Generazione di Codice

Esercizio1 - Testo

(a) Si dia uno schema di traduzione ascendente per la generazione di codice a 3 indirizzi che traduca il comando while descritto dalla seguente grammatica:

W::= HC endw
HC::= HE do C
HE::= HI E
HI::= while

(a1) assumendo epressioni con generazione per Loc
(a2) assumendo espressioni con generazione per short circuit

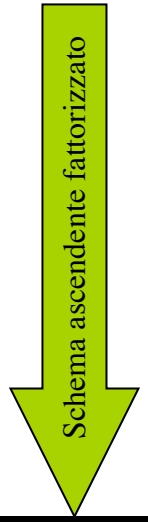
(b) Si mostri il codice generato dalla traduzione del seguente comando:

while x>5 do x:=x+7

mostrando prima il codice che si suppone sia generato da: x>5 e x:=x+7.

Esercizio 1- (a1)

[10]W::= while {init:= quad} E do
{w.next:= mk-L(quad);
emit('if' E.loc '= #false goto' --)} C {BK(C.next, init);
emit('goto' init)} endw



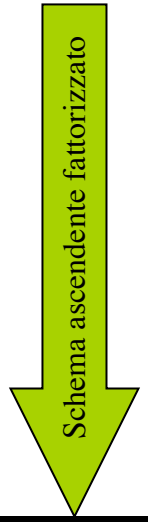
W::= HC endw
HC::= HE do C
HE::= HI E
HI::= while

Schema di traduzione

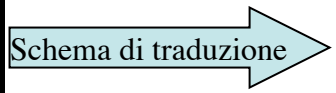
W::= HC endw
{emit('goto' HC.init);
W.next=HC.next;}
HC::= HE do C
{BK(C.next,HE.init);
HC.init = HE.init;
HC.next = HE.next;}
HE::= HI E
{HE.next = [quad];
emit('if' E.loc '= #false goto' --);
HE.init = HI.init;}
HI::= while
{HI.init = quad;}

Esercizio 1- (a2)

[10]W::= while {init:= quad} E do
{w.next:= E.false;
BK(E.true,quad);} C {BK(C.next, init);
emit('goto' init)} endw



W::= HC endw
HC::= HE do C
HE::= HI E
HI::= while



W::= HC endw
{emit('goto' HC.init);
W.next=HC.false;}
HC::= HE do C
{BK(C.next,HE.init);
HC.init = HE.init;
HC.false = HE.false;}
HE::= HI E
{HE.false = E.false;
BK(E.true,quad);
HE.init = HI.init;}
HI::= while
{HI.init = quad;}

Esercizio 1- (b)

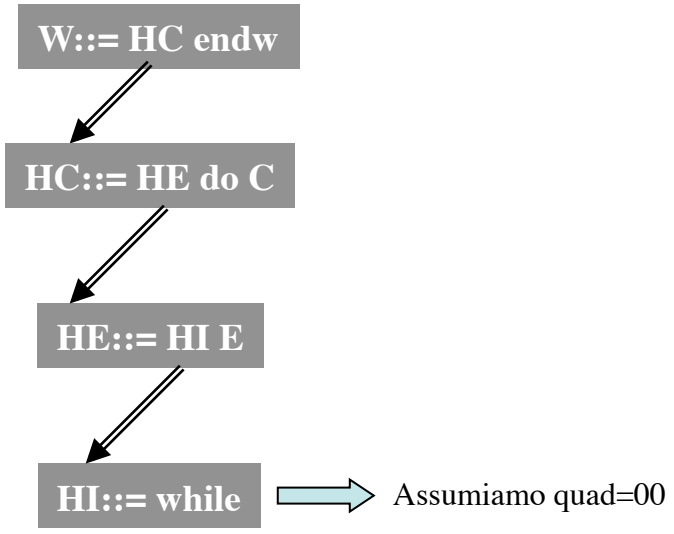
while x>5 do x:=x+7

Non possiamo usare Short-Circuit: Troppo complicato *emulare* operatori relazione:

x>5	loc1:=locx [>] #5	Assunto che: locx sia la locazione in symtab di x
x:=x+7	locx:=locx [+] #7	Assunto che: locx sia la locazione in symtab di x

