



# **Universidad Nacional de San Martín**

## **Sistemas de Procesamiento de Datos**

### UNIDAD 8 = Entrada/Salida, interrupciones, acceso directo a memoria

Tecnicatura en Programación Informática  
Tecnicatura en Redes Informáticas

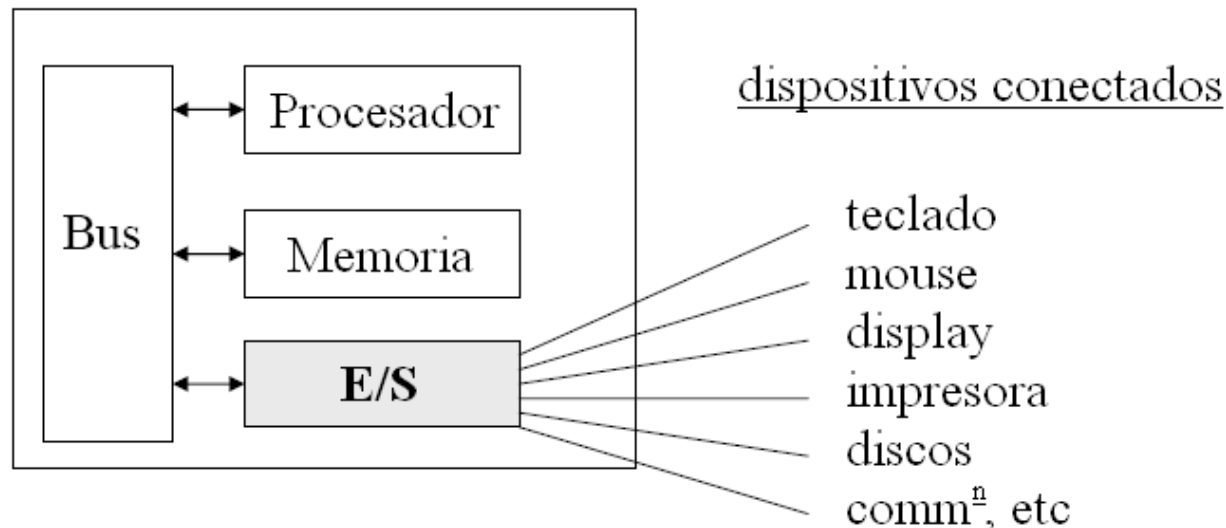
Profesor: Fabio Bruschetti

Ayudante: Pedro Iriso

2024 – 2C

# Unidades de Entrada/Salida (E/S) o Input/Output (I/O)

- Concepto básico de Entrada/Salida
  - Entrada/Salida es el intercambio de datos entre las buses internos del ordenador (conectados a la CPU y MP) y los dispositivos externos
- Diagrama de bloque del ordenador





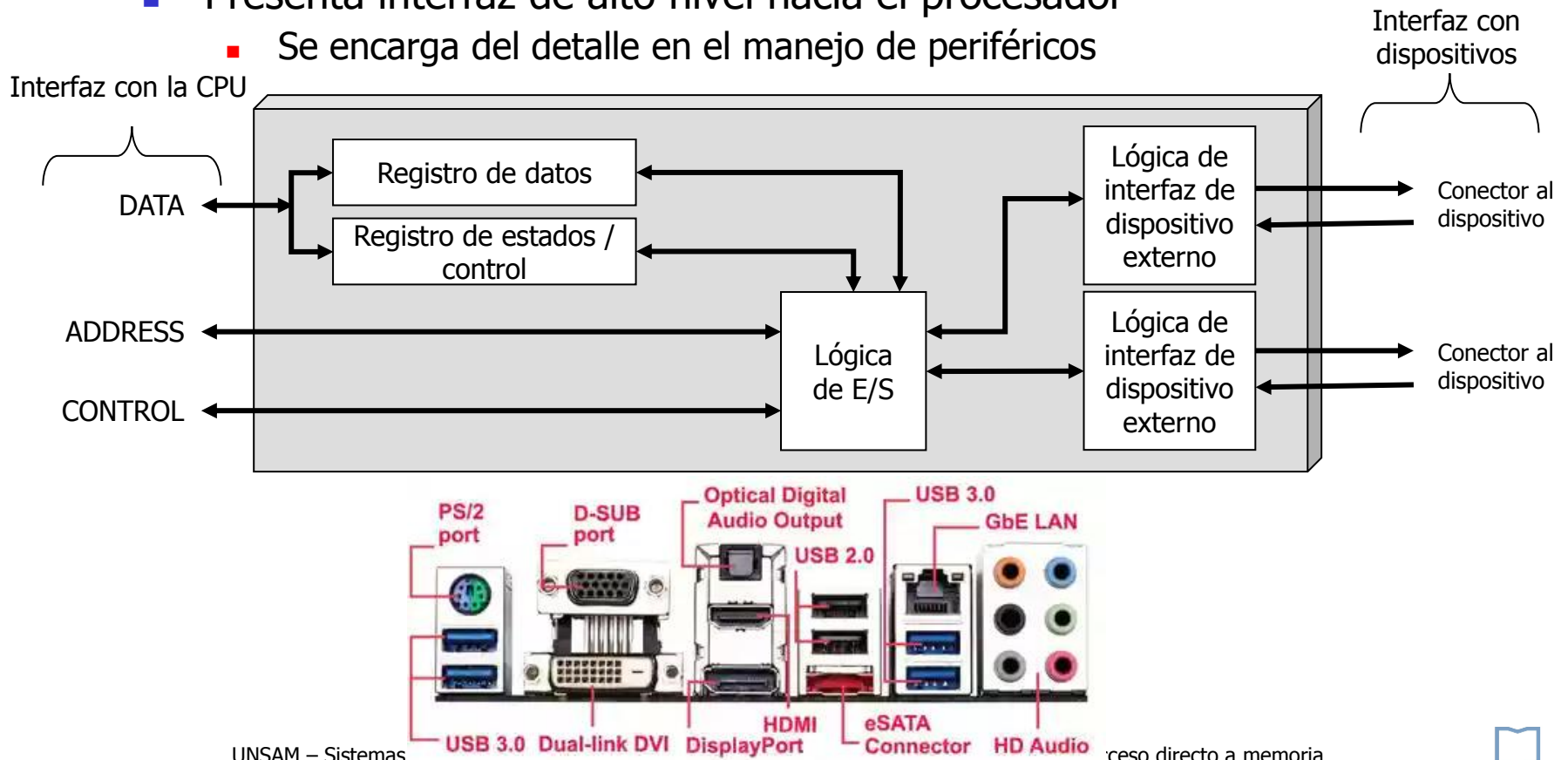
# Funciones

---

- Control y temporización
  - Coordinar el tráfico entre los recursos internos y dispositivos externos
- Comunicación con la CPU
  - Decodificar órdenes
  - Intercambiar datos
  - Proveer señalización de estados
  - Reconocimiento de direcciones
- Comunicación con los dispositivos
  - Ídem anterior excepto el reconocimiento de direcciones
- Almacenamiento temporal de datos
  - Adaptación de velocidades de transferencia de información
- Detección de errores
  - Informar errores al procesador
  - Paridad, Hamming

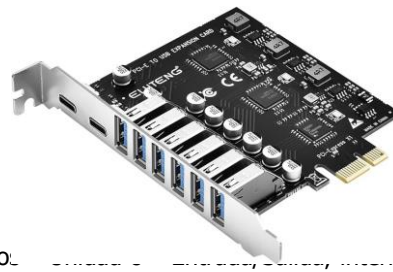
# Estructura

- Estructura de un módulo de E/S
  - Pueden tener uno o más registros para intercambio de datos
  - Cada registro tendrá una dirección única y unívoca
- Presenta interfaz de alto nivel hacia el procesador
  - Se encarga del detalle en el manejo de periféricos



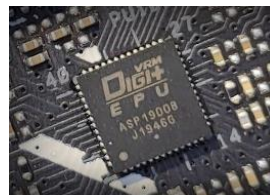
# Clasificación general de periféricos

- **Tarjetas de expansión**: Unidades independientes y conectables que se instalan en las ranuras de la placa base (por ejemplo, PCIe) para añadir o mejorar funcionalidades:
  - Placas gráficas avanzadas
  - Placas de sonido avanzadas
  - Placas de red
  - Placas de conectividad con almacenamiento
  - Placas de expansión USB
  - Placas WiFi



# Clasificación general de periféricos

- **Chips integrados o en la Placa Madre**: Circuitos integrados incorporados (soldados o insertados) que se encargan de funciones esenciales ya sea para la asistencia a la CPU como para agregar funcionalidades:
  - Gráficos integrados en la CPU
  - Sonido integrado
  - Red integrada
  - Almacenamiento integrado
  - BIOS/UEFI
  - Otros (PCH, PCI Express Lanes, USB, WiFi/Bluetooth, etc.)





# Interacción con periféricos

---

- La programación de los dispositivos de E/S es distinta a la programación de memoria dado que estos operan asincrónicamente de la CPU (y del programa en ejecución)
- Para transferir información, el dispositivo y el procesador se deben sincronizar para el intercambio.
- Hay 2 estrategias para programar interacciones con dispositivos de E/S
  - Sincrónicamente: espero a que el dispositivo finalice la operación de E/S
  - Asincrónicamente: disparo la operación de E/S en el dispositivo y la interfaz de E/S me avisa cuando se ha concretado (esta es la más eficiente)



# Puertos de E/S

---

- Un puerto permite el intercambio de datos entre el bus (conectado a la CPU y a la MP) y el componente de E/S (que a su vez está conectado a los dispositivos)
- Un componente de E/S tiene 3 clase de puertos:
  - Puerto de CONTROL: Se escriben valores que controlan el comportamiento de componente/dispositivo
  - Puerto de STATUS: Se leen valores que representan el estado actual del componente/dispositivo
  - Puerto de DATA: Se leen y escriben valores que hacen al intercambio de la información con el periférico
- Cuando se conectan a un sistema de computación, a cada puerto se le asigna una dirección de E/S
  - Un dispositivo (puerto) es identificado por la dirección de E/S
  - Un dispositivo tiene uno o más puertos de E/S y cada uno de ellos es identificado por una única y unívoca dirección de E/S

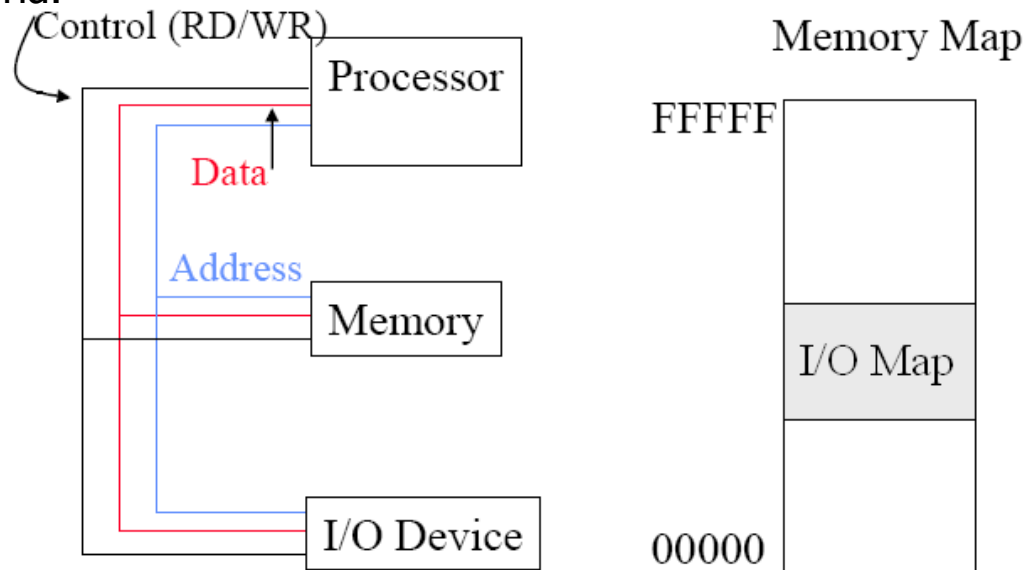


# Direccionamiento de puertos de E/S

En arquitecturas de microprocesadores, existen dos tipos de conexiones de direccionamiento de E/S: Isolated I/O y Memory-Mapped I/O

