

---

# Παραγωγή Ενδιάμεσου Κώδικα

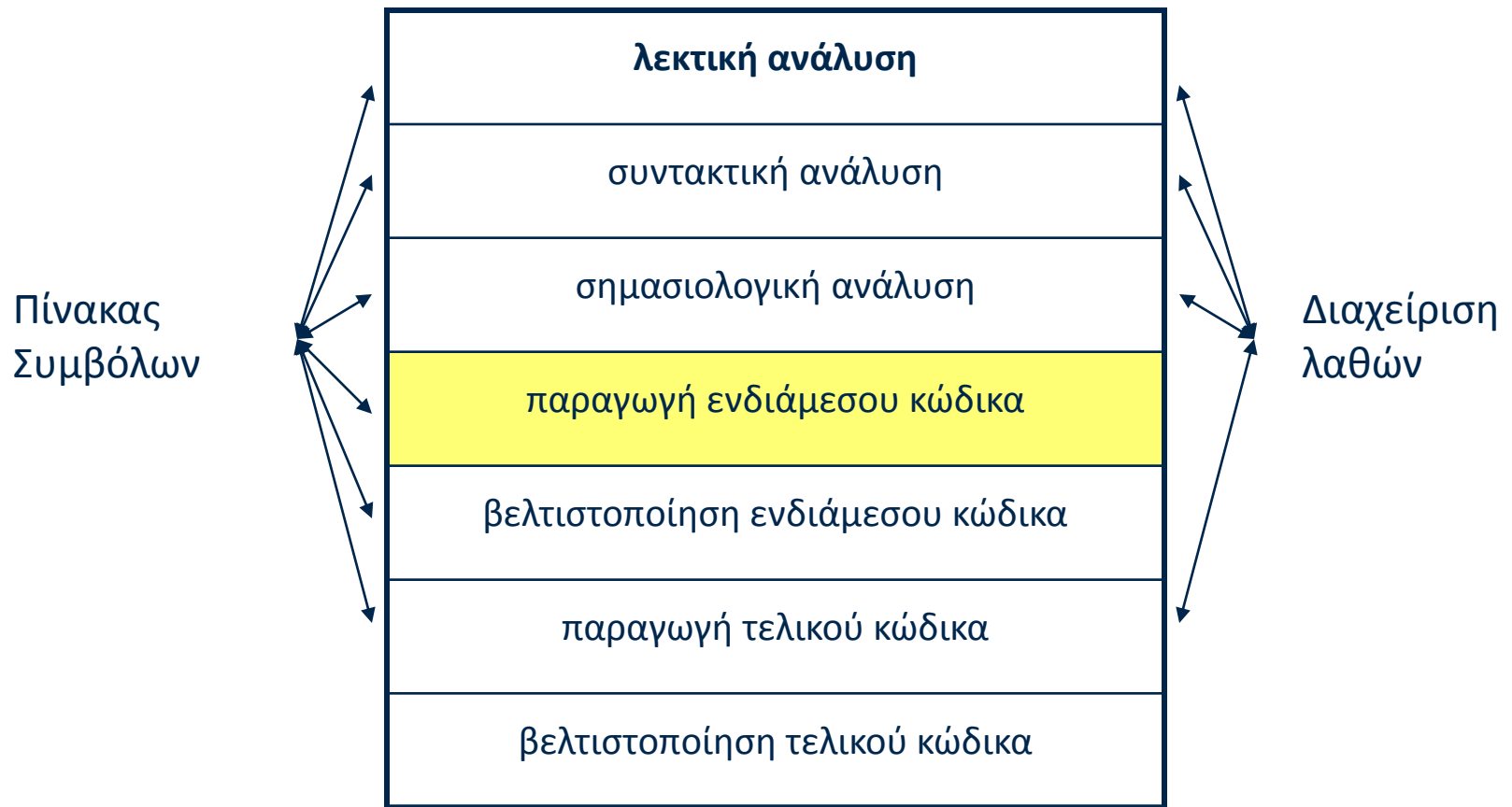


Διαλέξεις στο μάθημα: Μεταφραστές  
Γιώργος Μανής

---

## Οι Φάσεις της Μεταγλώττισης

---



## *Παραγωγή Ενδιάμεσου Κώδικα*

---



## Παραγωγή Ενδιάμεσου Κώδικα

---



## Ενδιάμεση Γλώσσα

---

- ‡ Ο ενδιάμεσος κώδικας είναι ένα σύνολο από τετράδες

- ένας τελεστής
- τρία τελούμενα

π.χ. + , a,b,t\_1

\*, t\_1,2,t\_2

:=,t\_2,\_,c

- ‡ Οι τετράδες είναι αριθμημένες. Κάθε τετράδα έχει μπροστά της έναν μοναδικό αριθμό που τη χαρακτηρίζει. Μόλις τελειώσει η εκτέλεση μίας τετράδας εκτελείται η τετράδα που έχει τον αμέσως μεγαλύτερο αριθμό, εκτός εάν η τετράδα που μόλις εκτελέστηκε υποδείξει κάτι διαφορετικό .

π.χ.: 100: +,a,b,c

110: +,d,e,f

## Οι τελεστές αριθμητικών πράξεων

---

τετράδες της μορφής:

$op, x, y, z$

- όπου το  $op$  μπορεί να είναι ένα εκ των:  $+, -, *, /$
- τα τελούμενα  $x, y$  μπορεί να είναι:
  - ονόματα μεταβλητών
  - αριθμητικές σταθερές
- το τελούμενο  $z$  μπορεί να είναι:
  - όνομα μεταβλητής

## Οι τελεστές αριθμητικών πράξεων

---

τετράδες της μορφής:

op, x, y, z

- εφαρμόζεται ο τελεστής op στα τελούμενα x και y και το αποτέλεσμα τοποθετείται στο τελούμενο z

π.χ.:

+ , a, b, c      αντιστοιχεί στην πράξη  $c = a + b$

/ , a, b, c      αντιστοιχεί στην πράξη  $c = a / b$

## Ο τελεστής εκχώρησης

---

