RISC

Reduced Instruction Set Computer ou **Computador com um Conjunto Reduzido de Instruções (RISC)**, é uma linha de arquitetura de processadores que favorece um conjunto simples e pequeno de instruções que levam aproximadamente a mesma quantidade de tempo para serem executadas. A maioria dos microprocessadores modernos são RISCs, por exemplo DEC Alpha, SPARC, MIPS, e PowerPC. O tipo de microprocessador mais largamente usado em desktops, o x86, é mais CISC do que RISC, embora chips mais novos traduzam instruções x86 baseadas em arquitetura CISC em formas baseadas em arquitetura RISC mais simples, utilizando prioridade de execução.

Os processadores baseados na computação de conjunto de instruções reduzido não tem micro-programação, as instruções são executadas diretamente pelo hardware. Como característica, esta arquitetura, além de não ter microcódigo, tem o conjunto de instruções reduzido, bem como baixo nível de complexidade.

A idéia foi inspirada pela descoberta de que muitas das características incluídas na arquitetura tradicional de processadores para ganho de desempenho foram ignoradas pelos programas que foram executados neles. Mas o desempenho do processador em relação à memória que ele acessava era crescente. Isto resultou num número de técnicas para otimização do processo dentro do processador, enquanto ao mesmo tempo tentando reduzir o número total de acessos à memória.

RISC é também a arquitetura adotada para os processadores dos videogames modernos, que proporcionam um hardware extremamente dedicado somente à execução do jogo, tornando-o muito mais rápido em relação a micro computadores com mais recursos, embora com processador x86.

História

Desde os primeiros momentos da indústria de computadores que os cientistas dos principais fabricantes têm estudado métodos e técnicas que possam aperfeiçoar o desempenho e a capacidade dos sistemas de computação.

Alguns aspectos atribuídos ao primeiro-RISC projetos marcados por volta de 1975 incluem as observações que os compiladores de memória restritos da época eram freqüentemente incapazes de tirar proveito dos recursos destinados a facilitar a montagem manual de codificação, e que os modos de endereçamento complexos levavão muitos ciclos para executar devido ao exigido acessos à memória adicional. Foi alegado que tais funções seriam melhor executadas por seqüências de instruções mais simples se isso poderia render implementações pequenas o suficiente para deixar espaço para muitos registros , reduzindo o número de acessos à memória lenta. Nestes projetos simples, a maioria das instruções são de tamanho uniforme e estrutura semelhante, as operações aritméticas são restritas a registros de CPU e instruções a carregar apenas separam e armazenam a memória de acesso. Essas propriedades permitem um melhor balanceamento de estágios no pipeline do que antes, fazendo gasodutos RISC significativamente mais eficiente e permitindo que as freqüências de clock mais altas.

No inicio da década de 1980, surgiram varias pesquisas cujo desejo era de aumentar o desempenho do sistema de computação. Se o desejo era esse, deve-se procurar aperfeiçoar o emprego das instruções que consomem mais tempo de execução, e não se preocupar tanto com instruções mais complexas que raramente são usadas.

Ao longo das décadas em que é usado o computador, observa-se que o hardware em geral tem evoluído, os processadores e a memória, tem evoluído mais rapidamente que o software.

Por outro lado, a manutenção e desenvolvimento de programas não evoluíram em custo/benefício. Apesar de a todo o momento surgirem novas criações e lançamentos, isso acarreta um custo muito elevado. A manutenção também pode ser um problemas, pois os programas oferecidos pelos fabricantes estão longe da ausência de falhas(bugs).

Entre as pesquisas realizadas nessa época, podemos citar a de David Patterson. Juntamente com Carlos Séquin, ele publicou em 1982 [PATT82] um estudo mostrando o desempenho, os parâmetros e elementos de linguagens de alto nível quando compiladas e executadas. Esse artigo descrevia uma nova arquitetura para um processador, propondo

solucionar os problemas de desempenho e custo existentes nas arquiteturas complexas vigentes (CISC). Esta arquitetura foi chamada de RISC, porque criava um processador com pequeno conjunto de instruções.

Esse trabalho é usado como referência para a avaliação do comportamento dos programas de alto nível e seu desempenho dinâmico, no qual os autores apresentaram resultados da compilação e da execução de oito programas diferentes, quatro em linguagem Pascal e quatro em linguagem C.

As análises efetuadas de programas compilados por máquinas de arquitetura CISC, mostraram que os compiladores não eram mais tão espertos quanto os programadores assembly na busca de instruções de máquina complexas. O programa compilador utiliza pouco da grande quantidade de instruções e dos modos de endereçamento que estão disponíveis, pois parece ser difícil analisar o programa de alto nível. Por exemplo, para efetuar a soma entre dois operandos em uma máquina com arquitetura CISC, sendo que um dos valores está em um registrador e o outro na memória, levara um certo tempo gasto para calcular o endereço de acesso a memória. Já em uma máquina com arquitetura RISC, são usadas duas instruções(diferente da arquitetura CISC que utiliza uma instrução), mas como são instruções mais simples, sua execução é bem mais rápida fazendo com que seu tempo total seja menor.

É uma estratégia de desenvolvimento de CPU com base na percepção de que simplificado (em oposição ao complexo) as instruções podem proporcionar maior desempenho, se essa simplicidade permite a execução muito mais rápida de cada instrução. Um computador com base nesta estratégia é um computador com um reduzido conjunto de instruções (também RISC). Existem muitas propostas de definições precisas , mas o termo está sendo lentamente substituído pela arquitetura mais descritiva load-store. famílias bem conhecidas incluem RISC DEC Alpha, a AMD 29k, ARC, ARM, Atmel AVR, MIPS, PA-RISC, Power (incluindo PowerPC), SuperH e SPARC.

Características das Arquiteturas CISC

O nome CISC (Complex Instruction Set Computer) advém do fato de se considerar complexo um conjunto constituído de grande quantidade de instruções, com múltiplos modos de endereçamento, entre outras críticas. Em uma época inicial da computação em que a memória era cara e pequena e, por isso, os códigos gerados pelos compiladores deveriam ser compactos e eficientes na execução. Dessa forma, os projetistas precisavam obter boa densidade do código de máquina, ou seja, cada instrução deveria fazer muito, de modo que o programa completo tivesse poucas instruções.

