Tabelle multilivello

G. Lettieri

4 Maggio 2018

1 MMU_1: tabella su 4 livelli

Proviamo a calcolare quanto sono grandi le tabelle di corrispondenza usate dalla Super-MMU. La memoria virtuale è di 2^{48} byte e ogni pagina è grande 2^{12} byte, quindi la memoria virtuale contiene

$$\frac{2^{48}}{2^{12}} = 2^{36} = 64\,\text{Gi pagine.}$$

La tabella di corrispondenza di ogni processo deve avere una entrata per ognuna di queste pagine. Ogni entrata deve contenere almeno i bit P, R/W, U/S e il numero di frame che fornisce i bit da 12 a 51 dell'indirizzo fisico, per un totale di 43 bit, arrotondati in 6 byte. Se poi vogliamo che la dimensione di ogni entrata sia una potenza di 2, dovremo usare almeno 8 byte. In conclusione, la tabella di corrispondenza dovrà essere di

$$64\,\text{Gi} \times 8\,\text{B} = 512\,\text{GiB}.$$

Difficilmente, quindi, possiamo pensare di avere un dispositivo di memoria che possa contenere anche una sola di queste tabelle.

Per affrontare il problema notiamo che la stragrande maggioranza dei programmi ha bisogno soltanto di una piccola frazione dei 2^{48} byte disponibili di memoria virtuale. Vorremmo avere una tabella che contiene le sole entrate effettivamente utilizzate. L'idea è di spezzare la tabella in parti più piccole ed evitare di creare le parti di tabella che non verranno mai richieste.

Introduciamo quindi MMU_1, una MMU del tutto identica alla Super-MMU, tranne che per il formato della tabella di corrispondenza. Inoltre, MMU_1 possiede una memoria in cui salvare le tabelle di corrispondenza e un registro, `cr3`, che serve ad individuare la tabella attiva ad ogni istante. MMU_1 intercetta tutte le operazioni di lettura e scrittura in memoria iniziate dalla CPU e tenta di tradurre l'indirizzo virtuale in fisico usando la tabella attiva. Questa volta, però, può non solo accadere che non sia presente traduzione corrispondente, ma anche che manchi proprio la parte di tabella a cui accedere. Per realizzare questa idea, ogni parte di tabella dovrà avere un suo bit P che dica alla MMU_1, se la parte di tabella è presente o assente. L'idea è di avere una ulteriore tabella,

