Capítulo 2.

Descripción general de un DSP.

1. DESCRIPCIÓN DE UN SISTEMA MICROPROCESADOR

Un sistema microprocesador es 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 siguientes:

1.1 Periférico en 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[1] de datos (normalmente 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[2], como puede ser la lectura de variables analógicas y el cálculo de su correspondiente valor digital (periféricos de conversión analógico–digital), etc.

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

[1] Periféricos denominados periféricos de memoria.

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

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 ó 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.
- **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 datos (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, entre las que se encuentran:
  - 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, pueden 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 Unidad de Control de Procesos (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.

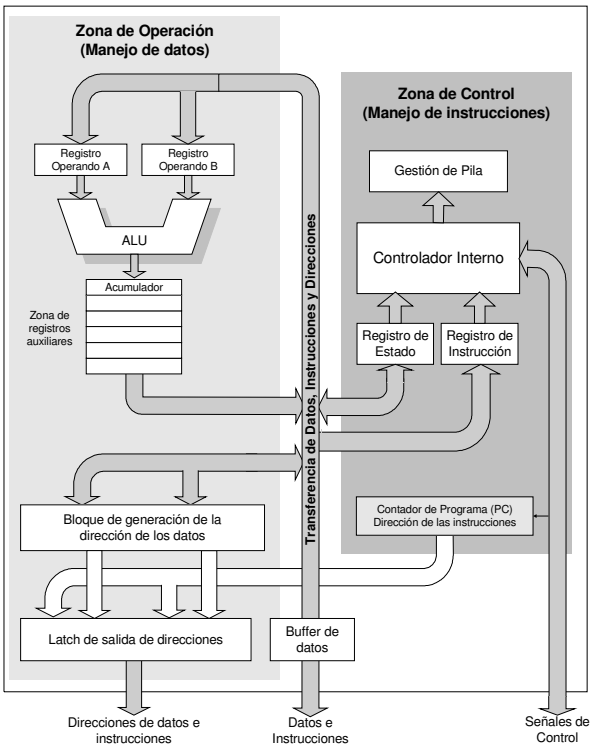

Fig. 2.1. Estructura básica de una CPU.

1.2.1 Zona de 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 *micro–órdenes*[1] 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 micro–órdenes que controlan, a su vez, 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:

[1] Conjunto de órdenes básicas que es capaz de implementar la CPU de un sistema microprocesador. Estas micro–órdenes 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.

- 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 micro−órdenes necesarias para su ejecución. Aunque las instrucciones son palabras 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.
- 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 donde 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 y con idea de ejecutar otro conjunto de instrucciones que no se ubican detrás de las que se encuentra ejecutando la CPU, la secuencia normal de ejecución del programa y cambiar el valor del registro PC; por ejemplo, al ejecutar un bucle, llamar a una subrutina o cuando aparecen interrupciones en el sistema microprocesador. Para la realización de esta tarea, la CPU dispone de un método automático de salvaguarda del registro PC y, por extensión, de cualquier otro dato 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 siguiente figura se muestra el proceso de ruptura de la secuencia normal de ejecución de instrucciones de una CPU por la ejecución de una llamada a una subrutina. A modo de ejemplo, y por simplicidad, 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.

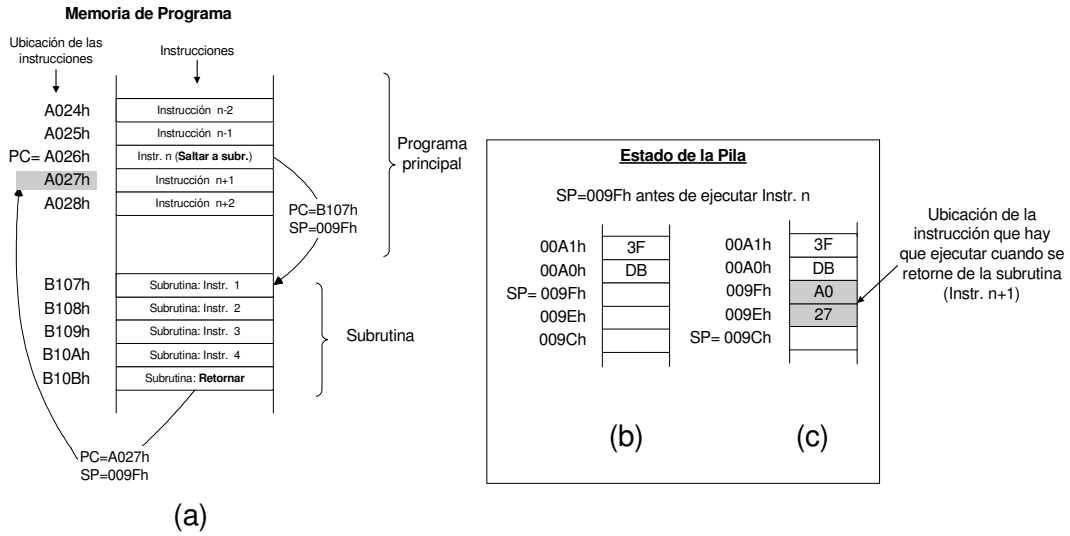

Fig. 2.2. Proceso de salto a una subrutina: a) flujo del programa, b) estado de la pila antes y después de ejecutar la subrutina y c) estado de la pila mientras se ejecuta la subrutina.

