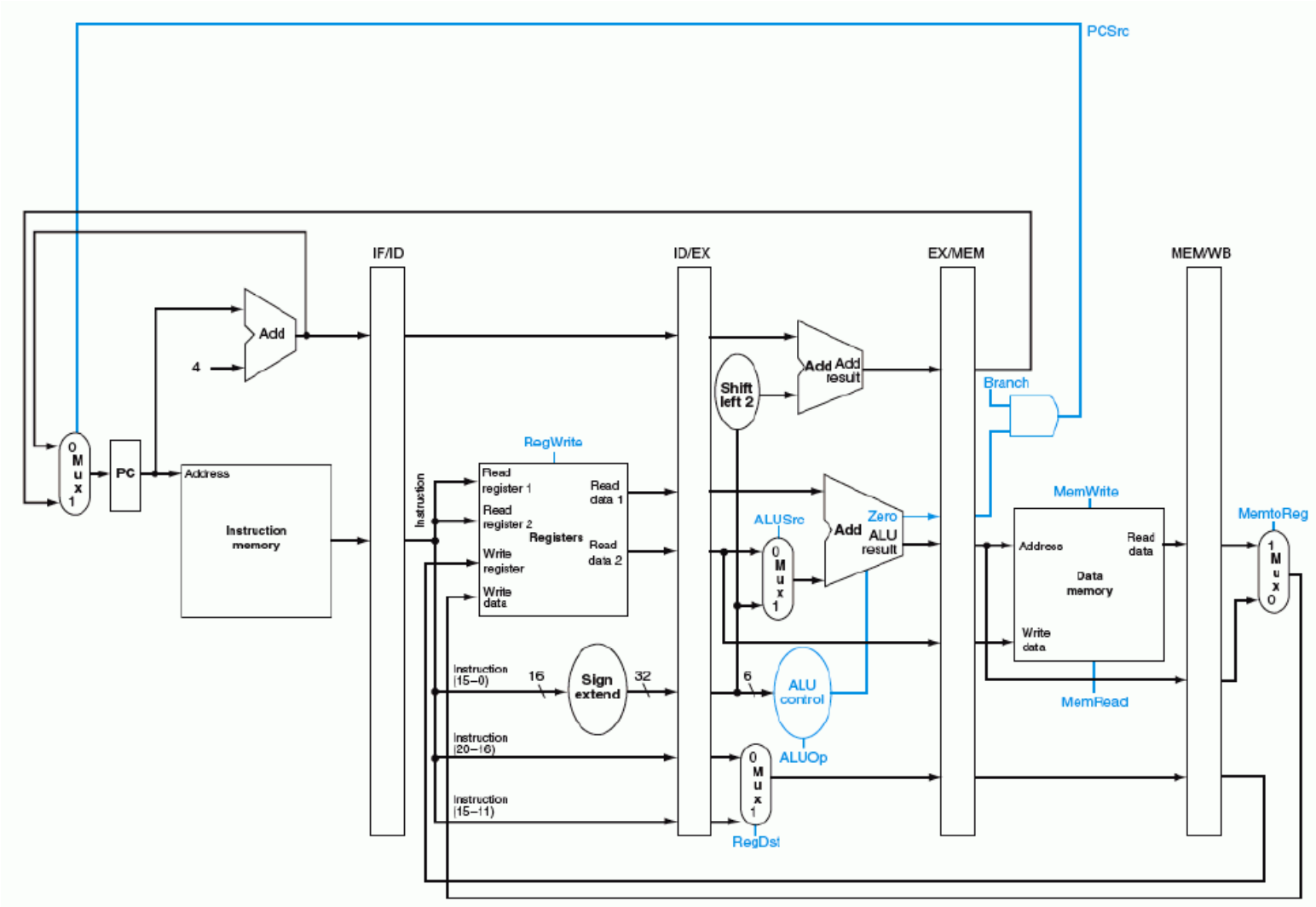
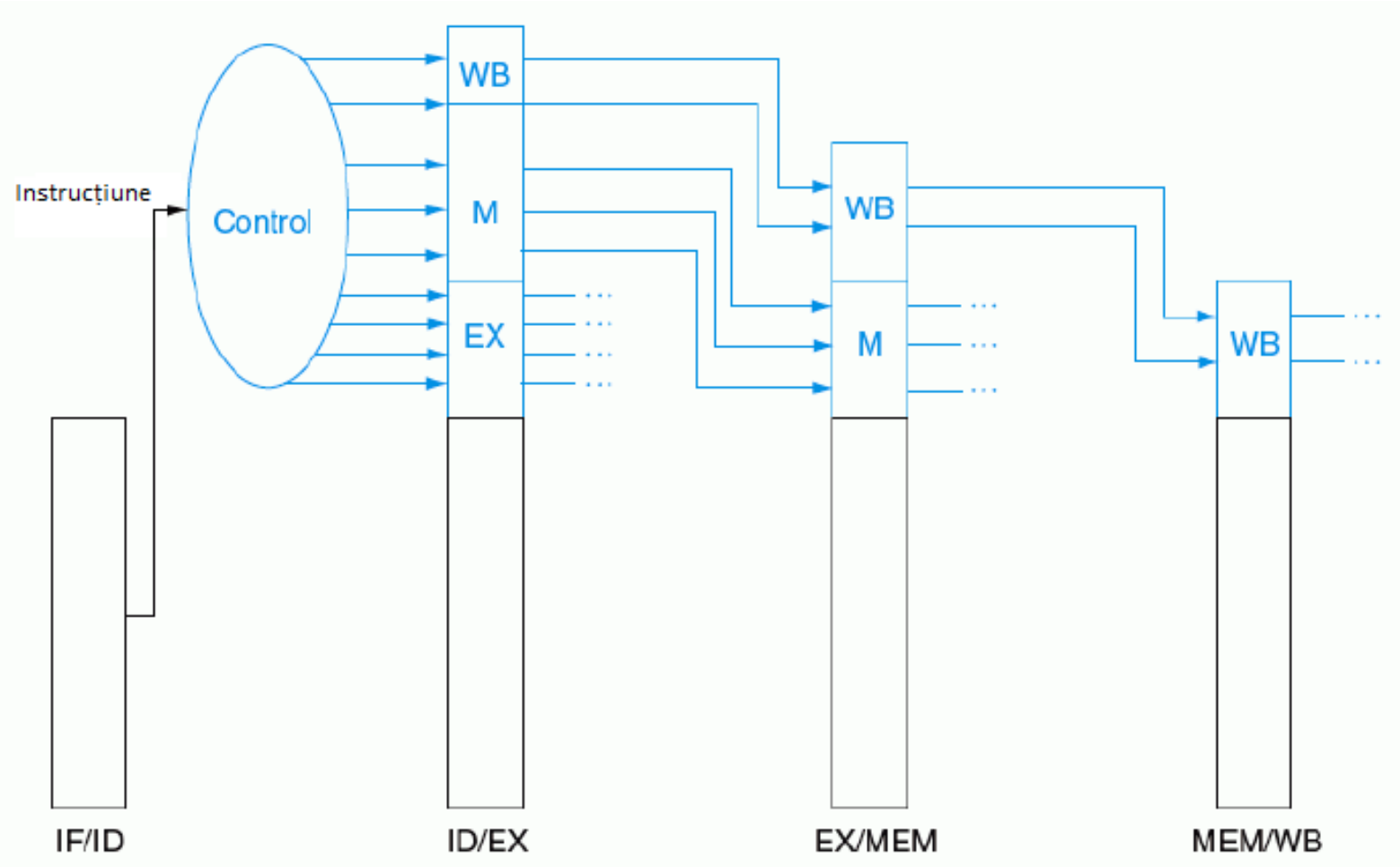
CONTROLUL PENTRU PIPELINE

