5. Introducere în arhitecturi paralele

- Tipuri și nivele de paralelism
- Clasificarea arhitecturilor paralele
- Arhitecturi vectoriale
- Arhitecturi SIMD
- Arhitecturi sistolice
- Arhitecturi MIMD
- Arhitecturi specifice unor domenii
- Arhitecturi cu fire de execuție multiple

Arhitecturi cu fire de execuție multiple

- Arhitecturi cu fire de execuție multiple
  - Prezentare generală
  - Modelul fluxului de control
  - Modelul fluxului de date
  - Arhitecturi cu flux de date
  - Exemple de calculatoare cu flux de date

Prezentare generală (1)

- Utilizează un mecanism pentru comutarea rapidă a contextului între firele de execuție
  - Metodă similară cu cea utilizată de sistemele *multi-tasking*
  - Firele de execuție care așteaptă operații de I/E sau de sincronizare sunt suspendate
  - Contextul unui fir de execuție suspendat este salvat în memorie sau într-un set de registre
  - La reluarea execuției, procesorul este încărcat cu contextul salvat

Prezentare generală (2)

- Cerința pentru o eficiență ridicată:
  - Reducerea timpului de inactivitate al procesorului
- Trebuie îndeplinite două condiții:
  - Comutarea foarte rapidă a contextului → se realizează prin mecanisme hardware
  - Disponibilitatea unui număr suficient de fire de execuție
    - Păstrarea contextului unui număr mare de fire de execuție este costisitoare → compromis

Prezentare generală (3)

- Avantajul acestor arhitecturi:
  - Tolerează întârzierile datorate accesului la memorii non-locale și sincronizărilor
- Accesul la memorii non-locale
  - Timpul de acces variază cu distanța
  - Se asigură ca timpul de comutare să fie mai mic decât cel mai redus timp de acces
- Sincronizarea între procese/fire de execuție
  - Mecanism hardware pentru detectarea unei situații de sincronizare

Arhitecturi cu fire de execuție multiple

- Arhitecturi cu fire de execuție multiple
  - Prezentare generală
  - Modelul fluxului de control
  - Modelul fluxului de date
  - Arhitecturi cu flux de date
  - Exemple de calculatoare cu flux de date

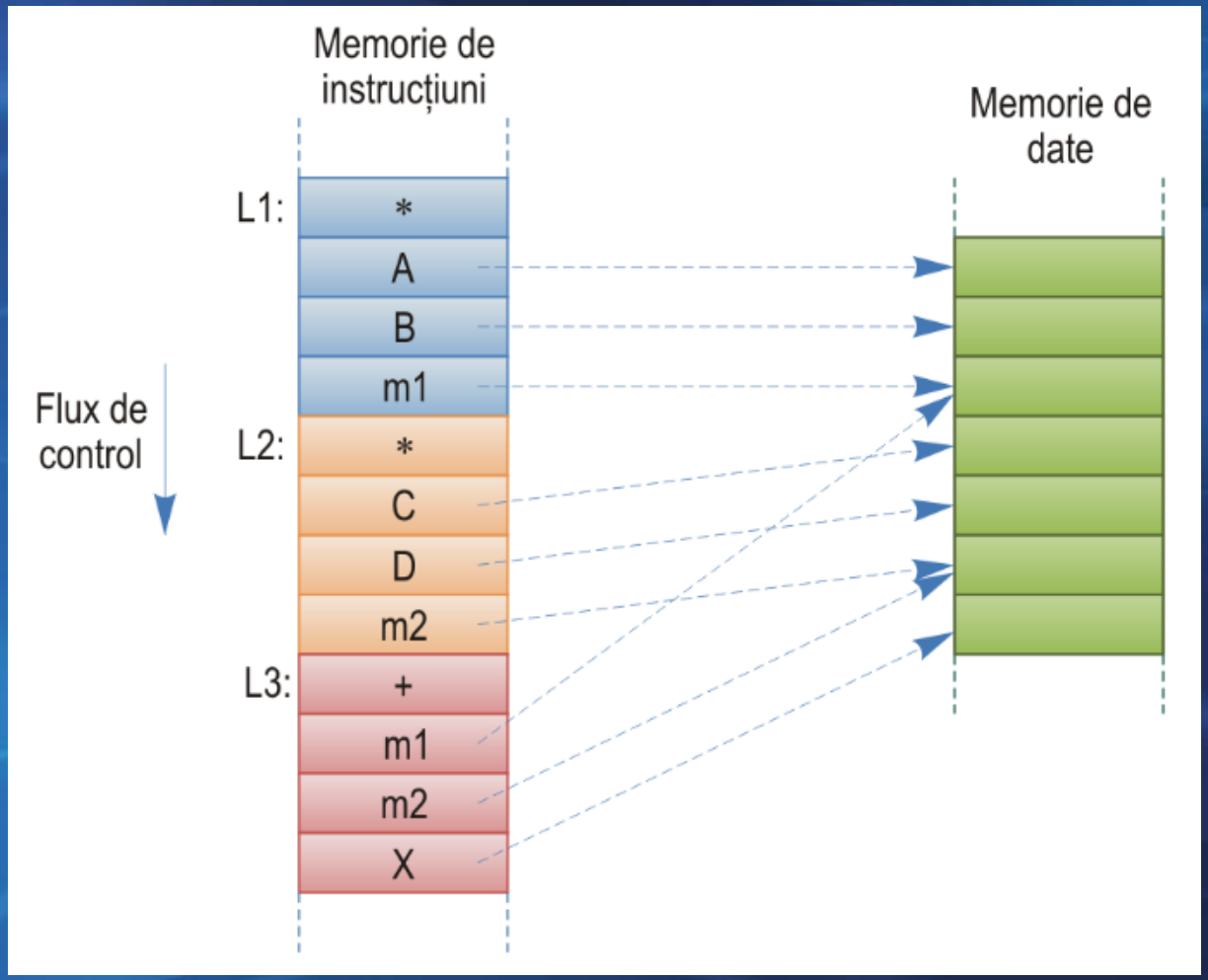
Modelul fluxului de control (1)

