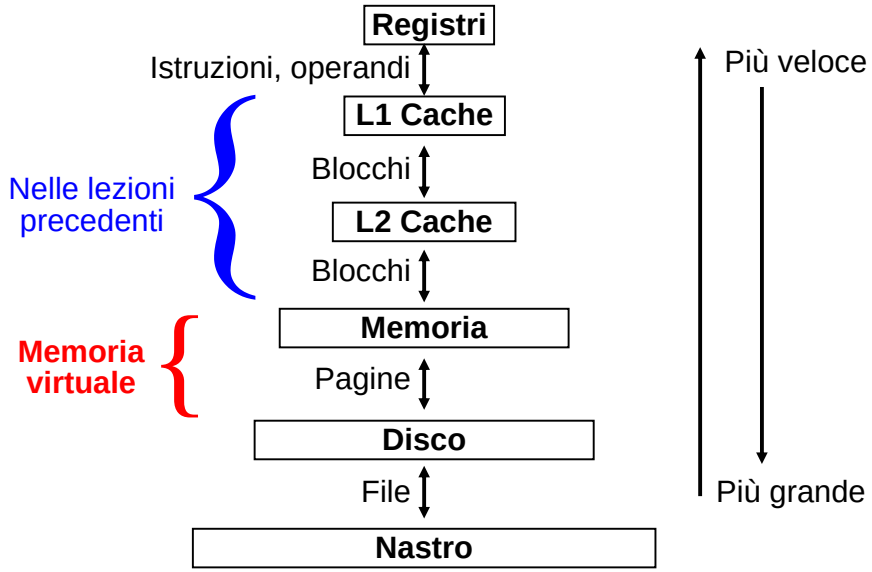
La memoria virtuale

Architetture Avanzate dei Calcolatori

Valeria Cardellini

La gerarchia di memorie



Memoria virtuale

- Memoria virtuale
  - Uso della memoria principale come una cache della memoria secondaria (disco)
- Permette di separare i concetti di
  - *Spazio di indirizzamento* di un processo
    - Condivisione della memoria tra più processi, garantendo la *protezione* (un processo non scrive/legge la porzione di memoria di un altro processo)
  - *Dimensione* effettiva della memoria fisica
    - Si ha l'illusione di avere a disposizione più memoria fisica
      - Solo le parti attive dei programmi sono in memoria
      - E' possibile eseguire più programmi, con codici e dati di dimensioni maggiori della memoria fisica

Indirizzo fisico

- **Indirizzo fisico** (o *reale*)
  - E' l'indirizzo della memoria fisica
- Spazio di indirizzamento fisico (o reale)
  - E' il numero di byte (o parole) indirizzabili fisicamente
  - Dipende esclusivamente dal numero di bit dell'indirizzo fisico
- Dimensione della memoria fisica
  - Numero di byte (o di parole) che la costituiscono

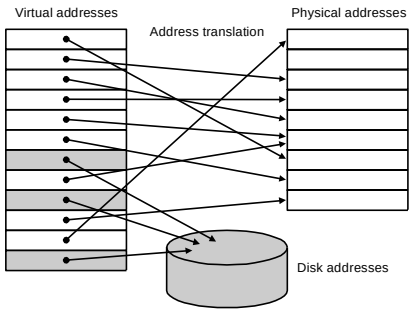
Indirizzo virtuale

- **Indirizzo virtuale** (o *logico*)
  - E' l'indirizzo usato dal programma in esecuzione (nelle istruzioni in linguaggio macchina)
  - Assegnato dal linker (è quindi un *indirizzo rilocabile*)
- Spazio di indirizzamento virtuale (o logico)
  - Dipende dal numero di bit dell'indirizzo virtuale
  - Tipicamente molto maggiore dello spazio di indirizzamento fisico
- Dimensione della memoria virtuale di un programma
  - *Iniziale*: calcolata dal linker dopo la generazione del codice eseguibile
  - *Durante l'esecuzione*: può variare per la modifica dello *stack* (chiamate a procedure) e per la modifica dello *heap* (gestione delle strutture dati dinamiche)

Traduzione da indirizzo virtuale a indirizzo fisico

- Il processore fa riferimento ad un indirizzo virtuale, che viene tradotto in un indirizzo fisico, utilizzato per accedere alla memoria principale
- Traduzione (*mapping*) degli indirizzi virtuali in indirizzi fisici
  - Coinvolge elementi hw e sw
  - Viene attivata durante l'esecuzione del programma ad ogni riferimento di memoria
  - E' trasparente al programmatore, al compilatore, al linker
  - Il meccanismo di traduzione degli indirizzi deve essere efficiente!

