SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Architettura degli Elaboratori
15 – Gestione eccezioni / Parallelizzazione

Prof. Andrea Sterbini – sterbini@di.uniroma1.it

Argomenti

- Eccezioni e Interrupt
- Parallelizzare la CPU MIPS

Argomenti

- Eccezioni e Interrupt
- Parallelizzare la CPU MIPS

- **Eccezioni e interrupt:** la CPU deve poter rispondere ad **eventi imprevisti:**

Argomenti

- Eccezioni e Interrupt
- Parallelizzare la CPU MIPS

- **Eccezioni e interrupt:** la CPU deve poter rispondere ad **eventi imprevisti:**
- **- segnalazioni da periferiche di I/O esterne alla CPU**
- si tratta di segnali **asincroni e imprevisti**

Argomenti

- Eccezioni e Interrupt
- Parallelizzare la CPU MIPS

- **Eccezioni e interrupt:** la CPU deve poter rispondere ad **eventi imprevisti:**
- **- segnalazioni da periferiche di I/O esterne alla CPU**
- si tratta di segnali **asincroni e imprevisti**
- **- segnalazioni di condizioni di errore interne alla CPU**
- **accesso ad indirizzi di memoria non esistenti**
- **errori nelle operazioni ALU o FPU (overflow/underflow o divisioni per zero)**
- **page faults (accesso a pagine della memoria virtuale non ancora in memoria)**
- **uso di istruzioni sconosciute (che volendo possono essere implementate in SW)**
- **malfunzionamenti hardware (interne o esterne)**

Argomenti

- Eccezioni e Interrupt
- Parallelizzare la CPU MIPS

- **Eccezioni e interrupt:** la CPU deve poter rispondere ad **eventi imprevisti:**
  - **- segnalazioni da periferiche di I/O esterne alla CPU**
    - si tratta di segnali **asincroni e imprevisti**
  - **- segnalazioni di condizioni di errore interne alla CPU**
    - **accesso ad indirizzi di memoria non esistenti**
    - **errori nelle operazioni ALU o FPU (overflow/underflow o divisioni per zero)**
    - **page faults (accesso a pagine della memoria virtuale non ancora in memoria)**
    - **uso di istruzioni sconosciute (che volendo possono essere implementate in SW)**
    - **malfunzionamenti hardware (interne o esterne)**
  - **- richieste al sistema operativo (syscall)**

Gestire l'eccezione

- La gestione di una eccezione cerca di
- **- rispondere all'evento** (se si tratta di una operazione di I/O o una syscall)
- **- oppure riparare la situazione** (se si tratta di un errore risolvibile)

Gestire l'eccezione

- La gestione di una eccezione cerca di
- - **rispondere all'evento** (se si tratta di una operazione di I/O o una syscall)
- - **oppure riparare la situazione** (se si tratta di un errore risolvibile)

- Per prima cosa bisogna **preservare lo stato della computazione**
- - **salvataggio del PC** in un registro (EPC) per poter ripartire (se possibile)
- - per i salti condizionati la CPU non usa flag di stato, che quindi non vanno salvati

Gestire l'eccezione

- La gestione di una eccezione cerca di
- - **rispondere all'evento** (se si tratta di una operazione di I/O o una syscall)
- - **oppure riparare la situazione** (se si tratta di un errore risolvibile)

- Per prima cosa bisogna **preservare lo stato della computazione**
- - **salvataggio del PC** in un registro (EPC) per poter ripartire (se possibile)
- - per i salti condizionati la CPU non usa flag di stato, che quindi non vanno salvati

- Rispetto alle altre istruzioni in pipeline dobbiamo **reagire come un control-hazard**
- ovvero **eliminare anche le istruzioni seguenti** e quella che ha generato l'eccezione
- - overflow o errore (EXE) → le 3 istruzioni in IF, ID ed EXE
- - interruzione esterna (ID) → l'istruzione in IF
- - syscall (EXE) → le 3 istruzioni in IF, ID ed EXE
- - istruzione non definita (ID) → l'istruzione in IF

Gestire l'eccezione

- Per poter rispondere all'evento bisogna **individuarne la causa**
- - si usa un registro dedicato (registro **Causa**)
- - in altre architetture si usano **interruzioni vettorizzate** in cui l'indirizzo del codice di gestione della risposta viene direttamente determinato dal tipo di eccezione

Gestire l'eccezione

- Per poter rispondere all'evento bisogna **individuarne la causa**
- - si usa un registro dedicato (registro **Causa**)
- - in altre architetture si usano **interruzioni vettorizzate** in cui l'indirizzo del codice di gestione della risposta viene direttamente determinato dal tipo di eccezione

- Poi è necessario **eseguire la routine di gestione dell'interruzione** (ovvero cambiare PC)
- - dobbiamo aggiungere al MUX del PC un ingresso con un indirizzo prefissato
- … e alla fine se possibile **tornare ad eseguire il codice iniziale** (ovvero ripristinare PC ← EPC)

