Utilización de la memoria

Los sistemas basados en procesadores MIPS suelen dividir la memoria en tres partes (véase figura B.5.1). La primera parte, junto al extremo inferior del espacio de direcciones (a partir de la dirección 400000_{hex}), es el *segmento de texto*, que contiene las instrucciones del programa.

Datos estáticos: parte de la memoria que contiene los datos cuyo tamaño es conocido por el compilador y cuya vida útil es la duración de la ejecución entera del programa.

La segunda parte, sobre el segmento de texto, es el *segmento de datos*, que se divide en dos partes. La parte de **datos estáticos** (a partir de la dirección 10000000_{hex}) contiene objetos cuyo tamaño es conocido por el compilador y cuya vida útil —el intervalo durante el cual un programa puede acceder a ellos— es el tiempo que tarda en ejecutarse el programa completo. Por ejemplo, en C, las variables globales son asignadas estáticamente, ya que se pueden referenciar en cualquier momento durante una ejecución del programa. El enlazador asigna objetos estáticos a las posiciones en el segmento de datos y resuelve las referencias a estos objetos.

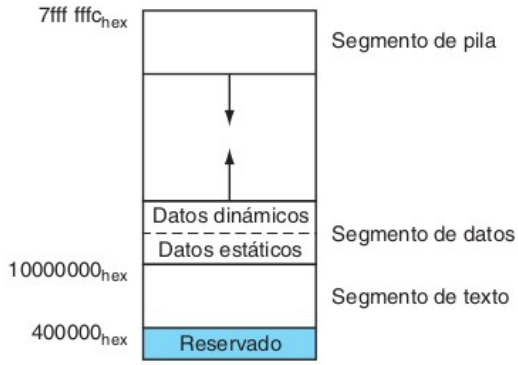

FIGURA B.5.1 Esquema de la memoria.

Inmediatamente por encima de los datos estáticos están los *datos dinámicos*. Estos datos, como su nombre indica, son asignados por el programa durante su ejecución. En programas en C, la rutina de biblioteca `malloc` busca y devuelve un bloque nuevo de memoria. Desde un compilador no se puede predecir la cantidad de memoria que asignará un programa, el sistema operativo amplía el área de datos dinámicos para satisfacer la demanda. Tal como indica en la figura la flecha hacia arriba, `malloc` amplía el área dinámica con la llamada al sistema `sbrk`, que hace que el sistema operativo añada más páginas al espacio de direcciones virtual del programa (véase sección 5.4 en el capítulo 5) inmediatamente por encima del segmento de datos dinámicos.

Segmento de pila: parte de la memoria utilizada por un programa para mantener las referencias de llamadas a procedimientos.

La tercera parte, el **segmento de pila (*stack segment*)** del programa, reside en la parte superior del espacio de direcciones virtual (empezando en la dirección $7fffffff_{hex}$). Como los datos dinámicos, el tamaño máximo de la pila de un programa no se conoce de antemano. A medida que el programa apila los valores en la pila, el sistema operativo expande el segmento de pila hacia abajo, hacia el segmento de datos.

Esta división de la memoria en tres partes no es la única posible. Sin embargo, tiene dos características importantes: los dos segmentos que se expanden dinámicamente están lo más alejados posible, y pueden crecer hasta llegar a utilizar el espacio de direcciones entero del programa.

Interfaz hardware software

Debido a que el segmento de datos comienza muy por encima del programa, en la dirección 10000000_{hex}, las instrucciones cargar y almacenar (*load* y *store*) no pueden referenciar directamente objetos de datos con su campo de desplazamiento de 16 bits (véase sección 2.5 en el capítulo 2). Por ejemplo, para cargar la palabra en el segmento de datos en la dirección 10010020_{hex} en el registro `$v0` requiere dos instrucciones:

```
lui $s0, 0x1001   # 0x1001 significa  1001 base 16
lw  $v0, 0x0020($s0) # 0x10010000 + 0x0020 = 0x10010020
```

(El *0x* delante de un número significa que es un valor hexadecimal. Por ejemplo, `0x8000` es 8000_{hex} ó 32768_{diez}).

Para evitar la repetición de la instrucción `lui` en cada *load* o *store*, los sistemas MIPS suelen dedicar un registro `($gp)` como un *puntero global* al segmento de datos estáticos. Este registro contiene la dirección 10008000_{hex}, de forma que las instrucciones cargar y almacenar puedan utilizar su campo de desplazamiento de 16 bits con signo para acceder a los primeros 64 KB del segmento de datos estático. Con este puntero global, podemos reescribir el ejemplo como una única instrucción:

```
lw $v0, 0x8020($gp)
```

Por supuesto, un registro de puntero global hace el direccionamiento a las posiciones 10000000_{hex}-10010000_{hex} más rápido que a otras posiciones dinámicas. El compilador MIPS normalmente almacena las *variables globales* en este área debido a que estas variables tienen una direcciones fijas y se ajustan mejor que otros datos globales, como las matrices.