O surgimento, em 1951, do conceito de microprogramação facilitou o trabalho de projetar instruções complexas, implementado-as em microcódigo. O microcódigo reside em memória de controle, pode-se acelerar suas execução com essas memórias sendo rápidas. A criação de novas instruções é, na maioria das vezes, quase sem custo e sem aumento de espaço, facilitando a implementação do conceito de famílias de processadores. Um bom exemplo disso é a arquitetura x86. Outra vantagem do emprego de microcódigo reside na rapidez da execução de instruções que estão armazenadas em memória (memória ROM de controle) bem mais rápido que a memória convencional.

O primeiro sistema de computação lançado com microcódigo e que originou, também, o conceito de família de computadores foi introduzido pela IBM em 1964, o Sistema IBM/360. Posteriormente, a DEC (Digital Equipament Corporation) introduziu sua família de PDP, mais tarde substituída pelo sistema VAX, um dos melhores exemplos de máquina CISC.

Pode-se concluir que os projetistas de arquiteturas CISC consideram três aspectos básicos: - uso de microcódigo; - construção de conjuntos com instruções completas e eficientes (completeza no conjunto); - criação de instruções de máquina de “alto nível”, ou seja, com complexidade semelhante á dos comandos de alto nível.

Colocados juntos, esses elementos do projeto nortearam a filosofia de construção de processadores CISC por longo tempo, como a família Intel x86, os processadores AMD K e, anteriormente, os sistemas IBM e VAX. Assim é que existem naqueles conjuntos instruções poderosas, do tipo:

- CAS - compare and swap operands (comparar valores e trocas operandos)
- RTR - return and restore codes (retornar e restaurar código)

- SWAP - swap register words (trocar palavras dos registradores)

Em geral o desenvolvimento das arquiteturas CISC tende a seguir algumas regras básicas:

a) Formato de dois operandos mais comum – instruções com campos de origem e destino, como a instrução:

ADD CX, mem (subtrair o valor na memória do valor no registrador CX e colocar resultado no registrador CX)

b) Uso de modos registrador para registrador; registrador para memória e memória para registrador.

c) Uso de múltiplos modos de endereçamento para a memória, incluindo indexação para o caso de vetores.

d) Instruções com largura variável, com a quantidade de bytes variando de acordo com o modo de endereçamento utilizado.

e) As instruções requerem múltiplos ciclos de relógio para sua complexa execução, além do que a quantidade desses ciclos varia de acordo com a largura das instruções. Por exemplo, se uma instrução realiza mais de um acesso à memória para buscar dois operandos, então gasta mais ciclos do que outra que só realiza um acesso.

f) O hardware possui poucos registradores devido ao fato de possuir muitas instruções com acesso à memória e por causa da limitação do espaço no chip usado para a memória de controle.

g) Há também registradores especializados, como o registrador de controle (flags...); de segmento para o ponteiro da pilha, para tratamento de interpretação e outros.

Como é usual acontecer em qualquer área da atividade humana, é raro que algum conceito ou tecnologia importante, obtenha unanimidade entre pesquisadores, técnicos, projetistas e administradores. Este é o caso da arquitetura CISC, a qual sempre foi alvo de críticas e comentários sobre desvantagens e problemas.

Neste texto não cabe posicionamento por este ou aquele fato ou tecnologia, mas sim apresentar todos os elementos possíveis das diversas tendências, no caso entre CISC e RISC.

No entanto, para se compreender o surgimento de processadores com arquitetura RISC deve-se analisar os eventuais problemas indicados para a arquitetura CISC, que levaram pesquisadores e projetistas de sistemas a criar uma alternativa, considerada por elem mais vantajosa.

Para entender melhor as raízes do surgimento da filosofia RISC, pode-se mencionar alguns pontos das arquiteturas CISC citados como problemáticos por um dos criadores de máquinas RISC, David Petterson, em um de seus artigos, induzindo ao projeto de processadores que pudessem, com sua especificação mais simples, reduzir ou eliminar os citados problemas. Na realidade, parece ter sido Patterson quem primeiro definiu ar arquiteturas com muitas e poderosas instruções de CISC e sua maquina protótipo de RISC (o nome esclhido foi RISC-1): Diferenças de velocidade entre memória e processador – no final da década de 1970, a IBM verificou que essa diferença era um problema em seus sistemas, algumas operações eram realizadas por programas, acarretando muitos acessos a uma memória lenta. A solução encontrada foi criar novas instruções de maquina para executar tais operações, podendo-se acreditar que esse foi o início do aumento da quantidade de instruções nas CISC.

Emprego de microcódigo – o surgimento e a real vantagem de custo/beneficio do emprego de micricódigo sobre programação diretamente no hardware induziram os projetistas a criar mais e mais instruções, devido a facilidade e a flexibilidade decorrentes. Desenvolvimento acelerado de linguagens de alto nível – na década de 1980, havia um crescimento acelerado do emprego de linguagens de ato nível, o que conduzia os projetistas de processadores a incluir cada vez mais instruções de máquinas em seus produtos, como o propósito de manter um suporte adequado na compilação.

Densidade de código a ser executado – as arquiteturas CISC procuram obter um código compacto após a compilação, de modo a não consumir memória em excesso. Isso era necessário em uma época em que as memórias eram caras e de reduzindo tamanho. Construindo conjuntos de instruções, cada uma delas mais próxima do significado do comando de ato nível, poder-se-ia obter códigos executáveis mais densos, mais compactos. Alega Patterson que isto acarretaria também mais bits nas instruções (códigos de operações com mais bits devido à quantidade delas, bem como mais modos de endereçamento), o que contrabalançaria aquela pretensa vantagem.

Necessidade de compatibilidade com processadores anteriores – uma das metas sempre seguida pela Intel e outros fabricantes foi a de conservar a compatibilidade entre as versões de seus processadores. Assim o processador 486 veio com apenas algumas instruções novas e todo o código do 386 junto, códigos executáveis para o 386 rodavam também no 486, e os usuários poderiam trocar de computador sem nenhum custo adicional de compilação, etc. O mesmo aconteceu com o Pentium I, II, III e 4. Mesmo isso, embora seja um notório requisito importante de marketing, acarreta uma limitação especificação de novas arquiteturas. Dessa forma, as arquiteturas novas só crescem em quantidade de instruções, visto que o fabricante nunca retira as instruções antigas devido ao problema de compatibilidade.

Não-RISC filosofia de desenvolvimento

Nos primeiros dias da indústria de computadores, a programação era feita em linguagem assembly ou código de máquina, o que incentivou instruções poderosas e fáceis de usar. projetistas da CPU, portanto tentaram fazer as instruções que iria fazer tanto trabalho quanto possível. Com o advento das linguagens de alto nível, os arquitetos do computador também começaram a criar instruções dedicadas a implementar diretamente alguns mecanismos centrais dessas línguagens. Outro objetivo geral foi o de fornecer todos os modos possíveis de endereçamento para cada instrução, conhecido como ortogonalidade, para facilitar a implementação do compilador. As operações aritméticas podiam, portanto, ter muitas vezes resultados, bem como operandos diretamente na memória (em adição ao registrador ou imediata).