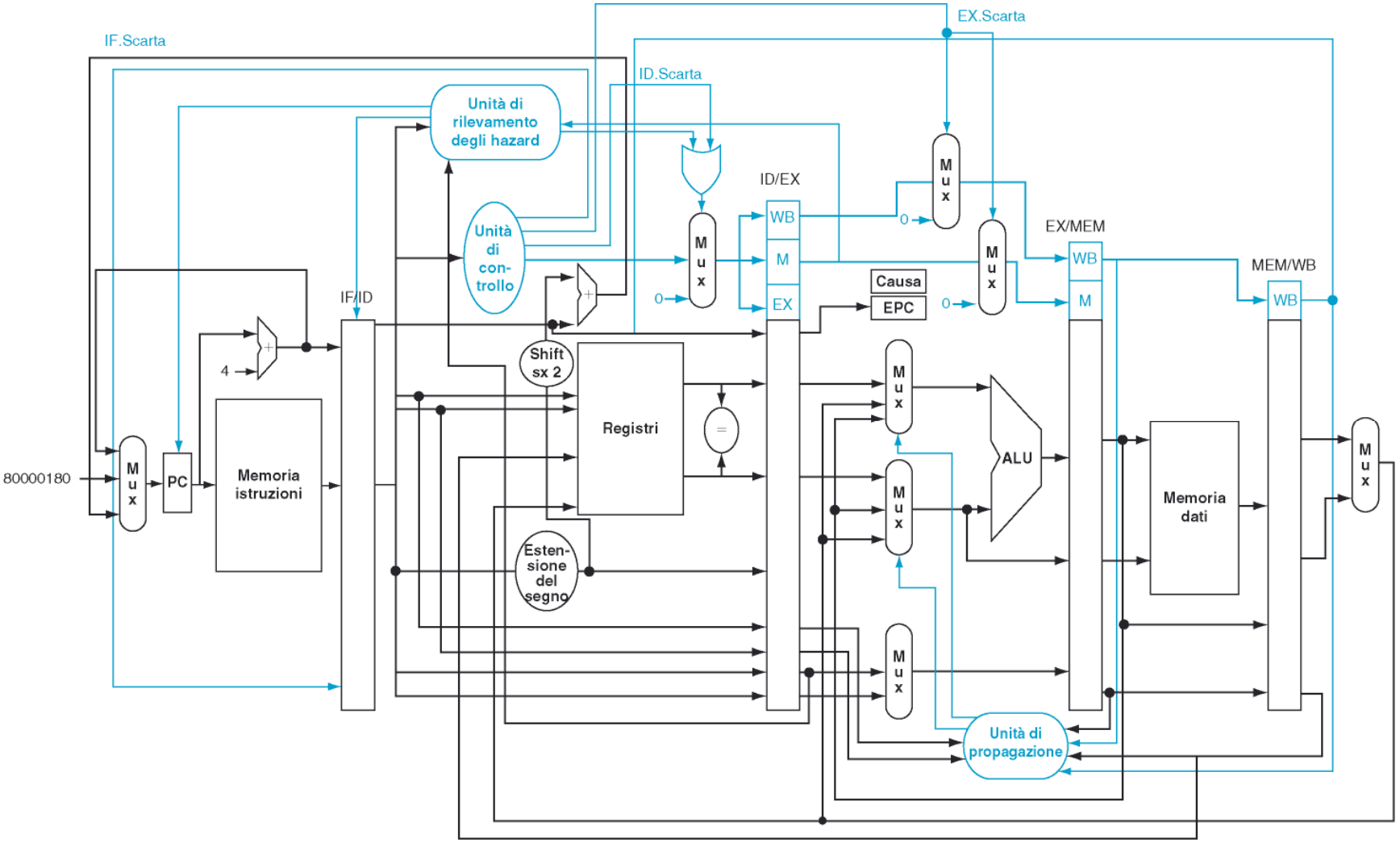
Gestire l'eccezione

- Per poter rispondere all'evento bisogna **individuarne la causa**
- - si usa un registro dedicato (registro **Causa**)
- - in altre architetture si usano **interruzioni vettorizzate** in cui l'indirizzo del codice di gestione della risposta viene direttamente determinato dal tipo di eccezione

