Lezione 20

CPU a ciclo multiplo

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

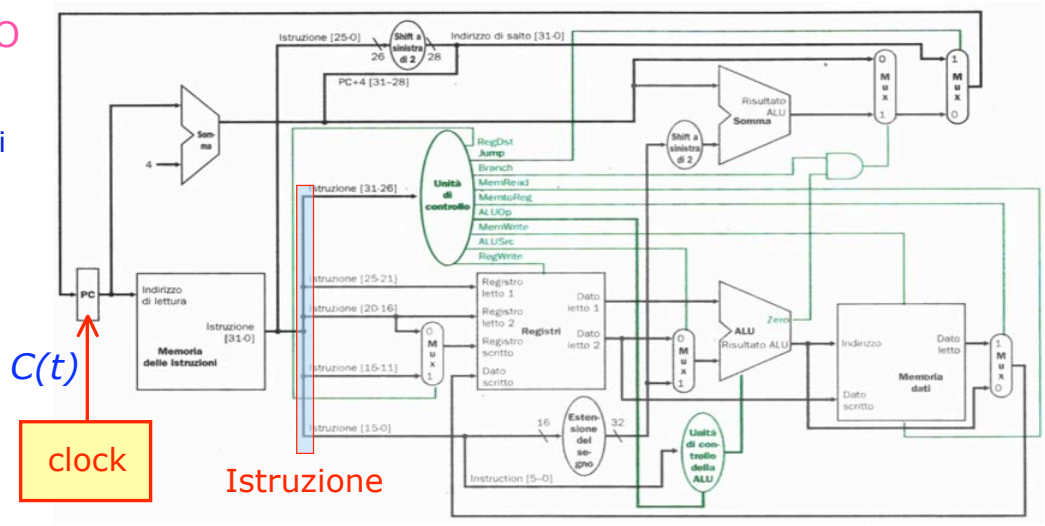
Sommario

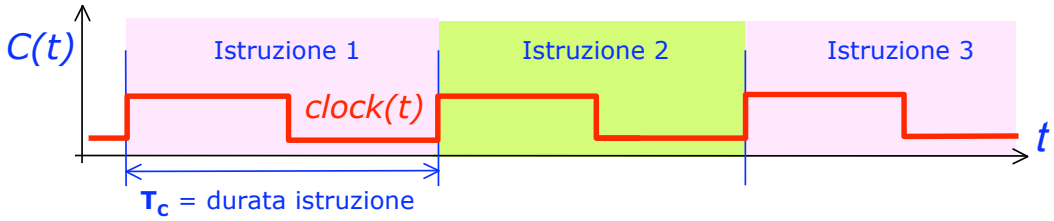
- I problemi della UC a singolo ciclo
- Principi ispiratori di una CPU multi-ciclo.
  - Le fasi di fetch e decodifica.
- Esecuzione multi-ciclo delle istruzioni R
- Esecuzione multi-ciclo delle istruzioni lw/sw.
- Esecuzione multi-ciclo dei salti ed analisi della CPU multi-ciclo.

CPU a Ciclo singolo

- CPU a CICLO SINGOLO:

  Ad ogni ciclo di clock viene eseguita un'istruzione completa.





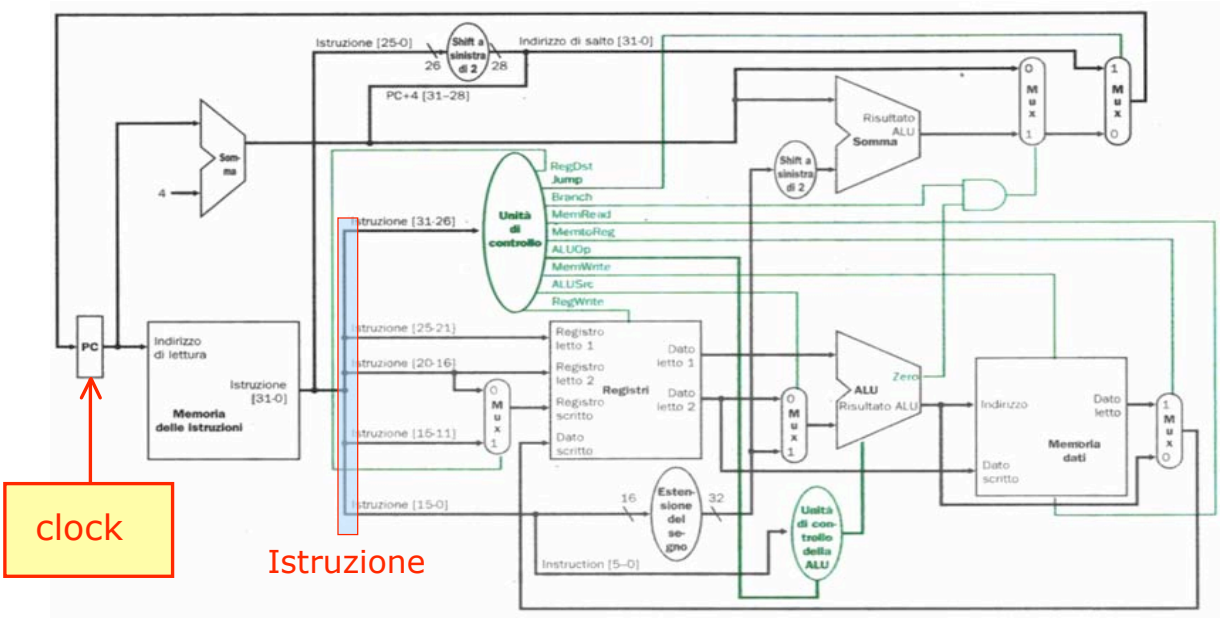
CPU singolo ciclo: caratteristiche

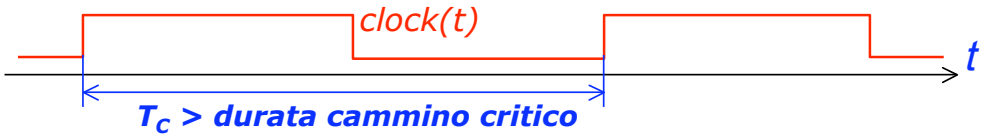
- Peculiarità:
  - Il segnale di CLOCK viene fornito solo al registro Program Counter
  - L' Unità di Controllo è una rete combinatoria

