

Apuntes sobre CPU

La unidad de proceso central [central processing unit (CPU)] es la parte de una computadora que interpreta y lleva a cabo las instrucciones contenidas en el software. En la mayoría de las CPUs, esta tarea es dividida entre una unidad de control que dirige el flujo del programa y una o mas unidades de ejecución que realizan las operaciones en datos. Casi siempre, una colección de registros es incluida para retener operandos y resultados del intermedio. El termino CPU es frecuente usado vagamente para incluir otras partes importantes centrales de una computadora tal como caches y controladores de entrada/salida, especialmente en computadoras con chips de microprocesadores modernos que incluyen varios de estas funciones en un circuito integrado físico. Los fabricantes y minoristas de computadoras de escritorios frecuente describen erróneamente el gabinete de la computadora y su contenido como la CPU la cual es engañoso. Una familia de diseños de CPU frecuentemente es referida como una arquitectura de CPU. Arquitecturas de CPU notables incluyen: Intel x86 Zilog IBM System/360 DEC PDP-11, y su sucesor, la VAX Motorola 68000 Sun Microsystems SPARC MIPS Computer Systems Inc. MIPS HP PA-RISC DEC's Alpha The AIM Alliance's PowerPC DEC y Acorn ARM StrongARM SuperH UNIVAC 1100/2200 series (actualmente soportada por [Unisys <http://www.wikipedia.org/wiki/Unisys Corporation>](http://www.wikipedia.org/wiki/Unisys_Corporation) ClearPath IX computers) 1750A, la computadora estándar militar de U.S. AP-101, la computadora del transbordador espacial Nuevas arquitecturas de CPU incluyen: Intel Itanium AMD x86-64

Historicas CPUs importantes han sido: EDSAC- la primera computadora de programa almacenado Apollo Guidance Computer, usado en los vuelos de la luna. **Arithmetic and Logical Unit** Una unidad aritmética y lógica [Arithmetic and Logical Unit (ALU)] es uno de los componentes del núcleo de todas las unidades de proceso central. Es capaz de calcular los resultados de una variedad amplia de cómputos comunes. Las operaciones disponibles mas comunes son las operaciones aritméticas enteras de adición, substracción y multiplicación las operaciones de la lógica bitwise [bitwise operation] de AND, NOT, OR y XOR, y varias operaciones de cambio. Típicamente, un ALU estándar no maneja la división entera ni cualquier operación de punto flotante. Para estos cálculos un componente separado, tal como un divisor o la unidad de punto flotante [Floating Point Unit (FPU)], se usa a menudo, aunque es también posible que unos programas de micro código puedan usar el ALU para emular estas operaciones. El ALU toma como entradas los datos a ser operados y un código de la unidad de control [control unit] indicando que operación para realizar, y para salida proporciona el resultado de la computación. En algunos diseños puede tomar como una entrada y salida un sistema de códigos de la conducción, que pueden usarse para indicar tal como llevar-dentro o llevar-fuera, sobre flujo u otras condiciones. **Véase también** unidad de ejecución ALU multiplicación **Unidad de control** Una unidad de control es la parte de un CPU u otro dispositivo que dirige su operación. Las salidas de la unidad controla la actividad del resto del dispositivo. Una unidad de control puede pensarse como una máquina estatal finita. Una vez las unidades de control para CPUs fueron la lógica ad-hoc, y estas fueron difíciles de diseñar. Ahora estos son frecuente implementados como un microprograma que es almacenado en un almacenamiento de control [control store]. Las palabras de el microprograma son seleccionadas por un micro secuenciador y los bits de es palabras directas controlan las partes diferentes de el dispositivo, incluyendo los registros, unidades aritmética y lógica, registro de instrucciones, buses, y la entrada/salida del off-chip. En computadoras modernas, cada de estos subsistemas puede tener su propio controlador subsidiario, con la unidad de control actuando como un supervisor. **Véase también** Diseño de CPU Arquitectura de la computadora **Diseño de CPU** A una larga magnitud, el diseño de un CPU, o unidad de proceso central, es el diseño de su unidad de control. La manera moderna (de 1965 a 1985) de diseñar lógica de control es escribir un microprograma. El diseño del CPU fue originalmente un proceso ad-hoc. Solo obtener un CPU para trabajo fue un evento substancial, gubernamental y técnico. Innovadores diseños incluyen: cache, memoria virtual, instrucción pipeline, superescalar, CISC, RISC, maquina virtual, emuladores, microprograma, y pila.

