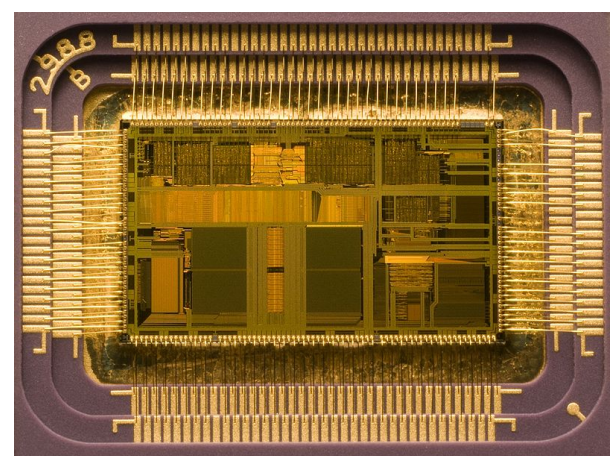


Unidad central de procesamiento

La **unidad central de procesamiento** o **CPU** (por el acrónimo en inglés de *central processing unit*), o simplemente el **procesador** o **microprocesador**, es el componente del computador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos. Los CPU proporcionan la característica fundamental de la computadora digital (la programabilidad) y son uno de los componentes necesarios encontrados en las computadoras de cualquier tiempo, junto con el almacenamiento primario y los dispositivos de entrada/salida. Se conoce como microprocesador el CPU que es manufacturado con circuitos integrados. Desde mediados de los años 1970, los microprocesadores de un solo chip han reemplazado casi totalmente todos los tipos de CPU, y hoy en día, el término "CPU" es aplicado usualmente a todos los microprocesadores.

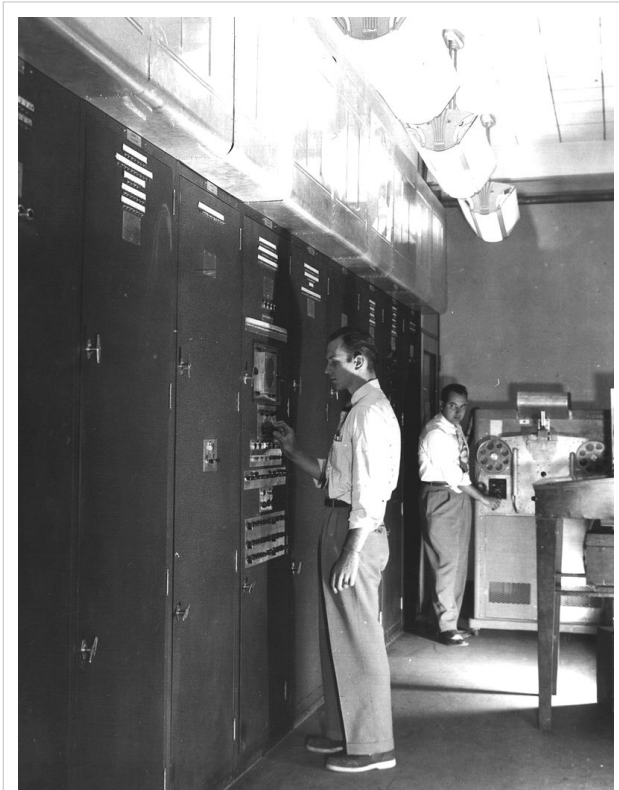


Oblea de un microprocesador Intel 80486DX2 (tamaño: 12x6,75 mm) en su empaquetado.

La expresión "unidad central de proceso" es, en términos generales, una descripción de una cierta clase de máquinas de lógica que pueden ejecutar complejos programas de computadora. Esta amplia definición puede fácilmente ser aplicada a muchos de los primeros computadores que existieron mucho antes que el término "CPU" estuviera en amplio uso. Sin embargo, el término en sí mismo y su acrónimo han estado en uso en la industria de la informática por lo menos desde el principio de los años 1960. La forma, el diseño y la implementación de los CPU ha cambiado drásticamente desde los primeros ejemplos, pero su operación fundamental ha permanecido bastante similar.

Los primeros CPU fueron diseñados a la medida como parte de una computadora más grande, generalmente una computadora única en su especie. Sin embargo, este costoso método de diseñar los CPU a la medida, para una aplicación particular, ha desaparecido en gran parte y se ha sustituido por el desarrollo de clases de procesadores baratos y estandarizados adaptados para uno o muchos propósitos. Esta tendencia de estandarización comenzó generalmente en la era de los transistores discretos, computadoras centrales, y microcomputadoras, y fue acelerada rápidamente con la popularización del circuito integrado (IC), éste ha permitido que sean diseñados y fabricados CPU más complejos en espacios pequeños (en la orden de milímetros). Tanto la miniaturización como la estandarización de los CPU han aumentado la presencia de estos dispositivos digitales en la vida moderna mucho más allá de las aplicaciones limitadas de máquinas de computación dedicadas. Los microprocesadores modernos aparecen en todo, desde automóviles, televisores, neveras, calculadoras, aviones, hasta teléfonos móviles o celulares, juguetes, entre otros.

Historia



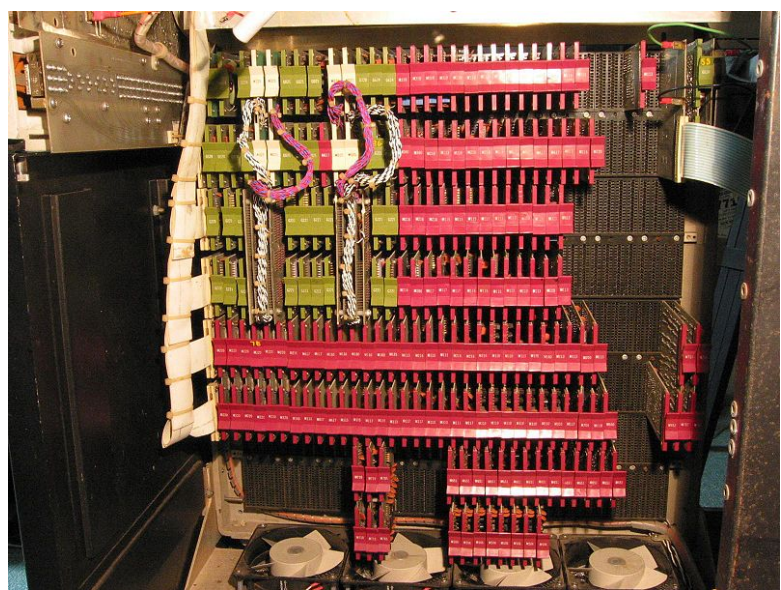
El EDVAC, uno de los primeros computadores de programas almacenados electrónicamente.

Casi todos los CPU tratan con estados discretos, y por lo tanto requieren una cierta clase de elementos de conmutación para diferenciar y cambiar estos estados. Antes de la aceptación comercial del transistor, los relés eléctricos y los tubos de vacío (válvulas termoiónicas) eran usados comúnmente como elementos de conmutación. Aunque éstos tenían distintas ventajas de velocidad sobre los anteriores diseños puramente mecánicos, no eran fiables por varias razones. Por ejemplo, hacer circuitos de lógica secuencial de corriente directa requería hardware adicional para hacer frente al problema del rebote de contacto. Por otro lado, mientras que los tubos de vacío no sufren del rebote de contacto, éstos deben calentarse antes de llegar a estar completamente operacionales y eventualmente fallan y dejan de funcionar por completo.^[1] Generalmente, cuando un tubo ha fallado, el CPU tendría que ser diagnosticado para localizar el componente que falla para que pueda ser reemplazado. Por lo tanto, los primeros computadores electrónicos, (basados en tubos de vacío), generalmente eran más rápidas pero menos confiables que las computadoras electromecánicas, (basadas en relés). Las computadoras

de tubo, como el EDVAC, tendieron a tener un promedio de ocho horas entre fallas, mientras que las computadoras de relés, (anteriores y más lentas), como el Harvard Mark I, fallaban muy raramente. Al final, los CPU basados en tubo llegaron a ser dominantes porque las significativas ventajas de velocidad producidas generalmente pesaban más que los problemas de confiabilidad. La mayor parte de estos tempranos CPU síncronos corrían en frecuencias de reloj bajas comparadas con los modernos diseños microelectrónicos, (ver más abajo para una exposición sobre la frecuencia de reloj). Eran muy comunes en este tiempo las frecuencias de la señal del reloj con un rango desde 100 kHz hasta 4 MHz, limitado en gran parte por la velocidad de los dispositivos de conmutación con los que fueron contruidos.