- Vantaggi
  - Semplicità circuitale
  - Velocità esecuzione, fissato T_C: 1 istruzione / ciclo di CLOCK
- Limiti
  - Utilizzo non efficiente delle risorse (3 ALU, 2 memorie) utilizzate in momenti diversi:
    - memoria istruzioni in fase di fetch, memoria dati in fase di accesso memoria
    - ALU PC (fetch), ALU branch (decodifica), ALU operazioni/test (esecuzione)
  - Tutte le istruzioni, semplici o complesse, durano sempre T_C

- ***T_C è dimensionato sul cammino critico dell'istruzione che dura di più***

CPU a Ciclo singolo





T_C > durata cammino critico

Esecuzione in un singolo ciclo di clock

- La durata del ciclo di clock T_C va dimensionata sul caso peggiore
  → istruzione di maggior durata
  → percorso logico più lungo (**cammino critico**)
  - Accesso memoria: 2 ns — ALU e sommatori: 2 ns
  - lettura/scrittura registri: 1 ns — decodifica: 2 ns
  - tempi trascurabili per gli altri elementi della CPU
  - componenti indipendenti possono lavorare in parallelo

- Percorso più lungo: istruzione di caricamento: **`lw`**

Istruzione	Memoria istruzioni (fetch)	Lettura registri (decodifica)	Operazioni ALU	Memoria dati	Write-back	Totale
`Tipo R`	2	2	2	0	1	**7 ns**
`lw`	2	2	2	2	1	**9 ns**
`sw`	2	2	2	2	0	**8 ns**
`beq`	2	2	2	0	0	**6 ns**
`j`	2	2	0	0	0	**4 ns**

Valutazione prestazioni: CPU a singolo ciclo

- Nella CPU a singolo ciclo la durata delle istruzioni è costante, e pari alla durata massima necessaria
- La durata delle istruzioni è maggiore della durata teorica media
  - In alcuni casi (caso II) decisamente maggiore della media pesata sulla durata delle istruzioni

La CPU a singolo ciclo è **inefficiente** (c'è uno spreco di tempo)

Istr.	lw	sw	beq	j	R	fp (add)	fp (mul)	Durata MAX	Durata media
Durata	9 ns	8 ns	6 ns	4 ns	7 ns	12 ns	20 ns		
Caso I	24%	12%	18%	2%	44%			**9 ns**	**7.36 ns**
Caso II	31%	21%	5%	2%	27%	7%	7%	**20 ns**	**8.98 ns**

CPU multi-ciclo

CPU a ciclo multiplo:

- Possibile soluzione: si spezza l'istruzione in **più passi parziali**
  - ciascun passo parziale impiega lo stesso tempo.
- In un ciclo di clock viene eseguito un singolo passo parziale di istruzione (non l'intera istruzione)
- Le istruzioni possono durare un numero diverso di cicli di clock

- Conseguenze:
  - Consentito riutilizzare le unità funzionali, in cicli di clock diversi
  - Necessità di registri di memoria temporanea: devono memorizzare lo stato delle unità funzionali, cioè l'informazione che può servire ai passi successivi.
  - L'unità di controllo diventa una FSM.
    - Nella CPU a singolo ciclo era una Rete Combinatoria

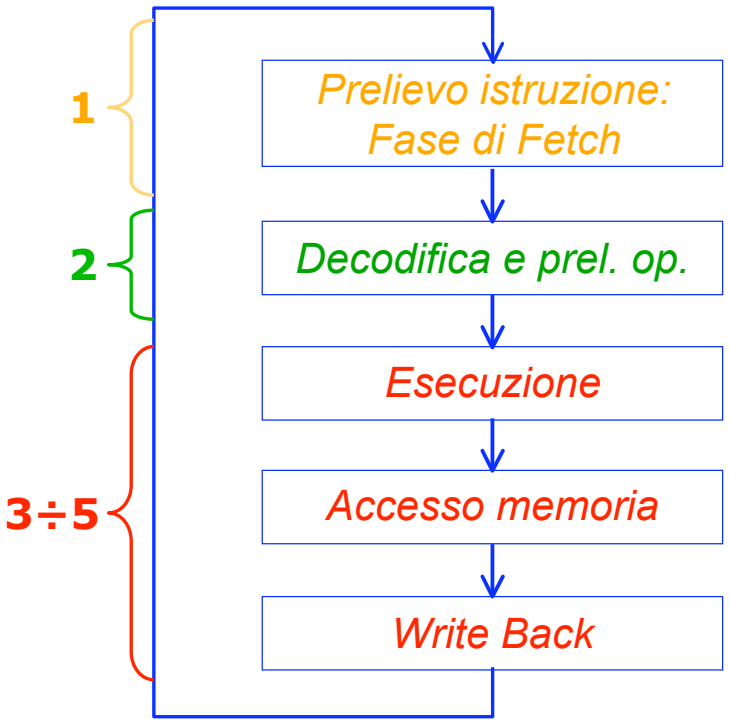
Valutazione prestazioni: CPU multi-ciclo

- CPU multi-ciclo:
  Ogni istruzione dura **solo il numero di cicli di clock necessario**
  - Durata effettiva = media pesata della durata delle istruzioni
  - La CPU multi-ciclo consente una riduzione del tempo di esecuzione
  - Dipende dal tipo di programma (frequenza delle diverse categorie di istruzione)

Istr.	lw	sw	beq	j	R	fp (add)	fp (mul)	Durata MAX	Durata media
Durata	9 ns	8 ns	6 ns	4 ns	7 ns	12 ns	20 ns		
Caso I	24%	12%	18%	2%	44%			**9 ns**	**7.36 ns**
Caso II	31%	21%	5%	2%	27%	7%	7%	**20 ns**	**8.98 ns**

