

1

A civilização avança estendendo o número de operações importantes que podem ser realizadas sem se pensar nelas.

Alfred North Whitehead
Uma Introdução à Matemática, 1911

Abstrações e Tecnologias Computacionais

- 1.1** **Introdução** 1
- 1.2** **Por trás do programa** 6
- 1.3** **Sob as tampas** 9
- 1.4** **Desempenho** 19
- 1.5** **A barreira da potência** 29
- 1.6** **Mudança de mares: Passando de processadores para multiprocessadores** 31

1.7	Vida real: Fabricação e benchmarking do AMD Opteron X4	34
1.8	Falácia e armadilhas	39
1.9	Comentários finais	41
1.10	Perspectiva histórica e leitura adicional	43
1.11	Exercícios	43

1.1

Introdução

Bem-vindo a este livro! Estamos felizes por ter a oportunidade de compartilhar o entusiasmo do mundo dos sistemas computacionais. Esse não é um campo árido e monótono, no qual o progresso é glacial e as novas ideias se atrofiam pelo esquecimento. Não! Os computadores são o produto da impressionante e vibrante indústria da tecnologia da informação, cujos aspectos são responsáveis por quase 10% do produto interno bruto dos Estados Unidos. Essa área incomum abraça a inovação com uma velocidade surpreendente. Nos últimos 25 anos, surgiram inúmeros novos computadores que prometiam revolucionar a indústria da computação; essas revoluções foram interrompidas porque alguém sempre construía um computador ainda melhor.

Essa corrida para inovar levou a um progresso sem precedentes desde o início da computação eletrônica no final da década de 1940. Se o setor de transportes, por exemplo, tivesse tido o mesmo desenvolvimento da indústria da computação, hoje nós poderíamos viajar de Nova York até Londres em aproximadamente um segundo por apenas alguns centavos. Imagine por alguns instantes como esse progresso mudaria a sociedade – morar em Taiti e trabalhar em São Francisco, indo para Moscou no início da noite a fim de assistir a uma apresentação do balé de Bolshoi. Não é difícil imaginar as implicações dessa mudança.

Os computadores levaram a humanidade a enfrentar uma terceira revolução, a revolução da informação, que assumiu seu lugar junto das revoluções industrial e agrícola. A multiplicação da força e do alcance intelectual do ser humano naturalmente afetou muito nossas vidas cotidianas, além de ter mudado a maneira como conduzimos a busca de novos conhecimentos. Agora, existe um novo veio de investigação científica, com a ciência da computação unindo os cientistas teóricos e experimentais na exploração de novas fronteiras na astronomia, biologia, química, física etc.

A revolução dos computadores continua. Cada vez que o custo da computação melhora por um fator de 10, as oportunidades para os computadores se multiplicam. As aplicações que eram economicamente proibitivas de repente se tornam viáveis. As seguintes aplicações, no passado recente, eram “ficção científica para a computação”:

- Computação em automóveis: até os microprocessadores melhorarem significativamente de preço e desempenho no início dos anos 80, o controle dos carros por computadores era considerado um absurdo. Hoje, os computadores reduzem a poluição e melhoram a eficiência do combustível, usando controles no motor, além de aumentarem a segurança por meio da prevenção de derrapagens perigosas e pela ativação de air-bags para proteger os passageiros em caso de colisão.

- Telefones celulares: quem sonharia que os avanços dos sistemas computacionais levariam aos telefones portáteis, permitindo a comunicação pessoa a pessoa em quase todo lugar do mundo?
- Projeto do genoma humano: o custo do equipamento computacional para mapear e analisar as sequências do DNA humano é de centenas de milhares de dólares. É improvável que alguém teria considerado esse projeto se os custos computacionais fossem 10 a 100 vezes mais altos, como há dez ou 20 anos. Além do mais, os custos continuam a cair; você poderá adquirir seu próprio genoma, permitindo que a assistência médica seja ajustada a você mesmo.
- World Wide Web: ainda não existente na época da primeira edição deste livro, a World Wide Web transformou nossa sociedade. Para muitos, a Web substituiu as bibliotecas.
- Máquinas de busca: à medida que o conteúdo da Web crescia em tamanho e em valor, encontrar informações relevantes tornou-se cada vez mais importante. Hoje, muitas pessoas contam com máquinas de busca para tantas coisas em suas vidas que seria muito difícil viver sem elas.

Claramente, os avanços dessa tecnologia hoje afetam quase todos os aspectos da nossa sociedade. Os avanços de hardware permitiram que os programadores criassem softwares maravilhosamente úteis e explicassem por que os computadores são onipresentes. A ficção científica de hoje sugere as aplicações que fazem sucesso amanhã: já a caminho estão os mundos virtuais, reconhecimento de voz prático e assistência médica personalizada.

Classes de aplicações de computador e suas características

Embora um conjunto comum de tecnologias de hardware (discutidas nas Seções 1.3 e 1.7) seja usado em computadores variando dos dispositivos domésticos inteligentes e telefones celulares aos maiores supercomputadores, essas diferentes aplicações possuem diferentes necessidades de projeto e empregam os fundamentos das tecnologias de hardware de diversas maneiras. Genericamente falando, os computadores são usados em três diferentes classes de aplicações.

Os computadores desktop são possivelmente a forma mais conhecida de computação e caracterizam-se pelo computador pessoal, que a maioria dos leitores deste livro provavelmente já usou extensivamente. Os computadores desktop enfatizam o bom desempenho a um único usuário por um baixo custo e normalmente são usados para executar software independente. A evolução de muitas tecnologias de computação é motivada por essa classe da computação, que só tem cerca de 30 anos!

Os servidores são a forma moderna do que, antes, eram os mainframes, minicomputadores e supercomputadores, e, em geral, são acessados apenas por meio de uma rede. Os servidores são projetados para suportar grandes cargas de trabalho, que podem consistir em uma única aplicação complexa, normalmente científica ou de engenharia, ou manipular muitas tarefas pequenas, como ocorreria no caso de um grande servidor Web. Essas aplicações muitas vezes são baseadas em software de outra origem (como um banco de dados ou sistema de simulação), mas frequentemente são modificadas ou personalizadas para uma função específica. Os servidores são construídos a partir da mesma tecnologia básica dos computadores desktop, mas fornecem uma maior capacidade de expansão tanto da capacidade de processamento quanto de entrada/saída. Em geral, os servidores também dão grande ênfase à estabilidade, já que uma falha normalmente é mais prejudicial do que seria em um computador desktop de um único usuário.

Os servidores abrangem a faixa mais ampla em termos de custo e capacidade. Na sua forma mais simples, um servidor pode ser pouco mais do que uma máquina desktop sem monitor ou teclado e com um custo de mil dólares. Esses servidores de baixa capacidade normalmente são usados para armazenamento de arquivos, pequenas aplicações comerciais

computadores desktop Um computador projetado para uso por uma única pessoa, normalmente incorporando um monitor gráfico, um teclado e um mouse.

servidor Um computador usado para executar grandes programas para múltiplos usuários quase sempre de maneira simultânea e normalmente acessado apenas por meio de uma rede.

ou serviço Web simples (veja Seção 6.10). No outro extremo, estão os supercomputadores, que, atualmente, consistem em centenas ou milhares de processadores e, em geral, de terabytes de memória e de petabytes de armazenamento, e custam desde milhões até centenas de milhões de dólares. Os supercomputadores normalmente são usados para cálculos científicos e de engenharia de alta capacidade, como previsão do tempo, exploração de petróleo, determinação da estrutura da proteína e outros problemas de grande porte. Embora esses supercomputadores representem o máximo da capacidade de computação, eles são uma fração relativamente pequena dos servidores e do mercado de computadores em termos de receita total.