Applichiamo lo schema di traduzione

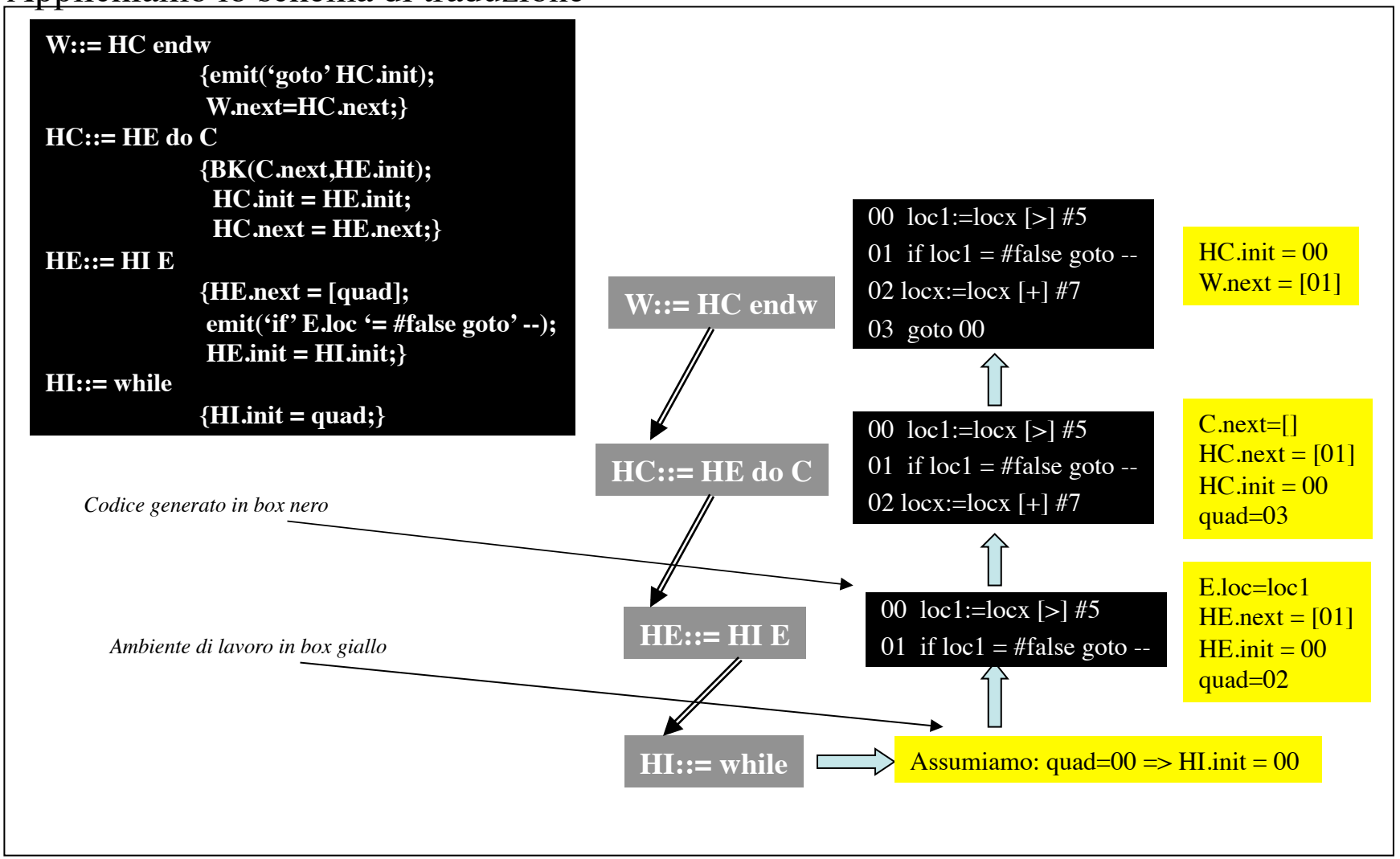
```
W::= HC endw
                {emit('goto' HC.init);
                 W.next=HC.next;}
HC::= HE do C
                {BK(C.next,HE.init);
                  HC.init = HE.init;
                  HC.next = HE.next;}
HE::= HI E
                {HE.next = [quad];
               emit('if' E.loc '= #false goto' --);
                 HE.init = HI.init;}
HI::= while
                {HI.init = quad;}
```



