Introducción a los DSP. Conceptos básicos

1

CONTENIDO DEL CAPÍTULO

1.1. Reseña histórica.

1.2. Descripción de un sistema microprocesador.

1.3. Dispositivos provistos de CPU: el DSP.

1.4. Estructura interna básica de un DSP.

1.5. Principales aplicaciones y fabricantes de DSP.

1.6. Ejercicios propuestos.

1.1. RESEÑA HISTÓRICA

La instrumentación y los modernos sistemas de control y telecomunicación son versátiles, flexibles y de bajo coste porque en la actualidad se dispone de los sistemas digitales denominados sistemas microprocesadores. Estos sistemas se caracterizan por la capacidad de procesar información mediante la ejecución de programas (conjunto de instrucciones que se encuentran almacenadas en dispositivos de memoria de tipo RAM y ROM) gracias a que van provistos de un módulo denominado CPU[1], que es el encargado de interpretar secuencialmente las instrucciones y de realizar la tarea programada. Esta capacidad les permite ser sistemas de propósito general que pueden ser utilizados en infinidad de aplicaciones con sólo cambiar el programa que han de ejecutar.

[1] Siglas que definen al dispositivo digital denominado unidad central de proceso y que proceden del inglés *Central Process Unit*.



Figura 1.1. *Gottfried Wilhelm von Leibnitz.*

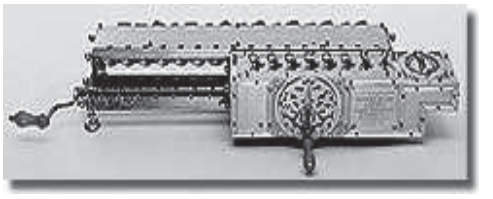


Figura 1.2. *Calculadora de Leibnitz.*

La historia de los sistemas capaces de procesar información se puede decir que comienza en 1694 con la primera calculadora mecánica diseñada por el matemático, físico, teólogo, filósofo, historiador e inventor alemán Leibnitz (Fig. 1.1). Esta calculadora, la primera que introdujo la posibilidad de acumular el resultado dotando de memoria al mecanismo, era capaz de sumar, restar, multiplicar, dividir e incluso realizar raíces cuadradas, demostrándose, ya en aquella época, la ventaja del sistema binario sobre el decimal en la implementación de mecanismos capaces de procesar información (Fig. 1.2).

El primer ordenador digital fue propuesto por el matemático inglés Charles Babbage, en 1821 (Fig. 1.3). Se trataba de un ingenio capaz de compilar tablas matemáticas. El diseño, fabricado en 1854, fue empleado por los gobiernos británico y estadounidense (Fig. 1.4).

Unos años más tarde, concretamente en 1854, el matemático inglés George Boole comenzó a desarrollar el álgebra que lleva su nombre y que es parte fundamental en el diseño de los sistemas digitales (Fig. 1.5).



Figura 1.3. *Charles Babbage.*



Figura 1.4. *Grabado que representa a la máquina diferencial de Babbage.*



Figura 1.5. *George Boole.*

En 1943, y en plena Segunda Guerra Mundial, un coronel del ejército de los Estados Unidos solicitó una máquina calculadora para la artillería. El reto fue aceptado por la Universidad de Pensilvania, que en 1946 presentó la máquina a la que se le asignó el nombre de ENIAC (*Electrical Numerical Integrator And Calculator*) (Fig. 1.6). Se trataba del primer ordenador de válvulas. Esta enorme construcción, instalada en una sala de 9×30 m, presentaba problemas tales como su falta de fiabilidad o una escasa capacidad de memoria y de funcionamiento, ya que únicamente podía funcionar unos dos minutos antes de que las válvulas empezaran a fundirse.

Sin embargo, el hito más importante en la historia de la computación fue un artículo publicado en 1946 por el matemático estadounidense de adopción, húngaro de origen, von Neumann, en el que se plantea la primera arquitectura abierta de un sistema microprocesador. En 1944, von Neumann tuvo noticias sobre el proyecto secreto ENIAC, del que elabora un informe, presentando el bosquejo del diseño de un nuevo ordenador, que recibirá el nombre de EDVAC (*Electronic Discrete VAriable Computer*) y que será construido según sus ideas. Muchos de los sistemas microprocesadores actuales responden a la arquitectura propuesta por von Neumann (Fig. 1.7).

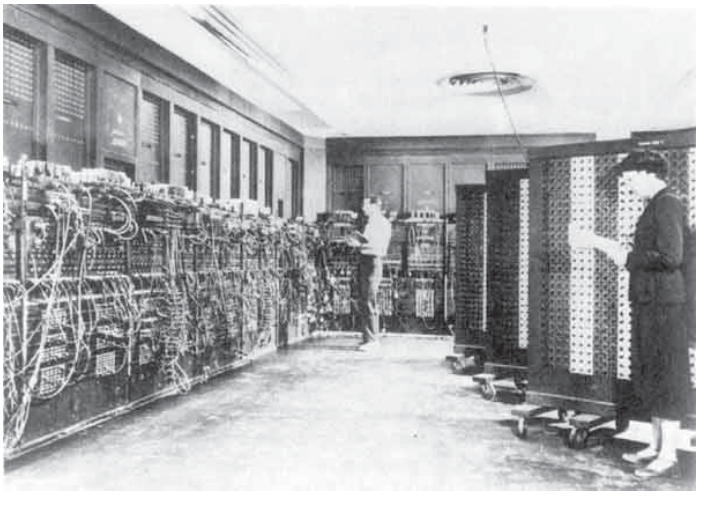


Figura 1.6. *ENIAC.*

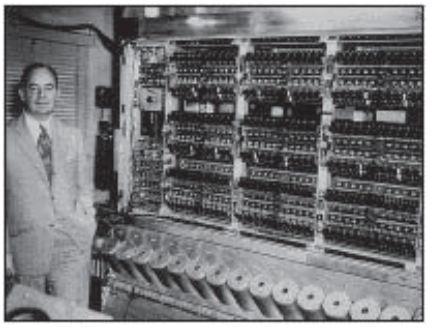


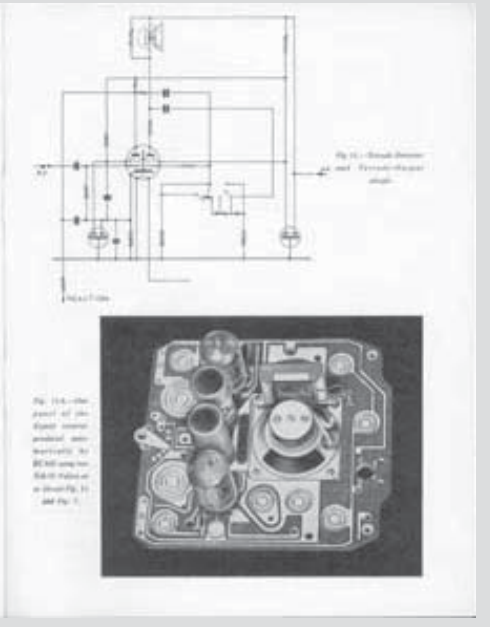
Figura 1.7. *John Louis von Neumann y el EDVAC.*

El desarrollo de la ciencia que conocemos por el nombre de electrónica comienza en esta época, a finales de la primera mitad del siglo XX. De hecho, los dispositivos electrónicos que existían en tiempos de la Segunda Guerra Mundial no eran excesivamente complicados (a título de ejemplo, los bombarderos B-29, la maquinaria de guerra más compleja utilizada por los norteamericanos en dicha contienda mundial, disponían de unas 300 válvulas de vacío, la mayoría de las cuales equivalía al dispositivo electrónico más usual hoy en día, el transistor). Por aquella época, la integración no era una prioridad, aunque antes de la guerra ya aparecieron algunos intentos tendentes a disminuir el coste, simplificar el ensamblaje y disminuir el tamaño y peso de los dispositivos electrónicos (radios de *Sargrove*, espoletas de proximidad para aumentar la efectividad de los antiaéreos norteamericanos durante la Segunda Guerra Mundial, etc.).

Las radios del británico de origen húngaro John Adolphe Szabadi (Sargrove)

Sargrove, para conseguir introducir y vender en el mercado indio radios de muy bajo coste, diseñó y fabricó una radio sobre dos láminas de plástico; puede decirse que había desarrollado el primer circuito impreso. Los conductores, capacidades e inductancias necesarios se montaron en el sustrato del sistema, y al avanzar en el diseño de la placa del circuito se conectaron dos válvulas de vacío. Esta idea, sin embargo, tuvo un escaso éxito comercial, hasta el punto de que cuando al final de la Segunda Guerra Mundial se define nuevamente la ubicación de la banda de frecuencia de la radio comercial, la radio diseñada se hizo obsoleta, y *Sargrove* no encontró fondos para financiar su actualización.





El elemento base de la electrónica, el transistor, es inventado en 1947 por John Bardeen, Walter Brattain y William Shockley, en los laboratorios de la compañía Bell Telephone. Entre los posibles usos del transistor no se encontraba, en principio, su empleo en el diseño de ordenadores, ya que la aparición del mercado y la industria de los ordenadores e informática era entonces algo totalmente imprevisible. Pocos científicos supieron ver la potencialidad de estos minúsculos dispositivos, que han servido de base al increíble desarrollo tecnológico que ha experimentado la humanidad en las últimas décadas (ejemplo de esto es la denominación que suele darse en numerosa bibliografía científica al siglo XX: el siglo de la electrónica). Inicialmente, los transistores se aplicaron en la mejora de los sistemas analógicos ya existentes (radios, televisiones, etc.). Todo el desarrollo inicial estaba orientado hacia los dispositivos electrónicos analógicos, cuyo principal inconveniente se relacionaba con el hecho de que cada circuito era diferente de los demás. En aquella época, los diseñadores de equipos de televisión no repetían dentro de un aparato los circuitos electrónicos y se sentían muy orgullosos de ese hecho, y así un aparato de televisión podía llegar a tener hasta 30 circuitos analógicos diferentes entre sí. La aparición de los primeros ordenadores ayudó a vislumbrar la necesidad de utilizar gran cantidad de componentes idénticos (circuitos electrónicos digitales) en un único sistema electrónico.

Los circuitos electrónicos digitales empezaron a desarrollarse por esta época, pero mucho más lentamente que los analógicos. En 1958, un ingeniero eléctrico recién llegado a la compañía norteamericana Texas Instruments, Jack St. Clair Kilby, presentó a un grupo de colaboradores la idea de integrar, en una única pastilla de germanio, un transistor junto a otros componentes: había surgido la idea del circuito

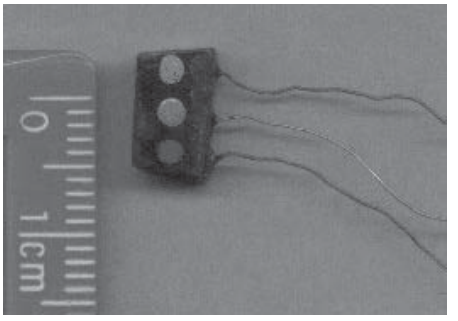


Figura 1.8. *El transistor.*

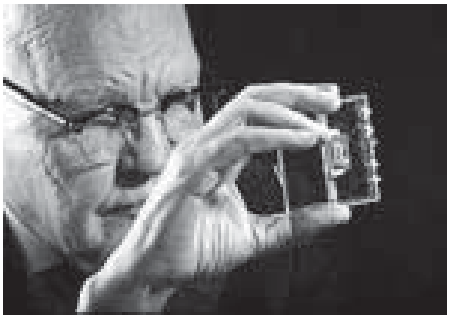


Figura 1.9. *Jack Kilby con un circuito integrado monolítico.*

integrado monolítico (idea que le permitió obtener el nobel de Física en el año 2000). A partir de ese momento, el transistor se convierte en la base de pequeños chips o circuitos electrónicos integrados (puertas AND, OR, etc.), que sirvieron de dispositivos elementales en el diseño y realización del sistema microprocesador. El empleo de tantos componentes repercutía en una escasa fiabilidad, ya que el fallo de un componente, o de una conexión soldada, implicaba que todo el circuito dejara de funcionar.

Simultáneamente, comenzó la estandarización, en cuanto a la forma y el tamaño, de los dispositivos electrónicos con vistas a facilitar su interconexión y se desarrollaron nuevos dispositivos basados en las características moleculares de los materiales, como es el caso del cristal de cuarzo, que es capaz de comportarse como una inductancia o una capacidad, sin ser ni lo uno ni lo otro. La principal oposición a este desarrollo provenía de los propios diseñadores de circuitos electrónicos, normalmente integrados dentro de las mismas compañías y centros encargados de financiar toda esta tarea de investigación, que veían el desarrollo de los circuitos integrados monolíticos como un competidor a su tarea.

El empuje definitivo llegó cuando, a principios de los años sesenta, el Gobierno norteamericano se embarca en dos grandes proyectos, el misil balístico *Minuteman* y la misión *Apollo*. El ordenador del *Apollo* era un dispositivo electrónico muy sencillo, mientras que el del misil *Minuteman* consistía en un sistema con más de 2.000 circuitos de 22 tipos diferentes. Ambos programas utilizaban como componentes circuitos integrados monolíticos. El buen desarrollo de los dos proyectos aumentó sustancialmente la reputación, en cuanto a fiabilidad, de los circuitos integrados monolíticos. A partir de ese momento, el desarrollo de la electrónica digital fue vertiginoso.

Múltiples compañías comienzan a utilizar este tipo de dispositivos en sus productos comerciales, y a finales de los años sesenta, la mayoría de los ingenieros había aceptado la idea de que los circuitos integrados estaban aquí para quedarse. Los siguientes pasos en cuanto a la implementación de los sistemas microprocesadores se dan para aumentar su capacidad de integración. Así, Gilbert Hyatt presenta en 1970 la patente del microprocesador, todo un sistema microprocesador integrado en un único chip digital con una pequeña CPU, memorias y periféricos que facilitaban su comunicación con el exterior. En 1971 Hyatt se separó de sus socios,

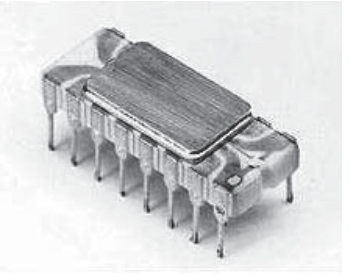


Figura 1.10. *Intel 4004.*

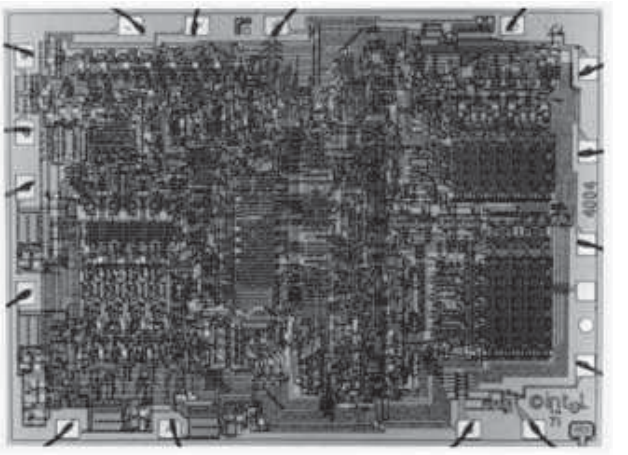


Figura 1.11. *Vista interna del Intel 4004.*

creando estos últimos una nueva compañía que se llamó Intel. Mientras se tramitaba la patente, al año siguiente de su presentación en la oficina de patentes estadounidense, tres ingenieros de Intel desarrollaron y pusieron en el mercado el primer microprocesador, el Intel 4004. Contenía 2.250 transistores y manejaba datos de 4 bits. Como curiosidad, comentar que la patente no le fue reconocida a Hyatt hasta 1990, después de una larga batalla legal.

Posteriormente apareció el Intel 8008, un dispositivo de 8 bits de datos mucho más avanzado, y en 1978 aparecieron los primeros microprocesadores de 16 bits: el Intel 8086 (que funcionaba con relojes de 5 MHz, 8 MHz o 10 MHz), el 68000 de Motorola y el Z8000 de Zilog. Los microprocesadores de última generación son capaces de manejar datos de 64 bits y trabajar con relojes de frecuencias superiores al GHz (Intel ofrece actualmente para PC el Pentium IV, que opera con relojes de hasta 3,4 GHz y frecuencias del bus del sistema de 800 MHz, y para estaciones de trabajo y multiproceso el Xeon, con relojes externos similares y frecuencias del bus del sistema de 533 MHz)[2].

Simultáneamente, y en paralelo con el desarrollo de la electrónica, se ha producido un importante desarrollo de la programación. Cuando aparecen los primeros ordenadores, la forma del usuario para programarle al sistema el conjunto de operaciones a realizar era mediante la utilización del código máquina (conjunto de ceros y unos con significado para el procesador). En 1952, Grace M. Hopper desarrolla el primer compilador, programa capaz de traducir instrucciones creadas por el usuario (cercanas al lenguaje hablado) a código máquina. Posteriormente interviene en el desarrollo de otros lenguajes de programación, como el COBOL. Desde entonces a nuestros días, la evolución experimentada en los sistemas de programación de los microprocesadores ha pasado del empleo del rústico ensamblador al potente C, que simplifica la programación del dispositivo electrónico enmascarando el *hardware* al usuario.

---

[2] Información obtenida de *http://www.intel.com*, mayo de 2004.

La aparición de los microprocesadores revolucionó completamente el diseño electrónico, especialmente de los sistemas de control. De una forma gráfica, podría decirse que el ingeniero electrónico cambió el soldador y el osciloscopio del laboratorio por el ordenador del despacho. En la actualidad el diseño se hace con ordenador, empleándose herramientas que permiten simular el comportamiento del microprocesador antes de pisar el laboratorio. El diseño de la etapa de control de cualquier sistema se ha convertido en la realización de un programa para un microprocesador, con lo que se ha aumentado la versatilidad del mismo, puesto que una modificación de las especificaciones implica tan sólo la modificación del programa (*software*) sin necesidad de cambiar ningún elemento de la placa (*hardware*).

¿Y el futuro? Claro está que la tendencia actual de los sistemas microprocesadores de tipo electrónico es la de aumentar la velocidad de procesamiento (la archiconocida ley de Moore establece que el número de transistores de un sistema microprocesador, y por tanto la potencia de cálculo de éste, se duplica aproximadamente cada dieciocho meses). Sin embargo, ¿está el futuro de la computación ligado al de los sistemas electrónicos digitales que conocemos? Esta última pregunta no parece que tenga una clara respuesta, a tenor de cómo evoluciona la ciencia. Baste comentar, por ejemplo, la reciente aparición, a nivel experimental, de nuevos artificios capaces de procesar información a partir de los niveles cuánticos de los dispositivos electrónicos prácticos, los denominados procesadores cuánticos o computación cuántica. Se estima que este tipo de dispositivos revolucione nuevamente el nivel de eficiencia computacional. De momento, existen diferentes empresas y organismos de investigación (NEC, Lucent Technologies Bell Laboratories, Universidad Estatal de Michigan, etc.) que han conseguido desarrollar dispositivos de estado sólido capaces de funcionar como bits (elemento básico de la computación) en su nivel cuántico. La realización de un ordenador cuántico aún no es factible, pero se encuentra mucho más cerca.

A partir de ahora, vamos a analizar el funcionamiento y componentes elementales de los sistemas microprocesadores (y por ende al microprocesador, tal y como lo concibió Hyatt), sistemas realizados electrónicamente y que conforman uno de los principales y más complejos elementos del área de la ciencia que denominamos electrónica digital.

1.2. DESCRIPCIÓN DE UN SISTEMA MICROPROCESADOR

Un sistema microprocesador es, por tanto, un sistema digital basado en un microprocesador (dispositivo digital provisto de CPU) que se caracteriza por su capacidad para procesar información y que normalmente constituye un sistema electrónico de pequeño tamaño, elevada fiabilidad, poco tiempo de diseño y bajo coste, si se compara con cualquier otro tipo de sistema electrónico. Su estructura, características básicas y componentes son los que expondremos a continuación.

1.2.1. *Periférico de un sistema microprocesador*

Un periférico es un dispositivo cuya finalidad es ayudar a la CPU, realizando alguna tarea que ésta no puede realizar, como por ejemplo el almacenamiento[3] de datos (nor-

[3] Periféricos denominados periféricos de memoria.

malmente dispositivos de tipo volátil o RAM) o de programa (normalmente dispositivos de tipo no volátil o ROM) o la implementación de algunas funciones relacionadas con la comunicación con el exterior del sistema[4], como puede ser la lectura de variables analógicas y el cálculo de su correspondiente valor digital (periféricos de conversión analógico-digital), etc. Su esquema general se muestra en la Figura 1.12.

Los periféricos suelen disponer de registros internos a los que accede la CPU mediante órdenes de lectura y escritura. El número de registros internos que puede tener un periférico de un sistema microprocesador depende de su complejidad y funcionalidad, y puede variar desde no tener ninguno o tener pocos (lo que ocurre en la mayoría de periféricos) a tener un elevado número (como es el caso de los dispositivos de memoria RAM o ROM).

Dentro de los registros internos de que dispone el periférico, cabe distinguir los siguientes:

- **Registros de control.** Son aquellos registros internos del periférico que permiten configurar las condiciones en las que va a trabajar el dispositivo. A estos registros se puede acceder desde la CPU en lectura y escritura y suelen programarse una única vez, al principio del programa y antes de que la CPU empiece a trabajar con el periférico. Por ejemplo, existen periféricos de conversión analógico-digital con capacidad de generar datos digitales de 8 o 10 bits, por lo que antes de empezar a trabajar con él, la CPU debe configurar el tipo de dato convertido.
- **Registros de estado.** La CPU suele acceder a las banderas de este tipo de registros únicamente en lectura, para observar el estado del periférico. Así, en un periférico de conversión de analógico a digital (CAD en adelante) puede existir un registro de estado en el que se active una bandera para indicar el final de un proceso de conversión.

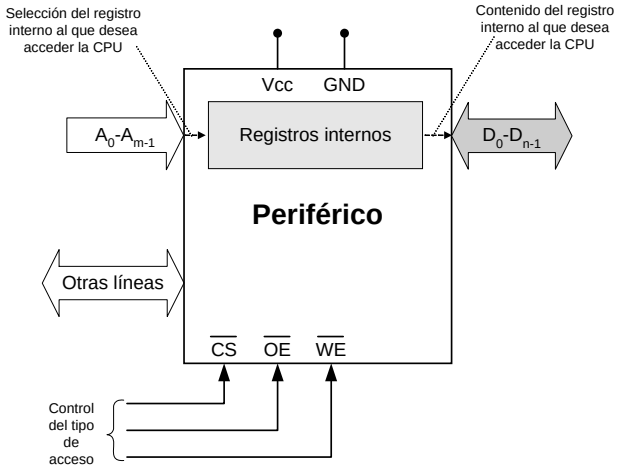


Figura 1.12. *Esquema general del periférico de un sistema microprocesador.*

[4] Periféricos a los que se denomina, de forma genérica, periféricos de entrada/salida (I/O).

- **Registros de datos.** Son los registros de trabajo del periférico y, por tanto, los que más utiliza la CPU. El periférico emplea estos registros como fuente para la realización de la tarea para la que fue seleccionado. Continuando con el ejemplo del periférico CAD, una vez convertido el valor analógico en su equivalente digital, la CPU accede a algún registro de dato interno al periférico para recoger el dato convertido. En general, son registros que pueden ser accedidos desde la CPU en lectura o escritura.

A veces puede ocurrir que el dispositivo carezca de algún tipo de registro interno de los antes analizados, como es el caso de las memorias que sólo disponen de registros de datos, pues ni necesitan ser configuradas ni constan de registro de estado. En cualquier caso, para que la CPU pueda acceder a los registros internos del periférico, éste dispone de un conjunto de conexiones eléctricas, o líneas de entrada y salida, entre las que se encuentran:

- Las líneas de dirección (que se suelen nombrar, de manera genérica, A_i). Sirven para indicar el registro interno del periférico al que se desea acceder. Para ello, el periférico dispone de su propio rango de direcciones de memoria en el que cada registro interno se asocia a un valor de las líneas de dirección. Como consecuencia, el número de líneas de dirección de entrada al periférico varía en función del número de registros internos de que disponga.
- Las líneas de dato (que se suelen nombrar, de manera genérica, D_i), que permiten la transferencia del contenido de los registros internos del periférico a la CPU.
- Otras líneas de entrada al periférico que permiten y controlan la transferencia de información entre el propio periférico y la CPU, como las siguientes:

  – La señal de selección del periférico, normalmente activa a nivel bajo y que se denomina $\overline{CS}$, *Chip Select*, o $\overline{CE}$, *Chip Enable*. Mientras no se seleccione el periférico (el valor que se impone en esta entrada es un uno lógico), las líneas de dato del mismo se encuentran en alta impedancia. Así, el periférico no impone ni un cero ni un uno lógico en ellas. Por tanto, cuando la CPU desee establecer una transferencia de información con el periférico, deberá seleccionarlo previamente.
  – Las señales de selección del tipo de acceso que realiza la CPU (lectura del o escritura al periférico). El periférico dispone de una señal (normalmente denominada $R/\overline{W}$, *Read/Write*) o de dos señales ($\overline{OE}$, *Output Enable*, y $\overline{WE}$, *Write Enable*) de entrada que utiliza la CPU para indicarle el tipo de transferencia que va a realizar con el periférico.