Os computadores embutidos são a maior classe de computadores e abrangem a faixa mais ampla de aplicações e desempenho. Os computadores embutidos incluem os microprocessadores encontrados em seu carro, os computadores em um telefone celular, os computadores em um *video game* ou televisão digital, e as redes de processadores que controlam um avião moderno ou um navio de carga. Os sistemas de computação embutidos são projetados para executar uma aplicação ou um conjunto de aplicações relacionadas como um único sistema; portanto, apesar do grande número de computadores embutidos, a maioria dos usuários nunca vê realmente que está usando um computador!

A Figura 1.1 mostra que, durante os últimos anos, o crescimento em telefones celulares que contam com computadores embutidos foi muito mais rápido do que a taxa de crescimento dos computadores desktop. Observe que os computadores embutidos também são encontrados em TVs digitais e sintonizadores, automóveis, câmeras digitais, players de música, *video games* e uma série de outros dispositivos do consumidor, o que aumenta ainda mais a lacuna entre o número de computadores embutidos e os computadores de desktop.

As aplicações embutidas normalmente possuem necessidades específicas que combinam um desempenho mínimo com limitações rígidas em relação a custo ou potência. Por exemplo, considere um telefone celular: o processador só precisa ser tão rápido quanto o necessário para manipular sua função limitada; além disso, minimizar custo e potência é o objetivo mais importante. Apesar do seu baixo custo, os computadores embutidos frequentemente possuem menor tolerância a falhas, já que os resultados podem variar desde um simples incômodo, quando sua nova televisão falha, até a completa devastação que poderia ocorrer quando o computador em um avião ou em um navio falha. Nas aplicações

supercomputador Uma classe de computadores com desempenho e custo mais altos; eles são configurados como servidores e normalmente custam milhões de dólares.

terabyte Originalmente, 1.099.511.627.776 (240) bytes, embora alguns sistemas de comunicações e de armazenamento secundário o tenham redefinido como significando 1.000.000.000.000 (1012) bytes.

petabyte Dependendo da situação, 1000 ou 1024 terabytes.

centro de dados Uma sala ou prédio criado para tratar das necessidades de energia, resfriamento e rede de um grande número de servidores.

computador embutido Um computador dentro de outro dispositivo, usado para executar uma aplicação predeterminada ou uma coleção de software.

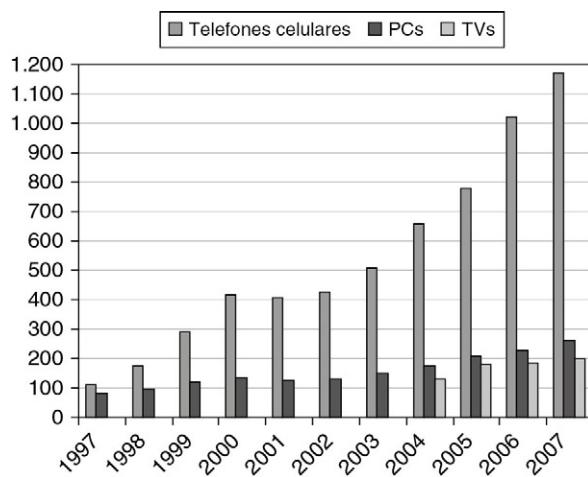


FIGURA 1.1 O número de telefones celulares, computadores pessoais e televisores fabricados por ano entre 1997 e 2007. (Temos dados de televisão somente a partir de 2004.) Mais de um bilhão de novos telefones celulares foram entregues em 2006. As vendas de telefones celulares ultrapassaram os PCs por um fator apenas de 1,4 em 1997, mas a taxa cresceu para 4,5 em 2007. O número total em uso em 2004 é estimado como sendo 2,0B de televisores, 1,8B de telefones celulares e 0,8B de PCs. Como a população do mundo era cerca de 6,4B em 2004, havia aproximadamente um PC, 2,2 telefones celulares e 2,5 televisores para cada oito pessoas no planeta. Um estudo de 2006 das famílias dos Estados Unidos descobriu que eles possuíam, na média, 12 aparelhos, incluindo 3 TVs, dois PCs e outros dispositivos, como consoles de jogos, tocadores de MP3 e telefones celulares.

embutidas orientadas ao consumidor, como um eletrodoméstico digital, a estabilidade é obtida principalmente por meio da simplicidade – a ênfase está em realizar uma função o mais perfeitamente possível. Nos grandes sistemas embutidos, em geral, são empregadas as mesmas técnicas de redundância utilizadas nos servidores (veja Seção 6.9). Embora este livro se concentre nos computadores de uso geral, a maioria dos conceitos se aplica diretamente – ou com ligeiras modificações – aos computadores embutidos.

Detalhamento: os detalhamentos são seções curtas usadas em todo o texto para fornecer mais detalhes sobre um determinado assunto, que pode ser de interesse. Os leitores que não possuem um interesse específico no tema podem pular essas seções, já que o material subsequente nunca dependerá do conteúdo desta seção.

Muitos processadores embutidos são projetados usando núcleos de processador, uma versão de um processador escrita em uma linguagem de descrição de hardware como Verilog ou VHDL (veja Capítulo 4). O núcleo permite que um projetista integre outro hardware específico de uma aplicação com o núcleo do processador para a fabricação de um único chip.

O que você pode aprender neste livro

Os bons programadores sempre se preocuparam com o desempenho de seus programas porque gerar resultados rapidamente para o usuário é uma condição essencial na criação bem-sucedida de software. Nas décadas de 1960 e 1970, uma grande limitação no desempenho dos computadores era o tamanho da memória do computador. Assim, os programadores em geral seguiam um princípio simples: minimizar o espaço ocupado na memória para tornar os programas mais rápidos. Na última década, os avanços em arquitetura de computadores e nas tecnologias de fabricação de memórias reduziram drasticamente a importância do tamanho da memória na maioria das aplicações, com exceção dos sistemas embutidos.

Agora, os programadores interessados em desempenho precisam entender os problemas que substituíram o modelo de memória simples dos anos 60: a natureza paralela dos processadores e a natureza hierárquica das memórias. Os programadores que desejam construir versões competitivas de compiladores, sistemas operacionais, bancos de dados e mesmo aplicações precisarão, portanto, aumentar seu conhecimento em organização de computadores.

Sentimo-nos honrados com a oportunidade de explicar o que existe dentro da máquina revolucionária, decifrando o software por trás do seu programa e o hardware sob a tampa do seu computador. Ao concluir este livro, acreditamos que você será capaz de responder às seguintes perguntas:

- Como os programas escritos em uma linguagem de alto nível, como C ou Java, são traduzidos para a linguagem de máquina e como o hardware executa os programas resultantes? Compreender esses conceitos forma o alicerce para entender os aspectos do hardware e software que afetam o desempenho dos programas.
- O que é a interface entre o software e o hardware, e como o software instrui o hardware a realizar as funções necessárias? Esses conceitos são vitais para entender como escrever muitos tipos de software.
- O que determina o desempenho de um programa e como um programador pode melhorar o desempenho? Como veremos, isso depende do programa original, da tradução desse programa para a linguagem do computador e da eficiência do hardware em executar o programa.
- Que técnicas podem ser usadas pelos projetistas de hardware para melhorar o desempenho? Este livro apresentará os conceitos básicos do projeto de computador moderno. O leitor interessado encontrará muito mais material sobre esse assunto em nosso livro avançado, Arquitetura de Computadores: Uma abordagem quantitativa.

- Quais são os motivos e as consequências da mudança recente do processamento sequencial para o processamento paralelo? Este livro oferece a motivação, descreve os mecanismos de hardware atuais para dar suporte ao paralelismo e estuda a nova geração de microprocessadores “multicore” (veja Capítulo 7).