- Calculatoare convenționale: se bazează pe arhitectura von Neumann
  - Fluxurile de control și de date sunt separate
- Instrucțiunile sunt executate secvențial
  - Instrucțiuni de control (de ex., salturi): specifică abaterea de la ordinea secvențială
  - Implementarea: registru contor de program
  - Arhitectură cu flux de control: funcționarea este controlată de secvența de instrucțiuni

Modelul fluxului de control (2)

- Datele sunt memorate în locații de memorie sau registre
  - Fluxul de date: determinat de referințele la locațiile de memorie
  - Datele sunt încărcate și prelucrate numai atunci când instrucțiunile le solicită
  - Fluxul de date nu are efect asupra ordinii de execuție a instrucțiunilor
- Exemplu: Modelul fluxului de control pentru calculul $X = (A * B) + (C * D)$

Modelul fluxului de control (3)



Arhitecturi cu fire de execuție multiple

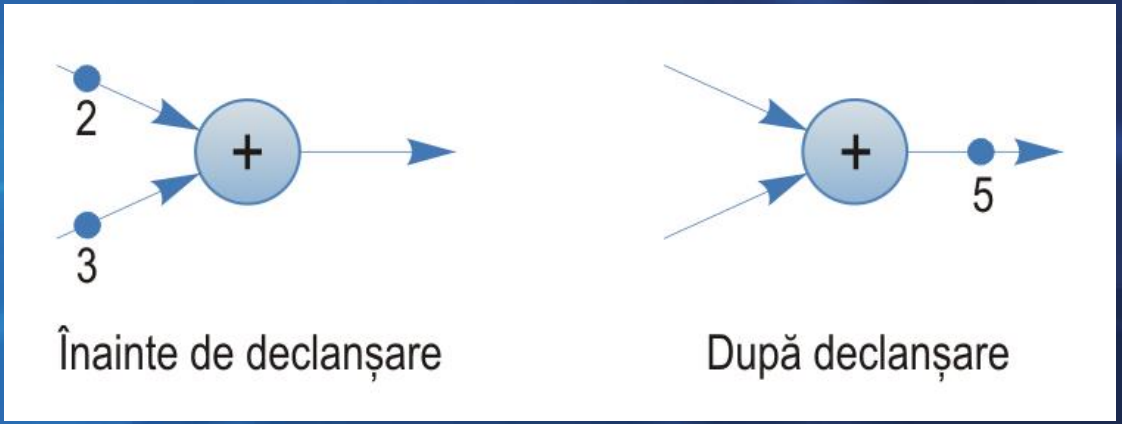
- Arhitecturi cu fire de execuție multiple
  - Modelul fluxului de control
  - Modelul fluxului de date
  - Arhitecturi cu flux de date
  - Exemple de calculatoare cu flux de date

Modelul fluxului de date (1)

- Secvența operațiilor nu este specificată
  - Nu există contor de program și nici conceptul clasic de variabile
  - O instrucțiune este gata pentru execuție atunci când operanzii devin disponibili
  - Arhitectură cu flux de date (*dataflow*): este controlată de date
  - Gradul de paralelism este ridicat
- Graful fluxului de date
  - Noduri: operatori (procesoare)

