Lezione 25

CPU pipeline – 4: le CPU moderne

Proff. A. Borghese, F. Pedersini

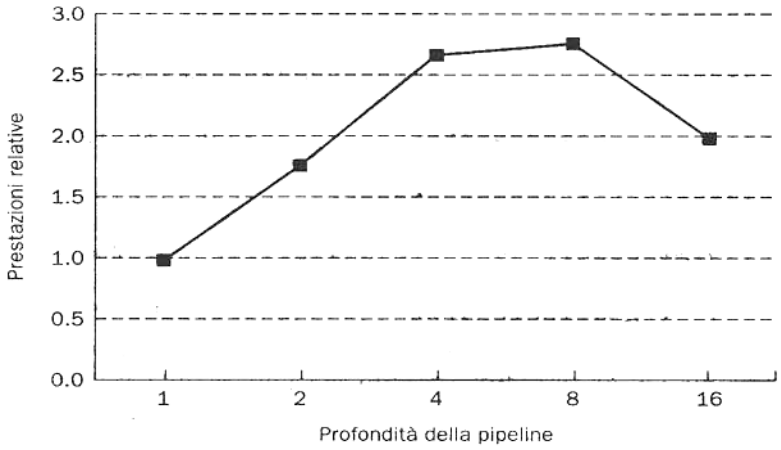
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Pipeline più spinte

- **Superpipelining**: pipelines più lunghe
- Architettura **Superscalare** (statica/dinamica): più di una istruzione viene iniziata nello stesso ciclo di pipeline
  - Scheduling statico: parallelismo definito a priori dal programmatore o dal compilatore (IA-64)
  - Scheduling dinamico: parallelismo deciso automaticamente dalla CPU. L'hardware cerca di individuare le istruzioni pronte per essere eseguite (MIPS, Pentium 4)
    - è possibile l'esecuzione in ordine diverso da quello dato.
    - è possibile avere un'unica istruzione per più dati: architettura ***SIMD:*** *Single Instruction – Multiple Data*

      Esempio: Intel Pentium IV: tempi medi di esecuzione della ALU sono metà del periodo di clock – fino a 126 istruzioni contemporanee

Superpipeline



- **Pipelines più lunghe:**
  teoricamente guadagno in velocità proporzionale al **n. di stadi**
  Es: Digital DEC Alpha: 5-7 stadi, Intel/AMD: oltre 20 stadi
- **Problemi:**
  - Criticità sui dati: stalli più frequenti.
  - Criticità sul controllo: numero maggiore di stadi di cui annullare l'esecuzione.
  - Maggior numero di registri → maggiori dimensioni chip → il clock non si riduce con il numero degli stadi

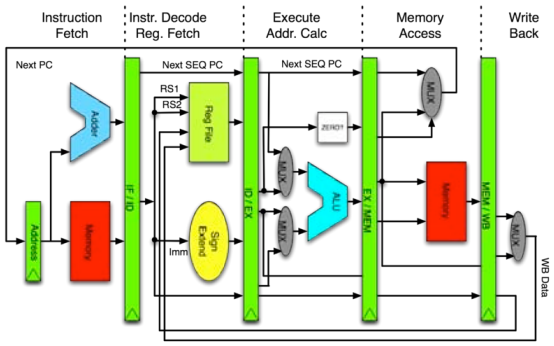
Schedulazione statica: IA-64

- **Scheduling statico**: parallelismo definito a priori dall'utente (programmatore / compilatore)
- **IA-64**: Instruction-level parallelism (ILP) implementato esplicitamente
  - **EPIC**: Explicitly Parallel Instruction Computer
- Le istruzioni sono raggruppate in bundles eseguiti in parallelo
  - 128 registri accedibili a finestre (in parallelo)
  - Istruzione: 128 bit = 5 bit: template field +
    3 x 41 bit: 3 istruzioni in parallelo
- **Predication**: trasformazione di branch in predicati:

```
If (p) then (statement 1) else (statement 2)
                ▼
   (p)   ➔ statement 1     # statements 1 e 2 possono
   (~p)  ➔ statement 2     # essere parallelizzati!
```

