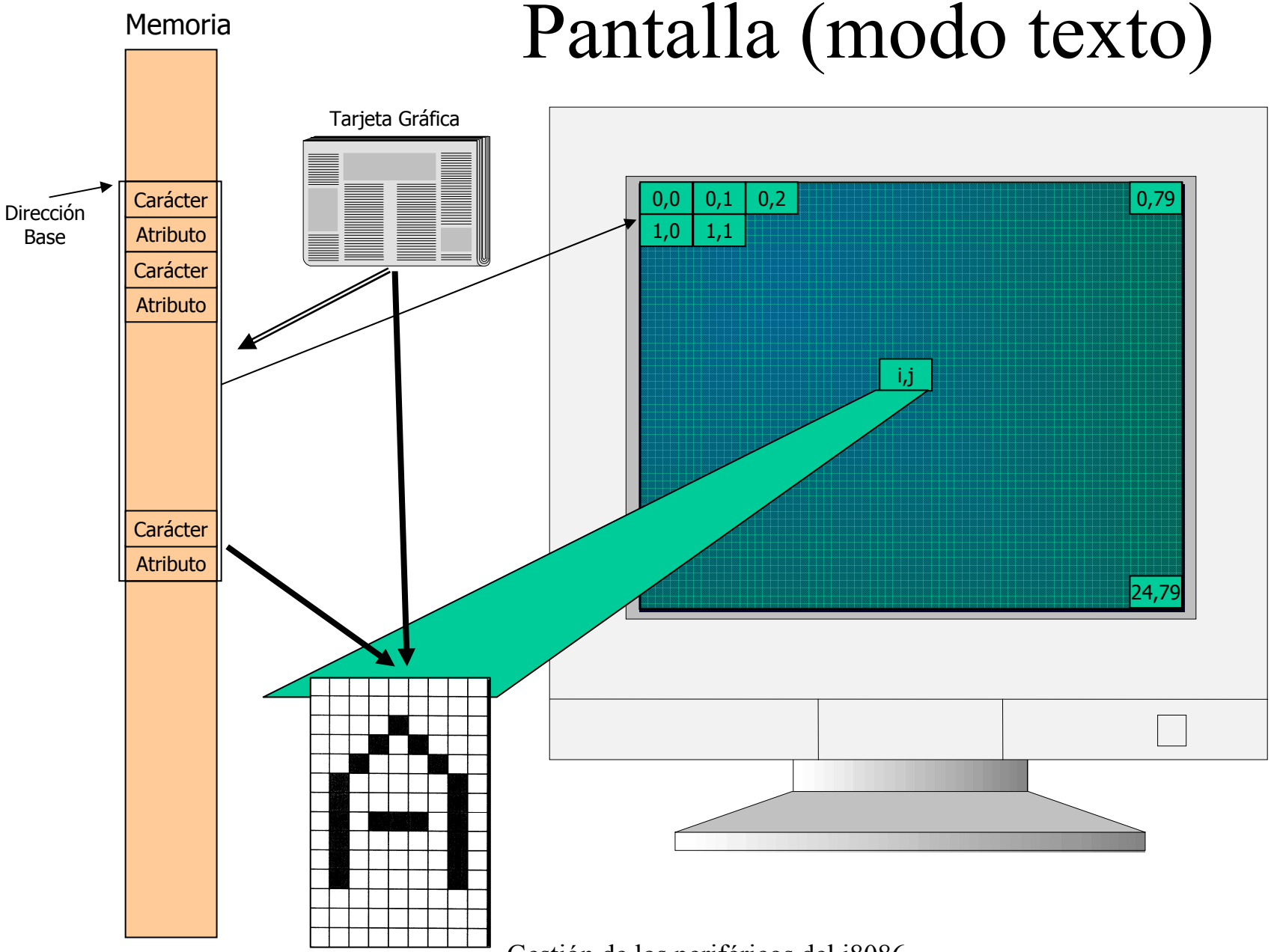
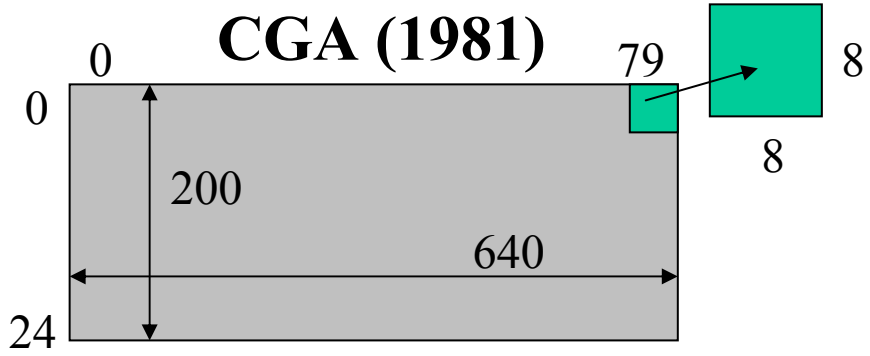
Gestión de los periféricos del i8086

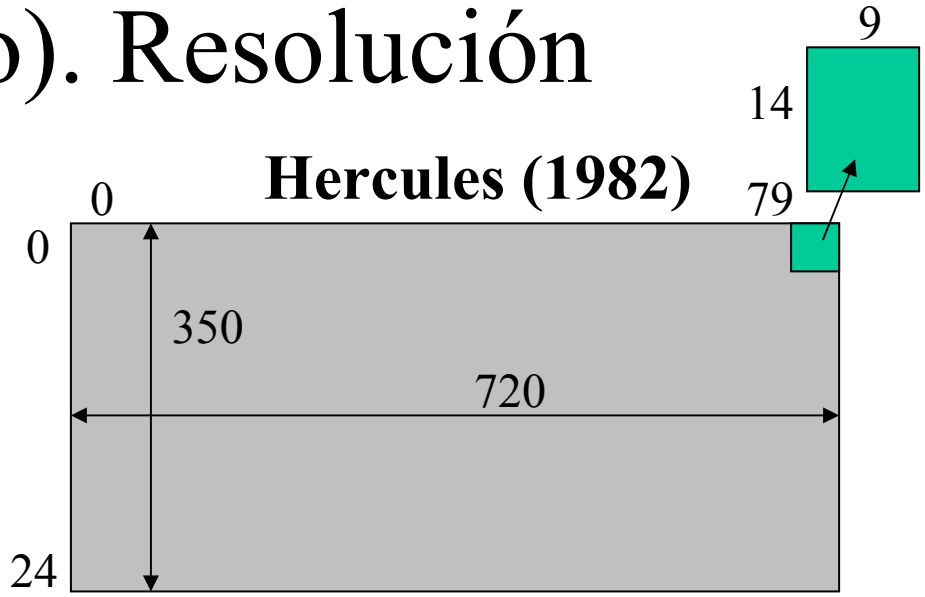
Arquitectura Computadores I

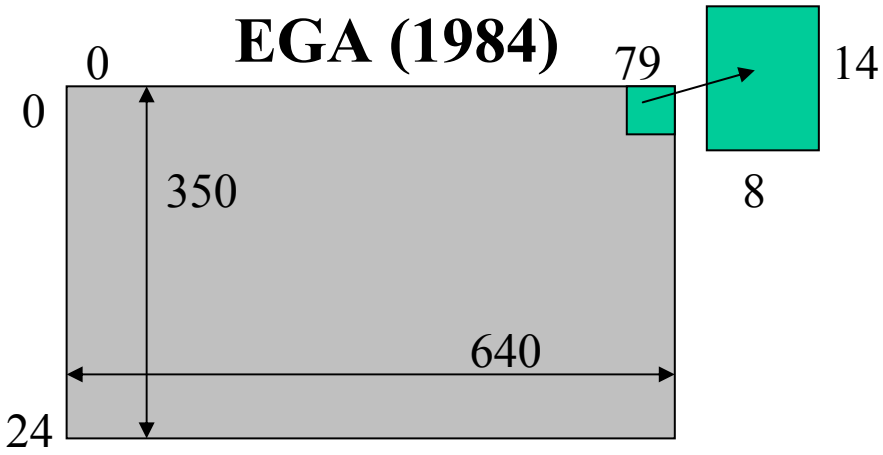
Pantalla (modo texto)