Fiecare linie de control este asociată cu o componentă care este activă doar într-o Singură etapă pipeline.



Implementarea controlului



TEMĂ

Să se arate trecerea prin pipeline a următoarelor 5 instrucțiuni:

```
lw      $10, 20 ($1)
sub     $11, $2, $3
and     $12, $4, $5
or      $13, $6, $7
add     $14, $8, $9
```

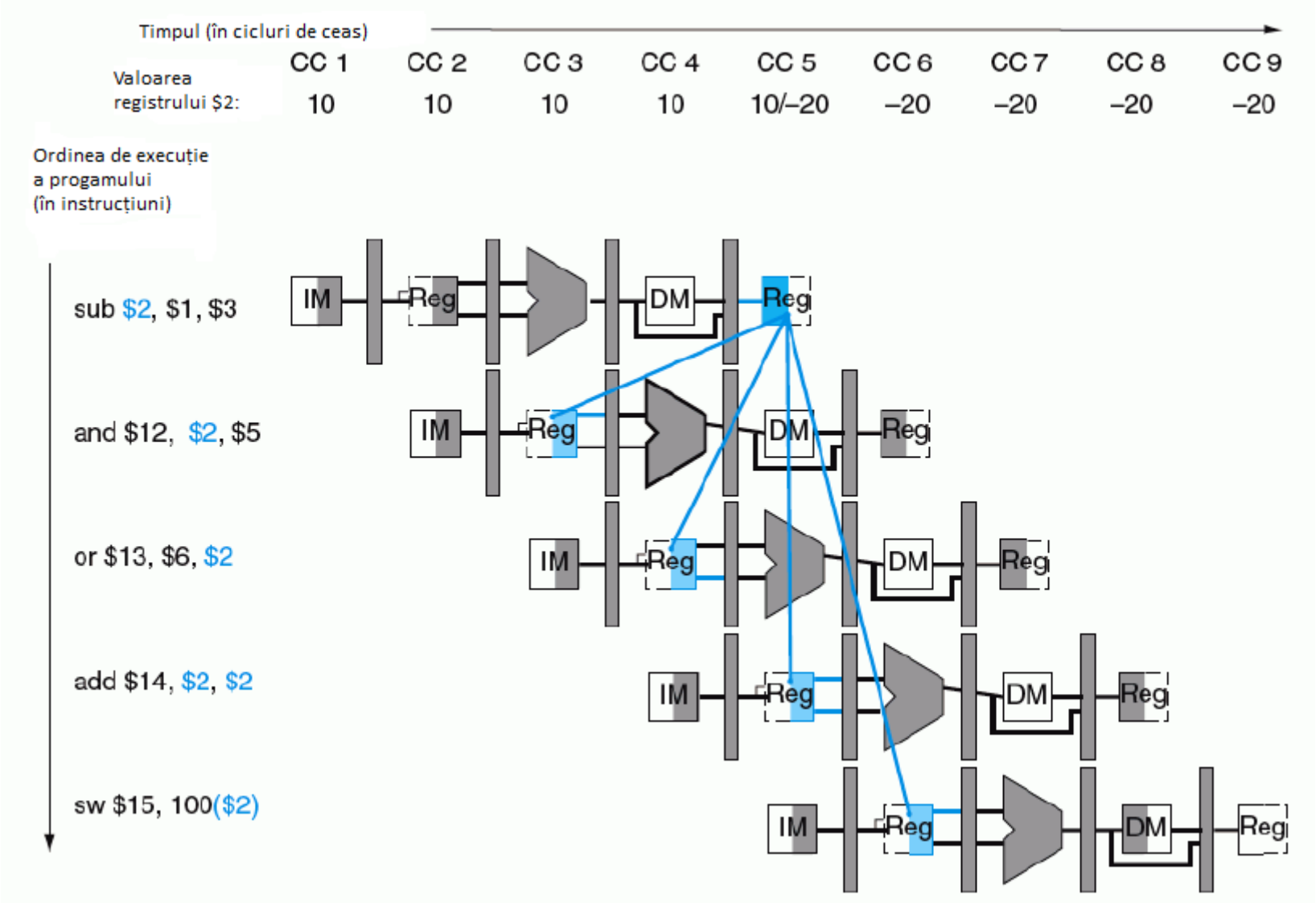
Să se eticheteze instrucțiunile din pipeline care precedă instrucțiunea lw sub forma