Traduzione da indirizzo virtuale a indirizzo fisico (2)



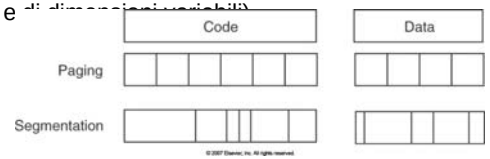
- Nella memoria virtuale, blocchi di memoria vengono mappati dallo spazio di indirizzamento virtuale allo spazio di indirizzamento fisico

Paginazione e segmentazione

- Sono due meccanismi per la gestione della memoria virtuale (trattati nel modulo di Sistemi Operativi)
- Il sistema operativo deve individuare dei blocchi di memoria fisica liberi (non necessariamente contigui)
- Paginazione
  - I blocchi di memoria virtuale hanno una dimensione *fissa* (detti *pagine*), tipicamente 4 KB o 8 KB
  - Vantaggi:
    - Riduce la frammentazione della memoria (presenza di zone di memoria piccole e libere inframmezzate a zone utilizzate): solo frammentazione interna (porzione di pagina non utilizzata)
    - Rimpiazzamento banale (tutte le pagine hanno la stessa dimensione)
    - Facilita la gestione della crescita di memoria di un processo durante l'esecuzione
  - Svantaggi:
    - Limitazione nella protezione

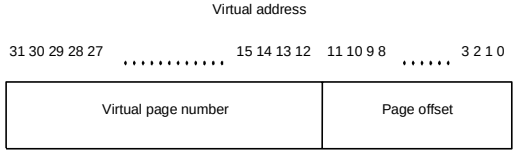
Paginazione e segmentazione (2)

- Segmentazione
  - I blocchi di memoria virtuale hanno una dimensione *variabile* (detti *segmenti*), da 1 byte come dimensione minima a 2^{16}-2^{32} byte come dimensione massima
  - Vantaggi:
    - Si presta alla protezione ed alla condivisione tra processi
    - Semplifica la gestione di strutture dati di dimensione variabile
    - Permette di fornire spazi di indirizzamento multipli
  - Svantaggi:
    - Frammentazione esterna (porzione di pagina non utilizzata)
    - Rimpiazzamento difficile (occorre trovare un'area di memoria contigua e di dimensioni variabili)



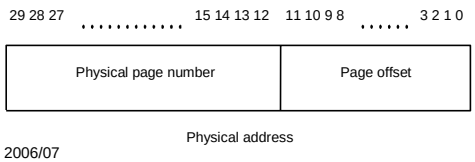
Paginazione

- Spazio di indirizzamento virtuale di un programma: *lineare* e suddiviso in un numero intero di *pagine* di dimensione *fissa* pari a una potenza di 2
  - Dimensioni tipiche di pagina: da 512 byte a 64 KB
- L'indirizzo virtuale è formato da due campi:
  - Il **numero di pagina virtuale** (NPV)
  - Lo spiazzamento (offset) del byte dall'inizio della pagina
- Esempio di indirizzo logico:



Paginazione (2)

- Spazio di indirizzamento fisico: suddiviso in un numero intero di *pagine* di dimensione pari a quelle utilizzate per lo spazio di indirizzamento virtuale
- Ogni pagina della memoria (*page frame*) può contenere una pagina dello spazio di indirizzamento virtuale
- L'indirizzo fisico è formato da due campi:
  - Il **numero di pagina fisica** (NPF)
  - Lo spiazzamento (offset) del byte dall'inizio della pagina
- Esempio di indirizzo fisico:



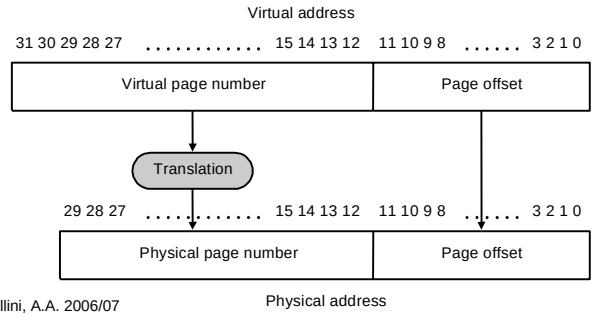
Traduzione degli indirizzi

- NPV tradotto in NPF
- L'offset non viene modificato:
  offset della pagina fisica = offset della pagina virtuale

