Pipeline Computation

Thoai Nam

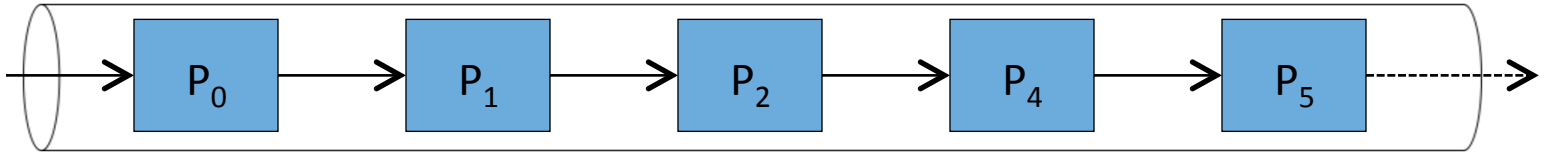
High Performance Computing Lab (HPC Lab)

Faculty of Computer Science and Technology

HCMC University of Technology

Pipeline computation

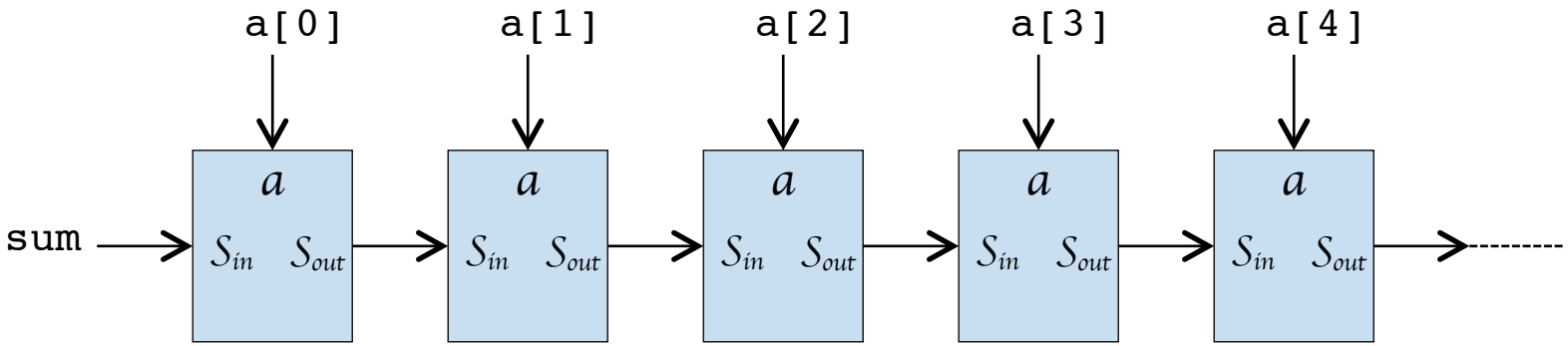
- Problem divided into a series of tasks that have to be completed one after the other (the basis of sequential programming)
- Each task executed by a separate process or processor



Example

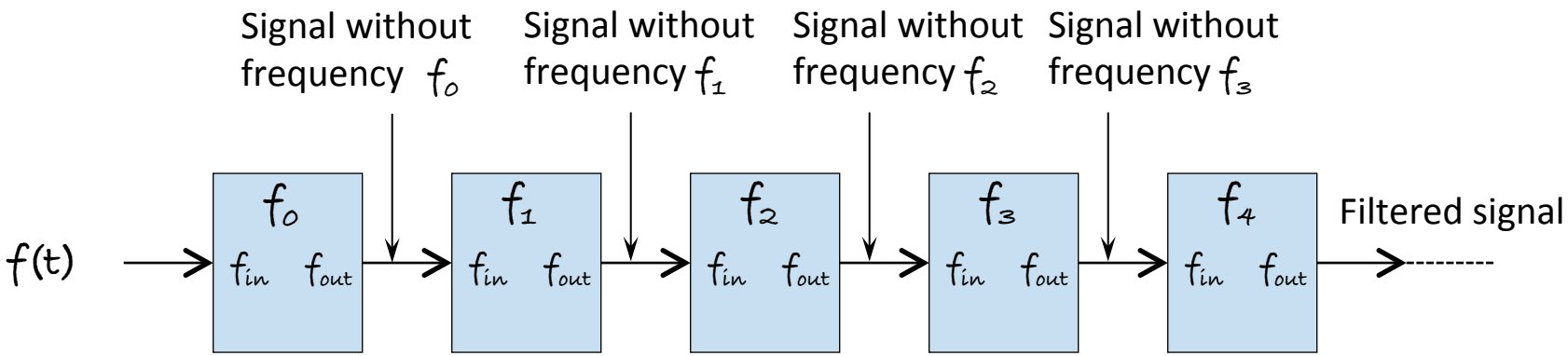
```
for (i=0; i<n; i++)
    sum = sum + a[i];
```

```
sum = sum + a[0];
sum = sum + a[1];
sum = sum + a[2];
...
sum = sum + a[n-1];
```



Pipeline computation

- Frequency filter - Objective to remove specific frequencies (f_0, f_1, f_2, f_3, etc.) from a digitized signal, $f(t)$
- Signal enters pipeline from left:



