Esercitazione su Memoria Virtuale

Gerarchie di Memoria

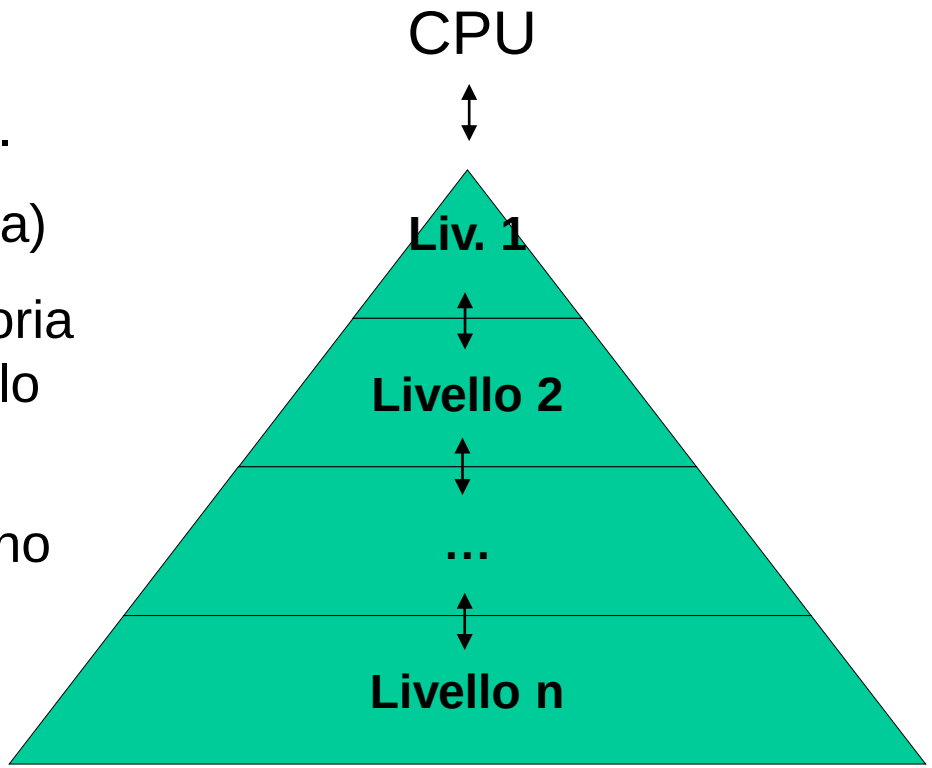
La memoria e organizzata in modo *gerarchico* (a livelli):

Liv. 1: memoria più veloce(costosa).

Liv. n: memoria più lenta (economica)

Il processore indirizza tutta la memoria di *Liv.n*, ma accede direttamente solo al *Liv. 1*.

Per *località* i blocchi cercati si trovano con alta probabilità a livello alto.



Obiettivo: dare l'illusione di avere un memoria veloce come a *Liv. 1* e grande come a *Liv. n*.

Memoria Virtuale

La memoria principale può a sua volta agire come “cache” per la memoria di massa (dischi, nastri).

Si parla in questo caso di memoria virtuale:

- I programmi sono compilati rispetto ad uno *spazio di indirizzamento virtuale*, diverso da quello fisico;
- Il sistema di memoria virtuale *traduce* gli indirizzi virtuali in indirizzi fisici.

Vantaggi:
- illusione di avere una maggiore memoria fisica;
- *rilocazione* dei codici;
- meccanismi di *protezione* (gli spazi virtuali di programmi diversi mappati su spazi fisici differenti).

Memoria Virtuale

I concetti fondamentali relativi alla memoria virtuale sono analoghi a quelli visti per la cache, ma per ragioni storiche la terminologia è diversa.

- Memoria fisica / memoria virtuale

- Pagine (blocchi)

- Page fault (miss) [la pagina non è in memoria e deve essere letta da disco]

Il costo dei *page fault* è estremamente elevato (milioni di cicli), e questo influenza molte scelte (dimensioni della pagina elevate *4, 16, 32, 64 KB*, completa associatività, gestione software, write-back).

