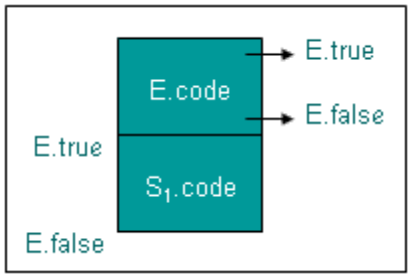


S ? if E then S_1

E.true = newlabel

E.false = S.next

S_1 .next = S.next

S.code = E.code ||

gen(E.true ':') ||

S_1 .code

Now we will consider the translation of boolean expressions into three address code generated by the following grammar:

S -> if E then S1

| if E then S1 else S2

| while E do S1

where E is the boolean expression to be translated.

Consider the following:

***newlabel* - returns a new symbolic label each time it is called.**

***E.true* - the label to which control flows if E is true.**

***E.false* - the label to which control flows if E is false.**

For *if-then*

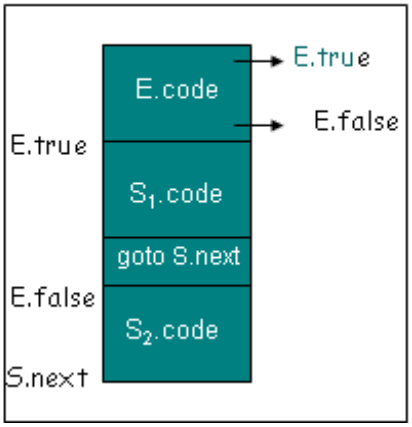
S -> if E then S1

E.true = newlabel //generate a new label for E.true

E.false = S.next //jump to S.next if E is false

S1.next = S.next

S.code = E.code || gen(E.true ':') || S1.code // associate the label created for E.true with the three address code for the first statement for S1



S ⟶ if E then S_1 else S_2

E.true = newlabel

E.false = newlabel

S_1 .next = S.next

S_2 .next = S.next

S.code = E.code ||

gen(E.true ':') ||

S_1 .code ||

gen(goto S.next) ||

gen(E.false ':') ||

S_2 .code

For *if-then-else*

S -> if E then S1 else S2

E.true = newlabel

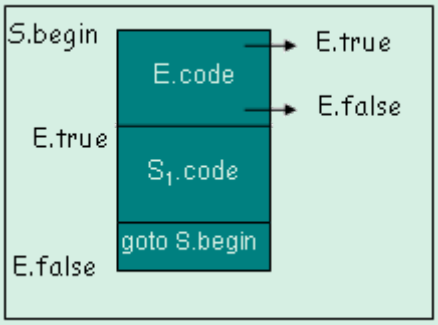
E.false = newlabel

S1.next = S.next

S2.next = S.next

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

In the above code, the labels E.true and E.false created are associated with the first three address code instructions for S1 and S2 respectively, so that if E is true, jump to S1 occurs and if E is false jump to S2 occurs. An explicit goto S.next is required after the code of S1 to ensure that after execution of code for S1 control moves to the statement after S instead of falling through the code of S2, in case E is true.



S ⟶ while E do S 1

S.begin = newlabel

E.true = newlabel

E.false = S.next

S 1 .next = S.begin

S.ocde = gen(S.begin ':') ||

E.code ||

gen(E.true ':') ||

S 1 .code ||

gen(goto S.begin)

For *while-do*

S -> while E do S1

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)

In case of while-do statement a new label S.begin is created and associated with the first instruction of the code for E, so that control can be transferred back to E after the execution of S1. E.false is set to S.next, so that control moves out of the code for S in case E is false. Since S1.next is set to S.begin, jumps from the code for S1 can go directly to S.begin.