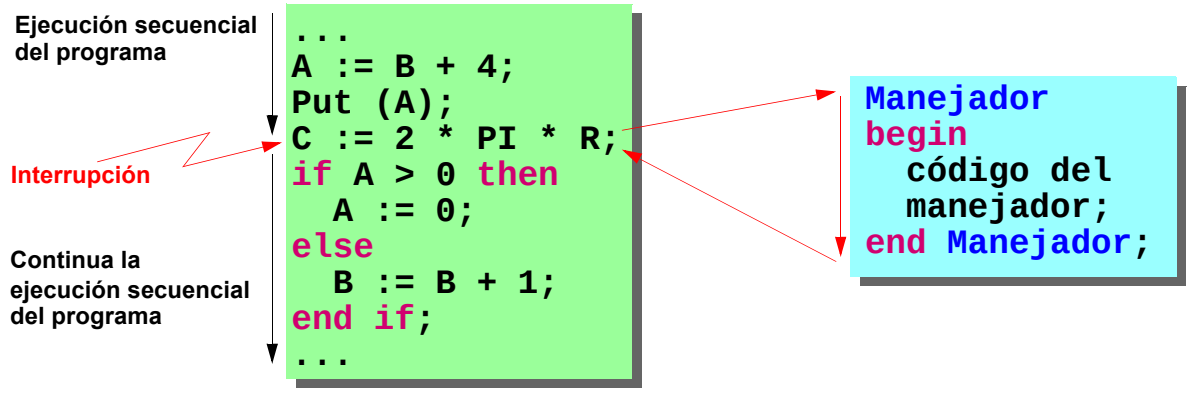
Instrumentación de Tiempo Real

UNIVERSIDAD DE CANTABRIA

Conceptos fundamentales

UNIVERSIDAD DE CANTABRIA

- ***Interrupción***: mecanismo mediante el cual es posible interrumpir la ejecución del programa ejecutado por la CPU
- Tras la interrupción el programa permanece suspendido mientras se ejecuta la *rutina de servicio de interrupción* (*ISR*) (también denominada *manejador de la interrupción*)



Conceptos fundamentales (cont.)

UNIVERSIDAD DE CANTABRIA

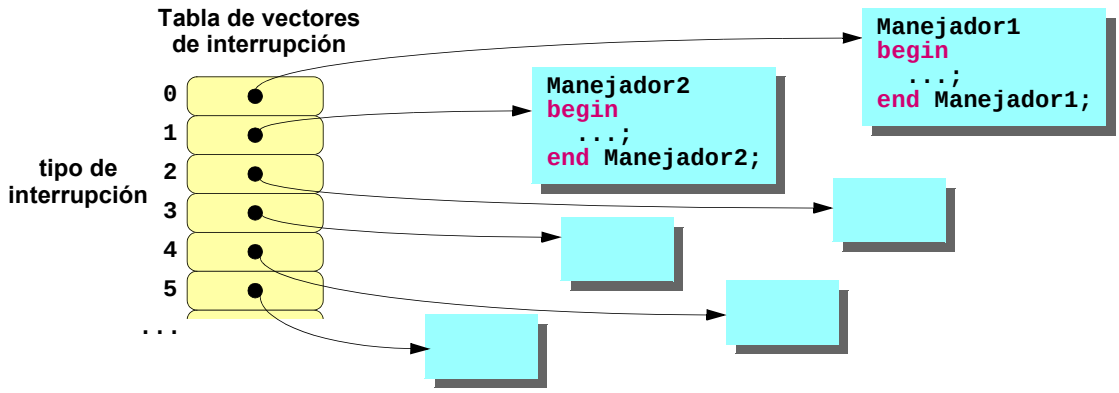
Las interrupciones pueden ser de origen interno o externo a la CPU:

- *excepción*: interrupción generada como consecuencia del resultado de una ejecución de una instrucción
  - P.e.: desbordamiento aritmético ("overflow"), división por cero, etc.
- *interrupción software*:
  - generadas expresamente por el programa con una instrucción ensamblador (`INT 4`)
- *interrupción hardware*: generada por un dispositivo
  - P.e.: expiración de un temporizador, nuevo dato disponible en un dispositivo de E/S, etc.

Conceptos fundamentales (cont.)

