TUTORÍA #3

Brando Miguel Palacios Mogollon

Operación: **MOVC**
Función: Mover el byte de código al acumulador
Sintaxis: MOVC A, *registro* @ A +

Instrucciones	OpCode	Bytes	Banderas
MOVC A, @ A + DPTR	0x93	1	Ninguna
MOVC A, @ A + PC	0x83	1	Ninguna

Descripción: MOVC mueve un byte de la Memoria de código al Acumulador. La dirección de la memoria de código desde la que se moverá el byte se calcula sumando el valor del acumulador con DPTR o el contador de programa (PC). En el caso del Contador de programas, la PC primero se incrementa en 1 antes de sumarse con el acumulador.

Ver también: MOV , MOVX

@R0: Memoria apuntada por R0.

#30H: valor de memoria

#=!@

Brando Miguel Palacios Mogollon

EJM:

```
org 0000h
Mov R0,#30h
Mov R3,#3

Cell:
        Mov A,#1
        Mov @R0,A       ;es lo mismo que decir mov 30h,A
Add A,#30h
DJNZ R3,cell
Lcall sndchr
end
```

Brando Miguel Palacios Mogollon

TABLA

Resultado:

Posiciones de memorias llenas del 30h al 3Eh com los valores de la subrutina obtengo_valor:

```
        org 0000h
        mov R0,#30h
        mov R4,#16
        mov R5,#0
lazo:
        mov A,R5
        lcall obtengo_valor
        mov @R0,A
        inc R5
        inc R0
        djnz R4,lazo
        sjmp $
obtengo_valor:
        inc A
        movc A,@A+PC
        ret
db 2Bh,7Fh,19h,88h,08h,1Dh,77h,55h,33h,9Fh,0CCh,0DEh,44h,0B0h,33h
END
```

Brando Miguel Palacios Mogollon

SUBRITUNA DOS_DIGITOS_ASCII

```
dos_digitos_decimales:
        mov B,#10
        div AB
        add A,#30h
        mov 30h,A
        mov A,B
        add A,#30h
        mov 31h,A
        ret
```

Almacena ellos datos en la posición 30h y 31h

Por ejm :

```
org 8000h
mov R3,#40
mov A,R3
lcall dos_dígitos_decimales
mov A,30h
lcall sndchr
mov A,31h
lcall sndchr
mov A,#0dh
lcall sndchr
end
```

OUT:

40

Brando Miguel Palacios Mogollon

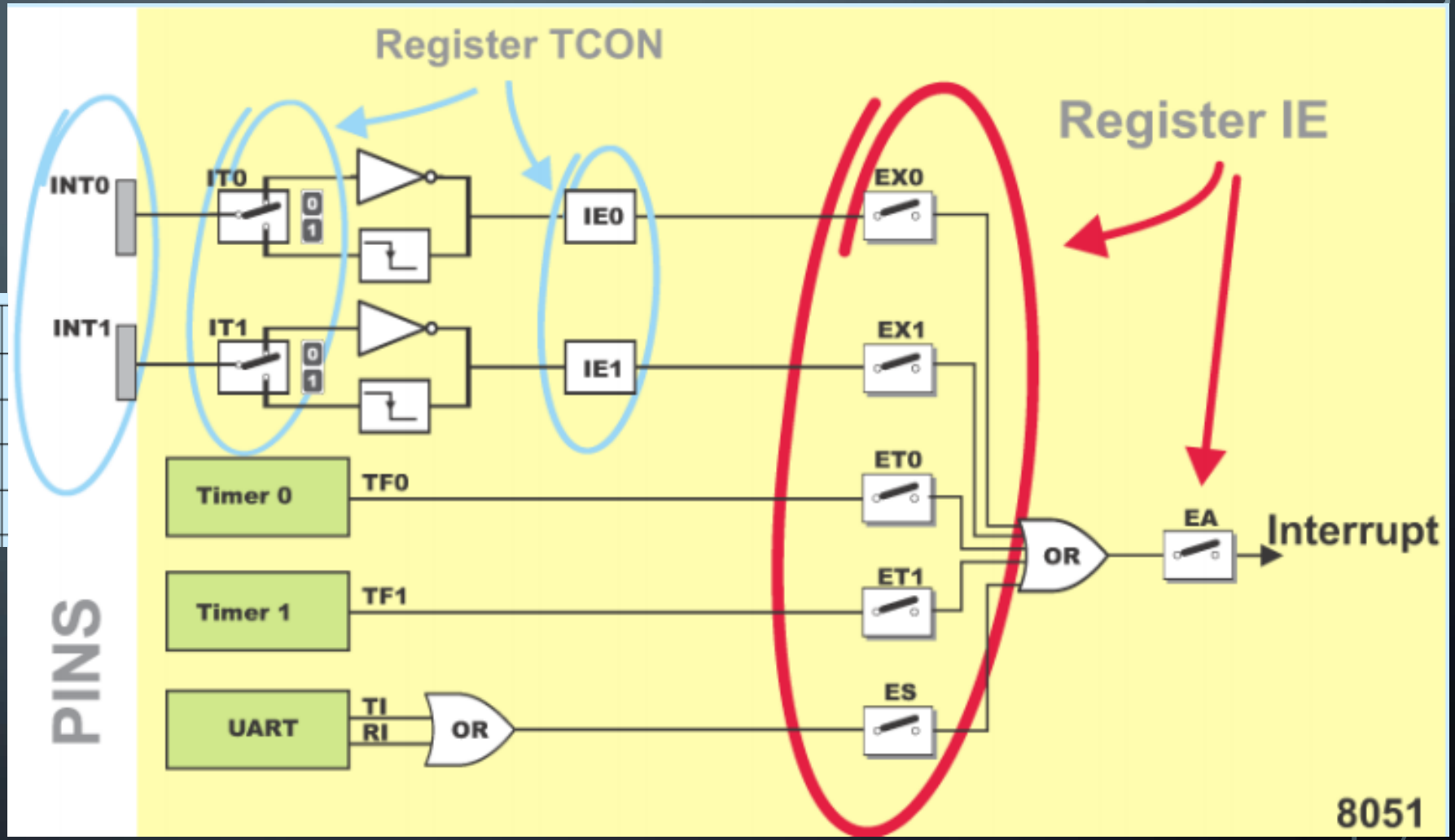
- tres_digitos_decimales
- mov B,#100
- div AB
- add A,#30h
- mov 30h,A
- mov A,B
- mov B,#10
- div AB
- add A,#30h
- mov 31h,A
- mov A,B
- add A,#30h
- mov 32h,A

30h:CENTE 31h: DEC 32h: UNI

Brando Miguel Palacios Mogollon

INTERRUPCIONES:

Interrupción	Bandera
Externo 0	IE0
Externo 1	IE1
Timer 0	TF0
Timer 1	TF1



Brando Miguel Palacios Mogollon

SUBRUTINA SETINTVEC (SETINVEC EQU 145H)

Permite el salto de la memoria a la interrupción necesaria se clasifica de la siguiente manera:

- Mov A,#0 ;interrupción externa 0
- Mov A,#2 ; interrupción externa 1
- Mov A,#1 ;interrupción interna 0 (timer 0)
- Mov A,#3 ; interrupción interna 1 (timer 1)

Brando Miguel Palacios Mogollon

INSTALACIÓN DE INTERRUPCIONES

```
        org 8000h
        mov A,#0 ; Selecciono interrupción  externa 0
        mov dptr,#rutina_interrup_ext0
        lcall setinvec
        setb EX0 ; Habilita la interrupción externa 0
        setb EA  ; Habilita interrupciones globales
        setb P3.2   ;Instalo la interrupción para el botón P3.2
        ;Código siguiente a la instalación
        sjmp $                  ;ljmp  2f0h
rutina_interrup_ext0:
        clr EX0
        ;La interrupción que deseas colocar
        setb EX0
        reti
        end
```

Brando Miguel Palacios Mogollon

```
        org 8000h
        setb IT0 ; Configuro para considerar el flanco de bajada
        mov A,#0
        mov dptr,#rutina_interrup_ext0
        lcall setinvec
        setb EX0
        setb EA
        setb P3.2
        setb P1.0
        sjmp $
rutina_interrup_ext0:
        clr EX0
        mov A,#100
        lcall delay
        cpl P1.0
        mov A,#200
        lcall delay
        setb EX0
        reti
        end
```

EJM: BOTÓN DE LED

PARA OPERACIONES CON BOTONES EN UNA INTERRUPCIÓN SE UTILIZA IT0 PARA QUE EL MICROCONTROLADOR PUEDA DETECTAR LAS PULSACIONES CUANDO EL BIT SEA 1.

CABE RECALCAR QUE DADO QUE EL BOTÓN REALIZA REBOTE ES BUENO PONER UN RETARDO.