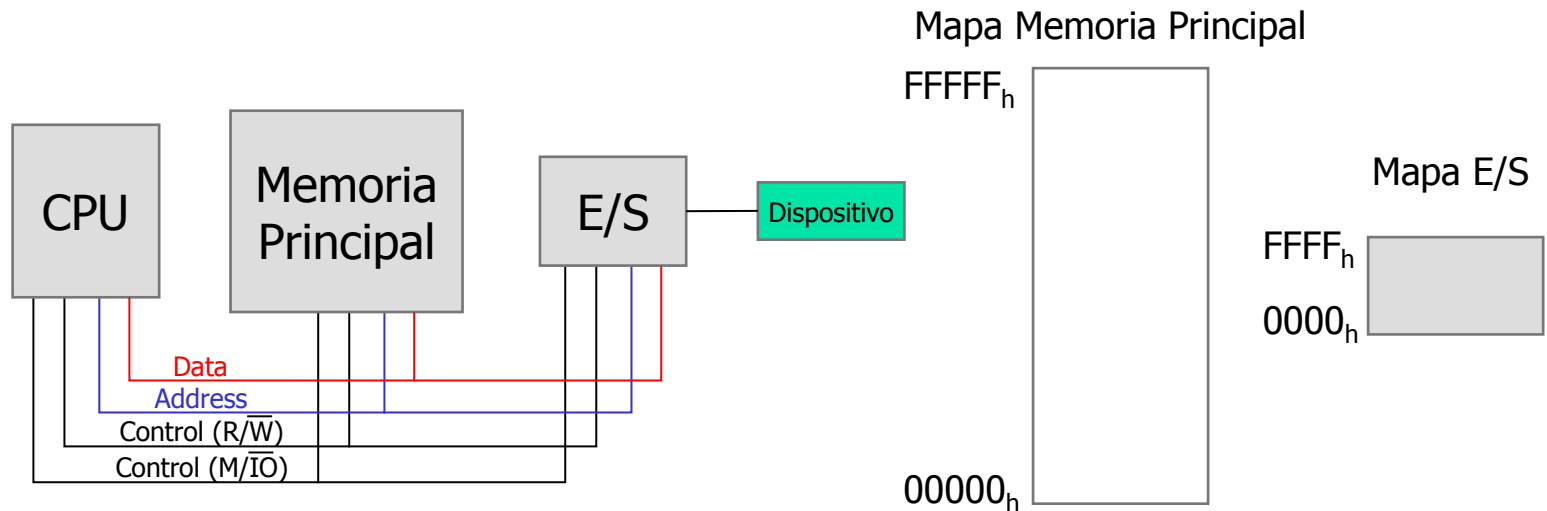
## ■ **Memory-Mapped I/O (Direccionamiento E/S Mapeada a Memoria)**

- El procesador utiliza el mismo set de instrucciones para accesos a memoria y para las operaciones de E/S
- Los dispositivos de E/S y la memoria se encuentran en el mismo espacio de memoria.

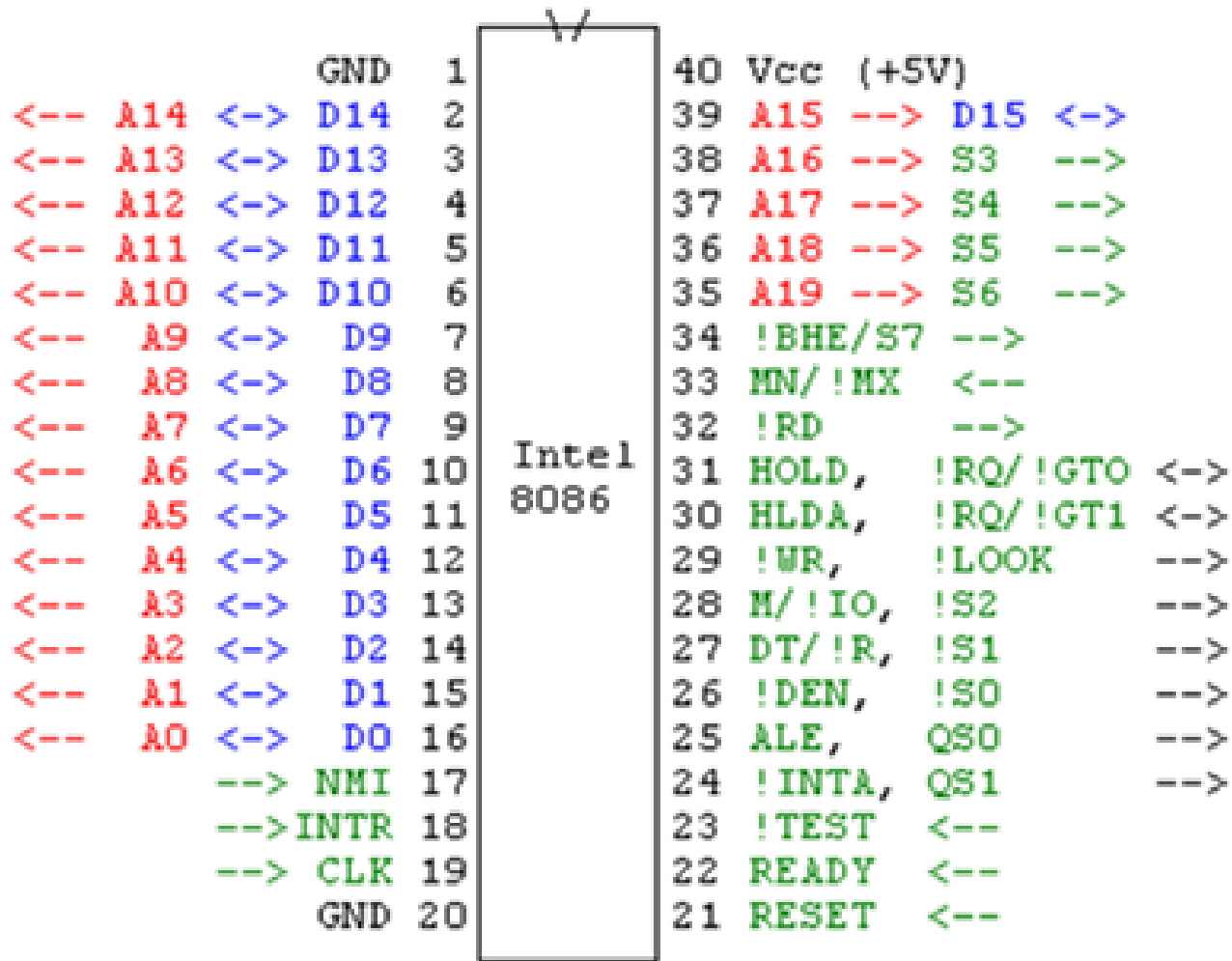


# Direcccionamiento de puertos de E/S

- **Isolated I/O (Direcccionamiento E/S Mapeada Aislada)**
  - El procesador tiene instrucciones dedicadas para operaciones de E/S
  - El procesador tiene un espacio de direcciones separado para dispositivos de E/S
  - 8086: 20 bits para MP y 16 bits para E/S



# 8086/88 Pint Out





# Direcccionamiento de puertos de E/S

---

## ■ **Intel – Direcccionamiento de E/S por Mapeo Aislado**

- Dentro de la familia 80x86, el rango de direcciones para E/S es:  
 $0000_h - FFFF_h$  (16 bits)
- En las PC, los dispositivos tienen asignados direcciones de E/S estándares (usado por todos los fabricantes)
  - Teclado  $60_h$
  - Parlante  $61_h$
  - LPT1  $3BC_h - 3BF_h$



# Direccionamiento de puertos de E/S

- **Intel – Direccionamiento de E/S por Mapeo Aislado**

- Existen instrucciones separadas que transfieren datos desde y hacia los puertos de E/S

- **Transferencia de Memoria**

**MOV Reg, [Memory]**  
**MOV [Memory], Reg**  
**MOV Reg, Reg**  
**MOV Reg, immediate 1-2 bytes**  
**MOV Seg-Reg, [Memory]**  
**MOV [Memory], Seg-Reg**  
**MOV Seg-Reg, Reg**  
**MOV Reg, Seg-Reg**

- **Transferencia de E/S**

**IN AL/AX, immediate byte**  
**IN AX/AX, DX**  
**OUT immediate byte, AL/AX**  
**OUT DX, AL/AX**

# Programación de puertos de E/S

## ■ Instrucción IN

- Lee desde un puerto de E/S y al dato lo guarda en AL o AX
- Sintaxis:
  - `IN AL, imm8` → Dato de **8** bits, puerto de **8** bits
  - `IN AX, imm8` → Dato de **16** bits, puerto de **8** bits
    - `imm8` especifica una dirección de E/S de 8 bits en el rango 00-FF<sub>h</sub>
  - `IN AL, DX` → Dato de **8** bits, puerto de **16** bits
  - `IN AX, DX` → Dato de **16** bits, puerto de **16** bits
    - `DX` contiene una dirección de E/S de 16 bits en el rango 0000-FFFF<sub>h</sub>

## ■ Instrucción OUT

- Escribe a un puerto de E/S con el dato contenido en AL o AX
- Sintaxis:
  - `OUT imm8, AL` → Dato de **8** bits, puerto de **8** bits
  - `OUT imm8, AX` → Dato de **16** bits, puerto de **8** bits
  - `OUT DX, AL` → Dato de **8** bits, puerto de **16** bits
  - `OUT DX, AX` → Dato de **16** bits, puerto de **16** bits

# Programación de puertos de E/S

Utilizo:



El dato es de 1 byte

AL

IN Destino, Origen

El dato es de 2 bytes

AX

Utilizo:



El puerto es de 1 byte

(00-FF<sub>h</sub>)

OUT Destino, Origen

El dato es de 2 bytes

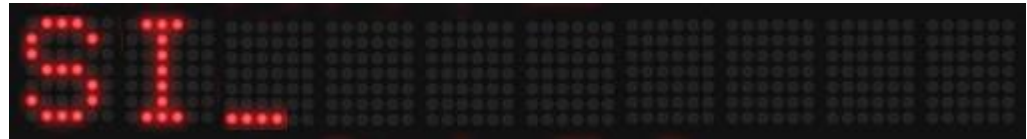
DX

# Programación de puertos de E/S

## ■ Ejemplo:

- Suponga tener un panel luminoso para mostrar caracteres ASCII
- Para el programador el puerto de escritura es 04E9<sub>h</sub>
- El carácter ASCII escrito en el puerto de datos se muestra en la posición actual del cursor del cartel
- Cuando se escribe un carácter, la posición del cursor se avanza.
- Escriba el fragmento de código que muestre el texto "SI"

- MOV DX, 04E9h
- MOV AL, 53h
- OUT DX, AL
- MOV AL, 49h
- OUT DX, AL





# Programación de puertos de E/S

## Data port: 03BCh

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write	Bit 7	Data 7
			Bit 6	Data 6
			Bit 5	Data 5
			Bit 4	Data 4
			Bit 3	Data 3
			Bit 2	Data 2
			Bit 1	Data 1
			Bit 0	Data 0

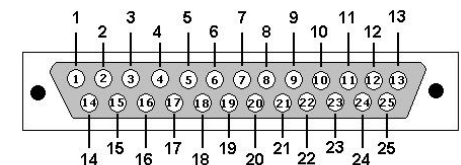
## Control port: 03BEh

Offset	Name	Read/Write	Bit No.	Properties
Base + 2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

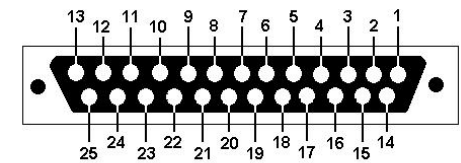
## Status port: 03BDh

Offset	Name	Read/Write	Bit No.	Properties
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Line	DB 25 male (computer)	Centronics (printer)
Strobe	1	⇒ 1
Data bit 0	2	⇒ 2
Data bit 1	3	⇒ 3
Data bit 2	4	⇒ 4
Data bit 3	5	⇒ 5
Data bit 4	6	⇒ 6
Data bit 5	7	⇒ 7
Data bit 6	8	⇒ 8
Data bit 7	9	⇒ 9
Acknowledge	10	⇐ 10
Busy	11	⇐ 11
Paper out	12	⇐ 12
Select	13	⇐ 13
Autofeed	14	⇒ 14
Error	15	⇐ 32
Reset	16	⇒ 31
Select	17	⇒ 36
Signal ground	18	⇐ 33
Signal ground	19	⇐ 19 + 20
Signal ground	20	⇐ 21 + 22
Signal ground	21	⇐ 23 + 24
Signal ground	22	⇐ 25 + 26
Signal ground	23	⇐ 27
Signal ground	24	⇐ 28 + 29
Signal ground	25	⇐ 16 + 30
Shield	Cover	⇐ Cover + 17



View looking into male connector

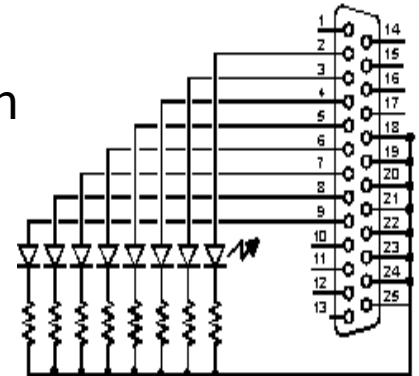


View looking into female connector

# Programación de puertos de E/S

## ■ Encendiendo Luces

- Hay leds conectados a los 8 bits de salida de una interfaz paralelo (LPT $n$ ) unidireccional, cada led es manejado por un bit en el puerto de E/S (datos)
- El puerto de datos es de solo escritura
- El puerto de datos es: 378 $_h$
- Configuración de Bits y LEDs
  - Para encender el led $_n$  hay que escribir un byte con el bit $_{n-1}$  en 1
  - Para apagar el led $_n$  hay que escribir un byte con el bit $_{n-1}$  en 0
- Cuando se desea apagar o prender un solo led, sin variar el estado de los demás, hay que mantener en una variable interna del programa el estado de los bits del puerto. ¿Por qué?

