Παράλληλη Επεξεργασία
Κεφάλαιο 2 – Παραλληλισμός Δεδομένων

Κωνσταντίνος Μαργαρίτης
Καθηγητής
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
kmarg@uom.gr
http://eos.uom.gr/~kmarg

Αρετή Καπτάν
Υποψήφια Διδάκτορας
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
areti@uom.gr
http://eos.uom.gr/~areti

Παραλληλισμός Δεδομένων

- Ο παραλληλισμός των δεδομένων
- Σειρές φωλιασμένων βρόχων
- Θέματα απόδοσης:
  - Δημιουργία διεργασιών
  - Ανταγωνισμός για πρόσβαση στην μνήμη
  - Καθυστέρηση για συγχρονισμό διεργασιών

Δημιουργία Διεργασιών

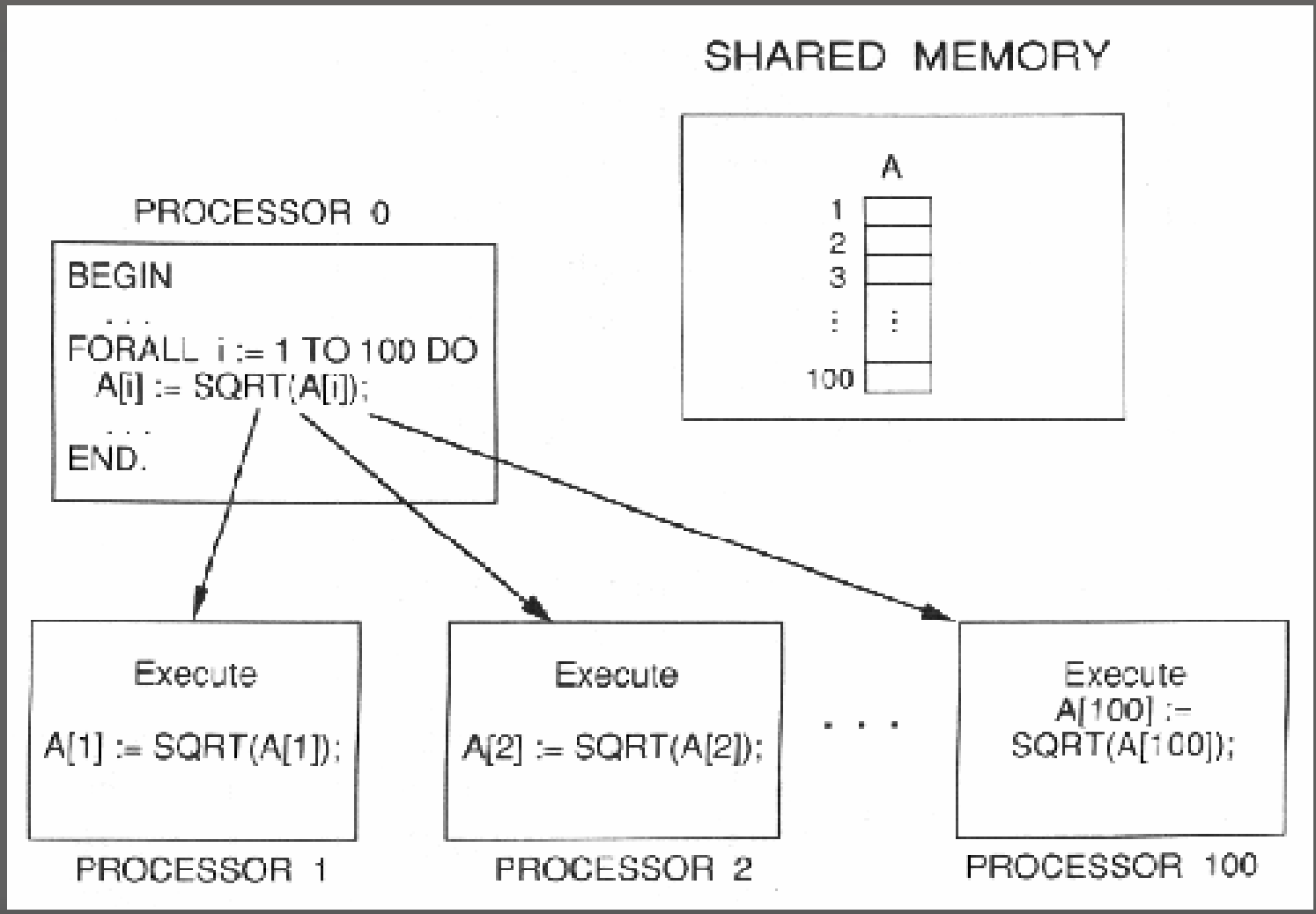
- Η έννοια της διεργασίας (σειριακό πρόγραμμα)
- Εντολές δημιουργίας παράλληλων διεργασιών
- Διεργασία-γονέας, διεργασία-παιδί

