[1 Introdução da Tecnologia RISC 2](#_Toc443214705)

[1.2 História 2](#_Toc443214706)

[1.2.1 Características da Arquitetura RISC 4](#_Toc443214707)

[1.2.2 Risc Inicial 5](#_Toc443214708)

[1.2.3 Risc Posterior 7](#_Toc443214709)

[2 Introdução da Arquitetura CISC 8](#_Toc443214710)

[2.1 Características da Arquitetura CISC 9](#_Toc443214711)

[3 Arquitetura X86 10](#_Toc443214712)

[3.1 Propriedades Básicas Da Arquitetura 11](#_Toc443214713)

[3.1.2 As implementações Atuais 12](#_Toc443214714)

[3.1.3 Registradores X86 13](#_Toc443214715)

[4 Arquitetura RISC x Arquitetura CISC 15](#_Toc443214716)

# 1 Introdução da Tecnologia RISC

RISC (acrônimo de Reduced Instruction Set Computer, em português, “Computador com um conjunto reduzido de instruções”) 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. Ela é constituída por um pequeno conjunto de instruções simples que são executadas diretamente pelo hardware,sem a intervenção de um interpretador(microcódigo),ou seja ,as instruções são executadas em apenas uma microinstrução.As máquinas RISC só se tornaram viáveis devido aos avanços de software no aparecimento de compiladores otimizados.Muitos dos microprocessadores modernos são RISCs,por exemplo DEC Alpha, SPARC,MIPS, e PowerPC.O tipo de microprocessador mais comum em desktops , o x86,é mais semelhante ao CISC do que ao RISC,embora chips mais novos traduzam instruções x86 baseadas em arquiteturas CISC em formas baseadas em arquitetura RISC mais simples,utilizando prioridade de execução.Risc é também a arquitetura adotada para os processadores dos videogames modernos,que proporcionam um hardware extremamente dedicado somente á execução do jogo,tornando-o muito mais rápido em relação a micro computadores com mais recursos,embora com processador x86.

## 1.2 História

Desde os primeiros momentos da indústria de computadores 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 frequentemente incapazes de tirar proveito dos recursos destinados a facilitar a montagem manual de codificação, e que os modos de endereçamento complexos levavam muitos ciclos para executar devido aos exigidos acessos à memória adicional. Foi alegado que tais funções seriam melhor executadas por sequências 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, as maiorias 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 pipelines RISC significativamente mais eficientes e permitindo que as frequências de clock fossem mais altas. No início da [década de 1980](https://pt.wikipedia.org/wiki/D%C3%A9cada_de_1980), surgiram várias 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.

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](https://pt.wikipedia.org/wiki/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](https://pt.wikipedia.org/wiki/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, levará 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.

## 1.2.1 Características da Arquitetura RISC

Existe um conjunto de características que permite uma definição de arquitetura básica RISC: são elas:

**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 até 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, consequentemente, o espaço 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](https://pt.wikipedia.org/wiki/Pipeline_(hardware)), 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

## 1.2.2 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”. 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 registro 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 registros 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.

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.

## 1.2.3 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, multiprocessador, 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](https://pt.wikipedia.org/wiki/D%C3%A9cada_de_1990) (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.

# 2 Introdução da Arquitetura CISC

**CISC** (sigla para **Complex Instruction Set Computer**, ou, em uma tradução literal, "Computador com um Conjunto Complexo de Instruções"): é uma linha de arquitetura de processadores capaz de executar centenas de instruções complexas diferentes sendo, assim, extremamente versátil. Exemplos de processadores CISC são os [386](https://pt.wikipedia.org/wiki/Intel_80386) e os [486](https://pt.wikipedia.org/wiki/Intel_80486) da [Intel](https://pt.wikipedia.org/wiki/Intel).

