



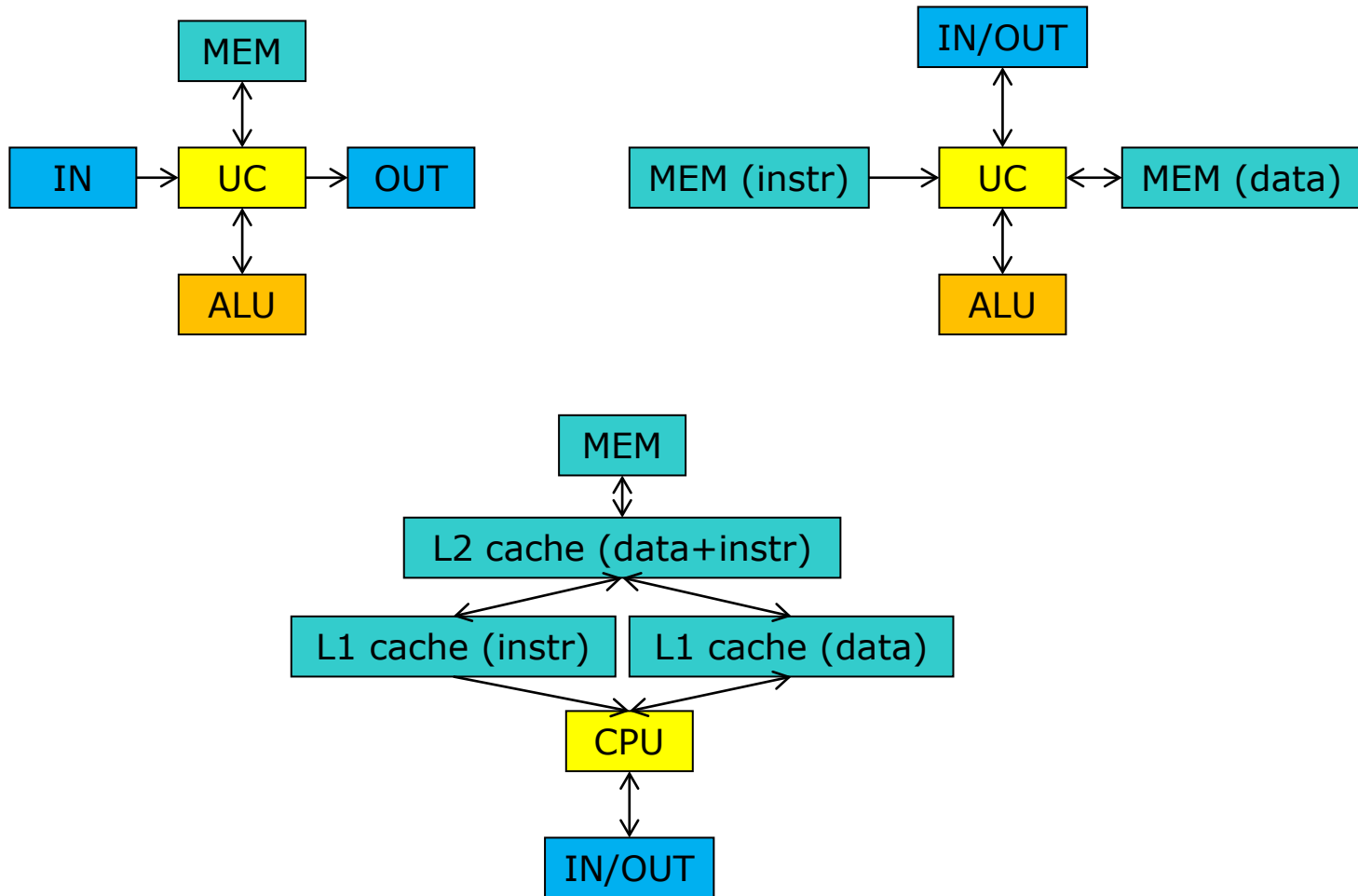
Sisteme cu microprocesoare

Cursul 13 Arhitecturi avansate de calcul

Ce se urmareste ?