Esercizio 1- (b)

while x>5 do x:=x+7

Applichiamo lo schema di traduzione



Esercizio 2

Esercizio2 - Testo

(a) Si dia uno schema di traduzione discendente per la generazione di codice a 3 indirizzi che traduca il comando for del C descritto dalla seguente grammatica:

FR::= for (E_1 ; E_2 ; E_3) C

(a1) si dia lo schema di generazione di codice per espressioni per loc;

(a2) assumendo epressioni con generazione per Loc;

(a3) si dia lo schema di generazione di codice per E_2 con short circuit;

(a3) assumendo espressione E_2 con generazione per short circuit;

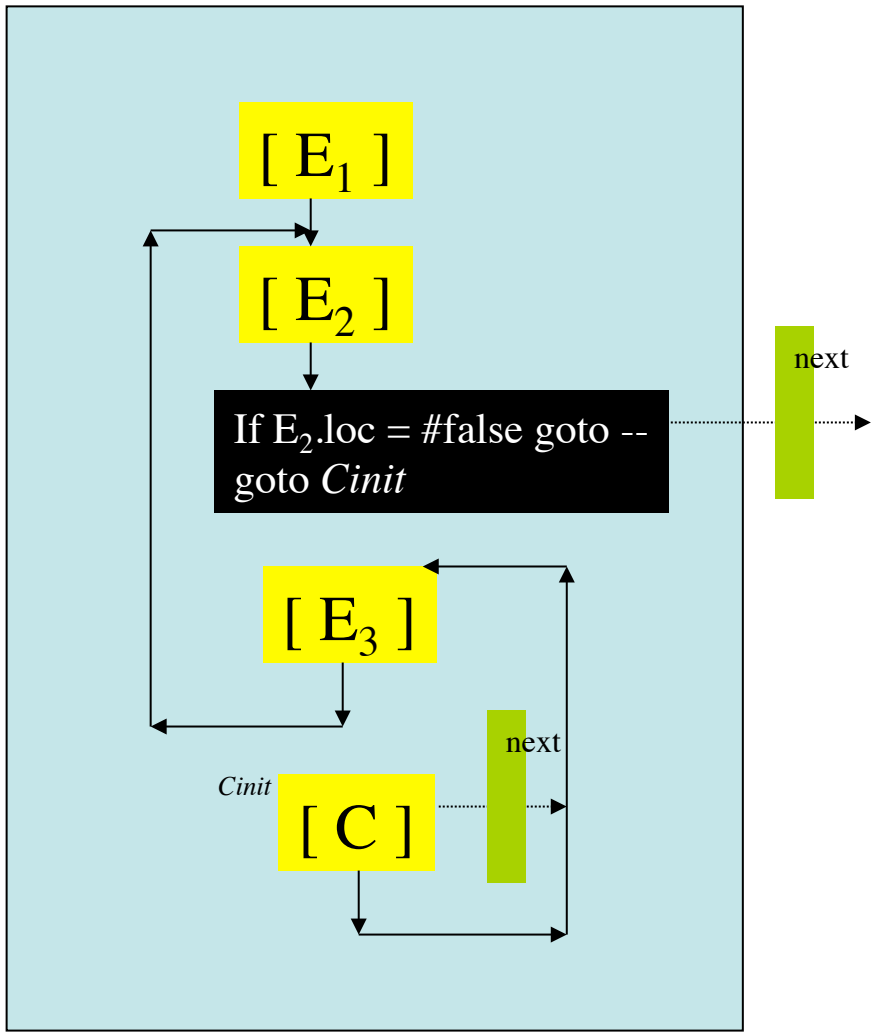
(b) Si mostri il codice generato dalla traduzione del seguente comando:

for (;true;)x:=y+7

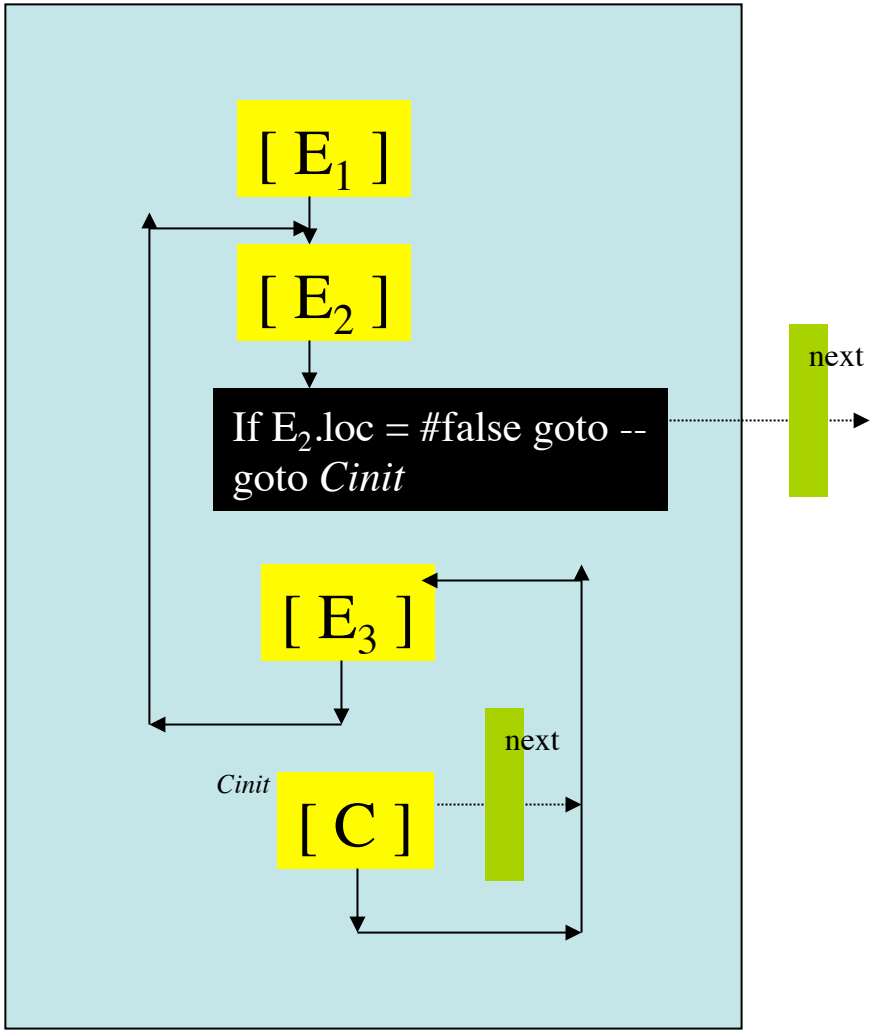
mostrando prima il codice che si suppone sia generato dai componenti.

Esercizio2 - (a1)

FR::= for (E_1 ; E_2 ; E_3) C



Esercizio2 - (a2)



FR::= for (E_1;{init:= quad}
E_2 {ifnext=[quad];
emit('if' E_2.loc '= #false goto' --);
Cinit=[quad];
emit('goto' --);
Einit=quad;}
E_3 {emit('goto' init);
BK(Cinit,quad);}
C {FR.next=ifnext;
BK(C.next,Einit);
emit('goto' Einit)}
endw

Esercizio 3

Esercizio3 - Testo

(a) Si dia uno schema di traduzione discendente per la generazione di codice a 3 indirizzi che traduca il comando for del C descritto dalla seguente grammatica:

CaSe::= case E of PairList Default
PairList::= E EList : C; PairList
PairList::= ε
EList::=, E EList
Default::= C

(a1) si dia lo schema di generazione di codice per espressioni per loc;
(a2) assumendo epressioni con generazione per Loc;

(b) Si mostri il codice generato dalla traduzione del seguente comando:

```
case x+y of x,x+2:x=y;
            5:x=y-x;
            x=3
```

mostrando prima il codice che si suppone sia generato dai componenti.

Esercizio 4

Esercizio3 - Testo

(a) Si cosiderino espressioni con sola somma di interi, identificatori di variabile, literals. Si dia uno schema di traduzione discendente che generi una traduzione source-to-source che fornisca un'espressione equivalente in cui sono stati calcolati tutti itermini che possono essere valutati a compile time. L'espressione risultante deve contenere al più un literal. Ad esempio:

x+3+z+y+2 è trasformata in x+z+y+5

(b) Si applichi lo schema dato alla trasformazione dell'espressione dell'esempio

(c) Lo si saprebbe estendere a espressioni con somme e prodotti, identificatori, literals, grouping e precedenza di prodotto?