CPU de transistores y de circuitos integrados discretos

La complejidad del diseño de los CPU se incrementó a medida que varias tecnologías facilitaron la construcción de dispositivos electrónicos más pequeños y confiables. La primera de esas mejoras vino con el advenimiento del transistor. Los CPU transistorizados durante los años 1950 y los años 1960 no tuvieron que ser construidos con elementos de conmutación abultados, no fiables, y frágiles, como los tubos de vacío y los relés eléctricos. Con esta mejora, fueron construidos CPU más complejos y más confiables sobre una o varias tarjetas de circuito impreso que contenían componentes discretos (individuales).



CPU, memoria de núcleo, e interfaz de bus externo de un MSI PDP-8/I.

Durante este período, ganó popularidad un **método** de fabricar muchos transistores en un espacio compacto. El circuito integrado (**IC**) permitió que una gran cantidad de transistores fueran fabricados en una simple oblea basada en semiconductor o "chip". Al principio, solamente circuitos digitales muy básicos, no especializados, como las puertas NOR fueron miniaturizados en IC. Los CPU basadas en estos IC de "bloques de construcción" generalmente son referidos como dispositivos de pequeña escala de integración "small-scale integration" (**SSI**). Los circuitos integrados SSI, como los usados en el computador guía del Apollo (Apollo Guidance Computer), usualmente contenían transistores que se contaban en números de múltiplos de diez. Construir un CPU completo usando IC SSI requería miles de chips individuales, pero todavía consumía mucho menos espacio y energía que diseños anteriores de transistores discretos. A medida que la tecnología microelectrónica avanzó, en los IC fue colocado un número creciente de transistores, disminuyendo así la cantidad de IC individuales necesarios para un CPU completo. Los circuitos integrados **MSI** y el **LSI** (de mediana y gran escala de integración) aumentaron el número de transistores a cientos, y luego a miles.

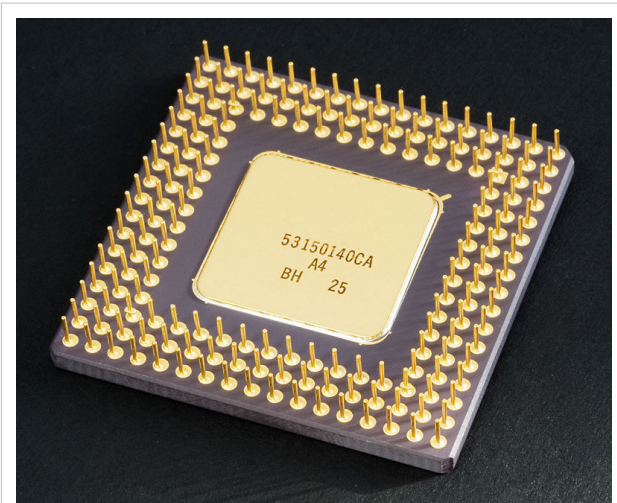
En 1964, IBM introdujo su arquitectura de computador System/360, que fue usada en una serie de computadores que podían ejecutar los mismos programas con velocidades y desempeños diferentes. Esto fue significativo en un tiempo en que la mayoría de las computadoras electrónicas eran incompatibles entre sí, incluso las hechas por el mismo fabricante. Para facilitar esta mejora, IBM utilizó el concepto de microprograma, a menudo llamado "microcódigo", ampliamente usado aún en los CPU modernos. La arquitectura System/360 era tan popular que dominó el mercado del mainframe durante las siguientes décadas y dejó una herencia que todavía aún perdura en las computadoras modernas, como el IBM zSeries. En el mismo año de 1964, Digital Equipment Corporation (DEC) introdujo otro computador que sería muy influyente, dirigido a los mercados científicos y de investigación, el PDP-8. DEC introduciría más adelante la muy popular línea del PDP-11, que originalmente fue construido con IC SSI pero eventualmente fue implementado con componentes LSI cuando se convirtieron en prácticos. En fuerte contraste con sus precursores hechos con tecnología SSI y MSI, la primera implementación LSI del PDP-11 contenía un CPU integrado únicamente por cuatro circuitos integrados LSI.

Los computadores basados en transistores tenían varias ventajas frente a sus predecesores. Aparte de facilitar una creciente fiabilidad y un menor consumo de energía, los transistores también permitían al CPU operar a velocidades

mucho más altas debido al corto tiempo de conmutación de un transistor en comparación a un tubo o relé. Gracias tanto a esta creciente fiabilidad como al dramático incremento de velocidad de los elementos de conmutación que por este tiempo eran casi exclusivamente transistores, se fueron alcanzando frecuencias de reloj del CPU de decenas de megahertz. Además, mientras que los CPU de transistores discretos y circuitos integrados se usaban comúnmente, comenzaron a aparecer los nuevos diseños de alto rendimiento como procesadores vectoriales SIMD (Single Instruction Multiple Data) (Simple Instrucción Múltiples Datos). Estos primeros diseños experimentales dieron lugar más adelante a la era de las supercomputadoras especializadas, como los hechos por Cray Inc.

Microprocesadores

Desde la introducción del primer microprocesador, el Intel 4004, en 1970, y del primer microprocesador ampliamente usado, el Intel 8080, en 1974, esta clase de CPUs ha desplazado casi totalmente el resto de los métodos de implementación de la Unidad Central de Proceso. Los fabricantes de mainframes y minicomputadores de ese tiempo lanzaron programas de desarrollo de IC propietarios para actualizar sus más viejas arquitecturas de computador, y eventualmente produjeron microprocesadores con conjuntos de instrucciones que eran compatibles hacia atrás con sus más viejos hardwares y softwares. Combinado con el advenimiento y el eventual vasto éxito del ahora ubicuo computadora personal, el término "CPU" es aplicado ahora casi exclusivamente a los microprocesadores.



