# MICROPROGRAMAREA

Microprogramarea este folosită pentru reducerea complexităţii proiectării controlului

Fiecare microinstrucţiune defineşte setul de semnale de control ale căii de date care trebuie activate într-o singură stare dată

Foarte importantă este succesiunea microinstrucţiunilor

Microinstrucţiunea poate fi privită ca o secvenţă de câmpuri ale căror funcţiuni sunt corelate.

***Cum definim formatul microinstrucţiunii ?***

Trebuie stabilite numărul de câmpuri dintr-o microinstrucţiune precum şi semnalele de control afectate de fiecare câmp.

1. Formatul microinstrucţiunii trebuie ales astfel încât să se simplifice reprezentarea.

2. Trebuie ca scrierea microinstrucţiunilor consistente să fie imposibilă

Semnalele care nu sunt activate simultan niciodată pot folosi în comun acelaşi câmp

Exemplu de împărţire pe câmpuri a unei microinstrucţiuni:

| | |
|---|---|
| ControlUAL | - specifică operaţia efectuată de UAL |
| SRC1 | - sursă operand 1 |
| SRC2 | - sursă operand 2 |
| CtrlReg | - scrire/citire a fişierului de registre şi sursa valorii pt scriere |
| Memorie | - scrierea/citirea şi sursa pentru memorie |
| CtrlScriePC | - specifică scrierea PC-ului |
| Secvenţă | - modalitatea de alegere a următoarei instr. |

## ALEGEREA INSTRUCŢIUNII URMĂTOARE

1. Incrementarea microinstrucţiunii curente – comportare secvenţială (mod implicit) – în câpul secveţă punem eticheta SEQ

2. Se transferă controlul la microinstrucţiunea care începe execuţia următoarei instrucţiuni – ciclu FETCH – în câpul secvenţă punem FETCH

3. Se alege următoarea microinstrucţiune pe baza intrării în unitatea de control – distribuţie

| Câmp | Valori pt câmp | Funcţie |
|---|---|---|
| Etichetă | Orice şir | Folosit pentru specificarea etichetelor în controlul secvenţelor de microcod. Etichetele terminate în 1 sau 2 sunt folosite pentru distribuţie cu tabel de salt, indexat cu cod operaţie |
| Control UAL | ADD | UAL adună |
| | Subst | UAL scade – implementeză comparaţia |
| | Func code | Foloseşte câmpul ***funct*** |

| Câmp | Valori pt câmp | Funcţie |
|---|---|---|
| SRC 1 | PC | PC intrare pentru UAL |
| | A | A folosit ca primă intrare în UAL |
| SRC 2 | B | B a doua intrare în UAL |
| | 4 | Constanta 4 ca a doua intrare în UAL |
| | Extend | Ieşirea unităţii de extindere semn ca a doua intrare în UAL |
| | Extshft | Foloseşte ieşirea unităţii deplasare cu 2 ca a doua intrare în UAL |
| Ctrl registru | Read | Citeşte 2 registre folosind pentru identificarea lor câpurile rs şi rt |
| | Write ALU | Scrie fişierul de registre folosind pt numărul registrului câmpul rt din IR, iar pentru dată conţinutul EşUAL |
| | Write MDR | Scrie fişierul de registre folosind pentru numărul registrului câmpul rt din IR iar pentru dată conţinutul MDR |
| Memorie | Read PC | Citeşte memoria având ca adresă PC-ul; scrie rezultatul în IR şi MDR |
| | Read ALU | Citeşte memoria având ca adresă EşUAL; scrie rezultat în MDR |
| | Write ALU | Scrie memoria având ca adresă EşUAL şi conţinutul lui B ca dată |

| Câmp | Valori pt câmp | Funcţie |
|---|---|---|
| Ctrl ScriePC | ALU | Scrie conţinutul lui UAL în PC |
| | ALUOut-cond | Dacă ieşirea UAL-Zero este activă, scrie în PC conţinutul registrului EşUAL |
| | Jump address | Scrie în PC adresa de salt a instrucţiunii |
| Secvenţă | seq | Alege secvenţa următoare în microinstrucţiune |
| | Fetch | Merge la prima microinstrucţiune pentru a începe o instrucţiune nouă |
| | Dispatch i | Distribuie adresa folosind ROM-ul specificat de i (1 sau 2) |

## Crearea microprogramului

Se vor eticheta instrucţiunile din microprogram cu etichete simbolice care pot fi folosite pentru specificarea conţinutului tablourilor de distribuţie.