τετράδες της μορφής:

$:=, x, \_, z$

- το τελούμενο  $x$  μπορεί να είναι:
  - όνομα μεταβλητής
  - αριθμητική σταθερά
- το τελούμενο  $z$  μπορεί να είναι:
  - όνομα μεταβλητής
- η τιμή του  $x$  εκχωρείται στη μεταβλητή  $z$ 
  - αντιστοιχεί στη εκχώρηση  $z:=x$



## Παράδειγμα

---

r:=4

pi:=3.14

area = pi \* r \* r

θα μπορούσε ισοδύναμα  
να είχαμε:



χωρίς να σημαίνει ότι  
αυτό ακριβώς θα βγάλει  
και ο μεταγλωττιστής

100: :=,4,\_,r

110: :=,3.14,\_,pi

120: \*,pi,r,T\_1

130: \*,T\_1,r,area

## Τελεστής άλματος χωρίς συνθήκη

---

jump, \_, \_, z

μεταπήδηση χωρίς όρους στη θέση z

π.χ. 100: :=,1,\_x

110: jump 130

120: :=,2,\_x

130 ...

όταν φτάσουμε στο 130 η τιμή του x θα είναι 1 και όχι 2

---

## Τελεστής άλματος χωρίς συνθήκη

---

relop, x, y, z

όπου relop ένας από τους τελεστές

=, >, <, <>, >=, <=

μεταπήδηση στη θέση z αν ισχύει η x relop y

π.χ. 100: =,a,4,120

110: jump,\_,\_,140

120: :=,1,\_b

130: jump 150

140: :=,2,\_b

150: ...

το b θα έχει την τιμή 1 αν ισχύει η συνθήκη a=4 και 2 αν δεν ισχύει

---

## *Αρχή και τέλος ενότητας*

---

- ‡ `begin_block, name, _, _`  
αρχή υποπρογράμματος ή προγράμματος με το όνομα `name`
  - ‡ `end_block, name, _, _`  
τέλος υποπρογράμματος ή προγράμματος με το όνομα `name`
  - ‡ `halt, _, _, _`  
τερματισμός προγράμματος
-

