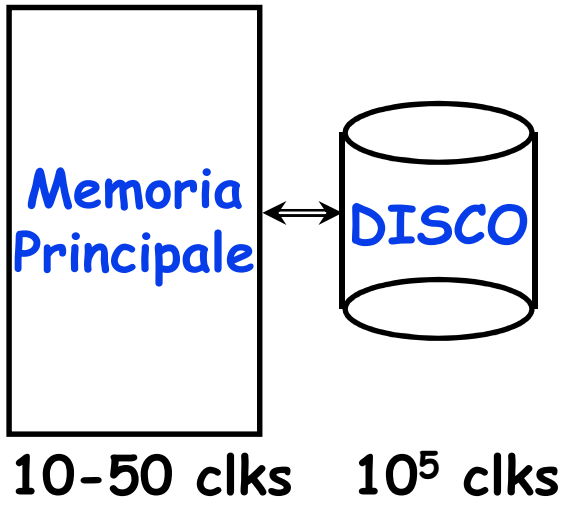
Lezione 19
Memoria Virtuale

http://www.dii.unisi.it/~giorgi/didattica/arcal1

Disk Caching (!= Memoria Virtuale)

- Idea: utilizzare la Memoria Principale come se fosse una cache del disco
  - Scopo: ridurre il tempo medio di accesso ai file che stanno su disco



- Funzionamento: e' una cache implementata completamente a software
  - Memorizza i blocchi di disco piu' recentemente utilizzati
  - Su hit, ottengo un tempo di accesso di $10^5/10=10^4$ volte maggiore
  - La tecnica e' usata da Windows, Linux e altri sistemi operativi

Memoria Virtuale

- Obiettivi delle tecniche di caching Memoria ←→ Disco
  - Disk Caching → migliorare il tempo di accesso file
  - Memoria Virtuale → creare uno spazio di indirizzamento piu' largo (con tempi di accesso simili alla memoria principale)

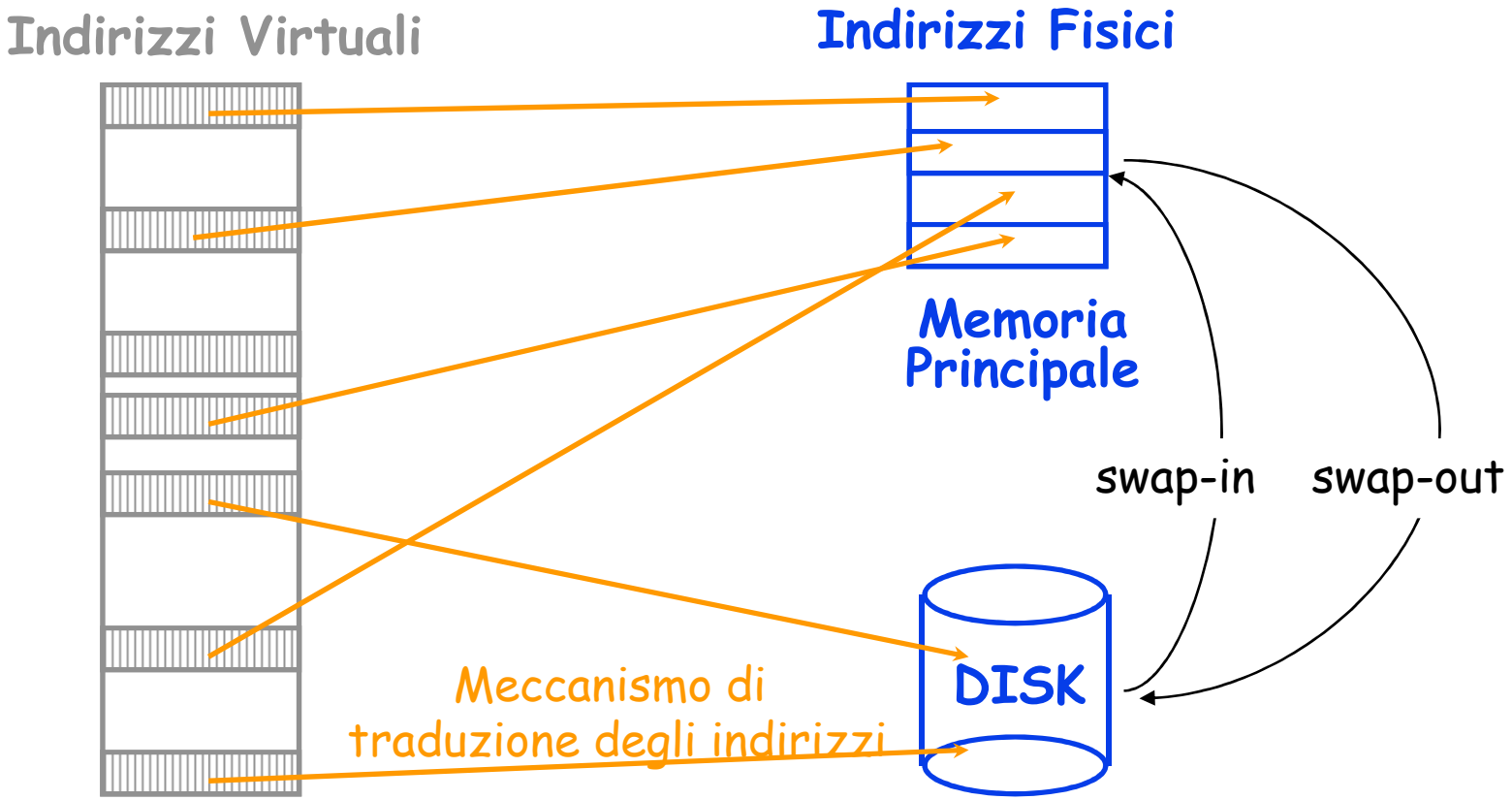
- Uso il disco come "memoria esterna" rispetto alla memoria principale: il livello successivo nella gerarchia di memoria
  - La memoria contiene una copia delle istruzioni/dati piu' recentemente usati
  - Tutto il resto sta su disco

Memoria Virtuale: vantaggi

- Crea l'illusione di un enorme quantita' di memoria (molto piu' ampia della memoria fisica realmente disponibile nel calcolatore)
  - Rimuove i vincoli di programmazione legati all'avere una memoria di dimensioni limitate
- Rilocazione dei programmi non necessaria
  - Il programma viene caricato in uno spazio di memoria che e' completamente vuoto per ogni programma, ovvero ogni programma ha a disposizione tutti gli indirizzi di uno proprio spazio virtuale privato
- Protezione dei programmi
  - Porzioni dello spazio di indirizzamento possono essere rese condivise in maniera controllata ed efficiente
  - Un programma non puo' accedere allo spazio di indirizzamento privato di un altro programma
- Il tutto funziona grazie ancora al principio di localita'
  - In un certo intervallo di tempo lo working set del programma e' abbastanza limitato