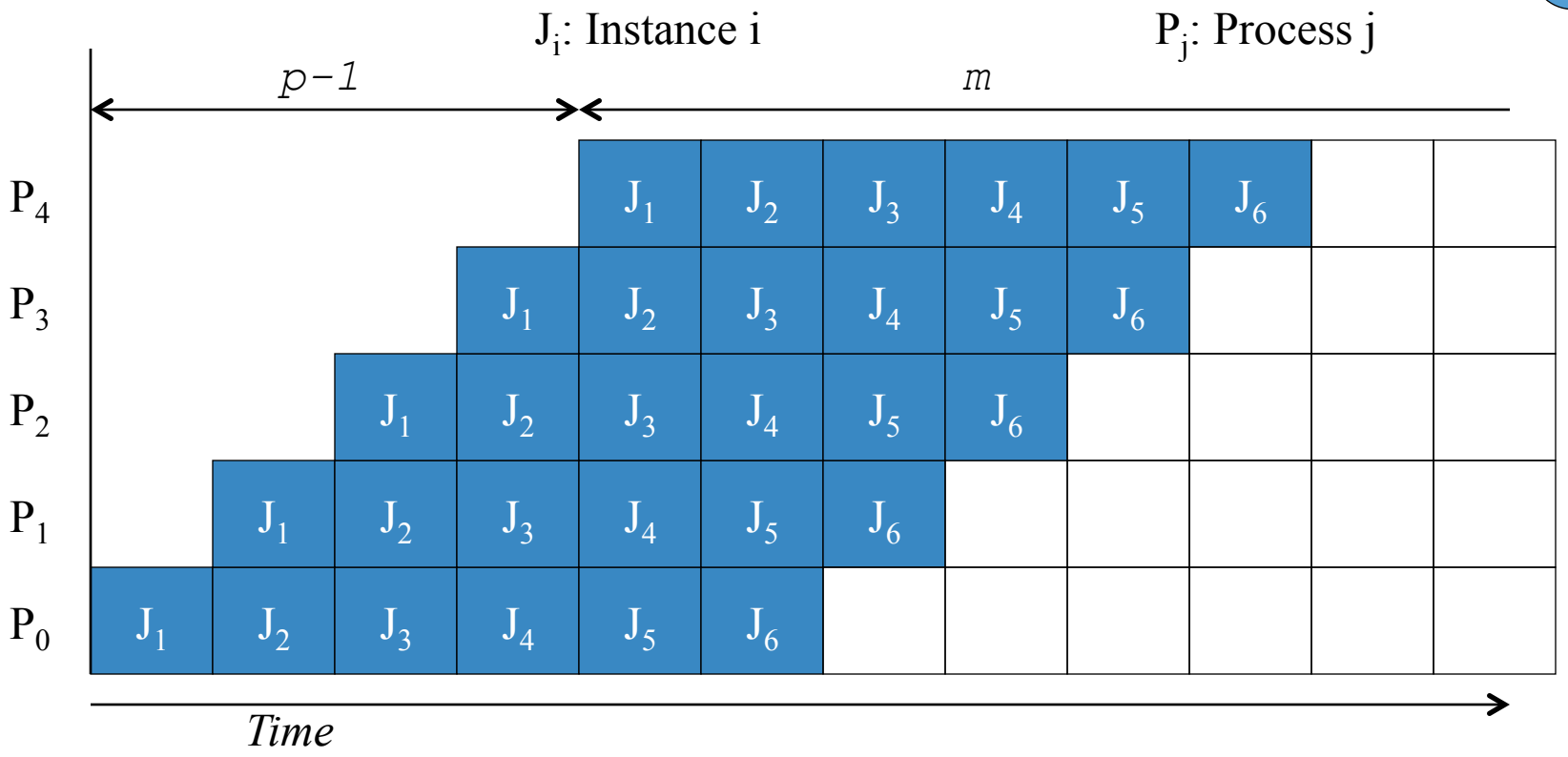
Where pipelining can be used to good effect

Assuming problem can be divided into a series of sequential tasks, pipelined approach can provide increased execution speed under the following three types of computations:

1. If more than one instance of the complete problem is to be executed
2. If a series of data items must be processed, each requiring multiple operations
3. If information to start the next process can be passed forward before the process has completed all its internal operations