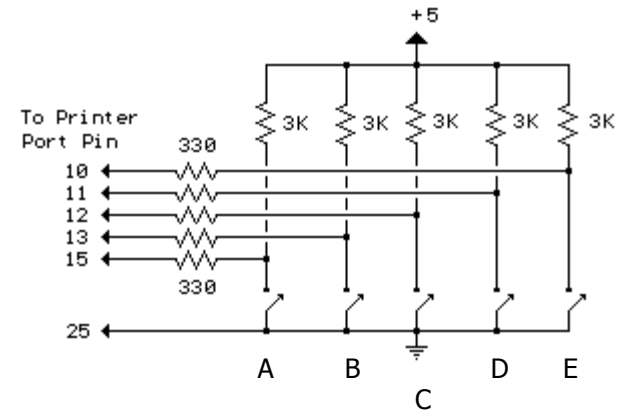


**El valor de cada bit del puerto determina si el led se encenderá o se apagará**

# Programación de puertos de E/S

## ■ Sensando Switches

- Hay 5 switches conectados a los bits del puerto de estados de la interfaz paralelo
- El puerto de estados es:  $379_h$
- Configuración de bits y switches
  - Bit 7 – switch E
  - Bit 6 – switch D
  - Bit 5 – switch C
  - Bit 4 – switch B
  - Bit 3 – switch A
- Si un programa lee el puerto varias veces a alta velocidad, el valor leído variará por un tiempo hasta que se estabiliza (oscila).



**Los switches tienen contactos de metal accionados por una palanca con resortes causando oscilaciones cuando pasa del estado abierto (no hace contacto) a cerrado (hace contacto)**



# Gestión centralizada de atención a dispositivos

---

- Polling de dispositivos de E/S - Espera Activa
  - La CPU pregunta de a un periférico por vez, si requiere atención. Si la necesita, ejecuta las acciones que le son solicitadas. Si no, continúa preguntando al siguiente dispositivo
  - Estas “encuestas” se ejecutan dentro del programa principal y se implementan con saltos condicionados
  - Mientras la CPU atiende a un dispositivo ningún otro puede ser atendido independientemente de su importancia
  - Desventaja: No permite anidamiento
  - Ventaja: No requiere HW adicional, se implementa por software



# Gestión centralizada de atención a dispositivos

## ■ Polling Dispositivos de E/S - Espera Activa (ejemplo ilustrativo)

Proc POLL

:REDO

MOV DX,0349H ; PRINTER STATUS PORT

IN AL,DX

AND AL,40H ; B6 INDICATES PRINTER ERROR

JNZ PRINTERRROR

MOV DX,E004H ; ETHERNET ADAPTER STATUS PORT

IN AX,DX ; 16 BIT DATA STATUS

AND AX,0008H ; B4 INDICATES ADAPTER DISABLED

JNZ DISABLEDETHERNET

RET

:PRINTERERROR

CALL PRINTER\_ERR

JMP REDO

:DISABLEDETHERNET

CALL NETADAPT\_ERR

JMP REDO

POLL endp



# Gestión distribuida de atención a dispositivos

---

## ■ Interrupciones

- Aprovecha los recursos de la CPU de manera más eficiente. Se ejecuta el main y se atienden a los periféricos solos si estos lo solicitan
- Cuando la CPU recibe un pedido de interrupción, finaliza la ejecución de la instrucción en curso y comienza la ejecución de la rutina de atención de la interrupción. Cuando finaliza esta rutina, el micro continúa donde había dejado



# Interrupciones por hardware (HW)

- Una interrupción es un procedimiento desencadenado por un **evento externo** que produce una **transferencia impredecible** del control de ejecución, **desde** el programa o **rutina que es interrumpida hacia otra rutina** (denominada de atención de la interrupción)
- Una interrupción se compone de 4 etapas
  - **Solicitud**: a través de una señal asincrónica de entrada a la CPU
  - **Reconocimiento**: dadas ciertas condiciones temporales y lógicas, la CPU reconoce el pedido y comienza con la atención de la interrupción. La transferencia del control de ejecución no es inmediata ni incondicional. Se identifica al periférico solicitante a través de su "*Tipo=n*". Se obtiene la dirección de memoria donde está la rutina de atención del periférico (*ISR<sub>n</sub>* = Interrupt Service Routine periférico *n*)
  - **Atención**: se ejecuta la rutina de atención ISR
  - **Retorno**: Finalizada la ejecución de la rutina de atención, se produce el retorno del control de ejecución a la rutina que fue interrumpida, permitiendo al micro continuar con la ejecución de la misma.

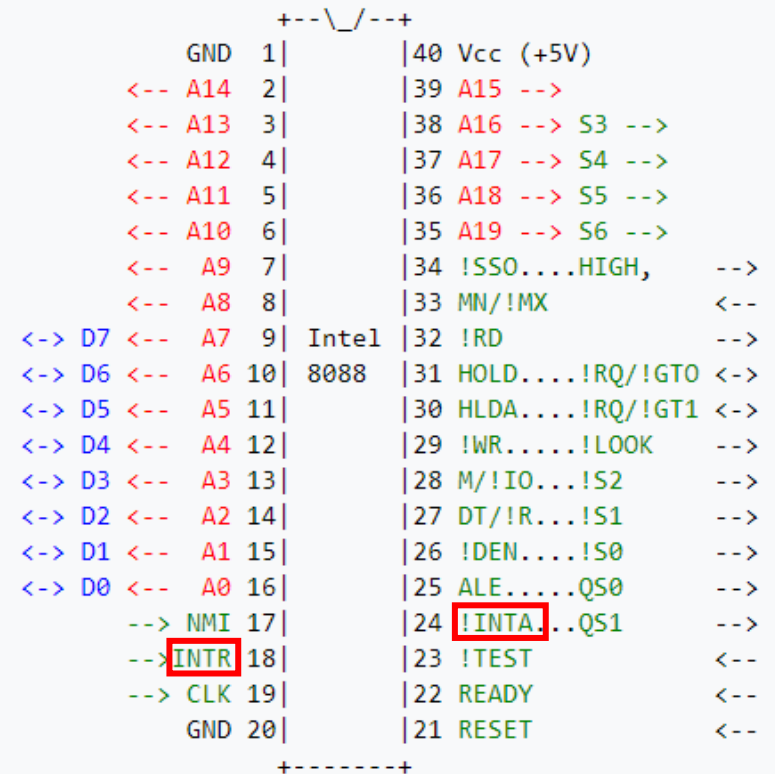
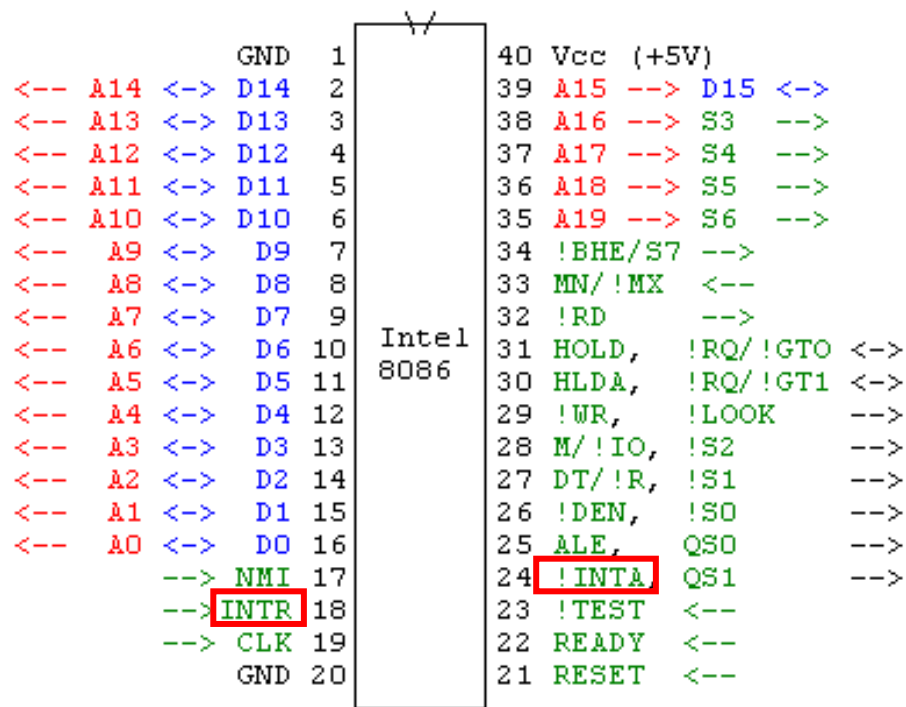


# Interrupciones por HW enmascarables

- Una máscara de interrupción es un bit que condiciona el reconocimiento de la interrupción
- Es la bandera o flag IF (Interrupt Flag)
  - Si  $IF=0$ , la solicitud es ignorada. Si  $IF=1$ , el pedido es aceptado y se inicia el ciclo de reconocimiento
- Las solicitudes se realizan a través de la señal INTR (Interrupt Request)
  - Se censa en el último ciclo de reloj de la instrucción en curso
  - Si es  $INTR=1$ , la CPU emite la señal !INTA o  $INTA=0$  (Interrupt Acknowledge)
  - Cuando comienza la ejecución de la rutina de atención de la interrupción, IF debe pasar a 0 hasta que la línea INTR pase a bajo. Sino fuera así, solo llegaría a ejecutarse la primera instrucción de esta rutina y la CPU comenzaría nuevamente con otro ciclo de reconocimiento.
- Como se leen 8 bits del tipo, hay 256 tipos distintos que pueden atenderse