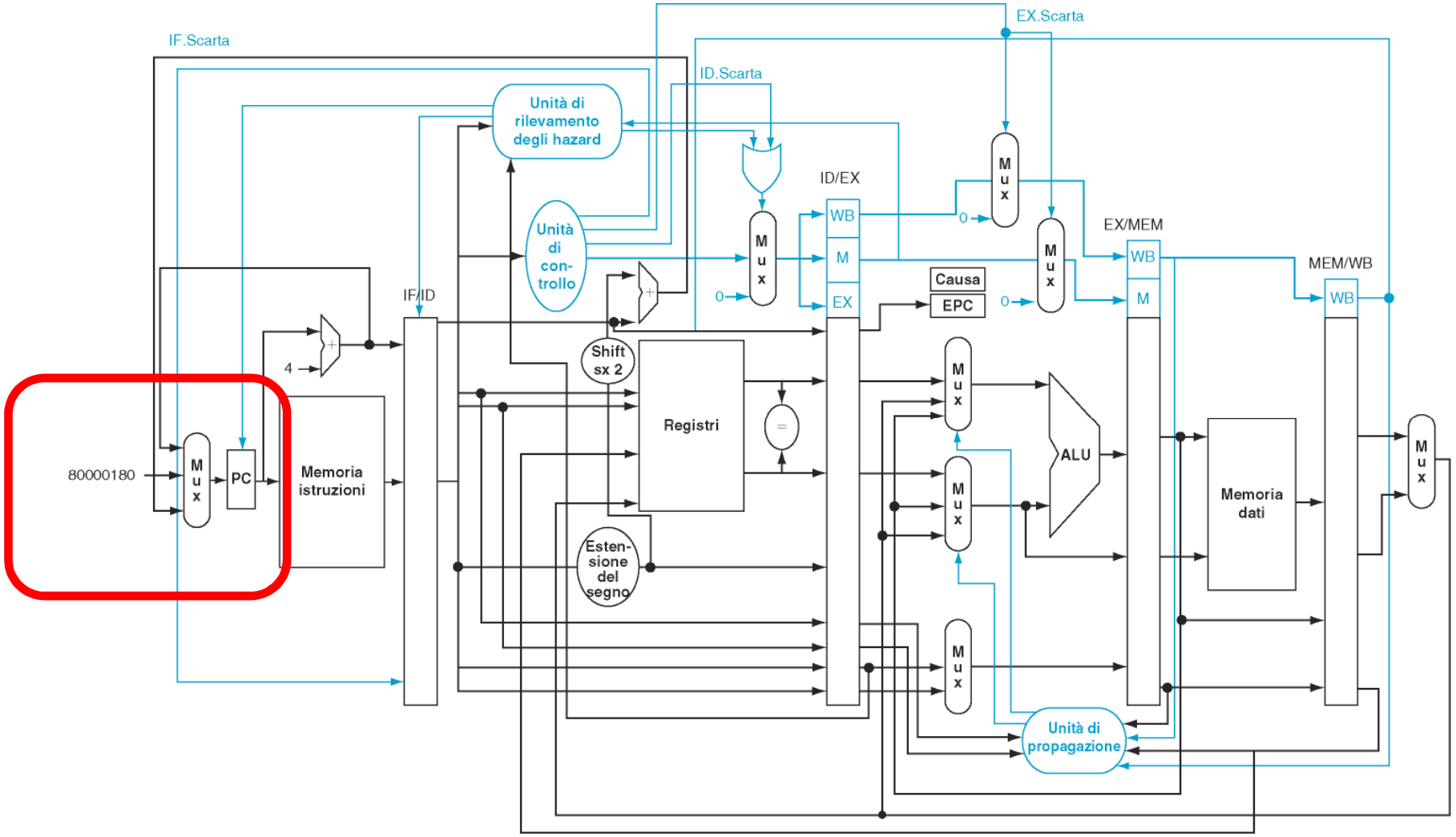
- Poi è necessario **eseguire la routine di gestione dell'interruzione** (ovvero cambiare PC)
- - dobbiamo aggiungere al MUX del PC un ingresso con un indirizzo prefissato
- … e alla fine se possibile **tornare ad eseguire il codice iniziale** (ovvero ripristinare PC ← EPC)

- Se **più interruzioni** avvengono **contemporaneamente**:
- - l'hardware dà una **priorità** alle interruzioni più importanti
- - nel registro Cause sono indicate tutte le interruzioni attive
- → quindi una volta risolta la prima eccezione sarà possibile gestire le altre
- - se necessario si possono disattivare ulteriori interruzioni oppure permettere di interrompere una interruzione meno importante per farne agire una più importante