Sem entender as respostas a essas perguntas, melhorar o desempenho do seu programa em um computador moderno ou avaliar que recursos podem tornar um computador melhor do que outro para uma determinada aplicação será um complicado processo de tentativa e erro, em vez de um procedimento científico conduzido por consciência e análise.

Este primeiro capítulo é a base para o restante do livro. Ele apresenta as ideias e definições básicas, coloca os principais componentes de software e hardware em perspectiva, mostra como avaliar o desempenho e a potência, apresenta os circuitos integrados, a tecnologia que estimula a revolução dos computadores, e explica a mudança para multicores.

Neste capítulo e em capítulos seguintes, você provavelmente verá muitas palavras novas ou palavras que já pode ter ouvido, mas não sabe ao certo o que significam. Não entre em pânico! Sim, há muita terminologia especial usada para descrever os computadores modernos, mas ela realmente ajuda, uma vez que nos permite descrever precisamente uma função ou capacidade. Além disso, os projetistas de computador (inclusive estes autores) adoram usar acrônimos, que são fáceis de entender quando se sabe o que as letras significam! Para ajudá-lo a lembrar e localizar termos, incluímos uma definição destacada de cada termo novo na primeira vez que aparece no texto. Após um pequeno período trabalhando com a terminologia, você estará fluente e seus amigos ficarão impressionados quando você usar corretamente palavras como BIOS, CPU, DIMM, DRAM, PCIE, SATA e muitas outras.

Para enfatizar como os sistemas de software e hardware usados para executar um programa irão afetar o desempenho, usamos uma seção especial, “Entendendo o desempenho dos programas”, em todo o livro; a primeira aparece a seguir. Esses elementos resumem importantes conceitos quanto ao desempenho do programa.

microprocessador multicore

Um microprocessador contendo múltiplos processadores (“cores” ou núcleos) em um único circuito integrado.

Acrônimo Uma palavra construída tomando-se as letras iniciais das palavras. Por exemplo: RAM é um acrônimo para Random Access Memory (memória de acesso aleatório) e CPU é um acrônimo para Central Processing Unit (unidade central de processamento).

Entendendo o desempenho dos programas

O desempenho de um programa depende de uma combinação entre a eficácia dos algoritmos usados no programa, os sistemas de software usados para criar e traduzir o programa para instruções de máquina e da eficácia do computador em executar essas instruções, que podem incluir operações de entrada/saída (E/S). A tabela a seguir descreve como o hardware e o software afetam o desempenho.

Componente de hardware ou software	Como este componente afeta o desempenho	Onde este assunto é abordado?
Algoritmo	Determina o número de instruções do código-fonte e o número de operações de E/S realizadas	Outros livros!
Linguagem de programação, compilador e arquitetura	Determina o número de instruções de máquina para cada instrução em nível de fonte	Capítulos 2 e 3
Processador e sistema de memória	Determina a velocidade em que as instruções podem ser executadas	Capítulos 4, 5 e 7
Sistema de E/S (hardware e sistema operacional)	Determina a velocidade em que as operações de E/S podem ser executadas	Capítulo 6

As Seções “Verifique você mesmo” se destinam a ajudar os leitores a avaliar se compreenderam os principais conceitos apresentados em um capítulo e se entenderam as implicações desses conceitos. Algumas questões “Verifique você mesmo” possuem respostas simples; outras são para discussão em grupo. As respostas às questões específicas

Verifique você mesmo

podem ser encontradas no final do capítulo. As questões “Verifique você mesmo” aparecem apenas no final de uma seção, fazendo com que fique mais fácil pulá-las se você estiver certo de que entendeu o assunto.

1. A [Seção 1.1](#) mostrou que o número de processadores embutidos vendidos a cada ano supera, e muito, o número de processadores para desktops. Você pode confirmar ou negar isso com base em sua própria experiência? Tente contar o número de processadores embutidos na sua casa. Compare esse número com o número de computadores desktop em sua casa.
2. Como mencionado anteriormente, tanto o software quanto o hardware afetam o desempenho de um programa. Você pode pensar em exemplos em que cada um dos fatores a seguir é o responsável pelo gargalo no desempenho?
 - O algoritmo escolhido
 - A linguagem de programação ou compilador
 - O sistema operacional
 - O processador
 - O sistema de E/S e os dispositivos

Em Paris, eles simplesmente olhavam perdidos quando eu falava em francês; nunca consegui fazer aqueles idiotas entenderem sua própria língua.

Mark Twain, *The Innocents Abroad*, 1869

software de sistemas Software que fornece serviços que normalmente são úteis, incluindo sistemas operacionais, compiladores e montadores.

1.2

Por trás do programa

Uma aplicação típica, como um processador de textos ou um grande sistema de banco de dados, pode consistir em milhões de linhas de código e se basear em bibliotecas de software sofisticadas que implementam funções complexas no apoio à aplicação. Como veremos, o hardware em um computador só pode executar instruções de baixo nível extremamente simples. Ir de uma aplicação complexa até as instruções simples envolve várias camadas de software que interpretam ou traduzem operações de alto nível nas instruções simples do computador.

A [Figura 1.2](#) mostra que essas camadas de software são organizadas principalmente de maneira hierárquica, na qual as aplicações são o anel mais externo e uma variedade de software de sistemas situa-se entre o hardware e as aplicações.

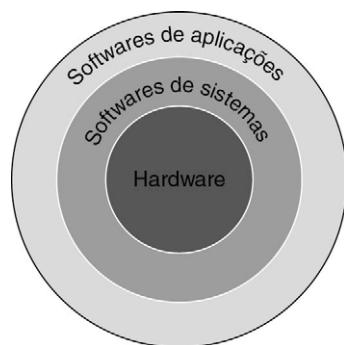


FIGURA 1.2 Uma visão simplificada do hardware e software como camadas hierárquicas, mostradas como círculos concêntricos, em que o hardware está no centro e as aplicações aparecem externamente. Nas aplicações complexas, muitas vezes existem múltiplas camadas de software de aplicação. Por exemplo, um sistema de banco de dados pode ser executado sobre o software de sistemas hospedando uma aplicação, que, por sua vez, executa sobre o banco de dados.

Existem muitos tipos de software de sistemas, mas dois tipos são fundamentais em todos os sistemas computacionais modernos: um sistema operacional e um compilador. Um sistema operacional fornece a interface entre o programa de usuário e o hardware e disponibiliza diversos serviços e funções de supervisão. Entre as funções mais importantes estão:

- Manipular as operações básicas de entrada e saída
- Alocar armazenamento e memória
- Possibilitar e controlar o compartilhamento do computador entre as diversas aplicações que o utilizam simultaneamente

Exemplos de sistemas operacionais em uso hoje são Linux, MacOS e Windows.

Os compiladores realizam outra função fundamental: a tradução de um programa escrito em uma linguagem de alto nível, como C, C++, Java ou Visual Basic, em instruções que o hardware possa executar. Em razão da sofisticação das linguagens de programação modernas e das instruções simples executadas pelo hardware, a tradução de um programa de linguagem de alto nível para instruções de hardware é complexa. Voltaremos a um breve resumo do processo aqui e depois entraremos em mais detalhes no Capítulo 2 e no Apêndice B.

Sistema operacional Programa de supervisão que gerencia os recursos de um computador em favor dos programas executados nessa máquina.

compilador Um programa que traduz as instruções de linguagem de alto nível para instruções de linguagem assembly.

