UNIT 3

SYNTAX DIRECTED TRANSLATION FOR CONTROL STRUCTURES

M.B.Chandak
www.mbchandak.com
hodcs@rknec.edu

CONTROL STRUCTURES

1. If (condition) then Statement
2. If (condition) then Statement – 1 else statement – 2
3. While (Expression) Do Statement
4. Do (Statement) While (Condition)
5. For(i=0; i<=10; i++) do (Statements)
6. Procedure calls
7. Arrays [One, Two and Three Dimensions]

IF (CONDITION) THEN STATEMENT

- If condition tested is true, statement will be executed.
- If condition tested is false, next statement will be executed.

If (Condition) then

Statement

Next Statement

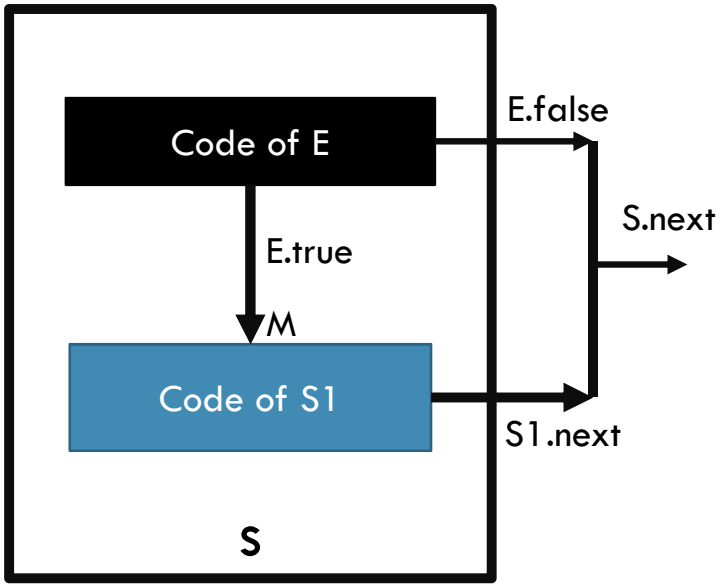
Grammar:

S → if (e) then S1

Modified Grammar:

S → if (e) then M1 S1 [M1 is dummy non terminal]

IF (CONDITION) THEN STATEMENT



Semantic Actions
{
Backpatch (E.True, M.Quad)
S.Next = merge(S1.next, E.false)
}
M → €
M.Quad = nextquad

EXAMPLE

Program Segment

IF (A>10) THEN
  X = 10

Three address code

Assume start address = 100

100 if(a>10) goto 102

101 goto 103

102 x = 10

103 exit

Semantics:

Backpatch(100,102)

S.next = merge(101,103)

IF(CONDITION) THEN S1 ELSE S2

Example of construct:

If (a > 10) then

x=10

Else

x=20

Expected Three Address Code

100: if (a>10) goto 103

101 x=20

102 goto 104

103 x=10

104 Exit

Representation:

S → if (E) then s1 else s2

Modified Grammar

S → if (E) then M1 s1 else M2 s2

M1 → €

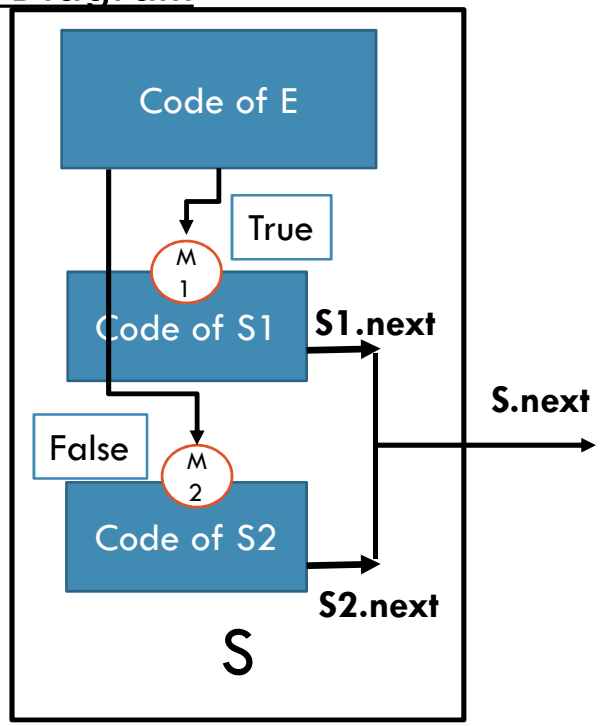
M2 → €

The non-terminal M1 will provide address of statement S1 and non-terminal M2 will address of statement S2.

***Nested If construct is not considered in present discussion.*

IF(CONDITION) THEN S1 ELSE S2

Flow Diagram



S → if (E) then M1 s1 else M2 s2

Semantic Actions:

{
  backpatch(E.true, M1.quad)
  backpatch(E.false,M2.quad)
  S.next = Merge(S1.next, S2.next)
}

M1.Quad = nextquad

M2.Quad = nextquad

Previous example description:

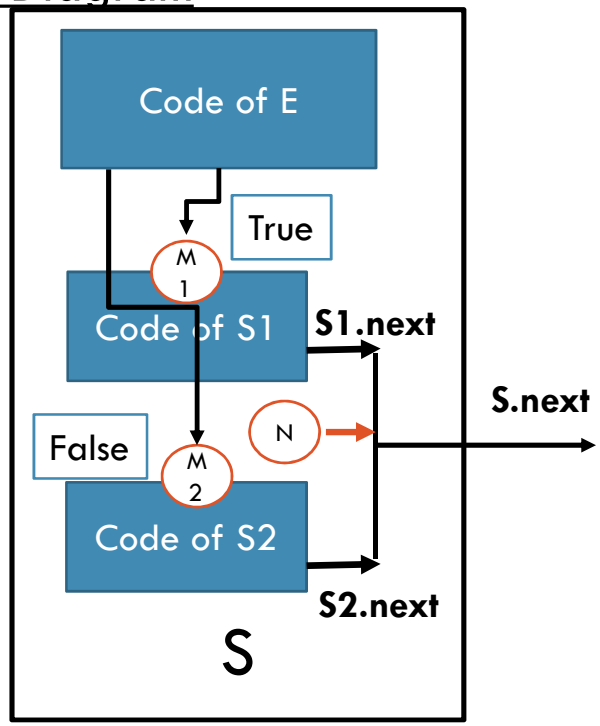
Backpatch(100,103)

Backpatch(100,101)

S.next = merge(102,104)

IF(CONDITION) THEN S1 ELSE S2 [NESTED IF]

Flow Diagram



S → if (E) then M1 s1 N else M2 s2

Semantic Actions:

{

backpatch(E.true, M1.quad)

backpatch(E.false,M2.quad)

S.next = Merge(S1.next, (N.quad, S2.next))

}

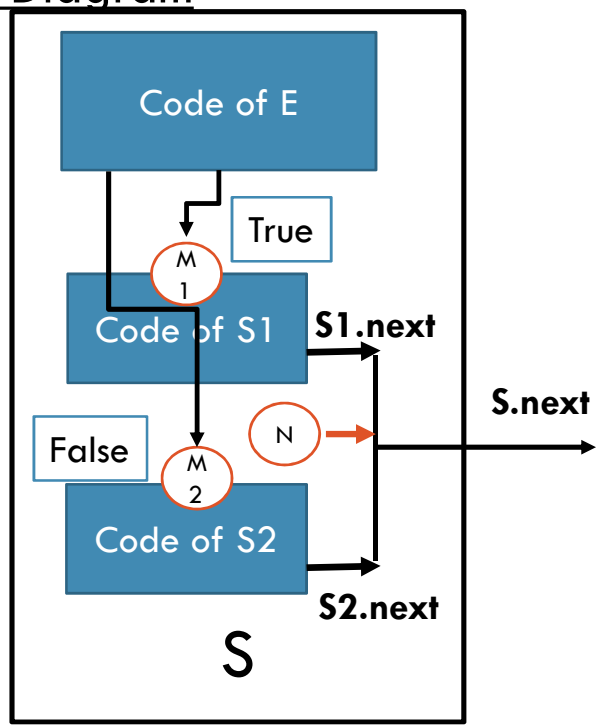
M1.Quad = nextquad

M2.Quad = nextquad

N.Quad = nextquad

IF(CONDITION) THEN S1 ELSE S2 [NESTED IF]