Υπολογισμός των τετραγωνικών ριζών των στοιχείων ενός πίνακα

```
PROGRAM Squareroot;
VAR
  A: ARRAY [1..100] OF REAL;
  i: INTEGER;
BEGIN
  .....
  FOR i := 1 TO 100 DO
    A[i]:= SQRT(A[i]);
    .....
END.
```

```
PROGRAM ParallelSquareroot;
VAR
  A: ARRAY [1..100] OF REAL;
  i: INTEGER;
BEGIN
  .....
  FORALL i:= 1 TO 100 DO
    A[i]:= SQRT(A[i]);
  .....
END.
```

Δημιουργία διεργασιών με την εντολή Forall



MPWin

- C02pg027.mpf
- Process Tree

Χαρακτηριστικά Διεργασίας

- Μέγεθος διεργασίας (Process Granularity)
- Χρόνος δημιουργίας της διεργασίας
- Χρόνος ανάθεσης της διεργασίας σε επεξεργαστή

MPWin

- C02pg027a.mpf
- Process Assignment chart
- Process Tree

Εντολή Forall

- Ομαδοποίηση διεργασιών
- Άριστο μέγεθος ομάδας
- Σύνταξη της εντολής Forall:

FORALL <μεταβλητή_μετρητής> := <αρχική_τιμή> TO <τελική_τιμή> {GROUPING <μέγεθος_ομάδας>} DO

    <Σώμα_Εντολής>;

Εντολή Grouping

```
PROGRAM ParallelSquareroot;
VAR
  A: ARRAY [1..100] OF REAL;
  i: INTEGER;
BEGIN
  .....
  FORALL i:= 1 TO 100 GROUPING N DO
    A[i]:= SQRT(A[i]);
  .....
END.
```

