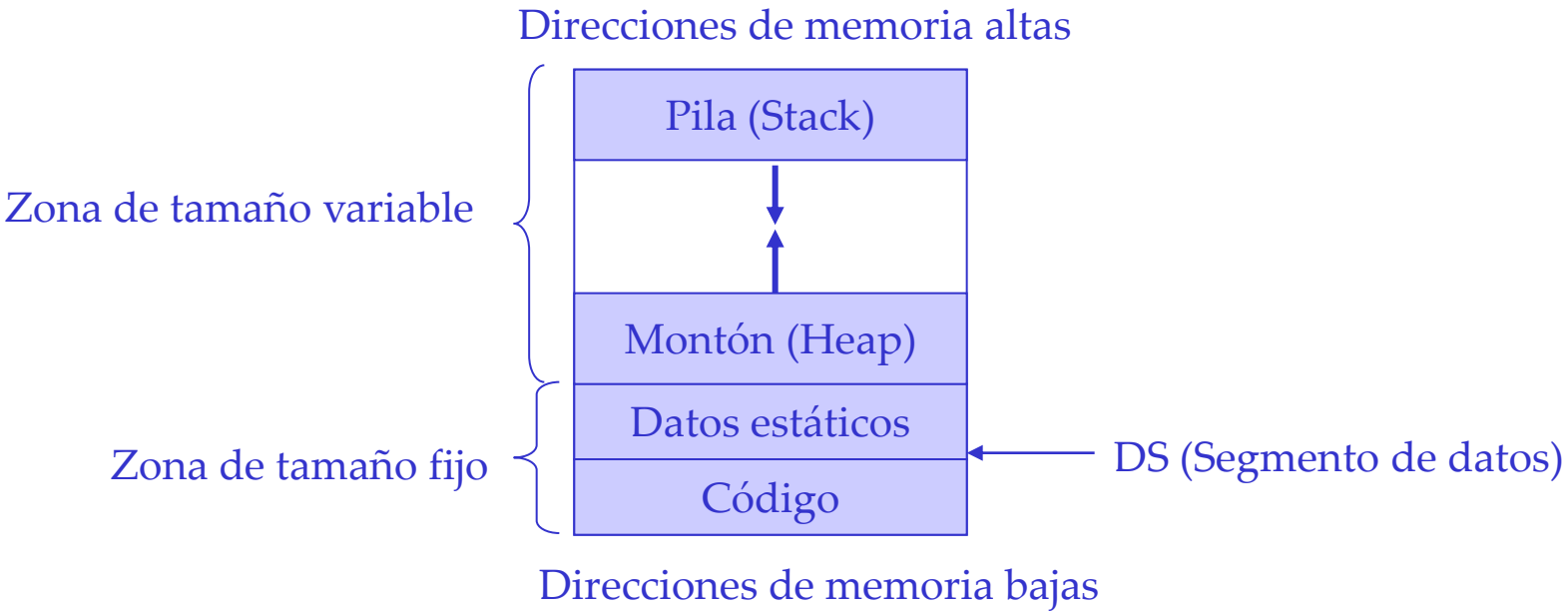
Universidad de Huelva

Tema 6
Organización y gestión de la memoria

Índice

- La organización depende del tipo de lenguaje (declarativos, imperativos), del compilador y del sistema operativo.
- Estructura general de los lenguajes imperativos (Fortran, C, C++) :