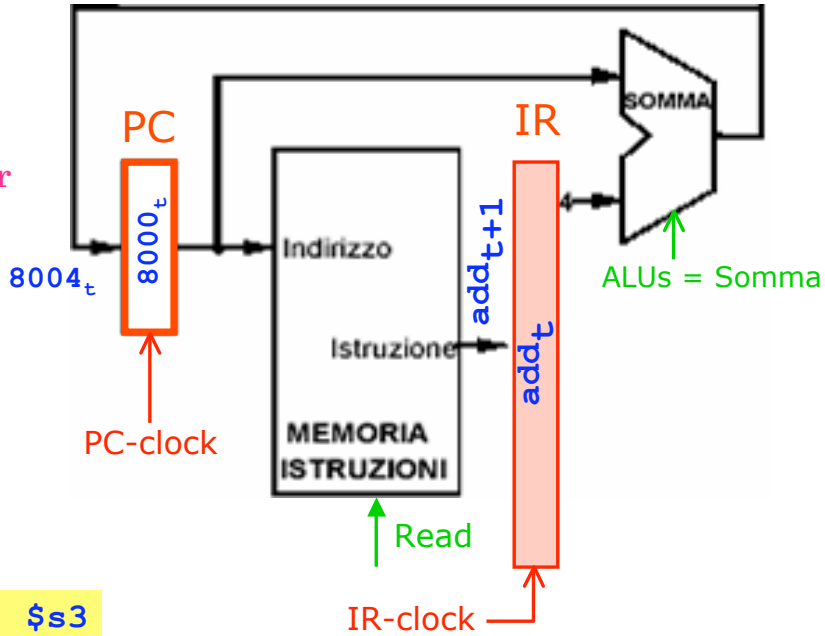
Ciclo di esecuzione di un'istruzione

- Le diverse fasi del ciclo di esecuzione vengono eseguite in momenti diversi
- L'esecuzione dell'istruzione richiederà da 3 a 5 cicli di clock



Circuito della fase di *fetch* multi-ciclo

- Novità:
  - Devo memorizzare l'istruzione: **IR – Instruction Register**
  - **Clock** differenti per **PC** e **IR**
  - Comando esplicito **MemRead** per la **memoria istruzioni**




```
8000:   add $s1, $s2, $s3
8004:   sub $s4, $s1, $t1
.....
```

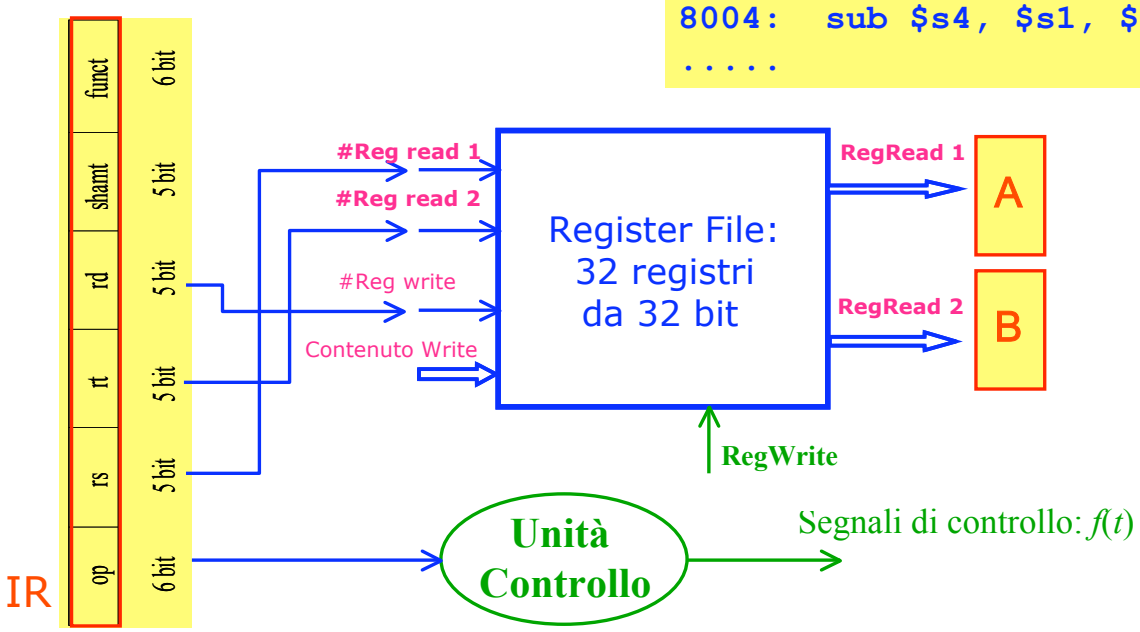

Sommario

- I problemi della UC a singolo ciclo di clock
- Principi ispiratori di una CPU multi-ciclo.
  - Fase di fetch
- Esecuzione multi-ciclo delle istruzioni R
- Esecuzione multi-ciclo delle istruzioni **`lw/sw`**
- Esecuzione multi-ciclo dei salti ed analisi della CPU multi-ciclo.

Tipo R: Decodifica e lettura dei registri (**fase 2**)

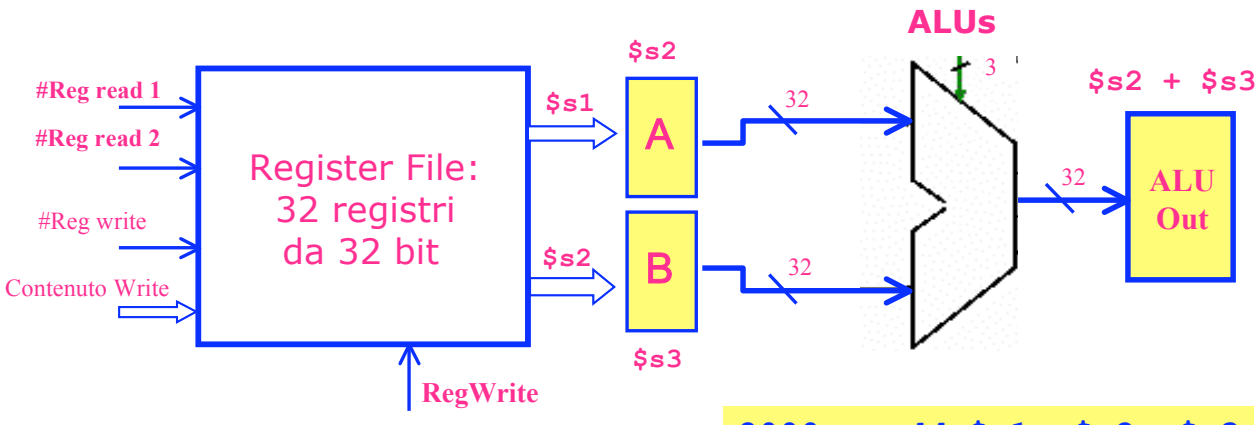
1. Leggo l'istruzione e genero i segnali di controllo opportuni.
2. Leggo il contenuto dei registri.

```
8000:  add $s1, $s2, $s3
8004:  sub $s4, $s1, $t1
.....
```