Memoria Virtuale: Funzionamento

- L'unita' di trasferimento fra memoria e disco e' la pagina
- La pagina ha una dimensione tipica di 4-16KB



- La Pagina Virtuale indirizzata puo' trovarsi sul disco anziche' nella memoria fisica→ swap-in

Tempi di accesso

- Rapporti tipici fra i tempi di accesso di vari tipi di memoria

L1→L2 $$\frac{t_{penalty,L1}}{t_{hit,L1}} = 5$$

L2→MEM $$\frac{t_{penalty,L2}}{t_{hit,L2}} = 5$$

MEM→DISCO $$\frac{t_{penalty,MEM}}{t_{hit,MEM}} = 10000$$

- Nel caso di un page-miss, o meglio di un page-fault, il tempo di penalty risulta molto grande

Esempio

- Il processore P, viaggia a 30 MIPS e 1/3 delle istruzioni generate da un certo programma sono di tipo LD
  → P fa 30/3 = 10 milioni di load/secondo
- Supponiamo che L1 e L2 insieme riescano a servire il 95% delle richieste → la Memoria Principale deve servire $0.05 \times 10^7 = 5 \times 10^5$ load/secondo
- Supponiamo che il Disco possa servire 50 richieste/secondo (considerando un rapporto di 1:10000 per servire un miss). Determinare il tempo di esecuzione del programma ipotizzando che l'hit-rate della memoria principale assuma i seguenti valori:

Hit Rate Memoria Principale	Richieste al disco per secondo	Tempo di Esecuzione del Programma
100%	0	1 (normalizzato)
99.999%	5	_____
99.99%	50	_____
99.9%	50	_____
90.0%	50	_____

Memoria Virtuale: Caratteristiche

- La memoria principale si comporta come una cache
- Le caratteristiche di tale cache sono
  - $A=A_{MAX}$ → cache fully associative: una pagina virtuale puo' essere mappata su qualsiasi pagina fisica
  - B=4-16KB
  - C=Dimensione della Memoria
  - Politica di Rimpiazzamento=LRU
  - Politica di Scrittura=WB (Write-Back)
  - Politica di Scrittura su Miss=WA (Write Allocate)

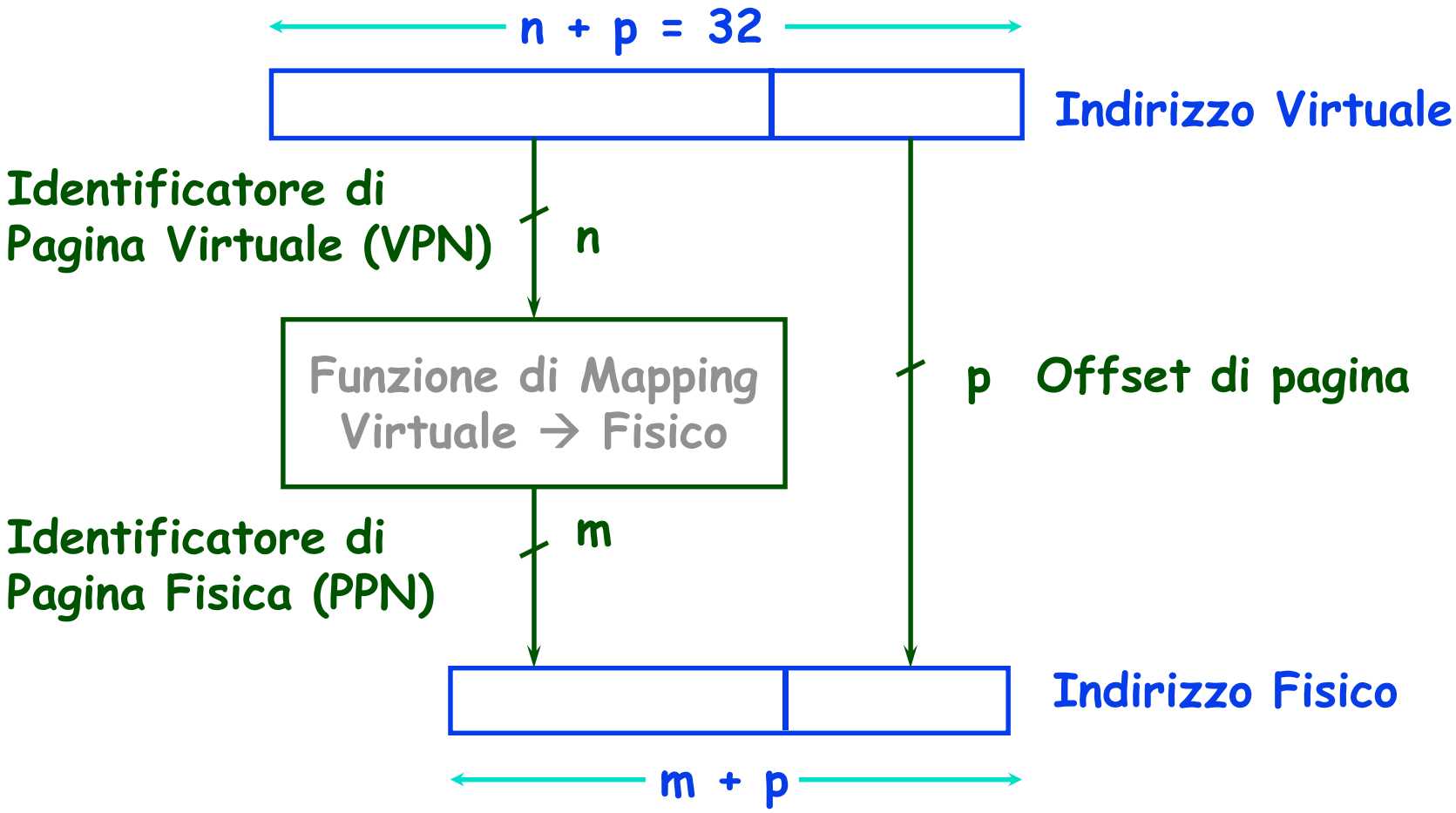
Memoria Virtuale: Terminologia

- Working set : Insieme di pagine che e' necessario mantenere in Memoria per ottenere un hit rate accettabile

- Thrashing : Perdita di prestazioni conseguente al fatto che lo working set e' diventato piu' grande della Memoria Principale