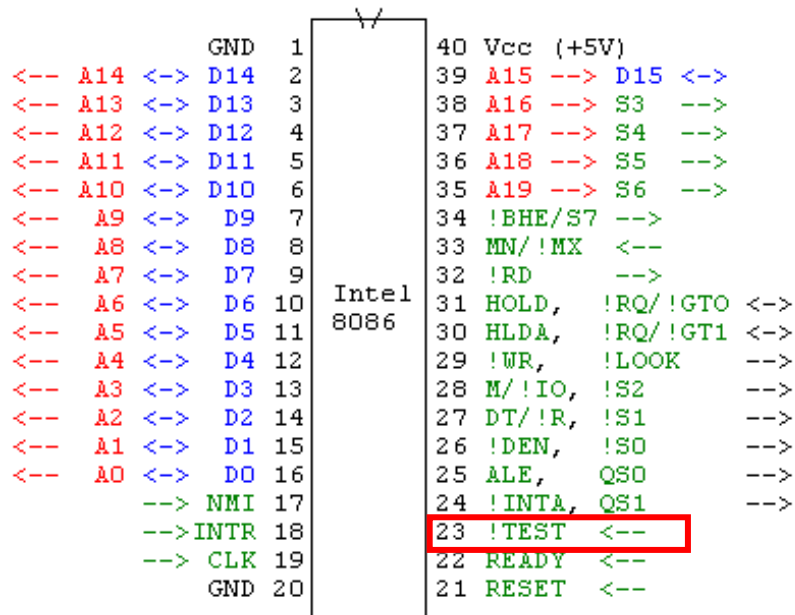


# Interrupciones por HW enmascarables: Señales involucradas



# Interrupciones por HW enmascarables

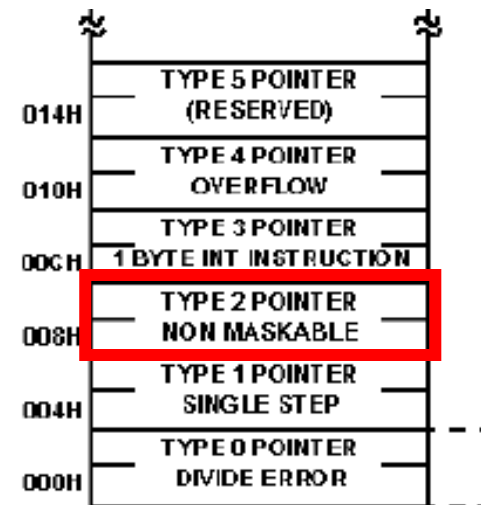
- Interrupciones Enmascarables (cont.)
  - Una excepción es la instrucción WAIT que espera un "0" en la entrada !TEST para continuar con el programa. Esta instrucción muestrea continuamente INTR durante su ejecución, permitiendo la atención de interrupciones en el interior de la espera. Cuando finaliza la rutina de atención, la CPU retorna a ejecutar la instrucción WAIT



# Interrupciones por HW NO enmascarables

## ■ Interrupciones No Enmascarables

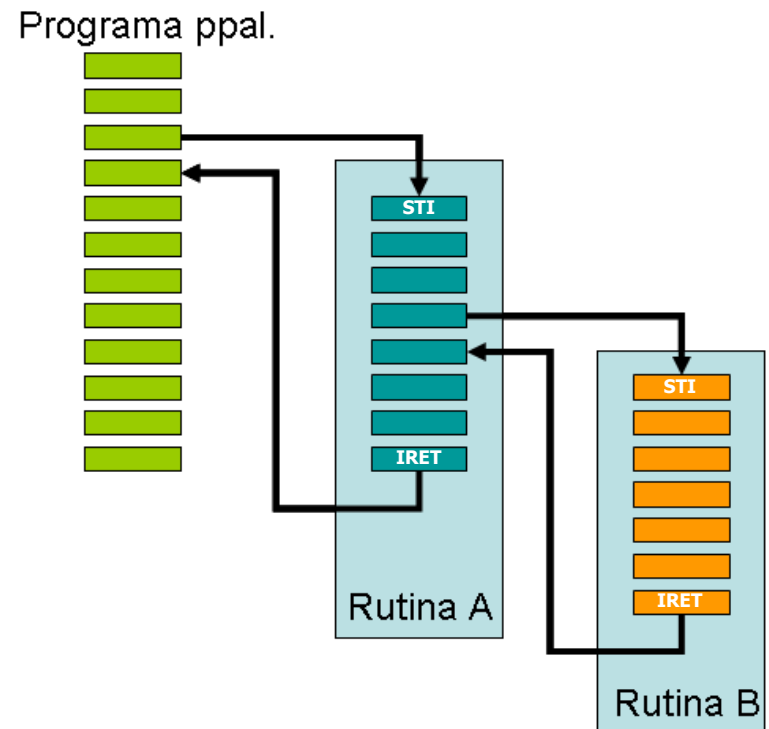
- La solicitud se realiza a través de la entrada NMI (Non Maskable Interrupt). Es una entrada asincrónica activa por flanco ascendente
- Está fijada como tipo "2" (o 02<sub>h</sub>)
- No es necesaria la señal de INTA
- Su atención es inevitable e incondicional. Es siempre la interrupción más prioritaria ya que puede interrumpir la ejecución de cualquier servicio generado por INTR
- NMI, se usa para situaciones graves que requieran atención incondicional de la CPU como errores de paridad, energía baja, etc.



<--	A6 <-->	D6 10	8086	31	HOLD,
<--	A5 <-->	D5 11		30	HLDA,
<--	A4 <-->	D4 12		29	!WR,
<--	A3 <-->	D3 13		28	M/!IO,
<--	A2 <-->	D2 14		27	DT/!R,
<--	A1 <-->	D1 15		26	!DEN,
<--	A0 <-->	D0 16		25	ALE,
	-->	NMI 17		24	!INTA,
	-->	INTR 18		23	!TEST
	-->	CLK 19		22	READY
		GND 20		21	RESET

# Interrupciones por HW: Anidamiento

- Una interrupción se anida en otra, cuando ésta interrumpe la ejecución de la ISR que está ejecutándose
- Esto puede evitarse si se utiliza el flag IF
  - No deseo anidamiento
    - Como la CPU coloca el  $IF=0$  antes de comenzar la ejecución de la ISR, no modifico el flag IF
  - Deseo anidamiento
    - La primera instrucción de atención a la interrupción tiene que ser la habilitación de atención de nuevas interrupciones colocando el  $IF=1$  (instrucción STI)
- En ambos casos, al ejecutarse la instrucción IRET presente en la ISR, se restaurarán los flags tal como estaban antes de la interrupción.



# Interrupciones por HW: Anidamiento

- La atención de una interrupción (ejecución de su ISR) se “anida” dentro de otra cuando ésta última interrumpe la ISR que está en curso

## Programa principal

```
.....
0430:0100 MOV BL, 34
0430:0102 MOV CX, 0012
0430:0105 MOV DX, 03BF
INTR (Tipo 0A) → 0430:0108 OR AL, BL
!INTA ← 0430:010A CMP AL, 0
0430:010C JNZ 0130
0430:010E IN AL, DX
0430:010F MOV [BX], AX
0430:0112 DEC BL
0430:0114 XOR AL, BL
0430:0116 INC AL
.....
```

## ISR<sub>0A</sub>

```
3F4C:0000 STI
3F4C:0102 PUSH AX
3F4C:0105 CALL 01BF
INTR (Tipo 1F) → 3F4C:0109 MOV AH, 9
!INTA ← 3F4C:010B SHR AL, 1
.....
3F4C:0231 XOR AX, AX
3F4C:0233 POP AX
3F4C:0235 IRET
```

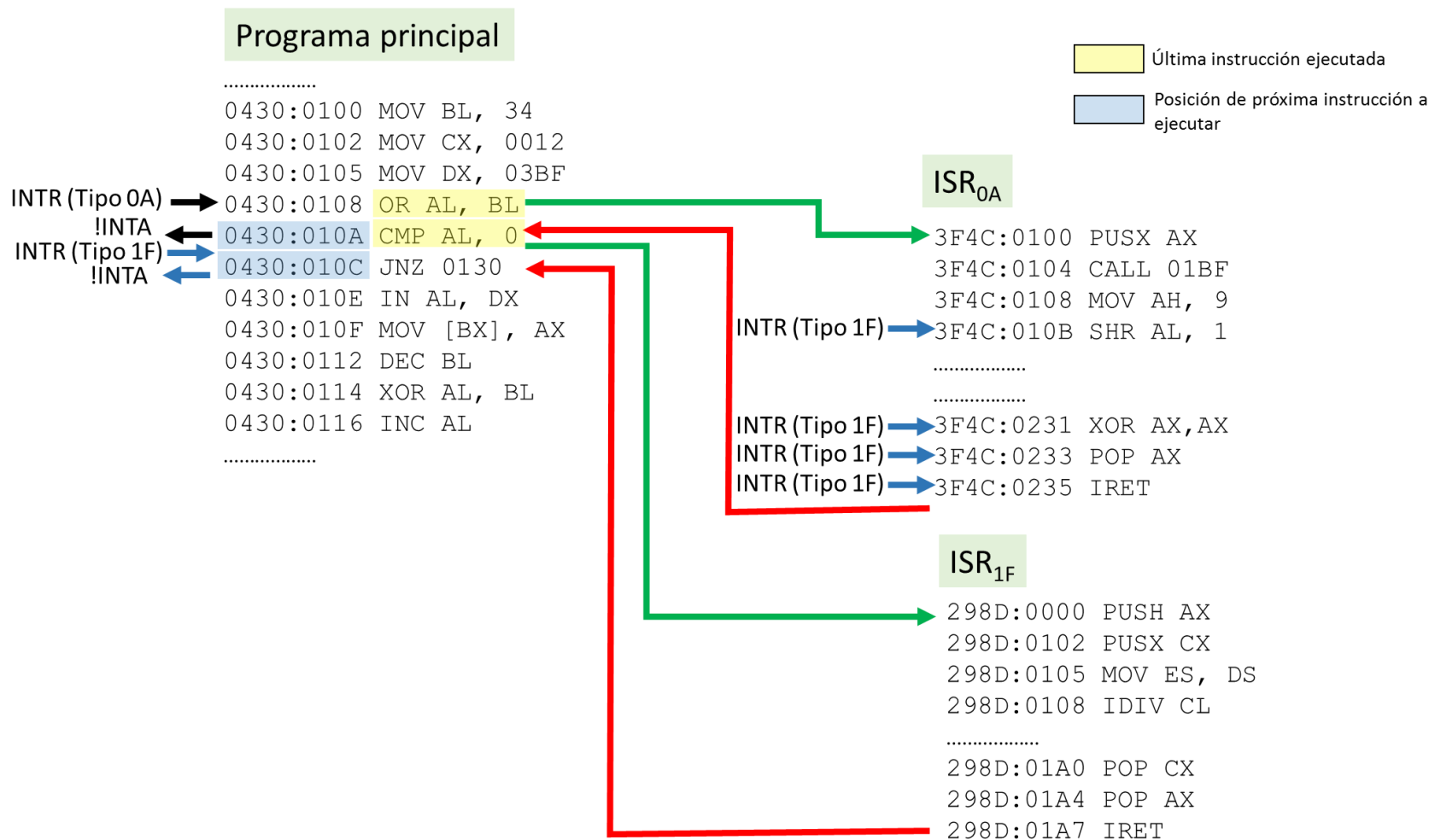
## ISR<sub>1F</sub>

```
298D:0000 PUSH AX
298D:0102 PUSH CX
298D:0105 MOV ES, DS
298D:0108 IDIV CL
.....
298D:01A0 POP CX
298D:01A4 POP AX
298D:01A7 IRET
```

- Última instrucción ejecutada
- Posición de próxima instrucción a ejecutar
- Instrucción STI

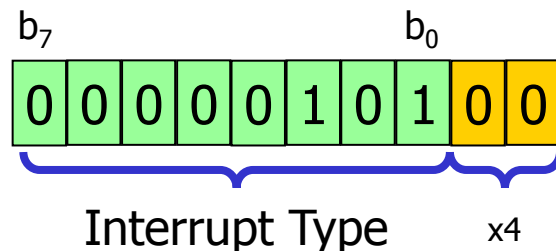
# Interrupciones por HW: Anidamiento

- ¿Cómo podemos evitar el anidamiento?



# Interrupciones por HW: Tablea de vectores de interrupción (IVT)

- Una vez reconocido el pedido, se debe individualizar al solicitante
- Se inicia el ciclo INTA leyendo un byte del bus de datos
- Este byte contiene el "tipo" (*Interrupt Type*) de la interrupción solicitada. En el 8086, habrá 256 tipos
- Este número actúa como puntero dentro de una tabla llamada vector de interrupciones (IVT = *Interrupt Vector Table*) que contiene a su vez el puntero a donde se encuentra la rutina de atención para cada tipo. Cada entrada tiene 4 bytes, 2 para el CS y 2 para el IP
- Por ejemplo, la interrupción 5 o  $\text{type} = 05_h$ , le corresponde el 6° vector de interrupciones cuyos 4 bytes se ubican en las direcciones  $00014_h$  a la  $00017_h$ .



