1 Conceptul de paralelism

Conceptul Von Neumann a dominat arhitecturile calculatoarelor. Acest concept a stat la baza sistemelor secventiale, monoprocesor.
Reamintim ca o structura Von Neumann este caracterizata de:

1) un singur element de procesare separat de memorie printr-o magistrala de comunicatie;
2) memoria este formata din locatii de dimensiune fixa cu organizare liniara si adresabila pe un singur nivel;
3) limbajul masinii este in general de nivel scazut avind instructiuni ce controleaza operatii simple si actioneaza asupra unor operanzi elementari;
4) executa operatiile elementare in mod secvential;
5) capabilitatile de intrare iesire sunt reduse;
6) reprezentarea interna a datelor si instructiunilor se face sub aceeasi forma.

Clasificarea sistemelor de calcul se poate face din mai multe puncte de vedere. Cele mai uzuale clasificari sunt bazate pe:

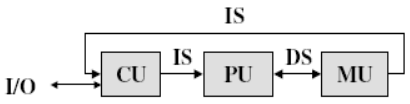
1.1 Taxonomia lui Flynn (bazata pe relatia dintre fluxul de instructiuni si fluxul de date:

1. SISD (single instruction single data flow)
2. SIMD (single instruction multiple data flow)
3. MISD (multiple instruction single data flow)
4. MIMD (multiple instruction multiple data flow)

SISD (Single Instruction Stream Over A Single Data Stream)

❖SISD
- ✓Conventional sequential machines

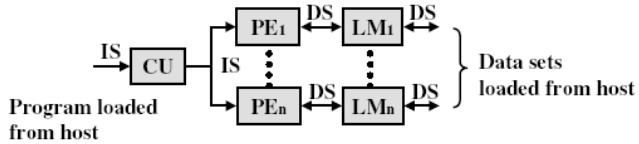
IS : Instruction Stream DS : Data Stream
CU : Control Unit PU : Processing Unit
MU : Memory Unit



SIMD (Single Instruction Stream Over Multiple Data Streams)

❖SIMD
- ✓Vector computers
- ✓Special purpose computations

PE : Processing Element LM : Local Memory



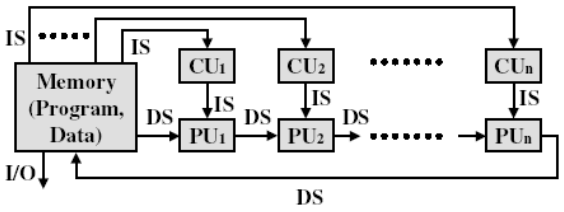
SIMD architecture with distributed memory

MISD (Multiple Instruction Streams Over A Single Data Streams)

❖MISD

✓Processor arrays, systolic arrays

✓Special purpose computations

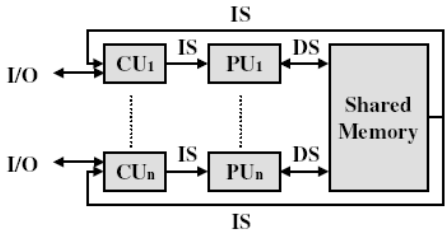


MISD architecture (the systolic array)

MIMD (Multiple Instruction Streams Over Multiple Data Stream)

❖MIMD

✓General purpose parallel computers



MIMD architecture with shared memory

1.2 Taxonomia lui Shore:

1. masina seriala pe cuvint paralela pe biti
2. masina paralela pe cuvint seriala pe biti
3. masina ortogonala
4. masina masiv neconectat
5. masina masiv conectat
6. masina masiv cu logica in memorie

1.3 Structurala

1. uniprocesoare seriale
2. unicalculatoare paralele bazata pe paralelism functional si pipeline
3. masive de procesoare

