Ambientes de Tiempo de Ejecución

Un compilador debe implementar eficientemente las abstracciones que se encuentran en la definición de un lenguaje. Estas abstracciones pueden ser:

- Nombres de variables
- Ámbitos
- Tipos de datos
- Operadores
- Procedimientos
- Parámetros
- Flujo de control

Un compilador debe de cooperar con el sistema operativo para poder implementar dichas abstracciones. Para dicha implementación, el compilador crea y administra *el ambiente de ejecución* en el cual asume que el programa se estará ejecutando.

Se trataran dos temas principales:

- Asignación de localidades de memoria
- Acceso a variables y datos.

Organización de la Memoria

Desde el punto de vista del diseñador del compilador, el programa se ejecuta en su propio espacio de direcciones de memoria, en el cual cada valor del programa tiene su dirección específica. La administración y organización de las direcciones lógicas queda a cargo del compilador, el sistema operativo y la máquina a donde se ejecuta. El sistema operativo mapea las direcciones lógicas a direcciones físicas, a donde se encuentran específicamente los datos almacenados.

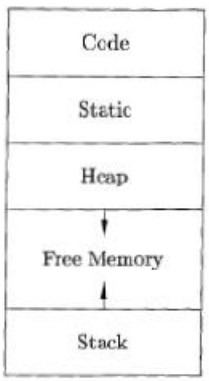

Figure 7.1: Typical subdivision of run-time memory into code and data areas

Asignación Estática (Compilación) Vs. Asignación Dinámica (Ejecución) de Memoria.

Se dice que una decisión de asignación de memoria es estática, cuando esta puede ser realizada por el compilador examinando únicamente el texto del programa. Una decisión es dinámica si solo puede ser tomada cuando el programa se encuentra en ejecución. Para la asignación dinámica de memoria se utiliza generalmente:

- Asignación en la Pila.
- Asignación en el Montículo

Asignación en la Pila

La utilización de la pila para la asignación de memoria es utilizada principalmente por los lenguajes de programación que utilizan procedimientos, métodos o funciones como unidades de programación. Cada vez que un procedimiento es llamado, espacio para las variables locales es asignado en la pila (push) y cuando este procedimiento termina el espacio es liberado de la misma (pop).

Las principales ventajas de este mecanismo son:

- El poder compartir direcciones de memoria
- Direccionamiento relativo de las variables locales sin cambio.

Árboles de Activación

La asignación de memoria en la pila es posible por la anidación (en el tiempo) de las llamadas a procedimientos, es decir si la activación de un procedimiento *p* llama a un procedimiento *q*, entonces la activación de *q* debe de terminar antes que termine la activación de *q*.

Se puede representar la activación de los procedimientos durante la ejecución de un programa por un árbol llamado *árbol de activación*, donde cada nodo corresponde a una activación y la raíz es la activación del procedimiento principal.

El uso del stack como método de asignación de memoria se basa en la relación entre este árbol de activación y el comportamiento de la ejecución del programa.

- La secuencia de llamadas corresponde a un recorrido en pre-orden. (Raíz-Izquierdo-Derecho)
- La secuencia de retorno corresponde a un recorrido en post-orden. (Izquierdo-Derecho-Raíz)
- Si el control del programa se encuentra en la activación de algún procedimiento, el cual corresponde al nodo N en un árbol de activación, entonces las activaciones que se encuentran actualmente vivas, son aquellas que corresponden al nodo N y a sus ancestros.

```
enter main()
    enter readArray()
    leave readArray()
    enter quicksort(1,9)
        enter partition(1,9)
        leave partition(1,9)
        enter quicksort(1,3)
            ...
        leave quicksort(1,3)
        enter quicksort(5,9)
            ...
        leave quicksort(5,9)
    leave quicksort(1,9)
leave main()
```

