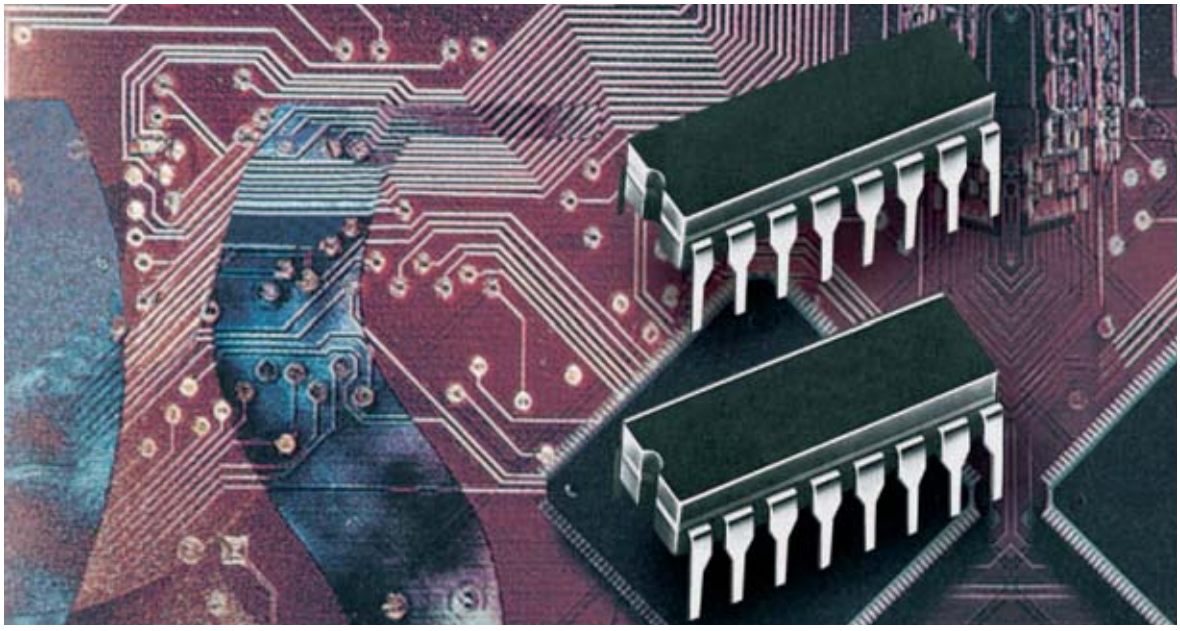

Programación con microcontroladores ST7 LITE 09 (8)

Continuamos el análisis detallado de las instrucciones Assembler para los microprocesadores ST7LITE09 iniciado en la revista N.246 afrontando las instrucciones correspondientes al 2° grupo, en este caso las instrucciones relativas al Puntero de Pila (registro Stack Pointer).

El contenido de este artículo aborda las instrucciones que en la presentación preliminar del conjunto de instrucciones (revista **Nº246**) corresponden al **segundo grupo**. Se trata de las instrucciones **Pop**, **Push** y **Rsp**, pertenecientes al mismo grupo al realizar todas operaciones sobre el **registro** que **apunta** a la **Pila** (**Stack Pointer**). Para dominar completamente estas instrucciones aconsejamos leer el artículo dedicado a la gestión del **Puntero de Pila** publicado en la revista **Nº233**, ya que trata en detalle el funcionamiento de la **Pila** (**Stack**) ilustrado con varios ejemplos. En este artículo, correspondiente a la serie dedicada al conjunto de instrucciones, se analiza la **sintaxis** de las instrucciones, incluyendo también algunos **ejemplos**.

EJEMPLOS para el 2° GRUPO de INSTRUCCIONES

La primera instrucción de cualquier programa ha de ser la instrucción **rsp**. Con esta instrucción el **Puntero de Pila** (**registro Stack Pointer**) se inicializa apuntando al valor más alto de la **Pila**, es decir a la dirección **FFh**.

```
St7_main
    rsp
```

NOTA: El término **main**, que significa **principal** en inglés, define el cuerpo principal del programa, como por ejemplo la rutina inicial.