Paşii 1 şi 2 din execuţia unei instrucţiuni:

Extragerea instrucţiunilor

Decodificarea instrucţiunilor şi calcularea PC-ului secvenţial cât şi a PC-ului obiectiv pentru ramificaţie

| Etch | Ctrl UAL | SRC1 | SRC2 | Ctrl Reg | Mem | Ctrl ScriePC | Secv |
|---|---|---|---|---|---|---|---|
| Fetch | Add | PC | 4 | | Read PC | UAL | Seq |
| | Add | PC | Extshft | Read | | | Dispatch 1 |

## Microprogramul pentru instrucţiunile de referire a memoriei

| Etch | Ctrl UAL | SRC1 | SRC2 | Ctrl Reg | Mem | Ctrl ScriePC | Secv |
|---|---|---|---|---|---|---|---|
| mem1 | Add | A | Extend | | | | Dispat ch 2 |
| lw2 | | | | | Read ALU | | Seq |
| | | | | Write MDR | | | Fetch |
| sw2 | | | | | Write ALU | | fetch |

## Microinstrucţiunile pentru instrucţiunile de tip R

| Etch | Ctrl UAL | SRC1 | SRC2 | Ctrl Reg | Mem | Ctrl ScriePC | Secv |
|---|---|---|---|---|---|---|---|
| Rformat 1 | Func code | A | B | | | | Seq |
| | | | | Write ALU | | | fetch |

**Instrucţiunea de ramificaţie**

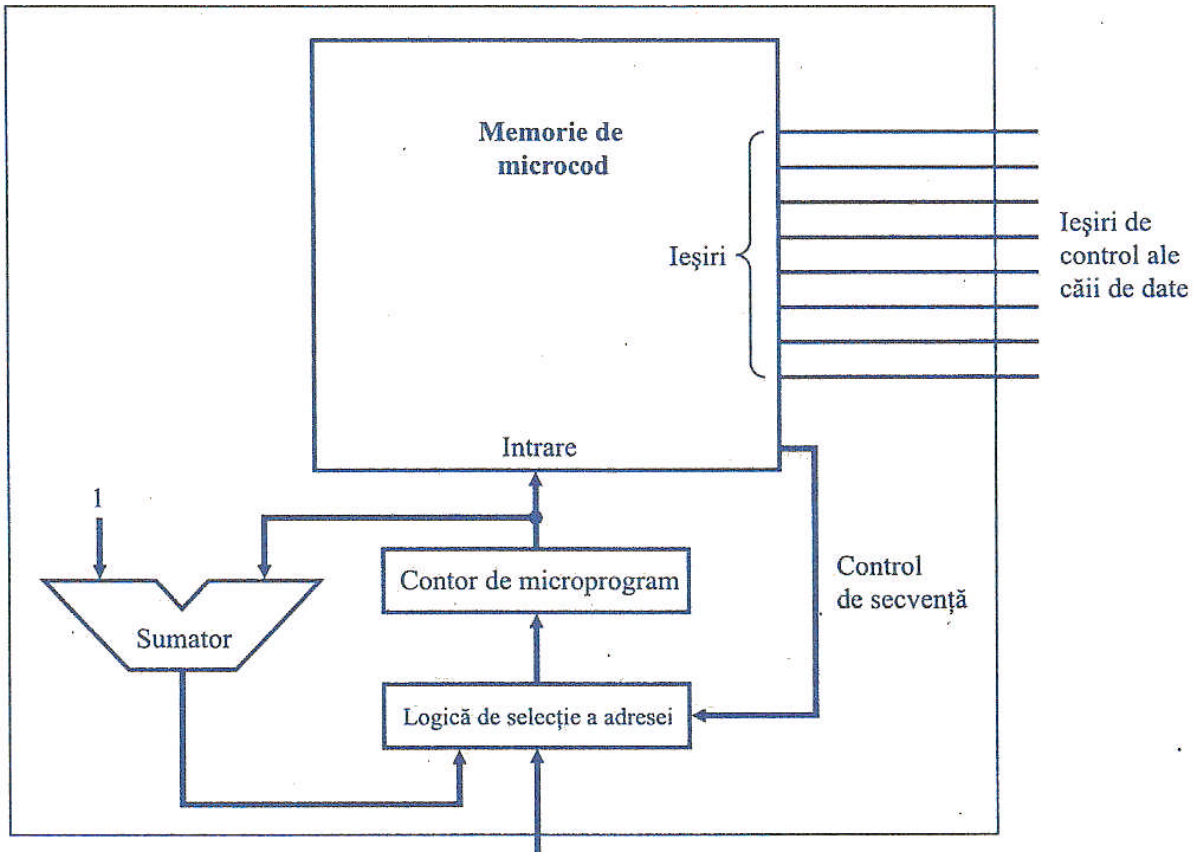
| Etch | Ctrl UAL | SRC1 | SRC2 | Ctrl Reg | Mem | Ctrl ScriePC | Secv |
|---|---|---|---|---|---|---|---|
| BEQ1 | Subt | A | B | | | ALUOut - cond | Fetch |