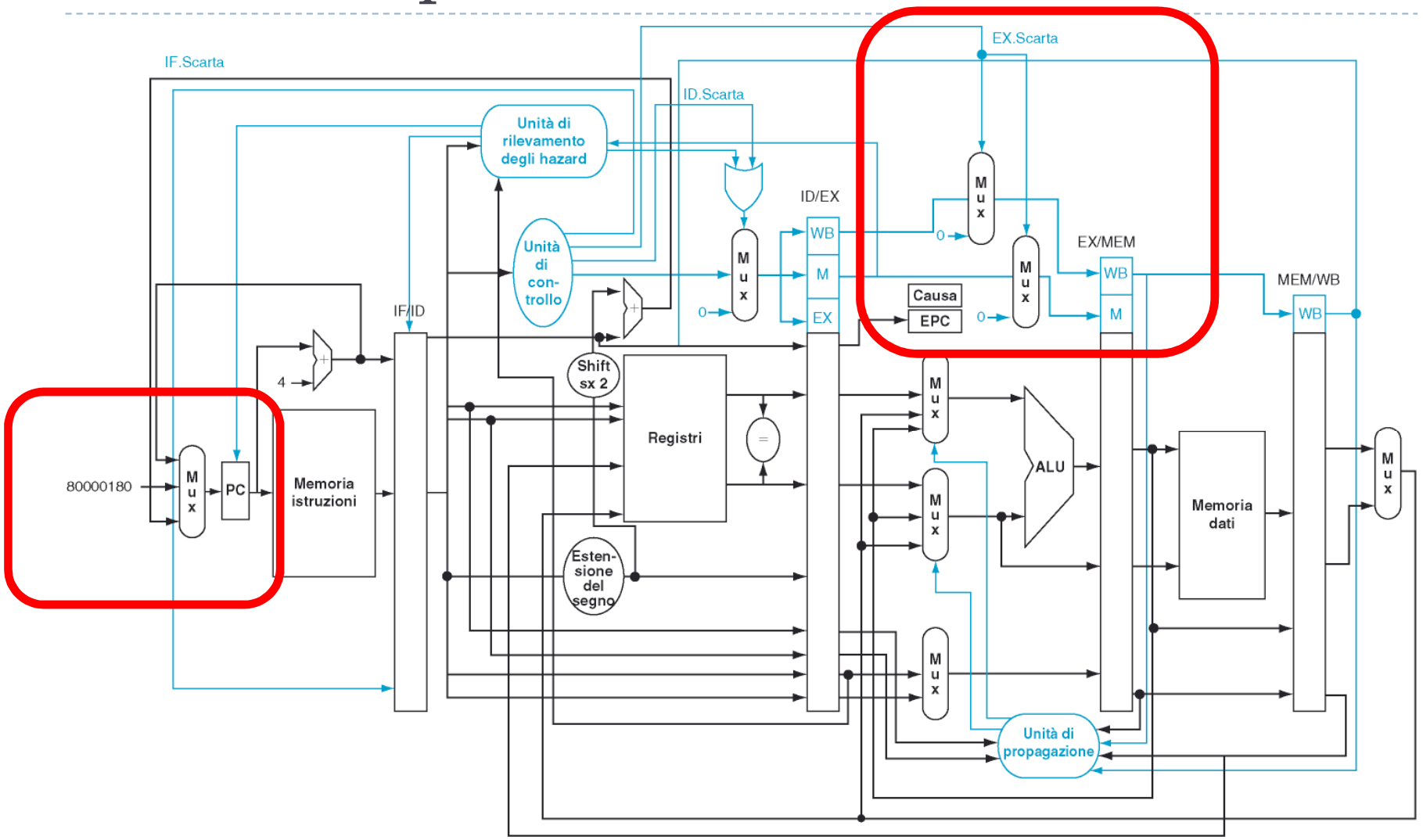
Modifiche per le eccezioni



Modifiche per le eccezioni



Modifiche per le eccezioni



Parallelismo a livello di istruzioni

- Per rendere la CPU **ancora più veloce**:
- possiamo aumentare il livello di parallelismo delle istruzioni usando **più fasi più piccole**
- oppure aggiungere unità funzionali per svolgere **più istr. in ciascuno stadio** della pipeline

Parallelismo a livello di istruzioni

- Per rendere la CPU **ancora più veloce**:
- possiamo aumentare il livello di parallelismo delle istruzioni usando **più fasi più piccole**
- oppure aggiungere unità funzionali per svolgere **più istr. in ciascuno stadio** della pipeline

- Per parallelizzare le istruzioni si può operare:

Parallelismo a livello di istruzioni

- Per rendere la CPU **ancora più veloce**:
- possiamo aumentare il livello di parallelismo delle istruzioni usando **più fasi più piccole**
- oppure aggiungere unità funzionali per svolgere **più istr. in ciascuno stadio** della pipeline

- Per parallelizzare le istruzioni si può operare:
  - Sulla generazione del codice (compilatore)

    → **parallelizzazione statica** dell'esecuzione
  - Oppure a run-time

    → **parallelizzazione dinamica** dell'esecuzione

Parallelismo a livello di istruzioni

- Per rendere la CPU **ancora più veloce**:
- possiamo aumentare il livello di parallelismo delle istruzioni usando **più fasi più piccole**
- oppure aggiungere unità funzionali per svolgere **più istr. in ciascuno stadio** della pipeline

- Per parallelizzare le istruzioni si può operare:
  - Sulla generazione del codice (compilatore)
    → **parallelizzazione statica** dell'esecuzione
  - Oppure a run-time
    → **parallelizzazione dinamica** dell'esecuzione

- Il compilatore, per preparare le istruzioni per la pipeline ad esecuzione parallela deve:

- **raggruppare le istruzioni** da eseguire assieme (**issue slot**)
- **gestire gli hazard** sui dati e sul controllo per evitare data-hazard nello stesso issue slot