- cresterea performantelor de calcul prin:
 - inovare arhitecturala
 - cresterea frecventei de lucru
- modelul clasic de calculator:
 - modelul Jon von Neumann
 - modelul bazat pe un procesor si pe o magistrala
- arhitecturi noi:
 - diferite forme de executie paralela:
 - arhitecturi paralele:
 - paralelism la nivel de date - UAL multiple
 - paralelism la nivel de instructiuni – arhitectura pipeline
 - paralelism la nivel de fire de executie/taskuri:
 - arhitecturi multi-core
 - arhitecturi multiprocesor
 - supercalculatoare – calculatoare paralele
 - sisteme distribuite
 - structuri simplificate de calcul
 - arhitectura RISC

Architettura von Neuman vs architettura Harvard



Architettura Harvard modificata

Arhitectura RISC – Reduced Instruction Set Computer

○ Observatii:

- statisticile arata ca desi procesorul are multe tipuri de instructiuni programatorii obisnuiesc sa foloseasca un set limitat de instructiuni simple
- un set complex de instructiuni cu instructiuni complexe necesita o UCP complexa, ce lucreaza la o frecventa mai scazuta
- cresterea de complexitate datorita setului extins de instructiuni in cele mai multe cazuri nu se justifica

○ Ideea:

- sa se simplifice la maximum UCP si implicit setul de instructiuni astfel incat procesorul sa lucreze la o frecventa de ceas mai mare

○ Principiu:

- sacrifica tot pentru viteza
- programatorul sa lucreze mai mult dar programul rezultat sa fie mai eficient din punct de vedere al timpului de executie

Evolutia arhitecturii RISC

- primele realizari:
 - RISC I si II – Univ. Berkley
 - MIPS – Univ. Stanford
 - IBM 801 – compania IBM
 - ALPHA – compania DEC
 - PowerPC – IBM si Motorola
- tendinte CISC:
 - cresterea complexitatii instructiunilor masina pentru a se apropia de instructiunile din limbajele de nivel inalt
 - programare mai simpla, mai eficienta d.p.d.v. al timpului de programare
 - consecinta: executie mai lenta a programului
 - utilizarea microprogramarii ca tehnica de implementare a UCP cu instructiuni complexe:
 - o instructiune complexa executata printr-o secventa de micro-operatii
- tendinta RISC:
 - reducerea numarului de instructiuni
 - renuntarea la instructiuni complexe
 - limitarea instructiunilor care lucreaza cu memoria – se prefera lucrul cu registre
 - moduri de adresare simple
 - cresterea frecventei de lucru
 - instructiunile executate intr-o singura perioada de ceas
 - promovarea arhitecturii pipeline

RISC v.s. CISC

| Parametru | RISC | CISC |
|----------------------------------|-----------------------|-----------------------|
| Tip de instructiune | Simplu | Complex |
| Numar de instructiuni | Redus | extins |
| Durata unei instructiuni | Un ciclu | Mai multe cicluri |
| Formatul instructiunii | Fix | Variabil |
| Mod de executie instr. | In paralel (pipeline) | Secvential |
| Moduri de adresare | Simple | Complexe |
| Instructiuni de acces la memorie | Doua: Load si Store | Aproape toate din set |
| Set de registre | multiplu | unic |
| Complexitatea | In compilator | In UCP (microprogram) |

Avantaje/dezavantaje RISC v.s. CISC

- RISC
 - Avantaje:
 - frecventa mai mare de lucru
 - mai multe instructiuni executate in unitatea de timp (MIPS)
 - obliga programatorul sa lucreze eficient
 - program executabil mai scurt (timp si spatiu)
 - Dezavantaje:
 - dificil de programat la nivel de asamblare,
 - timp mai lung pentru dezvoltarea unei aplicatii
 - daca se lucreaza in limbaj de nivel inalt se pierde eficienta castigata prin simplitate (discutabil)
- CISC:
 - Avantaje:
 - usor de programat
 - timp mai scurt pentru programare
 - compilatoarele de limbaje de nivel inalt se scriu mai usor
 - pot fi implementate usor structuri complexe de date si de program
 - Dezavantaje:
 - timp de executie mai mare pentru programe
 - cod mai lung
 - unitate centrala mai complexa, care consuma mai mult
- Concluzie:
 - combinarea celor 2 tehnici in cadrul aceluia procesor:
 - arhitectura Pentium: RISC in interior, CISC in exterior

