


La pipeline

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Università degli Studi di Milano

Riferimento al Patterson: 6 (fino a 6.2)


Sommario

Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

La CPU con pipeline del MIPS

Intuizione della pipeline

Anna, Bruno, Carla e Dario devono fare il bucato.

Devono lavare, asciugare, piegare e mettere via ciascuno un carico di biancheria (4 stadi per la lavorazione del bucato)

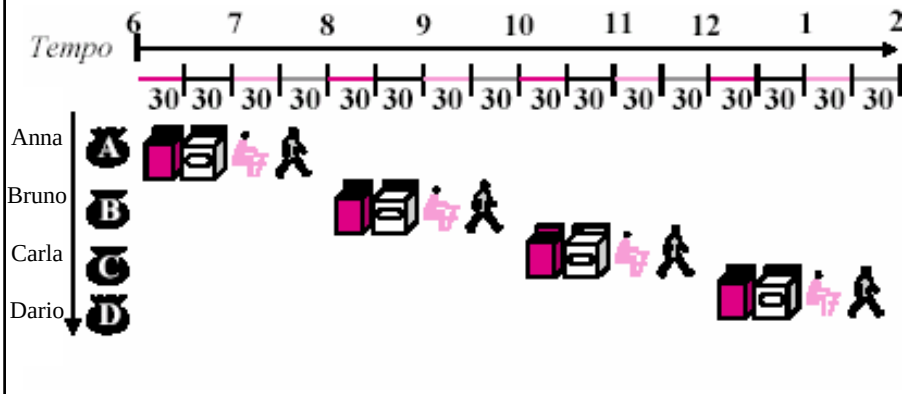
La lavatrice richiede 30 minuti.

L'asciugatrice richiede 30 minuti.

Stirare richiede 30 minuti.

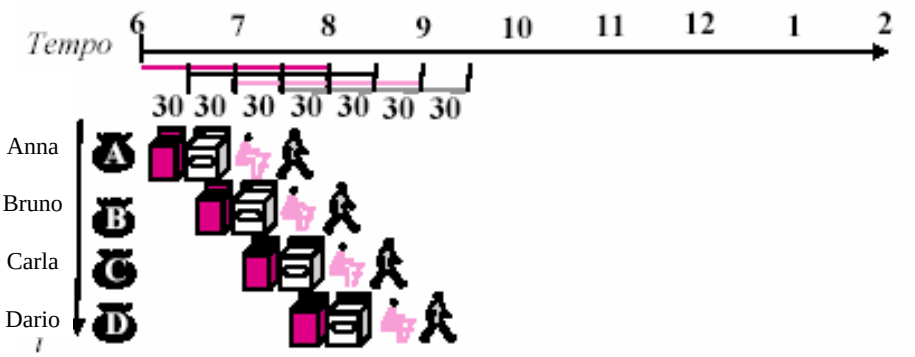
Piegare e mettere via richiede 30 minuti.

La lavanderia sequenziale



In totale vengono richieste 8 ore.

Lavanderia con pipeline



In totale vengono richieste 3.5 ore.

Osservazioni sulla pipeline

Il tempo di ciascuna operazione elementare non viene ridotto.

Gli stadi della pipe-line lavorano in contemporanea perché utilizzano unità funzionali differente.

Le unità funzionali lavorano sequenzialmente (in passi successivi) su istruzioni successive.

Viene aumentato il throughput.


Sommario

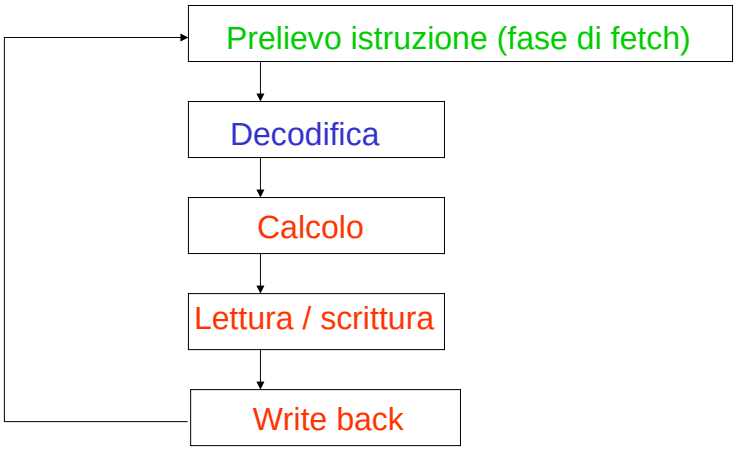
Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