UNIVERSIDAD DE CANTABRIA

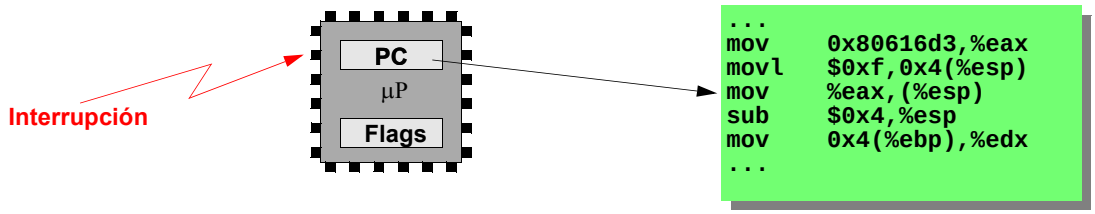
- Las distintas interrupciones que se pueden producir en un computador se identifican mediante un número (*tipo de la interrupción*)
- La *tabla de vectores de interrupción* establece el enlace entre cada tipo de interrupción y su ISR asociada



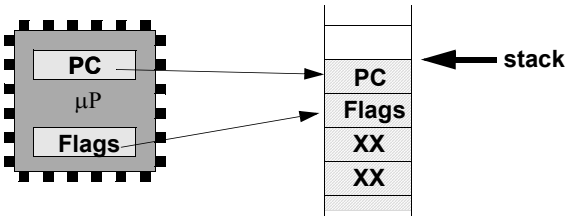
Aspectos hardware: Ciclo de atención a interrupción

UNIVERSIDAD DE CANTABRIA

1. Se genera una interrupción, el procesador termina la instrucción ensamblador que estaba ejecutando



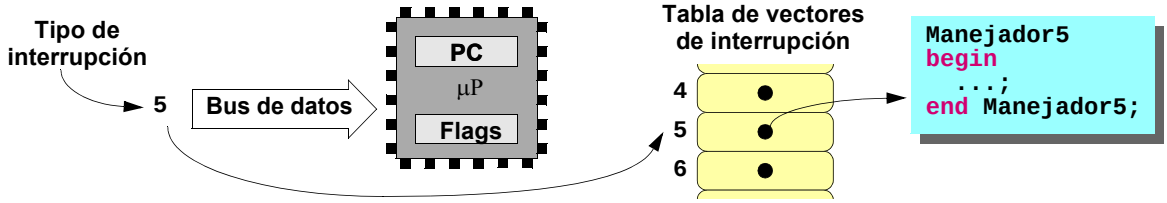
2. Se salva en el stack el estado del procesador (contador de programa y registro de estado)



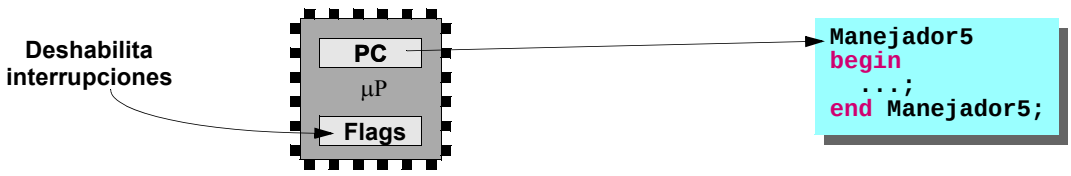
Aspectos hardware: Ciclo de atención a interrupción

UNIVERSIDAD DE CANTABRIA

3. La CPU lee el tipo de interrupción y obtiene la dirección de su ISR utilizando la tabla de vectores de interrupción



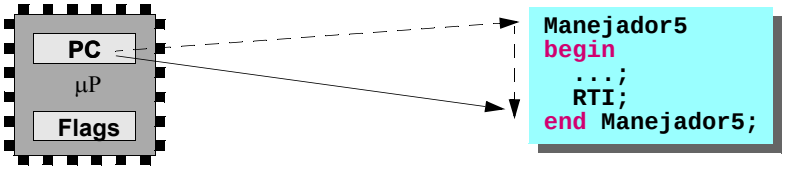
4. Se carga la dirección de comienzo de la ISR en el PC y se deshabilitan nuevas interrupciones