Memoria Virtuale: Traduzione degli indirizzi

Ad ogni istante solo una parte delle pagine di memoria “virtuali” è presente nella memoria fisica. L’indirizzo virtuale si suddivide in

Il *Virtual Page Address* indirizza le entry di una tabella, detta *Page Table*, che contiene informazioni sulla presenza / validità delle pagine in memoria

Per ragioni di efficienza non si accede direttamente alla *Page Table*, ma piuttosto al *TLB (translation Lookaside Buffer)* che svolge il ruolo di “cache” della *Page Table* …

Memoria Virtuale: Esercizio 1

Sia dato un sistema di memoria virtuale paginata con pagine di *2 KB* e una cache con blocchi di dati di *16 B*. Considerare il seguente loop che accede un array di interi (ognuno memorizzato su *4 B*)

```
for (i=0; i<2048; i++)
  if (a[i]%2==1)
     a[i]--;
```

Si supponga inoltre che la porzione di memoria virtuale acceduta dal loop non sia inizialmente presente, né in memoria fisica né in cache.

a. Che tipo di località sfrutta il loop?

Spaziale — elementi di memoria vicini vengono riferiti in sequenza (istruzioni ed elementi dell'array);

Temporale — la stessa locazione viene acceduta sia in lettura che in scrittura.

Memoria Virtuale: Esercizio 1 (continua)

Sia dato un sistema di memoria virtuale paginata con pagine di *2 KB* e una cache con blocchi di dati di *16 B*. Considerare il seguente loop che accede ad un array di interi (ognuno memorizzato su *4 B*)

```
for (i=0; i<2048; i++)
   if  (a[i]%2)
            a[i]--;
```

Si supponga inoltre che la porzione di memoria virtuale acceduta dal loop non sia inizialmente presente, né in memoria fisica né in cache.

b. Indicare il numero di page fault (certi), sia quando l'indirizzo virtuale *&a[0]* è un multiplo della dimensione della pagina, sia quando non lo è.

2048 valori interi → *2048* • *4B* = *8 KB* richiesti per memorizzare l'array.

Se &a[0] è un multiplo della dimensione della pagina, si avranno *8 KB / 2 KB = 4* page fault. Altrimenti si avranno *5* page fault.

Memoria Virtuale: Esercizio 1 (continua)

Sia dato un sistema di memoria virtuale paginata, con pagine di *2 KB* e una cache con blocchi di dati di *16 B*. Considerare il seguente loop che accede ad un array di interi (ognuno memorizzato su *4 B*)

```
for (i=0; i<2048; i++)
   if  (a[i]%2)
            a[i]--;
```

Si supponga inoltre che la porzione di memoria virtuale acceduta dal loop non sia inizialmente presente, né in memoria fisica né in cache.

c. Indicare il numero di cache miss (certi), sia quando l'indirizzo fisico corrispondente a *&a[0*] è un multiplo della dimensione del blocco di cache, sia quando non lo è.

Se l'indirizzo corrispondente a *&a[0]* è multiplo della dimensione del blocco si avranno $8\ KB / 16\ B = 2^{13}\ B / 2^4\ B = 2^9 = 512$ cache miss.

Altrimenti si avranno *513* cache miss.

Memoria Virtuale: Esercizio 1 (continua)

Sia dato un sistema di memoria virtuale paginata, con pagine di *2 KB* e una cache con blocchi di dati di *16 B*. Considerare il seguente loop che accede ad un array di interi (ognuno memorizzato su *4 B*)

```
for (i=0; i<2048; i++)
   if  (a[i]%2)
            a[i]--;
```

Si supponga inoltre che la porzione di memoria virtuale acceduta dal loop non sia inizialmente presente, né in memoria fisica né in cache.

d. Perché non è stato necessario dare informazioni sull'organizzazione della cache, o sulla politica di rimpiazzamento delle pagine della memoria virtuale?

Perché abbiamo considerato solo *Compulsory Miss*, che dipendono esclusivamente dalle dimensioni (dell'array, blocchi e pagine) e non dalle specifiche organizzazioni dei livelli di memoria.

Memoria Virtuale: Esercizio 2

Si consideri un sistema di memoria virtuale, la cui *Page Table* (con *valid* e *dirty* bit) ha dimensione *8MB.* La dimensione della pagine è *1024 B* e l'indirizzo fisico è di *24 bit*. Qual è la dimensione dell'*indirizzo virtuale*?

