8.6 Segmentación

Un aspecto importante de la gestión de memoria que se volvió inevitable con los mecanismos de paginación es la separación existente entre la vista que el usuario tiene de la memoria y la memoria física real. Como ya hemos comentado, la vista que el usuario tiene de la memoria no es la misma que la memoria física real, sino que esa vista del usuario se mapea sobre la memoria física. Este mapeo permite la diferenciación entre memoria lógica y memoria física.

8.6.1 Método básico

¿Piensan los usuarios en la memoria en forma de una matriz lineal de bytes, algunos de las cuales contienen instrucciones, mientras que otros contienen datos? La mayoría de nosotros diríamos que no, en lugar de ello, los usuarios prefieren ver la memoria como una colección de segmentos de tamaño variable, sin que exista necesariamente ninguna ordenación entre dichos segmentos (Figura 8.18).

Considere la forma en que pensamos en un programa en el momento de escribirlo. Tendemos a considerarlo como un programa principal con un conjunto de métodos, procedimientos o funciones y que también puede incluir diversas estructuras de datos (objetos, matrices, pilas, variables, etc.). Para referirnos a cada uno de estos módulos o elementos de datos, utilizamos su nombre. Hablamos acerca de "la pila", "la biblioteca matemática", "el programa principal", etc., sin preocuparnos de las direcciones de memoria que estos elementos puedan apuntar. No nos importa si la pila está almacenada antes o después de la función `Sqrt( )`. Cada uno de estos segmentos tiene una longitud variable, que está definida intrínsecamente por el segmento que cumpla ese segmento del programa. Los elementos dentro de un segmento están identificados por su desplazamiento con respecto al inicio del segmento: la primera instrucción del programa, la séptima entrada de la pila, la quinta instrucción de la función `Sqrt( )`, etc.

La **segmentación** es un esquema de gestión de memoria que soporta esta visión de la memoria que tienen los usuarios. Un espacio lógico de direcciones es una colección de segmentos y cada segmento tiene un nombre y una longitud. Las direcciones especifican tanto el nombre del segmento como el desplazamiento dentro de ese segmento. El usuario especifica, por tanto, cada dirección proporcionando dos valores: un nombre de segmento y un desplazamiento (compare

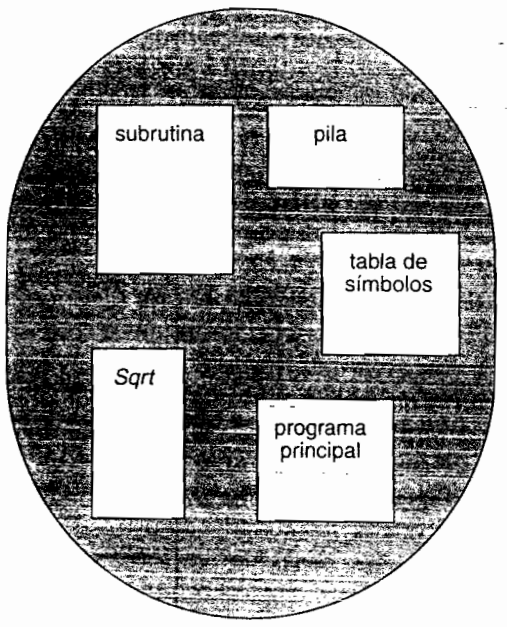

Figura 8.18 Vista de un programa por parte de un usuario.

este esquema con el de paginación, en el que el usuario especificaba una única dirección, que hardware particionaba en un número de páginas y en un desplazamiento, de forma invisible pa el programador.

Por simplicidad de implementación, los segmentos están numerados y se hace referencia ellos mediante un número de segmento, en lugar de utilizar un nombre de segmento. Así, u dirección lógica estará compuesta por una pareja del tipo:

<número-segmento, desplazamiento>.

Normalmente, el programa del usuario se compila y el compilador construye automáticame te los segmentos para reflejar el programa de entrada.

