

Considerazioni storiche (1/2)

Agli albori dell'informatica la memoria centrale di un computer raramente superava le 4k parole ed i sistemi operativi erano, a dir poco, essenziali.

A quell'epoca, chiunque si disponesse a scrivere un programma doveva prima informarsi di quale fosse la disponibilità di spazio nella memoria centrale del sistema da adoperare, al netto delle necessità della CPU (es. catasta operativa) e della "parte residente" del sistema operativo. Fatto il conteggio doveva scrivere un programma fatto di pezzi da mandare in esecuzione uno per volta, registrando i risultati parziali in gruppi di memoria i cui indirizzi fossero noti anche ai successivi pezzi di programma. Questi li prelevavano per usarli ed, a loro volta registravano i risultati in locazioni prefissate. Senza questa procedura il programma non poteva essere eseguito.

Considerazioni storiche (2/2)

Tutto questo è oggi completamente ignoto agli utilizzatori di computer.

Il fatto che talvolta siano indicati dei requisiti minimi del sistema per usare un certo programma, è un semplice suggerimento per ottenere dal programma un comportamento ragionevole, senza lunghissimi tempi di attesa tra due operazioni successive.

Oggi è assolutamente normale che programmi che tra occupazione diretta ed indiretta avrebbero bisogno di 25-30 Mbyte girino su macchine con memoria centrale da 16 Mbyte, grazie alla tecnica della "*memoria virtuale*", inventata dall'IBM negli anni '70.

La tecnica si avvale di un certo numero di supporti hardware forniti dal processore e ciò spiega perché essa è apparsa nei sistemi basati su microprocessori "single chip" relativamente di recente.

La tecnica della “*memoria virtuale*”

In estrema sintesi la tecnica prevede che **il sistema operativo assegni gli spazi disponibili nella memoria centrale ai segmenti di un programma su disco, che sia troppo grande per essere copiato interamente, secondo le esigenze dell’elaborazione in corso.**

La tecnica di gestione degli spazi ha molti punti in comune con il funzionamento di una memoria cache. Anche in questo caso, infatti, se gli spazi nella memoria principale sono saturati, e il programma prevede un accesso ad una locazione di memoria **nel campo di indirizzamento del processore**, che non corrisponde ad **un indirizzo esistente nella memoria centrale**, perché il relativo segmento non vi ha trovato posto, si deve ricorrere ad un *algoritmo di sostituzione*.

La parte di programma contenente l’indirizzo richiesto viene prelevato dal disco (trasferimento in DMA) e sostituita in memoria ad una delle parti che vi risiedevano.

Poiché il tempo di accesso al disco costituisce una penalità di fallimento molto più pesante, in termini di tempo rispetto a quella di una cache, l’entità minima da trasferire è presa moto più grande del blocco della cache ed è una *pagina di memoria* di solito compresa tra 2k e 16k.

La Memory Management Unit (MMU)

Quello a cui fa riferimento il processore nell’elaborazione è detto indirizzo “*virtuale*” ed ha come unica limitazione di essere entro il campo di indirizzamento massimo del processore.

Il compito di tradurre questo indirizzo in un indirizzo che punti ad una locazione effettivamente esistente nel sistema (*indirizzo fisico*) è demandato ad una struttura hardware che si chiama **Memory Management Unit (MMU)** e che opera sotto il controllo software del sistema operativo.

E’ opportuno, a questo punto aprire una parentesi per ricordare le condizioni al contorno in cui la MMU opera.

I compiti della MMU

Abbiamo visto, parlando dei dischi magnetici rigidi che i formati d'indirizzo forniscono una **corrispondenza biunivoca tra le informazioni (istruzioni o dati) su disco e le locazioni della memoria principale**. Il campo indirizzo su disco è, di solito, molto superiore, sia a quello coperto dalla memoria centrale, che al campo di indirizzamento del processore. Viceversa, nessun programma (in formato eseguibile) può, per definizione, eccedere i limiti dell'indirizzamento "*virtuale*" del processore e, quindi, **su disco possono essere ospitati molti programmi e tocca al sistema operativo gestire il trasferimento dei programmi nella memoria principale per l'esecuzione**.