De uma linguagem de alto nível para a linguagem do hardware

Para poder falar com uma máquina eletrônica, você precisa enviar sinais elétricos. Os sinais mais fáceis de serem entendidos pelas máquinas são ligado e desligado; portanto, o alfabeto da máquina se resume a apenas duas letras. Assim como as 26 letras do alfabeto português não limitam o quanto pode ser escrito, as duas letras do alfabeto do computador não limitam o que os computadores podem fazer. Os dois símbolos para essas duas letras são os números 0 e 1, e normalmente pensamos na linguagem de máquina como números na base 2, ou números binários. Chamamos cada “letra” de um dígito binário ou bit. Os computadores são escravos dos nossos comandos, chamados de instruções. As instruções, que são apenas grupos de bits que o computador entende e obedece, podem ser imaginadas como números. Por exemplo, os bits

1000110010100000

dizem ao computador para somar dois números. O Capítulo 2 explica por que usamos números para instruções e dados; não queremos roubar o brilho desse capítulo, mas usar números para instruções e dados é um dos conceitos básicos da computação.

Os primeiros programadores se comunicavam com os computadores em números binários, mas isso era tão maçante que rapidamente inventaram novas notações mais parecidas com a maneira como os humanos pensam. No início, essas notações eram traduzidas para binário manualmente, mas esse processo ainda era cansativo. Usando a própria máquina para ajudar a programá-la, os pioneiros inventaram programas que traduzem da notação simbólica para binário. O primeiro desses programas foi chamado de montador (assembler). Esse programa traduz uma versão simbólica de uma instrução para uma versão binária. Por exemplo, o programador escreveria

add A, B

e o montador traduziria essa notação como

1000110010100000

dígito binário

Também chamado bit. Um dos dois números na base 2 (0 ou 1) que são os componentes da informação.

Instrução

Um comando que o hardware do computador entende e obedece.

montador (assembler)

Um programa que traduz uma versão simbólica de instruções para a versão binária.

linguagem assembly

Uma representação simbólica das instruções de máquina.

linguagem de máquina

Uma representação binária das instruções de máquina.

linguagem de programação

de alto nível Uma linguagem, como C, C++, Java ou Visual Basic, composta de palavras e notação algébrica, que pode ser traduzida por um compilador para a linguagem assembly.

Essa instrução diz ao computador para somar dois números, A e B. O nome criado para essa linguagem simbólica, ainda em uso hoje, é linguagem assembly. Ao contrário, a linguagem binária que a máquina entende é a linguagem de máquina.

Embora seja um fantástico avanço, a linguagem assembly ainda está longe da notação que um cientista poderia desejar usar para simular fluxos de fluidos ou que um contador poderia usar para calcular seus saldos de contas. A linguagem assembly requer que o programador escreva uma linha para cada instrução que a máquina seguirá, obrigando o programador a pensar como a máquina.

A descoberta de que um programa poderia ser escrito para traduzir uma linguagem mais poderosa em instruções de computador foi um dos mais importantes avanços nos primeiros dias da computação. Os programadores atuais devem sua produtividade – e sua sanidade mental – à criação de linguagens de programação de alto nível e de compiladores que traduzem os programas escritos nessas linguagens em instruções. A Figura 1.3 mostra os relacionamentos entre esses programas e linguagens.

Um compilador permite que um programador escreva esta expressão em linguagem de alto nível:

A + B

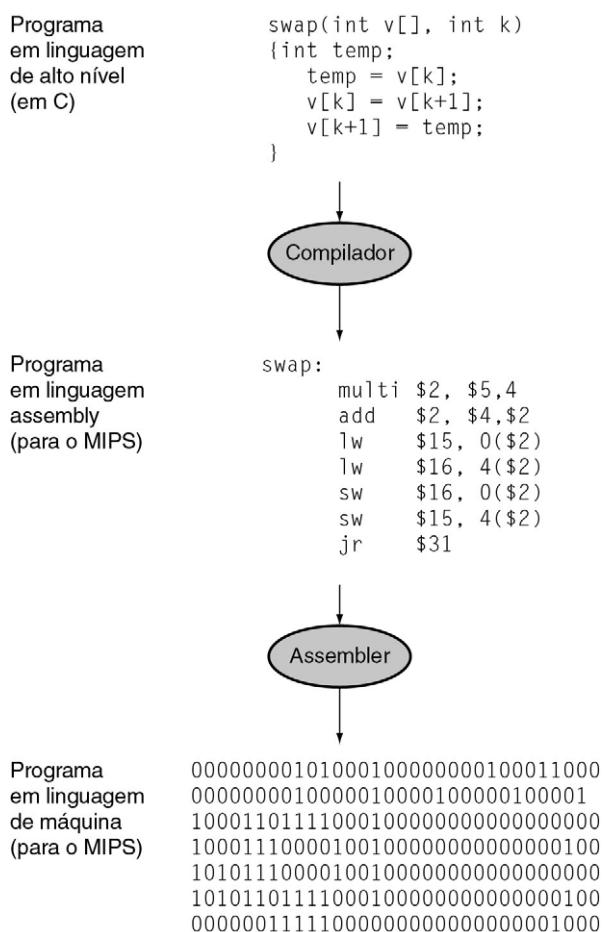


FIGURA 1.3 Programa em C compilado para assembly e depois montado em linguagem de máquina.
Embora a tradução de linguagem de alto nível para a linguagem de máquina seja mostrada em duas etapas, alguns compiladores removem a fase intermediária e produzem linguagem de máquina diretamente. Essas linguagens e esse programa são analisados com mais detalhes no Capítulo 2.

O compilador compilaria isso na seguinte instrução em assembly:

```
add A, B
```

Como podemos ver, o montador traduziria essa instrução para a instrução binária, que diz ao computador para somar os dois números, A e B.

As linguagens de programação de alto nível oferecem vários benefícios importantes. Primeiro, elas permitem que o programador pense em uma linguagem mais natural, usando palavras em inglês e notação algébrica, resultando em programas que se parecem muito mais com texto do que com tabelas de símbolos enigmáticos (veja a [Figura 1.3](#)). Além disso, elas permitem que linguagens sejam projetadas de acordo com o uso pretendido. É por isso que a linguagem Fortran foi projetada para computação científica, Cobol para processamento de dados comerciais, Lisp para manipulação de símbolos e assim por diante. Há também linguagens específicas de domínio para grupos ainda mais estreitos de usuários, como aqueles interessados em simulação de fluidos, por exemplo.

A segunda vantagem das linguagens de programação é a maior produtividade do programador. Uma das poucas áreas em que existe consenso no desenvolvimento de software é que é necessário menos tempo para desenvolver programas quando são escritos em linguagens que exigem menos linhas para expressar uma ideia. A concisão é uma clara vantagem das linguagens de alto nível em relação à linguagem assembly.

A última vantagem é que as linguagens de programação permitem que os programas sejam independentes do computador no qual elas são desenvolvidas, já que os compiladores e montadores podem traduzir programas de linguagem de alto nível para instruções binárias de qualquer máquina. Essas três vantagens são tão fortes que, atualmente, pouca programação é realizada em assembly.

1.3

Sob as tampas

Agora que olhamos por trás do programa para descobrir como ele funciona, vamos abrir a tampa do computador para aprender sobre o hardware dentro dele. O hardware de qualquer computador realiza as mesmas funções básicas: entrada, saída, processamento e armazenamento de dados. A forma como essas funções são realizadas é o principal tema deste livro, e os capítulos subsequentes lidam com as diferentes partes dessas quatro tarefas.

Quando tratamos de um aspecto importante neste livro, tão importante que esperamos que você se lembre dele para sempre, nós o enfatizamos identificando-o como um item “Colocando em perspectiva”. Há aproximadamente 12 desses itens no livro; o primeiro descreve os cinco componentes de um computador que realizam as tarefas de entrada, saída, processamento e armazenamento de dados.

Os cinco componentes de um computador são entrada, saída, memória, caminho de dados e controle; os dois últimos, às vezes, são combinados e chamados de processador. A [Figura 1.4](#) mostra a organização padrão de um computador. Essa organização é independente da tecnologia de hardware: você pode classificar cada parte de cada computador, antigos ou atuais, em uma dessas cinco categorias. Para ajudar a manter tudo isso em perspectiva, os cinco componentes de um computador são mostrados na primeira página dos capítulos seguintes, com a parte relativa ao capítulo destacada.