Dependiendo del tipo de periférico, puede aparecer otro tipo de líneas de entrada o salida asociadas a él. La función de éstas es muy diversa. Por ejemplo, es común encontrarse periféricos con líneas de entrada que permiten configurar su modo de funcionamiento. En el caso de los CAD, cuando el periférico termina el proceso de conversión, puede interesar generar un aviso a la CPU para indicarle dicho evento, por lo que será normal encontrar periféricos de CAD con líneas de salida que realicen esta función. Dada la variedad de periféricos que el lector puede encontrar en el mercado, no vamos a entrar en detalle en este tema.

1.2.2. *La unidad central de proceso (CPU)*

La CPU es el núcleo de un sistema microprocesador y el dispositivo que controla el resto de componentes del mismo. Es un dispositivo digital capaz de interpretar las instrucciones programadas por el usuario del sistema, de controlar el resto de componentes del mismo, para que se ejecute dicha secuencia de instrucciones, y de realizar una serie de operaciones elementales de manejo de datos, entre las que se pueden destacar las siguientes:

- Transferencia de datos: lee un dato ubicado en una determinada posición de memoria para trasladarlo a algún registro interno de la CPU, almacena un registro de la CPU en una determinada posición de memoria del sistema o transfiere el contenido de un registro de la CPU a otro.
- Operaciones aritméticas elementales: suma, resta, incrementa y decrementa el contenido de los registros internos de la CPU.
- Operaciones lógicas (AND, OR, XOR y NOT) con datos y de manejo de biestables de la CPU y desplazamiento o rotación, hacia la izquierda o hacia la derecha, del contenido de los registros internos de la CPU.

Se puede decir, por tanto, que la CPU está compuesta de dos partes, una que se encarga del control y del manejo de las instrucciones y otra que se encarga de la realización de las operaciones aritméticas y del manejo de los datos del sistema, cada una de las cuales necesita una serie de registros internos para desarrollar sus funciones. En la Figura 1.13 se muestra el esquema genérico de una CPU.

Control y manejo de instrucciones

La zona de control y manejo de instrucciones es la encargada de gobernar y coordinar el funcionamiento del sistema, enviando las denominadas *microórdenes*[5] a todos los elementos que deben intervenir en cada momento. Se trata de un circuito eminentemente secuencial y síncrono (básicamente está constituido por autómatas síncronos) que genera microórdenes, que a su vez controlan el funcionamiento de los registros internos y de la zona de manejo de datos de la CPU.

Para que pueda desempeñar correctamente su función, esta parte de la CPU necesita dos datos, que son (Fig. 1.14):

- La *instrucción* que ha programado el usuario para que sea ejecutada. Es un código binario que le indica a la CPU la función que debe realizar y, si hace falta, los operandos que debe emplear en la realización de dicha operación. El controlador interno interpreta el significado de dicho código y genera la secuencia de microórdenes necesarias para su ejecución. Aunque las instrucciones son pala-

[5] Conjunto de órdenes básicas que es capaz de implementar la CPU de un sistema microprocesador. Estas microórdenes realizan funciones como, por ejemplo, indicarle a la zona de manejo de datos de la CPU que realice una operación, activar la salida de un registro para que pueda ser usado como operando de una operación, etc.

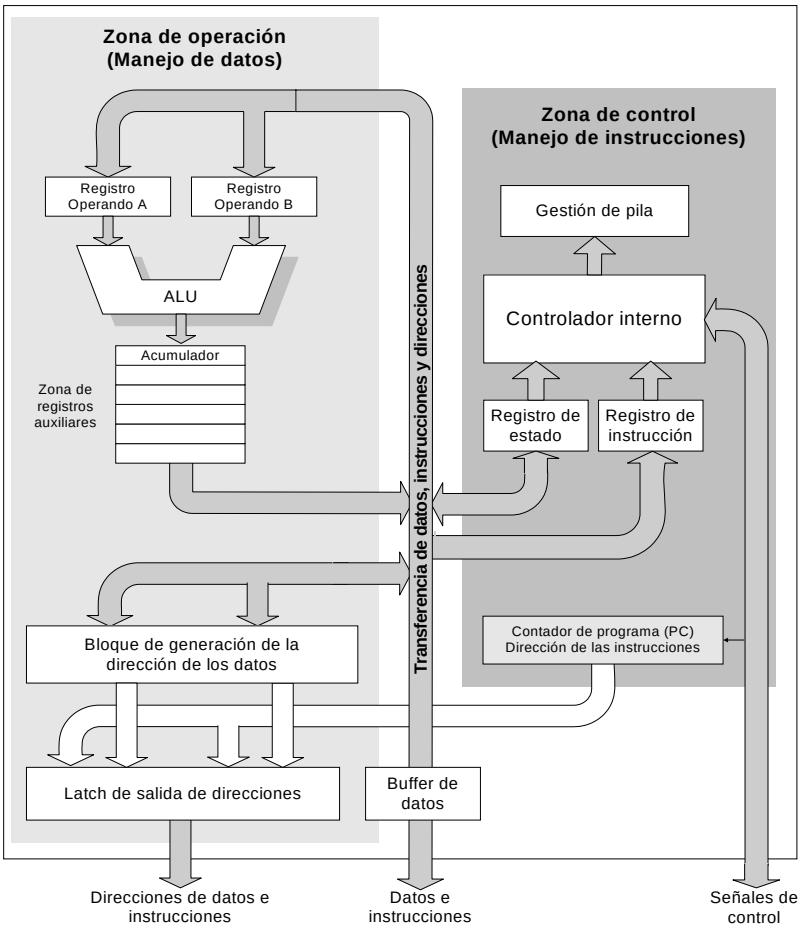


Figura 1.13. *Estructura básica de una CPU.*

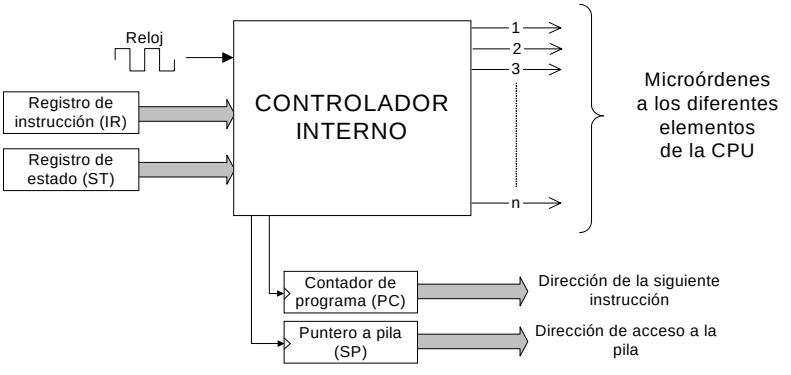


Figura 1.14. *Estructura básica de la zona de control y manejo de instrucciones de la CPU.*

bras binarias constituidas por ceros y unos, frecuentemente nos referiremos a ellas, y las utilizaremos, mediante una abreviatura o código mnemotécnico (lenguaje ensamblador) que simplificará la interpretación y utilización de las mismas durante la confección del programa y facilitará la tarea del programador. Téngase en cuenta que cada instrucción del juego de instrucciones que tiene asignado cualquier microprocesador involucra en su ejecución a varias de estas operaciones elementales. Por ejemplo, una instrucción de suma supone realizar operaciones elementales de recogida de la propia instrucción, decodificación de la instrucción, generación de las direcciones de los operandos, lectura de los operandos, realización de la operación aritmética y almacenamiento del resultado.

- El *estado* en el que se encuentra la CPU antes de ejecutarse una instrucción.

Para almacenar estos datos, en la zona de control y manejo de datos de la CPU aparecen dos registros internos: el registro de instrucción (IR, o *Instruction Register*), que mantiene la siguiente instrucción que debe ejecutar la CPU, y el de estado (ST, o *Status Register*), que está constituido por un conjunto de biestables (banderas) que pueden activarse independientemente unos de otros y que indican el estado en el que se encuentra la CPU.

Las instrucciones deben ser leídas e interpretadas por la CPU para que ésta pueda realizar la secuencia de operaciones que se le haya programado. De alguna manera, las instrucciones que programa el usuario del sistema microprocesador son recogidas por la parte de control de la CPU y trasladadas al registro de instrucción. La zona, dispositivos de almacenamiento de tipo volátil (RAM) y no volátil (ROM), donde se encuentran las instrucciones que programa el usuario de un sistema microprocesador, y que va recogiendo secuencialmente la CPU, se denomina memoria de programa. Para poder gestionar el acceso a ella, la CPU debe conocer en cada instante dónde se encuentra la instrucción que debe ejecutar. Para ello dispone de un registro conocido como contador de programa (PC, o *Program Counter*) que funciona como un contador que se incrementa con cada instrucción que se ejecuta, apuntando siempre a la siguiente instrucción que se va a ejecutar. De esta forma tan simple, la zona de control asociada a la CPU es capaz de secuenciar o serializar la ejecución de las instrucciones del programa del usuario, lo que, por otro lado, es el modo de funcionamiento típico de la CPU.

Sin embargo, el programador no siempre desea secuenciar la ejecución de sus instrucciones. A veces es necesario romper, temporalmente, la secuencia normal de ejecución del programa y cambiar la forma habitual de incrementar el registro PC; por ejemplo, al ejecutar un bucle, llamar a una subrutina o cuando aparecen interrupciones en el sistema microprocesador. En estos dos últimos casos, la CPU dispone de un método automático de salvaguarda del registro PC y, por extensión, de cualquier otro registro del sistema microprocesador, que hace uso de una zona de memoria RAM de tamaño variable que debe reservar el usuario expresamente para ese fin y que recibe el nombre de pila.

Asociado a la pila, la CPU dispone de un registro especial que se denomina puntero a la pila (SP, o *Stack Pointer*) para la gestión de ésta y que, como su nombre indica, apunta a la posición de memoria asociada a la pila en la que se guardará el siguiente dato que se desee almacenar. Este registro tiene la propiedad de incrementarse o decrementarse automáticamente cada vez que se ejecuta una instrucción de acceso a la pila. En la Figura 1.15 se muestra el proceso de ruptura de la secuencia normal de

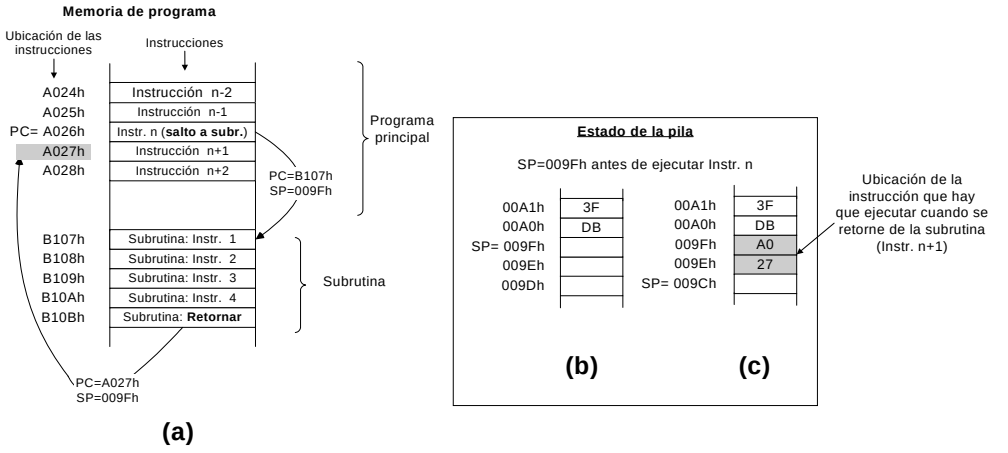


Figura 1.15. *Proceso de salto a una subrutina:* **(a)** *flujo del programa;* **(b)** *estado de la pila antes y después de ejecutar la subrutina, y* **(c)** *estado de la pila mientras se ejecuta la subrutina.*

ejecución de instrucciones de una CPU por la ejecución de una llamada a una subrutina. A modo de ejemplo, y para simplificar, se supone que cada instrucción ocupa un byte y que al salvar datos en la pila ésta aumenta de tamaño hacia direcciones decrecientes. Es importante resaltar que la CPU sólo dispone para la gestión de la pila del registro SP, y que físicamente, la pila es una zona de almacenamiento de datos que pueden ser escritos y leídos, por lo que coincidirá con una zona de memoria de tipo RAM, externa a la CPU.

Por su parte, muchas de las banderas del registro de estado se relacionan con el resultado de una operación aritmética realizada e indican si ésta ha sido cero (bandera de cero) o negativo (bandera de signo), si se ha producido un desbordamiento en el resultado generado por la unidad de manejo de datos (bandera de desbordamiento), si se ha generado un acarreo (bandera de acarreo), etc. Hay además otras banderas de configuración que sirven para enmascarar interrupciones, entrar en modos especiales de operación, controlar el modo de funcionamiento de la unidad de procesamiento de datos ante un desbordamiento, etc. El estado de todas estas banderas puede ser consultado por la zona de control de la CPU, condicionando su modo de actuación, y por el programador, para el que estas banderas serán claves en la toma de decisiones relacionadas con los programas.

Manejo de datos

La zona de manejo de datos y operaciones está compuesta fundamentalmente por una serie de operadores (la denominada unidad aritmético-lógica, o ALU, *Arithmetic Logic Unit*) y de registros internos que actúan como operandos, o variables de entrada, de dichos operadores y como registros de almacenamiento del resultado de una operación. Esta zona es controlada por la zona de manejo de instrucciones de la CPU y es la que realiza las operaciones indicadas por las instrucciones del usuario.

La ALU es un circuito formado por otros subcircuitos combinacionales capaces de realizar ciertas operaciones de tipo lógico (desplazamientos, AND, OR, XOR, etc.) y aritmético (suma y resta normalmente y multiplicación en algunos casos) entre uno o dos operandos, generando un resultado que se deposita en algún registro interno de la CPU. Las señales de entrada que provienen de la parte de control de la CPU permiten seleccionar el tipo de operación que se va a realizar, los operandos utilizados, así como el registro en el que se almacenará el resultado.

La mayoría de los registros internos de la parte de manejo de datos de la CPU son registros auxiliares que se pueden emplear como operandos de las operaciones aritméticas. Entre éstos destaca el denominado registro acumulador, que es un registro que puede servir de operando y que recoge el resultado de cualquier operación que realice la ALU. Cuando se está realizando una operación, la orden de carga del resultado en el acumulador, procedente de la parte de control de la CPU, debe llegar después de que los operandos hayan sido almacenados en sus respectivos registros y con el retraso suficiente como para que la salida de la ALU sea estable. En la mayoría de los casos, uno de los registros internos que funciona como operando para la ALU es el propio registro acumulador.

El número y propiedades de estos registros auxiliares, asociados a la parte de manejo de datos de la CPU, varían mucho de un microprocesador a otro y depende, en gran medida, de las características y potencia de la propia CPU. Así, es posible que la CPU tenga, en lugar de un único registro acumulador[6], todo un conjunto de registros internos que pueden funcionar como acumuladores[7].

La zona de manejo de datos de la CPU deberá recoger los datos que determinen las instrucciones del usuario para poder operar con ellos. A la zona, dispositivos de almacenamiento de tipo RAM y ROM, donde se encuentran dichos datos se denomina memoria de datos. Para poder gestionar el acceso a ella, la parte de manejo de datos de la CPU suele disponer de operadores auxiliares encargados de generar la dirección donde se encuentra el dato a emplear, así como de registros internos con los que pueden trabajar estos operadores auxiliares para generar las direcciones de memoria de los datos con los que va a operar, además de índices que permitan recorrer posiciones de memoria consecutivas y que facilitan el acceso a tablas.

1.2.3. *Arquitectura de un sistema microprocesador*

En el apartado anterior se ha visto la necesidad que tiene la CPU de acceder al exterior, por ejemplo a dispositivos externos de memoria de programa y de datos para recoger las múltiples instrucciones que deben ejecutarse y los datos con los que opera cada una de ellas.

El acceso a toda esta información se realiza mediante un número fijo de líneas que tiene la CPU dedicadas a la comunicación con su entorno denominadas buses del sistema (Fig. 1.16). Si la CPU no empleara los buses para acceder a la información, tendría que dedicar en exclusividad ciertas líneas a cada uno de los dispositivos exter-

---

[6] A este tipo de ALU se le suele denominar ALU orientada a acumulador.

[7] A este otro tipo de ALU se le suele denominar ALU orientada a registro.

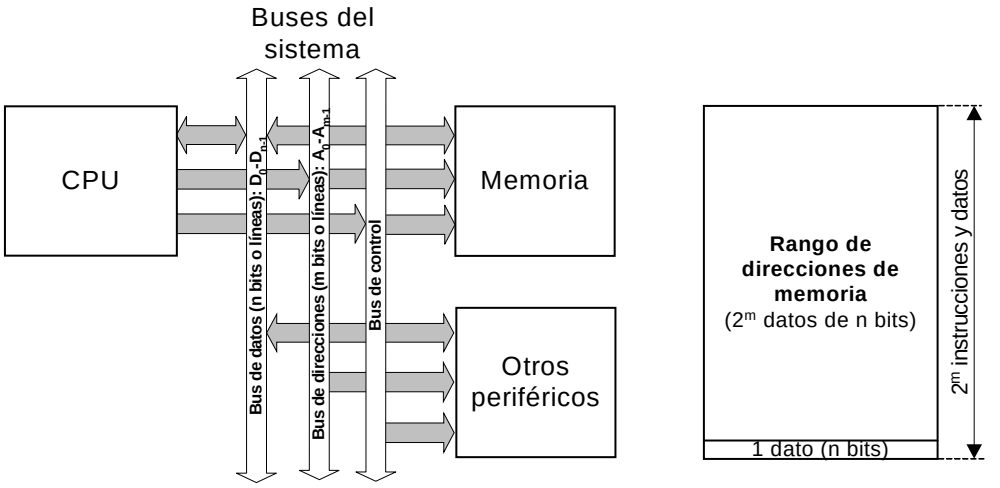


Figura 1.16. *Estructura básica de un sistema microprocesador.*

nos con los que debe comunicarse, encareciendo enormemente el sistema microprocesador y dificultando en exceso su diseño e implementación. Un bus es, por tanto, una agrupación de conexiones eléctricas que manejan niveles digitales de tensión y que facilita la conexión de la CPU con el resto de dispositivos del sistema microprocesador. Los buses que permiten que la CPU lea y escriba en dispositivos externos de memoria son:

- **Bus de control.** Conjunto de conexiones eléctricas o líneas, de entre las que destacan ciertas líneas de salida de la CPU que se emplean para controlar la comunicación de la propia CPU con los dispositivos de su sistema microprocesador (indicación de acceso a dispositivos externos, del tipo de acceso —lectura o escritura—, etc.). Además de las anteriores, aparecen algunas líneas de entrada a la CPU que sirven para hacer llegar la señal de reloj a la CPU del sistema, la inicialización de la CPU o que permiten a algún dispositivo externo requerir la atención de la CPU, provocando la interrupción en su funcionamiento secuencial. Conforme aumenta el grado de complejidad de la CPU, pueden aparecer otras líneas de entrada y salida, relacionadas, por ejemplo, con el control de la temporización en los accesos a periféricos externos que realice la CPU, y por tanto ligadas al bus de control, o que ofrecen una funcionalidad diferente de la expuesta anteriormente.
- **Bus de datos.** Conjunto de líneas o conexiones eléctricas por las que circulan los datos entre la CPU y los dispositivos externos del sistema microprocesador. Cuando la CPU desea almacenar un dato en un dispositivo externo (acceso en escritura), utiliza el bus de dato como líneas de salida en las que escribe lo que desea almacenar, mientras que cuando desea recoger algo de un dispositivo externo (acceso en lectura), emplea el bus de datos como camino entrada a la CPU del dato o instrucción a leer.

- **Bus de direcciones.** Son otro conjunto de líneas o conexiones eléctricas, normalmente de salida de la CPU, que le sirven a ésta para indicar la dirección del dato o la instrucción que desea leer o del dato que quiere almacenar. Este conjunto de líneas, junto con algunas de las líneas de salida del bus de control, sirven para fijar el número máximo de instrucciones y datos a los que es capaz de acceder la CPU. Una CPU con *m* líneas digitales asociadas al bus de dirección es capaz de direccionar 2^m elementos, y por tanto, una CPU con un bus de direcciones constituido por 16 líneas sería capaz de direccionar 2^{16} posiciones (64 K instrucciones y datos, Figura 1.17). Al conjunto de instrucciones y datos que es capaz de direccionar una CPU en un sistema microprocesador se le denomina rango de direcciones de memoria.

El sistema microprocesador debe entenderse como un sistema digital en el que la CPU funciona como único maestro (*master*) y en el que los periféricos o dispositivos de almacenamiento de instrucciones y datos, de conversión analógico-digital y digital-analógico, etc., funcionan como esclavos (*slaves*). El maestro es el dispositivo que se encarga de gestionar la comunicación en el sistema digital y de controlar la transferencia de información por los buses del sistema. Toda transferencia de información que se realice comienza y concluye cuando lo decide la CPU. A través de los accesos

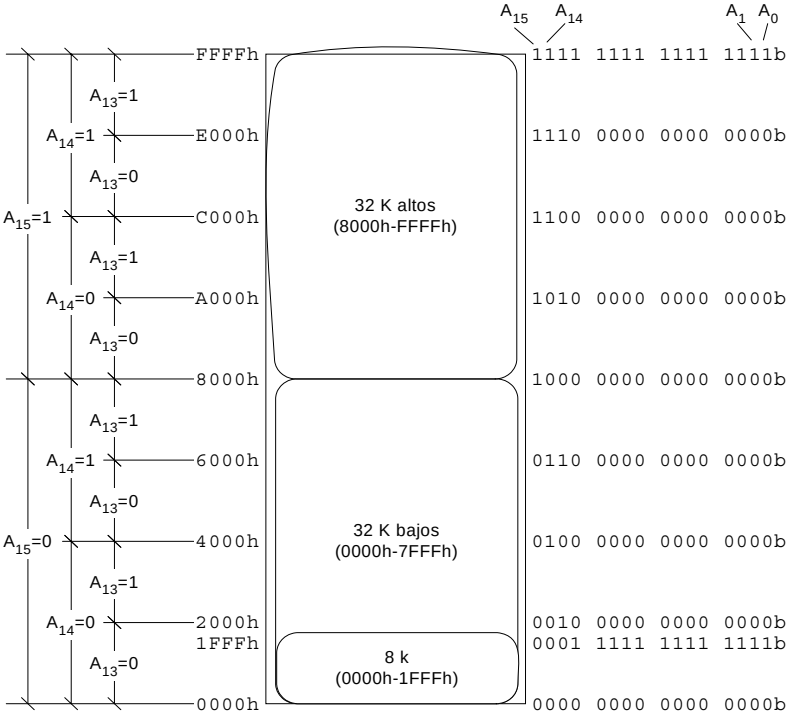


Figura 1.17. *Rango de direcciones de memoria de una CPU con 16 líneas de dirección (bus de direcciones con 16 líneas, desde A_0 hasta A_{15}).*

en lectura o escritura, la CPU comienza, ya sea en modo lectura o escritura, y termina el acceso al periférico al que quiere acceder mediante las líneas asociadas a los buses de control y dirección, siguiendo una estricta temporización que fija el fabricante y que indica mediante los cronogramas o diagramas de tiempo de las señales de la CPU que controlan dichos accesos.

Los sistemas microprocesadores actuales se suelen clasificar según la funcionalidad de la CPU y según la arquitectura interna del propio sistema microprocesador. Según su funcionalidad, se diferencia entre sistemas CISC (*Complex Instruction Set Computer*, con un juego complejo de más de 80 instrucciones, algunas de las cuales son muy complejas y potentes y requieren muchos ciclos de reloj del sistema para ejecutarse), RISC (*Reduced Instruction Set Computer*, con un juego reducido de instrucciones, disponiendo de pocas instrucciones que se ejecutan normalmente en un ciclo de reloj y que permiten optimizar el *hardware* del sistema) y SISC (*Specific Instruction Set Computer*, con un juego específico de instrucciones, o CPU destinadas a aplicaciones muy concretas cuyo juego de instrucciones se adapta a la aplicación concreta a la que se destina el sistema).

