


Le virtual machine e la memoria virtuale

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimento Patterson 5: 5.6, 5.7.


Sommario

Virtual machines (VM)

Sviluppate negli anni 60 e riprese recentemente per:

- Sicurezza e isolamento di sotto-sistemi.
- Condivisione dello stesso sistema da parte di molti utenti (e.g. cloud computing)

Velocità dei processori rende il costo della virtualizzazione accettabile.

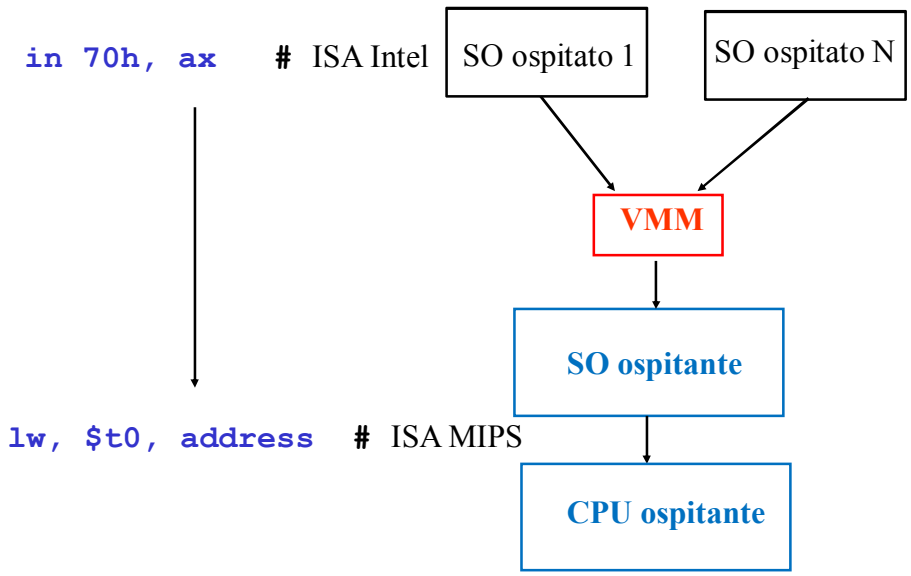
La VM fornisce un **ambiente completo** a **livello di sistema**, compatibile con una certa ISA.

Lo stesso calcolatore può **supportare più VM** e può quindi supportare più SO e ambienti, anche obsoleti quali ad esempio il DOS e quindi programmi non più eseguibili direttamente.

Le VM sono supportate da uno strato software particolare chiamato **Monitor** (VMM) o **Hypervisor**. Questo software fa da tramite tra le diverse VM e l'ambiente ospitante (hardware + SO).

Il costo principale è legato all'intervento del SO dell'ospitante per tradurre le richieste al SO della VM (I/O, memoria….)

Esempio



Il Monitor (VMM)

E' un ruolo delicato:

- Deve proteggere i SW ed i sistemi operativi ospitati.
- Deve proteggersi dai sistemi operativi ospitati.

Il software eseguito da una VM (software ospitato) deve essere eseguito esattamente **come se fosse eseguito sull'ISA nativa** (a parte le prestazioni).

Il software eseguito da una VM **non deve modificare nulla delle risorse della macchina ospitante**, **direttamente**, ma deve passare attraverso il SO della macchina ospitante.

Per esempio, nel caso in cui un processo ospitato (guest) sia basato su un timer (eccezione di sistema), questa dovrà essere simulata dal VMM, il quale potrebbe utilizzare a sua volta il timer di sistema o meno. Nel caso in cui due processi guest richiedano il timer di sistema, sicuramente in uno dei due casi, il timer sarebbe simulato dal VMM.

Solitamente un VMM ha privilegi di sistema, mentre una VM ha i privilegi di utente.

Il VMM e le istruzioni privilegiate

Cosa succede quando un SO ospitato fa una richiesta privilegiata (e.g. lancia un interrupt o un'eccezione?).

Queste devono essere intercettate (trap) dal VMM che le gestirà nel modo più appropriato tenendo conto dello stato della macchina e delle altre VM.

Questo processo può rilevarsi molto costoso soprattutto quando l'applicazione ospitata fa molto uso di I/O ma anche di altre funzionalità.