A atitude na época era que o desenvolvimento de hardware foi mais maduro do que o projeto do compilador e que esta foi, em si também uma razão para implementar partes da funcionalidade do hardware ou microcódigo em vez de um compilador de memória limitada (ou seu código gerado) sozinho. Esta filosofia de design tornou-se retroactivamente denominada computação de conjunto complexo de instruções (CISC ou Complex Instruction Set Computing), após a filosofia RISC esta entrou em cena. CPUs também tiveram registros relativamente baixos, por várias razões:

- Mais registros também implica uma maior economia de tempo e restauração de conteúdo registrar na máquina de pilha.
- Um grande número de registros requer um grande número de bits de instrução como especificadores de registo, ou seja, um código menos denso. Registradores da CPU são mais caros do que posições de memória externa; jogos de registo de grande porte foram pesados com placas de circuito limitado ou integração de chips.

Uma força importante incentivar a complexidade era o fato das memórias principais serem muito limitadas (da ordem de kilobytes). Foi, portanto, vantajosa para a densidade de informações contidas em programas de computador a ser elevado, levando a características tais como alta codificação, instruções de comprimento variável, fazendo o carregamento de dados, bem como o cálculo (como mencionado acima). Estas questões foram de maior prioridade que a facilidade de decodificação de instruções.

Uma razão igualmente importante foi que as memórias principais foram bastante lentas (um tipo comum foi a memória de núcleo de ferrite), usando a embalagem de informação densa, pode-se reduzir a freqüência com que a CPU tinha de aceder a este recurso lento. Os computadores modernos mostram fatores de limitação semelhantes: memórias principais são lentas em comparação com o CPU e as memórias cache rápidas empregadas para superar este são limitadas em tamanho. Isso pode explicar o fato de que conjuntos de instruções altamente codificados tem provado ser tão útil como projetos RISC em computadores modernos.

Filosofia de Desenvolvimento RISC

Em meados de 1970 investigadores (em especial John Cocke) da IBM (e projetos semelhantes em outros lugares) demonstraram que a maioria das combinações desses modos ortogonais de endereçamento e as instruções não foram utilizados pela maioria dos programas gerados por compiladores disponíveis no momento. Ele revelou-se difícil em muitos casos, para escrever um compilador com mais que a capacidade limitada de tirar proveito dos recursos oferecidos pelos processadores convencionais.

Também foi descoberto que, em implementações de arquiteturas microcodificadas certas operações complexas tendem a ser mais lentas do que uma seqüência de operações mais simples fazendo a mesma coisa. Isso foi em parte um efeito do fato de que muitos projetos foram levados às pressas, com pouco tempo para otimizar ou sintonizar todas as instruções, mas sim apenas aquelas usadas com mais freqüência. Um exemplo famoso foi a instrução do VAX de índice.

Como mencionado anteriormente, a memória de núcleo há muito havia sido mais lenta do que muitos projetos de CPU. O advento de memórias de semicondutores reduziu essa diferença, mas ainda era aparente que mais registradores (e mais tarde caches) permitiria maior freqüência de operação da CPU. registros adicionais exigiriam chips importantes ou áreas bordo que, na época (1975), poderiam ser disponibilizados se a complexidade da lógica de CPU havia sido reduzida.

Contudo um outro impulso de ambos os RISC e outros projetos veio a partir de medições práticas em programas no mundo real. Andrew Tanenbaum resumiu muitos destes, demonstrando que os processadores tiveram muitas vezes tamanhos desproporcionais imediatos. Por exemplo, ele mostrou que 98% de todas as constantes em um programa iriam caber em 13 bits, mas muitos projetos CPU dedicam de 16 ou 32 bits para armazená-los.

Isto sugere que, para reduzir o número de acessos à memória, uma máquina de comprimento fixo pode armazenar constantes em bits não utilizados da palavra de instrução em si, de modo que eles seriam imediatamente prontos quando a CPU precisa deles (muito parecido com endereçamento imediato em um desenho convencional). Estes necessários e pequenos opcodes, ocorreram a fim de deixar espaço para uma constante com um tamanho razoável em uma palavra de instrução de 32 bits.

Uma vez que muitos programas do mundo real passam a maior parte do seu tempo executando operações simples, alguns pesquisadores decidiram concentrar em fazer as operações o mais rápido possível. A velocidade do clock de uma CPU é limitado pelo tempo que leva para executar a mais lenta sub-operação de qualquer instrução, diminuindo o tempo de ciclo que, muitas vezes acelera a execução de outras instruções. O foco na "instruções reduzida" levou ao. resultando máquina que está sendo chamado de "computador conjunto reduzido de instruções" (RISC). O objetivo era fazer com que as instruções tão simples que poderiam ser facil conduta, a fim de atingir uma capacidade de clock única em altas freqüências.

Mais tarde verificou-se que uma das características mais importantes dos processadores RISC é que a memória externa só era acessível por uma instrução de armazenar carga. Todas as outras instruções foram limitadas aos registos internos. Isso simplificou muitos aspectos do design do processador: permitir que as instruções para ser de comprimento fixo, simplificação de condutas, e isolar a lógica para lidar com o atraso na conclusão de um acesso à memória (cache miss, etc) para apenas duas instruções. Isso levou a RISC projetos que estão sendo referidos como arquiteturas load / store.

Instrução de definição do tamanho e terminologia alternativa

Um equívoco comum da expressão "computador conjunto reduzido de instruções" é a idéia equivocada de que as instruções são simplesmente eliminadas, resultando em um menor conjunto de instruções. De fato, ao longo dos anos, conjuntos de instruções RISC cresceram em tamanho, e hoje muitos deles têm um conjunto maior de instruções do que muitos processadores CISC. Alguns processadores RISC, como o Transputer INMOS têm conjuntos de instruções tão grandes como, por exemplo, o CISC IBM System/370 e, inversamente, o DEC PDP-8 - claramente um

CPU CISC porque muitas de suas instruções envolvem múltiplos acessos à memória - tem apenas 8 instruções básicas, além de algumas instruções prorrogadas.

