Soluzioni del Capitolo 7

Questo documento contiene le soluzioni ad un numero selezionato di esercizi del Capitolo 7 del libro "Calcolatori Elettronici - Architettura e organizzazione", Mc-Graw Hill 2017.
Coloro che avessero sviluppato soluzioni alternative a quelle qui proposte, o soluzioni a esercizi non compresi tra quelli qui trattati, sono invitati a trasmetterle all'indirizzo sotto riportato. Serviranno a migliorare e tenere aggiornati i contenuti di questo sito.
L'autore sarà grato nei confronti di coloro che segnaleranno errori di qualunque genere, sia nella parte che segue sia nel libro menzionato.
giacomo.bucci@unifi.it

Aggiornato il 19 aprile 2017

7.1 Il termine programma indica una sequenza ordinata di istruzioni indipendentemente da come viene eseguita. Il termine processo indica l'attività che si instaura nella macchina per effetto dell'esecuzione del programma. Programma è l'aspetto *statico*, processo è l'aspetto *dinamico*.

E' possibile immaginare che un processo sia permanentemente in esecuzione e che in un dato momento carichi nel suo spazio di memoria un programma, sviluppato a parte, che svolge una specifica funzione che poi ha termine; successivamente lo stesso processo può mettere in esecuzione un differente programma. Ovviamente, perché questo possa accadere, i programmi dovranno rispettare le condizioni imposte dall'ambiente creato dall'esecuzione del processo.

7.4 Lo schema è in Figura 7.1. Ci si riferisce a un sistema multitasking **non** a divisione di tempo. Nello stato "Running" il processo è in possesso della CPU, nello stato "Ready" è potenzialmente in condizione di eseguire, ma è in attesa che gli venga assegnata la CPU. L'evento Dispatch, determinato dal fatto che il processo correntemente in possesso della CPU la rilascia, fa passare il processo dallo stato "Ready" a quello "Running". Dallo stato "Running" il processo può uscire perché termina oppure perché si mette in attesa di un qualche evento o del completamento di una operazione di I/O. Al verificarsi dell'evento, ovvero al completamento dell'operazione di I/O, il processo torna nello stato "Ready". Notare che un processo appena creato viene portato in stato "Ready".

L'arco etichettato con "Preempted" si riferisce al caso in cui diventa eseguibile un processo a più alta priorità di quello correntemente in esecuzione. Tale evento determina la sospensione del processo in esecuzione, che passa allo stato "Ready" e l'elezione del processo di maggior priorità a "Running". Non è detto che un sistema operativo preveda il meccanismo di *preemption* (prelazione).

Se ci fosse anche la divisione di tempo, il passaggio da "Running" a "Ready" sarebbe possibile anche per termine del quanto di tempo.

7.5 E' anzitutto necessario che il nucleo del sistema operativo sia in grado di gestire, eventi e code e di attuare la commutazione tra processi (*task switching*). A tale scopo ogni processo deve avere un suo descrittore o *Task Control Block* (TCB) che ne rappresenta il suo stato. Il TCB deve contenere tutte le informazioni che devono essere caricate in CPU al momento del passaggio in esecuzione (stato "Running"); in modo duale in TCB viene copiato lo stato del processore quando esso passa dallo stato di esecuzione ad altri

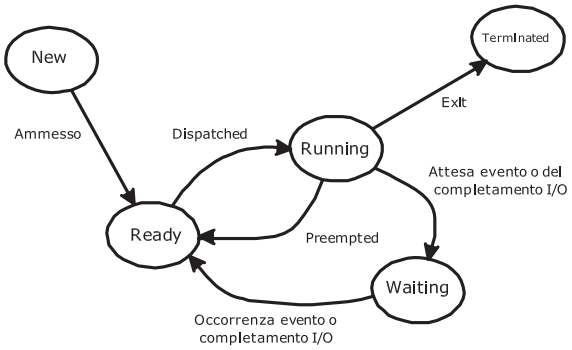

Figura 7.1 Stati di un processo (Esercizio 7.15).

stati. Nel funzionamento in modo protetto, l'architettura ×86 fornisce tutto il supporto necessario al *task switching*.