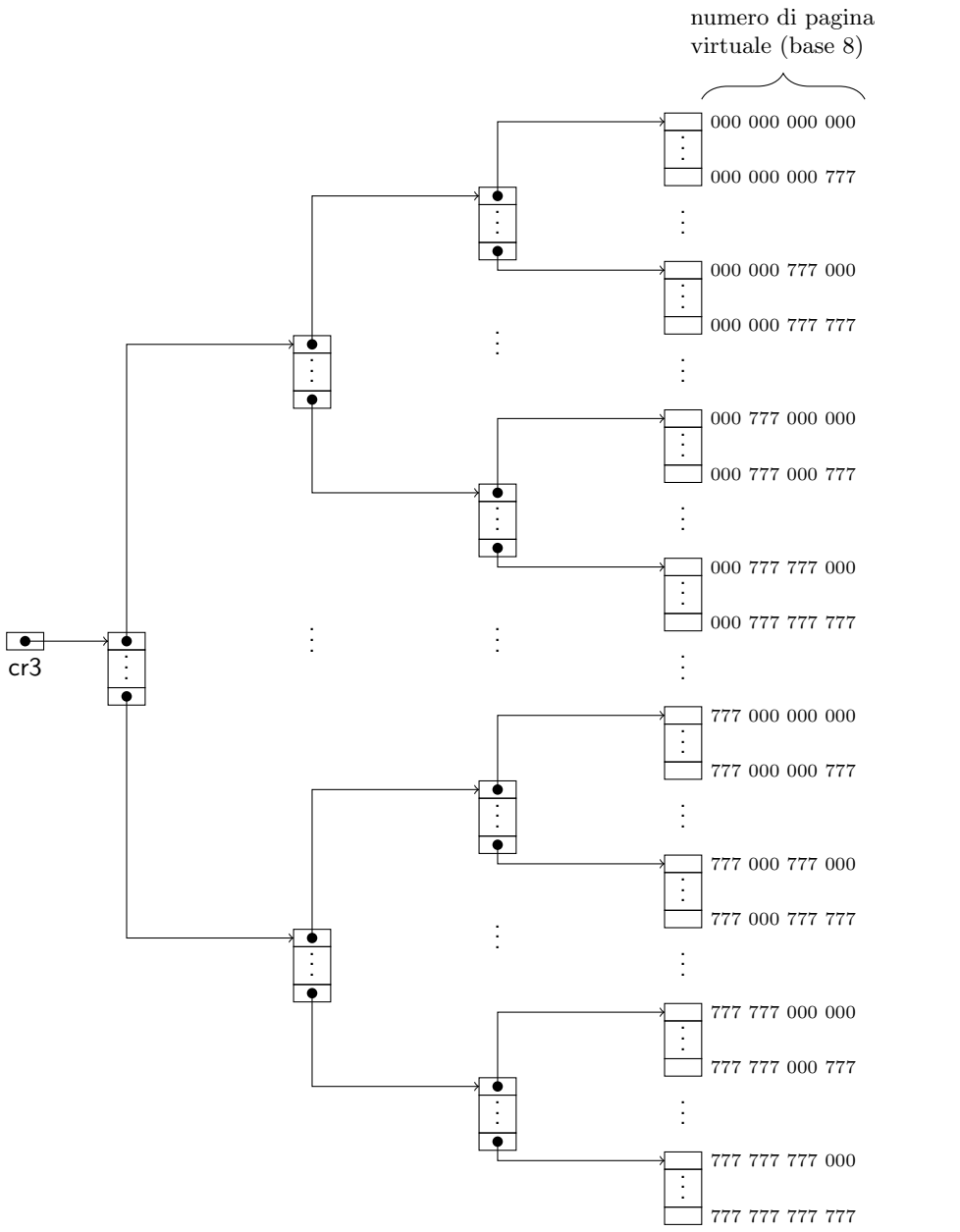

Figura 1: Tabella di corrispondenza su 4 livelli. La figura mostra quali entrate delle tabelle di livello 1 contengono la traduzione dei numeri di pagina virtuale mostrati sulla destra.

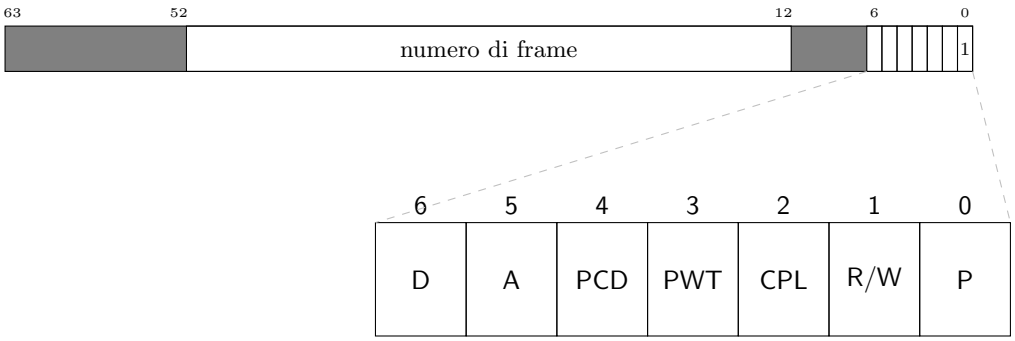

Figura 2: Descrittore di pagina virtuale (tabelle di livello 1).

da consultare per prima, con una entrata dedicata ad ognuna delle parti in cui abbiamo scomposto la tabella originaria.