inainte <1>, inainte <2>, …..

și instrucțiunile care urmează instrucțiunii add sub forma

dupa <1>, dupa <2>

HAZARDURILE DE DATE ȘI AVANSAREA



Soluția software – introducere nop-uri => cicluri de ceas în care nu se face nimic

Detecția hazardului

1a. EX/MEM.RegistruRd = DI/EX.RegistruRs

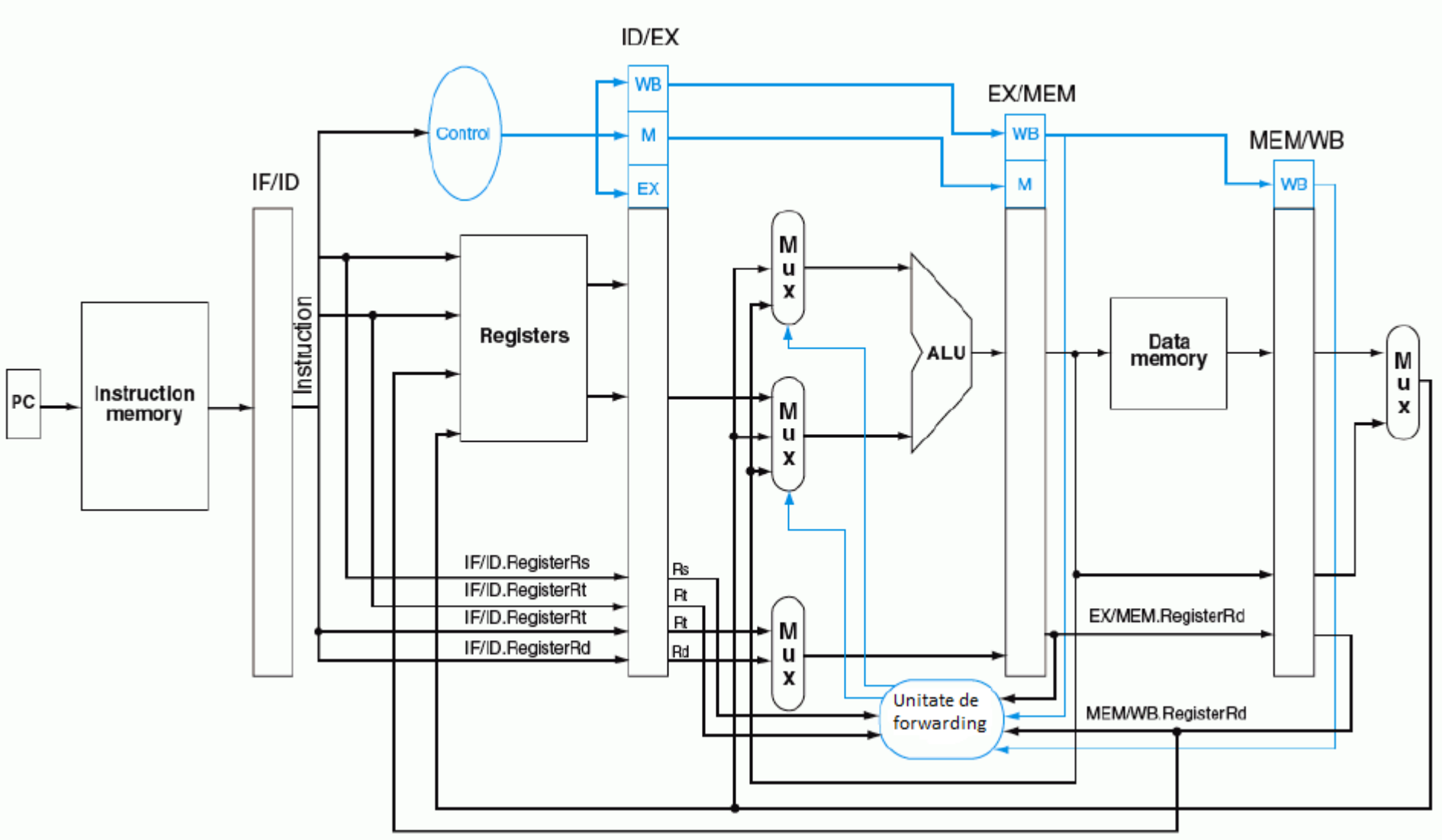
1b. EX/MEM.RegistruRd = DI/EX.RegistruRt