I processi in grado di eseguire, sono tenuti in una lista (ReadyL); in testa alla lista c'è sempre il processo "Running". La lista ReadyL potrà essere gestita tenendo conto della priorità (assegnata a ciascun processo). Se il processo "Running" avvia un'operazione di I/O e deve attenderne il completamento, il sistema deve deve sospenderne l'esecuzione e metterlo in attesa dell'evento che segnale il completamento. A tale scopo il sistema prevedere/gestire liste di attesa di eventi, riconoscere quale processo debba essere riattivato al verificarsi di un dato evento.

Il sistema dovrà prevedere anche le cosiddette primitive di sincronizzazione, ad esempio le primitive `wait` e `signal` su semafori. Un'operazione di wait su un semaforo "rosso", deve sospendere l'esecuzione del processo e metterlo in attesa del "verde". A tale scopo al semaforo si può associare una coda in cui vengono messi ordinatamente i processi che trovano il rosso. Nel momento in cui un processo esegue un'operazione di signal sul medesimo semaforo, il processo in testa alla coda viene messo in esecuzione, passando dallo stato "Waiting" allo stato "Ready". In termini pratici ciò equivale a estrarre il processo dalla lista di attesa associata allo specifico semaforo e a inserirlo nella lista "Ready".

7.8 Per poter trattare l'esercizio occorre fare alcune assunzioni. Stabiliamo che ciascuno dei due livelli della tabella delle pagine sia costituto da pagine contenti 1K PTE. Indichiamo con TH e TL le tabelle di primo e secondo livello e osserviamo che per ciascuno dei due processi bastano una tabella TH e una TL.

Nella Tabella di primo livello della PMT di $p1$ (TH_{p1}) risulterà occupata la sola prima posizione nella quale si troverà il riferimento a $\mathrm{TL}_{p1,0}$, ovvero all'unica tabella di secondo livello della PMT di $p1$. Dunque TH_{p1} avrà questo aspetto (N indica il numero d'ordine in tabella, P indica il bit di presenza)

N	P	Riferimento a
0	1	$\mathrm{TL}_{p1,0}$
1	0	–
2	0	–
..	0	–
1023	0	–

mentre $\mathrm{TL}_{p1,0}$ avrà questo aspetto (Pf($\leftarrow k$) indica la pagina fisica su cui è mappata la pagina virtuale k.

N	P	Riferimento a
0	1	Pf($\leftarrow$ 0)
1	1	Pf($\leftarrow$ 1)
2	1	Pf($\leftarrow$ 2)
3	1	Pf($\leftarrow$ 3)
4	0	–
5	1	Pf($\leftarrow$ 5)
6	0	–
7	1	Pf($\leftarrow$ 7)
8	1	Pf($\leftarrow$ 8)
9	1	Pf($\leftarrow$ 9)
10	0	–
..	0	–
1023	0	–

La PMT di $p2$ si costruisce in modo analogo.

7.9 Poiché la pagina è di 16 KB, il campo relativo allo scostamento entro pagina è di 14 bit, per cui, essendo l'indirizzo virtuale su 36 bit, il campo per il numero di pagina virtuale (VPN) è di 36−14=22 bit. Essendo l'indirizzo fisico a 32 bit, il campo per il numero di pagina fisica (RPN) è pari a 32-14= 18 bit.

Considerando che nel descrittore di pagina occorre prevedere tre bit per Presenza (P), Modifica avvenuta (M) e Riferimento avvenuto (R), oltre che 3 bit riservati per il sistema operativo, ogni descrittore deve essere di almeno 24 bit. Poiché la macchina legge/scrive a 32 bit è necessario portare a 32 bit la dimensione dell'elemento. Restano pertanto 8 bit non utilizzati. La struttura degli indirizzi è in Figura 7.2.