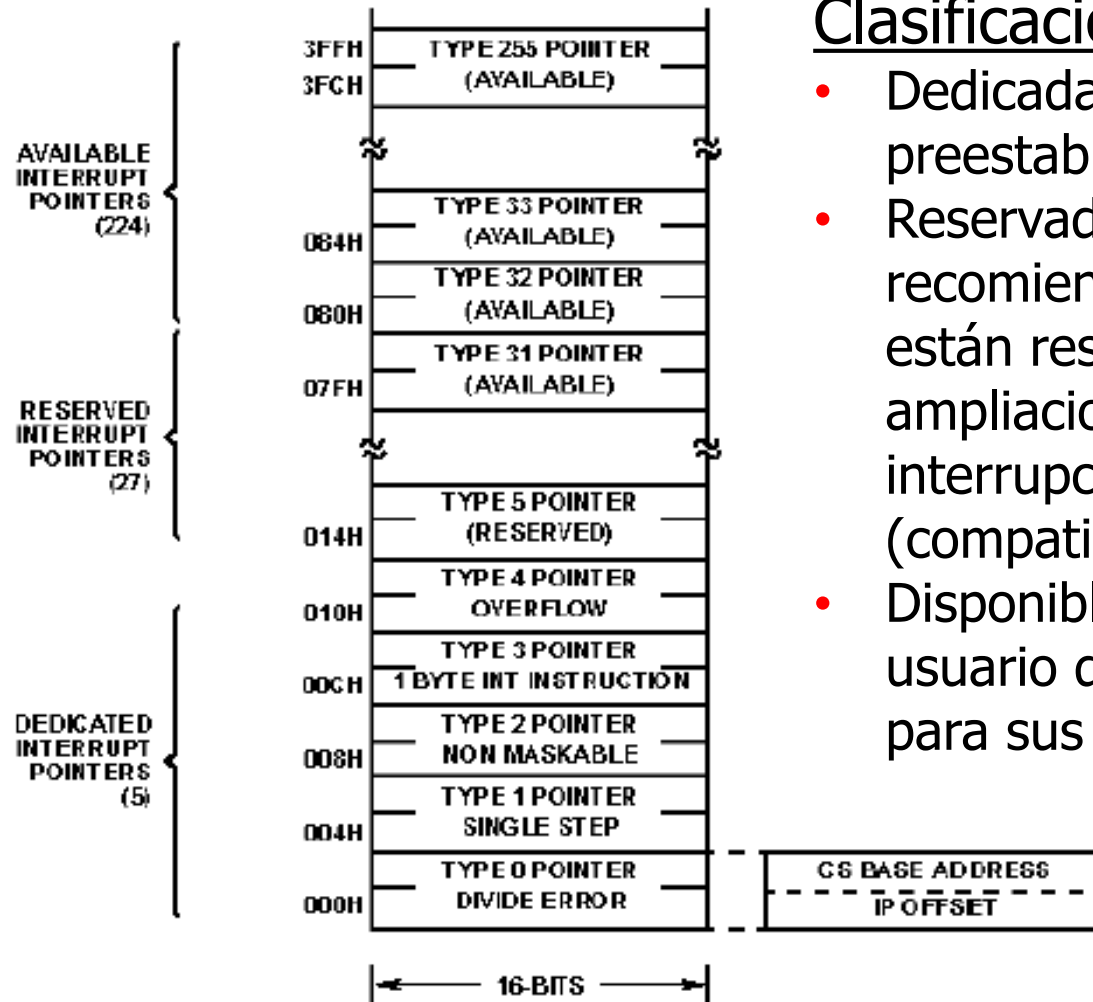
Dirección inicial =  $14_h$   
 $14-15-16-17 \rightarrow \text{CS:IP}$

# Interrupciones por HW: Tablea de vectores de interrupción (IVT)

- Es un sector de la memoria que se encuentra en los primeros 1024 bytes del mapa. Va desde la dirección:  $00000_h$  a la  $003FF_h$
- Contiene 256 vectores de interrupción, de 4 bytes cada uno
- Al residir en la memoria RAM, se podrá modificar el contenido del durante la ejecución de un programa
- Si un vector no se usa, se deberá apuntar a una ISR que solo contenga la instrucción IRET para retornar sin hacer nada
- Vector de interrupciones (4 bytes dentro de la IVT)
  - Cada vector está tiene 2 bytes para alojar el IP y 2 para el CS
  - El sentido es:
    - 1er byte → LSB del IP (nibble menos significativo del IP)
    - 2do byte → MSB del IP (nibble más significativo del IP)
    - 3er byte → LSB del CS
    - 4to byte → MSB del CS



# Interrupciones por HW: Tabla de vectores de interrupción (IVT)



## Clasificación Intel

- Dedicadas (tipos  $00_h$  a  $04_h$ ): están preestablecidas y son fijas.
- Reservadas (tipos  $05_h$  a  $1E_h$ ): no se recomienda usar estos tipos porque están reservados para futuras ampliaciones del esquema de interrupciones del procesador (compatibilidad futura)
- Disponibles (tipos  $1F_h$  a  $FF_h$ ): el usuario dispone libremente su uso para sus propias rutinas



# Interrupciones por HW: Detalle de eventos por etapa

---

- Solicitud
  - La CPU recibe un flanco ascendente en el pin INTR
- Reconocimiento
  - Se evalúa la máscara de interrupciones si está activa (IF=1) y el microprograma se encuentra en la última microinstrucción
  - Desactiva la máscara de interrupción IF (IF=0)
  - Se coloca un flanco descendente en el pin !INTA
  - El contador de programa se encuentra en la próxima instrucción a ejecutarse en el programa interrumpido
  - Guarda el CS, IP y Flags en la pila (STACK)
  - Efectúa un ciclo de lectura sobre el bus de datos para capturar el *Interrupt Type*
  - Calcula (*Interrupt Type* x 4) el puntero al vector de interrupciones (IVT)
  - Lee en MP la dirección del puntero obtenido y las siguientes 3 (base + 3)
  - Se carga el CS y el IP en con el contenido del IVT



# Interrupciones por HW: Detalle de eventos por etapa

---

- Atención
  - Comienza la ejecución de la Interrupt Service Routine (ISR) en la dirección apuntada por el IVT y cargada en CS e IP
- Retorno
  - Al ejecutar la instrucción Interrupt Return (IRET) se restablecen de la pila los valores de los Flags, IP y CS guardados previamente
  - Continúa la ejecución de la instrucción siguiente del programa interrumpido (que estaba guardada en la pila)



# Pseudo-Interrupciones

---

- Por software
- Por error
- Trampa (o Single Step)



# Pseudo-interrupciones: Por software

- **Son instrucciones** que provocan que la CPU ejecute alguna de las rutinas de atención de interrupciones del “tipo” que se indique. Finalizada la ejecución de la rutina de atención, la CPU continúa ejecutando la instrucción posterior a la instrucción de interrupción
- No son interrupciones propiamente dichas ya que al ser instrucciones, **no son impredecibles**
- Como son instrucciones, **no respetan ningún esquema de prioridades y se anidan siempre** que se incluyan dentro de una rutina de atención de interrupción. Obviamente no son enmascarables
- Estas instrucciones pueden usarse para transferir el control a rutinas que son reubicables en memoria (en las que el programa que las llama no conoce la dirección de inicio de las mismas)
- Guardan en la pila las banderas y el Program Counter (CS:IP)
- No ejecutan el ciclo de INTA, pero si inhiben Single Step y IF durante su ejecución

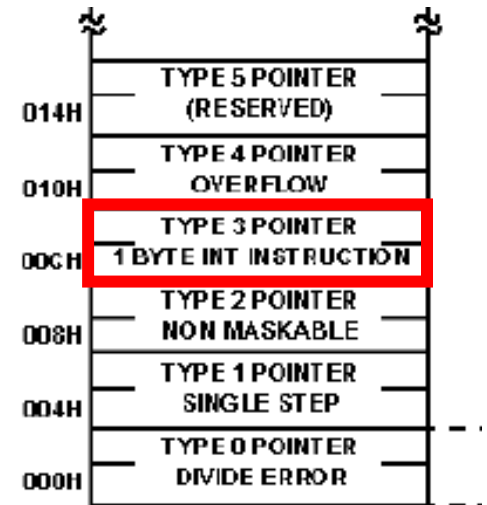
# Pseudo-interrupciones: Por software

## ■ Instrucción INT 3<sub>h</sub>

- La instrucción ocupa 1 byte de longitud
- Interrupción de Tipo=03<sub>h</sub>
- Se utiliza para insertar puntos de ruptura en rutinas que están siendo depuradas

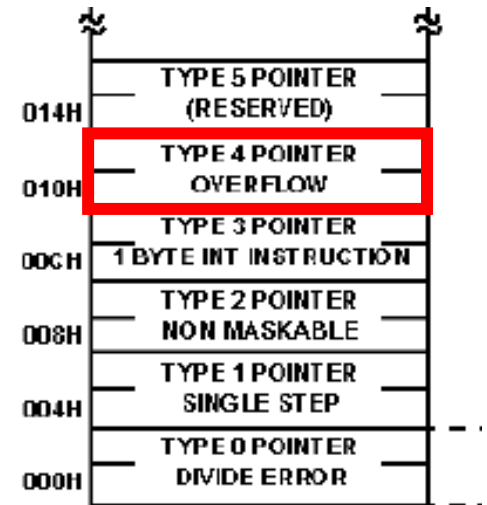
## ■ Instrucción INT XX<sub>h</sub>

- Ejecuta el servicio de la interrupción del Tipo=XX<sub>h</sub>
- Ocupa dos bytes, uno para el código de operación y otro para el para el dato del tipo
- Se puede usar para simular cualquier tipo de interrupción y para depuración
- ¿Cómo puede manejar varios servicios una rutina ISR?
  - El tipo de servicio se pasa como parámetro
  - Los parámetros se pasan por los registros, no por el stack
  - El valor de retorno, depende del servicio invocado



# Pseudo-interrupciones: Por software

- Instrucción INTO (Interrupt On Overflow)
  - Ocupa 1 byte
  - Tipo=04<sub>h</sub>
  - Interrumpe el programa si existe un overflow luego de una operación indicado por la activación del flag OF (Overflow Flag)
  - Si OF = 0 la instrucción no hace nada (se comporta como un NOP) y retorna a la próxima instrucción del INTO
  - Si OF=1 ejecuta la rutina que apunta el IVT en el vector 4
  - Esta instrucción permite corregir errores en el caso de overflow





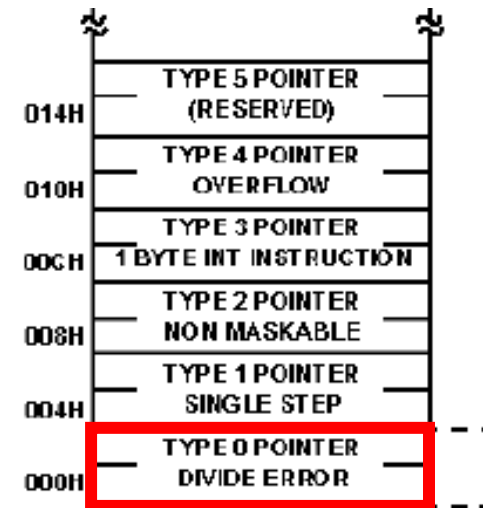
# Pseudo-interrupciones: Por software

- Funciones instalada por el DOS: (Int 21<sub>h</sub>)
  - Imprimir un mensaje (parámetro)      DS:DX = dir. Mensaje; AH = 9<sub>h</sub>
  - Salir      AH = 4C<sub>h</sub>
  - Leer carácter de teclado      AH = 1<sub>h</sub> retorna en AL el ASCII del teclado
  - Mostrar carácter en pantalla      DL = caracter ASCII ; AH = 2<sub>h</sub>
  - Imprimir carácter      DL = caracter ASCII ; AH = 5<sub>h</sub>
- Servicios del BIOS
  - Display      (Int 10<sub>h</sub>)
  - E/S a discos      (Int 13<sub>h</sub>)
  - Teclado      (Int 16<sub>h</sub>) [Lectura AH=0<sub>h</sub>]
  - Impresoras      (Int 17<sub>h</sub>) para funciones de impresora



# Pseudo-interrupciones: Por error

- Se producen por lo general por un error de software
- No respetan esquema de prioridades
- En el 8086 hay solo una: "Tipo"=0 – Error al dividir
  - Ocurre cuando se intenta dividir por 0
  - La dirección de retorno de la rutina de atención de este tipo apunta a la misma instrucción que se terminó de ejecutar antes de producirse este error
  - Permite corregir el error que provocó la interrupción en la rutina de atención y volver a empezar
  - No es enmascarable.