- Spazio di indirizzamento virtuale (in byte):
  # pagine virtuali · dimensione pagina virtuale =
  = $2^{|NPV|} \cdot 2^{|OFFSET|}$ byte
- Spazio di indirizzamento fisico (in byte):
  # pagine fisiche · dimensione pagina fisica =
  = $2^{|NPF|} \cdot 2^{|OFFSET|}$ byte

Esempio di traduzione degli indirizzi

- Indirizzo virtuale (32 bit)
  - 20 bit per NPV e 12 bit per offset
    - Pagina di dimensione pari a 4 KB (2^{12} B)
    - Spazio di indirizzamento virtuale di 4 GB ($2^{20} \cdot 4$ KB)
- Indirizzo fisico (30 bit)
  - 18 bit per NPF e 12 bit per offset
    - Spazio di indirizzamento fisico di 1 GB ($2^{18} \cdot 4$ KB)



Da indirizzo simbolico a indirizzo fisico

- Prima dell'esecuzione ogni programma attraversa quattro stadi:
  - Codice sorgente, codice oggetto, codice eseguibile, codice in esecuzione
  - In ciascuno di questi stadi il riferimento agli indirizzi è fatto in modo diverso

- Memory Management Unit (MMU): unità di gestione della memoria, responsabile della traduzione da NPV a NPF
- La MMU è incorporata nel processore

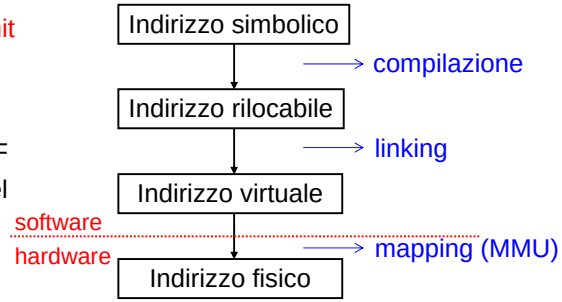
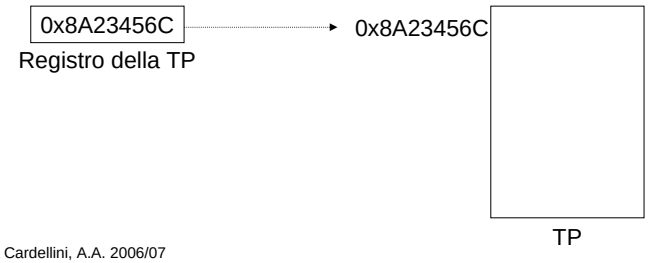


Tabella delle pagine

- La **tabella delle pagine** (TP o page table) contiene la corrispondenza tra pagine virtuali e pagine fisiche per un processo
  - La TP è indicizzata tramite il NPV
  - Ogni elemento della TP contiene il NPF per quella pagina virtuale
  - Quindi: TP[NPV] = NPF
- Ogni processo ha la sua TP
  - Lo spazio di indirizzamento virtuale è condiviso tra più processi
- Ogni pagina virtuale può corrispondere ad una qualsiasi pagina fisica
- In linea di principio, la TP contiene una riga per ogni pagina virtuale del processo

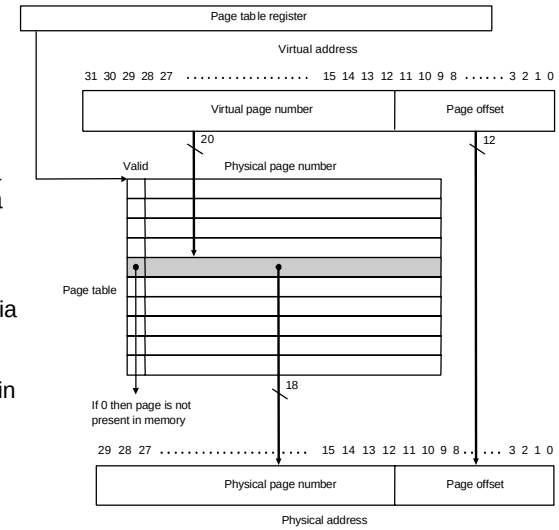
Tabella delle pagine (2)

- Il sistema operativo è responsabile dell'allocazione della memoria fisica e dell'aggiornamento delle TP
- La TP risiede in memoria
  - Il registro della TP (*page table register*) contiene l'indirizzo in memoria fisica della TP del processo correntemente in esecuzione



Esempio di tabella delle pagine

- 20 bit per NPV
- 12 bit per offset
- 18 bit per NPF
- La TP ha 2^{20} righe
- Il *bit di validità* indica se la corrispondenza registrata è legale
  - Uguale a 0: la pagina non è presente in memoria (*page fault*)
  - Uguale a 1: la pagina è presente in memoria

