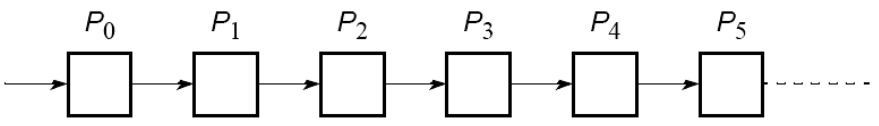
Υπολογισμοί βασισμένοι στη τεχνική της σωλήνωσης (Pipelined Computations)

Σωληνωμένοι (Pipelined) Υπολογισμοί

• Η τεχνική της σωλήνωσης εφαρμόζεται σε ένα μεγάλο εύρος προβλημάτων τα οποία είναι εν μέρει ακολουθιακά από τη φύση τους, δηλ. πρέπει να εκτελεστεί μία ακολουθία βημάτων.

• Πιο συγκεκριμένα, το πρόβλημα διαιρείται σε μία σειρά από έργα τα οποία θα πρέπει να ολοκληρωθούν το ένα μετά το άλλο. Κάθε έργο εκτελείται από μία διαφορετική διεργασία/επεξεργαστή:



• Για παράδειγμα, ο υπολογισμός του αθροίσματος των στοιχείων ενός πίνακα μπορεί να υλοποιηθεί με τη βοήθεια της τεχνικής της σωλήνωσης:

```
for (i = 0; i < n; i++)
sum = sum + a[i];
```

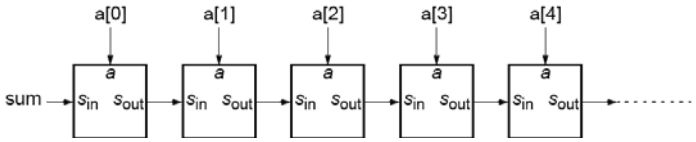
Ο βρόχος μπορεί να «ξεδιπλωθεί» ως εξής:

```
sum = sum + a[0];
sum = sum + a[1];
sum = sum + a[2];
sum = sum + a[3];
sum = sum + a[4];
```

....

.

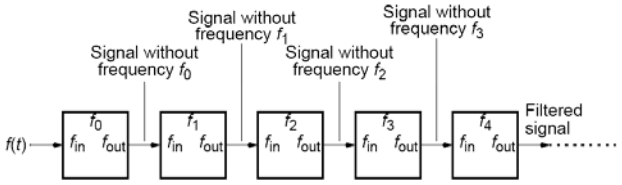
- Ο υπολογισμός αυτός μπορεί εύκολα να υλοποιηθεί με τη τεχνική της σωλήνωσης



•Στη λύση αυτή, κάθε στάδιο αντιστοιχεί σε μία ξεχωριστή εντολή του ξεδιπλωμένου βρόχου

• Κάθε στάδιο δέχεται ως είσοδο το συσσωρευμένο άθροισμα από τα προηγούμενα στάδια (είσοδος s_{in}) και το αντίστοιχο στοιχείο του πίνακα a και παράγει το νέο συσσωρευμένο άθροισμα στην έξοδο s_{out}. Δηλ. το στάδιο i εκτελεί τον υπολογισμό $s_{out} = s_{in} + a[i]$

• Εκτός από απλές εντολές, μία ακολουθία από συναρτήσεις μπορούν να εκτελεστούν με την τεχνική της σωλήνωσης. Ένα φίλτρο συχνοτήτων είναι ένα τέτοιο παράδειγμα. Στο συγκεκριμένο παράδειγμα ο στόχος είναι να αφαιρεθούν συγκεκριμένες συχνότητες (f_0, f_1, f_2 ,f_3, κτλ.) από ένα ψηφιακό σήμα $f(t)$. Το σήμα εισέρχεται στη σωλήνωση από αριστερά:



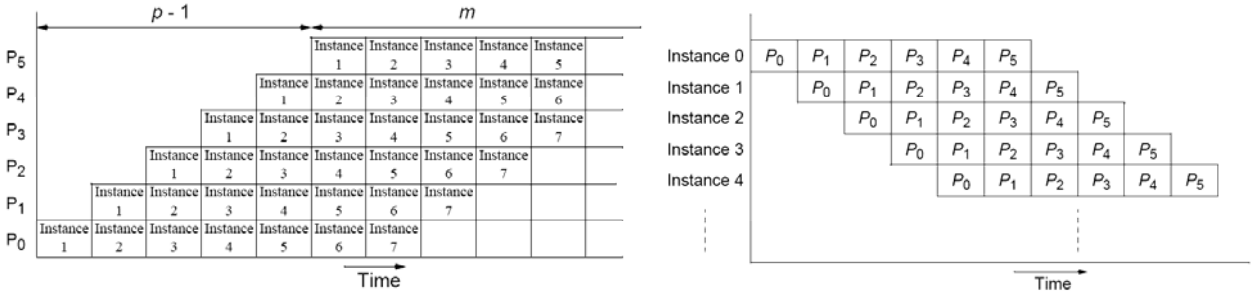
•Σε κάθε στάδιο εφαρμόζεται η συνάρτηση που αφαιρεί μία συχνότητα από το ψηφιακό σήμα

Πότε η τεχνική της σωλήνωσης έχει επιτυχία

Υποθέτοντας ότι το υπό εξέταση πρόβλημα μπορεί να διαιρεθεί σε μία ακολουθία από έργα, η τεχνική της σωλήνωσης μπορεί να δώσει χαμηλότερους χρόνους εκτέλεσης σε τρεις περιπτώσεις:

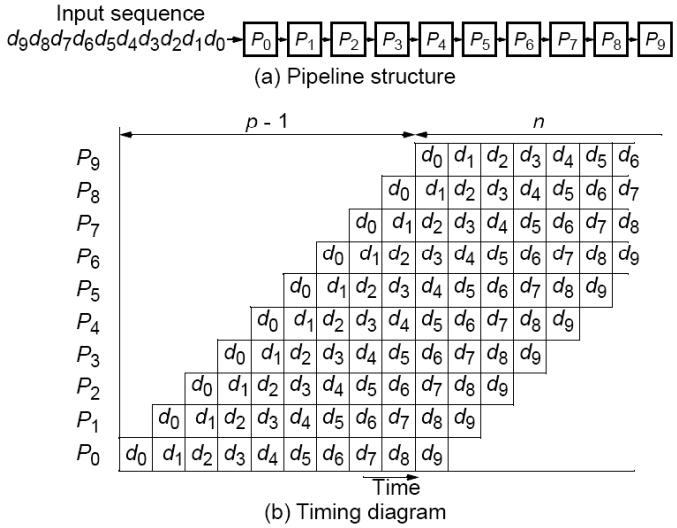
1. Αν υπάρχουν περισσότερα από ένα στιγμιότυπα του ίδιου προβλήματος που πρέπει να επιλυθούν (Type 1 pipeline)
2. Αν πρέπει να γίνει επεξεργασία μίας ακολουθίας δεδομένων και η επεξεργασία κάθε δεδομένου είναι σύνθετη απαιτώντας μία σειρά από λειτουργίες (Type 2 pipeline)
3. Αν κάθε στάδιο i της σωλήνωσης μπορεί να δώσει γρήγορα τμήμα του αποτελέσματος του στο επόμενο στάδιο πριν να ολοκληρωθεί πλήρως η επεξεργασία στο στάδιο i. Το επόμενο στάδιο μπορεί να αρχίσει την επεξεργασία του βασιζόμενο μόνο στην αρχική πληροφορία που στέλνει το στάδιο i. Στη πορεία, λαμβάνει και το υπόλοιπο αποτέλεσμα του σταδίου i (Type 3 pipeline)

