Introducción a la Entrada/Salida

Organización de entrada/salida

La familia de procesadores 80x86, presente en el IBM PC, utiliza la *arquitectura Von Neumann*, que puede verse en la figura 1. El denominado **bus del sistema** conecta las diferentes partes de una máquina von Neumann, y en la familia 80x86 se diferencian 3 clases de buses:

1. **Bus de datos**, de 8, 16, 32 ó 64 bits dependiendo del modelo (64 bits para los Pentiums de última generación). El número de bits se usa, en general, para determinar el tamaño (*size*) del procesador.
2. **Bus de direcciones**, para poder conectar la CPU con la memoria y con los dispositivos de entrada/salida.
3. **Bus de control**, para enviar señales que determinan cómo se comunica la CPU con el resto del sistema (por ejemplo, las líneas de lectura (*read*) y escritura (*write*) especifican qué es lo que se está haciendo en la memoria).

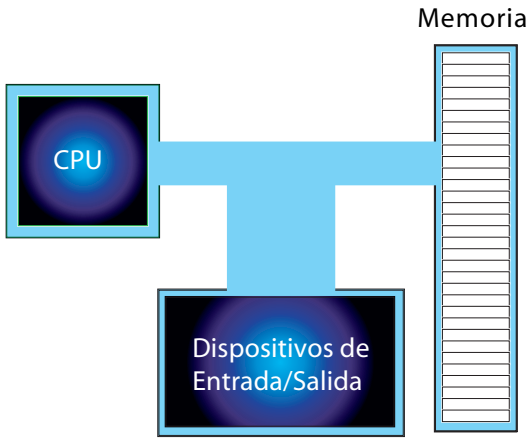

Figura 1: Arquitectura típica de una máquina Von Neumann

Por tanto, los dispositivos de entrada/salida son una de las partes fundamentales de la arquitectura del computador y su objetivo es la eficiencia en la gestión de las operaciones de entrada/salida, descargando a la CPU de tanto trabajo como sea posible. Estos dispositivos tienen velocidades muy variadas:

- Dispositivos lentos (como el ratón, el teclado, el joystick...)
- Dispositivos de velocidad media (como una impresora)
- Dispositivos rápidos (como un disco duro, una tarjeta de red...)

Por tanto, un dispositivo concreto no puede aceptar datos enviados a una tasa arbitrariamente alta (por ejemplo, una impresora no puede imprimir los millones de caracteres por segundo que sería capaz de enviarle un Pentium IV). Por ello, ha de haber alguna forma de *coordinar* el envío de datos entre la CPU y los periféricos. Esa es la misión de los circuitos de interfaz que aparecen en la figura 2.

Existen 2 formas básicas de conectar estos circuitos de interfaz, dependiendo del tipo de procesador en que se base la arquitectura. Ambos tipos de acceso requieren que la CPU realice el movimiento de los datos entre el periférico y la memoria principal:

1. **Conexión mapeada en memoria** (*memory–mapped I/O*), que usa "direcciones especiales" en el espacio normal de direcciones. En este caso el circuito de interfaz se conecta en el computador como si fuera memoria. Presentan este tipo de entrada/salida, por ejemplo, aquellas arquitecturas basadas en el M68000 de Motorola. Cualquier instrucción que acceda a la memoria (como "mov") puede acceder a un puerto I/O mapeado en memoria.
2. **Conexión mediante puertos especiales de entrada/salida** (*I/O–mapped I/O*), que usa "instrucciones especiales" de entrada/salida [1] y un espacio de direcciones específico. Este es el caso de las CPUs 80x86, y por tanto el que nos interesa para este proyecto.

La familia de procesadores 80x86 proporciona, por tanto, dos espacios de direcciones:

- Para memoria
- Para dispositivos de E/S

El bus de direcciones (*address bus*) varía de tamaño según el procesador de la familia que se emplee (puede ser de 20, 24 ó 32 bits), pero para la entrada/salida es siempre de **16 bits**. Esto permite al microprocesador direccionar hasta 65536 diferentes localizaciones especiales de entrada/salida, lo que es más que suficiente para la mayoría de los dispositivos, aunque muchas veces un dispositivo requiera más de una dirección de entrada/salida (por ejemplo, en el caso del ratón PS/2 veremos que habrá que acceder a las direcciones 60h y 64h; en el caso del puerto de juegos, suele ser la 201h). Hay 2 espacios de direcciones, pero **un solo bus** de direcciones y son las líneas de control las que deciden a qué espacio estamos accediendo en cada momento. De esta manera, el direccionamiento de entrada/salida (*I/O addressing*) se comporta exactamente igual que el direccionamiento de memoria (*memory addressing*). La memoria y los dispositivos I/O comparten el bus de datos y los 16 primeros bits del bus de direcciones.