Microprocesador Intel 80486DX2 en un paquete PGA de cerámica

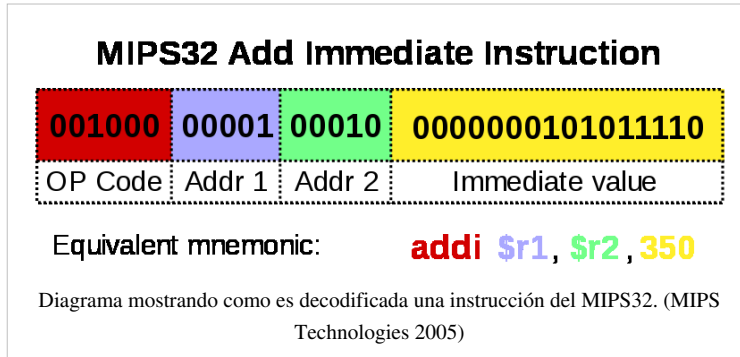
Las generaciones previas de CPUs fueron implementadas como componentes discretos y numerosos circuitos integrados de pequeña escala de integración en una o más tarjetas de circuitos. Por otro lado, los microprocesadores son CPUs fabricados con un número muy pequeño de IC; usualmente solo uno. El tamaño más pequeño del CPU, como resultado de estar implementado en una simple pastilla, significa tiempos de conmutación más rápidos debido a factores físicos como el decrecimiento de la capacitancia parásita de las puertas. Esto ha permitido que los microprocesadores síncronos tengan tiempos de reloj con un rango de decenas de megahercios a varios gigahercios. Adicionalmente, como ha aumentado la capacidad de construir transistores excesivamente pequeños en un IC, la complejidad y el número de transistores en un simple CPU también se ha incrementado dramáticamente. Esta tendencia ampliamente observada es descrita por la ley de Moore, que ha demostrado hasta la fecha, ser una predicción bastante exacta del crecimiento de la complejidad de los CPUs y otros IC.

Mientras que, en los pasados sesenta años han cambiado drásticamente, la complejidad, el tamaño, la construcción, y la forma general del CPU, es notable que el diseño y el funcionamiento básico no ha cambiado demasiado. Casi todos los CPU comunes de hoy se pueden describir con precisión como máquinas de programa almacenado de von Neumann.

A medida que la ya mencionada ley del Moore continúa manteniéndose verdadera, se han presentado preocupaciones sobre los límites de la tecnología de transistor del circuito integrado. La miniaturización extrema de puertas electrónicas está causando los efectos de fenómenos que se vuelven mucho más significativos, como la electromigración, y el subumbral de pérdida. Estas más nuevas preocupaciones están entre los muchos factores que hacen a investigadores estudiar nuevos métodos de computación como la computadora cuántica, así como ampliar el uso de paralelismo, y otros métodos que extienden la utilidad del modelo clásico de von Neumann.

Operación del CPU

La operación fundamental de la mayoría de los CPU, es ejecutar una secuencia de instrucciones almacenadas llamadas "programa". El programa es representado por una serie de números que se mantienen en una cierta clase de memoria de computador. Hay cuatro pasos que casi todos los CPU de arquitectura de von Neumann usan en su operación: **fetch**, **decode**, **execute**, y **writeback**, (leer, decodificar, ejecutar, y escribir).



El primer paso, **leer** (fetch), implica el recuperar una instrucción, (que es representada por un número o una secuencia de números), de la memoria de programa. La localización en la memoria del programa es determinada por un contador de programa (PC), que almacena un número que identifica la posición actual en el programa. En otras palabras, el contador de programa indica al CPU, el lugar de la instrucción en

el programa actual. Después de que se lee una instrucción, el Contador de Programa es incrementado por la longitud de la palabra de instrucción en términos de unidades de memoria.^[2] Frecuentemente la instrucción a ser leída debe ser recuperada de memoria relativamente lenta, haciendo detener al CPU mientras espera que la instrucción sea retornada. Este problema es tratado en procesadores modernos en gran parte por los cachés y las arquitecturas pipeline (ver abajo).

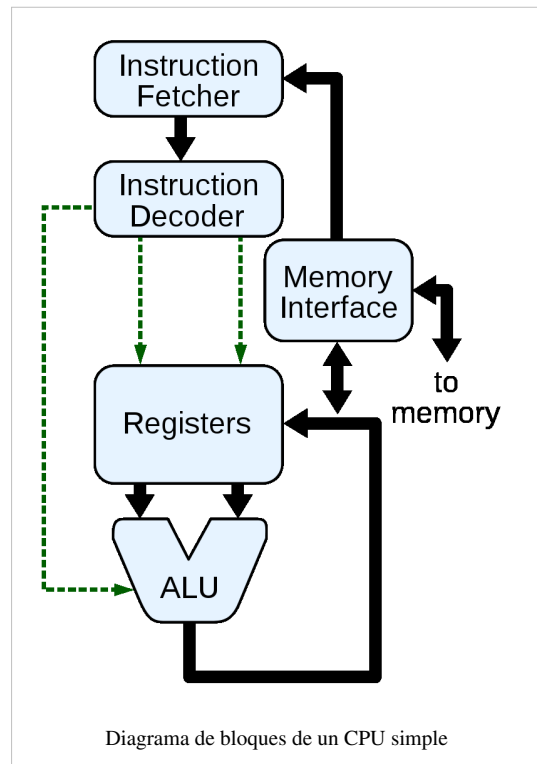
La instrucción que el CPU lee desde la memoria es usada para determinar qué deberá hacer el CPU. En el paso de **decodificación**, la instrucción es dividida en partes que tienen significado para otras unidades del CPU. La manera en que el valor de la instrucción numérica es interpretado está definida por la arquitectura del conjunto de instrucciones (el **ISA**) del CPU.^[3] A menudo, un grupo de números en la instrucción, llamados opcode, indica qué operación realizar. Las partes restantes del número usualmente proporcionan información requerida para esa instrucción, como por ejemplo, operandos para una operación de adición. Tales operandos se pueden dar como un valor constante (llamado valor inmediato), o como un lugar para localizar un valor, que según lo determinado por algún modo de dirección, puede ser un registro o una dirección de memoria. En diseños más viejos las unidades del CPU responsables de decodificar la instrucción eran dispositivos de hardware fijos. Sin embargo, en CPUs e ISAs más abstractos y complicados, es frecuentemente usado un microprograma para ayudar a traducir instrucciones en varias señales de configuración para el CPU. Este microprograma es a veces reescribible de tal manera que puede ser modificado para cambiar la manera en que el CPU decodifica instrucciones incluso después de que haya sido fabricado.

Después de los pasos de lectura y decodificación, es llevado a cabo el paso de la **ejecución** de la instrucción. Durante este paso, varias unidades del CPU son conectadas de tal manera que ellas pueden realizar la operación deseada. Si, por ejemplo, una operación de adición fue solicitada, una unidad aritmético lógica (**ALU**) será conectada a un conjunto de entradas y un conjunto de salidas. Las entradas proporcionan los números a ser sumados, y las salidas contendrán la suma final. La ALU contiene la circuitería para realizar operaciones simples de aritmética y lógica en las entradas, como adición y operaciones de bits (bitwise). Si la operación de adición produce un resultado demasiado grande para poder ser manejado por el CPU, también puede ser ajustada una bandera (flag) de desbordamiento aritmético localizada en un registro de banderas (ver abajo la sección sobre rango de números enteros).

El paso final, la **escritura** (writeback), simplemente "escribe" los resultados del paso de ejecución a una cierta forma de memoria. Muy a menudo, los resultados son escritos a algún registro interno del CPU para acceso rápido por subsecuentes instrucciones. En otros casos los resultados pueden ser escritos a

una memoria principal más lenta pero más barata y más grande. Algunos tipos de instrucciones manipulan el contador de programa en lugar de directamente producir datos de resultado. Éstas son llamadas generalmente "saltos" (jumps) y facilitan comportamientos como l bucles (loops), la ejecución condicional de programas (con el uso de saltos condicionales), y funciones en programas.^[4] Muchas instrucciones también cambiarán el estado de dígitos en un registro de "banderas". Estas banderas pueden ser usadas para influenciar cómo se comporta un programa, puesto que a menudo indican el resultado de varias operaciones. Por ejemplo, un tipo de instrucción de "comparación" considera dos valores y fija un número, en el registro de banderas, de acuerdo a cuál es el mayor. Entonces, esta bandera puede ser usada por una posterior instrucción de salto para determinar el flujo de programa.