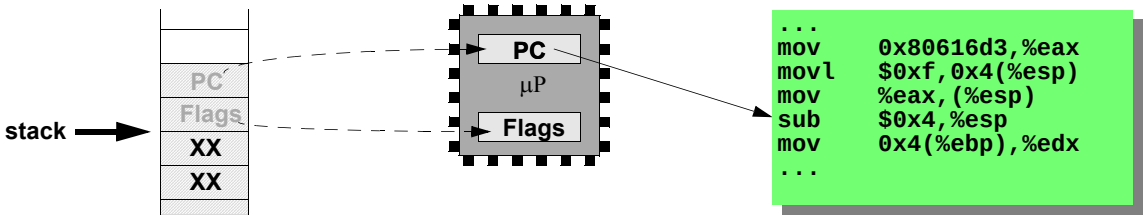
Aspectos hardware: Ciclo de atención a interrupción

UNIVERSIDAD DE CANTABRIA

5. Se ejecuta la ISR hasta llegar a la instrucción de retorno de interrupción (RTI)

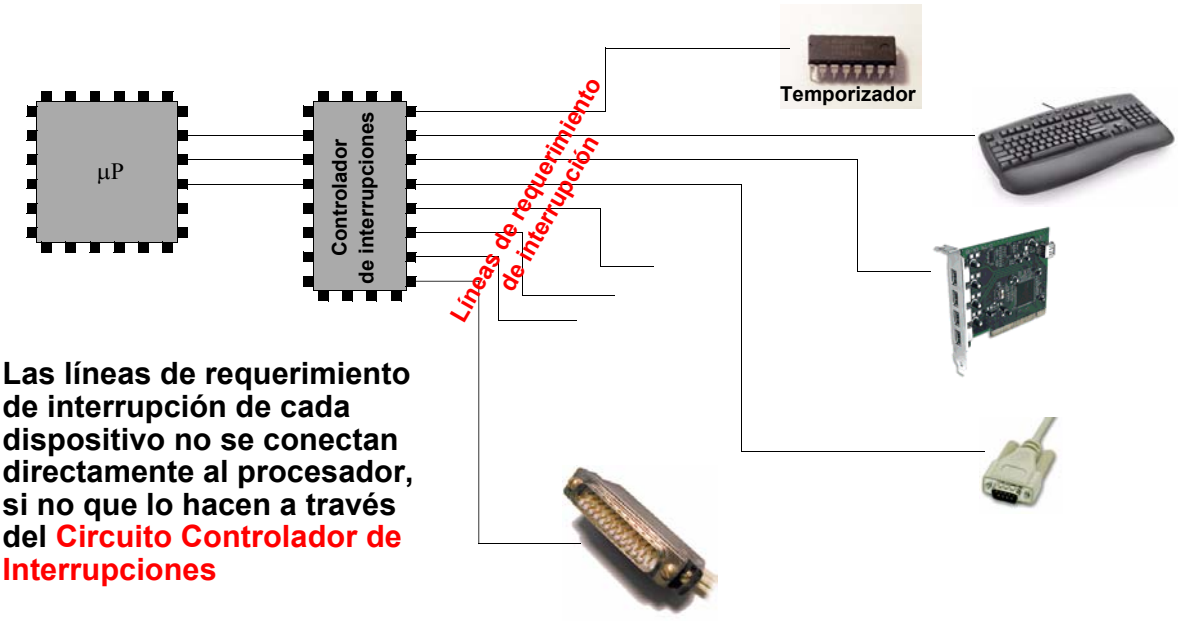


6. La instrucción RTI recupera del stack el estado original de la CPU, con lo que el procesador continua la ejecución del programa interrumpido



