United College of Engineering and Research, Naini

Computer Science and Engineering Department

Unit 3

Compiler Design

Synatx Directed Translation

Prepared By

Prabhat Shukla

SDT to generate 3AC for assignment statement

$S \rightarrow id=E$

$E \rightarrow E + E$

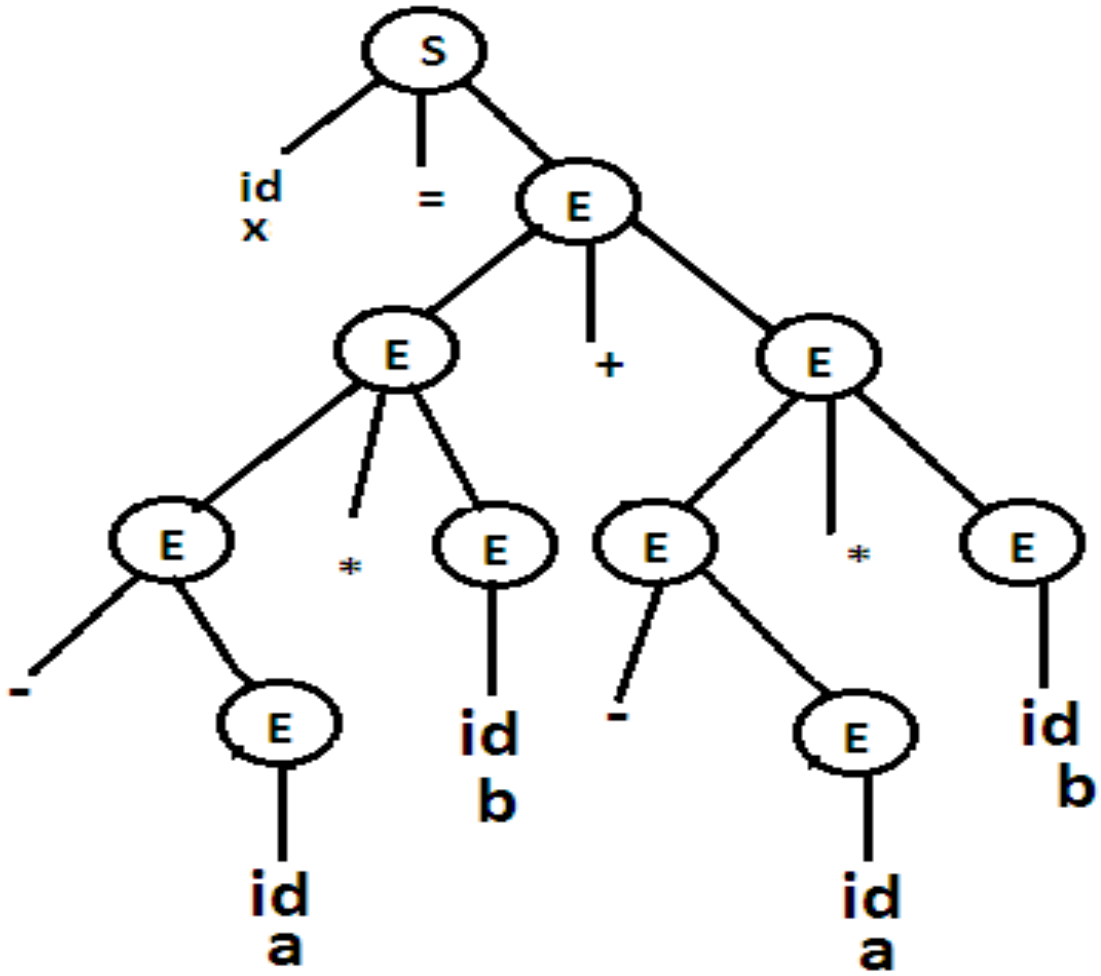
$E \rightarrow E * E$

$E \rightarrow -E$

$E \rightarrow id$

assume string is:

x= -a*b + -a*b


S
id
x
=
E
E
+
E
E
*
E
E
*
E
-
E
id
b
-
E
id
b
id
a
id
a

Answer

S → id=E {gen(id . lval ||'='|| E.place)}

E → E + E {E . Place = new temp();
gen(E . Place ||'='|| E . place ||'+' ||E . place}

E → E * E {E . Place = new temp();
gen(E . Place|| '='|| E . Place|| '*'|| E . place}

E → -E {E . Place = new temp();
gen(E . Place ||'='|| '-'|| E . place }

E → id {E . Place = id . lval }

SDT to generate 3AC for boolean expression

Que:-Define backpatching and sementic rules for the Boolean expression.Derive the three address code for the following expression.

