Procesare paralela

- Procesarea paralela presupune efectuarea de calcule simultane; probleme complexe sunt divizate in probleme mai simple si acestea sunt rezolvate in mod concurent (in paralel).
- Procesarea paralela – la diferite niveluri:
  - Bit;
  - Instructiune;
  - Date;
  - Task;
- Paralelismul – utilizat mai ales pana in urma cu cativa ani la calcul de inalta performanta; in ultimii ani, datorita limitarilor fizice care nu permit cresterea continua a frecventei microprocesoarelor paralelismul este utilizat tot mai intensiv in arhitectura microprocesoarelor, ca o alternativa la cresterea frecventei, in primul rand la procesoare multicore.
- Computerele paralele pot fi clasificate in conformitate cu nivelul la care hardware-ul suporta paralelism:
  - Un singur computer:
    - Procesoare multicore;
    - Multi-procesor: sistem cu mai multe procesoare;
  - Mai multe computere:
    - Clustere: conexiunea unui mare numar de computere;
    - Computere masiv paralele (MPP, MPPA);
    - Griduri: federalizarea sistemelor din mai multe locatii pentru realizarea unui calcul masiv paralel;

Procesare paralela

- Cresterea frecventei a fost, pana in 2004, principala metoda de crestere a vitezei de calcul;

- Formula pentru calculul puterii consumate intr-un microprocesor:

$$P = C \times V^2 \times F$$

P – puterea

C – capacitatea – proportionala cu numarul de tranzistori care basculeaza intr-un ciclu

V – tensiunea

F – frecventa

- Legea lui Moore (numarul de tranzistoare se dubleaza fiecare 18-24 de luni) a ramas neschimbata si dupa 2004 (cand Intel a renuntat la realizarea de procesoare cu frecventa tot mai mare).
- Incepand cu 2004, acest numar aditional de tranzistori nefolositi pentru cresterea frecventei sunt utilizati pentru paralelizare.