Fase di esecuzione tipo R (**fase 3**)

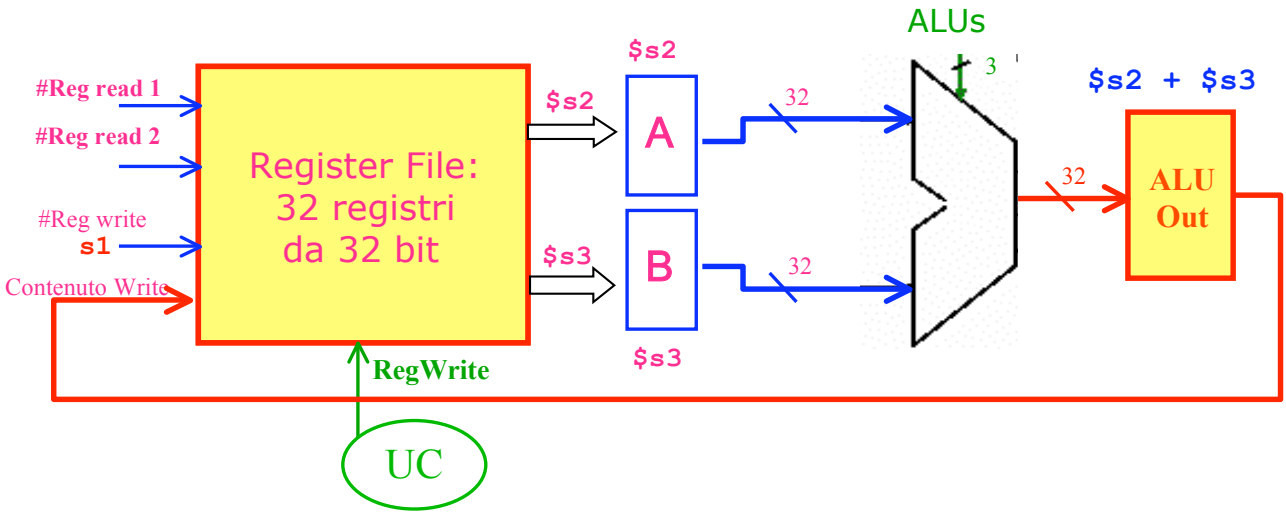
- Il contenuto dei registri **A** e **B** viene mandato all'ALU
- UC genera **ALUs** il codice di: **add**
- Il risultato viene memorizzato in **ALUOut**




```
8000:  add $s1, $s2, $s3
.....
```

Fase di Write-back – tipo R (**fase 4**)

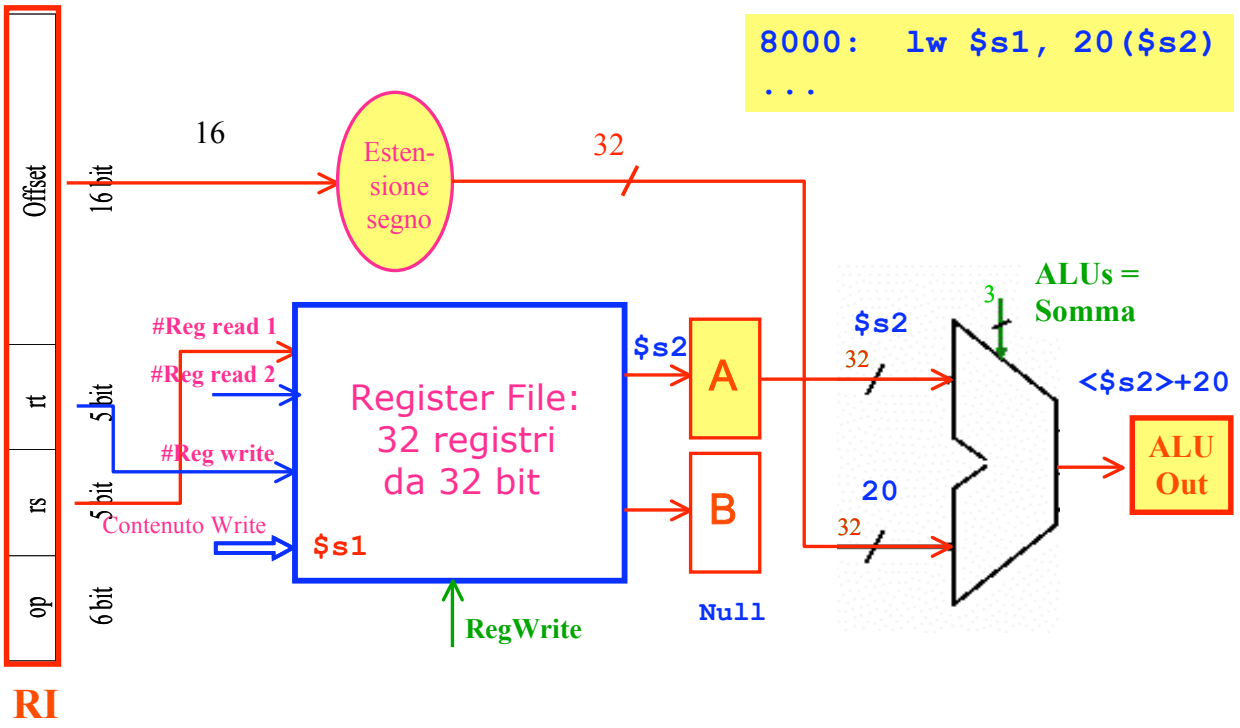
- Il contenuto di **ALUOut** viene mandato alla porta di scrittura del **RF**
- UC genera il segnale di: **RegWrite**
- TOTALE – **tipo R**: **4 cicli** di CLOCK



