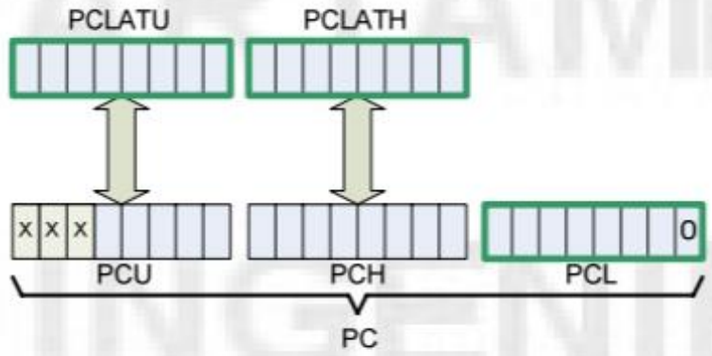
Microcontroladores

Método de direccionamiento y registro STATUS

1.- Contador de Programa

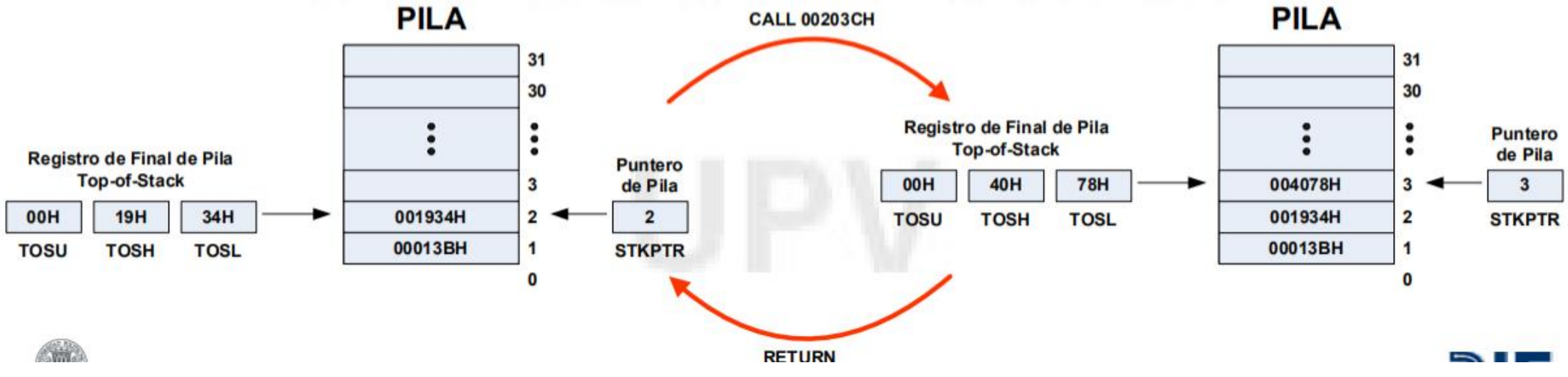


Puntero de 21 bits que indica la dirección en memoria de programa de la instrucción que se debe ejecutar. Está compuesto por 3 bytes:

- **PCU: parte superior del PC, registro no directamente accesible; las operaciones de lectura/escritura sobre este registro se hacen a través del registro PCLATU**
- **PCH: parte alta del PC, registro no directamente accesible; las operaciones de lectura/escritura sobre este registro se hacen a través del registro PCLATH**
- **PCL: parte baja del PC, registro directamente accesible. Una operación de lectura sobre PCL provoca que los valores de PCU y PCH pasen a PCLATU y PCLATH respectivamente. Y una operación de escritura sobre PCL provoca que los valores de PCLATU y PCLATH pasen a PCU y PCH respectivamente. El PCL siempre tiene el bit menos significativo a '0', debido a que las instrucciones siempre empiezan en direcciones pares.**

2.- Pila de Direcciones

- **La Pila es un bloque de memoria RAM independiente de 31 palabras de 21 bits que sirve para almacenar temporalmente el valor del PC cuando se produce una llamada a subrutina o una interrupción.**
- **El puntero de pila (contenido en el registro STKPTR) es un contador de 5 bits que indica la posición actual del final de pila. El contenido del final de pila es accesible mediante los registros TOSU, TOSH, TOSL.**
- **Cuando se procesa una interrupción o se ejecutan las instrucciones las instrucciones CALL o RCALL (el PC está apuntando a la siguiente instrucción) se incrementa el STKPR y se almacena en el final de pila el valor del PC.**
- **Cuando se ejecutan las instrucciones RETURN, RETLW o RETFIE se copia el valor almacenado en la cima de pila en el PC y se decrementa el STKPTR.**



3.- Registros de la PILA

REGISTRO STKPTR

	L/R-0	L/R-0	-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0
STKPTR	STKFUL	STKUNF	-	SP4	SP3	SP2	SP1	SP0

– **STKFUL: Flag de llenado de la pila (en modo escritura únicamente puede ser puesto a '0'):**
  * **STKFUL='0': No se ha producido el llenado o desbordamiento de la pila**
  * **STKFUL='1': Se ha producido el llenado o desbordamiento de la pila**
– **STKUNF: Flag de vaciado de la pila (en modo escritura únicamente puede ser puesto a '0'):**
  * **STKUNF='0': No se ha producido el desbordamiento por vaciado de la pila**
  * **STKUNF='1': Se ha producido el desbordamiento por vaciado de la pila**
– **SP4..SP0: Puntero de pila. Estos 5 bits indican la posición del final de la pila (valor de 0 a 31)**

4.- Vaciado/Llenado de la Pila

LA PILA DE DIRECCIONES:

- **Llenado de la Pila: si la pila llega al máximo de su capacidad (31 elementos apilados):**
  - **Si el bit de configuración STRVEN está a '0': el bit STKFUL del registro STKPTR se pone a '1' y si se producen nuevos apilamientos no afectarán a la pila.**
  - **Si el bit de configuración STRVEN está a '1': el bit STKFUL del registro STKPTR se pone a '1' y se producirá un reset del uC.**
- **Vaciado de la Pila: si la pila está vacía y se intenta desapilar de nuevo:**
  - **Si el bit de configuración STRVEN está a '0': el bit STKUNF del registro STKPTR se pone a '1', el PC se pondrá a 0000H y Puntero de pila permanecerá a 0.**
  - **Si el bit de configuración STRVEN está a '1': el bit STKUNF del registro STKPTR se pone a '1' y se producirá un reset del uC.**

PILA RAPIDA DE REGISTRO:

Se trata de una pila de un solo nivel en la que se apilan los valores del registro de estado, del W y del registro BSR cada vez que se produce una interrupción. Estos valores pueden ser recuperados si al salir de la interrupción se utiliza la instrucción "RETFIE, FAST". Si están habilitadas las interrupciones de baja y alta prioridad, esta pila no debe ser utilizada en interrupciones de baja prioridad.
Si no hay interrupciones habilitadas esta pila puede ser utilizada en llamadas a subrutinas ("CALL <eti>, FAST" y "RETURN, FAST").

5.- Lectura de datos en memoria de programa

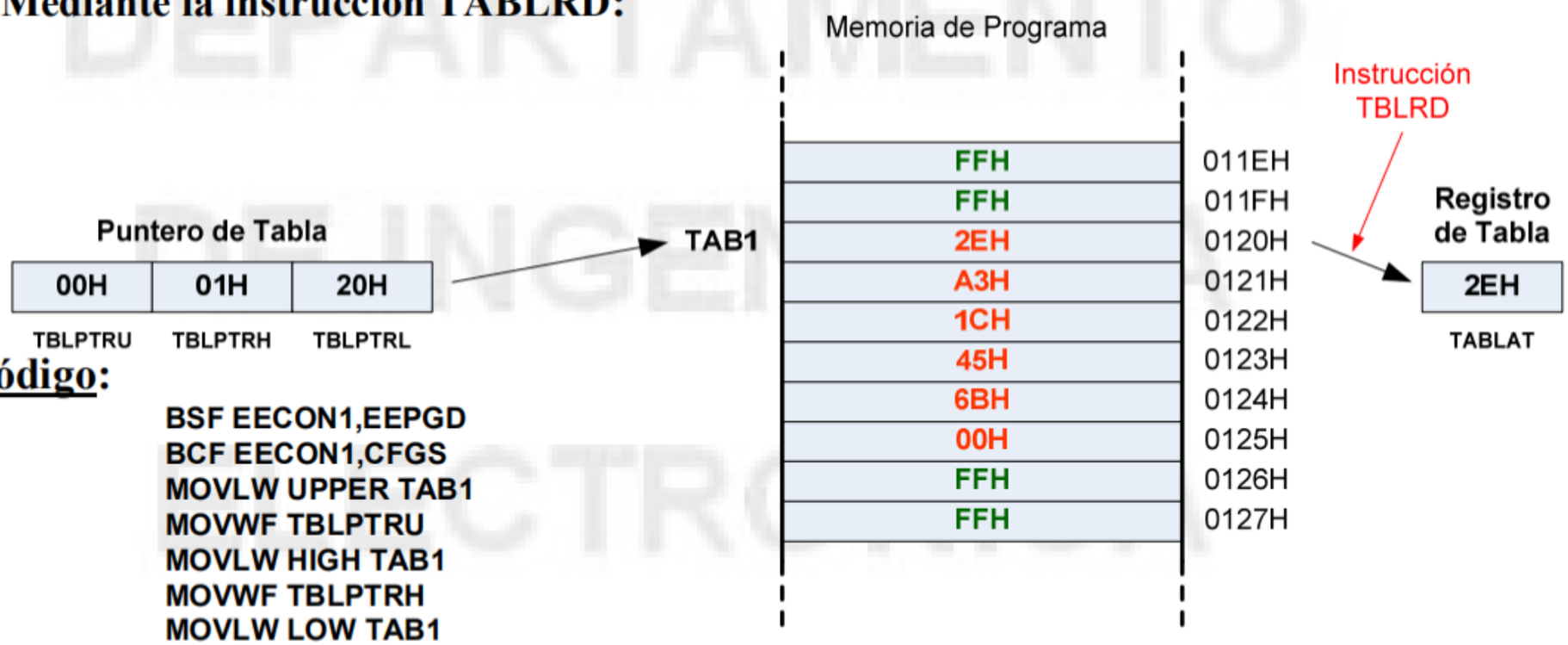
- La memoria de programa puede ser leída, borrada y escrita durante la ejecución del programa. La operación que se utiliza normalmente en tiempo de ejecución es la de lectura de tablas o datos almacenados en memoria de programa.
- Existen dos formas de leer tablas de memoria de programa:
  - Mediante la instrucción RETLW:

```
        MOVF <DESPL.>,W      ; Se almacena en valor constante de desplazamiento a W
        CALL TABLA           ; Se llama a la función TABLA (después de la llamada el valor
        .                    ; leído de la tabla quedará almacenado en W para poder ser
        .                    ; utilizado)
        .
ORG     <INICIO TABLA>       ; Dirección inicial de la tabla
TABLA   ADDWF PCL            ; Se suma al PCL actual el valor del desplazamiento
        RETLW <DATO0>        ; Se sale de la subrutina almacenado en W DATO0
        RETLW <DATO1>        ; Se sale de la subrutina almacenado en W DATO1
        .
        .
        .
        RETLW <DATON>        ; Se sale de la subrutina almacenado en W DATON
```

<DESPL.> indica la posición del elemento que se quiere leer. Su valor debe ser el doble del valor de la posición que queremos leer.
<INICIO TABLA> dirección de inicio de la tabla (debe ser un valor par)
INCONVENIENTE: Cada byte de la tabla ocupa dos bytes en memoria.

5.- Lectura de datos en memoria de programa

- **Mediante la instrucción TABLRD:**



Código:

```
        BSF EECON1,EEPGD
        BCF EECON1,CFGS
        MOVLW UPPER TAB1
        MOVWF TBLPTRU
        MOVLW HIGH TAB1
        MOVWF TBLPTRH
        MOVLW LOW TAB1
        MOVWF TBLPTRL
        TBLRD*
        MOVF TABLAT,W          ; El dato leído se almacena en TABLAT y de ahí lo pasamos
        .                      ; al acumulador para poder utilizarlo
        .
        .
        ORG 0x0120
TAB1:   DB 0x2E,0xA3,0x1C,0x45,0x6B
```

5.- Lectura de datos en memoria de programa

- **Mediante la instrucción TABLRD:**

Proceso:

1º Poner a '0' el bit CFGS del registro EECON1 (CFGS='0' acceso a la memoria EEPROM o a la memoria flash de programa / CFGS='1' acceso a la memoria de configuración)

2º Poner a '1' el bit EEPGD del registro EECON1 (EEPGD='0' acceso a la memoria EEPROM / EEPGD='1' acceso a la memoria flash de programa).

3º Inicializar el puntero de tabla (registros TBLPTRU, TBLPTRH, TBLPTRL)

4º Leer el dato apuntado por el puntero de tabla mediante la instrucción TBLRD. El valor leído queda almacenado en el registro TABLAT. La instrucción TBLRD tiene 4 formatos:

TBLRD*	Lee el dato
TBLRD*+	Lee el dato e incrementa el puntero
TBLRD*-	Lee el dato y decrementa el puntero
TBLRD+*	Incrementa el puntero y lee el dato

Si se quiere leer una posición de memoria especifica dentro de la tabla, habrá que sumarle al puntero de tabla el indice que nos lleve a dicha posición.

6.- Registro STATUS

- **N: Bit de de valor negativo. Bit utilizado para operaciones con signo (complemento a 2):**
  - **N='0': Resultado de la última operación positivo (MSB a '0')**
  - **N='1': Resultado de la última operación negativo (MSB a '1')**
- **OV: Bit de desbordamiento. Bit utilizado para operaciones con signo (complemento a 2). Indica si se ha producido desbordamiento del 7º bit (bit 6), es decir si se ha producido algún cambio en el bit 7 del resultado:**
  - **OV='0': No se ha producido desbordamiento en la operación aritmética**
  - **OV='1': Se ha producido desbordamiento en la operación aritmética**
- **Z: Bit de cero:**
  - **Z='0': El resultado de la operación aritmética o lógica ha sido diferente de 0**
  - **Z='1': El resultado de la operación aritmética o lógica ha sido 0**
- **DC: Bit de acarreo de dígito (para las instrucciones ADDWF, ADDLW, SUBLW y SUBWF):**
  - **DC='0': No se ha producido acarreo del 4º bit**
  - **DC='1': Se ha producido acarreo del 4º bit**
- **C: Bit de acarreo (para las instrucciones ADDWF, ADDLW, SUBLW y SUBWF):**
  - **DC='0': No se ha producido acarreo del 8º bit**
  - **DC='1': Se ha producido acarreo del 8º bit**

7.- Acceso a la memoria RAM de datos:

- La instrucción **MOVFF op1,op2** permite acceder directamente a cualquier posición de la memoria RAM de datos ya incorpora los 12 bits de la dirección de los dos operandos (es una instrucción de 2 words de 16 bits).
- El resto de instrucciones que permiten acceder a la memoria RAM de datos incorporan un modificador "a" que establece el modo de acceso:
  - Si a= 1: se accede a la totalidad de memoria mediante el BSR. Mediante los 4 bits menos significativos del BSR se selecciona el banco y mediante el operando de la instrucción se indica el byte del banco seleccionado que se quiere acceder. La instrucción MOVLB permite escribir directamente en el BSR el valor del banco a seleccionar (los bits BSR[7..4] no son considerados y se leen siempre como '0').

```
MOVLW .33       ; Se carga el valor 33 en el acumulador
MOVLB .1        ; Se selecciona el banco de registros 1
MOVWF VAR1,1 ; Se pasa el contenido del acumulador a VAR1 declarada en la posición 100H
```

  - Si a = 0: se ignora el BSR y se accede al banco de acceso rápido compuesto por los primeros 96 bytes del banco 0 y los 160 bytes de los SFR's. Este método permite acceder a dichos bytes con una sola instrucción sin necesidad de seleccionar previamente el banco.

```
MOVLW .33       ; Se carga el valor 33 en el acumulador
MOVWF VAR2,0 ; Se pasa el contenido del acumulador a VAR2 declara en la posición 000H
```

8.- Modos de direccionamiento:

- **El modo de direccionamiento es la forma en la que se obtienen el o los datos que van a ser utilizados en la instrucción. Existen 4 modos de direccionamiento: inherente, literal, directo e indirecto.**
  - **Modo de direccionamiento inherente: en este modo o bien la instrucción no tiene operando o bien el operando viene especificado en el propio código de operación de la instrucción.**

```
RESET  ; Realiza un reset por software (los operandos son todos los registros afectados por el reset)
DAW    ; Ajuste decimal del acumulador (el operando es el acumulador). Formato BCD
NOP    ; No realiza ninguna operación (no hay operando)
```

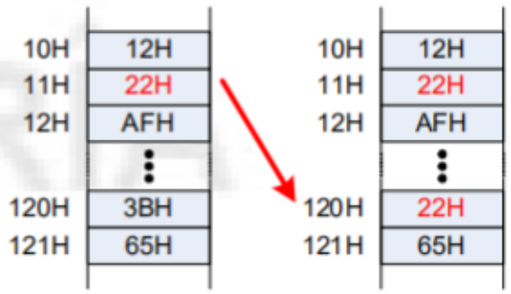
  - **Modo de direccionamiento literal: en este modo el valor del operando viene indicado de forma explicita en la instrucción.**

```
GOTO 0x100  ; Salto incondicional a la dirección 100H (el operando es 0x100)
MOVLW .23   ; Cargar en el acumulador el valor 23 (el operando es el .23)
```

9.- Modos de direccionamiento a la RAM:

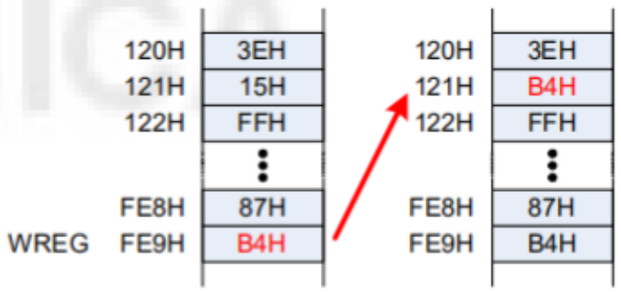
- **Modo de direccionamiento directo: en este modo la dirección en la que se encuentra el valor del operando viene indicada de forma explicita en la instrucción. El operando puede ser un byte o un bit:**
  - **Operando de tipo byte:**
    - **Mediante la instrucción MOVFF org,dest:**

```
MOVFF 0x011,0x120  ;
```



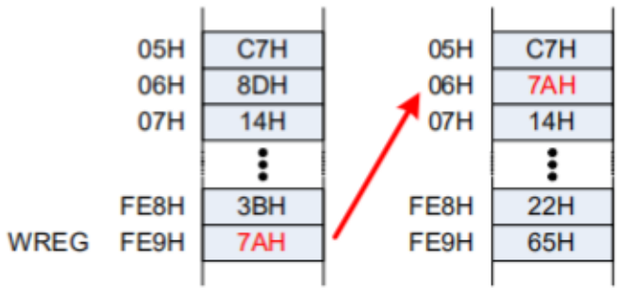
    - **Mediante la combinación del BSR y el valor de 8 bits indicado en la instrucción**

```
MOVLW 0xB4
MOVLB .1
MOVWF 0x21,1
```



    - **Mediante el banco de acceso rápido**

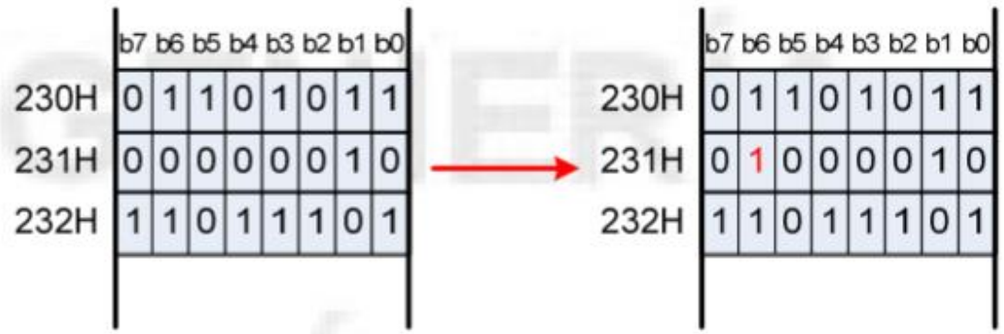
```
MOVLW 0x74
MOVWF 0x06,0   ; También es válido MOVWF 0x60,A
```



9.- Modos de direccionamiento a la RAM:

- **Modo de direccionamiento directo (cont.)**
  - **Operando de tipo bit: en este caso en la instrucción se especifica el registro en el que se encuentra el bit y luego la posición del bit dentro del registro.**

```
MOVLB .2
BSF 0x31,6
```

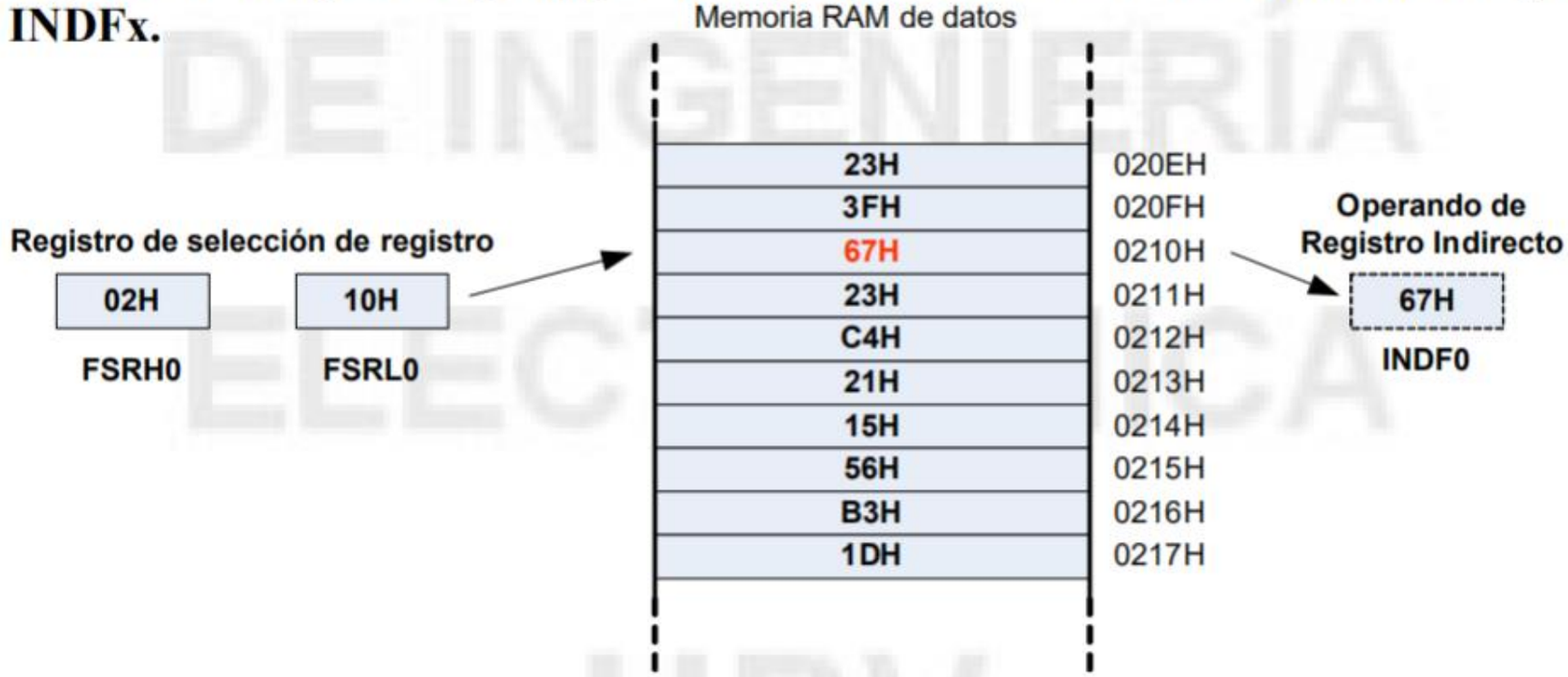


Para facilitar las tareas de programación, en el lenguaje ensamblador los valores numéricos que corresponden a datos literales o a direcciones de memoria se representan mediante etiquetas.

```
VAR1 EQU 0x010
CTE1 EQU .24
.
.
.
MOVLW CTE1
MOVWF VAR1,0
```

9.- Modos de direccionamiento a la RAM:

- **Modo de direccionamiento indirecto: en este modo la dirección de memoria en la que se encuentra el dato viene especificado en uno de los registros FSR0, FSR1 y FSR2. Para acceder al valor se debe escribir la dirección del dato (12 bits) en el registro FSRx (FSRxH[3..0] y FSRxL[3..0]) y se lee/escribe el dato en el registro INDFx.**



Los registros INDFx son registros virtuales, aunque tienen una dirección asignada en la zona de SFR's, físicamente se corresponden con la dirección de memoria apuntada por el correspondiente FSRx.

9.- Modos de direccionamiento a la RAM:

- **Modo de direccionamiento indirecto (cont.): Además de los INDFx existen otros registros virtuales que permiten acceder el dato apuntado por los FSRx, permitiendo operación adicionales:**
  - **POSTDEC0, POSTDEC1, POSTDEC2: al acceder a un registro POSTDECx se accede a la posición de memoria apuntada por el FSRx correspondiente y a continuación se decrementa el valor de dicho FSRx.**
  - **POSTINC0, POSTINC1, POSTINC2: al acceder a un registro POSTINCx se accede a la posición de memoria apuntada por el FSRx correspondiente y a continuación se incrementa el valor de dicho FSRx.**
  - **PREINC0, PREINC1, PREINC2: al acceder a un registro PREINCx se incrementa el valor del FSRx correspondiente y, a continuación, se accede a la posición de memoria apuntada por el nuevo valor del FSRx.**
  - **PLUSW0, PLUSW1, PLUSW2: al acceder a un registro PLUSWx se accede a la dirección de memoria RAM formada por la suma del valor del FSRx y del acumulador WREG (se considera en valor de WREG con signo [-127;128]). En este caso el valor del FSRx no se modifica.**

La lectura de los registros virtuales (INDFx, POSTDECx, POSTINCx, PREINCx) mediante direccionamiento indirecto da por resultado 0x00. La escritura de los registros virtuales mediante direccionamiento indirecto da por resultado un NOP.

No se deben modificar los valores de los FSRx's mediante direccionamiento indirecto; se debe acceder a estos registros siempre mediante direccionamiento directo.

Preguntas