El Procesador III:
Llamadas a procedimiento y uso de pila

Llamadas a procedimiento

- Un **procedimiento** (función, subrutina) **es un subprograma** que realiza una **tarea especifica** cuando se le invoca
  - Recibe argumentos o parámetros de entrada
  - Devuelve algún resultado

- En **ensamblador** un **procedimiento** se **asocia** con un **nombre simbólico** que denota su dirección de inicio.

Procedimientos

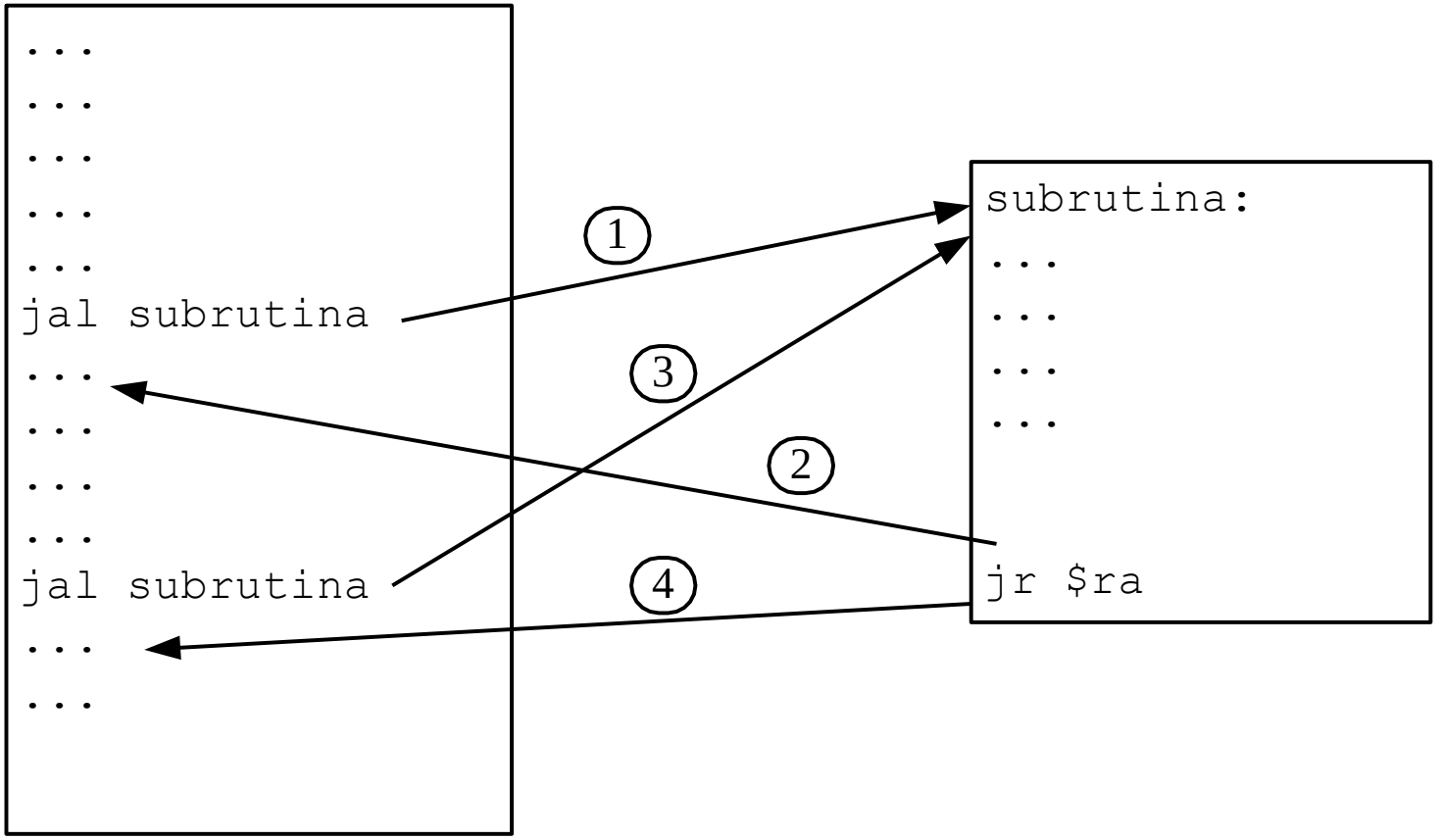
Un procedimiento puede:

1. Llamarse desde distintas posiciones.

2. Contener llamadas a otros procedimientos. Esto posibilita el anidamiento de procedimientos hasta una profundidad arbitraria.