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. Existen, 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.

1.2.2 Zona de 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, ...) y aritmético (suma y resta normalmente y, en algunos casos, multiplicación) 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 estos 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[1], todo un conjunto de registros internos que pueden funcionar como acumuladores[2].

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

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

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

trabajar estos operadores auxiliares y que facilitan la generación de 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 faciliten el acceso a tablas.

1.3 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**. 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 externos 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 (64K instrucciones y datos). 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**.

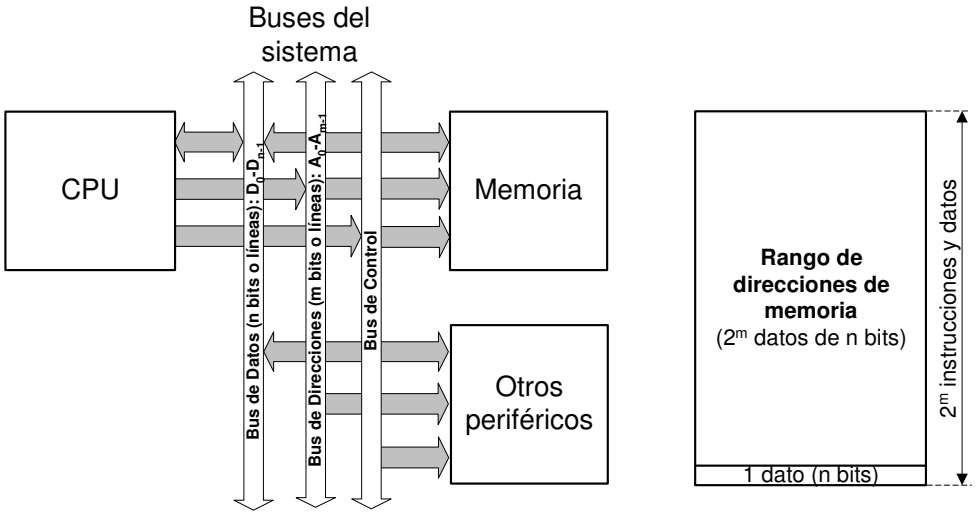

Fig. 2.3. Estructura básica de un sistema microprocesador.

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 en lectura o escritura, la CPU comienza, activa 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 cronogramas se analizarán en detalle cuando se estudie la comunicación entre la CPU y los periféricos de su sistema microprocesador.

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** (con un juego complejo de instrucciones que disponen de más de 80 instrucciones, algunas de las cuales son muy complejas y potentes y requieren muchos ciclos para ejecutarse), **RISC** (con un juego reducido de instrucciones, disponiendo de pocas instrucciones que se ejecutan normalmente en un ciclo y que permiten optimizar el hardware del sistema) y **SISC** (con un juego específico de instrucciones, o CPUs 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 estos, 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. El sistema microprocesador

accede a la memoria de programa para recoger la instrucción a ejecutar; posteriormente descodifica la instrucción para determinar la secuencia de operaciones 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. 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 ejecutando, la CPU está recogiendo otras instrucciones y preparándose para ejecutarlas, lo que disminuye el tiempo real o efectivo de ejecución de cada instrucción.

1.4 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:

- Que la CPU seleccione en modo lectura o escritura al periférico (active la señal de selección o habilitación del periférico, $\overline{CS}$ ó $\overline{CE}$, y especifique el tipo de acceso que debe atender el periférico, active las señales $R/\overline{W}$, $\overline{OE}$ ó $\overline{WE}$). 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 de o escriba en un registro interno del periférico.
- Que la CPU distinga entre el acceso a uno u otro registro interno del periférico, es necesario las líneas asociadas al bus de direcciones del sistema microprocesador 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.

1.5 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: hay que evitar conflictos al conectar periféricos a la CPU a través del bus de datos y 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 escuchan (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, $\overline{CS}$ ó $\overline{CE}$: 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[1]. 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.

Por tanto, 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.

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

[1] Normalmente se empleará un decodificador o una PAL.

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 y, por tanto, 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 en 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 *0nseg*.