Arhitecturi paralele

- Motivatii:
 - reducerea timpului de executie sub limita impusa de tehnologie
- principiu:
 - acolo unde un procesor nu face fata se pun mai multe procesoare
- Dificultati:
 - descompunerea problemei in secvente executabile in paralel
 - sincronizarea intre taskuri paralele
- Legea lui Amdahl
 - limiteaza performantele unui calculator paralel, datorita portiunii de program ce nu se poate paraleliza

$$\text{speed-up} = t_{\text{exec_vechi}} / t_{\text{exec_nou}} =$$

$$= \frac{t_{\text{exec_vechi}}}{t_{\text{secv.}} + t_{\text{paralel}}/n}$$

exemplu: 90% paralelizabil

$$\text{speed-up}_{\text{max}} = 10$$

$n \rightarrow \infty$

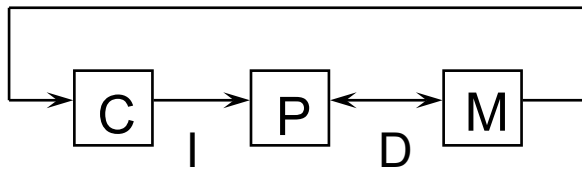
n- numarul de procesoare

Clasificare

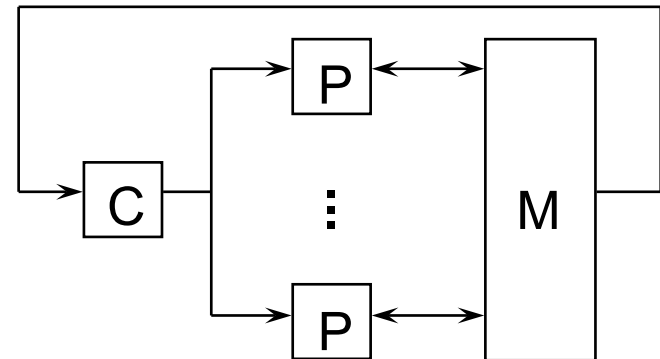
- Taxonomia lui Flynn:
 - numar de fluxuri de instructiuni
 - numar de fluxuri de date
- Tipuri de paralelism:
 - SISD – single instruction single data
 - arhitectura scalara (neparalela)
 - SIMD – single instruction multiple data
 - arhitectura cu mai multe UAL
 - paralelism de date
 - MISD – multiple instruction single data –
 - (?) arhitectura pipeline
 - MIMD – multiple instruction multiple data –
 - arhitectura paralela propriu-zisa
 - mai multe procesoare executa in paralel secvente diferite de program