La dimensione della tabella è pari a $2^{22} \times 4 = 2^{24}$ B.

Si noti che il numero 24 suggerirebbe descrittori di 3 B. Ma questa non è una soluzione accettabile, infatti porterebbe a costruire una PMT i cui elementi non sarebbero allineati ai confini di parola.

7.10 Facciamo l'ipotesi semplificativa che istruzioni e dati siano sempre di 32 bit. Con una pagina di 1K istruzioni, anche nell'ipotesi estrema che venga percorsa da cima a fondo senza mai dar luogo a loop, il tasso di miss sulla pagina, ovvero sulla stessa posizione in TLB, sarebbe 1/1024. Si tratta di un tasso di un ordine inferiore di grandezza rispetto al tasso di miss delle memorie cache.

Un TLB di 64 posizioni è sufficiente a rappresentare un programma di 64 K istruzioni (ovvero 256 KB). Tenuto conto della località dei programmi è assai probabile che questa misura sia sufficiente a rappresentare le pagine più frequentemente usate. Per esempio possiamo immaginare che un programma, comunque grande, usi normalmente il codice

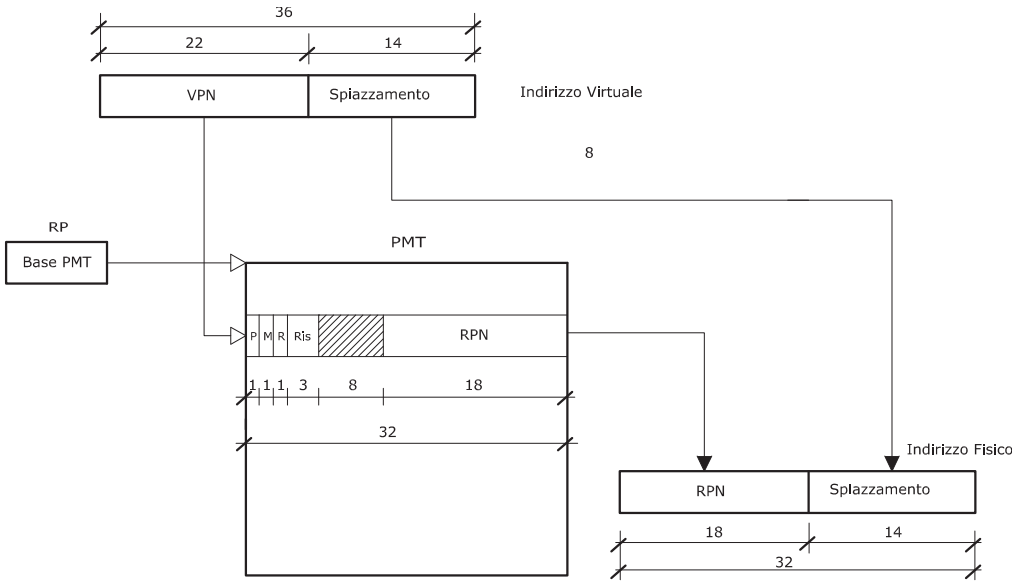

Figura 7.2 Struttura degli indirizzi dell'esercizio 7.9.

contenuto in 40 pagine; in tal caso resterebbero 24 posizioni per fare riferimento alle pagine usate raramente.

7.11 Il formato di una posizione del TLB è in Figura 7.3. Si noti che ciascuna via è di 32 posizioni e perciò richiede 5 bit per l'indirizzamento. Ne consegue che il campo TAG è formato da $22 - 5 = 17$ bit. Il numero di bit in ogni posizione è pari a $3 + 17 + 18 = 38$ bit, per un totale di $38 \times 64 = 2432$ bit.