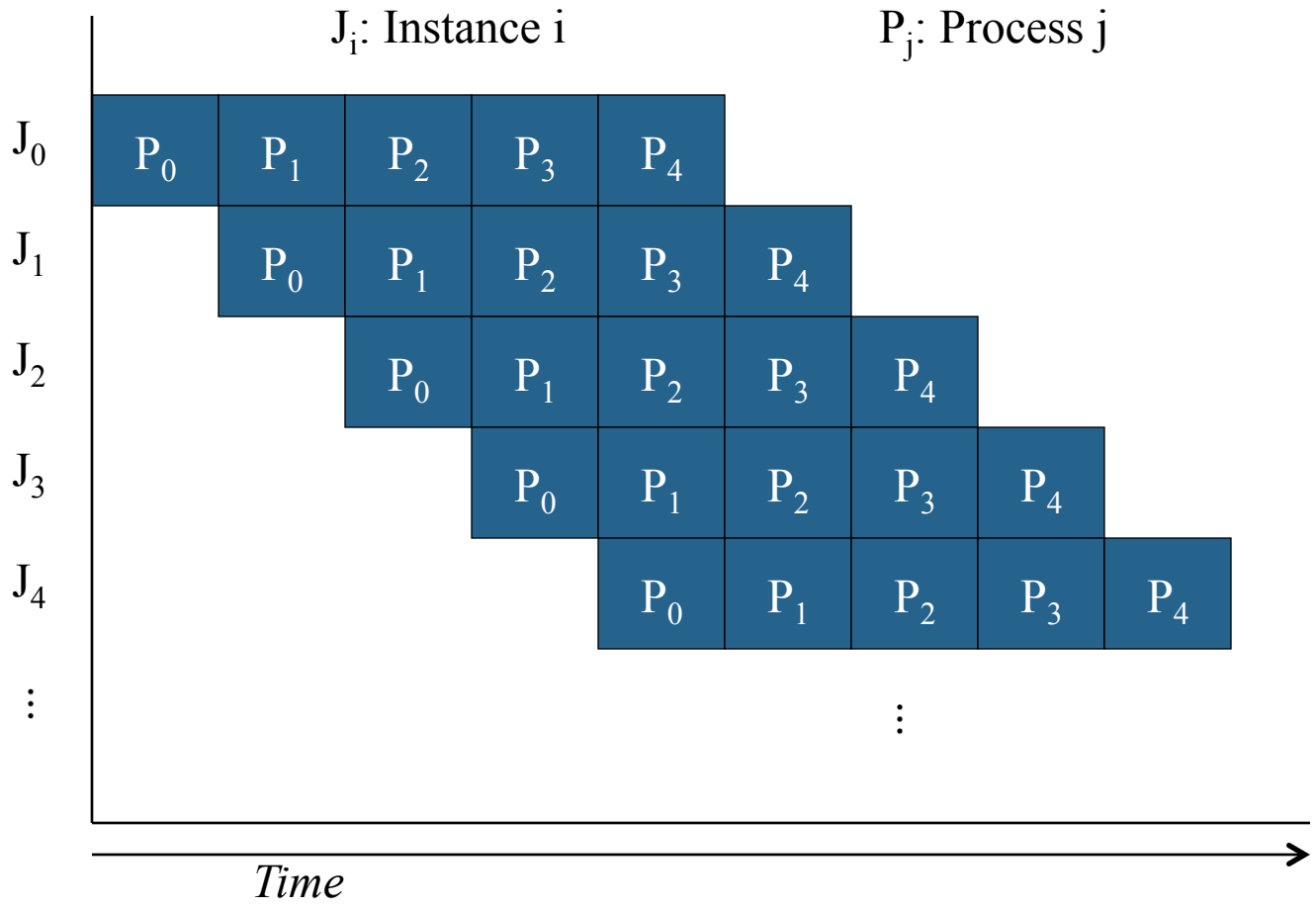
“Type 1” Pipeline Space-Time Diagram

More than one instance of the complete problem is to be executed



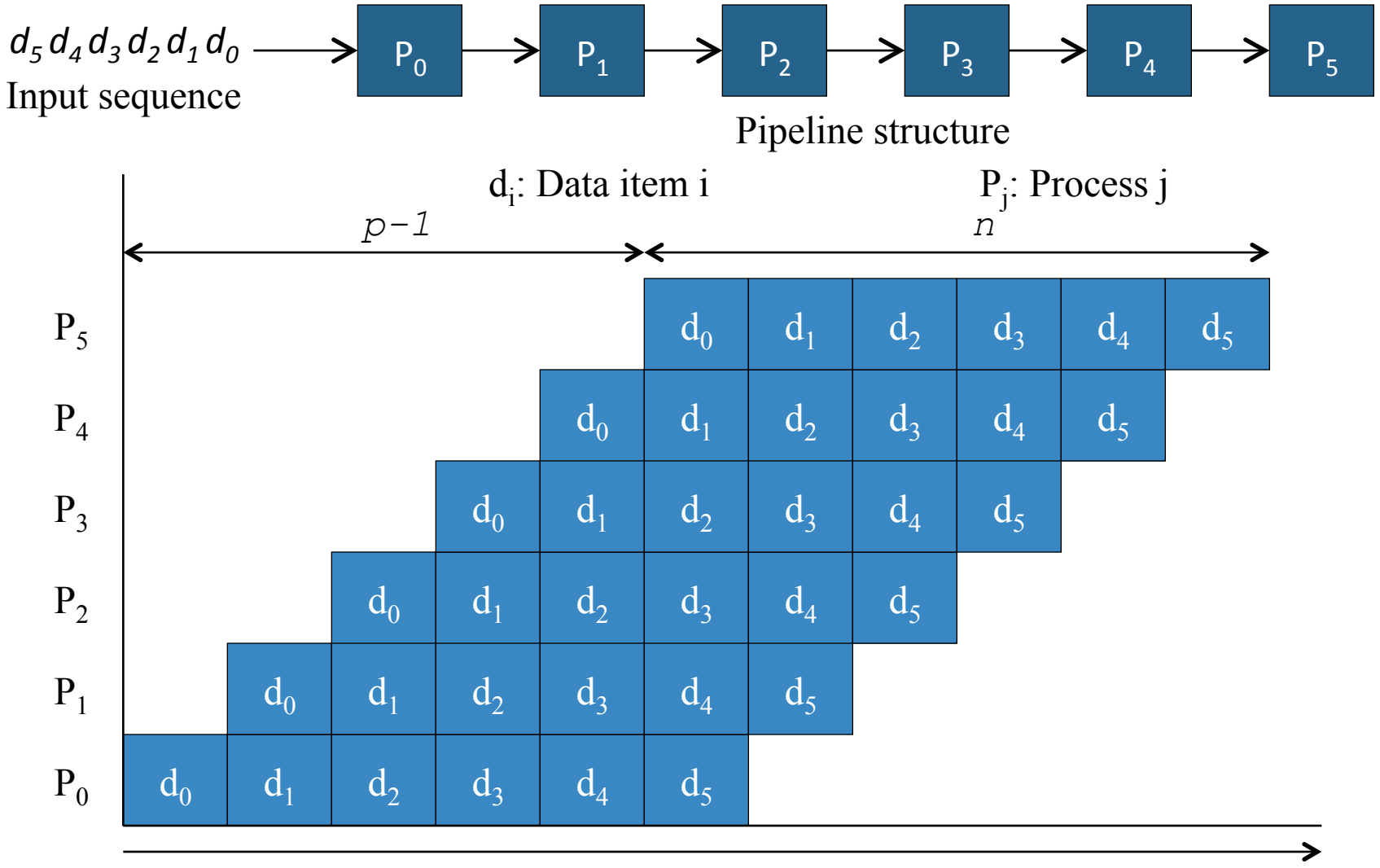
Alternative space-time diagram

More than one instance of the complete problem is to be executed



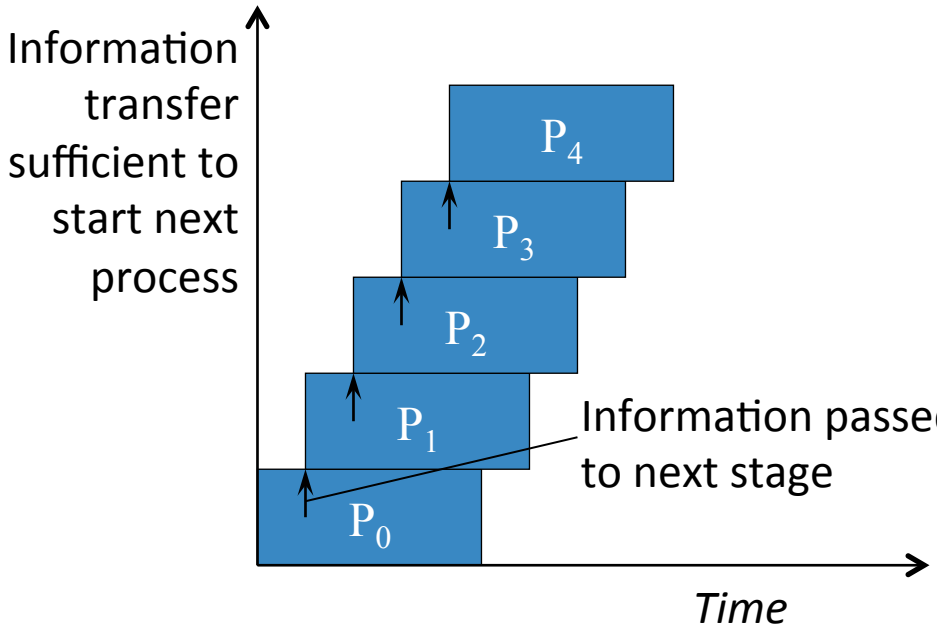