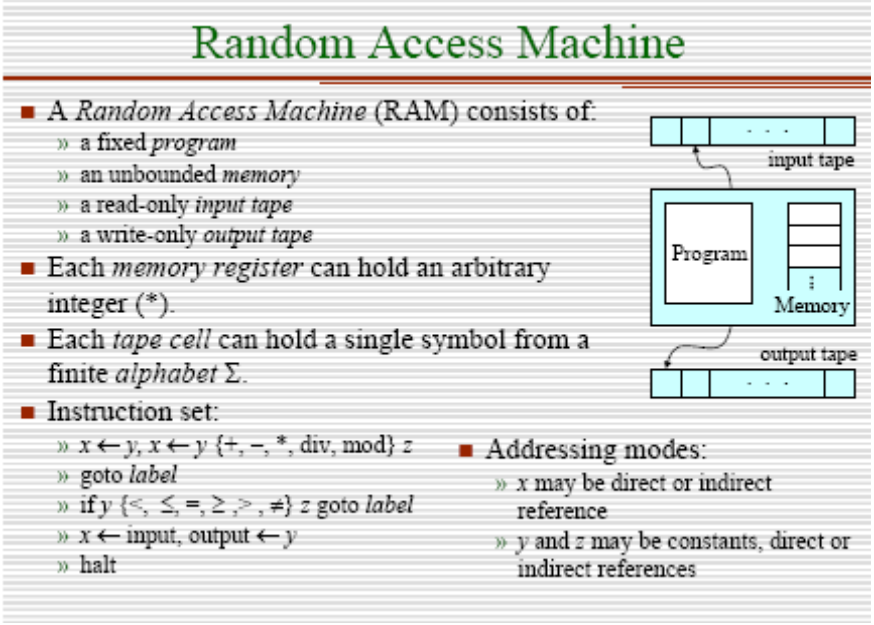
1.4 Modele ale calculului paralel

1. Model RAM (Random Acces Machine)
2. PIPELINE
3. PROCESOARE DE VECTORI
4. PROCESOARE DE MASIVE
5. MULTIPROCESOARE cu memorie partajata
6. PRAM (Parallel Random Acces Machine)
7. MULTIPROCESOARE cu transfer prin mesaje
8. PROCESARE SISTOLICA
9. PROCESARE DATA FLOW

2 Caracteristicile generale ale modelelor calculului paralel

2.1 Structura cu acces aleator RAM (random acces machine).

Masina de baza monoprocesor poate fi abstractizata ca o masina cu acces aleator. De altfel acest model este similar cu modelul masinii Turing. Un astfel de model este utilizat pentru dezvoltare de algoritmi si analiza complexitatii acestora si are rolul de a constitui punctul de plecare pentru modelele paralele.



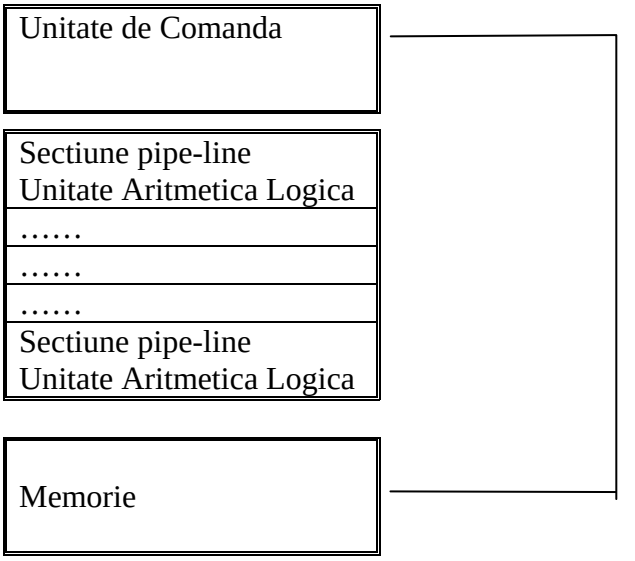
RAM = (S, Γ ,Σ, f, s_0, Z)