Taxonomia lui Flynn

SD

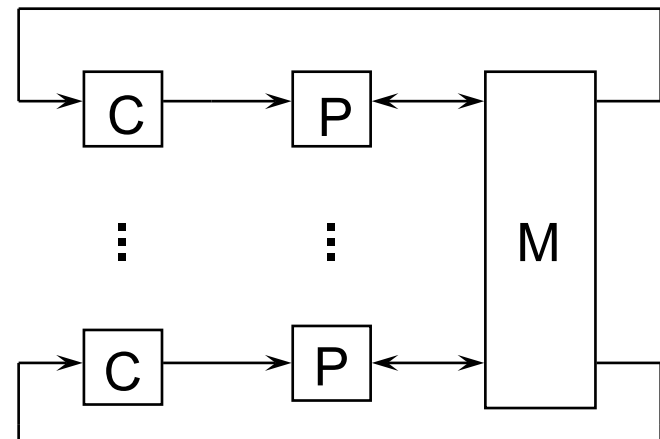
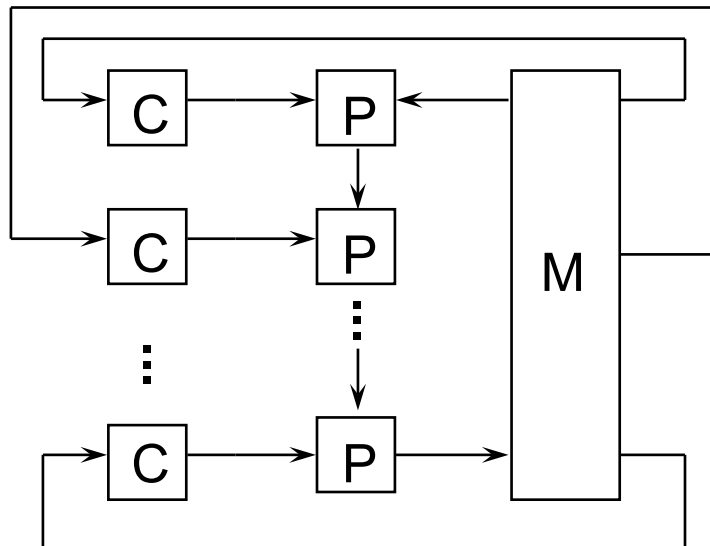


MD



SI

MI

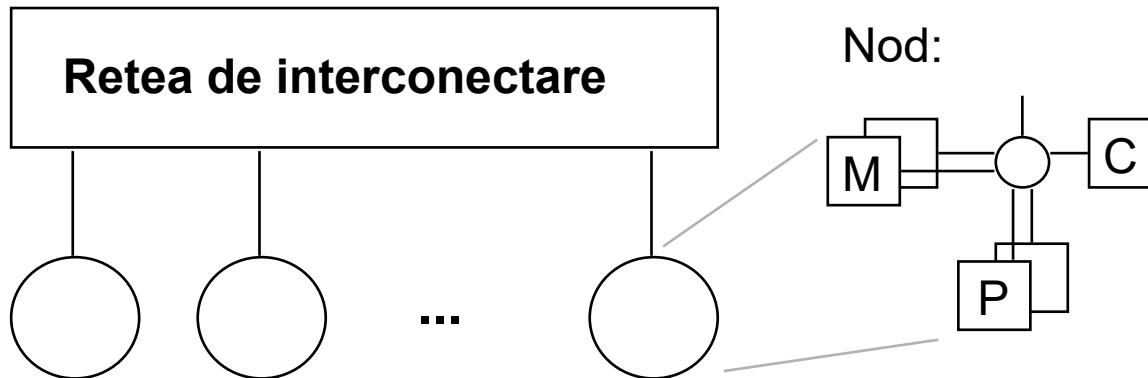


Alte forme de clasificare

- **arhitecturi functional-paralele**
 - ILP (Instruction Level Parallelism)
 - paralelism la nivel de instructiuni; exemple:
 - executie pipeline
 - VLIW - very large instruction word
 - TLP (Thread Level Parallelism)
 - paralelism la nivel de fire de executie; exemple:
 - arhitecturi multi-core
 - PLP (Process Level Parallelism)
 - paralelism la nivel de procese; exemple:
 - arhitecturi paralele, sisteme distribuite, GRID, cloud
- **arhitecturi cu paralelism de date**
 - arhitecturi de tip vector
 - arhitecturi sistolice; ex: arh. cu destinatie speciala, procesare de imagini
 - arhitecturi asociative si neurale
 - SIMDs

Caracteristicile arhitecturilor paralele

- Procesor si memorie
 - Granularitate
 - Autonomie
 - Cuplare
- Retea de interconectare
 - Topologie
 - Latenta si latime de banda