“Type 1” Pipeline - Χρονοδιάγραμμα



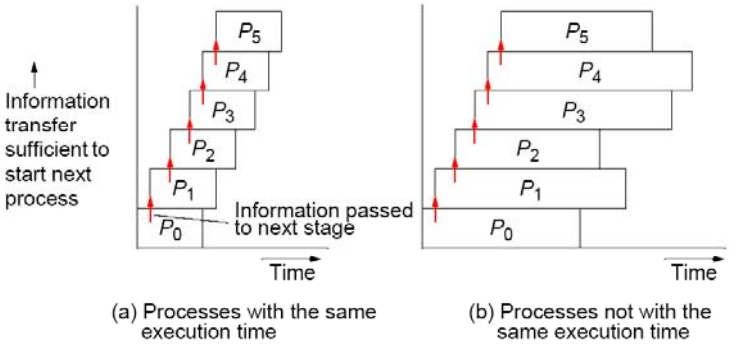
• Ο τύπος 1 της τεχνικής της σωλήνωσης χρησιμοποιείται ευρέως για το σχεδιασμό των επεξεργαστών στο επίπεδο του υλικού.
• Υπάρχουν πολλαπλά στιγμιότυπα του ίδιου προβλήματος και για την επίλυση ενός στιγμιότυπου απαιτούνται p σταδια επεξεργασίας. Κάθε στάδιο εκτελείται από μία διαφορετική διεργασία
• Όταν η διεργασία ολοκληρώσει την επεξεργασία της για ένα στιγμιότυπο του προβλήματος, λαμβάνει από τη προηγούμενη διεργασία τα ενδιάμεσα αποτελέσματα για το επόμενο στιγμιότυπο και στη συνέχεια εκτελεί το στάδιο που έχει αναλάβει για το νέο στιγμιότυπο.
• Ο συνολικός χρόνος για την επίλυση των m στιγμιότυπων θα είναι m+p-1 βήματα

“Type 2” Pipeline- Χρονοδιάγραμμα

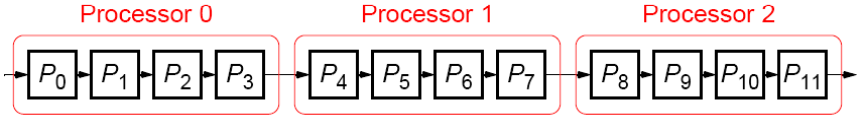


• Στο τύπο 2 της τεχνικής pipeline, υπάρχει μία ακολουθία δεδομένων στα οποία πρέπει να γίνει επεξεργασία διαδοχικά.
• Σε κάθε δεδομένο αυτής της ακολουθίας, υπάρχουν πολλαπλά στάδια επεξεργασίας που θα πρέπει να εκτελεστούν και τα στάδια αυτά τα αναλαμβάνουν ισάριθμες διεργασίες
• Ο συνολικός χρόνος για την επεξεργασία n δεδομένων είναι n+p-1 βήματα.

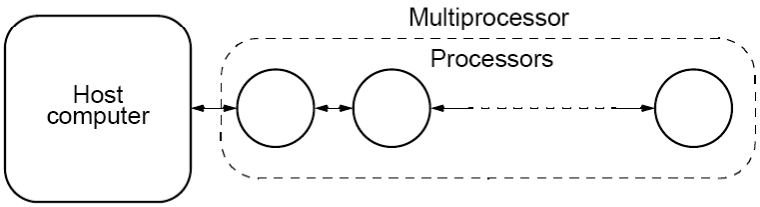
“Type 3” Pipeline - Χρονοδιάγραμμα



• Ο τύπος 3 της τεχνικής της σωλήνωσης είναι ο τύπος που χρησιμοποιείται στα παράλληλα προγράμματα. Πριν τελειώσει η επεξεργασία της, κάθε διεργασία i περνάει πληροφορία στην επόμενη διεργασία i+1. Η υπόλοιπη πληροφορία περνάει σταδιακά στην διεργασία i+1 καθώς παράγεται από τη διεργασία i.
• Αν το πλήθος των σταδίων σε μία σωλήνωση είναι μεγαλύτερος από το πλήθος των διαθέσιμων διεργασιών/επεξεργαστών, κάθε διεργασία μπορεί να αναλάβει μία ομάδα διαδοχικών σταδίων



