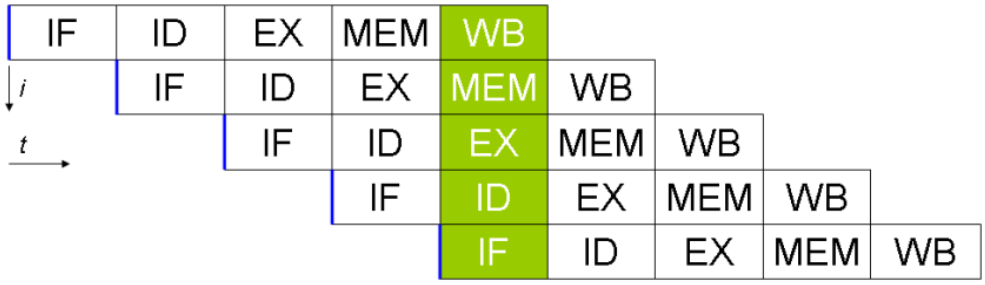
CURS 0x08

PIPELINING, BRANCH PREDICTION, OUT OF ORDER EXECUTION

Trei mecanisme pentru execuție cu viteză sporită
- **pipelining (conductă de date)**
- **branch prediction (predicția salturilor)**
- **out of order execution (execuția în ordine arbitrară)**

CPU execută instrucțiuni:
- **IF – Instruction Fetch**
- **ID – Instruction Decode**
    - **COA – Calculate Operand Address**
    - **FO – Fetch Operand**
    - **FOs – Fetch Operands**
- **EX – Execution**
- **MEM – Memory Access**
- **WB – Write Back**

PIPELINING



Imaginea arată bine, din păcate nu putem face așa ceva mereu:

1. **pipeline stalls (întârzieri în conducta de date)**
2. **aceste evenimente se numesc hazards (erori)**

- **structural hazards: o unitate de calcul este deja utilizată**
  - **două instrucțiuni încearcă să acceseze aceeași unitate**

- **data hazards: datele nu sunt pregătite pentru utilizare**
  - **o instrucțiune depinde de rezultatul unei instrucțiuni precedente**

  - **True Dependence (Read After Write – RAW)**

    add %ebx, %eax
    sub %eax, %ecx

  - **Anti-dependence (Write After Read – WAR)**

    add %ebx, %eax
    sub %ecx, %ebx

  - **Output Dependence (Write After Write – WAW)**

    mov $0x10, %eax
    mov $0x01, %eax

- **control hazards: nu știm următoarea instrucțiune**
  - **din cauza unor instrucțiuni de jump nu știm instrucțiunea următoare**

OUT OF ORDER EXECUTION

Ce se întâmplă în procesoarele moderne?
• se citesc câteva sute de instrucțiuni la un moment dat
• în hardware, se realizeaza un graf (data-flow graph) de dependențe între aceste instrucțiuni
• execută instrucțiunile în funcție de dependențe, ceea ce va rezulta un out of order execution

BRANCH PREDICTION

• predicție fixă: mereu sare / mereu nu sare
• predicția de la pasul anterior: ce a făcut instrucțiunea ultima dată
• predicția cu istoric: ce face de obicei instrucțiunea
• execuție speculativă (eager execution): calculează cu și fără salt