Διάλεξη 9: Αλγόριθμοι Αμοιβαίου Αποκλεισμού με τη χρήση μεταβλητών Ανάγνωσης/Εγγραφής

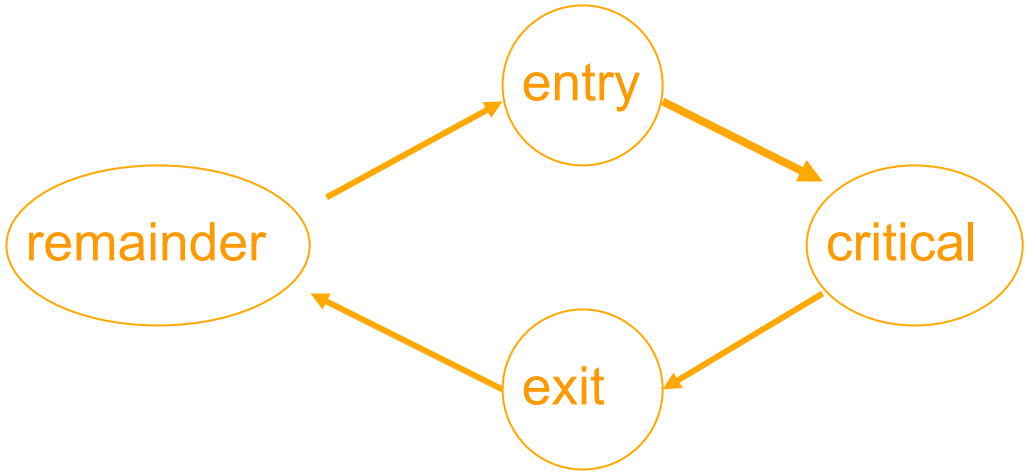
ΕΠΛ 432: Κατανεμημένοι Αλγόριθμοι

Τι θα δούμε σήμερα

- Αλγόριθμος Ψησταριάς (Bakery Algorithm)
- Αλγόριθμος 2-επεξεργαστών
- Αλγόριθμος n-επεξεργαστών (Δέντρο Ανταγωνισμού)

Το Πρόβλημα του Αμοιβαίου Αποκλεισμού

- Ο κώδικας κάθε επεξεργαστή χωρίζεται σε τέσσερις τομείς:



  - **entry**: συγχρονίσου με τους άλλους για να διασφαλιστεί αποκλειστική πρόσβαση στο …
  - **critical**: χρησιμοποίησε τον πόρο και ακολούθως…
  - **exit**: καθάρισε τα ίχνη σου και μπες σε κατάσταση όπου…
  - **remainder**: δεν σε ενδιαφέρει η χρήση του πόρου

Αλγόριθμοι Αμοιβαίου Αποκλεισμού

- Ένας αλγόριθμος αμοιβαίου αποκλεισμού προσδιορίζει τον κώδικα για τους τομείς εισόδου (entry) και εξόδου (exit) και διασφαλίζει:
  - **Αμοιβαίος Αποκλεισμός** (Συνθήκη Ασφαλείας): Σε κάθε εκτέλεση του αλγορίθμου το πολύ ένας επεξεργαστής βρίσκεται στο κρίσιμο τμήμα σε κάθε διάταξη.
  - Κάποιο είδος συνθήκης ζωτικότητας ή προόδου.

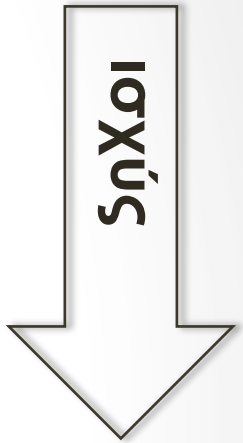
Συνθήκες Ζωτικότητας

- **Όχι Αδιέξοδο**: Αν κάποιος επεξεργαστής βρίσκετε στο τμήμα εισόδου (entry) σε κάποια διάταξη, τότε κάποιος επεξεργαστής θα βρεθεί μέσα στο κρίσιμο τμήμα (critical) σε κάποια μεταγενέστερη διάταξη

