Tabelle multilivello

G. Lettieri

6 Maggio 2019

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. Le parti di tabella presenti saranno compattatate in modo da risparmiare spazio (Figura 1).

Introduciamo quindi MMU_1, una MMU del tutto identica alla Super-MMU, tranne che per il formato della tabella di corrispondenza. Come la Super-MMU, MMU_1 possiede una memoria interna in cui salvare le tabelle di corrispondenza e un registro, `cr3`, che serve ad individuare la tabella attiva ad ogni istante. La speranza è di poter usare una memoria interna molto più piccola rispetto a quella richiesta dalla Super-MMU.

La 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 (quella puntata da `cr3`), esattamente come la Super-MMU. Questa volta, però, può non solo accadere che non sia presente traduzione corrispondente (bit P a zero), ma anche che manchi proprio la parte di tabella a cui accedere.

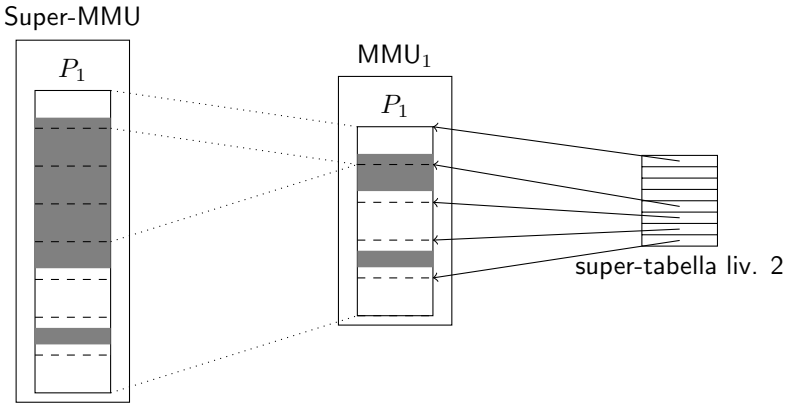

Figura 1: Esempio di tabelle di corrispondenza di un processo P_1 nella Super-MMU e nella MMU$_1$. Le tabelle sono suddivise in parti più piccole separate da linee tratteggiate. Le zone in grigio corrispondono ad entrate non utilizzate, cioè con bit P=0. La MMU$_1$ permette di non allocare spazio per le parti di tabella che risultano completamente inutilizzate. Per trovare le parti allocate si introduce una super-tabella con una entrata per ogni parte della tabella originaria.

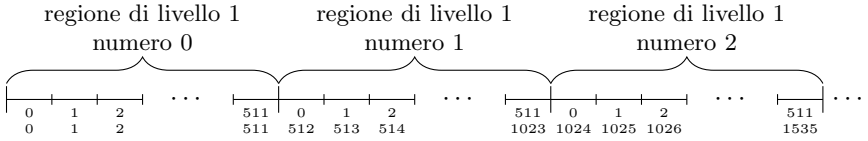

Figura 2: Suddivisione degli indirizzi in regioni di livello 0 (pagine) e regioni di livello 1. Le linee tra due tratti verticali rappresentano le pagine. Sotto le pagine sono riportati il numero di sottoregione e il numero di regione di livello 0 (o numero di pagina).

L'idea è di introdurre una ulteriore tabella, che la MMU$_1$ deve consultare per prima, con una entrata dedicata ad ognuna delle parti in cui abbiamo scomposto la tabella originaria. Di ogni parte dobbiamo sapere se è presente o assente e, nel caso in cui sia presente, anche *dove* si trova nella memoria interna di MMU$_1$.

Per semplicità di gestione conviene spezzare la tabella di corrispondenza in parti di dimensione uguale alle pagine virtuali e ai frame, che stiamo assumendo essere grandi 4 KiB. Chiamiamo ciascuna di queste parti *tabelle di livello 1*. Ogni tabella di livello 1 contiene 512 entrate (4 KiB/8) e si occupa quindi della traduzione di 512 pagine virtuali. Chiamiamo "super-tabella" di livello 2 la tabella aggiuntiva che ci deve dire quali tabelle di livello 1 sono presenti, e dove sono. La tabella usata dalla Super-MMU, invece, possiamo chiamarla super-tabella di livello 1.