Aspectos hardware: Controlador de interrupciones

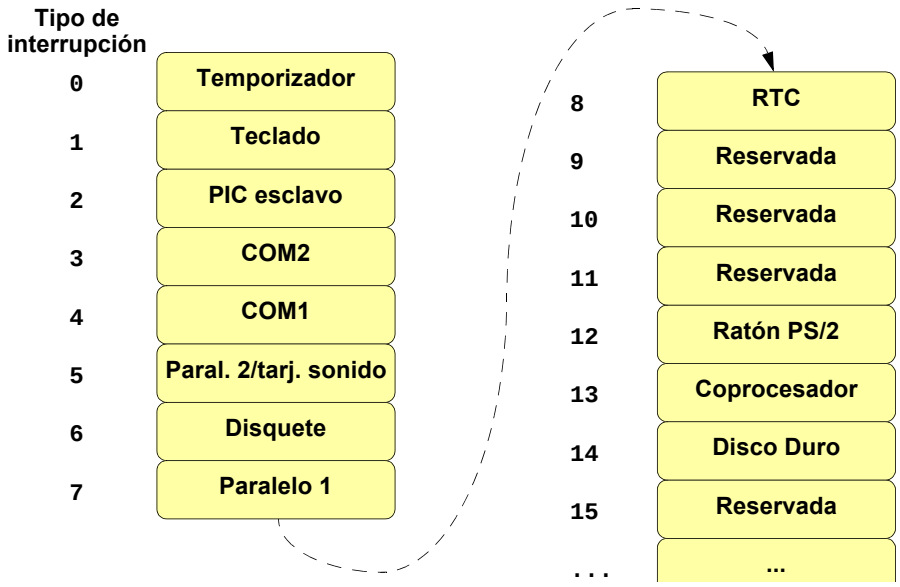
UNIVERSIDAD DE CANTABRIA



Las líneas de requerimiento de interrupción de cada dispositivo no se conectan directamente al procesador, si no que lo hacen a través del Circuito Controlador de Interrupciones

Aspectos hardware: Tabla de vectores de interrupción de un PC

UNIVERSIDAD DE CANTABRIA



Gestión de interrupciones en MaRTE OS

UNIVERSIDAD DE CANTABRIA

La gestión de interrupciones se realiza utilizando el paquete `MaRTE_Hardware_Interrupts`. Este paquete incluye:

- **constantes predefinidas para los 16 primeros tipos de interrupción:**

```
type Hardware_Interrupt is ...;
```

```
TIMER_INTERRUPT
CTLR2_INTERRUPT
SERIAL1_INTERRUPT
DISKETTE_INTERRUPT
RTC_INTERRUPT
RESERVED1_INTERRUPT
RESERVED3_INTERRUPT
FIXED_DISK_INTERRUPT
KEYBOARD_INTERRUPT
SERIAL2_INTERRUPT
PARALLEL2_INTERRUPT
PARALLEL1_INTERRUPT
SOFT_INTERRUPT
RESERVED2_INTERRUPT
COPROCESSOR_INTERRUPT
RESERVED4_INTERRUPT
```

Gestión de interrupciones en MaRTE OS (cont.)

UNIVERSIDAD DE CANTABRIA

- Definición del tipo de procedimiento manejador de interrupción:

```
type Interrupt_Handler_Function is
     access function (Area : in System.Address;
                      Intr : in Hardware_Interrupt)
                      return Handler_Return_Code;
```

- Posibles valores de retorno del procedimiento manejador:

```
POSIX_INTR_HANDLED_NOTIFY  --  para informar al
 --  sistema operativo de que el manejador ha atendido
 --  la interrupción

POSIX_INTR_NOT_HANDLED  --  para informar al sistema
 --  operativo de que el manejador NO ha atendido la
 --  interrupción
```


Gestión de interrupciones en MaRTE OS (cont.)

UNIVERSIDAD DE CANTABRIA

- Los manejadores de interrupción se ejecutan en el contexto del sistema operativo
- Una buena práctica de programación consiste en hacer los manejadores lo más cortos posibles
- Además, existen algunas limitaciones en las operaciones que puede realizar un manejador de interrupción
  - NO ejecutar operaciones potencialmente bloqueantes
  - como regla general NO realizar operaciones de entrada salida de texto (`Put`, `Get`):
    - sólo podrían utilizarse los procedimientos "Put" definidos en el paquete `Basic_Console_IO` (normalmente para depuración)

Gestión de interrupciones en MaRTE OS (cont.)