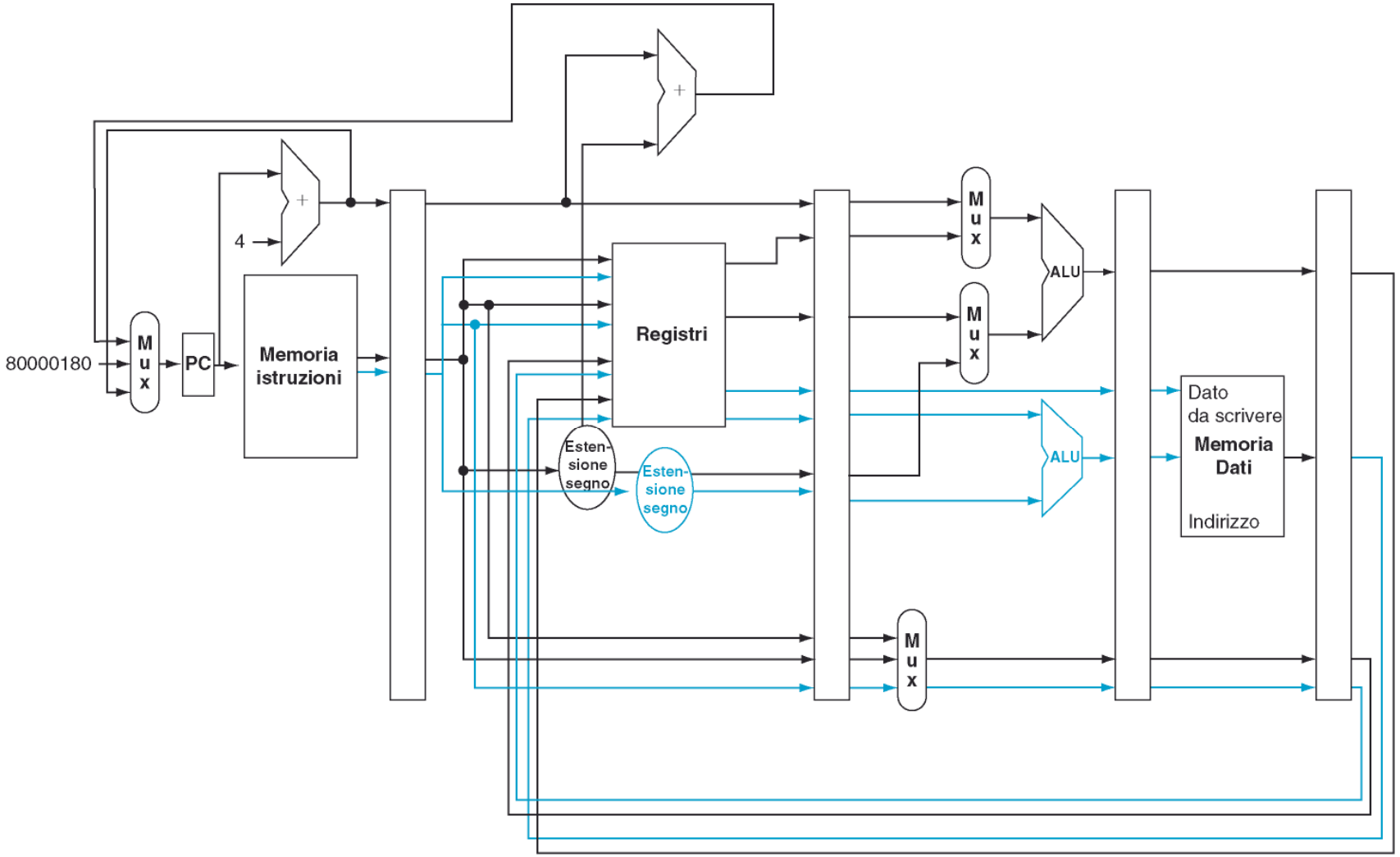
Parallelizzazione statica del MIPS

- Relizziamo **due canali** di esecuzione in parallelo
- il primo dedicato alle sole istruzioni di **load-store**
- il secondo dedicato alle istruzioni di **tipo R ed ai salti condizionati**

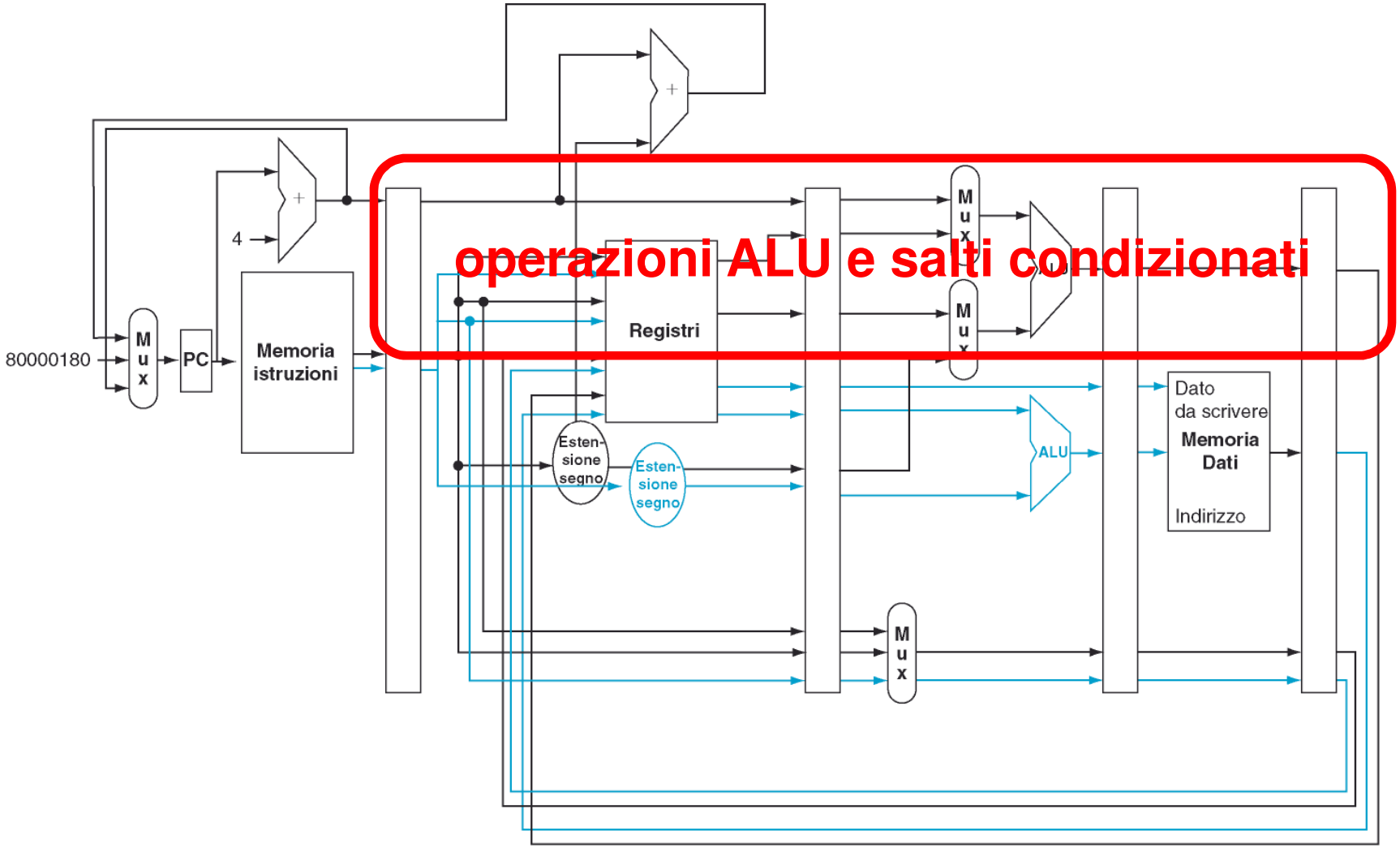
Tipo di istruzioni	Stadi della pipeline								
Load/Store	IF	ID	EXE	MEM	WB				
ALU e Branch	IF	ID	EXE	MEM	WB				
Load/Store		IF	ID	EXE	MEM	WB			
ALU e Branch		IF	ID	EXE	MEM	WB			
Load/Store			IF	ID	EXE	MEM	WB		
ALU e Branch			IF	ID	EXE	MEM	WB		
Load/Store				IF	ID	EXE	MEM	WB	
ALU e Branch				IF	ID	EXE	MEM	WB	