Sommario

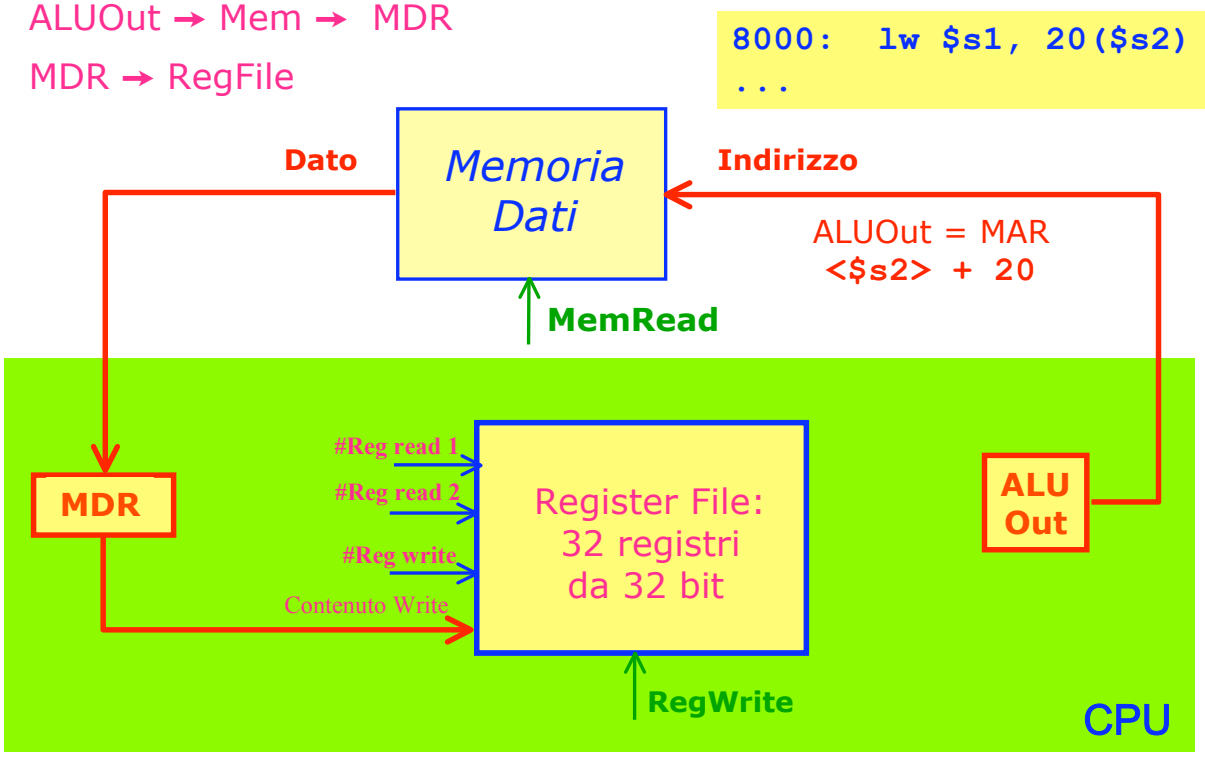
- I problemi della UC a singolo ciclo di clock
- Principi ispiratori di una CPU multi-ciclo. Le fasi di fetch e decodifica.
- Esecuzione multi-ciclo delle istruzioni R
- Esecuzione multi-ciclo delle istruzioni `lw/sw`
- Esecuzione multi-ciclo dei salti ed analisi della CPU multi-ciclo.

Fase 3: esecuzione (tipo I: `lw`)