UNIVERSIDAD DE CANTABRIA

Asociar un manejador con un tipo de interrupción:

```
function Associate
   (Intr      : in Hardware_Interrupt;
    Handler   : in Interrupt_Handler_Function;
    Area      : in System.Address;
    Area_Size : in Size_T)
    return Int;
--  Retorna 0 si no se ha producido ningún error
--  Cuando Area y Area_Size no se usan (lo normal):
--      Area      => System.Null_Address
--      Area_Size => 0

function Disassociate
   (Intr    : in Hardware_Interrupt;
    Handler : in Interrupt_Handler_Function)
    return Int;
--  Retorna 0 si no se ha producido ningún error
```


Gestión de interrupciones en MaRTE OS (cont.)

UNIVERSIDAD DE CANTABRIA

Deshabilitar ("lock") y habilitar ("unlock") una fuente de interrupción en el controlador de interrupciones:

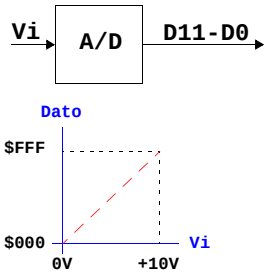
```
function Lock (Intr : in Hardware_Interrupt)
               return Int;
--  Retorna 0 si no se ha producido ningún error


function Unlock (Intr : in Hardware_Interrupt)
                 return Int;
--  Retorna 0 si no se ha producido ningún error
```

Ejemplo: Tarjeta A/D operando con interrupciones

UNIVERSIDAD DE CANTABRIA

LoDato	($240)	D3	D2	D1	D0	X	X	X	X
HiDato	($241)	D11	D10	D9	D8	D7	D6	D5	D4
Control	($242)	X	X	X	X	X	X	X	IH
Estado	($246)	B	X	X	X	X	X	X	X



- El bit `IH` del registro de control sirve para habilitar (1) o deshabilitar (0) las interrupciones por fin de conversión
  - la interrupción se deshabilita tras cada dato convertido, por lo que habrá que rehabilitarla en el manejador
- La conversión de un nuevo dato se inicia escribiendo cualquier valor en el registro `Estado`

Ejemplo: Tarjeta A/D operando con interrupciones (cont.)

UNIVERSIDAD DE CANTABRIA

```
with MaRTE_OS;
with Basic_Integer_Types; use Basic_Integer_Types;
with IO_Interface; use IO_Interface;
with MaRTE_Hardware_Interrupts;
use MaRTE_Hardware_Interrupts;
with Text_IO; use Text_IO;
with System;

procedure Lee_AD_Con_Interrupciones is

   --  Direcciones de los registros de E/S del
   --  convertidor A/D
   BASE_AD     : constant IO_Port := 16#240#;
   Reg_LoDato  : constant IO_Port := BASE_AD + 0;
   Reg_HiDato  : constant IO_Port := BASE_AD + 1;
   Reg_CNTR    : constant IO_Port := BASE_AD + 2;
   Reg_Estado  : constant IO_Port := BASE_AD + 6;
```

Ejemplo: Tarjeta A/D operando con interrupciones (cont.)

UNIVERSIDAD DE CANTABRIA

```
--  Variables globales utilizadas por el manejador
Datos : array (1 .. 10) of Unsigned_16;
pragma Volatile (Datos);

N_Leidos : Integer := 0;
pragma Volatile (N_Leidos);

Fin_Lectura_AD : Boolean := False;
pragma Volatile (Fin_Lectura_AD);
```


Ejemplo: Tarjeta A/D operando con interrupciones (cont.)

UNIVERSIDAD DE CANTABRIA

```
function Manejador (Area : in System.Address;
                    Intr : in Hardware_Interrupt)
                   return Handler_Return_Code is
   Hi, Lo : Unsigned_8;
begin
   N_Leidos := N_Leidos + 1;
   --  Lee el código obtenido por el A/D
   Hi := Inb (Reg_Hi_Dato); Lo := Inb (Reg_Lo_Dato);
   Datos (N_Leidos) := Unsigned_16 (Hi) * 16
                     + Unsigned_16 (Lo) / 16;
   Fin_Lectura_AD := N_Leidos = Datos'Last;
   if not Fin_Lectura_AD then
      -- Rehabilita inte. y comienza nueva conversión
      Outb (Reg_CNTR, 16#01#); Outb (Reg_Estado, 0);
   end if;
   return POSIX_INTR_HANDLED_NOTIFY;
end Manejador;
```

Ejemplo: Tarjeta A/D operando con interrupciones (cont.)

