Subrutinas - Pasaje de parámetros - Pila

**Preguntas relacionadas**:

| 1. ¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila. 2. Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el funcionamiento y uso de la pila. 3. Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el comportamiento con anidamiento de múltiples procedimientos/funciones. |
| --- |

# **Pasaje de argumentos a procedimientos o funciones**

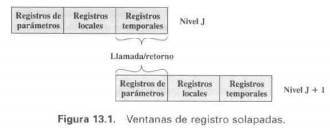
El pasaje de argumentos/parámetros a subrutinas/procedimientos/funciones puede ser de tres formas:

**Vía registros**:

* El número de registros es la principal limitación
* Es importante documentar qué registros se usan

**Ventanas de registros**: El uso de un conjunto amplio de registros debería reducir la necesidad de acceder a memoria. Un procedimiento típico emplea solo unos pocos parámetros de llamada y variables locales. Además, la profundidad de activación de procedimientos fluctúa dentro de un rango relativamente pequeño. Para explotar estas propiedades, se usan múltiples conjuntos de registros, cada uno asignado a un procedimiento distinto. Una llamada a un procedimiento hace que el procesador conmute automáticamente a una ventana de registros distinta de tamaño fijo, en lugar de salvaguardar los registros en memoria. Las ventanas de procedimientos adyacentes están parcialmente solapadas para permitir el paso de parámetros.

La ventana se divide en tres áreas de tamaño fijo. Los registros de parámetros contienen parámetros pasados al procedimiento en curso desde el procedimiento que lo llamó, y los resultados a devolver a éste. Los registros locales se usan para variables locales, según la asignación que realice el compilador. Los registros temporales se usan para intercambiar parámetros y resultados con el siguiente nivel más bajo (el procedimiento llamado por el procedimiento en curso). Los registros temporales de un nivel son físicamente los mismos que los registros de parámetros del nivel más bajo adyacente. Este solapamiento posibilita que los parámetros se pasen sin que exista una transferencia de datos real.



**Vía memoria**:

* Se usa un área definida de memoria (RAM)
* Difícil de estandarizar

**Vía pila (stack)**:

* Es el método más ampliamente usado
* El “verdadero pasaje de parámetros”
* Independiente de memoria y registros
* Hay que comprender bien cómo funciona porque la pila (stack) es usada por el usuario y por el sistema
* En x86, SP apunta al último lugar usado

# ¿Qué es una pila?

Una pila (*stack*) es un **conjunto ordenado de elementos**, en el que solo uno de ellos es accesible en un instante dado: la **cabecera** de la pila. Su **longitud es variable**. Solo se pueden añadir o eliminar elementos en la cabecera/tope de la pila (es conocida como lista LIFO).

# **¿Cómo funciona la pila?**

Operaciones:

* **PUSH**: apila/añade un nuevo elemento en la cabecera de la pila.
* **POP**: desapila/elimina el elemento de la cabecera de la pila.

Es una secuencia de dos acciones:

1. Movimiento de datos Reg-Mem o Mem-Reg
2. Modificación del puntero antes/después de la anterior

Hay que tener en cuenta:

* Dónde apunta el puntero
* Cómo crece la pila

Las **operaciones** **unarias** operan con el elemento de la cabecera (el operando está de forma implícita) y lo sustituyen por el resultado (ej: NOT).

Las **operaciones** **binarias** operan con dos elementos de la cabecera de la pila, los elimina y pone el resultado de la operación en la cabecera.

# **¿Cómo se usa la pila?**

En memoria principal se **reserva un bloque de posiciones contiguas** para la pila. Para su correcto funcionamiento se necesitan **3 direcciones**, normalmente **memorizadas en registros** de la CPU:

* **Puntero de pila (SP)**: Dirección de la cabecera de la pila. Si se añade o elimina un elemento de la pila, el mismo incrementa o decrementa para actualizarse.
* **Base de la pila (BP)**: Dirección base del bloque reservado para la pila. Si se hace un POP con la pila vacía, se informa un error.
* **Límite de la pila**: Dirección del otro extremo del bloque reservado. Si se hace un PUSH con la pila llena, se informa un error.

## **Anidamiento de múltiples funciones utilizando pila**

El comportamiento de la pila con anidamiento de procedimientos/funciones/subrutinas:

Antes de llamar a la subrutina se debe apilar los parámetros a pasar y la dirección de retorno, al llamar a la subrutina de debe:

1. **Salvar el estado de BP** (viejo BP): apilar el valor del BP
2. **Salvar el estado de SP** (BP=SP): cargar el valor del BP el del SP
3. **Reservar espacio para datos locales** (opcional): si es que los hay
4. **Salvar valores de otros registros** (opcional): si es que se van a modificar los registros
5. **Acceder a parámetros**: para accederlos se le debe sumar un desplazamiento al BP para acceder a la posición de la pila en la que estén.
   1. **En general el desplazamiento es**: 2 (es el tamaño de BP apilado) + tamaño de dirección de retorno + total de tamaño de parámetros entre el buscado y BP
6. **Escribir sentencias a ejecutar**: Aquí se puede llamar a otra subrutina o regresar a la anterior. Si se vuelve a hacer una llamada se debe apilar los parámetros a pasar a la subrutina y la dirección de retorno. La subrutina llamada debe repetir los pasos anteriores mas los que siguen. Sino se hace otra llamada a subrutina también se deberán seguir los siguientes pasos:
7. **Retornar parámetro** (opcional): si es que tiene que retornar datos
8. **Regresar correctamente del procedimiento**: desapilar todo lo que apilo, volver a cargar a cargar el valor del BP que tenía antes de entrar a la subrutina

Siempre apilo las direcciones de retorno (tantas como llamadas anidadas haya) y voy sacando en orden inverso al que fueron guardadas (con POP, siguiendo la estructura LIFO).

**Fuentes**:

* “Clase 01” (Diapositiva 47)
* “Apéndice 9A - Pilas” (Stalling 5ta ed. Pág 353)
* “Capítulo 12 - Computadores de repertorio reducido de instrucciones ” (Stalling 5ta ed. Pág 443)