


Le eccezioni
Introduzione alla pipe-line

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

Università degli Studi di Milano


Sommario

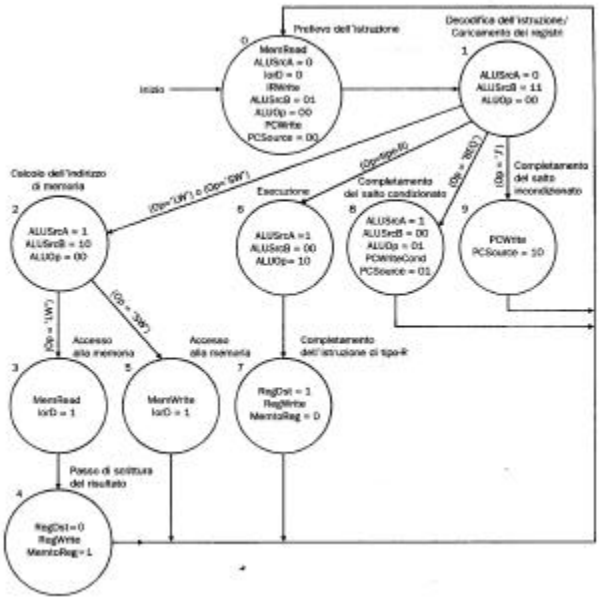
La gestione delle eccezioni in una CPU multi-ciclo

Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

FSM - STG



Eccezioni ed Interrput

Alterano il funzionamento di un programma (funzionalmente equivalenti ad una jump).

Eccezioni. Generamente internamente al processore (e.g. overflow).
Interrupt. Generate esternamente al processore (e.g. richiesta di attenzione da parte di una periferica).

Tipo di evento	Provenienza	Terminologia MIPS
Richiesta di un dispositivo di I/O	Esterna	Interrupt
Chiamata al SO da parte di un programma	Interna	Eccezione
Overflow aritmetico	Interna	Eccezione
Uso di un'istruzione non definita	Interna	Eccezione
Malfunzionamento dell'hardware	Entrambe	Eccezione o Interruzione

Azioni in risposta ad un'eccezione

Le eccezioni vengono gestite dal SO.

1) Salvataggio dell'indirizzo dell'istruzione incriminata (PC-4) nell'EPC.

2) Trasferimento del controllo al SO (questo potrà eventualmente terminare il programma segnalando errore).

Hardware addizionale

Registro EPC: è un registro a 32 bit utilizzato per memorizzare l'indirizzo dell'istruzione coinvolta.

Registro causa: è un registro utilizzato per memorizzare la causa dell'eccezione; in MIPS sono 32 bit:
- bit 0 = 0 istruzione indefinita.
- bit 0 = 1 overflow aritmetico.

Segnali di controllo:
EPCWrite – scrittura nel registro EPC.
CausaWrite – scrittura nel registro Causa.
CausaInt – Dato per il registro Causa.

Set degli input della FSM modificato:
Aggiunta di Overflow.