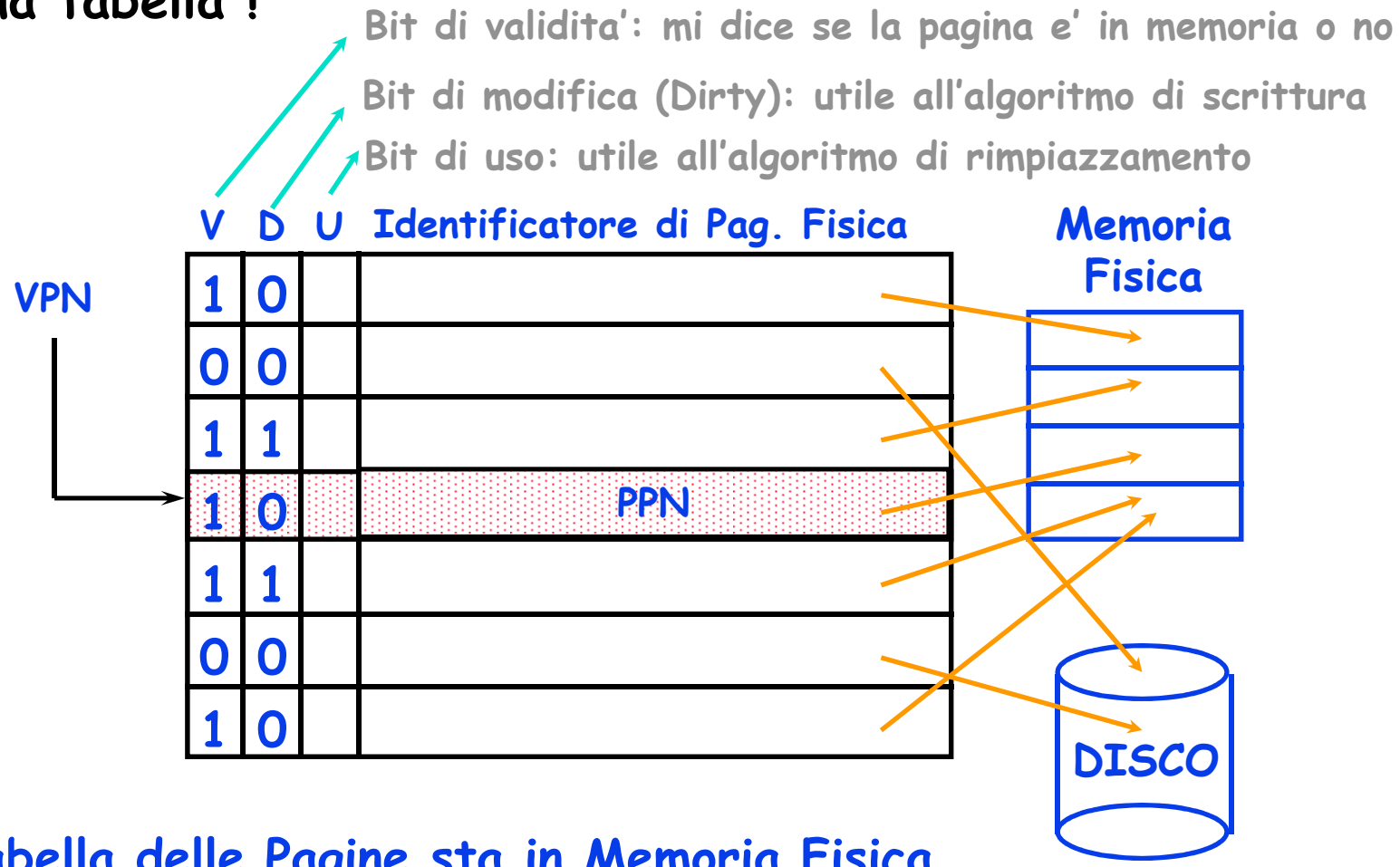
- Page Swap : Trasferimento di una pagina nuova in memoria principale e conseguente riscrittura su disco di una pagina "vittima"

Memoria Virtuale: schema logico



Funzione di Mapping Virtuale-Fisico

- E' una tabella !



La Tabella delle Pagine sta in Memoria Fisica

Quanto e' grande questa tabella?

Gestione del Page-Fault (Page-Fault Handler)

- Passi per leggere una locazione della Memoria Principale
  1) Presento un indirizzo virtuale al sistema di Memoria
  2) Accedo alla Tabella delle Pagine per vedere se la pagina e' in RAM
  3) Se la pagina e' in RAM accedo nuovamente alla RAM per prendere il dato
  4) Se la pagina non e' in RAM → lancio un interrupt al processore e chiamo una routine di servizio del Sistema Operativo denominata Page-Fault Handler
  5) Il Page-Fault Handler si occupa di fare il page-swap (tramite il DMA) e porta in RAM la pagina richiesta prelevandola da disco
  6) Viene ripetuta la lettura: stavolta la pagina si trovera' in RAM

