Unità di apprendimento 1

Le architetture dei sistemi di elaborazione

Unità di apprendimento 1
Lezione 7

Le architetture non Von Neumann

In questa lezione impareremo:

- a conoscere le principali tecniche che migliorano le prestazioni dei computer
- a capire come si sono evolute le tecniche di elaborazione
- a comprendere le metodologie di gestione evoluta della memoria
- ad approfondire lo sviluppo nella gestione dei dispositivi di I/O

Le evoluzioni dei sistemi di elaborazione (1)

- Per migliorare le caratteristiche delle CPU e dei sistemi di elaborazione:
  - Aumento della **frequenza di clock**, a parità di comportamento rende più veloci le elaborazioni
  - Aumento dell'**ampiezza di parola** e aumento dello **spazio di indirizzamento**, incrementando il numero di bit elaborati contemporaneamente e il numero di celle indirizzabili

Le evoluzioni dei sistemi di elaborazione (2)

- Entrambe le tecniche hanno raggiunto presto un punto nel quale non era più possibile ottenere prestazioni migliori
- Nasce una nuova direzione di sviluppo, chiamata **non Von Neumann** con funzioni di elaborazione parallele che consentono di aumentare la velocità di elaborazione a parità di frequenza di clock

Grado di parallelismo delle architetture

- SISD (*Single Instruction Single Data*), un unico processore esegue un'unica istruzione per volta e può prelevare e depositare in memoria un solo dato per volta;
- SIMD (*Single Instruction Multiple Data*), permette di eseguire contemporaneamente la stessa operazione su più dati (utile per le operazioni multimediali);
- MISD (*Multiple Instruction Single Data*), consente di iniziare l'esecuzione di diverse istruzioni, ognuna sui propri dati. Mentre si esegue un'istruzione si può cominciare a elaborare la successiva;
- MIMD (*Multiple Instruction Multiple Data*), è relativa ai sistemi multiprocessori.

Evoluzioni dell'elaborazione: Esecuzione fuori ordine

- Capacità delle CPU di eseguire istruzioni senza rispettarne necessariamente l'**ordine** imposto dal codice che le contiene
- Per fare questo la CPU analizza le istruzioni che dovrà eseguire, individuando le istruzioni non vincolate dalle altre, quelle che non hanno vincolo sequenziale vengono fatte eseguire in **parallelo**
- Questo metodo tuttavia non sempre può essere impiegato

Evoluzioni dell'elaborazione: Prefetch

- I processori implementano delle unità che analizzano il codice cercando di **prevedere** in anticipo quali dati o istruzioni serviranno al processore, provvedendo inoltre al loro **caricamento in cache** (o direttamente nel processore) prima del loro reale impiego
- Alcune architetture prevedono istruzioni per indicare quali blocchi precaricare

Evoluzioni dell'elaborazione: Speculative execution

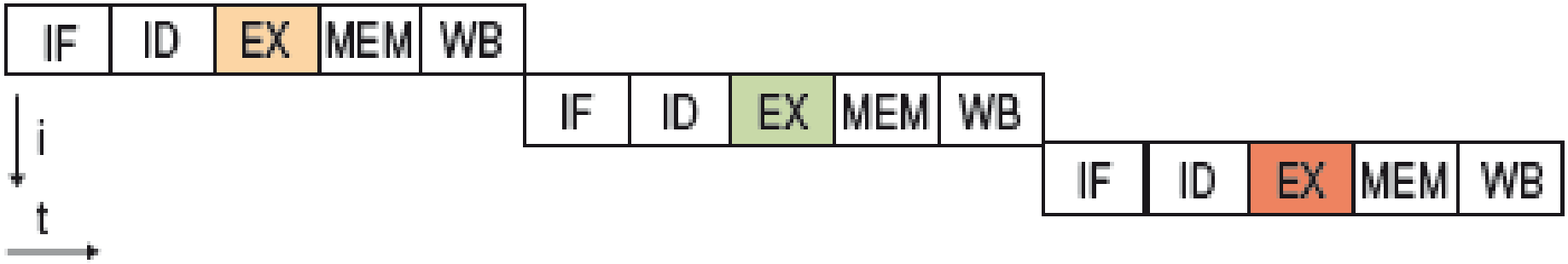
- Tecnica che consiste nell'eseguire entrambi i rami di un salto in modo da poter prevedere la direzione in una diramazione