Υπολογιστικές πλατφόρμες κατάλληλες για την τεχνική της σωλήνωσης

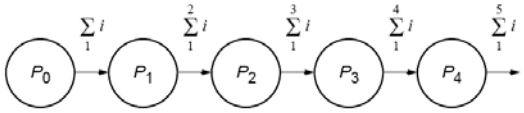


• Το ιδανικό διασυνδετικό δίκτυο για τη τεχνική της σωλήνωσης είναι η γραμμή ή ο δακτύλιος.
• Τέτοιου είδους δίκτυα μπορούν εύκολα να προσομοιωθούν από mesh δίκτυα και τα δίκτυα υπερκύβου.
• Παρά τις περιορισμένες δυνατότητες επικοινωνίας που έχει μία γραμμική διάταξη, πολλές εφαρμογές μπορούν να υλοποιηθούν σε τέτοιες διατάξεις με χαμηλό κόστος.
• Σε μία αρχιτεκτονική cluster, η τεχνική της σωλήνωσης μπορεί να εφαρμοσθεί επιτυχώς μόνο αν το διασυνδετικό δίκτυο των υπολογιστών επιτρέπει ταυτόχρονες μεταφορές μεταξύ επεξεργαστών

Παραδείγματα λύσεων με βάση την τεχνική της σωλήνωσης
(Παραδείγματα για κάθε τύπο υπολογισμού)

Άθροιση αριθμών

• Γίνεται εφαρμογή της τεχνικής σωλήνωσης τύπου 1



• Ο κώδικας που εκτελείται σε όλες τις διεργασίες P_i εκτός από τις διεργασίες P_0 και P_1 είναι ο ακόλουθος:

recv(&accumulation, P_{i-1});
accumulation = accumulation + number;
send(&accumulation, P_{i+1});

όπου accumulation είναι το μερικό άθροισμα όπως έχει σχηματιστεί από τις προηγούμενες διεργασίες και number είναι το στοιχείο a[i] που θα προστεθεί στο μερικό άθροισμα accumulation.

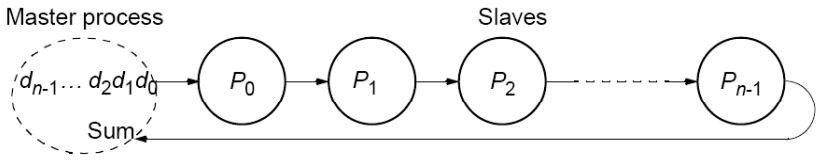
• Η διεργασία P_0 εκτελεί τον εξής κώδικα: **send(&number, P1)**

• Επίσης η διεργασία P_{n-1} εκτελεί τις εξής εντολές:

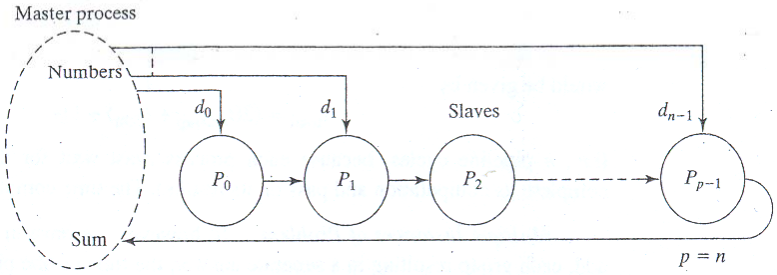
recv(&number, Pn-2);
accumulation = accumulation + number;

Συνολικά το SPMD πρόγραμμα θα έχει ως εξής:

```
if (process > 0) {
        recv(&accumulation, Pi-1);
        accumulation = accumulation + number;
}
if (process < n-1)
        send(&accumulation, P i+1);
```



