

Proiectarea procesorului MIPS care operează într-un singur ciclu de ceas.

– Curs 8_1 –

Subiecte abordate:

- Introducere
- Realizarea unei căi de date

MIPS = Microprocessor without Interlocked Pipeline Stages
= Procesor de tipul RISC

Performanțele unui procesor sunt impuse de către următorii factori:

- numărul de instrucțiuni din programul care se execută (n);
- durata ciclului de ceas (T);
- numărul mediu de cicluri de ceas pe instrucțiune (CPI).



Proiectarea procesorului (unitatea de execuție și unitatea de comandă) va determina:

- durata perioadei ceasului;
- numărul de cicluri de ceas pe instrucțiune.

Etapele proiectării:

1. Se examinează setul de instrucțiuni din care rezultă cerințele pentru unitatea de execuție:
 - semnificația fiecărei instrucțiuni este dată de transferurile între registre;
 - unitatea de execuție trebuie să includă elementele de memorare corespunzătoare: registrele necesare setului de instrucțiuni (eventual mai multe);
 - unitatea de execuție trebuie să asigure fiecare transfer între registre;
2. Se selectează componentele unității de execuție și se stabilește metodologia de sincronizare (aplicare a ceasului).
3. Se assemblează unitatea de execuție corespunzător specificațiilor.
4. Se analizează implementarea fiecărei instrucțiuni pentru determinarea semnalelor și a punctelor de comandă, care afectează implementarea fiecărui transfer între registre.
5. Se assemblează logica de comandă.

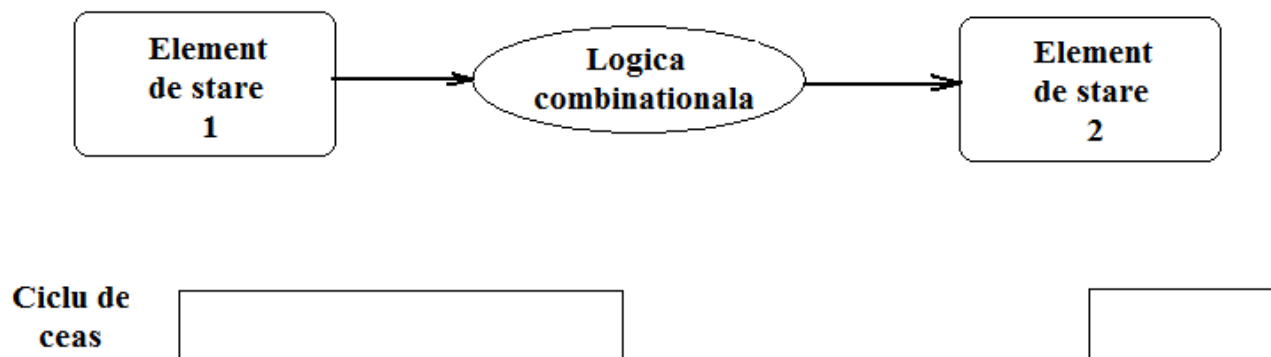
Unitățile funcționale în implementarea MIPS constau în două tipuri diferite de elemente logice:

 *elemente care operează asupra valorilor de date*  toate elementele combinaționale (ieșirile lor depind doar de intrările prezente la acel moment de timp)

 *elemente care conțin stare*  au o anumită memorie internă (aceste elemente de stare caracterizează complet mașina).

Metodologia folosirii ceasului

- ✓ Definește momentul în care semnalele pot fi citite și când pot fi scrise.
- ✓ Metodologia de acționare pe frontul ceasului înseamnă că orice valoare păstrată în mașină este actualizată numai pe un front al impulsului de ceas.
- ✓ Astfel, dacă elementul de stare este scris la fiecare front activ al ceasului, semnalul de control pentru scriere nu va mai fi indicat.



Toate semnalele trebuie să se propage de la elementul de stare 1, prin logica combinațională și să stabilească elementul de stare 2 pe perioada unui ciclu de ceas.

Formatele instrucțiunilor procesorului MIPS

Toate instrucțiunile MIPS au 32 de biți. Sunt prezente trei formate diferite:

Instrucțiunea de tip R	op	rs	rt	rd	shamt	funct
	31-26	25-21	20-16	15-11	10-6	5-0
Instrucțiunea de tip I	op	rs	rt	adresa		
	31-26	25-21	20-16	15-0		
Instrucțiunea de tip J	op	adresa țintă				
	31-26	25-0				

Tipul R = formatul instrucțiunii aritmetice

Tipul I = formatul instrucțiunii de transfer, ramificație, imediat.