Figure 7.3: Possible activations for the program of Fig. 7.2

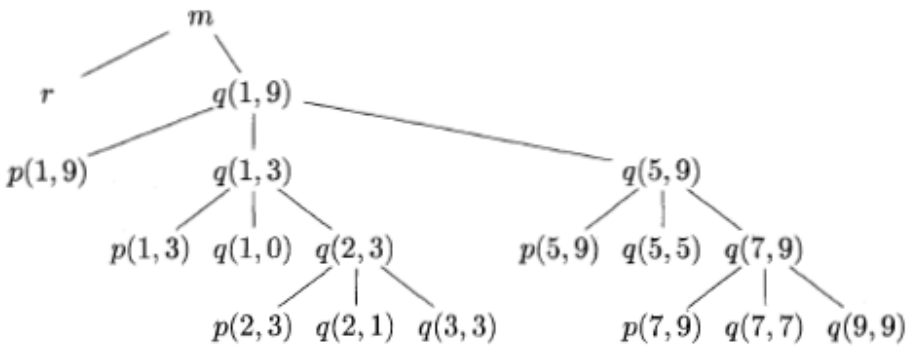

Figure 7.4: Activation tree representing calls during an execution of *quicksort*

Registros de Activación

Las llamadas y retornos de procedimientos son administrados por la *pila de control*. Cada activación *viva* tiene asignado un *registro de activación* en la pila de control, la raíz del árbol de activación en el fondo de la pila y la secuencia de registros de activación en la pila corresponden al camino en el árbol de activación hacia la activación a donde el control actualmente reside.

El contenido de los registros de activación varía con el lenguaje a ser implementado. Generalmente poseen la siguiente información:

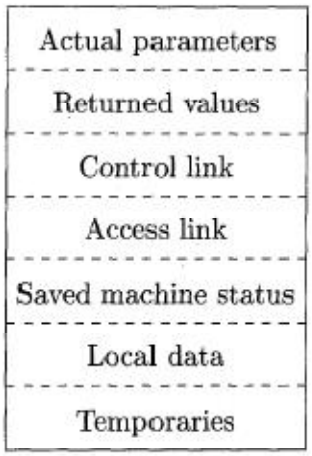

Figure 7.5: A general activation record

- **Valores temporales.** Aquellos que resulten de evaluación de expresiones y que no puedan guardarse en los registros.
- **Datos locales** (variables) que pertenecen al procedimiento.
- **Estado de la máquina.** Información acerca del estado de la máquina antes de llamar al procedimiento. Típicamente contiene la *dirección de retorno* y el contenido de los *registros* que estén siendo usados por el procedimiento que llama.
- **Enlace de acceso.** Puntero a datos que no se encuentran en el registro de activación actual. (Es decir, información en otros registros de activación)
- **Enlace de control.** Puntero al registro de activación del llamador.
- **Espacio para el valor de retorno** si es que hay.
- **Parámetros actuales** del procedimiento llamado. Estos parámetros se colocan a veces en registros por eficiencia.

Secuencia de llamadas

Las llamadas a los procedimientos son implementadas por lo que se conoce ***secuencias de llamadas***, las cuales consisten en *código* que asigna un registro de activación en la pila y asigna los valores a sus campos respectivos. Una ***secuencia de retorno*** es código similar que restaura el estado de la máquina de tal forma que el procedimiento que llama pueda continuar con su ejecución.

Consideraciones al generar secuencias de llamadas

- Los valores a comunicar (parámetros) entre el proceso llamador y el llamado son colocados generalmente en el registro de activación del llamado de tal forma que estén lo más cercano al registro de activación del llamador.
- Valores de longitud fija (enlace de control, enlace de acceso y el estado de la máquina) son colocados generalmente a la mitad del registro de activación. Si se sigue un estándar de colocación, las secuencias de llamadas y secuencias de retorno pueden ser efectuadas fácilmente.
- Valores cuya longitud no puede ser conocida al generar el código respectivo (arreglos, temporales), son colocados al final del registro de activación.
- Colocación del puntero a pila (stack pointer)