Procesare paralela: Legea lui Amdahl

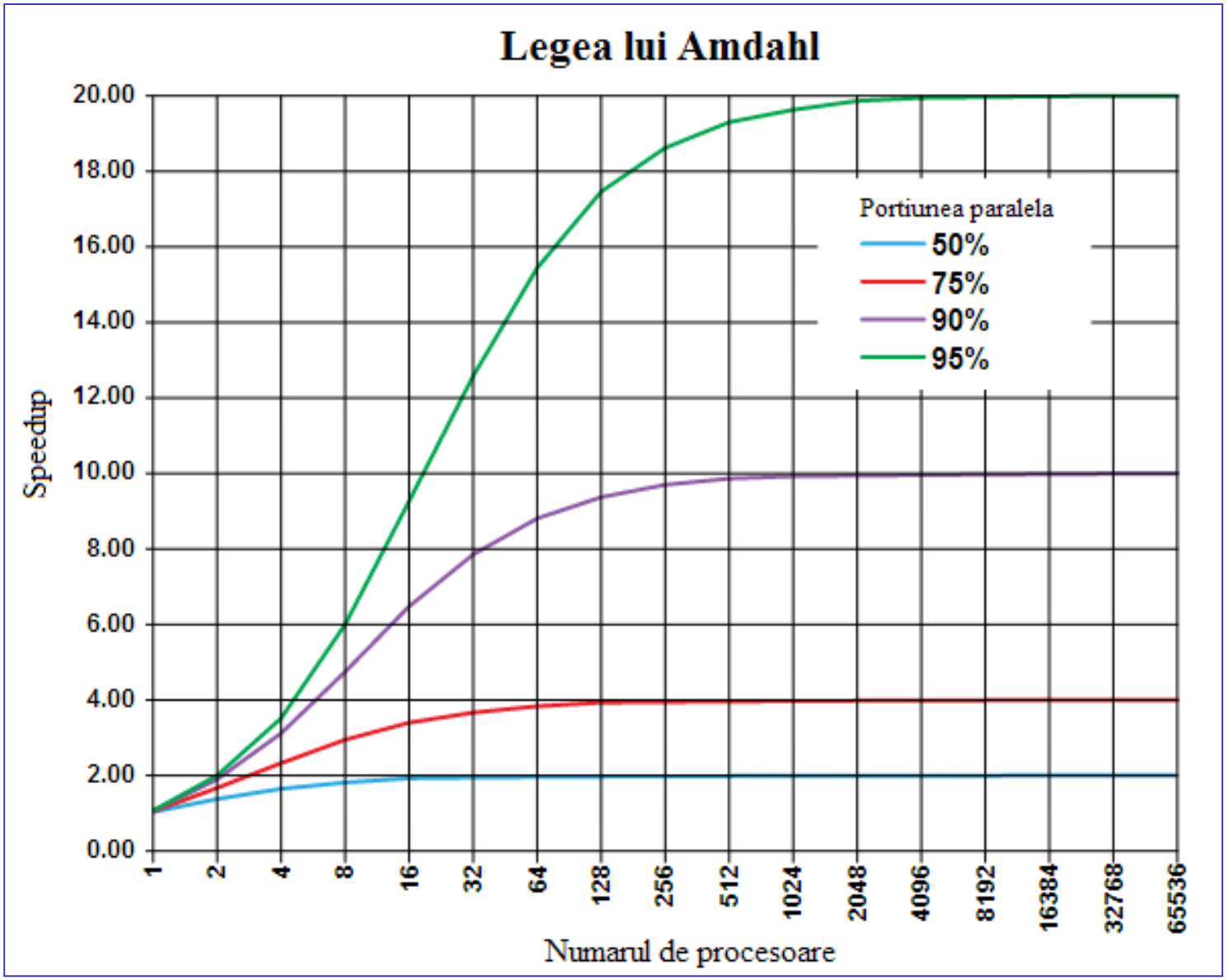
Legea lui Amdahl

- In mod ideal, daca un algoritm paralel ar putea sa fie scalat continuu, de fiecare data cand dublam numarul de procesoare, viteza de calcul ar trebui sa se dubleze;
- In realitate, majoritatea algoritmilor paraleli prezinta un speed-up liniar (in rare cazuri super-liniar) pentru un numar mic de procesoare si apoi cresterea vitezei se satureaza
- Legea lui ***Amdahl*** (approx. 1960): *o mica portiune dintr-un program, care nu poate fi paralelizata, limiteaza drastic scalabilitatea unui program paralel*.
- Daca α e acea fractiune din totalul calculelor care este folosit pentru calcule ne-paralele, cresterea maxima a vitezei e invers proportionala cu acea fractiune

$$S = \frac{1}{\alpha} \qquad = \lim_{P \to \infty} \frac{1}{\frac{1-\alpha}{P} + \alpha}$$

- Ex: daca 90% din program poate fi paralelizat, fractiunea care nu poate fi paralelizata va fi $\alpha = 0.1$. In consecinta, speed-up-ul maxim este de 1/0.1 = 10, indiferent de numarul de procesoare folosit.

Procesare paralela: Legea lui Amdahl



Procesare paralela: Legea lui Gustafson

Legea lui Gustafson

- Cresterea vitezei (S – speed-up) e definita in mod diferit decat in legea lui Amdahl. Gustafson a observat ca, odata cu cresterea numarului de procesoare, se mareste si dimensiunea problemei. El a introdus conceptul de crestere scalata a vitezei (scaled speedup). In consecinta, cresterea vitezei va fi proportionala (in formula lui Gustafson, cu numarul de procesoare si cu (1 - α), unde α este procentul din efortul total de calcul initial care nu poate fi paralelizat.