La principal clasificación de los sistemas microprocesadores es la que se basa en la estructura interna de éstos, que distingue entre sistemas microprocesadores con arquitectura de tipo von Neumann o de tipo Harvard.

En las arquitecturas de tipo von Neumann, las instrucciones y los datos son almacenados en memorias (de tipo RAM o de tipo ROM) a las que accede la CPU a través de un único bus de direcciones, de datos y de control (Fig. 1.18). El sistema microprocesador accede a la memoria de programa para recoger la instrucción a ejecutar; posteriormente decodifica la instrucción para determinar la secuencia de operaciones

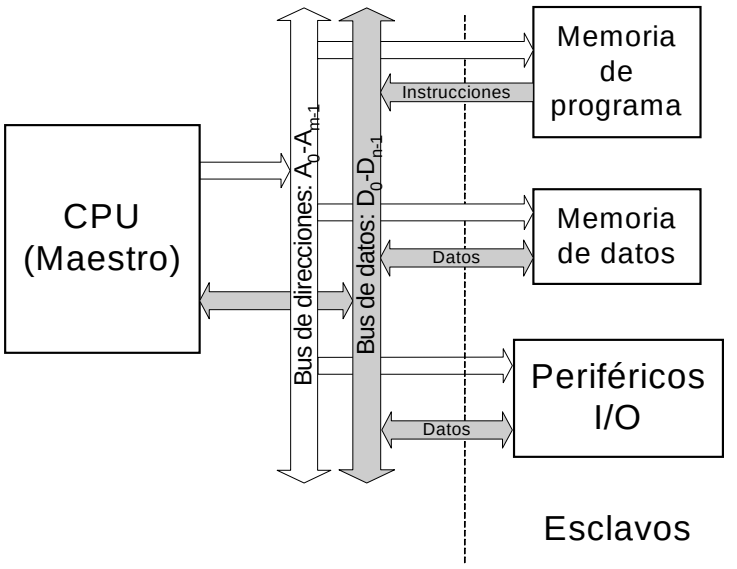


Figura 1.18. *Arquitectura de tipo von Neumann asociada a un sistema microprocesador (el bus de control del sistema no se ha representado para simplificar el esquema).*

que ésta lleva asociada y accede a la memoria de datos para leer los operandos asociados a la instrucción leída, y finalmente ejecuta la instrucción. Cuando se ha completado este ciclo, puede comenzar otro nuevo.

En la arquitectura Harvard, los buses de dirección, de datos y de control para acceder a la memoria de programa son diferentes de los de acceso a los datos (Fig. 1.19). Las instrucciones son recibidas por la CPU a través de un bus de datos reservado para las instrucciones, diferente del bus de datos reservado para los datos. En este caso, la CPU es capaz de leer instrucciones de la memoria de programa y de ejecutar simultáneamente instrucciones ya leídas. Esta estructura de acceso a datos y ejecución de instrucciones en paralelo se denomina en terminología anglosajona *pipelining* (en castellano «segmentación»). Se simultanea el acceso a las memorias de programa y de dato, lo que es posible gracias a la duplicidad de recursos o buses que existen en el sistema. Este tipo de estructuras es más complejo en el *hardware* que las de tipo von Neumann, pero permiten acelerar el tiempo efectivo de ejecución de la instrucción: la CPU es capaz de recoger instrucciones mientras realiza la manipulación de los datos asociados a otras instrucciones leídas previamente. La ejecución real de una instrucción se realiza en las mismas fases que en el caso de las arquitecturas von Neumann y en un número de ciclos de reloj similar, pero mientras una instrucción se está ejecu-

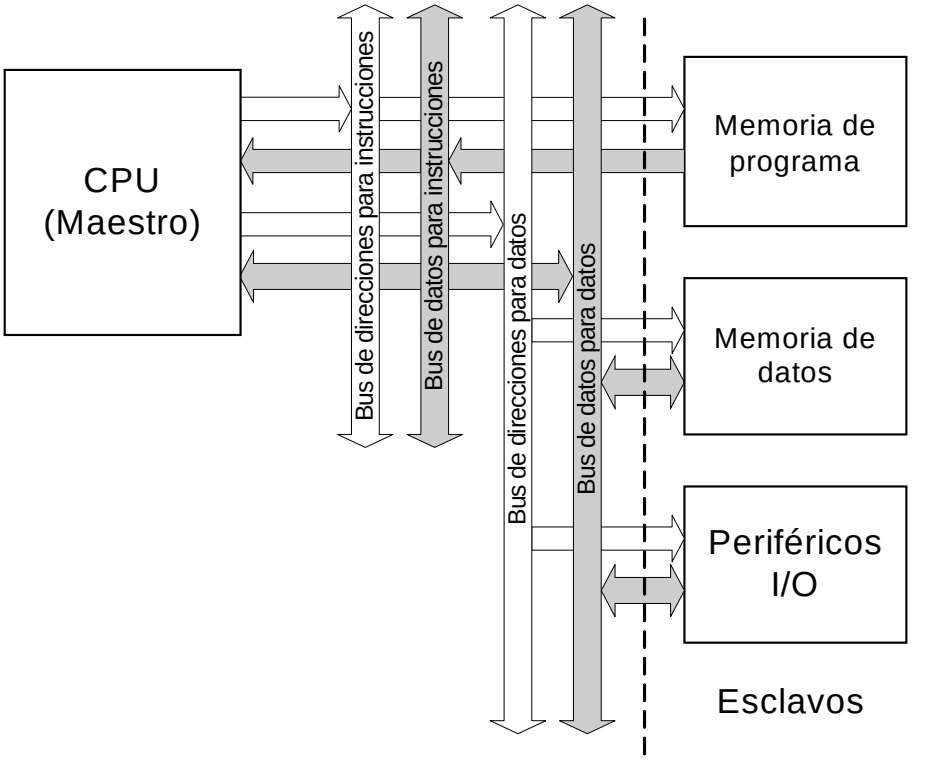


Figura 1.19. *Arquitectura de tipo Harvard asociada a un microprocesador (el bus de control del sistema no se ha representado para simplificar el esquema).*

tando, la CPU está recogiendo otras instrucciones y preparándose para ejecutarlas, lo que disminuye el tiempo real o efectivo de ejecución de cada instrucción.

Ubicación de un periférico en el rango de direcciones de la CPU

La comunicación del periférico con la CPU se realiza normalmente a través de los buses del sistema microprocesador. Cuando la CPU genera una orden de lectura a un periférico de su sistema microprocesador, debe provocar el acceso en lectura al contenido de uno de los registros internos del periférico. Por otro lado, al generar la CPU una orden de escritura al periférico, ésta debe acceder a uno de los registros internos del periférico para almacenar en él un dato. Para que la CPU realice estas funciones es necesario:

1. Que la CPU seleccione en modo lectura o escritura al periférico (active la señal de selección o habilitación del periférico, $\overline{CS}$ o $\overline{CE}$, y especifique el tipo de acceso que debe atender el periférico, señales $R/\overline{W}$, $\overline{OE}$ o $\overline{WE}$). El diseñador del sistema microprocesador utiliza, para activar las líneas antes comentadas de entrada al periférico, las señales de salida de la CPU asociadas al bus de control del sistema como entradas a una pequeña lógica combinacional, denominada lógica de selección del periférico. La salida de la lógica de selección será la que imponga los valores lógicos necesarios, en las entradas de selección y tipo de acceso del periférico, para que la CPU lea o escriba un registro interno del periférico.
2. Que la CPU distinga entre el acceso a uno u otro registro interno del periférico. Para ello es necesario que las líneas asociadas al bus de direcciones del sistema microprocesador se conecten a las líneas de dirección de entrada al periférico. De esta forma, el valor que imponga la CPU en las líneas del bus de direcciones determina el registro interno del periférico accedido.

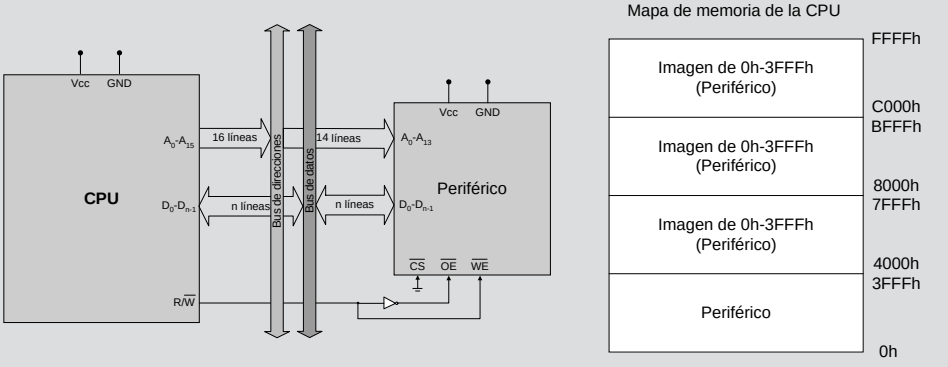
El rango de direcciones de memoria de la CPU asignado a cada dispositivo debe ser mayor o igual al número de registros internos que éste tiene. No importa que sea mayor, ya que las direcciones sobrantes no se utilizarán nunca y el programa que ejecuta la CPU no suele contener órdenes de lectura/escritura a dichas direcciones sobrantes. Ahora bien, si el rango de direcciones de memoria que se reserva a un dispositivo es inferior al número de datos internos que almacena o puede almacenar, estaríamos infrautilizándolo, puesto que la CPU no podría acceder a todos los registros internos de dicho periférico. Al rango de direcciones de memoria de la CPU que se encuentra asignado a los diferentes periféricos del sistema microprocesador se le suele denominar mapa de memoria del sistema microprocesador (Ejemplo 1).

Ejemplo 1. Ubicación de un periférico en el rango de direcciones de una CPU genérica

Supongamos una CPU con un bus de direcciones de 16 líneas (rango de direcciones de memoria de 2^{16}, es decir, permite accesos a 64 K, o 65.536 datos e instrucciones, ubicados

desde la posición 0h hasta la FFFFh), una señal de salida denominada $R/\overline{W}$ que indica que se está realizando un acceso a algún periférico externo en lectura o escritura y un bus de datos de *n* bits. Supongamos, además, que a esta CPU se conecta un periférico que dispone de 16 K registros internos, con 14 líneas de dirección para distinguir entre ellos, según se indica en la figura inferior.

Obsérvese que las señales asociadas al bus de direcciones de la CPU se utilizan para seleccionar el registro interno del periférico al que se desea acceder, conectándose a las líneas de dirección de entrada al periférico. En este ejemplo, la señal $\overline{CS}$ de entrada al periférico está conectada a GND (tierra del circuito de alimentación del sistema electrónico), lo que indica que el periférico siempre estará habilitado y se podrá acceder a él en lectura o escritura. El rango de direcciones del periférico es inferior al de la CPU, generándose el mapa de memoria que se indica en la figura. Esto será habitual en los sistemas microprocesadores, ya que no importa que la zona reservada al periférico en el mapa de memoria de una CPU sea mayor que el rango de direcciones del propio periférico, puesto que las direcciones sobrantes no se utilizarán nunca (el programa no suele contener órdenes de lectura/escritura a dichas direcciones sobrantes). Nótese que si el rango de direcciones de memoria que se reserva al dispositivo es inferior al número de datos internos que almacena o puede almacenar, estaríamos infrautilizando al periférico, ya que la CPU no podría acceder a todos sus registros internos.



Organización de la memoria y arbitraje del bus de datos

La existencia de buses y periféricos con la estructura descrita anteriormente hace que la arquitectura del sistema microprocesador sea abierta, evitándose la necesidad de que la CPU disponga de un conjunto de líneas dedicadas a cada periférico de su sistema microprocesador. Es decir, en cualquier momento se puede añadir un nuevo dispositivo al sistema con sólo conectarlo a los buses, aunque el hecho de emplear un único conjunto de líneas para conectar a la CPU con los dispositivos externos del sistema microprocesador establece ciertas limitaciones en el número de dispositivos con los que ésta se puede comunicar y en la forma en que se realiza dicha comunicación.

Básicamente, al diseñar un sistema microprocesador se deben seguir principalmente dos reglas:

1. Hay que evitar conflictos al conectar periféricos a la CPU a través del bus de datos.
2. Es necesario cumplir la temporización que indica el fabricante en los accesos que realiza la CPU.

La primera regla busca evitar cortocircuitos en las conexiones eléctricas asociadas a los buses del sistema, lo que implica que dos dispositivos no pueden, bajo ningún concepto, activar en el mismo instante de tiempo la misma línea de un bus. Esta limitación está habitualmente relacionada con el bus de datos y el diseño del mapa de memoria del sistema microprocesador. No afecta, sin embargo, al bus de dirección, en el que en principio sólo escribe la CPU, ni al de control, por motivos similares.

Cuando la CPU realiza un acceso en lectura o escritura a un periférico externo, lo primero que hace es seleccionar el tipo de acceso (lectura o escritura) y la dirección a la que desea acceder (establece un valor fijo en el bus de direcciones de salida de la CPU que se mantiene todo el tiempo que dura el acceso). La orden de escritura no producirá cortocircuitos, puesto que sólo un dispositivo, la CPU, pone el dato en el bus de datos, mientras que el resto de dispositivos escucha (normalmente la CPU sólo deseará que un periférico tome el dato). Sin embargo, en una orden de lectura hay grave peligro de cortocircuito. Si dos dispositivos se dan por aludidos ante la orden de lectura e intentan poner un dato en el bus de datos para que lo recoja la CPU, probablemente se produciría un cortocircuito en las salidas de los dispositivos que se conectan al bus, provocando que alguno de ellos se queme.

Para evitar esta situación, es necesario diseñar el sistema microprocesador de forma que cuando la CPU acceda al exterior sólo se comunique con un periférico, o lo que es lo mismo, asignando diferentes zonas del rango de direcciones de memoria del sistema microprocesador a cada periférico. Esto será posible gracias a las señales de selección de los periféricos, $\overline{CS}$ o $\overline{CE}$: mientras no se seleccione el dispositivo, éste mantiene las señales de salida que se conectan al bus de datos en alta impedancia y la CPU no puede hablar con él.

Dado que la CPU se diseña e implementa con un número limitado de líneas dedicadas a gestionar la comunicación con los periféricos externos, es habitual que el diseñador del sistema microprocesador necesite una circuitería externa a la CPU para diseñar la lógica combinacional que activa la señal de selección de cada periférico[8]. Dicha lógica tendrá como entradas algunas de las líneas del bus de control y de las líneas más significativas del bus de direcciones, generará como salidas las señales que se conectan con los selectores de los periféricos externos del sistema microprocesador y será la encargada de impedir los cortocircuitos.

---

[8] Normalmente se implementará la descodificación en un dispositivo lógico programable simple (PLD) o mediante dispositivos digitales de pequeña y mediana escala de integración (puertas lógicas, descodificadores, etc.).

Reglas de asignación de rango de direcciones de memoria de una CPU

Se describen, en este inciso, una serie de reglas básicas que pueden ayudar a la definición del mapa de memoria de un sistema microprocesador. Las reglas expuestas a continuación no son de obligado cumplimiento, ni siquiera se puede asegurar que, siguiéndolas, se llegará a definir el mejor mapa de memoria del sistema microprocesador, pero al menos pueden simplificar y estandarizar dicha tarea.

1. Ubicar los periféricos en zonas de memoria de la CPU de tamaño superior, o como mucho de igual tamaño, al del propio periférico. Por ejemplo, no se deberá ubicar un periférico de memoria RAM y tamaño 64 K en una zona de memoria de la CPU de tamaño 32 K, dado que al escoger ese criterio de diseño estaríamos renunciando a que la CPU pueda acceder a la mitad del periférico (o al menos a que el acceso a todo el periférico sea simple).
2. Relacionado con el punto anterior. Intentar que la zona de memoria reservada al periférico sea una única zona. Por ejemplo, no ubicar un periférico de 32 K en dos zonas no consecutivas de memoria de la CPU de tamaño 16 K. De esta forma, estaríamos simplificando la elección del conjunto de líneas del bus de direcciones que emplea la CPU para indicar al periférico el registro al que desea acceder, y que por tanto se conectan a las líneas de dirección de entrada del periférico.
3. Dividir el rango de direcciones de memoria de la CPU en tantas partes como periféricos se vayan a ubicar en el mapa de memoria y asignar cada zona a un periférico distinto. Si se prevén futuras ampliaciones del sistema, es conveniente dividir el rango de direcciones en alguna zona más de las estrictamente necesarias y dejar esa zona libre para los futuros periféricos del sistema. En todo caso, la división del rango de direcciones de memoria de la CPU siempre debe hacerse siguiendo las mismas pautas:
   - Si se dispone de 2^n periféricos, dividir el rango de direcciones de memoria en 2^n partes iguales, siempre que se cumplan los requisitos de diseño de los apartados 1 y 2. Por ejemplo, un sistema microprocesador basado en una CPU con 16 líneas de dirección y cuatro periféricos de tamaño menor de 16 K requieren una división del rango de direcciones en cuatro partes iguales de tamaño 16 K. Las líneas de dirección A_{15} y A_{14} de la CPU distinguirán entre accesos a uno u otro periférico del sistema.
   - Si se dispone de un número m de periféricos tal que $2^{n-1} < m < 2^n$, dividir el rango de direcciones de memoria en 2^n partes iguales siempre que se cumplan los requisitos de diseño de los apartados 1 y 2. En este caso, quedarán zonas de memoria en el diseño sin asignar a periféricos, disponibles para futuras ampliaciones. Con ello se simplifica notablemente la lógica de selección de los periféricos: se utilizan pocas líneas y siempre las más significativas del bus de dirección de la CPU. Se evitan, además, conflictos en el mapa de memoria del sistema microprocesador.
4. Excepción a la regla 3, por incumplimiento de alguna regla anterior (principalmente la regla 1). Si alguno o algunos de los periféricos tienen un tamaño tal que al hacer la división del rango de direcciones de memoria de la CPU en 2^n partes

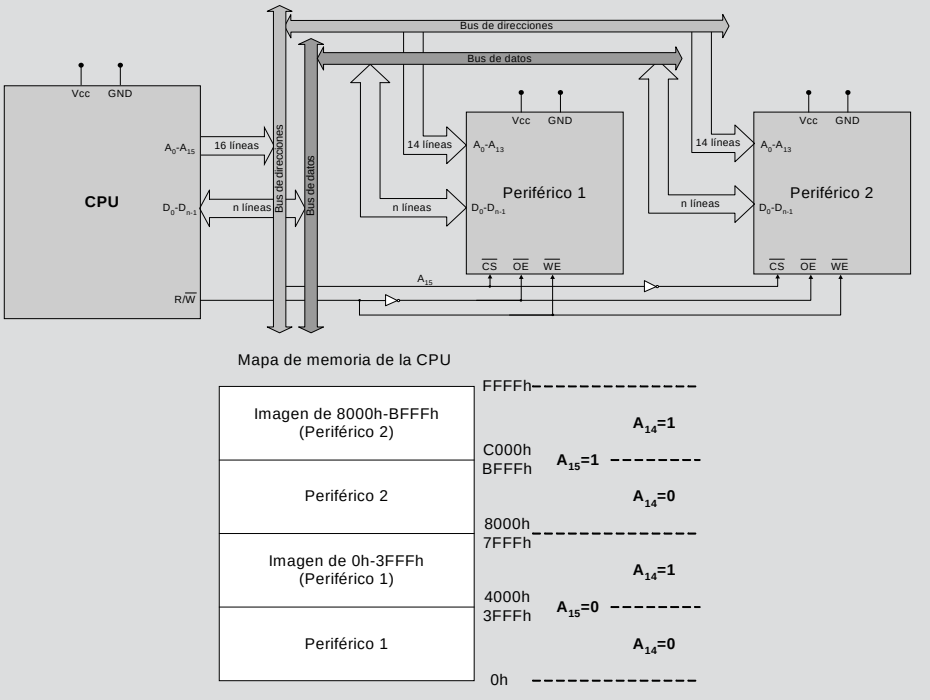
iguales el periférico no «cabe» en las zonas resultantes, la división del rango de direcciones de la CPU deberá de realizarse por 2 de forma iterativa, ubicándose en primer lugar aquellos periféricos de mayor tamaño. La división continúa con las zonas libres, no asociadas a periféricos, hasta asignar una zona a cada periférico del sistema. Por ejemplo, el diseño de un mapa de memoria de una CPU con 16 líneas de dirección (tamaño 64 K) y tres periféricos externos de tamaño 32, 16 y 4 K no se haría dividiendo el rango de direcciones de la CPU en cuatro partes iguales y asignándole una parte a cada periférico (dejando una zona libre), dado que las zonas resultantes de la división tienen un tamaño de 16 K, inferior al del periférico de 32 K. En su lugar, el rango de direcciones de la CPU se divide inicialmente en dos partes, cada una de tamaño 32 K. Una de estas zonas se asignaría al periférico de tamaño 32 K, y la zona libre resultante se volvería a dividir en dos partes, cada una de 16 K. De las dos partes resultantes de la segunda división, una sería asignada al periférico de tamaño 16 K y la otra se asignaría al periférico de tamaño 4 K, o nuevamente, se dividiría en otras dos partes, una de las cuales se asignaría al periférico de tamaño 4 K y la otra se dejaría disponible para futuras ampliaciones del sistema digital.

Por tanto, y a modo de resumen, aunque aparezcan muchos dispositivos externos conectados a los buses, la CPU sólo debe comunicarse con uno de ellos cada vez que realice una lectura o escritura, para evitar cortocircuitos en las conexiones eléctricas de los buses del sistema; se evitan los conflictos si no se seleccionan simultáneamente varios periféricos y si en la misma zona del rango de direcciones de memoria de la CPU no aparecen dos o más dispositivos. Es necesario que sólo un dispositivo sea aludido por la CPU cada vez que ésta realiza un acceso por el bus de datos, conectándose dicho periférico al bus de datos mientras el resto mantiene sus salidas al bus en estado de alta impedancia, lo que se consigue gracias a la denominada señal de selección del periférico. Dado que el número de señales que emplea la CPU para seleccionar un periférico externo es muy limitado, habrá que implementar un circuito combinacional extra, externo a la CPU, que ataque a la línea de entrada de selección de cada periférico del sistema microprocesador, de forma que se impida el conflicto.

Ejemplo 2. Ubicación de dos periféricos en el rango de direcciones de una CPU genérica

Supongamos una CPU como la del Ejemplo 1 y que a esta CPU se conectan dos periféricos similares a los de dicho ejemplo, según se indica en la figura inferior.

Se puede observar que la línea más significativa del bus de direcciones, A_{15}, divide el rango de direcciones de memoria de la CPU en dos mitades: la superior, en la que A_{15} toma el valor 1, y la inferior, en la que toma el valor 0. A su vez, la línea A_{14} divide cada una de estas dos mitades en otras dos submitades, en las que esta línea toma el valor 1 y 0, respectivamente, y así sucesivamente. Asignando un rango de direcciones único a cada dispositivo se tiene la garantía de que sólo ese dispositivo es el que se activa cuando la CPU da una orden de lectura o escritura a una dirección comprendida dentro de dicho



rango. Con esta premisa se suele diseñar la función lógica que, a partir de las señales del bus de control y de dirección, ataca a las señales de selección de los periféricos. En este ejemplo $\overline{CS}_1 = A_{15}$ y $\overline{CS}_2 = NOT(A_{15})$: el periférico 1 es seleccionado cuando la CPU realiza accesos a direcciones en las que A_{15} vale 0, mientras que el periférico 2 lo es cuando A_{15} vale 1. Obsérvese que, como en el Ejemplo 1, la zona reservada a cada periférico en el mapa de memoria de la CPU es mayor que el rango de direcciones del propio periférico. En este caso, a cada periférico se le reserva una zona en el rango de direcciones de la CPU de tamaño el doble del número de registros internos existentes.

La segunda limitación está relacionada con la comunicación entre la CPU y sus periféricos en el sistema microprocesador, y con el hecho de que la CPU consiga leer el dato o instrucción que requiere del periférico externo o escribir el dato que desea almacenar en él. Se ha comentado que es la CPU la que controla, en todo momento, la comunicación (la transferencia de los datos es iniciada y terminada por el maestro del sistema microprocesador), por lo que para que ésta sea fiable, es preciso que al periférico, que funciona como esclavo, le dé tiempo de responder a la CPU.