- Vengono caricate più istruzioni per volta: **VLIW** (Very Long Instruction Word)

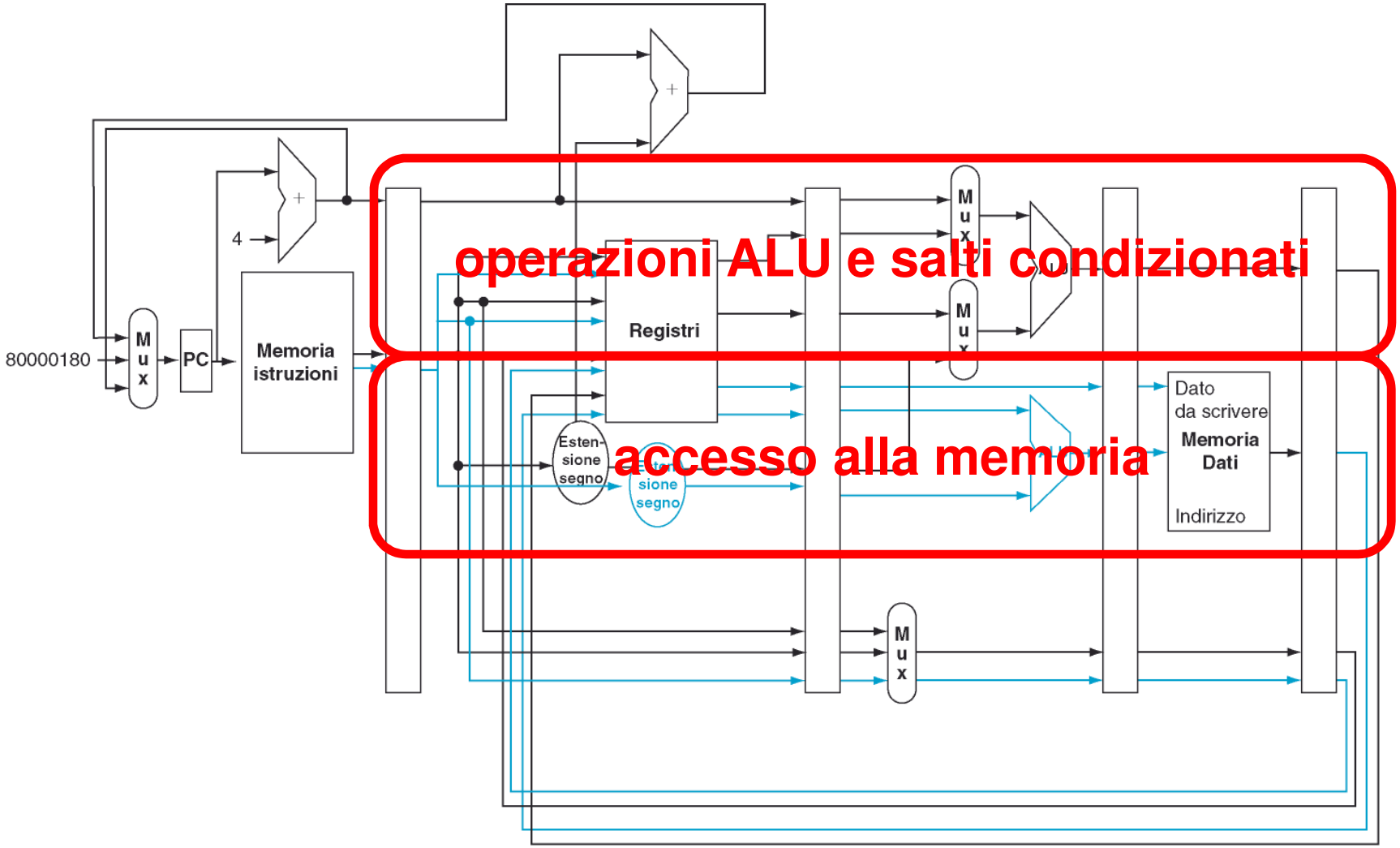
CPU MIPS a due canali



CPU MIPS a due canali



CPU MIPS a due canali



Esempio di parallelizzazione

- Sommiamo il valore di `$s2` a tutti gli elementi di un vettore, dal fondo all'inizio