Applicazioni che eseguono principalmente funzioni di I/O o fanno poco utilizzo della virtualizzazione, sono quelle eseguite più velocemente dalle VM.

Il VMM emula (via SW) l'architettura ospitata e traduce nell'hardware dell'architettura ospitante.

Il VMM e le istruzioni privilegiate

Cosa succede quando un SO ospitato fa una richiesta privilegiata (e.g. lancia un interrupt o un'eccezione?).

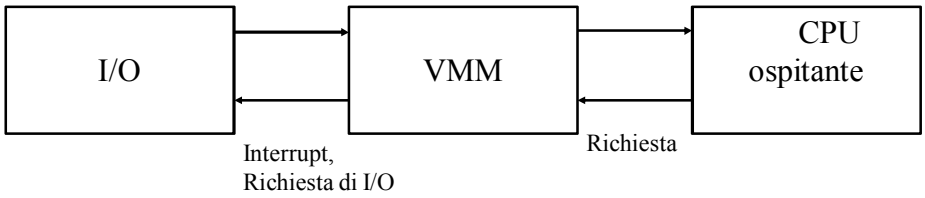
Queste devono essere intercettate (trap) dal VMM che le gestirà nel modo più appropriato tenendo conto dello stato della CPU opsitante e delle altre VM.

Questo processo può rilevarsi molto costoso soprattutto quando l'applicazione ospitata fa molto uso di I/O ma anche di altre funzionalità.

Applicazioni che eseguono principalmente funzioni di calcolo o fanno poco utilizzo della virtualizzazione, sono quelle eseguite più velocemente dalle VM.

Funzionamento del VMM

Comunicazione a 2 vie



Stato di una VM

- PC
- Registri
- Registro della tabella delle pagine.

A partire dallo stato è possibile ricostruire un processo e farne ripartire l'esecuzione.

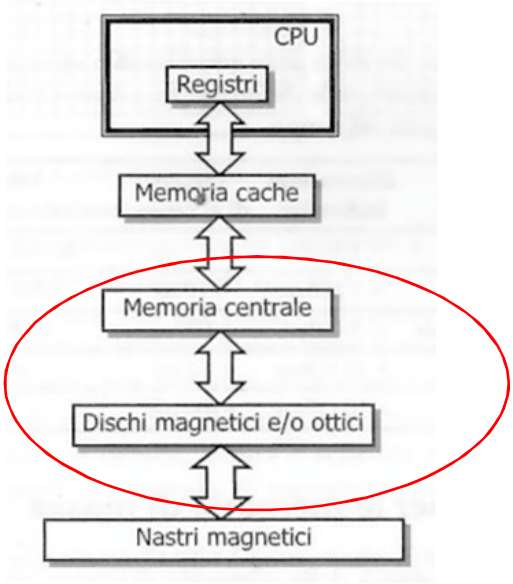
Sommario

Le virtual machine

La memoria virtuale

Accesso alla memoria virtuale

Gerarchia di memorie



La memoria virtuale

La memoria principale può essere vista come una memoria cache del disco.

Una volta si utilizzavano gli *overlay* per caricare/scaricare segmenti di codice+dati da disco a memoria principale.

La memoria principale è la stessa per più programmi in esecuzione, i quali condividono lo stesso spazio di indirizzamento e sono inizialmente residenti su disco.

La memoria virtuale consente di mappare lo spazio di memoria di ciascun programma nella memoria principale fisica, evitando sovrapposizioni e consente **una gestione sicura della memoria**.

La memoria virtuale consente anche di nascondere al programmatore la dimensione limitata della memoria principale rispetto alla memoria su disco.

La memoria virtuale consente di gestire in modo efficace le VM.

Terminologia

Un blocco di memoria virtuale e di memoria principale viene chiamato **pagina** (equivalente alla linea della cache).

L'accesso ai dati avviene mediante l'indirizzo virtuale.