La dimensione di ogni pagina è $1024B = 2^{10}\ B$, quindi sono richiesti *10 bit* per indirizzare ciascun byte di ogni pagina (*10 bit* OFFSET).

Pertanto i *24 bit* dell'indirizzo fisico, includono *10 bit* per l'OFFSET e *14 bit* per il NUMERO della pagina fisica.

Quindi ogni ingresso della *Page Table* è composto di *2 bit* per *valid* e *dirty*, e *14 bit* per indirizzare le pagine fisiche → *16 bit* in totale (*2 B*)

Il numero totale di ingressi della *Page Table* è $8\ MB / 2B = 2^{23} / 2 = 2^{22}$, indirizzabili con *22 bit*. La dimensione dell'indirizzo virtuale è quindi:

22 bit + *10 bit* (OFFSET) = *32 bit*

Memoria Virtuale e TLB

In un sistema di *memoria virtuale* ogni accesso alla memoria richiede due accessi: prima alla Page Table e quindi alla memoria fisica.

Per ridurre questo overhead si sfrutta un meccanismo detto *TLB (Translation Lookaside Buffer),* una sorta di cache della Page Table.

La TLB contiene copia delle entry della Page Table riferite recentemente, e, come la cache, può sfruttare l'associatività per ridurre i conflitti.

L'indirizzo virtuale viene scomposto in virtual page address e offset (nella pagina).
Il virtual page address viene utilizzato per accedere alla TLB come accadeva per la cache ovvero …

… se la TLB contiene 2^K set, i K bit meno significativi del virtual page address sono utilizzati per indirizzare un insieme della TLB, i bit rimanenti sono utilizzati come *tag* nelle entry della TLB.

Memoria Virtuale e TLB: Esercizio 3

Si consideri un sistema di memoria virtuale paginata, con *pagine* di *1 KB* e *indirizzo virtuale* di *32 bit*. Si consideri una *TLB 2-way-associative* di *512 B*, in cui ciascun elemento consta di *4 B* suddivisi in: *2 bit* per *valid* e *dirty*, *TAG* e corrispondente numero di pagina fisica. Calcolare la dimensione dell'*indirizzo fisico*.

Offset (o displacement) di pagina: la pagina ha dimensione *1 KB* = 2^{10} *B* e quindi servono *10 bit* per l'offset.

I rimanenti *22 bit* dell'indirizzo virtuale servono per indirizzare gli elementi della *Page Table (NUMERO di pagina virtuale)*.

22 bit per indirizzare la Page Table	**10 bit OFFSET**

Memoria Virtuale e TLB: Esercizio 3 (cont.)

Si consideri un sistema di memoria virtuale paginata, con *pagine* di *1 KB* e *indirizzo virtuale* di *32 bit*. Si consideri una *TLB 2-way-associative* di *512 B*, in cui ciascun elemento consta di *4 B* suddivisi in: *2 bit* per *valid* e *dirty*, *TAG* e corrispondente numero di pagina fisica. Calcolare la dimensione dell'*indirizzo fisico*.

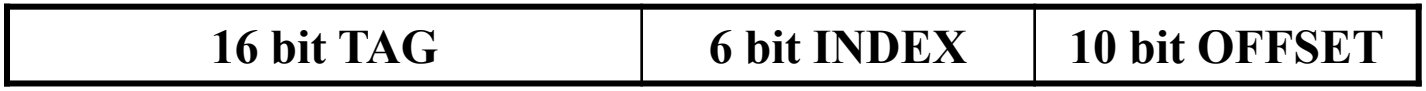
Vediamo come l'indirizzo virtuale viene scomposto nell'accesso alla *TLB*:

tot. ingressi TLB = 512 B / 4 B = 128

essendo la *TLB 2-way associative*

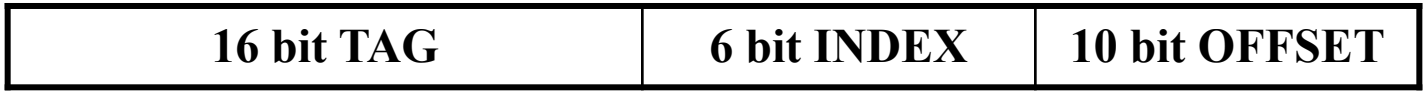
tot. insiemi TLB = 128 / 2 = 64 = 2^6

Quindi:



Memoria Virtuale e TLB: Esercizio 3 (cont.)