	Istruzioni ALU o Branch	Istruzioni di trasf. dati	Ciclo di clock
`Ciclo:`		`lw $t0, 0($s1)`	1
	`addi $s1, $s1, -4`		2
	`addu $t0, $t0, $s2`		3
	`bne $s1, $zero, Ciclo`	`sw $t0, 4($s1)`	4

- Come si vede, solo una coppia di istruzioni viene eseguita in parallelo, quindi questo frammento di programma **sembra difficile da parallelizzare** (CPI = 4/5 = 0.8)

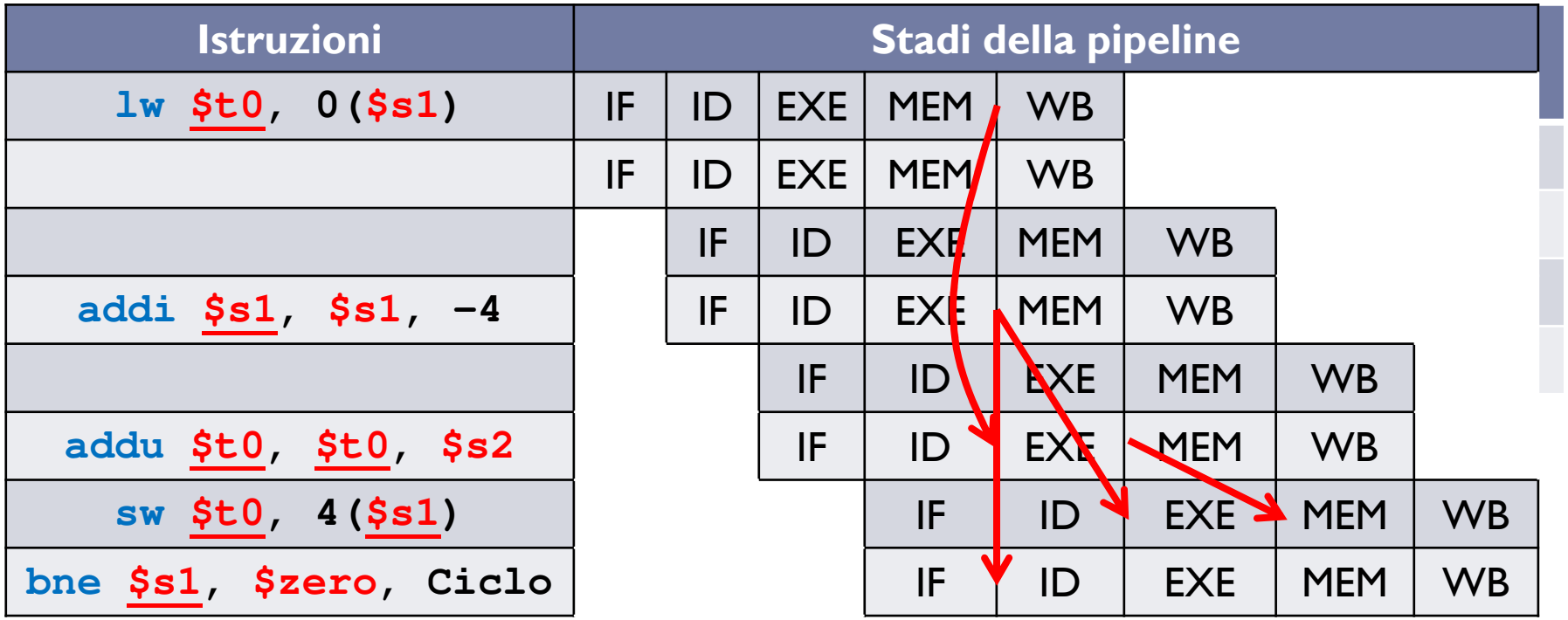
Esempio di parallelizzazione

- Sommiamo il valore di `$s2` a tutti gli elementi di un vettore, dal fondo all'inizio

Istruzioni	Stadi della pipeline							
`lw $t0, 0($s1)`	IF	ID	EXE	MEM	WB			
	IF	ID	EXE	MEM	WB			
		IF	ID	EXE	MEM	WB		
`addi $s1, $s1, -4`		IF	ID	EXE	MEM	WB		
			IF	ID	EXE	MEM	WB	
`addu $t0, $t0, $s2`			IF	ID	EXE	MEM	WB	
`sw $t0, 4($s1)`				IF	ID	EXE	MEM	WB
`bne $s1, $zero, Ciclo`				IF	ID	EXE	MEM	WB

Esempio di parallelizzazione

- Sommiamo il valore di `$s2` a tutti gli elementi di un vettore, dal fondo all'inizio



