La pila y su funcionamiento

¿Qué es la pila?

La pila (stack) es un área concreta de la memoria principal destinada a almacenar:

- Los parámetros y las variables locales de una función.
- La dirección de retorno al programa que generó la llamada a la función.
- El contexto de ejecución del programa que generó la llamada a la función, es decir, los valores que tienen los registros del procesador antes de la llamada.

La pila funciona igual que una estructura LIFO (Last In First Out). Es importante tener en cuenta que:

- La cima de la pila (o TOP) es la dirección del último dato introducido.
- Cuando se va a retirar un dato de la pila, se retira el dato que está en la cima.
- La dirección de la cima se almacena en un registro especial llamado ESP.
- La pila crece en sentido inverso, hacia las posiciones bajas de memoria principal.
- La instrucción PUSH añade un dato en la pila.
- La instrucción POP retira un dato de la pila.

Código ejemplo para explicar el funcionamiento de la pila

A través de este código, podremos aprender el funcionamiento de la pila. Un programa, llamado *Programa_Principal*, contiene todas las instrucciones necesarias para llamar una función y, la función, llamada *Subprograma*, contiene todas las instrucciones necesarias para tomar el control y después retornar al *Programa_Principal*.

Código del *Programa_Principal*

1	PUSHA
2	PUSH Param2
3	PUSH Param1
4	CALL Subprograma
5	ADD ESP, 8
6	POPA
7	… instrucciones específicas de Programa_Principal

Código del *Subprograma*

1	PUSH EBP
2	MOV EBP, ESP
3	… instrucciones específicas de la función
4	POP EBP
5	RET

1. Estudiando la instrucción Nº 1 del código *Programa_Principal*: **PUSHA**

Antes de llamar al subprograma hay que salvar el estado (contexto de ejecución) del *Programa_Principal*, es decir, hay que salvar el contenido de todos los registros de propósito general. Estos registros son: EAX, ECX, EDX, EBX, ESI y EDI. Para salvar el contenido de todos esos registros simplemente utilizaremos la instrucción PUSHA.

PUSHA es una instrucción que almacena en la pila el contenido de todos los registros de propósito general. Es lo mismo que hacer todo esto: PUSH EAX, PUSH ECX, PUSH EDX, PUSH EBX, PUSH ESI, PUSH EDI.

2. Estudiando la instrucciones Nº 2 y 3 del código *Programa_Principal*: **PUSH Param2, Push Param1**

Ahora, hay que almacenar en la pila los parámetros de la función en orden inverso. Si la llamada a la función en un lenguaje de alto nivel es *Subprograma(Param1, Param2) e*n la pila habrá que almacenar primero *Param2* y después *Param1*.

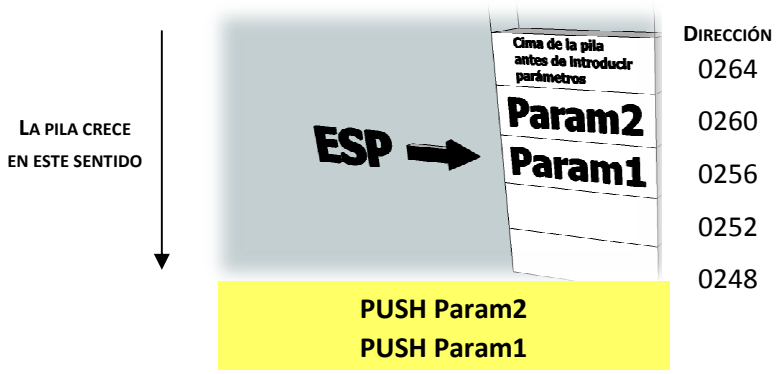

Figura 1: *Los parámetros se introducen en la pila en orden inverso: primero Param2 y después Param1. Hay que tener en cuenta que la pila crece hacia direcciones bajas.*

3. Estudiando la instrucción Nº 4 del código *Programa_Principal*: **CALL Subprograma**

La instrucción CALL realiza la llamada a la función *Subprograma*. Las acciones que efectúa son: 1º) almacenar en la pila la dirección de retorno al *Programa_Principal* (la dirección de retorno es la que apunta a la instrucción Nº 5 de *Programa_Principal*: ADD ESP, 8); 2º) saltar a la primera instrucción de la función *Subprograma*.

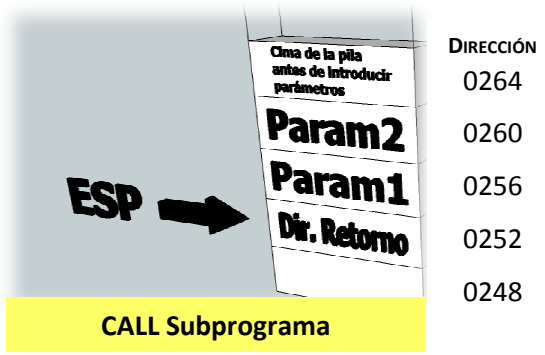

Figura 2: *Se almacena en la pila la dirección de retorno al Programa_Principal.*

4. Estudiando la instrucción Nº 1 del código *Subprograma*: **PUSH EBP**

En la siguiente instrucción, instrucción Nº 2 del código Subprograma, vamos a modificar el valor del registro EBP, así que antes debemos salvar su valor en la pila. Todavía no sabemos para qué sirve este registro, lo descubriremos en el siguiente punto.