7.12 La cache è di 256 KB e linee di 32 B (8 parole). Il numero delle linee è dato da $2^{18}/2^5 = 2^{13} = 8$ K linee. Il formato dell'indirizzo è mostrato in Figura 7.4.
la dimensione complessiva del TAG è data da $(14+1) \times 2^{13}$bit$= 15 \times 8$K bit $= 15$ KB.

7.13 Poiché le pagine fisiche sono di 8 KB, occorrono 13 bit per indirizzarle. Dunque il campo Offset è di 13 bit (A_{12}-A_0). Poiché l'indirizzo fisico è su 32 bit, i PTE della tabella di secondo livello devono presentare $32 - 13 = 19$ bit per identificare il numero di pagina fisica. Ne consegue che una PTE nella tabella di secondo livello è necessariamente di 4 byte[1].

Si ricorda che il numero di pagina fisica, esteso con tanti zeri quanti corrispondono al numero di bit di indirizzamenti in pagina (in questo caso 13) fornisce l'indirizzo fisico di partenza (la base) della pagina. Alternativamente, l'estensione di un numero con n zeri a destra equivale a moltiplicare per 2^n.

[1]Ovviamente non ha senso prendere 3 byte per le PTE, in quanto ciò determinerebbe problemi di allineamento, ecc.