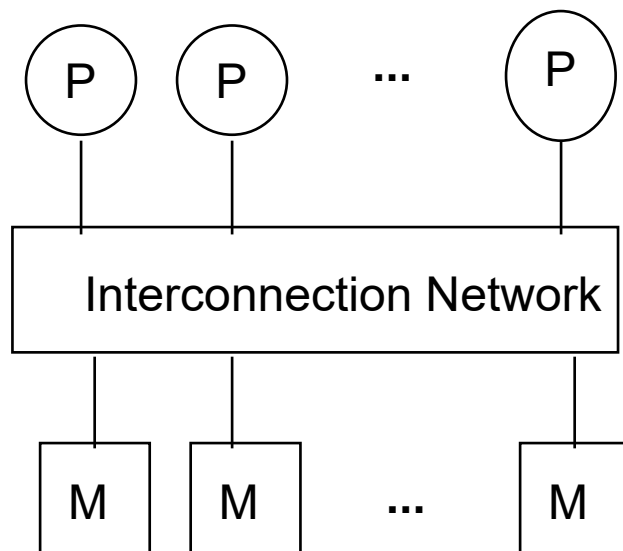


Organizarea memoriei la arhitecturile MIMD

- **Memorie partajata - Shared-Memory:**
 - spatiu liniar si global de adrese
 - variante:
 - UMA - Uniform Memory Access
 - timp uniform de acces la memorie
 - NUMA - Non-Uniform Memory Access
 - timp de acces = F (adresa de memorie)
 - COMA - Cache Only Memory Architecture
 - Symmetric Multiprocessor:
Shared-Memory + Shared I/O
- **Memorie privata - Private-Memory:**
 - multe adrese/memorii locale, private pentru fiecare procesor