Per semplicità di gestione, conviene spezzare la tabella di corrispondenza in parti di dimensione uguale a quella delle pagine virtuali e fisiche, che stiamo assumendo di 4 KiB. Chiamiamo ciascuna di queste parti "tabelle di livello 1". Ogni tabella di livello 1 contiene 512 entrate e si occupa quindi della traduzione di 512 pagine virtuali. Dal momento che non tutte le tabelle di livello 1 saranno presenti, la tabella aggiuntiva, oltre ad avere un bit P per ogni tabella di livello 1, dovrà anche fornire l'indirizzo di memoria fisica in cui la tabella in questione è stata caricata, visto che tale indirizzo non è facilmente prevedibile. Dunque, anche le entrate della tabella aggiuntiva saranno grandi 8 byte, e la tabella occuperà 1 GiB ($2^{36}/512 = 2^{27}$ entrate, ciascuna di 8 byte). Questa dimensione, pur se più realistica, non è comunque ancora accettabile, ma possiamo riapplicare la stessa tecnica anche a questa: la spezziamo in $2^{27}/512 = 2^{18}$ parti, ciascuna grande 4 KiB, che chiamiamo "tabelle di livello 2", e introduciamo una terza tabella da consultare prima di queste. Questa terza tabella sarà ora grande 2 MiB. Per uniformità, spezziamo anche questa tabella in $2^{18}/512 = 512$ parti grandi 4 KiB che chiamiamo "tabelle di livello 3" e introduciamo un'unica "tabella di livello 4", anch'essa di 4 KiB, che contenga i bit P e gli indirizzi delle 512 tabelle di livello 3. La situazione è dunque quella di Fig. 1, in cui si illustra il caso in cui tutte le tabelle, di tutti i livelli, siano presenti.

Si noti che ora `cr3` contiene l'indirizzo della tabella di livello 4. La traduzione da indirizzo virtuale a fisico è contenuta soltanto nelle entrate delle tabelle di livello 1, che sono le tabelle in cui abbiamo scomposto ogni tabella di corrispondenza della Super-MMU e che, prese in ordine, la ricompongono. Il formato delle entrate delle tabelle di livello 1 è mostrato in Fig. 2 e rispecchia quello dell'architettura Intel/AMD a 64 bit. Chiameremo queste entrate "descrittori di pagina virtuale", come abbiamo fatto fino ad ora, ma anche "descrittori di livello 1", essendo contenuti nelle tabelle di livello 1. Notiamo che i descrittori contengono altre informazioni oltre a quelle che già conosciamo (i bit P, A, D e il campo F). Dal momento che la MMU_1 intercetta tutte le operazioni di lettura

e scrittura eseguite dal programma nel suo spazio di indirizzamento, si trova in una buona posizione per svolgere anche altri compiti basati sugli indirizzi. A questo scopo i descrittori di livello 1 contengono i seguenti bit:

- il bit **P**, che dice se la traduzione è presente o assente;
- un bit **R/W** che vieta (0) o permette (1) le operazioni di scrittura all'interno della pagina virtuale;
- un bit **U/S** che dice se la pagina "appartiene" al livello sistema o utente; una pagina che appartiene al livello sistema non può essere riferita mentre il processore si trova al livello utente;
- un bit **PWT** che (se vale 1) specifica che la cache dovrà usare la politica *write-through* per tutte le scritture eseguite su questa pagina;
- un bit **PCD** che (se vale 1) specifica che la cache dovrà essere disabilitata completamente per tutti gli accessi (letture o scritture) eseguite su questa pagina.
- un bit **D** che viene posto a 1 dalla MMU_1 ogni volta che c'è un accesso in scrittura su questa pagina.

Vedremo in seguito il significato del bit A.

I bit P, R/W, U/S, PWT e PCD sono scritti dalla routine di sistema che attiva il processo (nel nostro caso, `activate_p()`) e soltanto letti dalla MMU_1. Per esempio, la routine potrebbe porre R/W=0 per tutte le pagine che contengono la sezione `.text` del programma utente da eseguire, per evitare che errori nel programma causino scritture nel suo stesso codice. La routine può porre R/W=0 anche per le pagine che contengono soltanto costanti. Durante l'esecuzione del programma, la MMU_1 solleverà un'eccezione ogni volta che il processore inizia una operazione di scrittura ad un indirizzo con associato R/W=0. La routine che gestisce l'eccezione interromperà il programma e stamperà un messaggio di errore.

