mento a una pagina. Poiché la tabella delle pagine invertita è ordinata per indirizzi fisici, mentre le ricerche si fanno per indirizzi virtuali, per trovare una corrispondenza occorre esaminare tutta la tabella; questa ricerca richiede molto tempo. Per limitare l'entità del problema si può impiegare una tabella hash (come si descrive nel Paragrafo 8.5.2), che riduce la ricerca a un solo, o a pochi, elementi della tabella delle pagine. Naturalmente, ogni accesso alla tabella hash aggiunge al procedimento un riferimento alla memoria, quindi un riferimento alla memoria virtuale richiede almeno due letture della memoria reale: una per l'elemento della tabella hash e l'altro per la tabella delle pagine. Per migliorare le prestazioni, la ricerca si effettua prima nella TLB, quindi si consulta la tabella hash.

Nei sistemi che adottano le tabelle delle pagine invertite, l'implementazione della memoria condivisa è difficoltosa. Difatti, la condivisione si realizza solitamente tramite indirizzi virtuali multipli (uno per ogni processo che partecipa alla condivisione) associati a un unico indirizzo fisico. Il metodo è pero inutilizzabile in presenza di tabelle invertite, perché, essendovi un solo elemento indicante la pagina virtuale corrispondente a ogni pagina fisica, questa non può avere più di un indirizzo virtuale associato. Una semplice tecnica per superare il problema consiste nel porre nella tabella delle pagine una sola associazione fra un indirizzo virtuale e l'indirizzo fisico condiviso; ciò comporta un errore dovuto all'assenza della pagina per ogni riferimento agli indirizzi virtuali non associati (*page fault*).

8.6 Segmentazione

Un aspetto importante della gestione della memoria, inevitabile alla presenza della paginazione, è quello della separazione tra la visione della memoria dell'utente e l'effettiva memoria fisica. Lo spazio d'indirizzi *visto* dall'utente non coincide con l'effettiva memoria fisica, ma lo si fa corrispondere alla memoria fisica. I metodi che stabiliscono questa corrispondenza consentono di separare la memoria logica dalla memoria fisica.

8.6.1 Metodo di base

Ci si potrebbe chiedere se l'utente può considerare la memoria come un vettore lineare di byte, alcuni dei quali contengono istruzioni e altri dati. Molti risponderebbero di no. Gli utenti la vedono piuttosto come un insieme di segmenti di dimensione variabile non necessariamente ordinati (Figura 8.18).

La tipica struttura di un programma con cui i programmatori hanno familiarità è costituita di una parte principale e di un gruppo di procedure, funzioni o moduli, insieme con diverse strutture dati come tabelle, matrici, pile, variabili e così via. Ciascuno di questi moduli o elementi di dati si identifica con un nome: "tabella dei simboli", "funzione `sqrt()`", "programma principale", indipendentemente dagli indirizzi che questi elementi occupano in memoria. Non è necessario preoccuparsi del fatto che la tabella dei simboli sia memorizzata prima o dopo la funzione `sqrt()`. Ciascuno di questi segmenti ha una lunghezza variabile, definita intrinsecamente dallo scopo che il segmento stesso ha all'interno del programma. Gli elementi che si trovano all'interno di un segmento sono identificati dal loro scostamento, misurato dall'inizio del segmento: la prima istruzione del programma, il settimo elemento della tabella dei simboli, la quinta istruzione della funzione `sqrt()`, e così via.

La **segmentazione** è uno schema di gestione della memoria che consente di gestire questa rappresentazione della memoria dal punto di vista dell'utente. Uno spazio d'indirizzi

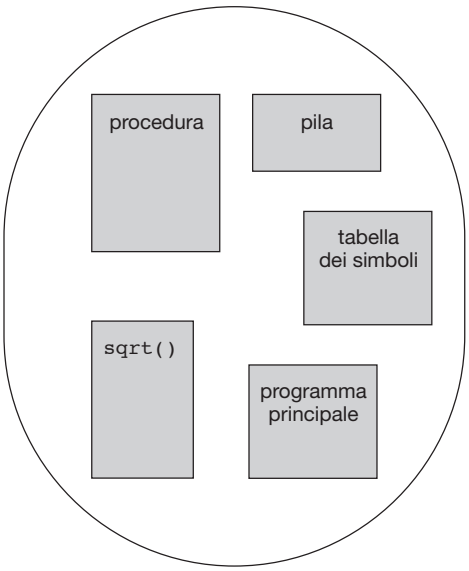

Figura 8.18 Un programma dal punto di vista dell'utente.

logici è una raccolta di segmenti, ciascuno dei quali ha un nome e una lunghezza. Gli indirizzi specificano sia il nome sia lo scostamento all'interno del segmento, quindi l'utente fornisce ogni indirizzo come una coppia ordinata di valori: un nome di segmento e uno scostamento. Questo schema contrasta con la paginazione, in cui l'utente fornisce un indirizzo singolo, che l'architettura di paginazione suddivide in un numero di pagina e uno scostamento, non visibili dal programmatore.