MPWin

- C02pg029.mpf
- Process Tree
- Process Activity chart

Αλγόριθμος Ταξινόμησης Σειράς

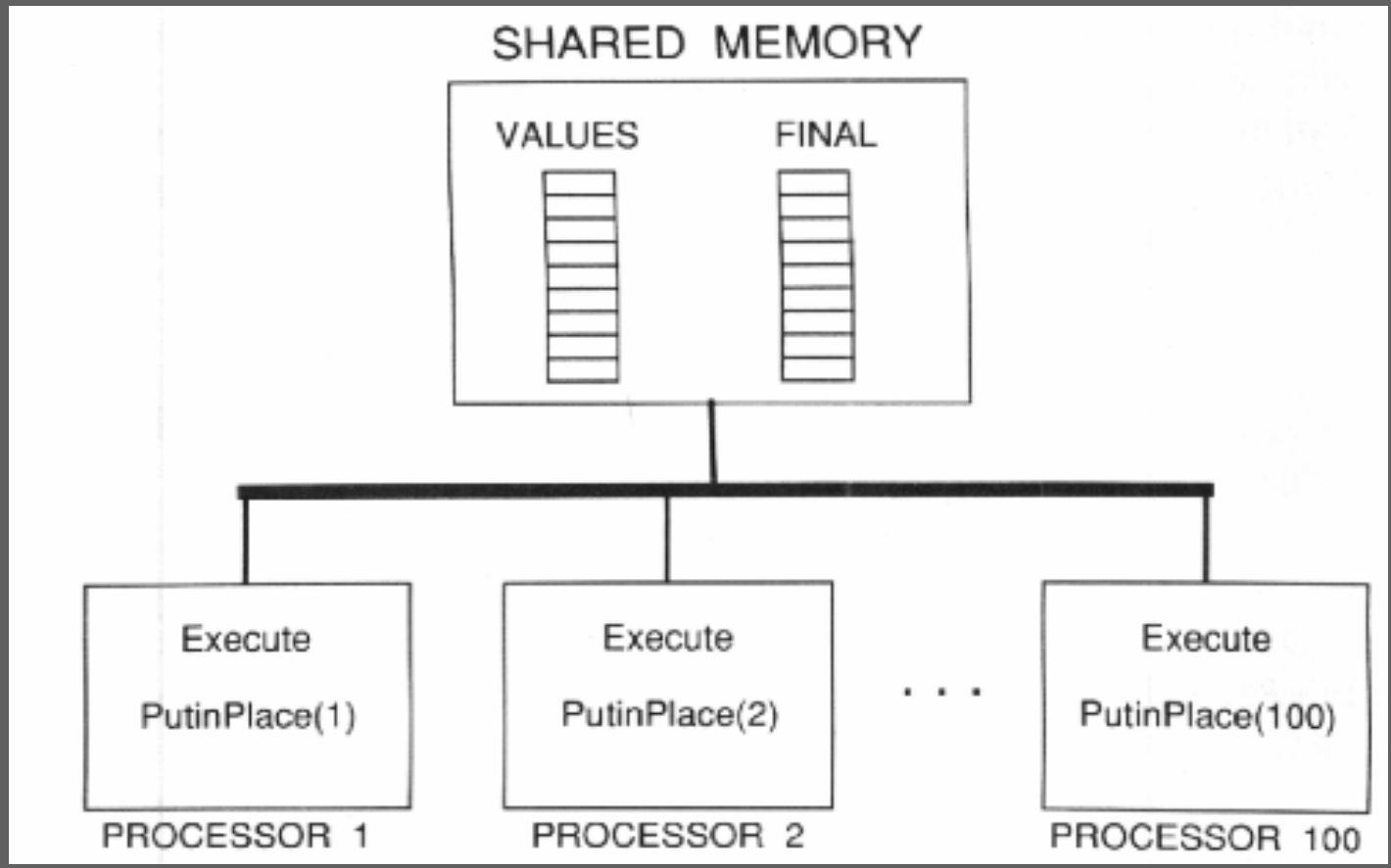
```
RANK_SORT;

FOR i := 1 TO n DO BEGIN
    e := values[i];
    rank := 0;
    FOR   κάθε στοιχείο t του πίνακα
      IF e >= t THEN rank := rank+1;
      τοποθέτησε το e στη θέση rank του
      ταξινομημένου πίνακα
END;
```

```
PARALLEL_RANK_SORT;

FORALL i := 1 TO n DO BEGIN
  e := values[i];
  rank := 0;
  FOR   κάθε στοιχείο t της λίστας
    IF e >= t THEN rank := rank+1;
      τοποθέτησε το e στη θέση rank του
      ταξινομημένου πίνακα
END;
```