Pipeline Superscalari: **MIPS**

MIPS **R2000**: pipeline classica

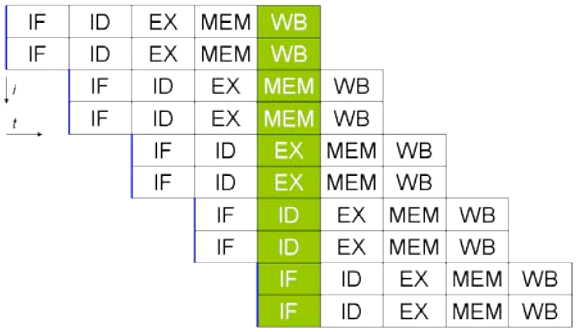


R3000:

- Register File a doppia banda (4 porte di lettura, 2 porte di scrittura)
- 2 ALU: general purpose, base+offset

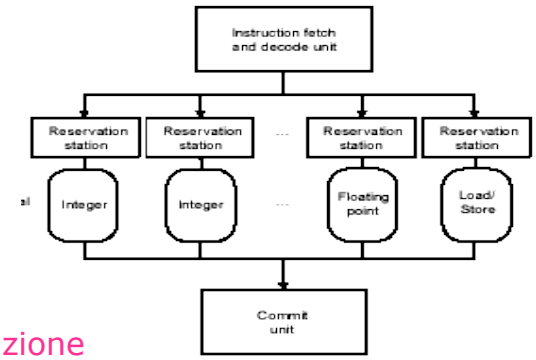
R8000: pipeline superscalare

- Replicate le unità funzionali per eseguire più istruzioni in parallelo
- Ottengo più di 1 istruzione per ciclo di clock: **N > 1 instr. / CC**



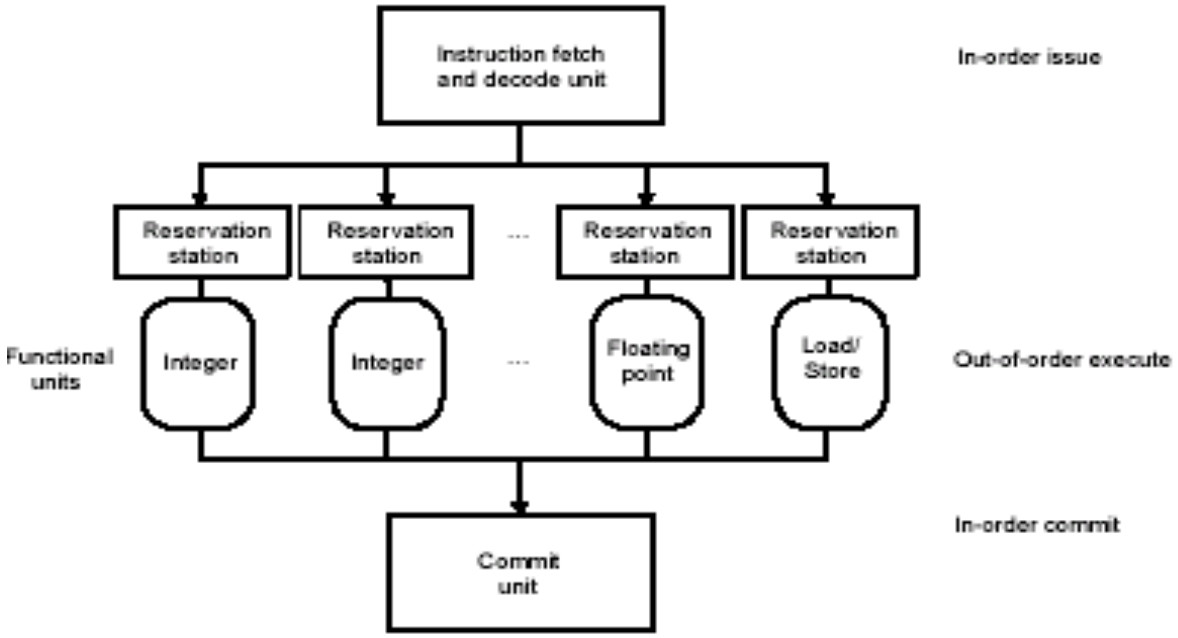
Schedulazione dinamica:

Schedulazione Dinamica:



- **Fetch**: Prelievo ed invio alla coda istruzioni
- **Decodifica**: Conversione in microistruzioni
  - Valutazione criticità (branch)
- **Esecuzione**:
  Le istruzioni vengono alla stazione di prenotazione corrispondente all'unità funzionale richiesta (smistamento)
  - Un riferimento all'istruzione viene inserito nel buffer di riordino.
  - Esecuzione vera e propria (calcolo)
  - Eventuali criticità portano a far attendere alcune istruzioni
- **Commit Unit:**
  Riordino istruzioni in modo che terminino in modo sequenziale
  - Può fornire in uscita più itruzioni in un ciclo di clock.
  - Commit Unit tiene traccia di tutte le istruzioni pendenti.