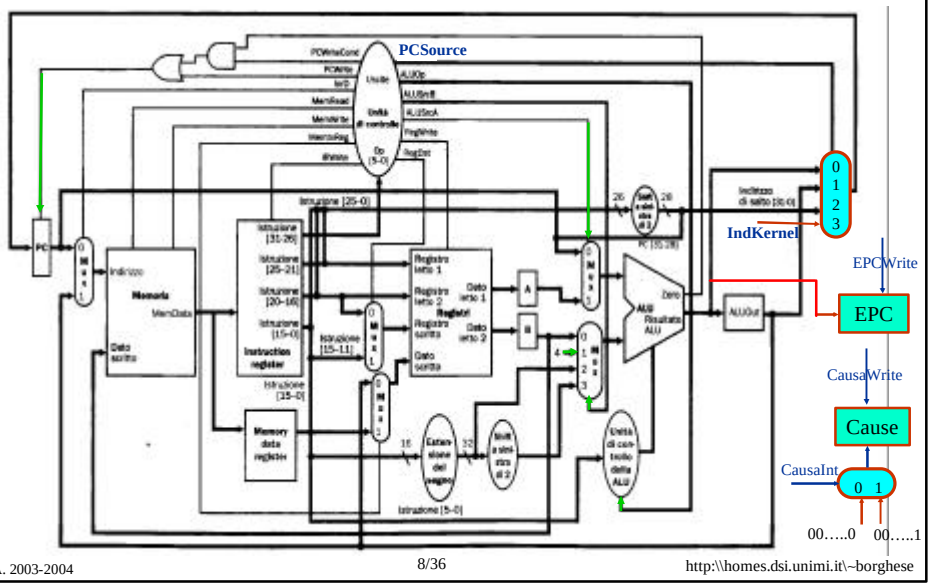
Modifiche alla FSM della CPU

Istruzione indefinita.
Non esiste, al passo 2 (decodifica), uno stato futuro valido.
Si aggiunge un nuovo stato futuro indicato con 'altro', è lo
Stato corrispondente al verificarsi dell'eccezione.

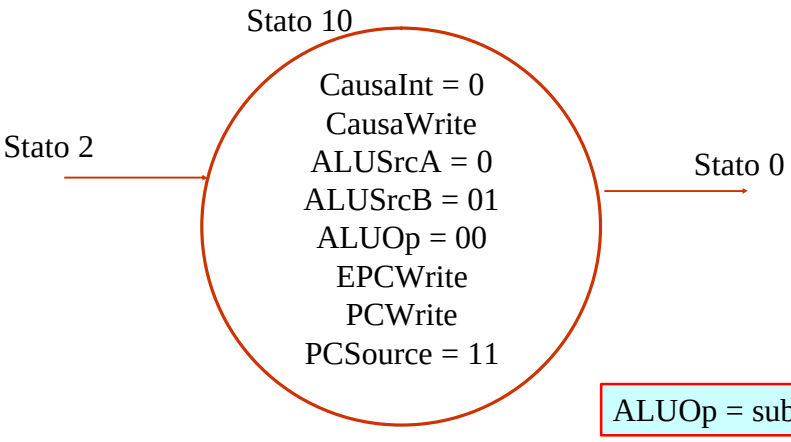
Overflow aritmetico.
Al passo 4 di esecuzione dell'operazione, lo stato futuro è scelto in funzione del segnale di overflow (nuovo input alla FSM). Ho già eseguito la somma e sto scrivendo il risultato nel register file.

Collegamenti HW per la risposta ad un'eccezione

Salvataggio del PC: PC = PC – 4 PC → EPC; Selezione della causa; Segnali di controllo in verde.