O termo "redução" em que a frase foi concebido para descrever o fato que a quantidade de qualquer trabalho único a instrução realizada é reduzida - no máximo um ciclo de memória de dados único - em comparação com as "instruções complexas" do CISC CPUs que podem exigir dezenas de dados ciclos de memória para executar uma única instrução. Em particular, os processadores RISC geralmente têm instruções separadas para I / O(input/output, entrada/saida) e de processamento de dados; consequentemente, os observadores da indústria começaram a utilizar os termos "registro-registro" ou "carga armazenada "para descrever os processadores RISC.

Alguns CPUs foram retroativamente apelidados RISC - um artigo da revista Byte certa vez se referiu à 6502 como "o processador RISC original", devido à sua instrução simplistas e quase ortogonais (mais instruções de trabalho, com a maioria dos modos de endereçamento), bem como a sua 256 página-zero "registos". O 6502 está sem carga design do armazenamento no entanto: operações aritméticas pode ler a memória e as instruções de como INC ROL e até mesmo modificar a memória. Além disso, ortogonalidade é igualmente associada frequentemente com "CISC". No entanto, o 6502 pode ser considerado semelhante ao RISC (máquinas e precoce) no fato de que ele não usa seqüência de microcódigo. No entanto, o fato bem conhecido que a empresa empregava mais longos, porém menos ciclos de clock em relação a muitos microprocessadores contemporâneos foi devido a um design mais assíncrono com menor subdivisão de ciclos de máquina interna. Isso é semelhante às máquinas de início, mas não para RISC.

Alguns processadores foram projetados especificamente para ter um conjunto muito pequeno de instruções - mas esses projetos são muito diferentes dos desenhos clássicos RISC, para que eles tenham recebido outras denominações, tais como computador instrução mínima set (MISC), a Instrução Zero Set Computer (ZISC) , um conjunto de instruções de computador (OISC), transporte acionado arquitetura (ATT), etc

Alternativas

RISC foi desenvolvida como uma alternativa ao que é hoje conhecido como CISC. Ao longo dos anos, outras estratégias foram implementadas como alternativa para o RISC e CISC. Alguns exemplos são a arquitetura VLIW, MISC, OISC, processamento paralelo maciço, matriz sistólica, computação reconfigurável, e o fluxo de dados.

Características das arquiteturas RISC

Em meados de 1970 investigadores (em especial John Cocke) da IBM (e projetos semelhantes em outros lugares) demonstraram que a maioria das combinações desses modos ortogonais de endereçamento e as instruções não foram utilizados pela maioria dos programas gerados por compiladores disponíveis no momento. Ele revelou-se difícil em muitos casos, para escrever um compilador com mais que a capacidade limitada de tirar proveito dos recursos oferecidos pelos processadores convencionais.

Também foi descoberto que, em implementações de arquiteturas microcodificadas certas operações complexas tendem a ser mais lentas do que uma seqüência de operações mais simples fazendo a mesma coisa. Isso foi em parte um efeito do fato de que muitos projetos foram levados às pressas, com pouco tempo para otimizar ou sintonizar todas as instruções, mas sim apenas aquelas usadas com mais freqüência. Um exemplo famoso foi a instrução do VAX de índice.

Como mencionado anteriormente, a memória de núcleo há muito havia sido mais lenta do que muitos projetos de CPU. O advento de memórias de semicondutores reduziu essa diferença, mas ainda era aparente que mais registradores (e mais tarde caches) permitiria maior freqüência de operação da CPU. registros adicionais exigiriam chips importantes ou áreas bordo que, na época (1975), poderiam ser disponibilizados se a complexidade da lógica de CPU havia sido reduzida.

Contudo um outro impulso de ambos os RISC e outros projetos veio a partir de medições práticas em programas no mundo real. Andrew Tanenbaum resumiu muitos destes, demonstrando que os processadores tiveram muitas vezes

tamanhos desproporcionais imediatos. Por exemplo, ele mostrou que 98% de todas as constantes em um programa iriam caber em 13 bits, mas muitos projetos CPU dedicam de 16 ou 32 bits para armazená-los. Isto sugere que, para reduzir o número de acessos à memória, uma máquina de comprimento fixo pode armazenar constantes em bits não utilizados da palavra de instrução em si, de modo que eles seriam imediatamente prontos quando a CPU precisa deles (muito parecido com endereçamento imediato em um desenho convencional). Estes necessários e pequenos opcodes, ocorreram a fim de deixar espaço para uma constante com um tamanho razoável em uma palavra de instrução de 32 bits.

Uma vez que muitos programas do mundo real passam a maior parte do seu tempo executando operações simples, alguns pesquisadores decidiram concentrar em fazer as operações o mais rápido possível. A velocidade do clock de uma CPU é limitado pelo tempo que leva para executar a mais lenta sub-operação de qualquer instrução, diminuindo o tempo de ciclo que, muitas vezes acelera a execução de outras instruções. O foco na "instruções reduzida" levou ao. resultando máquina que está sendo chamado de "computador conjunto reduzido de instruções" (RISC). O objetivo era fazer com que as instruções tão simples que poderiam ser facil conduta, a fim de atingir uma capacidade de clock única em altas freqüências.

Mais tarde verificou-se que uma das características mais importantes dos processadores RISC é que a memória externa só era acessível por uma instrução de armazenar carga. Todas as outras instruções foram limitadas aos registos internos. Isso simplificou muitos aspectos do design do processador: permitir que as instruções para ser de comprimento fixo, simplificação de condutas, e isolar a lógica para lidar com o atraso na conclusão de um acesso à memória (cache miss, etc) para apenas duas instruções. Isso levou a RISC projetos que estão sendo referidos como arquiteturas load / store.

O resultado do trabalho de vários pesquisadores de diferentes locais, fez com que surgissem processadores com características bem diferentes dos já conhecidos por CISC. Eles trabalhavam visando encontrar novas possibilidades para aperfeiçoar o desempenho dos processadores, usando caminhos diferentes dos adotados pela arquitetura CISC. Para o desenvolvimento de arquiteturas RISC, teve inicio por três caminhos próximos:

1. O primeiro deles, foi em meados da década de 1970 com um projeto da IBM, desenvolvido pelo pesquisador daquela empresa: John Cocke. Esse projeto foi denominado IBM 801, o qual nem se tornou comercial, mas serviu de base para os desenvolvimentos seguintes da IBM nessa área.
2. Anos depois em medos da década de 1980, na Universidade de Stanford, na Califórnia, EUA, foi desenvolvida outra pesquisa pela equipe de John Hennessey [HENN84]. Esse projeto se tornou redundante nos processadores MIPS e na criação da empresa MIPS Technology Inc.
3. Na mesma época (1980), na Universidade da Califórnia, EUA, foram desenvolvidas pesquisas muito semelhantes. Lideradas por David Patterson, foram criados os primeiros protótipos, RISC-1 e RISC-2, que acabaram se tornando base para o surgimento dos processadores SPARC.