3. Cada llamada a procedimiento está emparejada con un retorno en el programa llamado.

Secuencia de ejecución de instrucciones en presencia de llamadas a subrutinas



Procedimientos anidados

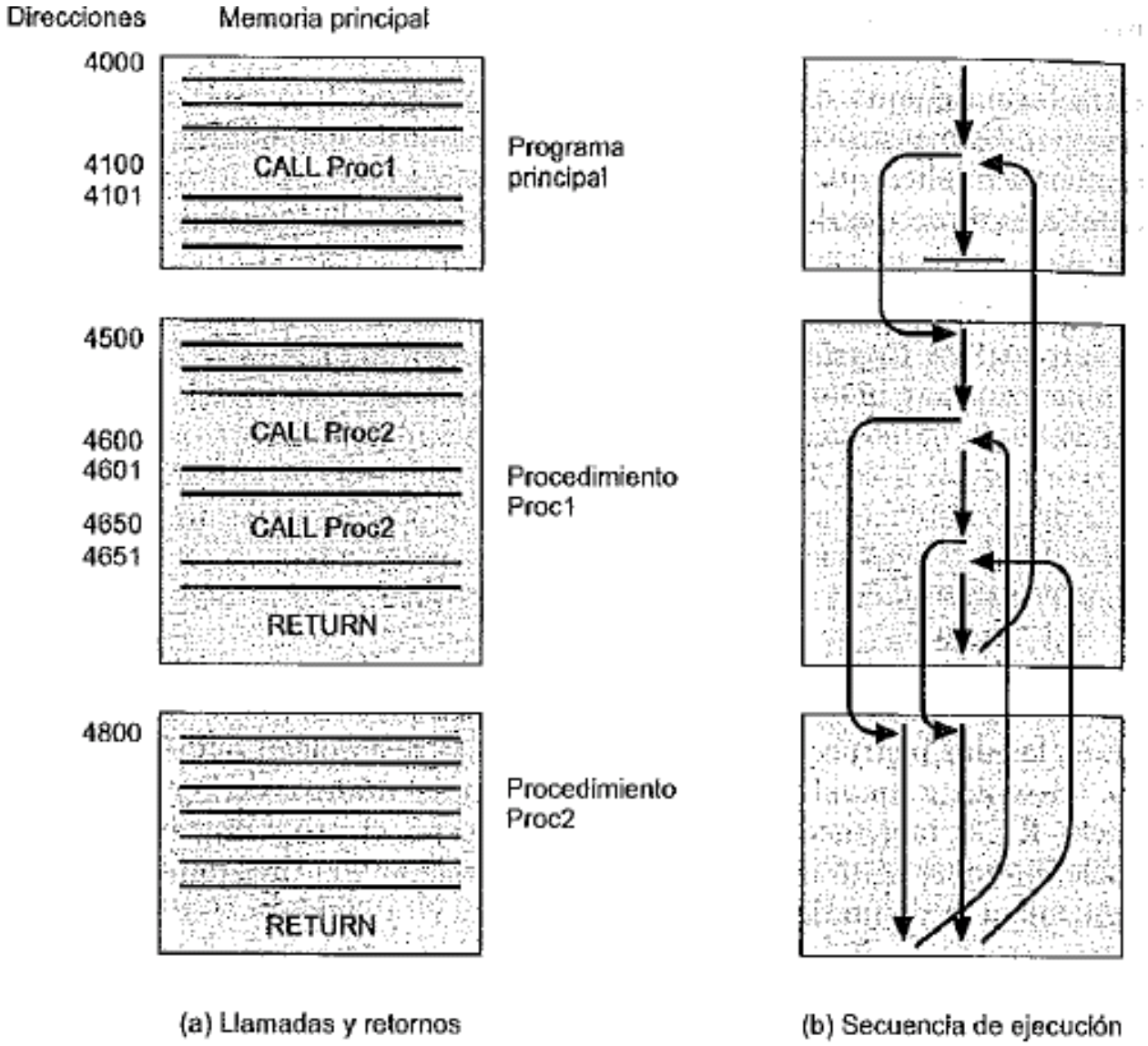
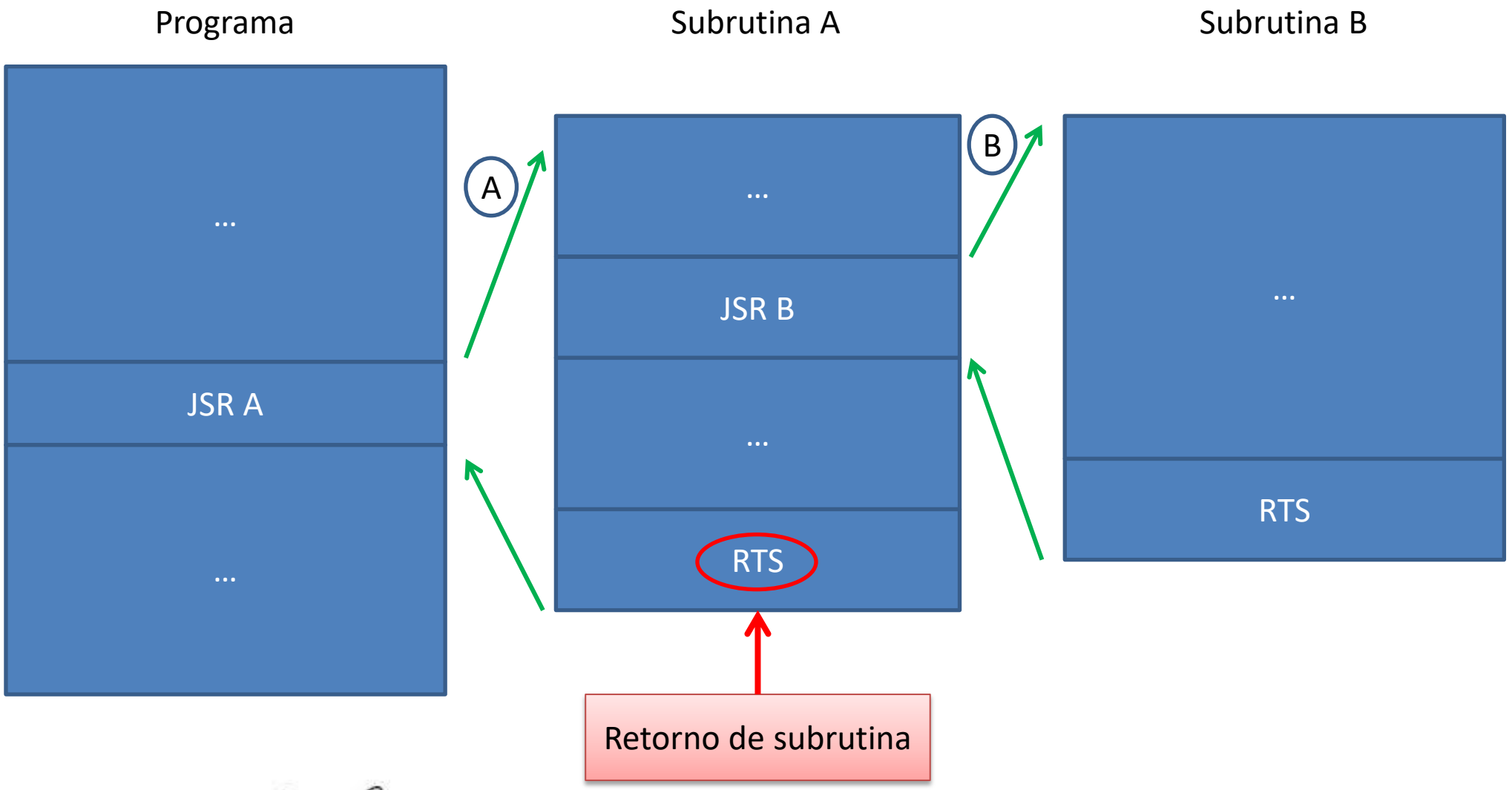