La trascrizione di un programma dal disco nella memoria principale non è un'operazione "trasparente", anche se le sue dimensioni sono inferiori allo spazio "fisico" di memoria disponibile, perché il suo posizionamento nel campo degli indirizzi del processore può comunque essere incompatibile con l'occupazione corrente della memoria principale. Questo significa che **il lavoro di trasposizione di indirizzi della MMU è indispensabile anche quando non ci sono problemi di spazio di memoria**.

Il processo di ricerca in memoria

La MMU, quando riceve un *indirizzo virtuale* lo converte in un *indirizzo fisico*, con un procedimento che analizzeremo in dettaglio più avanti e poi lo invia alla cache, dando inizio ad una prima ricerca della locazione.

- **Se la locazione fa parte di un blocco che risiede nella cache**, il contenuto viene inviato al processore ed il discorso si chiude.
- **In caso di insuccesso** si passa alla ricerca nella memoria principale dove, **se il programma è interamente residente, l'indirizzo è sicuramente contenuto**.
- **Nel caso in cui il programma non ha potuto essere trascritto interamente**, la locazione cercata potrebbe non esser presente ed, allora, va cercato su disco, dove deve essere necessariamente presente. In questo caso si trasferisce la pagina a cui appartiene l'indirizzo, in DMA, nella memoria, eventualmente al posto di una delle pagine precedentemente presenti, individuata attraverso un algoritmo di sostituzione.

Cache, memoria principale e disco, tre elementi di un'unica struttura

Il meccanismo che abbiamo delineato per sommi capi,
è alla base dell'affermazione che

- **un sistema ben configurato si comporta operativamente come se avesse una memoria con capacità corrispondente a quella dei suoi dischi rigidi, ma con un tempo d'accesso dell'ordine di quello della sua cache.**

La tabella dei numeri di pagina (1/3)

In pratica la "*tabella dei numeri di pagina*" non è altro che una parte della memoria centrale formata da tante locazioni quante sono le pagine "*virtuali*". Ciascuna locazione deve avere un numero di bit sufficiente a contenere il numero della pagina *fisica* corrispondente, più altri **bit di controllo** di cui parleremo più avanti.

Questa **struttura dati** residente in memoria consente di associare ad un numero elevato di pagine virtuali un numero ridotto di pagine fisiche.

La gestione della tabella è semplice perché per ogni locazione si utilizza almeno un bit in più rispetto a quelli necessari per contenere il numero delle pagine fisiche (in figura sono 18). Questo bit, di solito posto nella posizione più alta (30 nel caso specifico) è ad "1" solo quando la locazione contiene un numero di pagina fisica effettivamente presente in memoria. Esso è stato posto ad "1" quando la pagina è stata trasferita nella memoria centrale dal disco rigido.

In figura la locazione corrispondente al numero di pagina virtuale indirizzato è in grigio e contiene un numero di pagina fisica valido (il relativo bit di validità è in grigio più scuro) il contenuto della locazione viene trasferito nella parte più significativa del registro di indirizzo, mentre i 12 bit meno significativi vengono presi direttamente dall'indirizzo virtuale.

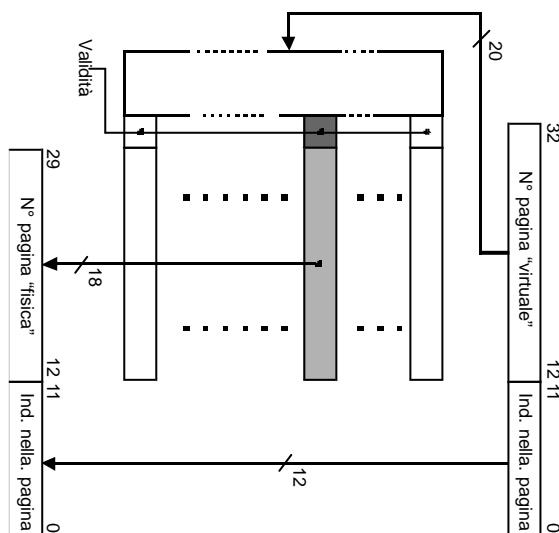
La tabella dei numeri di pagina (2/3)

In realtà il *bit di validità* non è l'unico bit di controllo inserito nelle locazioni della tabella delle pagine, perché anche qui occorre il *bit di modifica* che indica che la pagina è stata modificata durante la sua permanenza in memoria e deve essere ricopiata su disco prima di essere sovrascritta (*copy back*).