Tabla de contenidos [Diseños de CPU para propósito general](#) [1950s: Primeros diseños](#) [1960s: La revolución de la computadora, y CISC](#) [1970s: Integración larga escala](#) [Principios 1980s: Las lecciones de RISC](#) [Mediados 1980s y actualmente: Síntesis](#) [1990 y actualmente: Mirando adelante](#) [Diseño integrado](#) **Diseños de CPU para propósito general 1950s: Primeros diseños** Las computadoras a lo largo y a principios de 1950s fueron similares en que

todas contenían un procesador central que fuera único para esta máquina. Programas escritos para una máquina no correrían en otra, y más frecuente no correrían en otras máquinas de la misma compañía. Cada diseño difirió en los tipos de instrucciones que estas soportaban, y pocas máquinas podrían ser consideradas "propósito general". No había bastante espacio para alambrear en un sistema lleno de instrucciones usando la tecnología de el día (por instancia los sistemas SAGE llenaron los pisos enteros) para que cada máquina integrara una cierta solución. Para finales de los 50s fabricantes comerciales habían desarrollado grandes computadores. La computadora más ampliamente instalada fue la IBM 650, la cual usaba memoria de tambor sobre la cual los programas eran cargados usando cualquier cinta de papel o tarjetas perforadas. Algunas máquinas también incluyeron memoria de núcleo magnético la cual proporcionaba velocidades más altas. Los discos duros también estaban empezando a ponerse populares. Las computadoras son un ábaco automático. El tipo de sistema numérico afecta la manera en que trabajan. A principios de los 50s la mayoría de las computadoras fueron construidas por las tareas de procesamiento numérico específico, y muchas máquinas usaron números decimales como su sistema numérico básico - esto es, las funciones matemáticas de las máquinas trabajaron en base-10 en lugar de base-2 como es común actualmente. Estos no fueron meramente decimal codificado binario [binary coded decimal (BCD)]. Las máquinas actualmente tenían diez tubos de vacío por dígito en cada registro. Un viejo proyecto para la fuerza aérea americana, BINAC intentó hacer un peso ligero, computadora simple para usar aritmética binaria. Impresiono a la industria profundamente. Tan tarde como en 1970, los lenguajes de computadora tal como C eran incapaces de estandarizar su conducta numérica porque las computadoras decimales tenían grupos de usuarios demasiado grande para alienar. Incluso cuando diseñadores usaron un sistema binario, ellos aun tenían muchas ideas extrañas. Algunos usaron aritmética señal-magnitud (-1 = 10001), en lugar de la aritmética moderna complemento-dos (-1 = 11111). La mayoría de las computadoras usaron el juego de caracteres seis-bit, porque ellos adecuadamente codificaron tarjetas de Hollerith. Era una revelación mayor para diseñadores de este periodo para realizar que la palabra de datos debe ser un múltiplo de el tamaño del carácter. Ellos empezaron a diseñar computadoras con 12, 24 y 36 bit de palabra de datos. En esta era, la ley de Grolsch domino el diseño de computadoras: La capacidad de la computadora incremento como el cuadrado de su costo. **1960s: 1960s: La revolución de la computadora, y CISC** Un mayor problema con las primeras computadoras fue que un programa para una computadora no trabajaría en otras. Compañías de computo encontraron que sus clientes tenían algo de razón para permanecer