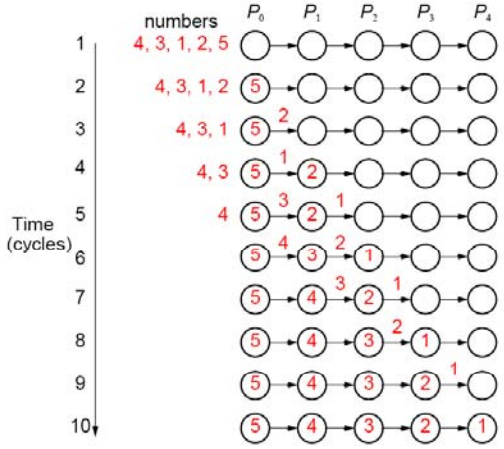
• Υπάρχουν δύο παραλλαγές στη βασική τεχνική όσον αφορά το τρόπο με τον οποίο εισάγονται οι αριθμοί στη διάταξη σωλήνωσης.
• Στη πρώτη περίπτωση η master και οι slave διεργασίες είναι σε διάταξη δακτυλίου και τα δεδομένα εισόδου εισάγονται από τη master διεργασία μέσω της διαδικασίας P_0 σε όλες τις υπόλοιπες διεργασίες.
• Εναλλακτικά η master διεργασία δίνει κατευθείαν τα δεδομένα στις slave διεργασίες όταν αυτές τα χρειάζονται δηλ. στο πρώτο κύκλο δίνει το στοιχείο d_0 στη P_0 στο δεύτερο κύκλο δίνει το στοιχείο d_1 στη P_1 κοκ.

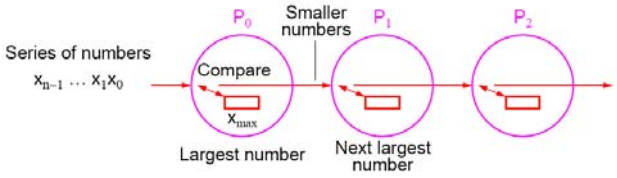


• Στη δεύτερη περίπτωση ο συνολικός χρόνος για το υπολογισμό του αθροίσματος n στοιχείων θα είναι $t_{total} = (t_{comp} + t_{comm})n$ όπου $t_{comp}(= 1)$ είναι το κόστος της άθροισης σε κάθε διεργασία και $t_{comm} = (t_{startup} + t_{data})$ είναι το κόστος επικοινωνίας μεταξύ γειτονικών διεργασιών.
• Αν υπάρχουν m στιγμιότυπα του προβλήματος της άθροισης των n αριθμών που πρέπει να επιλυθούν, η συνολική πολυπλοκότητα του θα είναι $t_{total} = (t_{comp} + t_{comm})(m+n-1)$. Ο μέσος χρόνος για την ολοκλήρωση ενός στιγμιότυπου θα είναι $t_a = t_{total}/m$. Για μεγάλη τιμή του m ο χρόνος αυτός θα είναι περίπου ίσος με $t_{comp} + t_{comm}$

Ταξινόμηση – Σωλήνωση Τύπου 2

• Ο αλγόριθμος ταξινόμησης παρεμβολής (insertion sort) είναι ένας γνωστός αλγόριθμος ταξινόμησης. Μπορεί να υλοποιηθεί παράλληλα με τη βοήθεια της τεχνικής της σωλήνωσης.
• Συγκεκριμένα, η διεργασία P_0 δέχεται διαδοχικά τα στοιχεία που θα ταξινομηθούν, αποθηκεύει το μεγαλύτερο αριθμό που έχει δει μέχρι τώρα και περνάει στην επόμενη διεργασία όλους τους υπόλοιπους αριθμούς. Αν ένα νέο στοιχείο ληφθεί που είναι μεγαλύτερο από αυτό που έχει αποθηκευθεί, το ήδη αποθηκευμένο στοιχείο στέλνεται στην επόμενη διεργασία και το νέο στοιχείο παίρνει τη θέση του.
• Οι υπόλοιπες διεργασίες εκτελούν την ίδια διαδικασία δηλ. κρατούν πάντα το μεγαλύτερο μέχρι εκείνη τη στιγμή στοιχείο και περνούν τους υπόλοιπους αριθμούς στην επόμενη διεργασία