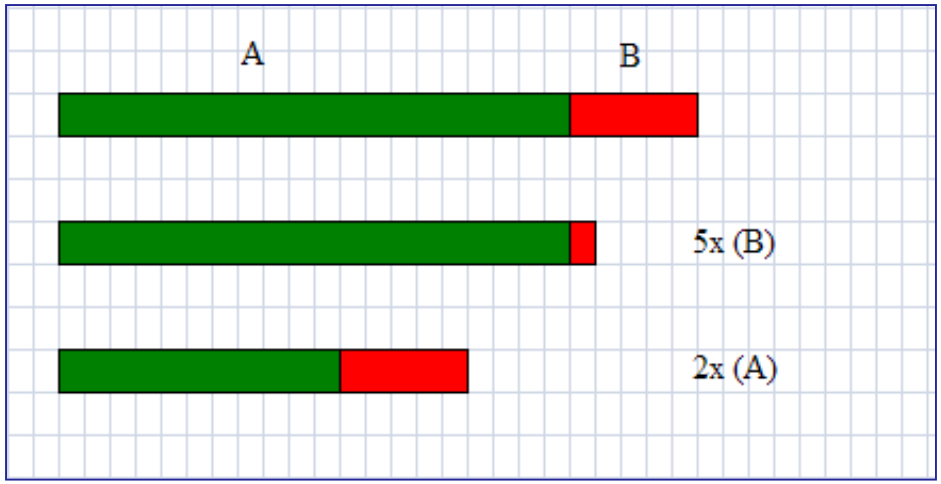
$$S(P) = P - \alpha(P - 1) \qquad = \alpha + P(1 - \alpha).$$

- Legea lui Gustafson furnizeaza alternativa optimista la legea lui Amdahl; pentru valori destul de mici ale lui α, speed-up-ul este chiar proportional cu numarul de procesoare; in practica, e posibil chiar ca odata cu numarul de procesoare, valoare lui α poate sa scada.

Legea lui Amdahl: problema e de dimensiune fixa si deci si cantitatea de lucru de facut in paralel nu depinde de numarul de procesoare;

Legea lui Gustafson: problema e de dimensiune crescatoare si deci cantitatea totala de lucru e proportionala cu numarul de procesoare.

Procesare paralela



Marind viteza de procesare pentru (A) doar de 2x obtinem o viteza de procesare mai mare decat daca marim viteza de procesare pentru (B) de 5x. In consecinta programatorii vor prefera probabil sa se concentreze asupra paralelizarii portiunii (A), chiar daca speedup-ul pentru aceasta e mai mic (pentru ca ponderea e mai mare).

Modele de consistenta a memoriei

Consistenta secventiala (Leslie Lamport): un sistem paralel produce aceleasi rezultate ca echivalentul sau secvential

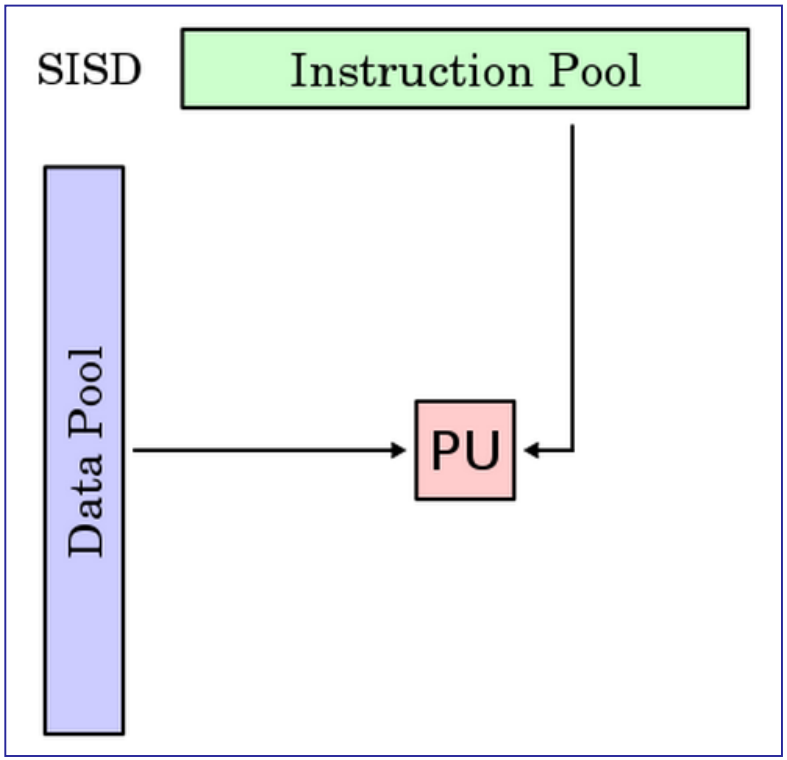
Consistenta software tranzactionala: concept/model de consistenta imprumutat de la bazele de date, implica notiunea de tranzactii atomice si se aplica in special la accesul concurent la memorie