Tipul J = formatul instrucțiunii de salt

Câmpurile de operație sunt următoarele:

- **op**: codul de operație al instrucțiunii;
- **rs, rt, rd**: adresele registrelor sursă și destinație
- **shamt**: cantitatea/numărul de biți cu care se efectuează deplasarea;
- **funct**: selectează varianta de operație specificată de către op;
- **adresa/imediat**: deplasarea adresei(offset)/valoarea imediata;
- **adresa țintă/target address**: deplasarea pentru adresa țintă de salt.

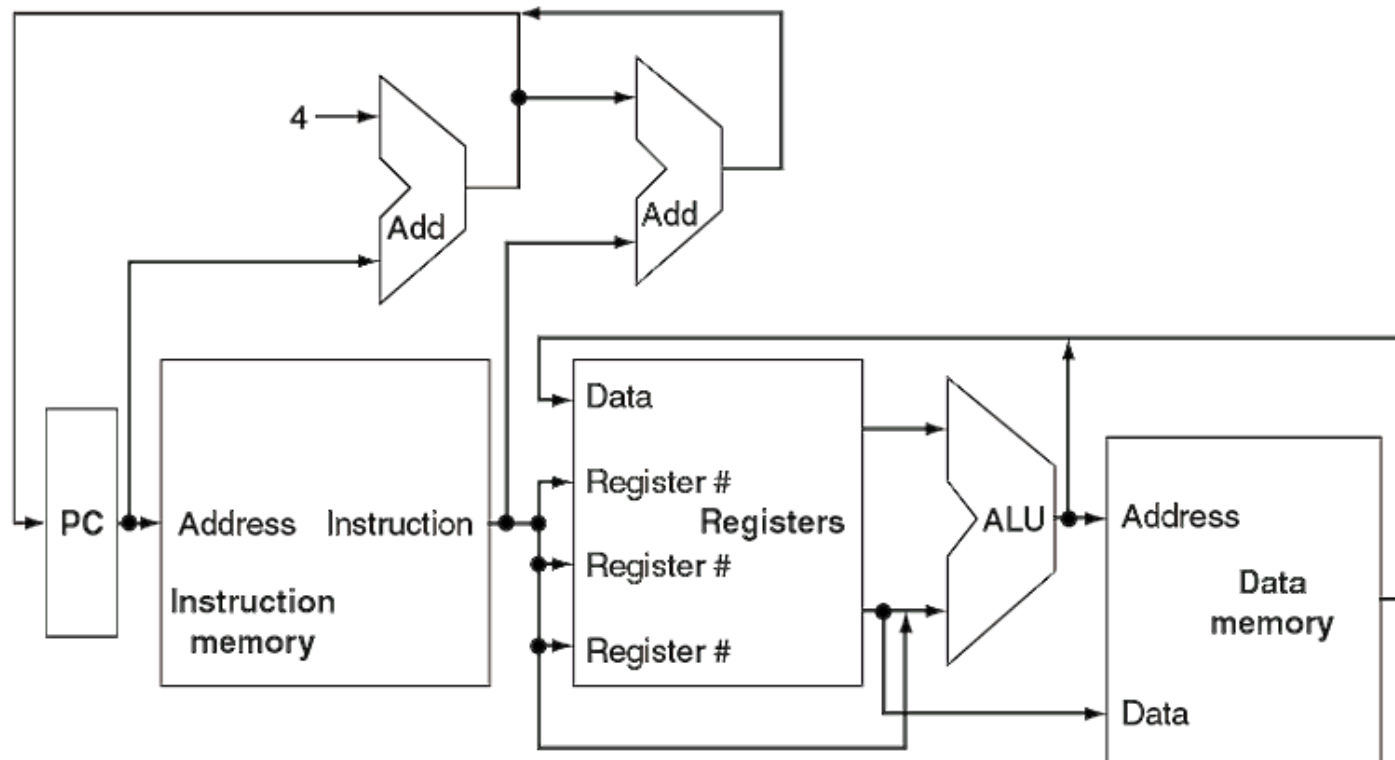
Ne propunem implementarea următoarelor instrucțiuni:

- ✓ Instrucțiuni de referire a memoriei - încărcare/memorare
- ✓ Instrucțiuni aritmetice și logice - add, sub, and, or și slt (set-on-less-than)
- ✓ Instrucțiunile de ramificație la egal și de salt

Indiferent de clasa instrucțiunii, primii pași de implementare a unei instrucțiuni sunt la fel:

- ✓ se va trimite memoriei PC-ul și se va extrage instrucțiunea din memorie
- ✓ se citesc unul sau două registre ➡ vom folosi câmpurile instrucțiunii pentru selectarea registrelor. Instrucțiunea de încărcare a unui cuvânt din memorie presupune citirea doar a unui registru.