Os processadores baseados na computação de conjunto de instruções complexas contêm uma [micro-programação](https://pt.wikipedia.org/w/index.php?title=Micro-programa%C3%A7%C3%A3o&action=edit&redlink=1), ou seja, um conjunto de códigos de instruções que são gravados no processador, permitindo-lhe receber as instruções dos programas e executá-las, utilizando as instruções contidas na sua micro-programação. Seria como quebrar estas instruções, já em baixo nível, em diversas instruções mais próximas do hardware (as instruções contidas no [microcódigo](https://pt.wikipedia.org/w/index.php?title=Microc%C3%B3digo&action=edit&redlink=1) do processador). Como característica marcante esta arquitetura contém um conjunto grande de instruções, a maioria deles em um elevado grau de complexidade.

Examinando do ponto de vista um pouco mais prático, a vantagem da arquitetura CISC é que já temos muitas das instruções guardadas no próprio processador, o que facilita o trabalho dos [programadores](https://pt.wikipedia.org/wiki/Programador) de [linguagem de máquina](https://pt.wikipedia.org/wiki/Linguagem_de_m%C3%A1quina); disponibilizando, assim, praticamente todas as instruções que serão usadas em seus programas. Os processadores CISC têm a vantagem de reduzir o tamanho do [código executável](https://pt.wikipedia.org/wiki/C%C3%B3digo_execut%C3%A1vel) por já possuírem muito do código comum em vários programas, em forma de uma única instrução.

Porém, do ponto de vista da performance, os CISCs têm algumas desvantagens em relação aos [RISCs](https://pt.wikipedia.org/wiki/RISC), entre elas a impossibilidade de se alterar alguma instrução *composta* para se melhorar a performance. O código equivalente às instruções *compostas* do CISC pode ser escrito nos RISCs da forma desejada, usando um conjunto de instruções simples, da maneira que mais se adequar. Sendo assim, existe uma disputa entre **tamanho do código X desempenho**.

## 2.1 Características da Arquitetura CISC

Existe um conjunto de características que permite uma definição de arquitetura básica CISC: são elas:

**Diferenças de velocidade entre memória e processador** – no final da [década de 1970](https://pt.wikipedia.org/wiki/D%C3%A9cada_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 máquina para executar tais operações, podendo-se acreditar que esse foi o início do aumento da quantidade de instruções no CISC.

**Emprego de microcódigo** – o surgimento e a real vantagem de custo/beneficio do emprego de microcódigo 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](https://pt.wikipedia.org/wiki/D%C3%A9cada_de_1980), havia um crescimento acelerado do emprego de linguagens de alto 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 alto 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](https://pt.wikipedia.org/wiki/486) veio com apenas algumas instruções novas e todo o código do [386](https://pt.wikipedia.org/wiki/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](https://pt.wikipedia.org/wiki/Pentium_II), 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.

# 3 Arquitetura X86

A arquitetura x86 é o nome genérico dada á família de processadores baseados no Intel 8086, da Intel Corporation. A arquitetura é chamada x86 porque os primeiros processadores desta família eram identificados somente por números terminados com a sequência "86": o [8086](https://pt.wikipedia.org/wiki/Intel_8086), o [80186](https://pt.wikipedia.org/wiki/Intel_80186), o [80286](https://pt.wikipedia.org/wiki/Intel_80286), o [80386](https://pt.wikipedia.org/wiki/Intel_80386) e o [80486](https://pt.wikipedia.org/wiki/Intel_80486). Como não se pode ter uma marca registrada sobre números, a Intel e a maior parte de seus competidores começaram a usar nomes que pudessem ser registrados como marca, como [Pentium](https://pt.wikipedia.org/wiki/Pentium) para as gerações de [processadores](https://pt.wikipedia.org/wiki/Processadores) posteriores, mas a antiga nomenclatura tinha forjado um termo para toda a [família](https://pt.wikipedia.org/wiki/Fam%C3%ADlia). O termo x86 não é sinônimo de compatibilidade com IBM PC, pois isso sugere uma infinidade de outros hardwares e sistemas embarcados bem como [computadores](https://pt.wikipedia.org/wiki/Computadores) usando chips x86 antes do moderno de mercado ser compatível com PC, alguns deles antes mesmo do IBM PC em si. Houve várias tentativas, dentro da [Intel](https://pt.wikipedia.org/wiki/Intel) em si, para quebrar o domínio do mercado da arquitetura x86, que descendia diretamente do simples [microprocessadores](https://pt.wikipedia.org/wiki/Microprocessadores) de 8 bits. Exemplos disso são as iAPX 432 (aliás, Intel 8800), o Intel 960, Intel 860 e Intel e a arquitetura do Hewlett Packard Itanium. No entanto, o aperfeiçoamento contínuo de 86 micro arquiteturas, circuitos, e fabricação de [semicondutores](https://pt.wikipedia.org/wiki/Semicondutores) provaria que é difícil substituir x86 em vários segmentos. O 64, da extensão bit da AMD x86 (que a Intel finalmente respondeu, com um[design](https://pt.wikipedia.org/wiki/Design) compatível) e a escala de processadores x86, como o eight-core Intel Xeon e 12-core AMD Opteron destaca-se o x86 como um exemplo de contínuo aperfeiçoamento e como as normas estabelecidas pela [indústria](https://pt.wikipedia.org/wiki/Ind%C3%BAstria) pode resistir à concorrência das novas arquiteturas.