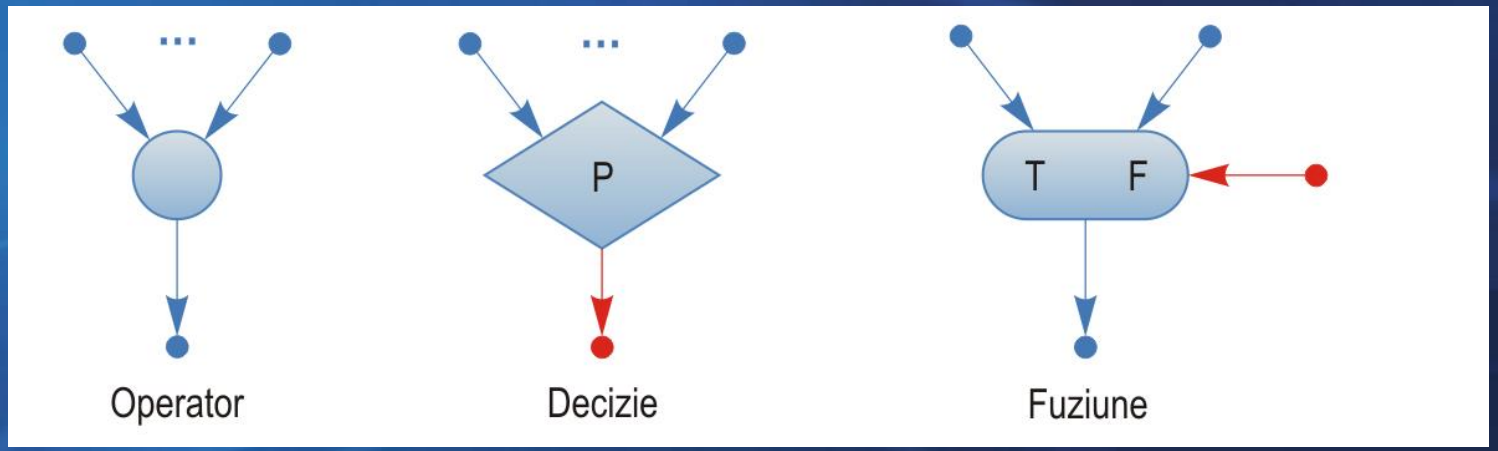
Modelul fluxului de date (2)

- Arce: căi pentru date sau valori de control
- Declanșarea (execuția operației) unui nod: sunt disponibile datele pe arcele de intrare
- Disponibilitatea datelor pe un arc este indicată printr-un simbol (*token*)



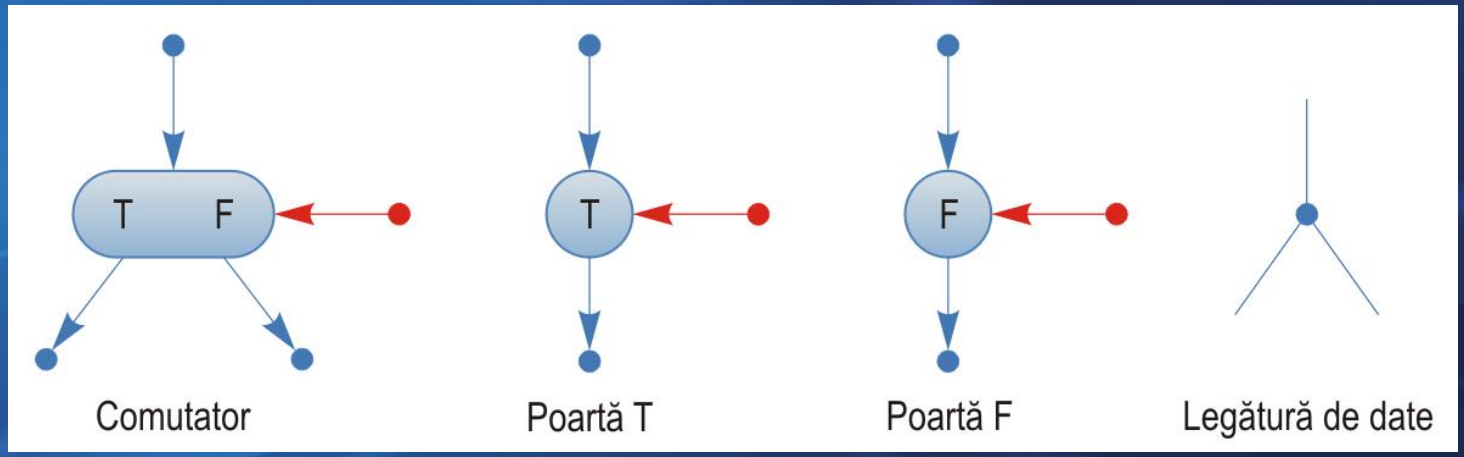
Modelul fluxului de date (3)

