Architettura dei Calcolatori e Sistemi Operativi
Memorie Cache

Chair

Prof. C. Brandolese

e-mail: carlo.brandolese@polimi.it
phone: +39 02 2399 3492
web: home.dei.polimi.it/brandole

Politecnico di Milano

Teaching Assistant

A. Canidio

e-mail: andrea.canidio@mail.polimi.it
material: github.com/acanidio/polimi_cr_acso_2018

Outline

- **Memorie Cache**
  - Struttura
  - Problemi
  - Tipologie di memorie cache
    - Indirizzamento diretto
    - Completamente associativa
    - Set associativa a N vie

Memorie Cache

- La **memoria cache** è una memoria **veloce** relativamente **piccola**, non visibile al software che memorizza i dati più recentemente usati della **memoria principale** del sistema.

- Il funzionamento della Cache Memory si basa principalmente su due *principi di località*:
  - **Località temporale**
    - Dati recentemente usati hanno un'alta probabilità di essere nuovamente usati a breve.
  - **Località spaziale**
    - Se un dato viene referenziato, è molto probabile che dati adiacenti siano a breve a loro volta acceduti.

- Accesso al dato in Cache:
  - **Cache Hit** Dato presente in cache
  - **Cache Miss** Dato non presente in cache

Struttura di Cache

- Ogni linea di cache contiene, in aggiunta al blocco di dati:
  - **Tag** - identica univocamente un blocco in cache
  - **D (dirty bit)** - quando e settato ad 1, indica che il blocco in cache e stato modicato, e viceversa (facoltativo)
  - **V (validity bit)** - quando e settato ad 1, indica che il blocco in cache e valido, e viceversa

Problemi Essenziali

- **Dove (block placement)**
  - Dove caricare un blocco proveniente da un livello gerarchico inferiore?
- **Come (block identication)**
  - Come individuare un blocco in un livello gerarchico superiore?
- **Quale (block replacement)**
  - Quale blocco sostituire in caso di miss per fare posto ad un blocco del livello gerarchico sottostante?
    - FIFO
    - LRU
    - RANDOM
- **Politica di Scrittura (write policy)**
  - Come gestire le modiche dei blocchi?
    - Write back
    - Write through

Cache a Indirizzamento Diretto

- Ciascun blocco di RAM va mappato su un preciso blocco di cache
- Struttura dell'indirizzo di RAM:

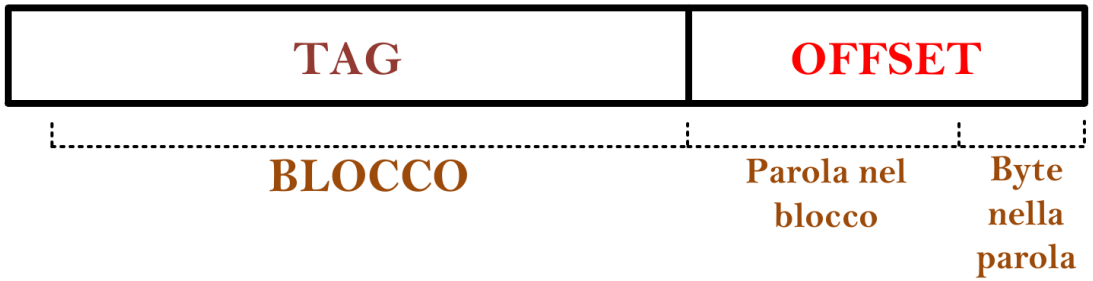
- **DOVE (block placement)**
  - (indirizzo blocco) MOD (numero blocchi in cache)
  - Equivale a considerare il campo INDICE dell'indirizzo in RAM

- **COME (block identication)**
  - Dato l'indirizzo in RAM, si considera il campo INDICE, che indichera il blocco di cache in cui cercare il dato. Se in quel blocco il tag e uguale al tag dell'indirizzo in RAM e il validity bit e settato ad 1, allora il dato e presente in cache ed e valido.

[Oss: il dato va cercato in una sola linea di cache!]

Cache Completamente Associativa

- Ciascun blocco di RAM puo essere mappato in qualsiasi blocco di cache
- Struttura dell'indirizzo di RAM:



- **DOVE (block placement)**
  - In qualsiasi blocco di cache

- **COME (block identication)**
  - Dato l'indirizzo in RAM, si considera il campo TAG, che indichera il blocco di RAM da cercare in cache. Tale campo deve essere confrontato con l'omonimo campo in tutte le linee di cache. Se si trova un matching sul campo TAG e il validity bit e settato ad 1, allora il dato e presente in cache ed e valido.

[Oss: il dato va cercato in tutte le linee di cache!]

Cache Set Associativa

- Ciascun blocco di RAM va mappato in uno qualsiasi dei blocchi di un preciso set nella cache
- Struttura dell'indirizzo di RAM:

- **DOVE (block placement)**
  - (indirizzo blocco) MOD (numero set in cache)
  - Equivale a considerare il campo SET dell'indirizzo in RAM

- **COME (block identication)**
  - Dato l'indirizzo in RAM, si considera il campo SET, che indichera il set nella cache in cui cercare il dato. All'interno del set individuato, bisogna effettuare una ricerca associativa sul campo TAG di tutte le linee di cache in quel set. Se si trova un matching sul campo TAG e il validity bit e settato ad 1, allora il dato e presente in cache ed è valido. [Oss: il dato va cercato in tutti i blocchi di un solo set!]