## 3.1 Propriedades Básicas Da Arquitetura

A arquitetura x86 é um comprimento variável de instrução, primeiramente são modelados dois endereços “CISC” com [ênfase](https://pt.wikipedia.org/wiki/%C3%8Anfase) em compatibilidade com versões anteriores. No entanto o conjunto de instruções não é típico CISC, mas, basicamente, uma versão estendida e [ortogonal](https://pt.wikipedia.org/wiki/Ortogonal) de simples oito bits, das arquiteturas 8008, 8080 e 8085. O [Byte](https://pt.wikipedia.org/wiki/Byte) de endereçamento é suportado e as palavras são armazenadas na memória com ordem de little-endian byte. O acesso à memória para endereços desalinhados é permitido em todos os tamanhos de palavras suportadas. O maior tamanho nativo para [aritmética](https://pt.wikipedia.org/wiki/Aritm%C3%A9tica) de inteiros e endereços de memória (ou deslocamentos) é de 16, 32 ou 64 bits dependendo da geração da [arquitetura](https://pt.wikipedia.org/wiki/Arquitetura) (processadores mais recentes incluem um suporte direto para inteiros menores também). Vários valores escalares podem ser tratados simultaneamente, através da unidade SIMD presente em gerações posteriores, conforme descrito abaixo. Compensações imediatas de endereçamento e dados de imediato podem ser expressos em quantidades de 8 bits conforme a frequência que ocorrem os casos ou contextos em que o [intervalo](https://pt.wikipedia.org/wiki/Intervalo) de 128. 127 é o suficiente. Instruções típicas são, portanto, de 2 ou 3 [bytes](https://pt.wikipedia.org/wiki/Bytes) de comprimento (embora alguns são muito mais longos, e algumas são de [byte](https://pt.wikipedia.org/wiki/Byte) único). Para melhor conservar o espaço de codificação, a maioria dos registros são expressos em opcodes usando três bits, e no máximo um operando em uma instrução que pode ser um local de memória (alguns projetos "CISC" altamente [ortogonais](https://pt.wikipedia.org/wiki/Ortogonal), como o PDP-11, podem utilizar dois). No entanto, esta [memória](https://pt.wikipedia.org/wiki/Mem%C3%B3ria) operando também pode ser de destinação (ou a combinação entre fonte e destinação), enquanto o outro operando, a fonte, pode ser tanto registrada quanto imediata. O [número](https://pt.wikipedia.org/wiki/N%C3%BAmero) relativamente pequeno de registros gerais (também herdados de seus antecessores de 8 bits) fez-se endereçamento de registro relativo (usando pequenas compensações imediatas) como um importante método de acesso a operandos, especialmente na [pilha](https://pt.wikipedia.org/wiki/Pilha). Muito trabalho foi, portanto, investido na [fabricação](https://pt.wikipedia.org/wiki/Fabrica%C3%A7%C3%A3o) desses acessos de modo tão rápido como registrar os acessos, ou seja, um ciclo de transferência de instrução, na maioria dos casos em que os dados acessados estão disponíveis no [cache](https://pt.wikipedia.org/wiki/Cache) de nível superior.