La pipeline (1)

- La CPU elabora un'istruzione in cinque passaggi fondamentali:
  - **IF**: lettura dell'istruzione da memoria (**Instruction Fetch**)
  - **ID**: decodifica dell'istruzione e lettura degli operandi da registri (**Instruction Decode**)
  - **EX**: esecuzione dell'istruzione (**Execute**)
  - **MEM**: attivazione della memoria (**Memory**)
  - **WB**: scrittura del risultato nel registro opportuno (**Write Back**)

La pipeline (2)

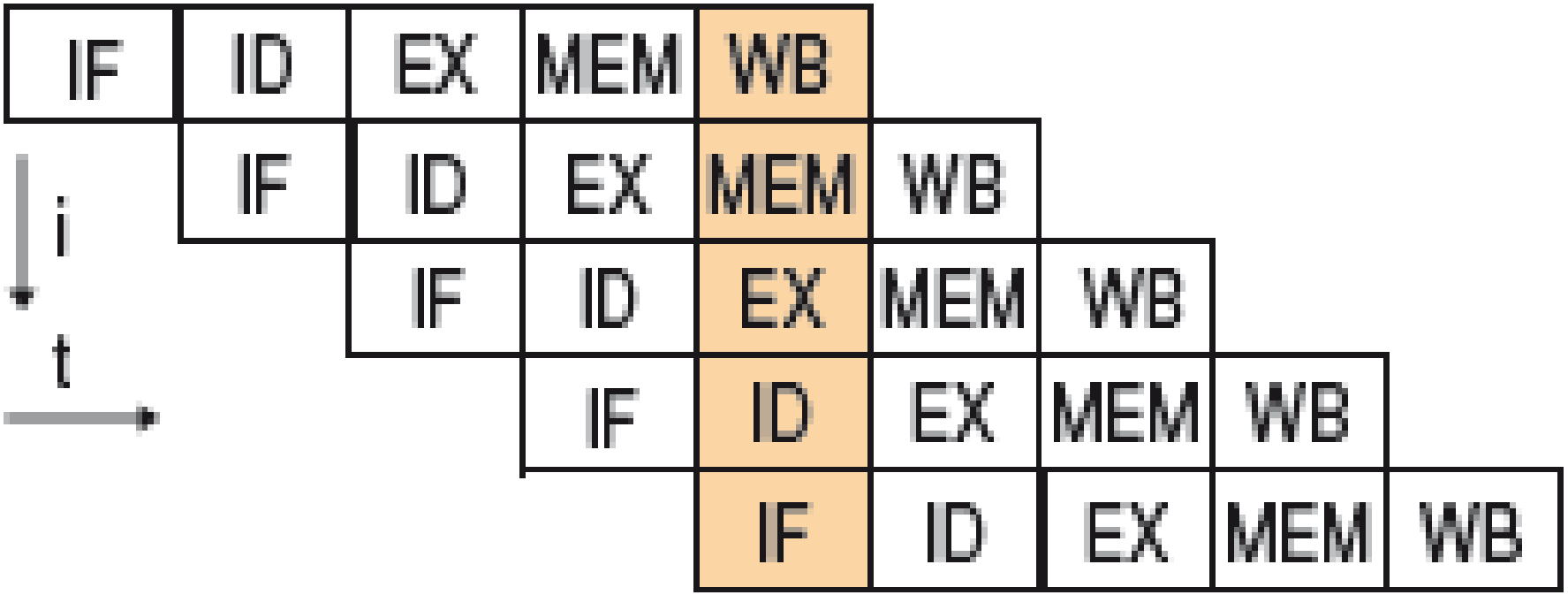
- Senza l'uso della pipeline avremmo la seguente sequenza di operazioni:



La pipeline (3)

- La CPU lavora in **pipeline** eseguendo a ogni stadio un solo compito specifico, a ogni **ciclo di clock** viene completata l'esecuzione dei **cinque stadi** per un'istruzione, nello stesso istante ogni unità elabora in parallelo i diversi stadi delle istruzioni

La pipeline (4)



Problematiche della pipeline

- **conflitti strutturali**: quando due fasi utilizzano contemporaneamente la stessa risorsa, come per esempio i bus, l'ALU o la memoria RAM
- **conflitti tra i dati**: quando un'istruzione utilizza un dato che è ancora in fase di elaborazione
- **conflitti di controllo**: quando le istruzioni che devono essere eseguite da un salto si trovano in una zona di memoria non prevista. In questo caso la coda delle istruzioni deve essere caricata nuovamente con conseguente perdita di tempo

Soluzione alla criticità della pipeline: lo stallo

- Consiste nell'inserimento di alcuni cicli per bloccare temporaneamente una parte della pipeline in attesa che si risolva il conflitto
- Se per esempio nella fase di **Execute** la seconda istruzione richiede il contenuto di un registro che dovrà essere aggiornato dalla fase successiva (**Write Back**), viene aggiunto un ciclo di stallo dopo la fase di **Execute** della seconda istruzione

I istruzione	Fetch	Decode	Execute	Write Back		
II istruzione	---	Fetch	Decode	Execute	Stallo	Write Back

Conflitti strutturali della pipeline

- soluzione **hardware**: aggiungendo all'**ALU** l'unità **FPU** (**Floating Point Unit**), consentiamo al sistema l'uso di questa unità in alternativa all'**ALU**, in modo tale che due stadi della pipeline utilizzino contemporaneamente le due unità (**ALU** e **FPU**), la prima per calcoli su numeri interi e la seconda per calcoli con numeri espressi in virgola mobile
- L'accesso contemporaneo alla memoria **RAM** è risolto da parte di due istruzioni successive dividendo la memoria cache di primo livello in due parti: memoria dati e memoria istruzioni, secondo l'architettura **Harvard**