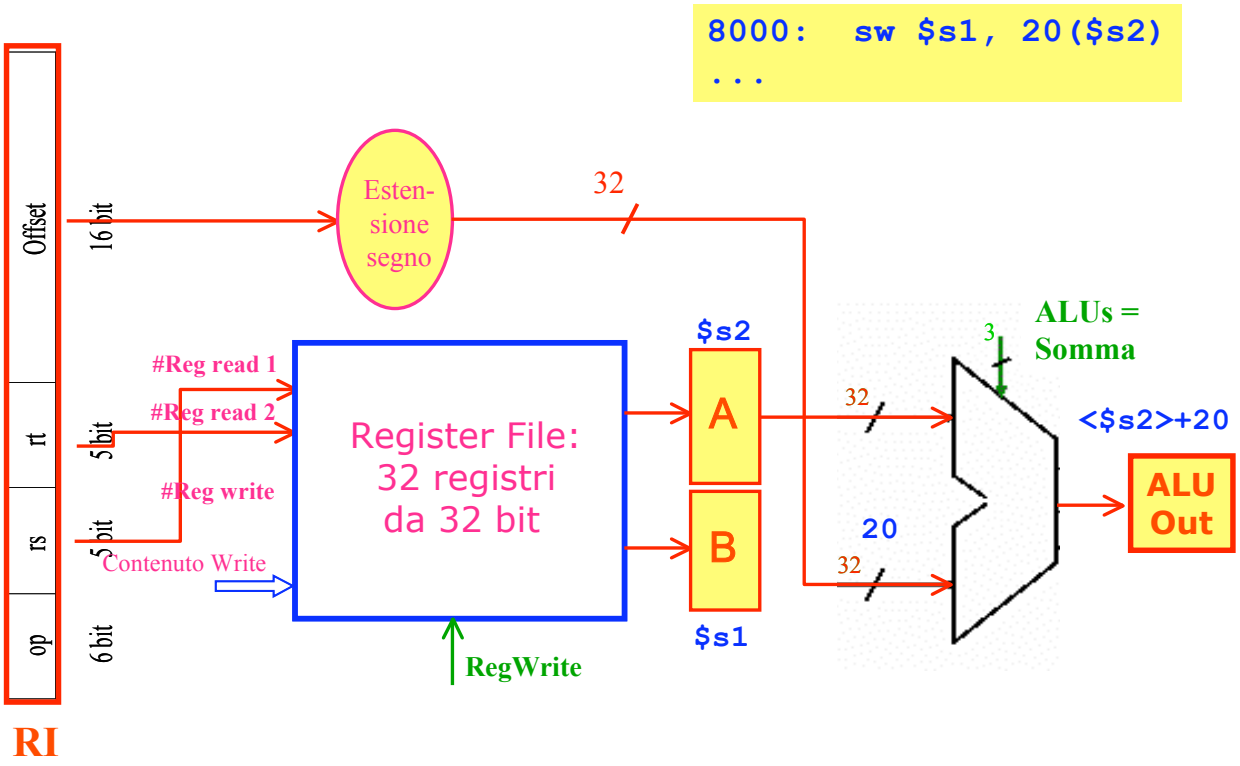


`lw` – Lettura memoria + write-back (fase 4,5)

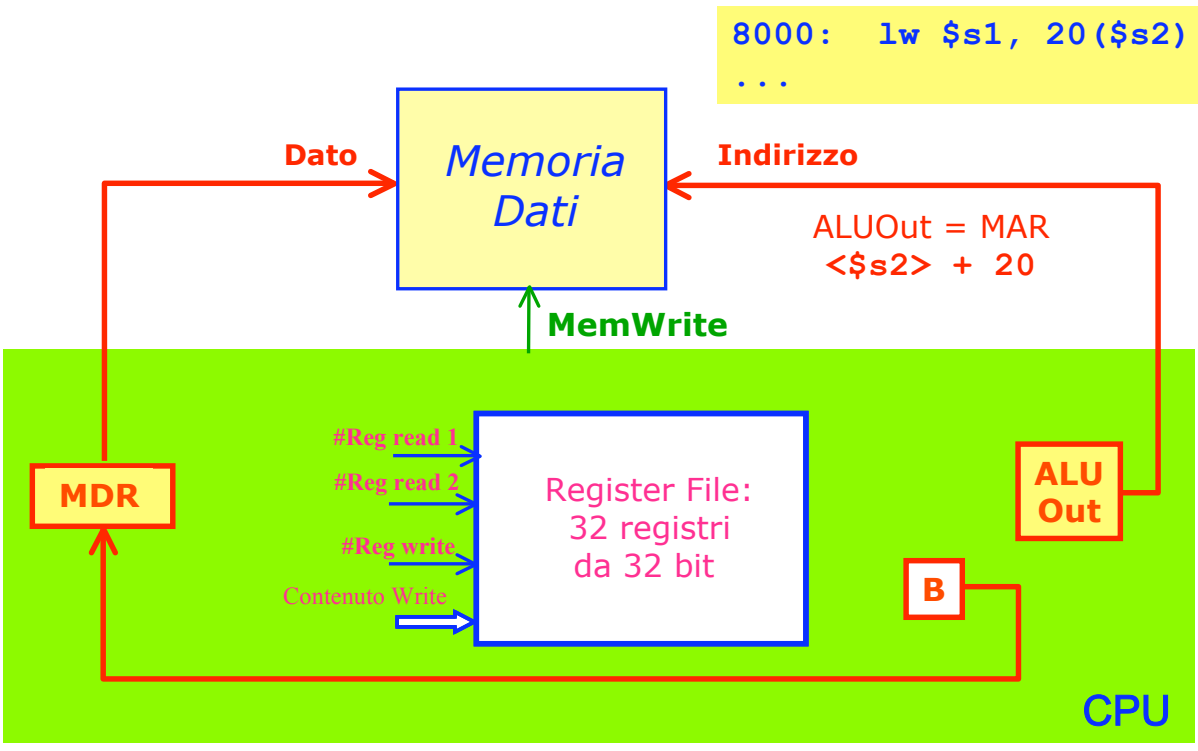
1. ALUOut → Mem → MDR
2. MDR → RegFile



Fase 3: esecuzione (tipo I: sw)