## *Αρχή και τέλος ενότητας*

---

Παράδειγμα begin\_block - end\_block, name - halt

100: begin\_block, add, \_, \_

110: :=, 1, \_, x

120: :=, 2, \_, y

130: +, x, y, z

140: halt, \_, \_, \_

150: end\_block, add, \_, \_

---

## Συναρτήσεις - Διαδικασίες

---

# par, x, m, \_

όπου x παράμετρος συνάρτησης και m ο τρόπος μετάδοσης

CV : μετάδοση με τιμή

REF: μετάδοση με αναφορά

RET: επιστροφή τιμής συνάρτησης

# call, name, \_, \_

κλήση συνάρτησης name

# ret, x, \_, \_

επιστροφή τιμής συνάρτησης

---

## Παράδειγμα κλήσης συνάρτησης

---

$x := \text{foo}(\text{in } a, \text{inout } b)$

θα μπορούσε ισοδύναμα  
να είχαμε:

χωρίς να σημαίνει ότι  
αυτό ακριβώς θα βγάλει  
και ο μεταγλωττιστής

100: par , a , cv , \_  
110: par , b , ref , \_  
120: par , T\_1 , ret , \_  
130: call , foo , \_ , \_ , \_  
140: ... τιμή στο x


## Παράδειγμα κλήσης διαδικασίας

---

call foo (in a, inout b)

θα μπορούσε ισοδύναμα  
να είχαμε:

χωρίς να σημαίνει ότι  
αυτό ακριβώς θα βγάλει  
και ο μεταγλωττιστής



```
100: par , a , cv , _  
110: par , b , ref , _  
120: call , foo , _ , _ , _  
130 ...
```



## Βοηθητικές Υπορουτίνες

---

- # nextquad()
  - επιστρέφει τον αριθμό της επόμενης τετράδας που πρόκειται να παραχθεί
- # genquad(op, x, y, z)
  - δημιουργεί την επόμενη τετράδα (op, x, y, z)
- # newtemp()
  - δημιουργεί και επιστρέφει μία νέα προσωρινή μεταβλητή
  - οι προσωρινές μεταβλητές είναι της μορφής  
T\_1, T\_2, T\_3 ...

## Βοηθητικές Υπορουτίνες

---

- # `emptylist()`
    - δημιουργεί μία κενή λίστα ετικετών τετράδων
  - # `makelist(x)`
    - δημιουργεί μία λίστα ετικετών τετράδων που περιέχει μόνο το  $x$
  - # `merge(list1, list2)`
    - δημιουργεί μία λίστα ετικετών τετράδων από τη συνένωση των λιστών  $list_1$ ,  $list_2$
  - # `backpatch(list,z)`
    - η λίστα  $list$  αποτελείται από δείκτες σε τετράδες των οποίων το τελευταίο τελούμενο δεν είναι συμπληρωμένο
    - η `backpatch` επισκέπτεται μία μία τις τετράδες αυτές και τις συμπληρώνει με την ετικέτα  $z$
-

## *Αρχή και Τέλος Block*

---

```
<PROGRAM>                                ::= program ID
                                           <PROGRAMBLOCK (ID)>

<PROGRAMBLOCK (name) > ::=
    <DECLARATIONS>
    <SUBPROGRAMS>
    genquad("begin_block",name,"_","_")
    <BLOCK>
    if (this is the main program block)
        genquad("halt","_","_","_")
    genquad("end_block",name,"_","_")
```

---

## Αριθμητικές Παραστάσεις

---

Παράδειγμα:

$$x + (y + z) \times w$$

ενδιάμεσος κώδικας:

1: +, y, z, T\_1

2: ×, T\_1, w, T\_2

3: +, x, T\_2, T\_3

## Αριθμητικές Παραστάσεις

$E \rightarrow T^1 ( + T^2 \{P_1\} )^* \{P_2\}$

$\{P_1\}$ :  $w = \text{newTemp}()$

Νέα προσωρινή μεταβλητή που θα κρατήσει το μέχρι στιγμής αποτέλεσμα

$\text{genquad}("+", T^1.\text{place}, T^2.\text{place}, w)$

Παραγωγή τετράδας που προσθέτει το μέχρι στιγμής αποτέλεσμα στο νέο  $T^2$

$T^1.\text{place} = w$

Το μέχρι στιγμής αποτέλεσμα τοποθετείται στην  $T^1$  ώστε να χρησιμοποιηθεί αν υπάρξει επόμενο  $T^2$