# Pseudo-interrupciones: Por trampa (ejecución paso-a-paso)

- Si el bit TF (Trap Flag) está activo, después de la ejecución de cada instrucción se produce una interrupción "tipo"=1
- Al aceptar esta interrupción, se borra el TF de manera de ejecutar la rutina de atención en forma normal
- Se usa para implementar una rutina para el monitoreo durante la depuración de programas
- Comienza a ejecutarse una instrucción después de que se activó el bit TF
- No es enmascarable

- Establecer el TF  
PUSHF  
POP AX  
OR AX,0100<sub>h</sub>  
PUSH AX  
POPF

014H	TYPE 5 POINTER (RESERVED)
010H	TYPE 4 POINTER OVERFLOW
00CH	TYPE 3 POINTER 1 BYTE INT INSTRUCTION
008H	TYPE 2 POINTER NON MASKABLE
004H	TYPE 1 POINTER SINGLE STEP
000H	TYPE 0 POINTER DIVIDE ERROR

C : acarreo en la suma y arrastre en la resta

P : paridad del dato (0, impar y 1, par)

A : acarreo auxiliar o arrastre entre los bits 3 y 4

Z : indicación de resultado igual a cero

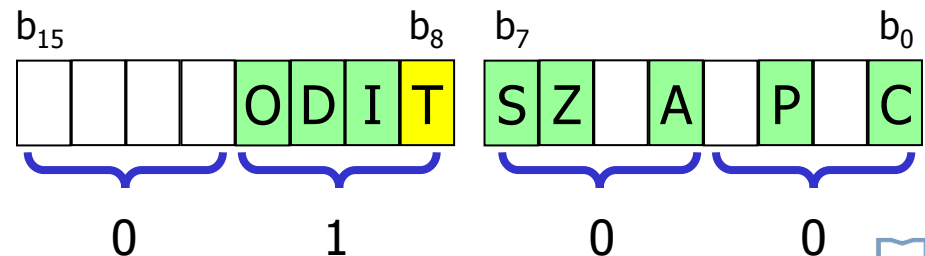
S : indicador de signo del resultado. 0, positivo y 1, negativo

T : trampa. Habilita la característica de depuración

I : habilitación de interrupciones de hardware

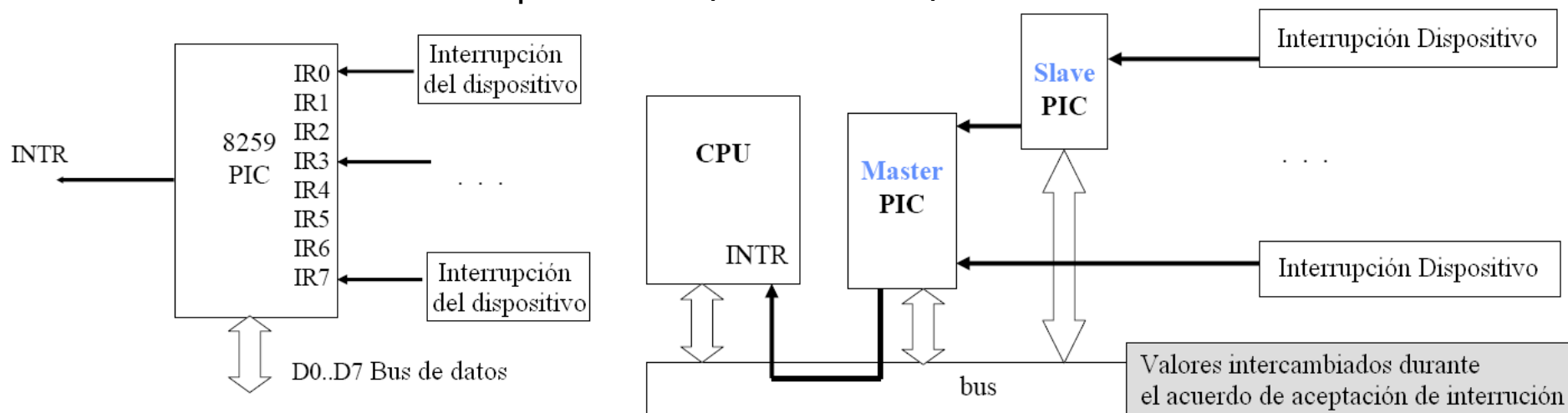
D : selección de incremento o decremento en los índices

O : sobreflujo

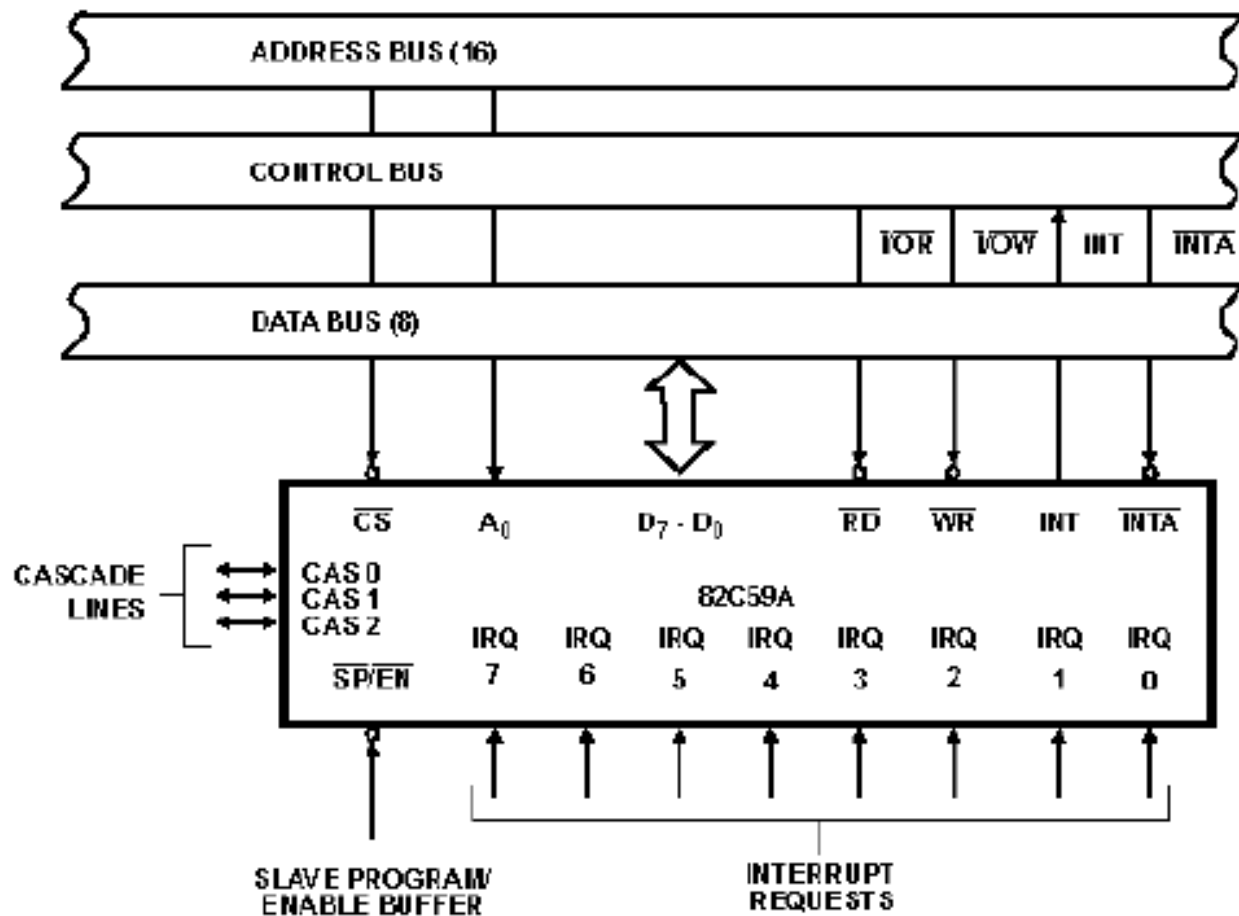


# Interrupciones por HW: Controlador de interrupciones programable autovectorizadas Intel 8259A

- Intel 8259A (PIC = Programmable Interrupt Controller)
  - Amplía la estructura de interrupciones del 8086
  - Es configurable y programable (asociación de tipos a dispositivos, prioridades)
  - Se lo considera como un dispositivo más del chipset
  - Conectándolo 8 8259A en cascada (8+1) → maneja 64 interrupciones
  - Durante el booteo, la BIOS programa el PIC maestro:
    - IR0 a IR7 se mapean a los tipos de interrupciones 08h a 0Fh



# Controlador de interrupciones Intel 8259A





# Controlador de interrupciones Intel 8259A

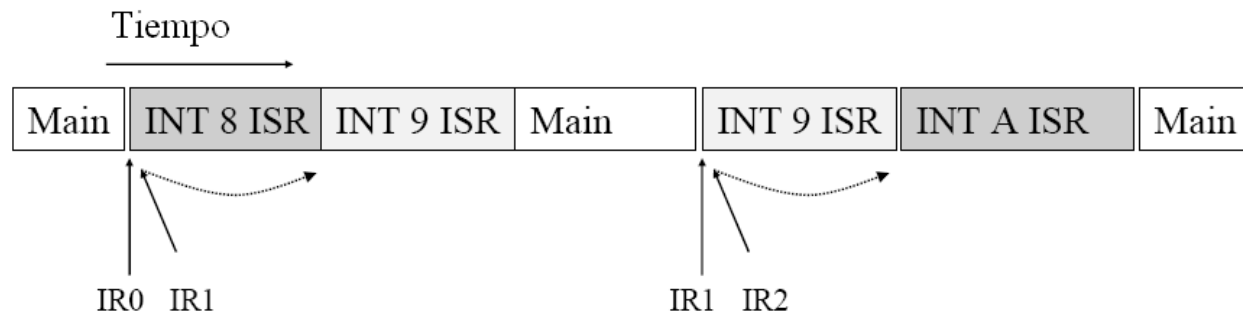
- Prioridad de Interrupción
  - Los dispositivos tienen asignados prioridades
    - Los dispositivos con alta prioridad tiene precedencia sobre los de baja prioridad
    - Las prioridades entran en juego cuando:
      - Varias interrupciones suceden en el mismo tiempo
      - Nuevas interrupciones ocurren mientras se está procesando la rutina ISR de interrupciones previas
  - El controlador de interrupciones hace cumplir las prioridades
    - En PCs
      - Los dispositivos tienen conexiones preconfiguradas hacia el controlador
        - Timer siempre en IR0 y el teclado en IR1
      - El DOS programa al 8259A para asignar prioridades basadas en las conexiones de los dispositivos
        - IR0== alta prioridad e IR7==baja prioridad

El número mas bajo  
tiene la prioridad  
más alta

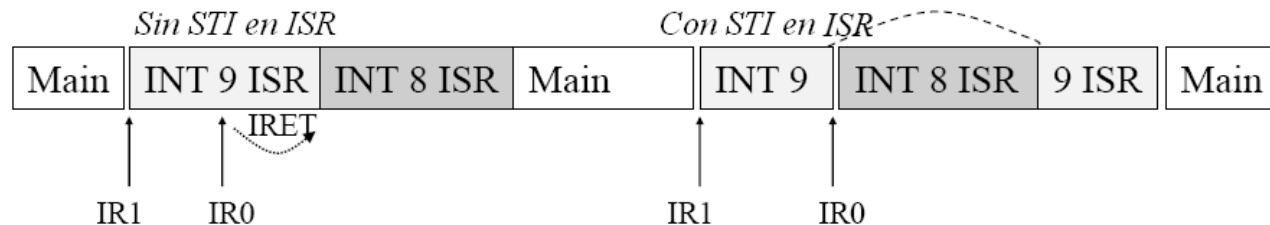
# Controlador de interrupciones Intel 8259A

## ■ Prioridad de Interrupción

- Que rutina ISR se ejecuta si dos interrupciones ocurren al mismo tiempo



- Si la CPU esta ejecutando una rutina ISR y un segundo dispositivo (de mayor prioridad) interrumpe, ¿que ISR sigue ejecutando?