Las CPUs 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 el ciclo máquina[1] de la CPU suele ser lo suficientemente grande (y el ciclo de bus[2] de la CPU lo suficientemente lento) 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 consumo−, está en torno a los *10nseg*, 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 *25nseg*.

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 CPUs 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

[1] 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.

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

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 máquina 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 que se suele denominar $\overline{RDY}$. 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 $\overline{RDY}$, realizada un ciclo máquina después de la comprobación previa. El proceso anterior se repita hasta que, en una comprobación, la CPU detecta que la señal está activada, completando entonces el acceso. La generación de los estados de espera en una CPU se suele realizar (la señal $\overline{RDY}$ de entrada a la CPU se suele obtener) a partir de la combinación lógica de dos señales:

La señal de fin de cuenta que genera un contador con precarga que existe dentro de la propia CPU (generación de una señal $\overline{RDY}$ interna en la propia CPU).

La señal de fin de cuenta que genera algún dispositivo externo a la CPU (generación de una señal $\overline{RDY}$ externa). Esta señal es de entrada a la CPU y se suele asociar al bus de control del sistema microprocesador.

1.6 La jerarquía de memoria: memoria CACHÉ y periférico DMA

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.

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. Hoy en día, el sistema microprocesador precisa, por tanto, de dispositivos de memoria de gran capacidad de almacenamiento, a pesar de ser más lentos en su acceso que la propia CPU. En cualquier caso, existen otros dispositivos de memoria de acceso mucho más rápidos y adecuados a los tiempos de acceso de la CPU, aunque con mucha menor capacidad de almacenamiento de información que los primeros.

Para conseguir mejorar el rendimiento del sistema microprocesador, la transferencia de datos e instrucciones entre la CPU y su memoria y disminuir su coste, los diseñadores de estos sistemas aprovechan las dos características esenciales asignadas a cualquier programa que se realice para ser ejecutado en una CPU, las denominadas reglas de localidad (localidad temporal y espacial). Por un lado, la localidad temporal determina que, si un elemento de la memoria de programa o dato ha sido accedido recientemente, pronto volverá a ser accedido. Por otro, la localidad espacial indica que, si la CPU acaba de acceder a un determinado elemento de la memoria de programa o dato, seguidamente accederá a otro elemento ubicado cerca del anterior. La utilización de las características asociadas a ambas reglas en el diseño del sistema microprocesador han permitido la aparición de estos sistemas a un coste razonable, una elevada capacidad de almacenamiento y un alto rendimiento. Ha sido necesario, para ello, organizar la memoria de forma jerárquica, por niveles, apareciendo el concepto de “jerarquía de memoria”.

La jerarquía de memoria pretende sacar partido a ambas reglas, aumentando el rendimiento del sistema. Así, para aprovechar la localidad temporal, el sistema debe mantener cerca de la CPU las instrucciones y los datos usados más recientemente, mientras que, para aprovechar la localidad espacial, el sistema debe ubicar cerca de la CPU, un bloque de datos o instrucciones contiguos a un dato o instrucción que esté usando la CPU. El establecimiento de la jerarquía de memoria implica ubicar una pequeña fracción de la memoria total del sistema, la memoria más rápida y con menores tiempos de acceso, para que sea directamente accesible por la CPU. En esta memoria, denominada memoria CACHÉ, se almacenarán los datos e instrucciones que está usando la CPU. Si una instrucción o dato no se encuentra en ella, será necesario acercarlo a la CPU desde una zona de memoria de mayor nivel en la jerarquía, más capacidad de almacenamiento y superiores tiempos de acceso (memoria principal o secundaria −formada por dispositivos que se conectan a la CPU mediante sistemas de entrada/salida como ocurre con los discos duro, …−). La tasa de aciertos en los accesos que realiza la CPU a la memoria CACHÉ determina el rendimiento del sistema microprocesador. Si ésta es alta, la jerarquía de memoria ofrece una velocidad efectiva cercana a la de la memoria más rápida y de menor capacidad, aquella que se ubica cercana a la CPU conformando el nivel más bajo de la memoria. Por el contrario, si la tasa de aciertos es baja, la velocidad efectiva de la jerarquía de memoria será mucho menor que la velocidad de la memoria CACHÉ y la jerarquía de memoria será poco eficiente. Las memorias CACHÉ cambian continuamente su contenido, manejando siempre los conjuntos de instrucciones y datos a los que más recientemente ha accedido la CPU. La ubicación en la jerarquía de un dispositivo memoria determina su función y cómo se realiza el acceso al mismo; el rol que se aplica a cada memoria es diferente y dependerá del nivel en el que se encuentre en la jerarquía.