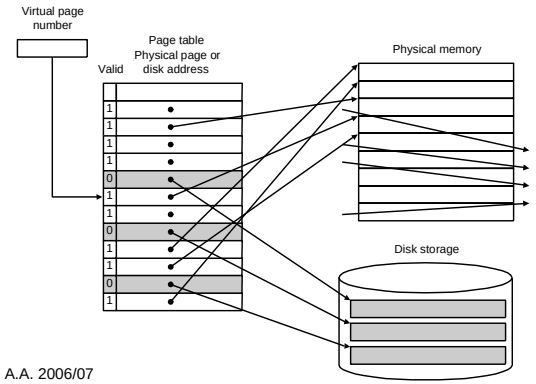


Page fault

- Quando un processo richiede una pagina che non è nella memoria principale, il processore genera un *errore di pagina* (*page fault*)
  - Page fault = cache miss per la cache
- In caso di page fault, la pagina deve essere caricata dal disco
  - Fault penalty grande
    - E' utile che le pagine siano grandi
    - Le letture dal disco hanno un costo iniziale elevato, dovuto ai movimenti meccanici dei dispositivi
  - E' quindi importante ridurre i page fault
  - Page fault gestiti a software tramite l'intervento del sistema operativo

Page fault (2)

- Al loading del processo, viene creato su disco l'immagine delle varie pagine del programma e dei dati (*swap space*)
- La TP può anche essere usata per memorizzare gli indirizzi delle pagine su disco
  - Indirizzi usati dal sistema operativo per gestire il page fault ed il rimpiazzo delle pagine



Bit nella tabella delle pagine

- Oltre al bit di validità, a ciascuna entry nella TP sono associati altri bit:
- *Bit di modifica* (dirty bit)
  - Indica se una pagina è stata modificata
  - Necessario in quanto per la memoria virtuale si usa la politica di scrittura write-back
- *Bit di riferimento* (reference bit)
  - Indica se, in un certo lasso di tempo, la pagina è stata acceduta
  - Settato ogni volta che la pagina è acceduta ed azzerato periodicamente dal SO
  - Usato dal SO per implementare la politica di rimpiazzamento delle pagine LRU (Least Recently Used)

Dimensioni della tabella delle pagine

- Per indirizzi virtuali grandi la TP diventa enorme
  - Es.: indirizzo virtuale a 32 bit e pagine da 4 KB
    - 20 bit per NPV
    - TP con 2^{20} entry
    - Se ciascuna entry è di 4 B, occorrono 4 MB ($4 \cdot 2^{20} = 2^{22}$ B) per la TP
- Se ci sono molti programmi in esecuzione, è necessaria una gran quantità di memoria solo per memorizzare le varie tabelle delle pagine!
- Si usano tecniche per ridurre lo spazio occupato dalla TP

Ridurre lo spazio della tabella delle pagine

- Diverse tecniche basate sull'osservazione che i programmi (piccoli) usano solo una piccola parte della TP a loro assegnata e che c'è anche una certa località nell'uso della TP

1. La tecnica più semplice consiste nell'usare un registro limite, che restringe la dimensione della tabella di ciascun processo
   - La dimensione della tabella delle pagine cresce dinamicamente in base al bisogno
   - Svantaggio: lo spazio di indirizzamento può crescere in una sola direzione
2. Per consentire la crescita in due direzioni, si usano due registri limite e due tabelle delle pagine
   - E' una forma di segmentazione della memoria invisibile al programma applicativo
   - Svantaggio: non funziona bene quando lo spazio degli indirizzi è usato in modo sparso anziché contiguo

Ridurre lo spazio della tabella delle pagine (2)

3. Tabella delle pagine multi-livello
   - Gerarchia di tabelle: ogni livello contiene una tabella che memorizza i puntatori alla tabella del livello sottostante
     - Il livello più basso è composto da tabelle che contengono la mappatura effettiva
   - Permette di memorizzare in locazioni non contigue la tabella delle pagine attivamente utilizzate dal processo
   - Svantaggio: il processo di traduzione delle pagine è più complicato
4. Tabella inversa delle pagine
   - Si usa una funzione hash per tradurre l'indirizzo virtuale in una entry della tabella delle pagine
   - La tabella delle pagine ha tante entry quante sono le pagine fisiche
   - Svantaggio: il processo di traduzione delle pagine è più complicato
5. Paginazione della tabella delle pagine