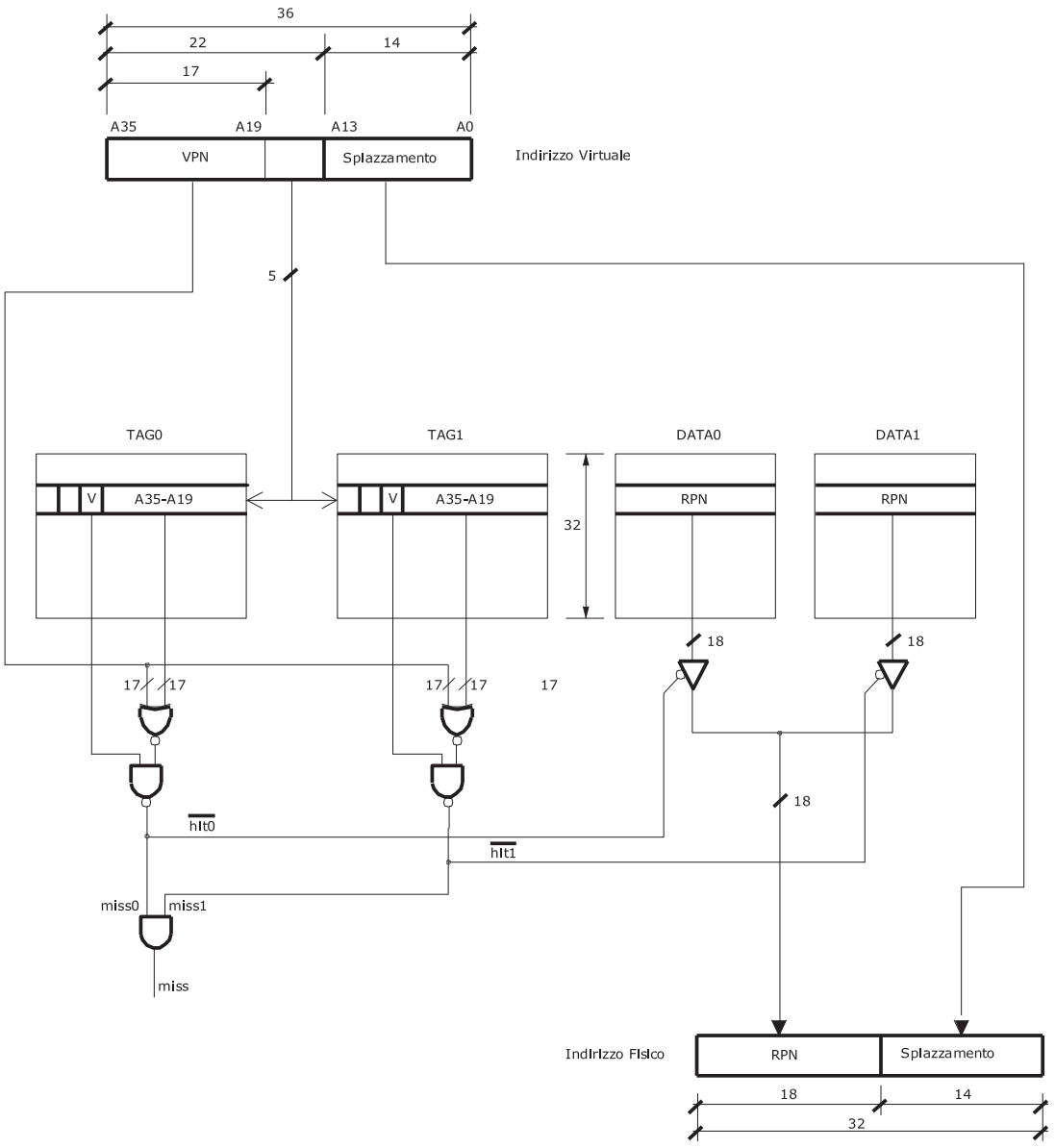

Figura 7.3 Struttura del TLB dell'esercizio 7.11 e formazione del'indirizzo fisico. I campi a sinistra di V contengono i bit M, A, ecc.

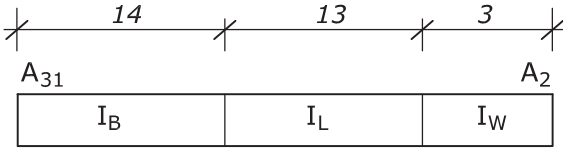

Figura 7.4 Formato dell'indirizzo relativo all'esercizio 7.12.

Poiché per indirizzare nella tabella di primo livello si usano i 12 bit più significativi dell'indirizzo, per indirizzare entro la tabella di secondo livello restano $32-12-13 = 7$ bit. Dunque la tabella di secondo livello contiene $2^7 = 128$ posizioni, pari a $128 \times 4 = 512$ B.

Un PTE della tabella di primo livello, deve fornire la base per una tabella di secondo livello, entro la quale si indirizza con $7+2 = 9$ bit (si ricordi che gli indirizzi sono assegnati ai byte e quindi occorre scalare di 4 il numero che corrisponde al PTE di secondo livello). Ne consegue che il campo del PTE della tabella di primo livello che contiene il numero di PTE nella tabella di secondo livello deve essere ampio $32 - 9 = 23$ bit. Tenuto conto dei bit accessori, un PTE di primo livello richiede sicuramente 4 B. La tabella di primo livello contiene $2^12 = 4$ K elementi, pari a 16 KB.