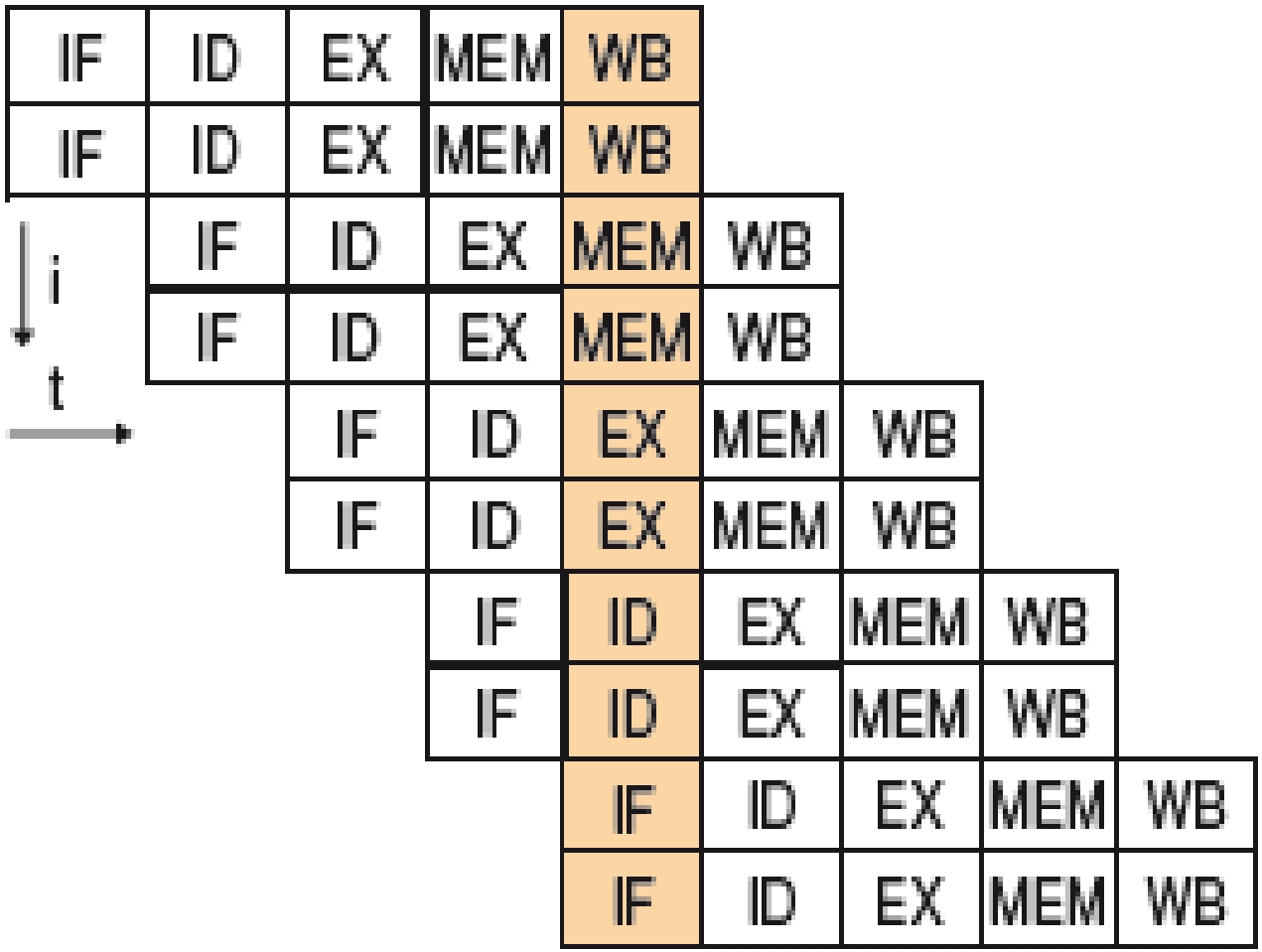
Conflitto tra dati nella pipeline

- Il conflitto tra i dati può essere semplicemente spostato a un livello **software**: il compilatore può individuare le istruzioni che generano il conflitto sui dati interponendo tra esse una serie di istruzioni assembly inutili chiamate **NOP** (**No Operation**) con il compito di ritardare l'esecuzione della seconda istruzione per consentire l'adeguamento delle pipeline

Evoluzioni dell'elaborazione: Tecnologie superscalari

- Per realizzare CPU con prestazioni sempre migliori si è affermata la tecnologia **superscalare** grazie alla quale vengono integrati in un unico microprocessore **più pipeline** che funzionano in **parallelo**
- Le pipeline sono composte da più di 5 stadi ed è stata innalzata la **frequenza** di **clock**, questo è possibile spezzando le singole operazioni elementari necessarie per completare un'istruzione
- Aumenta tuttavia la criticità dei salti condizionati

CPU superscalare a doppia pipeline



Evoluzioni dell'elaborazione: Branch prediction

- I **salti condizionati** rendono critica la pipeline, per la ripetizione dei cicli macchina
- Un processore incontra mediamente un salto condizionato ogni 6/7 istruzioni, è importante cercare di **prevederne** (**branch prediction**) la destinazione in anticipo per caricarne il blocco di istruzioni corretto nella pipeline

Evoluzioni della memoria centrale: Cache memory

- Memoria di tipo statico (**SRAM**) molto veloce e di dimensioni contenute, inserita tra la memoria centrale di tipo **SDRAM** e la **CPU**.
- **Rapido** accesso alla memoria
- Tuttavia, essendo di dimensioni ridotte rispetto alla **RAM**, la memoria cache è in grado di memorizzare soltanto una minima parte dei dati contenuti nella **RAM**