$\{P_2\}$ :  $E.\text{place} = T^1.\text{place}$

Όταν δεν υπάρχει άλλο  $T^2$  το αποτέλεσμα είναι στο  $T^1$

## Αριθμητικές Παραστάσεις

---

$T \rightarrow F^1 (\times F^2 \{P_1\})^* \{P_2\}$

$\{P_1\}$ :      $w = \text{newTemp}()$

$\text{genquad}(\text{"\times"}, F^1.\text{place}, F^2.\text{place}, w)$

$F^1.\text{place} = w$

$\{P_2\}$ :      $T.\text{place} = F^1.\text{place}$

Ανάλογη λογική με τον κανόνα E

---

## Αριθμητικές Παραστάσεις

---

$F \rightarrow ( E ) \{P_1\}$

Απλή μεταφορά από το E.place στο F.place

$\{P_1\}: F.place = E.place$

$F \rightarrow id \{P_1\}$

Απλή μεταφορά από το id.place στο F.place

$\{P_1\}: F.place = id.place$

---

## *Αριθμητικές Παραστάσεις*

---

```
procedure E (E.place)
  begin
    T ( T1.place )
    while token=plustk do begin
      lex();
      T (T2.place)
      w:=newTemp()
      genquad( "+", T1.place, T2.place, w)
      T1.place :=w
    end
    E.place := T1.place
  end
```



## Λογικές Παραστάσεις

---

Έστω η γραμματική

$B \rightarrow Q \text{ ( or } Q \text{ )}^*$

$Q \rightarrow R \text{ ( and } R \text{ )}^*$

$R \rightarrow ( B )$

$R \rightarrow E \text{ relop } E$

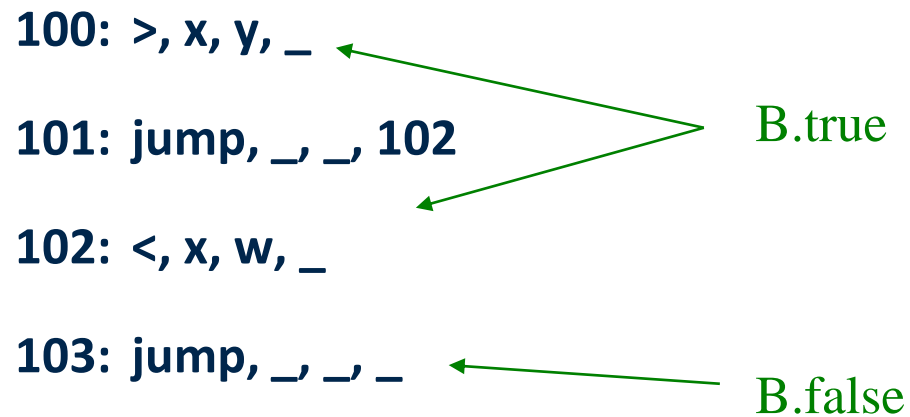
---

## Λογικές Παραστάσεις - OR

---

Παράδειγμα:

$B = x > y \text{ or } x < w$

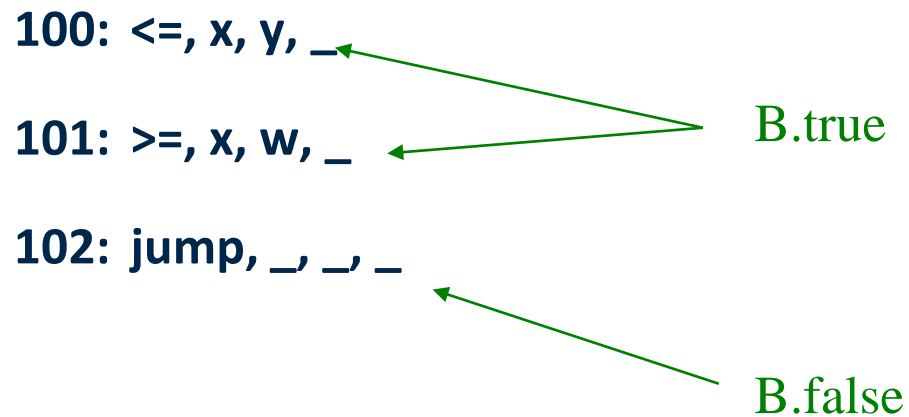


## Λογικές Παραστάσεις - OR

---

Παράδειγμα:

$B = x > y \text{ or } x < w$

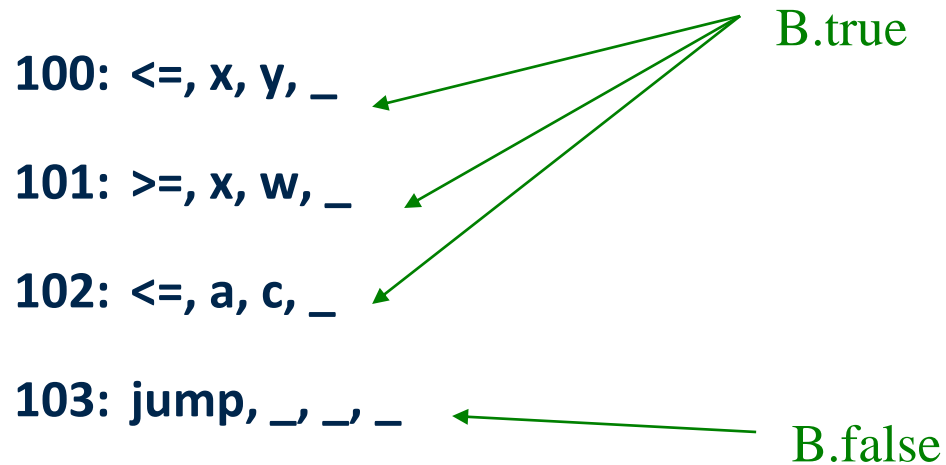


## Λογικές Παραστάσεις - OR

---

Παράδειγμα:

$B = x > y \text{ or } x < w \text{ or } a > c$



## Λογικές Παραστάσεις - OR

$B \rightarrow Q^1 \{P_1\} ( \text{or } \{P_2\} Q^2 \{P_3\} )^*$

$\{P_1\}$ :  $B.\text{true} = Q^1.\text{true}$

$B.\text{false} = Q^1.\text{false}$

$\{P_2\}$ :  $\text{backpatch}(B.\text{false}, \text{nextquad}())$

$\{P_3\}$ :  $B.\text{true} = \text{merge}(B.\text{true}, Q^2.\text{true})$

$B.\text{false} = Q^2.\text{false}$

Μεταφορά των τετράδων  
από τη λίστα  $Q^1$  στη λίστα  $B$

Συμπλήρωση όσων τετράδων μπορούν  
να συμπληρωθούν μέσα στον κανόνα

Συσώρευση στη λίστα  $\text{true}$  των τετράδων  
που δεν μπορούν να συμπληρωθούν και  
αντιστοιχούν σε αληθή αποτίμηση  
λογικής παράστασης

Η λίστα  $\text{false}$  περιέχει την τετράδα η οποία  
αντιστοιχεί σε στη μη αληθή αποτίμηση της  
λογικής παράστασης

## Λογικές Παραστάσεις - AND

---

Παράδειγμα:

$B = x > y \text{ and } x < w$

100: >, x, y, 102

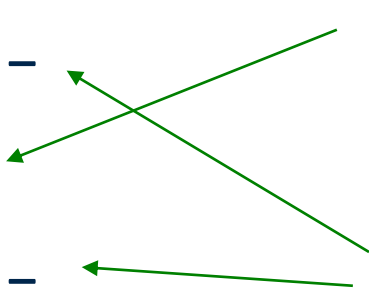
101: jump, \_, \_, \_

102: <, x, w, \_

103: jump, \_, \_, \_

B.true

B.false



## Λογικές Παραστάσεις - AND

$Q \rightarrow R^1 \{P_1\} ( \text{and } \{P_2\} R^2 \{P_3\} )^*$

$\{P_1\}$ :  $Q.\text{true} = R^1.\text{true}$

$Q.\text{false} = R^1.\text{false}$

$\{P_2\}$ :  $\text{backpatch}(Q.\text{true}, \text{nextquad}())$

$\{P_3\}$ :  $Q.\text{false} = \text{merge}(Q.\text{false}, R^2.\text{false})$

$Q.\text{true} = R^2.\text{true}$

Μεταφορά των τετράδων  
από τη λίστα  $R^1$  στη λίστα  $Q$

Συμπλήρωση όσων τετράδων μπορούν  
να συμπληρωθούν μέσα στον κανόνα

Συσώρευση στη λίστα false των τετράδων  
που δεν μπορούν να συμπληρωθούν και  
αντιστοιχούν σε μη αληθή αποτίμηση  
λογικής παράστασης