Estão descritos a seguir os elementos base que constituem a arquitetura RISC(Reduced Instruction Set Computer): - pequeno conjunto de instruções, todas com largura fixa; - execução otimizada de chamada de função; - menor quantidade de modos de endereçamento; - uso de pipelining; - execução rápida de cada instrução(uma a cada ciclo de clock).

Menor quantidade de instruções: talvez a característica mais marcante das arquiteturas RISC, seja a de possuir um conjunto de instruções menor(todas também com largura fixa), que as máquinas que possuíam a arquitetura CISC, porém com a mesma capacidade. Vem daí o nome dado a arquitetura RISC (computadores com um conjunto reduzido de instruções). A SPARC, da Sun, possuía um conjunto de cerca de 50 instruções, a VAX-11/780 tinha ate 300 instruções, o Intel 80486 foi apresentado com 200 instruções e os Pentium possuem mais de 200 instruções.

Com o conjunto de instruções reduzido e cada uma delas tendo suas funções otimizadas, os sistemas possuíam um resultado melhor em questão de desempenho. Em virtude do conjunto reduzido das instruções, acarretavam em programas um pouco mais longos.

Mesmo assim, o conjunto de instruções de máquina reduzido, possibilitava outras vantagens, que entre elas podemos citar: - Com uma menor quantidade de chips e transistores, conseqüentemente, o especo físico era maior com um custo reduzido;

- A redução da complexidade do decodificador de instruções, reduz também o tempo de decodificação.

A busca pelas instruções foi facilitada porque todas as instruções possuem o mesmo tamanho em bits e alinhadas a largura da palavra. Por isso não é mais necessário verificar o tamanho do contador de instruções, pois ele é incrementado sempre com o mesmo valor. Com isso, não tem risco da instrução ocupar duas páginas de dados diferentes, porque traria problemas para o sistema operacional na hora do acesso.

Execução otimizada de chamadas de função: outra evolução da arquitetura RISC para a arquitetura CISC tem relação com a chamada de retinas e passagem de parâmetros. Estudos indicam que as chamadas de funções consomem um tempo significativo de processador. Elas requerem poucos dados, mas demoram muito tempo nos acessos a memória.

Em virtude disso, na arquitetura RISC foram utilizados mais registradores. As chamadas de função que na arquitetura CISC ocorriam com acessos a memória, mas na RISC isso era feito dentro do processador mesmo, utilizando os registradores que foram colocados a mais.

Modo de execução com Pipelining: uma das características mais relevantes da arquitetura RISC é o uso de pipelining, mesmo sabendo que ela tem um funcionamento mais efetivo quando as instruções são todas bastante parecidas.

Imaginando estágios de uma linha de montagem, não é interessantes que um estágio termine antes do outro, pois nesse caso perde-se a vantagem da linha de montagem. O objetivo de cada instrução, é completar um estágio de pipeline em um ciclo de clock, mas esse objetivo nem sempre é alcançado.

O processamento de uma instrução é composto pelo menos por cinco fases:

Instruction fetch;

- Instruction decode;
- Operand fetch;
- Execution;
- Write back.

Hoje em dia o pipeline não se limita a apenas 5 estágios, mas pode chegar a 20 ou 30 estágios (Intel Pentium 4). No entanto, para que todo o processo funcione é necessário que determinadas restrições se verifiquem. A prioridade é que todas as instruções permaneçam em cada estágio o mesmo tempo, para que:

- O sinal de relógio seja usado como cadência de processamento;
- Não sejam necessários “buffers”;

Execução de cada instrução em um ciclo de clock: se o uso do pipelining se considera uma característica importante da arquitetura RISC, a execução de uma instrução por ciclo de clock é mais importante, segundo os que estabeleceram suas bases. Um dos pontos mais negativos das arquiteturas RISC é o longo tempo de execução de cada instrução. Com o surgimento dessa nova arquitetura, cada instrução passou a ser executada a cada ciclo de clock.

Características	Considerações
Menor quantidade de instruções que as máquinas CISC	• Simplifica o processo de cada instrução e torna este item mais eficaz; • Embora o processador RS/600 possua 184 instruções, ainda assim é bem menos que as 303 instruções dos sistemas VAX-11. Além disso, a maioria das instruções é realizada em 1 ciclo de clock, o que é considerado o objetivo maior dessa arquitetura.
Execução otimizada de chamadas de função	• As máquinas RISC utilizam os registradores da UCP (em maior quantidade que os processadores CISC) para armazenar parâmetros e variáveis em chamadas de rotinas e funções. Os processadores CISC usam mais a memória para a tarefa.
Menor quantidade de modos de endereçamento	• As instruções de processadores RISC são basicamente do tipo Load/Store, desvio e de operações aritméticas e lógicas, reduzindo com isso seu tamanho. • A grande quantidade de modos de endereçamento das instruções de processadores CISC aumenta o tempo de execução das mesmas.
Utilização em larga escala de pipelining	• Um dos fatores principais que permite aos processadores RISC atingir seu objetivo de completar a execução de uma instrução pelo menos a cada ciclo de clock é o emprego de pipelining em larga escala.

Tabela 1 - <Monteiro M. A. Introdução à organização de computadores, 2007, p. 383>

RISC Inicial

O primeiro sistema que hoje seria conhecido como RISC era o supercomputador CDC 6600, projetado em 1964, uma década antes que o termo foi inventado. O CDC 6600 tinha uma arquitetura load-store, com apenas dois modos de endereçamento (Registrar e registrar constante imediato) e 74 opcodes (enquanto um Intel 8086 tem 400). O 6600 tinha onze unidades funcionais pipeline para aritmética e lógica, além de cinco unidades de carga e duas unidades de armazenamento, a memória tinha bancos múltiplos assim todas as unidades de armazenamento de carga poderiam operar ao mesmo tempo. O ciclo de clock base / taxa de emissão de instrução foi de 10 vezes mais rápido que o tempo de acesso à memória. Jim Thornton e Seymour Cray projetou como um processador central de processamento de números, apoiados por 10 computadores simples, chamados de "processadores periféricos" para lidar com I / O e outras funções do sistema operacional. Assim, o comentário de brincadeira depois que a sigla RISC realmente significava " Realmente Inventado por Seymour Cray.