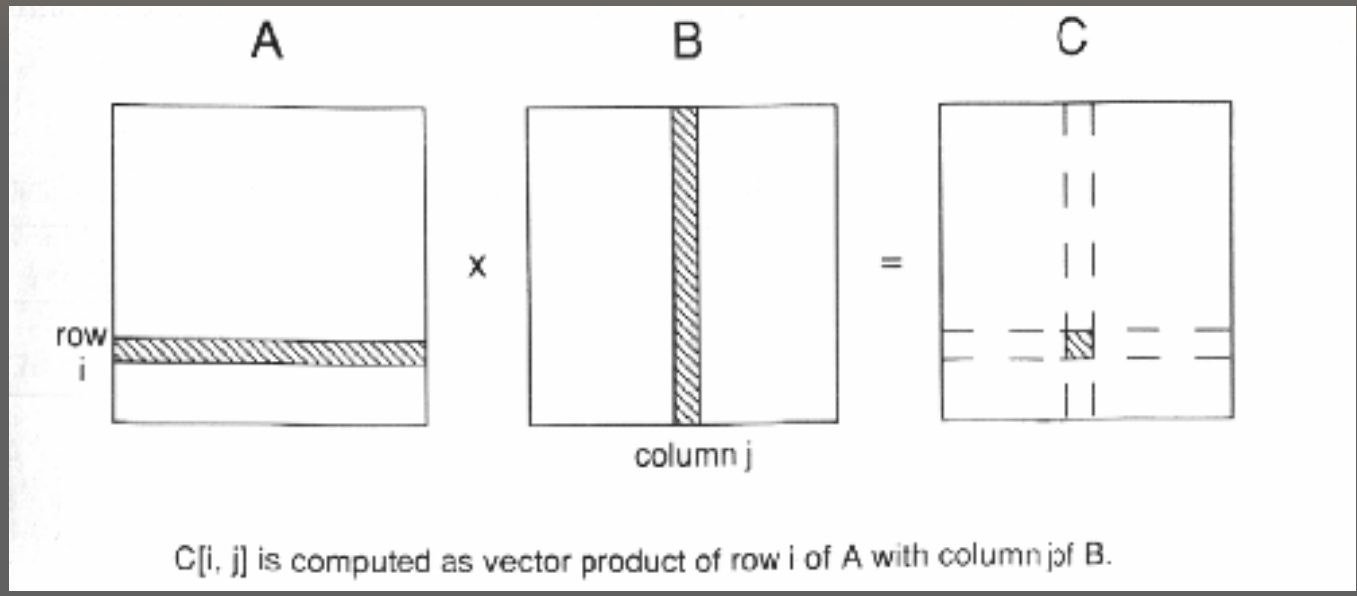
```
1   PROGRAM RankSort;
2   CONST
3     n = 100;
4   VAR
5     values,final : ARRAY [1..n] OF INTEGER;
6     i: INTEGER;

7   PROCEDURE PutinPlace(src : INTEGER);
8   VAR
9     testval,j,rank : INTEGER;
10  BEGIN
11    testval := values[src];
12    j := src;   (* το j θα κινηθεί σειριακά  σε όλο τον πίνακα *)
13    rank := 0;
14    REPEAT
15      j := j MOD n+1;
16      IF testval >= values[j] THEN rank := rank+1;
17    UNTIL  j=src;
18    final[rank]:=testval; (*βάλε την τιμή στην ταξινομημένη της θέση *)
19  END;

20  BEGIN    (* Κυρίως πρόγραμμα *)
21    FOR i := 1 TO n DO
22      READLN(values[i]); (*αρχικοποίηση των τιμών προς ταξινόμηση *)
23    FORALL i := 1 TO n DO
24      PutinPlace(i); (*Εύρεση της σειράς των values[i] και τοποθέτηση τους*)
25  END.
```

ΣΧΗΜΑ 2.3 Παράλληλη Ταξινόμηση Σειράς

Εκτέλεση παράλληλου αλγορίθμου ταξινόμησης σειράς



Πολλαπλασιασμός Μητρών



C[i, j] is computed as vector product of row i of A with column jof B.

```
Πολλαπλασιασμός Μητρών   C = A * B:

  FOR i := 1 TO n DO
    FOR j := 1 TO n DO
      υπολόγισε το C[i,j] σαν το εσωτερικό γινόμενο
      της γραμμής i της A με την στήλη j της B;
```

Σειριακός Αλγόριθμος Πολλαπλασιασμού Μητρών

```
PROGRAM MatrixMultiply;
CONST
  n=10;
VAR
  A, B, C: ARRAY [1..n, 1..n] OF REAL;
  i, j, k: INTEGER;
  sum: REAL;
BEGIN
  ...

  FOR i := 1 TO n DO
    FOR j := 1 TO n DO BEGIN
      sum := 0;
      FOR k := 1 TO n DO
        sum := sum + A[i,k] * B[k,j];
      C[i,j] := sum;
    END;
  ...
END.
```

Μετατροπή Σειριακού Πολλαπλασιασμού Μητρών σε Παράλληλο (Σφάλμα)

```
PROGRAM ErroneousMatrixMultiply;
CONST
  n=100;
VAR
  A, B, C : ARRAY [1..n,1..n] OF REAL;
  i, j, k : INTEGER;
  sum : REAL;
BEGIN
  .....
  FORALL i := 1 TO n DO
    FORALL j := 1 TO n DO BEGIN
      sum := 0;
      FOR k := 1 TO n DO
        sum := sum + A[i, k]*B[k, j];
      C[i, j] := sum;
    END;
  .....
END.
```

Παράλληλος Πολλαπλασιασμός Μητρών

```
PROGRAM ParallelMatrixMultiply;
CONST
  n=10;
VAR
  A, B, C: ARRAY [1..n, 1..n] OF REAL;
  i,j:integer;

PROCEDURE VectorProduct(i, j: INTEGER);
VAR
  sum: REAL;
  k: INTEGER;
BEGIN
  sum := 0;
  FOR k := 1 TO n DO
    sum := sum + A[i,k] * B[k,j];
  C[i,j] := sum;
END;
BEGIN (* Σώμα κυρίως προγράμματος*)
  ...
  FORALL  i := 1 TO n DO
    FORALL j := 1 TO n DO
      VectorProduct(i,j); (*υπολογίζει την γραμμή i της A από τη στήλη j της B*)
  ...

END.
```