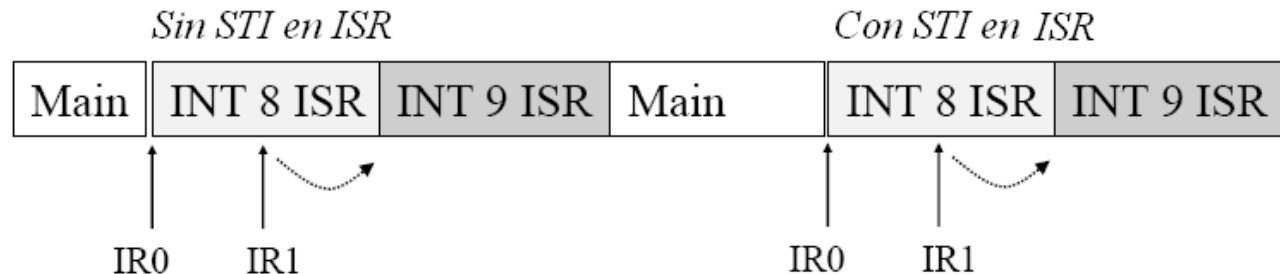


- El controlador intentará permitir a una interrupción de mayor prioridad interrumpir otra de menor. La segunda interrupción no será reconocida por el procesador sino hasta que IF=1 (habiliten las interrupciones)

# Controlador de interrupciones Intel 8259A

## ■ Prioridad de Interrupción

- Si una interrupción de baja prioridad sucede a otra de alta prioridad, el controlador mantiene la prioridad.



- El controlador recuerda la interrupción de baja prioridad hasta que la de alta prioridad haya terminado
- Cuando terminó, el controlador genera otra interrupción a favor del dispositivo lento
  - ¿Cuántas interrupciones puede recordar el controlador?
  - ¿Cómo sabe el controlador que la interrupción de alta prioridad terminó?



# Controlador de interrupciones Intel 8259A

---

- Interrupciones pendientes
  - Se llama pendiente a una señal de interrupción que es almacenada en un latch, pero no ha sido reconocida por el procesador
    - Las interrupciones pueden estar pendientes en el controlador o en el dispositivo
- El Intel 8259A tiene un registro interno de 8 bits
  - Un bit por cada entrada IR
  - Cuando se enciende IR, el bit asociado a él se enciende
  - Cuando la interrupción es reconocida, el bit asociado se apaga.
    - El controlador puede recordar hasta una interrupción pendiente por IR





# Controlador de interrupciones Intel 8259A

---

- Fin de Interrupción (EOI)
  - Después de enviar una interrupción al procesador, el controlador necesita saber cuando es seguro, generar una interrupción de menor prioridad
    - Esto hace necesario tener un feedback desde la CPU
  - Se envía un comando al controlador desde la CPU indicando el fin de la interrupción en curso.
    - No es parte del ciclo INTA, no es hecho por el hardware
    - Se trata de un comando software, que el programa debe enviar (ICW, OCW)



# Controlador de interrupciones Intel 8259A

- Modelo del programador (Controlador de Interrupciones)
  - El Controlador es un dispositivo de E/S → tiene una dirección de un puerto de E/S
  - Dos puertos de 8 bits cada uno
  - Interrupt Mask Register → (Puerto 21h) → lectura/escritura
    - Permite habilitar/deshabilitar interrupciones individuales en el controlador
    - $\text{bit}_n = 1 \rightarrow \text{IR}_n$  enmascarado (se ignora la interrupción)
    - $\text{bit}_n = 0 \rightarrow \text{IR}_n$  es habilitado
    - Al revés del IF (*Interrupt Flag*)
  - Command Register → (Puerto 20h) → escritura solamente
    - Se escribe en  $20_h$  para informar al controlador que terminó la interrupción



# Controlador de interrupciones Intel 8259A

---

- Modelo del programador (Teclado de PC)
  - El teclado de PC usa interrupciones
    - Se puede usar polling del puerto de estado para detectar si hay una tecla presionada pendiente de ser enviada.
    - Está conectado al IR1 del controlador, a través del chip 8255.
  - El 8255 Programable Peripheral Interface tiene 2 puertos
    - Data Port (Port PA) → dirección de E/S =  $60_h$ 
      - Lectura = tecla presionada
    - Control Port (Port PB) → dirección de E/S =  $64_h$ 
      - Lectura = Buffer con teclas presionadas



# Interrupciones de la BIOS

---

- Las Interrupciones de la BIOS
  - INT 10h: Servicios de Vídeo (texto y gráficos).
  - INT 11h: Informe sobre la configuración del equipo.
  - INT 12h: Informe sobre el tamaño de la memoria convencional.
  - INT 13h: Servicios de disco (muy elementales: pistas, sectores, etc.).
  - INT 14h: Comunicaciones en serie.
  - INT 15h: Funciones cassette (PC) y servicios especiales del sistema (AT).
  - INT 16h: Servicios de teclado.
  - INT 17h: Servicios de impresora.
  - INT 18h: Llamar a la ROM del BASIC (sólo máquinas IBM).
  - INT 19h: Reinicialización del sistema.
  - INT 1Ah: Servicios horarios.
  - INT 1Fh: Apunta a la tabla de los caracteres ASCII 128-255 (8x8 puntos).



# Interrupciones del D.O.S.

---

- Las Funciones del DOS

- INT 20h: Terminar programa (tal vez en desuso).
- INT 21h: Servicios del DOS. (devuelve acarreo si hay error)
- INT 22h: Control de finalización de programas.
- INT 23h: Tratamiento de Ctrl-C.
- INT 24h: Tratamiento de errores críticos.
- INT 25h: Lectura absoluta de disco (sectores lógicos).
- INT 26h: Escritura absoluta en disco (sectores lógicos).
- INT 27h: Terminar dejando residente el programa (en desuso).
- INT 28h: Idle (ejecutada cuando el ordenador está inactivo).
- INT 29h: Impresión rápida en pantalla (no tanto).
- INT 2Ah: Red local MS NET.
- INT 2Bh-2Dh: Uso interno del DOS.
- INT 2Eh: Procesos Batch.
- INT 2Fh: Interrupción Multiplex.
- INT 30h-31h: Compatibilidad CP/M-80.
- INT 32h: Reservada.



# Acceso Directo a Memoria (DMA)

---

- Es una técnica que permite a los periféricos realizar transferencias sobre la memoria sin la intervención de la CPU (ésta continúa haciendo otras tareas)
- En el momento en que el DMA realiza las transferencias, la CPU se desconecta de los buses (3-state) y cede el control
- Esta “petición de buses” y control es recibida por la CPU en el pin HOLD y concede la operación a través del pin HOLDA (HOLD Acknowledge) una vez terminado el ciclo de bus en curso
- El dispositivo que controla los buses se lo denomina “Master” y el que es controlado, “Slave”. En este caso, la CPU deja de ser el Master de los buses

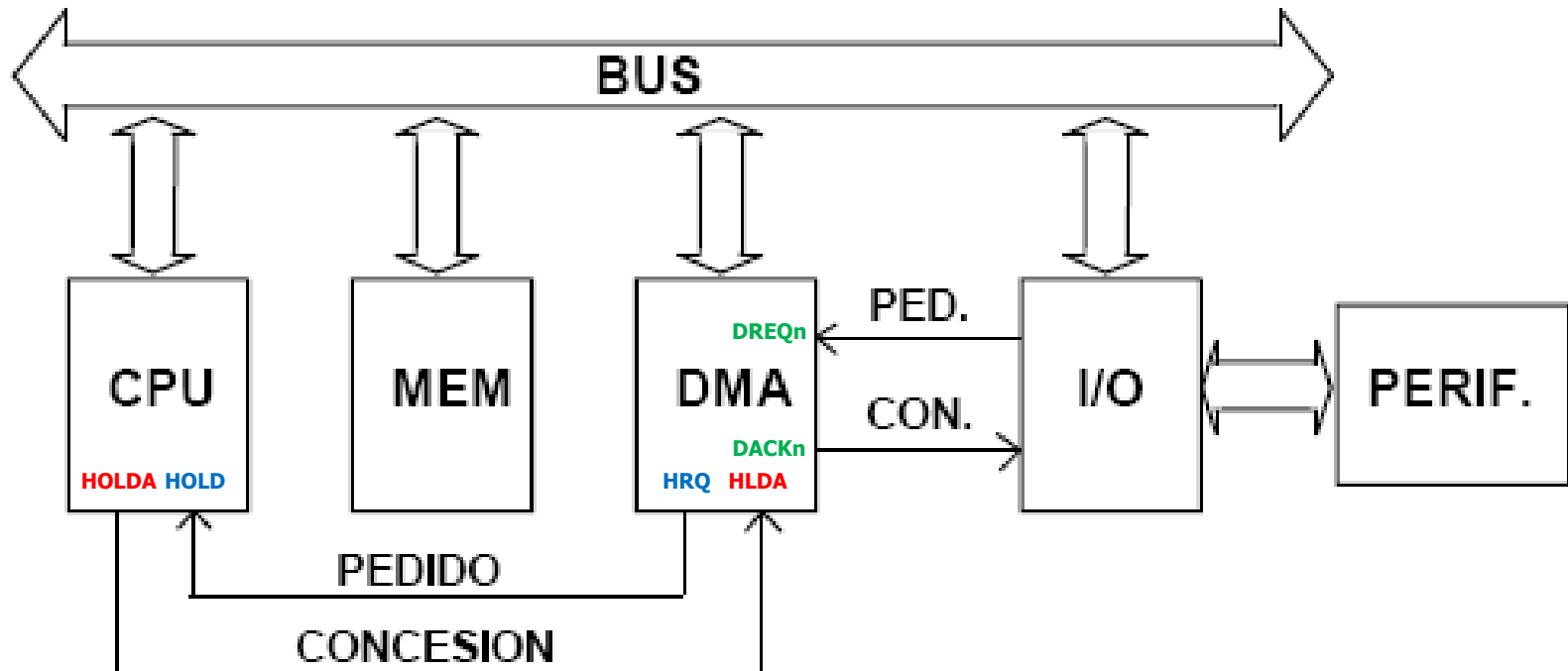


# Controlador de Acceso Directo a Memoria Intel 8237A

---

- DMAC 8237A (DMA Controller)
- Posee 4 canales programables en 3 modos diferentes y cada modo en 3 tipos de transferencia
- Permite transferencias “memoria-a-memoria”
- Direccionamiento por incrementos / decrementos
- Hasta 1.6Mb/s
- Expandible a  $n$  canales DMA
- Señales principales
  - $DREQ_n$  (DMAC)  $\leftarrow$  Dispositivo : solicitud de acceso DMA
  - $HRQ$  (DMAC)  $\rightarrow$  CPU (HOLD): solicitud de buses a la CPU
  - $HLDA$  (DMAC)  $\leftarrow$  CPU (HOLDA): aceptación de la CPU
  - $DACK_n$  (DMAC)  $\rightarrow$  Dispositivo : otorgamiento del servicio DMA

# Controlador de Acceso Directo a Memoria Intel 8237A







# Controlador de Acceso Directo a Memoria Intel 8237A

---

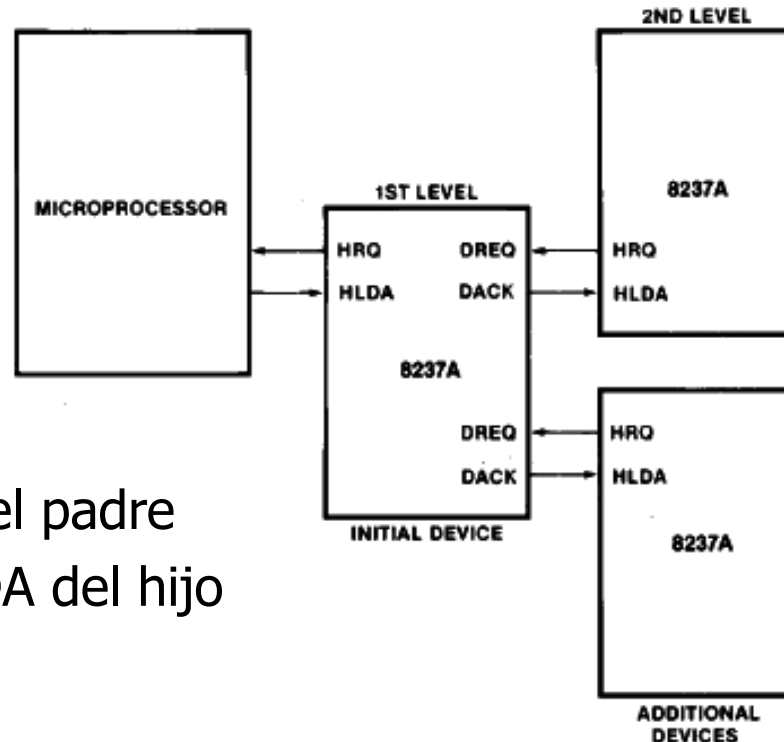
- Modos de operación

- Modo Simple (Single Transfer Mode): se realiza una única transferencia y se asegura que la CPU complete un ciclo antes de realizar una nueva transferencia
  - El “rollover” del Terminal Count (de  $0000_h$  a  $FFFF_h$ ) indica el fin de la transferencia. En este momento, se genera un pulso en End-of-Process (!EOP), pin bidireccional
- Modo En Bloques (Block Transfer Mode): Se realizan transferencias de bloques hasta el rollover del TC o hasta que se detecte un EOP externo
- Modo A Demanda (Demand Transfer Mode): La transferencia se realiza mientras DREQ permanece activo o hasta el rollover del TC o un EOP externo. Se pueden realizar todas las transferencias necesarias hasta agotar las posibilidades del dispositivo

# Controlador de Acceso Directo a Memoria Intel 8237A

## ■ Conexión en Cascada (Cascade Mode)

- Permite la interconexión de más de un 8237A. Permite la propagación de señales de petición y su correspondiente prioridad.