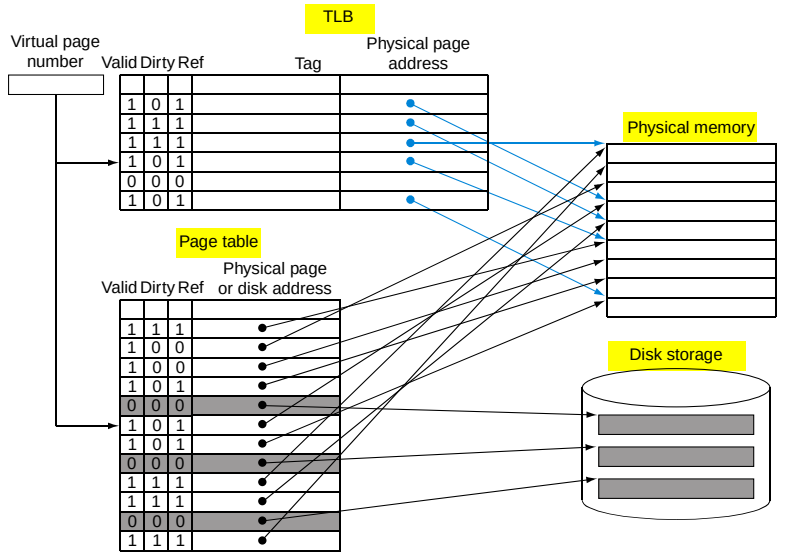
TLB: traduzione veloce degli indirizzi

- In linea di principio: ogni accesso alla memoria principale richiede due tempi di accesso
  - Uno per accedere alla TP ed ottenere l'indirizzo fisico, uno per accedere all'informazione richiesta
- Per migliorare le prestazioni si sfrutta la località dei riferimenti
  - Quando si traduce un numero di pagina virtuale, il risultato sarà probabilmente necessario di nuovo entro breve tempo (i riferimenti ad una stessa pagina hanno località spaziale e temporale)
- Si ricorre ad una cache di traduzione, detta **translation-lookaside buffer** (TLB), che registra le traduzioni effettuate di recente
- La traduzione dal NPV nel corrispondente NPF viene compiuta dalla MMU, che include il TLB

TLB: traduzione veloce degli indirizzi (2)

- Il TLB contiene un sottoinsieme delle corrispondenze tra NPV e NPV memorizzate nella TP
- Il TLB è una cache:
  - Ogni elemento del TLB contiene un campo etichetta ed un campo dati
  - Il campo tag contiene *una parte* del NPV
  - Il campo dati contiene un NPF
  - Inoltre, ciascuna riga del TLB contiene bit di validità, bit di riferimento e bit di modifica
    - Non si accede più necessariamente alla TP per ogni riferimento in memoria

TLB: traduzione veloce degli indirizzi (3)



Hit e miss nel TLB

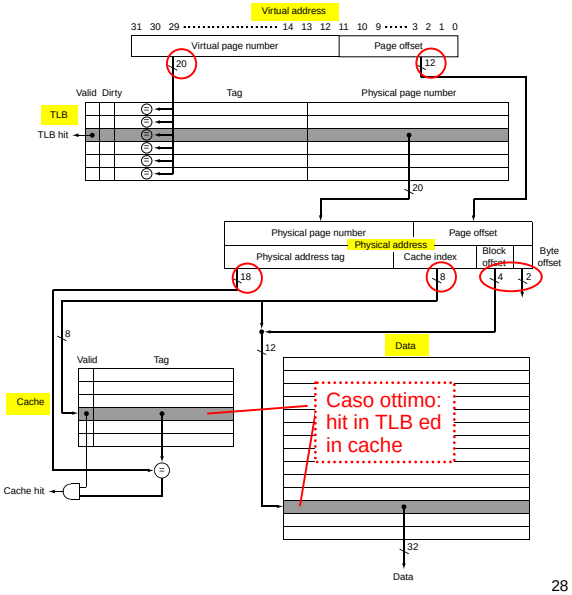
- Dato un indirizzo virtuale di memoria, si cerca il NPV nel TLB (confronto dei tag)
- In caso di hit nel TLB:
  - Si usa il NPF per generare l'indirizzo fisico, si pone a 1 il bit di riferimento; se l'operazione richiesta è una scrittura, si pone a 1 anche il bit di modifica
- In caso di miss nel TLB:
  - Si deve verificare se si tratta solo di miss nel TLB oppure di page fault
    - La pagina è in memoria: manca semplicemente la traduzione nel TLB, si carica la traduzione dalla TP nel TLB e si ripete l'accesso
    - La pagina non è in memoria: il miss nel TLB diventa un page fault, si genera un'eccezione che viene gestita dal sistema operativo
  - Il numero di elementi nel TLB è molto minore del numero di pagine nella memoria principale → miss sul TLB molto più frequenti dei veri page fault