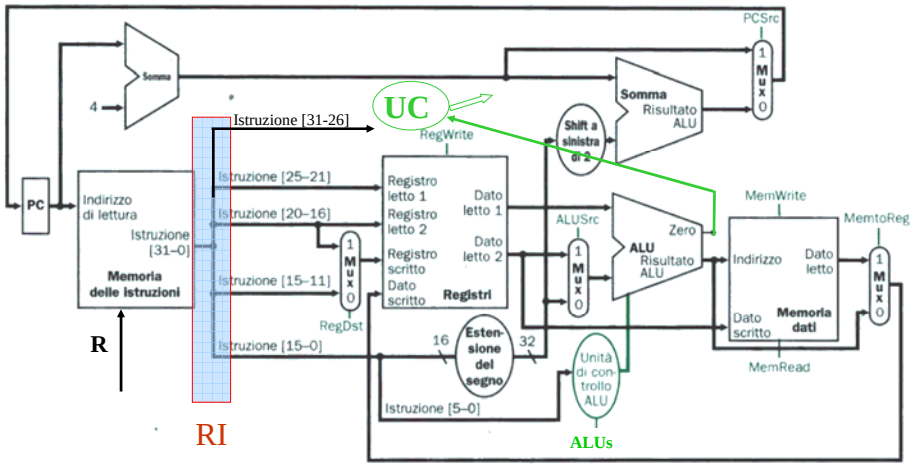
La CPU con pipeline del MIPS


Ciclo di esecuzione di un'istruzione MIPS



Le istruzioni richiedono 5 passi → 5 stadi della pipe-line

Schema generale CPU a ciclo singolo

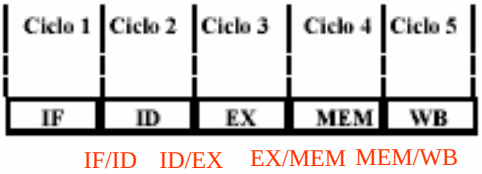


Utilizzo unità funzionali della CPU

Passo esecuzione	ALU	Memoria	Register File
Fase fetch	Yes	Yes	N0
Decodifica	NO*	N0	Yes
Calc – beq	Yes / Yes (salto)	N0	N0
Calc – j	N0	N0	N0
Calc – R	Yes	N0	N0
WB – R	Yes	N0	Yes
Calc – sw	Yes (calcolo indirizzo)	N0	N0
Mem – sw	N0	Yes	N0
Calc – lw	Yes (calcolo indirizzo)	N0	N0
Mem – lw	N0	Yes	N0
WB – lw	N0	N0	Yes

NO* In realtà nella multi-ciclo il calcolo dell'indirizzo di salto veniva anticipato nella fase di decodifica. Si considera qui che le operazioni vengono effettuate nella fase di calcolo.

I 5 stadi della pipeline

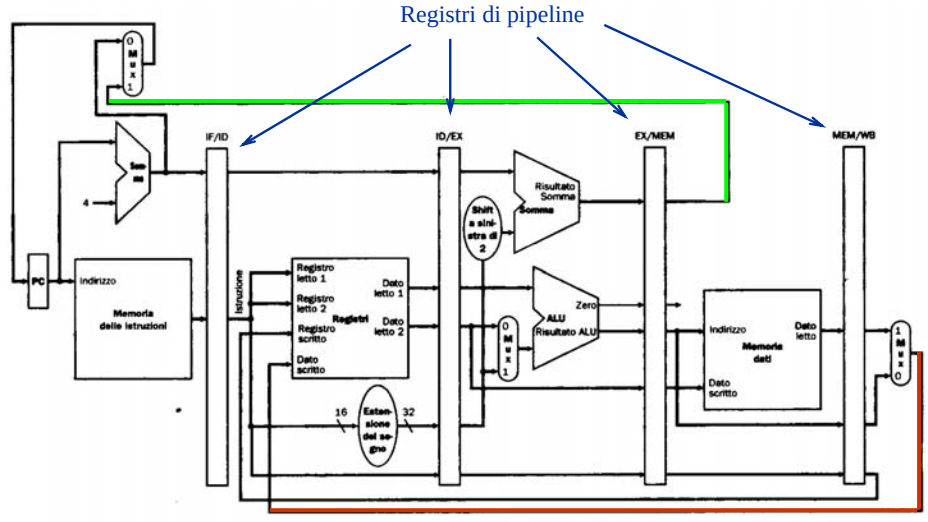


Tra due cicli sono posti dei registri denominati registri di pipe-line.

Nomi degli stadi di pipeline:
- IF: Prelievo istruzione (Instruction fetch)
- ID: Decodifica istruzione (+lettura register file)
- EX: Esecuzione
- MEM: Accesso a memoria (lettura/scrittura)
- WB: Scrittura del register file

NB Uno stadio inizia il suo lavoro quando il clock va basso e trasferisce in quello stadio l'elaborazione effettuata dallo stadio precedente

CPU con pipeline



Il ruolo dei registri

Ciascuno stadio produce un risultato. La parte di risultato che serve agli stadi successivi deve essere memorizzata in un registro.