Un compilador C, podría crear segmentos separados para los siguientes elementos:

1. El código.
2. Las variables globales.
3. El cúmulo de memoria a partir del cual se asigna la memoria.
4. Las pilas utilizadas por cada hebra de ejecución.
5. La biblioteca C estándar.

También pueden asignarse segmentos separados a las bibliotecas que se monten en tiempo d compilación. El cargador tomará todos estos segmentos y les asignará los correspondientes núme ros de segmento.

8.6.2 Hardware

Aunque el usuario puede ahora hacer referencia a los objetos del programa utilizando una direc ción bidimensional, la memoria física real continúa siendo, por supuesto, una secuencia unidi mensional de bytes. Por tanto, deberemos definir una implementación para mapear las direcciones bidimensionales definidas por el usuario sobre las direcciones físicas unidimensiona les. Este mapeo se lleva a cabo mediante una tabla de segmentos. Cada entrada de la tabla de seg mentos tiene una *dirección base del segmento* y un *límite del segmento*. La dirección base del segmento

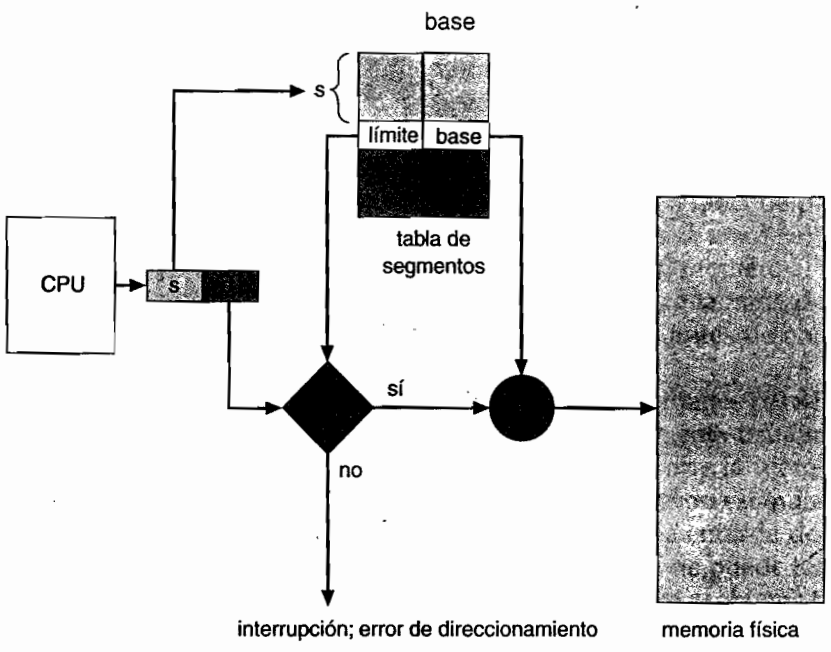

Figura 8.19 Hardware de segmentación.

