Escuela de "Ciencia de la Computación". Facultad de Ciencias - UNI. Lima - PERU.

SUBRUTINAS DEL TMC51

CURSO: "ARQUITECTURA DE COMPUTADORES" (CC212)

Prof.: LIC. Martín Cruz Salazar

```
; ========================================================
; subrutina chkbrk
; esta rutina chequea si se presionó Ctrl + C. Si es detectado el control
; es pasado de vuelta al programa monitor.
; ========================================================
;
chkbrk:  jnb   ri, nobrk              ; si no hay caracter entonces retorna
         mov   a,  sbuf               ; si lo hay consigue caracter del puerto serial
         clr   ri                     ; resetea el bit de estado rx
break:   cjne  a,  #03h,nobrk         ; si se ha presionado ctrl + C entonces
         lcall print                  ; muestra el mensaje 'break'
         db 0dh, 0ah," <break> ", 000h
         ljmp  return                 ; retorna al monitor
nobrk:   ret                          ; sino retorno normal

; ========================================================
; subrutina getchr
; esta rutina lee un caracter del puerto serie y lo guarda
; en el acumulador.
```

```
; ========================================================

;

getchr:  jnb  ri, getchr        ; espera hasta que el caracter sea recibido

        mov  a,  sbuf           ; consigue el caracter

        anl  a,  #7fh           ; mask off 8th bit

        clr  ri                 ; limpia el bit de estado serial

        ret



; ================================================================

; subrutina getbyt

; esta rutina lee un número ascii hexadecimal de 2 digitos

; desde el puerto serie. El resultado es retornado en el acumulador

; ================================================================

;

getbyt:

        lcall getchr            ; consigue el caracter ascii msb

        lcall ascbin            ; lo convierte a binario

        swap  A                 ; lo mueve a la mitad mas significativa del acumulador

        mov   B,  A             ; lo salva en B

        lcall getchr            ; consigue el caracter ascii lsb

        lcall ascbin            ; lo convierte a binario

        orl   A,  B             ; combina las dos mitades

        ret

; ========================================================
```

```
; subrutina ascbin
; esta rutina toma el caracter ascii pasado en el
; acumulador y lo convierte a un número binario de 4 bits
; que es retornado en el acumulador.
; ========================================================
ascbin:
        add   a,  #0d0h        ; si chr < 30 entonces error
        jnc   notnum
        clr   c                 ; chequea si chr es 0-9
        add   a,  #0f6h        ; lo ajusta
        jc    hextry            ; salta si chr no es 0-9
        add   a,  #0ah         ; si lo es entonces lo ajusta
        ret
hextry:
        clr   acc.5           ;
        clr   c                 ; chequea si chr es a-f
        add   a,  #0f9h        ; lo ajusta
        jnc   notnum            ; si no es a-f entonces error
        clr   c                    ; averigua si char es 46 o menos.
        add   a,  #0fah        ; ajusta el acumulador
        jc    notnum            ; si carry=1 entonces no es un número hex
        anl   a,  #0fh          ; limpia bits no usados
        ret
notnum:
        setb  errorf          ;
        ret
```

```
;================================================
;Subrutina prthex
;Esta rutina toma el contenido del acumulador y lo envía
; vía serie como 2 dígitos ascii hexadecimal.
;================================================
prthex:
        lcall binasc
    lcall sndchr
    mov A,R2
    lcall sndchr
    ret

;================================================
;Subrutina binasc
;Toma el contenido del acumulador y
;lo convierte en dos números ascii hexadecimales.
;El resultado es retornado en el acumulador y R2
;================================================
binasc:
        mov R2,A    ;se mueve a R2 para salvar el valor de A
    anl A,#0Fh  ;se convierte el dígito menos significativo
    add A,#0F6h ;lo ajusta
    jnc noadj1
    add A,#07h
noadj1:
```

```
        add A,#3Ah  ;lo hace ascii
      xch A,R2    ;pone el resultado en R2
        swap A          ;se convierte ahora el digito más significativo
        anl A,#0Fh
        add A,#0F6h ;lo ajusta
      jnc noadj2
      add A,#07h
noadj2:
         add A,#3Ah  ;lo hace ascii
      ret

;=====================================================
;Subrutina print
;toma la cadena inmediatamente que sigue a "lcall"
;y lo envia por el puerto serie. La cadena debe terminar
con un nulo(0).
;Esta subrutina retornará a la instrucción que sigue
;inmediatamente a la cadena.
;=====================================================
print:  pop dph ;coloca dirección de reterno en dptr
      pop dpl
prtstr:
         clr A
      movc A,@A+DPTR
      cjne A,#0h,mchrok
      sjmp prtdone
```

```
mchrok:
        lcall sndchr
    inc DPTR
    sjmp PRTSTR
prtdone:
        mov A,#1h
    jmp @A+DPTR

;=====================================
;Subrutina sndchr
;que envia un caracter
;que está contenido en el acumulador A
;=====================================
    sndchr:
        clr TI
        mov SBUF,A
    txloop:
        jnb TI,txloop
        ret

;================================================
;Subrutina init
;que configura el puerto serie
;a una velocidad de comunicación de 19200 baudios
;================================================
    init:
```

```
        mov PCON,#80H
        mov TMOD,#20H
        mov TCON,#41H
        mov TH1,#0FDH
        mov SCON,#50H
        ret

;=========================================================
;Subrutina display
;que convierte el valor del registro A
;en un patron de bits que corresponde a un número que
;se muestra en el display de 7 segmentos
;=========================================================
display:
        inc a           ;incremento el valor de A
        movc a,@a+pc    ;muevo a A el valor del contenido de la dirección de memoria (pc+a)