Brando Miguel Palacios Mogollon

INSTALACIÓN DE UN TIMER COMO INTERRUPCIÓN

```
setintvec equ 145h
print equ 136h
                org 8000h
;=================================================================
;               mov TMOD,#0h                ;modo 0 del timer 0
;               mov A,#1        ;timer 0 es la fuente de la interrupcion
;               mov dptr,#IntTimer0         ;IntTimer0 es la direccion inicial del ISR
;               lcall stintvec  ;Posibilita tener el ISR en RAM
;               setb TR0        ;inicia timer 0
;               setb ET0        ;habilita interrupcion del timer 0
;               setb EA                     ;habilita interrupcion general
;=========================================
mov TMOD,#20h               ;modo 0 del timer 0
                                            ;para que no interfiera con la comunicacion serial
                mov a,#1                    ;Timer 0 es la fuente de interrupcion
                mov dptr,#IntTimer0         ;IntTimer0 es la direccion inicial del ISR
                lcall setintvec             ;posibilita tener el ISR enn RAM
                setb TR0                    ;inicia el timer 0
                setb ET0                    ;habilita inerrupcion del timer0
                setb IT0        ;fija la interrupcion externa sensible en el flanco de bajada
                setb IT1        ;fija la int. ext. 1 en el flanco de bajada
```

Brando Miguel Palacios Mogollon

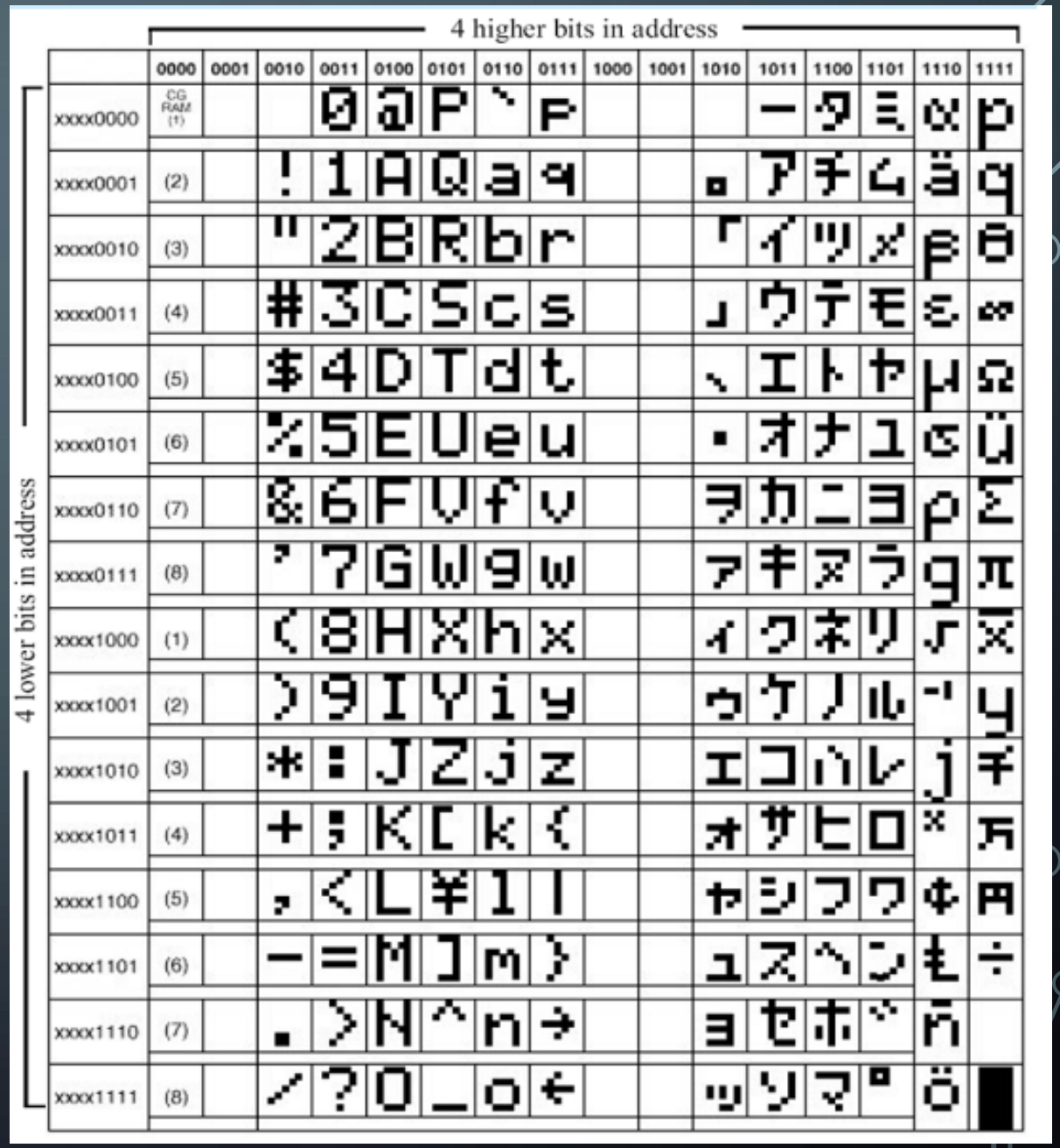
```
        mov A,#0  ;fuente de interrupcion externa 0
        mov dptr,#ISR0          ;fija dptr con la direccion de inicio
                                ;de la interrupcion externa 0 para llamar a setintvec
        lcall setintvec         ;fija el ISR para INT0
        mov A,#2  ;fuente de int. ext. 1
        mov dptr,#ISR1          ;fija dptr con la direccion
        lcall setintvec
        setb EX0    ;habilita interrupcion externa 0 en el registro IE
        setb EX1
        setb EA
        setb p3.2   ;fija el pin p3.2 como entrada boton conectado
        setb p3.3
;=================================================================
        sjmp $
```