Per semplicità i segmenti sono numerati, e ogni riferimento si compie per mezzo di un numero anziché di un nome; quindi un indirizzo logico è una *coppia*

<*numero di segmento*, *scostamento*>

Normalmente il programma utente è stato compilato, e il compilatore struttura automaticamente i segmenti secondo il programma sorgente. Un compilatore per il linguaggio C può creare segmenti distinti per i seguenti elementi di un programma:

1. il codice;
2. le variabili globali;
3. lo heap, da cui si alloca la memoria;
4. le pile usate da ciascun thread;
5. la libreria standard del C.

Alle librerie collegate dal linker al momento della compilazione possono essere assegnati dei nuovi segmenti. Il caricatore preleva questi segmenti e assegna loro i numeri di segmento.

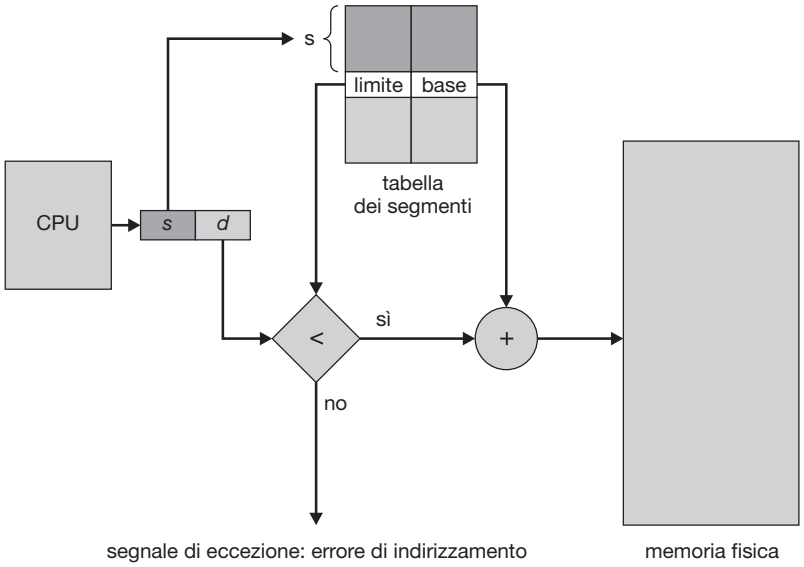

Figura 8.19 Architettura di segmentazione.

8.6.2 Architettura di segmentazione

Sebbene l'utente possa far riferimento agli oggetti del programma per mezzo di un indirizzo bidimensionale, la memoria fisica è in ogni caso una sequenza di byte unidimensionale. Per questo motivo occorre tradurre gli indirizzi bidimensionali definiti dall'utente negli indirizzi fisici unidimensionali. Questa operazione si compie tramite una **tabella dei segmenti**; ogni suo elemento è una coppia ordinata: la *base del segmento* e il *limite del segmento*. La base del segmento contiene l'indirizzo fisico iniziale della memoria dove il segmento risiede, mentre il limite del segmento contiene la lunghezza del segmento.

L'uso della tabella dei segmenti è illustrato nella Figura 8.19. Un indirizzo logico è formato da due parti: un numero di segmento *s* e uno scostamento in tale segmento *d*. Il numero di segmento si usa come indice per la tabella dei segmenti; lo scostamento *d* dell'indirizzo logico deve essere compreso tra 0 e il limite del segmento, altrimenti s'invia un segnale di eccezione al sistema operativo (tentativo di indirizzamento logico oltre la fine del segmento). Se tale condizione è rispettata, si somma lo scostamento alla base del segmento per produrre l'indirizzo della memoria fisica dove si trova il byte desiderato. Quindi la tabella dei segmenti è fondamentalmente un vettore di coppie di registri di base e limite.

