Antiguamente, era muy común que las placas base viniesen equipadas con soquetes apropiados para la memoria caché, que permitían añadir más memoria caché. Los módulos adicionales, llamados módulos *COAST* (*Cache On A STick*) eran relativamente accesibles, haciendo que mucha gente realizase la ampliación. Sin embargo, actualmente esta posibilidad ya no existe, pues la gran mayoría de los procesadores ya traen la memoria caché *L2* integrada, sin permitir ninguna modificación, ya que no es aconsejable abrir el procesador y soldar más memoria caché.

O sea que, actualmente, la cantidad de memoria caché que deseemos tener en el procesador, o raramente en la placa base, debe ser decidida antes de realizar la compra del equipo, eligiendo entre las distintas opciones disponibles. Una vez adquiridos el procesador y la placa base no nos será posible hacer ninguna modificación al respecto de forma simple y sencilla.

3.1.3 Procesadores RISC y procesadores CISC

Siempre ha existido una gran polémica en torno a cuál de estas dos plataformas es mejor. Tal vez podamos considerar inútil estar hablando sobre esto, pero es interesante comprender la diferencia entre estas dos plataformas para entender varios aspectos de los procesadores modernos.

Un procesador *CISC* (*Complex Instruction Set Computer*, u ordenador con un conjunto complejo de instrucciones), es capaz de ejecutar varios centenares de instrucciones complejas diferentes, siendo extremadamente versátil. Algunos ejemplos de procesadores *CISC* son el *386* y el *486*.

A inicios de la década de los años 80, la tendencia era construir chips con conjuntos de instrucciones cada vez más complejos. Sin embargo, algunos fabricantes consideraron seguir el camino contrario, creando el formato *RISC* (*Reduced Instruction Set Computer*, u ordenador con un conjunto reducido de instrucciones). Al contrario de los complejos procesadores *CISC*, los procesadores *RISC* sólo son capaces de ejecutar algunas instrucciones simples. Justamente por esto, los chips basados en esta arquitectura son más simples y más baratos. Otra ventaja de los procesadores *RISC*, es que al tener un número menor de circuitos internos, pueden trabajar a frecuencias más elevadas. Un ejemplo son los procesadores *Alpha,* que en el año 97 ya podían trabajar a unos nada despreciables 600 MHz para la época.

Puede parecer extraño que un chip que es capaz de ejecutar pocas instrucciones pueda ser considerado, por muchos, más rápido que otro que ejecuta centenares de instrucciones. Pero un procesador *RISC* es capaz de ejecutar tales instrucciones de forma mucho más rápida. La idea principal es que a pesar de que un procesador *CISC* sea capaz de ejecutar centenares de instrucciones diferentes, sólo algunas son usadas de forma frecuente.

Entonces, podríamos crear un procesador optimizado para ejecutar sólo las instrucciones simples que se utilizan con una mayor frecuencia. Como de cualquier forma, poca

gente programa directamente en lenguaje *ensamblador*, bastaría alterar los compiladores para que los programas fuesen compatibles con los nuevos procesadores.

Es indiscutible, sin embargo, que los procesadores *CISC* son mejores en la mayoría de tareas, principalmente por su gran número de recursos. Por eso, en vez de la consolidación de una de las dos tecnologías, actualmente vemos procesadores híbridos, que son esencialmente procesadores *CISC* pero que incorporan muchos recursos encontrados en los procesadores *RISC* (o viceversa).

Examinando desde un punto de vista un poco más práctico, la ventaja de una arquitectura *CISC* es que ya tenemos muchas de las instrucciones guardadas en el propio procesador, lo que facilita el trabajo de los programadores, que ya disponen de prácticamente todas las instrucciones que serán usadas en sus programas. En el caso de un chip estrictamente *RISC*, el programador tendría un poco más de trabajo, pues sólo dispondría de instrucciones simples y tendría que combinar varias instrucciones siempre que necesitase ejecutar alguna tarea más compleja.

En los chips actuales, que son una mezcla de las dos arquitecturas, unimos las dos capacidades. Internamente, el procesador lleva a cabo sólo instrucciones simples. Estas instrucciones internas, variando en función del tipo de procesador, se adaptan al proyecto del chip. Por ejemplo, las instrucciones internas de una *K6* son diferentes a las de un *Pentium*. Sobre estas instrucciones internas disponemos de un circuito decodificador, que convierte las instrucciones complejas utilizadas por los programas, en varias instrucciones simples que pueden ser entendidas por el procesador.

El conjunto básico de instrucciones usadas en los equipos PC es conocido por el conjunto *x86*. Este conjunto está compuesto por un total de 187 instrucciones, que son las más utilizadas por todos los programas. Además de este conjunto principal, algunos procesadores también traen añadidas instrucciones alternativas, que permiten a los programas ejecutar algunas tareas de forma más rápida de lo que sería posible sólo con el uso de las instrucciones *x86*. Algunos ejemplos de conjuntos alternativos de instrucciones son el *MMX* (usado a partir del *Pentium MMX*), el *3D-Now!* (usado por los procesadores de la casa AMD a partir del *K6-2*) y el *SSE* (soportado a partir del *Pentium III*).

3.1.4 Front End y Back End

Cualquier procesador actual puede ser dividido en dos bloques básicos, *Front End* y *Back End*. El *Front End* corresponde a los circuitos que decodifican las instrucciones, es el caso del *Hardware decoder* y el *Microcode decoder* junto con algunos componentes más, como los circuitos de *Branch Prediction*, que ordenan las instrucciones de forma que el procesador pueda procesar el mayor número posible de instrucciones por ciclo, y la memoria caché *L1*. Estos componentes son la "puerta de entrada" del procesador, teniendo la función de preparar las instrucciones para que sean realizadas a continuación por el procesador.