Unitățile funcționale și legăturile dintre ele

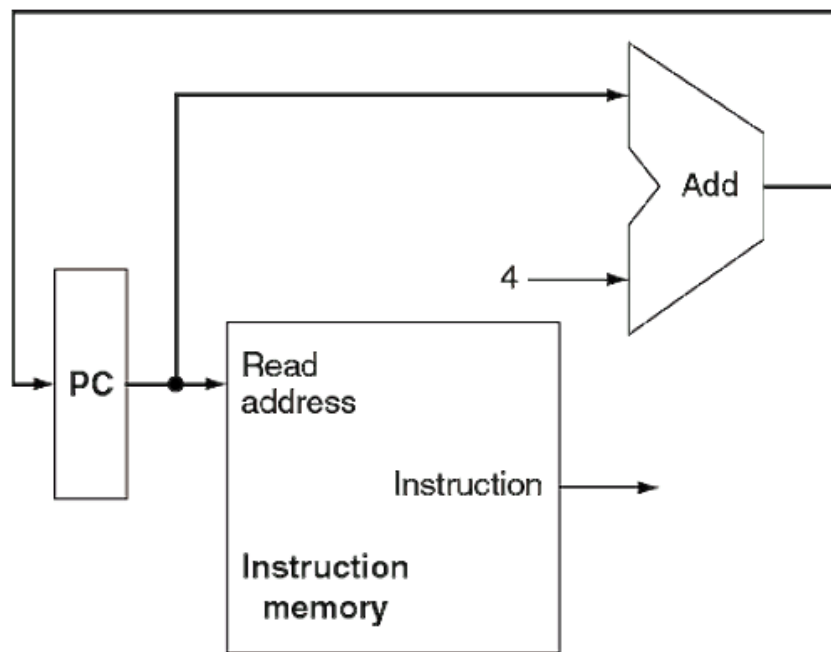


- ✓ Dacă instrucțiunea este de tip aritmetic sau logic ➡ rezultatul din UAL trebuie scris într-un registru
- ✓ Dacă operația este de încărcare/memorare ➡ rezultatul UAL va fi o adresă
- ✓ Pentru ramificații ➡ se folosește ieșirea UAL în determinarea adresei următoarei instrucțiuni de executat ➡ se introduce o logică de control

OBS. !!!

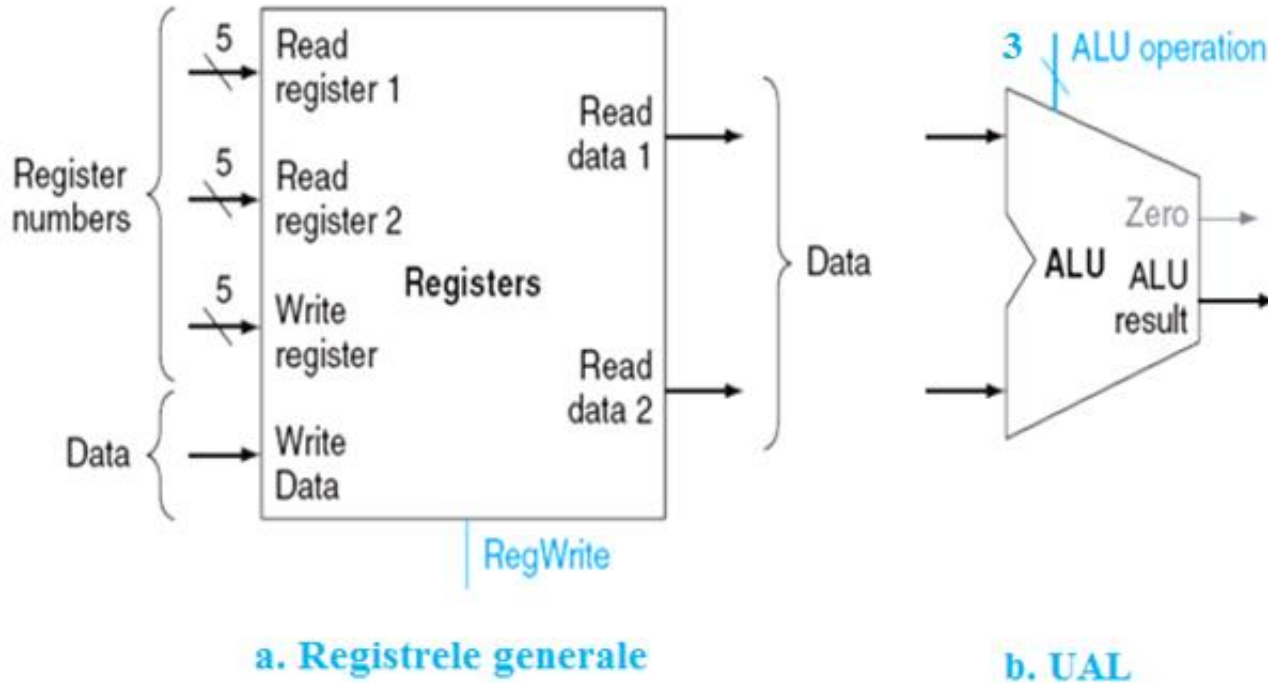
1. Utilizăm **PC** (contorul instrucțiunilor de program) pentru a oferi memoriei de instrucțiuni adresa instrucțiunii
2. Registrele operand utilizate de instrucțiune sunt specificate în câmpurile instrucțiunilor
3. După extragerea registrelor operand ele pot fi prelucrate pentru
 - a. Calcularea unei adrese de memorie - instr. de încărcare/memorare
 - b. Calcularea unui rezultat aritmetic
 - c. Pentru o comparație