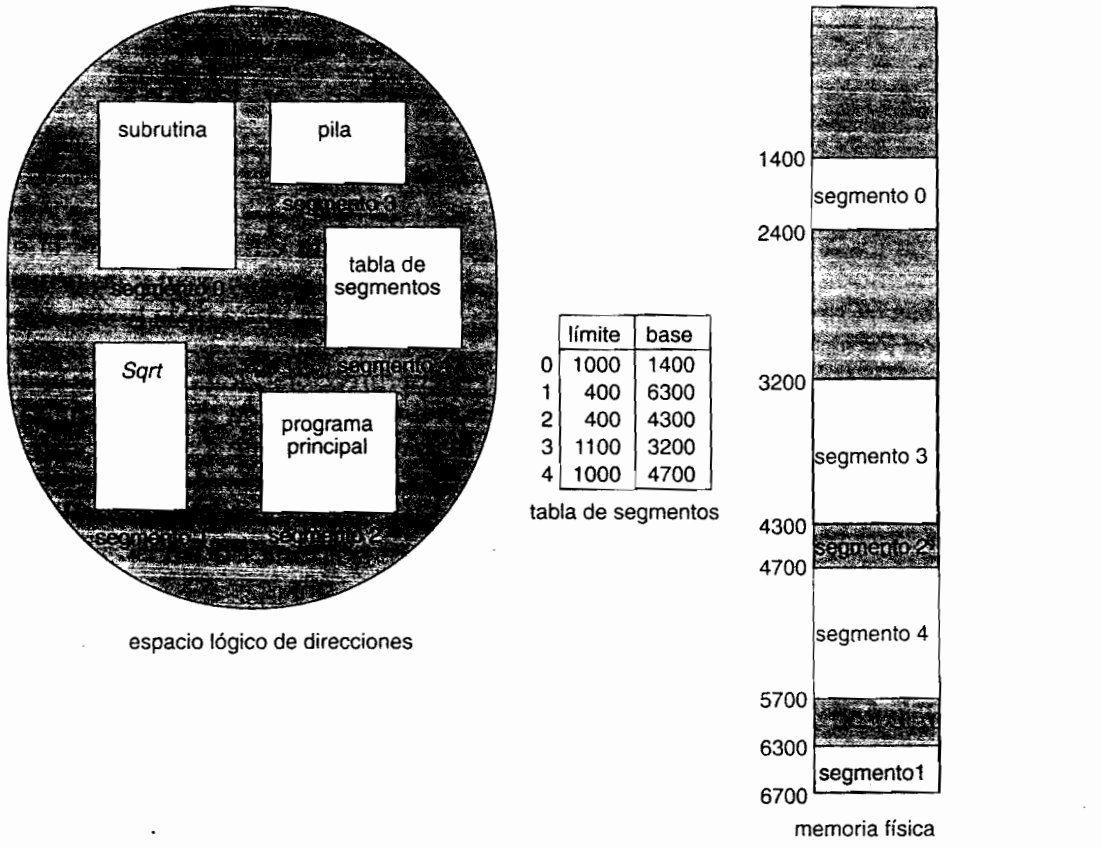

Figura 8.20 Ejemplo de segmentación.

contiene la dirección física inicial del lugar donde el segmento reside dentro de la memoria, mientras que el límite del segmento especifica la longitud de éste.

El uso de una tabla de segmentos se ilustra en la Figura 8.19. Una dirección lógica estará compuesta de dos partes: un número de segmento, *s*, y un desplazamiento dentro de ese segmento, *d*. El número de segmento se utiliza como índice para la tabla de segmentos. El desplazamiento *d* de la dirección lógica debe estar comprendido entre 0 y el límite del segmento; si no lo está, se producirá una interrupción hacia el sistema operativo (intento de direccionamiento lógico más allá del final del segmento). Cuando un desplazamiento es legal, se lo suma a la dirección base del segmento para generar la dirección de memoria física del byte deseado. La tabla de segmentos es, por tanto, esencialmente una matriz de parejas de registros base-límite.

Como ejemplo, considere la situación mostrada en la Figura 8.20. Tenemos cinco segmentos, numerados de 0 a 4. Los segmentos se almacenan en la memoria física de la forma que se muestra. La tabla de segmentos dispone de una tabla separada para cada segmento, en la que se indica la dirección inicial del segmento en la memoria física (la base) y la longitud de ese segmento (el límite). Por ejemplo, el segmento 2 tiene 400 bytes de longitud y comienza en la ubicación 4300. Por tanto, una referencia al byte 53 del segmento 2 se corersponderá con la posición 4300 + 53 = 4353. Una referencia al segmento 3, byte 852, se corresponderá con la posición 3200 (la base del segmento 3) + 852 = 4052. Una referencia al byte 1222 del segmento 0 provocaría una interrupción hacia el sistema operativo, ya que este segmento sólo tiene 1000 bytes de longitud.

8.7 Ejemplo: Intel Pentium

Tanto la paginación como la segmentación tienen ventajas y desventajas. De hecho, algunas arquitecturas proporcionan ambos mecanismos. En esta sección, vamos a analizar la arquitectura Intel Pentium, que soporta tanto una segmentación pura como una segmentación con paginación. No