Resumiendo, existen dos consideraciones básicas que debemos hacer en el subsistema de entrada/salida en un IBM PC:

[1] La familia Intel 80x86 usa las instrucciones en ensamblador **in** y **out** para acceder a los puertos de entrada/salida. Estas instrucciones funcionan igual que **mov** excepto por que ponen la dirección en el bus de direcciones de entrada/salida en vez de en el bus de direcciones de memoria.

1. Las CPUs 80x86 requieren "instrucciones especiales" para acceder a los dispositivos de entrada/salida.

2. No se pueden direccionar más de 65536 direcciones de entrada/salida. Esto es un problema porque algunos dispositivos requieren más direcciones, como por ejemplo una tarjeta gráfica VGA típica, que necesita más de 128000. Afortunadamente, existen medios, utilizando la circuitería adecuada, para mapear en memoria esos dispositivos.

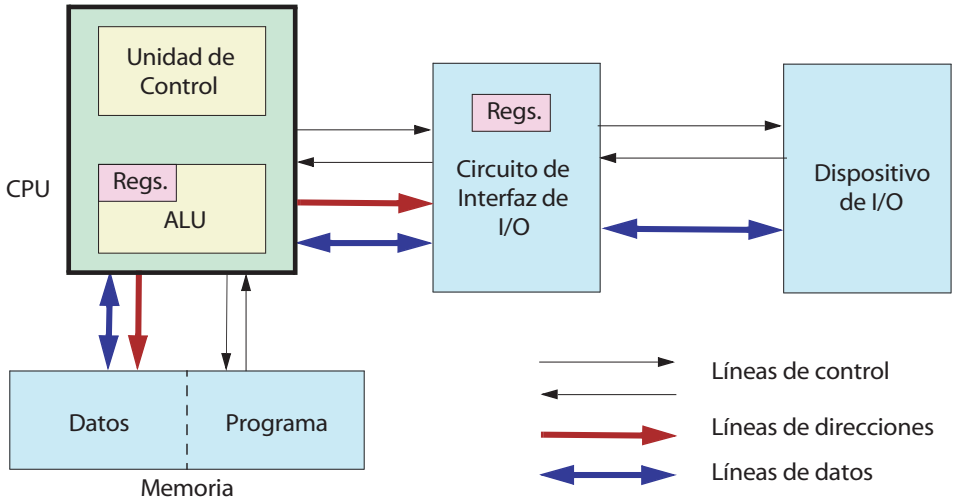

Figura 2: Interfaces de Entrada/Salida

Un *puerto I/O* tiene una apariencia de celda de memoria para el computador, pero tiene conexiones con el mundo exterior. Los puertos de entrada/salida se clasifican habitualmente en tres tipos diferentes:

- Puertos de *sólo lectura* (*read–only* o *input port*)
- Puertos de *sólo escritura* (*write–only* o *output port*)
- Puertos de *lectura–escritura* (*read–write* o *input–output*)

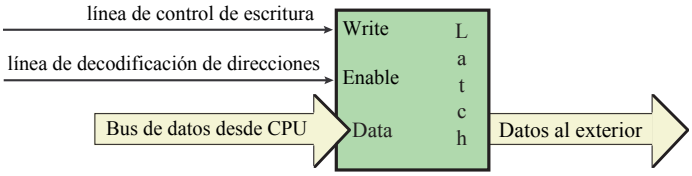

Figura 3: Puerto de *sólo–escritura* con un latch

Por regla general, los puertos de entrada/salida utilizan un latch como elemento de memoria. Por ejemplo, en el caso de un puerto de sólo escritura el latch captura los datos escritos

en él por la CPU y los hace disponibles en un conjunto de cables externos a la CPU y al sistema de memoria. En este tipo de puertos la CPU no puede leer el dato que ha escrito en el latch ya que para que éste funcione deben estar activas tanto la línea de direcciones como la de control, y en el caso de querer realizar una lectura la línea "control de escritura" (*write control*) no estaría activa. La figura 3 muestra una posible implementación de un puerto de sólo escritura con un latch.