Realizarea căii de date - determinarea instrucțiunii curente și trecerea la următoarea instrucțiune



- ✓ Extragem instrucțiunea din memorie
- ✓ Incrementăm PC-ul cu 4 pentru trecerea la instrucțiunea următoare.
- ✓ Instrucțiunea următoare este situată la o distanță de 4 octeți.
- ✓ PC este un element de stare

Realizarea căii de date - instrucțiunile aritmetice și logice



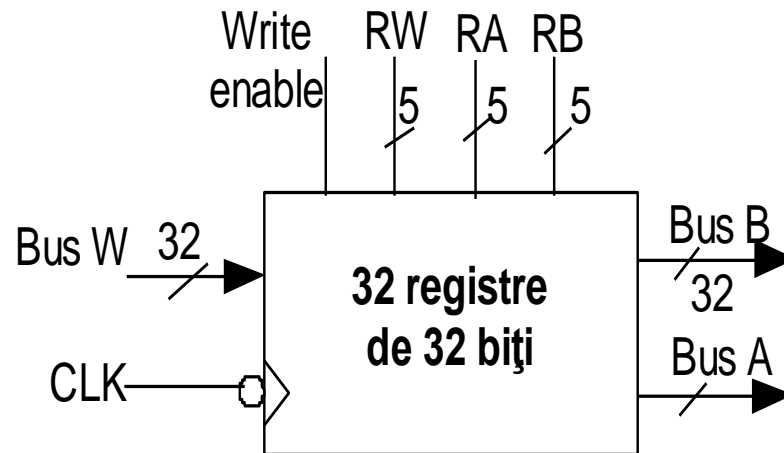
✓ Instrucțiunile de tipul R: add, sub, and, or, slt.

✓ Instrucțiunile de format R au ca operanzi 3 registre - 2 sunt citite și unul este scris

✓ UAL operează cu valorile citite din registre

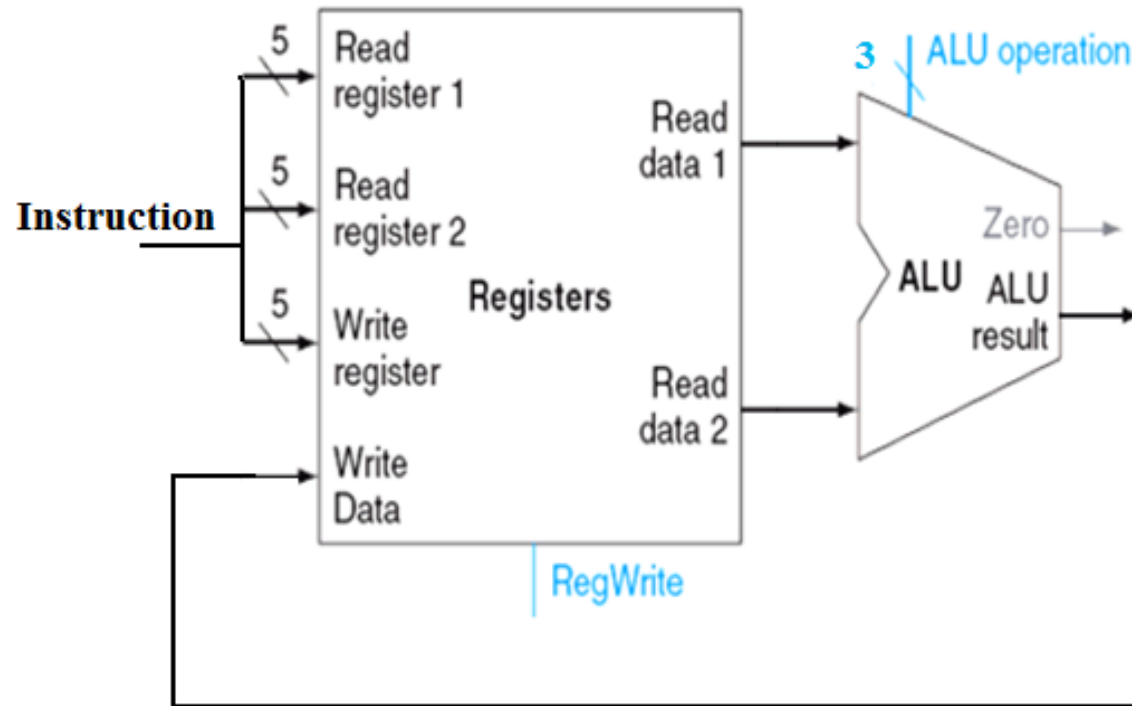
Remember !!!!