## 3.1.2 As implementações Atuais

Durante a execução, processadores x86 atuais empregam uma [decodificação](https://pt.wikipedia.org/wiki/Decodifica%C3%A7%C3%A3o) extra para dividir a maioria das instruções em pedaços menores (micro-operações). Estes são então entregues a uma unidade de controle que os tampões e os [cronogramas](https://pt.wikipedia.org/wiki/Cronograma) estão em conformidade de acordo com a x86-semântica, para que possam ser executadas, parcialmente em paralelo, por uma das várias unidades (mais ou menos especializadas) de execução. Quando foi introduzida, esta abordagem foi por vezes referida como um "núcleo RISC" ou como "tradução RISC", em parte por razões de [marketing](https://pt.wikipedia.org/wiki/Marketing), mas também porque essas micro-operações compartilham algumas propriedades com certos tipos de instruções RISC. No entanto, o [microcódigo](https://pt.wikipedia.org/w/index.php?title=Microc%C3%B3digo&action=edit&redlink=1) tradicional (usado desde a [década](https://pt.wikipedia.org/wiki/D%C3%A9cada) de 1950) também inerentemente compartilha muitas das mesmas propriedades; a nova abordagem difere, principalmente na medida em que a tradução para micro-operações agora ocorrem de forma [assíncrona](https://pt.wikipedia.org/wiki/Ass%C3%ADncrona). Não ter que sincronizar as unidades de execução com as etapas de decodificação abre possibilidades para uma análise mais aprofundada do fluxo de código (buffer) e, portanto, permite a detecção de operações que podem ser executadas em paralelo, alimentando simultaneamente mais de uma unidade de execução. Os mais recentes [processadores](https://pt.wikipedia.org/wiki/Processadores) também fazem o oposto quando for o caso, pois eles combinam certas seqüências x86 (como uma comparação seguida de um salto condicional) um processo mais complexo de micro-op, que se encaixa no modelo de execução melhor e, portanto, pode ser executado mais rápido ou com menos recursos da máquina envolvida. Outra forma de tentar melhorar o desempenho é o [cache](https://pt.wikipedia.org/wiki/Cache) decodificado de micro-operações, de modo que o [processador](https://pt.wikipedia.org/wiki/Processador) pode acessar diretamente o [decodificado](https://pt.wikipedia.org/w/index.php?title=Decodificado&action=edit&redlink=1) de micro-operações a partir de um cache especial, em vez de descodificá-lo novamente. A “Execution Trace Cache” é encontrada na microarquitetura da [Intel](https://pt.wikipedia.org/wiki/Intel) NetBurst (Pentium 4) é até agora o único exemplo generalizado desta técnica. Transmeta usa um método completamente diferente em seus [CPUs](https://pt.wikipedia.org/wiki/CPU) x86 compatíveis. Eles usam tradução para converter as instruções x86 para instruções nativas do [CPU](https://pt.wikipedia.org/wiki/CPU). Transmeta argumenta que sua abordagem permite projetos de energia mais eficientes, uma vez que a CPU pode abandonar a etapa de [decodificação](https://pt.wikipedia.org/wiki/Decodifica%C3%A7%C3%A3o) complicada para implementações x86 tradicionais.

## 3.1.3 Registradores X86

**Registro de 16 bits** O original Intel 8086 e 8088 tem quatorze registros de 16 bits. Quatro deles (AX, BX, CX, DX) são registradores de propósito geral (GPRS, embora cada um possa ter uma finalidade adicional: por exemplo, apenas CX pode ser usado como um [contador](https://pt.wikipedia.org/wiki/Contador) com a instrução de [loop](https://pt.wikipedia.org/wiki/Loop)). Cada um pode ser acessado como dois [bytes](https://pt.wikipedia.org/wiki/Bytes) separados (daí o byte mais alto BX pode ser acessado como BH e byte baixo como BL). Há dois registros de ponteiro: SP, que aponta para o topo da [pilha](https://pt.wikipedia.org/wiki/Pilha) e da BP (base pointer) que é usado para apontar para algum outro lugar na [pilha](https://pt.wikipedia.org/wiki/Pilha), geralmente acima de variáveis locais. Dois registros (SI e DI) são para a indexação de matriz.