Figura 1

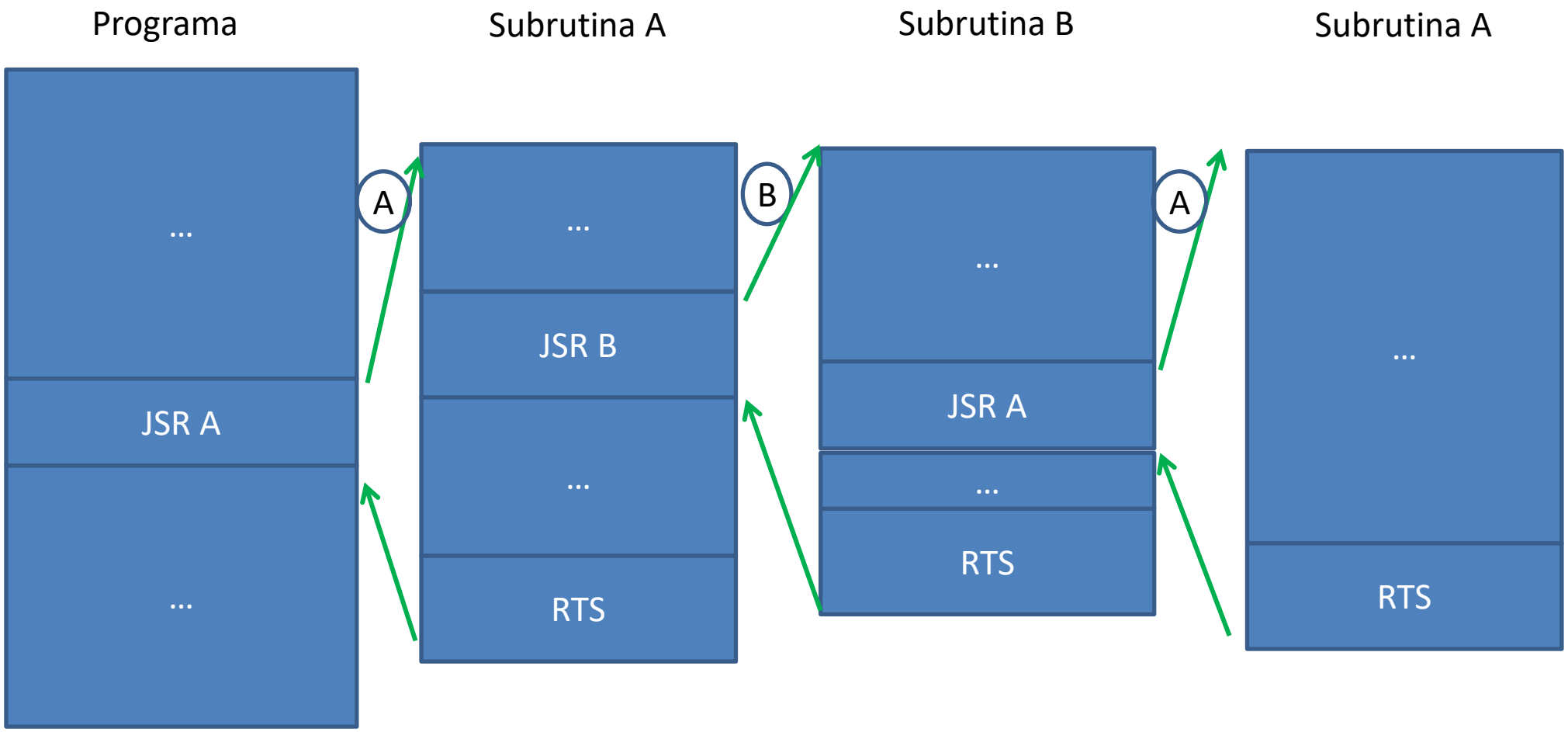
Llamadas anidadas

Una subrutina llama a otra



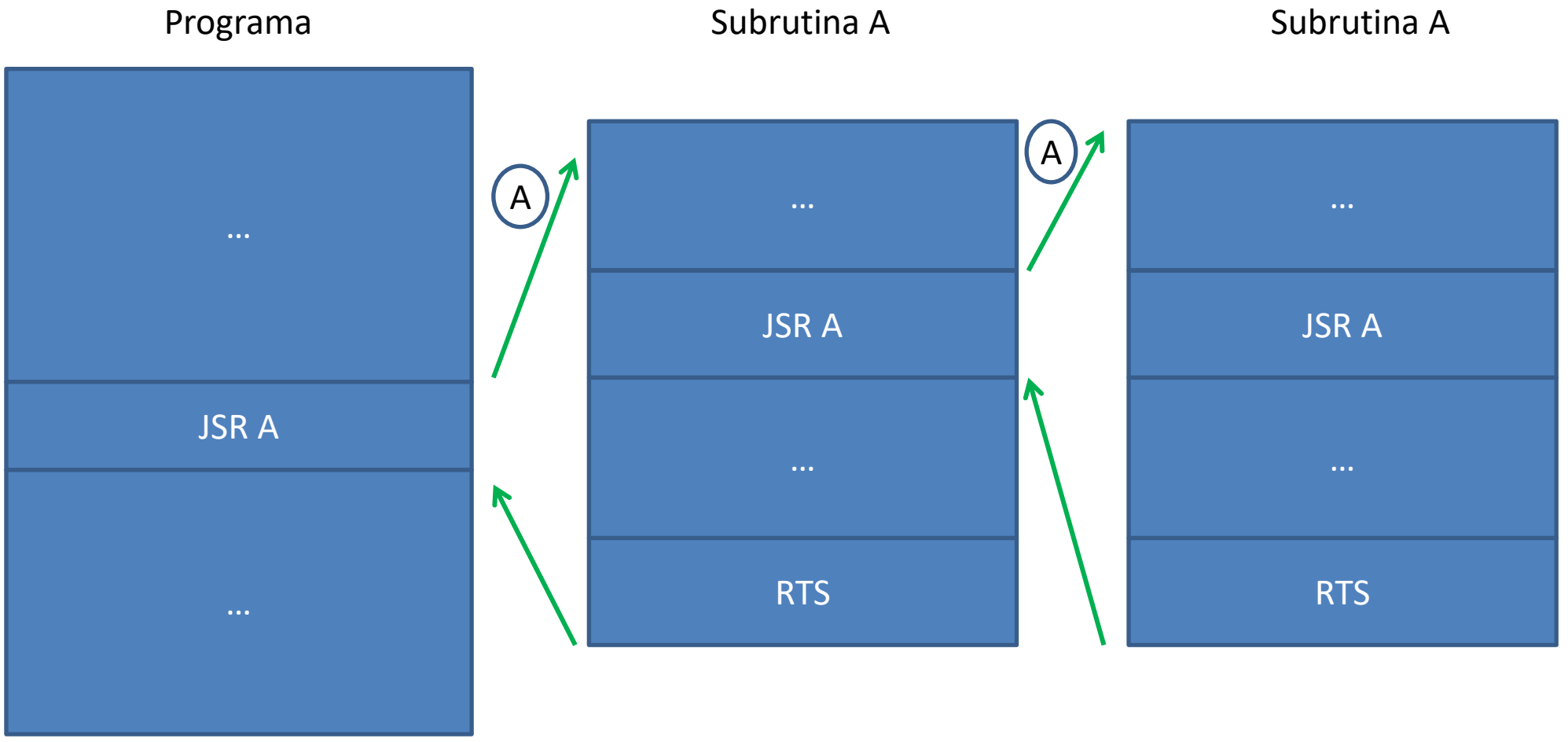
Llamadas reentrantes

Una subrutina ya activada vuelve a ser llamada



Subrutinas recursivas

Una subrutina se llama a sí misma



Procedimientos: reservar la dirección de retorno

Ya que debe permitirse que el procedimiento se llame desde distintos puntos, la CPU debe reservar la dirección de retorno en algún sitio, para que éste pueda realizarse correctamente.
Hay tres lugares habituales para guardar la dirección de retorno:

- Un registro
- Al principio del procedimiento
- En la cabeza de la pila

Procedimientos: reservar la dirección de retorno

Consideremos la instrucción en lenguaje máquina CALL X, que significa llamada al procedimiento de la posición X.
Si se utiliza un registro, la ejecución de CALL X produce las siguientes acciones:

RN ← PC + Δ
PC ← X

Donde RN es un registro que se utiliza siempre para este fin, PC es el contador de programa y Δ es la longitud de la instrucción.
El procedimiento llamado puede ahora consultar el contenido de RN para utilizarlo en el retorno posterior.