***Registrele generale
din cursul 5***



- RA (număr), care specifică registrul general al cărui conținut se plasează pe busA;
- RB (număr), care specifică registrul general al cărui conținut se plasează pe busB;
- RW (număr), care specifică registrul general al cărui conținut va fi modificat prin forțarea conținutului magistralei busW, când Write Enable este pe nivel ridicat.

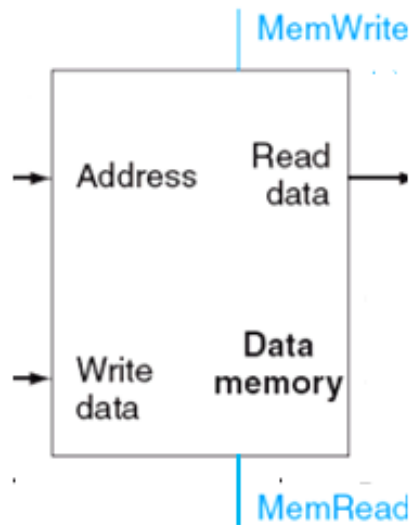
Realizarea căii de date - instrucțiunile aritmetice și logice (de tip R)



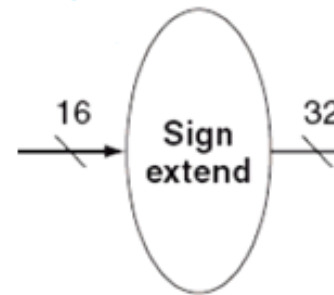
Instrucțiunile de încărcare și memorare

Dacă operația este de încărcare/memorare → rezultatul UAL va fi o adresă.

Pe lângă registrele generale și UAL, unitățile necesare implementării acestor instrucțiuni sunt:

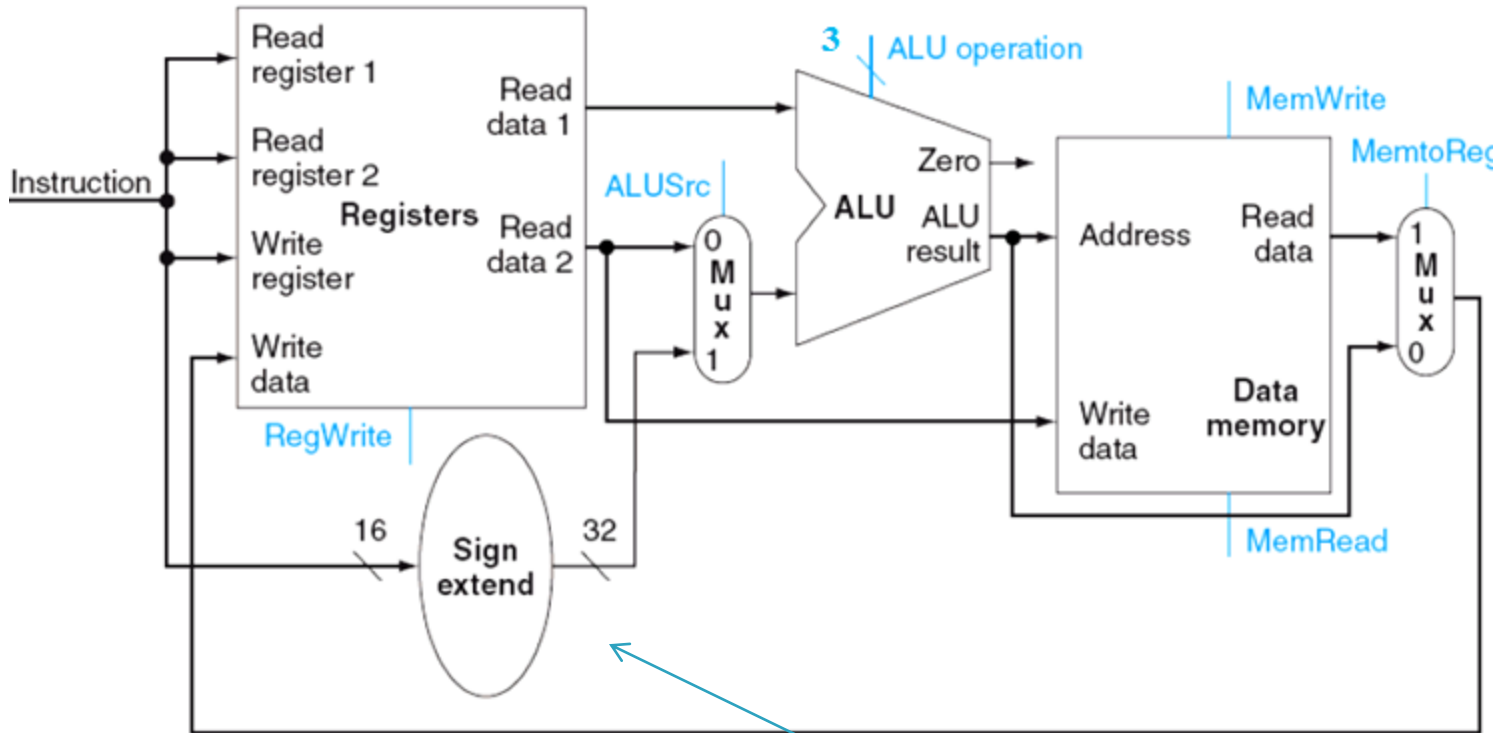


a. unitatea memoriei de date



b. unitatea de extindere a semnului

Instrucțiunile de încărcare și memorare



Instrucțiunile sunt:

lw \$t1, valoare_deplasare(\$t2)

sw \$t1, valoare_deplasare(\$t2)

unitate pentru a extinde
semnul câmpului de
deplasare din instrucțiune

Aceste instrucțiuni calculează o adresă de memorie prin adunarea registrului de bază \$t2 cu câmpul pentru deplasarea cu semn de 16 biți din instrucțiune. Extinderea se face până la o valoare de 32 de biți cu semn.

Dacă instrucțiunea este de memorare, valoarea de memorat trebuie citită din fișierul de registre, unde ea se găsește în \$t1.

Dacă instrucțiunea este de încărcare, valoarea citită din memorie trebuie scrisă în \$t1 care este registrul specificat din fișierul de registre.



Calea de date pentru instrucțiunile de încărcare sau memorare realizează un acces la un registru, urmat de calcularea unei adrese de memorie, iar în continuare o citire sau o scriere în memorie și o scriere în fișierul de registre, dacă instrucțiunea este de încărcare.

Instrucțiunea beq (de ramificație neîntârziată)

Instrucțiunea este formată din 2 registre care sunt comparate pentru egalitate și o deplasare de 16 biți, folosită în calculul adresei obiectiv pentru ramificație:

beq \$t1, \$t2, offset

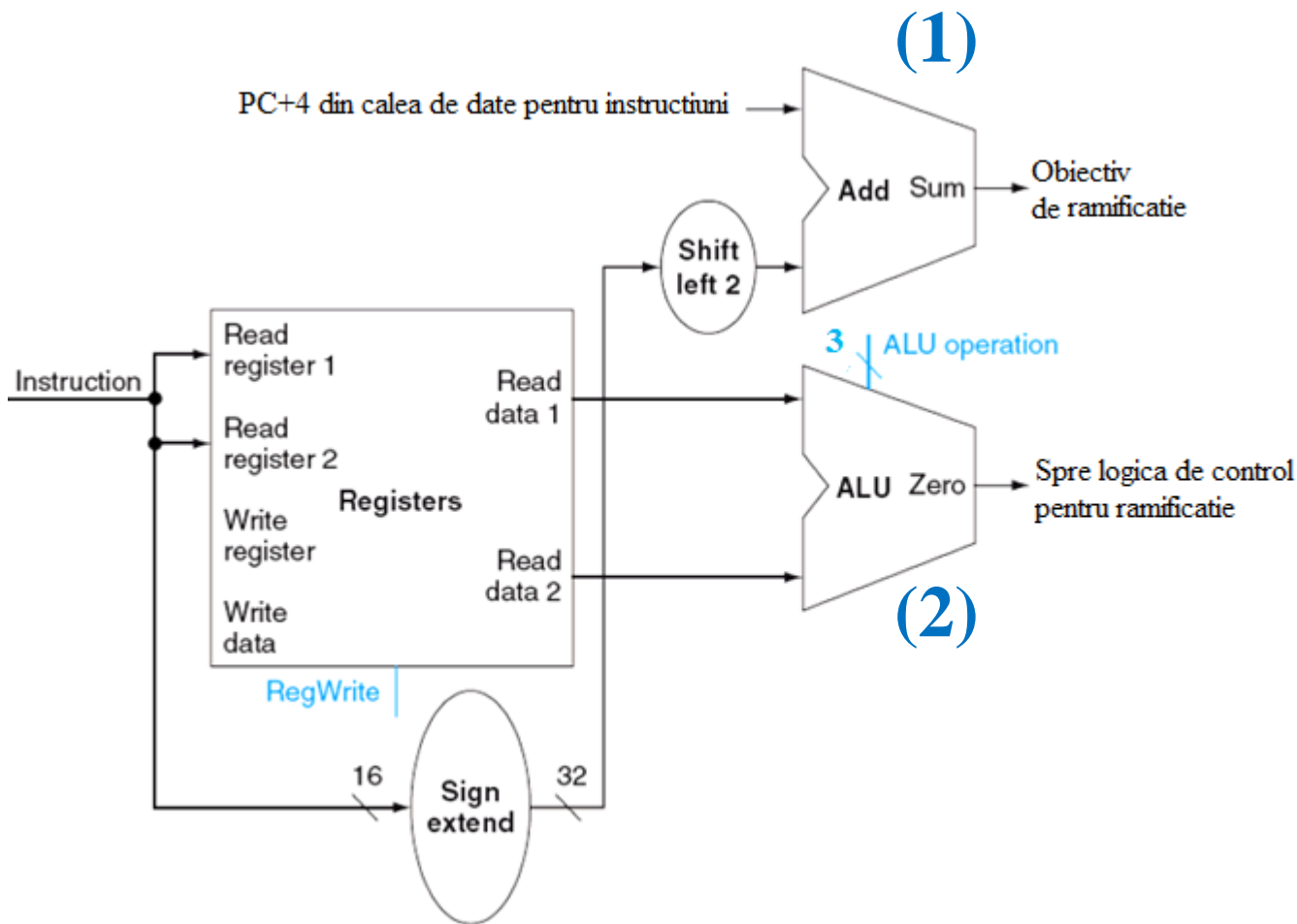
În vederea implementării acestei instrucțiuni trebuie determinată adresa obiectiv pentru ramificație ➡ PC + câmpul de deplasare al instrucțiunii cu semnul extins

Offset este folosit în calculul adresei obiectiv pentru ramificație, care este relativă la adresa instrucțiunii de ramificație.

Câmpul deplasării trebuie mutat cu 2 poziții la stânga.

Detalii legate de definiția instrucțiunilor de ramificație!!!

- ✓ Arhitectura setului de instrucțiuni specifică faptul că baza pentru calculul adresei de ramificație este adresa instrucțiunii care urmează ramificației. Deoarece automat se calculează $PC+4$ (adresa următoarei instrucțiuni), această valoare se utilizează ca bază pentru calculul adresei obiectiv pentru ramificație.
- ✓ Arhitectura mai stabilește că trebuie deplasat spre stânga cu 2 biți câmpul *deplasării* din instrucțiune, deoarece deplasarea este la nivel de cuvânt. Această deplasare spre stânga crește domeniul efectiv al câmpului deplasării cu un factor de 4.



Operanzii sunt egali



adresa obiectiv pentru
ramificatie devine noul
PC

Operanzii sunt diferiti



PC-ul incrementat este
noul PC

Calea de date pentru ramificatie implică realizarea a 2 operații:

1. Calcularea adresei obiectiv pentru ramificatie – se realizează prin unitatea pentru extinderea semnului și un sumator.
2. Compararea conținutului registrelor – cu ajutorul reg. generale, prin scădere.

- Sumatorul (2) evaluează condiția de ramificație.
- Sumatorul (1) calculează adresa obiectiv pentru ramificație, ca sumă între PC-ul implementat și ultimii 16 biți ai instrucțiunii (deplasarea ramificației) cu semnul extins, deplasați spre stânga cu 2 biți.

Trebuie determinat dacă următoarea instrucțiune de executat este instrucțiunea care urmează în secvență sau instrucțiunea care se găsește la adresa obiectiv pentru ramificație.

Instrucțiunea de salt

- operează prin înlocuirea celor mai puțin semnificativi 28 de biți ai PC-ului cu cei mai puțin semnificativi 26 de biți ai instrucțiunii deplasati spre stânga cu 2 biți ➡ concatenăm 00 la deplasarea saltului.

Implementarea unei singure căi de date

Instucțiunile:

- ☐ Încărcarea unui cuvânt (lw)
- ☐ Memorarea unui cuvânt (sw)
- ☐ Ramificație la egal (beq)
- ☐ Adunare (add)
- ☐ Scădere (sub)
- ☐ ȘI (and)
- ☐ SAU (or)
- ☐ Setare la mai mic decât (slt)

Toate instrucțiunile se execută într-un singur ciclu de ceas!!

