CISC FRENTE A RISC, UNA BATALLA EN BLANCO Y NEGRO

Habiendo tratado ya parte el inicio de la computación, hoy entraremos en una época algo más moderna. Muchos la considerarán la época dorada de la computación, aquella en la que germinó el concepto de los ordenadores actuales y se empezó a fraguar la era más contemporánea de la información.

Situémonos en los últimos años de la década de los 50: los procesadores se desarrollaban de forma completamente independiente, haciendo que un mismo programa tuviese que ser modificado para ejecutarse en máquinas diferentes. Es evidente que era una situación que había que cambiar.

La operación más sencilla para un procesador

En la primera entrada hablamos de la ISA (Instruction Set Architecture) y pusimos dos ejemplos: ARM y x86 son las dos ISA más utilizadas en la actualidad en los procesadores de Intel, Qualcomm, ARM, Samsung, etc.

Una ISA esta formada por múltiples componentes, tales como los tipos de datos que maneja un procesador, los registros y sus tamaños, los buffers e incluso los errores que es capaz de manejar. Uno de los componentes fundamentales de cada ISA son el conjunto de instrucciones que admite.

Estas instrucciones son fundamentales e imprescindibles, ya que es lo que el procesador ejecuta.

Dependiendo de la ISA existen múltiples tipos de instrucciones, aunque principalmente se engloban dentro de tres grandes categorías: operaciones con memoria, operaciones aritméticas y operaciones de control sobre la CPU.

Llegados a este punto, la complejidad del set de instrucciones es variable y depende enormemente de lo que sus diseñadores decidieran en el día de su creación. En nuestro ejemplo del algoritmo real del huevo frito recordaréis que teníamos un Paso 6: Verterlo con cuidado sobre el aceite caliente, que es una de sus instrucciones. Si realizamos una aproximación desde el punto de vista de un procesador esa sería una instrucción compleja que podría dividirse en varias instrucciones más simples:

Paso 6.1: Colocar el huevo partido sobre la sartén.
Paso 6.2: Acercar el huevo partido a un par de centímetros del aceite caliente.
Paso 6.3: Mover verticalmente el huevo partido.
Paso 6.4: Verter el contenido del huevo partido sobre el aceite hasta que esté vacío.
Paso 6.5: Retirar el huevo partido y ya vacío.
Paso 6.6: Tirar a la basura el huevo partido y ya vacío.

Como veis estamos realizando la misma labor con dos juegos de instrucciones diferentes. La primera instrucción Paso 6 es más compleja que las nuevas instrucciones Paso 6.X, que son más simples y sencillas de entender pero también son muchas más en cantidad. Elegir entre una u otra dependerá del set de instrucciones que admite el procesador.

Este razonamiento es la base para entender lo que ocurrió con CISC a mediados del siglo XX, cuando IBM se propuso unificar las instrucciones con las que trabajaban los procesadores. Unos años más tarde y teniendo en cuenta su experiencia, introdujo con otro enfoque: RISC. Ambas aproximaciones se fundamentan sobre la misma base de funcionamiento (un bucle infinito en el que en un mismo ciclo, un procesador recibe una nueva instrucción a ejecutar, la decodifica, la ejecuta y espera a la siguiente instrucción) pero son muy diferentes en cuanto al tipo de instrucción que admiten.

CISC acude en la búsqueda de lo más completo

En los años 50, todos los computadores se diseñaban de forma completamente aislada unos de otros. Esto hacía que sus instrucciones fuesen independientes, haciendo que un programa escrito para un cierto ordenador no se pudiese ejecutar en otro. A finales de la década, IBM reunió a un grupo de sus investigadores para estudiar la forma con la que un programa pudiese trabajar en múltiples computadores sin importantes cambios, ampliando la compatibilidad del software en diferentes máquinas. El resultado fue el enfoque CISC, Complex Instruction Set Computing, introducido por primera vez en los IBM System/360 el 7 de abril de 1964.
IBM System-360 Un System/360 en una fábrica de Volkswagen, año 1973 (vía Wikipedia)

CISC ofrece un conjunto de instrucciones bastante completas y lentas de ejecutar, pero que agrupaban varias operaciones de bajo nivel en la misma instrucción. Esto da lugar a programas pequeños y sencillos de desarrollar que además realizaban pocos accesos a memoria: esto que ahora podría parecer insignificante era vital en aquella época, cuando los ordenadores trabajaban con muchos menos recursos que los equipos actuales. En el algoritmo de ejemplo del huevo frito, con un enfoque CISC tendríamos una única instrucción: nuestro Paso 6: Verterlo con cuidado sobre el aceite caliente.