**Colocando EM
perspectiva**

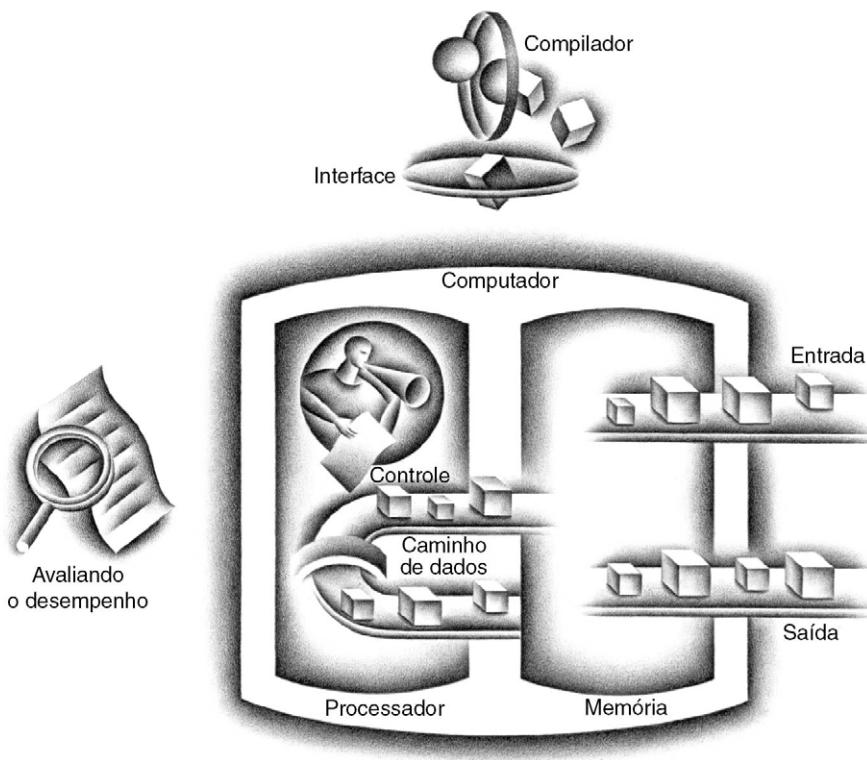


FIGURA 1.4 A organização de um computador, mostrando os cinco componentes clássicos. O processador obtém instruções e dados da memória. A entrada escreve dados na memória, e a saída lê os dados desta. O controle envia os sinais que determinam as operações do caminho de dados, da memória, da entrada e da saída.

dispositivo de entrada Um mecanismo por meio do qual o computador é alimentado com informações, como o teclado e o mouse.

dispositivo de saída Um mecanismo que transmite o resultado de uma computação para o usuário ou para outro computador.

Tive a ideia de criar o mouse enquanto ouvia uma palestra de uma conferência. O orador era tão chato que comecei a me distrair e conceber a ideia.

Doug Engelbart

A Figura 1.5 mostra um computador desktop típico com teclado, mouse sem fio e monitor. Essa fotografia revela dois dos principais componentes dos computadores: os dispositivos de entrada, como o teclado e o mouse, e os dispositivos de saída, como o monitor. Como o nome sugere, a entrada alimenta o computador, e a saída é o resultado da computação, enviado para o usuário. Alguns dispositivos, como redes e discos, fornecem tanto entrada quanto saída para o computador.

O Capítulo 6 descreve dispositivos de entrada e saída (E/S) em mais detalhes, mas vamos dar um passeio introdutório pelo hardware do computador, começando com os dispositivos de E/S externos.

Anatomia do mouse

Embora muitos usuários agora aceitem o mouse sem questionar, a ideia de um dispositivo apontador como um mouse foi mostrada pela primeira vez por Doug Engelbart usando um protótipo em 1967. A Alto, que foi a inspiração para todas as estações de trabalho, inclusive para o Macintosh, incluiu um mouse como seu dispositivo apontador em 1973. Na década de 1990, todos os computadores desktop tinham esse dispositivo, e as novas interfaces gráficas com o usuário e os mouses se tornaram regra.

O mouse original era eletromecânico e usava uma grande esfera que, quando rolada sobre uma superfície, fazia com que um contador x e um y fossem incrementados. A quantidade do aumento de cada contador informava a distância e a direção em que o mouse tinha sido movido.

O mouse eletromecânico está sendo substituído pelo novo mouse ótico. O mouse ótico é, na verdade, um processador óptico em miniatura incluindo um LED para fornecer iluminação, uma minúscula câmera em preto e branco e um processador óptico simples. O LED ilumina a superfície abaixo do mouse; a câmera tira 1.500 fotografias em cada segundo



FIGURA 1.5 Um computador desktop. O monitor de cristal líquido (LCD) é o principal dispositivo de saída, e o teclado e o mouse são os principais dispositivos de entrada. À direita está um cabo Ethernet que conecta o laptop à rede e à Web. O laptop contém o processador, a memória e os dispositivos de E/S adicionais. Esse sistema é um laptop Macbook Pro 15" conectado a um monitor externo.

sob a iluminação. Os quadros sucessivos são enviados para um processador ótico simples, que compara as imagens e determina se e quanto o mouse foi movido. A substituição do mouse eletromecânico pelo mouse eletro-óptico é uma ilustração de um fenômeno comum, no qual os custos cada vez menores e a segurança cada vez maior fazem uma solução eletrônica substituir a tecnologia eletromecânica mais antiga. Mais tarde, veremos outro exemplo: a memória flash.

Diante do espelho

Talvez o dispositivo de E/S mais fascinante seja o monitor gráfico. Todos os computadores laptop e portáteis, calculadoras, telefones celulares e quase todos os computadores desktop agora utilizam monitores de cristal líquido (LCDs) para obterem uma tela fina, com baixa potência. O LCD não é a fonte da luz; em vez disso, ele controla a transmissão da luz. Um LCD típico inclui moléculas em forma de bastão em um líquido que formam uma hélice giratória que encurva a luz que entra na tela, de uma fonte de luz atrás da tela ou, menos frequentemente, da luz refletida. Os bastões se esticam quando uma corrente é aplicada e não encurvam mais a luz. Como o material de cristal líquido está entre duas telas polarizadas a 90 graus, a luz não pode passar a não ser que esteja encurvada. Hoje, a maioria dos monitores LCD utiliza uma matriz ativa que tem uma minúscula chave de transistor em cada pixel para controlar a corrente com precisão e gerar imagens mais nítidas. Uma máscara vermelha-verde-azul associada a cada ponto na tela determina a intensidade dos três componentes de cor na imagem final; em um LCD de matriz ativa colorida, existem três chaves de transistor em cada ponto.

A imagem é composta de uma matriz de elementos de imagem, ou pixels, que podem ser representados como uma matriz de bits, chamada mapa de bits, ou bitmap. Dependendo do tamanho da tela e da resolução, o tamanho da matriz de vídeo variava de 640×480

*Pela tela do computador
aterrissei um avião no pátio
de uma transportadora,
observei uma partícula
nuclear colidir com uma fonte
potencial, voei em um foguete
quase na velocidade da luz
e vi um computador revelar
seus sistemas mais íntimos.*

Ivan Sutherland, o “pai” da computação gráfica,
Scientific American, 1984

monitor de cristal líquido

Uma tecnologia de vídeo usando uma fina camada de polímeros líquidos que podem ser usados para transmitir ou bloquear a luz conforme uma corrente seja ou não aplicada.

monitor de matriz ativa

Um monitor de cristal líquido usando um transistor para controlar a transmissão da luz em cada pixel individual.

pixel O menor elemento da imagem. A tela é composta de centenas de milhares a milhões de pixels, organizados em uma matriz.

a 2560×1600 pixels em 2008. Um monitor colorido pode usar 8 bits para cada uma das três cores primárias (vermelho, verde e azul), totalizando 24 bits por pixel, permitindo que milhões de cores diferentes sejam exibidas.