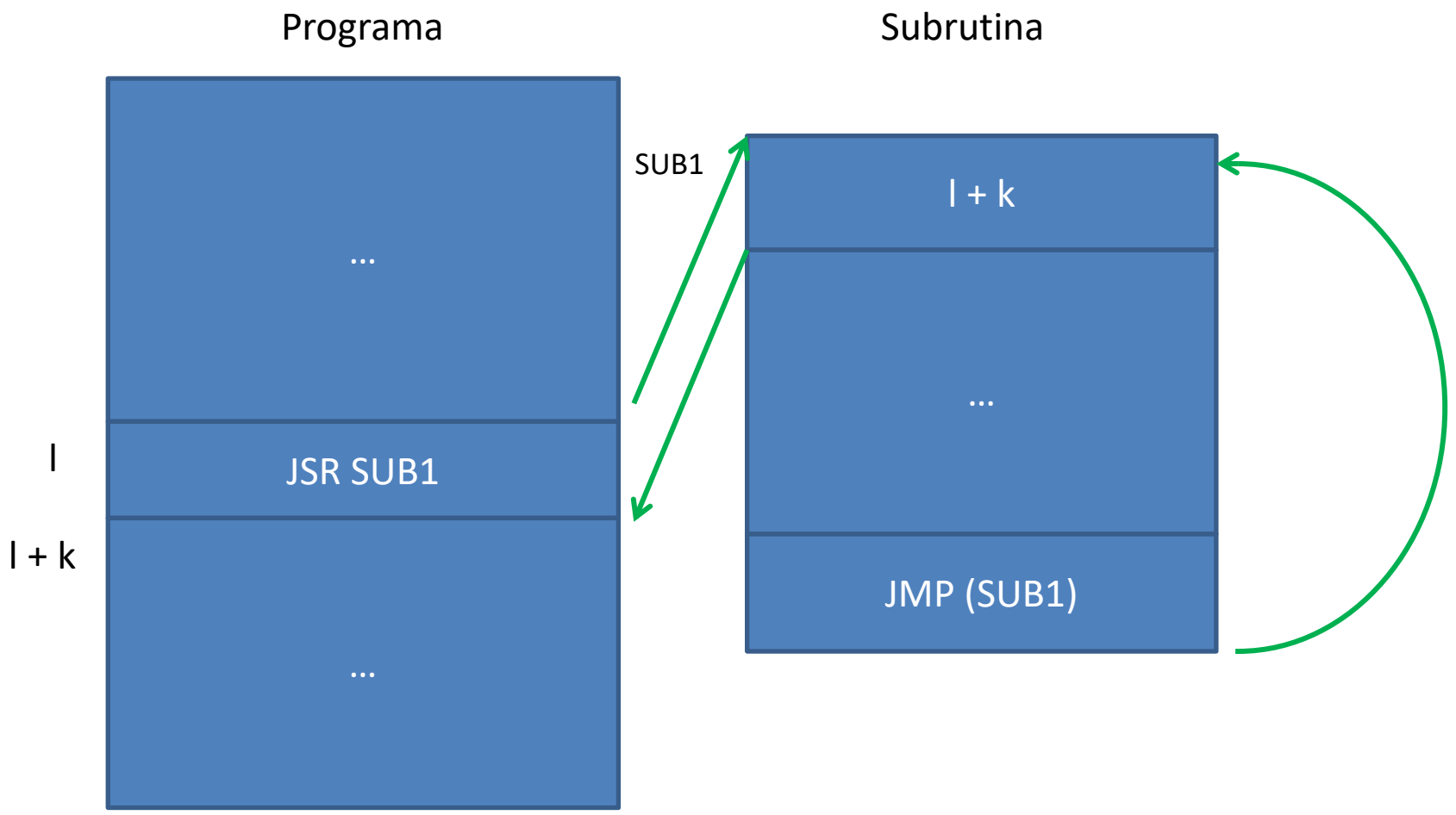
Una segunda posibilidad es almacenar la dirección de retorno al comienzo del procedimiento. En este caso, CALL X hace que:

X ← PC + Δ
PC ← X + 1

La dirección de retorno queda almacenada en un lugar seguro.

Guardando la dirección de retorno al principio de la subrutina

Consiste en reservar la primera palabra de la subrutina para almacenar la dirección de retorno



Guardando la dirección de retorno en la pila

Una pila es un conjunto ordenado de elementos en el que sólo uno de ellos es accesible en un instante dado.
El punto de acceso se denomina **cabecera de la pila**.
El número de elementos de la pila, o longitud de la pila, es variable. **Sólo se pueden añadir o eliminar elementos en la cabecera de la pila**. Por esta razón, un pila también denominada *lista último en entrar-primero-en salir (LIFO, last-in-first-out)*.

Guardando la dirección de retorno en la pila

Una operación **PUSH** (apilar) añade un nuevo elemento en la cabeza de la pila.
Una operación **POP** (desapilar) elimina el elemento de la cabecera de la pila.
Las **operaciones binarias**, que requieren de dos operandos (por ejemplo, multiplicar, dividir, sumar o restar), utilizan dos elementos de la cabecera de la pila como operandos, desapilando ambos y apilando el resultado de nuevo en la pila.
Las **operaciones unarias**, que requieren sólo un operando (por ejemplo la NOT), utilizan el elemento de la cabecera de la pila.

Operaciones sobre la pila

- Algunas arquitecturas de procesadores permiten realizar operaciones aritmético-lógicas directamente sobre la pila (p.e. Intel)
- Las instrucciones de operaciones sobre pila tienen cero operandos en la instrucción, pues implícitamente cogen los dos últimos datos de la pila como operandos, operan con ellos y los escriben en la pila

Operaciones orientadas al uso de la pila

PUSH	Añade un nuevo elemento en la cabecera de la pila
POP	Elimina el elemento de la cabecera de la pila
Operación unaria	Realiza una operación con el elemento de la cabecera de la pila. Sustituye el elemento de la cabecera con el resultado.
Operación binaria	Realiza una operación con dos elementos de la cabecera de la pila. Elimina de la pila dichos elementos. Pone el resultado de la operación de la cabecera de la pila.