Lo schema della traduzione degli indirizzi è in Figura 7.5

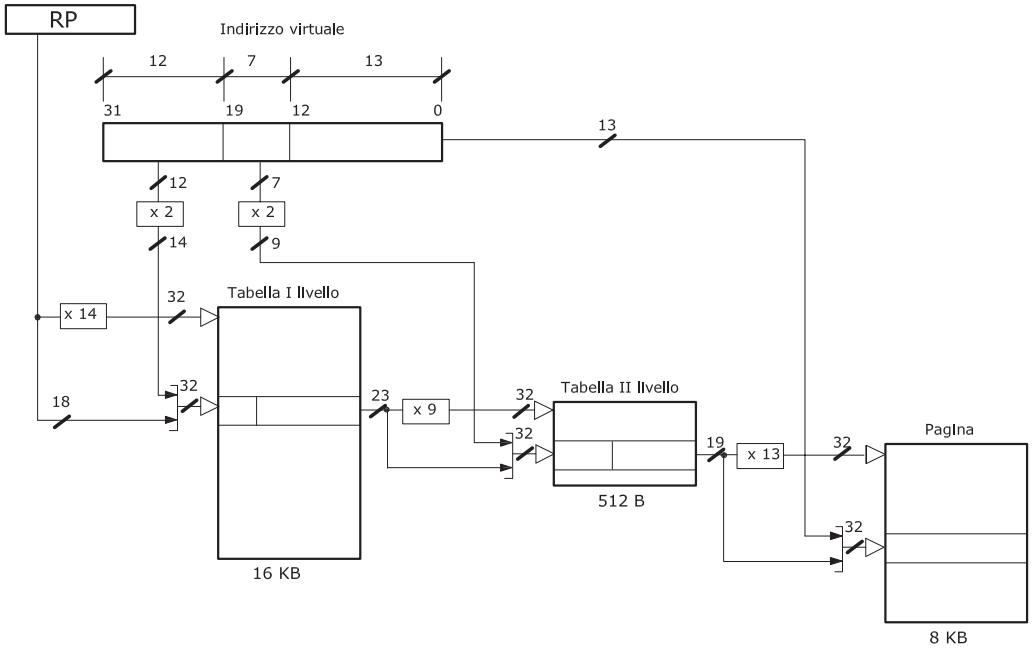

Figura 7.5 Traduzione degli indirizzi (esercizio 7.13). La notazione "$\times n$" sta a indicare l'estensione con n zeri del relativo campo, ovvero la concatenazione di n zeri a destra. Ciò equivale a moltiplicare per 2^n il valore contenuto nel campo. Si faccia attenzione al fatto che la notazione usata per indicare le estensioni dei campi è diversa rispetto a quella del libro di testo.

Un elemento di un TLB (completamente associativo) ha un TAG composto da $12 + 7 = 19$ bit, oltre ai bit di validità, ecc.. Esso avrebbe pure 19 bit per la parte corrispondente a numero di pagina fisica.

7.14 Per ovvi motivi di allineamento una PTE richiede 4 B (una parola allineata).

Le pagine fisiche hanno dimensione pari a 1 KB, corrispondenti a 256 parole. Dunque per indirizzare una parola entro la pagina si richiedono 8 bit (A_{9-2}).

La tabella di primo livello deve contenere $128 = 2^7$ posizioni e pertanto l'indirizzamento al suo interno richiede i 7 bit più significativi dell'indirizzo virtuale (A_{31-25}). La tabella ha dimensione pari a $2^9 = 512$ B.

La tabella di secondo livello deve contenere $256 = 2^8$ posizioni e pertanto l'indirizzamento al suo interno richiede i prossimi 8 bit dell'indirizzo virtuale (A_{24-17}). La tabella ha dimensione pari a $2^{10} = 1$ KB.

Per la tabella di terzo livello restano restano perciò 7 linee di indirizzo (A_{16-10}). La tabella ha dimensione pari a $2^9 = 512$ B.

Se la tabella di primo e secondo livello diventano di 512 ($= 2^9$) posizioni, per la tabella di terzo livello restano 4 linee di indirizzo (A_{13-10}) e la tabella stessa assume la dimensione di 64 B.

7.15 Cominciamo con l'osservare che le pagine sono di 8 KB, ovvero di 4 K parole, essendo la macchina a 16 bit. L'indirizzo fisico di una parola è dato dai bit A_{23-1}, mentre il bit A_0 (con la linea $\overline{\text{BHE}}$) viene usato per selezionare il byte.

L'indirizzamento delle 4 K parole richiede 12 linee di indirizzo (A_{12-1}). Poiché l'indirizzo è su 24 bit, occorrono $24 - 13 = 11$ linee per identificare il numero di pagina fisica attraverso i PTE della tabella di secondo livello. Il numero di pagina fisica rappresenta gli 11 bit più significativi (A_{23-13}) dell'indirizzo di base della pagina fisica (i restanti 13 bit dell'indirizzo di pagina sono a zero).

