Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Organización y gestión de la memoria

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Procesadores del Lenguaje

1. Organización de la memoria en tiempo de ejecución

La organización de la memoria depende del lenguaje, del compilador y del sistema operativo. Este último es el responsable de gestionar la memoria.
Toda referencia a una posición de memoria dentro del código debe ser relativa a la posición asignada al proceso por el sistema operativo.

2. Zona de código

- Contiene las instrucciones del programa a ejecutar, escritas en código máquina.
- El tamaño del código y su contenido se calculan en tiempo de ejecución.
- El fichero ejecutable contiene todo este código junto con información relativa al tamaño de los diferentes bloques de memoria necesitados. Esta información se usa para cargar el programa en memoria para su ejecución.
- El código se considera un bloque compacto.
- Algunos compiladores fragmentan el código en solapas (*overlays*) que son secciones de código que se cargan en memoria independientemente. El compilador decide cómo agrupar las funciones en solapas para que no realice demasiadas cargas de código durante la ejecución.

3. Memoria estática

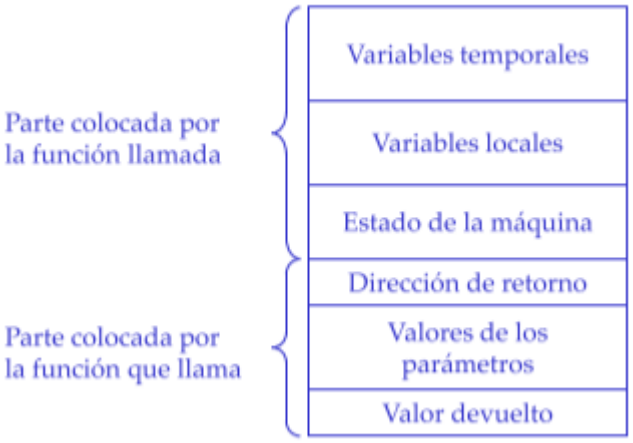
- Forma más sencilla de gestionar la memoria.
- El compilador asigna una posición de memoria fija a cada variable del programa.
- Se utiliza para almacenar constantes y variables globales.
- Se realiza en tiempo de compilación de forma consecutiva.
- La dirección es relativa al segmento de datos (DS).
- Los compiladores basados exclusivamente en memoria estática:
  - No pueden manejar funciones recursivas.
  - Cada función tiene asignado un registro de activación con los parámetros, variables locales y variables temporales de la función.
  - La memoria estática contiene las variables globales y la secuencia de registros de activación de cada función.

4. Memoria de Pila

- Maneja lenguajes estructurados con llamadas recursivas.
- Las funciones recursivas requieren manejar diferentes instancias del registro de activación almacenados en forma de pila. La cima corresponde con el registro de activación de la función en ejecución.
- Al terminar la ejecución, se desapila el registro y pasa el control a la función que se encuentre en la cima de la pila.
- Es necesario almacenar el estado de la máquina y la dirección del comienzo de registro de activación de la función llamante (*Frame Pointer*) para poder retomar el control.
- Esta información se añade al contenido del registro de activación de cada función.

4.1. Estructura del Registro de Activación

El registro de activación de cada función tiene la siguiente estructura:



4.2. Procedimiento de Llamada:

1. Se reserva espacio para el valor devuelto.
2. Se almacena el valor de los parámetros pasados a la función.
3. Almacenar el valor de la dirección de retorno.
4. Se almacena el estado de la máquina (SP y FP).
5. Se pasa el control a la función llamada.
6. Comienza la ejecución del código de la función llamada.

4.3. Procedimiento de Retorno

1. Asignar el valor devuelto.
2. Restaura el contenido del estado de la máquina. Modificando el valor de SP y FP y provocando la liberación de la memoria ocupada por el registro de activación de la función llamada.
3. Restaura el valor de la dirección de retorno.
4. Devuelve el control a la función que llama.
5. Almacena el valor devuelto en la variable local o temporal adecuada.