Si noti che la scomposizione della super-tabella di livello 1 in *tabelle* di livello 1 corrisponde a introdurre, nello spazio degli indirizzi, una ulteriore scomposizione in regioni oltre a quella delle pagine. Infatti, mentre ogni singola entrata di una tabella di livello 1 si occupa di tradure gli indirizzi di una pagina virtuale (4 KiB), una intera tabella di livello 1 si occupa, complessivamente, di tradurre gli indirizzi di una regione che copre 512 pagine e, dunque, si estende per 2 Mi indirizzi. Chiamiamo queste nuove regioni *regioni di livello 1* (si veda la Figura 2). Per uniformità chiamiamo *regioni di livello 0* le normali pagine di 4 KiB. Ogni regione di livello 1 sarà identificata dal suo numero di regione, come al solito. Per le regioni di livello 0 (le pagine) abbiamo invece una novità: ciascuna regione di livello 0 si trova interamente all'interno di una precisa regione di livello 1 e, dunque, possiamo considerarla una *sottoregione* di quella regione. All'interno di ogni regione di livello 1 possiamo assegnare ad ogni sottoregione un numero progressivo, da 0 a 511, che chiamiamo *numero di sottoregione 0 dentro il livello 1*. Questo comporta che, in alternativa al numero di pagina, possiamo identificare ogni regione di livello 0 anche specificando il numero della regione di livello 1 in cui cade e il *numero di sottoregione 0* al suo interno. Per esempio, dalla Figura 2 si vede come la pagina numero 1025 sia anche identificata dalla coppia "(numero di regione di livello 1, numero di sottoregione 0 dentro il livello 1)" pari a $(2, 1)$.

Il fatto che tutte le dimensioni siano potenze di due comporta che questi nuovi numeri si ottengono facilmente: dato il numero di pagina, sia r_0, possiamo scomporlo in due parti (r_1, r_0^1), con r_0^1 che contiene i 9 bit $(log_2(512))$ meno significativi di r_0 e r_1 i rimanenti bit. La parte r_1 è il numero di regione 1[1] mentre r_0^1 è il numero di sottoregione 0 nel livello 1. Prendiamo nuovamente, come esempio, la pagina numero 1025. Notiamo che $(1025)_{10} = (2\ 001)_8$ (il pedice rappresenta la base di rappresentazione) e dunque i 9 bit meno significativi del numero di pagina valgono 1 e i rimanenti valgono 2.

Un indirizzo virtuale v potrà essere ora scomposto in 3 parti: (r_1, r_0^1, o). Il numero di regione 1, r_1, dice alla MMU_1 quale entrata della super-tabella di livello 2 deve consultare per sapere se la corrispondente tabella 1 è presente o meno. L'entrata conterrà dunque un flag, che chiamiamo P per analogia a quello contenuto nelle entrate delle tabelle 1, destinato a fornire questa informazione. Se P=1, l'entrata contiene anche l'indirizzo (nella memoria interna di MMU_1) della tabella 1. Il numero di sottoregione 0 dentro il livello 1, r_0^1, dice alla MMU_1 quale entrata della tabella 1 consultare per ottenere la traduzione di v (o per sapere se v non ha traduzione).

Anche le entrate della super-tabella 2 saranno grandi 8 byte e la super-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. Riapplichiamo allora la stessa tecnica anche alla super-tabella 2: la spezziamo in $2^{27}/512 = 2^{18}$ parti, ciascuna grande 4 KiB, che chiamiamo *tabelle di livello 2*, e introduciamo una terza super-tabella (di livello 3) da consultare prima di queste. Ogni tabella di livello 2 si occupa della traduzione di una regione di livello 2, grande 1 GiB.

[1] Omettiamo di ripetere "di livello" quanto ciò non crea ambiguità.