Después de la ejecución de la instrucción y la escritura de los datos resultantes, el proceso entero se repite con el siguiente ciclo de instrucción, normalmente leyendo la siguiente instrucción en secuencia debido al valor incrementado en el contador de programa. Si la instrucción completada era un salto, el contador de programa será modificado para contener la dirección de la instrucción a la cual se saltó, y la ejecución del programa continúa normalmente. En CPUs más complejos que el descrito aquí, múltiples instrucciones pueden ser leídas, decodificadas, y ejecutadas simultáneamente. Esta sección describe lo que es referido generalmente como el "entubado RISC clásico" (Classic RISC pipeline), que de hecho es bastante común entre los CPU simples usados en muchos dispositivos electrónicos, a menudo llamados microcontroladores.^[5]



Diseño e implementación

Prerrequisitos
Arquitectura informática
Circuitos digitales

Rango de enteros

La manera en que un CPU representa los números es una opción de diseño que afecta las más básicas formas en que el dispositivo funciona. Algunas de las primeras calculadoras digitales usaron, para representar números internamente, un modelo eléctrico del sistema de numeración decimal común (base diez). Algunas otras computadoras han usado sistemas de numeración más exóticos como el ternario (base tres). Casi todos los CPU modernos representan los números en forma binaria, en donde cada dígito es representado por una cierta cantidad física de dos valores, como un voltaje "alto" o "bajo".^[6]



Con la representación numérica están relacionados el tamaño y la precisión de los números que un CPU puede representar. En el caso de un CPU binario, un **bit** se refiere a una posición significativa en los números con que trabaja un CPU. El número de bits (o de posiciones numéricas, o dígitos) que un CPU usa para representar los números, a menudo se llama "tamaño de la palabra", "ancho de bits", "ancho de ruta de datos", o "precisión del número entero" cuando se ocupa estrictamente de números enteros (en oposición a

números de coma flotante). Este número difiere entre las arquitecturas, y a menudo dentro de diferentes unidades del mismo CPU. Por ejemplo, un CPU de 8 bits maneja un rango de números que pueden ser representados por ocho dígitos binarios, cada dígito teniendo dos valores posibles, y en combinación los 8 bits teniendo 2^8 ó 256 números discretos. En efecto, el tamaño del número entero fija un límite de hardware en el rango de números enteros que el software corre y que el CPU puede usar directamente.^[7]

El rango del número entero también puede afectar el número de posiciones en memoria que el CPU puede **direccionar** (localizar). Por ejemplo, si un CPU binario utiliza 32 bits para representar una dirección de memoria, y cada dirección de memoria representa a un octeto (8 bits), la cantidad máxima de memoria que el CPU puede direccionar es 2^{32} octetos, o 4 GB. Ésta es una vista muy simple del espacio de dirección del CPU, y muchos diseños modernos usan métodos de dirección mucho más complejos como paginación para localizar más memoria que su rango entero permitiría con un espacio de dirección plano.

Niveles más altos del rango de números enteros requieren más estructuras para manejar los dígitos adicionales, y por lo tanto, más complejidad, tamaño, uso de energía, y generalmente costo. Por ello, no es del todo infrecuente, ver microcontroladores de 4 y 8 bits usados en aplicaciones modernas, aun cuando están disponibles CPU con un rango mucho más alto (de 16, 32, 64, e incluso 128 bits). Los microcontroladores más simples son generalmente más baratos, usan menos energía, y por lo tanto disipan menos calor. Todo esto pueden ser consideraciones de diseño importantes para los dispositivos electrónicos. Sin embargo, en aplicaciones del extremo alto, los beneficios producidos por el rango adicional, (más a menudo el espacio de dirección adicional), son más significativos y con frecuencia afectan las opciones del diseño. Para ganar algunas de las ventajas proporcionadas por las longitudes de bits tanto más bajas, como más altas, muchas CPUs están diseñadas con anchos de bit diferentes para diferentes unidades del dispositivo. Por ejemplo, el IBM System/370 usó un CPU que fue sobre todo de 32 bits, pero usó precisión de 128 bits dentro de sus unidades de coma flotante para facilitar mayor exactitud y rango de números de

coma flotante. Muchos diseños posteriores de CPU usan una mezcla de ancho de bits similar, especialmente cuando el procesador está diseñado para usos de propósito general donde se requiere un razonable equilibrio entre la capacidad de números enteros y de coma flotante.

Frecuencia de reloj

La mayoría de los CPU, y de hecho, la mayoría de los dispositivos de lógica secuencial, son de naturaleza síncrona.^[8] Es decir, están diseñados y operan en función de una señal de sincronización. Esta señal, conocida como **señal de reloj**, usualmente toma la forma de una onda cuadrada periódica. Calculando el tiempo máximo en que las señales eléctricas pueden moverse en las varias bifurcaciones de los muchos circuitos de un CPU, los diseñadores pueden seleccionar un período apropiado para la señal del reloj.

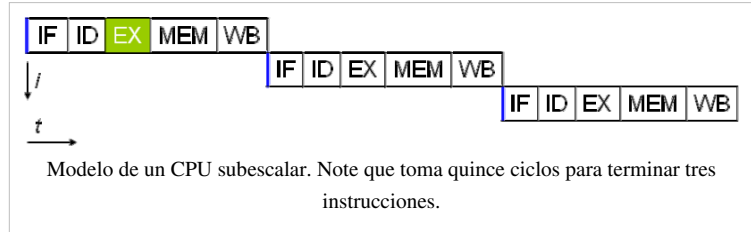
Este período debe ser más largo que la cantidad de tiempo que toma a una señal moverse, o propagarse, en el peor de los casos. Al fijar el período del reloj a un valor bastante mayor sobre el retardo de la propagación del peor caso, es posible diseñar todo el CPU y la manera que mueve los datos alrededor de los "bordes" de la subida y bajada de la señal del reloj. Esto tiene la ventaja de simplificar el CPU significativamente, tanto en una perspectiva de diseño, como en una perspectiva de cantidad de componentes. Sin embargo, esto también tiene la desventaja que todo el CPU debe esperar por sus elementos más lentos, aun cuando algunas unidades de la misma son mucho más rápidas. Esta limitación ha sido compensada en gran parte por varios métodos de aumentar el paralelismo del CPU (ver abajo).

Sin embargo, las solamente mejoras arquitectónicas no solucionan todas las desventajas de CPUs globalmente síncronas. Por ejemplo, una señal de reloj está sujeta a los retardos de cualquier otra señal eléctrica. Velocidades de reloj más altas en CPUs cada vez más complejas hacen más difícil de mantener la señal del reloj en fase (sincronizada) a través de toda la unidad. Esto ha conducido que muchos CPU modernos requieran que se les proporcione múltiples señales de reloj idénticas, para evitar retardar una sola señal lo suficiente significativamente como para hacer al CPU funcionar incorrectamente. Otro importante problema cuando la velocidad del reloj aumenta dramáticamente, es la cantidad de calor que es disipado por el CPU. La señal del reloj cambia constantemente, provocando la conmutación de muchos componentes (cambio de estado) sin importar si están siendo usados en ese momento. En general, un componente que está cambiando de estado, usa más energía que un elemento en un estado estático. Por lo tanto, a medida que la velocidad del reloj aumenta, así lo hace también la disipación de calor, causando que el CPU requiera soluciones de enfriamiento más efectivas.