Quatro registradores de segmento (CS, DS, SS e ES) são usados para formar um endereço de memória. O registrador de flags contém sinalizadores como de proceder com tais flags, overflow de flag e underflow de flag. Finalmente, o ponteiro de instrução (IP) aponta para a próxima instrução que será obtida a partir da memória e depois executados.

Na Intel 80286, três registros especiais contém endereços da tabela descrita (GDTR, LDTR, IDTR), e um registro de quarta tarefa (TR).

**Registro 32 bits** Com o advento do [processador](https://pt.wikipedia.org/wiki/Processador) de 32 bits 80386, os registradores de 16 bits, registradores base, registradores de índice, ponteiro de instrução, e registradores de flag, mas não os registradores de segmento, foram ampliados para 32 bits. Este é representado por um [prefixo](https://pt.wikipedia.org/wiki/Prefixo) "E" (para Extended) para registrar os nomes em [linguagem assembly](https://pt.wikipedia.org/wiki/Assembly) x86. Assim, o registro AX diferencia o registrador de 16 bits do novo EAX 32-bit, SI corresponde ao menor de 16 bits do ESI, e assim por diante. Os [registradores](https://pt.wikipedia.org/wiki/Registradores) de uso geral, registradores de base e registradores de índice podem ser usados como base em modos de endereçamento, e todos os registros, exceto o ponteiro de [pilha](https://pt.wikipedia.org/wiki/Pilha) pode ser utilizado como [índice](https://pt.wikipedia.org/wiki/%C3%8Dndice) de endereçamento.

Dois novos registros de segmento (FS e GS) foram adicionados. Com um maior número de registros, instruções e operandos, o formato do binario|código de [máquina](https://pt.wikipedia.org/wiki/M%C3%A1quina) foi alargado. Para fornecer compatibilidade com versões anteriores, os [segmentos](https://pt.wikipedia.org/wiki/Segmentos) com código executável pode ser marcado como contendo ou 16 bits ou 32 bits de instruções. Prefixos especiais permitem a inclusão de instruções de 32 bits em um segmento de 16 bits, ou vice-versa.

Com o 80486 uma unidade de processamento de ponto flutuante (FPU) foi adicionada, com oito registros de 80 bits de largura.

Com o Pentium II, oito [registradores](https://pt.wikipedia.org/wiki/Registradores) de 64 bits MMX foram adicionados. Com o Pentium III, um de 32 bits Streaming SIMD Extension (SSE) de controle / registrador de status (mxcsr) e oito de 128 bits SSE [registradores](https://pt.wikipedia.org/wiki/Registradores) de ponto flutuante (XMM0. XMM7) foram adicionados [17]. Com o Pentium III, um Streaming SIMD Extension (SSE) de 32 bits, um registrador de controle (MXCSR) e oito registradores SSE de pontos flutuantes de 128 bits (XMM0. XMM7) foram adicionados.

**Registro 64 bits** Com o [processador](https://pt.wikipedia.org/wiki/Processador) AMD Opteron, a [arquitetura](https://pt.wikipedia.org/wiki/Arquitetura) x86 estendeu os registradores de 32 bits para registradores de 64 bits de uma forma similar como a extensão de 16 bits para 32 bits foi feita, foram adicionadas oito registradores de 64 bits (R8, R9 R15.), entre outras. No entanto, essas extensões são apenas utilizáveis em modo de 64 bits. Os modos de endereçamento não mudaram drasticamente a partir do modo de 32 bits, com exceção que a [resolução](https://pt.wikipedia.org/wiki/Resolu%C3%A7%C3%A3o) foi estendida para 64 bits, endereços [virtuais](https://pt.wikipedia.org/wiki/Virtual) foram estendidos para 64 bits, a fim de impedir endereços [virtuais](https://pt.wikipedia.org/wiki/Virtual) que se têm revelado problemáticos em outras [arquiteturas](https://pt.wikipedia.org/wiki/Arquitetura), e mais alguns detalhes foram drasticamente reduzidos.

# 4 Arquitetura RISC x Arquitetura 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 resultarão 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 à 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. Processadores RISC são também otimizados para operações de uma única tarefa devido ao grande número 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.