Ομάδες Εντολών με Δηλώσεις Μεταβλητών

```
VAR <declaration 1>;
    <declaration 2>;
     ...
    <declaration m>;
BEGIN
    <statement 1>;
    <statement 2>;
     ...
    <statement n>;
END;
```

Ομάδες Εντολών με Δηλώσεις Μεταβλητών

```
PROGRAM Sample;
VAR
  A: ARRAY [1..10,1..10] OF REAL;
  x,p:REAL;
  ...

BEGIN(*Σώμα κυρίως προγράμματος*)
  ...
  x := A[1,4]*3;
  p := Sqrt(x);
  VAR (* Αρχή της ομάδας εντολών*)
    x,y: INTEGER;
    value:REAL;
  BEGIN
    value := 1;
    FOR x := 1 TO 10 DO
      FOR y := 1 TO 10 DO BEGIN
        A[x,y] := value;
        value := value * 2;
      END;
  END; (*Τέλος της ομάδας εντολών*)
  Writeln(x);(* Αναφέρεται στο x που δηλώνεται στην αρχή του Κ.Π.*)
  Writeln(A[1,1]);
  ...
END.
```

Ομάδες Εντολών με Δηλώσεις Μεταβλητών - Παράλληλος Πολλαπλασιασμός Μητρών

```
1   PROGRAM ParallelMatrixultiply_2;
2    CONST
3      n = 10;
4    VAR
5       A, B, C:  ARRAY [1..n, 1..n] OF REAL;
6       i, j: INTEGER;
7    BEGIN
8      ...
9      FORALL  i := 1 TO n DO
10       FORALL  j := 1 TO n DO
11         VAR
12           sum: REAL;
13           k: INTEGER;
14         BEGIN
15           sum := 0;
16           FOR k  :=   1 TO n DO
17             sum := sum+ A[i,k] * B[k, j];
18           C[i, j] := sum;
19         END;
           ...
```

Εμβέλεια των μετρητών της Forall

```
PROGRAM Sample;
VAR
  i: INTEGER;

PROCEDURE Tree;
BEGIN
  ...
END;

BEGIN
  ...
  FORALL i := 1 TO 20 DO BEGIN
    ...
    Tree;
  END;
  ...
END.
```

Σύνταξη της εντολής Fork

```
FORK <γραμμή εντολής>;
```

Παραδείγματα:

```
x := y * 3;
FORK  FOR i := 1 TO 10 DO A[i] := i;
z := SQRT(x);

FORK Normalize(A);
FORK Normalize(B);
FORK Normalize(C);
```

Σύγκριση εντολών Forall και Fork

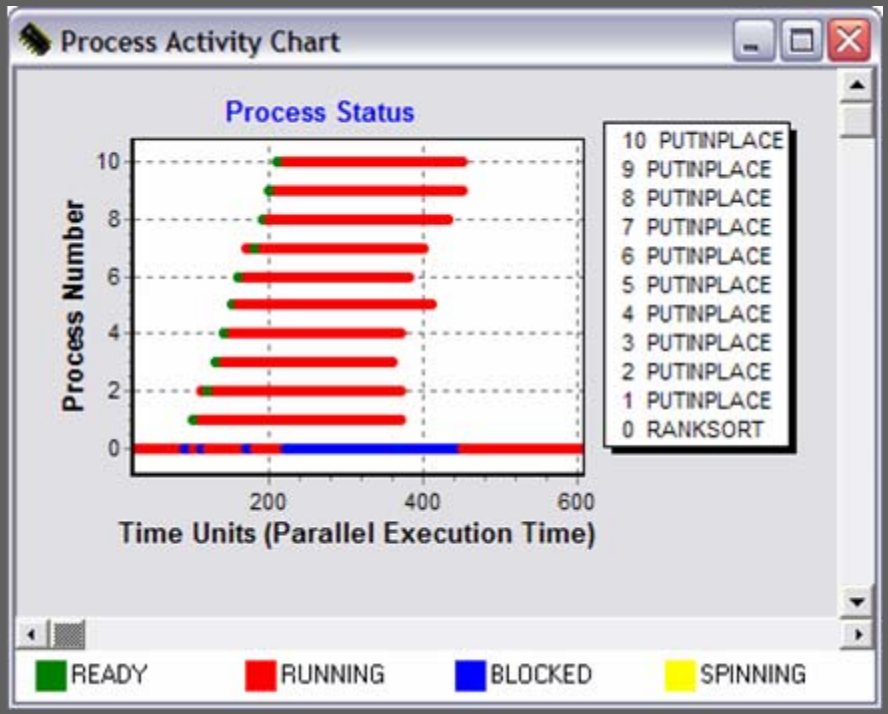
Η δομή FORALL έχει σαν αποτέλεσμα την παρακάτω ακολουθία ενεργειών:

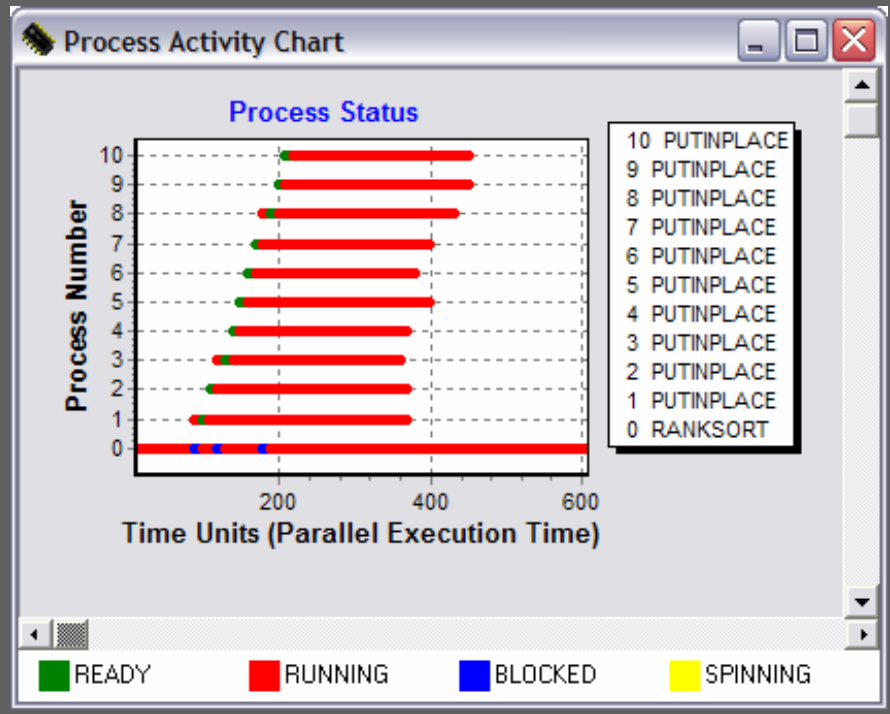
Δημιούργησε n διεργασίες παιδιά από την <γραμμή εντολής>
Περίμενε έως ότου τερματίσουν και οι n διεργασίες παιδιά
Συνέχισε την εκτέλεση του προγράμματος μετά την FORALL

Η δομή FORK έχει σαν αποτέλεσμα την παρακάτω ακολουθία ενεργειών:

Δημιούργησε 1 διεργασία παιδί από την <γραμμή εντολής>
Συνέχισε παράλληλα την εκτέλεση του προγράμματος μετά την FORK

Forall vs Fork





Εντολή Join

```
FOR i := 1 TO 10 DO
   FORK Compute(A[i]);
FOR i := 1 TO 10 DO
   JOIN;
```

Εντολή Fork και Παραλληλισμός Λειτουργιών

```
PROGRAM ParallelListApply;
TYPE
  pnttype = ^elementtype;
  elementtype = RECORD
    data:REAL;
    next: pnttype;
  END;

VAR
  pnt, listhead: pnttype;

PROCEDURE Compute(value: REAL);
BEGIN
  ...
END;

BEGIN (*Κυρίως πρόγραμμα*)
  ...
  pnt := listhead;
  WHILE  pnt <> nil DO BEGIN
    FORK Compute(pnt^.data);  (* Δημιουργεί την διεργασία παιδί*)
    pnt := pnt^.next;  (*Μετακινείται στο επόμενο στοιχείο  της λίστας*)
  END;
  ...
END.
```

Ο νόμος του Amdahl

Μέγιστη Επιτάχυνση= $$\frac{1}{f + \frac{1-f}{p}}$$