Si consideri un sistema di memoria virtuale, con *pagine* di *1 KB* e *indirizzo virtuale* di *32 bit*. Si consideri una *TLB 2-way-associative* di *512 B*, in cui ciascun elemento consta di *4 B* suddivisi in: *2 bit* per *valid* e *dirty*, *TAG* e corrispondente numero di pagina fisica. Calcolare la dimensione dell'*indirizzo fisico*.



Per calcolare la dimensione dell'indirizzo fisico, ricordiamo che ogni entry della *TLB* è del tipo:

Dato che ogni entry comprende *32 bit*, il numero della pagina fisica occupa *32 - #(valid, dirty) - # TAG = 32 – 2 – 16 = 14*. Quindi

 # Ind. Fisico = #(num. Pagina) + #OFFSET = 14 + 10 = 24

Memoria Virtuale e TLB: Esercizio 4

Si consideri un sistema di memoria virtuale, con

- indirizzi virtuali di *40 b*
- pagine di *16 KB*
- indirizzi fisici di *26 b*

Calcolare la dimensione della tabella delle pagine, assumendo che ogni entry includa *4 bit* per valid, dirty, protezione e uso.

L'indirizzo virtuale si suddivide nel modo seguente:

26 bit per indirizzare la Page Table	14 bit OFFSET

quindi il numero di entry della Page Table (numero di pagine virtuali) è 2^{26}.
Ogni entry della Page Table ha la forma seguente:

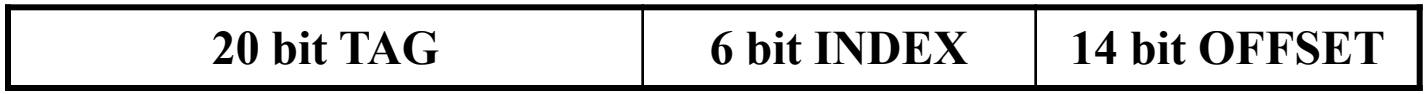
V-D-P-U (4 bit)	Indirizzo Pagina fisica (12 bit)

Quindi complessivamente la Page Table occuperà uno spazio di $2^{26} \bullet 16$ *bit,* ovvero 2^{27} *B = 128 MB.*

Memoria Virtuale e TLB: Esercizio 4 (cont.)

Si supponga che il sistema di memoria virtuale precedente (indirizzi virtuali di *40 b,* pagine di *16 KB,* indirizzi fisici di *26 b)* sia implementato con una TLB 4-way associative di *256* elementi. Si indichi dimensione e composizione delle entry della TLB e dimensione della TLB stessa.

L'indirizzo virtuale è ora utilizzato per accedere alla TLB. I *256* elementi sono suddivisi in *64* set di quattro elementi, che richiedono *6 bit* per essere indirizzati. Dunque:



quindi le entry della TLB conterranno:

Quindi la dimensione complessiva della TLB risulta essere *256 • 36 bit,* ovvero $2^8 * 4 * 9 = 2^{10} * 9 = 9$ *Kb.*

Memoria Virtuale e TLB: Esercizio 4 (cont.)

Si supponga che il sistema di memoria precedente (indirizzi virtuali di *40 b,* pagine di *16 KB,* indirizzi fisici di *26 b)* abbia una cache di tipo set-associative a due vie con blocchi di 8 B e una dimensione totale di *4 KB* (dati). Spiegare le modalità di accesso alla memoria (struttura e dimensione delle varie parti degli indirizzi).

L'indirizzo virtuale viene utilizzato per accedere alla TLB e, nel caso di TLB-miss, alla Page Table. Una volta risolti i miss verrà generato un indirizzo in memoria fisica.

L'indirizzo fisico viene quindi utilizzato per accedere alla cache. Il numero di blocchi della cache è

$$dim.\ tot. / dim.\ blocco = 4 * 2^{10} / 8 = 2^9$$

e quindi si avranno $2^9 / 2 = 2^8$ set costituiti da due blocchi.

Pertanto l'indirizzo fisico è strutturato nel modo seguente (#tag = IND_FIS - OFFSET – INDEX = 26 – 3 – 8 = 15):

15 bit TAG	8 bit INDEX	3 bit OFFSET