Para que la CPU consiga leer el dato o instrucción que requiere del periférico externo o escribir el dato que desea almacenar en él, es necesario que en el sistema digital se garantice que «el tiempo que transcurre desde que inicia el acceso la CPU hasta que lo termina ($T_{acceso\text{-}CPU}$) sea mayor que el tiempo que tarda en responder el periférico ($T_{respuesta\text{-}periférico}$)». Es decir, si la CPU desea leer un dato o una instrucción de

un periférico externo, y para ello comienza un acceso externo a través de los buses del sistema, al periférico le tiene que dar tiempo a poner el dato requerido en el bus de datos antes de que la CPU concluya el acceso al exterior, y si la CPU quiere escribir un dato en un periférico externo, debe mantener el dato en el bus de datos el tiempo que al periférico le haga falta para almacenarlo en su interior. En caso contrario, la CPU no leería la instrucción o el dato deseado o no almacenaría el dato en el periférico externo, por lo que el sistema digital no funcionaría, o al menos no lo haría correctamente. La segunda limitación asociada al diseño de un sistema microprocesador se puede resumir en el cumplimiento de la siguiente desigualdad: $T_{acceso\text{-}CPU} > T_{respuesta\text{-}periférico}$.

Un cronograma o diagrama de tiempo es un gráfico que representa en el tiempo cómo se realiza un acceso en lectura o escritura a cualquier dispositivo (CPU o periférico). Su correcta interpretación conlleva el diseño adecuado del sistema microprocesador desde el punto de vista de los tiempos de acceso en la estructura *maestro-esclavo* ($T_{acceso\text{-}CPU} > T_{respuesta\text{-}periférico}$).

El diagrama de tiempo asociado a una CPU representa cómo se inicia y concluye la comunicación e indica el tiempo de acceso de la CPU. Asociado a un periférico, el diagrama de tiempo establece el tiempo de acceso al periférico, que en condiciones ideales coincide con el tiempo de respuesta del periférico. La diferencia entre tiempo de respuesta y de acceso en un periférico depende del tiempo de retraso ($T_{retraso}$) asociado a la lógica de selección del propio dispositivo de la siguiente forma: $T_{respuesta\text{-}periférico} = T_{acceso\text{-}periférico} + T_{retraso}$. La CPU inicia el acceso, pero el periférico al que desea acceder no se entera del comienzo de dicho acceso hasta unos pocos nanosegundos después, $T_{retraso}$, debido a que todo periférico tiene asociada una lógica de selección que, a su vez, lleva asociado un tiempo de retardo mayor de 0 ns.

En las Figuras 1.20 y 1.21 se describe cómo debe ser el acceso en lectura y escritura de una CPU a un periférico de su sistema microprocesador. Se pueden observar

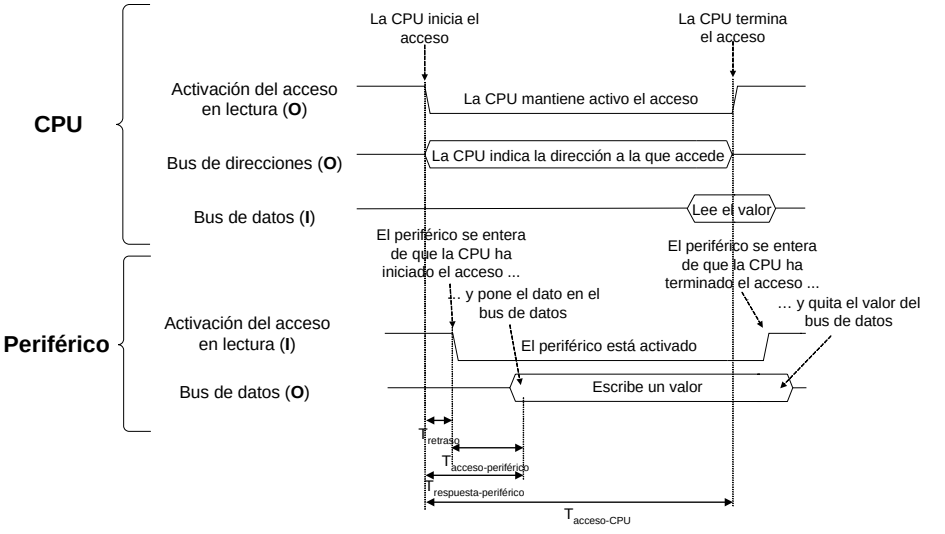


Figura 1.20. *Ejemplo de cómo debe realizar una CPU un acceso en lectura a un periférico de su sistema microprocesador.*

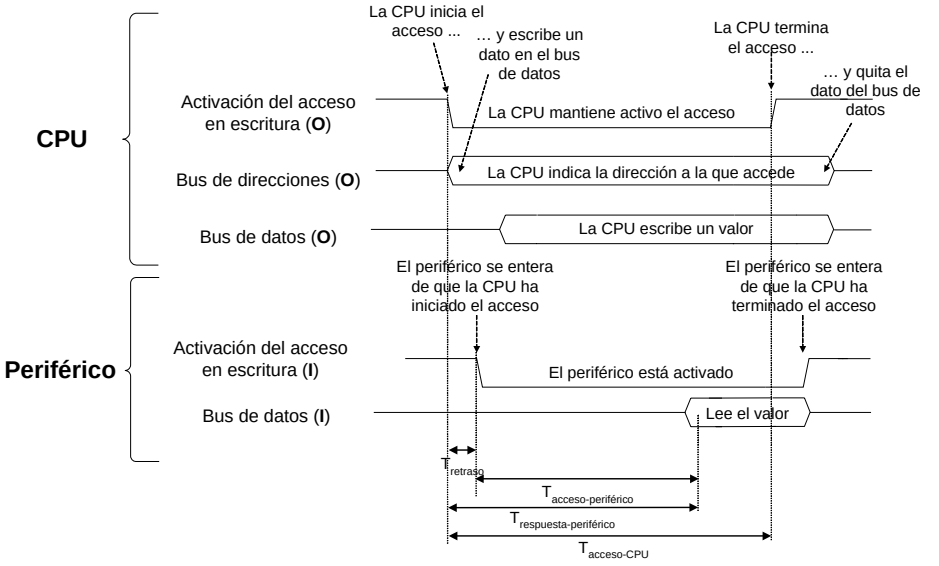


Figura 1.21. *Ejemplo de cómo debe realizar una CPU un acceso en escritura a un periférico de su sistema microprocesador.*

los diagramas de tiempo de la CPU y del periférico. Las señales de salida se indican con la etiqueta (O) y las de entrada con (I). La CPU realiza bien el acceso en lectura si al periférico le da tiempo a responderle imponiendo un valor en el bus de datos antes de que lea el contenido del bus. El acceso en escritura de la CPU al periférico se realiza sin problemas, siempre y cuando el valor que ha impuesto en el bus de datos la CPU no sea retirado antes de que el periférico lo haya recogido.

Las CPU pequeñas en prestaciones y potencia de cálculo, como son la mayoría de las CPU que forman parte de los sistemas microprocesadores de tipo von Neumann, no suelen controlar el tiempo de acceso de la CPU a los periféricos externos, y el fabricante de la CPU impone un tiempo de acceso fijo. Esto es debido a que los ciclos máquina[9] y de bus[10] de la CPU suelen ser lo suficientemente grandes como para que la mayoría de los periféricos puedan ser accedidos sin problemas de tiempo.

El caso de las CPU de elevada potencia de cálculo, la mayoría de las CPU que forman parte de los sistemas microprocesadores de tipo Harvard, es diferente. El ciclo máquina de estas CPU es muy pequeño y no todos los periféricos que se pueden encontrar en el mercado presentan unos tiempos de respuesta tan bajos. Así, el tiempo de respuesta y estabilización de la salida en una función lógica elemental —OR, AND...—, asociada a un dispositivo diseñado con tecnología LS —TTL Schottky de bajo consu-

[9] El ciclo máquina hace referencia al tiempo efectivo que tarda la CPU en ejecutar una instrucción, que es un múltiplo entero de la frecuencia del oscilador de entrada a la CPU.

[10] El ciclo de bus se refiere al tiempo mínimo que tarda la CPU en realizar un acceso a través de los buses del sistema microprocesador.

mo—, está en torno a los 10 ns, muy cerca del ciclo máquina de una CPU de las que estamos hablando que coincide, a su vez, con los tiempos de acceso de la CPU al exterior, que pueden llegar a ser inferiores a los 25 ns.

Ello lleva a plantearnos si es posible emplear cualquier periférico externo en un sistema digital basado en una CPU de elevada potencia de cálculo, y si es el caso, cómo se puede gestionar el acceso de la CPU al periférico. Impedir el empleo de un periférico, por lento que éste sea, no deja de ser una limitación para el diseñador de un sistema microprocesador. Por ello, los fabricantes diseñan las CPU de la manera más versátil posible, lo que implica permitir el empleo del mayor número de periféricos externos. La respuesta, por tanto, a la primera de las cuestiones anteriores es que es posible emplear cualquier periférico externo en un sistema digital basado en una CPU de elevada potencia de cálculo. Obviamente, para facilitar esta característica es necesario que el fabricante permita al usuario controlar el tiempo que tarda la CPU en realizar el acceso al exterior.

En un sistema microprocesador, los estados de espera representan la manera que tiene el usuario de controlar el tiempo de acceso de una CPU a un periférico externo. El acceso de la CPU al periférico se puede ralentizar cierto tiempo, normalmente un periodo de tiempo equivalente a un número entero múltiplo del ciclo de bus de la CPU. A este entero se le denomina número de estados de espera del acceso. Esta característica permitirá a la CPU acceder a dispositivos externos con tiempos de acceso altos, superiores a los que realmente necesita el *maestro* para acceder a un *esclavo*.

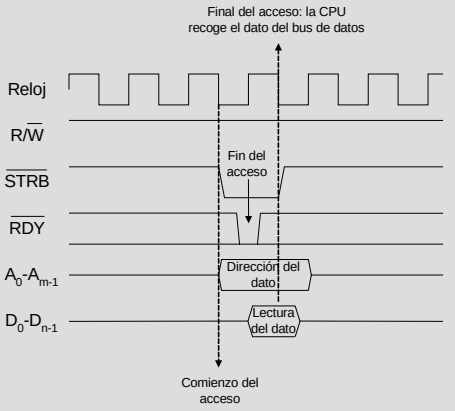
El número de estados de espera en un acceso se controla mediante una señal de entrada a la CPU. Poco antes de completar el acceso, la CPU comprueba el estado de esta señal de control, y si no está activada, no lo completa, retrasándolo hasta una posterior comprobación de la señal, realizada un ciclo de bus después de la comprobación previa. El proceso anterior se repite hasta que, en una comprobación, la CPU detecta que la señal está activada, completando entonces el acceso.

Temporización de los accesos realizados por una CPU genérica

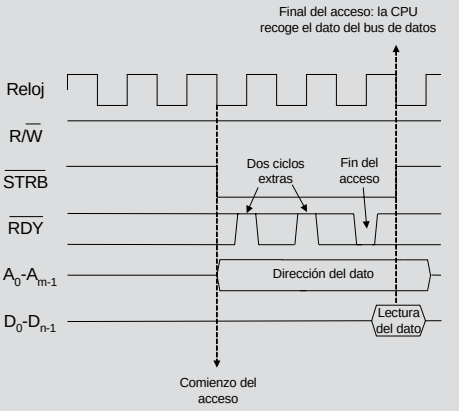
Supongamos un sistema microprocesador que dispone de una CPU a la que se le puede programar el tiempo de acceso a los periféricos externos.

En las figuras de este ejemplo se muestra un diagrama de tiempo típico de una CPU que accede al exterior con cero estados de espera (el acceso que realiza es el más rápido posible) y con dos estados de espera (se genera a la CPU una señal para ralentizar el acceso al exterior dos ciclos máquina extra). En ambos casos, la CPU es tal que:

- La señal $\overline{RDY}$ es de entrada a la CPU y sirve para indicarle a ésta si debe o no concluir el acceso al exterior (en este caso se ha considerado que cuando vale cero, la CPU debe concluir el acceso).
- La señal que se denomina *reloj* es de entrada a la CPU y genera el ciclo básico para la temporización del acceso.
- La señal $\overline{STRB}$ es de salida de la CPU y sirve para indicar que está realizando un acceso al exterior.
- La señal $R/\overline{W}$ es de salida de la CPU. Con ella se distingue entre acceso en lectura y escritura.
- El conjunto de líneas $A_0 - A_{m-1}$ y $D_0 - D_{n-1}$ representan el bus de direcciones y de datos, respectivamente.



Acceso en lectura con cero estados de espera.



Acceso en lectura con dos estados de espera.

La jerarquía de memoria: memoria caché

El sistema electrónico digital que hemos denominado sistema microprocesador dispone de una CPU, que gestiona la transferencia de información, y diversos periféricos, entre los que destacan los dispositivos de almacenamiento de instrucciones y datos (periféricos de memoria). La transferencia de información entre la CPU y los periféricos de memoria determina la capacidad de procesamiento y cálculo, así como el rendimiento de todo el sistema.

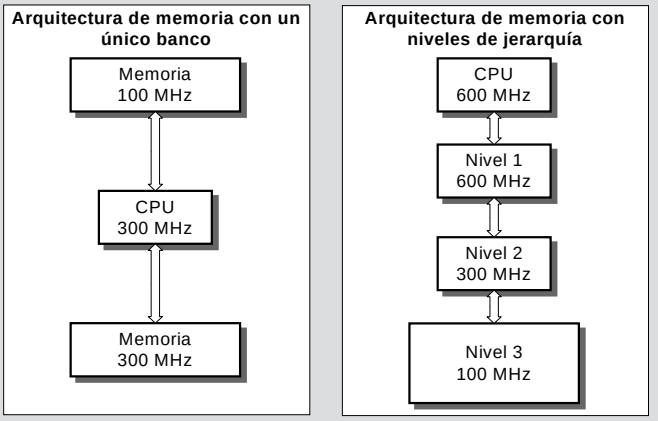
En la práctica, la tecnología impone ciertas restricciones a la CPU y a los dispositivos de memoria. Así, cuanto más rápido es el acceso a un dispositivo de memoria, menor es la capacidad de almacenamiento del mismo, y viceversa. Por su parte, la CPU ha ido evolucionando hacia sistemas más rápidos, más precisos, capaces de realizar tareas más complejas y, por tanto, con mayores necesidades en cuanto al almacenamiento de datos e instrucciones. En las modernas aplicaciones en las que se emplean los sistemas microprocesadores, siempre es preciso contar con una elevada capacidad de memoria integrada dentro del propio sistema, aunque estos dispositivos de almacenamiento no sean capaces de trabajar a la misma velocidad que puede llegar a alcanzar una CPU. El resultado es la aparición de una clara brecha entre la velocidad de procesamiento y la velocidad que se puede alcanzar en los accesos a las memorias incluidas en los sistemas microprocesadores. Esto no significa que no existan también memorias muy rápidas que se podrían incorporar en el sistema, pero son mucho más caras y de menor capacidad de almacenamiento que las memorias más lentas.

Desde el punto de vista del diseñador del sistema microprocesador, que busca al menor coste posible las mayores prestaciones en capacidad de procesamiento de datos, la solución a este dilema pasa por organizar la memoria de forma jerárquica, por niveles, apareciendo el concepto de «jerarquía de memoria». En ella, una memoria pequeña y rápida se encuentra próxima a la CPU, ofreciéndole un acceso a la máxima velocidad posible. Los siguientes niveles de memoria, cada vez con mayores capacidades de alma-

cenamiento pero con velocidades de acceso más bajas, se van sucesivamente alejando de la CPU, provocando accesos aún más lentos de los que podrían ofrecer. Cuando la CPU accede a una dirección de una memoria con una elevada capacidad de almacenamiento (y por tanto más baja en la jerarquía), ésta se asigna en realidad a una posición de una memoria más pequeña y rápida, perteneciente a un nivel más alto de jerarquía. Para ello será necesario que, mediante algún método, las memorias de mayor velocidad de acceso funcionen como dispositivos de almacenamiento temporal de información (instrucciones o datos), que es, a su vez, copia de la información ubicada en las memorias de menores velocidades de acceso y mayores capacidades de almacenamiento. Si eso se consigue, el tiempo de acceso medio a la memoria del sistema se aproximará más al tiempo de acceso de las memorias más rápidas que al de las memorias más lentas. A estas memorias del nivel más alto de la jerarquía de memoria de la CPU se les denomina memorias caché.

Comparación entre sistemas microprocesadores con un único banco de memoria (sin jerarquía de memoria) y con una jerarquía de memoria de tres niveles

La figura del ejemplo muestra una CPU sin jerarquía de memoria, con un único banco de memoria, frente a otra que posee tres niveles de memoria, siguiendo una estructura jerárquica.



Cuando existe un único banco de memoria, figura de la izquierda, la CPU normalmente accede a ella a la máxima velocidad (en el ejemplo, memoria de velocidad de acceso de 300 MHz), sin que exista ningún ciclo de espera en el acceso. Ahora bien, si se duplicase la velocidad de esta CPU hasta los 600 MHz, y si se siguiese usando un único banco de memoria, existiría una penalización de un ciclo de espera en cada acceso. Esto llega a ser particularmente crítico en aplicaciones que fuerzan a la CPU a realizar múltiples accesos a la memoria del sistema, en cuyo caso la velocidad de procesamiento de la CPU disminuye hasta aproximarse a la velocidad de acceso a la memoria. Para aprovechar las nuevas prestaciones de la CPU, manteniendo una arquitectura de memoria con un único banco, es necesario duplicar la velocidad de acceso a la memoria, lo que nos llevaría a un coste prohibitivo si se desease conservar su misma capacidad de almacenamiento. Frente a esta filosofía de diseño, las razones económicas ya comentadas han

terminado por imponer la organización jerárquica de memoria en dos o más niveles, mostrada en la figura de la derecha. Típicamente, los niveles más altos están formados por dispositivos de memoria de elevada velocidad de acceso y baja capacidad (memoria caché asociada al nivel 1 de la jerarquía de memoria). Cuando la CPU intenta acceder a una posición de memoria que no se encuentra en los niveles más altos y rápidos de la jerarquía, la busca en los niveles más bajos (niveles 2 y 3 de la jerarquía), y al tiempo que accede a la información, un controlador asociado a la propia memoria caché aprovecha el acceso para almacenarla en el nivel más alto. Esto ocurriría, por ejemplo, en la ejecución por parte de la CPU de la primera iteración de las instrucciones de un bucle; las siguientes iteraciones que realiza la CPU encuentran las instrucciones en el nivel más alto y rápido de la jerarquía. Mediante esta técnica de diseño del sistema microprocesador se consigue, sin aumentar en exceso el coste del sistema, aumentar la velocidad de procesamiento de la CPU, aproximándola a la máxima posible, a pesar de que los dispositivos de memoria de mayor capacidad de almacenamiento conservan unas velocidades de acceso menores a las de procesamiento de la CPU.

Los principios básicos de funcionamiento de una memoria caché son los denominados principios de localidad. Estos principios asumen el hecho de que cuando se accede a una posición de memoria existe una elevada probabilidad de que esa u otra posición cercana serán accedidas en un corto intervalo de tiempo. El acceso a una misma posición de memoria en un corto intervalo de tiempo es lo que se denomina localidad temporal, mientras que el acceso a posiciones cercanas es la denominada localidad espacial. Tanto la localidad espacial como la temporal se manifiestan de manera notable durante la ejecución de un programa: las instrucciones se ejecutan normalmente de manera secuencial o en bucles de instrucciones próximas, y los datos se reutilizan o bien se accede a otros datos que se encuentran en posiciones próximas en un pequeño intervalo de tiempo. La primera vez que se accede a una instrucción o dato se lee de una memoria de un nivel jerárquico bajo. Durante ese acceso, el controlador de la memoria caché aprovecha para guardar una copia de la instrucción o dato recogido, de manera que las siguientes veces que la CPU intente acceder a ellas se encontrarán ya en la caché de programa o de datos, respectivamente. En conclusión, la organización jerárquica de la memoria aprovecha el principio de localidad espacial y temporal para guardar los datos e instrucciones de mayor uso en memorias rápidas que ocupan los niveles superiores en la jerarquía.

Ejemplo de aprovechamiento de la localidad: filtro FIR de orden seis

El ejemplo siguiente ilustra los principios de localidad espacial y temporal. Consideremos un filtro FIR de orden seis, donde $h[i]$ son los coeficientes del filtro y $x[i]$ son las muestras entrantes. Los cálculos involucrados en las dos primeras salidas, $y[0]$ e $y[1]$, serían:

$$y[0] = h[0]x[0] + h[1]x[1] + \cdots + h[5]x[5]$$

$$y[1] = h[0]x[1] + h[1]x[2] + \cdots + h[5]x[6]$$

Es decir, para computar una salida es necesario leer las seis muestras anteriores de un búfer de memoria y los seis coeficientes de otro búfer. Cuando se accede a la muestra $x[0]$, algunos controladores de caché no recogen únicamente ese dato, sino un conjunto o bloque de datos de un cierto número de posiciones que siguen a la primera. Este rango de direcciones constituye lo que se llama una línea de la caché. El motivo es aprovechar la localidad espacial, ya que las muestras entrantes ocuparán posiciones consecutivas en el búfer de memoria. Lo mismo ocurriría con el acceso a los coeficientes. Para ello, los diseñadores de los sistemas microprocesadores modernos integran buses de datos de tamaño superior al de los propios datos. Cuando se vaya a calcular la salida $y[1]$, los coeficientes y cinco de las muestras entrantes siguen siendo las mismas, en tanto que sólo la muestra $x[6]$ es nueva. En este caso, existe una alta localidad espacial y temporal.

Al acceder la CPU a un nivel de memoria dentro de la jerarquía para buscar un dato, puede ocurrir que se encuentre o no. En el primer caso tendríamos un acierto y en el segundo un fallo, que nos llevaría a buscar el dato en un nivel inferior. La frecuencia de aciertos en los accesos a un determinado nivel de memoria se define como la fracción de accesos a dicha memoria con éxito, mientras que la frecuencia de fallos es la fracción de accesos a la memoria sin éxito. El tiempo de acierto viene dado por la suma del tiempo necesario para determinar que ha habido un acierto más el tiempo consumido en acceder al dato. La penalización de un fallo viene dada por el tiempo necesario para traer el dato o bloque de datos del nivel inferior al superior, lo que supone un tiempo de acceso más un tiempo de transferencia entre niveles. La estrategia de diseño de una jerarquía de memoria debe ser aquella que permita minimizar la penalización del fallo y maximizar los aciertos.

En conclusión, el nombre de caché, que se utilizó inicialmente para designar el nivel de memoria que existía entre la CPU y la memoria principal en los primeros procesadores, se sigue empleando hoy en día para designar cualquier dispositivo de almacenamiento de código o datos que haga uso de los principios de localidad en los accesos. Así, las memorias caché ocupan los niveles más altos dentro de la jerarquía de memoria y son las más cercanas a la CPU. Se trata, por tanto, de dispositivos de un acceso rápido, baja capacidad de almacenamiento y cuya utilización se ha convertido en indispensable para la mayoría de computadores y procesadores de un cierto nivel de prestaciones, que pueden incluso contar con varias de estas memorias.

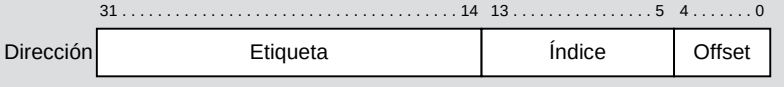
Dado que la caché tiene un tamaño reducido, mucha información tendrá que ser buscada en otros niveles inferiores de memoria. Si la caché se encuentra llena, un bloque de información (que contiene el dato que provocó el fallo y cuyo tamaño debe coincidir con el de la línea de la memoria caché) se cargará en la caché sobre una línea previamente ocupada de la misma. Se produce entonces un reemplazo de la información almacenada en la caché. Dado que el acceso a niveles inferiores siempre supone una penalización en el tiempo de acceso, es muy importante diseñar una estrategia óptima de reemplazo de líneas de caché que permita maximizar la tasa de acierto. Asimismo, es importante minimizar la penalización en tiempo que provoca el fallo. Este objetivo se conseguirá mediante una organización de la caché que permita determinar rápidamente si ha existido fallo, y en caso afirmativo, el acceso rápido al dato solicitado por la CPU mediante una transferencia optimizada de bloques de datos y líneas.

Para determinar si el dato o la instrucción buscada se encuentra en la caché, el controlador del acceso se basa en la dirección de memoria accedida, que suele dividirse en tres campos, y un bit de validez asociado a cada línea de la caché:

- Campo índice, que se usa para localizar las líneas de la caché que pueden contener el dato que busca la CPU.
- Campo etiqueta, que determina cuál de las líneas anteriormente seleccionadas con el campo índice puede contener el dato.
- Bit de validez, que indica si el dato que busca la CPU se encuentra presente en la línea seleccionada con los campos índice y etiqueta.
- Campo *offset*, que permite identificar el dato dentro de la línea seleccionada de la caché.