- **Όχι Παρατεταμένη Στέρηση**: Αν κάποιος επεξεργαστής βρίσκετε στο τμήμα εισόδου (entry) σε κάποια διάταξη, τότε ο *ίδιος επεξεργαστής* θα βρεθεί μέσα στο κρίσιμο τμήμα (critical) σε κάποια μεταγενέστερη διάταξη

Τύποι μεταβλητών

- Ανάγνωσης/Μεταβολής/Εγγραφής (Read/Modify/Write)
- Ελέγχου/Ενημέρωσης (Test&Set)
- Ανάγνωσης/Εγγραφής (Read/Write)



Μεταβλητή Read / Write

- Απλές λειτουργίες Εγγραφής και Ανάγνωσης

- Σε ένα ***ατομικό*** βήμα ένας επεξεργαστής μπορεί να:
  - Ανάγνωση: Επιστρέψει την τιμή της κοινόχρηστης μεταβλητής
  - Εγγραφή: Αλλάξει την τιμή της κοινόχρηστης μεταβλητής
  - Αλλά όχι και τα δύο

Αλγόριθμος Ψησταριάς (Bakery Algorithm)

- Ιδέα: Πελάτες σε Ψησταριά (στο Δημόσιο στη περίπτωση της Κύπρου)
  - Ο κάθε πελάτης παίρνει ένα αριθμό και ο πελάτης με τον μικρότερο αριθμό είναι ο επόμενος που εξυπηρετείται

- Ο αλγόριθμος εγγυάται
  - Αμοιβαίο Αποκλεισμό χωρίς Παρατεταμένη Στέρηση

- Χρησιμοποιεί 2 πίνακες κοινόχρηστων μεταβλητών ανάγνωσης και εγγραφής:
  - `Number`: πίνακας n ακεραίων μεταβλητών
  - `Choosing`: πίνακας n διαδικών μεταβλητών

Κοινόχρηστες Μεταβλητές

- booleans `Choosing[i]`: αρχικά FALSE
  - Μεταβάλλεται από τον επεξεργαστή p_i και διαβάζεται από όλους
  - Χρησιμοποιείται για να δείξει ότι ένας επεξεργαστής είναι στην διαδικασία επιλογής αριθμού

- integers `Number[i]`: αρχικά 0
  - Μεταβάλλεται από τον επεξεργαστή p_i και διαβάζεται από όλους
  - Δείχνει τον αριθμό που επέλεξε ο επεξεργαστής p_i

- Επιλογή Αριθμού:
  - Βρες τον μεγαλύτερο επιλεγμένο αριθμό και αύξησέ τον κατά 1

- Πότε ο p_i μπαίνει στο κρίσιμο τμήμα:
  - Όταν `Number[i]` είναι ο μικρότερος αριθμός
  - Σύγκριση ζευγών `(Number[i], i)` για αποφυγή συμμετρικών τιμών

Αλγόριθμος Ψησταριάς (Bakery Algorithm)

Code for entry section:

```
Choosing[i] := true
Number[i] := max{Number[0], …,
                 Number[n-1]} + 1
Choosing[i] := false
for j := 0 to n-1 (except i) do
   wait until Choosing[j] = false
   wait until Number[j] = 0 or
    (Number[j],j) > (Number[i],i)
endfor
```