Operaciones básicas de la pila

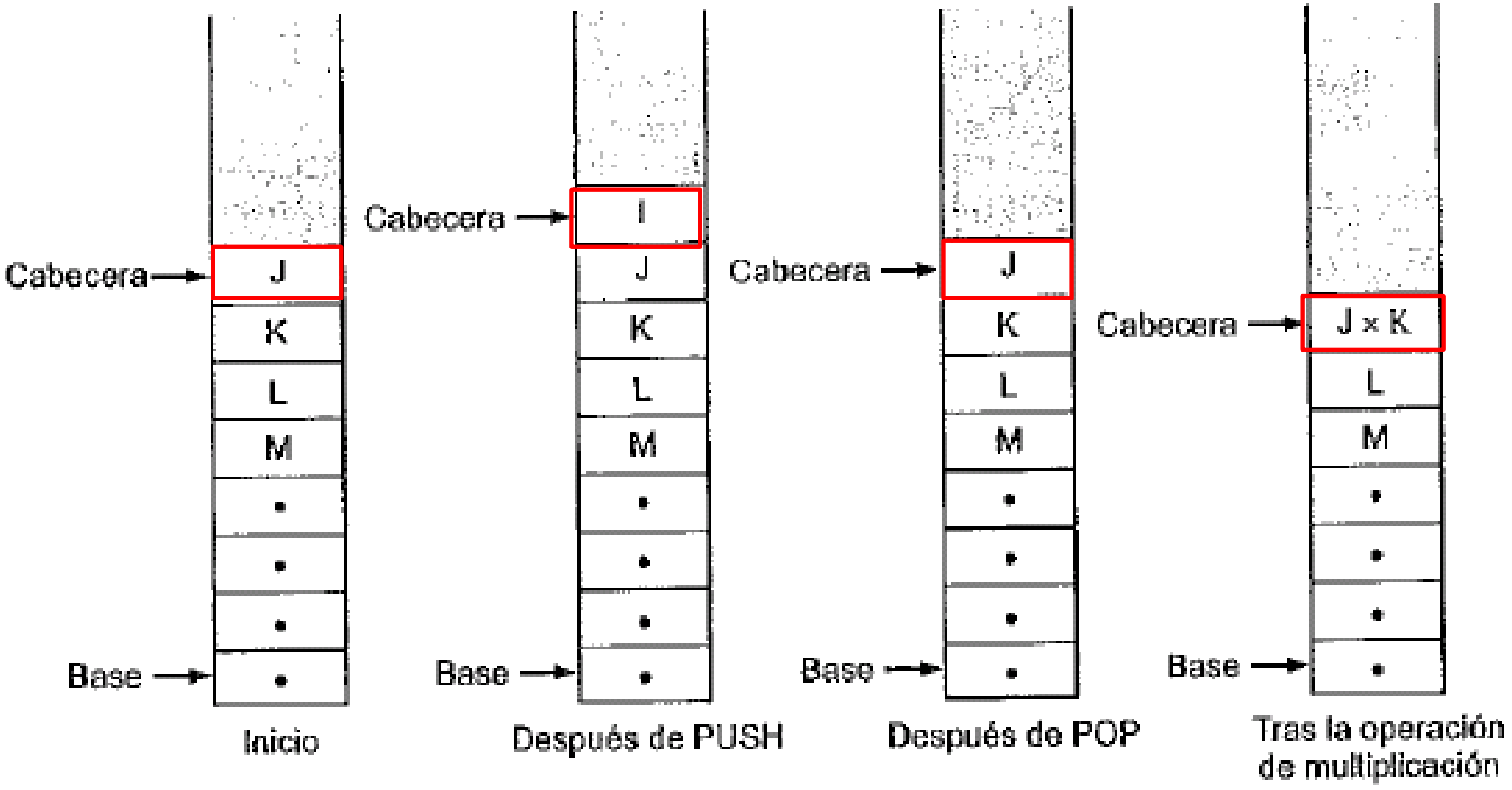


Figura 2

Uso de una pila para implementar el anidamiento de procedimientos

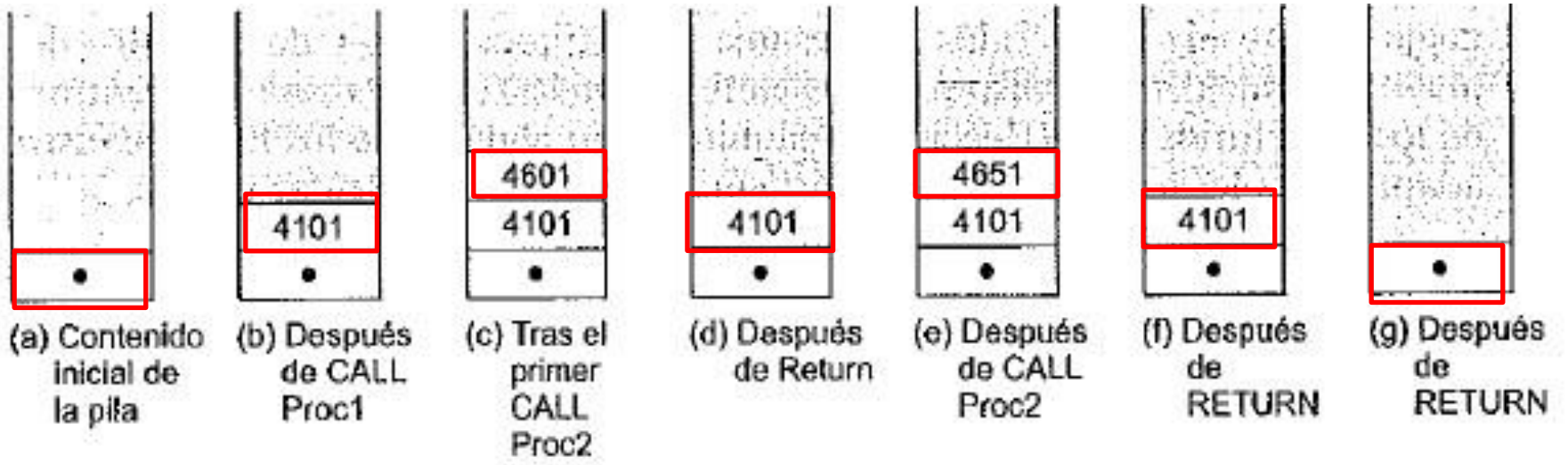


Figura 3 (representación la Figura 1)