“Type 2” Pipeline Space-Time Diagram

A series of data items must be processed, each requiring multiple operations

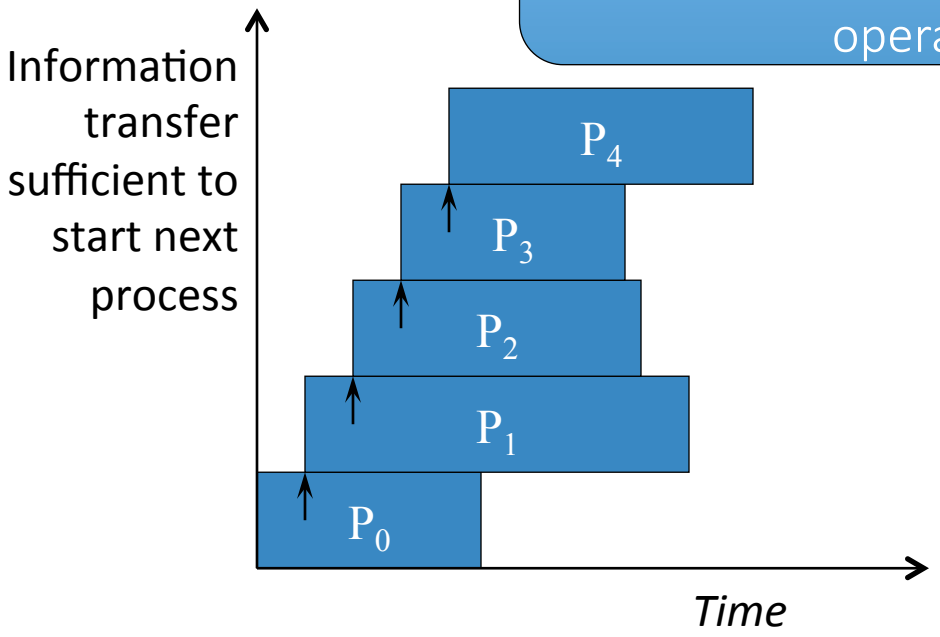


"Type 3" Pipeline Space-Time Diagram

Information to start the next process can be passed forward before the process has completed all its internal operations