Code for exit section:

```
Number[i] := 0
```

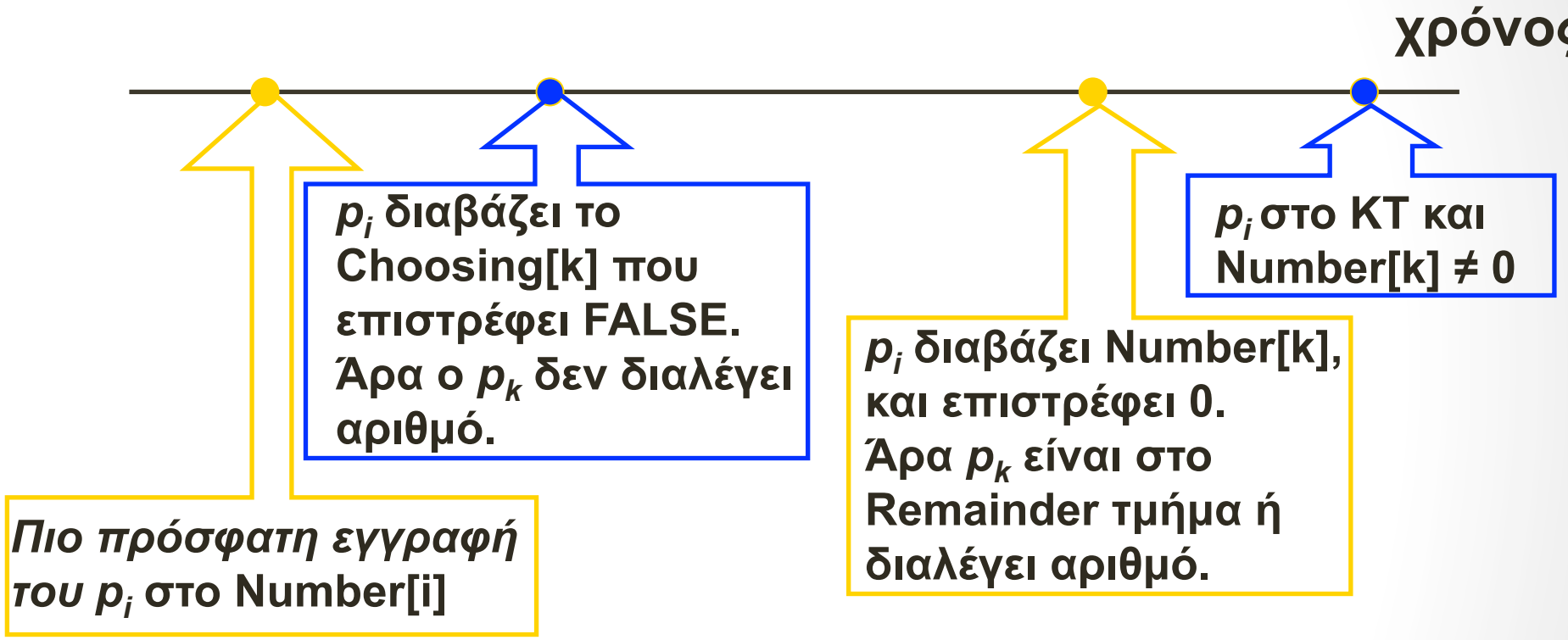
Απόδειξη Αμοιβαίου Αποκλεισμού

- **Λήμμα 1:** Αν ένας επεξεργαστής p_i μπει στο κρίσιμο τμήμα και $\texttt{Number}[k] \neq 0$ ($k \neq i$), τότε $(\texttt{Number}[k],k) > (\texttt{Number}[i],i)$.