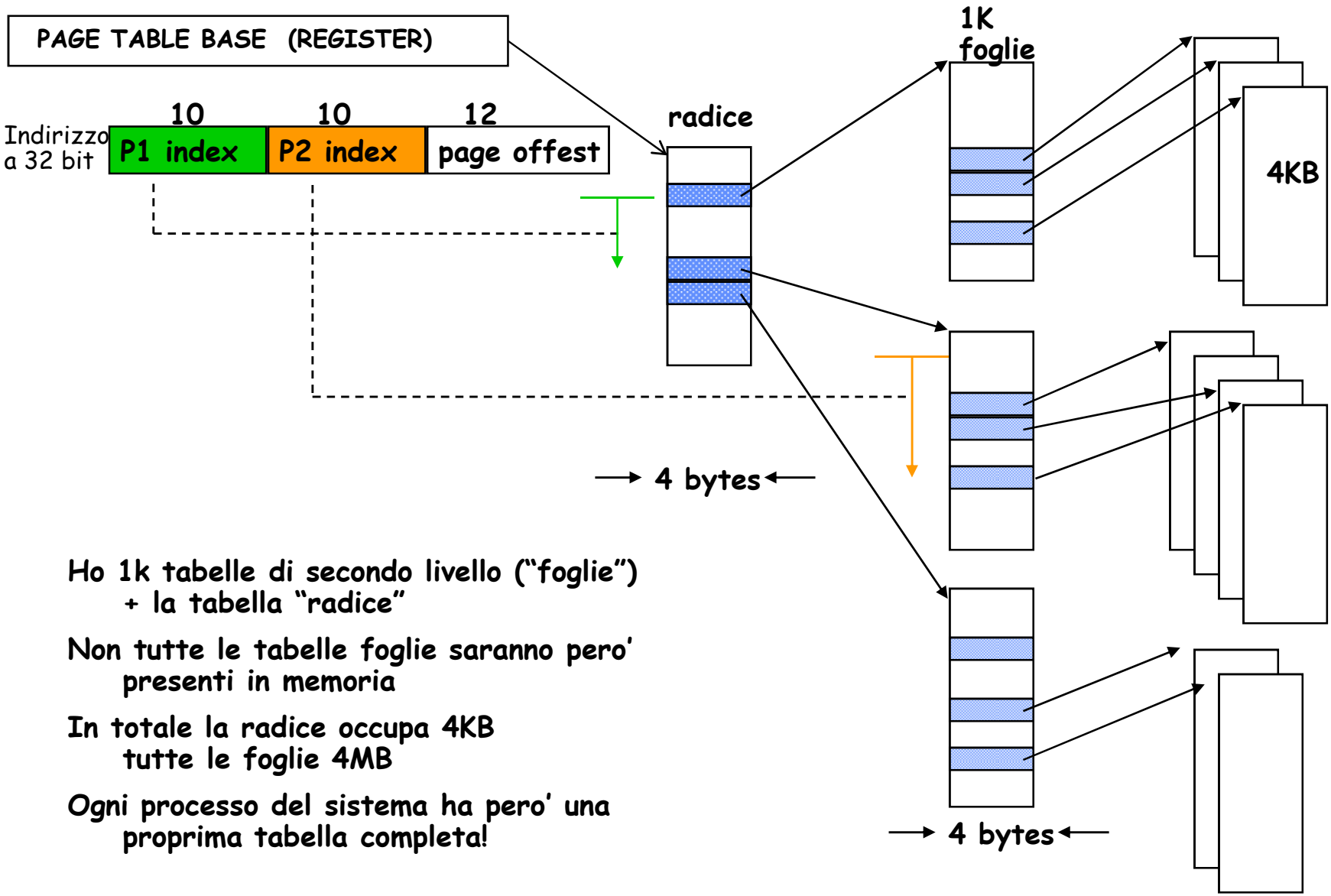
- Anche se avessi una percentuale del 100% di Hit-Rate in Memoria Principale, la penalty dovuta all'uso della memoria virtuale comporterebbe un raddoppio dei tempi di accesso alla memoria

Tecniche per ridurre la Tabella delle Pagine

- A) Paginazione della Tabella delle Pagine
  - Parte della tabella finisce in memoria virtuale anziche' fisica (→ si potra' generare un doppio page-fault !)
  - Staranno in RAM solo le porzioni della tabelle piu' usate…

- B) Tabella delle Pagine Inversa
  - Si applica una funzione di hashing all'indirizzo virtuale cosi' che il numero di entry nella tabella delle pagine e' al piu' pari al numero totale di pagine fisiche

A) Tabella delle Pagine a Due Livelli



Ho 1k tabelle di secondo livello ("foglie") + la tabella "radice"

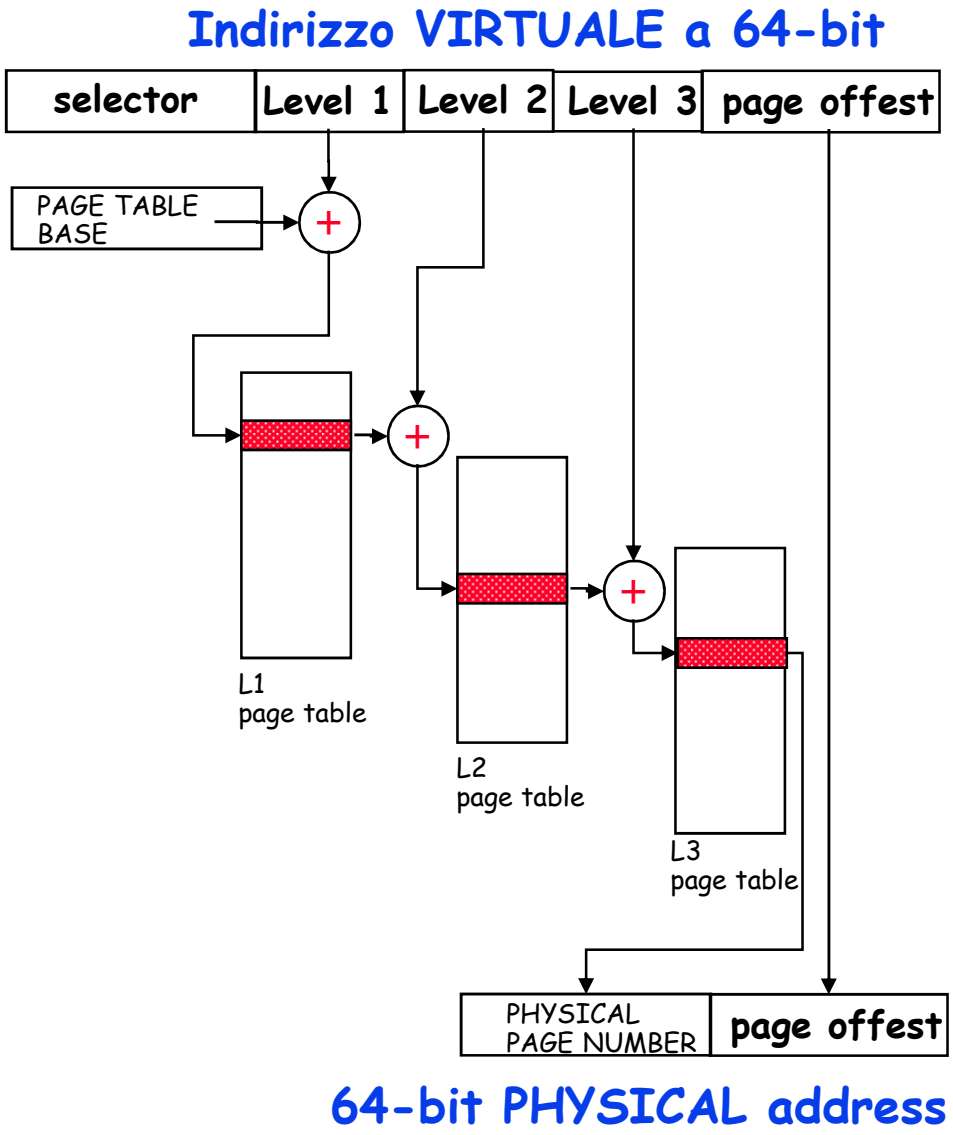
Non tutte le tabelle foglie saranno pero' presenti in memoria

In totale la radice occupa 4KB tutte le foglie 4MB

Ogni processo del sistema ha pero' una proprima tabella completa!