Un método de tratar la conmutación de componentes innecesarios se llama el clock gating, que implica apagar la señal del reloj a los componentes innecesarios, efectivamente desactivándolos. Sin embargo, esto es frecuentemente considerado como difícil de implementar y por lo tanto no ve uso común afuera de diseños de muy baja potencia.^[9] Otro método de tratar algunos de los problemas de una señal global de reloj es la completa remoción de la misma. Mientras que quitar la señal global del reloj hace, de muchas maneras, considerablemente más complejo el proceso del diseño, en comparación con diseños síncronos similares, los diseños asíncronos (o sin reloj) tienen marcadas ventajas en el consumo de energía y la disipación de calor. Aunque algo infrecuente, CPUs completas se han construido sin utilizar una señal global de reloj. Dos notables ejemplos de esto son el AMULET, que implementa la arquitectura del ARM, y el MiniMIPS, compatible con el MIPS R3000. En lugar de remover totalmente la señal del reloj, algunos diseños de CPU permiten a ciertas unidades del dispositivo ser asíncronas, como por ejemplo, usar ALUs asíncronas en conjunción con pipelining superescalar para alcanzar algunas ganancias en el desempeño aritmético. Mientras que no está completamente claro si los diseños totalmente asíncronos pueden desempeñarse a un nivel comparable o mejor que sus contrapartes síncronas, es evidente que por lo menos sobresalen en las más simples operaciones matemáticas. Esto, combinado con sus excelentes características de consumo de energía y disipación de calor, los hace muy adecuados para sistemas embebidos.

Paralelismo

La descripción de la operación básica de un CPU ofrecida en la sección anterior describe la forma más simple que puede tomar un CPU. Este tipo de CPU, usualmente referido como **subescalar**, opera sobre y ejecuta una sola instrucción con una o dos piezas de datos a la vez.



Este proceso da lugar a una ineficacia inherente en CPUs subescalares. Puesto que solamente una instrucción es ejecutada a la vez, todo el CPU debe esperar que esa instrucción se complete antes de proceder a la siguiente instrucción. Como resultado, el CPU subescalar queda "paralizado" en instrucciones que toman más de un ciclo de reloj para completar su ejecución. Incluso la adición de una segunda unidad de ejecución (ver abajo) no mejora mucho el desempeño. En lugar de un camino quedando congelado, ahora dos caminos se paralizan y aumenta el número de transistores no usados. Este diseño, en donde los recursos de ejecución del CPU pueden operar con solamente una instrucción a la vez, solo puede, posiblemente, alcanzar el desempeño **escalar** (una instrucción por ciclo de reloj). Sin embargo, el desempeño casi siempre es subescalar (menos de una instrucción por ciclo).

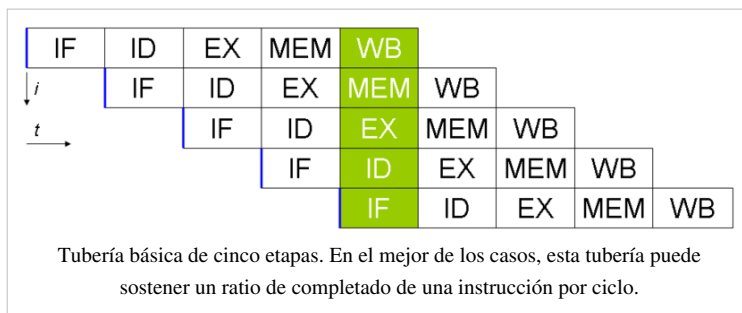
Las tentativas de alcanzar un desempeño escalar y mejor, han resultado en una variedad de metodologías de diseño que hacen comportarse al CPU menos linealmente y más en paralelo. Cuando se refiere al paralelismo en los CPU, generalmente son usados dos términos para clasificar estas técnicas de diseño.

- El paralelismo a nivel de instrucción, en inglés Instruction Level Parallelism (ILP), busca aumentar la tasa en la cual las instrucciones son ejecutadas dentro de un CPU, es decir, aumentar la utilización de los recursos de ejecución en la pastilla
- El paralelismo a nivel de hilo de ejecución, en inglés thread level parallelism (TLP), que se propone incrementar el número de hilos (efectivamente programas individuales) que un CPU pueda ejecutar simultáneamente.

Cada metodología se diferencia tanto en las maneras en las que están implementadas, como en la efectividad relativa que producen en el aumento del desempeño del CPU para una aplicación.^[10]

ILP: Entubado de instrucción y arquitectura superescalar

Artículo principal: Entubado de instrucción y superescalar



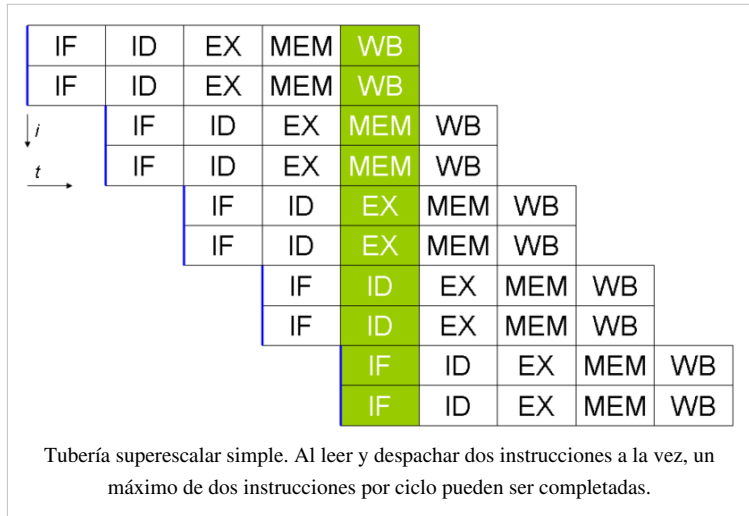
Uno de los más simples métodos usados para lograr incrementar el paralelismo es comenzar los primeros pasos de leer y decodificar la instrucción antes de que la instrucción anterior haya terminado de ejecutarse. Ésta es la forma más simple de una técnica conocida como instruction pipelining (entubado de instrucción), y es utilizada en casi todos los CPU de propósito

general modernos. Al dividir la ruta de ejecución en etapas discretas, la tubería permite que más de una instrucción sea ejecutada en cualquier tiempo. Esta separación puede ser comparada a una línea de ensamblaje, en la cual una instrucción es hecha más completa en cada etapa hasta que sale de la tubería de ejecución y es retirada.

Sin embargo, la tubería introduce la posibilidad de una situación donde es necesario terminar el resultado de la operación anterior para completar la operación siguiente; una condición llamada a menudo como conflicto de dependencia de datos. Para hacer frente a esto, debe ser tomado un cuidado adicional para comprobar estas clases de condiciones, y si esto ocurre, se debe retrasar una porción de la tubería de instrucción. Naturalmente, lograr esto

requiere circuitería adicional, los procesadores entubados son más complejos que los subescalares, pero no mucho. Un procesador entubado puede llegar a ser casi completamente escalar, solamente inhibido por las abruptas paradas de la tubería (una instrucción durando más de un ciclo de reloj en una etapa).

Una mejora adicional sobre la idea del entubado de instrucción (instruction pipelining) condujo al desarrollo de un método que disminuye incluso más el tiempo ocioso de los componentes del CPU. Diseños que se dice que son superescalares incluyen una larga tubería de instrucción y múltiples unidades de ejecución idénticas. En una tubería superescalar, múltiples instrucciones son leídas y pasadas a un despachador, que decide si las instrucciones se pueden o no ejecutar en paralelo (simultáneamente). De ser así, son despachadas a las unidades de ejecución



disponibles, dando por resultado la capacidad para que varias instrucciones sean ejecutadas simultáneamente. En general, cuanto más instrucciones un CPU superescalar es capaz de despachar simultáneamente a las unidades de ejecución en espera, más instrucciones serán completadas en un ciclo dado.