fiel a una marca particular, como la siguiente computadora ellos adquirieron sería incompatible de cualquier modo. A ese punto el precio y desempeño fueron usualmente las únicas preocupaciones. En 1962, IBM apostó la compañía en una nueva manera para diseñar las computadoras. El plan era hacer una familia entera de computadoras que puedan todas correr el mismo software, pero con diferentes desempeños, y en diferentes precios. Como los requerimientos de los usuarios crecieron ellos pudieron subir a las grandes computadoras, y aun mantener todo de sus inversión en programas, datos y medio de almacenamiento. Para hacer esto ellos diseñaron una sola computadora de referencia llamada el System 360 (S/360). El System 360 fue una computadora virtual, un sistema de instrucción de referencia y capacidades que todas las máquinas en la familia soportarían. En orden para proporcionar diferentes clases de máquinas, cada computadora en la familia usaría más o menos la emulación del hardware, y más o menos emulación de microprograma, para crear una máquina capaz de correr el entero sistema de instrucción del System 360. Por instancia una máquina de bajo-fin podría incluir un procesador muy simple para bajo costo. Sin embargo, este requeriría el uso de un emulador de micro código largo para proporcionar el resto de el sistema de instrucción, el cual la haría lenta esta. Una máquina de alto-fin usaría un procesador mucho más complejo que podría procesar directamente más de el diseño de el System 360, así correr un emulador mucho más simple y rápido. IBM eligió el hacer el sistema de instrucción de referencia bastante complejo, y muy capaz. Esta era una opción consciente. Incluso la idea de la computadora era compleja, su "almacén de control" que contiene el microprograma estaría relativamente pequeño, y podría hacerse con memoria muy rápida. Otro efecto importante fue que una sola instrucción podía describir bastante una secuencia compleja de operaciones. Así las computadoras generalmente tendrían que sacar menos instrucciones de la memoria principal, la cual pudo estar hecha más lenta, más pequeña y menos costosa para una combinación dada de velocidad y precio. Una característica de frecuente-descuido de la S/360 era que esta fue el primer sistema de

instrucción diseñada para el procesamiento de datos, en lugar del cálculo matemático. El sistema de instrucción fue diseñado no solo para manipular simplemente números enteros, pero texto, el punto flotante científico (similar para los números usados en una calculadora), y la aritmética decimal necesitada por sistemas de contabilidad. El sistema S/360 fue la primera computadora en hacer mayor uso de decimal codificado binario. Casi todas las siguientes computadoras incluyeron estas innovaciones en alguna forma. Este sistema básico de características es llamado "procesador de conjunto de instrucciones complejas" [complex instruction set computer (CISC)]. En muchos CISCs, una instrucción podía acceder cualquier registro o memoria, usualmente en varios diferentes modos. Esto hizo la CISC fácil de programar, porque un programador podía recordar solo treinta de cien instrucciones, y un conjunto de tres o diez "modos de direccionamiento" en lugar de las miles de instrucciones distintas. Esto fue llamado un "grupo de instrucciones ortogonales". Las arquitecturas PDP-11 y Motorola 68000 son ejemplos de cercanas instrucciones ortogonales.

