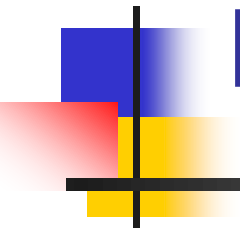


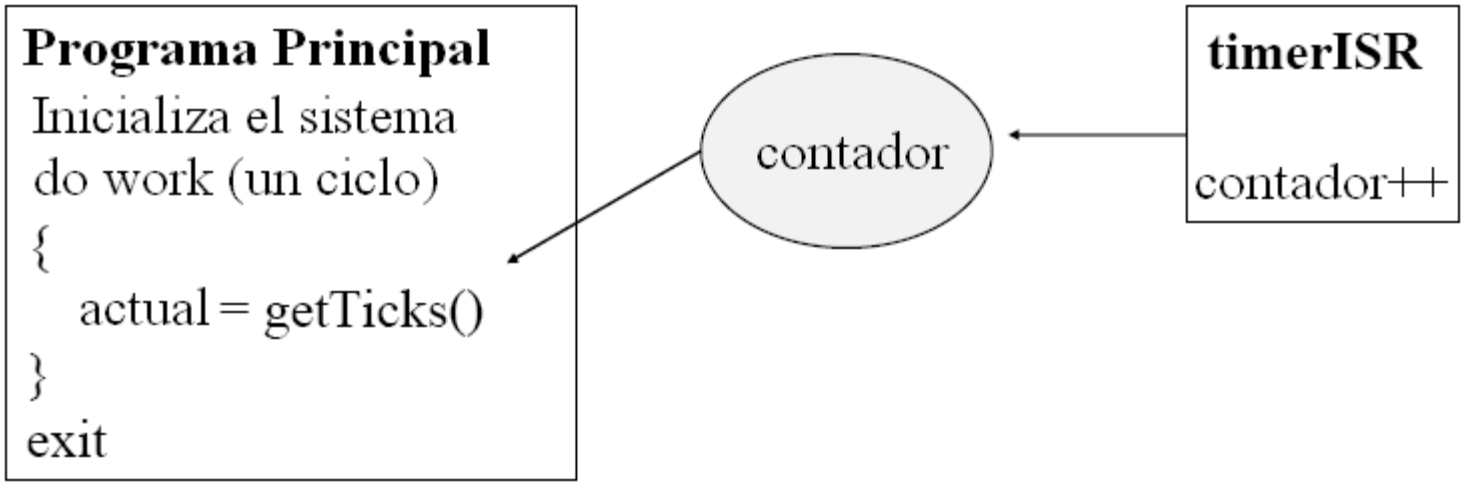
Sistemas de Procesamiento de Datos

Assembler (parte 6)

Profesor: Fabio Bruschetti
Aydte: Pedro Iriso
Ver 2019

Cronómetros (Programa)

- Generación de un reloj de tiempo real a 20 Hz
  1) Desarrollo de la arquitectura del programa
     - Dos componentes de software - programa principal y la rutina ISR para timer
     - Comparten una variable contador de 32-bits
     - La rutina ISR, incremente a la variable contador por cada tick
     - El programa principal lee la variable cuando sea necesario usando:
       - double DameTicks(void) que su implementación accede a la variable



Cronómetros (Programa)

- Generación de un reloj de tiempo real a 20 Hz

  2) Determinar los valores para el Timer 0 del 8253

  - Escribir al registro de control del 8253 (en 43h)
    - SC1 SC0 = 00 (selecciona el timer 0)
    - RL1 RL0 = 11 (byte Menos Signif. luego Mas Signif.)
    - M2 M1 M0 = 011 (Onda cuadrada)
    - BCD = 0 (contador de 16-bit)
    - Registro de Control 00 11 011 0 --> 36h
  - Determinar el valor inicial y el de recarga
    - factor de escala: 1.19318 / 20 Hz = 59659d = **0E90Bh**
  - Escribir al registro de datos del timer 0 (en 40h)
    - Primero se escribe el Byte Menos Significativo = 0Bh
    - Luego el Byte Mas significativo = E9h

Cronómetros (Programa)

- Generación de un reloj de tiempo real a 20 Hz
  - Código Fuente:
    - ; Symbolic Definitions
      ; PIC registers and constants

```
PIC_COMMAND_REG         equ     20H
PIC_IMR_REG             equ     21H
EOI                     equ     20H
;Timer registers and constants
TIMER_0_REG             equ     40h
TIMER_CTRL_REG          equ     43H
TIMER_0_MODE  equ       36H
```

    - .data
      ; Program modifies state of system; need variables to save
      ; state to allow restoring of state upon exit

```
old_pic_imr             db      ?
old_int8_vector_offset  dw      ?
old_int8_vector_segment dw      ?
; 32-bit count: shared by ISR and main program
count_low               dw      ?
count_high              dw      ?
```

Cronómetros (Programa)

- Generación de un reloj de tiempo real a 20 Hz
  - Código Fuente:
    - .code
      MAIN PROC
      ; Initialize data structures used by ISR
      SUB AX , AX ; trick! AX = 0 !
      MOV count_low , AX ; tick count = 0
      MOV count_high , AX
      CLI ; Disable interrupts while installing and setting up timer. Why ?
      ; Install Timer ISR at interrupt type = 8 (As before)
      ; Program timer 0 to interrupt at 20 Hz
      MOV AL , TIMER_0_MODE ; Control Register
      MOV DX , TIMER_CTRL_REG
      OUT DX , AL
      MOV AL , 0BH ; scaling factor = E90BH
      MOV DX , TIMER_0_REG
      OUT DX , AL; write low byte
      MOV AL , 0E9H
      OUT DX , AL; write high byte

Cronómetros (Programa)

- Generación de un reloj de tiempo real a 20 Hz
  - Código Fuente:
    - ; Enable TIMERinterrupts at the PIC (As before)

```
        MOV DX , PIC_IMR_REG
        IN AL , DX
        MOV old_pic_imr, AL          ; for later restore
        AND AL , 0FEH                ; clear bit 0 of IMR
        OUT DX , AL
; Enable interrupts at the processor
        STI
        forever: ….                  ; Main loop of program
        CALL get_ticks               ; Returns tick count in dx:ax
        ; Use value as needed
        JNE forever
exit :          ; Restore state before returning to DOS (not shown)
```

Cronómetros (Programa)

- Código Fuente:
  - timerisr PROC FAR

```
        STI                     ; Re-enable ints
        PUSH DS                 ; Save EVERY register used by ISR
        PUSH DX
        PUSH AX
        MOV AX , @data
        MOV DS , AX
        ADD count_low,1
        JNC t1
        INC count_high
t1:     CLI                     ; Lock out all intsuntil IRET
        MOV AL , EOI            ; Send EOI to PIC
        MOV DX , PIC_COMMAND_REG
        OUT DX , AL
        POP AX                  ; Restore registers
        POP DX
        POP DS
        IRET
```

  - timerisr ENDP

Cronómetros (Programa)

- Código Fuente:
  - get_ticks PROC NEAR

```
        CLI                     ; Lock out intswhile accessing shared data
                                ; (ensure mutual exclusion)
        MOV DX , count_high             ; Return 32 bit count
        MOV AX , count_low
        STI                     ; Re-enable ints
        RET
get_ticks ENDP
```