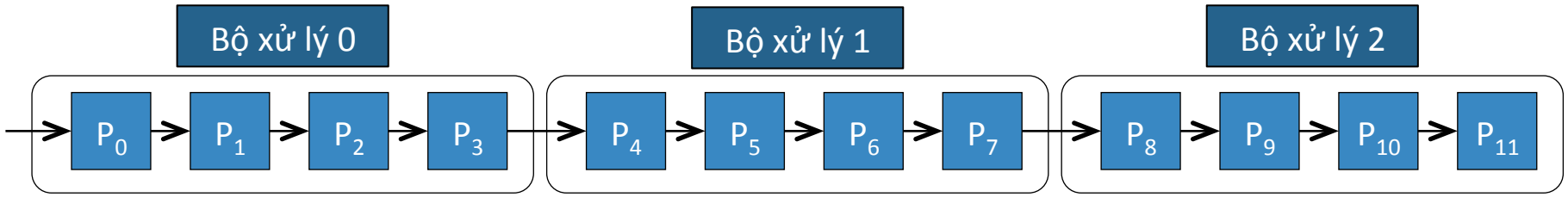


(a) Processes with the same execution time



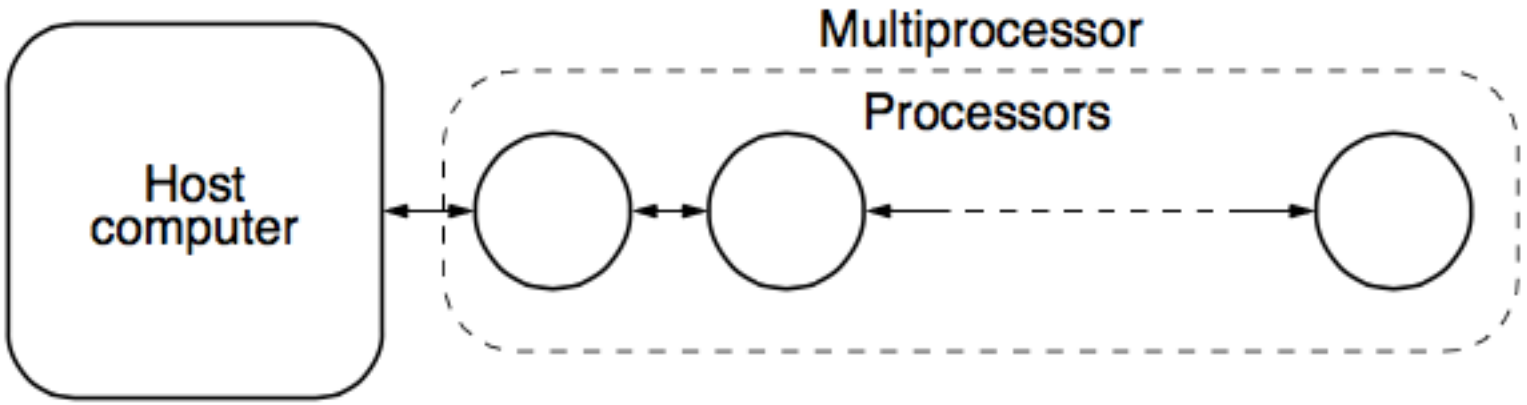
(b) Processes not with the same execution time

- Pipeline processing where information passes to next stage before end of process
- If the number of stages is larger than the number of processors in any pipeline, a group of stages can be assigned to each processor:



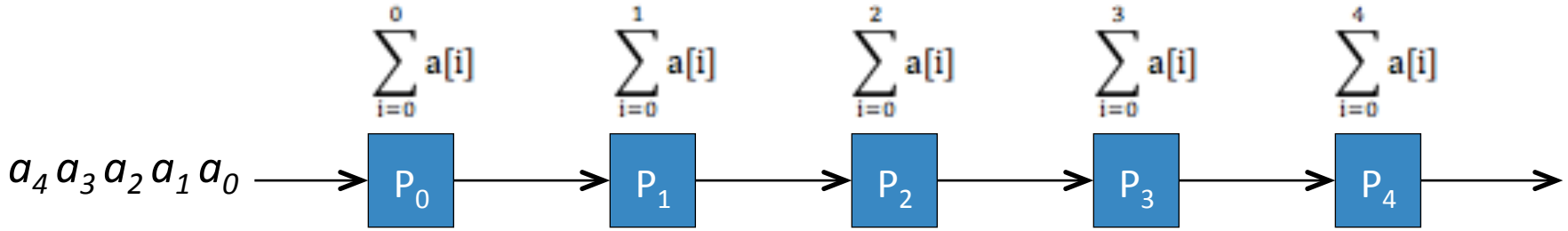
Computing Platform for Pipelined Applications

- Multiprocessor system with a line configuration:



Strictly speaking pipeline may not be the best structure for a cluster - however a cluster with switched direct connections, as most have, can support simultaneous message passing

Prefix sums or Scan



Khởi động:

```
1.  k = 0;     // số giá trị nhận được
2.  psum = 0; // psum là giá trị tổng tiền tố tính tại Pi
```

Tính toán tại đoạn ống khi nhận dữ liệu:

```
3.  x = sin;  // nhận dữ liệu nhập
4.  if (++k <= i)
5.    psum += x;
6.  sout = x;
```