Ejemplo de división de la dirección de una posición de memoria en etiqueta, índice y *offset*

Suponiendo que el bus de direcciones es de 32 bits, la figura siguiente muestra un ejemplo de división de la dirección de memoria en los tres campos citados.



En este caso, la caché tiene 512 líneas, cada una con 32 datos. Los 18 bits más significativos componen la etiqueta, que la caché comparará con la etiqueta interna de sus líneas. Los siguientes 9 bits constituyen el índice, que se usa para localizar o ubicar una línea dentro del conjunto de líneas de la caché. Finalmente, el *offset* de 5 bits ubica un dato dentro de la línea. En consecuencia, en este ejemplo, cada línea está formada por 32 datos. Internamente, la caché contiene las líneas formadas por la etiqueta, los datos y un bit de validez que sirve para identificar si los datos o instrucciones contenidos en la línea son copia de un bloque de información ubicado en la memoria del sistema.

Básicamente, y siguiendo esta estructura, existen tres formas de organizar la caché: correspondencia directa, asociativa por vías y completamente asociativa.

Caché de correspondencia directa

Cada bloque de datos de la memoria del sistema microprocesador sólo dispone de una línea donde puede estar ubicado dentro de la memoria caché. La búsqueda de la información en la caché se basa en el campo índice de la dirección de memoria, que apunta a una línea concreta de entre todas las que forman la caché. Si la etiqueta de la dirección coincide con la etiqueta de la línea de la caché y el bit de validez está activo, el dato solicitado se encuentra en la caché (*cache hit*), y no es necesario el acceso de la CPU a la memoria física donde se encuentra la información (la CPU recoge una copia de la información contenida en la caché). Si alguna de las dos condiciones anteriores no se cumpliese, tendríamos un fallo, o *miss*, por lo que la CPU tendría que acceder a

la memoria física para acceder a la información. En ese caso, el controlador de la caché aprovecha el acceso que realiza la CPU para almacenar la información en la correspondiente línea de la caché. La Figura 1.22 muestra una memoria caché de correspondencia directa de 16 K × 32 bits, organizada en 16 K líneas con 32 bits para almacenar información cada una. El campo índice de la dirección accedida, de 14 bits, permite seleccionar una de las 16 K líneas. Si los 16 bits más significativos de dicha dirección coinciden con la etiqueta de la línea seleccionada, y su bit de validez se encuentra activo, la memoria caché contiene el dato que busca la CPU, determinado por el campo *offset*.

El principal problema de la caché descrita es que aprovecha poco la localidad espacial. Si se observa con detenimiento la Figura 1.22, se puede ver que cada línea de la caché sólo contiene 32 bits. Si la CPU trabajase con datos de 8 bits, la localidad espacial sólo contemplaría un bloque de cuatro datos como posiciones cercanas con alta probabilidad de ser accedidas, pero si la CPU trabajase con datos de 32 bits, sencillamente no se aprovecharía la localidad espacial. Para aprovechar mejor esta localidad se necesitaría una memoria caché en la que cada línea fuese de un tamaño mayor que la palabra que utiliza el procesador, como ilustra la Figura 1.23. Aunque el tamaño de la caché es el mismo que el de la Figura 1.22, en este caso el tamaño de la

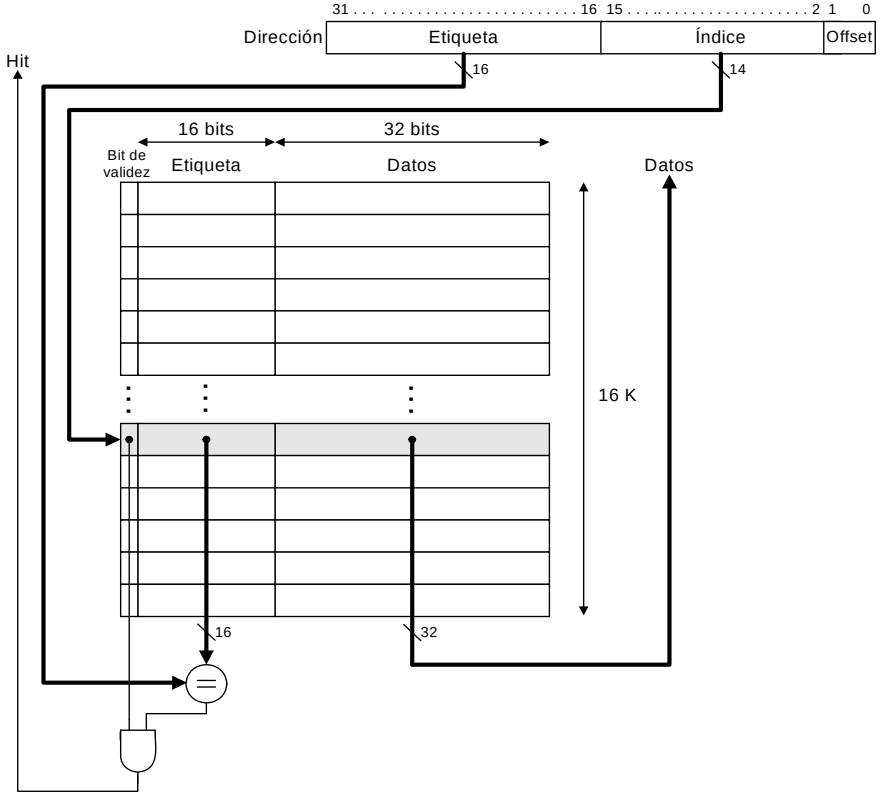


Figura 1.22. *Estructura de una caché de correspondencia directa de 16 K × 32 bits.*

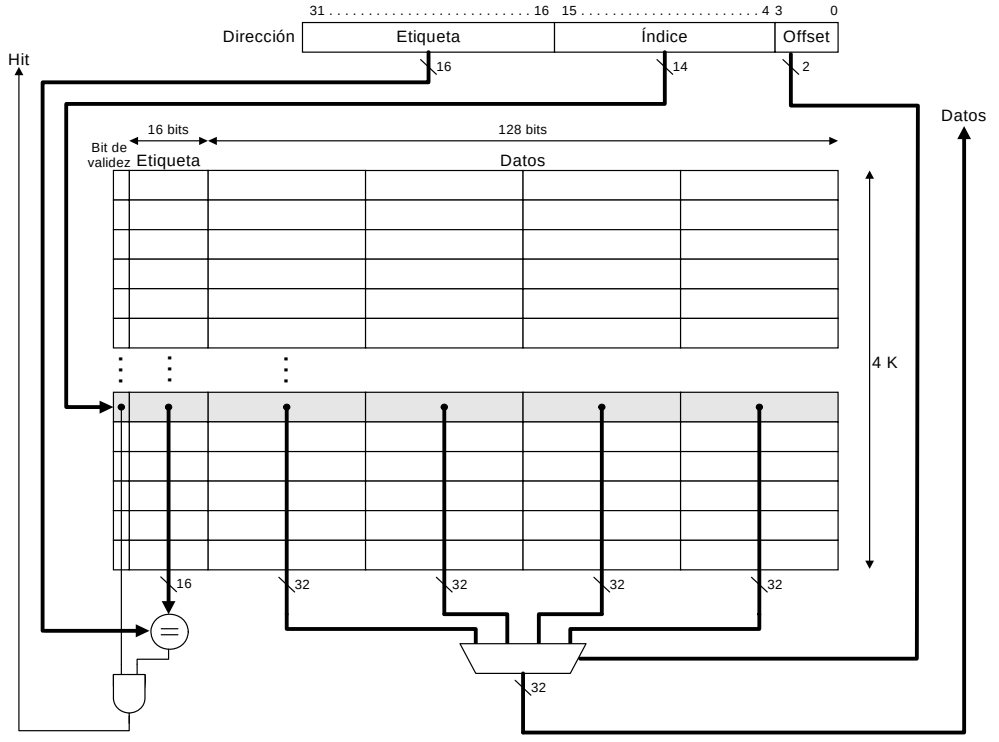


Figura 1.23. *Estructura de una caché de correspondencia directa de 4 K × 128 bits.*

línea es de 128 bits, de modo que cada vez que se guarda en la caché una posición de memoria, se guarda mayor número de posiciones adyacentes, aprovechando mejor la localidad espacial: el *offset* de la dirección ocupa ahora 4 bits, permitiendo acceder a 16 datos de un byte o cuatro datos de 32 bits.

Manteniendo la capacidad de almacenamiento de información de la caché, esta nueva forma de organizarla, incrementando el tamaño de la línea, implica las siguientes ventajas:

- El número total de etiquetas y bits de validez que utiliza la caché disminuye al disminuir el número de líneas. Con esto se consigue disminuir el tamaño físico que ocupa la memoria caché.
- Disminuye la tasa de fallos al aprovechar mejor la localidad espacial. No obstante, llegaría un momento en que una línea demasiado grande dejaría de mejorar la tasa de fallo, dado que el número de líneas de la caché sería demasiado pequeño; fijado un tamaño de caché, mayor tamaño de la línea implica menor número de líneas y los contenidos de las mismas serían desalojados más rápidamente, incluso antes de que muchos de esos datos fuesen accedidos. Además, líneas excesivamente grandes incrementan la penalización del fallo, puesto que el tiempo de transferencia de las líneas se incrementa con el tamaño de éstas (si se mantiene igual el ancho del bus de datos).

Caché asociativa por vías

Una caché asociativa de *n* vías es aquella que posee un conjunto de *n* posibles localizaciones para cada valor apuntado por el campo índice de la dirección del dato o instrucción accedido.

Así, por ejemplo, la caché asociativa de la Figura 1.24 es de cuatro vías, ya que cada bloque de datos de la memoria puede ubicarse en cuatro líneas (el campo índice de la dirección del bloque de memoria que contiene la información apunta a cuatro líneas que se corresponden con las cuatro vías de la caché). En este sentido, las líneas de la memoria caché asociativa por vías se pueden asimilar a las celdas de una matriz donde el índice equivaldría a la fila y la vía haría referencia a la columna de la matriz. Cada línea de la caché, o celda de la matriz, estaría constituida por un bit de validez, un campo de etiqueta más un campo de datos.

A diferencia de la caché de correspondencia directa, donde sólo existía un único comparador, la caché asociativa por vías posee tantos comparadores como vías; en el caso de la Figura 1.24, cuatro. Al igual que en el caso de la caché de correspondencia directa, el acceso a la caché asociativa por vías se realiza usando el campo índice, que

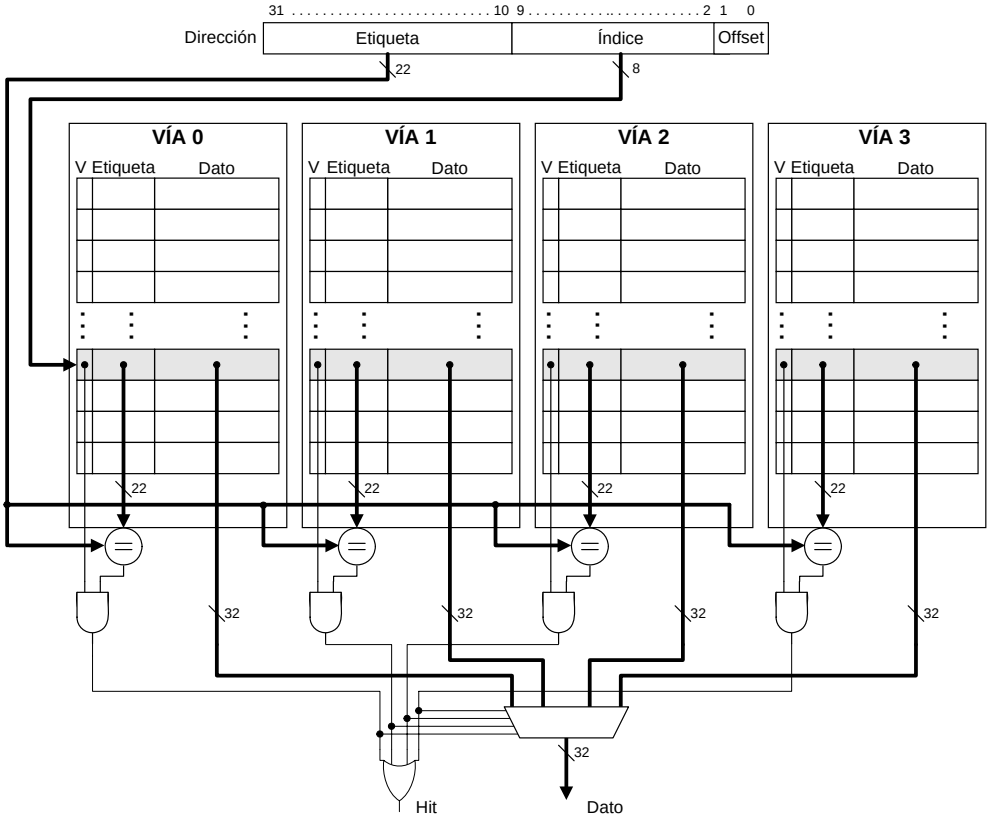


Figura 1.24. *Estructura de una memoria caché asociativa de cuatro vías.*

selecciona un conjunto de líneas (en el caso expuesto en la Figura 1.24, cuatro líneas). A continuación se comparan las etiquetas de esas cuatro líneas con la etiqueta de la dirección buscada, haciendo uso de los cuatro comparadores. Si alguno de ellos coincide, se produce un acierto o *hit*. En caso contrario, tendremos un fallo o *miss*. Al contar con varias líneas alternativas para situar un bloque de información sin necesidad de reemplazar en caso de acceso fallido, como ocurría en la caché de correspondencia directa, la caché asociativa por vías aumenta la tasa de acierto. Presenta, sin embargo, algunas desventajas, como:

- La necesidad de usar un mayor número de comparadores, tantos como vías.
- El incremento que se produce del tiempo de acceso a la memoria, necesario para determinar si ha habido acierto. Este incremento se debe al retardo adicional causado por la comparación y selección de la línea dentro del conjunto de posibles alternativas.

Caché completamente asociativa

Se caracteriza porque cada bloque de datos se puede colocar en cualquier línea de la caché. Se puede decir que se trata de una memoria asociativa por vías con el máximo número de vías posibles. Para realizar la búsqueda será necesario comparar la etiqueta de la dirección con todas las etiquetas almacenadas en la caché. Dado que existen tantos comparadores como vías, el coste en área de la caché totalmente asociativa es muy elevado, por lo que sólo se usan cuando el número de líneas de la caché es pequeño.

La Tabla 1.1 compara las diferentes formas de organizar una memoria caché. Cada vez que ocurre un fallo en el acceso a la caché, la CPU busca en el nivel jerárquico de memoria inmediatamente inferior, y el propio sistema de gestión de acceso a la memoria coloca, a continuación, el bloque de información en la caché para aprovechar tanto la localidad temporal (futuros accesos) como la localidad espacial (posiciones próximas).

En este proceso es importante la política de reemplazo, puesto que de ella dependerá la tasa de aciertos. En la caché de correspondencia directa, la política de reemplazo viene ya determinada por el hecho de que cada bloque de información sólo puede ocupar una línea fija de la caché, de modo que si ocurre un fallo, el nuevo bloque de información se guardará en la línea que le corresponda, reemplazando su contenido

Tabla 1.1. *Comparación de las diferentes formas de organizar una caché*

Organización de la caché	Número de líneas por vía	Número de vías
Correspondencia directa	Coincide con el número de líneas	1
Asociativa por vías	Número de líneas/número de vías	Típicamente, 2-8
Completamente asociativa	1	Coincide con el número de líneas

anterior. A diferencia de ella, en la caché asociativa por vías y en la totalmente asociativa sí que existe posibilidad de elección. En caso de fallo, el nuevo bloque de información puede guardarse en cualquiera de las líneas seleccionadas por el campo índice, tantas como vías tenga la caché (en el caso de la totalmente asociativa, todas las líneas son candidatas). Existen varios métodos para determinar qué línea va a ser reemplazada:

- Aleatorio. La línea a reemplazar se elige de forma pseudoaleatoria.
- Menos recientemente utilizado (LRU, o *Least Recently Used*). Se basa en el principio de localidad temporal, de manera que se reemplaza la línea que lleva más tiempo sin ser utilizada. Para ello es necesario llevar la cuenta del uso de las líneas asignadas a un único índice. Por ejemplo, en una caché asociativa de dos vías, bastaría con un bit que indicase cuál de las dos líneas asignadas a un único campo índice es la que menos recientemente se ha utilizado. Evidentemente, cuanto mayor sea el número de vías de la caché, más costosa resultará esta estrategia de reemplazo, por lo que sólo se suele emplear en memorias asociativas de dos, cuatro o, como máximo, ocho vías.
- El primero en llegar es el primero en salir. Las líneas se van sustituyendo en el mismo orden en el que se fue almacenando en ellas la información.

En las operaciones de lectura se procede a leer la información de la caché a la vez que se efectúa la comprobación de la etiqueta y el bit de validez. Así se ahorra tiempo en el caso de acierto. Si el acceso generase un fallo, se leería del nivel inmediatamente inferior en la jerarquía y se guardaría el correspondiente bloque de información en la caché, siguiendo alguna de las estrategias de reemplazo comentadas.

Las operaciones de escritura funcionan de una manera diferente con la caché. Si mediante una típica instrucción de almacenamiento sólo se procediese a modificar el contenido de la caché, el dato actualizado en la caché tendría un valor diferente al que aparece en la memoria principal del sistema. En ese caso se dice que la caché y la memoria principal son incoherentes. Existen dos estrategias para garantizar la necesaria coherencia de los datos:

- **Escritura directa.** La forma más simple de mantener la coherencia consiste en modificar tanto la caché como la memoria principal. El inconveniente que plantea este sistema es que obliga a realizar otra escritura en la memoria principal, que ocupa un nivel inferior en la jerarquía y, en consecuencia, es más lenta que la caché.
- **Postescritura.** En este caso se escribe únicamente en la caché, y sólo en el caso en el que una línea previamente modificada vaya a ser reemplazada, se escribe en los niveles de memoria jerárquicamente inferiores antes del reemplazo. De ese modo se garantiza la coherencia entre los diferentes niveles de memoria. Para llevar a cabo esta estrategia hay que añadir un bit de modificado (*dirty bit*) a cada línea de la caché, que indique si es necesario realizar la postescritura en un nivel inferior de la jerarquía de memoria antes del desalojo. Aunque esta estrategia es más compleja que la anterior, mejora las prestaciones de la caché, ya que requiere de menos escrituras en las memorias más lentas y de inferior nivel en la jerarquía.

Organización de la jerarquía de memoria

La memoria principal es el nivel que sigue a la memoria o memorias caché en la jerarquía de memoria de un sistema microprocesador. La relación entre memoria principal y memoria caché condiciona el funcionamiento de esta última, ya que la penalización de fallo depende del tiempo de acceso al nivel inferior de memoria y del tiempo de transferencia entre ambos niveles. Así pues, es importante encontrar una organización adecuada con el fin de reducir ambos tiempos y conseguir una penalización pequeña de fallos para la caché.

Normalmente, es mucho más fácil reducir el tiempo de transferencia entre dos niveles de memoria que reducir el tiempo de acceso; es decir, que la forma más sencilla de reducir la penalización de fallos de la caché consiste en incrementar el número de palabras por ciclo de reloj que se pueden transferir entre ésta y la memoria principal.

La Figura 1.25 muestra las tres opciones de organización de la jerarquía de memoria:

- La primera (Fig. 1.25a) muestra el caso de una memoria principal conectada a una memoria caché con un bus de datos de tamaño igual al ancho de los datos típicos que maneja la CPU. En este caso, o bien la línea de la caché contiene un único dato, en cuyo caso no se aprovecha la localidad espacial, o bien para completar una línea de la caché que contenga varios datos consecutivos (aprovechando la localidad espacial) es necesario que el controlador de la caché realice varias transferencias desde la memoria principal para completar una línea de la caché.
- La segunda opción (Fig. 1.25b) aumenta el ancho de banda de las transferencias entre la memoria principal y la caché, aumentando el tamaño del bus de datos entre ambas memorias. Esta organización permite que, a diferencia que en el caso anterior, el controlador de la caché complete una línea de ésta en una única transferencia de memoria.
- La tercera opción (Fig. 1.25c) se conoce como memoria entrelazada. También permite el incremento del ancho de banda, aunque en este caso se solapan en el tiempo los accesos que sobre la memoria principal realiza la caché. La memoria principal se divide en distintos bancos, incrementándose el ancho del bus de direcciones, pero no el del bus de datos de la memoria principal ni el de la caché. Se pueden enviar varias direcciones al mismo tiempo, cada una dirigida hacia un

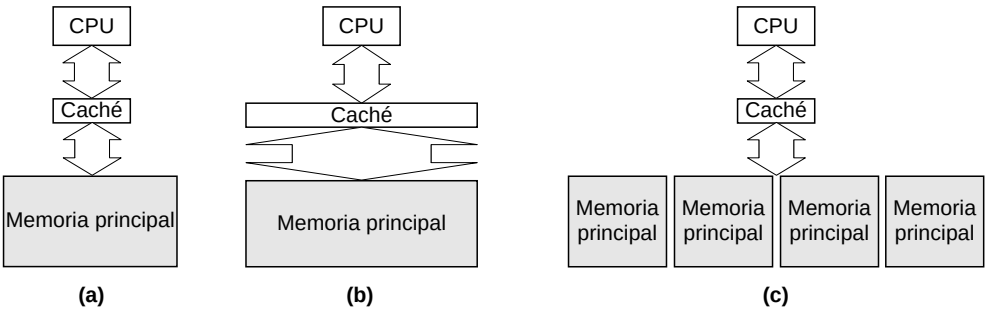


Figura 1.25. *Diferentes métodos de organización de la jerarquía de memoria.*

banco en concreto. De esta forma, el tiempo de latencia de un conjunto de accesos es equivalente al de un solo acceso, reduciéndose el número de ciclos de penalización de fallo de la caché. Tras realizarse los accesos, las palabras son transferidas secuencialmente por el bus de datos entre los dos niveles de la jerarquía.

Periféricos de acceso directo a memoria: DMA

Establecida la jerarquía de memoria, otros dispositivos o periféricos que intervienen en la mejora del rendimiento del sistema son los denominados periféricos DMA (*Direct Memory Access*), que son capaces de gestionar, sin coste alguno en tiempo de CPU para el microprocesador (Fig. 1.26), transferencias de bloques de información desde una posición en el mapa de memoria del sistema microprocesador (origen) hacia otra (destino).

Una transferencia DMA hace referencia al trasvase de un grupo de instrucciones o datos en el mapa de memoria del sistema digital. El número de instrucciones o datos que se van a trasvasar es uno de los parámetros característicos de la transferencia DMA. El periférico DMA dispone de un contador de información transferida que se inicializa al número de palabras que se desean transvasar. Cada vez que el periférico transfiere una palabra en el mapa de memoria del sistema, disminuye en una unidad dicho contador. Se dice que la transferencia DMA ha concluido cuando el registro contador de palabras transferidos alcanza el valor cero.

Cada palabra transvasada en el mapa de memoria implica dos operaciones elementales a realizar por parte del periférico DMA: una lectura de la información en la dirección origen (evento *Read-DMA*) y una escritura de la misma en la dirección destino (evento *Write-DMA*). Al final de cada evento de tipo *Read-DMA* (*Write-DMA*), la dirección origen (destino) automáticamente se incrementa, decrementa o permanece constante.

Hasta ahora se han analizado los sistemas microprocesadores como sistemas digitales en los que había un único maestro (el microprocesador) y multitud de esclavos (periféricos de almacenamiento de programas y datos, de conversión analógico-digital y digital-analógico, etc.). El maestro era el que gestionaba la transferencia de información por los buses del sistema: activaba en modo lectura o escritura al periférico al que quería

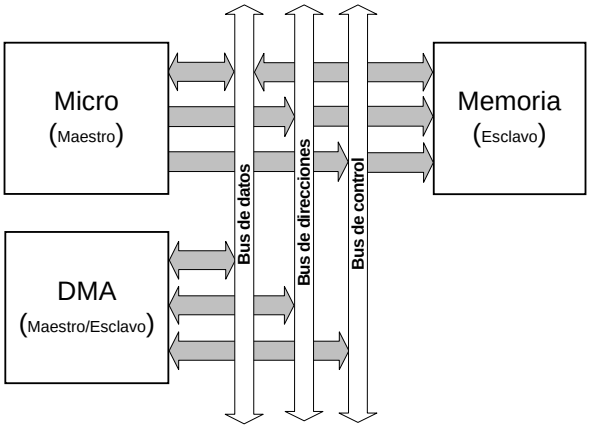


Figura 1.26. *Estructura maestro-esclavo en un sistema microprocesador: modo básico de funcionamiento del periférico DMA.*

acceder, mediante las líneas asociadas a los buses de control y dirección, y completaba el acceso, siguiendo una cierta temporización en el acceso suministrada por el fabricante, leyendo (o escribiendo) información del (en el) bus de datos del sistema. El periférico DMA es un dispositivo que puede funcionar, en el sistema digital en el que se encuentre inmerso, como maestro o esclavo. El microprocesador puede acceder a él para configurar su modo de funcionamiento, como haría con cualquier otro periférico, comportándose, en ese momento, el periférico DMA como esclavo en el sistema. La característica fundamental que tiene este periférico es que, una vez configurado y habilitado, su funcionamiento normal consiste en realizar transferencias de información a través de los buses de datos del sistema, sin que dicha transferencia suponga coste de CPU para el microprocesador. Para poder realizar esa tarea, el periférico DMA necesita poder gobernar los buses y funcionar como maestro en el sistema digital. Obviamente, para no provocar conflictos en los accesos a los buses, sólo puede haber un maestro en el sistema.