- Două tipuri de simboluri: de date; booleene
- Nod operator
- Nod de decizie: generează un simbol boolean
- Nod de fuziune: plasează unul din simbolurile de intrare pe arcul de ieșire



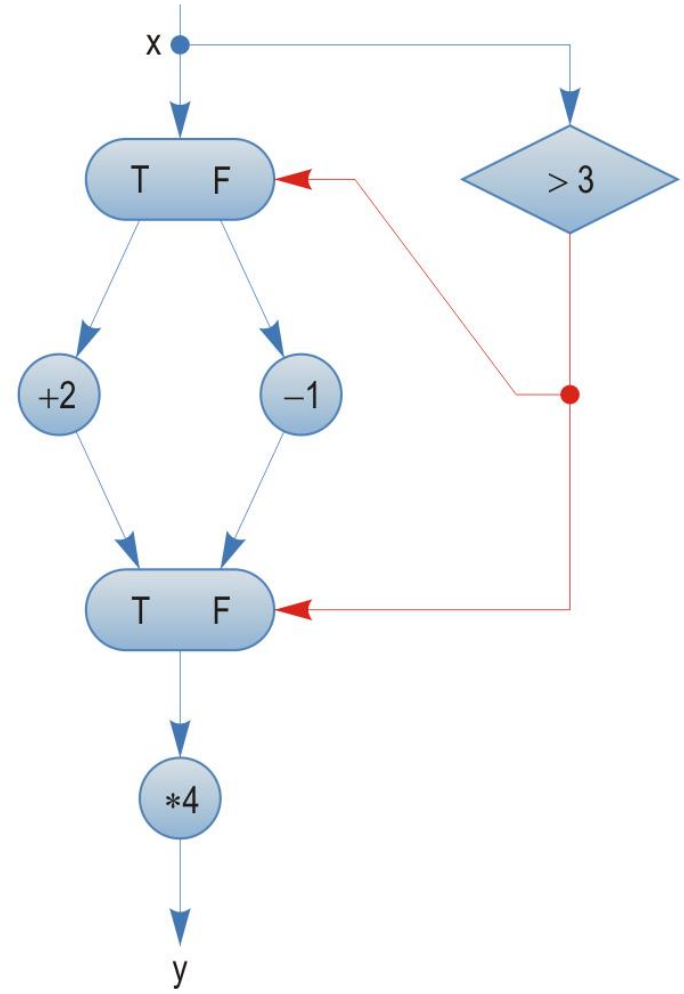
Modelul fluxului de date (4)

- Nod comutator: plasează simbolul de intrare pe unul din arcele de ieșire
- Poartă T: plasează simbolul de intrare la ieșire dacă simbolul boolean este TRUE
- Poartă F: idem, dacă simbolul boolean este F



Modelul fluxului de date (5)

```
if (x > 3) {
   y = (x + 2) * 4;
}
else {
   y = (x - 1) * 4;
}
```



Modelul fluxului de date (6)

- Modelul static al fluxului de date
  - Declanșarea: numai dacă fiecare arc de intrare conține un simbol; arcele de ieșire nu conțin simboluri
  - Este necesară o confirmare de la nodul urm.
- Modelul dinamic al fluxului de date
  - Lipsa simbolurilor la ieșire nu este necesară
  - Un arc poate conține mai multe simboluri
  - Un simbol trebuie asociat cu un set de date

Arhitecturi cu fire de execuție multiple

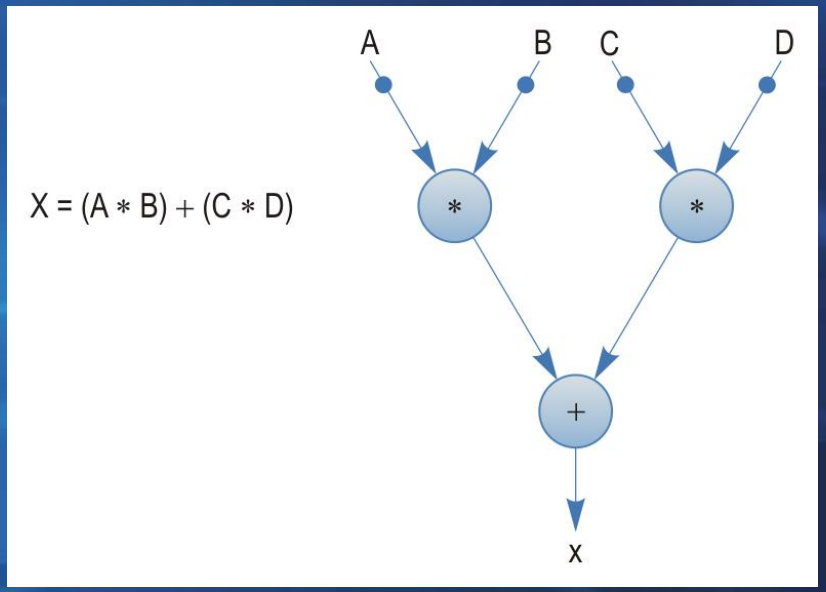
- Arhitecturi cu fire de execuție multiple
  - Modelul fluxului de control
  - Modelul fluxului de date
  - Arhitecturi cu flux de date
  - Exemple de calculatoare cu flux de date