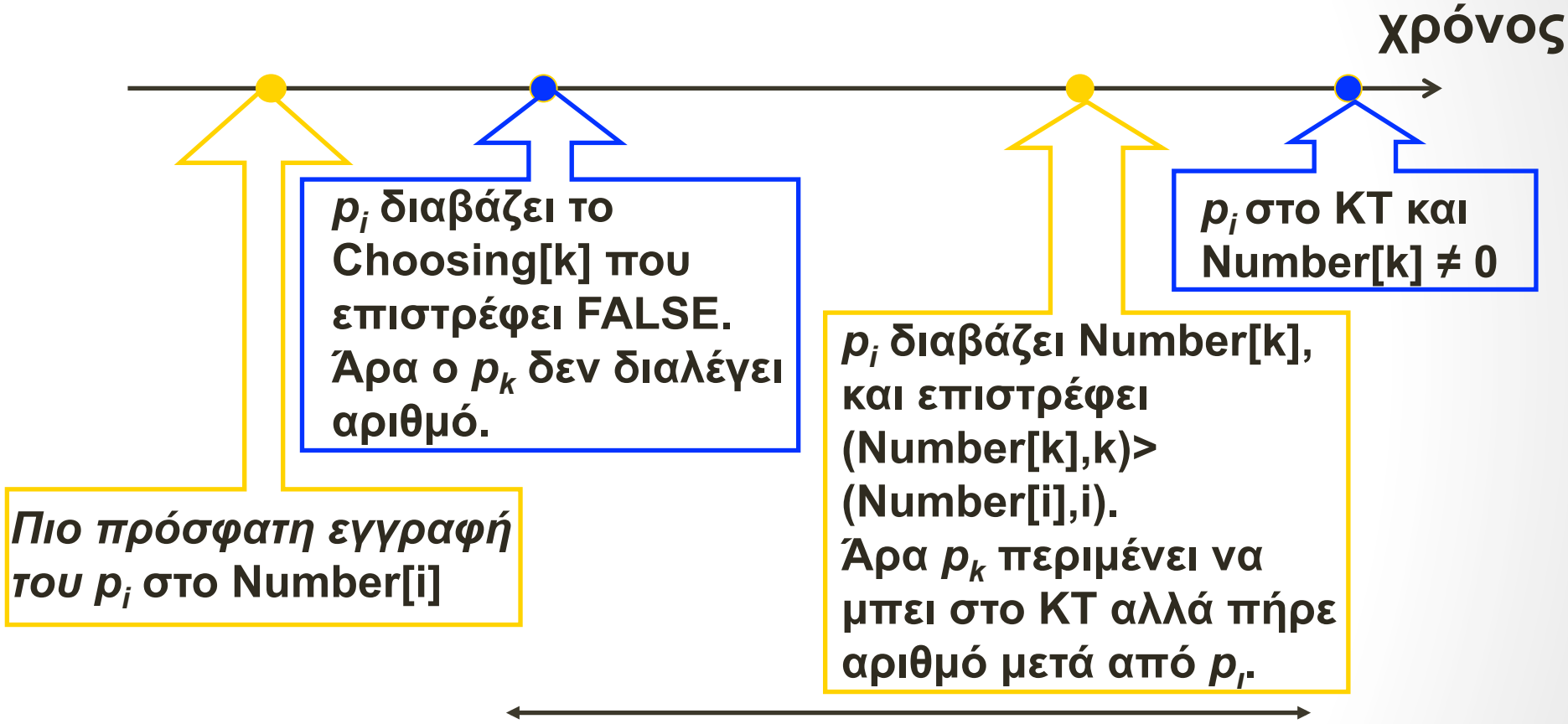
- Ο p_i διαβάζει την τιμή του $\texttt{Number}[k]$ πριν μπει στο ΚΤ

- Υπάρχουν 2 περιπτώσεις
  - Περίπτωση 1: $\texttt{Number}[k] = 0$
  - Περίπτωση 2: $(\texttt{Number}[k], k) > (\texttt{Number}[i], i)$

Περίπτωση 1: Number[k] =0



Περίπτωση 2: (Number[k],k)>(Number[i],i)



p_k διαλέξε ένα αριθμό μεγαλύτερο από Number[i]. Άρα πρέπει να διάλεξε σε αυτό το διάστημα.

Απόδειξη Αμοιβαίου Αποκλεισμού

- **Λήμμα 2:** Αν ένας επεξεργαστής p_i μπει στο κρίσιμο τμήμα, τότε `Number[i]` > 0.
  - Μπορεί να αποδειχτεί εύκολα με επαγωγή.

- Αμοιβαίος Αποκλεισμός: Υποθέτουμε ότι μπορούν 2 επεξεργαστές p_i και p_k να μπουν στο ΚΤ ταυτόχρονα
  - Από Λήμμα 2 έπεται: `Number[i]`, `Number[k]` > 0
  - Από Λήμμα 1 έπεται:
    - (`Number[k]`,k) > (`Number[i]`,i) και
    - (`Number[i]`,i) > (`Number[k]`,k)