La mayor parte de la dificultad en el diseño de una arquitectura superescalar de CPU descansa en crear un despachador eficaz. El despachador necesita poder determinar rápida y correctamente si las instrucciones pueden ejecutarse en paralelo, tan bien como despacharlas de una manera que mantenga ocupadas tantas unidades de ejecución como sea posible. Esto requiere que la tubería de instrucción sea llenada tan a menudo como sea posible y se incrementa la necesidad, en las arquitecturas superescalares, de cantidades significativas de caché de CPU. Esto también crea técnicas para evitar peligros como la predicción de bifurcación, ejecución especulativa, y la ejecución fuera de orden, cruciales para mantener altos niveles de desempeño.

- La predicción de bifurcación procura predecir qué rama (o trayectoria) tomará una instrucción condicional, el CPU puede minimizar el número de tiempos que toda la tubería debe esperar hasta que sea completada una instrucción condicional.
- La ejecución especulativa frecuentemente proporciona aumentos modestos del desempeño al ejecutar las porciones de código que pueden o no ser necesarias después de que una operación condicional termine.
- La ejecución fuera de orden cambia en algún grado el orden en el cual son ejecutadas las instrucciones para reducir retardos debido a las dependencias de los datos.

En el caso donde una porción del CPU es superescalar y una parte no lo es, la parte que no es superescalar sufre en el desempeño debido a las paradas de horario. El Intel Pentium original (P5) tenía dos ALUs superescalares que podían aceptar, cada una, una instrucción por ciclo de reloj, pero su FPU no podía aceptar una instrucción por ciclo de reloj. Así el P5 era superescalar en la parte de números enteros pero no era superescalar de números de coma (o punto [decimal]) flotante. El sucesor a la arquitectura del Pentium de Intel, el P6, agregó capacidades superescalares a sus funciones de coma flotante, y por lo tanto produjo un significativo aumento en el desempeño de este tipo de instrucciones.

El entubado simple y el diseño superescalar aumentan el ILP de un CPU al permitir a un solo procesador completar la ejecución de instrucciones en ratios que sobrepasan una instrucción por ciclo (IPC).^[11] La mayoría de los modernos diseños de CPU son por lo menos algo superescalares, y en la última década, casi todos los diseños de CPU de propósito general son superescalares. En los últimos años algo del énfasis en el diseño de computadores de alto ILP se ha movido del hardware del CPU hacia su interface de software, o ISA. La estrategia de la muy larga

palabra de instrucción, very long instruction word (VLIW), causa a algún ILP a ser implícito directamente por el software, reduciendo la cantidad de trabajo que el CPU debe realizar para darle un empuje significativo al ILP y por lo tanto reducir la complejidad del diseño.

TLP: ejecución simultánea de hilos

Otra estrategia comúnmente usada para aumentar el paralelismo de los CPU es incluir la habilidad de correr múltiples hilos (programas) al mismo tiempo. En general, CPUs con alto TLP han estado en uso por mucho más tiempo que los de alto ILP. Muchos de los diseños en los que Seymour Cray fue pionero durante el final de los años 1970 y los años 1980 se concentraron en el TLP como su método primario de facilitar enormes capacidades de computación (para su tiempo). De hecho, el TLP, en la forma de mejoras en múltiples hilos de ejecución, estuvo en uso tan temprano como desde los años 1950. En el contexto de diseño de procesadores individuales, las dos metodologías principales usadas para lograr el TLP son, multiprocesamiento a nivel de chip, en inglés chip-level multiprocessing (CMP), y el multihilado simultáneo, en inglés simultaneous multithreading (SMT). En un alto nivel, es muy común construir computadores con múltiples CPU totalmente independientes en arreglos como multiprocesamiento simétrico (symmetric multiprocessing (SMP)) y acceso de memoria no uniforme (Non-Uniform Memory Access (NUMA)).^[12] Aunque son usados medios muy diferentes, todas estas técnicas logran la misma meta: incrementar el número de hilos que el CPU(s) puede correr en paralelo.

Los métodos de paralelismo CMP y de SMP son similares uno del otro y lo más directo. Éstos implican algo más conceptual que la utilización de dos o más CPU completos y CPU independientes. En el caso del CMP, múltiples "núcleos" de procesador son incluidos en el mismo paquete, a veces en el mismo circuito integrado.^[13] Por otra parte, el SMP incluye múltiples paquetes independientes. NUMA es algo similar al SMP pero usa un modelo de acceso a memoria no uniforme. Esto es importante para los computadores con muchos CPU porque el tiempo de acceso a la memoria, de cada procesador, es agotado rápidamente con el modelo de memoria compartido del SMP, resultando en un significativo retraso debido a los CPU esperando por la memoria. Por lo tanto, NUMA es considerado un modelo mucho más escalable, permitiendo con éxito que en un computador sean usados muchos más CPU que los que pueda soportar de una manera factible el SMP. El SMT se diferencia en algo de otras mejoras de TLP en que el primero procura duplicar tan pocas porciones del CPU como sea posible. Mientras es considerada una estrategia TLP, su implementación realmente se asemeja más a un diseño superescalar, y de hecho es frecuentemente usado en microprocesadores superescalares, como el POWER5 de IBM. En lugar de duplicar todo el CPU, los diseños SMT solamente duplican las piezas necesarias para lectura, decodificación, y despacho de instrucciones, así como cosas como los registros de propósito general. Esto permite a un CPU SMT mantener sus unidades de ejecución ocupadas más frecuentemente al proporcionarles las instrucciones desde dos diferentes hilos de software. Una vez más esto es muy similar al método superescalar del ILP, pero ejecuta simultáneamente instrucciones de múltiples hilos en lugar de ejecutar concurrentemente múltiples instrucciones del mismo hilo.

Procesadores vectoriales y el SIMD

Artículos principales: Procesador vectorial y SIMD

Un menos común pero cada vez más importante paradigma de CPU (y de hecho, de computación en general) trata con **vectores**. Los procesadores de los que se ha hablado anteriormente son todos referidos como cierto tipo de dispositivo **escalar**.^[14] Como implica su nombre, los procesadores vectoriales se ocupan de múltiples piezas de datos en el contexto de una instrucción, esto contrasta con los procesadores escalares, que tratan una pieza de dato por cada instrucción. Estos dos esquemas de ocuparse de los datos son generalmente referidos respectivamente como SISD (Single Instruction, Single Data) (Simple Instrucción, Simple Dato) y SIMD (Single Instruction, Multiple Data) (Simple Instrucción, Múltiples Datos). La gran utilidad en crear CPUs que se ocupen de vectores de datos radica en la optimización de tareas que tienden a requerir la misma operación, por ejemplo, una suma, o un producto escalar, a ser realizado en un gran conjunto de datos. Algunos ejemplos clásicos de este tipo de tareas son las aplicaciones multimedia (imágenes, vídeo, y sonido), así como muchos tipos de tareas científicas y de ingeniería. Mientras que un

CPU escalar debe completar todo el proceso de leer, decodificar, y ejecutar cada instrucción y valor en un conjunto de datos, un CPU vectorial puede realizar una simple operación en un comparativamente grande conjunto de datos con una sola instrucción. Por supuesto, esto es solamente posible cuando la aplicación tiende a requerir muchos pasos que apliquen una operación a un conjunto grande de datos.