S	multimea finita a starilor automatului;
Γ	alfabetul finit al benzii;
Σ	Σ C S alfabetul finit al benzii de intrare;
f	functia de tranzitie, eventual partial definita f : S * Γ --> S * Γ
s_0	s_0 € S starea initiala;
Z	inclus in S - multimea starilor finale

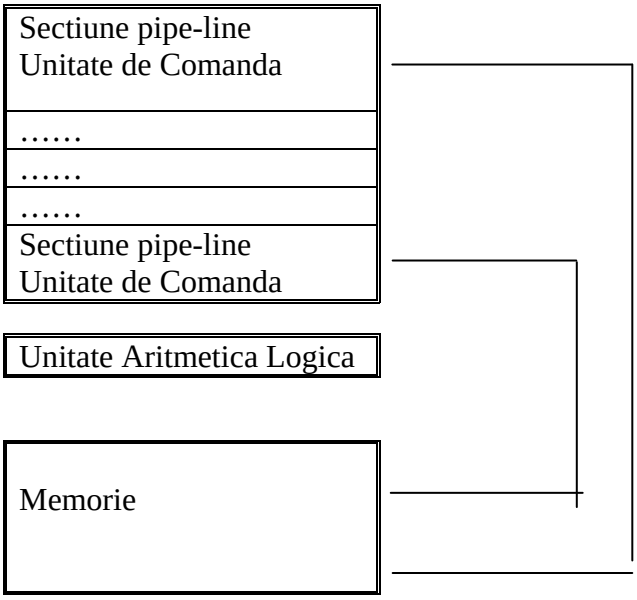
2.2 Structura pipeline.

Conceptul de pipeline consta in a divide un task T in subtask-uri T_1 , T_2 , ...,T_k si de a le atribui unor elemente de procesare. Paralelismul se poate realiza fie la nivel de

- prelucrare aritmetica;
- citire, interpretare, executie instructiuni



- paralelism la nivel de prelucrare aritmetica;



- paralelism la nivel de citire, interpretare, executie instructiuni

2.3 Procesoare de vectori.

In cadrul acestui model, exista un set de instructiuni care trateaza vectorul ca operand simplu. In cadrul acestei structuri trebuie sa existe o UAL de tip pipeline si registre de vectori.
O astfel de structura o gasim in sisteme monoprocesor care au ca extensie procesare de vectori. Acopera o gama redusa de probleme cu caracter vectorial.

2.4 Procesoare de masive.

Procesoarele de masive se refera la calculatoarele paralele sincrone care constau din UAL multiple, supervizate de o singura unitate de control, care efectueaza aceeasi operatie la un moment dat.
UAL efectueaza aceeasi operatie asupra unor date diferite. Acest model exprima categoria de sisteme SIMD si este cunoscut si sub denumirea de model date paralel.

2.5 Multiprocesoare cu memorie partajata.

Acest model descrie structurile in care fiecare procesor contine propria unitate de prelucrare, propria memorie si propria unitate de comanda si comunica prin intermediul unei retele de comutare cu module de memorie partajata.
Fiecare procesor executa propriul set de instructiuni din memoria locala sau memoria divizata. Reteaua de comutare permite schimbul de informatii intre procesoare si intre procesoare si memoria partajata divizata.

2.6 Structura paralela cu acces aleator (PRAM).

Este o extensie a modelului RAM.
Programele sunt diferite unele de altele. Comunicatia intre procesoare si memorie este considerata ca se face fara introducere de intirziere.Memoria comuna este disponibila tuturor programelor.
Exista patru posibilitati de a avea acces la memorie:

- EREW - exclusive read exclusive write (programele nu pot avea acces simultan la eceasta memorie
- ERCW - exclusive read concurent write
- CREW - concurent read exclusive write
- CRCW - concurent read concurent write

Acest model este utilizat numai pentru proiectarea algoritmilor paraleli in structuri cu memorie divizata si studiul proprietatilor acestora

2.7 Multiprocesoare cu transfer prin mesaje.

In acest model memoria este distribuita fiecarui procesor astfel incit fiecare procesor dispune de propriul program si propria memorie de date.
Comunicarea datelor divizate este realizata prin schimburi de mesaje prin intermediul unei retele de comutare mesaje.
Acest model este diferit de modelul cu memorie divizata deoarece interconexiunea se realizeaza prin mesaje si nu prin acces direct.