Outra máquina de armazenar carga inicial foi o minicomputador Data General Nova, projetada em 1968 por Edson de Castro. Tinha um conjunto de instruções RISC quase puro, muito semelhante ao dos processadores ARM de hoje, porém, não foi citado como tendo influenciado os designers ARM, embora Novas estavam em uso na Universidade de Cambridge, Laboratório de Informática no início de 1980.

A primeira tentativa de fazer um chip baseado em RISC CPU era um projeto da IBM, que começou em 1975. Nomeando após o edifício onde funcionou o projeto, o trabalho levou à família de processadores IBM 801, que foi amplamente utilizado no interior de hardware da IBM. O 801 acabou por ser produzida em uma única forma de chip como o ROMP, em 1981, que era para 'Pesquisa OPD [Office Products Division] micro processador. Como o nome indica, este processador foi projetado para "mini" tarefas, e quando a IBM lançou o IBM RT-PC baseado no projeto em 1986, o desempenho não era aceitável. No entanto, o 801 inspirado vários projectos de investigação, incluindo novos na IBM que acabaria por levar à sua rede eléctrica.

A maioria dos projetos públicos RISC, no entanto, foram os resultados dos programas de universidade de pesquisa executado com financiamento do Programa de VLSI DARPA. O Programa de VLSI, hoje praticamente desconhecido, levou a um grande número de avanços no design de chips, fabricação, gráficos e até mesmo um computador.

Projeto UC Berkeley RISC começou em 1980 sob a direção de David Patterson e Carlo H. Lantejoula, com base no ganho de desempenho através da utilização de pipelining e um uso agressivo de uma técnica conhecida como registro de janelas. Em uma CPU normal tem um pequeno número de registros, e um programa pode usar qualquer

registo a qualquer momento. Em uma CPU com o registro de Janelas, há um grande número de registros, por exemplo, 128, mas os programas podem usar apenas um pequeno número deles, por exemplo, 8, a qualquer momento. Um programa que limita-se a 8 registros por procedimento pode fazer chamadas de procedimento muito rápido: A chamada simplesmente move a janela "para baixo" por 8, para o conjunto de 8 registros utilizados por esse procedimento, e o retorno move a janela traseira. (Em um CPU normal, a maioria das chamadas deve poupar pelo menos os valores poucos registros 'para a pilha, a fim de utilizar esses registos como espaço de trabalho, e restaurar os valores de retorno).

O projeto RISC entregou o processador RISC-I em 1982. Composta de apenas 44.420 transistores (em comparação com as médias de cerca de 100 mil em projetos novos CISC da época) RISC eu tinha apenas 32 instruções, e ainda completamente superado qualquer outro projeto de chip único. Eles seguiram para o transistor 40.760, de 39 anos instruções RISC-II, em 1983, que decorreu ao longo de três vezes mais rápido que RISC-I.

Na mesma época, John L. Hennessy começou um projeto semelhante chamado MIPS na Universidade de Stanford em 1981. MIPS centrou-se quase inteiramente sobre o gasoduto, certificando-se que poderia ser executado como "completa" possível. Embora pipelining já estava em uso em outros projetos, várias características do chip MIPS fez seu pipeline muito mais rápido. O mais importante, e talvez chato, desses recursos foi a exigência de que todas as instruções de ser capaz de concluir em um ciclo. Essa exigência permitiu que o pipeline para ser executado em taxas de dados muito maior (não houve necessidade de atrasos induzidos) e é responsável por muito do desempenho do processador. No entanto, também teve o efeito colateral negativo de eliminação de muitas instruções potencialmente úteis, como uma multiplicação ou uma divisão.

Nos primeiros anos, os esforços RISC eram bem conhecidos, mas confinado aos laboratórios universitários, que havia criado. O esforço de Berkeley tornou-se tão conhecido que acabou se tornando o nome de todo o conceito. Muitos na indústria de informática criticou que os benefícios de desempenho foram pouco provável que se traduzem em ajustes do mundo real, devido à diminuição da eficiência da memória de várias instruções, e que essa foi a razão que ninguém estava usando. Mas a partir de 1986, todos os projetos de pesquisa RISC começaram a entregar os produtos.

RISC posterior

A pesquisa de Berkeley não estava diretamente comercializada, mas o projeto RISC-II foi usado pela Sun Microsystems para desenvolver o SPARC, pela Pyramid tecnologia para desenvolver sua linha de máquinas de gama média, multi-processador, e em quase todas as outras empresas, alguns anos depois. Foi o uso da Sun de um chip RISC em suas novas máquinas, que demonstrou que os benefícios do RISC eram reais, e as suas máquinas rapidamente ultrapassou a concorrência e, essencialmente, assumiu o mercado de workstations inteiro.

John Hennessy deixou Stanford (temporariamente) para comercializar o projeto MIPS, começando a companhia conhecida como MIPS Computer Systems. Seu primeiro projeto foi uma segunda geração de chips MIPS conhecido como o R2000. projetos MIPS passou a se tornar um dos chips RISC mais utilizada quando foram incluídos na PlayStation e Nintendo 64 consoles de jogos. Hoje eles são um dos processadores mais comuns embutidos em uso para aplicações high-end.

IBM aprendeu com o fracasso RT-PC e passou a projetar o RS/6000 com base na sua nova arquitetura POWER. Ela, então, mudou seu sistema AS/400 existentes para chips POWER, e encontrou para sua surpresa que, mesmo o conjunto de instruções muito complexas correu consideravelmente mais rápido. Power também iria encontrar-se em movimento "para baixo" na escala para produzir o desenho PowerPC, o que elimina muitos dos "IBM apenas" instruções e criou uma aplicação de um único chip. Hoje, o PowerPC é um dos CPUs mais comumente usados para aplicações automotivas (alguns carros têm mais de 10 deles no interior). Foi também o CPU utilizado em máquinas Apple Macintosh 1994-2006. (A partir de fevereiro de 2006, a Apple mudou sua linha de produção principal para processadores Intel x86).

Quase todos os outros fabricantes rapidamente se juntou. Do Reino Unido esforços de pesquisa, resultou na transputer INMOS, a Acorn Archimedes e do Advanced RISC Machine line, que é um enorme sucesso hoje. Empresas com projetos existentes CISC também rapidamente se juntaram à revolução. Intel lançou o i860 e i960 pelo final dos anos 1980, embora não tenham sido muito bem sucedida. Motorola construiu um novo projeto chamado de 88.000 em homenagem ao seu famoso CISC 68000, mas viu quase nenhum recurso, e, eventualmente, abandonou e se juntou a IBM para produzir o PowerPC. A AMD lançou o seu 29000, que viria a se tornar o mais popular de projeto RISC do início de 1990.