1970s: Integración larga escala En los 1960s, la computadora Apollo Guidance and Missile Minuteman hicieron el circuito integrado económico y práctico. Alrededor de 1971, los primeros chips de calculadora y reloj empezaron a mostrar que computadoras muy pequeñas podrían ser posibles. El primer microprocesador fue el 4004, diseñado en 1971 para una compañía de calculadoras, y producido por Intel. El 4004 es el ancestro directo de el Intel 80386, incluso ahora mantienen algo de compatibilidad de código. Solo unos pocos años después, la palabra tamaño de el 4004 fue doblada para formar el 8008. Por los mediados de 1970s, el uso de circuitos integrados en computadoras fue común. Esto hizo posible para poner un entero CPU en una sola tarjeta de circuito impreso. El resultado fue que mini computadoras, usualmente con palabras de 16-bit, y 4k a 64k de memoria, vinieron a ser comunes. Se había creído que CISCs eran los tipos de computadora más poderosas, porque su micro código era pequeño y podía ser almacenado en memoria de muy alta velocidad. Los CISCs se creía que eran los tipos de computadoras más poderosas, por que sus micro código era pequeño y podía ser almacenado en una memoria de muy alta velocidad. Los CISCs fueron construidas usando lógica de computadora "bit slice" tal como los chips AMD 2900, con micro código de costumbre. Un componente bit slice es una pieza de un ALU, archivo de registro o micro secuencia. La mayoría de circuitos integrados bit-slice fueron ancho de 4-bits. Por los finales de 1970s, el PDP-11 fue desarrollado, discutiblemente la computadora pequeña más avanzada de su día. Casi de inmediato, CISCs de 32-bit fueron introducidos, VAX y PDP-10. También, para controlar un misil cruise, Intel desarrollo una versión más capaz de su microprocesador 8008, el 8080 IBM continuo para hacer computadoras grandes y rápidas. Sin embargo la definición de grande y rápida ahora significa más que un megabyte de RAM, las velocidades de reloj cerca de cien mega hertz, y de unidades de disco duro de diez de megabytes. El System 370 de IBM era una versión de el 360 para correr ambientes de la computación virtuales. La computadora virtual fue desarrollada en orden para reducir la posibilidad de una falla de software irreparable. Las series de Burrough B300 alcanzaron su porción del mercado grande. Fue una computadora de pila programada en un dialecto de Algo. Uso aritmética de punto-fijo de 64-bit, en lugar de punto-flotante. Todos esos diferentes desarrollos compitieron por una desenfadada porción del mercado.

Principios 1980s: Las lecciones de RISC En los primeros 1980s, investigadores de UC Berkley y de IBM descubrieron que la mayoría de los lenguajes de computadora produjeron solo un pequeño subconjunto de las instrucciones de un CISC. Mucho de el poder de la CPU fue simplemente siendo ignorada en el uso del mundo real. Ellos comprendieron que por hacer la computadora más simple, menos ortogonal, ellos podrían hacerla más rápida y menos costosa al mismo tiempo. Al mismo tiempo las CPUs fueron creciendo rápido en relación a la memoria ellos diseccionaron. Los diseñadores también experimentaron con el uso de sistemas largos de registros internos. La idea fue el caché de resultados del intermedio en los registros bajo el control de el compilador. Esto también redujo el número de modos de direccionamiento y ortogonalidad. Los diseñadores de computadora basados en esta teoría era llamada computadoras de sistema de instrucción reducida [Reduced Instruction Set Computers (RISC)]. Los RISCs generalmente tienen números largos de registros, accedidos por instrucciones más simples, con unas pocas instrucciones específicamente para cargar y guardar datos a memoria. El resultado fue un núcleo CPU muy simple corriendo a muy alta velocidad, soportando las clases exactas de operaciones los compiladores fueron usados de cualquier manera. Un lado bajo para el diseño RISC ha sido que los programas que corren en ellos tienden a ser más grandes. Eso es por que los compiladores tienen que generar secuencias más largas de las instrucciones más simples para