2a. MEM/RS.RegistruRd = DI/EX.RegistruRs

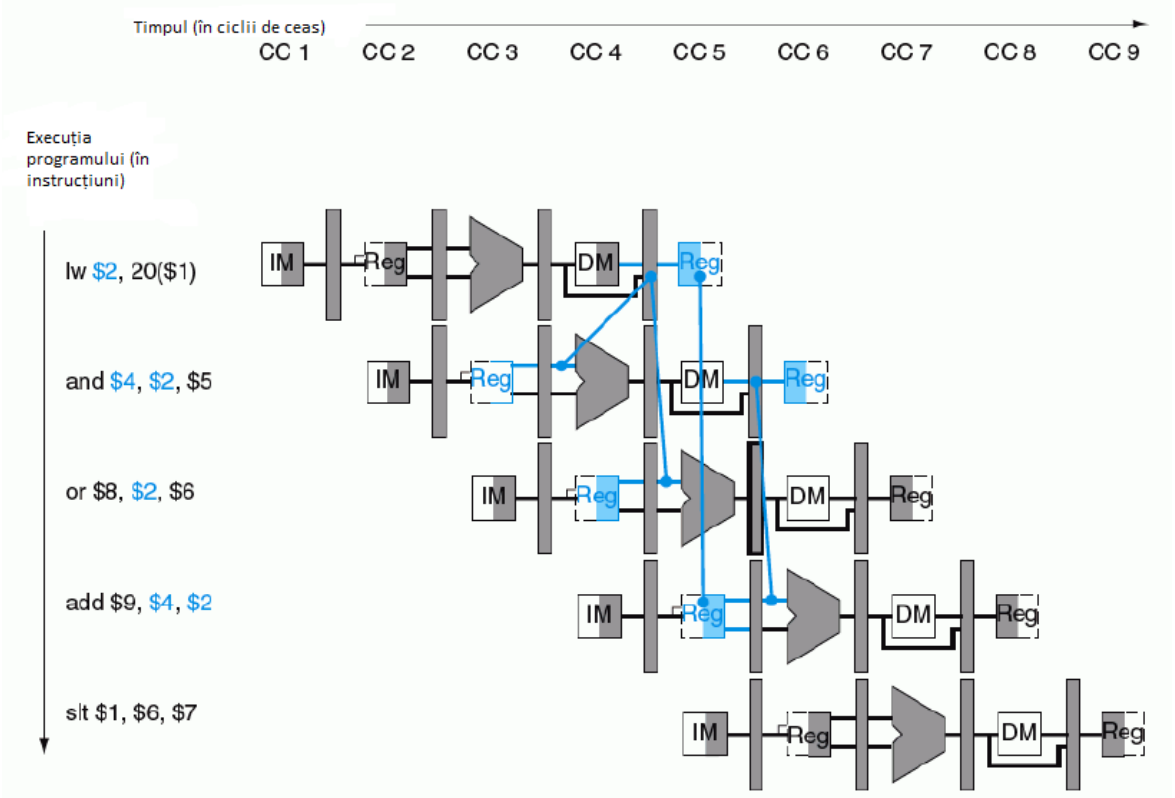
2b. MEM/RS.RegistruRd = DI/EX.RegistruRt

Dacă intrările UAL pot fi luate de la orice registru pipeline, nu numai de la EI/DI, atunci avansarea ar fi corectă.