Flow Diagram



Example:

```
If (a < 10) then
   x = 5
Else
   if (a > 20) then
        x = 10
   else
     if(a < 18) then
        x = 7
     else
        x = 15
EXIT IF
```

Above example is ladder "else" example, in which "non-terminal" "N" will be used to determine "EXIT" points.

TRANSLATION

Example:

```
If (a < 10) then
   x = 5
Else
   if (a > 20) then
         x = 10
   else
      if(a < 18) then
         x = 7
      else
         x = 15
EXIT IF
```

Label	Code
100	If (a<10) goto 109
101	If(a>20) goto 107
102	If(a<18) goto 105
103	X=15
104	Goto 106
105	X=7
106	Exit
107	X=10
108	Goto 106
109	X=5
110	Goto 106

Translation
Backpatch(100,109) Backpatch(100,101)
Backpatch((101,107) Backpatch(101,102)
Backpatch(102,105)
Backpatch(102,103)
S.Next = merge(110,(104,106,108)

SDTS FOR WHILE..DO CONSTRUCT

- While .. Do is "Pre-test" loop execution construct.
- The loop is executed till the value of condition remains TRUE and when condition becomes FALSE, loop terminates.
- Example:

  While (a <= 10) do

  {

  Execute loop till condition is true

  } → Control goes back to condition check, if false execute next instruction

  Next Instruction

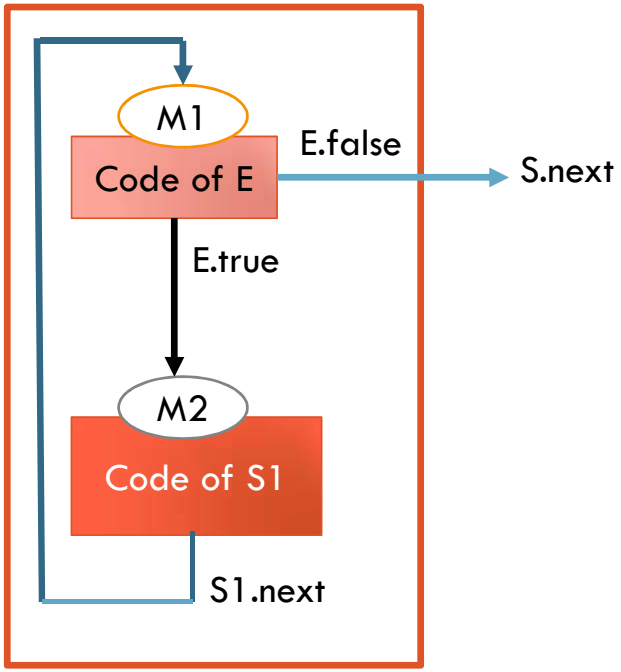
- Grammar:
- **S → While (E) do S1**

SDTS FOR WHILE..DO CONSTRUCT

- **Modified Grammar**
- **S → While M1 E1 DO M2 S1**
- **M1 → €**
- **M2 → €**
- **Semantic action:**

{
Backpatch(E.true,M2.quad)
Backpatch(S1.next, M1.quad)
S.next = E.false
Gencode(goto(M1.quad))
}

Flow Diagram



EXAMPLE

While (a > 10) do

x = 10

For above program segment:

100 if(a>10) goto 102

101 Goto 104

102 x = 10

103 Goto 100

104 Next statement

Semantics:

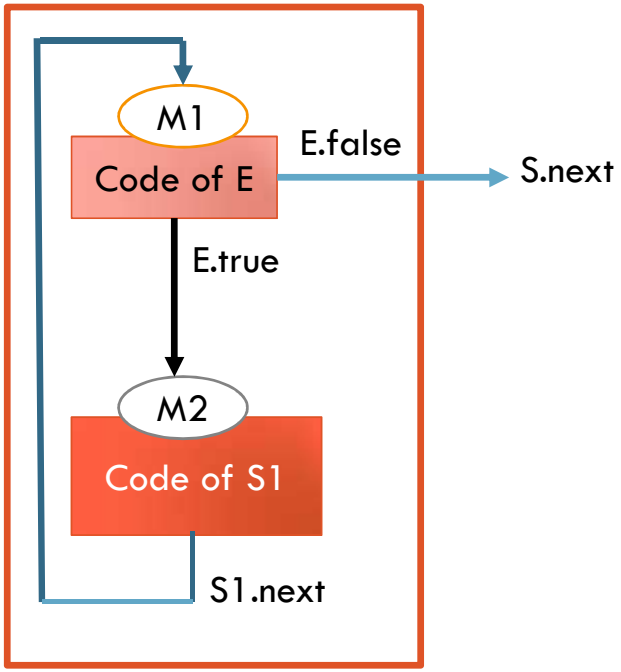
Backpatch(100,102)

Backpatch(103,100)

S.Next = 104

Goto(100)

Flow Diagram



DETAIL EXAMPLE

```
i = 1; j=1; x = 0; y=0;
While ( i <= 10) do
{
    x = x+1
    while (j < = 10) do
    {
        y = y + 1;
        j = j + 1;
     }
     i = i + 1;
}
```

LABEL	CODE	LABEL	CODE
100	I = 1	113	T4 = I+1
101	J = 1	114	I=T4
102	X = 0	**115**	**GOTO 104**
103	Y =0	116	NEXT STATEMENT
104	IF(I < = 10) GOTO 106		
105	**GOTO 116**		
106	T1 = X +1		
107	X = T1		
108	IF (J<=10) GOTO 110		
109	**GOTO 113**		
110	T2=Y+1		
111	Y=T2		
110	T3 = J+1		
111	J = T3		
112	**GOTO 108**		

SEMANTICS

Backpatch(E.true,m2.quad)
Backpatch(s1.next, m1.quad)
S.next = E.false
Gencode(goto M1.quad)

Backpatch(104,106)

Backpatch(108,110)
Backpatch(112,108)
S.next = 113
Gencode(goto 104)

S.next = 116

S.next = 116 [E.false]
S.next = 113 [E.false]

Gencode(goto 104) : Location M1

Since there is no construct to handle E.false value: gencode(goto M1) construct is used as explicit statement.

LABEL	CODE	LABEL	CODE
100	I = 1	113	T4 = I+1
101	J = 1	114	I=T4
102	X = 0	**115**	**GOTO 104**
103	Y =0	116	NEXT STATEMENT
104	IF(I < = 10) GOTO 106		
105	**GOTO 116**		
106	T1 = X +1		
107	X = T1		
108	IF (J<=10) GOTO 110		
109	**GOTO 113**		
110	T2=Y+1		
111	Y=T2		
110	T3 = J+1		
111	J = T3		
112	**GOTO 108**		

EXAMPLE: WHILE – IF CONSTRUCT

```
Program Segment
X=1
While (x <= 10) do
{
      if ( z <= 10)
         t = a + b
     else
         t = c + d
 x = x + 1
}
```

Label	Code	Semantics
100	X=1	
101	If(x<=10) goto 103	Backpatch(101,103)
102	Goto 112	S.Next = 112
103	If(z<=10) goto 107	Backpatch(103,107)
104	T1=c+d	::
105	T=t1	::
106	Goto 109	::
107	T2=a+b	::
108	T=t2	::
109	T3 = x+1	::
110	X = t3	::
111	Goto 101	::
112	Exit	::