Establecida la jerarquía de memoria, otros dispositivos o periféricos que intervienen en la mejora del rendimiento del sistema son los periféricos DMA. Estos periféricos pueden realizar, una vez han sido programados por la CPU, transferencias en la jerarquía de memoria para acercar información, principalmente datos, a la CPU sin que ésta intervenga. Para ello, disponen en muchos casos de buses de dirección, datos y control propios.

1.7 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 período 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) permitieron su uso en la implementación de la CPU aunque aún no se podían usar como soporte de la memoria, que se implementaron 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 períodos 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, surgieron los sistemas microprocesadores de tipo CISC. Estos sistemas incorporan complejas instrucciones que tardan en completarse varios ciclos de reloj y que requieren, para su ejecución, la realización de numerosas operaciones elementales. De esta manera, se consigue 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 requiere, para ser ejecutada por la CPU, 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) dio 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 mayores 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, que comienza en 1981 y se extiende hasta la actualidad, 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, ayudar a los compiladores a analizar los datos y controlar los flujos de instrucciones de forma óptima.

El diseño de los sistemas microprocesadores se encuentra en la actualidad 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 microprocesador y explotar al máximo la tecnología existente. 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 ó 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 CPUs 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 aquel 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[1] podemos destacar las siguientes:

Los denominados sistemas microprocesadores supersegmentados. Representan un ejemplo típico de arquitectura microprocesadora que incorpora un paralelismo 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, Se consigue, de esta manera, disminuir el tiempo necesario para la realización de cada fase y el ciclo máquina de la CPU.

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

[1] 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.

mediante técnicas de paralelismo que suponen innovaciones en la planificación de la ejecución de instrucciones que realiza el hardware.

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.

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.

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.

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 necesario la búsqueda de modelos para poder clasificarlos y compararlos. Así, en 1966, 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**, 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**, con un único flujo de instrucciones y múltiples flujos de datos. El coste de un sistema microprocesador con múltiples CPUs 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**, 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**, con múltiples flujos de instrucciones y datos. Se trata de sistemas que disponen de varias CPUs, cada una de las cuales ejecuta un conjunto diferente de instrucciones sobre distintos conjuntos de datos.

1.8 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 en 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 direccionamiento. 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.
- **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. 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 100 posiciones de memoria más 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 memoria que se incluye como dato en la instrucción que se está ejecutando. Por ejemplo: salta a la posición de memoria 8100h.

1.9 Interrupciones e inicialización

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 recogiendo continuamente 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.

1.9.1 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:

- 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.
- 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.

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:

- 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, para permitir una conclusión adecuada del proceso de atención de la interrupción.

- 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 quien 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 la interrupción.

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 breve 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 CPUs pero que, por defecto, se encuentra inactivo y sólo debe ser utilizado por usuarios expertos del sistema microprocesador. Un mal uso de esta técnica garantiza problemas tan graves como el desbordamiento de la pila asignada al sistema microprocesador, lo que afecta a la integridad de los datos e instrucciones almacenadas en las memorias volátiles del sistema.

1.9.2 Inicialización, reset

Por otro lado, 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.

2. DISPOSITIVOS PROVISTOS DE CPU: EL DSP

En la actualidad, los dispositivos provistos de una (o varias) CPU se denominan microcontroladores, microprocesadores y DSPs 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 implementación física si ha variado. En la década de los años 50 y 60, 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), memoria semiconductoras de tipo ROM/PROM/EPROM/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 PCs). 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 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.

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 CPUs 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 modem 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 DSPs, especialmente los teléfonos móviles y modems.

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

La principal diferencia de los DSPs 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 DSPs para aplicaciones con multiprocesadores.

Las aplicaciones basadas en DSPs 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 interna de los DSPs se enfocó inicialmente hacia el aumento del paralelismo del sistema. Recientemente han aparecido en el mercado DSPs con múltiples CPUs que pueden trabajar en paralelo (familia de DSPs TMS320C8x de Texas Instrument), grado de paralelismo 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 DSPs) ha centrado el desarrollo de sus DSPs (lo que nos daría una idea de la tendencia actual de otros fabricantes) en tres grandes familias de DSPs, TMS320C2000 para aplicación en el control de procesos industriales que requieren un procesamiento 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 DSPs, 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 DSPs, son:

- **ASICs de función fija ó FPGAs**. 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 FPGAs ó ASICs 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 DSPs. 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 DSPs.
- **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 DSPs 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 DSPs 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 DSPs. Los de 32 bits de bus de dato son tan rápidos como los DSPs 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 los DSPs 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 posibles, 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 DSPs son casi nulas.

3. ESTRUCTURA INTERNA BÁSICA DE UN DSP

Los DSPs 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), aunque se pueden encontrar en el mercado DSPs 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 ALUs 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 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 DSPs), 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 DSPs 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.