Brando Miguel Palacios Mogollon

```
IntTimer0:
        djnz R6,sale
        mov R6,#17  ;X
        clr TF0
        clr TR0
        mov TH0,#06h
        mov TL0,#0ach
        setb TR0
        jnb TF0,$
        clr TF0
        cpl P1.0
 sale:
        clr TF0
        reti
ISR0:
        lcall print
        db 0dh,0ah,"me interrumpes Nino rata >:V p3.2",0dh,0ah,0
        reti
ISR1:
        lcall print
        db 0dh,0ah,"me interrumpes p3.3",0dh,0ah,0
        reti
        END
```

Brando Miguel Palacios Mogollon

PANTALLA LCD 16X2



FORMATO DEL LCD

- El LCD posee una forma diferente de operar con comandos independientes del TMC51 por lo que se requiere incluir estas subrutinas a la placa mediante el formado **$INCLUDE(subrutinasLCD.inc)**

Brando Miguel Palacios Mogollon

INSTRUCCIONES DE INICIALIZACIÓN

- inicioLCD : inicializa el LCD

lcall inicioLCD

INSTRUCCIONES DE CONTROL DEL CURSO (APUNTADOR) (R0)

- offCur: Apaga el cursor (lo desaparece)

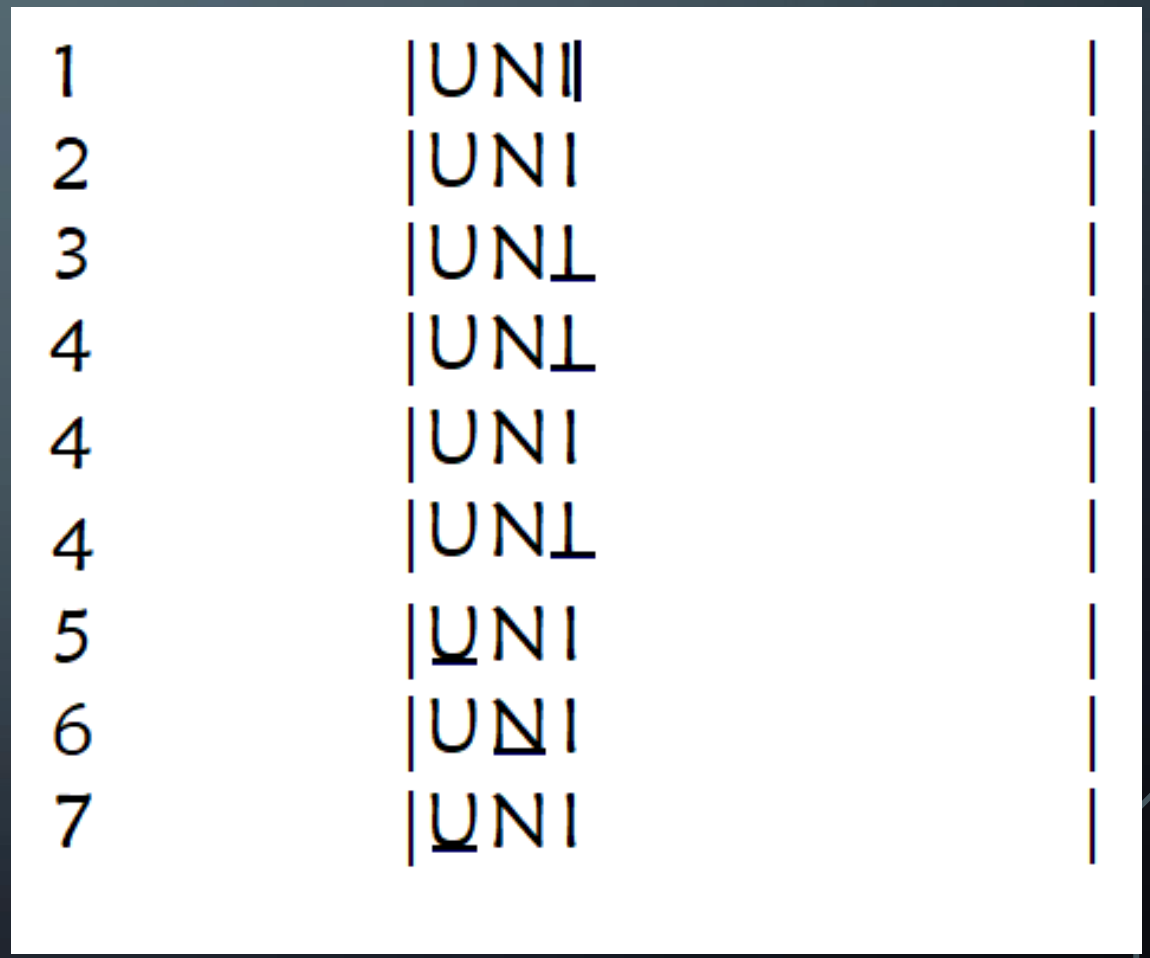
  Mov R0,#offCur

- lineCur: Cursor aparece como una línea.
- blinkCur: Cursor parpadeante.
- homeCur: Colocal el cursor en el lado izquierdo.
- shLfCur: Mueve el cursor 1 espacio a la izquierda.
- shRtCur: Mueve el cursor 1 espacio a la derecha.

Brando Miguel Palacios Mogollon

wrLCDcom4: escribe una palabra comando al LCD

```
1       ;Suponiendo que el código anterior a impreso en el terminal la palabra 'UNI'
2       Mov R0,#offCur
        lcall wrLCDcom4
3       Mov R0,#lineCur
        lcall wrLCDcom4