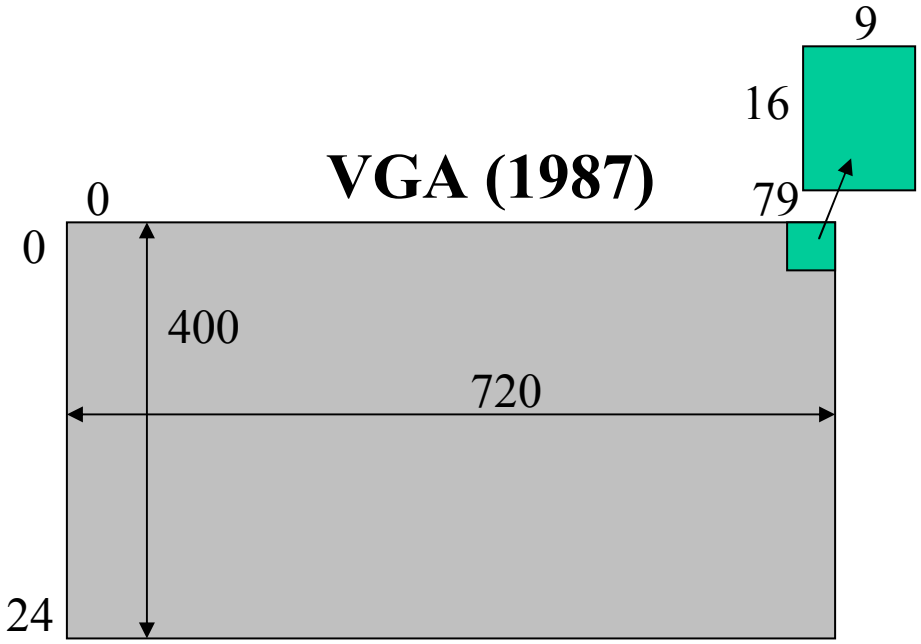


Pantalla (modo texto). Resolución



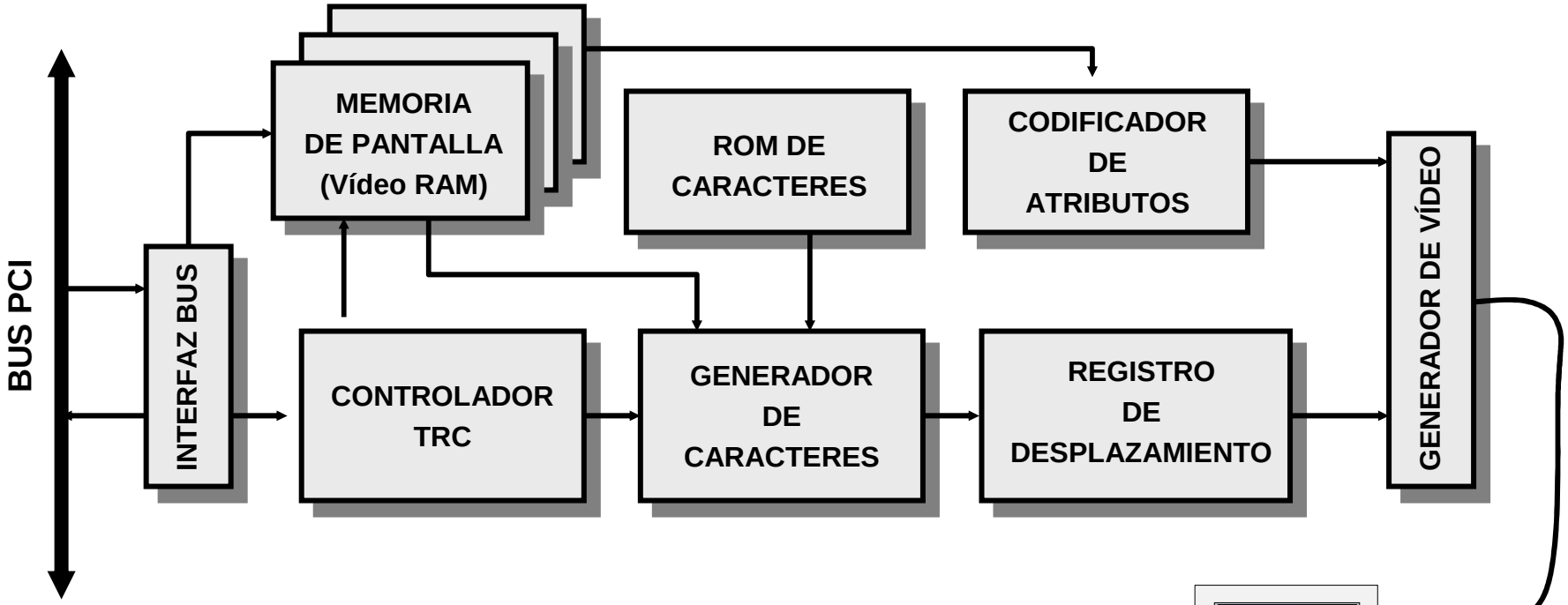


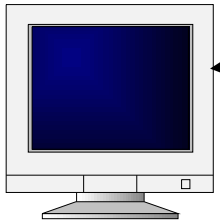




SVGA (1989) → 60f x 132c

Pantalla (modo texto)





Referencias:

1.- "The Indispensable PC Hardware Book",
H.P. Messmer, Addison-Wesley, 1997.

2.- "Programmer's Guide to the EGA, VGA, and Super VGA Cards,
R.F. Ferraro, Addison-Wesley, 1994.

3.- "Apuntes de la Asignatura: Periféricos e Interfaces", J.M. Valiente,
Dpto. de Informática de Informática de Sistemas y Computadores, UPV, 2000.

4.- "Solucionario del Programador para IBM PC, XT, AT y Compatibles, R. Jourdain,
Anaya Multimedia, 1986.

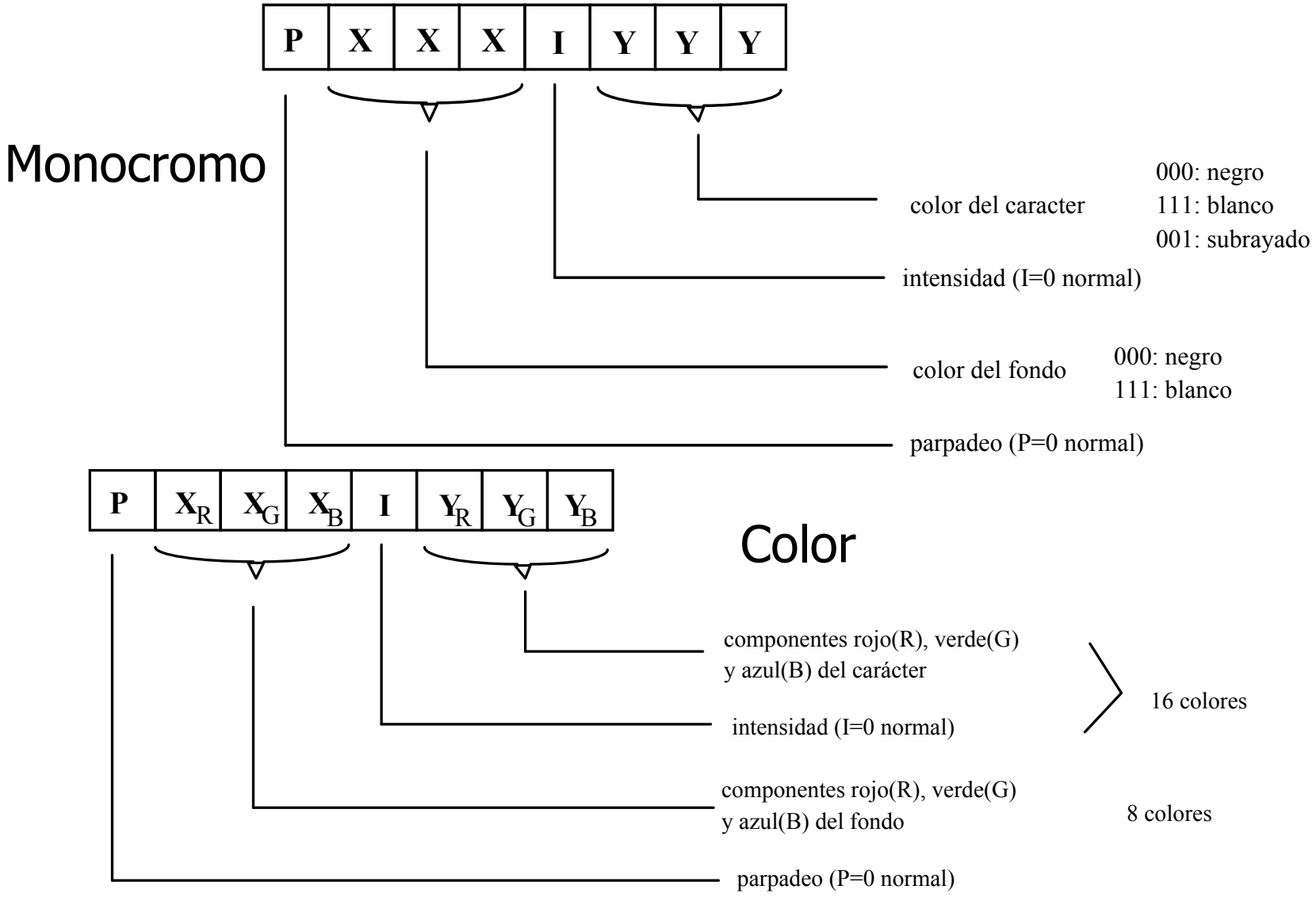
Pantalla (modo texto)

- Periférico mapeado en memoria
  - @base depende de la tarjeta gráfica
    - consultar el modo de vídeo activo en @0449H
      - si el modo es 7 → @base es @B000H
      - si el modo es 2 ó 3 → @base es @B800H

- Matriz de 25 filas y 80 columnas: carácter+atributo
  - *@carácter(i,j) = @base + i*80*2+j*2*
  - *@atributo (i,j) = @base + i*80*2+j*2+1*

- El número de filas y columnas depende de la resolución

Pantalla (modo texto). Atributos



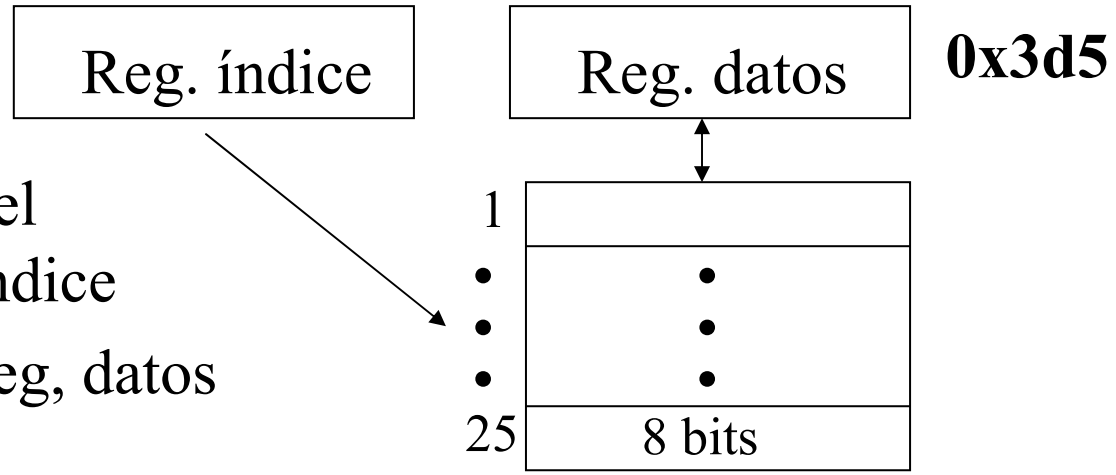
Pantalla (modo texto). Atributos

Color

I	R	G	B	COLOR
0	0	0	0	negro
0	0	0	1	azul
0	0	1	0	verde
0	0	1	1	cyan
0	1	0	0	rojo
0	1	0	1	magenta
0	1	1	0	marrón
0	1	1	1	blanco
1	0	0	0	gris
1	0	0	1	azul brillante
1	0	1	0	verde brillante
1	0	1	1	cyan brillante
1	1	0	0	rojo brillante
1	1	0	1	magenta brillante
1	1	1	0	amarillo
1	1	1	1	blanco brillante

Cursor

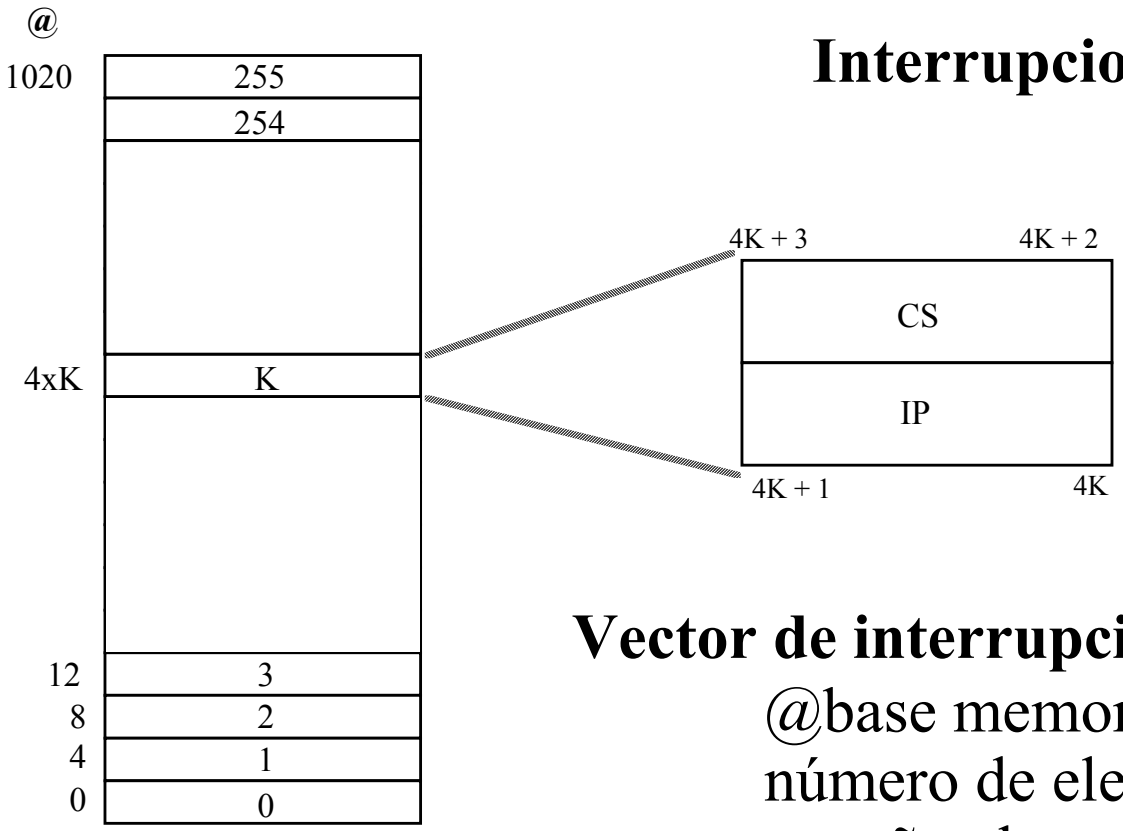
- Controlador tarjeta gráfica
  Motorola 6845 CRTC *(Cathode Ray Tube Controller)*



  - Primero → indicar el registro en el reg. índice
  - Luego → datos al reg, datos

- **Posición** del cursor: valor entre 0 y 1999 (25x80)
  →registros 14 (parte alta del valor) y 15 (parte baja del valor)
- **Forma** del cursor:
  →gordo: línea de comienzo 0, línea de fin: máxima dimensión matriz pixels
  →normal: misma línea de comienzo y de fin
  →registros 10 (línea de comienzo) y 11 (línea de fin)

Interrupciones en el i8086

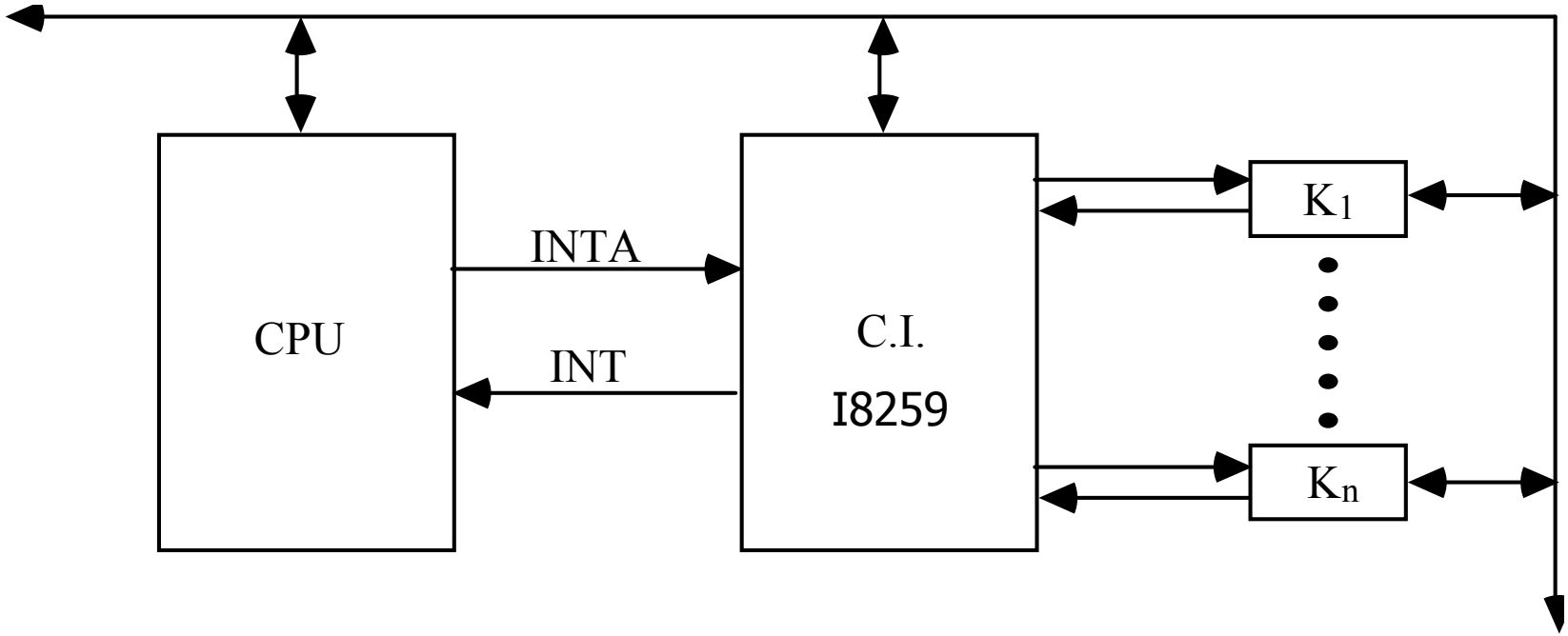


Interrupciones vectorizadas

Vector de interrupciones:

@base memoria: 0000H
número de elementos: 256
tamaño elemento: 4bytes

Interrupciones en el i8086

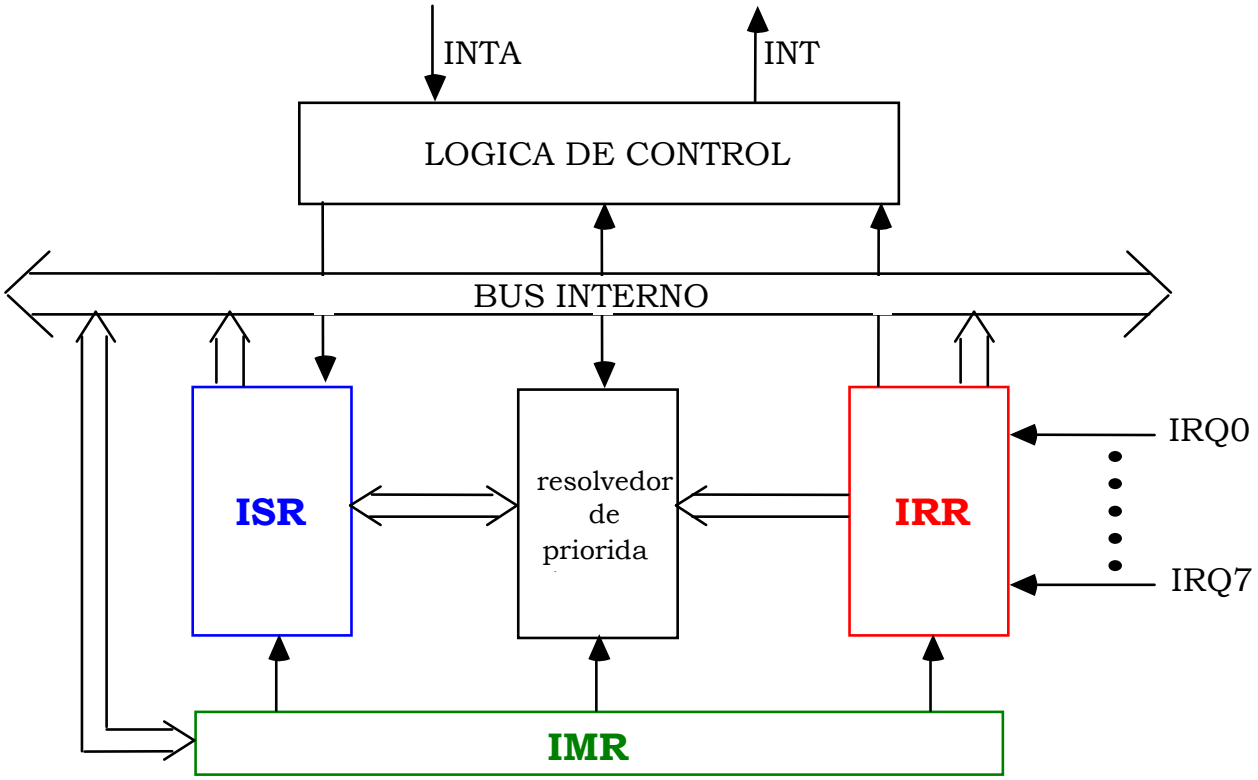


→ i8086 tiene **3 señales** para la gestión de interrupciones:

- **INT** (*INTerrup request*): petición de interrupción
- **INTA** (*INTerrup Acknowledge*): reconocimiento de interrupción
- **NMI** (*Non Maskable Interrupt*): interrupción no evitable

→controlador de interrupciones i8259

Controlador de interrupciones i8259



IRQ	DISPOSITIVO
0	reloj del sistema
1	teclado
2	reservada
3	línea serie COM2
4	línea serie COM1
5	controlador de disco duro
6	controlador de disco flexible
7	puerto paralelo 1 (impresora)

IRR *Interrupt Request Register* Registro de peticiones de interrupción.

ISR *In-Service Register* Registro indicador de interrupción en servicio

IMR *Interrupt Mask Register* Registro máscara de interrupciones

Controlador de interrupciones i8259

- Registro IRR
  - indica qué líneas *IRQi* se han activado: la activación de *IRQi* activa el bit *i* de IRR
- Registro ISR
  - indica qué interrupciones se están sirviendo: si el bit *i* de ISR está activo, se está sirviendo la petición *i*
- Registro IMR
  - indica qué interrupciones están permitidas: si el bit *i* de IMR está activado, la interrupción *IRQi* NO está permitida

Controlador de interrupciones i8259

- Características del i8259
  - las líneas IRQ se activan por flanco
  - prioridad fijada: IRQ0 > IRQ1 > ... > IRQ7
  - identificación de la rutina de servicio: **0000 1iii**, siendo *iii* el número de la línea IRQ activada (IRQ0 a IRQ7). Por tanto, entradas 08H a 0FH del vector de interrupciones

- Programación de i8259
  - registro de control (ISR/IRR): @20H dentro del espacio E/S
  - registro de máscara (IMR): @21H dentro del espacio E/S

Controlador de interrupciones i8259

- Registro de máscara IMR
  - lectura/escritura en la @21H de E/S
  - utilización: por ejemplo, inhibir alguna IRQ por programa

```
CLI                     ;inhibir interrupciones
IN AL,21H
OR AL,00000010B         ;inhibe la IRQ1 → teclado
OUT 21H, AL
STI                     ;permitir interrupciones

DisableInts();          //código equivalente en C
var=InPort(0x21);
var=var | 0x02;
OutPort(0x21,var);
EnableInts();
```

Controlador de interrupciones i8259

- Registro de control ISR/IRR (@20H de E/S)
  - utilización:

    1. fin de la rutina de servicio a una interrupción, **EOI**:

    → la rutina de servicio debe finalizar escribiendo un 20H en la @20H para indicar que se ha tratado la interrupción en curso

```
MOV AL,20H
OUT 20H, AL
```

```
OutPort(0x20,0x20);
```

así se avisa al i8259 para que dé paso a la siguiente interrupción en prioridad (IRQ con índice más alto)

Controlador de interrupciones i8259

- Registro de control ISR/IRR (@20H de E/S)
  - utilización:

    2. lectura de los registros ISR/IRR

    → seleccionar IRR: escribir el valor 0AH en 20H

    ```
    MOV AL,0AH          OutPort(0x20,0x0A);
    OUT 20H, AL
    ```

    → seleccionar ISR: escribir el valor 0BH en 20H

    ```
    MOV AL,0BH          OutPort(0x20,0x0B);
    OUT 20H, AL
    ```

    → una vez seleccionado el registro, la siguiente lectura sobre 20H dará el contenido de ese registro

    ```
    IN AL,20H           var=InPort(0x20);
    ```

Controlador de interrupciones i8259: **protocolo con la CPU**

- *IRQi* periférico → controlador (activa bit *i* de IRR)
  - + Resolvedor de prioridad: comprueba si la interrupción *i* está enmascarada (IMR) y si se está sirviendo una de mayor prioridad (no existe $j<i$, tal que bit *j* de ISR está a 1)
  - + Si resuelve atender la interrupción, activa la señal INT hacia la CPU

- CPU
  - + Al recibir INT, si IF==1 (int. permitidas), activa dos veces la señal INTA → indica al controlador que la interrupción va a ser permitida

Controlador de interrupciones i8259: protocolo con la CPU

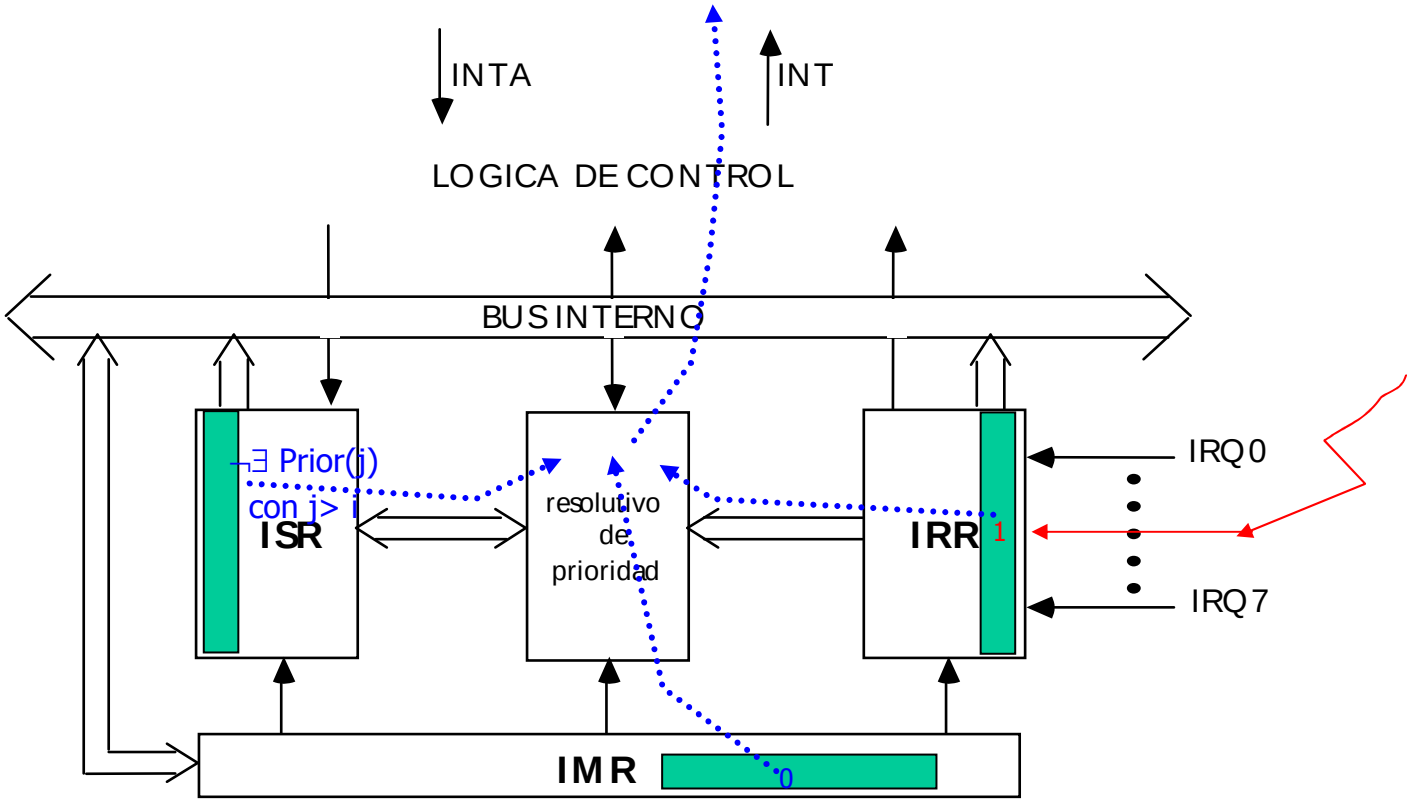
- Controlador 1ª INTA
  - + Elimina la interrupción de la lista de pendientes (IRR*i* a 0)
  - + Añade la interrupción a la lista de interrupciones en servicio (ISR*i* a 1)
- Controlador 2ª INTA
  - + Desactiva la señal INT
  - + Envía por el bus de datos la información acerca de la rutina de servicio a la interrupción (índice en el vector de interrupciones, *n*): *n=0000 1iii*

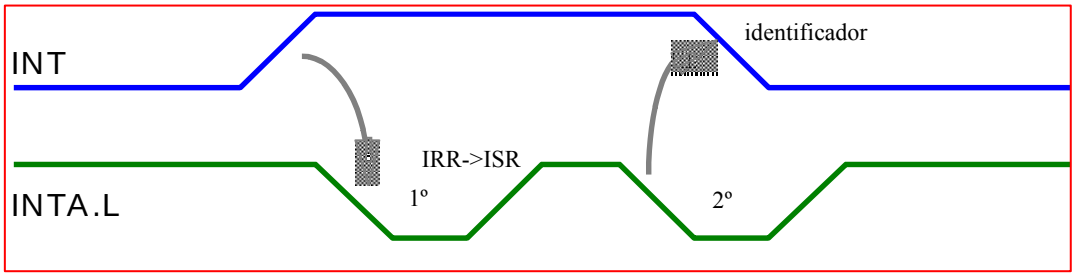
Controlador de interrupciones i8259: protocolo con la CPU

- CPU: al recibir la información
  - + Guarda en la PILA: PSW, CS e IP
  - + Pone a 0 el flag IF
  - + Actualiza CS e IP con el contenido de la entrada en el vector de interrupciones: *IP=MEM[n*4]* y *CS=MEM[n*4+2]*

    → se ejecuta la rutina de servicio
- La rutina finaliza con EOI e IRET
- Controlador: al recibir EOI
  - + Elimina la interrupción de la lista de interrupciones en servicio (ISR*i* a 0)

Controlador de interrupciones i8259: protocolo con la CPU

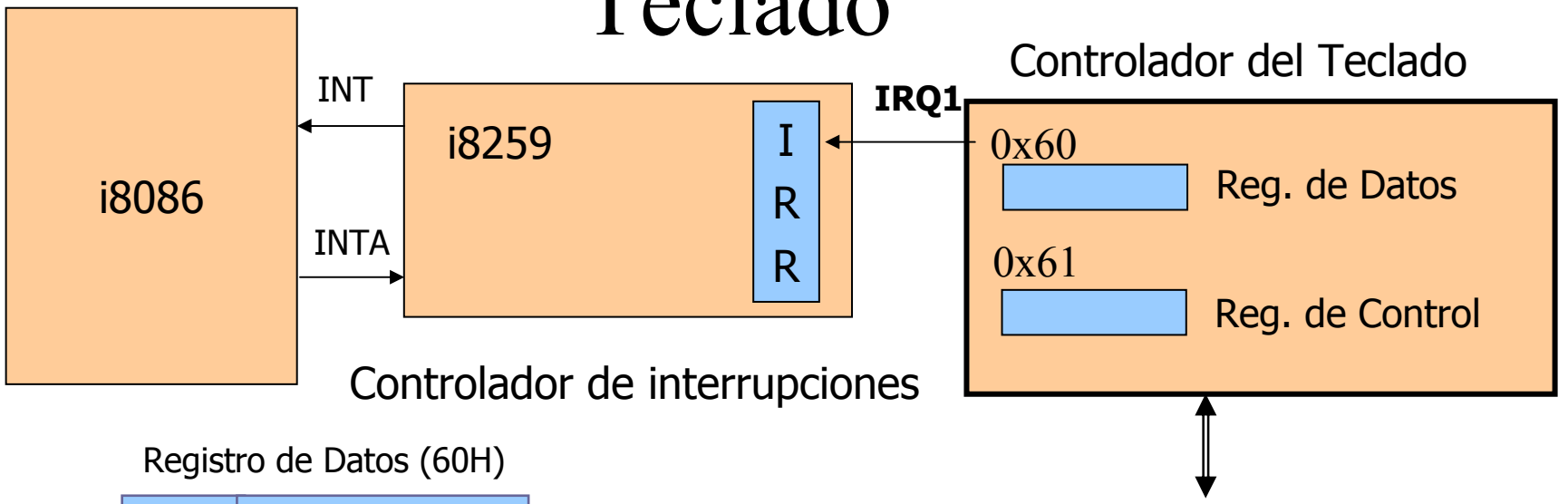




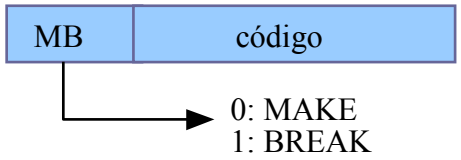
Teclado

- Controlador por el micro i8048
  - registro de datos en la @60H de E/S
  - registro de control en la @61H de E/S
  - buffer para 20 teclas
- Al pulsar una tecla, el controlador genera dos códigos de rastreo o *scan code* (!=código ASCII)
  - pulsación o *MAKE* / liberación de la tecla o *BREAK*
  - códigos de 8 bits, que se diferencian en el bit de más peso: 0 para MAKE y 1 para BREAK
- La CPU traduce el código de rastreo al carácter ASCII, dependiendo de la tabla de traducción del teclado

Teclado



Registro de Datos (60H)



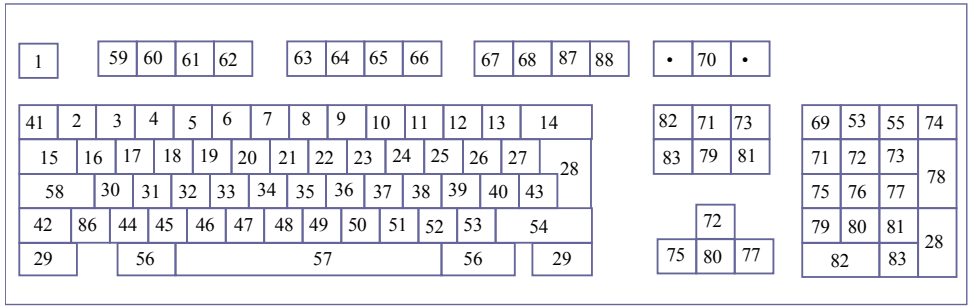
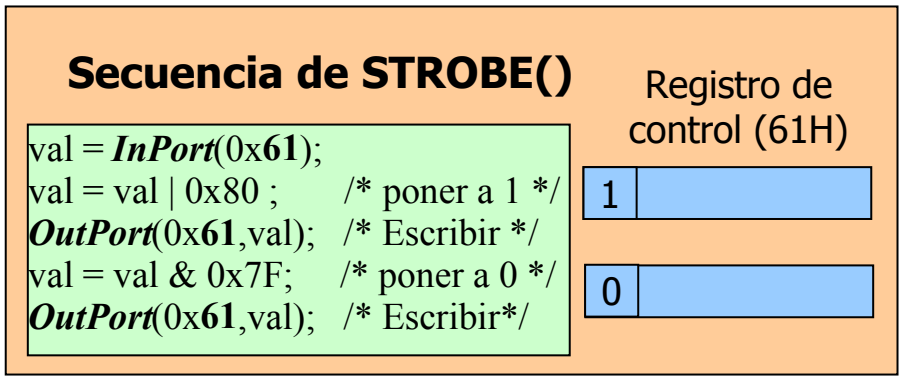


Tabla de traducción a código ASCII

```
unsigned char TABLA_ASCII[] = {
    '*', 27 , '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '\", '-',   8,
     9 ,'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', '`', '+',  13, '*',
    'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', '¥', '*', '§', 14, '‡',
    'Z', 'X', 'C', 'V', 'B', 'N', 'M', ',', '.', '-',  15, '*', '*', ' ', '*',
    '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '7', '8', '9',
     '-', '4', '5', '6', '+', '1', '2', '3', '0',  '.', '*', '*', '<', '*', '*', '*', '*',
};
```

Teclado

- Sincronización con la CPU:
  - al pulsar o liberar una tecla, i8048 genera **IRQ1**
  - sincronización por **encuesta** o por **interrupciones**
  - tras el tratamiento, es necesaria una secuencia **STROBE** sobre el bit 7 (1/0) del registro de control, para desactivar IRQ1



Teclado por encuesta

- Sincronización por encuesta:
  - inhibir interrupciones del teclado → bit 1 de IMR a 1 (i8259)
  - encuesta del teclado, ¿tecla pulsada o liberada?
    → testear bit 1 de IRR (IRQ1 activada)
  - leer el registro de datos de i8048, identificar MAKE o BREAK consultando el bit de más peso
  - tratar el caracter
  - secuencia de STROBE
  - habilitar interrupciones del teclado → bit 1 de IMR a 0

Teclado por encuesta

```
Mi_irr = Leer_IRR();
while (bit_1(Mi_irr) == 0)
        Mi_irr = Leer_IRR();
Codigo_Rastreo = Leer_Reg_datos_Teclado();
if (Make(Codigo_Rastreo) )
        Cod_ASCII = Tabla(Codigo_Rastreo);
Strobe();
```

Encuesta del teclado

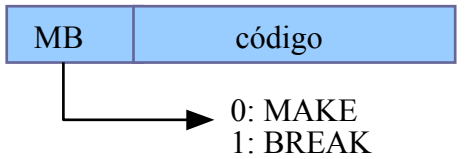
Leer_IRR()

```
OutPort(0x20, 0x0A);    // Selec. el reg. IRR
val = InPort(0x20);  // Leer el IRR
```

Leer_Reg_Datos_Teclado

```
CodTecla = InPort(0x60)
```

Registro de Datos (60H)




```
unsigned char TABLA_ASCII[] = {
   '*', 27 , '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '\', '-',   8,
    9 ,'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', '`', '+',  13, '*',
   'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', '¥', '*', '§', 14, '‡',
   'Z', 'X', 'C', 'V', 'B', 'N', 'M', ',', '.', '-',  15, '*', '*', ' ', '*',
   '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '7', '8', '9',
    '-', '4', '5', '6', '+', '1', '2', '3', '0',  '.', '*', '*', '<', '*', '*', '*', '*',
};
```

Secuencia de STROBE()

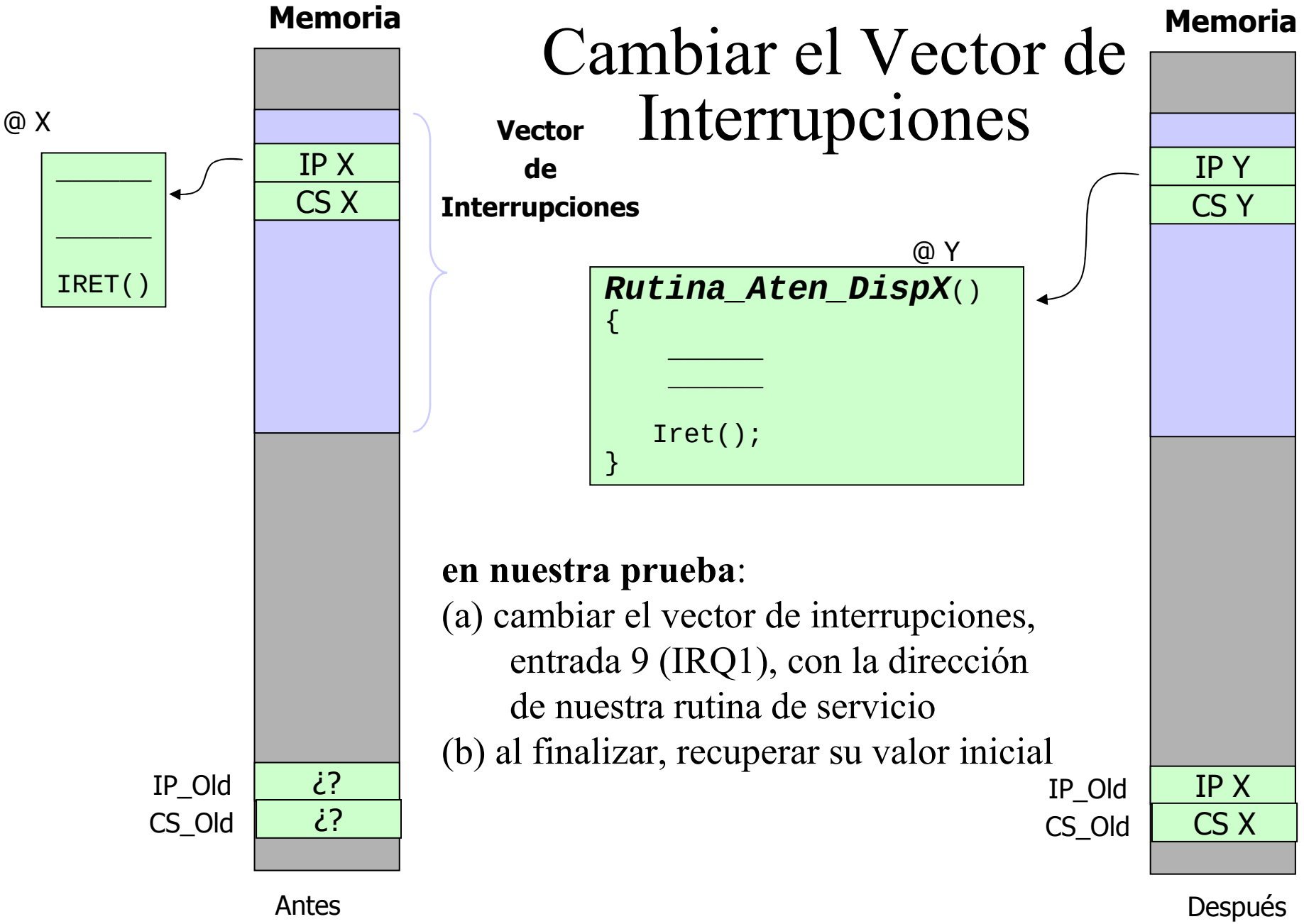
```
val = InPort(0x61);
val = val | 0x80 ;      /* Poner a 1 */
OutPort(0x61,val);   /* Escribir */
val = val & 0x7F;      /* Poner a 0 */
OutPort(0x61,val);   /* Escribir */
```

Registro de control (61H)

Teclado por interrupción

- Sincronización por interrupciones:
  - cuando IRQ1 interrumpa, la CPU dará paso a la rutina de servicio de la interrupción del teclado
  - rutina de servicio:
    - → leer código de rastreo
    - → detectar MAKE o BREAK
    - → si MAKE, traducir el código de rastreo
    - → STROBE
    - → EOI
    - → IRET

Cambiar el Vector de Interrupciones



en nuestra prueba:

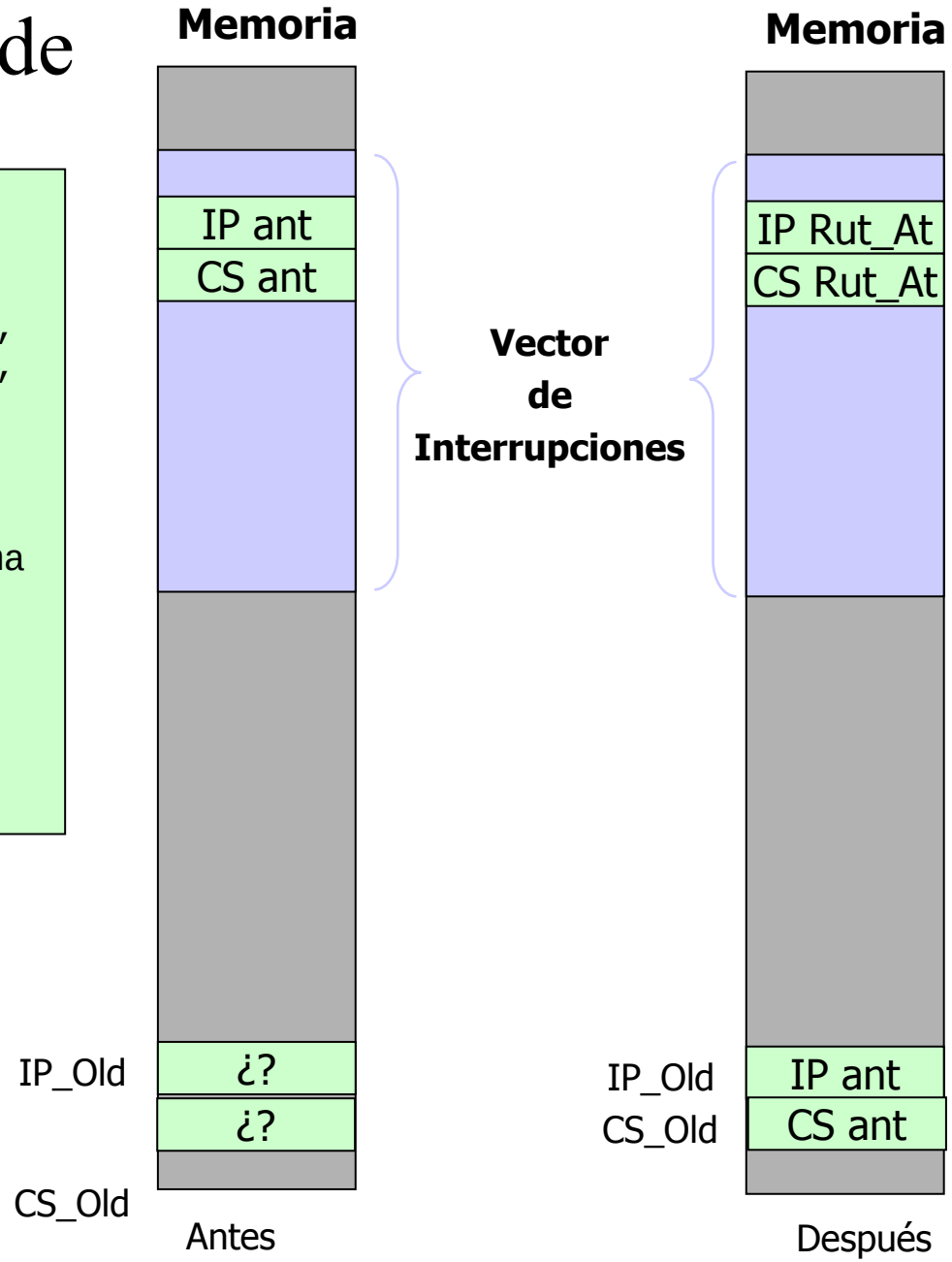
(a) cambiar el vector de interrupciones, entrada 9 (IRQ1), con la dirección de nuestra rutina de servicio

(b) al finalizar, recuperar su valor inicial

Cambiar el Vector de Interrupciones

```
main()
{
    Cambia_VI(Num_Int,
         IP(Rut_Atencion_DispX),
         CS(Rut_Atencion_DispX),
         &IP_Old,
         &CS_Old);


   // Código de nuestro programa


    Recupera_VI(Num_Int,
                IP_Old,
                CS_Old);
}
```

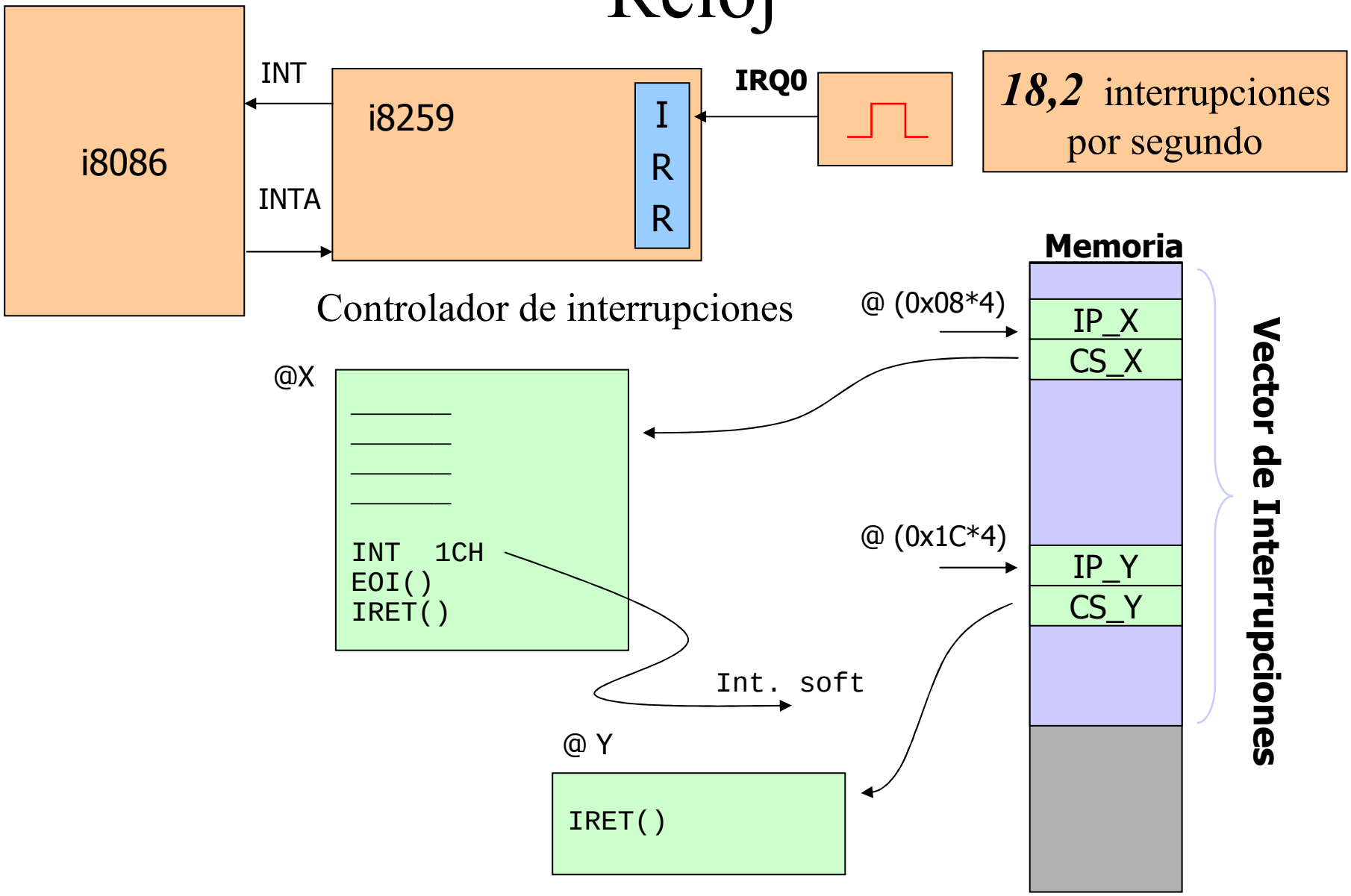
```
Rutina_Atencion_DispX()
{
   Tratar_la_inter_DispX();

   Strobe_DispX(); // No siempre
   Eoi();
   Iret();
}
```



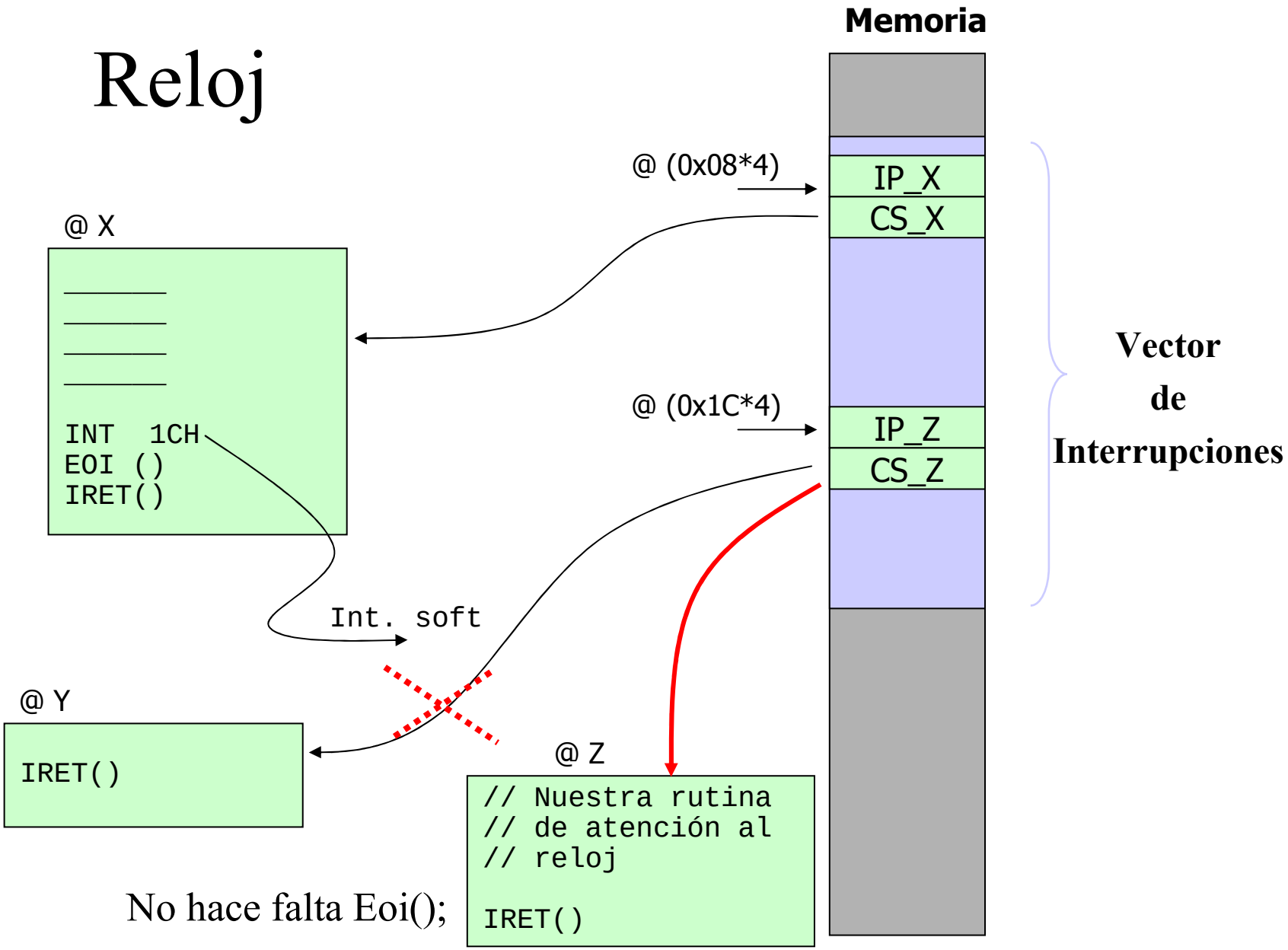
Reloj

- Controlador por el micro i8253
  - IRQ0 (entrada 08H del vector de interrupciones)
  - frecuencia interrupción: 18,2 veces/segundo
- Al interrumpir, se ejecuta una rutina de servicio que no se puede cambiar (controla parámetros del sistema)
  - la rutina de servicio ejecuta INT 1CH (interrupción software) al finalizar
  - la rutina que se ejecuta (entrada 1CH del vector de interrupciones) tiene como instrucción única IRET
- Nuestra rutina de servicio al reloj suplantará a la rutina de la entrada 1CH en el vector de interrupciones
  - se ejecutarán las dos rutinas: sistema + nuestra

Reloj


i8086
INT
INTA
i8259
I
R
R
IRQ0
18,2 interrupciones por segundo
Controlador de interrupciones
Memoria
@ (0x08*4)
IP_X
CS_X
@X
INT 1CH
EOI()
IRET()
@ (0x1C*4)
IP_Y
CS_Y
Vector de Interrupciones
Int. soft
@ Y
IRET()

Reloj



No hace falta Eoi();