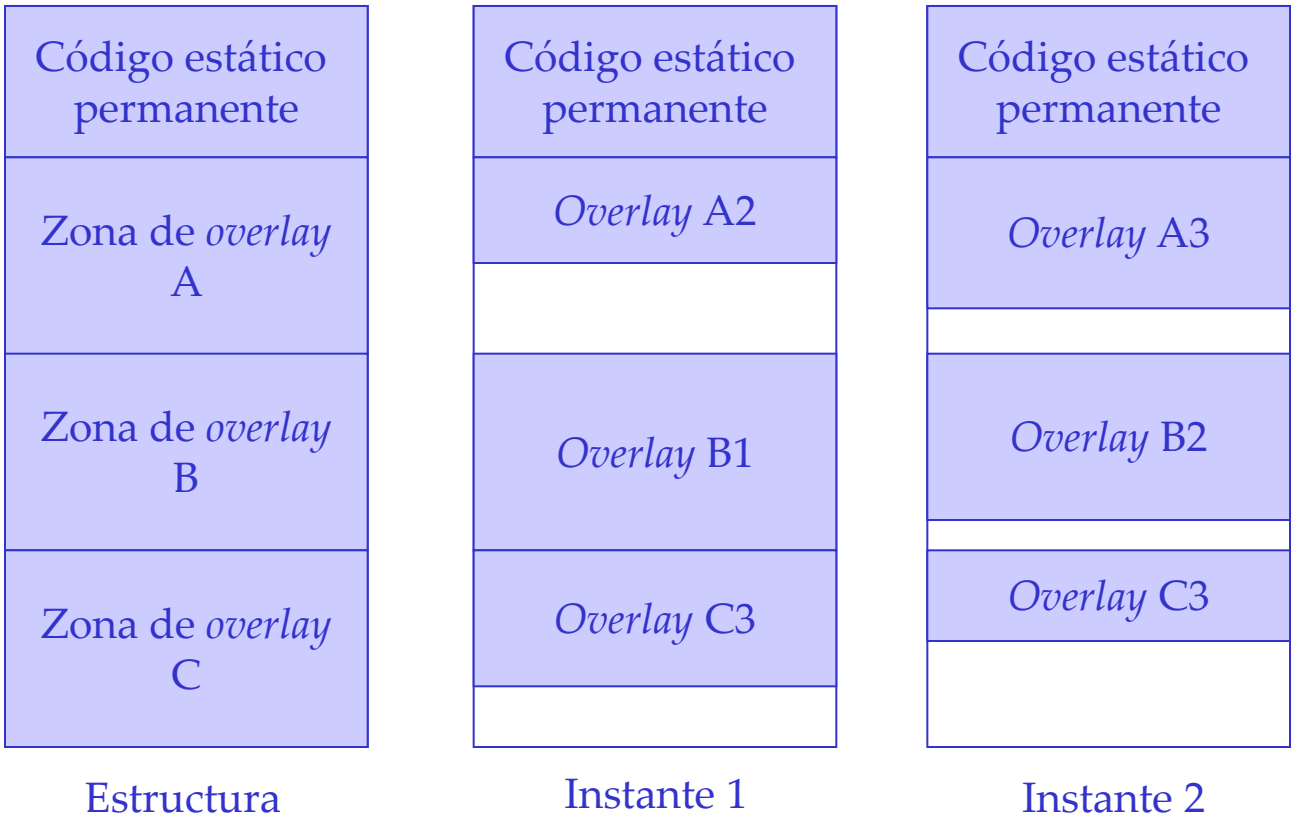
- El primer responsable de la gestión de la memoria es el sistema operativo.
- Al ejecutar un programa, un módulo del sistema operativo, llamado *cargador*, asigna la cantidad de memoria y carga el código a ejecutar (almacenado en un fichero) en la zona de código.
- El sistema operativo debe detectar la posible colisión entre el montón y la pila. En estos casos puede abortar el programa o aumentar la cantidad de memoria asignada.

- En sistemas con paginación de memoria, el esquema anterior puede estar troceado y disperso entre la memoria real del sistema y la memoria virtual.
- Toda referencia a una posición de memoria dentro del código debe ser relativa a la posición asignada al proceso por el sistema operativo.

- Contiene las intrucciones del programa a ejecutar, escritas en código máquina, es decir, la traducción a código máquina de todas los procedimientos y funciones del programa.
- El tamaño del código y su contenido se calculan en tiempo de compilación.
- El fichero ejecutable contiene todo este código, además de información relativa al tamaño de los diferentes bloques de memoria necesitados. Esta información es utilizada para cargar el programa en memoria para su ejecución.