I bit PWT e PCD, invece, sono un mezzo tramite il quale la routine di attivazione può indirettamente inviare ordini alla cache, sfruttando il fatto che la MMU_1 si trova sul percorso che porta dal processore alla cache (la cache si trova tra la MMU_1 e la memoria fisica). La MMU_1 si preoccuperà di inoltrare l'ordine alla cache ogni volta che traduce un indirizzo. La routine di inizializzazione porrà PCD=1 per tutte le pagine che contengono indirizzi di registri di I/O mappati in memoria, invece che locazioni di memoria. Un esempio è l'APIC, i cui indirizzi sono appunti mappati nello spazio di memoria. Porre PWT=1 e PCD=0 può essere utile per la parte di indirizzi relativa alla memoria video: quando il programma scrive vogliamo che la scrittura arrivi nella vera memoria video e non si fermi in cache, in modo che il controllore possa visualizzare l'informazione sul video; ma se il programma vuole leggere dalla memoria video, possiamo tranquillamente farlo leggere dalla cache.

Ogni volta che il sistema carica le pagine di un processo in memoria (alla prima attivazione, oppure dopo uno *swap-in* in seguito ad uno *swap-out*), dovrebbe

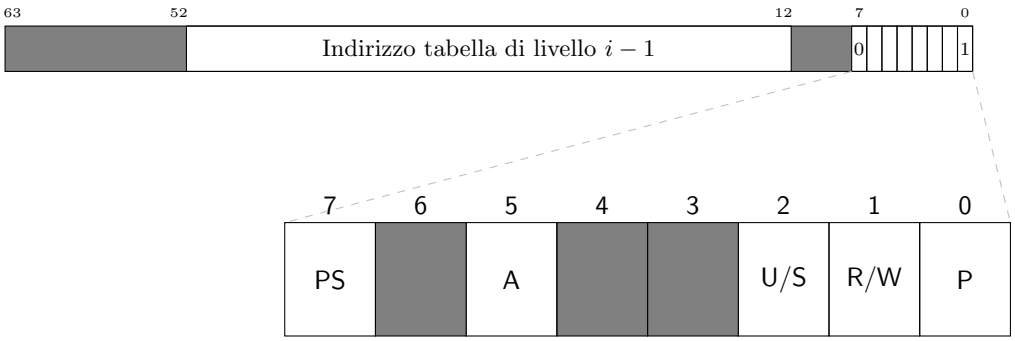

Figura 3: Descrittore di livello i, con $i = 2, 3, 4$.

porre D=0 in tutte le entrate della tabella di corrispondenza. Al momento di eseguire uno swap-out del processo, il sistema può evitare di salvare tutte le pagine del processo nel dispositivo di swap ri-esaminando le entrate e notando in quali di esse il bit D è diventato 1: queste puntano a pagine che sono state modificate e vanno risalvate; per tutte le altre il salvataggio si può evitare, in quanto la copia già presente nello swap è ancora valida.

Tutte le tabelle di livello 2, 3 e 4 hanno lo stesso formato. Ciascuna di esse contiene 512 entrate con il formato illustrato in Fig. 3. Il formato è simile a quello dei descrittori di livello 1 mostrati in Fig. 2: ci sono ancora i bit P, R/W e U/S e A, nelle stesse posizioni dei bit omonimi di Fig. 2; il campo "Indirizzo tabella di livello $i-1$" occupa la stessa posizione del campo "numero di frame" di Fig. 2; per il momento non ci preoccupiamo del nuovo bit PS e assumiamo che valga 0. Si noti che all'indirizzo della tabella mancano i bit da 0 a 11: la MMU_1 assume che questi bit siano 0, cioè che tutte le tabelle partano da indirizzi che sono multipli di 4 KiB (allineamento naturale). Questo vale anche per la tabella di livello 4: i 12 bit meno significativi di `cr3` devono essere tutti a 0.