TLB e gestione di miss

- I miss (fallimenti) di accesso al TLB sono gestibili da hw o da sw
  - La differenza di prestazioni è modesta
- Miss nel TLB: scelta dell'elemento da sostituire
  - Al momento della sostituzione, si devono copiare i bit di validità e di modifica nell'elemento della TP
    - La copia avviene solo in caso di sostituzione, non ad ogni modifica (strategia *write-back*)
- Grado di associatività del TLB
  - Da completamente associativi (TLB piccoli) a soluzioni set-associative
- Valori tipici del TLB
  - Da 16 a 512 entry, con blocchi di 1-2 elementi (da 4 a 8 byte per blocco)
  - Hit time: 0,5-1 cicli di clock
  - Miss rate: 0,01% - 1%
  - Miss penalty: da 10 a 100 cicli di clock

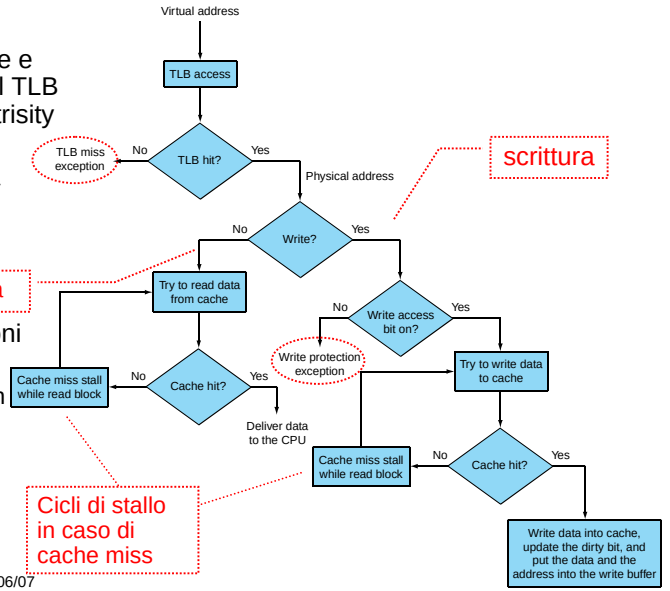
Esempio: il TLB dell'Intrisity FastMATH

- Pagine da 4 KB
- Indirizzi fisici da 32 bit
- Indirizzi logici da 32 bit
  - Struttura dell'indirizzo logico: 20 bit per NPV e 12 bit per offset, essendo 12=$\log_2$(4K)
- TLB completamente associativo con 16 elementi e condiviso per dati e istruzioni
- Ciascun elemento del TBL da 64 bit
  - 20 bit per tag
  - 20 bit per NPF
  - bit di validità, bit di modifica ed altri bit (anche per protezione)



TLB e cache

- Gestione di letture e scritture usando il TLB e la cache nell'Intrisity FastMATH
- Cache indirizzata fisicamente
- Notare le eccezioni
  - TLB miss
  - Write protection



Diversi tipi di miss

- In una gerarchia di memorie con TLB, memoria virtuale e cache abbiamo tre tipi di miss: TLB miss, page fault, cache miss
- Analizziamo le combinazioni possibili con almeno un miss (sono 7)

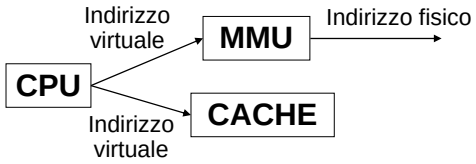
TLB	TP	Cache	E' possibile? Quando?
hit	hit	miss	Possibile, ma la TP non viene mai consultata in caso di hit nel TLB!
miss	hit	hit	Miss in TLB ma elemento in TP; si ritenta l'operazione e si trova l'elemento in cache
miss	hit	miss	Miss in TLB, elemento in TP, miss in cache; trasferimento in cache
miss	miss	miss	Miss in TLB seguito da page fault; si ritenta l'operazione e si verifica miss in cache
hit	miss	miss	Impossibile: non si ha traduzione dal TLB se la pagina non è in memoria
hit	miss	hit	Impossibile: non si ha traduzione dal TLB se la pagina non è in memoria
miss	miss	hit	Impossibile: non si hanno dati in cache se la pagina non è in memoria

Quale indirizzo per la cache?