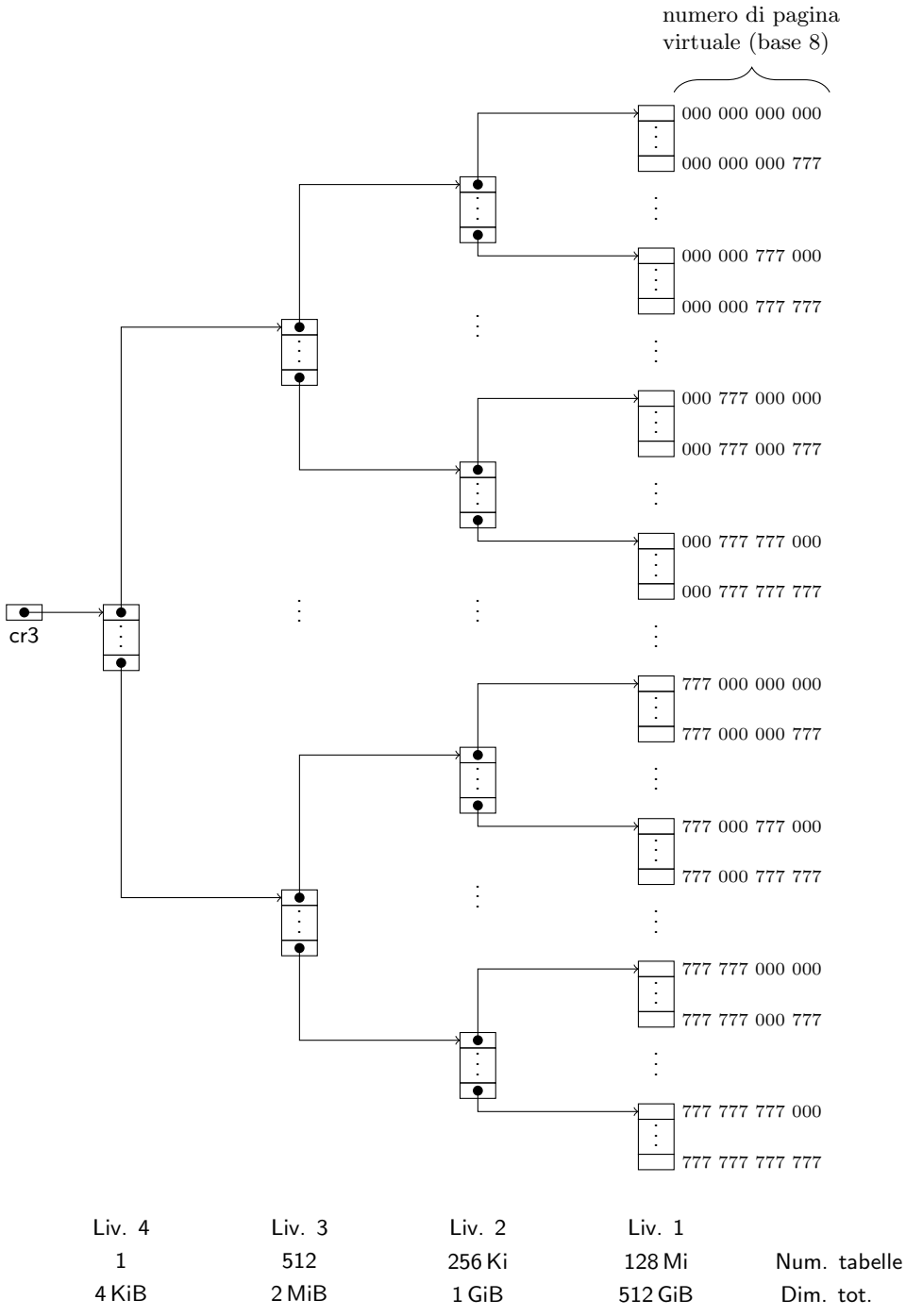

Figura 3: Tabella di corrispondenza su 4 livelli. La figura mostra quali entrate delle tabelle di livello 1 contengono la traduzione dei numeri di pagina virtuale mostrati sulla destra.

Ciascuna entrata della super-tabella di livello 3 che abbia P=0 dice alla MMU_2 che una intera regione di livello 2 è assente.