Αντίφαση

Αποφυγή Παρατεταμένης Στέρησης

- Ας υποθέσουμε ότι υπάρχει ΠΣ
- Που μπορεί να συμβεί;
  - Γραμμές 5 ή 6 όπου περιμένουμε
  - Όχι στην επιλογή αριθμού
- Ας υποθέσουμε ότι ο p_i δεν μπορεί να μπει στο ΚΤ (στέρηση) και έχει το μικρότερο ζεύγος `(Number[i],i)`
- Ας υποθέσουμε ότι ο επεξεργαστής p_k ήρθε μετά τον p_i αλλά μπαίνει στο ΚΤ πριν από τον p_i (αφού ο p_i έχει ΠΣ)

Αποφυγή Παρατεταμένης Στέρησης

- Για να μπορεί όμως ο p_k να μπεί στο ΚΤ πρέπει να περιμένει όλους τους επεξεργαστές με μικρότερο αριθμό να μπουν και να βγούν από το ΚΤ.

- Αφού ο p_k έρχεται μετά από το p_i σημαίνει ότι
  - `Number[k] > Number[i]`

- Επομένως για να μπει ο p_k στο ΚΤ, ο p_i δεν μπορεί να κολλήσει στις γραμμές 5 ή 6

Αντίφαση

Πολυπλοκότητα Αλγορίθμου

- Αριθμός Κοινόχρηστων Μεταβλητών: Θ(2n)
- Μεταβλητές `Choosing` είναι δυαδικές
- Μεταβλητές `Number` είναι μη φραγμένοι ακέραιοι
- Μπορούμε να βρούμε ένα αλγόριθμο που χρησιμοποιεί πιο λιγότερο κοινόχρηστο χώρο;

Φραγμένος αλγόριθμος 2-επεξεργαστών

- Ιδέα: Προσπαθούμε να πάρουμε ένα φραγμένο αλγόριθμο 2-επεξεργαστών και μετά να τον επεκτείνουμε σε αλγόριθμο που υποστηρίζει n-επεξεργαστές

- Χρήση 3 δυαδικών κοινόχρηστων μεταβλητών ανάγνωσης/εγγραφής:
  - `W[0]`: αρχικά 0, μεταβάλλεται από p_0 και διαβάζεται από p_1
  - `W[1]`: αρχικά 0, μεταβάλλεται από p_1 και διαβάζεται από p_0
  - `Priority`: αρχικά 0, μεταβάλλεται και διαβάζεται από τους δυο

- Κάθε μεταβλητή W[] δηλώνει την επιθυμία χρήσης του ΚΤ

Αλγόριθμος επεξεργαστή p_0

Code for p_0 's entry section:

```
1   .
2   .
3   W[0] := 1
4   .
5   .
6   wait until W[1] = 0
```

Code for p_0 's exit section:

```
7   .
8   W[0] := 0
```

Αλγόριθμος επεξεργαστή p_1

Code for p_1 's entry section:

```
1   W[1] := 0
2   wait until W[0] = 0
3   W[1] := 1
4   .
5       if (W[0] = 1) then goto Line 1
6   .
```

Code for p_1 's exit section:

```
7   .
8   W[1] := 0
```

Χαρακτηριστικά αλγορίθμου

- Ικανοποιεί τον Αμοιβαίο Αποκλεισμό
- Επιτυγχάνει αποφυγή Αδιεξόδου
- Δεν επιτυγχάνει Παρατεταμένη Στέρηση
  - Δίνει προτεραιότητα στον p_0
- Για να επιφέρουμε δικαιοσύνη θα χρησιμοποιήσουμε την μεταβλητή `Priority`

Αλγόριθμος 2-επεξεργαστών

Code for entry section:

```
1   W[i] := 0
2   wait until W[1-i] = 0 or Priority = i
3   W[i] := 1
4   if (Priority = 1-i) then
5         if (W[1-i] = 1) then goto Line 1
6   else wait until (W[1-i] = 0)
```

Code for exit section:

```
7   Priority := 1-i
8   W[i] := 0
```

Χαρακτηριστικά αλγορίθμου

- Ικανοποιεί τον Αμοιβαίο Αποκλεισμό
- Επιτυγχάνει αποφυγή Αδιεξόδου
- Επιτυγχάνει αποφυγή Παρατεταμένης Στέρησης