2.8 Procesare sistolica.

Modelul procesarii sistolice consta din elemente de procesare identice aranjate intr-o structura pipeline. Procesarea sistolica este caracterizata prin interconexiuni simple si regulate ceea ce permite integrarea VLSI.
Presupunem ca avem un element de procesare sistolica
Yout=A*X+Yin

Systolic Architecture

Orchestrate data flow for high throughput with less memory access

Different from pipelining

Nonlinear array structure, multidirection data flow, each PE may have (small) local instruction and data memory

Different from SIMD

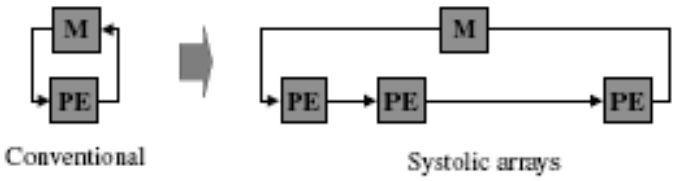
Each PE may do something different

Initial motivation

VLSI enables inexpensive special-purpose chips

Represent algorithms directly by chips connected in regular pattern

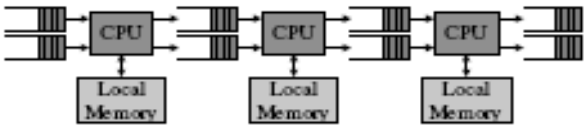
Systolic Architectures



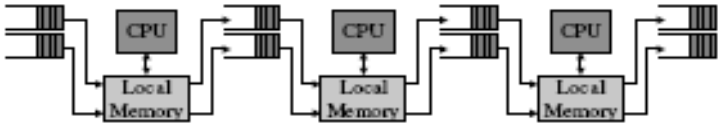
Replace a processing element(PE) with an array of PE's
without increasing I/O bandwidth

Two Communication Styles

Systolic communication



Memory communication



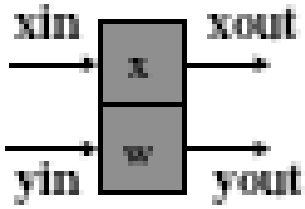
Matrix Multiplication

$$y_i = \sum_{j=1}^{n} a_{i,j} x_j, i = 1,\ldots,n$$

Recursive algorithm

```
for i = 1 to n
   y(i,0) = 0
   for j = 0 to n
      y(i,0) = y(i,0) + a(i,j) * x(j,0)
```

Use the following PE

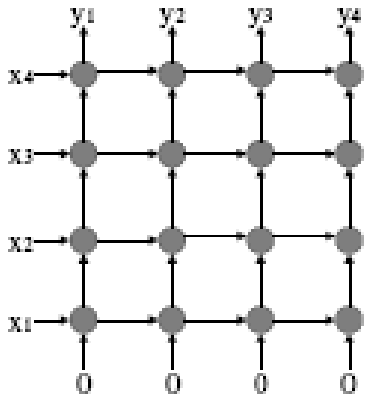


xout = x

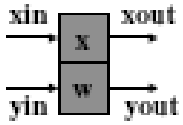
x = xin

yout = yin + w * xin

Systolic Array Representation of Matrix Multiplication

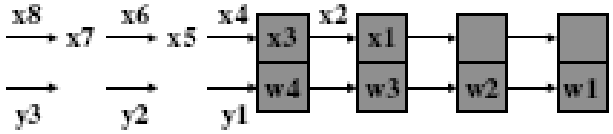


Example of Convolution



xout = x
x = xin
yout = yin + w * xin

y(i) = w1 * x(i) + w2 * x(i+1) + w3 * x(i+2) + w4 * x(i+3)
y(i) is initialized as 0.