Il registro mantiene l'informazione anche se lo stadio in questione riutilizza l'unità funzionale.

Esempio: l'istruzione letta viene salvata nel registro IF/ID (cf. Instruction Register).

Sommario

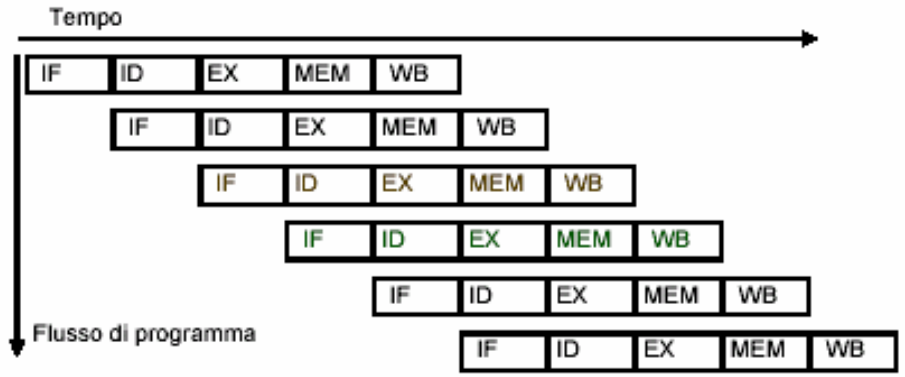
Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

La CPU con pipeline del MIPS

Rappresentazione convenzionale della pipeline



MIPS e pipeline

Fase di fetch semplificata: tutte le istruzioni hanno la stessa lunghezza.

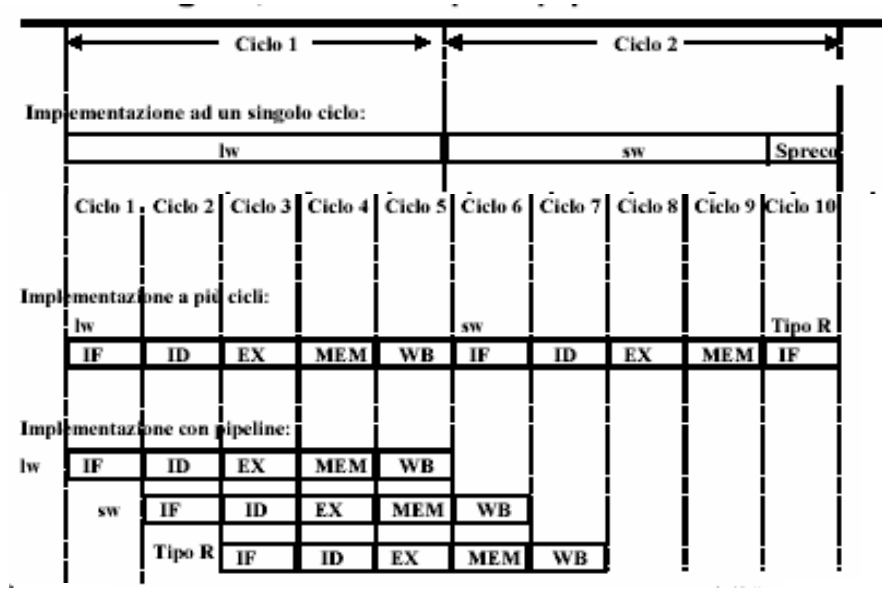
Numero ridotto di formati, i registri sono sempre nella stessa posizione (si può decodificare il codice operativo e leggere i registri).

Non ci sono operazioni sui dati in memoria: se utilizzo la ALU per effettuare dei calcoli, non dovrò accedere alla memoria. Se utilizzo la ALU per calcolare l'indirizzo, accederò alla memoria nell'istante successivo.

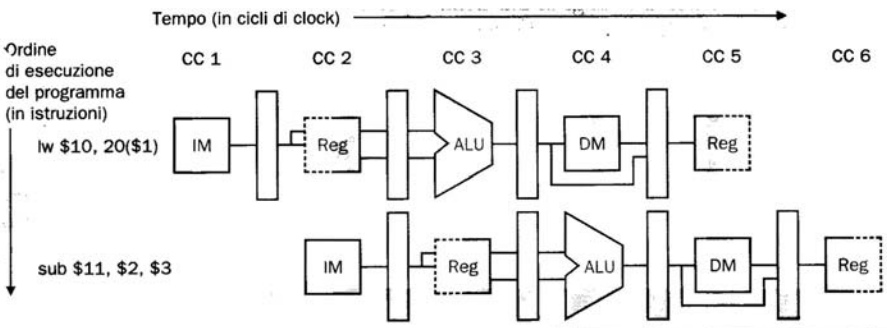
Allineamento delle istruzioni al byte.