Ciascuna regione di livello 2 ha al suo interno 512 sottoregioni di livello 1 (ciascuna delle quali contiene 512 sottoregioni di livello 0). Ciascun numero di regione 1, sia r_1, può scomporsi in $r_1 = (r_2, r_1^2)$, dove r_2 è il numero di regione 2 e r_1^2 è il numero di sottoregione 1 nel livello 2. Il numero di sottoregione r_1^2 è ancora una volta dato dai 9 bit meno significativi del numero di regione. Un indirizzo virtuale v si scomporrà in (r_2, r_1^2, r_0^1, o). Il numero r_2 fornisce alla MMU_1 l'indice della entrata da consultare nella super-tabella 3, da cui ottiene (se trova P=1) l'indirizzo della tabella 2 che si occupa della traduzione della regione di livello 2 numero r_2. Userà poi il numero di sottoregione r_1^2 come indice nella tabella 2 per trovare (se P=1) l'indirizzo della tabella 1 che si occupa della traduzione della regione di livello 1 numero (r_2, r_1^2). Quindi userà r_0^1 come indice nella tabella 1 per trovare la traduzione della pagina numero (p_2, p_1^2, p_0^1).

La super-tabella 3 deve avere 2^{18} entrate, ciascuna di 8 byte, e sarà dunque grande 2 MiB Questa sembra una dimensione accettabile, ma dobbiamo ricodare che ne serve una per ogni processo e, quindi, è ancora troppo grande. Provvediamo a dunque a scomporre anche questa in $2^{18}/512 = 512$ parti grandi 4 KiB che chiamiamo *tabelle di livello 3*, ciascuna delle quali si occupa della traduzione di una regione di livello 3 grande 512 GiB. Questa volta dobbiamo introdurre un'unica *tabella di livello 4*, grande solo 4 KiB, che contenga i bit P e gli indirizzi delle 512 tabelle di livello 3.

Un indirizzo virtuale risulta dunque scomposto in $(r_3, r_2^3, r_1^2, r_0^1, o)$, dove sia r_3 che r_2^3, r_1^2 e r_0^1 sono grandi 9 bit. La situazione finale è quella di Fig. 3, in cui si illustra il caso in cui tutte le tabelle, di tutti i livelli, siano presenti. La struttura dati che permette la traduzione ha dunque la forma di un albero. Si noti che ora `cr3` contiene l'indirizzo della tabella di livello 4, che è la radice dell'albero. La traduzione da indirizzo virtuale a fisico è contenuta soltanto nelle entrate delle tabelle di livello 1—le foglie dell'albero. Queste sono le tabelle in cui abbiamo scomposto la super-tabella di livello 1 che usava la Super-MMU e che, prese in ordine, la ricompongono. Le altre tabelle servono solo a raggiungere le tabelle di livello 1, o a dire quali di queste sono assenti.

Per eseguire una traduzione la MMU_1 deve attraversare tutto l'albero di Fig. 3, partendo dalla tabella di livello 4, puntata da `cr3`, e seguendo i puntatori fino a raggiungere una tabella di livello 1. Come abbiamo visto, il percorso nell'albero è determinato dal numero di pagina virtuale (bit 12–47) dell'indirizzo virtuale che MMU_1 sta traducendo, che dobbiamo immaginare scomposto in $(r_3, r_2^3, r_1^2, r_0^1)$.

Facciamo un esempio supponendo che la MMU_1 si trovi a dover tradurre l'indirizzo virtuale $v = (000\ 777\ 000\ 777\ 1234)_8$. Il numero di pagina è $(000\ 777\ 000\ 777)_8$. Vogliamo dunque che la MMU_1 arrivi a leggere l'ultima entrata della terza tabella di livello 1 dall'alto della Figura 3: questa è l'entrata che contiene il numero di frame da sostituire al numero di pagina. Avremo $r_3 = (000)_8$, $r_2^3 = (777)_8$, $r_1^2 = (000)_8$, $r_0^1 = (777)_8$. I primi 9 bit del numero di pagina, r_3, identificano il numero di regione di livello 3, che in questo caso