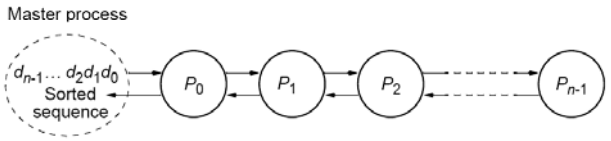




Ο βασικός αλγόριθμος για τη διεργασία *Pi* είναι

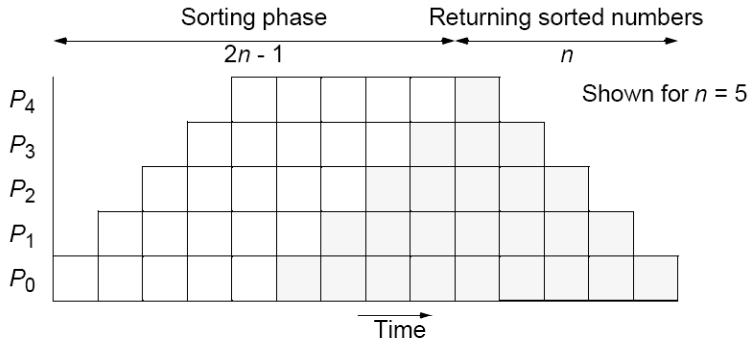
```
right_procNum = n – i -1;
recv(&x, P_{i-1});
for (j=0; j< right_procNum; j++) {
  recv(&number, P_{i-1} );
  if (number > x) {
            send(&x, P_{i+1});
            x = number;
  } else send(&number, P_{i+1});
}
/* τα στοιχεία του ταξινομημένου πίνακα ολισθαίνουν προς τα
«αριστερά» και συλλέγονται από τη Master διεργασία */

send(&x, P_{i-1})
for (j=0; j < right_procNum; j++) {
  recv(&number, P_{i+1});
  send(&number, P_{i-1});
}
```

Κάθε διεργασία γνωρίζει ότι θα λάβει n-i αριθμούς. Επίσης γνωρίζει ότι τα περάσει n – i - 1 στοιχεία στην επόμενη διεργασία αφού ένα στοιχείο θα αποθηκευθεί στη διεργασία



Συνολικά τα στοιχεία κινούνται προς τα δεξιά κατά τη φάση της ταξινόμησης και προς τα αριστερά όταν συλλέγεται ο ταξινομημένος πίνακας από τη Master διεργασία

Χρονοδιάγραμμα της ταξινόμησης παρεμβολής

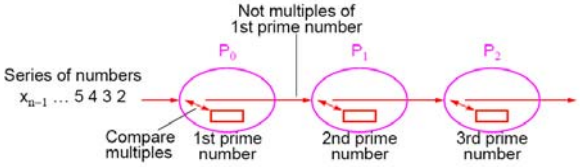


Ο χρόνος εκτέλεσης της φάσης ταξινόμησης θα είναι $(2n -1)*(T_{comp} + T_{comm})$ ενώ ο χρόνος για τη συλλογή του ταξινομημένου πίνακα από τη master διεργασία είναι $n*T_{comm}$.

Παράγωγη Πρώτων Αριθμών

Κόσκινο του Ερατοσθένη – Τύπος 2 σωλήνωσης

• Με τον αλγόριθμο αυτό βρίσκουμε όλους τους πρώτους αριθμούς που είναι μικρότεροι από ένα αριθμό n.
• Ο ακολουθιακός αλγόριθμος διαγράφει πρώτα όλα τα πολλαπλάσια του 2 που είναι μικρότερα του n. Στη συνέχεια, διαγράφει όλα τα πολλαπλάσια του μικρότερου των στοιχείων που έχουν απομείνει από το πρώτο γύρο. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να απομείνουν μόνο οι πρώτοι αριθμοί μικρότεροι του n.
• Η διαδικασία αυτή μπορεί να υλοποιηθεί παράλληλα με τη βοήθεια της τεχνικής σωλήνωσης



Ο κώδικας για την i-οστή διεργασία έχει ως εξής:

```
recv(&x, P_{i-1});
for (i = 0; i < n; i++) {
        recv(&number, P_{i-1});
         if (number % x) != 0 or number == terminator) send(&number, P_{i+1});
        If (number == terminator) break;
        }
```