UNIVERSIDAD DE CANTABRIA

```
begin
 --  Instala manejador de la interrupción
 if MaRTE_Hardware_Interrupts.Associate
                  (PARALLEL2_INTERRUPT,
                   Manejador'Unrestricted_Access,
                   System.Null_Address, 0) /= 0 then
    Put_Line ("Error: instalando manejador");
 end if;
 --  Después de acabar la configuración se habilita
 --  la interrupción en el controlador
 if MaRTE_Hardware_Interrupts.Unlock
                       (PARALLEL2_INTERRUPT) /= 0 then
    Put_Line ("Error: habilitando interrupción");
 end if;
```


Ejemplo: Tarjeta A/D operando con interrupciones (cont.)

UNIVERSIDAD DE CANTABRIA

```
   --  Habilita interrupciones en el A/D y comienza la
   --  primera conversión
   Outb (Reg_CNTR, 16#01#);
   Outb (Reg_Estado, 0);

   loop
      exit when Fin_Lectura_AD;  --  dejar el lazo?
      el programa principal puede hacer cosas
      mientras, de forma concurrente, el sistema va
      llamando al manejador de interrupción;
   end loop;

end Lee_AD_Con_Interrupciones;
```

Variables "Volátiles"

UNIVERSIDAD DE CANTABRIA

Cualquier variable compartida entre el manejador de la interrupción y la aplicación debe ser marcada como *volátil*

- indica al compilador que la variable puede cambiar de forma inesperada (una interrupción puede ocurrir en cualquier momento)
- el compilador no deberá realizar ninguna optimización sobre esa variable
  - cada lectura y escritura sobre ella se realizará directamente sobre la memoria
- en Ada, la forma de indicar que una variable es volátil es:

```
Nuevo_Dato : Boolean;
pragma Volatile (Nuevo_Dato);
```


Variables "Volátiles" (cont.)

UNIVERSIDAD DE CANTABRIA

Ejemplo de uso del pragma "Volatile":

```
Nuevo_Dato : Boolean;
pragma Volatile (Nuevo_Dato);

function Manejador (Area : in System.Address;
                    Intr : in Hardware_Interrupt)
                   return Handler_Return_Code is
begin
  Nuevo_Dato := True;
  return POSIX_INTR_HANDLED_NOTIFY;
end Manejador;

begin
  ...
  loop
    exit when Nuevo_Dato; -- sale cuando hay nuevo dato
  end loop;
  ...
```

Acceso a datos compartidos con un manejador de interrupción

UNIVERSIDAD DE CANTABRIA

El acceso a datos compartidos "grandes" debe realizarse de forma mutuamente exclusiva entre el manejador y la aplicación

- **"grande": su lectura/escritura no es *atómica* (necesita más de una instrucción ensamblador)**

La exclusión mutua se consigue deshabilitando temporalmente la interrupción:

```
MaRTE_Hardware_Interrupts.Lock (LA_INTERRUPCIÓN);
Accede a los datos compartidos
MaRTE_Hardware_Interrupts.Unlock (LA_INTERRUPCIÓN);
```


Acceso a datos compartidos con un manejador de interrupción (cont.)

UNIVERSIDAD DE CANTABRIA

```
...
type Dato is record
  Codigo_Error : Unsigned_8;
  Byte         : Unsigned_8;
end record;

Dato_Leido : Dato;
pragma Volatile (Dato_Leido);

function Manejador (Area : in System.Address;
                    Intr : in Hardware_Interrupt)
                    return Handler_Return_Code is
begin
  Dato_Leido.Codigo_Error := Inb (REG_ERROR);
  Dato_Leido.Byte := Inb (REG_DATO);
  return POSIX_INTR_HANDLED_NOTIFY;
end Manejador;
```

Acceso a datos compartidos con un manejador de interrupción (cont.)