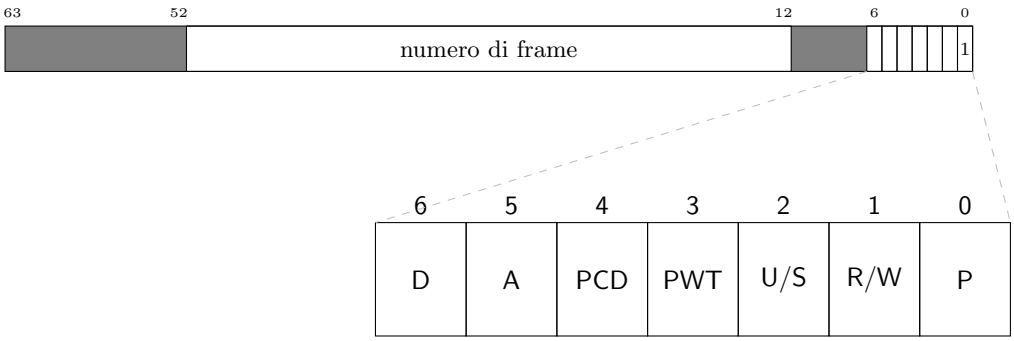

Figura 4: Descrittore di pagina virtuale (tabelle di livello 1).

è 0. La MMU_1 usa quindi l'entrata numero 0 della tabella 4 per trovare la prossima tabella da consultare. Come possiamo confermare dalla Figura 3, infatti, tutte le tabelle di livello 1 che traducono gli indirizzi che appartengono a questa regione (vale a dire, tutti i numeri di pagina che iniziano per $(000)_8$) si trovano nel sottoalbero che parte da questa entrata. La MMU_1 passerà dunque alla tabella di livello 3 in alto in Figura 3. Qui usa i successivi 9 bit, r_2^3 per sapere quale entrata di questa tabella deve percorrere per raggiungere la prossima tabella. Nel nostro esempio abbiamo $r_2^3 = (777)_8$. Questo indica che la pagina di cui stiamo cercando la traduzione si trova nell'ultima sottoregione di livello 2 all'interno della regione di livello 3 numero 0. Dunque la MMU_1 userà l'ultima entrata. Per conferma, si noti come in Figura 3 tutte e sole le tabelle di livello 1 che traducono gli indirizzi che iniziano con $(000\ 777)_8$ si trovano nel sottoalbero che parte dall'ultima entrata della tabella di livello 3 in alto. La MMU_1 passerà dunque alla seconda tabella di livello 2 dall'alto in Figura 3. Qui userà i bit $r_1^2 = (000)_8$ e proseguirà verso la terza tabella di livello 1 dall'alto. Infine, troverà la traduzione che stava cercando nell'entrata $r_0^1 = (777)_8$ di questa tabella.

Il formato delle entrate delle tabelle di livello 1 è mostrato in Fig. 4 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, oltre ai bit che già conosciamo, due bit A e D il cui significato è il seguente:

- bit A: viene posto a 1 dalla MMU_1 ogni volta che c'è un accesso (sia in lettura che in scrittura) su questa pagina;
- bit D: viene posto a 1 dalla MMU_1 ogni volta che c'è un accesso in scrittura su questa pagina.

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.

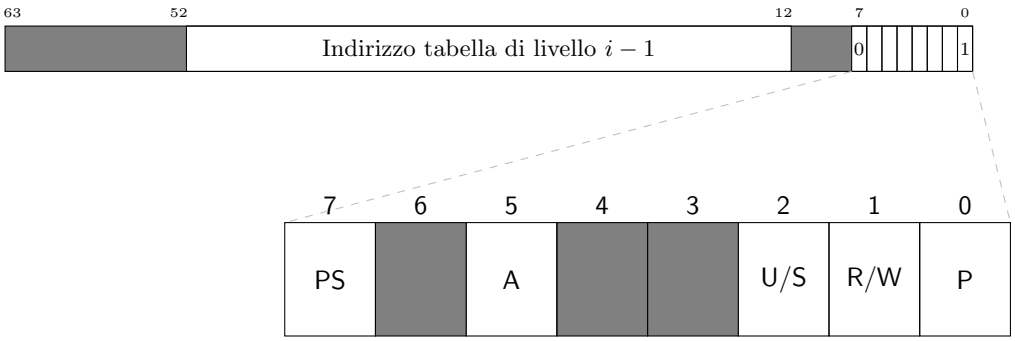

Figura 5: Descrittore di livello i, con $i = 2, 3, 4$.