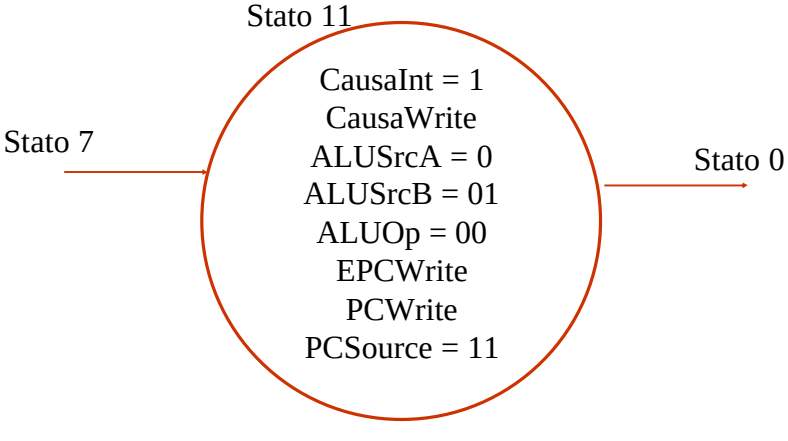


Stato OpCode not valid

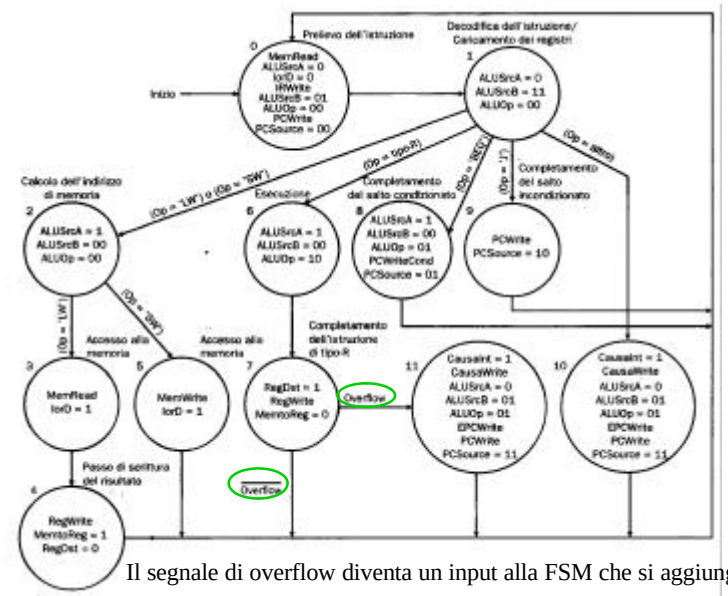


ALUOp = sub: PC-4

Stato Overflow



FSM per la CPU multi-ciclo con gestione delle eccezioni



Il segnale di overflow diventa un input alla FSM che si aggiunge ad OpCode.

Il coprocessore 0

Contiene i registri richiesta per la gestione delle eccezioni e per la gestione della memoria

Nome del registro	Numero del registro in coprocessore 0	Utilizzo
Bad/Addr	8	Registro contenente l'indirizzo di memoria a cui si è fatto riferimento
Stato	12	Maschera delle interruzioni e bit di abilitazione. Gestisce i diversi livelli di priorità.
Causa	13	Tipo dell'interruzione e but dell'interruzione pendente
EPC	14	Registro contenente l'indirizzo dell'istruzione che ha causato l'interruzione.

Interrupt gerarchici

Accodamento degli input: FIFO.
Annidamento degli input: LIFO (cf. stack)

Occorre definire una priorità.

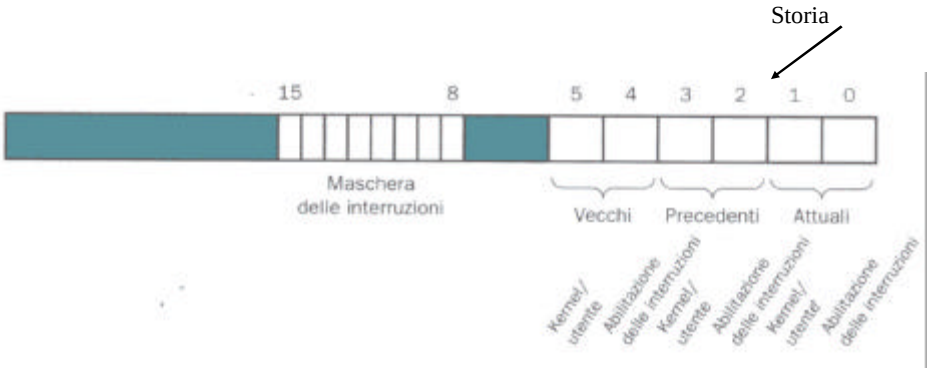
Maschere di interrupt.
- Ad ogni interrupt viene associato un livello.
- Il livello corrente è associato allo stato del sistema.
- La maschera dell'interrupt viene memorizzata nello status register (8 livelli per il MIPS)
- La maschera degli interrupt pending viene caricata nel cause register. Per ogni bit a 1, esiste pending un interrupt di quel livello.

Codifica di priorità.

Mascheramento delle interruzioni

Registro di stato codifica le priorità.

5 livelli di interruzione HW + 3 livelli di interruzione SW.
Inizializzazione con tutti i bit = 1. Interruzione al livello k, possibile solo se il bit corrispondente = 1.



Il coprocessore 0 - Istruzioni

Set di istruzioni che lavora sul coprocessore 0:

lwc0 $<reg_c0> <offset>($reg)
swc0 $<reg_c0> <offset>($reg)

mfc0 $<reg>, $<reg_c0> pseudo-istruzione
mtc0 $<reg_c0>, $<reg> pseudo-istruzione

Tipo di risposta ad un'eccezione

E' software (SO)

Vettorializzata: ciascuna eccezione rimanda ad un indirizzo diverso del SO. Gli indirizzi sono spaziati equamente (8 parole). Dall'indirizzo si può ricavare la causa dell'eccezione.

Tramite registro: detto registro **causa**. Il SO ha un unico entry point per la gestione delle eccezioni. La prima istruzione è di decodifica della causa dell'eccezione. L'entry point è forzato tramite PCSource = 11.