Para evitar que aparezcan dos maestros que gobiernen las líneas de los buses de datos y direcciones del sistema se añade al bus de control de muchos microprocesadores un par de líneas, una de entrada (normalmente HLD) y otra de salida (normalmente HLDA), que implementan un protocolo de acuse de recibo. El protocolo que se establece implica que, por defecto, el maestro en el sistema es el microprocesador. Si algún periférico desea tomar el control de las líneas asociadas a los buses de datos, dirección y control, primero debe solicitarlo al maestro (esta petición se realiza activando la línea HLD de entrada al microprocesador). El microprocesador analiza internamente la petición, y si el usuario que ha programado el micro lo permite, atiende la petición (deja en alta impedancia las líneas de los buses de dirección, control y datos). Una vez que el microprocesador ha dejado de gobernar los buses del sistema, le indica, activando la línea HLDA, al periférico que quería ser maestro (normalmente un periférico de tipo DMA) que puede gobernar los buses.

A partir de ese momento, el dispositivo que controla como maestro las líneas de los buses del sistema digital es el periférico DMA, que realiza entonces la transferencia de información (programada previamente por el microprocesador) en el mapa de memoria del sistema digital. Esta transferencia de información es un proceso repetitivo, que consiste en un acceso en lectura a una posición de memoria (el periférico DMA impone, en las líneas asociadas al bus de direcciones del sistema, el valor de la dirección de la posición a la que desea acceder, activa la señal de lectura asociada al bus de control y copia el valor que aparece en el bus de datos en un registro interno del periférico) y un acceso posterior en escritura a otra zona de memoria (el periférico DMA impone, en las líneas asociadas al bus de direcciones del sistema, el valor de la dirección de la posición a la que accede, activa la señal de escritura asociada al bus de control y copia el valor que aparece en el registro interno del periférico en el bus de datos). La dirección del bloque de memoria que se desea copiar, el tamaño de dicho bloque (que indica cuántas veces consecutivas accede en lectura y escritura la DMA como maestro del sistema digital) y la dirección donde se desea copiar el bloque de memoria son datos que programó el microprocesador cuando accedió, para configurarlo, al periférico DMA. Una vez que el periférico DMA es maestro, el microprocesador no puede efectuar accesos al exterior por los buses del sistema hasta que la DMA no pase a ser esclavo, lo que ocurre cuando concluye el trasvase programado.

La utilidad que ofrece un periférico de este tipo es descargar al microprocesador de la realización de operaciones de transvase de gran cantidad de información, normal-

mente datos, por lo que tendrá interés sólo si el usuario tiene previsto desarrollar aplicaciones en las que se realice gran cantidad de trasvases de datos en el mapa de memoria del sistema digital.

El funcionamiento de los periféricos DMA que se ha descrito se extiende en los sistemas microprocesadores modernos. En ellos, los periféricos DMA se incluyen dentro del mismo chip que constituye el microprocesador, trabajando en paralelo con la CPU. La DMA normalmente cuenta con sus propios buses de direcciones y datos que le permiten acceder al resto de recursos *hardware* del sistema, independientemente de la tarea que esté desempeñando la CPU. En el caso de que ambos subsistemas, CPU y DMA, intenten acceder al mismo recurso (por ejemplo, una misma posición de memoria), un sistema de gestión de prioridades en el acceso determinará quién accede al recurso en primer lugar.

Evolución histórica de las arquitecturas del sistema microprocesador

El progreso de la tecnología, especialmente en todo lo relacionado con la Electrónica, ha afectado fuertemente a la evolución de la arquitectura del sistema microprocesador. Evaluando el periodo transcurrido desde la aparición de los primeros sistemas microprocesadores hasta nuestros días, se pueden distinguir, al menos, cinco etapas.

En la época inicial o de nacimiento del microprocesador, *primera etapa* (1946-1957), se extendió la estructura propuesta por von Neumann. En esta etapa, la CPU y la memoria soportan la mayor actividad en el sistema microprocesador. La CPU gestiona el envío a la memoria de las direcciones de las instrucciones, la recepción e interpretación de las instrucciones, y en la fase de ejecución, selecciona la operación que debe realizarse en la ALU (lo que implica la búsqueda de los operandos y el almacenamiento del resultado). En estos primeros sistemas microprocesadores existe un constante trasiego de instrucciones y datos entre la memoria y la CPU, que además se encarga de gestionar la transferencia de datos entre los periféricos de entrada y salida del sistema. La tecnología empleada en la implementación de esta estructura consistía en válvulas de vacío, y dado que tanto la CPU como la memoria estaban construidas con dichos dispositivos, la velocidad de respuesta de ambos bloques era similar. El tiempo de procesado de las instrucciones se repartía entre la unidad de control y la memoria, y al ser semejantes sus velocidades, se obtenía un rendimiento óptimo. En esta época se usaba el lenguaje máquina binario para programar los sistemas. Otros avances tecnológicos significativos en esta etapa son la aparición de las memorias de ferrita y las cintas y discos magnéticos.

Con el tiempo, se produjo un distanciamiento entre la tecnología empleada en la construcción de la CPU y de la memoria del sistema. La aparición de los transistores y de los primeros circuitos integrados de pequeña y mediana escala de integración (SSI y MSI) permite su uso en la implementación de la CPU, aunque aún no se pueden usar como soporte de la memoria, que se implementa con tecnologías tales como los núcleos de ferrita con elevados tiempos de acceso. La velocidad de acceso a la memoria era en esta época unas diez veces inferior a la de la CPU, y el rendimiento del sistema inicial deja de ser óptimo al provocar largos periodos de inactividad de la CPU en las fases de acceso a memoria. Aparecen entonces las que se conocen como *segunda y tercera etapas* (de 1958 a 1963 y de 1964 a 1971, respectivamente) en la historia de los sistemas microprocesadores. Para compensar la diferencia de velocidad, surgen los sistemas microprocesadores de tipo CISC. Estos sistemas incorporan complejas instrucciones que

tardan en completarse varios ciclos de reloj y que para su ejecución requieren la realización de numerosas operaciones elementales. De esta manera se consiguen reducir los accesos a memoria de la CPU (al menos aquellos dedicados a la recogida de los códigos de las instrucciones). A su vez, la CPU se complica al requerir una pequeña memoria, denominada memoria de control, para almacenar los códigos de las operaciones elementales en las que se descompone cada instrucción. La instrucción pasa a ser una macroinstrucción que para ser ejecutada por la CPU requiere ser descompuesta en varias operaciones elementales o microinstrucciones, grabadas en la memoria de control. Todo esto, con la finalidad de incrementar el proceso de manipulación de la instrucción en la CPU, reducir el número de accesos de la CPU a la memoria y paliar el desfase entre la velocidad de la CPU y la de la memoria del sistema microprocesador.

El aumento posterior de la densidad de integración de los circuitos integrados (aparición de los circuitos integrados de alta escala de integración, LSI) da lugar a una *cuarta etapa* (1972-1980) en la historia de los sistemas microprocesadores, al permitir la construcción de memorias electrónicas rápidas (memorias semiconductoras). Estos dispositivos se incorporan al sistema microprocesador, equilibrándose la velocidad de la memoria con la de la CPU. Además, entre la CPU y la memoria se intercalan, en la zona de memoria de programa, dispositivos de memoria ultrarrápida aunque de pequeña capacidad (dispositivos de memoria caché), con la misión de disminuir los tiempos de acceso de la CPU a las instrucciones y equiparar, aún más si cabe, las velocidades de la CPU y la memoria. La memoria caché, con velocidades de acceso de entre cinco y diez veces mayor que la de los dispositivos que conforman la memoria del sistema, abastece a la CPU de instrucciones y evita a veces que ésta acceda a la zona de memoria de programa del sistema (la efectividad de estos dispositivos está en lograr contener las instrucciones que más frecuentemente precisa la CPU). En esta etapa decae el interés en los sistemas microprocesadores de tipo CISC. Se empiezan a imponer sistemas microprocesadores con juegos de instrucciones más simples que las que manejan las arquitecturas CISC, en los que la CPU es capaz de ejecutar una instrucción en un único ciclo máquina. Estos sistemas microprocesadores, que se conocen con el nombre de RISC, terminarán por imponerse en la siguiente etapa. En esta etapa se imponen, además, las arquitecturas de tipo Harvard frente a las de tipo von Neumann, especialmente en aquellos sistemas microprocesadores diseñados con unas altas prestaciones y una elevada potencia de cálculo.

El uso de los periféricos de memoria caché se extiende en la *quinta etapa*, a partir de 1981-..., caracterizada tecnológicamente por disponer de los denominados circuitos integrados de muy alta escala de integración, VLSI. Estos dispositivos consiguen acelerar la velocidad efectiva de acceso a las memorias, haciendo necesaria la mejora de la velocidad de la CPU. Para ello se establecen nuevos criterios en el diseño de la estructura del microprocesador: la eliminación de la microcodificación (todas las instrucciones pasan a ser sencillas para evitar la existencia de la memoria de control), la reducción del tiempo del ciclo máquina (consecuencia de la simplificación de las instrucciones), la interpretación directa por el *hardware* de las instrucciones y su ejecución en un ciclo máquina, la selección de un número mínimo de instrucciones en el repertorio y la ampliación de los periféricos de memoria caché (cada vez de mayor tamaño, jerarquizados en varios niveles con diferentes velocidades y capacidades y dedicados a acelerar el acceso por parte de la CPU a las instrucciones y a los datos). Este nuevo enfoque termina por imponer los sistemas microprocesadores de tipo RISC, en los que se eleva el número de

registros internos de propósito general de que dispone la CPU para evitar accesos a memoria. Aparecen los primeros compiladores capaces de analizar los datos e instrucciones y de generar los flujos óptimos de ejecución de las instrucciones en la CPU.

Las nuevas arquitecturas del sistema microprocesador: taxonomía de Flynn

El diseño de los sistemas microprocesadores se encuentra en plena evolución. Incluso cuando los límites planteados por la frecuencia de reloj y la escala de integración de los circuitos microelectrónicos se convierten en insalvables, se buscan nuevas formas de aumentar la potencia de procesamiento del sistema, explotando al máximo la tecnología existente. En la actualidad, la clave para conseguir sistemas de mayor rendimiento recae en la habilidad de explotar el paralelismo en la ejecución de las instrucciones por parte de la CPU. Los principales procedimientos para conseguir este paralelismo son:

- La *segmentación, «pipelining» o paralelismo temporal.* Hoy en día, es la técnica que más se utiliza para conseguir aumentar el rendimiento del sistema microprocesador. Persigue simultanear en el tiempo la ejecución de varias instrucciones. Para ello, la ejecución de cada instrucción se descompone en partes más pequeñas, etapas de la segmentación o segmentos, cada una conectada con la siguiente, formando un cauce en el que las instrucciones entran por un extremo, son procesadas en varias etapas y salen por el otro extremo. Además, la CPU tiene la capacidad de procesar simultáneamente todas las etapas de la segmentación, por lo que mientras una instrucción se encuentra dentro del cauce, procesándose en alguno de sus segmentos, otras instrucciones se encuentran también dentro de él, aunque procesándose en segmentos diferentes del anterior. De esta manera, puede entenderse que la CPU se encuentra ejecutando varias instrucciones a la vez en un ciclo máquina, y que bajo estas condiciones la mejora de velocidad debida a la segmentación equivale al número de etapas de la propia segmentación. Dado que las etapas se conectan entre sí y son procesadas simultáneamente por la CPU, todas deben tardar el mismo tiempo en completarse. El tiempo que se requiere para desplazar una instrucción un segmento a lo largo de su cauce de ejecución es el ciclo máquina de la CPU (normalmente uno o dos ciclos de reloj). Por tanto, la duración del ciclo máquina estará determinada por el tiempo que requiere la CPU para realizar la etapa más lenta de la segmentación de cualquier instrucción. En estos sistemas microprocesadores se busca, como objetivo, equilibrar la duración de las diferentes etapas de la segmentación. Si las etapas están perfectamente equilibradas, el tiempo por instrucción de la máquina segmentada, suponiendo condiciones ideales, se puede calcular dividiendo el tiempo que tarda en completarse la ejecución de la instrucción entre el número de etapas en las que se segmenta. Sin embargo, habitualmente las etapas no están perfectamente equilibradas, por lo que el tiempo por instrucción en un sistema microprocesador segmentado no siempre será el menor valor posible, aunque estará próximo. En cualquier caso, con la segmentación se consigue una reducción en el tiempo de ejecución medio de cada instrucción, determinándose la productividad de este método de paralelismo a partir de la frecuencia con que salen las instrucciones del cauce. Frente a otras técnicas para

aumentar la velocidad del sistema microprocesador, la segmentación tiene la ventaja de ser transparente al programador (todo el paralelismo es resuelto e implementado por *hardware* por la CPU), aunque presenta problemas que impiden la consecución del máximo rendimiento posible; aparecen situaciones, denominadas riesgos o *hazards*, que ralentizan la ejecución de algunas instrucciones del flujo determinado por el programador y que reducen el rendimiento en velocidad, respecto del ideal, logrado por la segmentación. Se distinguen tres clases de riesgos: los denominados riesgos estructurales (que surgen de conflictos en los recursos disponibles en la CPU, cuando el *hardware* no puede soportar todas las posibles combinaciones en la ejecución solapada de las instrucciones), los riesgos por dependencias de datos (que aparecen cuando una instrucción depende de los resultados de una instrucción anterior, de forma que ambas no pueden llegar nunca a ejecutarse de forma solapada) y los riesgos de control (que surgen en la segmentación de los saltos y otras instrucciones que cambian el contador de programa). Los riesgos son resueltos por la CPU de forma transparente al programador, deteniendo (y por tanto ralentizando) la ejecución de algunas instrucciones, las últimas que entraron en el flujo que está ejecutando la CPU, mientras otras prosiguen, las más antiguas que se encuentran en el flujo de ejecución de la CPU; cuando una instrucción está detenida, todas las instrucciones posteriores en el cauce a ésta también se detienen, y la CPU deja de buscar instrucciones nuevas mientras no avance el flujo de ejecución.

- El *multiprocesamiento o paralelismo asíncrono.* Dentro de este método de paralelismo aparece el denominado multiprocesamiento de uso específico, o uso de varias CPU que comparten una memoria común, conectadas entre sí de forma jerárquica, funcionando una de ellas como dispositivo principal o maestro y el resto como esclavas de la anterior. Sin embargo, la forma más habitual de multiprocesamiento es aquella que se caracteriza por constar de varios sistemas microprocesadores similares, fuertemente acoplados, que interaccionan, que disponen de y comparten recursos (una memoria común, periféricos de entrada y salida de datos al sistema) y que se gobiernan con un único sistema operativo.
- El *paralelismo espacial.* El aumento de la potencia de la CPU se debe al empleo de operadores vectoriales, que operan de forma independiente con otros cálculos previos, por lo que no aparecen riesgos por dependencia de datos, y que manejan gran cantidad de datos en cada operación (vectores).

Como consecuencia de la aparición y desarrollo de las técnicas anteriores, tendentes a aumentar y explotar el paralelismo en la ejecución de instrucciones por parte de la CPU, aparecen nuevas implementaciones o arquitecturas del sistema microprocesador. Dentro de estas nuevas arquitecturas[11], podemos destacar las siguientes:

1. Los denominados *sistemas microprocesadores supersegmentados.* Representan un ejemplo típico de arquitectura microprocesadora que incorpora un pa-

[11] Los sistemas microprocesadores modernos no emplean una única técnica de paralelismo, por lo que estos sistemas no quedan claramente incluidos en sólo una de las arquitecturas que se plantean en este apartado.

ralelismo de tipo temporal. La profundidad de las diferentes fases, en que se divide la ejecución de cada instrucción, se extiende para simplificarlas y disminuir así la complejidad y el conjunto de operaciones que debe de realizar la CPU para completar su ejecución. De esta manera se consigue disminuir el tiempo necesario para la realización de cada fase y el ciclo máquina de la CPU.

2. La denominada *implementación superescalar,* que extiende los conceptos de paralelismo temporal o segmentación y paralelismo asíncrono. Estas arquitecturas se basan en sistemas microprocesadores segmentados en los que se busca disminuir el número de ciclos de reloj necesarios para la realización de las instrucciones, permitiendo a la CPU gestionar por *hardware* más de una instrucción por ciclo de reloj. Con ello se consigue aumentar la frecuencia de ejecución de las instrucciones, que puede llegar a ser superior a la frecuencia de reloj. Los sistemas microprocesadores emiten varias instrucciones independientes (normalmente pocas, entre dos y cuatro) por ciclo de reloj. Si las instrucciones del flujo son dependientes y crean riesgos, éstos son resueltos in situ por la CPU, ralentizándose la velocidad de ejecución del programa del usuario; la CPU planifica la ejecución de las instrucciones de forma dinámica, intentándose mejorar la capacidad de procesamiento de instrucciones de la CPU mediante técnicas de paralelismo que suponen innovaciones en la planificación de la ejecución de instrucciones que realiza el *hardware*.
3. La *arquitectura VLIW* (sistemas microprocesadores con una palabra de instrucción muy larga, *Very Long Instruction Word*). Esta arquitectura es similar a la superescalar, en tanto en cuanto aprovecha los paralelismos temporal y asíncrono. Sin embargo, en este tipo de sistema microprocesador, la planificación de la ejecución de las instrucciones no la realiza la CPU en tiempo de ejecución, sino que la realiza el compilador antes de la propia ejecución del programa (planificación estática). Obsérvese que el aumento del número de instrucciones que ejecuta la CPU en un ciclo máquina genera un aumento de la interdependencia entre dichas instrucciones, lo que a su vez limita las posibilidades de ejecución simultánea de las mismas. Por otro lado, la lógica que se requiere dentro de la CPU para planificar de forma adecuada la ejecución simultánea de estas instrucciones llega a ocupar una gran parte del silicio de la propia CPU (tan es así que muchas veces la mayor parte de los recursos y del silicio de la CPU están dedicados a dicha planificación). La arquitectura VLIW plantea la utilización de ese silicio en la implementación de más unidades funcionales, lo que permitiría ejecutar más instrucciones por ciclo y aumentar el paralelismo en la ejecución de las instrucciones (como en los superescalares). Para ello es preciso que el compilador realice la tarea de planificación en la ejecución de las instrucciones, asignada a la CPU en las arquitecturas superescalares, y es necesario que lo haga en el tiempo de compilación, al generar el código binario que se ejecutará en el sistema. Una ventaja inmediata de este tipo de planificación de la secuencia de ejecución de las instrucciones es que permite dedicar más tiempo a encontrar la mejor optimización, si bien el compilador se vuelve más complejo, ya que sobre él recae la responsabilidad de agrupar de la mejor forma posible las instrucciones. Los compiladores juegan un papel fundamental en el diseño de las arquitecturas VLIW.

Conocen la fuente de código del programa, que suele contener información acerca del comportamiento del mismo, y pueden utilizar dicho conocimiento para lograr un alto nivel de paralelismo en la ejecución de las instrucciones. Para mejorar la calidad del código máquina producido, el compilador realiza una compilación previa denominada *trace-driven*, que genera un programa que funciona, aunque sin llegar a ser óptimo. El programa generado contiene rutinas intercaladas en el código del usuario que controlan y analizan el comportamiento del programa (qué ramas se usan, con qué frecuencia, etc.), lo que es usado por el propio compilador para, en una segunda fase, producir un código máquina más eficiente. Utilizando estas técnicas, se consigue que la CPU abarque y controle ventanas de instrucciones mayores que las que un sistema microprocesador superescalar es capaz de controlar.

4. Los *sistemas microprocesadores vectoriales.* Son sistemas que disponen, además de los operadores clásicos que permiten el manejo de datos escalares, de operadores aritméticos específicos que permiten a la CPU operar directamente con vectores (suma, resta, multiplicación, división). Representan un claro ejemplo de paralelismo espacial.
5. Los *sistemas microprocesadores matriciales.* Disponen de múltiples operadores aritméticos de carácter general que trabajan en paralelo y operan con escalares. Son capaces de realizar operaciones con vectores y matrices, aunque descomponiendo estas operaciones en un conjunto de operaciones escalares que se programan para ser ejecutadas en los operadores genéricos que incorpora la CPU. Son un ejemplo de paralelismo asíncrono.
6. Los *sistemas microprocesadores simbólicos.* Este tipo de sistemas, también denominados sistemas microprocesadores PROLOG, LISP o manipuladores simbólicos, son empleados en aplicaciones relacionadas con la inteligencia artificial, como reconocimiento de modelos, comprobación de teoremas..., en las que no se necesitan los elementos numéricos (datos y resultados, operaciones elementales o primitivas, etc.) que aparecen habitualmente en los sistemas microprocesadores antes comentados.

La gran diversidad y las diferencias existentes en la estructura interna de los sistemas microprocesadores hace necesaria la búsqueda de modelos para poder clasificarlos y compararlos. Así, en 1996, Michael J. Flynn, observando los flujos de instrucciones y datos, propuso un sencillo modelo, que se ha mantenido con pequeñas modificaciones hasta nuestros días y denominado taxonomía de Flynn, para clasificar en una de cuatro categorías a los sistemas microprocesadores. Estas categorías son las siguientes:

- Sistemas microprocesadores de tipo SISD (*Single Instruction, Single Data steam*), con un único flujo de instrucciones y de datos. En estos sistemas, una CPU interpreta una única secuencia de instrucciones que procesan datos individuales de tipo escalar. Los sistemas microprocesadores segmentados se engloban dentro de esta categoría.
- Sistemas microprocesadores de tipo SIMD (*Single Instruction, Multiple Data steam*), con un único flujo de instrucciones y múltiples flujos de datos. El coste de un sistema microprocesador con múltiples CPU es muy elevado, por lo que

los diseñadores han venido considerando opciones de diseño que abaratan el coste, sin degradar seriamente la potencia o eficiencia del sistema. Una de estas opciones ha consistido en centralizar uno de los componentes principales del sistema, la CPU, para que una única instrucción controle la ejecución sincronizada y simultánea de un cierto número de operaciones con datos, disponiendo cada operación de una memoria asociada. Un ejemplo típico de sistema microprocesador que tiene este tipo de organización es el sistema microprocesador vectorial.

- Sistemas microprocesadores de tipo MISD (*Multiple Instruction, Single Data steam*), con un flujo múltiple de instrucciones y un único flujo de datos. Esta categoría es la más difícil de entender, dado que ninguna de las arquitecturas comentadas previamente tiene clara cabida en ella, y representa sin duda el mejor ejemplo de la necesidad de ampliar, concretando más, la clasificación de Flynn (de hecho, otros autores han propuesto sus propias taxonomías, taxonomías de Skillicorn, Feng, etc.). Algunos autores consideran como ejemplo de sistema microprocesador MISD a los sistemas microprocesadores superescalares, al emitir varias instrucciones por ciclo y disponer de varias unidades aritmético-lógicas aunque con una única memoria donde buscar los datos. Sin embargo, la ejecución de las instrucciones en paralelo no se realiza en el mismo instante de tiempo, por lo que no sería extraño definirlos como de tipo SISD.
- Sistemas microprocesadores de tipo MIMD (*Multiple Instruction, Multiple Data steam*), con múltiples flujos de instrucciones y datos. Se trata de sistemas que disponen de varias CPU, cada una de las cuales ejecuta un conjunto diferente de instrucciones sobre distintos conjuntos de datos.

1.2.4. *Modos de direccionamiento de la CPU*

La CPU dispone de un circuito de salida de direcciones para gestionar la transferencia de información en el sistema microprocesador. Este circuito impone un valor fijo durante todo el acceso de lectura o escritura de la CPU, en los *latches* de salida del bus de direcciones que corresponda: el bus de direcciones asignado a los datos, si la CPU accede en lectura o escritura a un dato, o el bus de direcciones de las instrucciones, si la CPU accede al exterior para recoger una instrucción.