O suporte de hardware do computador para a utilização de gráficos consiste principalmente em um buffer de atualização de varredura, ou buffer de quadros, para armazenar o mapa de bits. A imagem a ser representada na tela é armazenada no buffer de quadros, e o padrão de bits de cada pixel é lido para o monitor gráfico a uma certa taxa de atualização. A Figura 1.6 mostra um buffer de quadros com 4 bits por pixel.

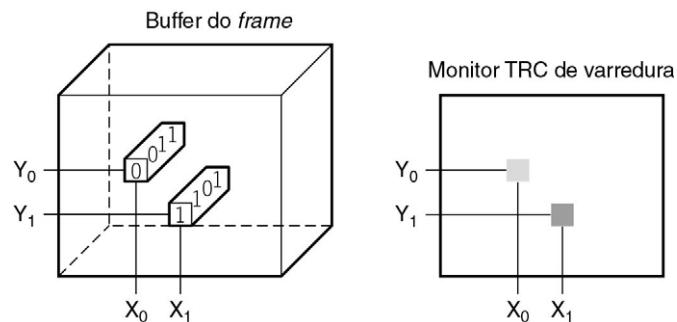


FIGURA 1.6 Cada coordenada no buffer de quadros à esquerda determina o tom da coordenada correspondente para o monitor TRC de varredura à direita. O pixel (X_0, Y_0) contém o padrão de bits 0011, que, na tela, é um tom de cinza mais escuro do que o padrão de bits 1101 no pixel (X_1, Y_1) .

placa-mãe Uma placa de plástico contendo pacotes de circuitos integrados ou chips, incluindo processador, cache, memória e conectores para dispositivos de E/S, como redes e discos.

circuito integrado Também chamado chip, é um dispositivo que combina de dezenas a milhões de transistores.

memória A área de armazenamento temporária em que os programas são mantidos quando estão sendo executados e que contém os dados necessários para os programas em execução.

RAM dinâmica (DRAM) Memória construída como um circuito integrado para fornecer acesso aleatório a qualquer local.

dual inline memory module (DIMM) Uma pequena placa que contém chips DRAM em ambos os lados. (Os SIMMs possuem DRAMs em apenas um lado.)

unidade central de processamento (CPU) Também chamada de processador. A parte ativa do computador, que contém o caminho de dados e o controle e que soma e testa números e sinaliza dispositivos de E/S para que sejam ativados etc.

O objetivo do mapa de bits é representar fielmente o que está na tela. As dificuldades dos sistemas gráficos surgem porque o olho humano é muito bom em detectar mesmo as mais sutis mudanças na tela.

Abrindo o gabinete

Se abrirmos o gabinete de um computador, veremos uma interessante placa de plástico, coberta com dezenas de pequenos retângulos cinzas ou pretos. A Figura 1.7 mostra o conteúdo do computador laptop da Figura 1.5. A placa-mãe é mostrada na parte superior da foto. Duas unidades de disco estão na frente – o disco rígido à esquerda e uma unidade de DVD à direita. O furo no meio é para a bateria do laptop.

Os pequenos retângulos na placa-mãe contêm os dispositivos que impulsionam nossa tecnologia avançada, os circuitos integrados, apelidados de chips. A placa é composta de três partes: a parte que se conecta aos dispositivos de E/S mencionados anteriormente, a memória e o processador.

A memória é onde os programas são mantidos quando estão sendo executados; ela também contém os dados necessários aos programas em execução. Na Figura 1.8, a memória é encontrada nas duas pequenas placas, e cada pequena placa de memória contém oito circuitos integrados. A memória na Figura 1.8 é construída de chips DRAM. DRAM significa **RAM dinâmica** (*Dynamic Random Access Memory*). Várias DRAMs são usadas em conjunto para conter as instruções e os dados de um programa. Ao contrário das memórias de acesso sequencial, como as fitas magnéticas, a parte RAM do termo DRAM significa que os acessos à memória levam o mesmo tempo independente da parte da memória lida.

O **processador** é a parte ativa da placa, que segue rigorosamente as instruções de um programa. Ele soma e testa números, sinaliza dispositivos de E/S para serem ativados e assim por diante. O processador é o quadrado abaixo da ventoinha e coberto por um dissipador de calor à esquerda na Figura 1.7. Ocionalmente, as pessoas chamam o processador de CPU, que significa o termo pomposo **unidade central de processamento**.

Penetrando ainda mais no hardware, a Figura 1.9 revela detalhes de um microprocessador. O processador contém dois componentes principais: o caminho de dados e o controle, correspondendo, respectivamente, aos músculos e ao cérebro do processador. O **caminho**