UNIVERSIDAD DE CANTABRIA

```
--  Cuerpo del procedimiento principal
begin
   ...
   --  Deshabilita la interrupción
   if Lock (LA_INTERRUPCIÓN) /= 0 then
      Put_Line ("Error: deshabilitando interrupción");
   end if;
   --  Accede de forma segura al dato compartido
   Error := Dato_Leido.Codigo_Error;
   Byte  := Dato_Leido.Byte;
   --  Habilita la interrupción
   if Unlock (LA_INTERRUPCIÓN) /= 0 then
      Put_Line ("Error: habilitando interrupción");
   end if;
   --  Utiliza "Byte" y "Error"
   ...
end Proc_Principal;
```


Controlador de Interrupciones Programable 8259A (PIC)

UNIVERSIDAD DE CANTABRIA

El circuito 8259A (PIC) es el controlador de interrupciones utilizado en los PCs

- Permite gestionar 8 líneas de requerimiento de interrupción independientes

Configurable:

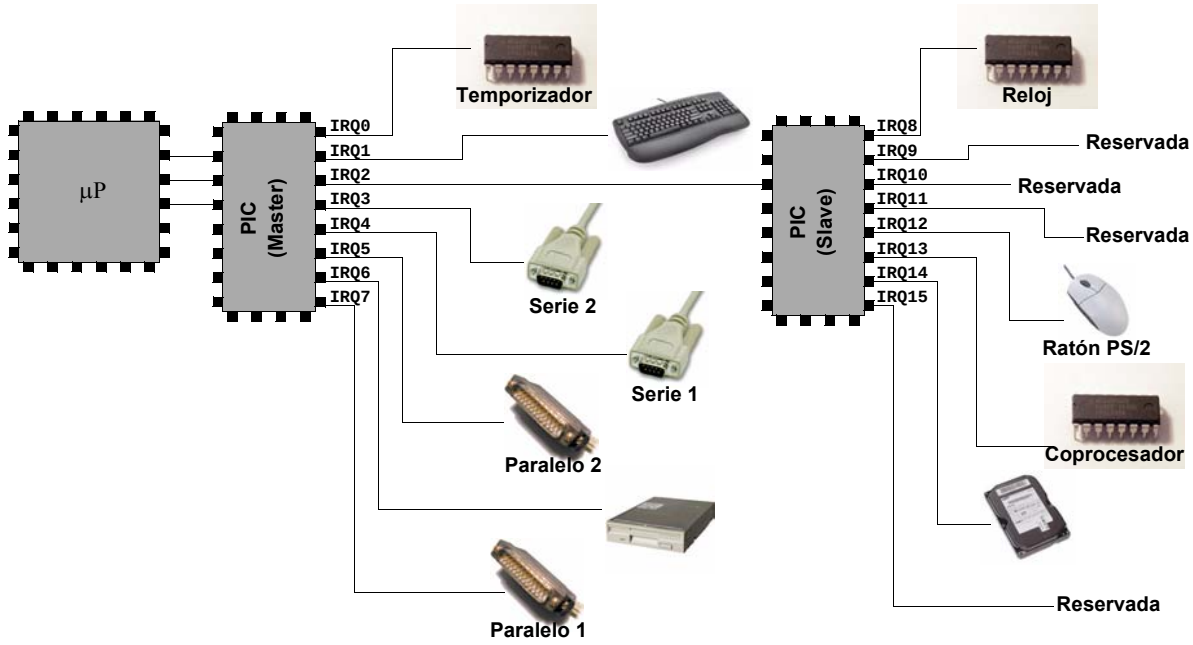
- prioridad de interrupciones rotativa o fija
- fin de interrupción automático o explícito

Permite habilitar y deshabilitar cada una de las 8 líneas de interrupción de forma independiente

Pueden conectarse varios PIC en cascada para permitir gestionar más de 8 líneas de requerimiento de interrupción

Configuración de las interrupciones hardware en un PC estándar

UNIVERSIDAD DE CANTABRIA



Controlador de Interrupciones Programable 8259A (PIC) (cont.)

UNIVERSIDAD DE CANTABRIA

Nosotros no tenemos que preocuparnos por la programación del PIC:

- **MaRTE OS se encarga de realizar la configuración inicial del PIC antes de ejecutar la aplicación del usuario**
- **El comando para rehabilitar las interrupciones es automáticamente incluido por MaRTE OS en cada manejador de interrupción instalado por el usuario**
- **Las funciones `MaRTE_Hardware_Interrupts.Lock` y `MaRTE_Hardware_Interrupts.Unlock` programan los PIC para deshabilitar una interrupción de forma individual**