Procesare paralela: taxonomia lui Flynn

	Single instruction	Multiple instruction
Single data	**SISD**	**MISD**
Multiple data	**SIMD**	**MIMD**

SISD – Single instruction Single data

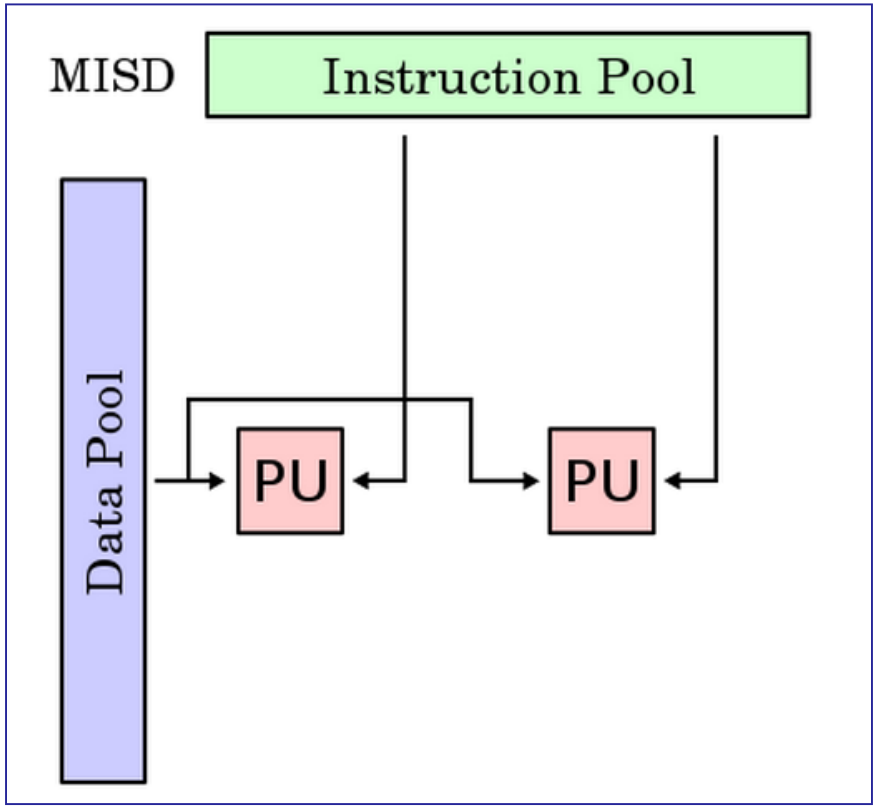
Un singur microprocesor executa un singur set de instructiuni pentru a opera asupra unui unic set de date. Acest model corespunde arhitecturii von Neumann. SISD pot avea caracteristici paralele. Prin operatiuni de preluare de instructiuni (instruction fetching) si executie in pipeline (pipelined execution) se poate mari viteza de procesare.