• Στο τέλος η διεργασία i θα αποθηκεύσει τον i-στο μικρότερο πρώτο αριθμό.
• Επίσης κάθε διεργασία δεν θα λάβει το ίδιο πλήθος δεδομένων και επιπλέον η ποσότητα αυτή δεν είναι γνωστή εκ των προτέρων. Για αυτό, χρησιμοποιείται και ένα μήνυμα “terminator” που στέλνεται στο τέλος της ακολουθίας των αριθμών. Κάθε διαδικασία που λαμβάνει αυτό το μήνυμα, γνωρίζει ότι δεν θα λάβει άλλα δεδομένα και αφού στείλει το μήνυμα στην επόμενη διεργασία στη συνέχεια τερματίζει.
• Η ανάλυση πολυπλοκότητας είναι παρόμοια με αυτή του αλγόριθμου ταξινόμησης με τη διαφορά ότι τώρα κάθε διεργασία εκτελεί λιγότερα βήματα από ότι η προηγούμενη διεργασία επειδή δεν θα λάβει όλους τους αριθμούς που θα λάβει η προηγούμενη διεργασία

Επίλυση συστήματος Γραμμικών εξισώσεων

Άνω Τριγωνική μορφή – Τύπος 3 σωλήνωσης

$$a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 \quad \ldots \quad + a_{n-1,n-1}x_{n-1} = b_{n-1}$$

$$\vdots$$

$$a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 = b_2$$

$$a_{1,0}x_0 + a_{1,1}x_1 = b_1$$

$$a_{0,0}x_0 = b_0$$

Στόχος είναι η επίλυση ενός συστήματος γραμμικών εξισώσεων που είναι σε άνω τριγωνική μορφή όπου τα στοιχεία a_{ij} και b_i είναι σταθερές και x_i είναι οι άγνωστές μεταβλητές που πρέπει να προσδιορισθούν

Back Substitution

• Για την επίλυση του γραμμικού συστήματος χρησιμοποιούμε την προς τα πίσω αντικατάσταση.
• Πρώτα , ο άγνωστος x_0 βρίσκεται κατευθείαν από την τελευταία εξίσωση δηλ.

$$x_0 = \frac{b_0}{a_{0,0}}$$

Στη συνέχεια γίνεται αντικατάσταση της τιμής του x_0 στην επόμενη εξίσωση και έτσι λαμβάνεται η μεταβλητή x_1 δηλ.,

$$x_1 = \frac{b_1 - a_{1,0}x_0}{a_{1,1}}$$

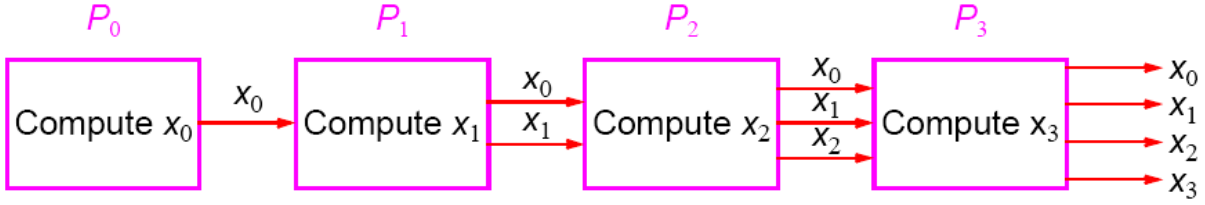
Έπειτα, οι τιμές για τους αγνώστους x_1 and x_0 αντικαθίσταται στη επόμενη εξίσωση για να λάβουμε τη τιμή για τη μεταβλητή x_2:

$$x_2 = \frac{b_2 - a_{2,0}x_0 - a_{2,1}x_1}{a_{2,2}}$$

Η διαδικασία αυτή επαναλαμβάνεται μέχρι να προσδιορισθούν οι τιμές για όλους τους αγνώστους

Η λύση με την τεχνική σωλήνωσης

•Η επίλυση ενός γραμμικού άνω τριγωνικού συστήματος μπορεί να υλοποιηθεί παράλληλα με την τεχνική της σωλήνωσης.
•Το πρώτο στάδιο της σωλήνωσης (διεργασία P_0) υπολογίζει τον άγνωστο x_0 και περνάει τη τιμή x_0 στο δεύτερο στάδιο (διεργασία P_1), το οποίο υπολογίζει τη τιμή της μεταβλητής x_1 από τη x_0 και περνάει και τις δύο τιμές x_0 και x_1 στο επόμενο στάδιο (διεργασία P_2), το οποίο υπολογίζει τη τιμή της x_2 από την x_0 και x_1, κοκ.