- Finora abbiamo considerato che l'accesso in cache avvenga usando l'indirizzo fisico di memoria
  - Prima dell'accesso in cache si traduce l'indirizzo virtuale in indirizzo fisico
  - Cache **indirizzata fisicamente**: *indice fisico* e *tag fisico*

Quale indirizzo per la cache? (2)

- Alcune architetture consentono di accedere alla cache con un indirizzo completamente o parzialmente virtuale
- Cache **indirizzata virtualmente**: si usa solo l'indirizzo virtuale
  - *Indice virtuale* e *tag virtuale*
  - Problema dell'ambiguità degli indirizzi (address aliasing): due indirizzi virtuali per la stessa pagina fisica



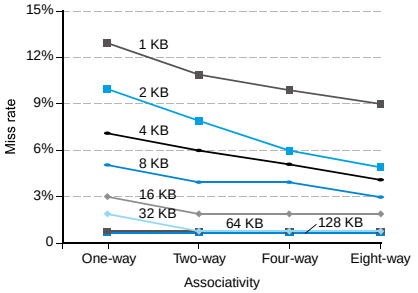
- Cache **indirizzata virtualmente ma con tag fisico**: si usa parzialmente l'indirizzo virtuale
  - *Indice virtuale* e *tag fisico*

Riepilogo per la gerarchia di memorie

1) Posizionamento di un blocco

Organizzazione	Numero di set	Blocchi per set
Indirizzamento diretto	Numero di blocchi in cache	1
Set associativa	Numero di blocchi in cache/grado associatività	Grado di associatività (da 2 a 16)
Completamente associativa	1	Numero di blocchi in cache

Soluzione usata per la memoria virtuale (una pagina virtuale ovunque in memoria principale)



Riepilogo per la gerarchia di memorie (2)

2) Individuare un blocco

Associatività	Metodo di individuazione	Confronti necessari
Indirizzamento diretto	Indice del blocco	1
Set associativa	Indice del set, ricerca i blocchi del set	Grado di associatività
Completamente associativa	Ricerca tutti i blocchi	Dimensione della cache
	Tabella di lookup separata	0

Soluzione usata per la memoria virtuale (tabella delle pagine)

Riepilogo per la gerarchia di memorie (3)

3) Sostituire un blocco

- Le due strategie principali sono
  - LRU
  - Random
- Per la cache, la politica di sostituzione è implementata in hardware
  - Politica di sostituzione semplice (Random o LRU approssimata)
- Per la memoria virtuale, la politica di sostituzione è implementata dal software
  - La politica può essere più sofisticata, perché i miss sono meno frequenti ma più costosi
    - Tempo di accesso in memoria dell'ordine di 10^{-8} sec; tempo di accesso al disco dell'ordine di 10^{-3} sec

Riepilogo per la gerarchia di memorie (4)

4) Strategia di scrittura

- Le due principali strategie di scrittura sono
  - Write-through
  - Write-back
- Per la cache, si possono usare entrambe
- Per la memoria virtuale, si usa solo write-back
  - Troppo lungo il tempo di latenza del disco

Esempi: Pentium 4 e AMD Opteron

Characteristic	Intel Pentium P4	AMD Opteron
Virtual address	32 bits	48 bits
Physical address	36 bits	40 bits
Page size	4 KB, 2/4 MB	4 KB, 2/4 MB
TLB organization	1 TLB for instructions and 1 TLB for data Both are four-way set associative Both use pseudo-LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative, LRU replacement Both L2 TLBs are four-way set associativity, round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TLB misses handled in hardware

- Pentium 4 ha 1 TLB per dati e 1 TLB per istruzioni
- Opteron ha 2 TLB per dati e 2 TLB per istruzioni
  - TLB L1 e L2 per dati
  - TLB L1 e L2 per istruzioni

Esempi: Pentium 4 e AMD Opteron (2)

Characteristic	Intel Pentium P4	AMD Opteron
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	8 KB for data, 96 KB trace cache for RISC instructions (12K RISC operations)	64 KB each for instructions/data
L1 cache associativity	4-way set associative	2-way set associative
L1 replacement	Approximated LRU replacement	LRU replacement
L1 block size	64 bytes	64 bytes
L1 write policy	Write-through	Write-back
L2 cache organization	Unified (instruction and data)	Unified (instruction and data)
L2 cache size	512 KB	1024 KB (1 MB)
L2 cache associativity	8-way set associative	16-way set associative
L2 replacement	Approximated LRU replacement	Approximated LRU replacement
L2 block size	128 bytes	64 bytes
L2 write policy	Write-back	Write-back