Cuando la CPU accede a un dato necesario para la ejecución de una instrucción, éste es especificado a la CPU en el código de operación de la propia instrucción. De este modo, en el código de operación no sólo se le indica a la CPU la instrucción que debe realizar, sino también los datos u operandos que debe utilizar, así como el registro en el que debe almacenar el resultado de la operación. Habitualmente, en el código de operación no se incluye directamente el operando, el valor del dato con el que debe operar la CPU, y si se incluye, sólo es incluido directamente en el código de operación de la instrucción uno de los operandos. En su lugar se suele incluir, total o parcialmente, o bien la dirección del registro en el que se encuentra el dato o una referencia a algún registro interno de la CPU que contiene el dato o la dirección en la que se encuentra.

La forma que tiene la CPU de acceder al dato a partir de la referencia incluida en el código de operación de la instrucción se denomina modo de direccionamiento. Cualquier instrucción que pueda ejecutar una CPU suele tener uno o más modos de direc-

cionamiento. A continuación citamos los más típicos en el acceso a los datos (puede ocurrir que una CPU no tenga alguno de ellos o que tenga alguno especial no citado):

- **Direccionamiento implícito o inherente.** Este modo de direccionamiento está asociado a instrucciones que pueden operar con un único registro interno de la CPU (generalmente el acumulador). No es necesario hacer ninguna referencia al registro a emplear, pues ya es conocido por la CPU. Por ejemplo, incrementar el acumulador.
- **Direccionamiento inmediato.** En este modo, las instrucciones contienen el dato con el que va a operar la CPU. Por ejemplo, cargar en el acumulador el número 100h.
- **Direccionamiento directo.** Las instrucciones contienen la dirección, o parte de la dirección, de memoria donde se encuentra el dato. Por ejemplo, cargar en el acumulador el contenido de la posición de memoria 300Ah (Fig. 1.27).
- **Direccionamiento indirecto.** Una parte de la instrucción sirve para indicarle a la CPU el registro interno que le va a servir de base para operar. Este registro interno, que no es el operando, contiene la dirección del operando. Por ejemplo, cargar el acumulador con el contenido de la posición de memoria cuya dirección se encuentra en la posición de memoria 16h (Fig. 1.28). Dentro de este modo se pueden incluir otras formas que tiene la CPU de acceder al dato en las que la instrucción se emplea como antes se ha comentado, incluyendo además un registro auxiliar que actúa como desplazamiento en el cálculo de la dirección del operando. Con matices, a este modo se le suele denominar direccionamiento indexado o indirecto indexado.

Cuando la CPU accede a una instrucción, puede modificar el contador de programa utilizando los siguientes modos de direccionamiento:

- **Direccionamiento relativo al PC.** Al registro contador de programa se le suma un valor incluido en la instrucción. La CPU localiza la siguiente instrucción a ejecutar en una posición de memoria desplazada del PC actual un valor que va incluido en la instrucción. Por ejemplo, salta a una posición que se encuentra cien posiciones de memoria más hacia delante de donde está ahora el contador de programa.
- **Direccionamiento absoluto.** En el registro PC se copia un valor incluido en la instrucción. La CPU localiza la siguiente instrucción a ejecutar en la posición de

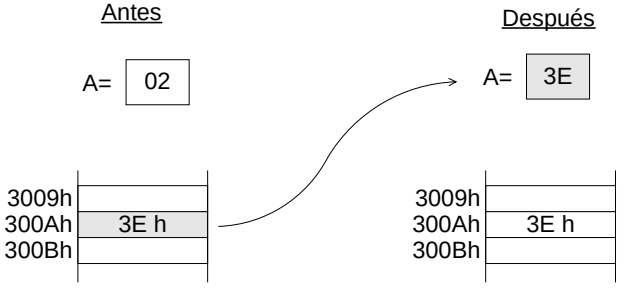


Figura 1.27. *Ejemplo de direccionamiento directo. La instrucción contiene la dirección, o parte de la dirección, de memoria donde se encuentra el dato (en este caso 300Ah).*

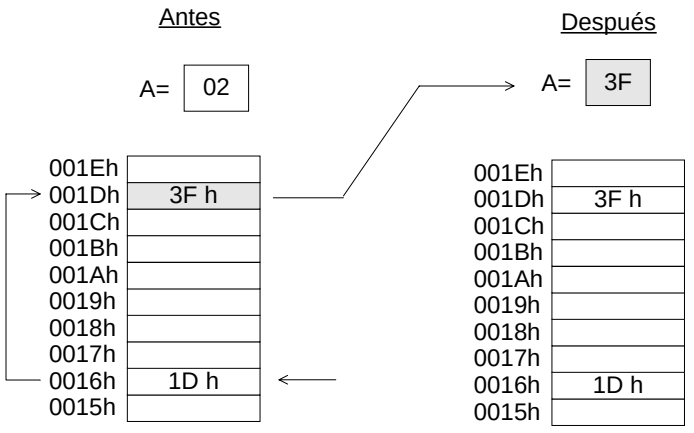


Figura 1.28. *Ejemplo de modo de direccionamiento indirecto. La instrucción contiene la dirección del «puntero» al dato (en este caso 16h).*

memoria que se incluye como dato en la instrucción que se está ejecutando. Por ejemplo, salta a la posición de memoria 8100h.

1.2.5. *Interrupciones e inicialización del sistema*

Hemos visto que la CPU debe entenderse como un dispositivo digital cuya función es ejecutar instrucciones secuencialmente. Por ello, la estructura de un programa que va a ejecutar una CPU es la de un bucle infinito en el que la parte de control está continuamente recogiendo un conjunto fijo de instrucciones. Cualquier ruptura en la «secuencia» normal de ejecución de un programa por parte de la CPU es debida a dos posibles causas: la aparición de un evento que se conoce como interrupción o a la inicialización del sistema.

Interrupciones

El periférico del sistema microprocesador permite la interacción de la CPU con el exterior. Así, un CAD permite a la CPU recoger valores digitales equivalentes a la señal analógica que se está muestreando. Cuando el periférico cumple su misión, la CPU debe tener conocimiento de dicho cumplimiento para poder actuar en consecuencia. En el caso del CAD, la CPU debe leer el valor del dato convertido para luego poder procesarlo. Esta tarea la puede realizar la CPU de dos formas:

1. Mediante el denominado sondeo (*polling*). Cuando el periférico cumple su misión, suele activar alguna bandera asociada a un registro de estado interno al propio periférico. La CPU puede leer dicho registro de estado, en cada iteración del bucle principal de control, y comprobar si se ha producido su activación, lo que delataría la presencia del evento.

2. Mediante interrupciones. En este caso, el periférico dispone de un mecanismo *hardware* que le permite activar una señal externa cuando se produce el evento. La activación de la línea es utilizada por el periférico para demandar, de forma inmediata, la atención por parte de la CPU. Esta línea de salida del periférico se conectaría a alguna línea de entrada y a un bloque interno de la parte de control de la CPU, que responde a la activación cortando la secuencia normal de ejecución del programa y atendiendo al periférico mediante la ejecución de un conjunto de instrucciones agrupadas en una rutina que se denomina rutina de servicio de la interrupción. La principal peculiaridad de esta forma de atención al evento externo es que, al terminar la ejecución de la rutina de servicio de la interrupción, la CPU continúa ejecutando la secuencia normal de instrucciones, asociadas al programa principal, por donde iba cuando se produjo el evento. Este método de interacción entre la CPU y un periférico externo suele ser más ventajoso que el sondeo, ya que no necesita consumir recursos de la CPU para la detección del evento. De hecho, la propia CPU puede generar peticiones de interrupción de la secuencia normal de ejecución de instrucciones para avisar de una operación aritmética extraña (por ejemplo, una división por cero, el desbordamiento en una operación, etc.), como consecuencia del intento de ejecución de un código de operación que no está contemplado por la propia CPU, etc. Por tanto, una CPU dispone de múltiples fuentes de petición de interrupción, cada una asignada a uno o varios periféricos externos o a la propia CPU. En ocasiones, cuando el número de fuentes de interrupciones es elevado, varias de éstas se pueden multiplexar sobre una única petición de interrupción a la CPU.

El proceso de atención de interrupciones por parte de la CPU puede ser controlado por el usuario mediante la configuración de uno o varios registros internos de ella. Así, la CPU dispondrá de una bandera de habilitación global del proceso de atención a las interrupciones que mientras esté inactiva impedirá que la CPU atienda peticiones de interrupción. Además, existirán banderas de habilitación del proceso de atención de cada una de las fuentes de interrupción posibles que tiene la CPU, de manera que mientras esté inactiva dicha bandera se impedirá el proceso de atención de la interrupción asociada a ella. En conclusión, para que la CPU atienda una interrupción es necesario que ésta se encuentre activa, generándose la petición de interrupción a la CPU, debe estar permitido el proceso global de atención de las interrupciones y la interrupción en cuestión debe encontrarse habilitada.

Los pasos que da la CPU para atender una petición de interrupción activa son los siguientes (Fig. 1.29):

- Termina de ejecutar la instrucción actual y almacena en la pila la dirección de memoria en la que se ubica la siguiente instrucción que debe ejecutar (se almacena en la pila el valor del registro contador de programa, PC). Además del registro PC, otros registros internos suelen ser almacenados en la pila de la CPU de manera automática, especialmente el registro de estado, salvaguardándose así el contexto de programación. De este modo, la CPU puede concluir de forma adecuada el proceso de atención de la interrupción, recuperando el estado que tenía antes de atender la interrupción.

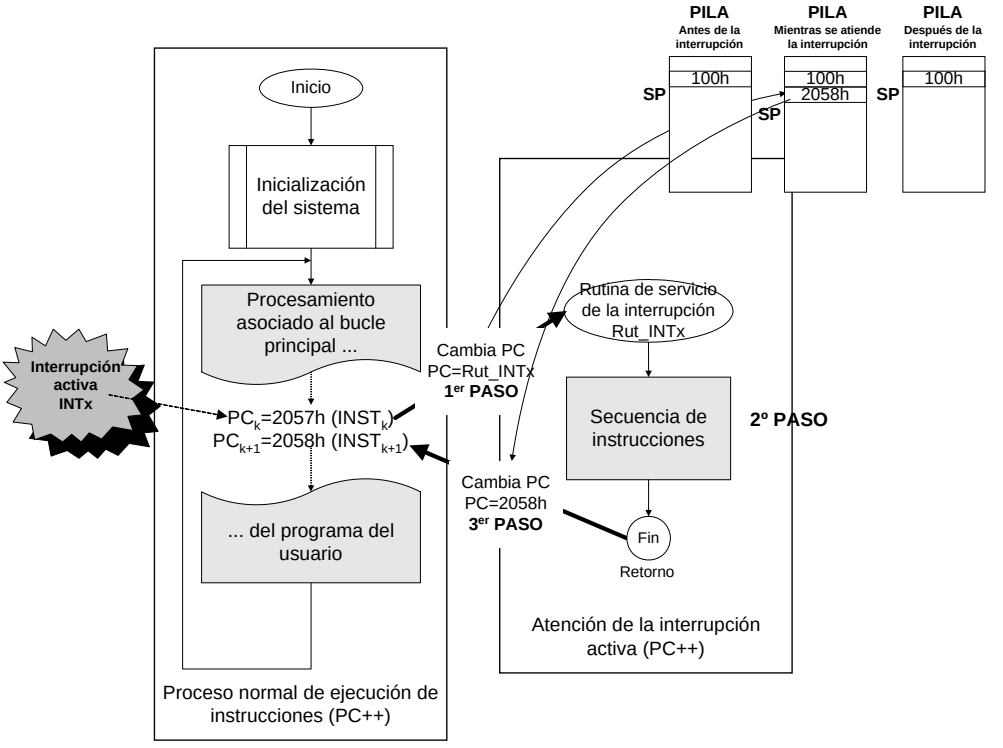


Figura 1.29. *Proceso típico de atención de una petición de interrupción activa. La interrupción, INT_x, aparece cuando la CPU está ejecutando la instrucción $INST_k$ y empieza a atenderse cuando la CPU completa la ejecución de esa instrucción. Cuando termina de ejecutarse la rutina de servicio de la interrupción, la CPU ejecuta $INST_{k+1}$.*

- Modifica el contenido del registro PC para que apunte a la dirección de comienzo de la rutina de servicio de la interrupción. A partir de este momento, la CPU comienza a ejecutar secuencialmente las instrucciones de dicha rutina. Al producirse la interrupción, la CPU reconoce quién la ha generado, y dado que tiene internamente asignados unos valores predeterminados para cada interrupción, localiza el comienzo de su correspondiente rutina. La actualización que realiza la CPU del contenido del registro PC se hace de una de las dos formas que se indican a continuación:
  - *Interrupciones vectorizadas.* En el registro PC se carga un valor predeterminado asignado a la interrupción. La primera instrucción de la rutina de servicio de la interrupción se encuentra, por tanto, en la dirección que indica dicho valor. En este caso, el rango de direcciones de memoria de la CPU dispone de una zona reservada a cada interrupción, para que el usuario ubique en ella su rutina de servicio. El espacio reservado debe permitir que el usuario introduzca como mínimo una instrucción (el salto a otra dirección del rango de direcciones de memoria donde encontrar la rutina de servicio de la interrupción).

  - *Interrupciones autovectorizadas.* En el registro PC se carga el dato contenido en la dirección que indica un valor predeterminado asignado a la interrupción. La primera instrucción de la rutina de servicio de la interrupción se encuentra en la dirección que indica el contenido de la posición de memoria asociada al valor asignado a la interrupción. En este supuesto, el mapa de memoria de la CPU dispone de una zona reservada a cada interrupción, en la que el usuario debe ubicar la dirección de comienzo de la rutina de servicio de la interrupción. El espacio reservado para cada interrupción debe tener un ancho de bits igual, o superior, al número de líneas asignadas al bus de direcciones de la CPU.

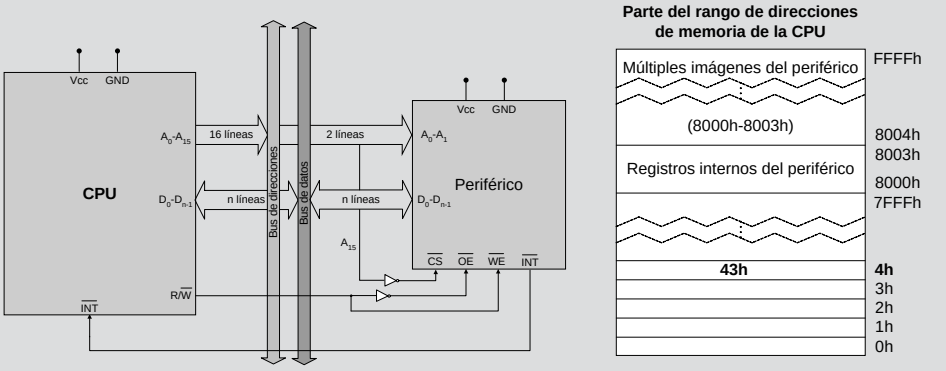
- Se ejecuta la rutina de servicio de la interrupción, terminando con una instrucción específica que tiene la CPU de retorno de interrupción. Esta instrucción provocará que el registro PC recupere de la pila la dirección que almacenó antes de comenzar el proceso de atención de la interrupción. Además del registro PC, los otros registros internos que se almacenaron en la pila al comenzar el proceso de atención de la interrupción deben ser recuperados de ella. Al concluir la ejecución de la rutina de servicio de la interrupción, la siguiente instrucción que se ejecuta es la que tocaba en el proceso secuencial normal y que no se ejecutó al comenzar el proceso de atención de la interrupción. La CPU continúa justo por donde iba antes de producirse la interrupción y la ejecución del programa principal no se ve alterada por ésta gracias a que se salvó el contexto de programación.

En cuanto al diseño de las rutinas de interrupción, como no se sabe en qué momento se va a producir la interrupción, ni el número de ellas que pueden activarse durante cada una de las iteraciones del programa, es conveniente que sean lo más breves posible y que no implementen largos algoritmos. Se evita así parar durante mucho tiempo el programa principal y que éste deje de atender otras tareas que no puedan esperar. Supongamos el caso de un periférico de conversión analógico-digital. Cuando se ha determinado el valor digital equivalente a una entrada analógica al periférico, éste puede dar aviso a la CPU mediante una interrupción para que lo lea. La CPU terminará de procesar la instrucción actual y ejecutará la rutina de servicio del CAD, que leerá el valor digital de algún registro interno del periférico sin procesarlo, guardándolo en alguna variable global, para permitir el posterior procesamiento del dato en una iteración del programa principal.

Comentar, por último, que la forma normal de atención de interrupciones suele impedir que la CPU atienda una petición de interrupción mientras está ejecutando la rutina de servicio de otra, evento al que se denomina encadenamiento o anidamiento de interrupciones. El anidamiento es un método de procesamiento permitido por la mayoría de CPU, pero que, por defecto, se encuentra inactivo y sólo debe ser utilizado por usuarios expertos del sistema microprocesador. Una mala utilización de esta técnica garantiza problemas tan graves como el desbordamiento de la pila asignada al sistema microprocesador (el tamaño de los datos que se han ubicado en ésta supera el tamaño definido por el usuario durante el proceso de compilación, almacenándose estos datos en posiciones previstas para otro uso), lo que afecta a la integridad de los datos e instrucciones almacenados en las memorias volátiles del sistema.

Diferencia entre interrupción vectorizada y autovectorizada

Supongamos una CPU que dispone de una señal de entrada ($\overline{INT}$) de petición de interrupción, que tiene habilitado el proceso de atención de dicha interrupción (siendo 4h el valor predeterminado que se le asigna) y que el rango de direcciones de memoria es el que se indica en la figura inferior. Si el proceso de atención de la interrupción es de tipo vectorizado, la CPU copia el valor 4h en el registro PC (PC = 4h) al comenzar el proceso de atención de la interrupción. Por el contrario, si es de tipo autovectorizado, en el registro PC se copia el contenido de la posición de memoria 4h (PC = 43h). Obsérvese que el periférico tiene sólo dos líneas de dirección que permiten acceder a sus registros internos y que se encuentra ubicado en la mitad superior del rango de direcciones de memoria de la CPU (desde 8000h a FFFFh), por lo que la CPU ve múltiples imágenes del periférico (hasta 8 K o 2^{13} imágenes).



Inicialización de una CPU: *reset*

Se denomina secuencia de *reset* o de inicialización de una CPU al conjunto de operaciones que debe ejecutar necesariamente la CPU para llegar a un estado inicial conocido. Para poder generar este proceso de inicialización, la CPU dispone de una señal de entrada que se suele denominar *reset*. Cuando se activa esta línea, la CPU inicializa, a un valor predeterminado, todos sus registros internos, así como el registro contador de programa, PC, que apunta al denominado vector de *reset* o de inicio del sistema. Obsérvese, por tanto, que después del proceso de *reset* la CPU tiene almacenado en el registro PC la dirección de comienzo del programa del usuario. Dependiendo de cada CPU, el arranque del sistema podrá generarse de forma vectorizada o autovectorizada, de forma análoga a los diferentes procesos de atención de las interrupciones. Para que un reset sea válido, debe permanecer activo durante un cierto número de ciclos máquina especificados por el fabricante.

1.2.6. *Controlador interno para la emulación*

Un emulador se puede definir como una herramienta de ayuda al diseño, desarrollo y puesta a punto de un sistema microprocesador. Tradicionalmente, en los primeros años

de desarrollo de los sistemas microprocesadores, los emuladores han consistido en un *hardware* conectado a un ordenador personal, más un *software* que controla dicho *hardware*. La salida de este *hardware* consiste en un conector que «imita» el conexionado eléctrico asociado al sistema microprocesador al que emula. Estos sistemas electrónicos son dispositivos extremadamente caros, respecto a otros sistemas de ayuda al desarrollo de *software* o *hardware* para un sistema microprocesador, por lo que su manejo no suele ser muy habitual.

En la actualidad, los emuladores del tipo antes comentado están cayendo en desuso debido a que los sistemas microprocesadores modernos incluyen un *hardware* de emulación, como por ejemplo el estándar JTAG (IEEE/ANSI 1149.1). Este estándar surgió inicialmente para comprobar la funcionalidad y las conexiones dentro de circuitos integrados que no tenían necesariamente que ser sistemas microprocesadores. No obstante, su uso se ha extendido, hasta generalizarse en los sistemas microprocesadores, para comprobación y emulación de la circuitería interna del sistema. En este caso, el ordenador personal se comunica (mediante puerto paralelo, serie, USB...) con una interfaz externa que se encarga de recibir las órdenes desde un *software* de depuración que se ejecuta en el propio ordenador y de transformarlas en comandos JTAG. Al otro lado de la interfaz se sitúa el sistema microprocesador, conectado a través del puerto JTAG con la interfaz externa anterior. Desde el *software* de depuración del ordenador se puede controlar completamente al sistema microprocesador: ejecución paso a paso o hasta una línea de código, colocación/eliminación de *breakpoints* en el código, visualización del contenido y posibilidad de modificar o reprogramar cualquier registro interno o memoria interna volátil, e incluso no volátil, que posea el sistema microprocesador, etc. La ventaja sobre los emuladores tradicionales estriba en que las pruebas se realizan sobre el propio sistema microprocesador, que además requiere pocas líneas de conexión. La necesidad de usar un interfaz entre el ordenador personal y el integrado se debe a que tanto el *hardware* de emulación como los comandos que se envían por el puerto JTAG son propiedad del fabricante y no se encuentran a disposición pública, por lo que es necesario adquirir un interfaz intermedio, que nos proporciona el propio fabricante del sistema microprocesador, que interprete las órdenes del PC para actuar sobre el puerto JTAG.

1.3. DISPOSITIVOS PROVISTOS DE CPU: EL DSP

En la actualidad, los dispositivos provistos de una (o varias) CPU se denominan microcontroladores, microprocesadores y DSP, y todos ellos se caracterizan por ser capaces de ejecutar programas ubicados en una memoria. Las diferencias entre ellos aparecen en el diseño realizado y dependen, a su vez, de la futura aplicación que se le piense dar al sistema electrónico:

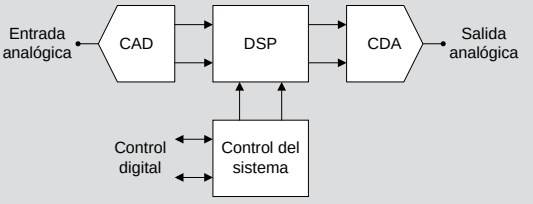
- **Microcontrolador.** Un controlador es un dispositivo cuyo objetivo es el gobierno de uno o varios procesos (por ejemplo, la gestión del nivel de un depósito de agua en el que se disponga de dos sensores que indiquen el nivel mínimo y máximo de seguridad entre los que debe encontrarse el agua). Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su imple-

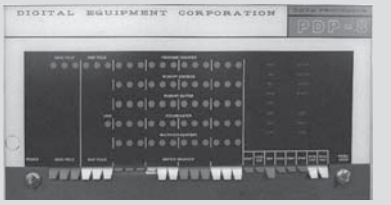
mentación física sí ha variado. En la década de los años cincuenta y sesenta, los controladores se diseñaban con componentes de lógica discreta. Posteriormente apareció la placa de circuito impreso, que incorporaba a su alrededor una CPU y varios dispositivos externos de memoria y de entrada/salida. En la actualidad, todos los elementos necesarios de un controlador se han incluido en un único chip, el cual recibe el nombre de *microcontrolador*. El microcontrolador suele incorporar, integrados en el chip, múltiples dispositivos: una CPU (normalmente no muy rápida y con una capacidad de procesamiento de información no excesivamente elevada, aunque suficiente para controlar la mayoría de los procesos reales), memorias semiconductoras de tipo ROM/PROM/EPROM/EEPROM/FLASH para contener el programa de control y RAM para almacenar los datos (dispositivos con unos tiempos de respuesta muy rápidos y con una capacidad de almacenar información baja, la suficiente para contener un programa de control y sus datos), líneas de entrada/salida para comunicarse con el exterior, así como periféricos tales como temporizadores, canales serie y paralelo, convertidores A/D y D/A, etc., fundamentales en la realización y monitorización del control de cualquier sistema físico. Las ventajas de la utilización de un microcontrolador provienen sobre todo del aumento de la fiabilidad que ofrece el sistema de control, ya que el microcontrolador necesita un reducido número de componentes externos, y de la disminución de los costes asociados, tanto en la etapa de diseño como en la realización del sistema final de control.