Essendo la macchina a 16 bit, nel PTE restano $16 - 11 = 5$ bit per le informazioni accessorie (bit P, M, A, Acc). Si può dunque ritenere che un PTE occupi 2 B (una parola).

Poiché la tabella di secondo livello è di 4 KB, essa viene a contenere 2 K PTE. L'indirizzamento entro questa tabella richiede pertanto 11 linee di indirizzo (A_{11-1}). Essendo l'indirizzo fisico su 24 bit, il numero di tabella (di secondo livello) viene espresso su $24 - 12 = 12$ bit. Dunque nelle PTE di primo livello restano $16 - 12 = 4$ bit per le informazioni accessorie. Ciò implica che il campo Acc deve necessariamente ridursi a 1 bit.

Restano infine $30 - 11 - 13 = 6$ bit per indirizzare entro la tabella di primo livello. Lo schema complessivo è in Figura 7.6. In questa figura si è usata una notazione diversa da quella di Figura 7.6. Il simbolo "$\oplus$" sta a rappresentare la solita concatenazione dove l'elemento che si concatena a destra è eventualmente scalato. Più precisamente, vengono scalati di 1 "Scost in TL1" e "Scost in TL2". Ovviamente non viene scalato lo scostamento della parola. Si è evidenziato che il bit A_0 non viene direttamente usato per indirizzare la parola.

7.16 Per quanto si riferisce alla prima parte si faccia riferimento al libro di testo ed eventualmente ai manuali Intel. Rispondiamo invece alle domande che si riferiscono all'esecuzione del salto, nell'ipotesi che il salto non determini un *task switch*.

Ovviamente il salto ha effetto sulla coppia dei registri di CPU (CS, EIP) nei quali vengono rispettivamente depositati CSx (il selettore del segmento di destinazione) e EIPx (l'OFFSET nel segmento di destinazione). Inoltre, poiché la coppia (CS0, EIP0) viene salvata nello stack, ne consegue che viene aggiornato il registro ESP. Infine, il cambiamento del selettore di segmento determina l'aggiornamento del descrittore di segmento nella posizione di cache associata a CS.

Se il segmento di destinazione è presente in memoria (assumiamo che sia un segmento locale, cioè descritto in LDT), la logica di CPU, riconoscendo che il selettore di segmento