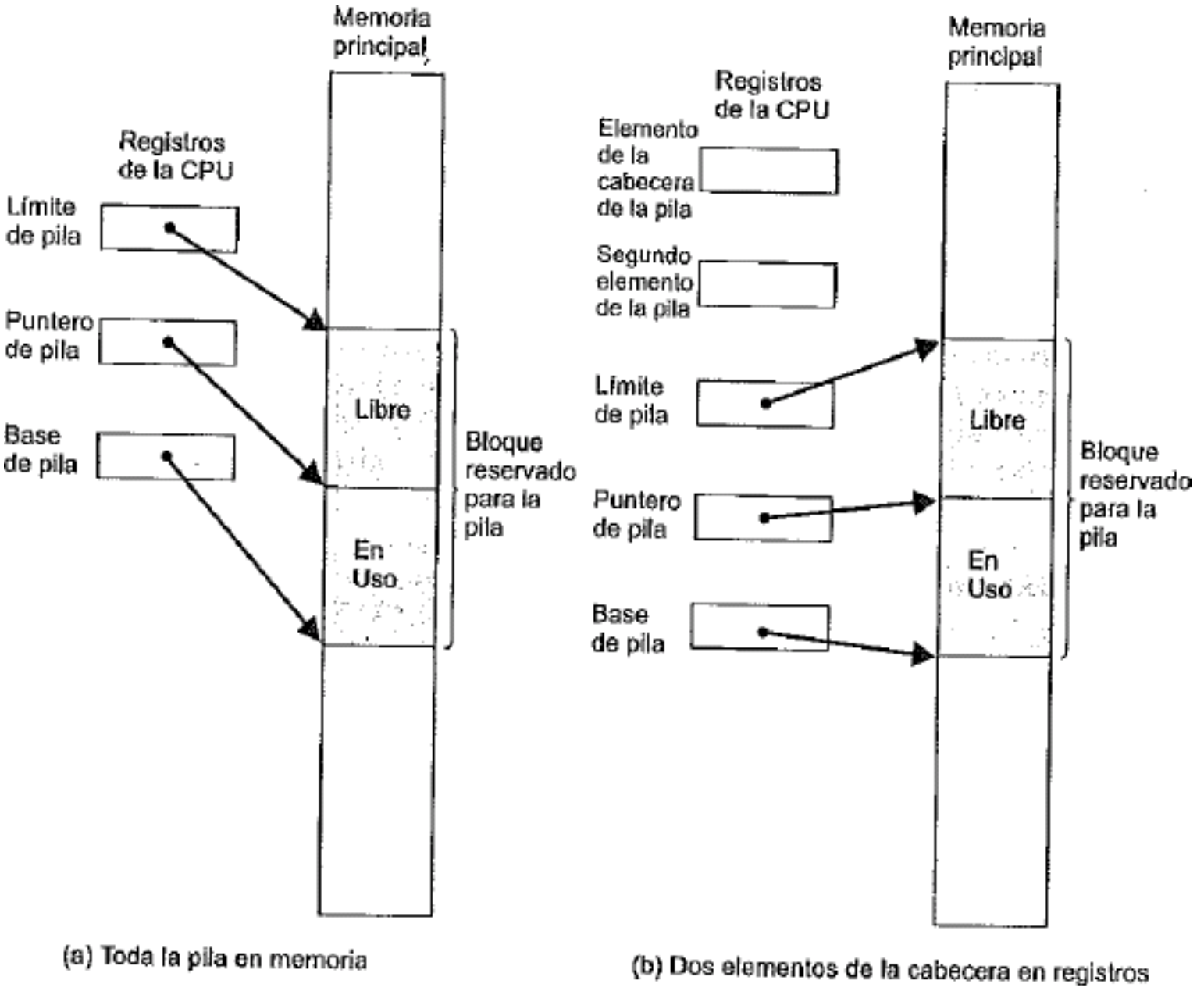

Memoria principal
Registros de la CPU
Límite de pila
Puntero de pila
Base de pila
Libre
En Uso
Bloque reservado para la pila
(a) Toda la pila en memoria
Memoria principal
Registros de la CPU
Elemento de la cabecera de la pila
Segundo elemento de la pila
Límite de pila
Puntero de pila
Base de pila
Libre
En Uso
Bloque reservado para la pila
(b) Dos elementos de la cabecera en registros

Figura 4

Ejemplo 1:

Comparación de tres programas para calcular f=(a – b)/(c + d * e)

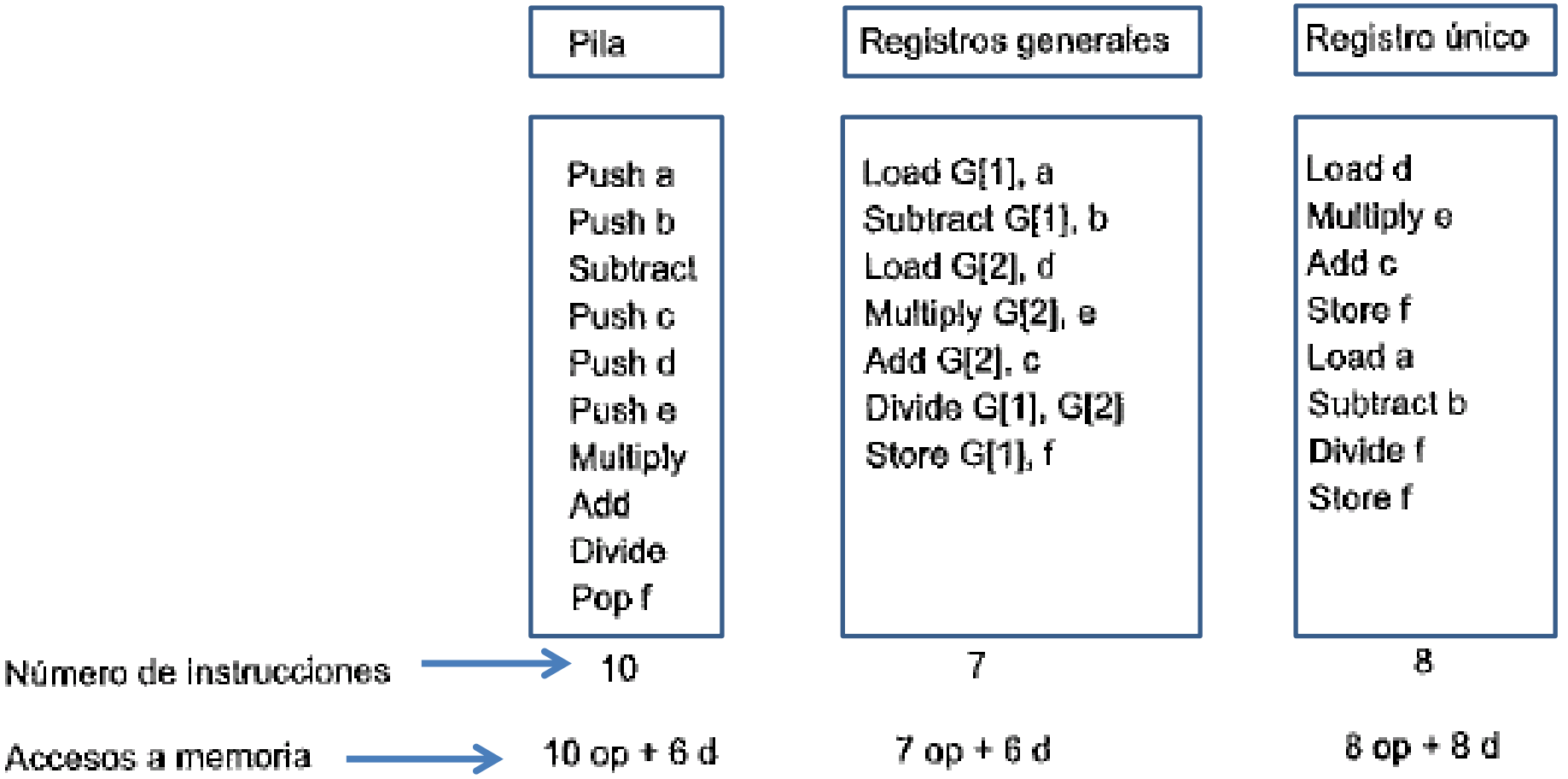


Figura 5

Ejemplo 1.1:

Utilización de la pila para calcular f=(a – b)/(c + d * e)

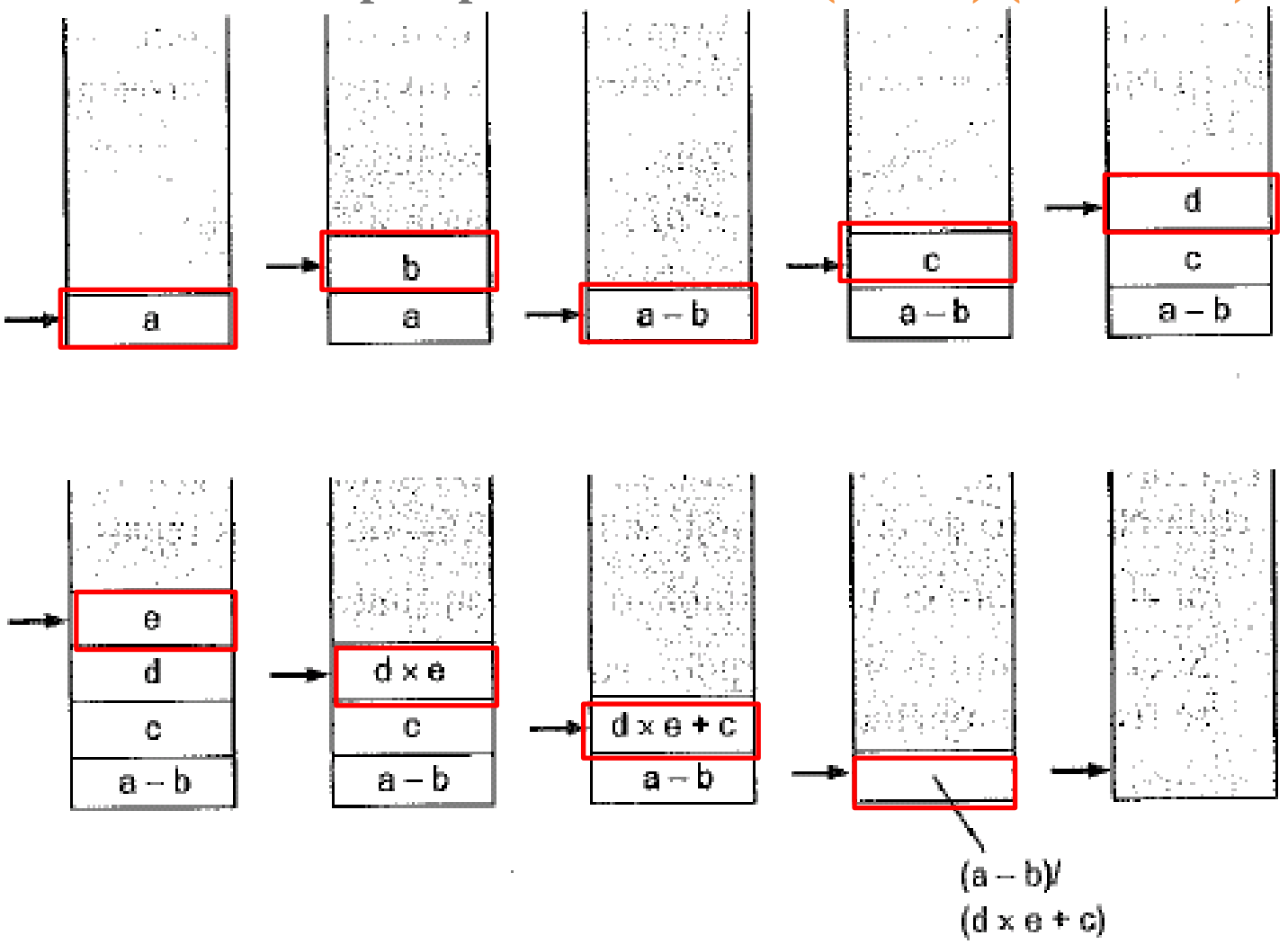


Figura 6

Operaciones PUSH y POP sobre la pila

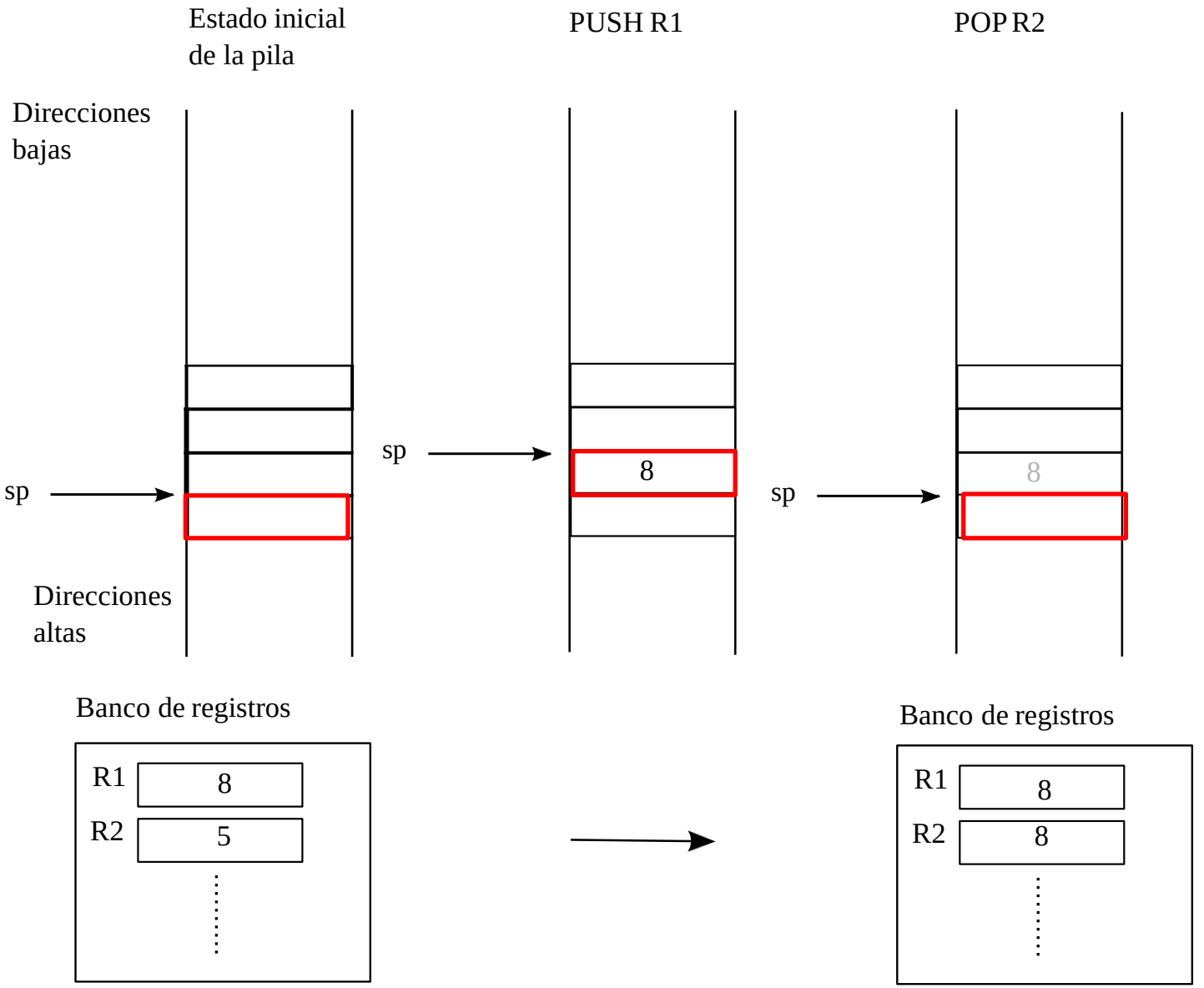
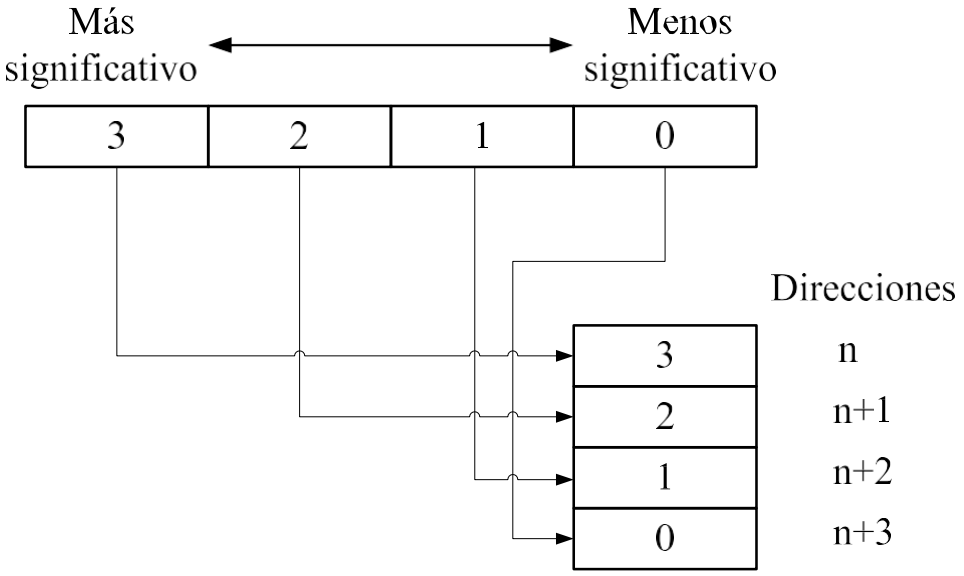
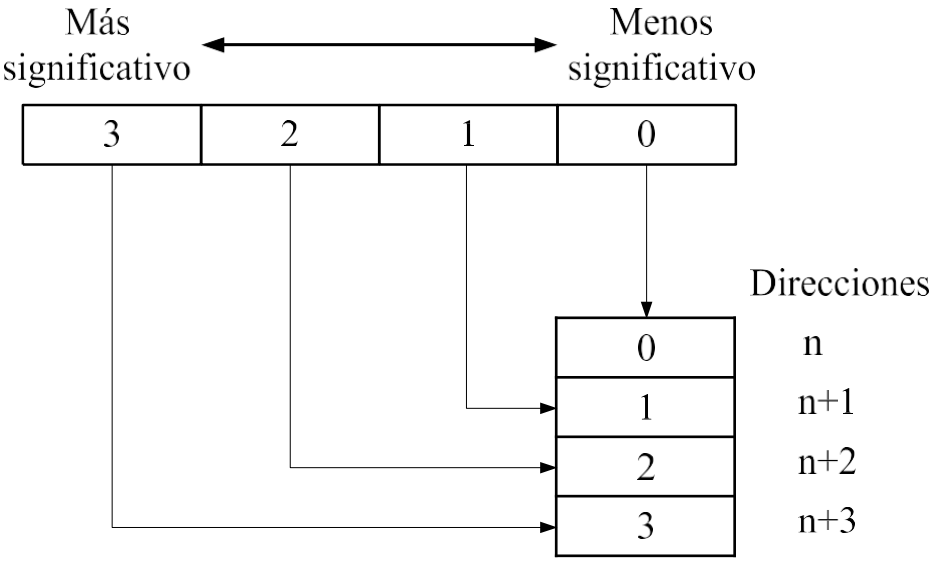


Figura 7

Diferencia entre ordenamiento big endian y little endian



(a) Big endian



(b) Little endian

Figura 8

Diferencia entre ordenamiento big endian y little endian
Ejemplo

```
/*Main function to call above function for 0x01234567*/