Da notare che la tabella da luogo, essa stessa, ad una occupazione di memoria non irrilevante perché nel caso numerico ipotizzato richiede ben 1024 K locazioni da almeno 20 bit che debbono essere necessariamente arrotondati a 32 (4 byte) e ciò da una occupazione di 4 Mbytes.

Se, però si fosse fatta un'ipotesi più realistica sull'indirizzo fisico disponibile, per esempio 64 Mbyte (24 bit d'indirizzamento) il numero di pagina fisica sarebbe stato di soli 12 bit e l'occupazione della tabella si sarebbe dimezzato.

La tabella dei numeri di pagina (3/3)



Il registro base della tabella delle pagine

La tabella dei numeri di pagina della figura precedente per poter funzionare senza altre aggiunte dovrebbe essere posizionata a partire dalla locazione iniziale della memoria centrale, ma questo sarebbe un pesante condizionamento per il sistema, non solo, ma impedirebbe di avere più di una tabella dei numeri di pagina, presente in memoria.

Il problema è facilmente superabile introducendo un registro detto, **registro base della tabella delle pagine** di memoria, che consente di sistemare la tabella dovunque in memoria esista uno spazio sufficiente a contenerla.

L'algoritmo di indirizzamento completo è, allora, il seguente: Il numero di pagina virtuale viene prima sommato al contenuto del **registro base della tabella delle pagine** e poi usato per l'accesso in memoria per la ricerca della corrispondente pagina fisica. Il numero di bit di questo registro deve essere sufficiente a posizionare la tabella dei numeri di pagina in qualsiasi zona della memoria centrale di cui il sistema dispone.

Ciò consente anche di avere **varie tabelle delle pagine, una per ogni programma in esecuzione e da la possibilità di passare dall'una all'altra attraverso l'aggiornamento del registro.**

Indirizzo delle pagine su disco

La conversione degli indirizzi virtuali in indirizzi fisici è solo uno dei problemi da risolvere, perché bisogna, anche, **garantire la corrispondenza tra le pagine virtuali ed il loro posizionamento sul disco** o sui dischi del sistema.

Questi, oltretutto, hanno un campo di indirizzamento più ampio, che richiede un maggior numero di bit.

Una possibile soluzione è quella di ospitare **l'indirizzo su disco della pagina nella stessa tabella dei numeri di pagina**, estendendo adeguatamente il numero di bit per locazione.

In questo modo se il bit di validità della locazione è ad "1" il contenuto è il numero di pagina fisico, se è a "0" il contenuto della locazione è l'indirizzo su disco. In tal modo quando la pagina cercata non si trova in memoria, invece dell'indirizzo fisico può essere letto l'indirizzo dove prelevarla su disco.

Considerazioni aggiuntive sulla MMU

La *tabella dei numeri di pagina*, come abbiamo visto può essere molto grande e non trova, evidentemente, spazio nel chip del processore, dove invece necessariamente è situata la MMU, con la cache primaria.

La *Memory Management Unit* rappresenta la parte hardware del sistema di memoria virtuale che risiede nel chip del processore e che sovrintende ai collegamenti tra questo e la memoria. Ma, poiché la *tabella dei numeri di pagina* risiede nella memoria centrale, la traduzione dell'indirizzo virtuale in indirizzo fisico, necessaria anche per la ricerca di dati od istruzioni nella cache primaria, comporterebbe sempre e comunque un collegamento con un altro dispositivo esterno al chip.

Ciò vanificherebbe tutti i vantaggi della cache.

Come si esce da questo apparente circolo vizioso?