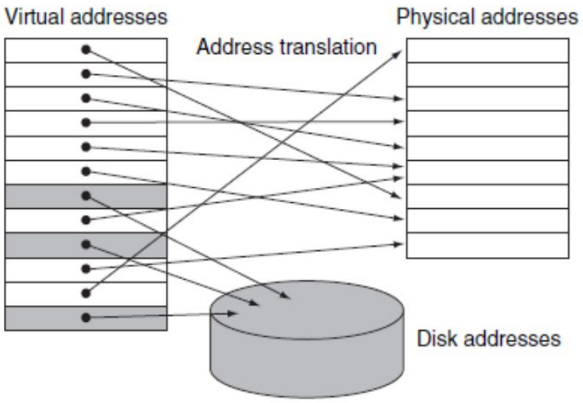
Una miss dalla memoria principale in giù viene chiamata **page fault**. L'accesso avviene mediante l'indirizzo fisico derivato dall'indirizzo virtuale.

L'indirizzo virtuale viene tradotto in un indirizzo fisico della memoria principale. Se si verifica una hit, l'esecuzione continua correttamente, se si verifica un page fault, occorre caricare la pagina in memoria principale.

La traduzione da indirizzo virtuale ad indirizzo fisico viene fatta al momento del caricamento del programma mediante la **rilocazione**.

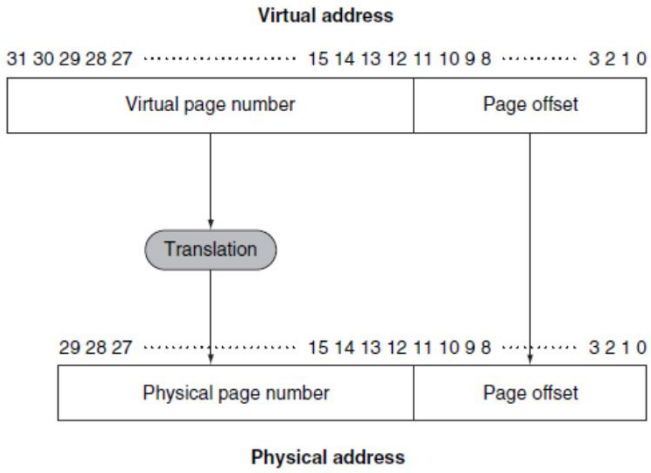
Indirizzi fisici e indirizzi virtuali

Tabella delle pagine



Ciascun programma ha il suo spazio di indirizzamento (virtuale) che viene mappato su disco e sulla memoria principale (indirizzi fisici).

La memoria virtuale



Altro esempio di base (numero di pagina) + offset

- Il numero di pagine fisico è determinato dalla quantità di memoria fisica.
- Il numero di pagine virtuali è virtualmente infinito.

La memoria virtuale

Strategie diverse da quelle utilizzate nelle cache. Obbiettivo principale è **nascondere la penalità di page fault.**

Memory technology	Typical access time	$ per GiB in 2012
SRAM semiconductor memory	0.5–2.5 ns	$500–$1000
DRAM semiconductor memory	50–70 ns	$10–$20
Flash semiconductor memory	5,000–50,000 ns	$0.75–$1.00
Magnetic disk	5,000,000–20,000,000 ns	$0.05–$0.10

L'accesso a disco (magnetico) può essere quasi un milione di volte più lento. 100 volte più lento nel caso ottimale di disco (a stato solido).

NB Questa penalità è dovuta largamente alla penalità per accedere alla prima parola della pagina. Con il trasferimento a burst il resto dei dati viene trasferito più velocemente.

Criteri di organizzazione della memoria principale

Le pagine devono essere sufficientemente ampie per ammortizzare i tempi di accesso (anche a seguito di una hit). Pagine di 4 KByte sono tipiche.

Il posizionamento delle pagine virtuali nella memoria principale è **completamente associativo** per massimizzare il riempimento ed evitare i page fault.

La gestione dei page fault può essere software (SO) visto il lungo tempo a disposizione per gestirli. Si ha così un posizionamento ottimizzato delle pagine nella memoria fisica.

La scrittura in memoria **di una pagina** viene gestita in modalità *write-back*.