La mayoría de los primeros CPU vectoriales, como el Cray-1, fueron asociados casi exclusivamente con aplicaciones de investigación científica y criptografía. Sin embargo, a medida que la multimedia se desplazó en gran parte a medios digitales, ha llegado a ser significativa la necesidad de una cierta forma de SIMD en CPUs de propósito general. Poco después de que comenzara a ser común incluir unidades de coma flotante en procesadores de uso general, también comenzaron a aparecer especificaciones e implementaciones de unidades de ejecución SIMD para los CPU de uso general. Algunas de estas primeras especificaciones SIMD, como el MMX de Intel, fueron solamente para números enteros. Esto demostró ser un impedimento significativo para algunos desarrolladores de software, ya que muchas de las aplicaciones que se beneficiaban del SIMD trataban sobre todo con números de coma flotante. Progresivamente, éstos primeros diseños fueron refinados y rehechos en alguna de las comunes, modernas especificaciones SIMD, que generalmente están asociadas a un ISA. Algunos ejemplos modernos notables son el SSE de Intel y el AltiVec relacionado con el PowerPC (también conocido como VMX).^[15]

Véase también

- Arquitectura de CPU
 - Unidad de control
 - Unidad aritmético lógica
 - Unidad de punto flotante
 - Coprocesador
 - Bus interface unit
 - Unidad de gestión de memoria
 - Unidad de ejecución
 - Unidad de proceso
 - Registro (hardware)
 - Microcódigo
 - Barrel shifter
 - Microprocesador
 - CISC
 - RISC
 - Bus de computadora
 - Bus de datos
 - Bus de direcciones
 - Bus de control
 - Conjunto de instrucciones
 - Diseño de CPU
 - Estado de espera
 - Ingeniería de computación
 - Lista de procesadores AMD Athlon 64
 - Tipos de datos máquina
 - Socket de CPU
 - Voltaje del núcleo del CPU
 - Enfriamiento del CPU
-

Referencias

- [1] Vacuum tubes eventually stop functioning in the course of normal operation due to the slow contamination of their cathodes that occurs when the tubes are in use. Additionally, sometimes the tube's vacuum seal can form a leak, which accelerates the cathode contamination. See vacuum tube.
 - [2] Since the program counter counts *memory addresses* and not *instructions*, it is incremented by the number of memory units that the instruction word contains. In the case of simple fixed-length instruction word ISAs, this is always the same number. For example, a fixed-length 32-bit instruction word ISA that uses 8-bit memory words would always increment the PC by 4 (except in the case of jumps). ISAs that use variable length instruction words, such as x86, increment the PC by the number of memory words corresponding to the last instruction's length. Also, note that in more complex CPU, incrementing the PC does not necessarily occur at the end of instruction execution. This is especially the case in heavily pipelined and superscalar architectures (see the relevant sections below).
 - [3] Because the instruction set architecture of a CPU is fundamental to its interface and usage, it is often used as a classification of the "type" of CPU. For example, a "PowerPC CPU" uses some variant of the PowerPC ISA. Some CPU, like the Intel Itanium, can actually interpret instructions for more than one ISA; however this is often accomplished by software means rather than by designing the hardware to directly support both interfaces. (See emulator)
 - [4] Some early computers like the Harvard Mark I did not support any kind of "jump" instruction, effectively limiting the complexity of the programs they could run. It is largely for this reason that these computers are often not considered to contain a CPU proper, despite their close similarity as stored program computers.
 - [5] This description is, in fact, a simplified view even of the Classic RISC pipeline. It largely ignores the important role of CPU cache, and therefore the **access** stage of the pipeline. See the respective articles for more details.
 - [6] The physical concept of voltage is an analog one by its nature, practically having an infinite range of possible values. For the purpose of physical representation of binary numbers, set ranges of voltages are defined as one or zero. These ranges are usually influenced by the operational parameters of the switching elements used to create the CPU, such as a transistor's threshold level.
 - [7] While a CPU's integer size sets a limit on integer ranges, this can (and often is) overcome using a combination of software and hardware techniques. By using additional memory, software can represent integers many magnitudes larger than the CPU can. Sometimes the CPU's ISA will even facilitate operations on integers larger that it can natively represent by providing instructions to make large integer arithmetic relatively quick. While this method of dealing with large integers is somewhat slower than utilizing a CPU with higher integer size, it is a reasonable trade-off in cases where natively supporting the full integer range needed would be cost-prohibitive. See Arbitrary-precision arithmetic for more details on purely software-supported arbitrary-sized integers.
 - [8] In fact, all synchronous CPU use a combination of sequential logic and combinatorial logic. (See boolean logic)
 - [9] One notable late CPU design that uses clock gating is that of the IBM PowerPC-based Xbox 360. It utilizes extensive clock gating in order to reduce the power requirements of the aforementioned videogame console it is used in.
 - [10] It should be noted that neither ILP nor TLP is inherently superior over the other; they are simply different means by which to increase CPU parallelism. As such, they both have advantages and disadvantages, which are often determined by the type of software that the processor is intended to run. High-TLP CPU are often used in applications that lend themselves well to being split up into numerous smaller applications, so-called "embarrassingly parallel problems." Frequently, a computational problem that can be solved quickly with high TLP design strategies like SMP take significantly more time on high ILP devices like superscalar CPU, and vice versa.
 - [11] Best-case scenario (or peak) IPC rates in very superscalar architectures are difficult to maintain since it is impossible to keep the instruction pipeline filled all the time. Therefore, in highly superscalar CPU, average sustained IPC is often discussed rather than peak IPC.
 - [12] Even though SMP and NUMA are both referred to as "systems level" TLP strategies, both methods must still be supported by the CPU's design and implementation.
 - [13] While TLP methods have generally been in use longer than ILP methods, Chip-level multiprocessing is more or less only seen in later IC-based microprocessors. This is largely because the term itself is inapplicable to earlier discrete component devices and has only come into use recently.
For several years during the late 1990s and early 2000s, the focus in designing high performance general purpose CPU was largely on highly superscalar IPC designs, such as the Intel Pentium 4. However, this trend seems to be reversing somewhat now as major general-purpose CPU designers switch back to less deeply pipelined high-TLP designs. This is evidenced by the proliferation of dual and multi core CMP designs and notably, Intel's newer designs resembling its less superscalar P6 architecture. Late designs in several processor families exhibit CMP, including the x86-64 Opteron and Athlon 64 X2, the SPARC UltraSPARC T1, IBM POWER4 and POWER5, as well as several video game console CPU like the Xbox 360's triple-core PowerPC design.
 - [14] Earlier the term **scalar** was used to compare most the IPC (instructions per cycle) count afforded by various ILP methods. Here the term is used in the strictly mathematical sense to contrast with vectors. See scalar (mathematics) and vector (spatial).
 - [15] Although SSE/SSE2/SSE3 have superseded MMX in Intel's general purpose CPU, later IA-32 designs still support MMX. This is usually accomplished by providing most of the MMX functionality with the same hardware that supports the much more expansive SSE instruction sets.
- a b Amdahl, G. M., Blaauw, G. A., & Brooks, F. P. Jr.(1964)." *Architecture of the IBM System/360* (<http://www.research.ibm.com/journal/rd/441/amdahl.pdf>). IBM Research.