De remarcat că avem doar o singură microinstrucţiune. **DE CE ?**

**Instrucţiunea de salt**

| Etch | Ctrl UAL | SRC1 | SRC2 | Ctrl Reg | Mem | Ctrl ScriePC | Secv |
|---|---|---|---|---|---|---|---|
| JUMP1 | | | | | | JUMP address | Fetch |

## Implementarea controlului microcodului



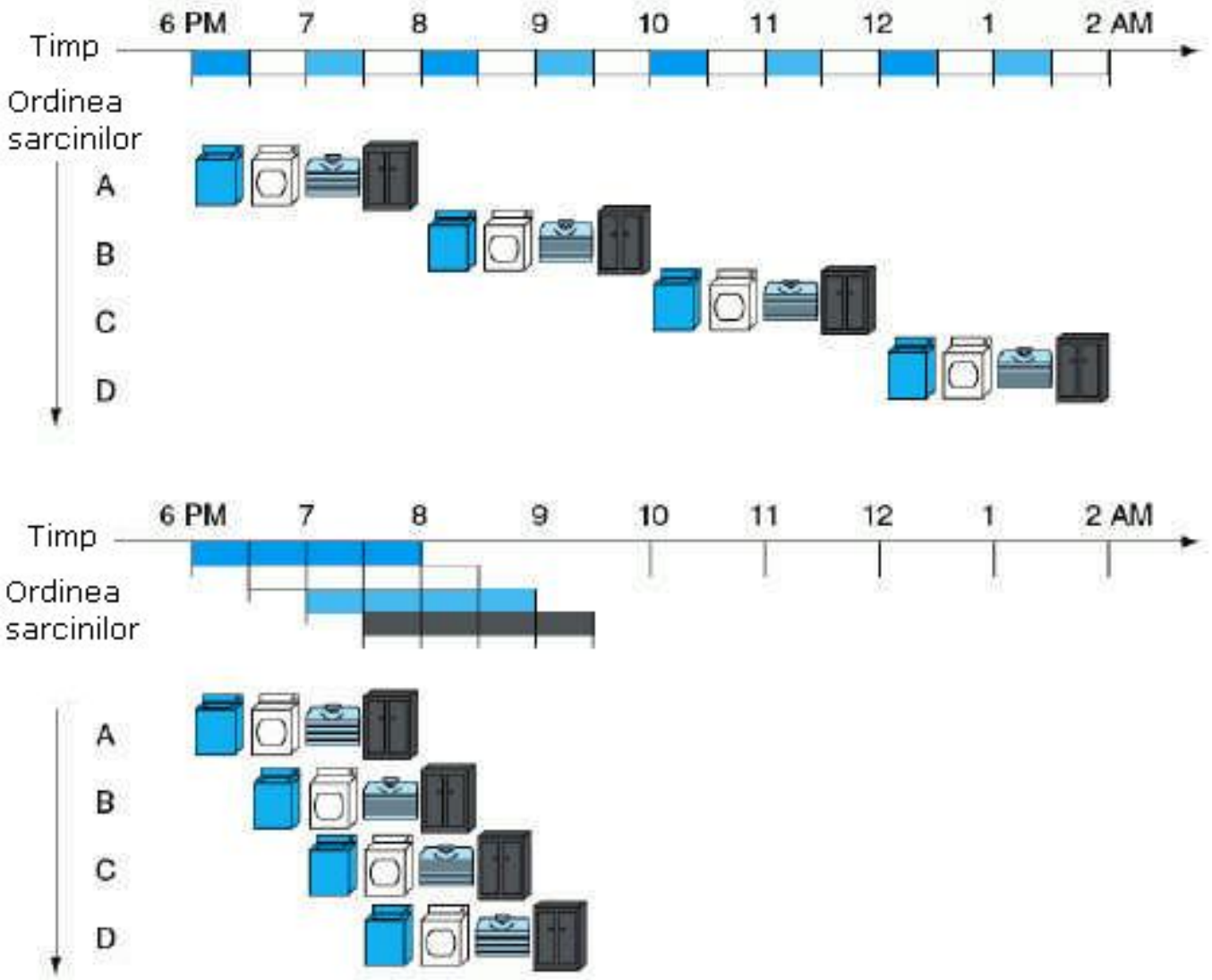

Unitate explicită pentru incrementare – calculează starea secvenţială următoare implicită

Memoria de microcod poate fi doar citită

# PIPELINE

## Introducere

Pipeline – execuţia mai multor instrucţiuni se suprapune în timp



Etajul 1 – extragerea instrucţiunii din memorie

Etajul 2 – citirea registrelor în timp ce instrucţiunea este decodificată

Etajul 3 – executarea operaţiei sau calcularea adresei

Etajul 4 – accesul la un operand din memorie

Etajul 5 – scrierea rezultatului într-un registru

Timpul între instrucţiuni $_{\text{pipeline}}$ = Timpul între instrucţiuni $_{\text{fără pipeline}}$ / nr de etaje