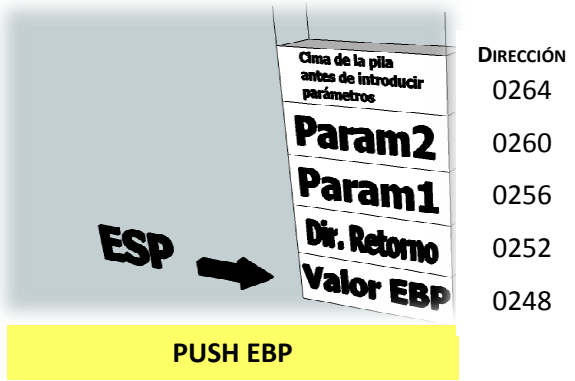

Figura 3: *el valor de EBP se salva en pila porque va a ser modificado en la siguiente instrucción.*

5. Estudiando la instrucción Nº 2 del código *Subprograma*: **MOV EBP, ESP**

Esta instrucción introduce en el registro EBP la dirección del TOP de la pila, es decir, almacena en EBP la dirección 0248. Pero, ¿para qué sirve el registro EBP? La explicación la encontraremos en el siguiente recuadro.

¿Cómo se accede a los parámetros de una función?

Cuando en *Subprograma* queramos acceder a un parámetro no lo sacaremos de la pila, simplemente proporcionaremos su dirección.

Y, ¿cuál es la dirección de Param1?

Muy fácil, la dirección de Param1 es 0256, lo sabemos porque estamos viendo una fotografía (figura 3) del estado actual de la pila. Pero un programa en ejecución no dispone de tal fotografía, así que para averiguar su dirección, el programa tendrá que hacer uso de una dirección de referencia.

Y, ¿cuál es esa dirección de referencia?

La dirección de referencia es el TOP que tiene la pila en este instante, justo después de ejecutar la instrucción PUSH EBP. Pero el TOP de la pila puede ir cambiando durante la ejecución del *Subprograma*, por ejemplo, al almacenarse las variables locales (si las tiene). Para conservar el valor que tiene el TOP en ese instante, lo almacenaremos en un registro, ese registro es EBP. Precisamente, la instrucción MOV EBP, ESP realiza esta acción.

Ahora, ¿cómo accedemos a Param1?

Cuando en *Subprograma* queramos acceder a Param1 utilizaremos la dirección [EBP + 8]. Esto significa que a EBP le sumamos 8 posiciones de memoria: [EBP + 8] = 0248 + 8 = 0256

Y, ¿a Param2?

Para acceder a Param2 utilizaremos la dirección [EBP + 12]. Esto significa que a EBP le sumamos 12 posiciones de memoria: [EBP + 12] = 0248 + 12 = 0260

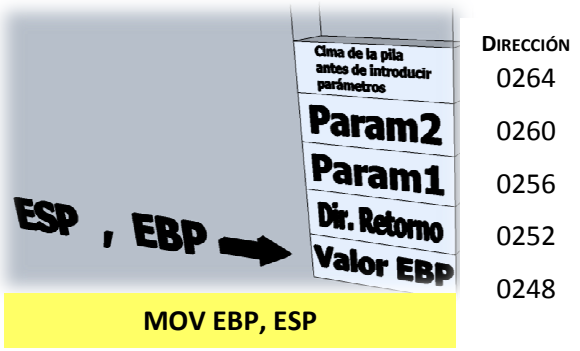

Figura 4: *Ahora, EBP apunta al TOP que tiene la pila en ese instante. El valor de EBP se mantendrá constante durante toda la ejecución del Subprograma, aunque se sigan introduciendo datos en la pila.*

6. Estudiando la instrucción Nº 4 del código *Subprograma*: **POP EBP**

Una vez finalizada la ejecución de las instrucciones específicas de la función, hay que ejecutar una serie de instrucciones para retornar a *Programa_Principal*. La primera instrucción a ejecutar es POP EBP. Esta instrucción extrae de la pila el antiguo valor que tenía el registro EBP y lo vuelve a almacenar en EBP.

7. Estudiando la instrucción Nº 5 del código *Subprograma*: **RET**

La instrucción RET extrae de la pila la dirección de retorno y salta a esa dirección. Como ya vimos antes, esta dirección es la correspondiente a la instrucción Nº 5 del *Programa_Principal*.

8. Estudiando la instrucción Nº 5 del código *Programa_Principal*: **ADD ESP, 8**

Ahora, toca extraer los parámetros de la pila. No utilizaremos la instrucción pop porque obliga a guardar en algún registro el contenido sacado de la pila y además solo permite extraer una posición cada vez. En su lugar, simplemente moveremos el TOP de la pila ¿cómo? Sumando al registro ESP el número de de bytes ocupados por los parámetros. Dado que la pila crece al revés (de posiciones altas a posiciones bajas), incrementar en 8 posiciones la dirección que almacena ESP es lo mismo que decrecer la pila en 8 posiciones. El contenido de las posiciones que queden debajo del TOP se consideran datos basura.

NOTA: si un parámetro tiene menos de 4 bytes, sumaremos 4 al ESP. Esto es así, porque cuando se introdujo el parámetro en la pila, la instrucción PUSH lo extendió a 32 bits.

9. Estudiando la instrucción Nº 6 del código *Programa_Principal*: **POPA**

Esta instrucción extrae de la pila los valores de los registros de propósito general que se almacenaron en el paso 1 y restaura los registros con esos valores.