        ret             ;Retorno de la subrutina al programa principal
        db      0C0h    ;Con este código se muestra "0" en el display
        db      0F9h    ;Con este código se muestra "1" en el display
        db      0A4h    ;"2"
        db      0B0h    ;"3"
        db      99h     ;"4"
        db      92h     ;"5"
        db      82h     ;"6"
        db      0F8h    ;"7"
        db      80h     ;"8"
```

```
        db      90h     ;"9"
        db      88h     ; a
        db      83h     ; b
        db      0c6h    ; c
        db      0a1h     ; d
        db      86h     ; e
        db      8eh     ; f


; ================================================
;  Subrutina delay
;   retardo de un milisegundo
;   el acumulador contiene los milisegundos que se va retardar
;   2 microsegundos son reservados para la llamada
;   a esta subrutina.
;  input    : milisegundos a retardar en el acumulador
;  output   : nada
;  destruye : nada - usa A
; ------------------------------------------------
; 100h-a6h=5ah=(90)decimal
; 90 * 11 = 990 microsegundos
; mas 10 da 1 milisegundo
;
;                               microsegundos (cycles)
;                               ----------------------
delay:
```

```
      dec   a             ;    1
d_olp:
     push acc            ;    2          \
     mov   a, #0a6h    ;    1          |
                    ;              |
d_ilp:  inc   a        ;    1  \       |
      nop            ;    1  |       |
      nop            ;    1  |       |
      nop            ;    1  |       |
      nop            ;    1  |       |
      nop            ;    1  |- 11     | (acc-1)
      nop            ;    1  | cycles  |- msec
      nop            ;    1  |       |
      nop            ;    1  |       |
      jnz   d_ilp     ;    2  /       |
                   ;              |
      nop            ;    1          |
      nop            ;    1          |
      nop            ;    1          |
      pop   acc       ;    2          |
                   ;              |
      djnz  acc,d_olp ;  2           /

; necesita esperar 998 microsegundos más

      mov   a, #0a6h  ;    1
```

```
d_lp2:  inc   a       ;    1  \
        nop           ;    1  |
        nop           ;    1  |
        nop           ;    1  |
        nop           ;    1  |
        nop           ;    1  |- 11
        nop           ;    1  | cycles
        nop           ;    1  |
        nop           ;    1  |
        jnz   d_lp2    ;    2  /
        nop           ;    1
        nop           ;    1
        nop           ;    1
        nop           ;    1
        nop           ;    1

        ret           ;    2


; ================================================
;  Subrutina sdelay
; Nos dá un retardo de 1 segundo
; demora por 999998 microsegundos
; 2 microsegundos son reservados para
;  la llamada a esta subrutina.
```

```
;  input    : nada
;  output   : nada
;  destruye : nada - usa A
; ------------------------------------------------
; 100h-91h=6fh=(111)decimal
; 9008 * 111 = 999888 microsegundos
; mas 102 del segundo lazo
; mas 8 da 999998 microsegundos
;                                  microsegundos (ciclos)
;                                  --------------------------------
sdelay:
          push acc             ;   2
          mov  A, #91h      ;   1

sd_olp:
      inc   A          ; \
      lcall mdelay ; 1000 microsegundos |
      lcall mdelay ; 1000 microsegundos |
      lcall mdelay ; |
      lcall mdelay ; |
      lcall mdelay ; |
      lcall mdelay ; |
      lcall mdelay ; |- loop toma 9008 microsegundos
      lcall mdelay ; |
      lcall mdelay ; |
      nop         ; |
```

```
        nop         ; |
        nop         ; |
        nop         ; |
        nop         ; |
        jnz   sd_olp ;  2/
        mov   a, #33h       ;   1
sd_ilp:
        djnz  acc, sd_ilp    ; -loop takes 2*33h=66h=(102)dec
        pop   acc           ;   2
        ret                 ;   2


; ================================================
;  subrutina mdelay - retardo de un milisegundo
;  retardo por 998 microsegundos
;  donde 2 microsegundos son
;   reservados para la llamada a esta subrutina.
;  input    : none
;  output   : none
;  destruye : nada- usa A
; ------------------------------------------------
; 100h-a6h=5ah=(90)decimal
;  90 * 11 = 990
; mas 8 da 998 microsegundos
;
;                                   microseconds (cycles)
;                                    ----------------------
```

```
mdelay:
;3 microsegundos
        push acc            ;    2
           mov   a, #0a6h    ;    1
;990 microsegundos
md_olp:
        inc   a        ;    1  \
        nop            ;    1  |
        nop            ;    1  |
        nop            ;    1  |
        nop            ;    1  |
        nop            ;    1  |- 11  ciclos
        nop            ;    1  |
        nop            ;    1  |
        nop        ;    1  |
        jnz   md_olp  ; 2  /
;5 microsegundos
        nop        ;    1
        pop   acc    ;    2
        ret            ;    2


; ===================================
;  subrutina setintvec - fija vector de interrupción
;  entrada    : "a" contiene fuente de interrupción [0..5]
;                  "dptr" contiene dirección del ISR
;  salida   : nada
```

```
;  destruye : a, dptr
; ---------------------------------------------------------------
setintvec:
        push dpl                ; salva dirección de ISR
        push dph    ; en la pila
        anl  a, #0fh            ; solo para estar seguro
        rl   a                  ; multiplica por 4
        rl   a
        mov  dph, #0ffh ; tabla del vector en ram
        mov  dpl, a         ; dptr apunta al vector de la tabla
        mov  a, #2          ; coloca la instrucción ljmp
        movx @dptr, a
        inc  dptr
        pop  acc            ; extrae el byte alto de la dirección ISR
        movx @dptr, a
        inc  dptr
        pop  acc            ; extra el byte bajo de la dirección ISR
        movx @dptr, a  ; nuevo vector de interrupción localizado
        ret
```