Sommario

La gestione delle eccezioni in una CPU multi-ciclo

Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

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)

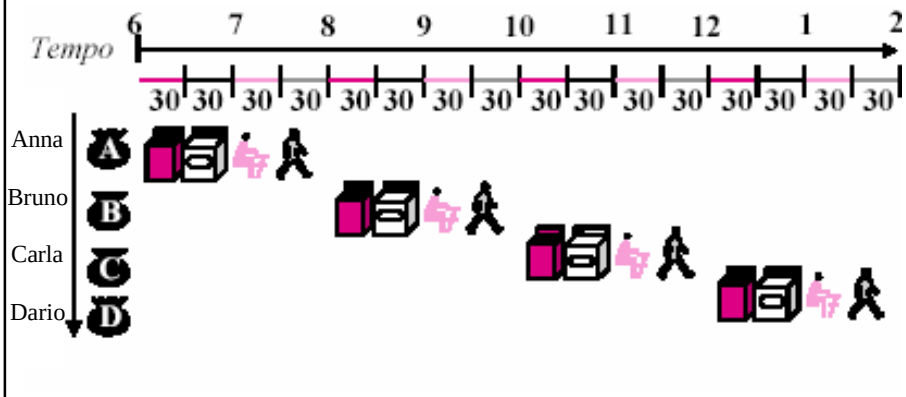
La lavatrice richiede 30 minuti.

L'asciugatrice richiede 30 minuti.

Stirare richiede 30 minuti.

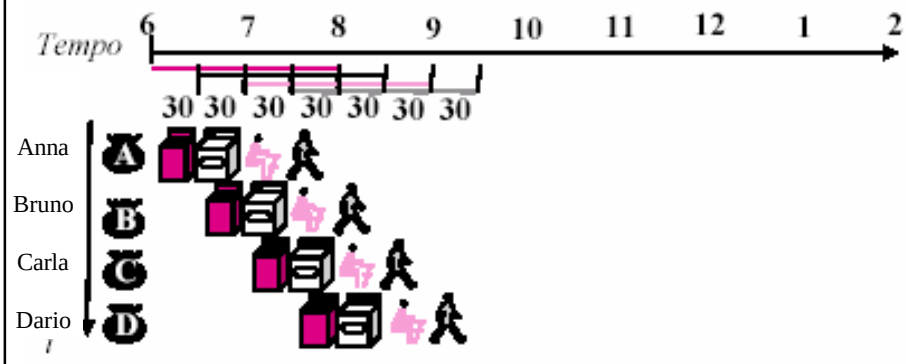
Piegare e mettere via richiede 30 minuti.

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

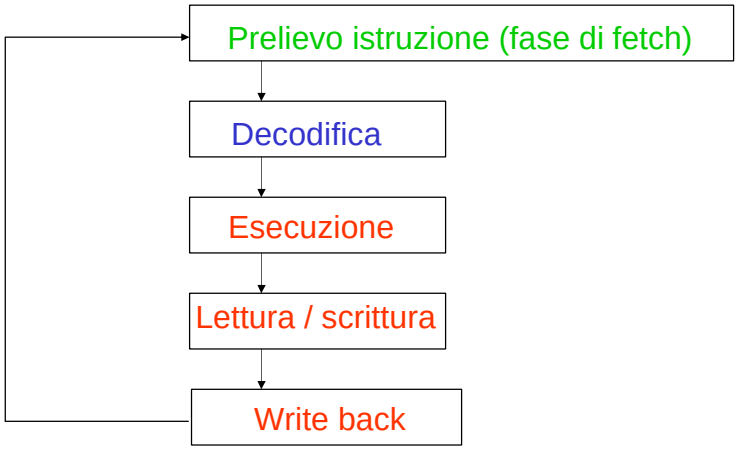
La gestione delle eccezioni in una CPU multi-ciclo