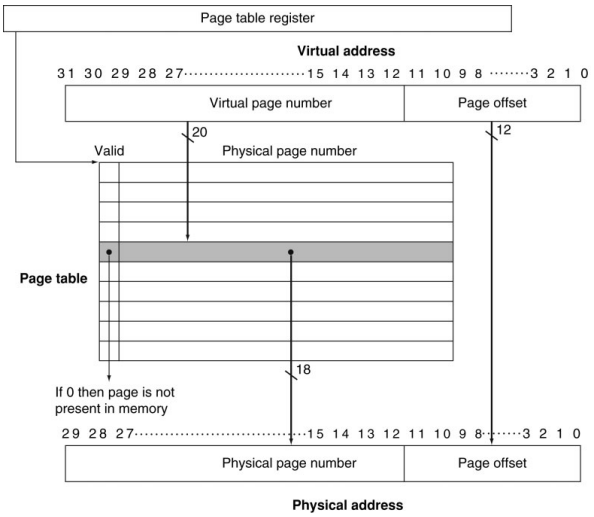
Posizionamento delle pagine

Non si può cercare una pagina virtuale, controllando tutte le pagine contenute nella memoria fisica.

Si utilizza la **Tabella delle pagine**, a sua volta residente nella memoria principale, indirizzata dal ***registro della tabella* delle *pagine***.

La tabella delle pagine è indicizzata con il numero della pagina virtuale e contiene la traduzione del numero di pagina virtuale in numero di pagina fisica. Controlla anche se la pagina fisica è già presente nella memoria fisica.

Tabella delle pagine

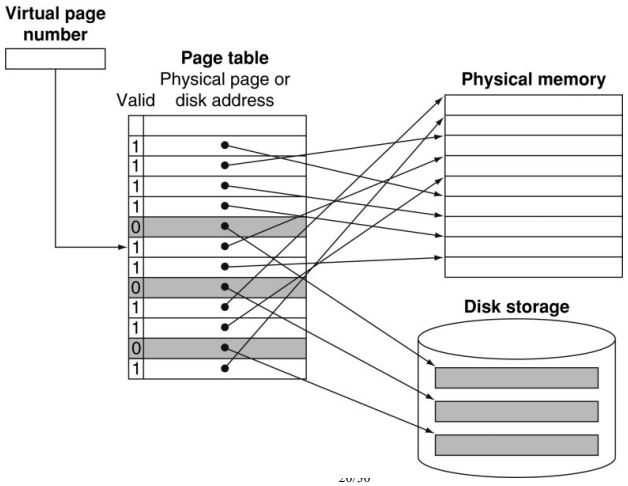


L'altezza della memoria è pari al numero di pagine virtuali. Per ogni pagina virtuale è riportata la traduzione.

Gestione dei page fault

La pagina richiesta può non essere contenuta in memoria, ma sarà contenuta su disco: sarà contenuta nello **swap space** di ogni processo.

La posizione su disco può essere data dalla stessa tabella delle pagine.



Scelta della pagina da sostituire

Il SO registra quali processi e quali indirizzi virtuali sono associati a ogni pagina fisica (eventualmente più processi)

LRU approssimato. **Use bit** (**Reference bit**)**.** Quando una pagine viene utilizzata viene impostato a 1 (cf. dirty bit). Periodicamente tutti gli use bit vengono azzerati.

In caso di page fault, la pagina virtuale viene trasferita in una pagina il cui Use bit è = 0.

NB Lo use bit è differente dal bit di validità.

Sommario

Le virtual machine

La memoria virtuale

Accesso alla memoria virtuale

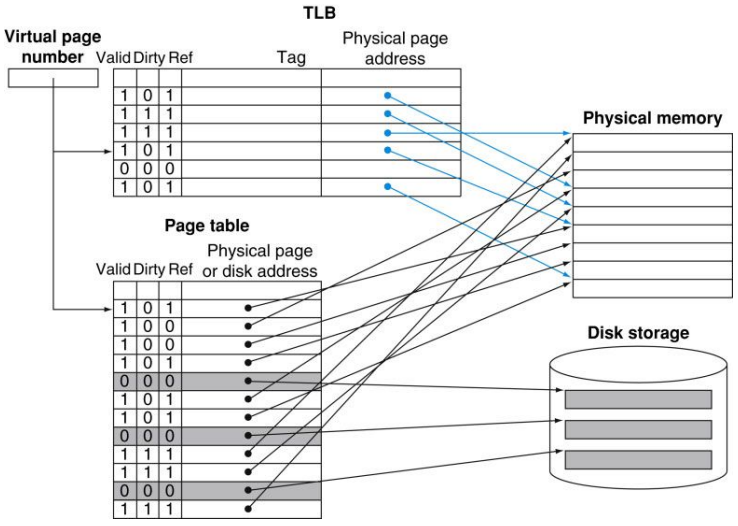
IL TLB – Translation Lookaside buffer

Con le PT (Tabelle delle pagine) ogni accesso alla memoria principale richiede in realtà 2 accessi.