Cache memory: principi di località

- **Località spaziale**: i programmi in esecuzione possono accedere a celle di memoria presenti all'interno di una zona definita, oppure possono accedere alle stesse locazioni di memoria già utilizzate
- **Località temporale**: il programma eseguirà molto probabilmente le istruzioni accedendo alle istruzioni immediatamente successive a quella appena eseguita

Livelli di Cache

- **cache integrata** direttamente nel processore, chiamata cache di primo livello (**L1**)
- **cache esterna** collegata direttamente al processore, chiamata cache di secondo livello (**L2**), è collegata alla **CPU** tramite il **BSB** (**Back Side Bus**)
- **cache esterna** collegata sulla **motherboard**, denominata cache di terzo livello (**L3**)

Cache memory: organizzazione

- È organizzata in blocchi di celle chiamate **linee,** ad ogni accesso in memoria centrale le informazioni vengono copiate in una linea della cache
- Durante ogni operazione di lettura da parte della CPU avviene una ricerca delle informazioni nella cache: se presenti (operazione di **Cache Hit**) non serve accedere alla memoria centrale, altrimenti (operazione di **Cache Miss**) viene effettuata la lettura in memoria e le informazioni prelevate vengono memorizzate anche nella cache

Cache memory: operazioni di scrittura in memoria

- cache **Write Through**: le operazioni di scrittura vengono effettuate sia nella cache che nella memoria centrale
- cache **Write Back**: le operazioni di scrittura avvengono solo nella cache, mentre la linea di cache viene scritta in memoria solo quando viene sostituita

Cache memory: gestione

- La cache viene gestita attraverso tre possibili metodi:
- metodo **diretto** (o mappatura diretta)
- metodo **completamente associativo**
- metodo **associativo a N vie**

Memoria virtuale

- La **memoria virtuale** rappresenta un metodo per far apparire la memoria centrale di dimensione maggiore rispetto a quella effettivamente allocata
- Se ne occupa la **MMU** (**Memory Management Unit**)

MMU (1)

- Si tratta di un processore in grado di traslare o **tradurre** gli indirizzi di **memoria virtuali** in indirizzi di memoria fisici, di effettuare la **protezione** della memoria, il controllo della **cache** della **CPU**, l'**arbitraggio** del BUS e, in architetture più semplici, la **commutazione** di banchi di memoria

Mmu (2)

- Suddividono lo spazio degli **indirizzi virtuali** (l'intervallo di indirizzi accessibili dal processore) in pagine di memoria di dimensione di pochi kB
- Gli **N** bit meno significativi dell'indirizzo rappresentano l'**offset** interno della pagina e i bit restanti rappresentano il numero virtuale della pagina
- La **MMU** contiene una tabella delle pagine indicizzata dal numero della pagina, in cui ogni elemento si chiama PTE (**Page Table Entry**) e restituisce il numero fisico della pagina corrispondente a quello virtuale che, combinato con l'offset della pagina, forma l'idirizzo fisico completo

Le evoluzioni degli I/O: DMA (Direct Memory Access)

- Consente l'impostazione dei trasferimenti tramite un insieme di porte di configurazione visibili dalla CPU
- Il programma imposta l'indirizzo iniziale del blocco di memoria da trasferire e le dimensioni. Una volta iniziato il trasferimento il controller **DMA** provvederà autonomamente al trasferimento dei dati da o per la memoria
- Il controller **DMA** condivide i **BUS** con la **CPU**, quindi il **DMA** trasferisce quando la **CPU** non usa i **BUS** (elaborazioni interne)

I coprocessori

- Sono CPU secondarie dedicate a scopi specifici che scaricano la CPU principale da alcune elaborazioni consentendo un maggior parallelismo:
  - **coprocessore numerico**: realizza una grande quantità di elaborazioni numeriche soprattutto in virgola mobile non realizzate dalla CPU principale. Può anche essere integrato nello stesso chip della CPU principale
  - **coprocessore grafico**: mantiene una memoria immagine della presentazione video e realizza elaborazioni grafiche per la costruzione della presentazione