Introduzione sulla pipeline

Gli stadi della pipeline

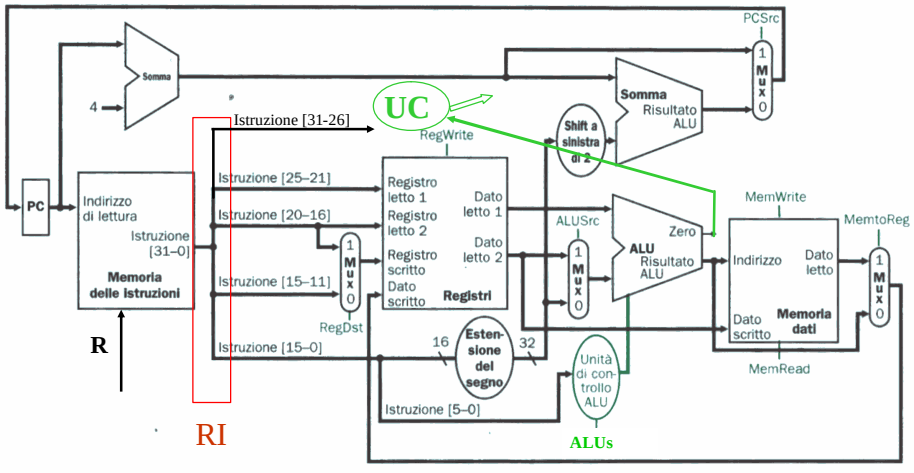
Rappresentazione del flusso di esecuzione in una pipeline

Ciclo di esecuzione di un'istruzione MIPS



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

Schema generale CPU a ciclo singolo

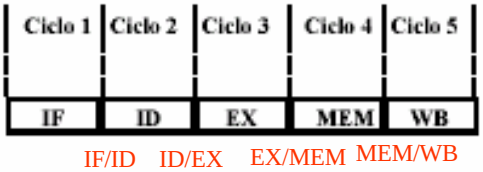


Utilizzo unità funzionali della CPU

Passo esecuzione	ALU	Memoria	Register File
Fase fetch	Yes	Yes	N0
Decodifica	NO*	N0	Yes
Exec I – beq	Yes / Yes (salto)	N0	N0
Exec I – j	N0	N0	N0
Exec I – R	Yes	N0	N0
Exec II – R	Yes	N0	Yes
Exec I – sw	Yes (calcolo indirizzo)	N0	N0
Exec II – sw	N0	Yes	N0
Exec I – lw	Yes (calcolo indirizzo)	N0	N0
Exec II – lw	N0	Yes	N0
Exec III – lw	N0	N0	Yes

NO* In realtà nella multi-ciclo il calcolo dell'indirizzo di salto veniva anticipato nella fase di decodifica.

I 5 stadi della pipeline

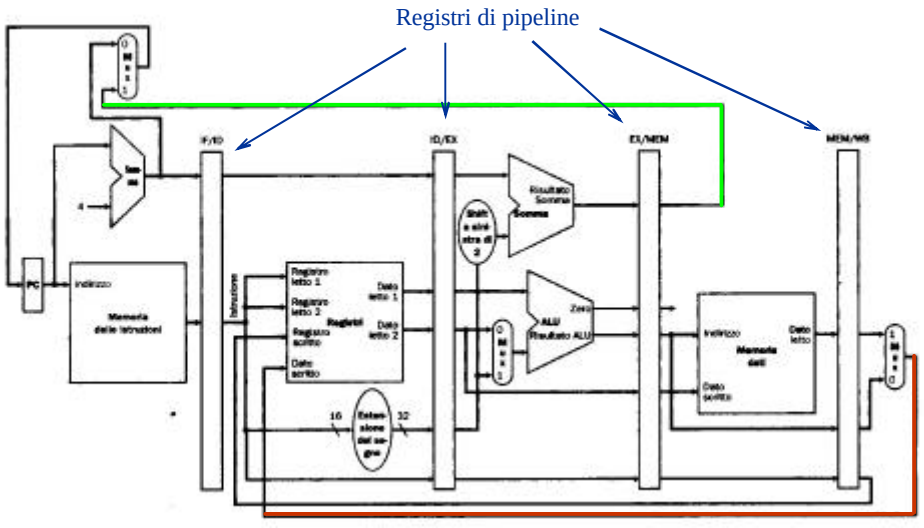


Tra due cicli sono posti dei registri denominati registri di pipe-line.