• Γενικά, η *i-οστή* διεργασία ($0 < i < n$) λαμβάνει τις τιμές x_0, x_1, x_2, …, x_{i-1} και υπολογίζει το x_i από την εξίσωση:

$$x_i = \frac{b_i - \sum_{j=0}^{i-1} a_{i,j}x_j}{a_{i,i}}$$

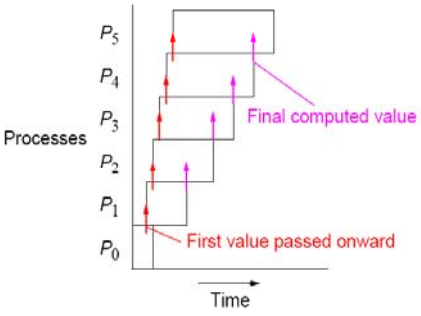
Ο ακολουθιακός κώδικας για τη συγκεκριμένη μέθοδο έχει ως εξής:

```
x[0] = b[0]/a[0][0];            /* computed separately */
for (i = 1; i < n; i++) {       /*for remaining unknowns*/
        sum = 0;
        For (j = 0; j < i; j++)
                sum = sum + a[i][j]*x[j];
        x[i] = (b[i] - sum)/a[i][i];
}
```

Στη παράλληλη υλοποίηση της ίδιας μεθόδου, ο κώδικας της διεργασίας P_i έχει ως εξής:

```
sum=0;
for (j = 0; j < i; j++) {
        recv(&x[j], P_{i-1});
        send(&x[j], P_{i+1});
         sum = sum + a[i][j]*x[j];
}
x[i] = (b[i] - sum)/a[i][i];
send(&x[i], P_{i+1});
```

• Η διεργασία i λαμβάνει τις τιμές των στοιχείων x[0], x[1], x[2],..., x[i-1] διαδοχικά. Κάθε φορά που λαμβάνει μία τιμή x[j] υπολογίζει το γινόμενο με τη τιμή a[i][j] και συσσωρεύει το αποτέλεσμα στη μεταβλητή sum. Μετά τη λήψη των τιμών όλων των μεταβλητών μπορεί να υπολογίσει την τιμή της μεταβλητής x[i].



Χρόνος

P0	P1	P2	P3	P4
divide				
send(x0) end	recv(x0) send(x0) mult/add div/sub send(x1) end	recv(x0) send(x0) mult/add recv(x1) send(x1) mult/add div/sub send(x2) end	recv(x0) send(x0) mult/add recv(x1) send(x1) mult/add recv(x2) send(x2) mult/add div/sub send(x3) end	recv(x0) send(x0) mult/add recv(x1) send(x1) mult/add recv(x2) send(x2) mult/add recv(x3) send(x3) div/sub send(x4) end

• Παρατηρείστε ότι κάθε διεργασία προωθεί αποτελέσματα στις επόμενες διεργασίες πριν να ολοκληρωθεί πλήρως η τοπική επεξεργασία
• Η πρώτη διεργασία P_0 εκτελεί μία διαίρεση και ένα send
• Η i-οστή διεργασία (0< i < p-1) εκτελεί i recv(), i send(), i πολλαπλασιασμούς /προσθέσεις, μία διαίρεση/αφαίρεση και ένα τελικό send, συνολικά (2i+1) βήματα επικοινωνίας και (2i+2) βήματα υπολογισμού.
• Η διαδικασία P_{p-1} εκτελεί p-1 βήματα επικοινωνίας και 2p-1 βήματα υπολογισμού.
• Στο δεξιό σχήμα αναπαρίσταται η εκτέλεση των εντολών στις διάφορες διεργασίες. Γίνεται η υπόθεση ότι όλες οι λειτουργίες (υπολογισμού και επικοινωνίας) απαιτούν το ίδιο χρόνο εκτέλεσης.
• Ο συνολικός χρόνος εκτέλεσης δίνεται από το χρόνο εκτέλεσης της τελευταίας διεργασίας συν το χρόνο εκτέλεσης p-1 send συν το χρόνο μίας διαίρεσης