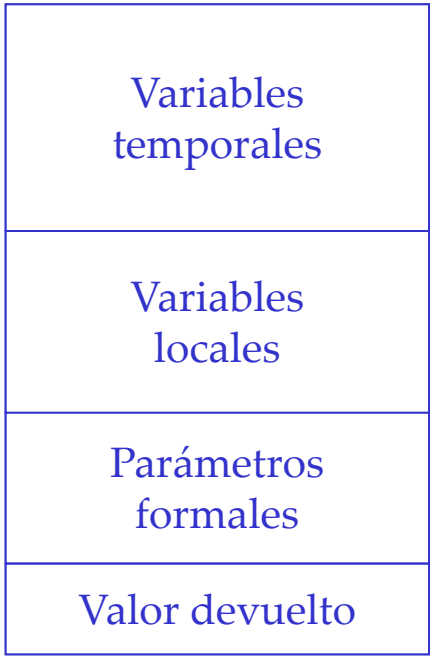
- Normalmente el código se considera un bloque compacto.
- Algunos compiladores, por el contrario, fragmentan el código en solapas (*overlays*) cuando la memoria disponible es menor que el tamaño del programa.
- Las solapas son secciones de código que se cargan en memoria de manera independiente. A lo largo del tiempo, dos solapas pueden haber ocupado la misma zona de memoria.
- El compilador debe decidir cómo agrupar las funciones en solapas de manera que no se realicen demasiadas cargas de código a lo largo de la ejecución.

- Ejemplo de un programa con varios overlays:

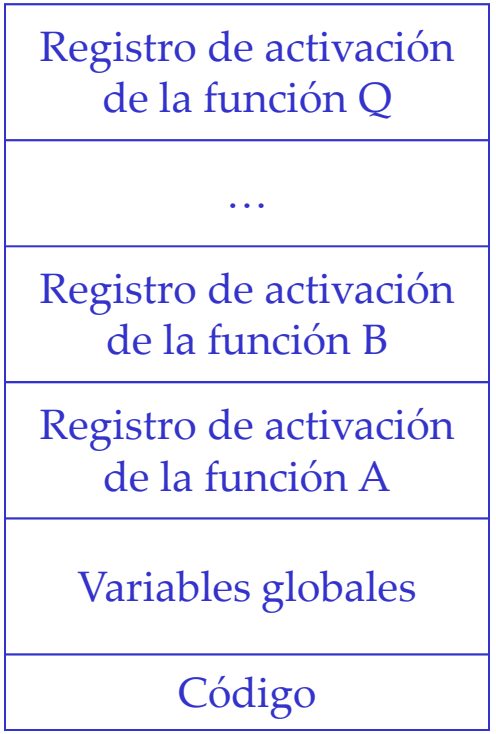


- Es la forma más sencilla de gestión de la memoria.
- El compilador asigna una posición de memoria fija a cada variable que se utilice en el programa.
- Era la única forma de gestión de memoria en los primeros compiladores (primeras versiones de FORTRAN).
- Actualmente sólo se utiliza para almacenar las constantes y variables globales del programa.
- La asignación de memoria se realiza en tiempo de compilación de forma consecutiva, teniendo en cuenta el tamaño de la variable a asignar.

- La dirección asociada a cada variable es constante y relativa al comienzo del segmento de datos (DS).
- Características de los compiladores basados exclusivamente en memoria estática:
  - no pueden manejar funciones recursivas, ya que estas requieren varias instancias de cada variable de la función.
  - cada función tiene asignado un registro de activación que contiene 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.