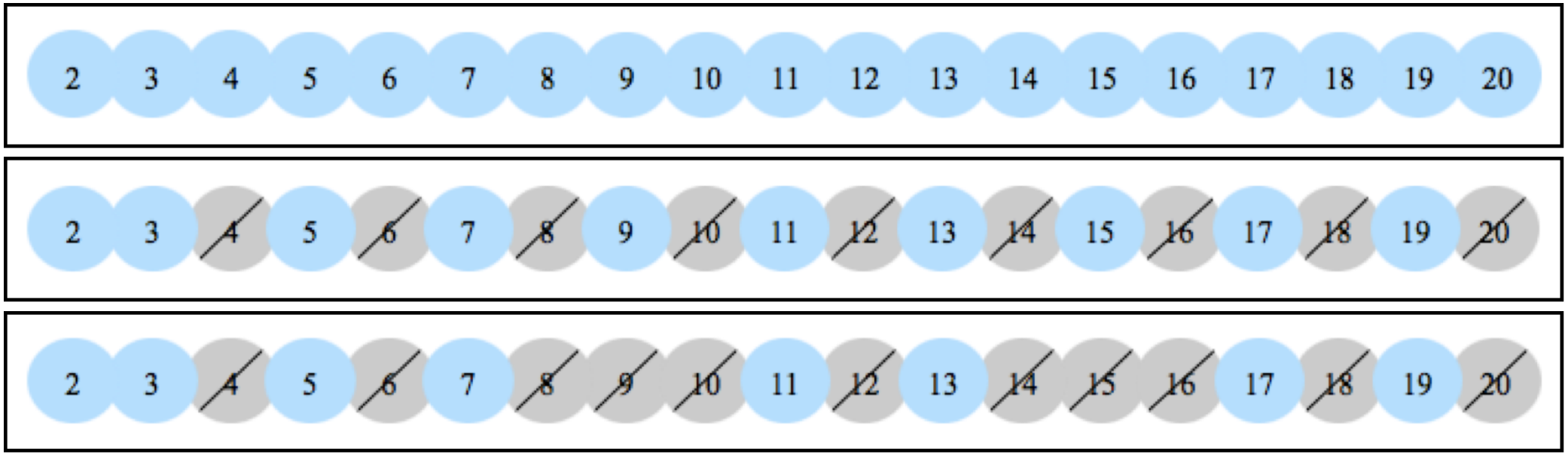
Kết quả:

```
7.  return(psum);    // psum = ∑_{k=0}^{i} x_k
```

Prime number generation

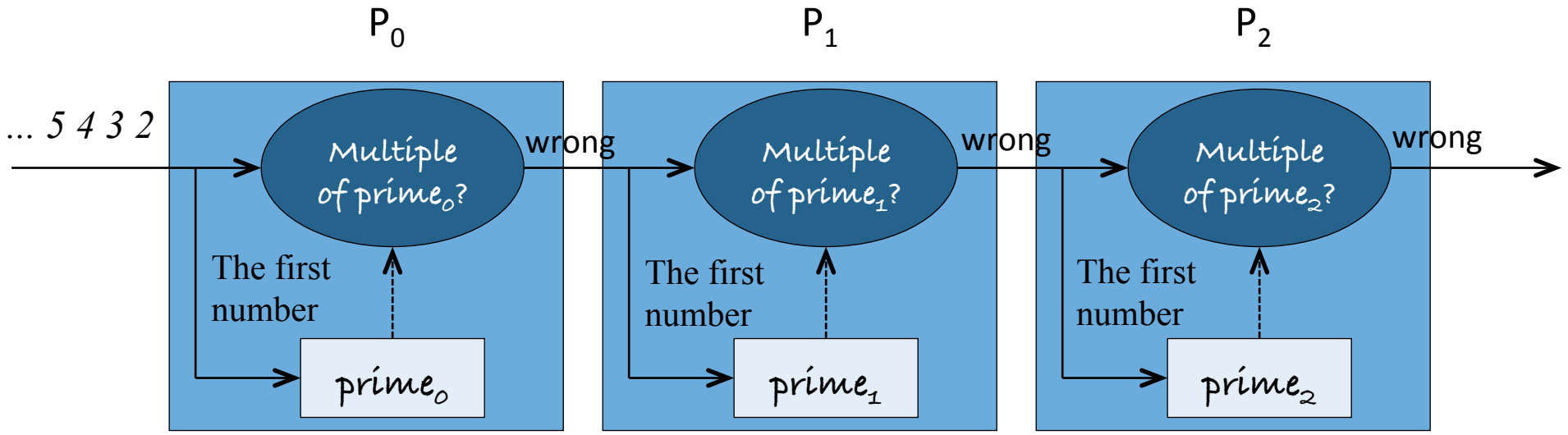
- The sieve of Eratosthenes

```
  // Khởi tạo xem tất cả là số nguyên tố
1.  for (i=2; i<=n; i++)
2.    prime[i] = 1;
  // Loại bỏ các bội số của các số nguyên tố từ 2 đến √n
3.  for (i=2; i<=sqrt(n); i++)
4.    if (prime[i] == 1)
5.      for (j=i+i; j<=n; j=j+i)
6.        prime[j] = 0;
```