- **Microprocesador.** En este caso, se diseña un sistema microprocesador con el objeto de manejar un gran volumen de datos e instrucciones de forma eficiente (el ejemplo más típico de sistema microprocesador que dispone de un microprocesador son los ordenadores personales o PC). Para ello, se suele dotar al microprocesador de una CPU integrada con una capacidad de procesamiento de información elevada, lo que limita el número de periféricos que se pueden integrar en el chip. Por otro lado, los sistemas microprocesadores basados en un microprocesador suelen necesitar periféricos de almacenamiento de instrucciones y datos masivos (memorias ópticas o magnéticas), que tienen una gran capacidad de almacenamiento de información pero que, por el contrario, son lentos al transferir información con la CPU. Para evitar o mitigar los problemas derivados de la lentitud del acceso de la CPU a los dispositivos de almacenamiento masivo de información, se integran memorias semiconductoras de tipo RAM en los microprocesadores que funcionan como periféricos de memoria caché (son, por tanto, memorias que cambian continuamente su contenido, manejando siempre las instrucciones y datos a las que más recientemente ha accedido la CPU).
- **DSP.** Un procesador digital de señal es un dispositivo con capacidad de procesamiento en línea, o en tiempo real, de información que presenta, a la vez, características de microcontrolador y microprocesador. Posee una CPU de gran potencia de cálculo preparada para el tratamiento digital de señales en tiempo real y para la realización del mayor número de operaciones aritméticas en el menor tiempo posible. Por tanto, su circuitería interna ha sido optimizada para la implementación de funciones tales como el filtrado, la correlación, el análisis espectral, etc., de una señal digital de entrada al sistema.

Reseña histórica sobre los DSP

El procesamiento de señales digitales es una tecnología que podemos clasificar como «dominante» en nuestra sociedad actual. Constituye la base de lo que conocemos como electrónica de consumo, multimedia, telefonía IP, telefonía sin hilo de tipo digital, procesamiento de imágenes (especialmente con fines médicos), y actualmente se empieza a utilizar en automoción y en aplicaciones domésticas (domótica). El procesado de señales digitales en línea o en tiempo real, tal y como lo conocemos hoy día, comenzó realizándose en los años sesenta sobre máquinas VAX y DEC PDP (en 1965 se presenta el PDP-8, que podía colocarse en una mesa de laboratorio, cuando la mayoría de los computadores necesitaban una habitación con aire acondicionado y cuyas últimas versiones incorporan por primera vez estructura de bus).





En la década de los años setenta aparecieron algunas máquinas de cálculo que usaban chips de lógica discreta en tecnología TTL y que iban acompañados por una unidad aritmética programable de cuatro bits, para realizar las funciones de procesamiento de los datos en tiempo real. Estos primeros sistemas eran lentos y muy voluminosos.

El primer multiplicador integrado en un chip lo comercializa en 1973 la compañía TRW, aunque a un excesivo coste. Posteriormente, en 1978, AMI anuncia el primer circuito integrado diseñado específicamente para el procesado de señales en tiempo real, el S2811, con capacidad para procesar datos de 12 bits. Este dispositivo no llegó a alcanzar niveles importantes de venta en su comercialización. En 1979, Intel introduce el Intel 2920, al que denominó «chip procesador analógico de señales», dado que disponía de periféricos de conversión analógico-digital y digital-analógico (después de todo, se trataba de un dispositivo capaz de procesar digitalmente señales analógicas). El Intel 2920 no disponía de multiplicador interno y era extremadamente lento en la realización de cualquier operación aritmética (ciclo máquina de 600 ns) para hacerlo útil en el procesamiento de señales de audio (el ámbito de aplicación que inicialmente estaba reservado a los DSP).

El primer DSP, con una unidad multiplicadora y otra acumuladora (MAC) integradas en el chip, lo comercializa a principios de 1980 NEC (el µPD7720). Tenía un ciclo máquina de 122 ns, operando su unidad MAC en dos ciclos máquina. Se trató del primer dispositivo comercial útil en el procesamiento de señales de audio y con aplicación en el espectro audible (hasta unos 20 kHz).

Ese mismo año, Ed Cordell diseñó para Texas Instruments el boceto de lo que sería la primera arquitectura de DSP fabricada por la compañía. Dicha arquitectura fue posteriormente optimizada por Surendar Magar, para favorecer la ejecución de los algoritmos de procesamiento de la señal conocidos en la fecha. Así, en abril de 1982 se lanza comercialmente el primer DSP de la familia TMS320 de Texas Instruments, el TMS32010, diseñado en tecnología TTL. Posteriormente se incorpora la tecnología CMOS al proceso de fabricación y se complica el diseño del chip, hasta llegar a los potentes sistemas de

hoy en día, optimizados en el *hardware* para el procesado de la señal. El desarrollo inicial de los DSP por parte de Texas Instruments se puede considerar casi como un trabajo visionario, dado que la rentabilidad de estos dispositivos no aparece hasta la década de los años noventa. Hoy en día, los DSP suponen un negocio anual para Texas Instruments que supera los 5.000 millones de dólares estadounidenses (sin contar los 10.000 millones de dólares que implica la venta de periféricos específicamente diseñados para ellos).

Actualmente, la frontera entre microcontrolador, microprocesador y DSP cada vez está más diluida. Es fácil encontrar en el mercado microprocesadores y procesadores digitales de señal que incorporan memoria y periféricos internos y microcontroladores con CPU tan potentes como los de un DSP. A veces la diferencia entre ellos es nula, y llamarlos microcontrolador, microprocesador o DSP se convierte más que nada en una cuestión de *marketing.*

Inicialmente, los DSP disponían de un mercado reducido (sólo se empleaban en sistemas que requerían una elevada potencia de cálculo, como ocurre en instrumentación electrónica de precisión, osciloscopios digitales, etc.), aunque con la revolución y expansión relacionada con las telecomunicaciones, esta tendencia ha cambiado claramente (raro es el teléfono móvil o el módem que no dispone de un DSP en su interior y la familia que no dispone de alguno de estos dispositivos electrónicos). Debido al elevado coste que inicialmente suponían, su uso quedó relegado a aplicaciones en grandes sistemas. A partir de 1988, el coste decreció sustancialmente y varios fabricantes japoneses comenzaron una producción masiva de productos que incorporaban DSP, especialmente los teléfonos móviles y módems (Fig. 1.30).

En conclusión, podemos decir que un DSP es un microprocesador orientado al procesamiento de señales digitales y a la realización de cálculos a alta velocidad. Estos microprocesadores se caracterizan por tener arquitecturas especiales, orientadas a la realización *hardware* de los cálculos que otro tipo de microprocesadores implementan vía *software*, mediante la ejecución secuencial de varias instrucciones. El *hardware* de la CPU de este tipo de sistemas digitales es por ello, generalmente, mucho más complejo que el de otros microprocesadores o microcontroladores. El área de

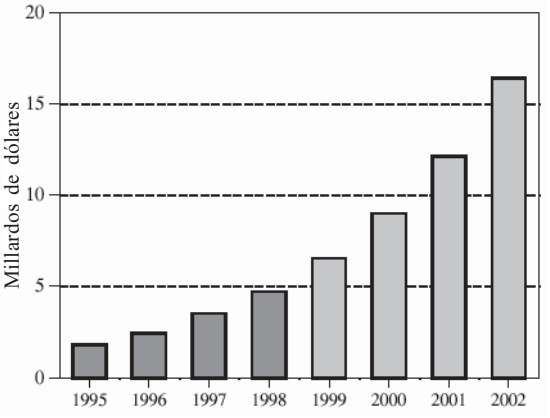


Figura 1.30. *Evolución de las ventas de DSP en los últimos años.*

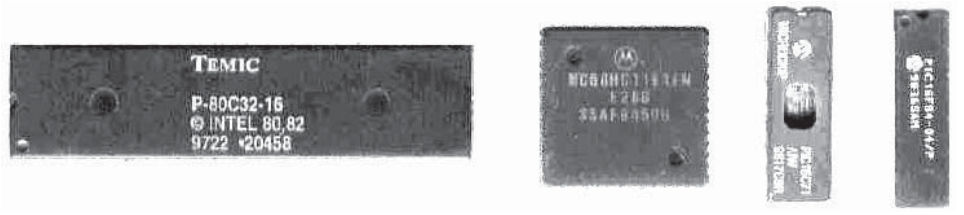


Figura 1.31. *Fotografía de varios microcontroladores: 80C32 de Intel, MC68HC11A8 de Motorola y PIC16C71 y PIC16F84 de Microchip.*

silicio es mucho mayor y por ello el coste del producto aumenta respecto a los microprocesadores y microcontroladores.

La principal diferencia de los DSP y otros procesadores modernos es que los primeros se diseñan para ser escalables, es decir, se diseñan para poder operar en paralelo con otros dispositivos similares. Para ello, se le añaden periféricos de control y bloqueo del programa (como líneas de entrada-salida que pueden bloquear la ejecución de ciertas instrucciones si se encuentran a un determinado valor) y periféricos de entrada-salida de alta velocidad (como puertos serie síncronos) que permiten la conexión sencilla de varios DSP para aplicaciones que requieran múltiples procesadores.

Las aplicaciones basadas en DSP son cada día mayores en número y cubren prácticamente todos los campos de la industria (telecomunicaciones, control, instrumentación, análisis de imagen y voz, automóvil, medicina). Esto hace que los fabricantes investiguen nuevas arquitecturas y, sobre todo, compiladores más inteligentes y mejores herramientas de desarrollo y depuración.

La principal tendencia en la mejora de las arquitecturas internas de los DSP se ha enfocado hacia el aumento del paralelismo del sistema, que es directamente proporcional al número de operaciones que el DSP será capaz de realizar en un ciclo de reloj. En la actualidad, el fabricante Texas Instruments (uno de los más importantes fabricantes de DSP) ha centrado el desarrollo de sus DSP (lo que nos daría una idea de la tendencia actual de otros fabricantes) en tres grandes familias de DSP: TMS320C2000, para aplicación en el control de procesos industriales que requieren un procesamiento



Figura 1.32. *Fotografía de un DSP, TMS230C32, de Texas Instruments, y de un microprocesador Pentium de Intel.*

complejo y un tiempo de respuesta bajo (como es el caso de las máquinas eléctricas, aquí se confunde el término DSP y microcontrolador de elevada potencia de cálculo); TMS320C5000, que incorpora optimización del consumo energético, muy necesario en telefonía móvil, con idea de aumentar la disponibilidad de un teléfono con la misma batería, y TMS320C3x y TMS320C6000, buscando una elevada potencia de cálculo en el dispositivo (para poder implementar tareas cada vez más complejas en el DSP).

La programación de este tipo de procesadores, que incorporan muchos la posibilidad de procesamiento en paralelo de los datos por parte de varios DSP, se hace cada vez más y más compleja. El empleo de lenguaje ensamblador complica en exceso el desarrollo de aplicaciones *software*. Se hace preciso (más aún que en el caso de otros microprocesadores y microcontroladores) el empleo de lenguajes de programación de alto nivel, que simplifiquen el desarrollo del *software* al usuario. Dado que estos sistemas se diseñan para el procesamiento de datos en el menor tiempo posible, este tipo de compiladores debe ser capaz de optimizar el programa en tiempo de ejecución, tarea que dificulta el desarrollo de los mismos.

Las principales alternativas que aparecen al uso de DSP son:

- ASIC de función fija o FPGA. A diferencia de cualquier microprocesador, su labor no se realiza mediante la ejecución secuencial de instrucciones, sino que se programa en la circuitería que lleva dentro. Cuando aparecieron estos dispositivos en el mercado, sus principales inconvenientes eran la falta de versatilidad (no valían para otra cosa distinta de aquella para la que fueron diseñados) y el coste de desarrollo que tenían asociado (el diseño de una aplicación basada en FPGA o ASIC era mucho más costosa en tiempo que el desarrollo de un programa en C o ensamblador para un DSP) y su utilidad principal estaba en el empleo como coprocesadores de los DSP. En la actualidad, y gracias al desarrollo de las herramientas de programación y de las tecnologías de fabricación y técnicas de integración asociada a estos dispositivos, los inconvenientes relacionados con la falta de versatilidad se han solucionado, si bien el coste de desarrollo sigue siendo superior al ligado a los DSP.
- Ordenador personal o estación de trabajo (*Native Signal Processing*). Está alcanzando popularidad por el cada vez menor coste de los sistemas PC multimedia. El sistema operativo se ejecuta en paralelo con la aplicación de procesado de la señal. Esto resta velocidad de procesado a la aplicación e impide su empleo en tiempo real, limitando su uso al procesado de la señal fuera de línea. Son más caros que las tarjetas basadas en DSP, y aunque la potencia de los microprocesadores es elevada, no pueden competir con el DSP en tiempo de ejecución de las operaciones aritméticas. El manejo de datos que realizan equivale en rapidez con el de los DSP, pero las operaciones aritméticas suelen realizarse en tiempos más elevados.
- Microcontroladores. Se diseñan principalmente para el control de procesos en tiempo real. Los microcontroladores se clasifican en función del tamaño del bus de datos. Los microcontroladores de 8 y 16 bits de bus de datos no alcanzan, en ningún caso, las velocidades de los DSP. Los de 32 bits de bus de datos son tan rápidos como los DSP y pueden emplearse en aplicaciones con pequeñas constantes de tiempo, del orden de los microsegundos, o frecuencias de trabajo elevadas. Las estructuras internas de estos microcontroladores se asemejan a la de

Tabla 1.2. *Comparación entre algunos DSP y microcontroladores*

Procesador	Bus datos	Reloj (MHz)	±ns	×ns	%µs	√µs
DSP						
TMS32010	16 bits	5	200	200	<12,8	<59
TMS32020	16 bits	5	200	200	<3,4	<59
TMS32025	16 bits	12,5	80	80	<1,32	
TMS320C30	32 bits	33	60	60	<2	<2
TMS320C40	32 bits	40	50	50	<1	<1
MC56000	32 bits	10	100	100	<3,7	
MC96002	32 bits	16,5	60	60		
Microcontroladores						
M68HC11	8 bits	8	1.000	5.000	20,5	
MCS51	8 bits	12	1.000	4.000	8	
MCS96	16 bits	16	800	1.750	2,5	
MC68020	32 bits	16	240	1.620	3,36	
MC68030	32 bits	20	200	1.400	2,8	
IAPX80286	32 bits	8	375	3.000	3,12	>2

los DSP, pero no están preparados para el trabajo en paralelo mediante arquitecturas multiprocesadores, sino que están pensados más bien para el funcionamiento en solitario y con el menor número de dispositivos externos (a ser posible, el producto final dispondrá de un único integrado, el propio microcontrolador). Lo cierto es que las diferencias entre los microcontroladores de elevada potencia de cálculo y los DSP son casi nulas.

En la Tabla 1.2 se muestran los tiempos empleados por varios DSP y microprocesadores en realizar algunas operaciones aritméticas básicas.

1.4. ESTRUCTURA INTERNA BÁSICA DE UN DSP

Los DSP que se comercializan se pueden clasificar, según su funcionalidad, como sistemas CISC (las instrucciones son complejas y requieren de varios ciclos de reloj para poder ser ejecutadas por la CPU) y sistemas RISC, aunque se pueden encontrar en el mercado DSP de tipo SISC (dedicados a aplicaciones concretas como telefonía móvil, etc.). Su estructura interna básica responde fundamentalmente a una arquitectura de tipo Harvard, en la mayoría de los casos mejorada y optimizada para acelerar la ejecución de las instrucciones y la realización de las operaciones aritméticas (buses de datos para trasvase de instrucciones o datos de tamaño superior al estrictamente

necesario, más de un bus de direcciones y de datos para el acceso a los datos, implementación de técnicas de paralelismo para favorecer la segmentación y la ejecución de varias operaciones elementales por ciclo máquina, operadores lógicos y aritméticos avanzados, etc.).

La zona de manejo de datos de la CPU suele estar especialmente diseñada. Esta zona dispone habitualmente de múltiples ALU y multiplicadores, capaces de realizar varias operaciones aritméticas en un único ciclo máquina del sistema. De entre todos los registros internos de esta zona, destaca el registro acumulador (operando fuente y destino de los operadores aritméticos). Si la CPU dispone de un único registro acumulador, cada vez que se realice una operación, debe salvarse el resultado obtenido si queremos que éste no se pierda en el futuro, como consecuencia de alguna operación que se vaya a realizar. Las CPU que disponen de un registro acumulador se dice que tienen una estructura interna orientada a acumulador (Fig. 1.33) y su potencia de cálculo se ve limitada por las continuas transferencias que hay que realizar para salvar el valor contenido en el acumulador, o para cargar en el acumulador un nuevo operando. Las CPU de mayor potencia de cálculo (caso de los DSP), gracias a que disponen de varios registros que pueden funcionar como acumulador, no se ven tan limitadas en su potencia de cálculo como las anteriores. Los DSP disponen de tablas de registros internos que funcionan como acumuladores y que se pueden emplear como variables de almacenamiento temporal de datos, con lo que disminuye el número de transferencias entre el acumulador de la CPU y la memoria de datos. Disponen de una estructura interna que se denomina orientada a registro (Fig. 1.34).

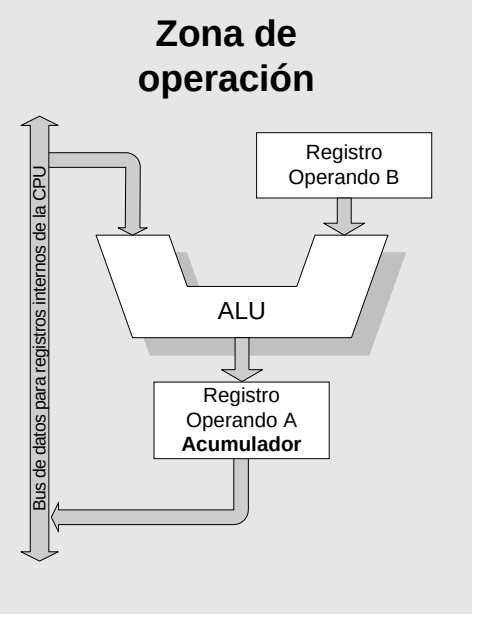


Figura 1.33. *Estructura interna de manejo de datos de tipo «orientada a acumulador».*

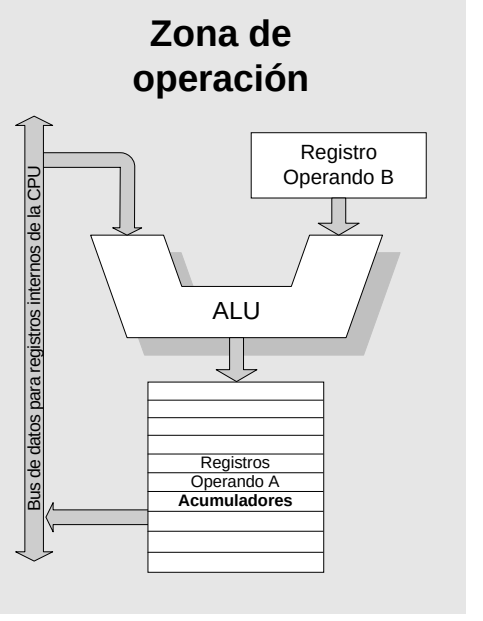


Figura 1.34. *Estructura interna de manejo de datos de tipo «orientada a registro».*

1.5. PRINCIPALES APLICACIONES Y FABRICANTES DE DSP

Las principales aplicaciones que podemos destacar actualmente se enumeran en la Tabla 1.3.

Tabla 1.3. *Principales aplicaciones de los DSP*

Telecomunicaciones y campos relacionados	
Telecomunicaciones: • Transmisión de voz: teléfonos móviles (GSM), celulares e inalámbricos. • Cancelación de ecos. • ADPCM Transcoders. • PBX digitales. • ATM. • Repartidores de línea. • Multiplexado de canales. • Módems de alta velocidad. • Ecualizadores adaptativos. • Fax. • Interpolación digital del habla. • Conmutación de paquetes. • Transmisión de voz. • Adaptadores de terminal. • Controladores HDLC. **Instrumentación:** • Analizadores de espectro. • Generadores de función. • Osciloscopios digitales: procesado de datos. • Procesado de datos sísmicos.	**Análisis de voz y conversación:** • Compresión de voz. • Codificación de conversación. • Reconocimiento de voz. • Enriquecimiento de voz • Conversión de texto en voz, y viceversa. • Encriptado de voz. **Procesamiento de imágenes:** • Animación. • Estaciones de trabajo. • Rotación tridimensional. • Reconocimiento de patrones. • Proceso homomórfico. • Compresión y transmisión de imágenes. **Aplicaciones digitales:** • Ventanas de adquisición. • Convolución. • Correlación. • Transformación de Hilbert. • Transformada rápida de Fourier. • Filtrado digital. • Generación de formas de onda.
Otros campos	
Aplicaciones de control: • Robótica. • Regulación en velocidad de motores. • Servocontrol. • Impresoras.	**Automóvil:** • Cancelación de ruidos. • Suspensión activa. • ABS. • Máquinas. • Navegación. • Control de entrada de voz. • Posicionamiento. • Análisis de vibraciones.

(Continúa)

Tabla 1.3. *Principales aplicaciones de los DSP* (Continuación)

Otros campos	
Consumo: • Sistemas de respuesta. • Autorradios. • Juguetes. • Herramientas. • Televisión y música digital. • Sintetizadores musicales.	**Militar:** • Navegación. • Sónar. • Procesado de imágenes. • Radar. • Guía de misiles. • Seguridad en las comunicaciones.
Medicina: • Ayuda a deficientes auditivos. • Monitorización de pacientes. • Equipos de ultrasonido. • Monitor fetal.	**Industria:** • Medida y control. • Robots. • Control numérico. • Seguridad de acceso. • Medida de la red de alimentación.

Por otro lado, algunos de los principales fabricantes de DSP, y por ende de circuitos electrónicos digitales, que existen en la actualidad son (el orden en que se disponen dichos fabricantes obedece simplemente a la ordenación alfabética de los mismos, en ningún caso se pretende realizar una ordenación que se salga del objetivo anterior):

1. Analog Devices.
2. Hitachi.
3. Motorola.
4. NEC.
5. SGS-Thomson.
6. Texas Instruments.
7. Zilog.

Cada fabricante introduce en el mercado sus DSP, con características similares. Algunos de los principales productos que ofrecen estos fabricantes se muestran en la Tabla 1.4.

Tabla 1.4. *Ejemplos de dispositivos DSP comerciales*

Fabricante	DSP 16 bits	DSP 24 bits	DSP 32 bits
Analog Devices	ADSP-2100 ADSP-21cspxx		ADSP-21020 ADSP-2106x
Hitachi	SH-DSP		
Motorola	DSP561xx DSP568xx	DSP5600x DSP563xx	DSP96002

(Continúa)

Tabla 1.4. *Ejemplos de dispositivos DSP comerciales* (Continuación)

Fabricante	DSP 16 bits	DSP 24 bits	DSP 32 bits
NEC	µPD7701		
SGS-Thomson	D950		
Texas Instruments	TMS320C1x TMS320C2000 TMS320C5000 TMS320C8x		TMS320C3x TMS320C4x TMS320C6000
Zilog	Z893xx Z894xx		

1.6. EJERCICIOS PROPUESTOS

1.1. El matemático estadounidense de adopción, húngaro de origen, John Louis von Neumann contribuye de forma notable al diseño de los sistemas microprocesadores. Indique la principal aportación de este científico en el desarrollo de dichos sistemas.

1.2. Reflexione sobre la estructura maestro-esclavo de un sistema microprocesador. Describa las principales características de un dispositivo periférico, frente a una CPU, en dicho sistema.

1.3. Los dispositivos de un sistema microprocesador se comunican entre sí por un conjunto de líneas denominadas buses del sistema. Enumere los diferentes buses que aparecen en el sistema microprocesador y describa las principales características de cada uno de ellos.

1.4. Indique las principales diferencias entre una arquitectura clásica de tipo von Neumann frente a otra de tipo Harvard.

1.5. ¿En qué se diferencia el rango de direcciones de memoria de una CPU del mapa de memoria del sistema microprocesador en el que se encuentra dicha CPU?

1.6. ¿Qué utilidad tienen los estados de espera en la gestión de los accesos de una CPU a los periféricos externos de su sistema microprocesador?

1.7. Históricamente, las arquitecturas de tipo CISC han dejado paso a otras arquitecturas microprocesadoras más simples en cuanto al tipo de instrucciones que procesa la CPU. ¿Cómo se denominan estas últimas arquitecturas microprocesadoras?

1.8. Las modernas arquitecturas microprocesadoras de tipo VLIW y superescalar se definen como supersegmentadas. ¿Cuál es la principal diferencia entre una y otra?

1.9. ¿Qué es y para qué sirve en la actualidad un controlador interno para la emulación en un microprocesador?

1.10. Los términos microprocesador, microcontrolador y DSP identifican a dispositivos digitales provistos de CPU. Enumere las principales diferencias entre ellos.