4        Mov R0,#blinkCur
        lcall wrLCDcom4
5       Mov R0,#homeCur
        lcall wrLCDcom4
6       Mov R0,#shRtCur
        lcall wrLCDcom4
7       Mov R0,#shLfCur
        lcall wrLCDcom4
```



Brando Miguel Palacios Mogollon

IMPRIMIR DATOS AL LCD

- wrLCDdata4: Escribe una palabra de datos al LCD.

EJM: Inprimir el numero 40 al LCD

```
1       mov A,#40
2       lcall dos_dígitos_decimales
3        mov R0,30h
4       lcall wrLCDdata4
5       mov R0,31h
6       lcall wrLCDdata4
```

```
1       Acumulador <-- 40
2       30h <- 4 (ascii) 31h <- 0 (ascii)
3       Registro 0 <- 30h <- 4 (ascii)
4       |4|                    |
5       Registro 0 <- 31h <- 0 (ascii)
6       |40|                   |
```

Brando Miguel Palacios Mogollon

- lcall placeCur4 ;localiza el cursor en los valores indicados
- lcall prtLCD4 ;Imprime palabras seleccionadas

EJM: ‘Arqui ez’

```
1       mov A,#1        ;Con Acumulador defines la fila que se va a imprimir
2       mov B,#4        ;Posicion 4 del cursor
3       lcall placeCur4 ;Coloca las direcciones de A  y B
4       lcall prtLCD4 ;Ímprimer el texto siguente
        db "Arqui ez",0
5       mov R0,#offcur          ;Oculta el cursor
        lcall wrLCDcom4         ;Lee el comando anterior
```

```
1       A <- 1 (FILA 1)
2       B <- 4 (Apunta a la posicion 4)
3       |________________|
        |________________|

4       |___Arqui_ez_____|
        |________________|

5       |___Arqui_ez_____|
        |________________|
```

Brando Miguel Palacios Mogollon