Si el puerto es de lectura–escritura, la CPU puede comprobar que el dato se ha escrito correctamente leyendo del puerto. Un puerto de entrada/salida se comportaría como se describe en la figura 4 (la parte inferior de la figura correspondería a un puerto de sólo lectura (input port)). Los datos que se escriban en un puerto de sólo lectura son ignorados por el sistema.

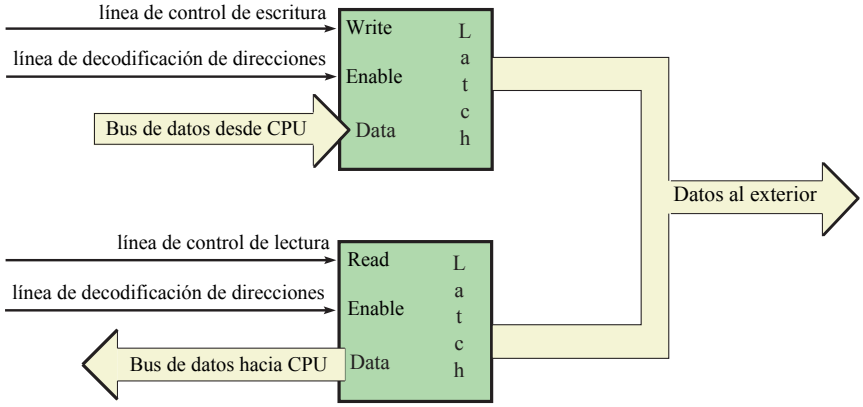

Figura 4: Puerto de *lectura–escritura*

Formas de realizar la entrada/salida

Ya hemos visto que para coordinar el envío de datos entre la CPU y los dispositivos se usan circuitos de interfaz especiales. Existen básicamente tres formas de realizar la gestión de entrada/salida en un dispositivo, es decir, tres formas en que estos circuitos de interfaz pueden operar:

1. Entrada/salida por **encuesta** (también llamada por consulta o programada, en inglés *"polling"*).
2. Entrada/salida por **interrupciones**.
3. Entrada/salida por **acceso directo a memoria**.

Entrada/salida por consulta

El polling, utilizado en las primeras computadoras personales (como Apple II), consiste en que la CPU sondea periódicamente al dispositivo para ver cuál es su estado. Ese sondeo se puede hacer, por ejemplo, leyendo de una dirección de entrada/salida correspondiente a uno

o varios registros de estado del dispositivo. Los bits de esos registros de estado nos dirán cuál es la situación concreta del dispositivo (por ejemplo, si se trata de una impresora, podemos saber si está lista para recibir nuevos caracteres; si se trata de un teclado, podemos saber si el usuario ha presionado una tecla y el carácter aún no se ha leído, etc). Otras veces no hay tal registro de estado, y simplemente se lee lo que haya en la dirección I/O desde programa, decidiendo luego qué hacer con el dato leído. Este es el caso, por ejemplo, de un joystick analógico conectado al puerto de juegos del PC.

Esta forma de entrada/salida es sencilla, pero claramente ineficiente. Por ejemplo, si un usuario tarda 10 segundos en mover el ratón, se habrán realizado miles de encuestas al dispositivo sin detectar un nuevo evento, con la consecuente pérdida de tiempo para realizar otras tareas en la CPU. Por otra parte, el ritmo de transferencia de datos está limitado por la velocidad de la CPU, ya que no podremos encuestar al dispositivo con una frecuencia arbitrariamente alta. Por tanto, esta forma de entrada/salida debe evitarse en lo posible. Sin embargo, en algunas ocasiones no quedará otra opción, ya que el dispositivo en cuestión no genera interrupciones (como es el caso del joystick que veremos en este proyecto).

Entrada/Salida por interrupciones

En el caso de la entrada/salida por interrupciones, es el dispositivo quien establece el momento en que se realiza la transferencia de los datos, avisando a la CPU de que ha ocurrido un evento (por ejemplo, que el usuario haya presionado una tecla). En este punto, debemos aclarar que en la familia 80x86 existen tres tipos de interrupciones, que a veces producen confusión por la nomenclatura empleada en diversos textos:

1. Las *traps* o interrupciones software son interrupciones invocadas por el usuario desde programa. En este caso, la CPU pasa a ejecutar el manejador de trap asociado (su rutina de atención a la interrupción o ISR[2]).