- HRQ del hijo va al DRQ del padre
- DACK del padre va al HLDA del hijo



# Controlador de Acceso Directo a Memoria Intel 8237A

---

- Tipos de transferencias
  - Lectura (Read): Movimiento de datos desde la memoria a la E/S
  - Escritura (Write): Movimiento de datos desde la E/S a la memoria
  - Verificación (Verify): Se generan direcciones pero no se utilizan las señales de control (son pseudo-transferencias)
- Transferencia Memoria-a-Memoria
  - Se emplean los canales 0 y 1
  - Se realiza un pedido de DMA normal
    - Por canal 0 y se lee dato desde la memoria y se guarda en el 8237A
    - Por canal 1 se escribe en la memoria el dato guardado en el 8237A
  - Autoinicialización
  - Se restablecen los registros internos a los valores predeterminados



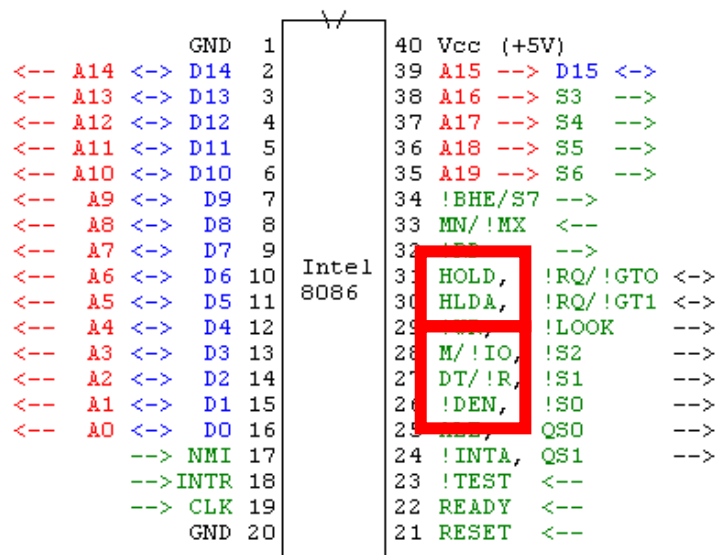
# Controlador de Acceso Directo a Memoria Intel 8237A

---

- Prioridades
  - Fija: Canal 0 = máxima, Canal 3 = mínima
  - Rotativa: El último canal servido es el de menor prioridad (evita monopolizar la atención)
- Generación de direcciones
  - Tiene multiplexada la parte alta del bus de direcciones
  - Los 8 bits más significativos de la dirección son guardados en un latch externo a través del bus de datos (señal AEN, Address Enable)
- Registros
  - Current Address, Current Word, Base Address y Base Word Count, Command, Mode, Request, Mask, Status, Temporary

# Acceso Directo a Memoria: Modos

- Control de transferencias en el bus del 8086
- Modo mínimo: señales
  - HOLD y HOLDA
  - M/!IO → Indica direccionamiento a MP o E/S
  - DT/!R → Indica Transmisión / Recepción de datos
  - !DEN → Habilita el buffer de datos 8286/7
  - MN/!MX → es "1" (Minimum Mode)



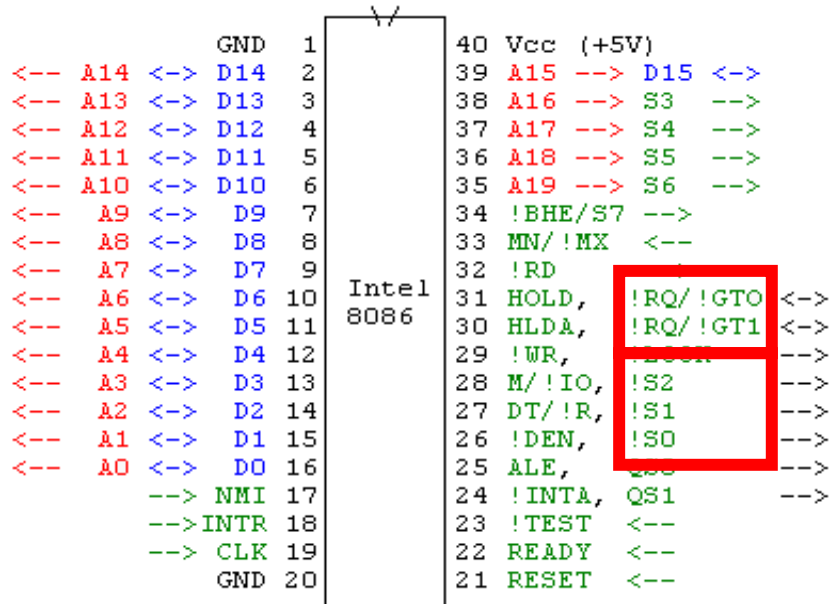
# Acceso Directo a Memoria: Modo máximo

## ■ Modo máximo

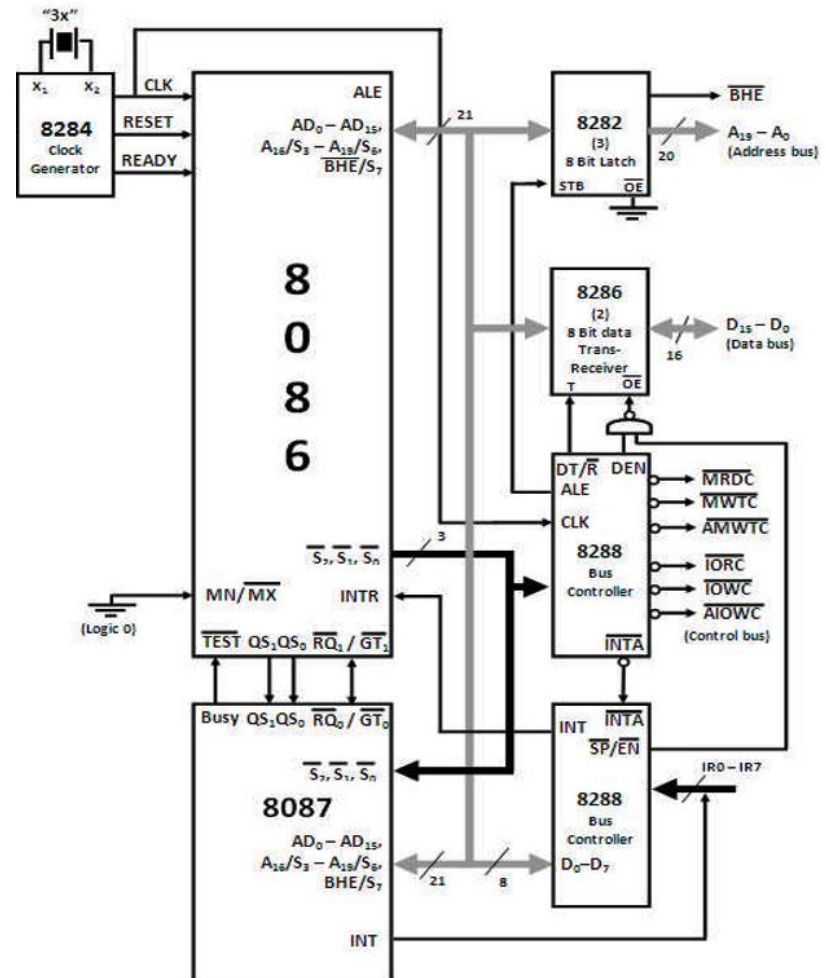
- Para conectar un coprocesador Intel 8087
- Se utilizan: controlador de buses Intel I8288
- RQ (ReQuest) y GT (GranT)
- MN!/MX = "0" (Maximum Mode)
- !RQ<sub>0</sub>!/GT<sub>0</sub>, !RQ<sub>1</sub>!/GT<sub>1</sub> → Utilizados para solicitar al la liberación de los buses al finalizar el ciclo de bus en curso
- El 8288 usa las señales !S0, !S1 y !S2
- Los COP iniciados con 11011<sub>b</sub> indicarán instrucciones para el coprocesador
- Cuando el 8087 ejecuta una instrucción, pone su pin "BUSY" en "1", cuando termina lo pasa a "0"
- El pin BUSY del 8087 está conectado con el TEST del 8086/8. El MAIN deberá ejecutar la instrucción WAIT para esperar el resultado del coprocesador.

S2	S1	S0	Significado
0	0	0	Ciclo INTA
0	0	1	Lectura E/S
0	1	0	Escritura E/S
0	1	1	Halt
1	0	0	Operand fetch
1	0	1	Lectura MP
1	1	0	Escritura MP
1	1	1	Estado pasivo

# Acceso Directo a Memoria: Modo máximo



- 8086 – Microprocessor
- 8087 – Math coprocessor
- 8282 – Octal latch
- 8284 – Clock generator and driver
- 8288 – Bus controller
- 8286 – Bus transceiver



# Acceso Directo a Memoria: Modo máximo

intel®

## 8087 MATH COPROCESSOR

- Adds Arithmetic, Trigonometric, Exponential, and Logarithmic instructions to the Standard 8086/8088 and 80186/80188 Instruction Set for All Data Types
- CPU/8087 Supports 7 Data Types: 16-, 32-, 64-Bit Integers, 32-, 64-, 80-Bit Floating Point, and 18-Digit BCD Operands
- Compatible with IEEE Floating Point Standard 754
- Available in 5 MHz (8087), 8 MHz (8087-2) and 10 MHz (8087-1): 8 MHz 80186/80188 System Operation Supported with the 8087-1
- Adds 8 x 80-Bit Individually Addressable Register Stack to the 8086/8088 and 80186/80188 Architecture
- 7 Built-In Exception Handling Functions
- MULTIBUS System Compatible Interface

intel®

## M8286/M8287 OCTAL BUS TRANSCEIVER

*Military*

- Data Bus Buffer Drive for Military M8086, M8088, M80186, M8085A, and M8048AH Processors
- High Output Drive Capability for Driving System Data Bus
- Fully Parallel 8-Bit Transceivers
- 3-State Outputs
- No Output Low Noise When Entering or Leaving High Impedance State
- Military Temperature Range: -55°C to +125°C (T<sub>C</sub>)

intel®

## M8282/M8283 OCTAL LATCH

*Military*

- Fully Parallel 8-Bit Data Register and Buffer
- Transparent During Active Strobe
- Supports M8085AH, M8048AH, M8086, M8088 and M80186
- High Output Drive Capability for Driving System Data Bus
- 3-State Outputs
- No Output Low Noise When Entering or Leaving High Impedance State
- Military Temperature Range: -55°C to +125°C (T<sub>C</sub>)

intel®

## M8288 BUS CONTROLLER FOR M8086, M8088, M80186 PROCESSORS

*Military*

- Bipolar Drive Capability
- Provides Advanced Commands
- Provides Wide Flexibility in System Configurations
- Military Temperature Range: -55°C to +125°C (T<sub>C</sub>)
- 3-State Command Output Drivers
- Configurable for Use with an I/O Bus
- Compatible with M8086, M8088, M8089 and M80186
- Facilitates Interface to One or Two Multi-Master Busses