A) Tabella delle Pagine a Tre Livelli (Alpha)

- Avendo indirizzi a 64 bit conviene suddividere in tre livelli
  - Nell'Alpha uso in realta' solo 43 bit per lo spazio virtuale



A) Metodi di accesso Top-Down e Bottom-Up

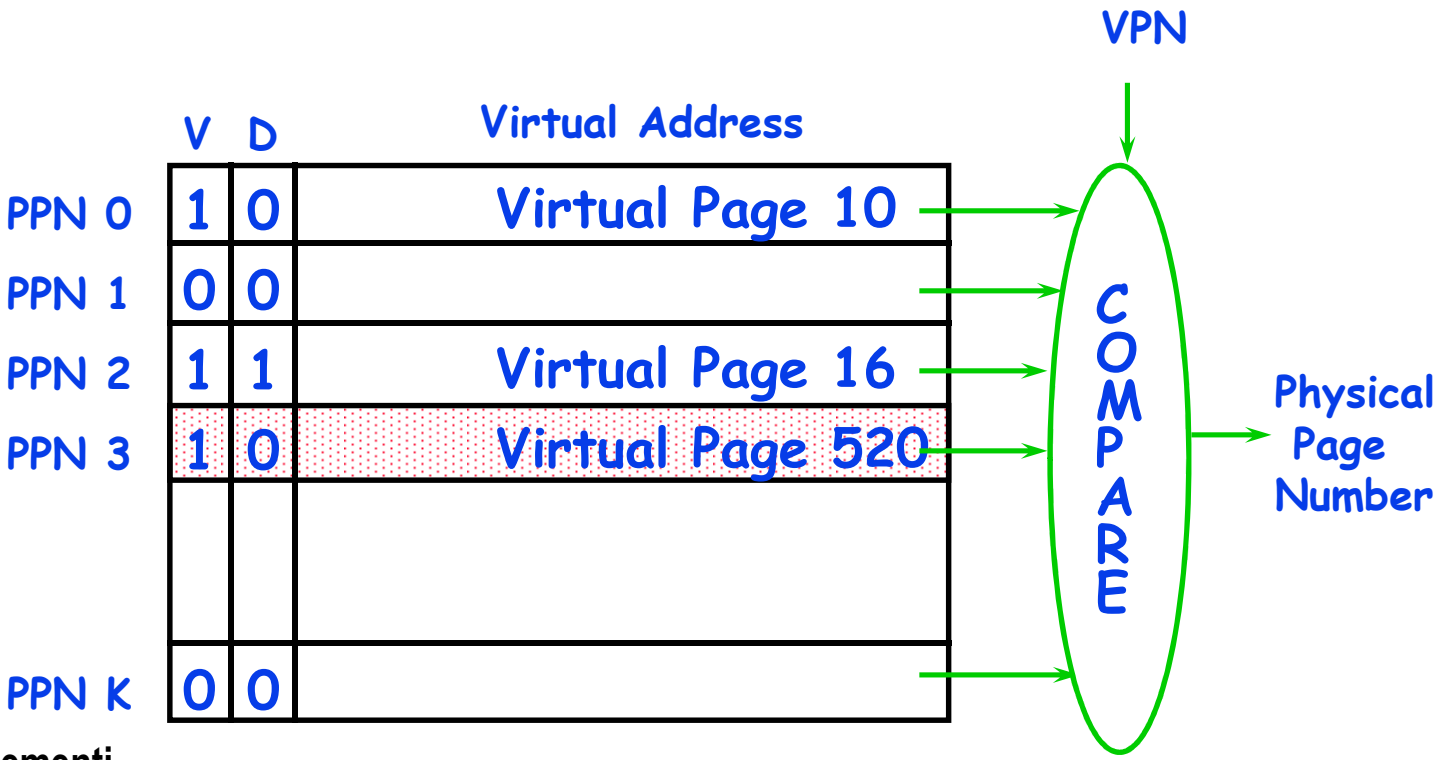
- Top-Down
  - Accesso alla root page table (sempre in memoria)
  - Accesso alla leaf page table
  - Accesso alla pagina fisica
  - Totale 3 accessi alla memoria + eventuale fault o doppio fault
  - In generale ho un numero di accessi alla memoria pari a 1+Num_Livelli
- Bottom-Up
  - Accesso diretto all'insieme delle leaf page table* (supposte contigue in memoria e allineate su boundary di 4MB per semplificare l'accesso)
    - Se l'operazione ha successo: bene (ho solo 2 accessi in memoria)
    - Altrimenti: si procede con un accesso bottom-up completo
  - Se va tutto bene, faccio 2 accessi fissi indipendentemente dal numero di livelli della tabella delle pagine

* affinche' questo funzioni devo avere un metodo per sapere se ci sono leaf al loro posto oppure no

B) Tabella delle Pagine Inversa (1)

- Una entry nella tabella delle pagine per ogni pagina fisica
- Vantaggi
  - La dimensione scala con la dimensione della memoria fisica anziche' con quella della memoria virtuale
  - La dimensione e' fissa a prescindere dal numero di processi !
- Chiamata "inversa" perche' le entry sono indicizzate per PPN anziche' per VPN
  - In ogni caso la ricerca avviene fornendo un VPN e ottenendo un PPN…
  - Il PPN e' dedotto dalla posizione della VPN trovata
- Tecnica
  - Si applica una funzione di hash al VPN
  - L'hash costituisce un indice nella tabella inversa corrispondente al PPN
  - Poiche' diversi VPN possono produrre lo stesso hash e' necessario un meccanismo per risolvere le collisioni (es. lista concatenata memorizzata all'interno della tabella stessa): la nuova entry e' aggiunta in fondo alla lista
  - E' possibile che si perda un po' tempo per percorrere tale lista concatenata

B) Tabella delle Pagine Inversa (2)