Soluția hardware



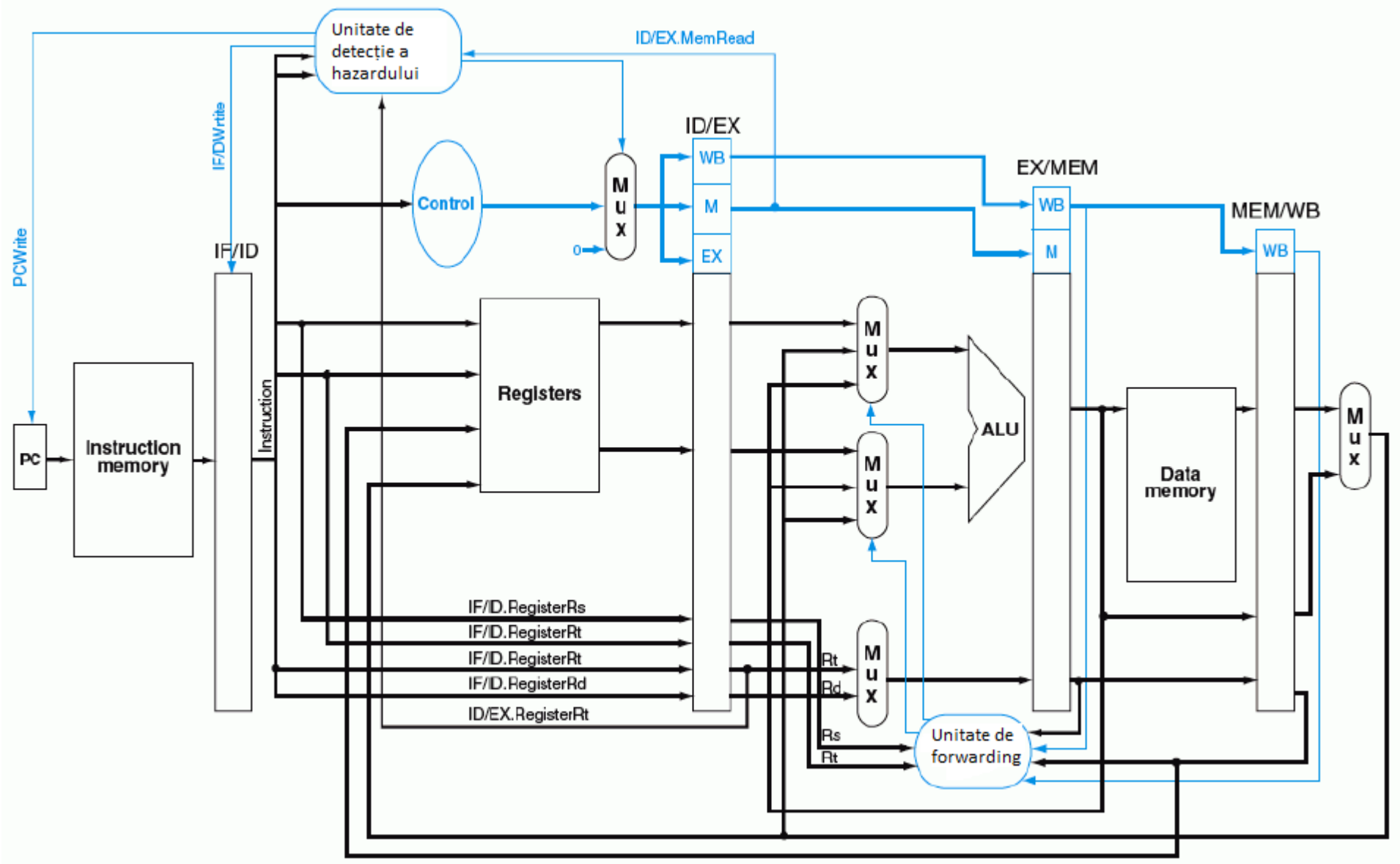
HAZARDURILE DE DATE ȘI STAȚIONĂRILE



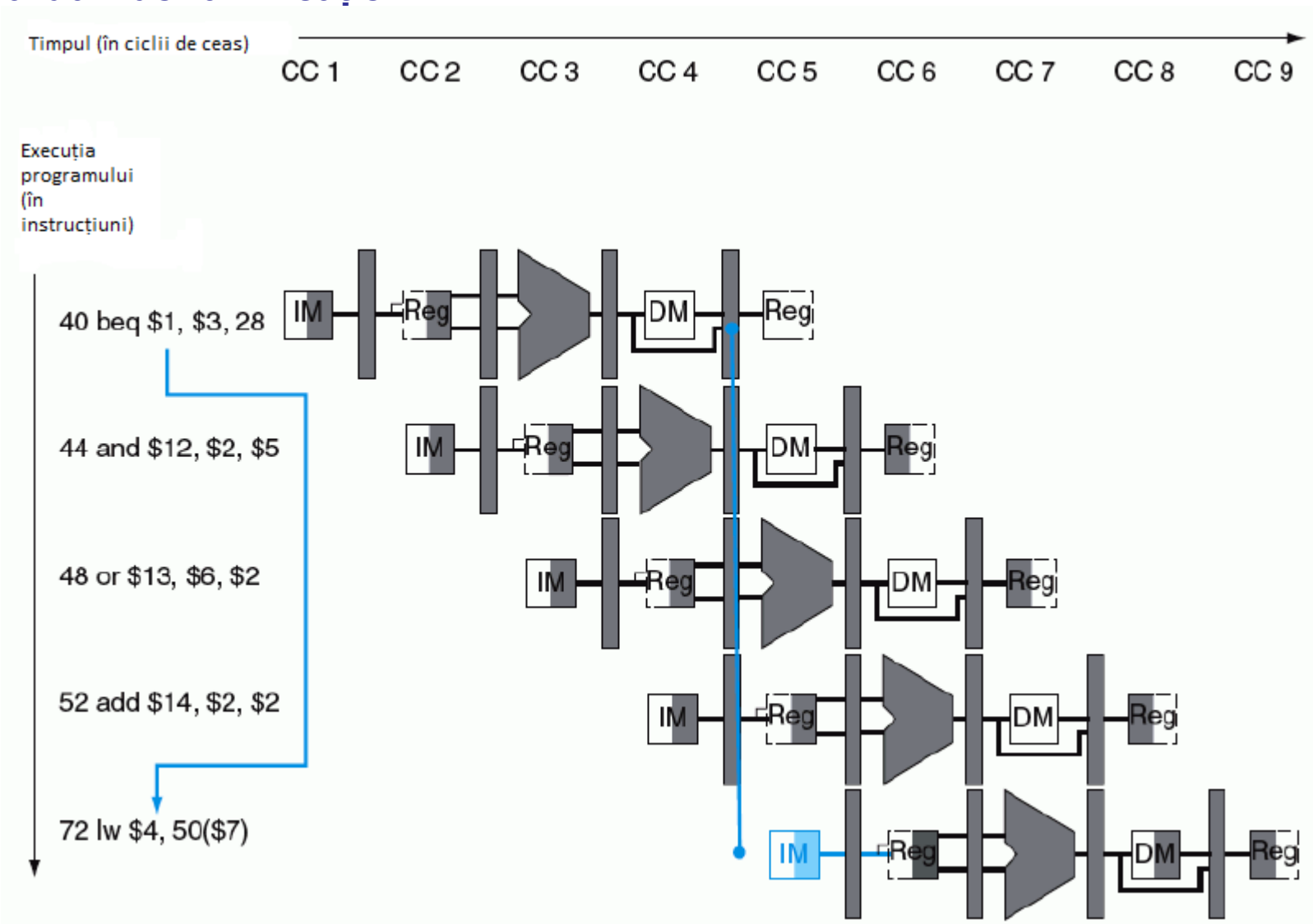
Pe lângă unitatea de avansare trebuie să existe o unitate de detectare a hazardului.

If (ID/EX.CitesteMem and ((ID/EX.RegistruRt = IE/ID.RegistruRs) or
(ID/EX.RegistruRt = IE/ID.RegistruR)))
staționare pipeline

Rezultatul final



Hazarduri de ramificație



Predicţia dinamică a ramificaţiei

Presupunem că ramificaţia nu va fi acceptată

O altă schemă de predicţie se bazează pe observarea adresei instrucţiunii

Deficienţă

Chiar dacă ramificaţia va fi acceptată aproape întotdeauna, predicţia va fi incorectă de două ori atunci când ramificaţia nu este acceptată.

Exemplu:
Se consideră o ramificaţie de buclă care ramifică de 9 ori consecutiv şi apoi nu este acceptată o dată.

Soluţie: introducerea schemelor de predicţie cu 2 biţi – o predicţie trebuie să greşească de 2 ori înainte de a fi schimbată.

Pipeline superscalar şi pipeline dinamic

Superpipeline – presupune pipeline-uri mai lungi

O altă îmbunătăţire prevede repetarea componentelor hardware astfel încât
Să crească numărul de instrucţiuni prelucrate în fiecare etapă de pipeline.

Pipeline dinamic – realizat de hardware pentru evitarea hazardurilor pipeline. De regulă se adaugă resurse hardware suplimentare pentru executarea instrucţiunilor următoare în paralel.