lograr los mismos resultados. Desde que estas instrucciones necesitan ser cargadas desde memoria de todas formas, los desplazamientos de tamaño de código largo alguno del manejo de memoria rápida del diseño de RISC. Recientemente, los ingenieros han encontrado maneras de comprimir los sistemas de instrucciones reducidas así que ellos encajan incluso en sistemas de memoria mas pequeña que CISCs. Ejemplos de tales esquemas de compresión incluyen los Acorn RISC Machine (ARM)s sistema de instrucción "thumb". En aplicaciones que no necesitan ejecutar software binario mas viejo, RISCs comprimidos están viniendo a dominar las ventas. Otro acercamientos a RISCs fue el sistema de instrucción "niladic" o "cero-dirección" [zero-address]. Este acercamiento realizo que la mayoría de espacio en una instrucción era el identificar los operandos de la instrucción. Estas maquinas pusieron los operandos en una pila push-down [last-in, first out (LIFO)]. El sistema de instrucción se complemento con unas instrucciones para sacar y almacenar memoria. La mayoría uso caché simple para proporcionar máquinas de RISC extremadamente rápidas, con código muy compacto. Otro beneficio era que las latencias de la interrupción fueron extremadamente pequeñas, mas pequeñas que la mayoría de las maquinas de CISC (un rasgo raro en maquinas de RISC). La primera computadora cero-dirección fue desarrollada por Charles Moore, coloco seis instrucciones de 5-bit en una palabra de 32-bit, y era un precursor para el diseño VLIW (véase debajo: 1990 y actualmente). Variantes comerciales fueron principalmente caracterizadas como máquinas "FORTH" (adelante), y probablemente fallo porque ese lenguaje se convirtió impopular. También, las maquinas fueron desarrolladas por contratistas de la defensa en exactamente el momento que la guerra fría termino. La perdida de fondo puede haber separado los equipos de desarrollo antes de las compañías podrían realizar mercadeo comercial adecuado. Chips de RISC ahora dominaban el mercado para sistemas-intercalados de 32-bit. Los chips RISC mas pequeños incluso están poniéndose comunes en el costo-sensitivo del mercado de sistemas-intercalados de 8-bit. El mercado principal para CPUs RISC ha sido sistemas que requieren energía baja o el tamaño pequeño Incluso algunos procesadores CISC (basados en arquitecturas que fueron creados antes del RISC se volvieron dominantes) traducen instrucciones internamente en un sistema de instrucción como RISC. Estos chips de CISC incluyen mas nuevos modelos x86 y VAX. Estos números pueden sorprender muchos, porque el "mercado" se percibe para ser computadoras de escritorio [desktop computers]. Sin embargo las computadoras de escritorio son solo una fracción diminuta de las computadoras ahora vendidas. La mayoría de las personas poseen mas computadoras en su automóvil y casa que en sus escritorios. Con los diseños de Intel dominando la inmensa mayoría de todas las ventas del desktop, RISC es encontrado solo en las líneas de computadoras de Apple. **Mediados**

1980s y actualmente: Síntesis En los mediados a últimos de 1980s, los diseñadores empezaron usando una técnica conocida como pipeline instrucción [instruction pipelining], en el cual el procesador trabaja en instrucciones múltiples en diferentes fases de realización. Por ejemplo, el procesador puede estar recuperando los operandos para la siguiente instrucción mientras calcula el resultado de el actual. CPUs modernos pueden usar mas de una docena de tales fases. Una idea similar, introdujo solo unos años después, era ejecutar instrucciones múltiples en paralelo en unidades aritmética-lógica separadas [arithmetic-logic units (ALU)s]. En lugar de operar en solo una instrucción en un tiempo, la CPU buscara varias instrucciones similares que no son dependientes entre si, y correrlos todos al mismo tiempo. Los resultados son entonces entrelazados cuando ellos terminan, haciendo que esto parezca un simple CPU que estaba corriendo dos veces rápido mientras aun esta utilizando solo un bus. Este acercamiento, llamado como diseño de procesador escalar, esta limitado por el grado de instrucción de paralelismo nivel [instruction level parallelism (ILP)], el numero de instrucciones no dependientes en el código del programa. Algunos programas son capaces para correr muy bien en procesadores escalares, notablemente los gráficos. Sin embargo mas problemas generales requieren lógica compleja, y esta casi siempre resulta en instrucciones cuyo resultados son basados en otros resultados -- así haciéndolos incapaces de correr en formas paralelizadas. Ramificarse es una culpable mayor. Por ejemplo, el programa puede agregar dos números y ramificar para un segmento de código diferente si el numero es mas grande que un tercer numero. En este caso incluso si la operación ramificada es enviada a el segundo ALU para procesamiento, todavía debe esperar para los resultados desde la suma. Así corre no muy rápido que si había un solo ALU. Para llegar alrededor de este limite, las tan llamadas diseños superescalares fueron desarrolladas. La lógica adicional en la CPU mira hacia el código como esta enviándose dentro del CPU, y "fuerzas" para ser paralelo. En el caso de