Arhitecturi cu flux de date (1)

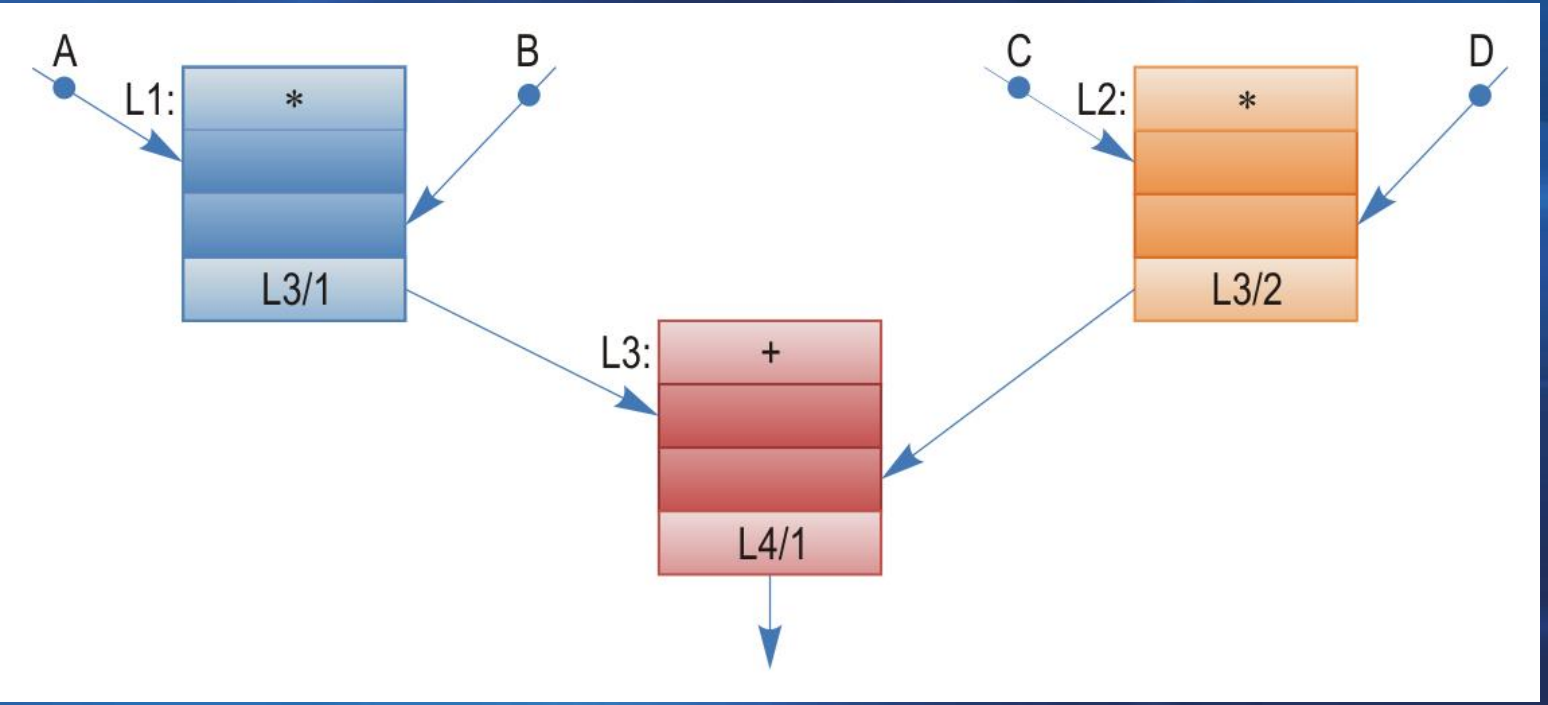
- Instrucțiunile cu flux de date nu adresează variabile → conțin variabilele utilizate
- Execuția instrucțiunilor nu afectează alte instrucțiuni gata pentru execuție
- Arhitectură statică
  - Validarea unei instrucțiuni: atunci când operanzii sunt recepționați și o altă instrucțiune așteaptă rezultatul
  - Constrângere impusă prin semnale de confirmare