I bit A e D, invece, sono letti e scritti sia dal software (di sistema) che dalla MMU_1.

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 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. Vedremo in seguito il significato del bit A, quando parleremo di paginazione su domanda.

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. 5. Il formato è simile a quello dei descrittori di livello 1 mostrati in Fig. 4: ci sono ancora i bit P, R/W e U/S e A, nelle stesse posizioni dei bit omonimi di Fig. 4; il campo "Indirizzo tabella di livello $i-1$" occupa la stessa posizione del campo "numero di frame" di Fig. 4; 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. I descrittori di tabella servono a trovare le tabelle di livello

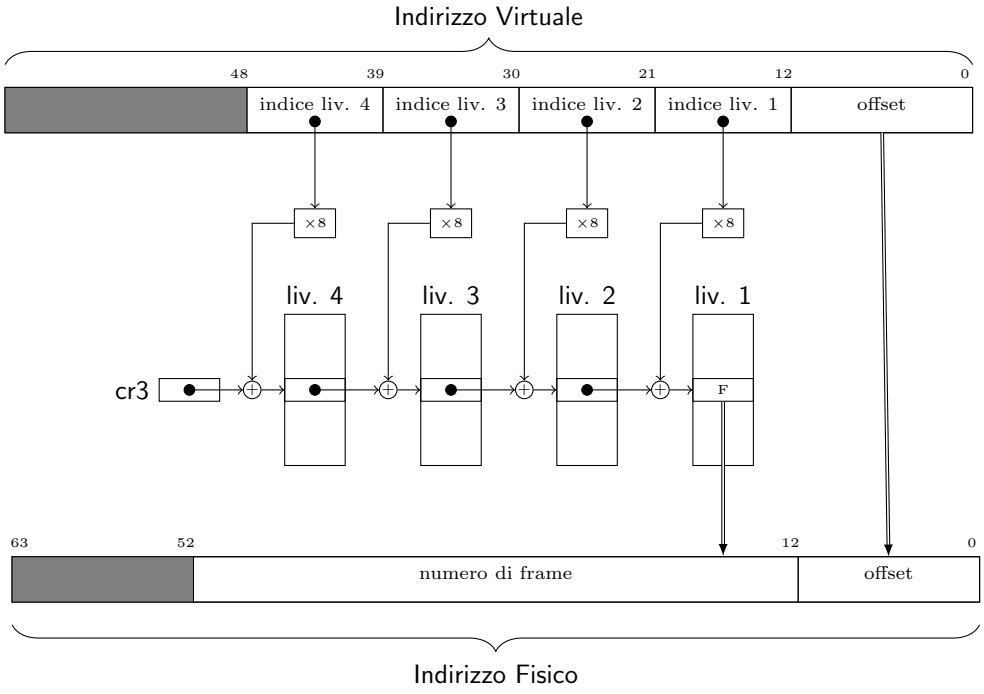

Figura 6: Traduzione da indirizzo virtuale a fisico (pagine di 4 KiB).

inferiore, ma solo le tabelle foglia contengono la traduzione cercata.

In Fig. 6 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, analoghi ai compiti aggiuntivi che svolgeva la Super-MMU, ma applicati alla struttura multi-livello:

- controlla tutti i bit R/W: una operazione di scrittura è permessa solo se tutti e 4 i bit lungo il percorso la permettono;
- controlla tutti i bit U/S: una operazione (di lettura o scrittura) è permessa solo se tutti e 4 i bit lungo il percorso la permettono;
- passa al controllore cache le informazioni contenute nei bit PWD e PCD nel descrittore di livello 1;
- 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 (i descrittori di livello maggiore di 1 non hanno il bit D).

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. 3 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.