2. Las *excepciones* son traps generadas automáticamente en respuesta a alguna condición excepcional producida al intentar ejecutar una instrucción: división por cero, código de operación ilegal... También en este caso se ejecuta la ISR asociada, decidiendo, en su caso, qué hacer con la situación anómala.

3. Las *interrupciones hardware*, a las que llamaremos simplemente "interrupciones", se basan en un evento hardware externo a la CPU y no relacionado con la secuencia de instrucciones que se esté ejecutando en ese momento. Son las que un ingeniero electrónico más intuitivamente relacionaría con el término "interrupción", y con las que vamos a tratar en este proyecto.

Para cada tipo de interrupción, por tanto, se puede instalar una *rutina de atención o servicio de interrupción*. Cuando la CPU recibe notificación de la interrupción detiene el programa en ejecución, ejecuta la ISR (es decir, sirve al dispositivo haciendo que cese su petición de interrupción (si es necesario, se accede al controlador de interrupciones para hacer lo mismo)) y finalmente devuelve el control al programa, restaurando su estado anterior. Por tanto, una condición importante a cumplir es que una ISR debe *preservar el estado del*

[2]ISR: Interrupt Service Routine. Procedimiento creado específicamente para manejar una trap, excepción o interrupción.

procesador (el contenido de todos sus registros) para poder volver al estado anterior a la llamada.

Existe una diferencia entre las interrupciones hardware y las demás: cuando se entra en una ISR de una interrupción hardware, el procesador 80x86 *deshabilita* posteriores interrupciones hardware poniendo a 0 el *flag de interrupción*. Esto no ocurre con las traps ni con las excepciones [3]. Por tanto, si queremos inhabilitar las interrupciones en medio de una ISR de trap o excepción, hay que hacerlo explícitamente, con la instrucción "cli" en ensamblador o con los medios que proporcione el sistema operativo. De la misma forma, para permitir nuevas interrupciones hardware en medio de una ISR hardware, hay que usar una instrucción "sti". Las fuentes primarias de *interrupción* en el PC son variadas:

- timer chip del PC
- teclado
- puertos series y paralelos
- discos
- reloj de tiempo real CMOS
- ratón
- tarjetas de sonido
- otros dispositivos

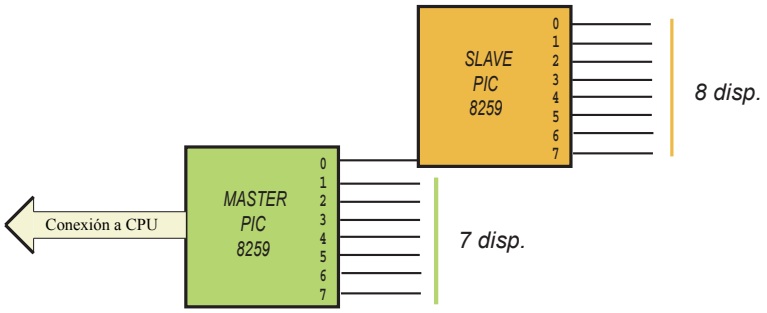

Figura 5: Esquema de conexión de los PIC

En la tabla 1 pueden consultarse las fuentes de interrupción más frecuentes en el PC, junto con su número de interrupción asociado. Todos estos dispositivos se conectan al *controlador programable de interrupciones* o **PIC** (*Programmable Interrupt Controller*) Intel 8259A, que establece prioridad entre las interrupciones y se conecta con la CPU. Este controlador de interrupciones (que se denomina genéricamente "8259") acepta interrupciones de hasta 8 dispositivos. Si cualquier dispositivo pide servicio, el 8259 se "conectará" con la CPU y le pasará un vector de interrupción programable. En un principio, el PC IBM original sólo soportaba ocho interrupciones diferentes ya que usaba un solo chip 8259. Más tarde IBM (y todos los fabricantes de PCs) añadieron un segundo PIC en el PC/AT y posteriores máquinas. De hecho, podrían encadenarse hasta nueve chips del 8259 para soportar un total de 64

[3] Nótese que el flag de interrupción del 80x86 sólo afecta a las interrupciones hardware, y no evita que haya una trap o una excepción.

dispositivos que generen interrupciones. Lo normal, sin embargo, es tener solamente dos, tal como se muestra en la figura 5, y soportar un total de 15 interrupciones.