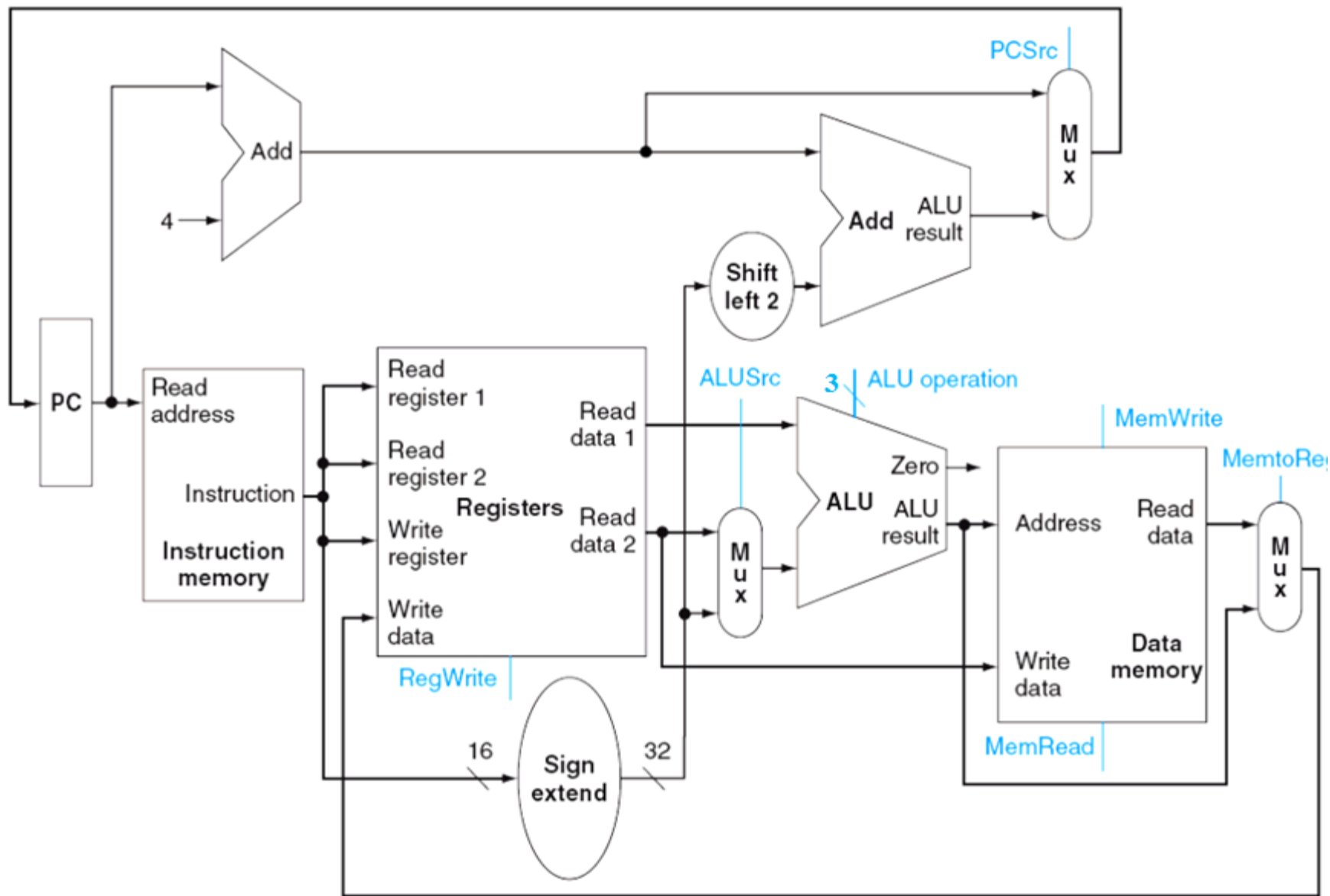


Nici o resursă a căii de date nu poate fi utilizată mai mult de o singură dată la fiecare instrucțiune.

✓ Pentru ca două clase diferite de instrucțiuni să folosească în comun un element al căii de date, trebuie să se permită conexiuni multiple la intrarea unui element și să existe un semnal de control care să facă selecția între aceste intrări.



Folosirea unui **multiplexor (selector de date)**





Care sunt diferențele dintre calea de date pentru instrucțiunile aritmetice și logice și calea de date pentru instrucțiunile de încărcare și memorare?

1. A doua intrare pentru UAL este un registru (instrucțiunile de tip R) sau este jumătatea inferioară, cu semnul extins, a instrucțiunii (dacă este instrucțiune de tip I).
2. Valoarea păstrată în registrul de destinație vine de la UAL (cazul tipului R) sau de la memorie (cazul tipului I).