Chiameremo le entrate delle tabelle di livello 2 "descrittori di livello 2" o "descrittori delle tabelle di livello 1". Si noti il decremento del livello: i descrittori di livello 2 sono contenuti in tabelle di livello 2 e descrivono tabelle di livello 1. Allo stesso modo chiameremo le entrate delle tabelle di livello 3 "descrittori di livello 3" o "descrittori delle tabelle di livello 2" e, infine, chiameremo le entrate della tabella di livello 4 "descrittori di livello 4" o "descrittori delle tabelle di livello 3".

Anche se simili, i descrittori di livello 2, 3 e 4 non vanno confusi con i descrittori di livello 1: i primi descrivono tabelle, mentre quelli di livello 1 descrivono pagine. Solo i descrittori di livello 1 contengono la traduzione da indirizzo virtuale a fisico, mentre i descrittori di tabelle contengono solo dei puntatori ad altre tabelle (di livello inferiore): servono soltanto per raggiungere i descrittori di livello 1 e trovare finalmente la traduzione.

Per eseguire una traduzione la MMU_1 deve attraversare tutto l'albero di Fig. 1, partendo dalla tabella di livello 4, puntata da `cr3`, e seguendo i puntatori

fino a raggiungere una tabella di livello 1 (che è un pezzo della vecchia tabella di corrispondenza). Ogni volta che arriva ad una tabella, MMU_1 deve sapere quali delle 512 entrate consultare per poter andare avanti. La scelta è dettata dal numero di pagina virtuale (bit 12–47) dell'indirizzo virtuale che MMU_1 sta traducendo. Sulla destra di Fig. 1, sotto la scritta "numero di pagina virtuale (base 8)", abbiamo mostrato il numero di pagina virtuale della pagina di cui si occupa il descrittore di pagina subito a sinistra. È comodo esprimere i numeri di pagina virtuale in base 8, ricordando che ogni cifra in base 8 rappresenta 3 bit in base 2. Vediamo che passando dal primo descrittore (indice 0) della tabella di livello 4 si raggiungono tutti e soli i descrittori che si occupano dei numeri di pagina virtuale che cominciano con 000 (9 bit a 0). Passando dall'ultimo descrittore, invece (numero 511, o 777 in base 8), si raggiungono tutti e soli i descrittori che si occupano dei numeri di pagina virtuale che cominciano con 777 (9 bit a 1). Lo stesso vale anche per gli altri 510 descrittori della tabella di livello 4: il descrittore numero xyz (in base 8) porta a trovare la traduzione di tutti e soli i numeri di pagina virtuale che iniziano per xyz. Quindi, dato il numero di pagina virtuale, MMU_1 non deve fare altro che estrarre i primi 9 bit per sapere quale descrittore di livello 4 consultare. Fatto questo, conosce ora il puntatore alla tabella di livello 3 che le interessa. Sempre ragionando sulla Fig. 1, si può vedere che i successivi 9 bit del numero di pagina virtuale determinano ora l'entrata della tabella di livello 3 da consultare. I successivi 9 bit determinano l'entrata da seguire nella corrispondente tabella di livello 2, e gli ultimi 9 bit selezionano il descrittore di pagina nella corrispondente tabella di livello 1.

In Fig. 4 abbiamo riassunto il processo di traduzione operato dalla MMU_1, mostrando più in dettaglio le operazioni svolte da MMU_1, assumendo che tutti i bit P valgano 1. Al primo passo MMU_1 deve leggere il corretto descrittore di livello 4, che si trova nella sua memoria. Per farlo deve eseguire una operazione di lettura in memoria, ovviamente specificando l'indirizzo. La tabella di livello 4 è un vettore di descrittori, ciascuno grande 8 byte, e la MMU_2 vuole leggere il descrittore il cui indice è contenuto nei bit 39–47 di V (indice di livello 4), sia i_4. L'indirizzo a cui vuole leggere è dunque $\texttt{cr3} + i_4 \times 8$. Si noti che, per quanto detto sull'allineamento naturale delle tabelle, questa operazione non comporta una vera somma, ma solo una concatenazione di bit. Anche la moltiplicazione per 8 consiste, ovviamente, nella concatenazione con tre bit costanti pari a 0. Letto il descrittore di livello 4, MMU_2 ne estrae il campo indirizzo (bit 12–51), lo concatena con i bit 30–38 di V e con altri tre bit a 0, ottenendo così l'indirizzo del descrittore di livello 3. E così anche per i descrittori di livello 2 e 1. Arrivata al livello 1, la MMU_2 estrae il campo F (bit 12–51), lo concatena con l'offset di V e ottiene così la traduzione.