En la actualidad CISC tiene a x86 como su mayor exponente, con AMD y sobre todo Intel a la cabeza de su desarrollo. Hay muchos ejemplos históricos como los PDP, Motorola 68000, Intel 4004 o Intel 8086, quizá los más representativos. Prácticamente cualquier ordenador de sobremesa o portátil desde los años 80 ha utilizado un procesador x86.

RISC, con la simpleza por bandera

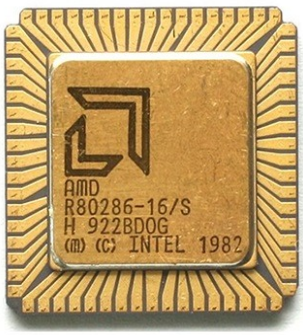
Tras el lanzamiento de CISC, los científicos de IBM empezaron a comprobar que los diseñadores de software creaban sus propias instrucciones más simples y precisas. Entonces, ya en la década de los 70, empezaron a diseñar una alternativa que posteriormente se introdujo en el mercado bajo el acrónimo RISC, Reduced Instruction Set Computing. El IBM 801 que empezó a crearse en 1975, fue diseñado por John Cocke y es considerado el primer procesador RISC de la historia.
IBM 801 John Cocke ya en los 80 junto a uno de los prototipos que usó el procesador 801

La principal virtud de RISC es tener un conjunto de instrucciones muy simples que se ejecutarán más rápidamente en el procesador. Existe un catálogo de pocas instrucciones y éstas son muy sencillas, lo cual implica también que para una cierta tarea compleja necesitaremos un mayor número de ellas, y por esto el programa final tendrá una longitud mayor y además accederá en un mayor número de ocasiones a los datos almacenados en la memoria. En nuestro ejemplo del algoritmo del huevo frito, un procesador RISC estaría compuesto por las instrucciones descritas entre Paso 6.1 y Paso 6.6.

Un procesador de tipo RISC es más simple tanto en software (instrucciones) como en hardware (registros de memoria), lo cual hace que sea un dispositivo notablemente más barato que otras CPU. En la actualidad el mayor ejemplo de procesador RISC son los productos ARM, utilizados ampliamente en dispositivos móviles pero también en otros campos como los supercomputadores. ARM será el principal protagonista de una de nuestras próximas entradas de esta saga.

RISC frente a CISC y la gran batalla actual

CISC nació con la finalidad de homogeneizar los diferentes computadores en los años 50 y 60. RISC buscó en los 70 ir un paso más allá y mejorar el rendimiento con instrucciones más simples pero programas más largos y más difíciles de desarrollar.



Tanto CISC como RISC han evolucionado de forma muy notable desde su nacimiento, adoptando mejoras provenientes del contrario en ambos casos y nuevos conjuntos de instrucciones para adaptarse a los usos de los ordenadores. Si bien es cierto que en la época de su creación la diferencia era muy amplia –principalmente debido a las limitaciones técnicas de la época tanto en tamaño de memoria como en velocidad de proceso -, en la actual informática moderna los requisitos son muy diferentes. Los límites en capacidad de almacenamiento son casi inexistentes y los procesadores son capaces de ejecutar millones de instrucciones en un solo segundo.
IBM PowerPC 601 e Intel 80286 ¿Adivináis qué enfoque siguen estos dos?

La gran batalla actual es la de sus dos grandes exponentes, ARM y x86, que han actualizado sus objetivos a lo que les importa a los usuarios del siglo XXI. Veremos que el punto fuerte de ARM está en la eficiencia energética. Un chip ARM consume mucha menos energía que un procesador x86 que tiene en su alto rendimiento su gran virtud, a costa de consumir bastante más energía.