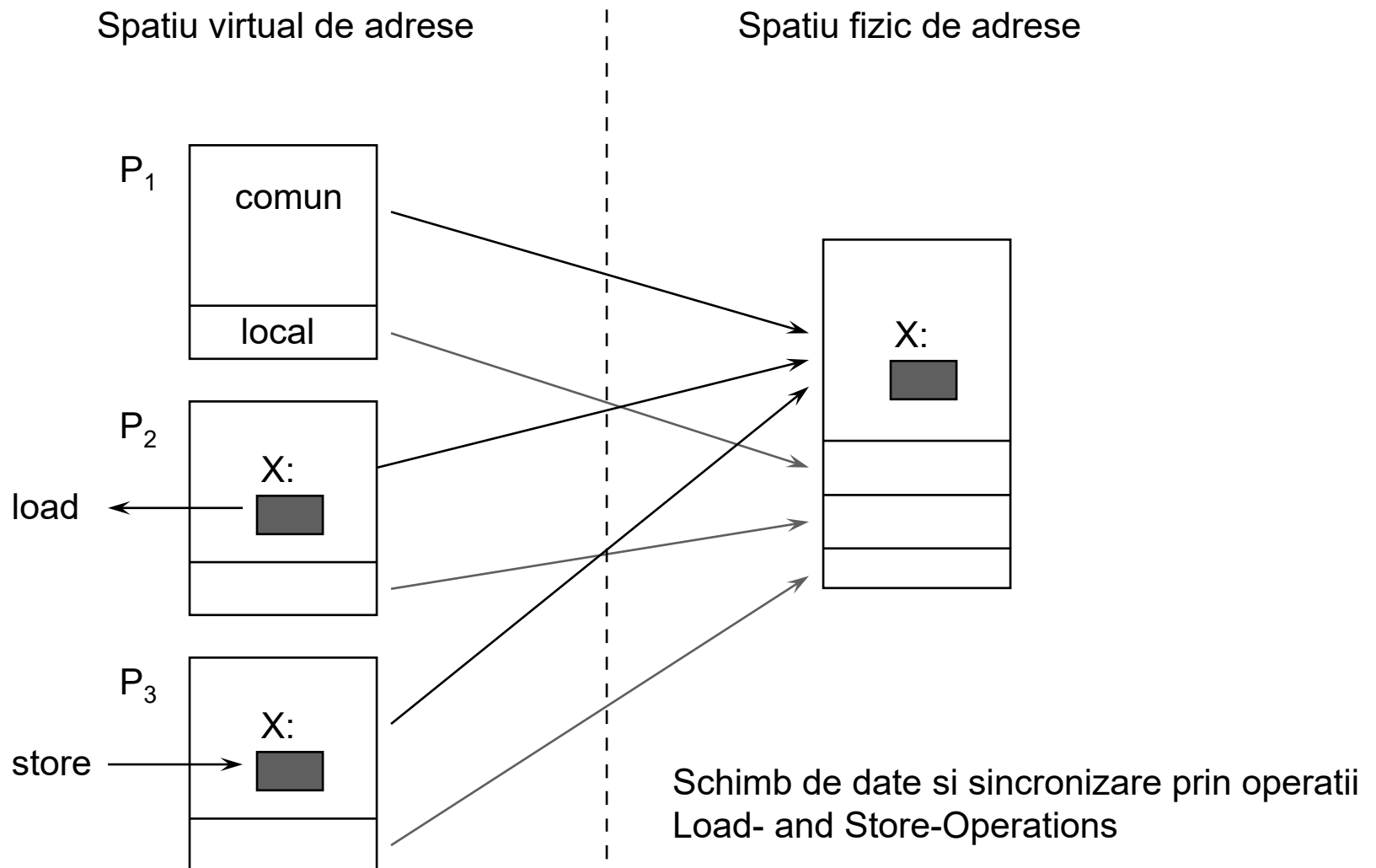
Arhitecturi cu memorie partajata

- Toate procesoarele (P) au acces direct la toate modulele de memorie (M)
- mecanism rapid si simplu de comunicatie
- probleme pot apare la accesul simultan la mediul de comunicatie (magistrala) sau la acelasi modul de memorie