Hoje a grande maioria de todas as CPUs de 32 bits em uso são CPUs RISC e microcontroladores. técnicas de projeto RISC oferece potência mesmo em tamanhos pequenos, e assim tornou-se dominante para a baixa potência processadores de 32 bits. Sistemas embarcados são, de longe o maior mercado de processadores: enquanto que uma família pode possuir um ou dois PCs, telefones de seu carro (s), celulares e outros dispositivos que podem conter um total de dezenas de processadores embarcados. RISC também tinha tomado conta do mercado de estações de trabalho maior para muito dos anos 90 (até a retomada pelo baixo custo de soluções baseadas em PC). Após o lançamento do Sun SPARCstation os outros vendedores se apressaram para concorrer com soluções RISC própria. O mercado hoje servidor high-end é quase totalmente RISC [carece de fontes?], E o 1 º lugar entre os supercomputadores a partir de 2008 [atualização] é realizada pelo sistema da IBM Roadrunner, que usa processadores Power Architecture baseado Cell para oferecer parte de seu poder de computação, embora muitos outros supercomputadores usam processadores x86 CISC vez.

RISC e x86

No entanto, apesar de muitos sucessos, RISC tem feito poucas incursões no PC desktop e servidor mercados de commodities, onde a plataforma Intel x86 continua a ser a arquitetura do processador dominante. Existem três razões principais para isso:

1. A base muito grande de aplicações do PC proprietários são escritos para 86, enquanto que nenhuma plataforma RISC tem uma base semelhante instalado, e isto significava usuários de PC foram fechados na 86.

2. Embora RISC era de fato capaz de ampliar em muito o desempenho rápido e barato, a Intel se aproveitou de seu grande mercado gastando vastas quantias de dinheiro sobre o desenvolvimento do processador. Intel poderia gastar muitas vezes, tanto quanto qualquer fabricante RISC de melhorar o projeto e fabricação de baixo nível. O mesmo não pode ser dito sobre as empresas menores, como Cyrix e NexGen, mas eles perceberam que poderiam aplicar (bem) práticas de design de conduta também para a arquitetura x86, como nos 486 e Pentium. A série 6x86 MII e fez exatamente isso, mas era mais avançada, implementou execução especulativa superescalar através do registo de renomeação, diretamente no nível 86-semântico. Outros, como o AMD K5 Nx586 e fez o mesmo, mas indiretamente, via buffering microcódigo dinâmica e semi-independente de agendamento superescalares e instrução expedição ao nível micro-operação (mais velhos ou mais simples "CISC" projetos normalmente executa seqüências rígidas micro-operação diretamente). O primeiro chip de buffer disponível implantação dessas dinâmicas e técnicas de programação foi o Nx586 NexGen, lançado em 1994, o AMD K5 foi severamente adiado e lançado em 1995.

3. Mais tarde, os processadores mais poderosos, como Intel P6, AMD K6, AMD K7, Pentium 4, etc buffering semelhantes, utilizavam dinâmicas e princípios de programação e execução superescalar flexível (e especulativa) realização de operação de micro-seqüências geradas a partir de 86 paralelos diversos estágios de decodificação. Hoje, essas idéias foram aperfeiçoadas (cerca de 86 pares em vez disso são fundidas numa mais complexa operação de micro, por exemplo) e ainda são usados pelos processadores x86 modernos, tais como Intel Core 2 e AMD K8.

Enquanto os primeiros projetos RISC foram significativamente diferentes designs contemporâneos CISC, em 2000 o maior desempenho em CPUs da linha RISC eram quase indistinguíveis dos CPUs mais eficientes da linha CISC.

Um número de fornecedores, incluindo a Qualcomm, estão tentando entrar no mercado de computadores com dispositivos baseados em ARM apelidado smartbooks, cavalgando a tendência de netbooks e crescente aceitação das distribuições Linux, um número que já constrói ARM. Outras empresas estão optando por usar o Windows CE.

RISC X CISC

Embora haja manualmente um número razoável de adeptos das máquinas que possuem arquiteturas RISC, também há, e em grande quantidade, aqueles que relacionam diversas desvantagens desses processadores, advogando em favor da arquitetura CISC. Vários podem ser os temas para discussão sobre RISC e CISC, um dos quais se refere ao desempenho do processador na execução de um programa. De modo geral, os vendedores e outros pesquisadores tendem a medir o desempenho através de programas de teste (benchmarks). No entanto, os referidos programas possuem uma série de complicações na interpretação de seus resultados em função do tipo de ambiente que utilizaram e da natureza dos testes.

Os defensores da arquitetura CISC propugnam que instruções mais complexas redundarão em um código-objeto menor, o que reduz um consumo de memória, com reflexos no custo do sistema. Isso não é necessariamente correto se considerarmos que uma menor quantidade de instruções nem sempre acarreta menor quantidade de bits (e é a quantidade efetiva de bits que consome menos memória e a menor custo). Se cada instrução CISC possuir mais operandos que as instruções RISC e se cada um de seus operandos ocupar uma boa quantidade de bits na instrução, então poderemos ter um programa CISC maior em bits do que um programa em máquina RISC, apesar de o programa para o processador RISC possuir maior quantidade de instruções.

Por exemplo, um programa escrito para rodar em um processador CISC pode gastar 150 instruções de máquina; cada uma das instruções possui código de operação de 8 bits, podendo ser de um, dois, e três operandos. Cada campo operando ocupa 18 bits e ainda há um campo para outras ações, com 4 bits de tamanho. Em média, as instruções têm um total de 50 bits. Um programa para realizar o mesmo problema, escrito para rodar em um processador RISC, pode ter 220 instruções, que em média ocupam 32 bits. As instruções são, em sua esmagadora maioria, de dois operandos, porém os operandos são valores em registradores e, por isso, as instruções não consomem muitos bits para endereçar os dois registradores. O programa para a máquina CISC gastaria 7.500 bits, enquanto o programa para a máquina RISC, mesmo possuindo mais 70 instruções que o processador CISC, consumiria 7.040 bits.

Os defensores da arquitetura CISC alegam que estas máquinas executam mais rapidamente os programas escritos em linguagem de alto nível devido á pouca quantidade de códigos binários executáveis. No entanto, o tempo que cada instrução leva para ser executada nem sempre conduz à confirmação dessa assertiva. Máquinas RISC, tendem a executar instruções bem mais rápido por que:

a)As instruções possuem C.Op. com menor quantidade de bits e, portanto, o tempo de decodificação é menor que o das maquinas CISQ;

b)As instruções são executadas diretamente pelo hardware e não por um microprograma. Máquinas RISC não são microprogramadas e, assim, tendem a executar as instruções de modo mais rápido.