sw: Scrittura nella memoria

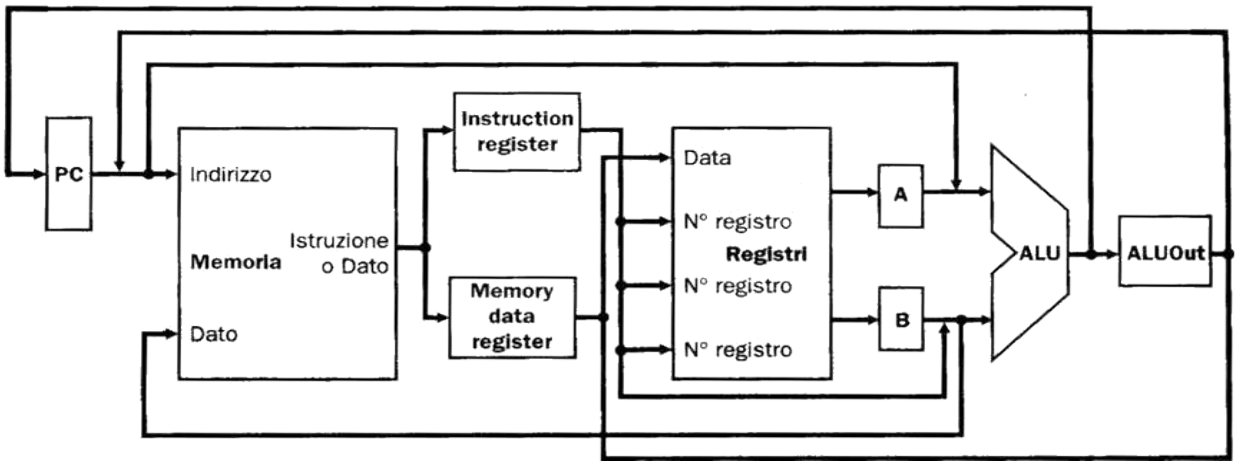


CPU multi-ciclo per istruzioni: tipo R, `lw/sw`

2 Memorie ➔ una sola memoria
a patto di creare 2 registri: un **registro istruzione (RI)** ed un **registro dati (MDR)**

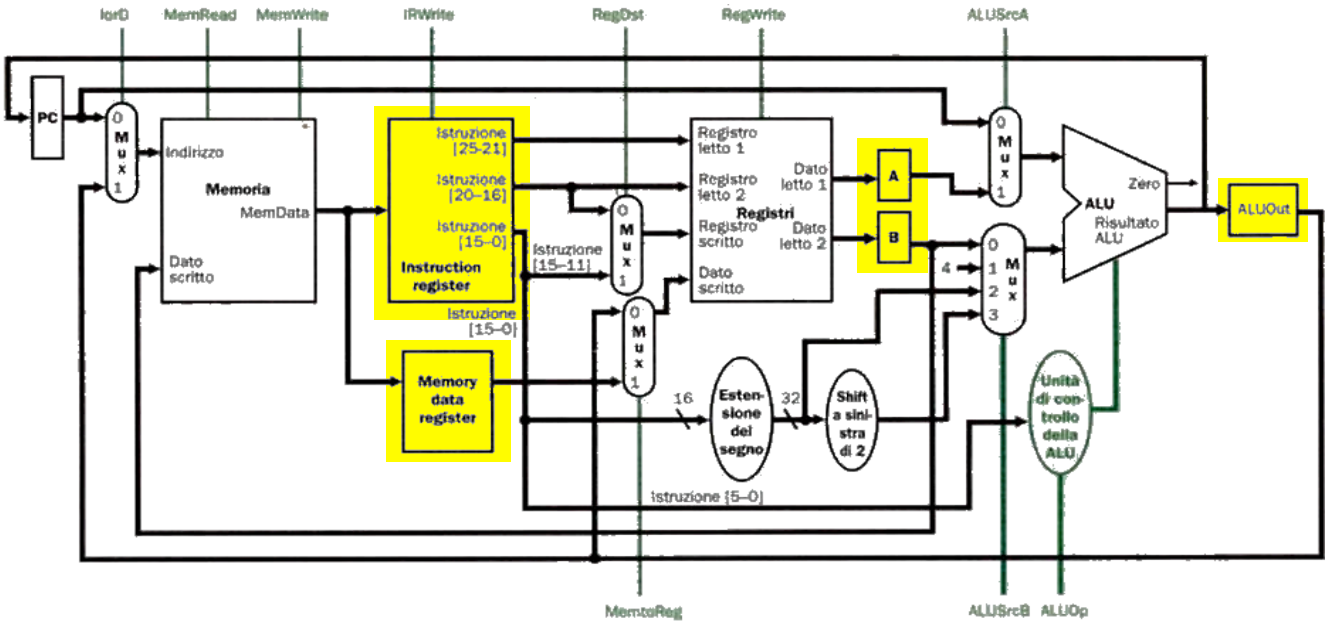
3 ALU ➔ un'unica ALU se viene utilizzata in 3 fasi diverse

- Fase 1: **+ 4 (aggiornamento PC)**
- Fase 2: **`offset + rs`**
- Fase 3: Operazione di **tipo R**



CPU multi-ciclo per istruzioni: tipo R, `lw/sw`

- Una sola ALU
  Una sola memoria (data + istruzioni)



Analisi dell'utilizzo dei registri

PC – Sincronizza il ciclo di esecuzione di un'istruzione. Viene aggiornato in fase di fetch. Dall'inizio della fase di decodifica, contiene l'indirizzo dell'istruzione successiva.

RI – Viene aggiornato alla fine della FF. Mantiene per tutte le fasi, il numero dei registri su cui operare. Questa informazione viene utilizzata da:

Istruzioni di tipo **`beq, sw:`** in fase di decodifica

Istruzioni di tipo **R** e **`lw:`** in fase di decodifica e WriteBack

A, B – Vengono aggiornati alla fine della decodifica. Mantiene il contenuto dei registri Reg Read 1 e 2. Il contenuto del 2° registro può non essere utilizzato (**`lw`**)

ALUOut – Viene aggiornato nei primi 3 stadi.

FF – contiene PC + 4

DECOD – contiene PC + Offset * 4

EXEC – contiene RegRead1 + [RegRead2 *oppure* (Offset x 4)]

MDR – Viene aggiornato solo nelle istruzioni **`lw/sw`**

Register File – Viene aggiornato nella fase di WriteBack nelle istruzioni **R** e **`lw`**

Scrittura nei registri della CPU multi-ciclo

- Tutti i registri della CPU vengono scritti ad ogni colpo di clock.
- Non c'è bisogno di sintetizzare nella UC il segnale di scrittura dei registri…
- …tranne che per **Register File**, **IR** e **PC**:
  - **IR** che deve mantenere il suo valore per tutta la durata dell'esecuzione di un'istruzione
  - **PC** viene aggiornato in fase di fetch.