Arhitecturi cu flux de date (2)

- Fiecare arc din graful fluxului de date poate conține cel mult un simbol
- Exemplu de graf al fluxului de date într-o arhitectură statică



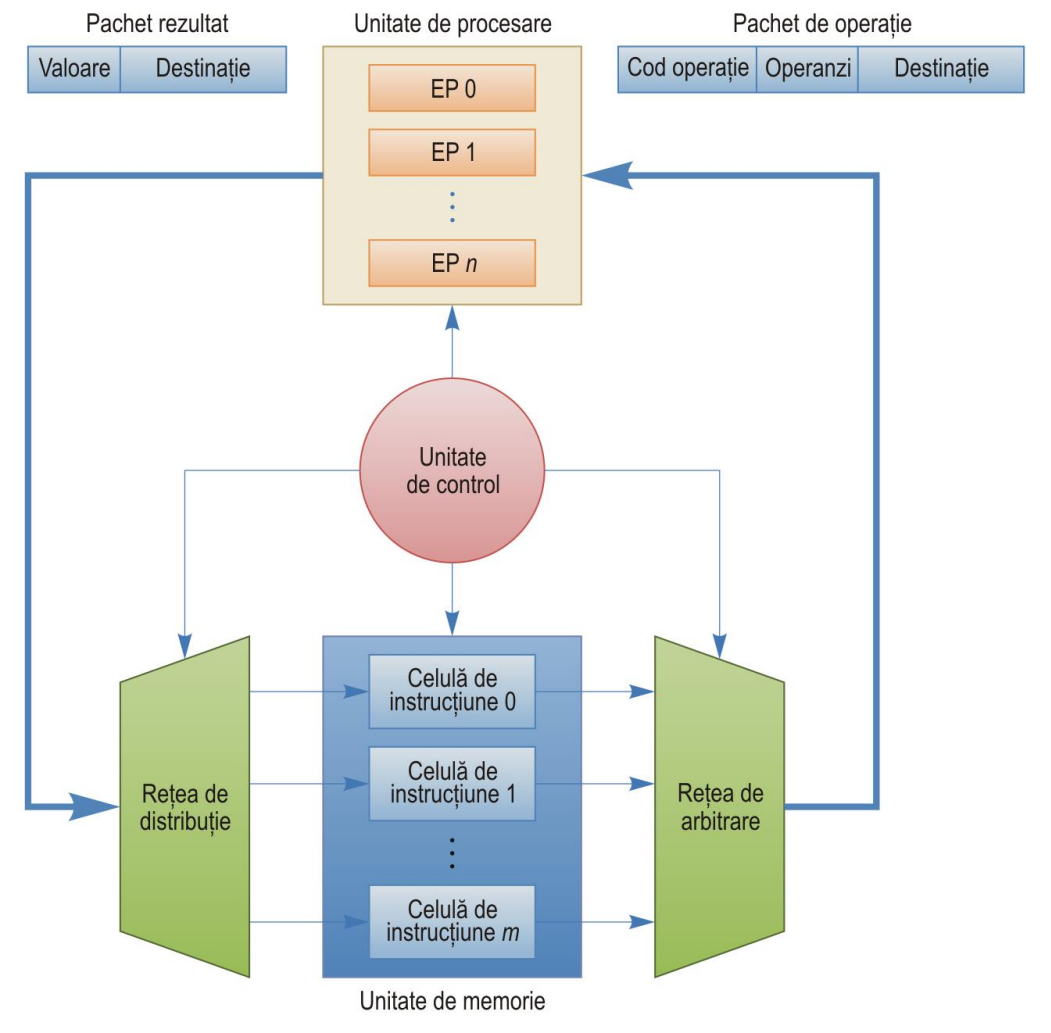
Arhitecturi cu flux de date (3)



Arhitecturi cu flux de date (4)

- Structura calculatorului static cu flux de date MIT
  - Unitatea de procesare: EP multiple
  - Unitatea de memorie: celule de instrucțiuni
  - Celulă de instrucțiune: conține reprezentarea unui nod al grafului → șablon de activitate
    - Codul operației, simboluri (date) de intrare, pointeri la celulele destinație
  - Pachet de operație: o instrucțiune validată
  - Pachet rezultat: valoare, adresă destinație