Procesare paralela: taxonomia lui Flynn

MISD – Multiple instruction Single data

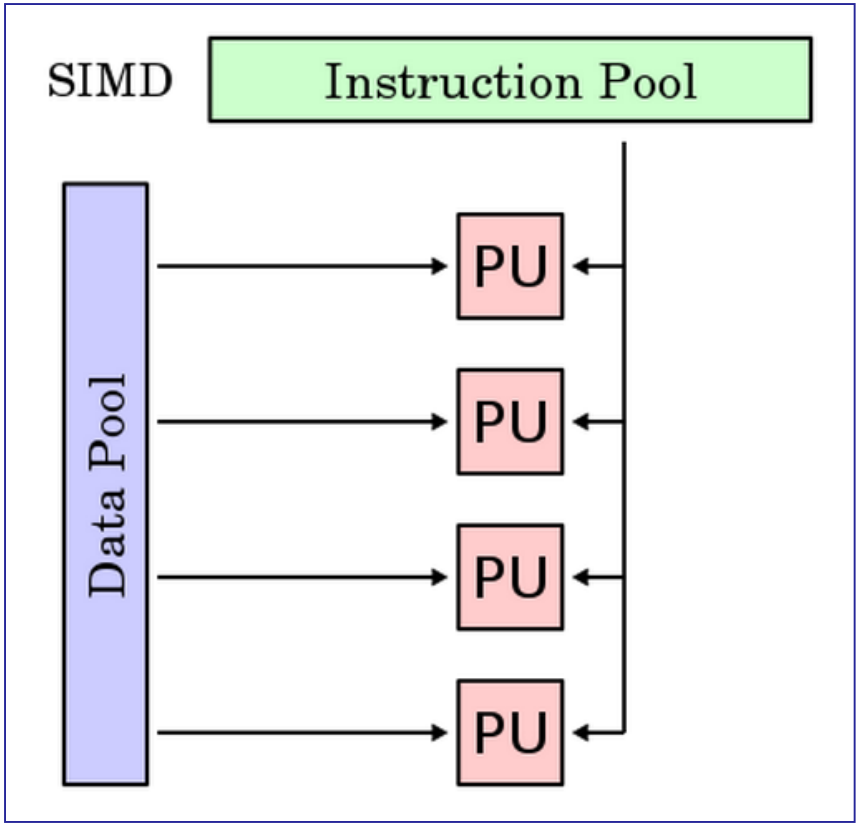
Mai multe unitati de procesare efectueaza instructiuni diferite pe acelasi set de date. Executia in pipeline poate fi clasificata de asemenea ca MISD (in unele analize se considera ca dupa executia unui set de instructiuni asupra setului de date, datele nu mai sunt aceleasi). Un exemplu de implementare practica a acestei paradigme sunt computerele utilizate pentru controlul de zbor pentru programul NASA al navetelor spatiale.



Procesare paralela: taxonomia lui Flynn

SIMD – Single instruction Multiple data

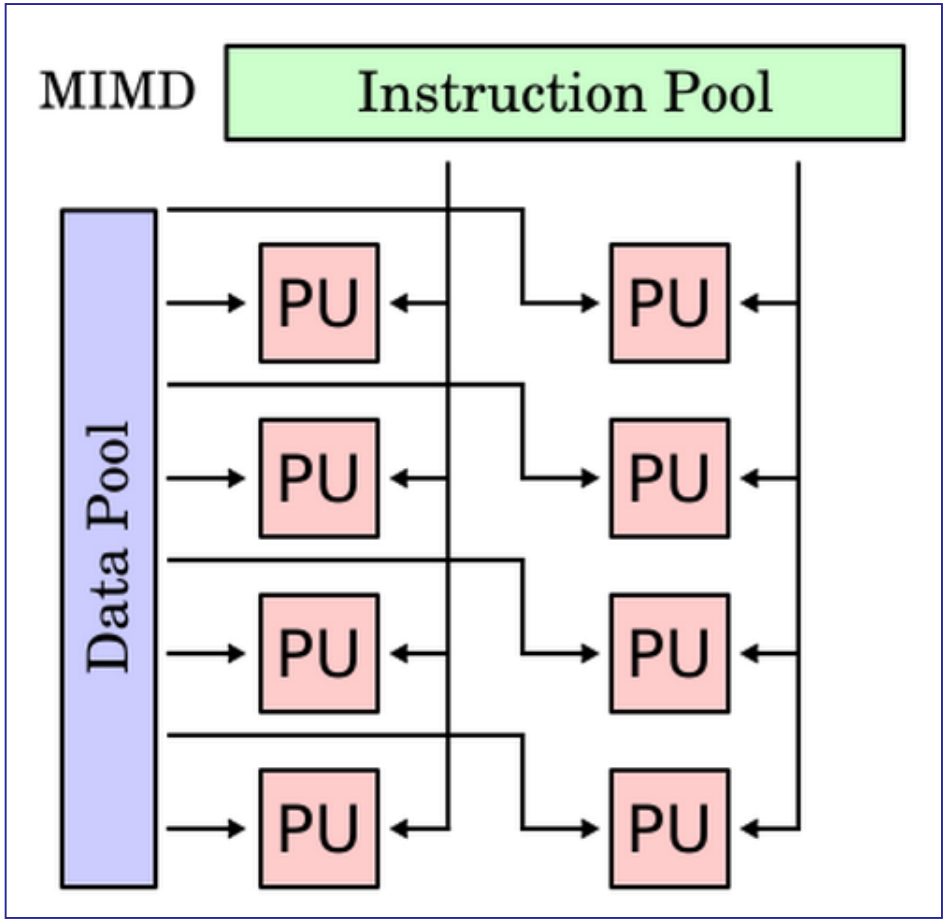
Descrie computere care executa aceleasi seturi de instructiuni pe mai multe seturi de date. Aplicatii tipice: pentru operatii de tip multimedia, pentru aplicarea de filtre imaginilor, pentru ajustarea volumului sunetului.
Primele utilizari ale arhitecturii SIMD au fost in ***computerele vectoriale*** (acestea executa aceleasi seturi de instructiuni asupra unor seturi multiple de date).
Ex. foarte cunoscut: **Cray 1** (1970-1980).
Avantaje: procesare simultana (si identica) a n unitati de memorie (de ex., n pixeli).
Dezavantaje: operatii care implica control intensiv de flux de instructiuni nu pot fi implementate foarte eficient.