**Pipeline-ul îmbunătăţeşte performanţa prin creşterea productivităţii instrucţiunilor, nu prin micşorarea timpului de execuţie al unei instrucţiuni individuale**

## OBSERVAŢII

Instrucţiunile au aceeaşi lungime => extragerea instrucţiunilor şi decodificarea lor se poate face în doar două etaje de pipe.

Câpurile registrelor sursă sunt localizate în acelaşi loc în cadrul instrucţiunii => în etajul 2 putem începe citirea fişierului de registre în acelaşi timp în care hardware-ul determină instrucţiunea ce a fost extrasă.

Operanzii din memorie apar numai în instrucţiunile de încărcare şi memorare => putem utiliza etapa de execuţie pentru calcularea adresei de memorie, iar accesul la memorie se efectuează în etapa următoare.

Operanzii trebuie să fie aliniaţi în memorie => data poate fi transferată între procesor şi memorie într-o singură etapă pipeline.

## *HAZARDURI PIPELINE*

Evenimentul în urma căruia instrucţiunea următoare nu poate fi executată în următorul ciclu de ceas se numeşte ***hazard***.

### HAZARD SRUCTURAL

Hardware-ul nu poate suporta combinaţia de instrucţiuni pe care dorim să le executăm în acelaşi ciclu de ceas.