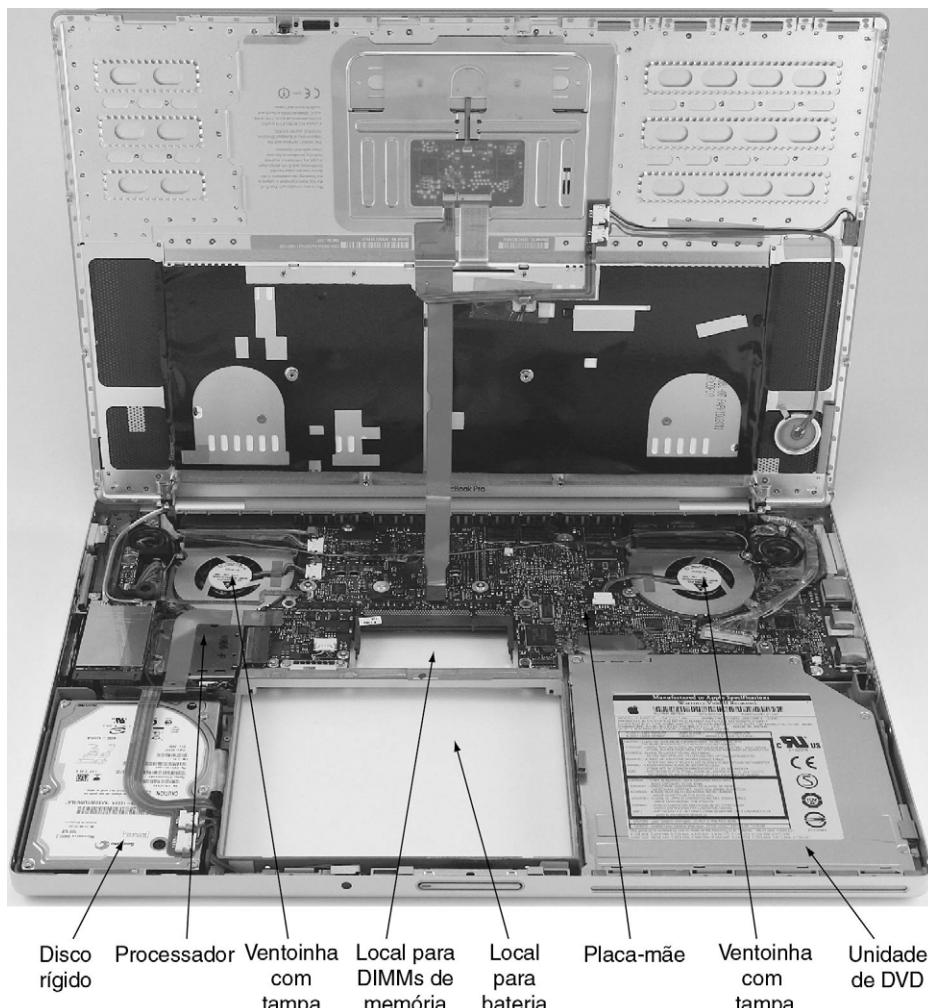


FIGURA 1.7 Dentro do computador laptop da Figura 1.5. A pequena caixa com o rótulo branco no canto inferior esquerdo é uma unidade de disco rígido SATA de 100 GB, e a pequena caixa de metal no canto inferior direito é a unidade de DVD. O espaço entre elas é onde ficaria a bateria do laptop. O pequeno furo acima do espaço da bateria é para as memórias DIMMs. A [Figura 1.8](#) é uma imagem ampliada das DIMMs, que são inseridas por baixo nesse laptop. Acima do espaço da bateria e da unidade de DVD existe uma placa de circuito impresso, chamada *placa-mãe*, que contém a maior parte da eletrônica do computador. Os dois pequenos círculos na metade superior da figura são duas ventoinhas com tampas. O processador é o grande retângulo elevado logo abaixo da ventoinha da esquerda. Foto por cortesia da OtherWorldComputing.com.

de dados realiza as operações aritméticas, e o **controle** diz ao caminho de dados, à memória e aos dispositivos de E/S o que fazer de acordo com as instruções do programa. O Capítulo 4 explica o caminho de dados e o controle para um projeto de desempenho mais alto.

Descer até as profundezas de qualquer componente de hardware revela os interiores da máquina. Dentro do processador, existe outro tipo de memória – a memória cache. A **memória cache** consiste em uma memória pequena e rápida que age como um buffer para a memória DRAM. (A definição não-técnica de *cache* é um lugar seguro para esconder as coisas.) A cache é construída usando uma tecnologia de memória diferente, a RAM estática – **Static Random Access Memory (SRAM)**. A SRAM é mais rápida, mas menos densa e, portanto, mais cara do que a DRAM.

Você pode ter observado um conceito comum nas descrições de software e de hardware: penetrar nas profundezas do hardware ou software revela mais informações, ou seja, detalhes de nível mais baixo estão ocultos para oferecer um modelo mais simples aos níveis mais altos. O uso dessas camadas, ou **abstrações**, é uma técnica importante para projetar sistemas computacionais extremamente sofisticados.

caminho de dados O componente do processador que realiza operações aritméticas.

controle O componente do processador que comanda o caminho de dados, a memória e os dispositivos de E/S de acordo com as instruções do programa.

memória cache Uma memória pequena e rápida que age como um buffer para uma memória maior e mais lenta.

Static Random Access Memory (SRAM) Também uma memória montada como um circuito integrado, porém mais rápida e menos densa que a DRAM.

abstração Um modelo que revela detalhes de nível inferior dos sistemas computacionais temporariamente invisíveis, a fim de facilitar o projeto de sistemas sofisticados.

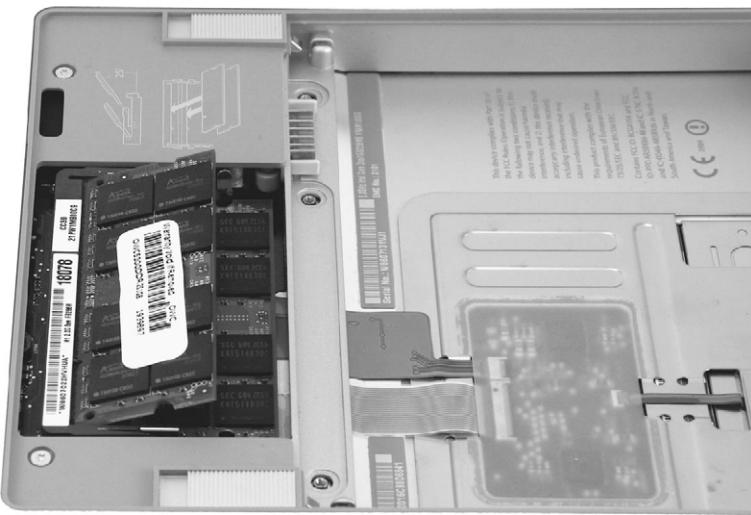


FIGURA 1.8 Close do fundo do laptop revela a memória. A memória principal está contida em uma ou mais placas pequenas mostradas à esquerda. O furo para a bateria está à direita. Os chips de DRAM são montados nessas placas (chamadas *Dual Inline Memory Modules – DIMMs*) e, então, ligados aos conectores. Foto por cortesia da OtherWorldComputing.com.

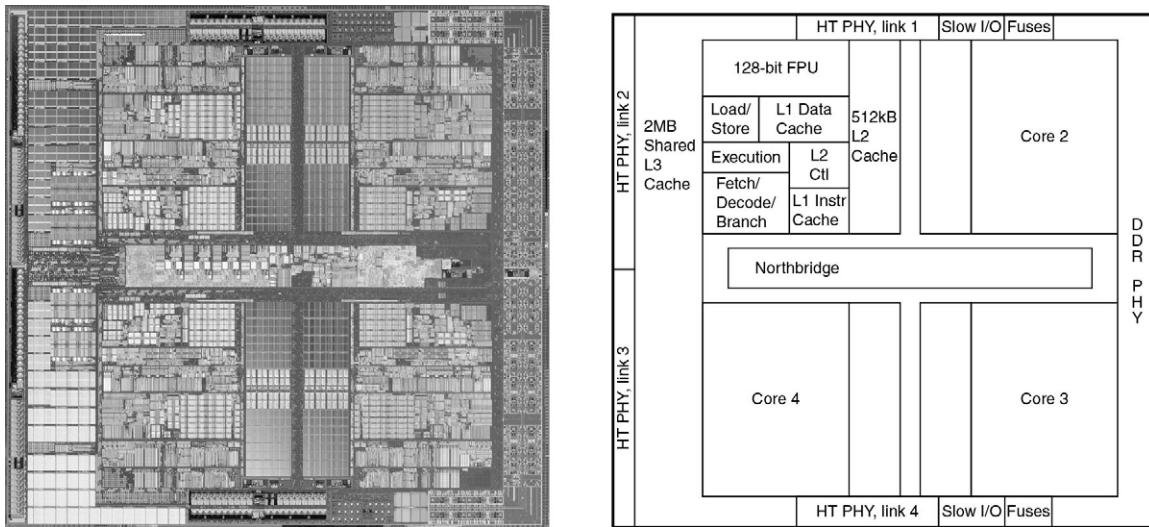


FIGURA 1.9 Dentro do microprocessador AMD Barcelona. O lado esquerdo é uma microfotografia do chip processador AMD Barcelona, e o lado direito mostra os principais blocos no processador. O chip tem quatro processadores ou “cores”. O microprocessador no laptop da Figura 1.8 tem dois cores por chip, chamado Intel Core 2 Duo.

arquitetura do conjunto de instruções Também chamada simplesmente de **arquitetura**. Uma interface abstrata entre o hardware e o software de nível mais baixo de uma máquina que abrange todas as informações necessárias para escrever um programa em linguagem de máquina que será corretamente executado, incluindo instruções, registradores, acesso à memória, E/S e assim por diante.

Uma das abstrações mais importantes é a interface entre o hardware e o software de nível mais baixo. Em decorrência de sua importância, ela recebe um nome especial: a **arquitetura do conjunto de instruções**, ou simplesmente **arquitetura**, de uma máquina. A arquitetura do conjunto de instruções inclui tudo o que os programadores precisam saber para fazer um programa em linguagem de máquina binária funcionar corretamente, incluindo instruções, dispositivos de E/S etc. Em geral, o sistema operacional encapsulará os detalhes da E/S, da alocação de memória e de outras funções de baixo nível do sistema, para que os programadores das aplicações não precisem se preocupar com esses detalhes. A combinação do conjunto de instruções básico e a interface do sistema operacional fornecida para os programadores das aplicações é chamada de **interface binária de aplicação (ABI)**.