Deficienţă
Creşterea complexităţii controlului.

```
Lw      $t0, 20($s2)
Addu    $t1, $t0, $t2
Sub     $s4, $s4, $t3
Slti    $t5, $s4, 20
```

Exploatarea ierarhiei de memorie

Un exemplu intuitiv

Un student aflat în biblioteca facultăţii are pe masă o serie de cărţi

Subiectul căutat nu se regăseşte în cărţile aflate pe masă

Studentul se reîntoarce la rafturi şi extrage o nouă carte

Existența unui număr mare de cărţi pe masă reduce timpul de căutare

Probabilitatea de căutare nu este aceeaşi pentru toate cărţile din bibliotecă

Un program nu accesează toată secțiunea sa de cod sau de date cu aceeaşi probabilitate.

Principiul localizării - stă la baza modului de operare a programelor

stabilește faptul că programele accesează o porțiune relativ redusă a spațiului lor de adrese la orice moment de timp

Localizarea temporală - localizare în timp - dacă se face referire la un anumit obiect, este posibil ca acesta să fie referit din nou cât de curând

Localizarea spațială - localizare în spațiu - dacă se face referire la un anumit obiect din memorie, obiectele ale căror adrese sunt învecinate cu acesta, tind să fie adresate cât de curând.

Principiul localizării temporare este folosit la implementarea memoriei unui calculator sub forma unei ierarhii de memorie.