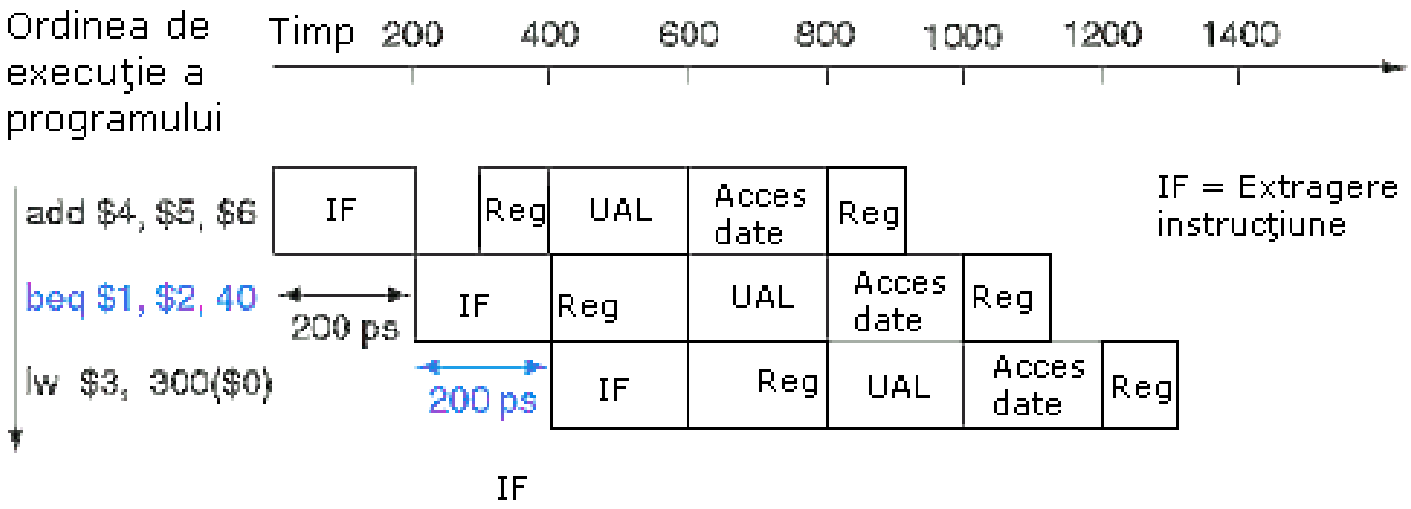
### HAZARD DE CONTROL

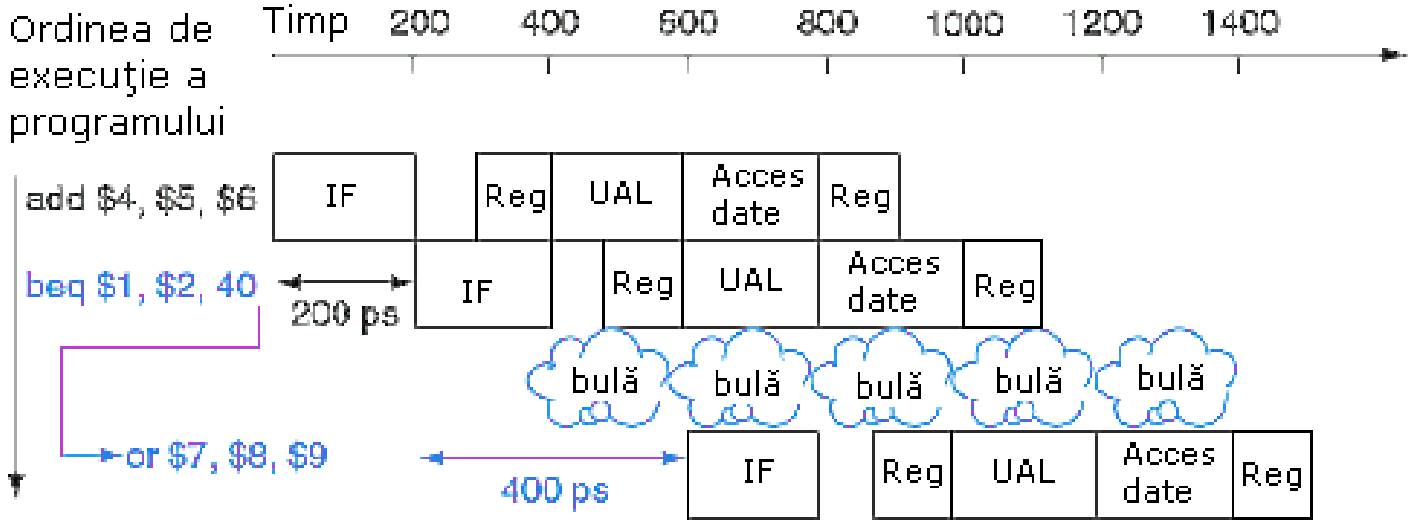
Apare din necesitatea de a lua o decizie pe baza rezultatelor unei instrucţiuni, în timp ce altele sunt în execuţie

#### SOLUŢII

1. Staţionarea – primul ciclu se va executa secvenţial şi apoi se va trece la execuţia pipeline – soluţie viabilă în practică dar prea lentă în cazul pipe-urilor lungi

2. Predicţia – se presupune că ramificaţiile nu vor reuşi => că doar în cazul reuşitelor pipe-ul va staţiona.



## Cursul următor – pipeline superscalar şi pipeline dinamic

### *CONCLUZII*

1. Pipeline-ul creşte numărul instrucţiunilor executate simultan şi viteza cu care instrucţiunile sunt începute şi terminate

2. Execuţia pipeline nu reduce timpul necesar derulării unei instrucţiuni individuale