p<q and r<s or t>u **(AKTU 2015-16,2013-14,2009-10)**

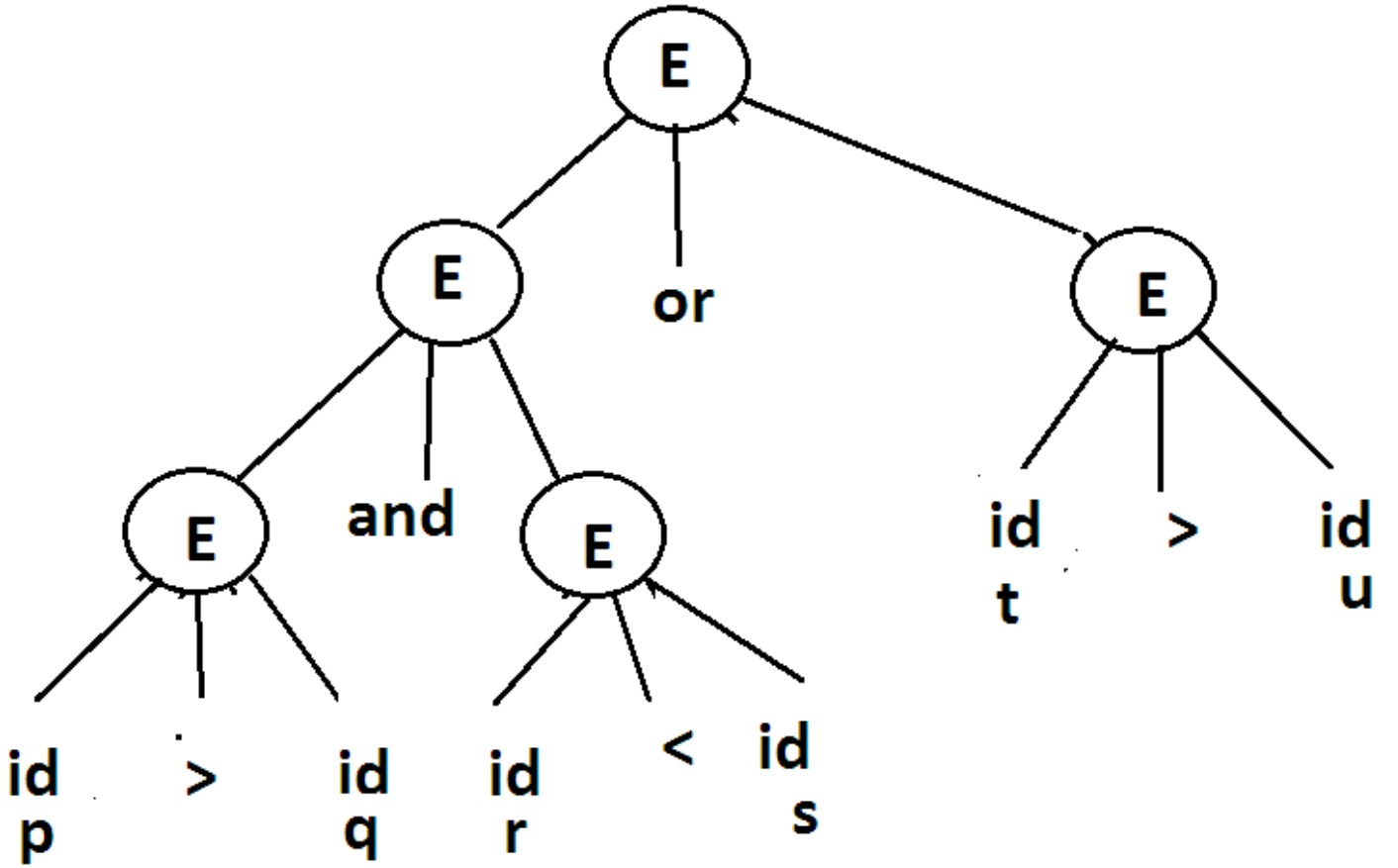
assume grammar is :-

E→E or E

E→E and E

E→ NOT E

E→ id relop id


E
E
or
E
E
and
E
id
t
>
id
u
id
p
>
id
q
id
r
<
id
s

E→E or E {E.place=new temp();
gen(E.place=E.place ‘or’ E.place}
E→E and E {E.place=new temp();
gen(E.place=E.place ‘and’ E.place}
E→ NOT E {E.place=new temp();
gen(E.place= ‘not’ E.place}
E→ id relop id { E.place=newtemp()
gen(‘if’ id1.place relop id2.place ‘goto’ next_state+3)
gen(E.place=’0’)
gen(‘goto’ next_state+2)
gen(E.place=’1’)
}

Backpatching

It is the activity of filling up unspecified information of labels using appropriate semantic actions in during the code generation process.