Vamos a analizar las instrucciones **push** y **pop** en uno de sus usos más habituales: **Guardar** el estado de los **Flags** de **Condition Code** antes de ejecutar una **rutina** que afecta a los

Flags y **restaurarlos** al estado previo después de la ejecución de la rutina.

```
sub     a,VARBL1
.....   .....
push    CC
ld      a,VARBL2
ld      VARBL1,a
pop     CC
.....   .....
jreq    labzer
```

Con la primera instrucción el valor contenido en la variable **VARBL1** es restado del **acumulador** (**A**) y el resultado se almacena en **A**. En base al resultado los **Flags N-Z-C** del registro **Condition Code** quedan influenciados. Suponiendo que el valor en **A** sea igual al contenido en **VARBL1**, la diferencia sería **00h** y, por lo tanto, los **Flags** tendrán estos valores:

Flag N=0 - Flag Z=1 - Flag C=0

NOTA: Los detalles sobre los **Flags** del registro **Condition Code** se pueden consultar en la revista **Nº229**.

La instrucción **push CC** salva en la **Pila** (**Stack Memory**) el valor del registro **Condition Code**, es decir el estado de los **Flags**. Las instrucciones **ld a,VARBL2** y **ld VARBL1,a** llevan el valor contenido en la variable **VARBL2** a la variable **VARBL1** a través del acumulador. Como hemos expuesto en la revista **Nº246** la instrucción **ld** influencia el **Flag Z** de **Condition Code**.

Si el valor es **igual** a **00h** el **Flag Z** se **activa**, es decir se pone a **1**, mientras que si el valor es **distinto** de **00h** el **Flag Z no** se **activa**, es decir se pone a **0**. Suponiendo que **VARBL2** contiene el valor **0Fh**, después de estas instrucciones también **VARBL1** contendrá el mismo valor y el **Flag Z** se pone a **0**, ya que **0Fh** es distinto de **00h**. Los Flags quedan actualizados con los siguientes valores:

Flag N=0 - Flag Z=0 - Flag C=0

Con la instrucción **pop CC** se restablecen en **Condition Code** los valores anteriormente salvados en la **Pila**. En nuestro caso los **Flags** de **Condition Code** vuelven a tomar los siguientes valores:

Flag N=0 - Flag Z=1 - Flag C=0

La última instrucción (**jreq labzer**) provoca el **salto** a la instrucción con etiqueta **labzer**, ya que el **Flag Z** es igual a **1**.

2° GRUPO INSTRUCCIONES de PILA (PUSH - POP - RSP)

PUSH (Introducir en la Pila)

Esta instrucción **salva** el valor del registro destino en la **Pila** y **decrementa** en **1** el **Puntero de Pila** (registro **Stack Pointer**). La **sintaxis** de la instrucción es:

push dst

Donde **dst** es un registro, incluyendo el registro Condition Code (Flags).

Cuadro Sinóptico

neumo.	dst	H	I	N	Z	C
push	A					
push	B					
push	C					
push	CC					

push dst

dst	direccionamiento	op-code				ciclos	bytes
A	inherente		88			3	1
X	inherente		89			3	1
Y	inherente	90	89			4	2
CC	inherente		8A			3	1

Condition Flags

H	I	N	Z	C
no influenciado	no influenciado	no influenciado	no influenciado	no influenciado

NOTA: Los **Flags H-I-N-Z-C no** son **influenciados** directamente por esta instrucción, pero pueden **cambiar** de estado en las instrucciones siguientes.

POP (Extraer de la Pila)

Esta instrucción **recupera** en el registro destino el valor de la **Pila** apuntado por el registro **Stack Pointer**. Después **incrementa** en **1** el valor del registro **Stack Pointer**. La **sintaxis** de la instrucción es:

pop dst

Donde **dst** es un registro, incluyendo el registro Condition Code (Flags).

Cuadro Sinóptico

neumo.	dst	H	I	N	Z	C
pop	A					
pop	B					
pop	C					
pop	CC	H	I	N	Z	C

pop dst

dst	direccionamiento	op-code				ciclos	bytes
A	inherente		84			4	1
X	inherente		85			4	1
Y	inherente	90	85			5	2
CC	inherente		86			4	1

Condition Flags

H	I	N	Z	C
influenciado	influenciado	influenciado	influenciado	influenciado

NOTA: Los **Flags H-I-N-Z-C solo** son influenciados si el operando destino es el registro **Condition Code**, en este caso los Flags se restablecen con los valores almacenados en la **Pila**. Con el acumulador (**A**) y los registros índice (**X** e **Y**) los Flags **no** son **influenciados**.

RSP (Reiniciar Puntero de Pila)

Esta instrucción pone el **Puntero de Pila** (registro **Stack Pointer**) a su **valor inicial** (**FFh**). La **sintaxis** de la instrucción es:

rsp

Cuadro Sinóptico

neumo.	dst	H	I	N	Z	C
rsp						

rsp

dst	direccionamiento	op-code				ciclos	bytes
	inherente		9C			2	1

Condition Flags

H	I	N	Z	C
no influenciado	no influenciado	no influenciado	no influenciado	no influenciado