Opteron: L1 cache indirizzata virtualmente ma con tag fisico

Opteron: L1 cache e L2 cache gestite con multi-level exclusion

Traduzione dell'indirizzo virtuale nell'Opteron

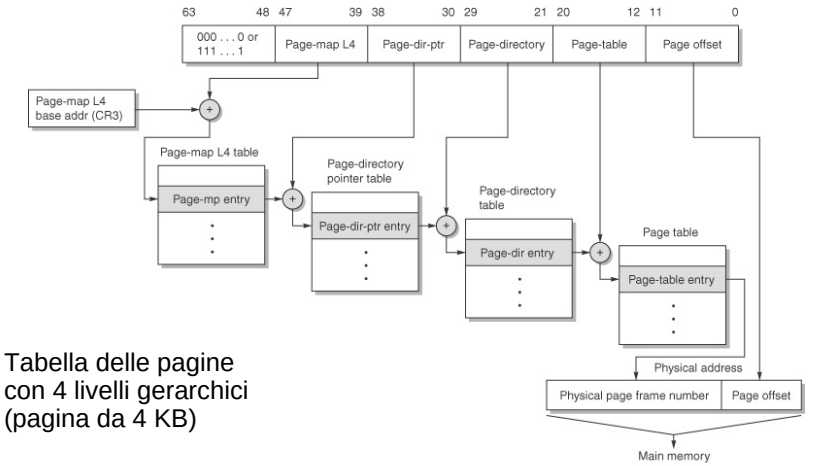
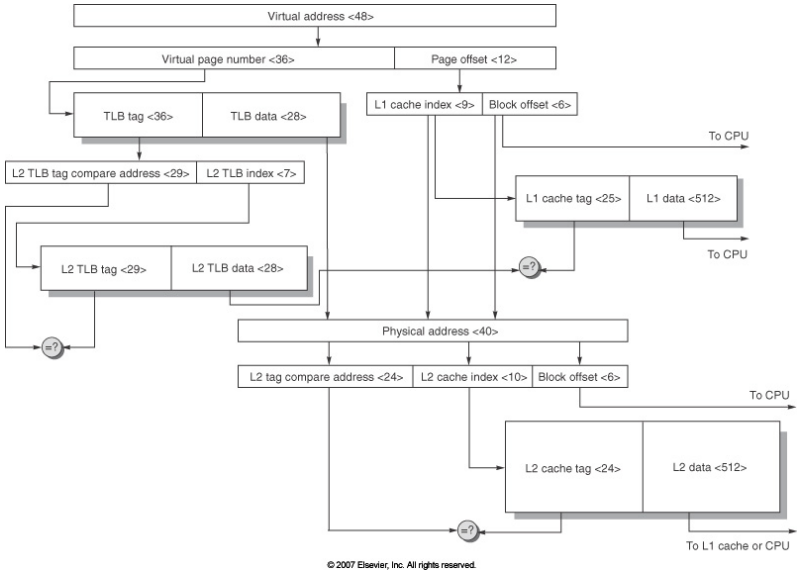


Tabella delle pagine con 4 livelli gerarchici (pagina da 4 KB)

© 2007 Elsevier, Inc. All rights reserved.

Traduzione degli indirizzi nell'Opteron



© 2007 Elsevier, Inc. All rights reserved.

La gerarchia di memoria dell'Opteron

Istruzioni

Dati

L1 cache

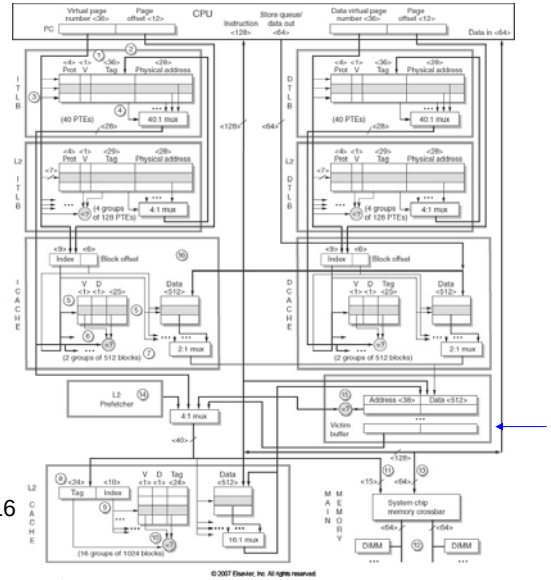
2^s = 64KB/64B·2

→ s = 9

L2 cache

2^s = 1024KB/64B·16

→ s = 10



© 2007 Elsevier, Inc. All rights reserved.