Arhitecturi cu flux de date (5)

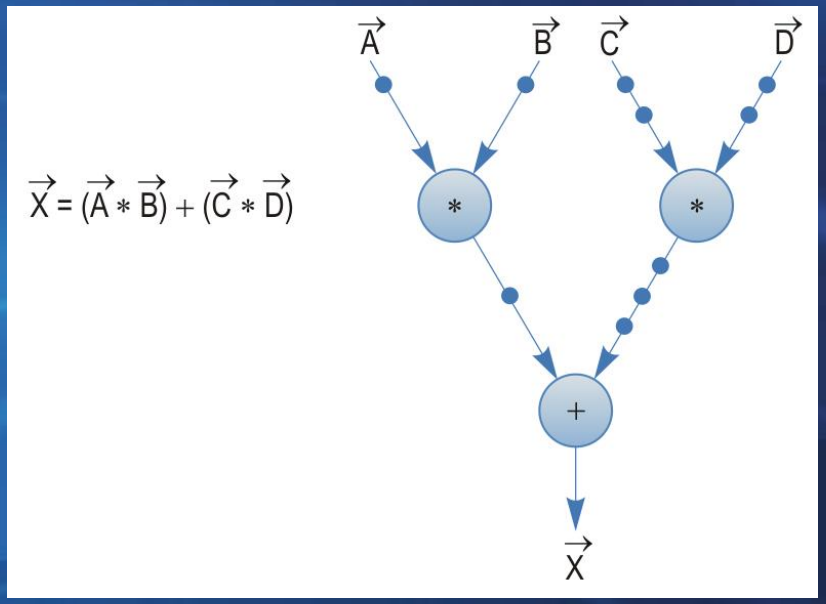


Arhitecturi cu flux de date (6)

- Arhitectură dinamică
  - Validarea unei instrucțiuni: atunci când operanzii sunt recepționați
  - Pot deveni disponibile simultan mai multe seturi de operanzi ale instrucțiunii
  - Comparativ cu arhitecturile statice: permit un grad mai ridicat de paralelism
  - Este necesar un mecanism pentru a distinge diferitele seturi de operanzi pentru o instrucțiune

Arhitecturi cu flux de date (7)

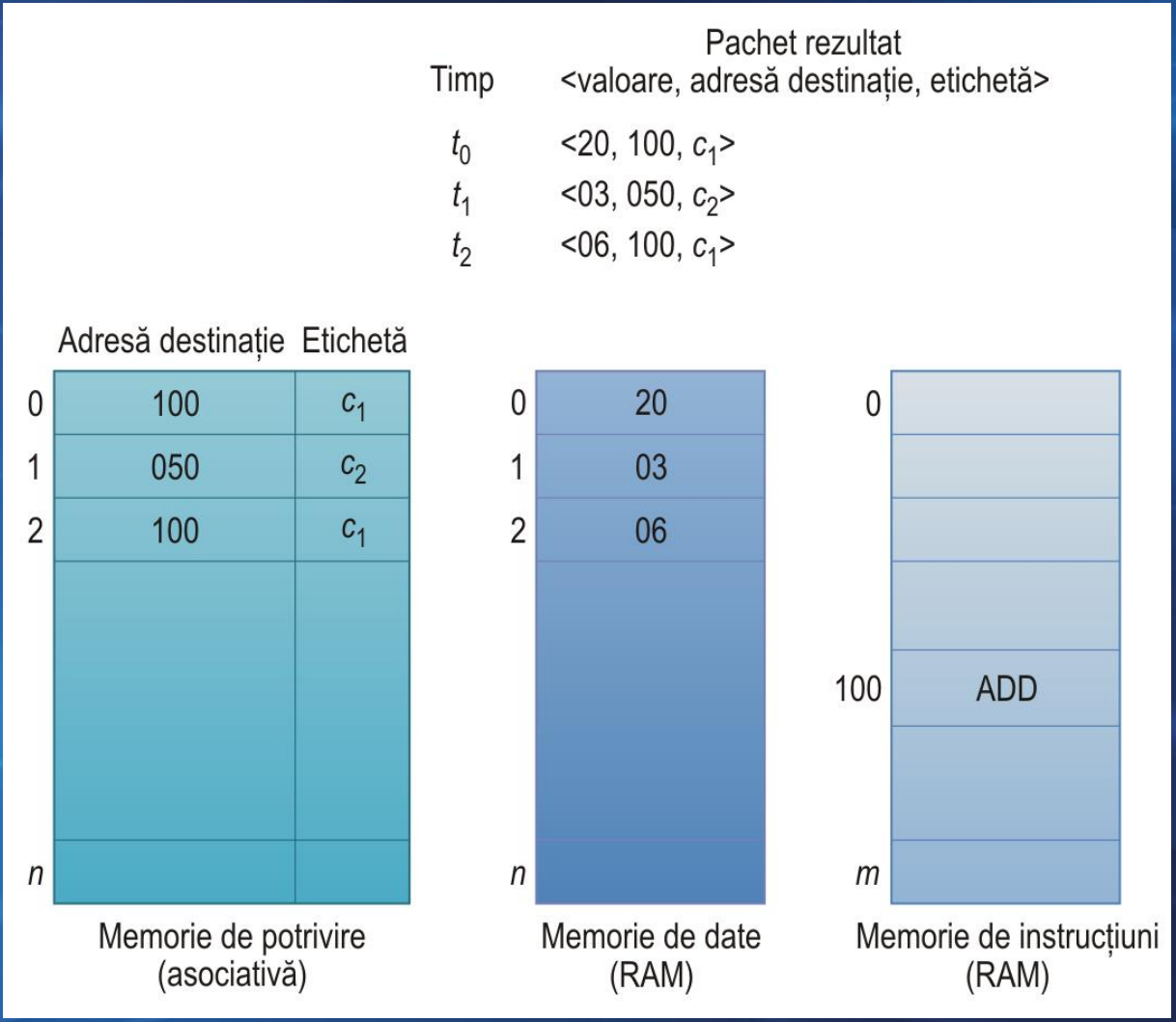
- Un arc din graful fluxului de date poate conține mai mult de un simbol
- Exemplu de graf al fluxului de date într-o arhitectură dinamică