Come esempio si può considerare la situazione illustrata nella Figura 8.20. Sono dati cinque segmenti numerati da 0 a 4, memorizzati in memoria fisica. La tabella dei segmenti ha un elemento distinto per ogni segmento, indicante l'indirizzo iniziale del segmento in memoria fisica (la base) e la lunghezza di quel segmento (il limite). Per esempio, il segmento 2 è lungo 400 byte e inizia alla locazione 4300, quindi un riferimento al byte 53 del segmento 2 si fa corrispondere alla locazione 4300 + 53 = 4353. Un riferimento al segmento 3, byte 852, si fa corrispondere alla locazione 3200 (la base del segmento 3) + 852 = 4052. Un riferimento al byte 1222 del segmento 0 causa l'invio di un segnale di eccezione al sistema operativo, poiché questo segmento è lungo 1000 byte.

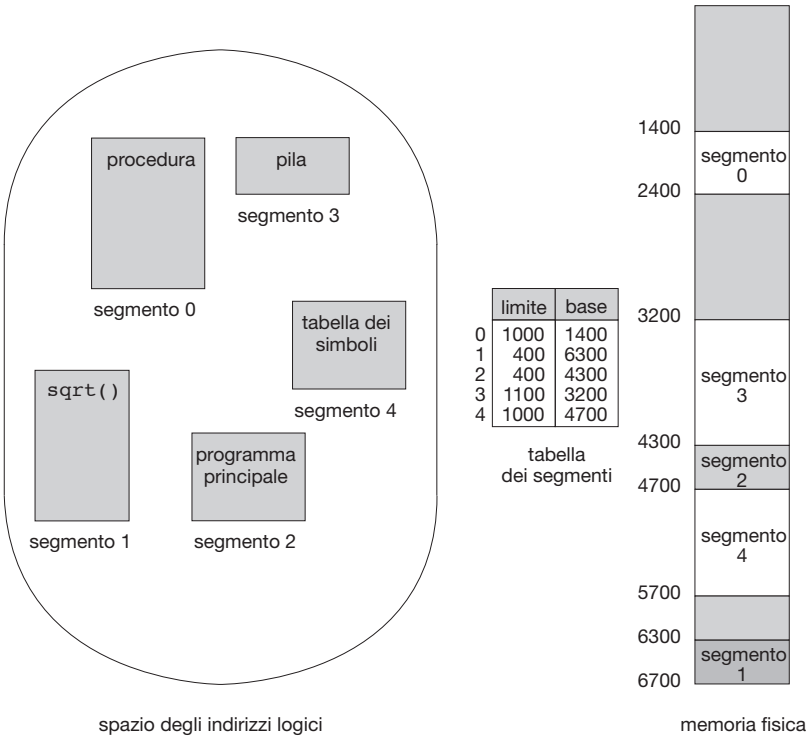

Figura 8.20 Esempio di segmentazione.

8.7 Un esempio: Pentium Intel

Paginazione e segmentazione presentano vantaggi e svantaggi, tanto che alcune architetture dispongono di entrambe le tecniche. In questo paragrafo prendiamo in esame l'architettura del Pentium Intel che utilizza, oltre alla segmentazione pura, la segmentazione mista a paginazione. Non ci inoltreremo in una trattazione completa della gestione della memoria del Pentium, ma ci limiteremo a illustrare i concetti fondamentali su cui è basata. La nostra analisi si concluderà con una panoramica della traduzione degli indirizzi di Linux nei sistemi Pentium.

In questi sistemi la CPU genera indirizzi logici, che confluiscono nell'unità di segmentazione. Questa produce un indirizzo lineare per ogni indirizzo logico. L'indirizzo lineare passa quindi all'unità di paginazione, la quale, a sua volta, genera l'indirizzo fisico all'interno della memoria centrale. Così, le unità di segmentazione e di paginazione formano un equivalente dell'unità di gestione della memoria (MMU). Il modello è rappresentato nella Figura 8.21.

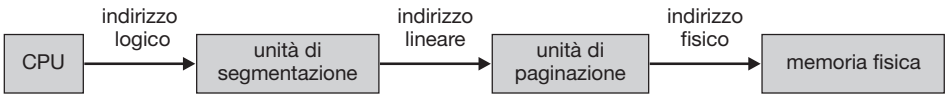

Figura 8.21 Traduzione degli indirizzi logici in indirizzi fisici in Pentium.