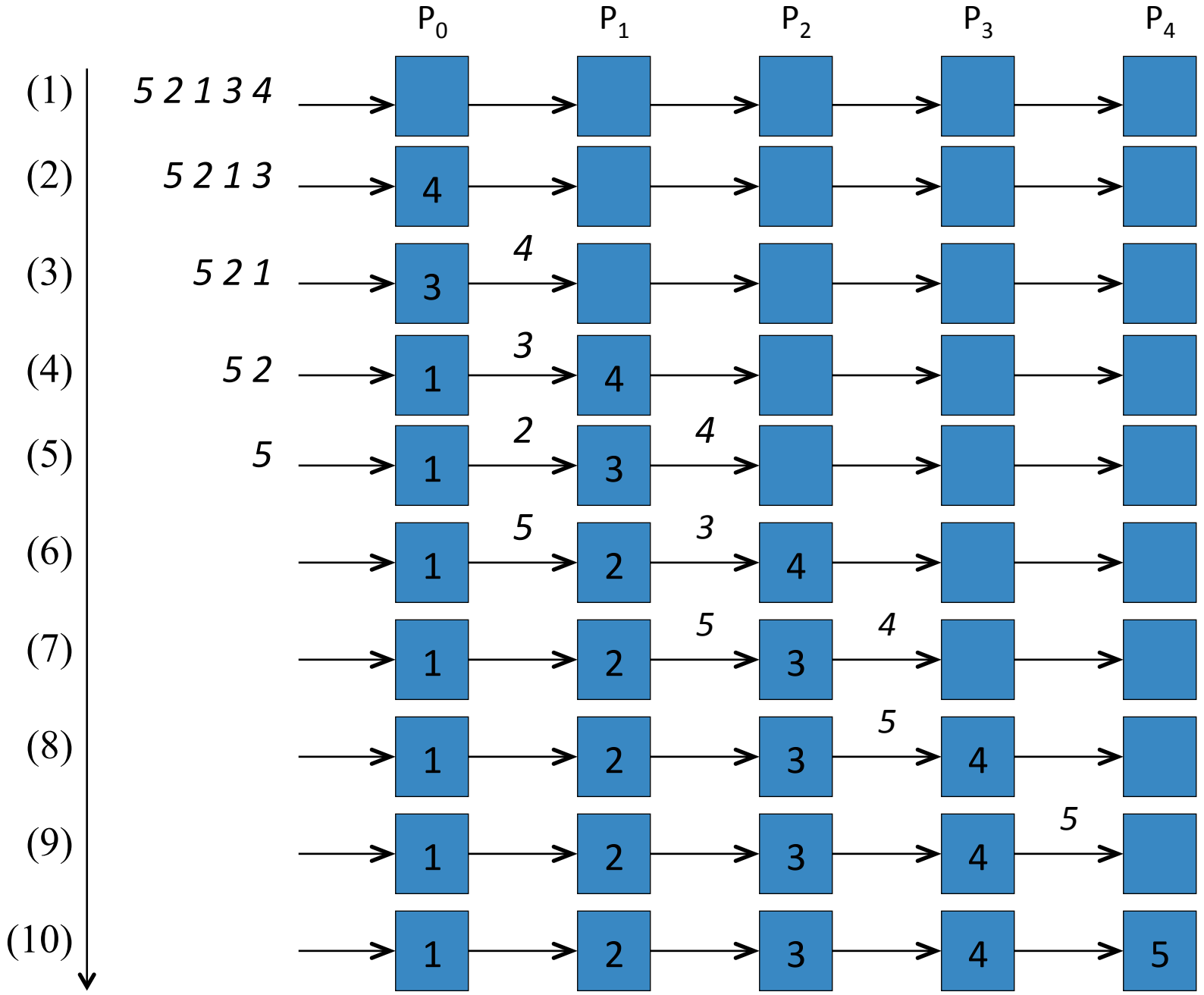


Prime number generator: pipeline

```
   Khởi động:
1.  prime = 0;  // Chưa có giữ số nào nguyên tố nào
   Tính toán tại đoạn ống khi nhận dữ liệu:
2.  x = s_in;   // nhận dữ liệu nhập
3.  if (prime == 0)
4.    prime = x;  // Giữ lại số nguyên tố thứ i+1
   // Loại bỏ bội số của số nguyên tố thứ i+1 (prime)
5.  else  if ((x % prime) != 0)
6.            s_out = x;
   Kết quả:
7.  return(prime);
```



Pipeline sorting

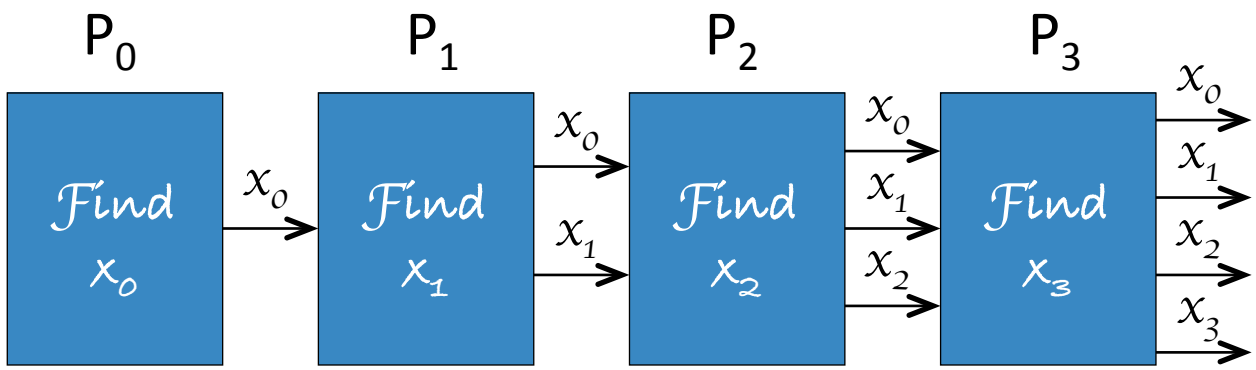


Solving a system of linear equations – special case

$$
\begin{aligned}
& a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 + \quad \ldots \quad + a_{n-1,n-1}x_{n-1} && = b_{n-1} \quad (n\text{-}1) \\
& \ldots \\
& a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 && = b_2 \quad (2) \\
& a_{1,0}x_0 + a_{1,1}x_1 && = b_1 \quad (1) \\
& a_{0,0}x_0 && = b_0 \quad (0)
\end{aligned}
$$