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



Sommario

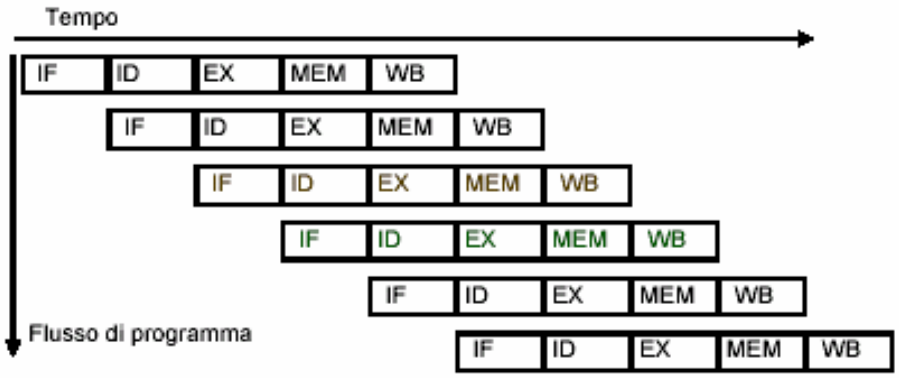
La gestione delle eccezioni in una CPU multi-ciclo

Introduzione sulla pipeline

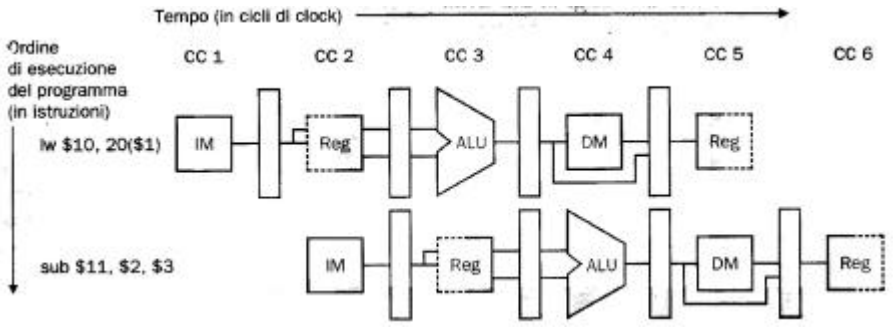
Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline

Rappresentazione convenzionale della pipeline

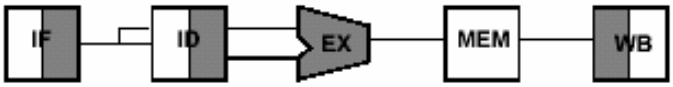


Visualizzazione grafica di una pipeline



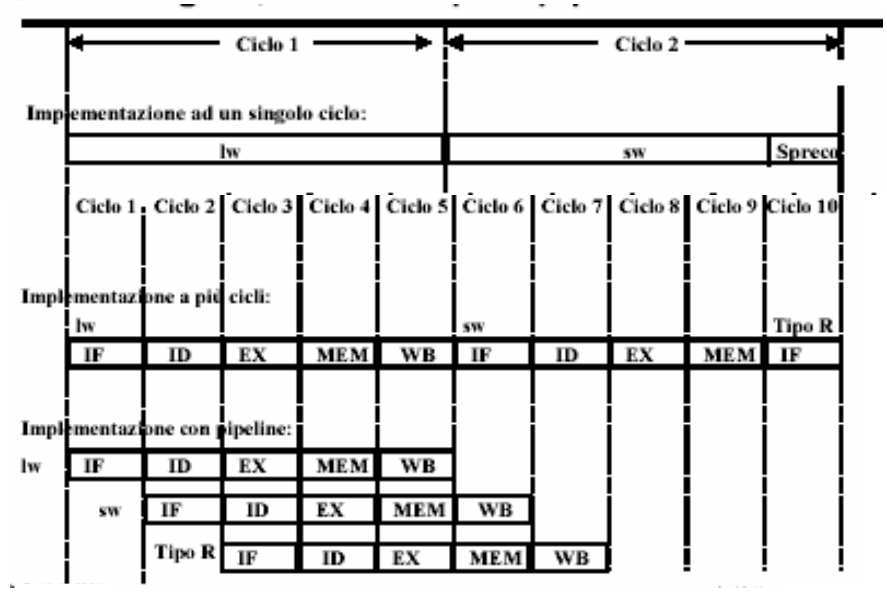
Rappresentazione grafica di una istruzione di add in pipeline

Esempio: add $s0, $t0, $t1



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

I vantaggi della pipeline



Esempio: pipeline per istruzioni di lw

Passo esecuzione	ALU	Memoria	Register File
IF (Fase fetch)	Yes	Yes	N0
ID (Decodifica)	Yes (salto)	N0	Yes
EX (Esecuzione)	Yes (calcolo indirizzo)	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)

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.

Sommario

La gestione delle eccezioni in una CPU multi-ciclo

Introduzione sulla pipeline

Gli stadi della pipeline

Rappresentazione del flusso di esecuzione in una pipeline