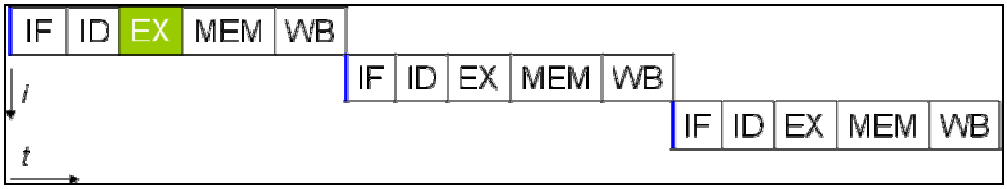
EVOLUZIONE DEI SISTEMI DI ELABORAZIONE

ELABORAZIONE DELLE ISTRUZIONI CON LA TECNICA PIPELINE

L'elaborazione di un'istruzione da parte di un processore si compone di cinque passaggi fondamentali:

1. **IF** (Instruction Fetch): Lettura dell'istruzione da memoria e incremento
2. **ID** (Instruction Decode): Decodifica istruzione e prelievo operandi
3. **EX** (Execution): Esecuzione dell'istruzione
4. **MEM** (Memory): Attivazione della memoria per lettura o scrittura dati (solo per certe istruzioni)
5. **WB** (Write Back): Scrittura del risultato nel registro opportuno

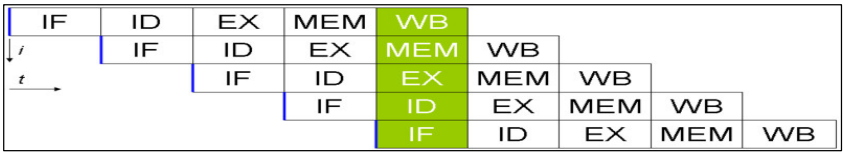
Praticamente ogni CPU in commercio è gestita da un clock centrale e ogni operazione elementare richiede almeno un ciclo di clock per poter essere eseguita.
Le prime CPU erano formate da un'unità polifunzionale che svolgeva in rigida sequenza tutti e cinque i passaggi legati all'elaborazione delle istruzioni. Una CPU classica richiede quindi almeno cinque cicli di clock per eseguire una singola istruzione.



Esecuzione delle istruzioni in un microprocessore senza pipeline

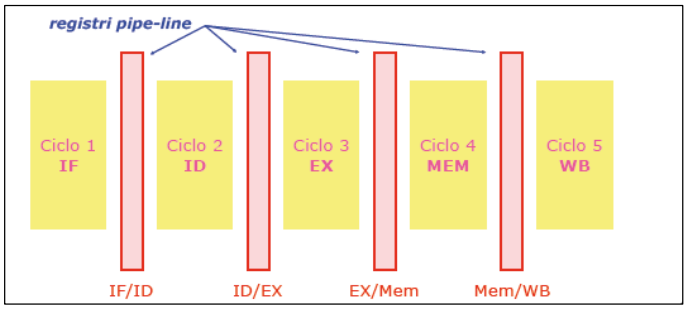
Con il progresso della tecnologia si è potuto integrare un numero maggiore di transistor in un microprocessore e quindi si sono potute **parallelizzare** alcune operazioni riducendo i tempi di esecuzione.

La **pipeline** è una tecnologia utilizzata nell'architettura hardware dai microprocessori dei computer per incrementare il throughput, ovvero la quantità di istruzioni eseguite in una data quantità di tempo, parallelizzando i flussi di elaborazione di più istruzioni.