Numero di elementi della tabella = K

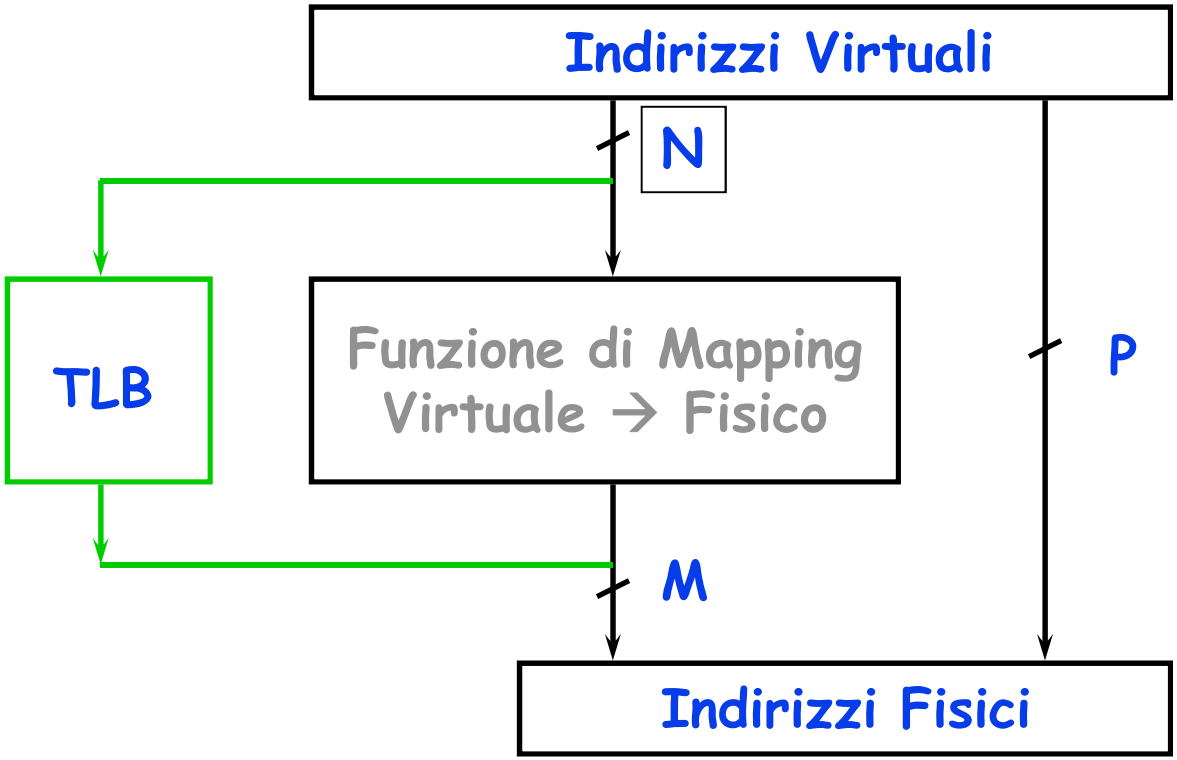
L' IBM "System 38" (AS400) implementa indirizzi a 64-bit

48 bit costituiscono il numero di pagina virtuale

e tali elementi hanno un tag di soli 12-bit

Funziona come una cache fully associative !

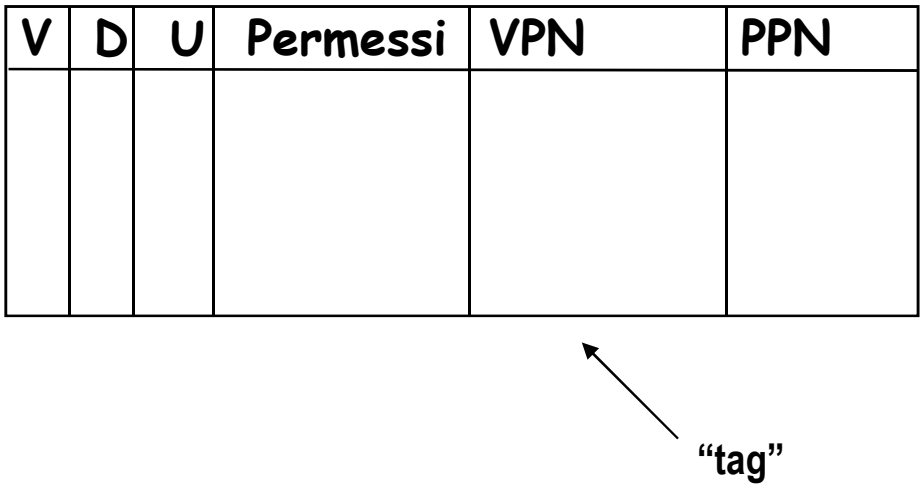
Velocizzare il meccanismo di traduzione



Uso una piccola cache (TLB=Translation Lookaside Buffer) per avere "a portata di mano" un po' di elementi della Tabella delle Pagine

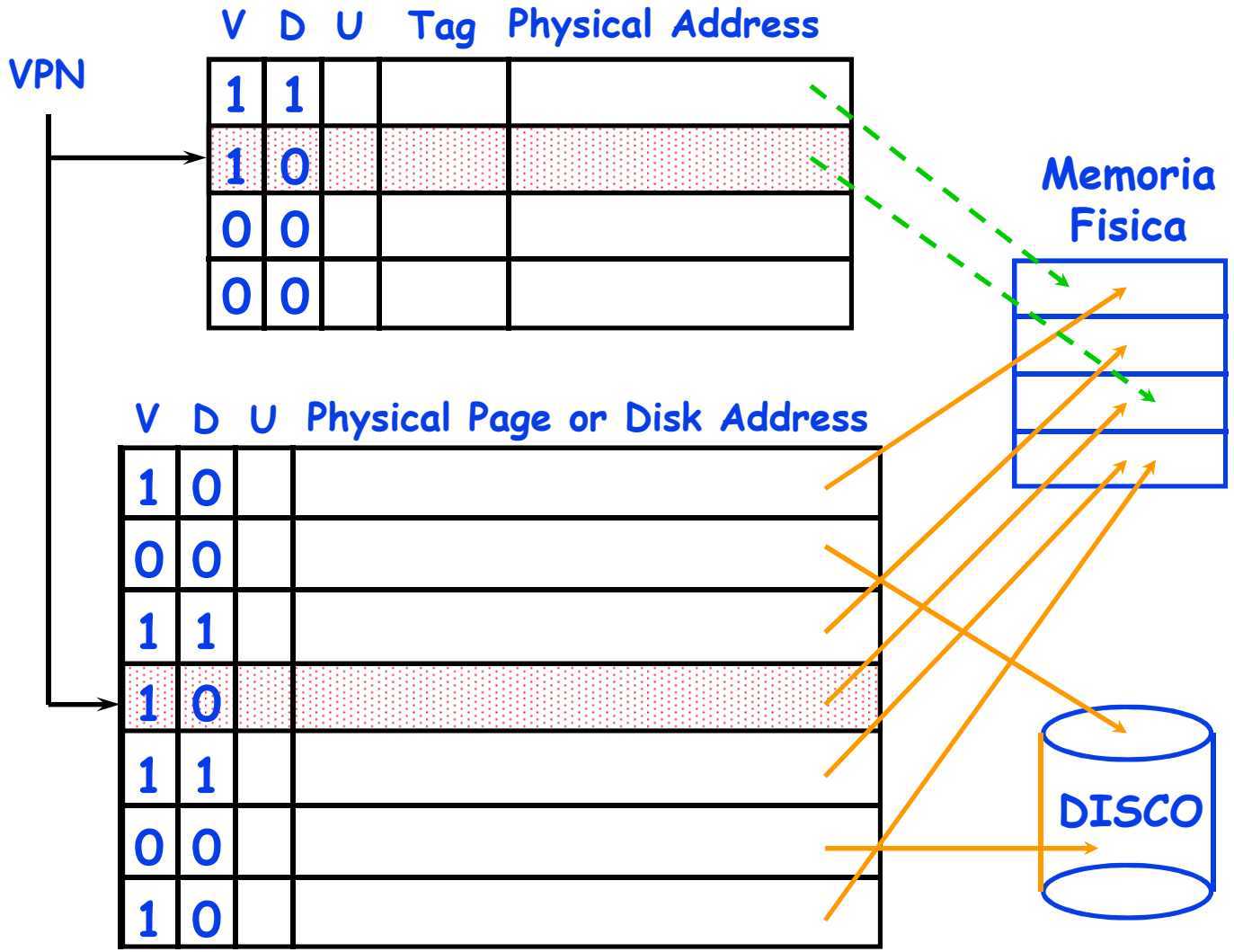
Funziona come una cache L1 per la Tabella delle Pagine (L2) !