Durante la traduzione la MMU_1 esegue anche altri compiti:

- controlla tutti i bit R/W: una operazione di scrittura è permessa solo se tutti e 4 i bit lungo il percorso la permettono (analogamente per il bit CPL);

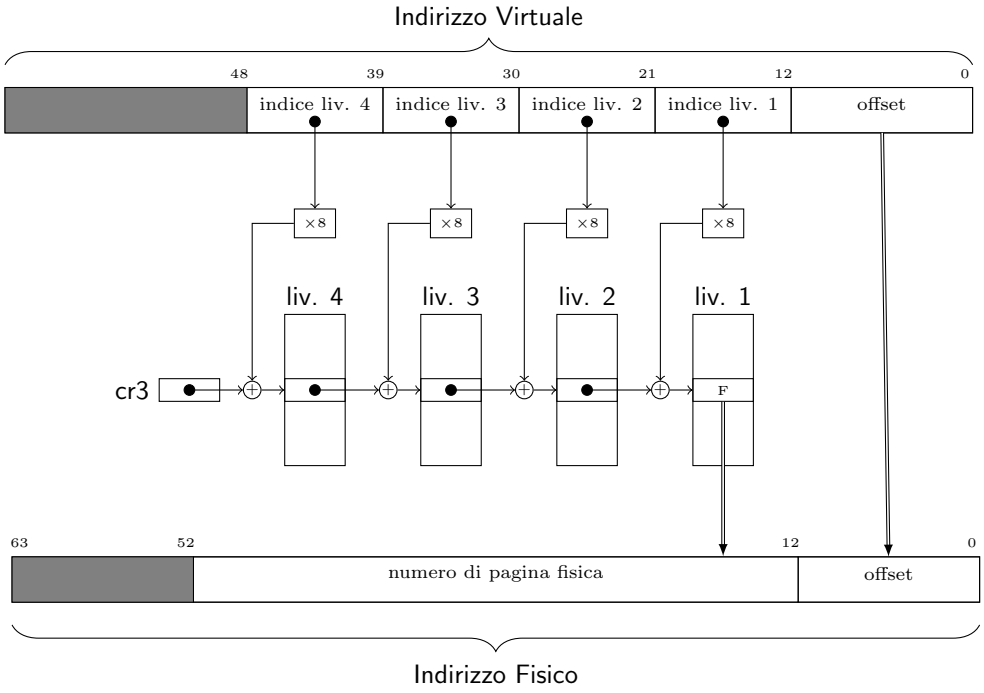

Figura 4: Traduzione da indirizzo virtuale a fisico (pagine di 4 KiB).

- pone a 1 tutti e 4 i bit A incontrati, se non lo erano già (vedremo in seguito perché questa operazione è utile);
- in caso di scrittura, pone a 1 il bit D nella tabella di livello 1.

Si noti che la tabella di livello 4 deve essere sempre presente in memoria fisica (non c'è un bit P in `cr3` che dica alla MMU_2 che la tabella è assente), ma tutte le altre possono essere assenti. L'assenza di una tabella comporta anche l'assenza di tutte le tabelle del sottoalbero di Fig. 1 di cui lei è la radice, e anche l'assenza di tutte le pagine virtuali descritte da tutte le tabelle di livello 1 che si trovano nelle foglie del sottoalbero. Se uno qualunque dei bit P incontrati durante la traduzione vale 0, la MMU_1 smette di tradurre e solleva una eccezione di page fault. La routine di sistema che gestisce il fault terminerà il processo con un errore.