- Equation (0): $x_0 = \dfrac{b_0}{a_{0,0}},$
- Equation (1): $x_1 = \dfrac{b_1 - a_{1,0}x_0}{a_{1,1}},$
- Equation (2): $x_2 = \dfrac{b_2 - a_{2,0}x_0 - a_{2,1}x_1}{a_{2,2}},$
- Equation (i): $x_i = \dfrac{b_i - \sum_{j=0}^{i-1} a_{i,j}x_j}{a_{i,i}}.$

Solving a system of linear equations – pipeline version 1



Khởi động:

```
1.  sum = 0;
```

Tính toán tại đoạn ống khi nhận dữ liệu:

// Nhận dữ liệu nhập $x_0, x_1, x_2, \dots, x_{i-1}$

```
2.  for (j=0; j<i; j++) {
3.    x[j] = s_in;    // Nhận dữ liệu
4.    s_out = x[j];   // Truyền x[j] sang cho P_{i+1}
5.  }
```

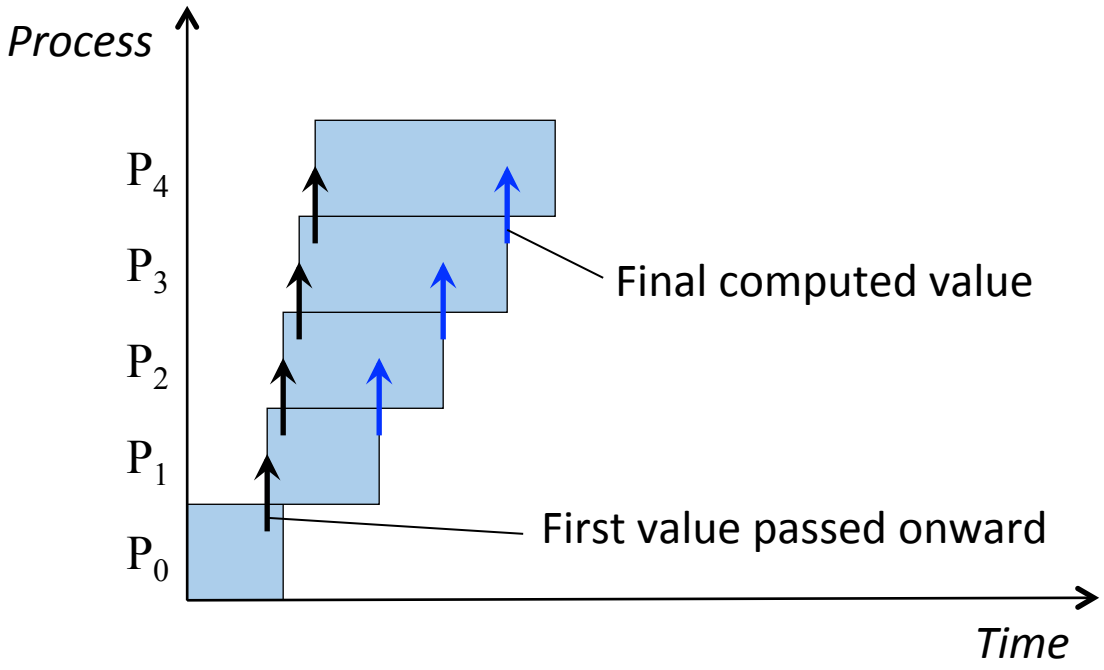
// Tính nghiệm x_i (x[i]) từ phương trình thứ (i)

```
6.  for (j=0; j<i; j++)
7.    sum = sum + a[i][j]*x[j];
8.  x[i] = (b[i] - sum)/a[i][i];
9.  s_out = x[i];  // Truyền x[i] (nghiệm x_i) sang cho P_{i+1}
```

Kết quả:

```
10. return(x[i]); // Nghiệm x_i
```

Solving a system of linear equations – pipeline version 2



Khởi động:

```
1.  sum = 0;
```

Tính toán tại đoạn ống khi nhận dữ liệu:

// Nhận dữ liệu nhập $x_0, x_1, x_2, \ldots, x_{i-1}$

```
2.  for (j=0; j<i; j++) {
3.    x[j] = s_in;    // Nhận dữ liệu
4.    s_out = x[j];   // Truyền x[j] sang cho P_{i+1}
5.    sum = sum + a[i][j]*x[j];
6.  }
```

// Tính nghiệm x_i (x[i]) từ phương trình thứ (i)

```
7.  x[i] = (b[i] - sum)/a[i][i];
8.  s_out = x[i];  // Truyền x[i] (nghiệm x_i) sang cho P_{i+1}
```

Kết quả:

```
9.  return(x[i]); // Nghiệm x_i
```