TLB: Caratteristiche



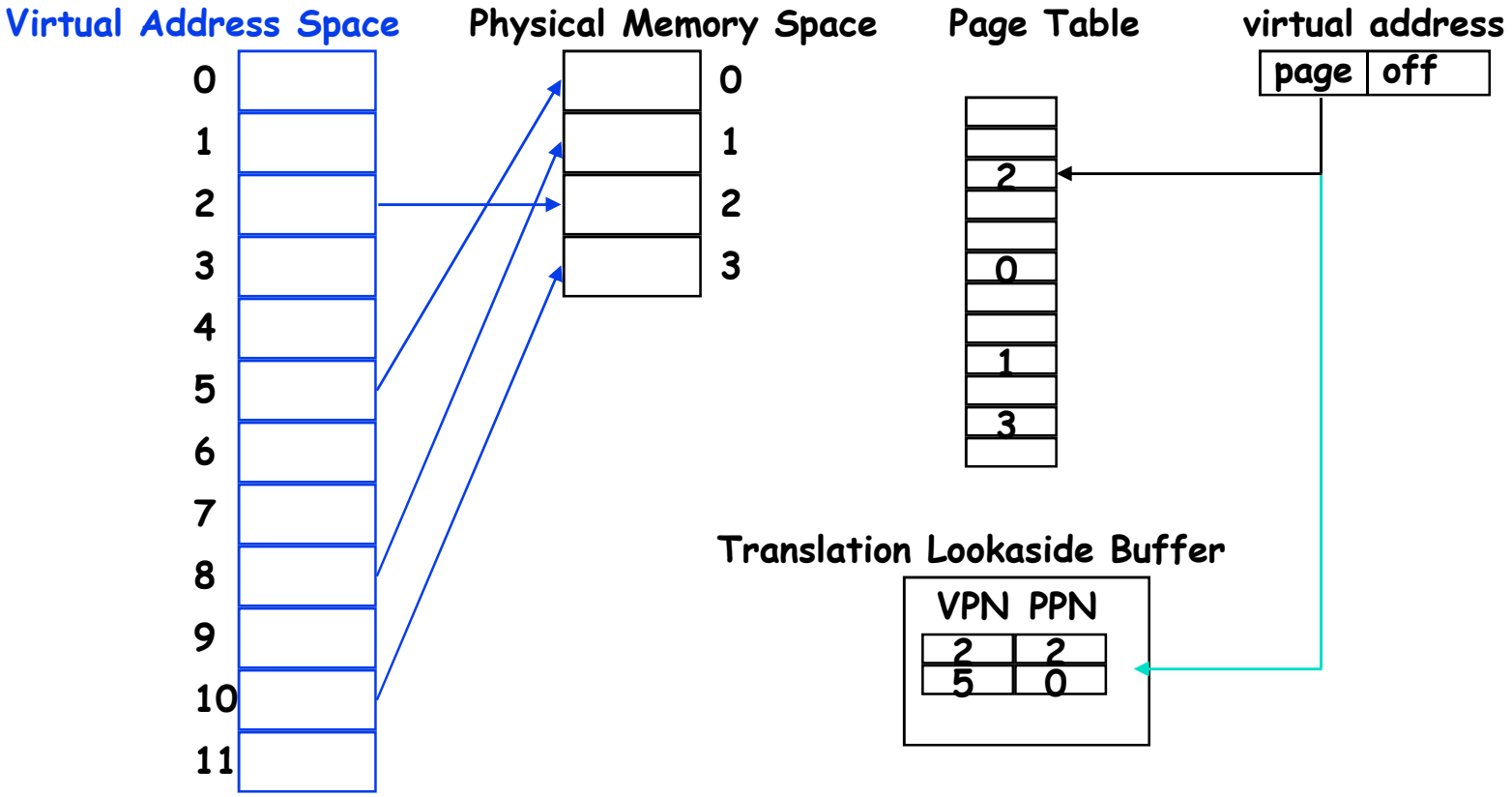
- La TLB e' una cache che puo' essere fully associative, set-associative o ad accesso diretto
- La TLB ha tipicamente non piu' di 128-256 elementi
- Il tempo di accesso alla TLB e' confrontabile col tempo di accesso alla cache (molto piu' basso del tempo di accesso alla memoria principale)