O ierarhie de memorii poate fi alcătuită din mai multe niveluri, însă datele la un moment dat sunt copiate doar între două niveluri adiacente.

Unitatea minimă de informație care poate fi prezentă sau absentă într-o ierarhie de memorie se numește bloc.

Unitatea minimă de informație care poate fi prezentă sau absentă într-o ierarhie de memorie se numește bloc.

Regăsirea informației necesare procesorului într-un bloc de memorie superior se numește **HIT**

Neregăsirea datelor pe nivelul superior se numește **MISS**

Rata de succes - fracțiunea din accesele la memorie ce au găsit datele în nivelul superior de memorie.

Rata de eşec = 1 - rata de succes fracţiunea din accesele la memorie care nu au găsit datele în nivelul superior de memorie.

Timpul de succes - reprezintă timpul necesar accesului la un nivel superior al ierarhiei de memorie, ce include și timpul necesar determinării tipului de acces.

Penalizarea de eșec - reprezintă timpul necesar înlocuirii blocului din nivelul superior cu blocul corespunzător din nivelul inferior, incluzând și timpul trimiterii acestui bloc către procesor.

Construirea sistemelor de memorie afectează:

1. modul în care SO-ul administrează memoria și perifericele
2. modul în care compilatoarele generează codul
3. modul în care aplicațiile folosesc mașina de calcul

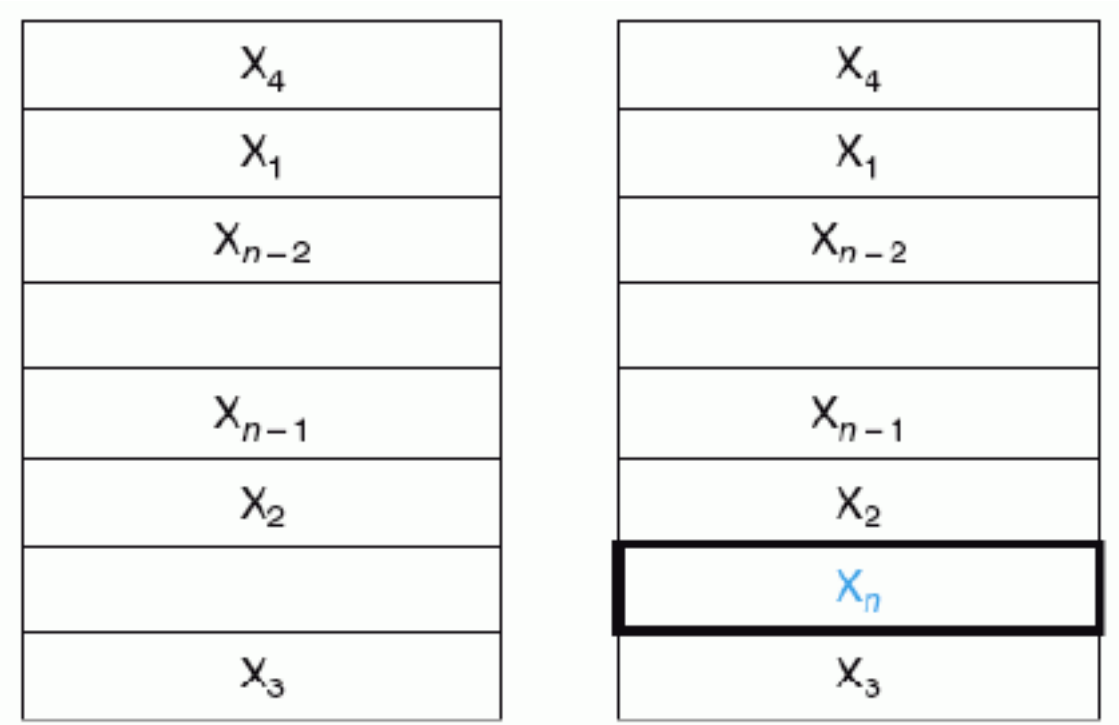
Principiu de bază

Programele prezintă localizare temporală cât și localizare spaţială.

Ierarhiile de memorie folosesc avantajul localizării temporale păstrând datele accesate recent cât mai aproape de procesor.
Ierarhiile de memorie folosesc avantajul localizării spațiale prin mutarea blocurilor conținând cuvinte învecinate din memorie în nivelurile superioare ale ierarhiei.

În anii ’60 s-a folosit cuvântul ***cache*** pentru a desemna nivelul ierarhiei de memorie aflat între UCP și memoria principală.