ramificación un número de soluciones son aplicadas, incluyendo mirando hacia los ejemplos pasados de la ramificación para ver cual es más común (llamada predicción de la ramificación), y simplemente corriendo este caso como si no había ninguna rama en absoluto. Un concepto similar es ejecución especulativa, donde ambos lados de una ramificación corren al mismo tiempo, y los resultados de una o la otra son lanzadas afuera una vez la respuesta es conocida. Estos adelantos, los cuales fueron originalmente desarrollados de la investigación para diseños estilo-RISC, permiten procesadores CISC modernos para ejecutar doce o más instrucciones por ciclo del reloj, cuando tradicionales diseños CISC pueden tomar doce o más ciclos para ejecutar solo una instrucción. El micro código resultante es complejo y error-pronso, principalmente debido a las dependencias entre instrucciones diferentes. Además, los electrónicos para coordinar estas ALUs requiere de más transistores, incrementando consumo de poder y calor. En este respecto RISC es superior porque las instrucciones tienen menos interdependencia y hacen implementaciones superescalares más fáciles. Sin embargo, como Intel ha demostrado, los conceptos pueden ser aplicados a un diseño CISC, dando suficiente tiempo y dinero. Nota histórica: Más de estas técnicas (pipeline, predicción de ramificación, etc) fueron originalmente desarrolladas en los últimos 50s por IBM en su computadora mainframe de extensión. **1990 y actualmente: Mirando adelante** El micro código que hace un procesador superescalar es solo -- el código de la computadora. A principios de los 90s, una innovación significativa era realizar que la coordinación de una computadora de múltiple-ALU pudiera moverse dentro de el compilador, el software que traduce unas instrucciones del programador en instrucciones de nivel-máquina. Este tipo de computadora es llamado una computadora de palabra de instrucción muy larga [very long instruction word (VLIW)]. Paralelizando el código en el compilador tiene muchas ventajas prácticas encima de hacer en el CPU. Extrañamente, la velocidad no es ninguno de ellos. Con bastantes transistores, la CPU pedía hacer todo en seguida. Sin embargo todos esos transistores hacen el chip más grande, y por consiguiente más costoso. Los transistores también usan poder, que significa que estos generan calor que debe removerse. El calor también hace el diseño menos fiable. Desde que compilar pasa solo una vez en la máquina del desarrollador, la lógica de control es "en conserva" en la realización final de el programa. Esto significa que no consume ningún transistor, y ningún poder, y por consiguiente es libre, y no genera calor. El CPU resultante es más simple, y corre por lo menos tan rápido como si la predicción estuviera en el CPU. Había varios intentos sin éxitos para comercializar el VLIW. El problema básico es que una computadora VLIW no hace escala a diferente precio y puntos de desempeño, como una computadora micro programada puede. También, las computadoras VLIW maximizan a través de poner, no la latencia, así que para ellos no eran atractivos para los ingenieros diseñando controladores y otras computadoras intercaladas en maquinaria. Los mercados de sistemas intercalados eran frecuentemente pioneros de otros perfeccionamientos de la computadora por proporcionar un largo mercado que no se cuida sobre la compatibilidad con software más viejo. En enero del 2000, una compañía llamada Transmeta tomó el paso interesante de poner a un compilador en la unidad de proceso central, y haciendo al compilador traducir desde un código de byte de referencia (en su caso, instrucciones x86) a un sistema de instrucción VLIW interna. Este acercamiento combina la simplicidad del hardware, bajo poder y velocidad de RISC VLIW con el sistema de memoria principal compacta y software de compatibilidad reversa proporcionada por el CISC popular. Después en el año 