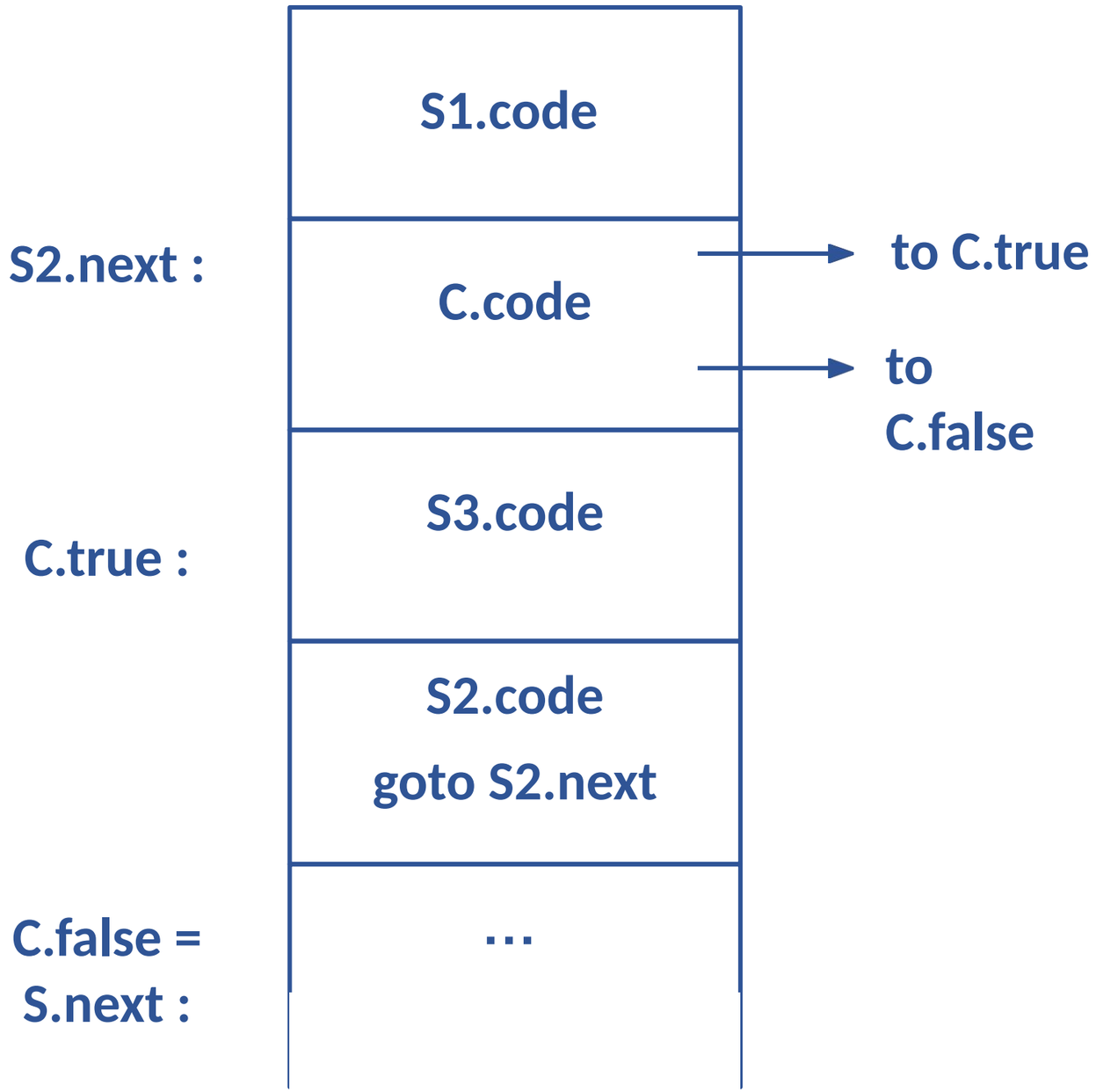
  S3.next = NULL ;

  S.code = S1.code || label(S1.next) || C.code || label(C.true) || S3.code || S2.code || gen("goto S1.next); }

SDD to generate Intermediate Code

- For

  S -> for (S1 ; C; S2) S3



SDD to generate Intermediate Code

- **Boolean Expressions**

B -> B1 || B2

{ B1.true = B.true ;

B1.false = new label();

B2.true = B.true;

B2.false = B.false;

B.code = B1.code || label (B1.false) || B2.code; }

SDD to generate Intermediate Code

- **Boolean Expressions**

B -> B1 && B2

{ B1.false = B.false ;

B1.true = new label();

B2.true = B.true ;

B2.false = B.false ;

B.code = B1.code || label (B1.true) || B2.code; }

SDD to generate Intermediate Code

- **Boolean Expressions**

B -> ! B1 { *B1.true = B.false; B1.false = B.true; B.code = B1.code;* }

B → true { B.code = gen(“goto” B.true);}

B → false { B.code = gen(“goto” B.false);}

SDD to generate Intermediate Code

- Generate Intermediate Code for the following example :

```
if ( x > 10)
{
    x = x + 1
}
```

- Grammar -

S -> if (B) { S1 }

B -> id1 > id2

S1 -> id = E;

E -> E1 + E2

E2 -> id

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu