

Unidad 5

Entrada / Salida

SISTEMAS BASADOS EN MICROPROCESADORES

Índice

5. Entrada / Salida.

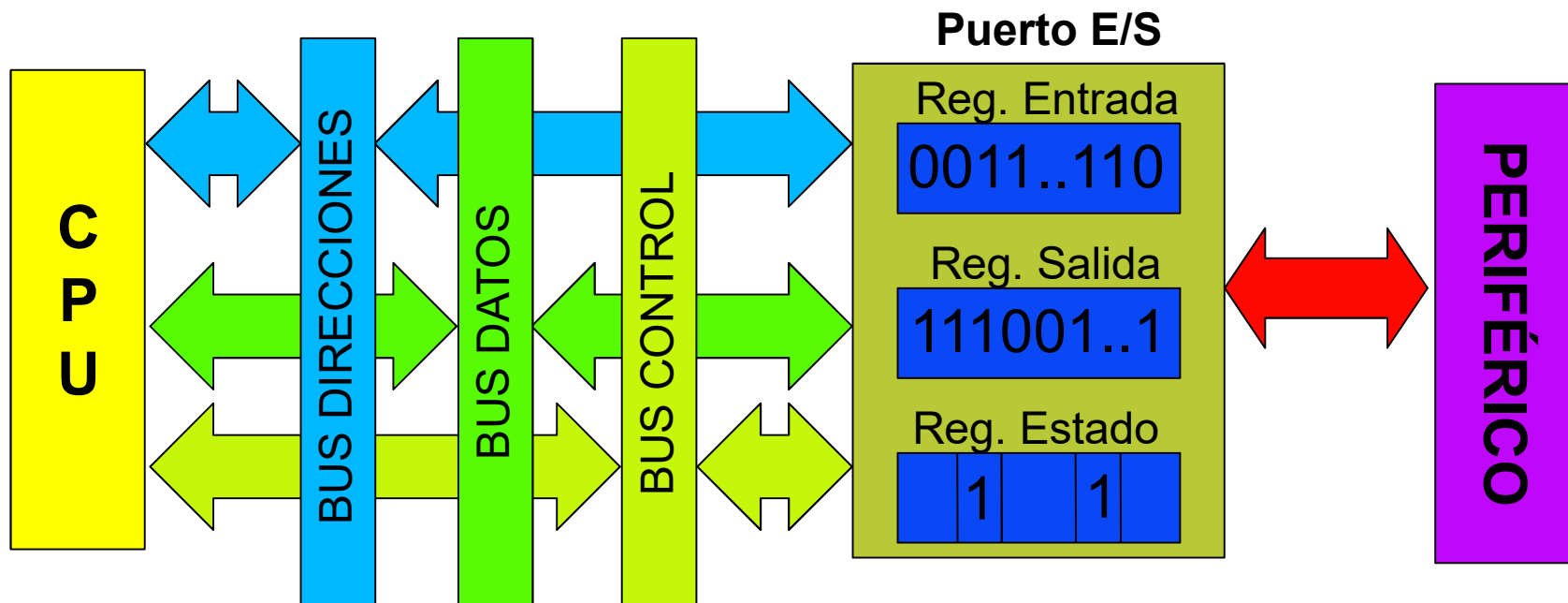
- 5.1. Técnicas de programación de entradas y salidas (E/S).
- 5.2. Sondeo.
- 5.3. Interrupción.
- 5.4. DMA.
- 5.5. Gestión y programación de las interrupciones en el 80x86: el controlador programable de interrupciones 8259A.

5.1. Técnicas de programación de E/S (I)

- Un sistema basado en microprocesador debe transferir datos desde (entrada) y hacia (salida) dispositivos externos.
- Tres formas de realizar la transferencia:
 - Sondeo (encuesta, *polling*)
 - La CPU es responsable de enviar y recibir datos, y de la sincronización con el periférico (esperar la llegada de datos o esperar la petición de envío de datos).
 - Interrupción
 - La CPU es responsable de enviar y recibir datos.
 - La sincronización se realiza mediante interrupciones *hardware* que recibe la CPU.
 - DMA (*Direct Memory Access*)
 - La CPU configura un controlador de DMA que se encarga de enviar y recibir datos.
 - La sincronización la realiza el propio controlador de DMA mediante interrupciones *hardware*.
 - Utilizado para transferencia de bloques.

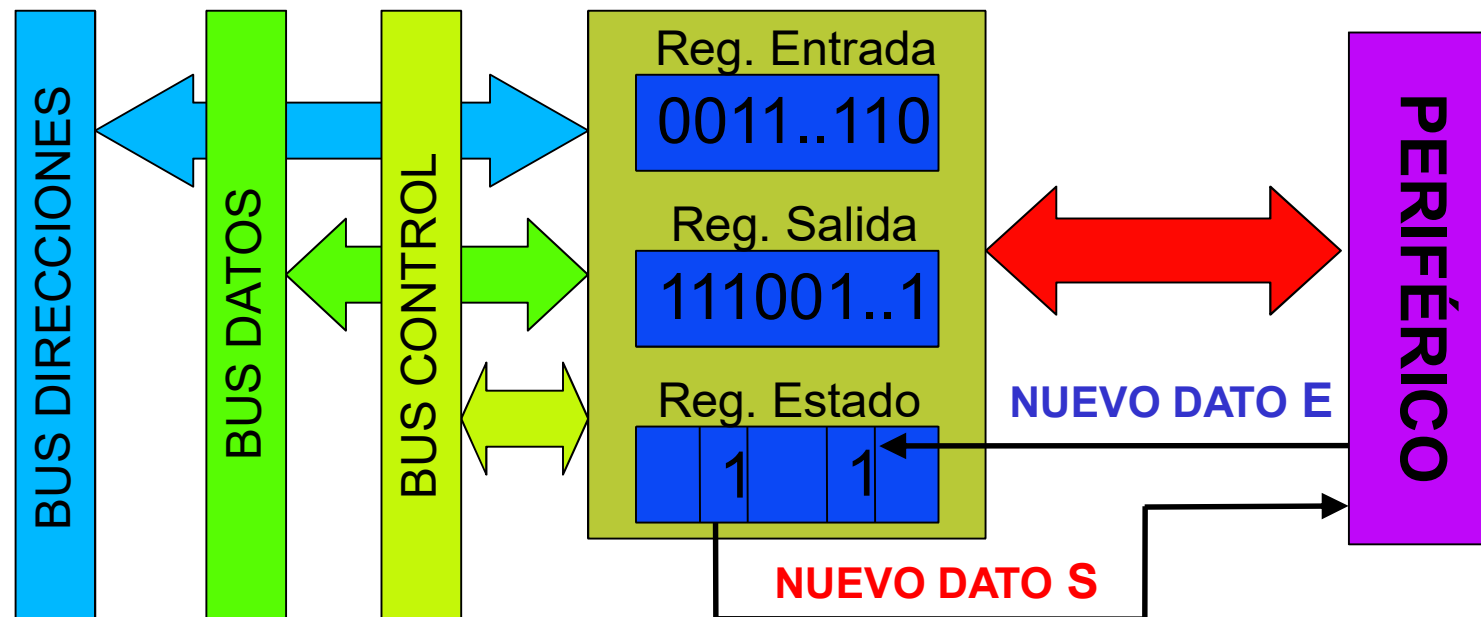
5.1. Técnicas de programación de E/S (II)

- Cada periférico es accedido por la CPU a través de un puerto de E/S (*controlador*) que actúa de interfaz.
- La CPU suele acceder a los puertos leyendo y escribiendo registros internos (cada registro tiene una dirección de E/S):
 - **Registros de entrada:** Datos del periférico hacia la CPU.
 - **Registros de salida:** Datos de la CPU hacia el periférico.
 - **Registros de estado:** Estado actual del periférico.



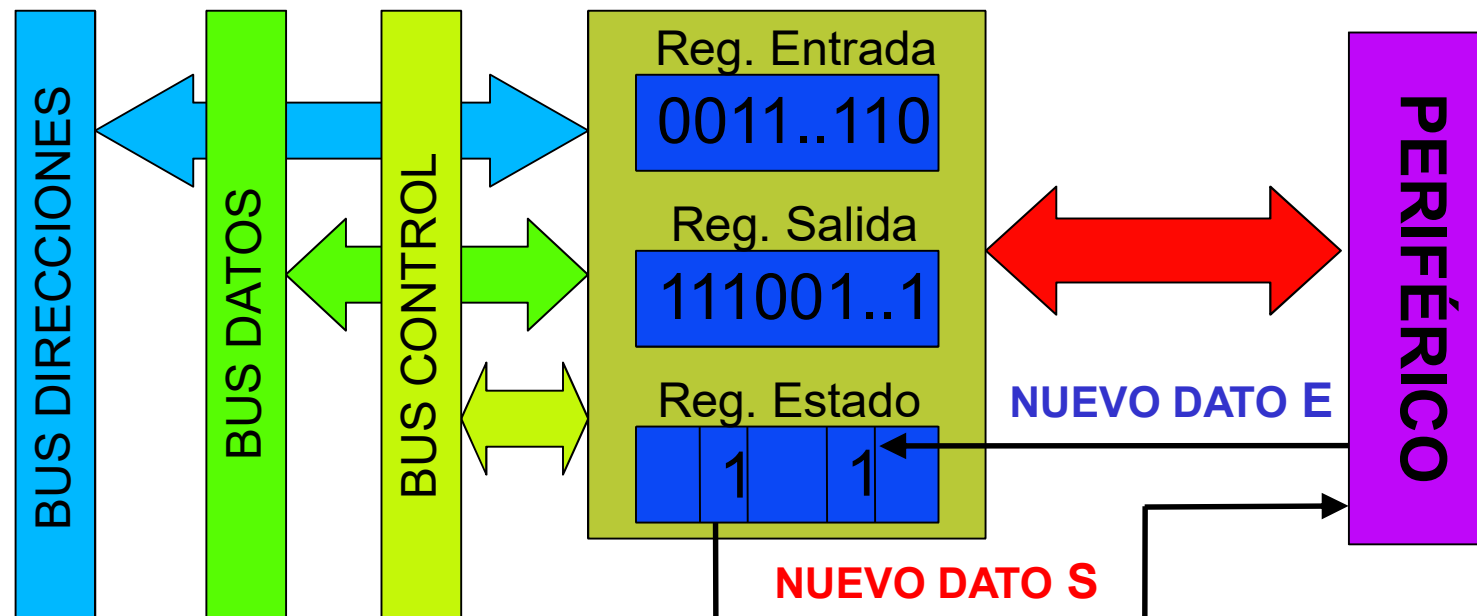
5.2. Sondeo

- La CPU ha de enviar y recibir datos, y sincronizarse con el periférico.
- Sincronización mediante espera activa: un bucle consulta continuamente el registro de estado (muy ineficiente).
- Protocolo de “apretón de manos” (*handshaking*) mediante dos líneas de control: **NUEVO DATO E** y **NUEVO DATO S**.



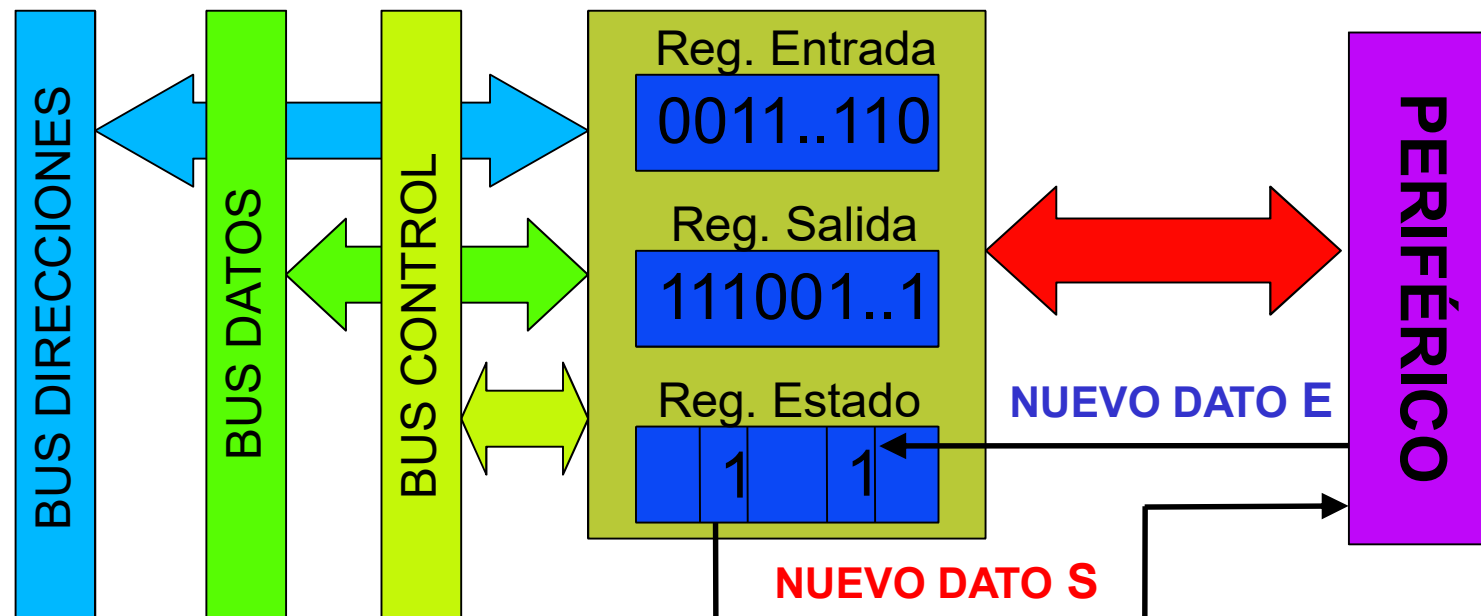
5.2. Sondeo

- Protocolo básico de escritura de datos hacia el periférico:
 - CPU escribe dato en registro de salida.
 - CPU activa señal **NUEVO DATO S** en registro de estado.
 - CPU espera activación de señal **NUEVO DATO E** en registro de estado (espera activa).
 - Periférico recibe dato del puerto y activa señal **NUEVO DATO E** (*Acknowledge, ACK*).



5.2. Sondeo

- Protocolo básico de lectura de datos desde el periférico:
 - CPU espera activación de señal **NUEVO DATO E** en registro de estado (espera activa).
 - Periférico envía dato al puerto y activa señal **NUEVO DATO E**.
 - CPU lee dato desde registro de entrada.
 - CPU activa señal **NUEVO DATO S** en registro de estado.



5.2. Sondeo

- **Ejemplo:** Lectura de datos desde puerto de E/S, almacenándolos en buffer de memoria y con envío de byte de control.
- Puerto de E/S:
 - 52h \Rightarrow @Reg.Entrada
 - 53h \Rightarrow @Reg.Salida
 - 54h \Rightarrow @Reg.Estado

```
datos SEGMENT
      buffer 200 DUP (0)
datos ENDS

codigo SEGMENT
      .....
      mov ax, datos
      mov ds, ax
      mov si, 0

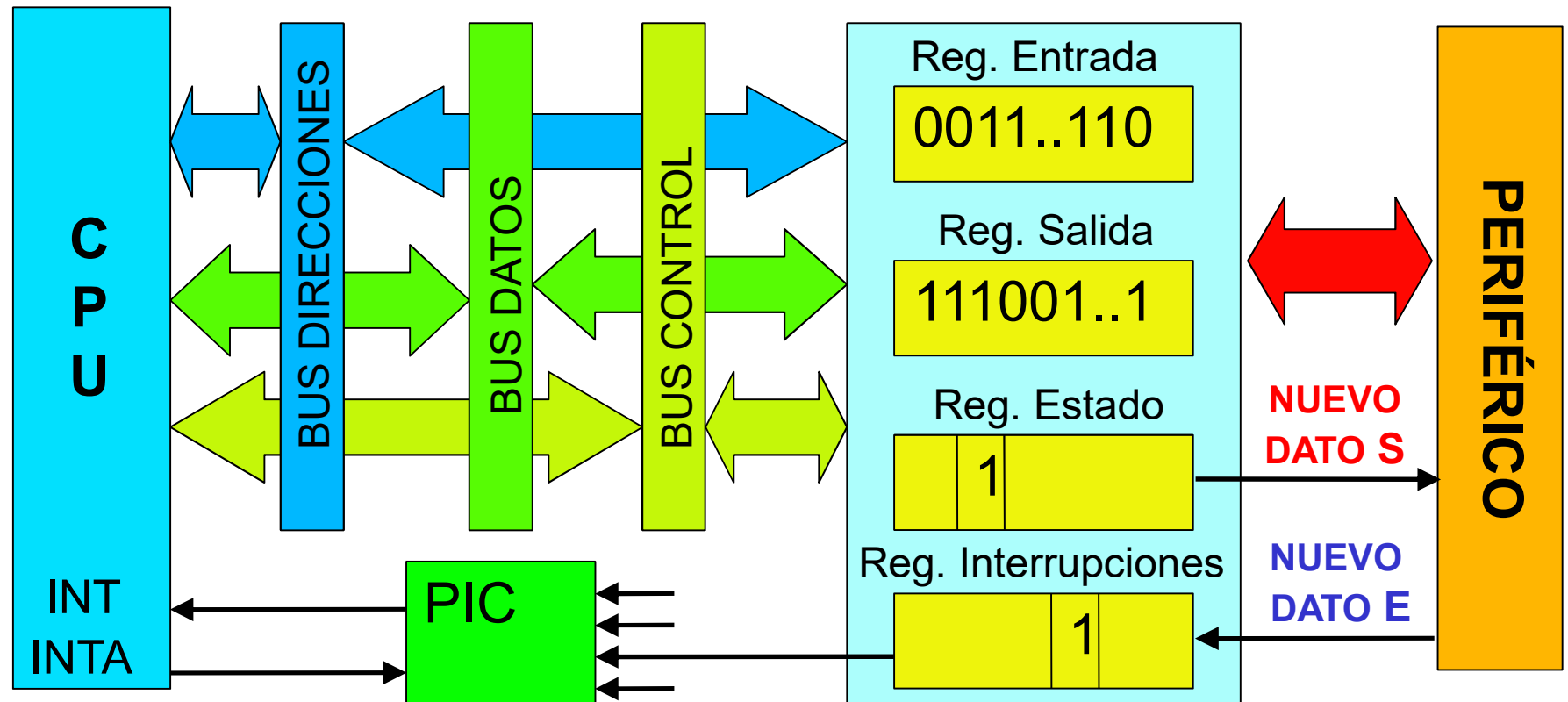
esperar: in al, 54h ; Espera activa
         test al, 00000001b
         jz esperar

         in al, 52h ; Lectura de dato
         mov buffer[ si ], al
         inc si
         cmp si, 200
         jne esperar

         mov al, 0FFh
         out 53h, al ; Byte de control
```

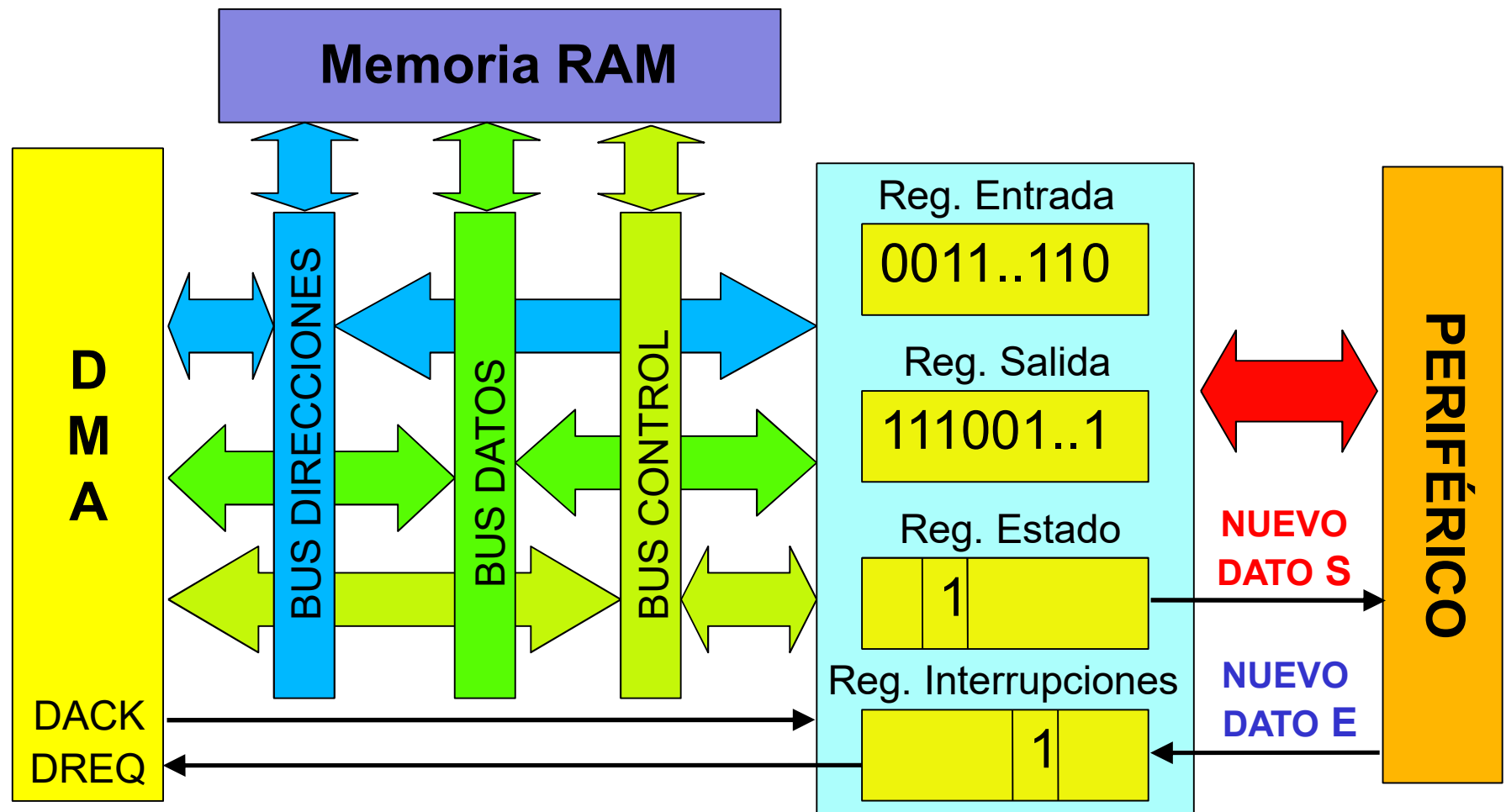

5.3. Interrupción

- La CPU es interrumpida cuando periférico manda dato (entrada) o cuando manda petición de recepción de dato (salida).
- La CPU ejecuta una rutina de servicio que lee dato o envía dato a través de los registros del puerto.



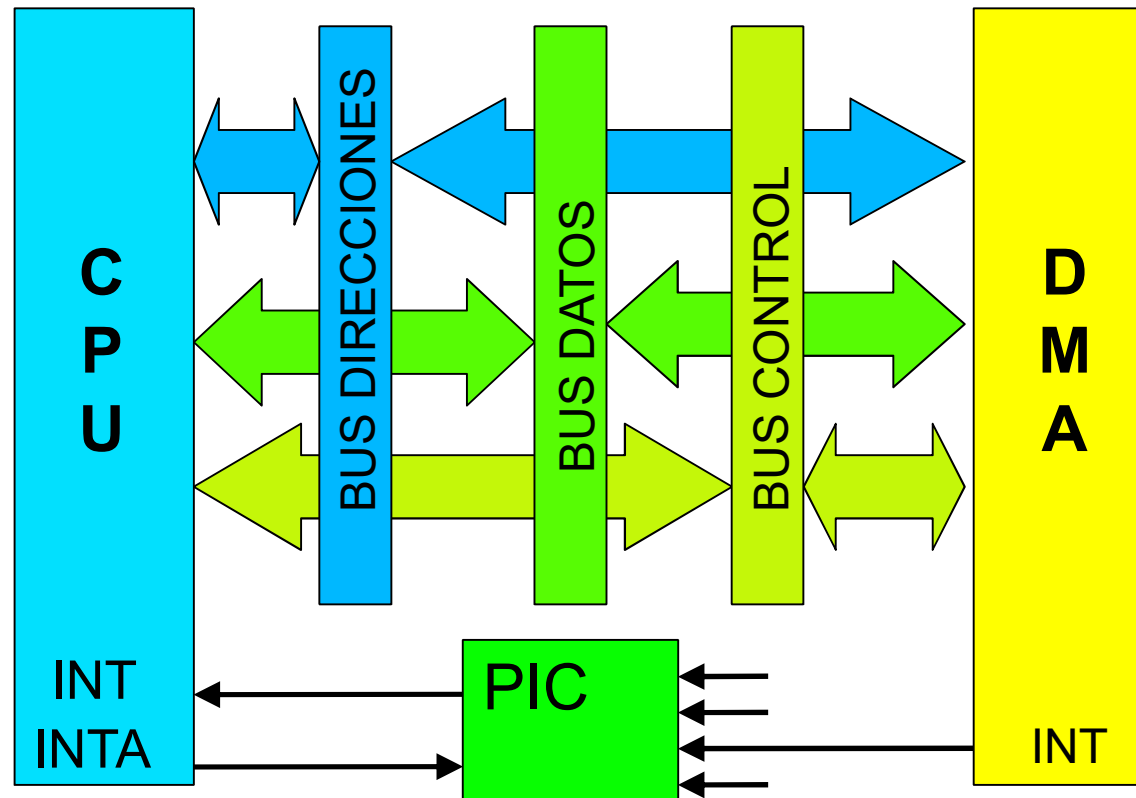
5.4. DMA

- La CPU programa el controlador de DMA para que transfiera un bloque de datos desde memoria al puerto de E/S (salida) o desde el puerto a memoria (entrada).



5.4. DMA

- El controlador de DMA interrumpe a la CPU cuando se ha transferido un bloque completo.
- La CPU ejecuta una rutina de servicio que accede al bloque (entrada) o genera un nuevo bloque (salida) y reprograma el controlador de DMA.

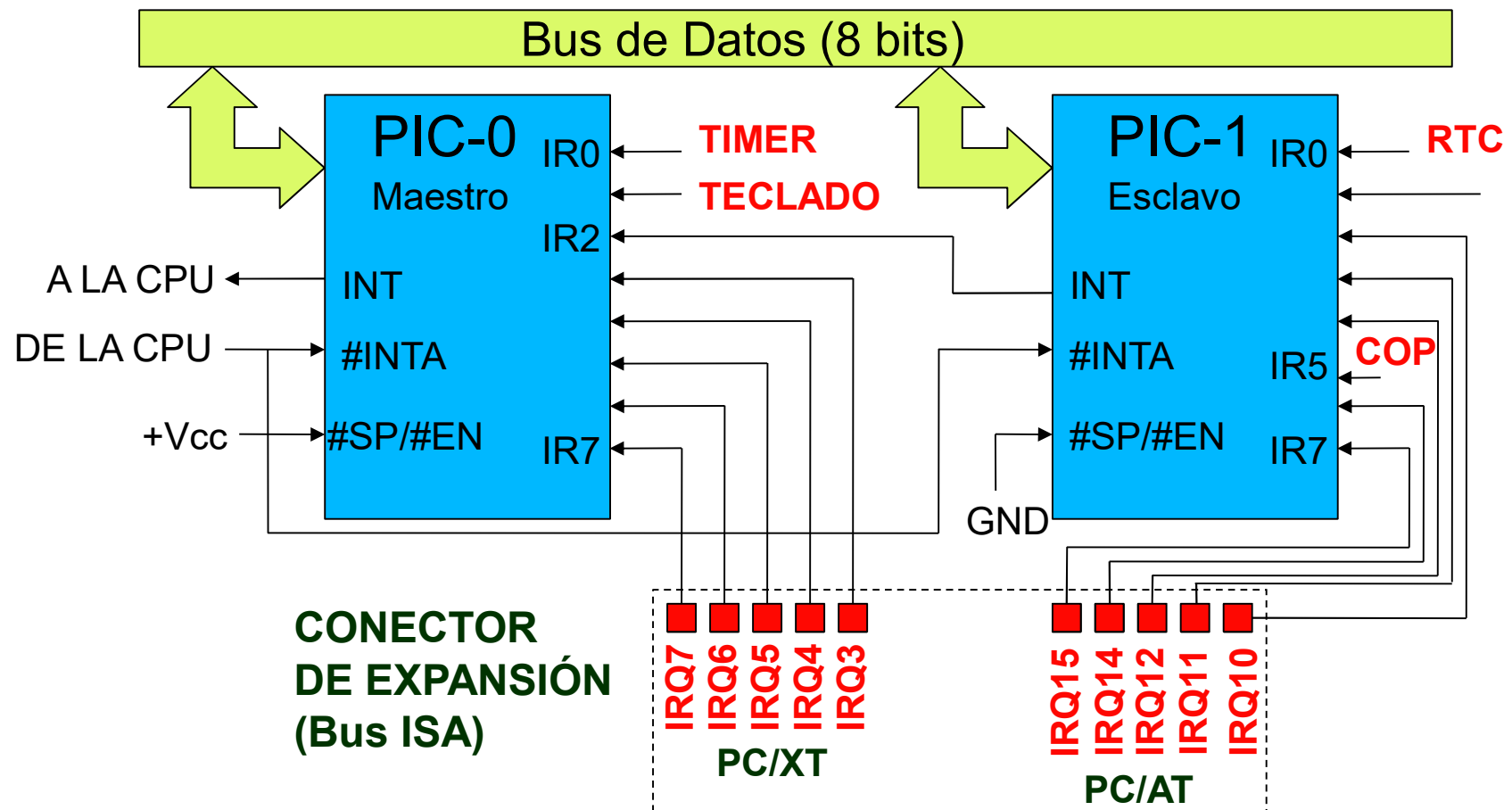


5.5. Gestión y programación de las interrupciones en el 80x86

- Las interrupciones *hardware* enmascarables son gestionadas mediante el controlador de interrupciones programable (PIC) 8259.
- Un PIC tiene 8 entradas de interrupción y una salida.
- Pueden gestionarse múltiples interrupciones enmascarables dependiendo del número de 8259 instalados (1 en PC/XT, 2 en PC/AT y superiores).
- La CPU recibe una única petición de interrupción por parte del PIC principal (maestro).
- Cada interrupción puede ser enmascarada de forma independiente a través del 8259.
- Se pueden establecer distintos esquemas de prioridad sobre las interrupciones enmascarables.

5.5. Gestión y programación de las interrupciones en el 80x86 (II)

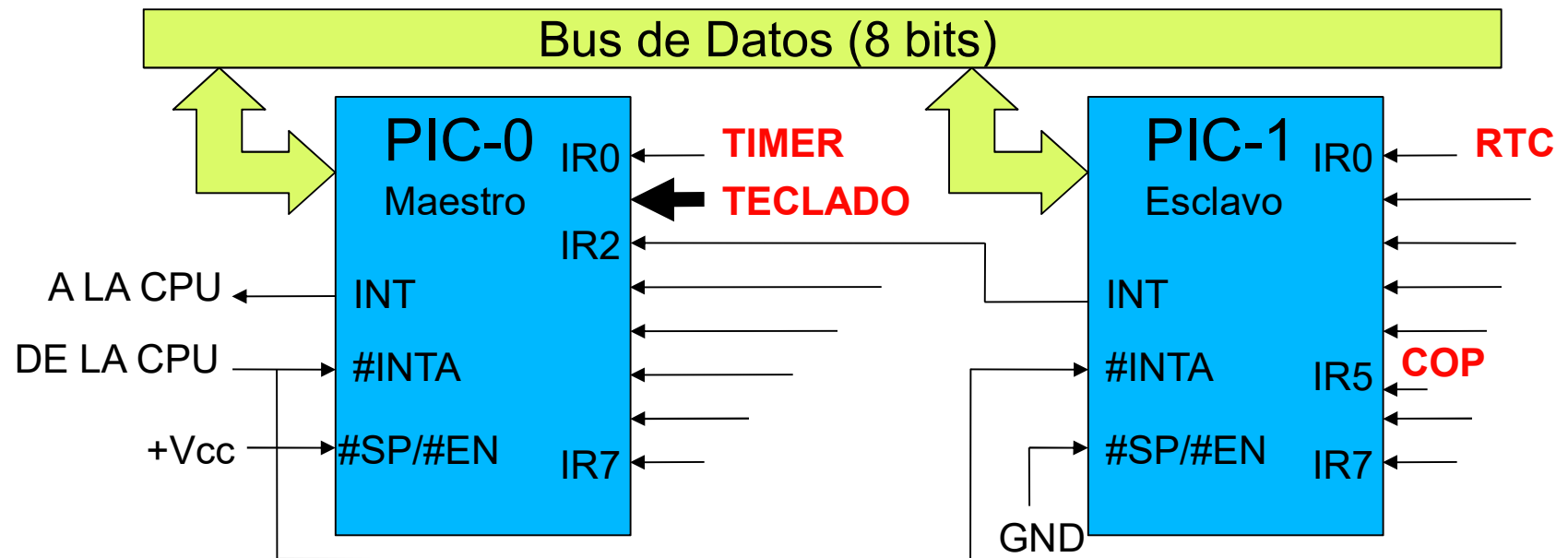
- Direcciones PIC-0 : 20h, 21h (PC/XT, PC/AT y posteriores)
- Direcciones PIC-1 : A0h, A1h (PC/AT y posteriores)
- Interrupciones: PIC-0 \Rightarrow 08h (IR0) , ... , 0Fh (IR7)
PIC-1 \Rightarrow 70h (IR0) , ... , 7Fh (IR7)



5.5. Gestión y programación de las interrupciones en el 80x86 (III)

● Petición de interrupción a través del PIC maestro:

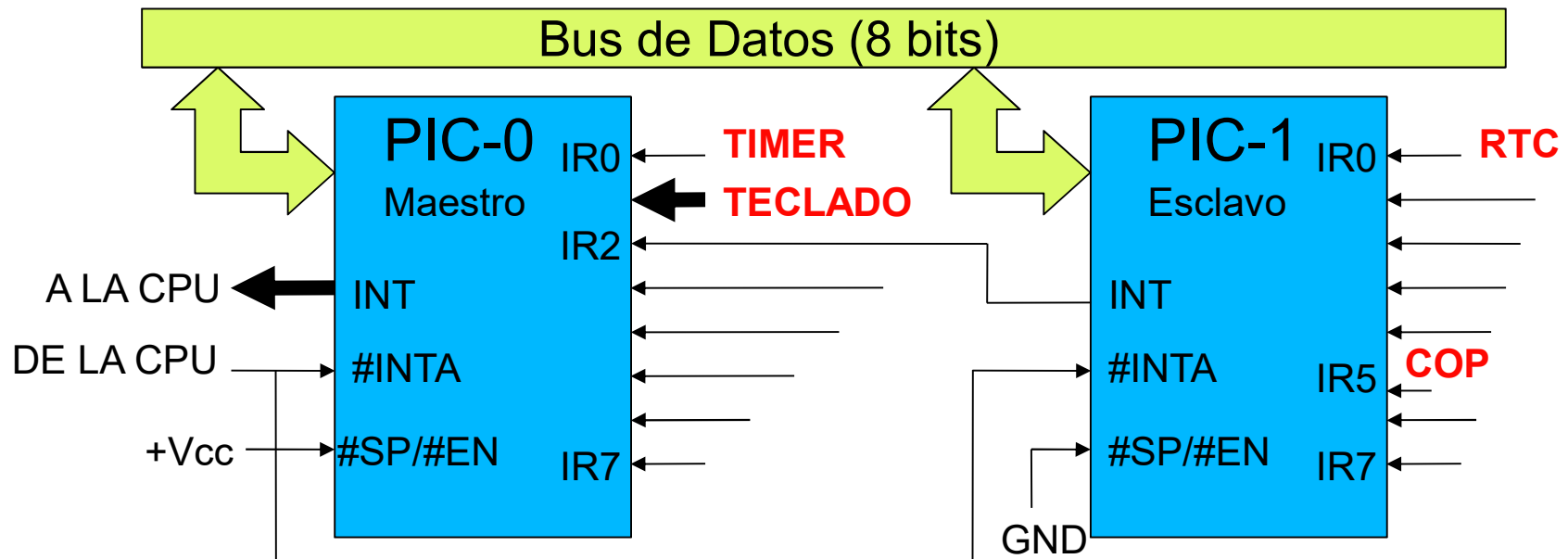
1. Puerto de E/S activa petición de interrupción de maestro.



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC maestro:

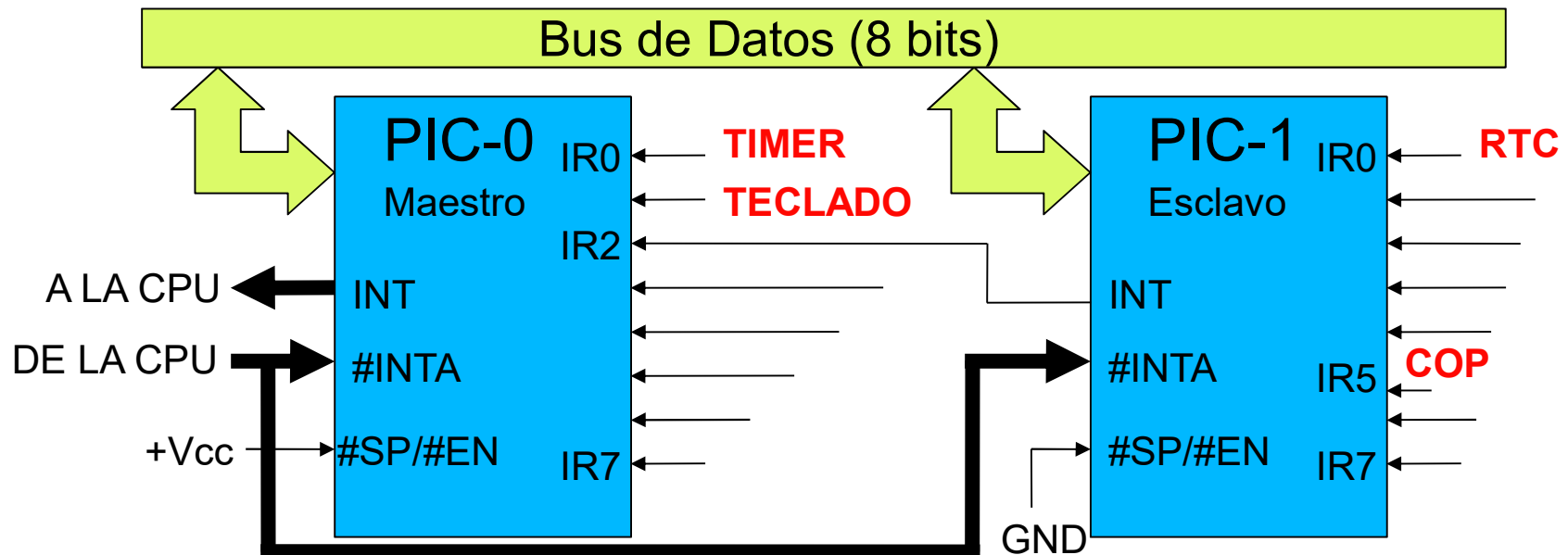
1. Puerto de E/S activa petición de interrupción de maestro.
2. Si interrupción tiene prioridad suficiente, PIC maestro activa petición de interrupción de CPU (**señal INT**).



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC maestro:

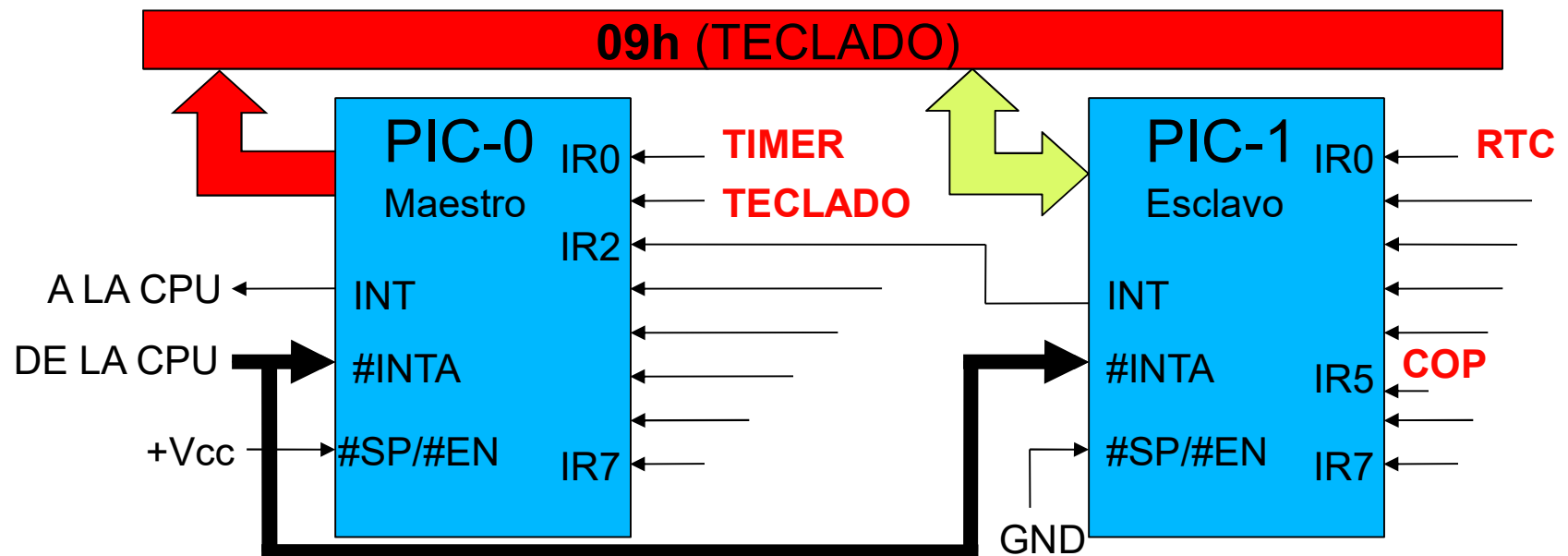
1. Puerto de E/S activa petición de interrupción de maestro.
2. Si interrupción tiene prioridad suficiente, PIC maestro activa petición de interrupción de CPU (**señal INT**).
3. Si CPU acepta interrupción, activa señal de aceptación (**señal #INTA**) dos veces seguidas.



5.5. Gestión y programación de las interrupciones en el 80x86

- Petición de interrupción a través del PIC maestro:

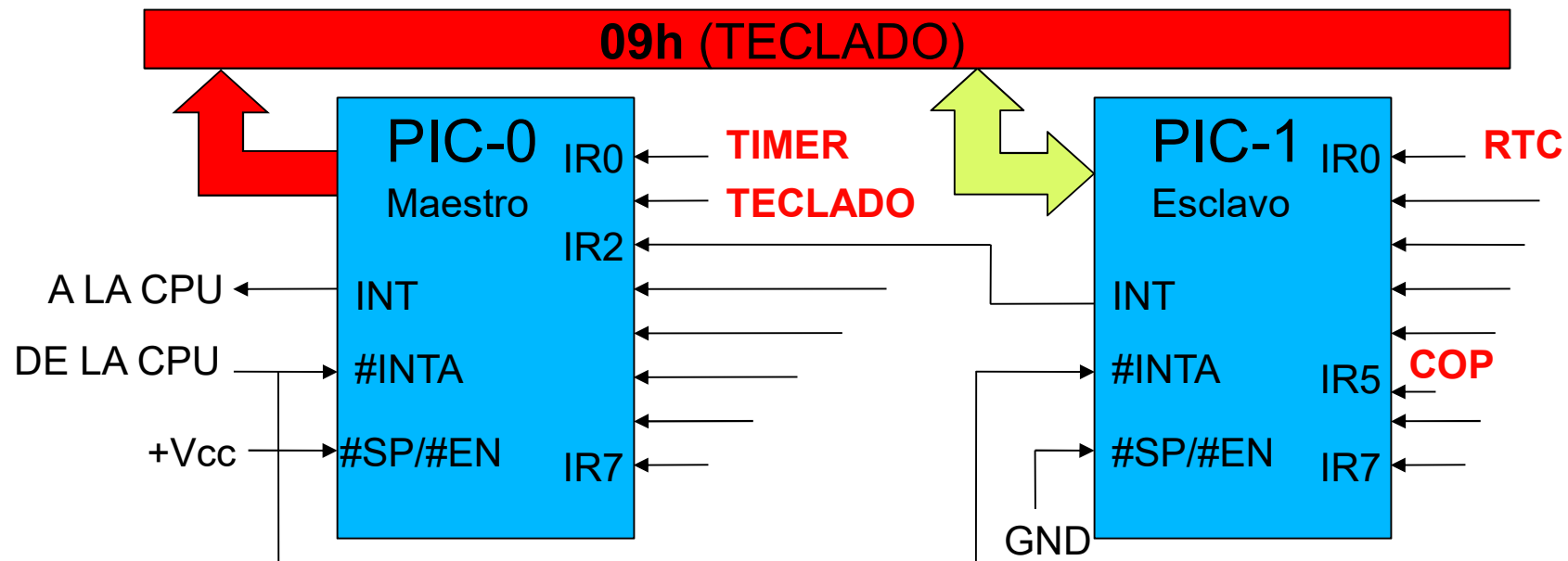
- En la segunda aceptación, PIC maestro escribe número de interrupción en bus de datos.



5.5. Gestión y programación de las interrupciones en el 80x86 (VII)

- **Petición de interrupción a través del PIC maestro:**

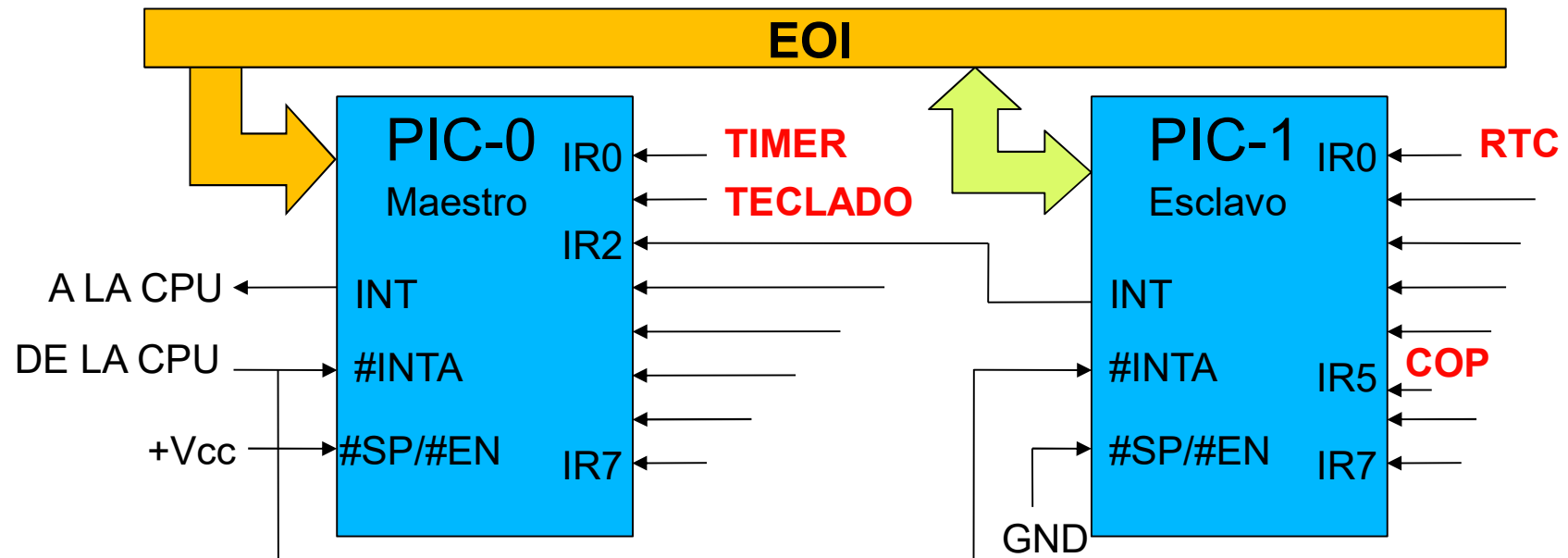
4. En la segunda aceptación, PIC maestro escribe número de interrupción en bus de datos.
5. CPU obtiene vector de interrupción y ejecuta rutina de servicio (**RSI**).



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC maestro:

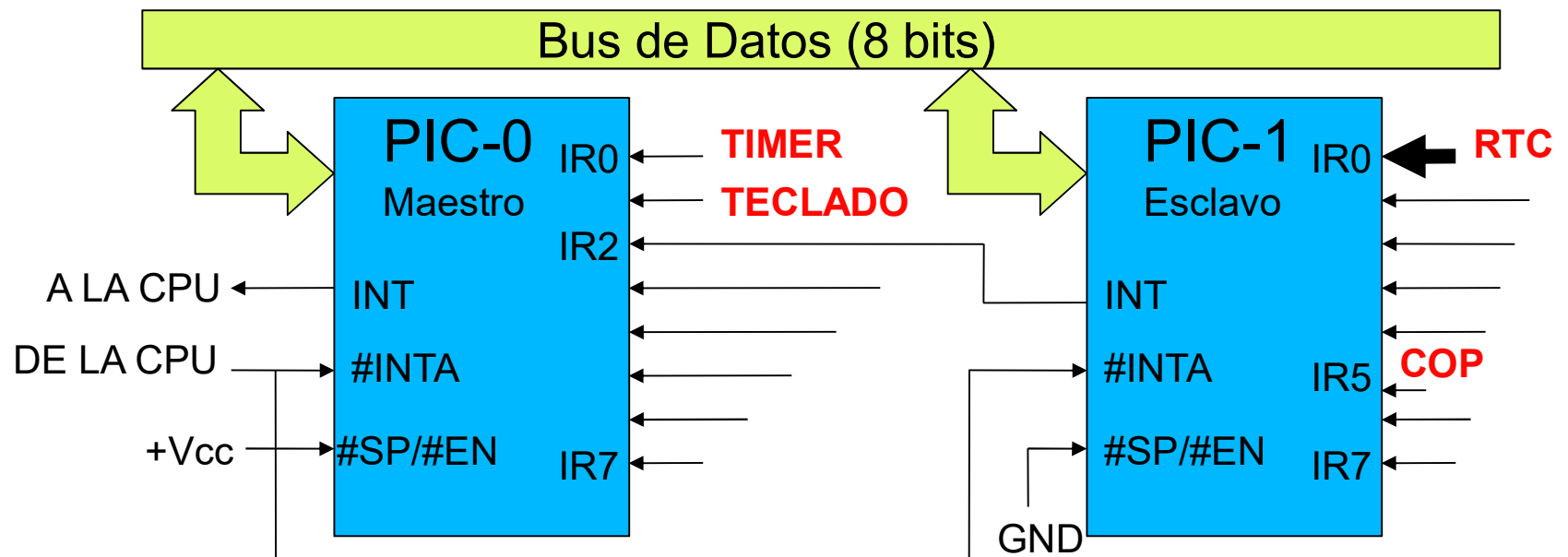
4. En la segunda aceptación, PIC maestro escribe número de interrupción en bus de datos.
5. CPU obtiene vector de interrupción y ejecuta rutina de servicio (**RSI**).
6. Antes de acabar, RSI envía comando de fin de interrupción (**End Of Interrupt, EOI**) a PIC maestro.
7. PIC maestro da por concluida la petición.



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC esclavo:

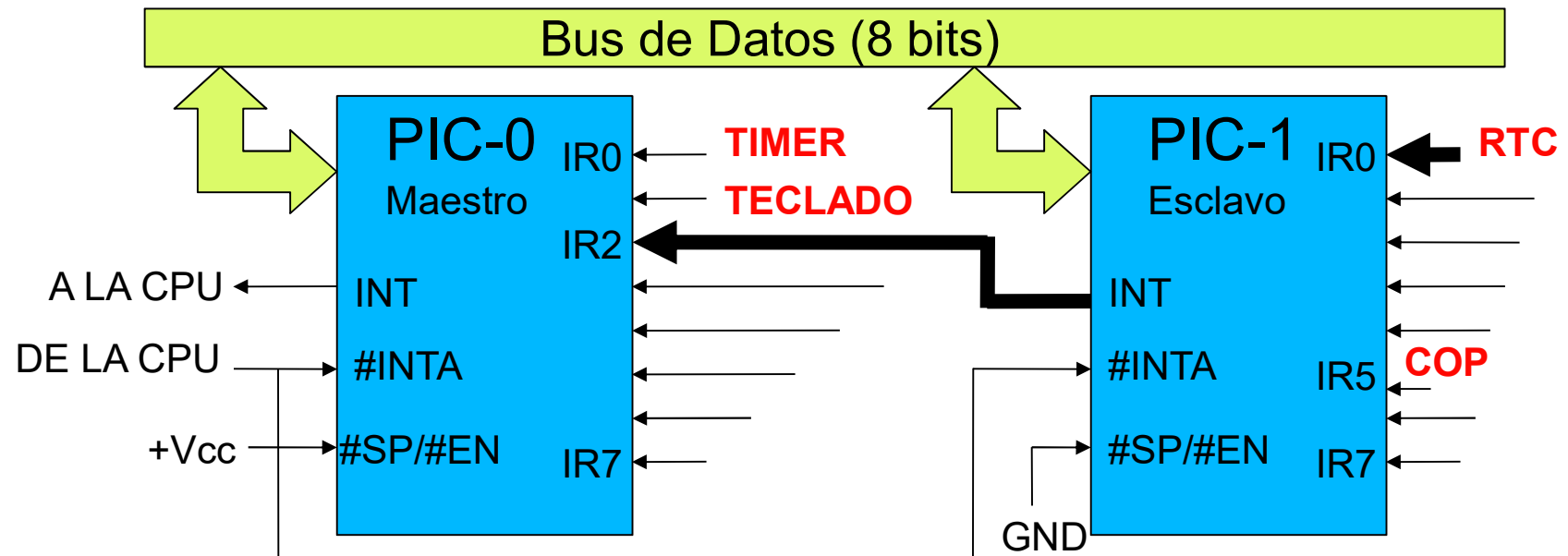
1. Puerto de E/S activa petición de interrupción de esclavo.



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC esclavo:

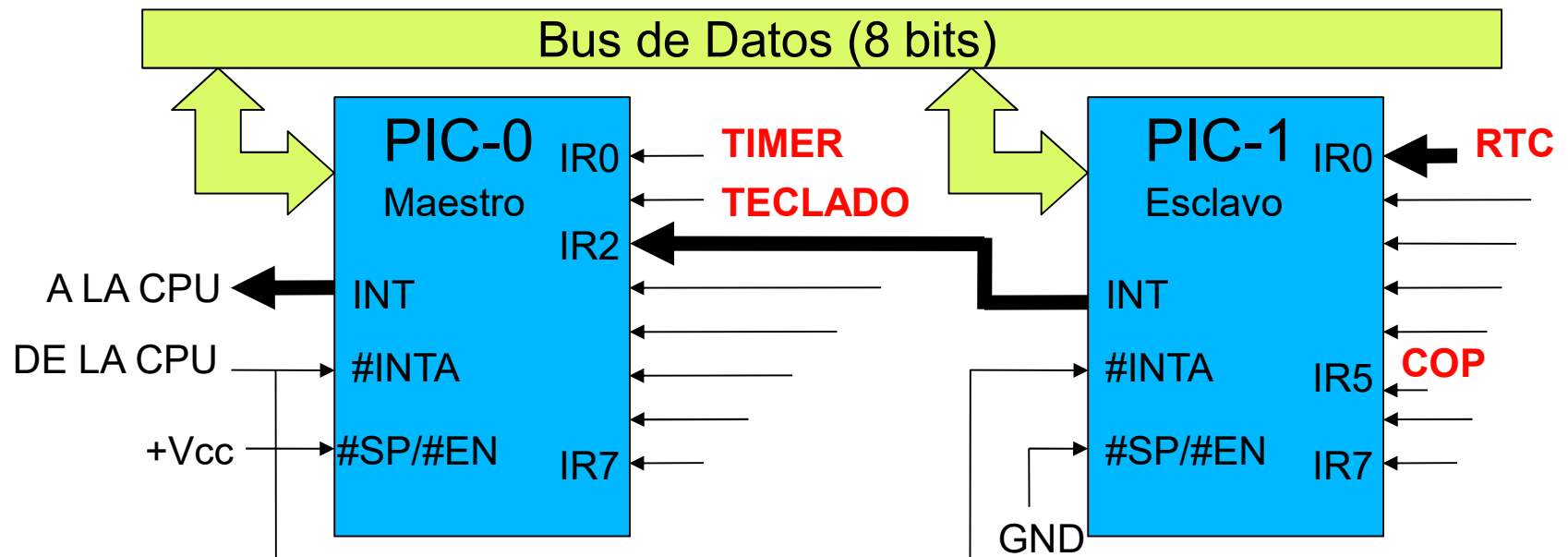
1. Puerto de E/S activa petición de interrupción de esclavo.
2. Si interrupción tiene prioridad suficiente, PIC esclavo activa petición de interrupción de PIC maestro (IR2).



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC esclavo:

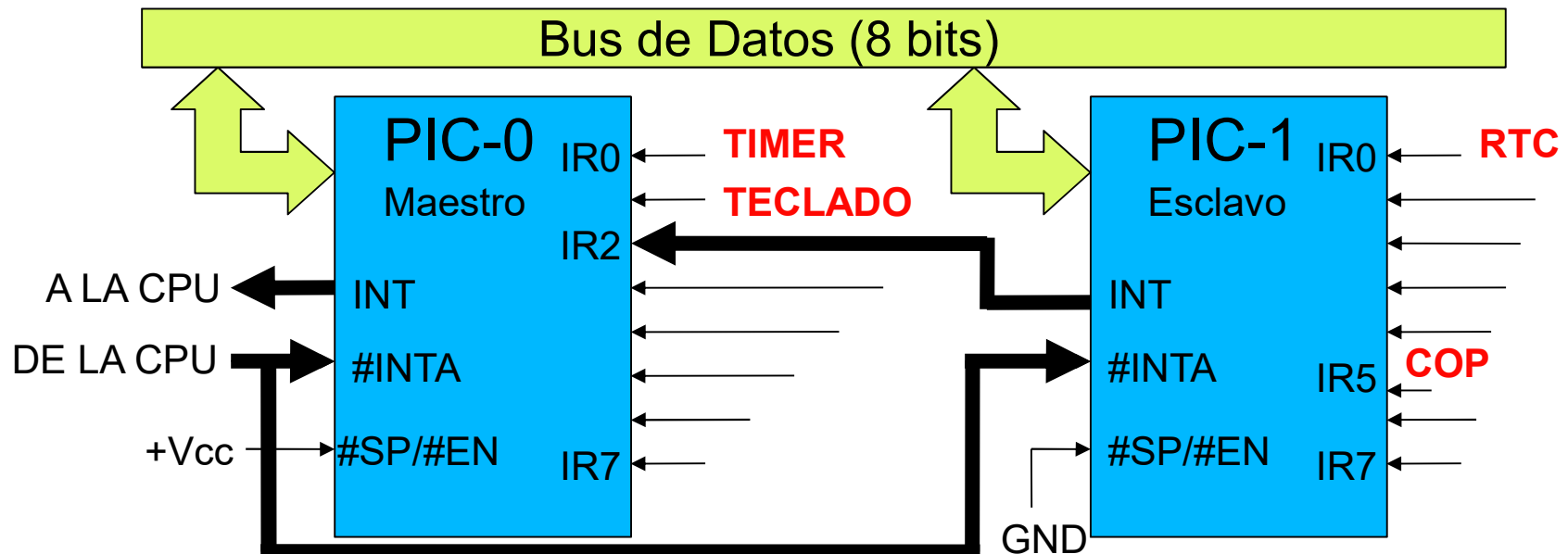
1. Puerto de E/S activa petición de interrupción de esclavo.
2. Si interrupción tiene prioridad suficiente, PIC esclavo activa petición de interrupción de PIC maestro (**IR2**).
3. Si interrupción 2 tiene prioridad suficiente, PIC maestro activa interrupción de CPU (**INT**).



5.5. Gestión y programación de las interrupciones en el 80x86

● Petición de interrupción a través del PIC esclavo:

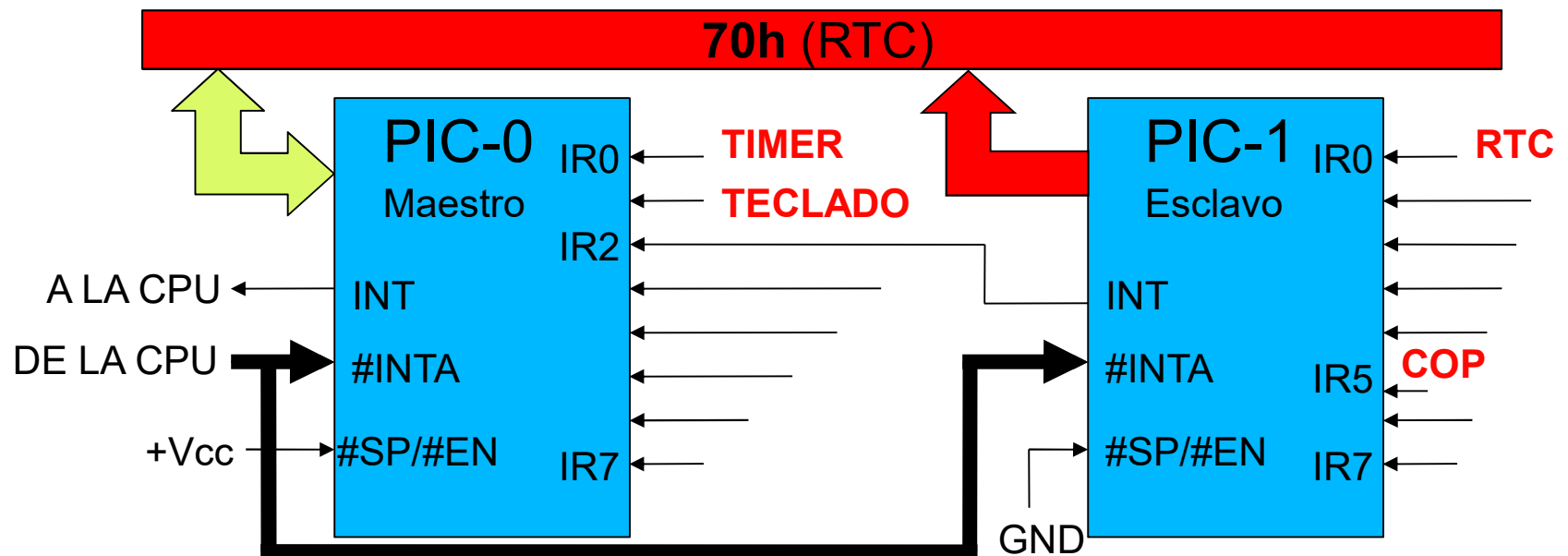
1. Puerto de E/S activa petición de interrupción de esclavo.
2. Si interrupción tiene prioridad suficiente, PIC esclavo activa petición de interrupción de PIC maestro (IR2).
3. Si interrupción 2 tiene prioridad suficiente, PIC maestro activa interrupción de CPU (INT).
4. CPU envía dos aceptaciones seguidas (#INTA).



5.5. Gestión y programación de las interrupciones en el 80x86

- Petición de interrupción a través del PIC esclavo:

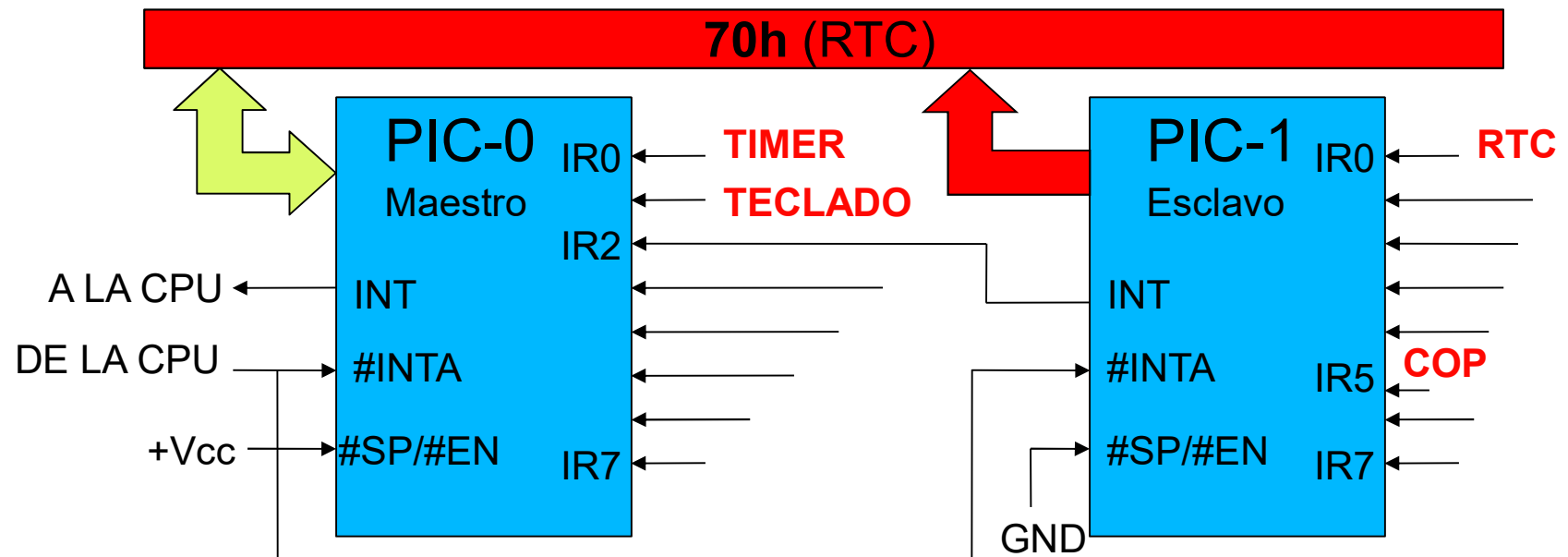
- En la segunda aceptación, PIC esclavo escribe número de interrupción en bus de datos.



5.5. Gestión y programación de las interrupciones en el 80x86 (XIV)

- **Petición de interrupción a través del PIC esclavo:**

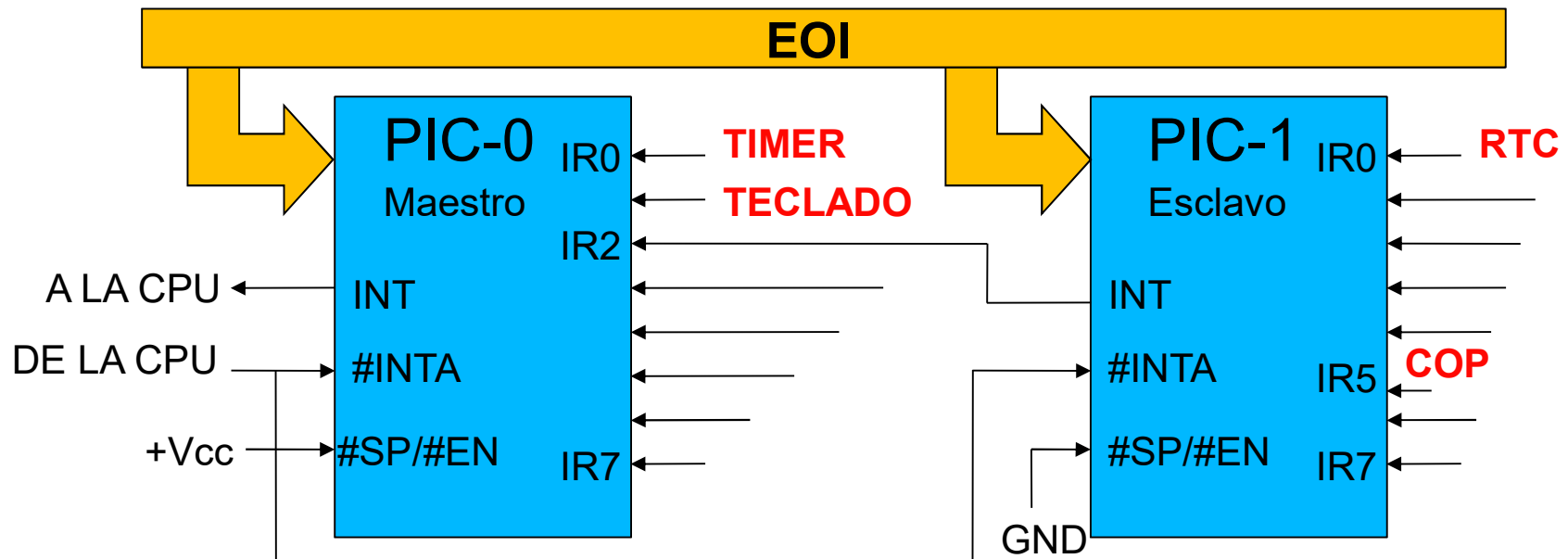
5. En la segunda aceptación, PIC esclavo escribe número de interrupción en bus de datos.
6. CPU obtiene vector de interrupción y ejecuta rutina de servicio **(RSI)**.



5.5. Gestión y programación de las interrupciones en el 80x86 (XV)

● Petición de interrupción a través del PIC esclavo:

5. En la segunda aceptación, PIC esclavo escribe número de interrupción en bus de datos.
6. CPU obtiene vector de interrupción y ejecuta rutina de servicio **(RSI)**.
7. Antes de acabar, RSI envía comando EOI a PICs maestro y esclavo.
8. PICs maestro y esclavo dan por concluida la petición.



5.5. Gestión y programación de las interrupciones en el 80x86

- Estructura de la rutina de servicio de una interrupción enmascarable:

RSI:

```
sti      ; Permite que la RSI pueda ser interrumpida  
GUARDAR REGISTROS EN LA PILA  
.....  
.....  
.....  
RECUPERAR REGISTROS DE LA PILA  
ENVIAR EOI(s) (*)  
iret
```

(*) Un **EOI** al MAESTRO si la interrupción procede del MAESTRO.
Un **EOI** al MAESTRO y un **EOI** al ESCLAVO si la interrupción
procede del ESCLAVO.

5.5. Gestión y programación de las interrupciones en el 80x86

	IR	IRQ	Número de interrupción	PC/XT	PC/AT
PIC-0	IR0	IRQ0	08h	Timer 18.2 i/s	Timer 18.2 i/s
	IR1	IRQ1	09h	Teclado	Teclado
	IR2	IRQ2	0Ah		PIC-1
	IR3	IRQ3	0Bh	COM2	COM2
	IR4	IRQ4	0Ch	COM1	COM1
	IR5	IRQ5	0Dh	Disco duro	LPT2
	IR6	IRQ6	0Eh	Floppy	Floppy
	IR7	IRQ7	0Fh	LPT	LPT1
PIC-1	IR0	IRQ8	70h		RTC
	IR1	IRQ9	71h		
	IR2	IRQ10	72h		
	IR3	IRQ11	73h		
	IR4	IRQ12	74h		
	IR5	IRQ13	75h		Coprocesador
	IR6	IRQ14	76h		Disco duro
	IR7	IRQ15	77h		

5.5. Gestión y programación de las interrupciones en el 80x86

- El 8259 tiene tres registros internos de 8 bits que permiten a la CPU controlar su funcionamiento y conocer su estado.
- Cada bit asociado a una entrada de interrupción (IR0 a IR7).
- **IMR** (Registro de máscara de interrupciones)
 - Un bit a uno inhibe las peticiones de su entrada de interrupción asociada.
 - Se puede leer y modificar en cualquier momento.
- **IRR** (Registro de petición de interrupción)
 - Un bit a uno indica una petición de interrupción aún no aceptada por la CPU.
 - Sólo se puede leer.
- **ISR** (Registro de interrupciones en servicio)
 - Un bit a uno indica petición de interrupción ya aceptada y que aún no ha finalizado (la CPU está ejecutando su RSI).
 - El bit se pone a cero cuando su RSI envía el comando EOI.
 - Sólo se puede leer.

5.5. Gestión y programación de las interrupciones en el 80x86

- El 8259 (PIC) dispone de dos familias de “comandos” para poder ser configurado por la CPU.
- Comandos de inicialización (*Initialization Command Word*)
 - 4 comandos consecutivos enviados de la CPU al PIC.
 - Se ejecutan durante la fase de arranque del PC.
 - Configuran el funcionamiento inicial del PIC.
- Comandos de operación (*Operation Command Word*)
 - 3 comandos que pueden enviarse desde la CPU en cualquier momento durante la ejecución de un programa.
 - Definen el funcionamiento de ciertos aspectos del PIC y permiten consultar su estado interno.
 - Han de enviarse con las interrupciones enmascarables inhibidas (**IF** = 0).
- La CPU inicia la ejecución de un comando enviando un byte al PIC mediante la instrucción **out**.

5.5. Gestión y programación de las interrupciones en el 80x86

- El tipo de comando depende de la dirección especificada y de algunos bits del byte enviado.
- Direcciones: **PIC-0** (20h y 21h) , **PIC-1** (A0h y A1h)

Nombre de comando	Dirección	Se distingue por
ICW1	A0 = 0 (par)	D4 a 1
ICW2	A0 = 1 (impar)	Sigue a ICW1
ICW3	A0 = 1 (impar)	Sigue a ICW2
ICW4	A0 = 1 (impar)	Sigue a ICW3
OCW1	A0 = 1 (impar)	
OCW2	A0 = 0 (par)	D3 a 0 y D4 a 0
OCW3	A0 = 0 (par)	D3 a 1 y D4 a 0

A0 (bus de direcciones) , **D3 y D4** (bus de datos)

(5)

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

- Dirección par (20h o A0h) y $D4 = D3 = 0$.
- Utilizado para:
 - Envío de **EOIs** (Fin de rutina de servicio de interrupción):
 - No específicos.
 - Específicos.
 - Cambio de prioridad de interrupciones (**rotaciones**)
 - Específicas.
 - Automáticas.
 - Con un único comando OCW2 se puede:
 - Enviar un EOI.
 - Realizar una rotación de prioridades.
 - Enviar un EOI y realizar una rotación de prioridades.

(5)

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

- **EOI No Específico**

- No especifica qué interrupción acaba.
- El PIC pone a 0 el bit del **ISR** de la IR con más prioridad.
- EOI habitual.

- **EOI Específico**

- Especifica bit del **ISR** que ha de ponerse a 0.
- Necesario cuando la interrupción que acaba no es la de mayor prioridad.

- **EOI Automático (debe activarse mediante ICW)**

- Rutina de servicio no ha de enviar EOI.
- El PIC pone el bit del **ISR** a 0 al recibir el segundo #INTA (antes de ejecutar la rutina de servicio).
- Una interrupción de cualquier prioridad puede interrumpir a la rutina de servicio que se esté ejecutando.

(5)

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

- Gestión de prioridades en **modo normal** (por defecto)
 - Cada IR tiene una prioridad fija.
 - Inicialmente \Rightarrow IR7 (mínima) $<$ IR6 $<$... $<$ IR0 (máxima)
 - Cuando se está atendiendo a una interrupción, el PIC no sirve las de menor o igual prioridad.
 - Se puede cambiar el orden de las prioridades mediante un comando de **rotación**.
 - La rotación ha de indicar la IR con mínima prioridad.
 - La prioridad de las demás IRs se reajusta cíclicamente.
 - Ejemplo: **Rotación de IR2**
 - IR2 (mín) $<$ IR1 $<$ IR0 $<$ IR7 $<$ IR6 $<$ IR5 $<$ IR4 $<$ IR3 (máx)
 - Rotación **específica** \Rightarrow Comando OCW2 indica la IR que pasa a tener mínima prioridad.
 - Rotación **automática** \Rightarrow La IR que finaliza (indicado por EOI) pasa a tener mínima prioridad (**round-robin**)

(5)

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

- Gestión de prioridades en **modo normal** (por defecto)
 - Problema de “**inversión de prioridades**”:
 - PIC-0 (maestro) **no sirve** (no envía a la CPU) **nuevas** interrupciones del PIC-1 (esclavo) hasta que no finalice la última que haya enviado ese esclavo (maestro reciba EOI).
 - Si esclavo envía interrupción de baja prioridad (ej. disco duro) y a continuación envía una de más prioridad (ej. RTC), maestro retiene la segunda hasta que finalice la primera \Rightarrow IR de más prioridad pasa a tener menos prioridad efectiva (**se invierten prioridades**).
 - **Solución:** Configurar PIC maestro en **modo especial** de gestión de prioridades.

(5)

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

● Gestión de prioridades en **modo especial**

- Se activa o desactiva mediante un comando OCW3.
- Sólo aplicable al PIC-0 (**maestro**).
- Maestro sirve cualquier petición nueva del esclavo aunque haya otras interrupciones del esclavo pendientes de finalización.
- Sólo se debe enviar **EOI** al maestro cuando no hay interrupciones en servicio en el esclavo (**ISR del esclavo valga 0**).

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

• D2,D1,D0

- IR que finaliza (si EOI específico) o con menor prioridad (si rotación).

• D4,D3

- 0,0 (fijos)

A0	D7	D6	D5	D4	D3	D2	D1	D0
0				0	0			

• D7,D6,D5

- 0,0,1 : **EOI no específico (20h)**
- 0,1,1 : EOI específico (**especificar D2-D0**)
- 1,0,1 : EOI no específico con rotación automática
- 1,0,0 : Activar rotación automática en EOI automático
- 0,0,0 : Desactivar rotación automática en EOI automático
- 1,1,1 : EOI específico con rotación automática (**especificar D2-D0**)
- 1,1,0 : Rotación específica (**especificar D2-D0**)
- 0,1,0 : no usado

5.5. Gestión y programación de las interrupciones en el 80x86

OCW2

• Ejemplos:

- **mov** al, 20h
out 20h, al ; EOI no específico a PIC-0 (maestro)
- **mov** al, 01100011b
out A0h, al ; EOI 3 a PIC-1 (esclavo)
- **mov** al, 10100000b
out A0h, al ; EOI no específico con rotación automática

	Antes	Después
ISR (PIC-1)	10010000	10000000
Prioridad	76543210	21076543

- **mov** al, 11100010b
out 20h, al ; EOI 2 con rotación automática

	Antes	Después
ISR (PIC-0)	10000100	10000000
Prioridad	76543210	43210765

5.5. Gestión y programación de las interrupciones en el 80x86

OCW3

- Dirección par (20h o A0h), D4 = 0 y D3 = 1.
 - Utilizado para lectura de registro de servicio (**ISR**) o de peticiones (**IRR**), activación/desactivación de “modo especial” y ejecución de comando POLL (**sondeo**).
- D1,D0: Registro leído en próxima lectura
 - 1,0 = **IRR**; 1,1 = **ISR**
- D2: Comando POLL
 - 0 = Inactivo, 1 = Activo
- D4,D3: 0,1 (fijos)
- D6,D5: Modo especial de gestión de prioridades
 - 1,0 = Desactivar; 1,1 = Activar; 0,0 = 0,1 = Ignorar.
- D7: 0 (fijo)

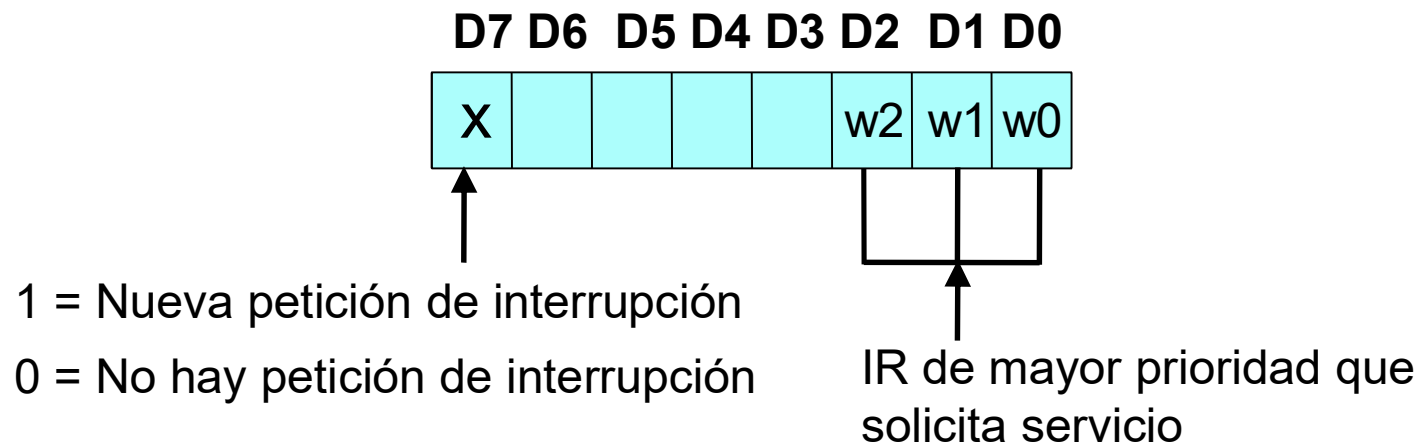
A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0			0	1			

5.5. Gestión y programación de las interrupciones en el 80x86

OCW3

● Comando POLL (sondeo).

- Permite a la CPU usar el PIC con las interrupciones enmascarables inhibidas (**IF** = 0).
- CPU envía comando POLL al PIC.
- CPU lee del PIC a continuación:
 - En caso de existir peticiones de interrupción en **IRR**, PIC pone a 1 el bit correspondiente del **ISR** según el esquema de prioridades (equivale a **#INTA**).
 - PIC escribe en el bus de datos el siguiente byte de estado:



5.5. Gestión y programación de las interrupciones en el 80x86

OCW3

• Ejemplos:

- **mov** al, 00001010b
out 20h, al ; Petición de lectura de **IRR** de PIC-0
in al, 20h ; Lectura de **IRR** de PIC-0
- **mov** al, 00001011b
out A0h, al ; Petición de lectura de **ISR** de PIC-1
in al, A0h ; Lectura de **ISR** de PIC-1
- **mov** al, 01101000b
out 20h, al ; Activa modo especial en PIC-0
- **mov** al, 01001000b
out 20h, al ; Pasa a modo normal en PIC-0
- **mov** al, 00001100b
out 20h, al ; Envía comando POLL al PIC-0
in al, 20h ; Recibe byte de estado del PIC-0