Esecuzione delle istruzioni in un microprocessore con pipeline

Una CPU con pipeline è composta da cinque stadi specializzati, capaci di eseguire ciascuno un'operazione elementare di quelle sopra descritte. La CPU lavora come in una catena di montaggio cioè ad ogni stadio provvede a svolgere in maniera sequenziale un solo compito specifico per l'elaborazione di una certa istruzione.
Quando la catena è a regime, ad ogni ciclo di clock dall'ultimo stadio esce un'istruzione completata. Nello stesso istante ogni unità sta però elaborando in parallelo i diversi stadi di successive altre istruzioni.
In sostanza quindi si guadagna una maggior velocità di esecuzione a prezzo di una maggior complessità circuitale del microprocessore, che non è più composto da una sola unità, ma da cinque unità che devono collaborare tra loro.
Per riuscire a compiere una tale operazione sono necessari dei registri che mantengano l'informazione parzialmente elaborata in ogni fase.
Attualmente tutti i microprocessori utilizzano una struttura a pipeline per migliorare le loro prestazioni.



Problematiche

L'implementazione di una pipeline non sempre moltiplica il throughput finale. L'analisi delle problematiche legate alla gestione delle pipeline per ottenere le migliori prestazioni teoriche ricadono sotto la ricerca del **instruction level parallelism**, il parallelismo a livello d'istruzione, cioè le istruzioni che possono essere eseguite in parallelo senza creare conflitti o errori di esecuzione.
Comunque le singole pipeline affrontano due problemi principalmente:

a) **il problema legato alla presenza di istruzioni che possono richiedere l'utilizzo di risorse (dati o registri) non ancora disponibili**

b) **il problema legato alla presenza di salti condizionati.**

- **Il primo problema deriva dal lavoro parallelo delle unità**.
  Supponiamo che la CPU con pipeline debba eseguire il seguente frammento di codice:

  1. C=A+B
  2. D=C-1

  La prima istruzione deve prelevare i numeri contenuti nelle variabili A e B, sommarli e porli nella variabile C. La seconda istruzione deve prelevare il valore contenuto nella variabile C, sottrarlo di uno e salvare il risultato in D. Ma la seconda istruzione non potrà essere elaborata (EX) fino a quando il dato della prima operazione non sarà disponibile in memoria (MEM) e quindi la seconda operazione dovrà bloccarsi per attendere il completamento della prima e quindi questo ridurrà il throughput complessivo.
  Questo problema può essere attenuato facendo in modo che i dati elaborati dalla prima istruzione siano resi disponibili alla seconda istruzione prima del loro salvataggio definitivo in memoria.
  Questo si ottiene inserendo dei **registri temporanei** nell'unità di esecuzione dove salvare dati che serviranno all'istruzione successiva.

- **Il secondo problema consiste nei salti condizionati.**
  I programmi contengono delle istruzioni condizionate che se una specifica condizione logica è verificata provvedono a interrompere il flusso sequenziale del programma e a mandare in esecuzione un altro pezzo di programma indicato dall'istruzione di salto. Ogni volta che questo accade il microprocessore si trova a dover eseguire un nuovo flusso di operazioni e quindi deve svuotare la pipeline del precedente flusso e caricare il nuovo flusso. Ovviamente queste operazioni fanno sprecare cicli di clock e quindi peggiorano il throughput. Per ridurre questo problema le CPU adottano delle unità chiamate **unità di predizione delle diramazioni** (in inglese *Branch Prediction Unit*) che fanno delle previsioni sul flusso del programma.
  Queste unità riducono notevolmente i cicli persi per i salti.

Evoluzioni

La sempre maggior richiesta di potenza di calcolo ha spinto le industrie produttrici di microprocessori a integrare **in un unico chip più microprocessori** (**architetture a multiprocessore**). Questo strategia consente al computer di avere più CPU separate dal punto di vista logico, ma fisicamente risiedenti nello stesso chip.
Questa strategia progettuale attenua i problemi di coerenza e di predizione dei salti. Infatti ogni CPU logica esegue un programma separato e quindi tra i diversi programmi non si possono avere problemi di coerenza tra le istruzioni.
Questa scelta progettuale aumenta le prestazioni solo nel caso in cui il sistema operativo e i programmi applicativi siano **progettati secondo una logica parallela** anziché sequenziale, per poter utilizzare tutte le CPU disponibili.