PRINCIPIILE DE BAZĂ ALE MEMORIILOR CACHE



Iniţial cuvântul de date X_n nu se găsește în cache

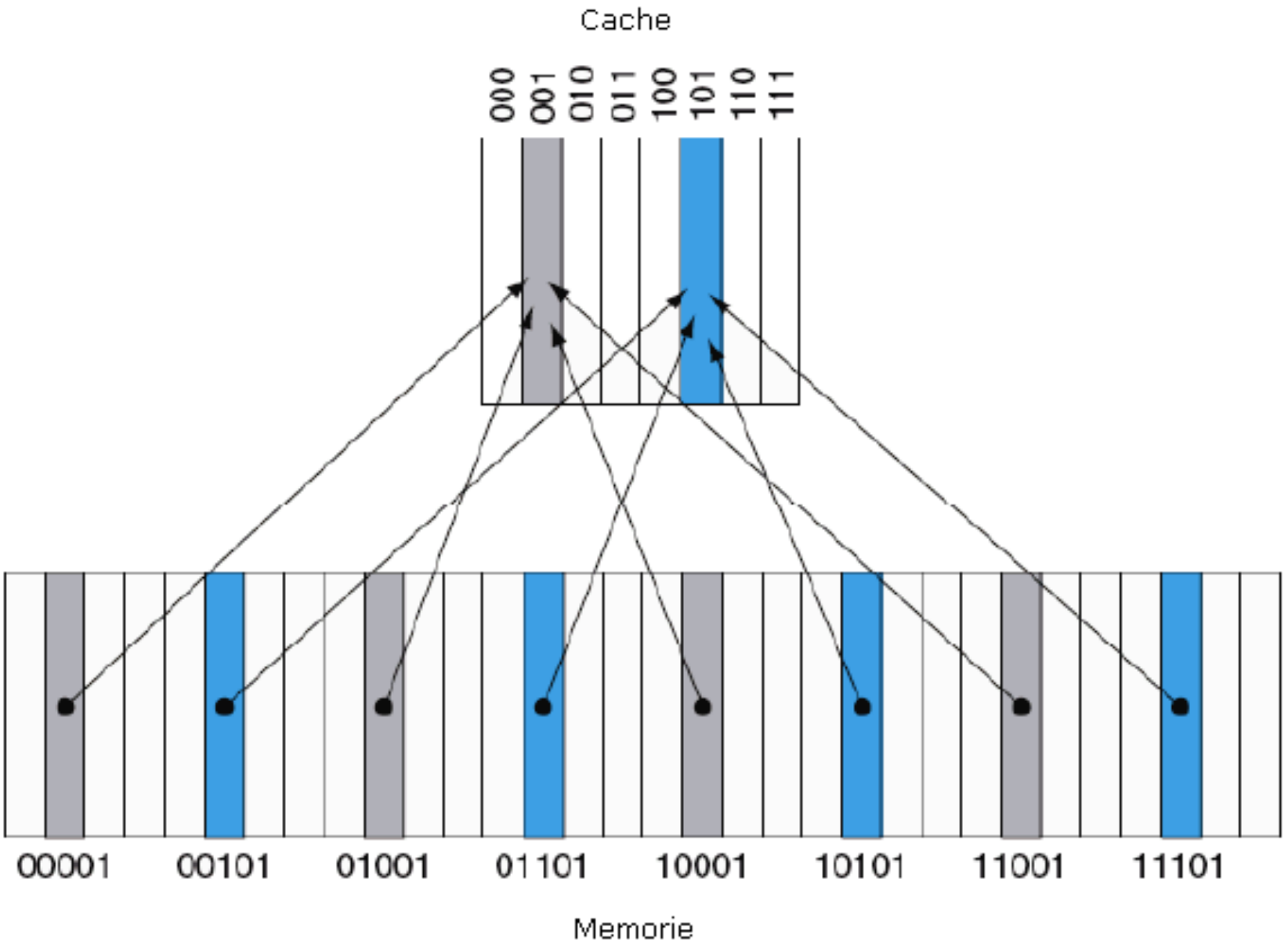
Cum se poate determina dacă o dată este în memoria cache ?

Dacă o dată există în memoria cache cum poate fi ea gasită ?

!!!!! Fiecare cuvânt poate fi memorat doar într-o anumită locație a memoriei cache => corespondenţă directă.

Corespondenţa dintre adrese și locaţiile memoriei cache se determină astfel:

(Adresa blocului) modulo (numărul de blocuri din memoria cache)



Cum determinăm dacă o dată este în memoria cache ?

Cum determinăm dacă o dată este validă sau nu ?

Metoda cea mai des folosită este cea de a aduna un ***bit de validare*** pentru a indica dacă o locație conţine o adresă validă.

Accesarea unei memorii cache cu corespondeță directă

Cerere	Adresa	HIT / MISS	Blocul din cache
22	10110		
26	11010		
22	10110		
26	11010		
16	10000		
3	00011		
16	10000		
18	10010		

Index	V	Marcaj	Date
0	N		
1	N		
10	N		
11	N		
100	N		
101	N		
110	N		
111	N		

Index	V	Marcaj	Date
0	N		
1	N		
10	N		
11	N		
100	N		
101	N		
110	Y	10	mem (10110)
111	N		

Index	V	Marcaj	Date
0	N		
1	N		
10	Y	11	mem (11010)
11	N		
100	N		
101	N		
110	Y	10	mem (10110)
111	N		

Index	V	Marcaj	Date
0	Y	10	mem (10000)
1	N		
10	Y	11	mem (11010)
11	N		
100	N		
101	N		
110	Y	10	mem (10110)
111	N		

Acest tip de accesare permite folosirea principiului localizării temporale - cuvintele accesate recent le înlocuiesc pe cele accesate mai puţin recent.