Estructura de un registro de activación



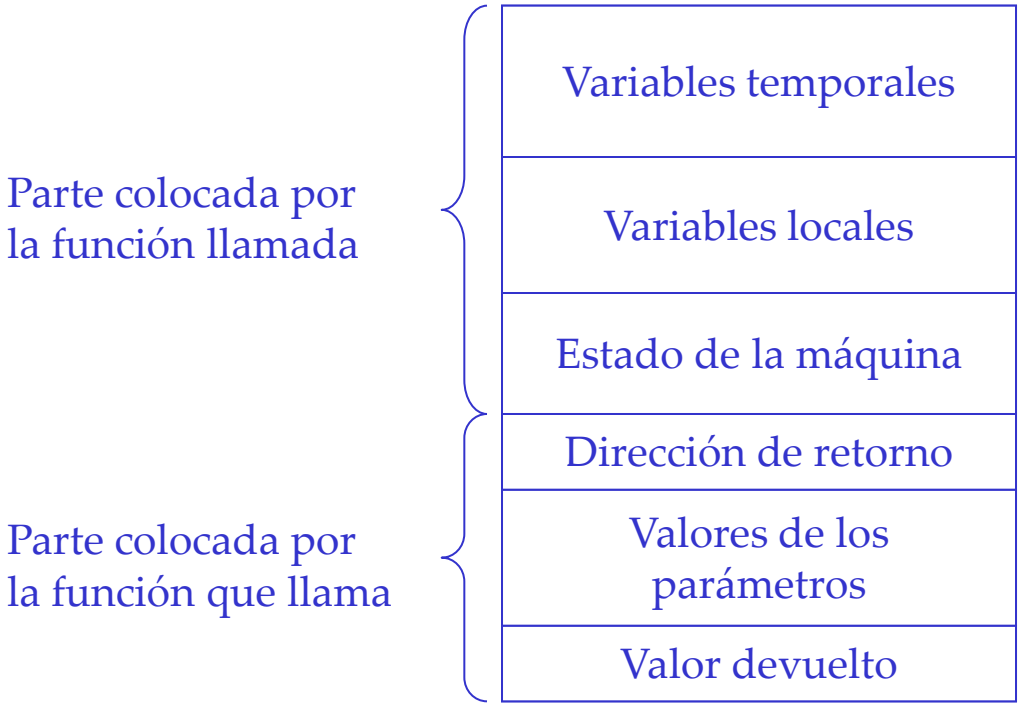
Estructura de memoria de un programa basado en memoria estática

Índice

- Se introdujo para manejar lenguajes estructurados con llamadas recursivas.
- Las funciones recursivas requieren manejar diferentes instancias del registro de activación, es decir, del conjunto de valores de sus variables en cada ejecución de la función.
- Estos registros de activación se almacenan en forma de pila, de manera que el registro de la cima de la pila corresponde a la función en ejecución.
- Al terminar la ejecución de una función debe desapilarse el registro de activación y pasar el control a la función que se encuentre en la cima de la pila.

- Al traspasar el control de una función a otra es necesario almacenar el estado de la máquina, es decir, el conjunto de valores de los registros del procesador.
- También es necesario almacenar la dirección del comienzo del registro de activación de la función llamante (*Frame Pointer*), para poder retomar el control una vez terminada la función llamada.
- Esta información se añade al contenido del registro de activación de cada función.

- Estructura de un registro de activación para un programa con gestión de memoria basada en una pila:



- El procesador maneja dos registros especiales:
  - Stack Pointer (SP): dirección de la cima de la pila.
  - Frame Pointer (FP): dirección base del registro de activación de la función activa.

- El procedimiento de llamada es el siguiente:
  - Se reserva espacio para el valor devuelto.
  - Se almacena el valor de los parámetros que se pasan a la función llamada.
  - Se almacena el valor de la dirección del código de la función que llama (dirección de retorno).
  - Se almacena el estado de la máquina (incluye SP y FP).
  - Se pasa el control a la función llamada.
  - Comienza la ejecución del código de la función llamada.

- El procedimiento de retorno es el siguiente:
  - Se asigna el valor devuelto.
  - Se restaura el contenido del estado de la máquina. Esto modifica el valor de FP y SP y provoca la liberación de la memoria ocupada por el registro de activación de la función llamada.
  - Se restaura el valor de la dirección del código de la función que llama (dirección de retorno).
  - Se devuelve el control a la función que llama.
  - Se almacena el valor devuelto en la variable local o temporal adecuada.

- Ejemplo del estado de la pila en un proceso de llamadas entre funciones.