2002, Intel tiene la intención de lanzar un chip basado en lo que llaman un diseño de computadora de instrucción paralela explícita [Explicitly Parallel Instruction Computer (EPIC)]. Este diseño supuestamente proporciona la ventaja de VLIW de instrucción aumentada a través de colocar. Sin embargo, esto evita alguno de los problemas de escala y complejidad, por proporcionar explícitamente en cada "bulto" de información de las instrucciones concerniendo sus dependencias. Esta información es calculada por el compilador, cuando estaría en un diseño VLIW. Las primeras versiones también serán compatibles-inverso con el actual software del x86 por medio de un modo de emulación en chip. También, nosotros podemos ver pronto CPUs multi-hilos. Los diseños actuales trabajan mejor cuando la computadora esta corriendo solamente un solo programa, sin embargo casi todos los sistemas operativos modernos le permiten al usuario ejecutar múltiples programas al mismo tiempo. Para la CPU para cambiar encima de y hacer el trabajo en otro programa requiere un contexto-interruptor costoso. En contraste, una CPU multi-hilo podría manejar instrucciones de múltiples programas en seguida. Para hacer esto, tales CPUs incluyen varios sistemas de registros. Cuando un interruptor del contexto ocurre los volúmenes de los

"registros trabajando" son simplemente copiados dentro de uno de un sistema de registros para este propósito. Tales diseños frecuentemente incluyen miles de registros en lugar de los centenares como en un diseño típico. En el lado bajo, los registros tienden a ser algo costosos en un espacio del chip necesitado para implementarlos. Este espacio del chip podría por otra parte usarse para algún otro propósito. Otra huella de desarrollo es combinar lógica reconfigurable con un CPU de propósito general. En este esquema, un lenguaje de computadora especial compila rutinas rápido-ejecutando dentro de un bit-mascara para configurar la lógica. Las partes mas lentas o menos-criticas de el programa pueden ser corridas compartiendo su tiempo en el CPU. Este proceso tiene la capacidad para crear dispositivos tal como radios de software, usando proceso de señal digital para realizar funciones usualmente realizadas por electrónicos analógicos. Como las líneas entre el hardware y software cada vez mas se oscurece debido al progreso en la metodología del diseño y disponibilidad de chips tal como FPGAs y los procesos de producciones económicos, incluso el hardware de fuente-abierta [open-source hardware] ha empezado a aparecer. Comunidades flojamente unida como OpenCores han anunciado recientemente arquitecturas de CPU abiertas completamente tal como el OpenRISC que puede llevarse implementado en FPGAs o en chips producidos de costumbre, por cualquiera, sin pagar cuotas de la licencia. **Diseño intercalado** La mayoría de sistemas de computo en uso actualmente están intercalados en otra maquinaria, tal como teléfonos, relojes, aparatos, e infraestructura. Estos "sistemas intercalados" usualmente tienen requerimientos pequeños para la memoria, tamaños de programa modestos, y muchas veces sistemas de entrada/salida simples pero inusuales. Por ejemplo, la mayoría de los sistemas intercalados carecen de teclados, pantallas, discos, impresoras, u otro dispositivo de entrada/salida reconocible de una computadora personal. Estos pueden controlar motores eléctricos, relays o voltajes, y leer interruptores, resistencias variables u otros dispositivos electrónicos. Frecuente, el único dispositivo de entrada/salida leíble por el humano es un solo diodo de luz emitida, y costo severo o contención de poder pueden incluso eliminar eso. En contraste a computadoras de propósito general, sistemas intercalados frecuentemente busca minimizar la latencia de la interrupción sobre la instrucción puesta a través. Por ejemplo, CPUs de baja latencia generalmente tienen relativamente algunos registros en sus unidades de proceso central. Cuando un dispositivo electrónico causa una interrupción, el intermedio resulta, los registros, tienen que ser guardados antes del software responsable para manejar la interrupción puede correr y entonces debe volverse a poner después de que es terminada. Si hay mas registros, este ahorro y proceso de restauración tarda mas tiempo y aumenta la latencia. **Véase también** BIOS Ingeniería en computacion Unidad de punto flotante bus lado frontal Microcodigo Tarjeta madre