Memorie centralizata sau distribuita

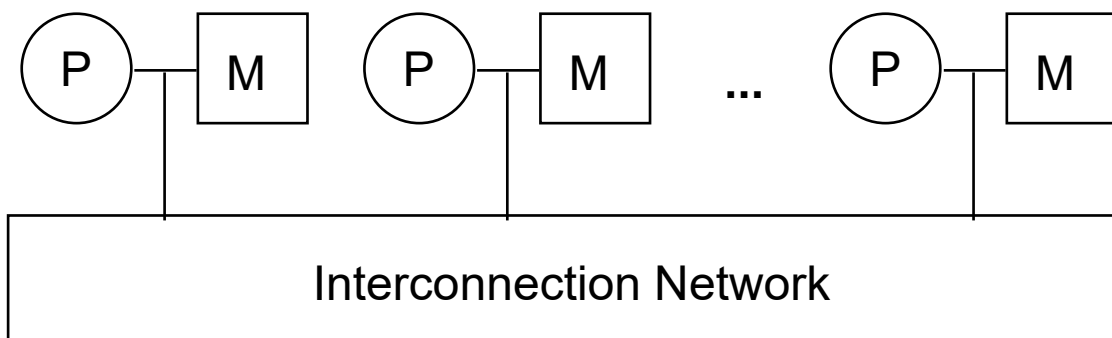
Comunicatie prin memorie partajata



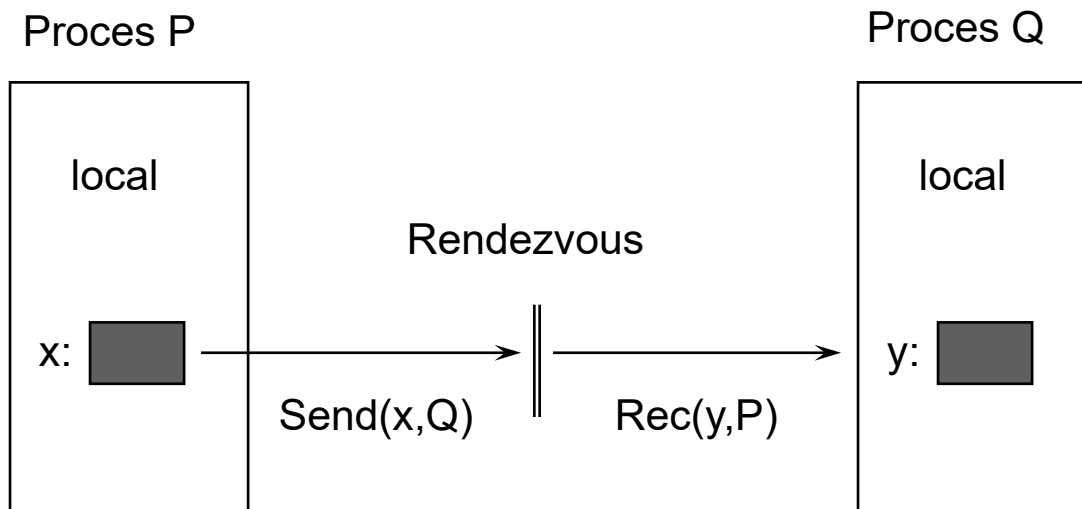
Arhitectura cu memorie privata

Procesoarele au acces direct numai la memoriile proprii (locale)

- comunicatia intre procesoare se face prin schimb de mesaje



Comunicatia in cazul memoriilor private



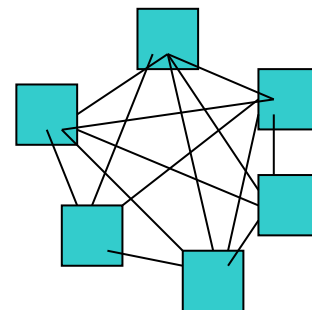
Schimb de date si sincronizare prin operatii **Send + Receive**

Procedura de comunicatie:

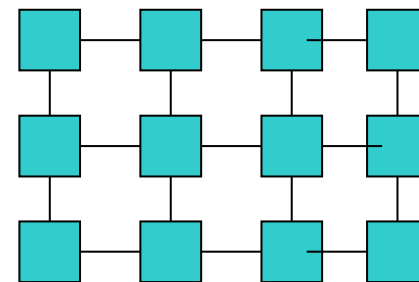
- construirea antetului (header)
- copierea datelor in bufferul emitentului
- transmiterea datelor
- copierea datelor din bufferul de intrare al receptorului

Retele de interconectare

- Probleme de proiectare
 - performanta ridicata – conexiuni multiple
 - Cost redus – conexiuni putine
- Tipuri de topologii de interconectare:
 - fiecare cu fiecare
 - costisitor pentru numar mare de procesoare
 - $nr. \text{ canale} = n(n-1)/2$
 - structura de tip matrice (bidimensionala)
 - fiecare procesor dialogheaza direct cu 4 vecini si indirect cu orice alt nod prin intermediul unor noduri intermediare
 - compromis intre viteza si cost
 - implementari practice: transputere
 - procesor simplu cu 4 canale de comunicatie, conectabil in structura matriciala



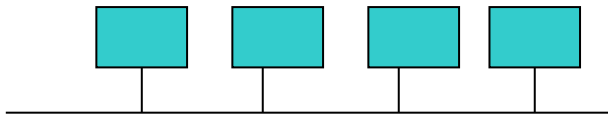
Fiecare cu fiecare



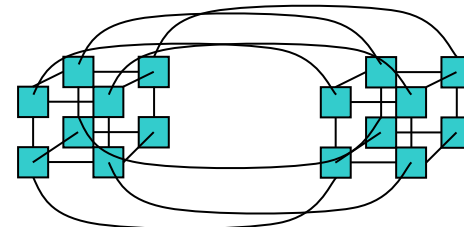
Matrice bidimensionala

Retele de interconectare

- structura de tip magistrala
 - un singur mediu de comunicatie
 - necesita protocol de acces la mediu de comunicatie
 - transfer rapid intre toate nodurile, vizibilitate directa
 - pot fi implementate transmisii multicast si broadcast
- structura de tip hipercub
 - structura ce optimizeaza numarul de conexiuni si calea minima intre colturile structurii



Magistrala



Hipercub de ordinul 2,3 si 4