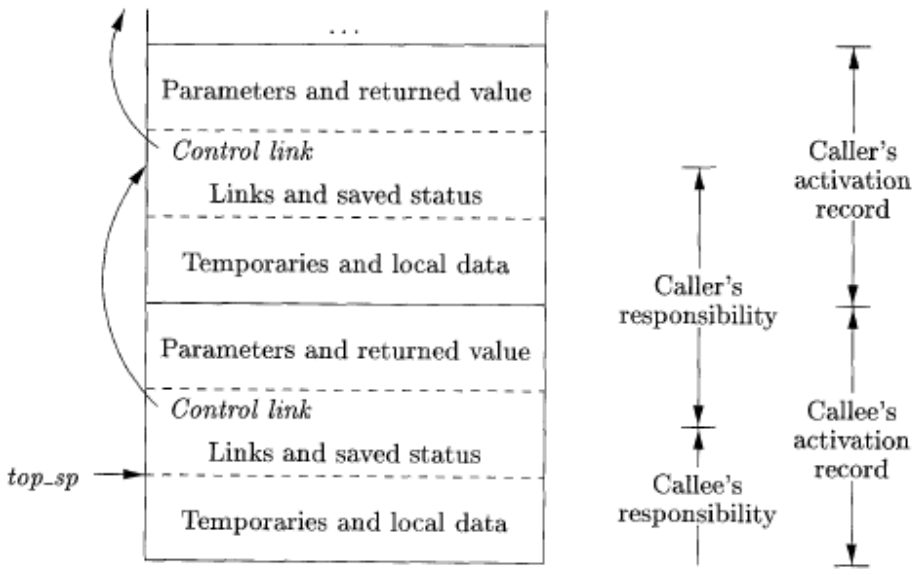

Figure 7.7: Division of tasks between caller and callee

Pasos en la secuencia de llamada

- El llamador evalúa y guarda los parámetros

- El llamador guarda la dirección de retorno en el registro de activación del llamado. Luego incrementa top_sp para almacenar el estado de la máquina.
- El llamado almacena los valores de los registros (estado de la máquina) y otra información de estado.
- El llamado inicializa sus valores locales y empieza la ejecución.

Pasos en la secuencia de retorno

- El llamado coloca el valor de retorno junto a los parámetros
- El llamado restaura el estado de la máquina y restaura el valor de *top_sp* y otros registros, luego salta a la dirección de retorno guardada el cambo de estado.
- El llamador luego utiliza el valor de retorno, si es necesario.

Datos de longitud variable en la pila.

En lenguajes modernos, los objetos cuyo tamaño no es determinado en tiempo de compilación son asignados en el *montículo*, sin embargo es posible también asignar dichos objetos en la pila. La pila es utilizada para objetos si solamente son locales para los procedimientos, ya que estos son inaccesibles una vez el procedimiento termina.

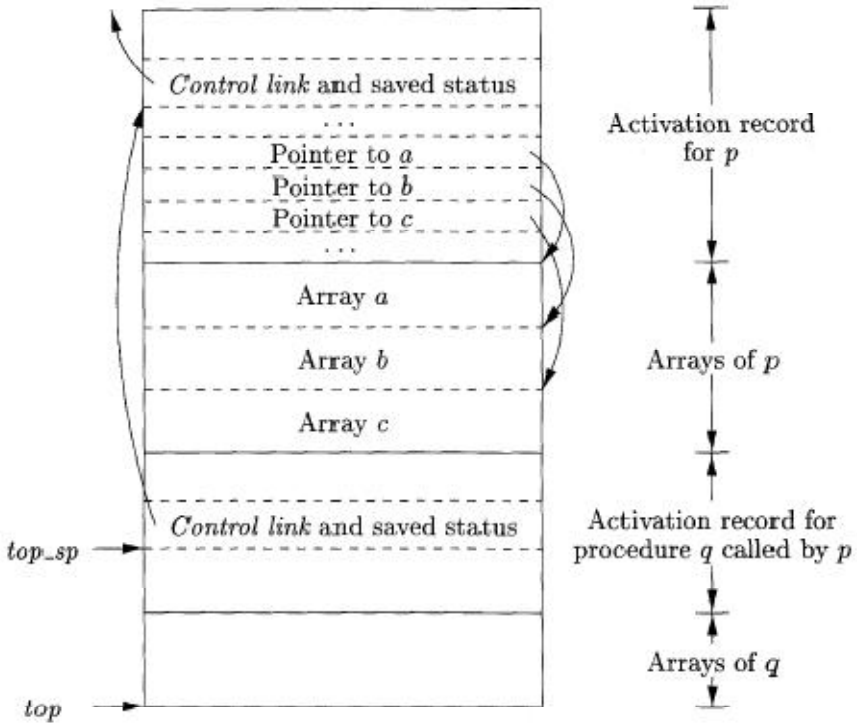

Figure 7.8: Access to dynamically allocated arrays