E→E or M E

E→E and M E

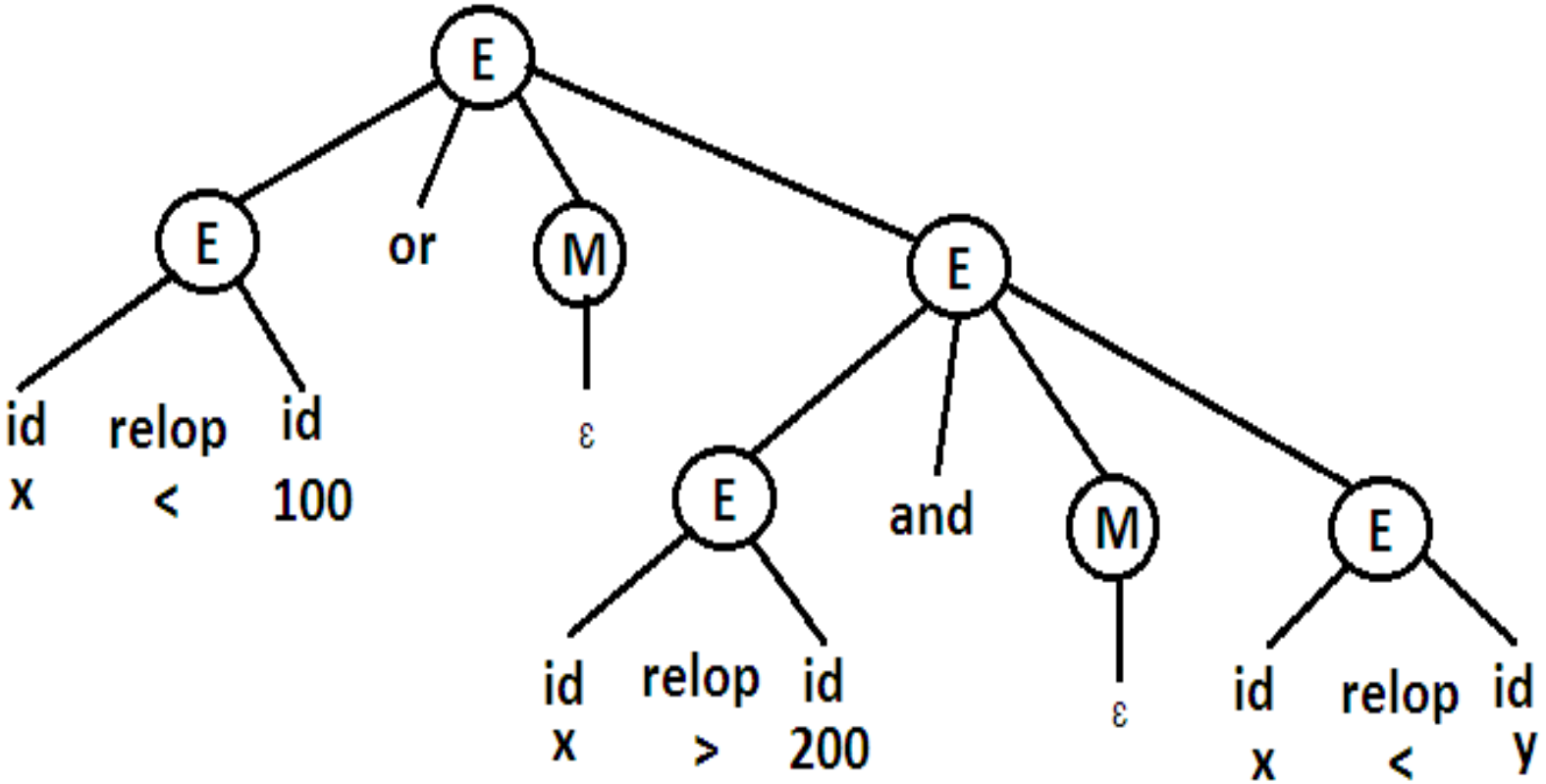
E→id relop id

M→ϵ

String is:

x<100 || x>200 && x < y

1. If x<100 goto()
2. goto()
3. If x>200 goto()
4. goto()
5. If x<y goto()
6. goto()


E
E
or
M
E
id
x
relop
<
id
100
ε
E
and
M
E
id
x
relop
>
id
200
ε
id
x
relop
<
id
y

makelist(*i*) creates a new list containing only *i*, an index into the array of quadruples; *makelist* returns a pointer to the list it has made.

merge(p_1,p_2) concatenates the lists pointed to by p_1 and p_2, and returns a pointer to the concatenated list.

backpatch(*p*,*i*) inserts i as the target label for each of the statements on the list pointed to by *p*.

$E \rightarrow E_1$ **or** $M\ E_2$ { *backpatch* (E_1.*falselist*, *M.quad*);
E.truelist : = *merge*(E_1.*truelist*, E_2.*truelist*);
E.falselist : = E_2.*falselist* }

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

$E \rightarrow$ **id**$_1$ **relop id**$_2$ { *E.truelist* : = *makelist* (*nextquad*);
E.falselist : = *makelist*(*nextquad* + *1*);
gen (‘**if**’ **id**$_1$.*place* **relop**.*op* **id**$_2$.place ‘**goto**_’)
gen (‘**goto**_’) }

$M \rightarrow \varepsilon$ { *M.quad* : = *nextquad* }