Procesare paralela: taxonomia lui Flynn

MIMD – Multiple instruction Multiple data

Arhitectura cu multiple procesoare care pot actiona asincron, ruland seturi de instructiuni independente. Aplicatii tipice: pentru proiectare asistata de computer, ca switch-uri de comunicatie. MIMD pot fi fie cu memorie partajata fie cu memorie distribuita. ***In modelul cu memorie partajata***, exista avantajul modelului de memorie mai simplu conceptual; sistemul de operare va asigura coerenta memoriei. ***In modelul cu memorie distribuita***, coerenta memoriei trebuie asigurata de aplicatie. Un model cartezian este frecvent folosit (reduce numarul de conexiuni intre unitati: 3D ori 2D)



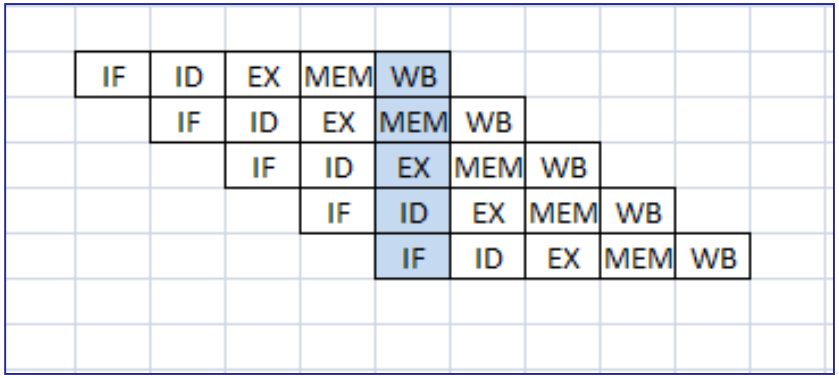
Procesare paralela

Procesare paralela la nivel de bit

Incepand cu perioada anilor 1970, marirea vitezei de procesare (independent de frecventa de operare) s-a facut in principal prin marirea lungimii cuvantului. Un cuvant mai lung va insemna un numar de instructiuni mai mic necesar pentru aceeasi procesare. Evolutia de la 4 la 8, 16 apoi la 32 de biti (standard pentru 20 de ani). Dupa 2003-2004 se introduc arhitecturile pe 64 de biti.

Procesare paralela la nivel de instructiune

Un program este in esenta o secventa de instructiuni executate de un procesor. Instructiunile pot fi reordonate in asa fel incat sa fie executate in paralel. Procesoarele moderne implementeaza un model de tip pipeline pe mai multe etape/pasi. Modelul clasic este procesorul de tip RISC. Pentium 4 implementeaza un model de pipeline cu 35 de pasi. In cazul in care nu exista dependente de date intre instructiuni, acestea poti fi executate in paralel.



IF – Instruction fetching (pregatirea instructiunii)
ID – Instruction decode (decodarea instructiunii)
EX – Execution (executie)
MEM – Memory access (acces memorie)
WB – Registry write back (scriere in registru)