- **MA:** è la stessa serie di operazioni da svolgere su tutti gli elementi del vettore, SENZA dipendenze tra le operazioni su **elementi diversi** del vettore … facilmente parallelizzabile.
- **IDEA:** eseguire assieme in parallelo più cicli diversi (su elementi diversi)

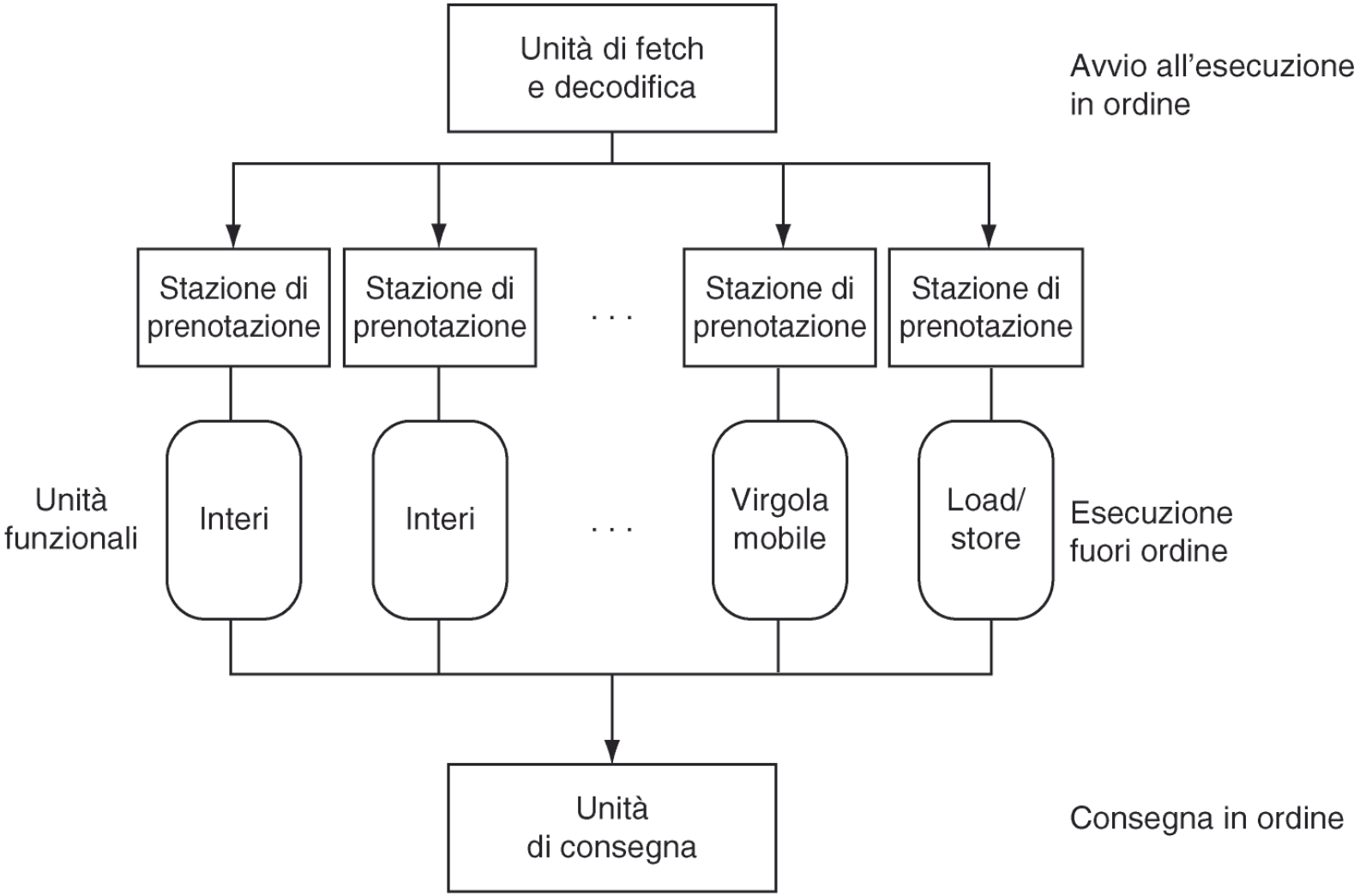
Loop unrolling

- Se il vettore contiene un multiplo di 4 di elementi possiamo **svolgere in parallelo** le operazioni di **4 cicli consecutivi** ed ottenere **maggior parallelismo** (CPI = 8/14 = 0,57)

	Istruzioni ALU o Branch	Istruzioni di trasf. dati	Ciclo di clock
Ciclo:	addi $s1, $s1, -16	lw $t0, 0($s1)	1
		lw $t1, 12($s1)	2
	addu $t0, $t0, $s2	lw $t2, 8($s1)	3
	addu $t1, $t1, $s2	lw $t3, 4($s1)	4
	addu $t2, $t2, $s2	sw $t0, 16($s1)	5
	addu $t3, $t3, $s2	sw $t1, 12($s1)	6
		sw $t2, 8($s1)	7
	bne $s1, $zero, Ciclo	sw $t3, 4($s1)	8

- **NOTA:** è necessario usare più registri (**register renaming**) e modificare gli offset
- **NOTA:** se il vettore è formato da un numero di elementi non multiplo di 4
- possiamo gestire i rimanenti 3 o meno con un loop normale

Parallelizzazione dinamica



AMD Opteron X4 (Barcelona)