Si utilizza una tecnica che conosciamo già, quella di una struttura intermedia che svolge nei confronti della tabella delle pagine lo stesso ruolo che la cache svolge nei confronti della memoria principale.

Il “Traslation Lookaside Buffer”, TLB

Una piccola parte della tabella detta *buffer per la traduzione degli indirizzi* (**Tranlation Lookaside Buffer, TLB**), è contenuta nello stesso chip del processore ed ospita le corrispondenze tra un certo numero (piccolo) di pagine virtuali e le relative pagine fisiche che sono state utilizzate più di recente.

La ricerca del numero di pagina virtuale nel TLB viene, di solito fatta con la tecnica associativa, solo in qualche caso è usata quella set-associativa.

La parte più significativa dell'indirizzo virtuale (tolto lo spiazzamento) costituisce l'etichetta della CAM, il contenuto del registro è l'indirizzo della corrispondente pagina fisica, con aggiunti in posizione opportuna un certo numero di bit di controllo.

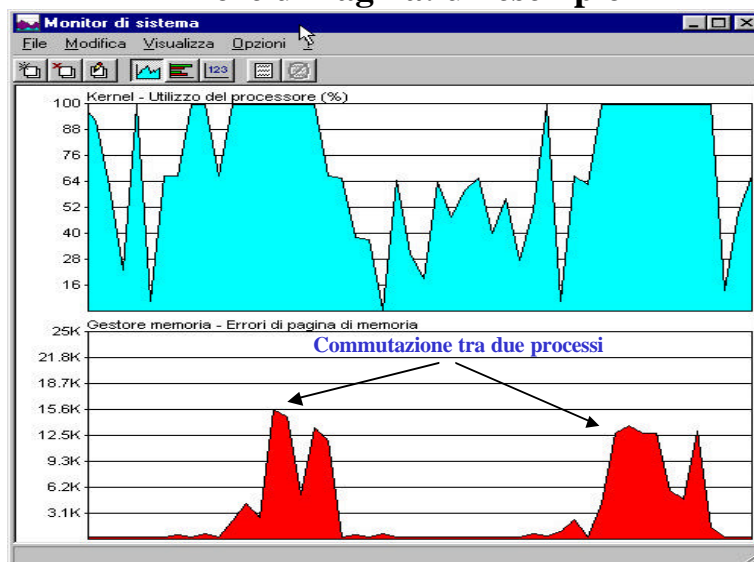
Terminologia della TLB

Per evitare confusione tra memoria cache e TLB si usa una terminologia diversa per dire che la ricerca della pagina fisica corrispondente alla pagina virtuale cui appartiene l'indirizzo chiesto dal processore è stata infruttuosa.

Non si parla di *'miss'* ma di *"page fault"*, ovvero, *errore di pagina*, ma le conseguenze sono le stesse, bisogna caricare l'intera pagina da disco in memoria, aggiornare la tabella delle pagine ed il TLB.

Tutto ciò viene gestito dal sistema operativo, che se sta gestendo anche un altro processo, potrebbe decidere di riempire il tempo non piccolo richiesto per l'accesso al disco per far procedere l'altra esecuzione in corso.

Errore di Pagina: un esempio



Considerazioni finali

La MMU, in conclusione, è la struttura logica hardware che dall'interno del chip del processore gestisce la traduzione degli indirizzi virtuali in indirizzi fisici, utilizzando da una parte il TLB che è anch'esso nel chip e dall'altra la tavola dei numeri di pagina che è nella memoria centrale.

Abbiamo già detto che la gestione della *memoria virtuale* ha molti meccanismi in comune con quella delle *cache memory*, in particolare sono applicabili sia le considerazioni sugli *algoritmi di sostituzione* sia quelle sul modo di procedere in caso di "errore di pagina", per non ritardare la ripresa dell'elaborazione.

A questo proposito è bene sottolineare che nel TLB la *pagina fisica corrispondente ad una certa pagina virtuale* è accompagnato dai **bit di controllo**, tra cui quello di *validità* che, ovviamente, quando un numero approda nel buffer viene posto ad "1".