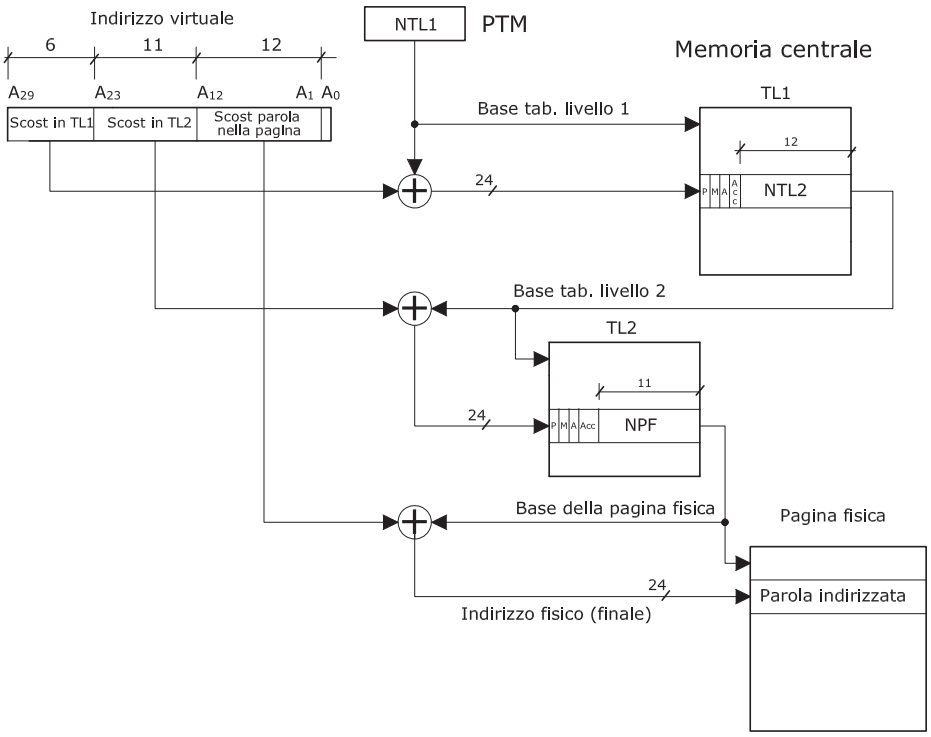

Figura 7.6 Traduzione degli indirizzi (esercizio 7.15). NPF sta per "numero di pagina fisica"; NTL1 per "numero di tabella di primo livello"; NTL2 per "numero tabella di secondo livello".

deve cambiare, preleva il LDT il descrittore individuato tramite il selettore CSx (in realtà tramite la componente INDEX di CSx) e lo copia nella posizione di cache associata a CS (vedi sopra).

Se il segmento di destinazione non è presente in memoria, l'accesso alla LDT alla posizione individuata da CSx determina un *segment fault*. A tale evento, che si manifesta come eccezione, segue necessariamente un cambiamento di contesto (passaggio all'*exception handler* e quindi al sistema operativo). Ne consegue che tutti i registri di CPU ne sono influenzati.

Notare che se il salto determina un *task switch*, il processo che si instaura porta ovviamente ad aggiornare la coppia (CS,IP), ma ciò comporta anche l'aggiornamento di tutti i registri di CPU con il nuovo TSS.

7.17 Occorre anzitutto ipotizzare che il sistema tenga una lista LRU con in numeri di pagina virtuale). Supponendo che la lista LRU sia completamente piena, ovvero che siano allocate un numero di pagine fisiche pari alla capienza massima della lista, un algoritmo molto rozzo potrebbe essere questo:

- al verificarsi di un *page fault* vengono esaminati tutti i PTE delle pagine presenti in memoria e quelle che hanno avuto un riferimento (A = 1) vengono messe in testa alla lista LRU; il bit A viene riportato a 0;

- concluso l'aggiornamento della lista LRU, viene designata come vittima la pagina che risulta ultima;
- (il PTE de) la nuova pagina (quella per cui si è fatto posto) viene inserita in testa alla lista.

Notare che questo algoritmo può essere reso meno oneroso. A tale scopo basta procedere dal fondo della LRU operando in questo modo:

- se la pagina in fondo alla lista ha ricevuto un riferimento allora essa viene portata in testa e si procede con la prossima;
- se la pagina non ha avuto un riferimento essa è certamente la più stagionata e viene designata come vittima.
- L'algoritmo prosegue riportando a zero tutti i bit A, ma senza guardare ulteriormente il contenuto della lista LRU, salvo introdurre in testa la nuova pagina.

7.18 Il meccanismo di protezione per blocchi di memoria contigui può essere il seguente. Gli indirizzi vengono generati relativamente al registro di CPU RB. Indicando con EA il valore dell'*Effective Address* il valore calcolato attraverso i campi dell'istruzione ed eventualmente altri registri, l'indirizzo fisico della cella indirizzata è dato da RB+EA. La protezione di memoria deve verificare che il valore corrispondente cada nello spazio di memoria assegnato al processo.

Si indichino con RL e RH due registri di CPU che, quando un dato processo viene messo in esecuzione, vengono caricati con l'indirizzo della posizione iniziale e con l'indirizzo della posizione finale dello spazio occupato in memoria dal processo stesso. Il controllo della protezione deve perciò verificare – ad ogni indirizzamento in memoria – la condizione RL$\leq$RB+EA$\leq$RH.