Arhitecturi cu flux de date (8)

- Diferențierea seturilor de operanzi într-o arhitectură dinamică
  - Câmp suplimentar adăugat în pachetul rezultatului → etichetă
  - Se compară etichetele
  - Se poate utiliza o memorie asociativă → memorie de potrivire
  - Determinarea instrucțiunii validate: pe baza adresei destinației și a etichetei din pachetul rezultat

Arhitecturi cu flux de date (9)



Arhitecturi cu fire de execuție multiple

- Arhitecturi cu fire de execuție multiple
  - Modelul fluxului de control
  - Modelul fluxului de date
  - Arhitecturi cu flux de date
  - Exemple de calculatoare cu flux de date

Exemple de calculatoare cu flux de date (1)

- Calculatoarele cu flux de date nu au avut un succes comercial
  - Complexitatea hardware semnificativă
  - Dificultatea transmiterii eficiente a simbolurilor de date
  - Necesită memorii asociative de dimensiuni mari $\rightarrow$ păstrarea dependențelor de date
  - Dificultatea dezvoltării și compilării limbajelor cu flux de date

Exemple de calculatoare cu flux de date (2)

- Concepte ale fluxului de date incluse în procesoare specializate: DSP, GPU
- Procesoare superscalare
  - Permit execuția instrucțiunilor într-o ordine diferită de cea din program (*out-of-order*)
  - Formă restrânsă de flux de date → flux local
  - Fereastră de execuție: instrucțiunile sunt executate când operanzii sunt disponibili
  - O fereastră de execuție: 32 .. 200 instrucțiuni

Exemple de calculatoare cu flux de date (3)

- Calculatorul dinamic Manchester
  - MDM – *Manchester Dataflow Machine*
- Multiprocesorul Monsoon (MIT, Motorola)
  - Utilizează limbajul paralel Id
- Calculatorul EDDEN (Sanyo Electric)
  - EDDEN – *Enhanced Data-Driven Engine*
  - Versiunea comercială: Cyberflow/64
- Calculatoare hibride: flux de date + flux de control

Rezumat

- Arhitecturile cu flux de date nu conțin un contor de program, ci sunt controlate de date
  - Două tipuri de arhitecturi cu flux de date: statice și dinamice
  - Arhitecturile dinamice trebuie să conțină un mecanism pentru a distinge între ele diferitele seturi de operanzi ai unei instrucțiuni
  - Diferențierea se poate realiza cu un câmp de etichetă adăugat la pachetul rezultatului
  - Procesoarele superscalare utilizează o formă restrânsă de flux de date

Noțiuni, cunoștințe

- Prezentare generală a arhitecturilor cu fire de execuție multiple
- Modelul fluxului de control
- Modelul fluxului de date
- Graful fluxului de date
- Modelul static și cel dinamic al fluxului de date
- Arhitectură statică cu flux de date
- Structura calculatorului cu flux de date MIT
- Arhitectură dinamică cu flux de date
- Diferențierea seturilor de operanzi într-o arhitectură dinamică cu flux de date