/* function to show bytes in memory, from
location start to start+n*/
void show_mem_rep(char *start, int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf(" %.x", start[i]);
    printf("\n");
}
```

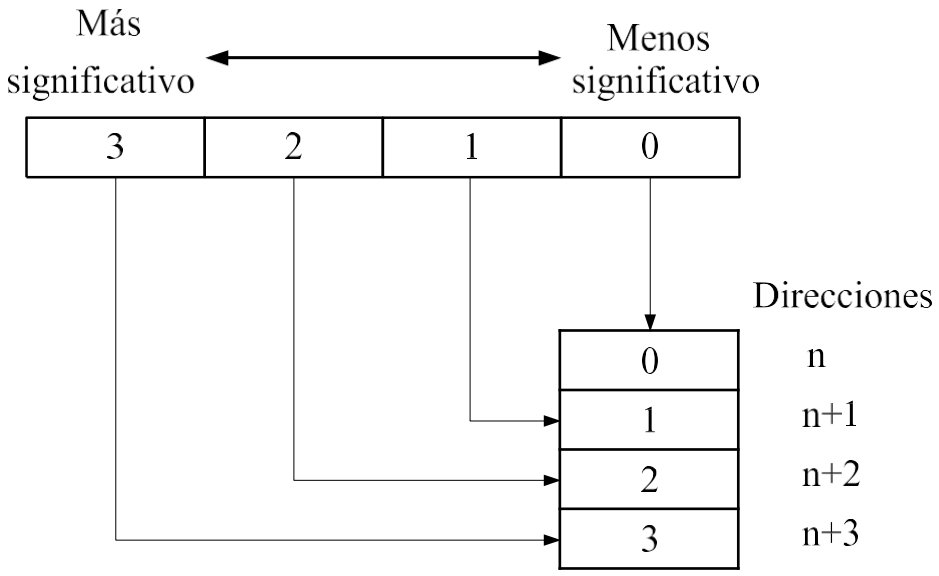
Se incrementa el valor «i» en 1

Resultado little endian
76543210

Resultado big endian
01234567

Sistema RISC-V

Eligió little-endian ya que es el ordenamiento de bytes predominante comercialmente.



(b) Little endian

Figura 9