Si sfrutta il principio di località: quando si accede la dato di una certa pagina, facilmente si avrà bisogno di un altro dato della stessa pagina. Si può evitare di dovere ritradurre una seconda volta. Come?

Si introduce un buffer HW: **Translation Lookaside buffer** che memorizza le traduzioni.
- Dimensioni: 16-512 elementi (linee)
- Dimensione della linea (1-2 pagine, 4-8 byte ciascuna)
- Tempo di hit: 0,5-1 ciclo di clock
- Penalità di miss: 10-100 cicli di clock (per caricare la traduzione)
- Frequenza di miss: 0,01%-1%

TLB



Il TLB è una cache a tutti gli effetti, associativa con un numero di elementi molto più ridotto del numero delle pagine virtuali.
Si può, in alternativa, configurare come una hash table.

Accesso alla memoria principale

1) Invio al TLB dell'indirizzo virtuale
2) Hit -> Indirizzo della pagina fisica (se TAG = num_pagina e valid = 1).
   Se la richiesta è in scrittura, dirty_bit = 1.

3) Miss -> Si verificano due casi:
3a) La traduzione non è nel TLB ma la pagina **è** in memoria principale.
3b) La traduzione non è nel TLB e la pagina **non è** in memoria principale (page fault).
Le miss del TLB saranno molto più frequenti dei page fault.

3a) La traduzione viene recuperata dalla tabella delle pagine e scritta del TLB. Il dirty bit riferito a quella pagina viene scritto nella tabella delle pagine e reimpostato a zero. La pagina viene infine caricata dalla memoria principale.

Gestione dei page fault

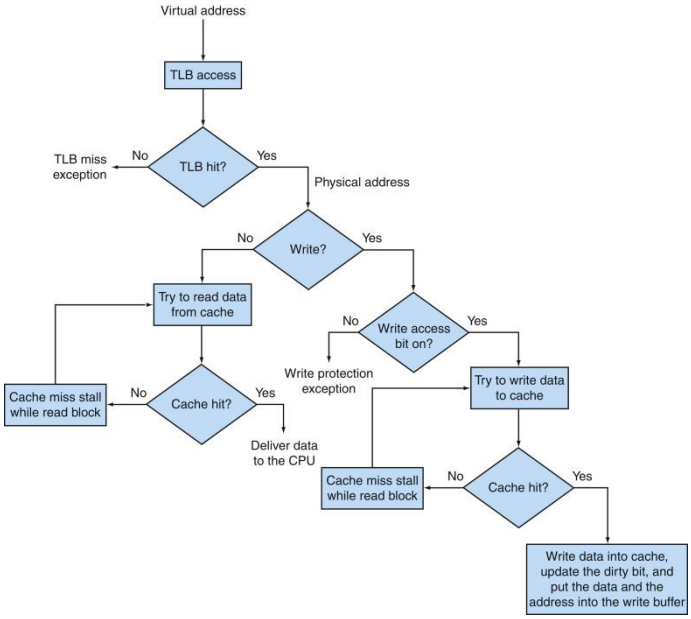
Cercare la pagina virtuale nella tabella delle pagine e localizzare la pagina fisica corrispondente (su disco).

Scegliere una pagina fisica da sostituire; se la pagina fisica identificata ha il bit dirty = 1 (ci sono delle parole modificate), deve essere scritta su disco prima di caricare la nuova pagina nella memoria principale.

Iniziare il trasferimento della pagina da disco a memoria principale.

Aggiornare la traduzione della pagina virtuale nella pagina fisica e reset del bit diry.

Gestione della richiesta di un dato



Possibili situazioni in Cache, PT e TLB

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

Implementazione della protezione

Page Table pointer and the TLB are stored inside the OS space. They can be accessed only in OS Kernel mode.