Entrada al 8259	**INT 80x86**	**Dispositivo**
IRQ 0	8	Chip temporizador del sistema (timer chip)
IRQ 1	9	Teclado
IRQ 2	0Ah	Conexión con el PIC 2(IRQ's 8–15)
IRQ 3	0Bh	Puerto serie 2 (COM2/COM4)
IRQ 4	0Ch	Puerto serie 1 (COM1/COM3)
IRQ 5	0Dh	Puerto paralelo 2 en AT, reservado en sistemas PS/2
IRQ 6	0Eh	Controladora de floppy
IRQ 7	0Fh	Puerto paralelo 1
IRQ 8/0	70h	Reloj de tiempo real
IRQ 9/1	71h	Redirección de la IRQ 2
IRQ 10/2	72h	Reservado
IRQ 11/3	73h	Reservado
IRQ 12/4	74h	Reservado en AT, **Ratón PS/2** en sistemas PS/2
IRQ 13/5	75h	Interrupción de la FPU (Floating Point Unit)
IRQ 14/6	76h	Controladora de disco duro
IRQ 15/7	77h	Reservado

Tabla 1: Fuentes de interrupción en el PC. Entradas del PIC 8259

Sin entrar en demasiados detalles sobre la programación del PIC, que puede resultar compleja, sí diremos que tiene un **registro de máscara de interrupción** (mask register). Este registro de 8 bits permite habilitar o inhabilitar las interrupciones de los dispositivos del sistema (un "0" habilita la interrupción del dispositivo y un "1" la inhabilita), como puede verse en la figura 6. Cuando ocurre una cierta interrupción, el 8259 **enmascara** las posteriores interrupciones de ese dispositivo hasta que recibe "fin de interrupción" de la ISR.

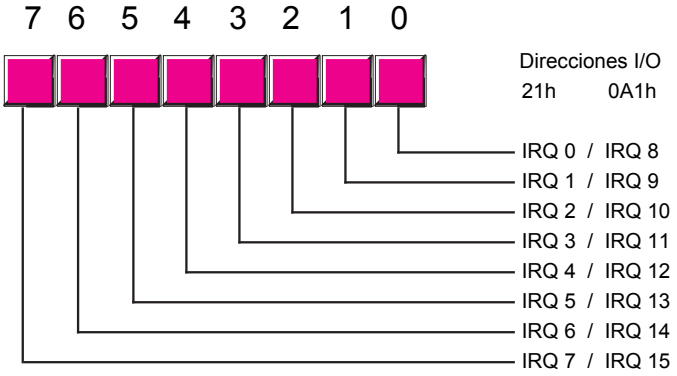

Figura 6: Registro de máscara de interrupción del 8259

El PIC proporciona varios esquemas de prioridad, pero el BIOS lo inicializa para usar prioridad fija. En este caso, el dispositivo con IRQ 0 (el timer) tiene la prioridad más alta

y el IRQ 7 la más baja. Si tenemos una tarea que queremos que sea de máxima prioridad (como por ejemplo el control de un reactor nuclear) seguramente usaríamos la *interrupción no enmascarable* (int 2), no conectada con el 8259, que no puede desactivarse mediante software y es de mayor prioridad que cualquiera que venga del PIC. Esto es así porque en realidad los chips de la familia 80x86 presentan dos posibles entradas de interrupción. La primera es la *interrupción enmascarable*, a la que se conecta el 8259, que como su nombre indica se puede enmascarar o desenmascarar con las instrucciones *cli* y *sti*. La segunda es para la *interrupción no enmascarable*, que siempre está activada por defecto.

Entrada/Salida por acceso directo a memoria (DMA)

Tanto los subsistemas de entrada/salida mapeada en memoria como los que usan puertos de entrada/salida necesitan que sea la CPU la que realice la transferencia de los datos entre el dispositivo y la memoria. Para algunos dispositivos de alta velocidad, esta forma de procesar los datos puede resultar demasiado lenta. En el caso del DMA, el dispositivo puede operar directamente sobre la memoria, siempre que la CPU le haya concedido el permiso para hacerlo. De esta manera, sólo se avisa a la CPU al comienzo o al final de una operación sobre memoria y se consigue una tasa de transferencia de datos superior a la de los otros métodos. Además esto a menudo permite que las operaciones de entrada/salida se realicen en paralelo a otras operaciones de la CPU, aumentando así el rendimiento global del sistema. Sin embargo, nótese que no es posible que la CPU y el dispositivo utilicen a la vez el bus de direcciones y de datos, por lo que el procesamiento paralelo se dará sólo cuando la CPU esté operando sobre datos contenidos en una memoria cache.