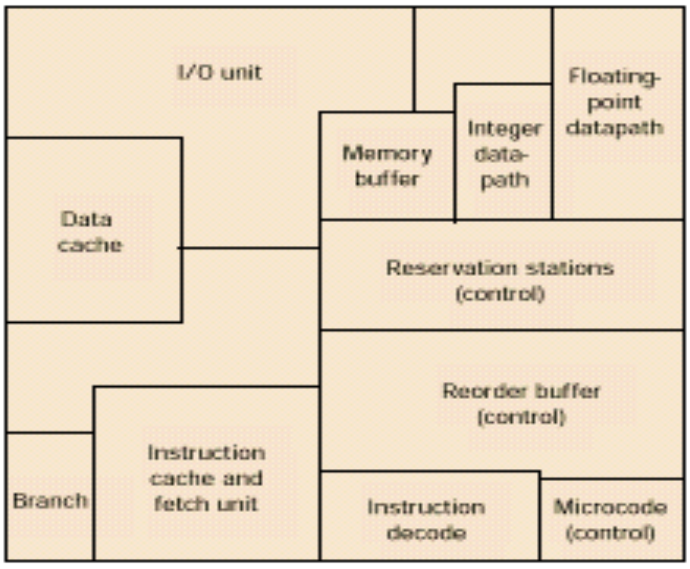
Pipeline con schedulazione dinamica



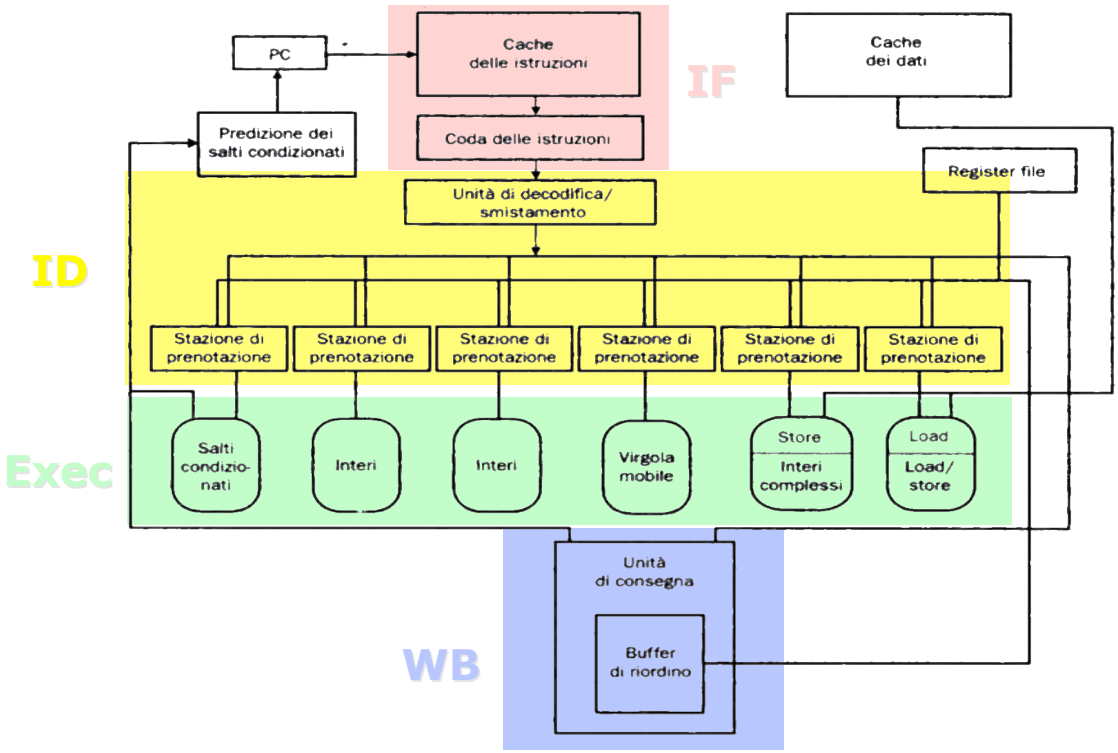
- Esistono cammini paralleli per le diverse fasi dell'istruzione (Exec/Mem)
- Durante lo stallo eseguo istruzioni successive → **supero gli stalli**