2.9 Procesare data flow.

Modelul de procesare data flow specifica efectuarea unei operatii imediat ce operanzii devin disponibili. Aceasta elimina necesitatea existentei contorului de program . Rezultatul produs de o instructiune este utilizat ca o data pura sau token si este furnizat direct urmatoarei instructiuni.
Masinile data flow nu necesita adrese pentru memorarea variabilelor ceea ce contrasteaza complet cu masinile Von Neumann.
Operatia se efectueaza numai cind token-ul este prezent pe ambele intrari ale operatorului.

2.9.1 Data flow architecture:

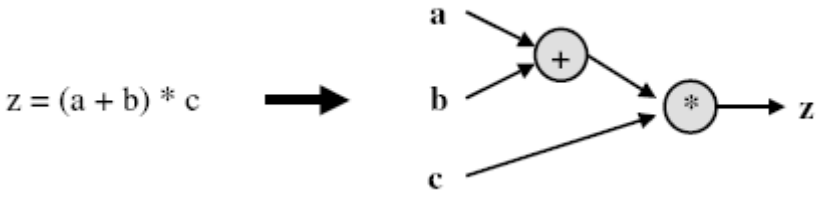
Data-driven model

- A program is represented as a directed acyclic graph in which a node represents an instruction and an edge represents the data dependency relationship between the connected nodes
- Firing rule
  - A node can be scheduled for execution if and only if its input data become valid for consumption

Dataflow languages

- Id, SISAL, Silage, ...
- Single assignment, applicative(functional) language, no side effect
- Explicit parallelism

2.9.2 Data Graph



2.9.3 Data flow vs Control flow

2.9.3.1 Control-flow computer

Program control is explicitly controlled by instruction flow in program
Basic components

- PC (Program Counter)
- Shared memory
- Control sequencer

Advantages

Full control
Complex data and control structures are easily implemented

Disadvantages

Less efficient
Difficult in programming
Difficult in preventing runtime error

2.9.3.2 Data-flow computer

Execution of instructions is driven by data availability
Basic components

- Data are directly held inside instructions
- Data availability check unit
- Token matching unit
- Chain reaction of asynchronous instruction executions