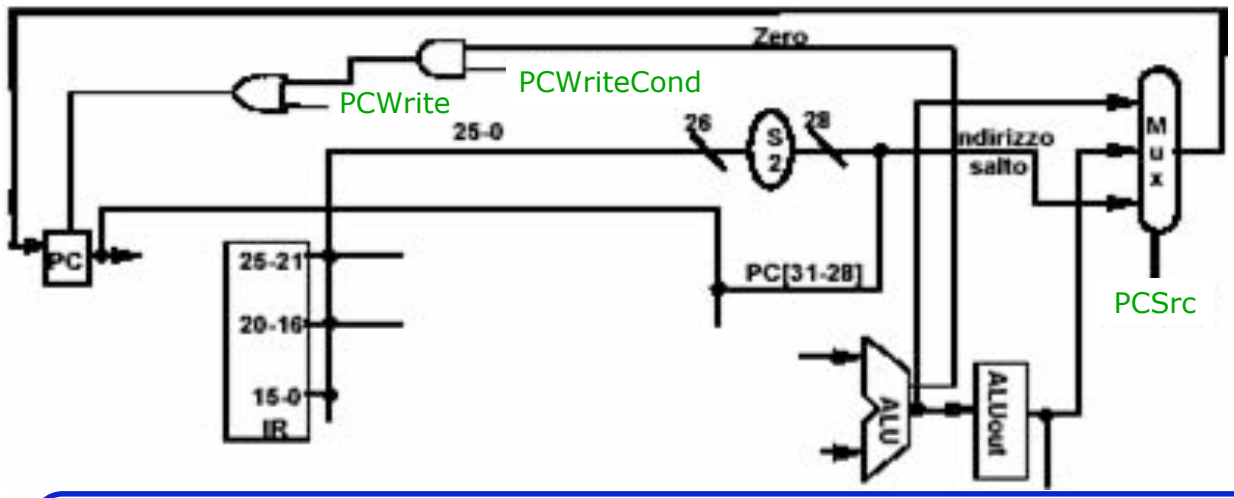
Sommario

- I problemi della UC a singolo ciclo di clock
- Principi ispiratori di una CPU multi-ciclo. Le fasi di fetch e decodifica.
- Esecuzione multi-ciclo delle istruzioni R
- Esecuzione multi-ciclo delle istruzioni **`lw/sw`**
- Esecuzione multi-ciclo dei salti ed analisi della CPU multi-ciclo

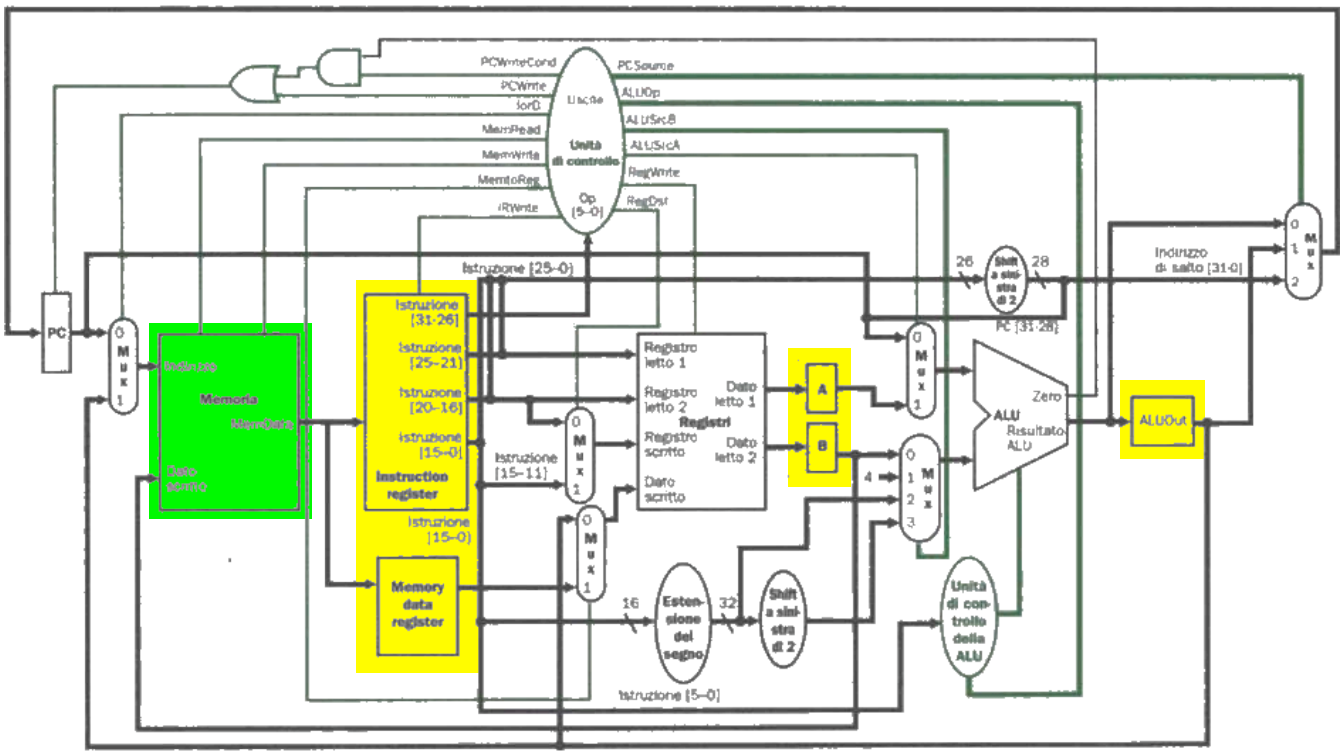
CPU multi-ciclo: i salti

- 3 possibili valori per PC → segnale **PCSrc**: 2 bit
  - Indirizzo ottenuto sommando 4 (fetch)
  - Indirizzo ottenuto dall'ALU (branch)
  - Indirizzo ottenuto dal campo dato dell'IR (jump)
- I 3 valori sono calcolati in sequenza!

PCWrite, PCSrc e PCWriteCond sono coordinati.



CPU multi-ciclo: schema completo



CPU Multi-ciclo: segnali di controllo

- Nuovi segnali, rispetto alla CPU singolo ciclo:

Segnale	Valore	Effetto
ALUSrcA	0	1° operando è il valore attuale del PC
	1	1° operando proviene dalla porta di lettura 1 del RF
ALUSrcB	00	2° operando proviene dalla seconda porta di lettura del RF
	01	2° operando è la costante: **+ 4**
	10	2° operando è l'estensione del segno del campo **offset**
	11	2° operando: **offset** → estensione segno → shift sx di due posizioni
IorD	0	L'indirizzo della memoria proviene dal PC
	1	L'indirizzo della memoria proviene dalla ALU (**ALUOut**)
PCSrc	00	In PC viene scritta l'uscita della ALU (**PC + 4**)
	01	In PC viene scritta il contenuto di **ALUOut** (indirizzo di una **branch**)
	10	In PC viene scrittto l'indirizzo di destinazione della **jump**
PCWrite	0	–
	1	Viene scritto il registro **PC** (indirizzo in PC controllato da **PCSrc**)
PCWriteCond	0	–
	1	Il **PC** viene scritto se anche l'uscita **zero** della ALU è affermata