Η λίστα true περιέχει την τετράδα η οποία  
αντιστοιχεί σε αληθή αποτίμηση της  
λογικής παράστασης

## Λογικές Παραστάσεις

---

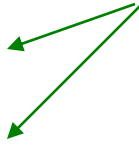
$R \rightarrow (B) \{P_1\}$

$\{P_1\}$ :

$R.true = B.true$

$R.false = B.false$

Μεταφορά των τετράδων  
από τη λίστα B στη λίστα R





## Λογικές Παραστάσεις

---

$R \rightarrow E^1 \text{ relop } E^2 \{P_1\}$

$\{P_1\}$ :      $R.\text{true} = \text{makelist}(\text{nextquad}())$   
                  $\text{genQuad}(\text{relop}, E^1.\text{place}, E^2.\text{place}, \text{"\_"} )$

Δημιουργία μη συμπληρωμένης  
τετράδας και εισαγωγή στη λίστα  
μη συμπληρωμένων τετράδων για  
την αληθή αποτίμηση της relop

$R.\text{false} = \text{makelist}(\text{nextquad}())$   
 $\text{genQuad}(\text{"jump"}, \text{"\_"}, \text{"\_"}, \text{"\_"} )$

Δημιουργία μη συμπληρωμένης  
τετράδας και εισαγωγή στη λίστα  
μη συμπληρωμένων τετράδων για  
τη μη αληθή αποτίμηση της relop

## *Κλήση Υποπρογραμμάτων*

---

Κλήση διαδικασίας:

call assign\_v (in a, inout b)

par, a, CV, \_

par, b, REF, \_

call, assign\_v , \_ , \_

---

## *Κλήση Υποπρογραμμάτων*

---

Κλήση συνάρτησης:

error = assign\_v (in a, inout b)

par, a, CV, \_

par, b, REF, \_

w = newTemp()

par, w, RET, \_

call, assign\_v , \_ , \_

---

## *Εντολή return*

---

**S -> return (E) {P1}**

**{P1}:      genquad("retv",E.place,"\_","\_")**

## Εκχώρηση

---

$S \rightarrow id := E \{P1\};$

$\{P1\} : \quad \text{genQuad}(":=", E.\text{place}, \_ , id)$

---

## Δομή while

---

**S -> while {P1} B do {P2} S<sup>1</sup> {P3}**

**{P1}:      Bquad:=nextquad()**

**{P2}:      backpatch(B.true,nextquad())**

**{P3}:      genquad("jump","\_","\_",Bquad)  
            backpatch(B.false,nextquad())**

Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, το true πάνω στην S και το false έξω από τη δομή

Μετάβαση στην αρχή της συνθήκης ώστε να ξαναγίνει έλεγχος

## Δομή Repeat...Until


---

**S -> repeat {P1} S<sup>1</sup> until (cond) {P2}**


**{P1}:      sQuad:=nextquad()**

**{P2}:      backpatch(cond.False,sQuad)  
            backpatch(cond.True,nextquad())**

Οι τετράδες αυτές πρέπει να μεταβούν στην αρχή της συνθήκης για να επανελεγχθεί



Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, δηλαδή έξω από τη δομή



## Δομή if

#  $S \rightarrow \text{if } B \text{ then } \{P1\} S^1 \{P2\} \text{TAIL } \{P3\}$

$\{P1\}$ :     `backpatch(B.true,nextquad())`

$\{P2\}$ :     `ifList=makelist(nextquad())`

`genquad("jump","_","_","_")`

`backpatch(B.false,nextquad())`

$\{P3\}$ :     `backpatch(ifList,nextquad())`

Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, στο if και else αντίστοιχα

Εξασφαλίζουμε ότι εάν εκτελεστούν οι εντολές του if δε θα εκτελεστούν στη συνέχεια οι εντολές του else

#  $\text{TAIL} \rightarrow \text{else } S^2 \mid \text{TAIL} \rightarrow \epsilon$



## *Είσοδος - Έξοδος*

---

**S -> input (id) {P1}**

**{P1}:      genquad("inp",id.place,"\_","\_")**

**S -> print (E) {P2}**

**{P2}:      genquad("out",E.place,"\_","\_")**

---