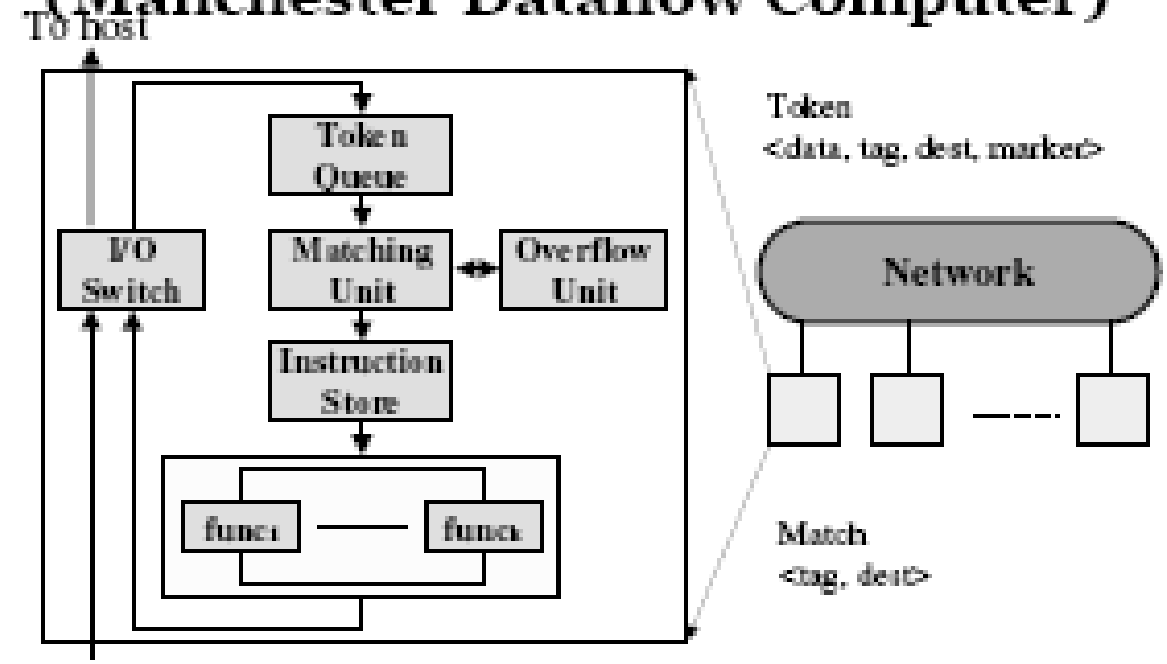
Advantages

- Very high potential for parallelism
- High throughput
- Free from side-effect

Disadvantages

- Time lost waiting for unneeded arguments
- High control overhead
- Difficult in manipulating data structures

Dataflow Machine (Manchester Dataflow Computer)



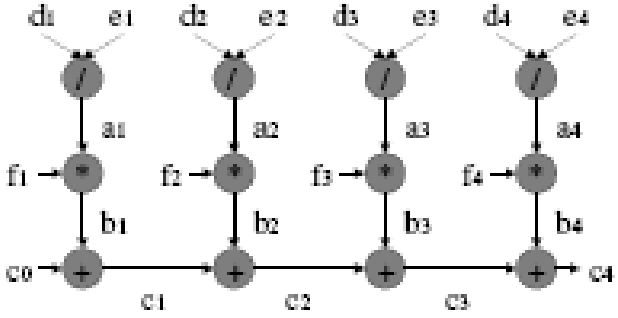
First actual hardware implementation

Dataflow Representation

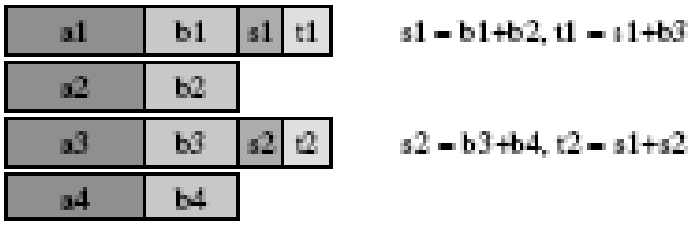
```
input d,e,f
   c0 = 0

for i from i to 4 do
   begin
      ai := di / ei
      bi := ai * fi
      ci := bi + ci-1
   end

output a, b, c
```



Execution On A Data Flow Machine



Parallel execution on a shared-memory 4-processor system in 7 cycles

Data-driven execution on a 4-processor dataflow computer in 9 cycles

Problems

Excessive copying of large data structures in dataflow operations

I-structure : a tagged memory unit for overlapped usage by the producer and consumer

- Retreat from pure dataflow approach (shared memory)
- Handling complex data structures

Chain reaction control is difficult to implement

Complexity of matching store and memory units

Expose too much parallelism

Convergence of Dataflow Machines

Converged to use conventional processors and memory

Support for large, dynamic set of threads to map to processors

Operations have locality across them, useful to group together

Typically shared address space as well

But separation of programming model from hardware (like data-parallel)

Contribution

Integration of communication with thread (handler) generation

Tightly integrated communication and finegrained synchronization

Each instruction represents a synchronization operation.

Absorb the communication latency and minimize the losses due to synchronization waits.