Pipeline: processori INTEL

- Esecuzione "speculativa" (Intel 80x86 dal Pentium)
  - scheduling dinamico
  - predizione dei salti
- Dal Pentium 4:
  - architettura super-scalare
  - "super-pipeline"



Scheduling dinamico: **PentiumPro**



Pipeline PentiumPro: funzionamento

- **Fetch**: Prelievo di 16 byte dalla MI ed invio alla coda istruzioni
- **Decodifica**:
  Conversione in microistruzioni di lunghezza fisssa: **72 bit**
  - In questa fase vengono valutate le branch (su un certo numero di istruzioni) attraverso una **Branch Prediction Table** di 512 elementi
  - Recupero operandi ed assegnazione registri replicati (rename buffer)
- **Esecuzione**: Le istruzioni vengono alla stazione di prenotazione corrispondente all'unità funzionale richiesta (smistamento)
  - Un riferimento all'istruzione viene inserito nel buffer di riordino.
  - Esecuzione vera e propria (calcolo)
- **Riconsegna:**
  Riordino istruzioni in modo che terminino in modo sequenziale
  - L'unità di consegna tiene traccia di tutte le istruzioni pendenti all'interno del buffer di riordino.
  - Può fornire in uscita più istruzioni in un ciclo di clock.

Esecuzione dell'istruzione

- Per considerare un'istruzione serve spazio nel Buffer di riordino e nella stazione di prenotazione della pipeline richiesta
- La CPU contiene registri duplicati
  - con più istruzioni in esecuzione, il registro destinazione potrebbe essere occupato → scrivo su un registro ausiliario: **rename registers/buffers**
  - In attesa che l'unità di consegna dia il permesso di scrivere i registri effettivi
- L'istruzione viene eseguita quando:
  - nella stazione di prenotazione ci sono **tutti gli operandi**,
  - l'unità funzionale è **libera**.
- L'unità di consegna decide quando l'istruzione è terminata
  - Se la predizione di un **salto è scorretta**, vengono scartate le istruzioni sbagliate dalle stazioni di prenotazione e dai buffer di riordino e liberati i registri interni

Pipeline del **Pentium 4**

- **Fetch** delle istruzioni della Memoria (Cache)
- **Decodifica:** Ciascuna istruzione viene tradotta in una o più microistruzioni di lunghezza fissa (RISC)
- Il processore esegue le micro-istruzioni in una pipeline superscalare a schedulazione dinamica
- Il processore restituisce ai registri il risultato dell'esecuzione delle micro-istruzioni relative alla stessa istruzione
  - rispettando l'ordine di esecuzione delle microistruzioni

Il Pentium 4 trasforma un **ISA CISC** in un **ISA interno RISC** costituito dalle micro-operazioni