We have to avoid that a process changes its own PT, making a virtual page mapping into a physical page of a different program. When accessing a physical page, a program should own that page.

This can be obtained by a personalized PT that can be modified only by the OS.

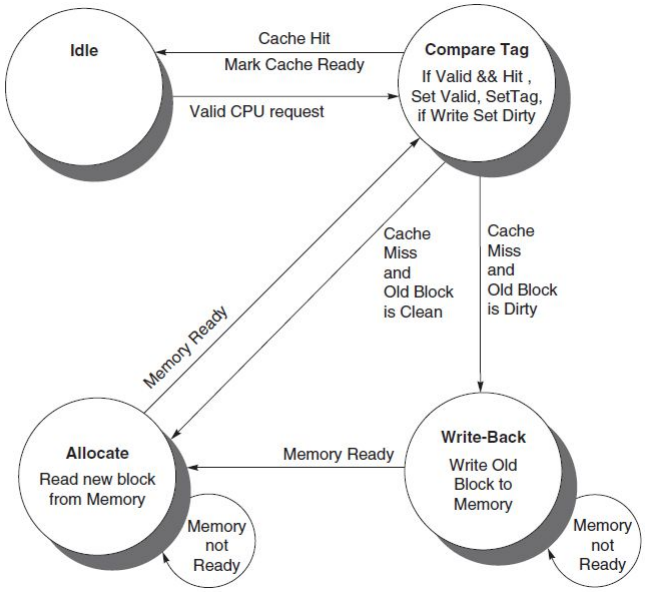
The write access bit can be used to protect the page from writing.
If a process wants to share a page with another, the two virtual pages should point to the same physical page.

All these mechanisms are mediated by the OS in kernel mode.

Alternative alla paginazione

La memoria non paginata viene suddivisa in segmenti, i quali non hanno dimensione fissa.

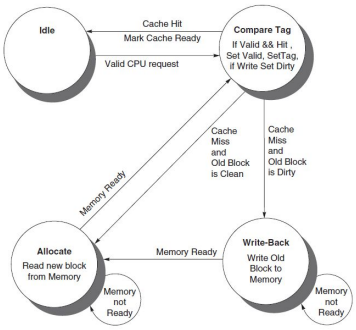
Controllore di una memoria cache



Funzionamento del controller

4 STATES:

- **Idle:** waits for a valid read/write from the processor
- **Compare tags:** identify the cache line and check TAG and validity bit.
  - HIT -> tag is found and line is valid
  - MISS -> tag is not found.
    - Dirty bit is set to 1 ➔ the line has to be copied in Main Memory before overwriting (write-back)
    - Dirty bit i set to 0 ➔ the data can be immediately be trasferred from Main Memory (write allocate)
- **Allocate:** Read a new cache line from main memory into cache.
- **Write-Back:** Write old cache line into main memory.



Controllore di una memoria cache

Macchina a stati finiti:

Stati = {Idle, CompareTag, Write-Back, Allocate}
Input = {Valid_CPU_Request, Cache_Hit, Dirty, Memory_Ready}
Uscite = {Read_new_Block, Write_old_block, SetHit, SetMiss, SetValid, SetTag, SetDirty}

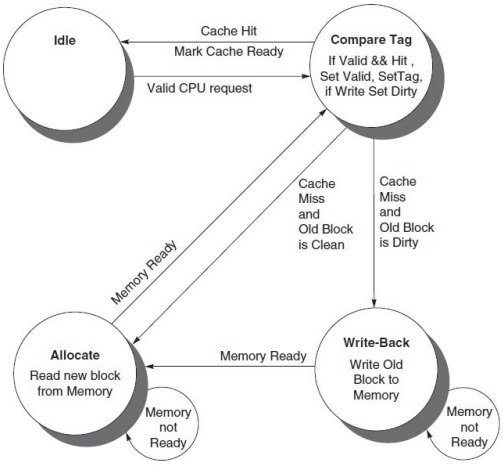
NB SetDirty imposta il bit "Dirty", ma costituisce anche un input.

Stato inziale: idle.

Stato prossimo: f(x,u)

Uscita: g(x)

Controllore di una memoria cache



Sintetizzare come macchina di Huffman

Sommario

Le virtual machine

La memoria virtuale

Accesso alla memoria virtuale