Processadores RISC são também otimizados para operações de uma única tarefa devido ao grande numero de registradores que possuem e à grande quantidade de estágios de pipelining. A melhor maneira de obter um bom desempenho dos processadores RISC é executar um programa de teste (um benchmark), o qual possui exatamente esta característica: um grande número de operações similares, em uma única tarefa. Neste ponto, e antes de serem apresentados alguns exemplos de arquiteturas clássicas RISC, deve-se observar que a discussão e o detalhamento de características de processadores que seguem a filosofia CISC e os que seguem a filosofia RISC são atualmente em menos crítica do que foi em anos anteriores, quando havia realmente uma nítida distinção entre ambas. Com o passar do tempo, o avanço da tecnologia em hardware, modificou a visão de alguns projetistas e as adaptações foram surgindo de ambas as partes, de modo que atualmente não se pode afirmar com absoluta certeza que um determinada processador segue rigorosamente a linha RISC nem que outro segue rigorosamente a linha CISC.

Os últimos processadores Intel possuem um núcleo de execução RISC, assim como os processadores de 64 bits, Itanium, seguem, em grande parte, as características definidas para um componente RISC. E é bem verdade, que processadores Power sempre possuíram uma quantidade apreciável de instruções, embora todas com largura fixa.

Benefícios Decrescentes

Ao longo do tempo, as melhorias nas técnicas de fabricação de chips têm melhorado o desempenho de forma exponencial, de acordo com a lei de Moore, enquanto melhorias na arquitetura foram relativamente pequenas. implementações modernas CISC têm implementado muitas das melhorias introduzidas pelo desempenho RISC, tais como transferência de um único clock de instruções simples. Compiladores também se tornaram mais sofisticados e são mais capazes de explorar complexos, bem como as instruções simples sobre arquiteturas CISC, muitas vezes com cuidado otimizar tanto a seleção de instrução e da instrução e da aquisição de dados em dutos e caches. A distinção RISC CISC borrou significativamente na prática.

Histórias de Sucesso RISC

Projetos RISC levaram a um certo número de plataformas e arquiteturas bem sucedidas, algumas das maiores sendo:

- ARM - A arquitetura ARM domina o mercado de baixa potência e sistemas de baixo custo embutido (normalmente 100-500 MHz em 2008). ARM Ltd., que licencia propriedade intelectual, em vez de chips de fabricação, relatou que 10 bilhões de chips licenciados tinham sido enviados no início de 2008. As várias gerações, as variantes e as implementações do núcleo ARM são utilizados em mais de 90% da eletrônica móvel dispositivos, incluindo quase todos os modernos telefones celulares, mp3 players e players de vídeo portáteis. Alguns exemplos de destaque são:
- O iPod da Apple (costume ARM7TDMI SoC)
- O iPhone da Apple e o iPod Touch (Samsung ARM1176JZF, ARM Cortex-A8, a Apple A4)
- iPad o SoC (A4 Apple baseados em ARM)
- O Palm e Pocket PC PDAs e smartphones (Marvell família XScale, Samsung SC32442 - ARM9)
- O RIM BlackBerry smartphone dispositivos de E / e-mail.
- Microsoft Windows Mobile
- O Nintendo Game Boy Advance (ARM7TDMI)
- O Nintendo DS (ARM7TDMI, ARM946E-S)
- O Sony Network Walkman (Sony in-house chip baseado em ARM)
- O T-Mobile G1 (Android HTC Dream, MSM7201A Qualcomm ARM11 @ 528 MHz)
- PowerPC Arquitetura - A arquitetura PowerPC é uma arquitetura RISC popular baseado que domina a restrição de desempenho e de alimentação integrada mercados de dispositivos, tais como equipamentos de comunicação (Roteadores, Switches), equipamentos de armazenamento, etc
- Linha MIPS do MIPS, encontrado na maioria dos computadores da SGI e do PlayStation, PlayStation 2, Nintendo 64 (descontinuado), PlayStation Portable Consolas e gateways residenciais como a Linksys WRT54G série.
- IBM ea Freescale (antiga Motorola SPS) Power Architecture, usado em todos os supercomputadores da IBM, servidores midrange e estações de trabalho, em computadores baseados em PowerPC da Apple Macintosh (descontinuado), no Nintendo Gamecube e Wii, da Microsoft, Xbox 360 e PlayStation 3 consoles de videogame , faixa da EMC DMX da SAN Symmetrix, e em muitos aplicativos embutidos, tais como impressoras e carros.
- SPARC, da Oracle (anteriormente Sun Microsystems) e Fujitsu
- Hewlett-Packard-RISC, também conhecido como HP-PA, interrompido 31 de dezembro de 2008.
- Alpha, utilizados em computadores de placa única, workstations, servidores e supercomputadores da Digital Equipment Corporation, Compaq e HP, suspenso em 2007.
- XAP processador usado em fios de baixa potência muitos (Bluetooth, wi-fi) chips da RSE.
- SuperH Hitachi, originalmente em amplo uso na Super Sega 32X, Saturn e Dreamcast, agora no coração de muitos dispositivos eletrônicos de consumo. O SuperH é a plataforma base para a Mitsubishi - Hitachi grupo de semicondutores comuns. Os dois grupos se fundiram em 2002, caindo arquitetura RISC própria Mitsubishi, o M32R.

- Atmel AVR usado em uma variedade de produtos, incluindo desde os controladores de Xbox portátil para carros BMW.

Ligações externas

- RISC vs. CISC [1]
- What is RISC [2]
- RISC vs. CISC from historical perspective [3]

Referências

[1] http://cse.stanford.edu/class/sophomore-college/projects-00/risc/risccisc/
[2] http://cse.stanford.edu/class/sophomore-college/projects-00/risc/whatis/index.html
[3] http://www.cpushack.net/CPU/cpuAppendA.html

Fontes e Editores da Página

RISC *Fonte*: http://pt.wikipedia.org/w/index.php?oldid=25293370 *Contribuidores*: !Silent, Carlos28, Dbarbosa², FML, Francisco Leandro, Johnnypersebs, Leonardo.stabile, Leszek Jańczuk, Luizfduartejr, Marudiniz, Mateusc, Mca.leite, Mecanismo, Nossedotti, Nuno-rafael, Rautopia, Rhe, X spager, Yurik, 28 edições anónimas

Licença