LA CPU del Pentium 4

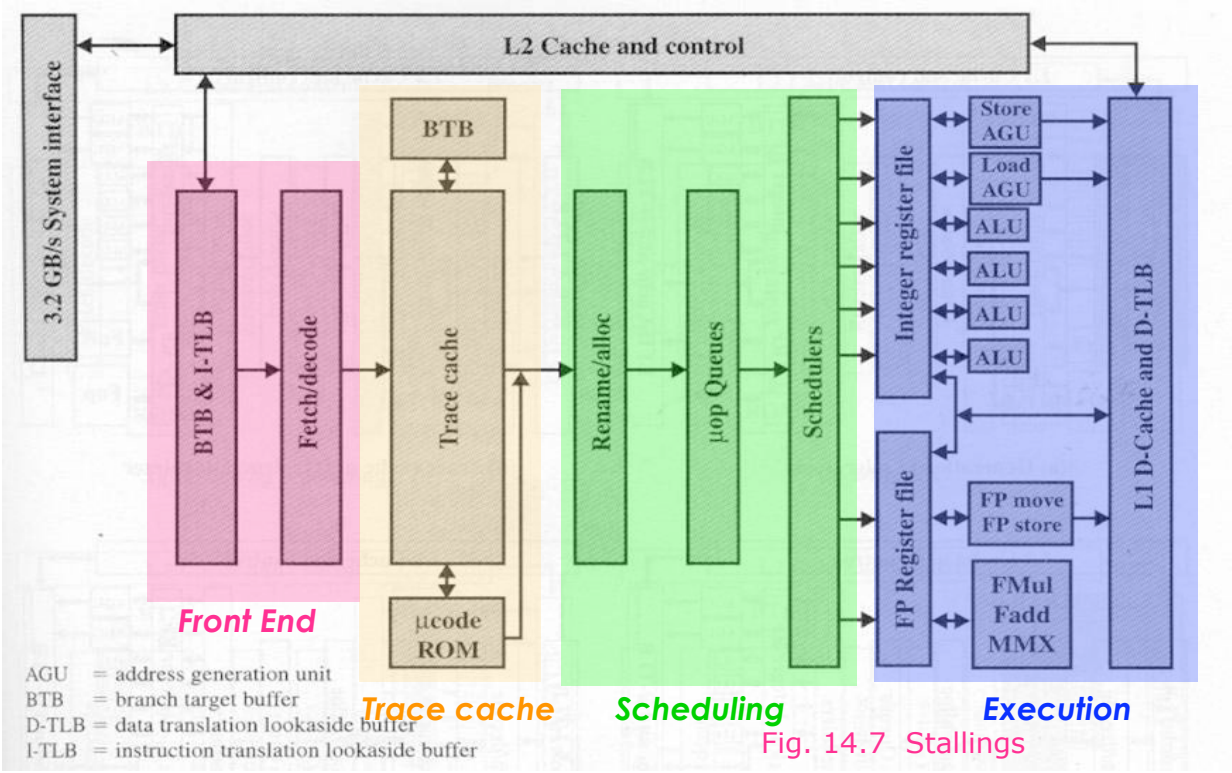


Fig. 14.7 Stallings

Pipeline del Pentium 4: **Front end**

- Le **istruzioni** passano dalla cache primaria al buffer **L2** (64 byte).
  - Qui vengono gestiti casi di "missing" ed i trasferimenti da cache
- In parallelo i **dati** vengono caricati da cache al buffer dati **L1**
  - Nel trasferimento da L1 ed L2 agiscono Branch Target Buffer e look-aside buffer che gestiscono le predizioni sulle branch
- Inizia la fase **Fetch** e **Decodifica** che legge il numero di byte pari alla lunghezza dell'istruzione
- Il **decoder** traduce l'istruzione in micro-operazioni (da 1 a 4):
  - **Microistruzione: 118 bit, appartenenti ad un'ISA RISC.**

Pipeline del Pentium 4: **Trace cache**

Trace cache: riorganizzazione delle istruzioni in decodifica, per ottimizzare il flusso di esecuzione

- Funzioni svolte:
  - **Riordino locale del codice,** in modo da minimizzare le criticità
    - la pipeline ha 20 stadi.
  - Invio alla fase di esecuzione (Allocate e Renaming).
  - **Istruzioni complesse** (> 4 microistruzioni) vengono create in questa fase con una macchina a stati finiti
    - Implementata con una ROM + memoria

Pipeline del Pentium 4: **Esecuzione**

- Elemento centrale è il **Reorder Buffer** o **Scheduler**
  - È un **buffer circolare** – può contenere fino a **126 micro-istruzioni**
  - Contiene le micro-istruzioni con il loro stato, la presenza o assenza dei dati, per tutta la durata dell'esecuzione.
  - Alloca i registri (traducendo i registri simbolici dell'Assembly in uno dei 128 registri di pipeline).
  - Avvia alla coda di esecuzione l'istruzione (coda per le **`lw/sw`** e coda per le altre istruzioni).
  - Lo scheduling avviene prendendo la prima istruzione che può essere eseguita nella coda. Dagli scheduler si può passare alle ALU in modi diversi.
- La fase di esecuzione scrive poi i risultati nei registri o nella cache L1
- C'è una parte dedicata ai flag che vengono poi inviati alla **BTB** per predirre correttamente i salti.