- a Brown, Jeffery (2005). « Application-customized CPU design (<http://www-128.ibm.com/developerworks/power/library/pa-fpfxbox/?ca=dgr-lnxw07XBoxDesign>)». IBM developerWorks. Consultado el 17-12-2005.
- a Digital Equipment Corporation (November de 1975). « LSI-11 Module Descriptions (<http://www.classiccmp.org/bitsavers/pdf/dec/pdp11/1103/EK-LSI11-TM-002.pdf>)». *LSI-11, PDP-11/03 user's manual* (2nd edition edición). Maynard, Massachusetts: Digital Equipment Corporation. pp. 4-3.
- a Garside, J. D., Furber, S. B., & Chung, S-H(1999). " *AMULET3 Revealed* (http://www.cs.manchester.ac.uk/apt/publications/papers/async99_A3.php)". University of Manchester Computer Science Department.
- Hennessy, John A.; Goldberg, David (1996). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers. ISBN 1-55860-329-8.
- a MIPS Technologies, Inc.(2005). " *MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set* (<http://www.mips.com/content/Documentation/MIPSDocumentation/ProcessorArchitecture/doclibrary>)". MIPS Technologies, Inc..
- a Smotherman, Mark (2005). « History of Multithreading (<http://www.cs.clemson.edu/~mark/multithreading.html>)». Consultado el 19-12-2005.
- a von Neumann, John(1945). " *First Draft of a Report on the EDVAC* (<http://www.virtualtravelog.net/entries/2003-08-TheFirstDraft.pdf>)". Moore School of Electrical Engineering, University of Pennsylvania.
- a b Weik, Martin H.(1961). " *A Third Survey of Domestic Electronic Digital Computing Systems* (<http://ed-thelen.org/comp-hist/BRL61.html>)". Ballistic Research Laboratories.

Enlaces externos

Diseños de microprocesador

- Advanced Micro Devices (<http://www.amd.com/>) - Advanced Micro Devices, a designer of primarily x86-compatible personal computer CPU.
- ARM Ltd (<http://www.arm.com/>) - ARM Ltd, one of the few CPU designers that profits solely by licensing their designs rather than manufacturing them. ARM architecture microprocessors are among the most popular in the world for embedded applications.
- Freescale Semiconductor (<http://www.freescale.com/>) (formerly of Motorola) - Freescale Semiconductor, designer of several embedded and SoC PowerPC based processors.
- IBM Microelectronics (<http://www-03.ibm.com/chips/>) - Microelectronics division of IBM, which is responsible for many POWER and PowerPC based designs, including many of the CPU utilized in late video game consoles.
- Intel Corp (<http://www.intel.com/>) - Intel, a maker of several notable CPU lines, including IA-32, IA-64, and XScale. Also a producer of various peripheral chips for use with their CPU.
- MIPS Technologies (<http://www.mips.com/>) - MIPS Technologies, developers of the MIPS architecture, a pioneer in RISC designs.
- Sun Microsystems (<http://www.sun.com/>) - Sun Microsystems, developers of the SPARC architecture, a RISC design.
- Texas Instruments (http://www.ti.com/home_p_allsc) - Texas Instruments semiconductor division. Designs and manufactures several types of low power microcontrollers among their many other semiconductor products.
- Transmeta (<http://www.transmeta.com/>) - Transmeta Corporation. Creators of low-power x86 compatibles like Crusoe and Efficeon.

Lectura adicional

- Processor Design: An Introduction (http://www.gamezero.com/team-0/articles/math_magic/micro/index.html) - Detailed introduction to microprocessor design. Somewhat incomplete and outdated, but still worthwhile.
- How Microprocessors Work (<http://computer.howstuffworks.com/microprocessor.htm>)

- **Pipelining: An Overview** (<http://arstechnica.com/articles/paedia/cpu/pipelining-2.ars/2>) - Good introduction to and overview of CPU pipelining techniques by the staff of Ars Technica
 - **SIMD Architectures** (<http://arstechnica.com/articles/paedia/cpu/simd.ars/>) - Introduction to and explanation of SIMD, especially how it relates to personal computers. Also by Ars Technica
 - **Listado de procesadores** (http://users.erols.com/chare/current_cpus.htm) - Nombres de CPUs y principales características
-

Fuentes y contribuyentes del artículo

Unidad central de procesamiento *Fuente:* <http://es.wikipedia.org/w/index.php?oldid=43883557> *Contribuyentes:* A ver, Airunp, Almiux2009, Alvaro qc, Antur, Antón Francho, Biasoli, Bucephala, Camilo, Claudio Segovia, Cobalttempest, Cratón, Cronos x, Dagavi, Dangarcia, Dario nar, Dark Bane, David0811, Diegusjaimes, Dreitmen, Edmenb, Egaida, Elabra sanchez, Eligna, Elisardojm, Er Komandante, Er conde, Esoya, Fanattiq, Feosegura, Fernando Rosso R, Fibonacci, FolkenX, Gabriel Fernando Rosso R., Galindojotaka, Gargula, GermanX, Humberto, Instigate cjsc (Narine), Isha, Isra00, Jarisleif, Jarke, Javierito92, Javiermhlo, Jesuja, Joseaperez, Juanito1, Jugones55, Julio Cardmat, King of Hearts, Kotxe, Leitzaran, M S, Makahaxi, Marcavia, Marianox, Matdrodes, McMalamute, Mercenario97, Muro de Aguas, Mushii, Nixón, Olea, Oscar ., PabloCastellano, Pan con queso, Piero71, PoLuX124, Poco a poco, Prietoquilmes, Queninosta, RGLago, Raiden32, RaizRaiz, Retama, Rimac, Roberto Fiadone, RoyFocker, Sabbut, Santiperez, Satin, Shooke, Snakefang, Speedplus, Sr Beethoven, Super braulio, The worst user, Tirithel, Tonylash60, TorQue Astur, Tostadora, TraveHacks, Txo, Vatelys, Vitamine, Wilfredor, Willi4m, Willigulip, Yeza, Yrithinnd, Ál, 403 ediciones anónimas

Fuentes de imagen, Licencias y contribuyentes

Archivo:80486dx2-large.jpg *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:80486dx2-large.jpg> *Licencia:* desconocido *Contribuyentes:* A23cd-s, Adambro, Admrboltz, Artnnerisa, CarolSpears, Denniss, Greudin, Kozuch, Martin Kozák, Mattbuck, Rjd0060, Rocket000, 11 ediciones anónimas

Archivo:Edvac.jpg *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Edvac.jpg> *Licencia:* Public Domain *Contribuyentes:* User Matt Britt on en.wikipedia

Archivo:PDP-8i cpu.jpg *Fuente:* http://es.wikipedia.org/w/index.php?title=Archivo:PDP-8i_cpu.jpg *Licencia:* Public Domain *Contribuyentes:* Robert Krten

Archivo:Intel 80486DX2 bottom.jpg *Fuente:* http://es.wikipedia.org/w/index.php?title=Archivo:Intel_80486DX2_bottom.jpg *Licencia:* desconocido *Contribuyentes:* Denniss, Solipsist

Archivo:Mips32 addi.svg *Fuente:* http://es.wikipedia.org/w/index.php?title=Archivo:Mips32_addi.svg *Licencia:* GNU Free Documentation License *Contribuyentes:* German, Nachcommonsverschieber

Archivo:CPU block diagram.svg *Fuente:* http://es.wikipedia.org/w/index.php?title=Archivo:CPU_block_diagram.svg *Licencia:* GNU Free Documentation License *Contribuyentes:* R. S. Shaw (PNG version), and (conversion to SVG)

Archivo:MOS 6502AD 4585 top.jpg *Fuente:* http://es.wikipedia.org/w/index.php?title=Archivo:MOS_6502AD_4585_top.jpg *Licencia:* GNU Free Documentation License *Contribuyentes:* EugeneZelenko, German, Idrougge, Morkork

Archivo:Nopipeline.png *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Nopipeline.png> *Licencia:* GNU Free Documentation License *Contribuyentes:* User:Poil

Archivo:Fivestagespipeline.png *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Fivestagespipeline.png> *Licencia:* GNU Free Documentation License *Contribuyentes:* User:Poil

Archivo:Superscalarpipeline.png *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Superscalarpipeline.png> *Licencia:* GNU Free Documentation License *Contribuyentes:* User:Poil

Licencia