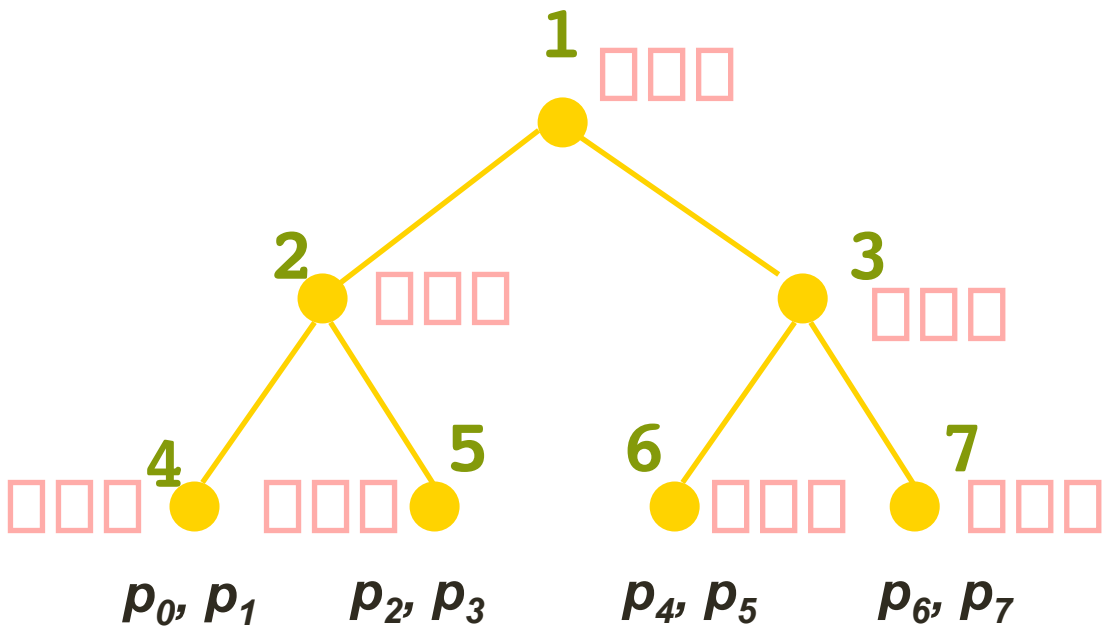
Φραγμένος Αλγόριθμος για n-επεξεργαστές

- Πως μπορούμε να επεκτείνουμε την ιδέα των 2-επεξεργαστών για να λύσουμε το πρόβλημα με n-επεξεργαστές;

- Χρησιμοποιώντας την ιδέα του Δέντρου Ανταγωνισμού (Tournament Tree)
  - Πλήρες Δυαδικό Δέντρο

- Τρέχουμε τον αλγόριθμο 2-επεξεργαστών σε κάθε επίπεδο του δέντρου
  - Κάθε επίπεδο έχει τα δικά του αντίγραφα των κοινόχρηστων μεταβλητών
  - Ο νικητής ανταγωνίζεται στο πιο πάνω επίπεδο
  - Αυτός που κερδίζει στην ρίζα μπαίνει στο ΚΤ

Δέντρο Ανταγωνισμού



Χαρακτηριστικά Αλγορίθμου

- **Ορθότητα:** Βασίζεται στην ορθότητα του αλγορίθμου 2-επεξεργαστών
  - Αμοιβαίος Αποκλεισμός: έπεται από τον αμοιβαίο αποκλεισμό 2-επεξεργαστών στην ρίζα του δέντρου
  - Μη Παρατεταμένη Στέρηση: έπεται από την μη παρατεταμένη στέρηση του αλγόριθμου 2-επεξεργαστών σε κάθε κόμβο του δέντρου.

- **Χωρική πολυπλοκότητα:** 3n δυαδικές κοινόχρηστες μεταβλητές
  - n: ο αριθμός των κόμβων του δέντρου

Ερωτήσεις;