Su architetture CISC la pipeline sarebbe più complicata...., ma vedremo che le gerarchie di memoria aiutano a semplificare il problema.

I vantaggi della pipeline



Visualizzazione grafica di una pipeline



Rappresentazione grafica di una istruzione di add in pipeline

Esempio: add $s0, $t0, $t1

I rettangoli grigi a destra indicano lettura, a sinistra indicano scrittura. I componenti bianchi, indicano il loro non utilizzo.

Esempio: pipeline per istruzioni di lw

Passo esecuzione	ALU	Memoria	Register File
IF (Fase fetch)	Yes	Yes	N0
ID (Decodifica)	Yes (indirizzo salto)	N0	Yes
EX (Esecuzione)	Yes	N0	N0
MEM (Accesso memoria)	N0	Yes	N0
WB (riscrittura)	N0	N0	Yes

Istruzioni	IF	ID	EX	MEM	WB	
lw $t0, 8($t2)	MemI, ALUPC	RF	ALU	MemD	RF	
lw $t1, 12($t2)		MemI, ALUPC	RF	ALU	MemD	RF
lw …..			MemI, ALUPC	RF	ALU	MemD
				MemI, ALUPC	RF	ALU
					MemI, ALUPC	RF

Miglioramento delle prestazioni

Il miglioramento massimo è una riduzione del tempo di un fattore pari al numero di stadi della pipe-line.

Nell'esempio precedente (2 istruzioni di lw), il tempo richiesto con la pipe-line è di 12ns contro i 20ns senza pipe-line. Il miglioramento teorico, prevedrebbe 4ns. Allora?

Throughput!! Miglioramento relativo al lavoro globale (con pipe-line senza stalli, 2ns ad istruzione)

Sommario

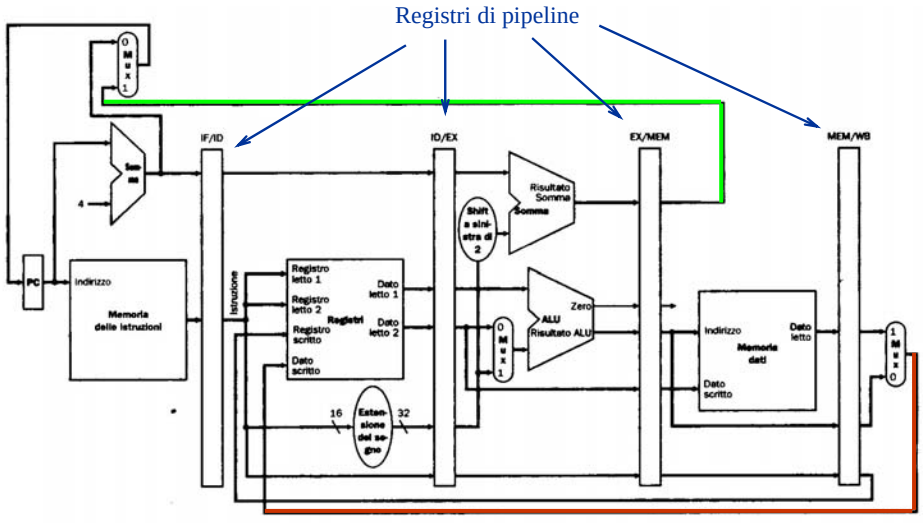
Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

La CPU con pipeline del MIPS

CPU con pipeline



Gli stadi di esecuzione

IF – Instruction Fetch
ID – Instruction Decode (e lettura register file)
EX – Esecuzione o calcolo dell'indirizzo di memoria.
MEM – Accesso alla memoria dati.
WB – Write Back (scrittura del risultato nel register file).

NB: I registri al termine di ogni fase prendono il nome dalle 2 fasi:
IF/ID ID/EX EX/MEM MEM/WB
Perchè non c'è un registro WB/IF?

Il data-path procede da sx a dx.

Da dx a sx si ha la scrittura del PC e la scrittura nel Register File che creano criticità (vanno contro-corrente).

Supponiamo che ciascuno stadio abbia la sua unità di controllo.

Esempo di esecuzione

Cosa si trova nella pipeline durante l'esecuzione di questo segmento di codice?

```
add $s0, $s1, $s2
sub $t0, $t1, $t2
lw  $t1, 24($t2)
beq $s1, $s2, 20
sw $t2, 36($t3)
```

NB Occorre specificare il contenuto della parte master e slave dei registri di pipeline.

Sommario

Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

La CPU con pipeline del MIPS