TLB: Organizzazione Logica

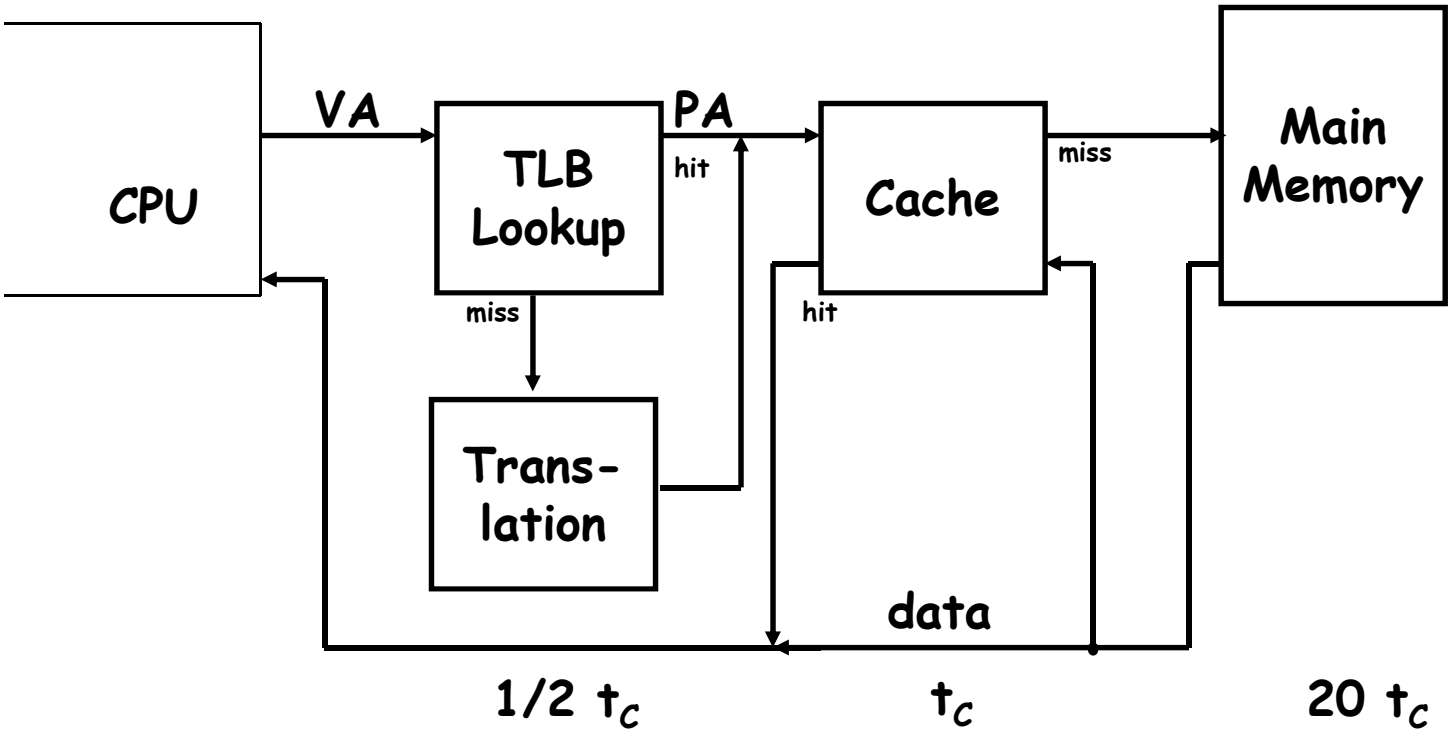


TLB: esempio

- Memoria Virtuale → la memoria agisce da cache per il disco
- La tabella delle pagine mappa indirizzi virtuali in fisici
- La TLB e' una cache delle piu' recenti traduzioni

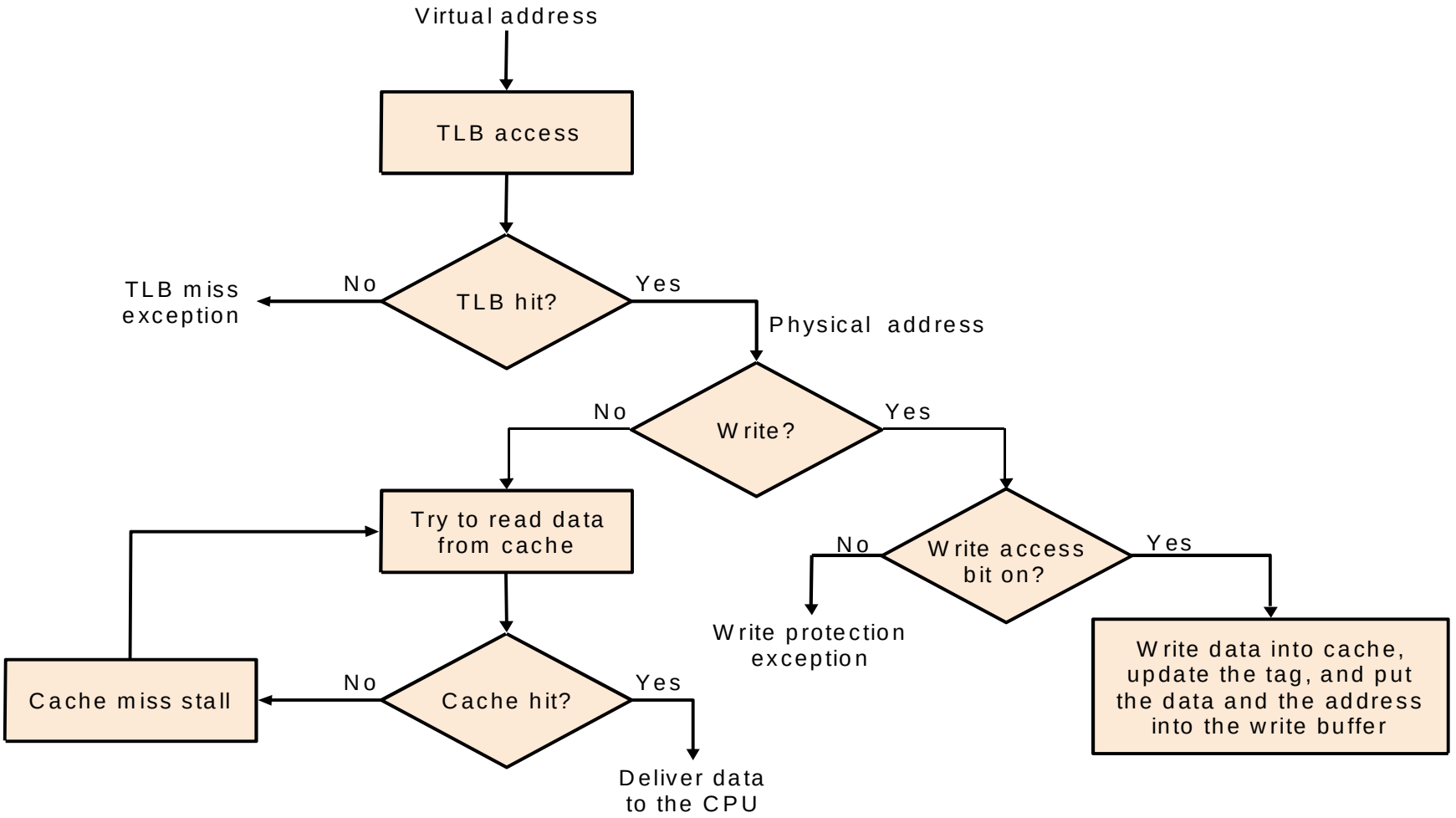


TLB: Schema e interazione con la Cache



VA = Virtual Address
PA= Physical Address

TLB+cache: diagramma di funzionamento



TLB: Valori tipici

Dimensione del blocco: 8 byte

Hit-time: 0.5 - 1 cicli

Miss-penalty: 10-50 cicli

Hit-rate: da 99% a 99.99%

Dimensione della TLB: 32-1024 elementi