Uma arquitetura do conjunto de instruções permite aos projetistas de computador falarem sobre funções independentes do hardware que as realiza. Por exemplo, podemos falar sobre as funções de um relógio digital (marcar as horas, exibir as horas, definir o

alarme) sem falar sobre o hardware do relógio (o cristal de quartzo, os visores de LEDs, os botões plásticos). Os projetistas de computador distinguem entre a arquitetura e uma **implementação** da arquitetura da mesma maneira: uma implementação é o hardware que obedece à abstração da arquitetura. Esses conceitos nos levam a outra seção “Colocando em perspectiva”.

Tanto o hardware quanto o software consistem em camadas hierárquicas, com cada camada inferior ocultando detalhes do nível acima. Esse princípio de *abstração* é o modo como os projetistas de hardware e os de software lidam com a complexidade dos sistemas computacionais. Uma interface-chave entre os níveis de abstração é a *arquitetura do conjunto de instruções* — a interface entre o hardware e o software de baixo nível. Essa interface abstrata permite que muitas *implementações* com custo e desempenho variáveis executem um software idêntico.

Um lugar seguro para os dados

Até agora, vimos como os dados são inseridos, processados e exibidos. Entretanto, se houvesse uma interrupção no fornecimento de energia, tudo seria perdido porque a memória dentro do computador é **volátil** — ou seja, quando perde energia, ela se esquece. Por outro lado, um DVD não se esquece da música gravada quando você desliga o aparelho de DVD e, portanto, é uma tecnologia de **memória não volátil**.

Para distinguir entre a memória usada para armazenar dados e programas enquanto estão sendo executados e essa memória não volátil usada para armazenar programas entre as execuções, o termo **memória principal** é usado para o primeiro e o termo **memória secundária** é usado para o último. As DRAMs dominam a memória principal desde 1975; e os **discos magnéticos** dominam a memória secundária desde 1965. O principal armazenamento não volátil usado em todos os computadores servidores e estações de trabalho é o **disco rígido** magnético. A **memória flash**, uma memória semicondutora não volátil, é usada no lugar dos discos em dispositivos móveis, como telefones celulares, e está cada vez mais substituindo os discos em tocadores de música e laptops.

Como mostra a [Figura 1.10](#), um disco rígido magnético consiste em uma série de discos, que giram em torno de um eixo a velocidades que variam entre 5.400 e 15.000 rotações por minuto. Os discos de metal são cobertos por um material de gravação magnético em ambos os lados, semelhante ao material encontrado em uma fita cassete ou de vídeo. Para ler e gravar informações em um disco rígido, um *braço móvel* com um pequena bobina eletromagnética, chamada de *cabeça de leitura/gravação*, é localizada pouco acima de cada superfície. A unidade inteira é selada permanentemente para controlar o ambiente dentro da unidade, o que, por sua vez, permite que as cabeças do disco estejam muito mais próximas da superfície da unidade.

O uso de componentes mecânicos significa que os tempos de acesso para os discos magnéticos são muito maiores do que para as DRAMs: os discos em geral levam de 5 a 20 milissegundos, enquanto as DRAMs levam de 50 a 70 nanossegundos — tornando as DRAMs cerca de 100.000 vezes mais rápidas. Entretanto, os discos possuem custos muito mais baixos do que a DRAM para a mesma capacidade de armazenamento, pois os custos de produção para uma determinada quantidade de armazenamento em disco são menores do que para a mesma quantidade de circuito integrado. Em 2008, o custo por gigabyte de disco era aproximadamente de 30 a 100 vezes menor do que o custo da DRAM.

interface binária de aplicação (ABI)

A parte voltada ao usuário do conjunto de instruções mais as interfaces do sistema operacional usadas pelos programadores das aplicações. Define um padrão para a portabilidade binária entre computadores.

implementação Hardware que obedece à abstração de uma arquitetura.

Colocando em perspectiva

memória volátil Armazenamento, como a DRAM, que conserva os dados apenas enquanto estiver recebendo energia.

memória não volátil Uma forma de memória que conserva os dados mesmo na ausência de energia e que é usada para armazenar programas entre execuções. O disco magnético é não volátil.

memória principal A memória usada para armazenar os programas enquanto estão sendo executados; normalmente consiste na DRAM nos computadores atuais.

memória secundária Memória não volátil usada para armazenar programas e dados entre execuções; normalmente consiste em discos magnéticos nos computadores atuais.

disco magnético (também chamado de disco rígido) Uma forma de memória secundária não volátil composta por discos giratórios cobertos com um material de gravação magnético.

memória flash Uma memória semicondutora não volátil. Ela é mais barata e mais lenta que a DRAM, porém mais cara e mais rápida que os discos magnéticos.

gigabyte Tradicionalmente, 1.073.741.824 (2^{30}) bytes, embora alguns sistemas de comunicações e de armazenamento secundário o tenham definido como 1.000.000.000 (10⁹) bytes. De modo semelhante, dependendo do contexto, o megabyte é 2²⁰ ou 10⁶ bytes.

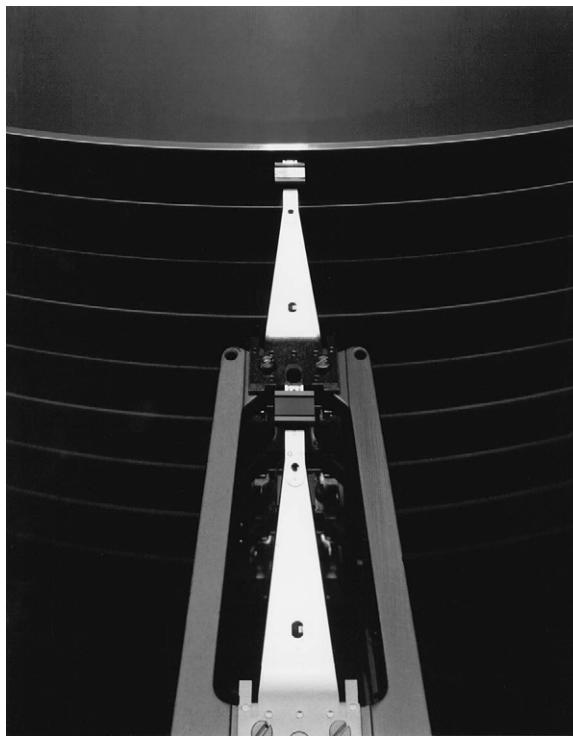


FIGURA 1.10 Uma unidade de disco mostrando 10 discos e as cabeças de leitura/gravação.

Os diâmetros dos discos rígidos atuais variam por um fator de mais de 3, de menos de 2,5cm até 9cm, e têm sido reduzidos ao longo dos anos para se adequarem a novos produtos; servidores, estações de trabalho, computadores pessoais, laptops, palmtops e câmeras digitais têm inspirado novos tamanhos de disco. Tradicionalmente, os discos maiores possuem melhor desempenho, os discos menores possuem o menor custo, e o melhor custo por gigabyte varia. Embora a maioria dos discos rígidos apareça dentro dos computadores (como na Figura 1.7), os discos rígidos também podem ser conectados usando-se interfaces externas, como Universal Serial Bus (USB).

Assim, existem três principais diferenças entre os discos magnéticos e a memória principal: os discos não são voláteis porque são magnéticos; possuem um tempo de acesso maior porque são dispositivos mecânicos; e são mais baratos por gigabyte porque possuem capacidade de armazenamento muito alta a um custo razoável.

Muitos tentaram inventar uma tecnologia mais barata que a DRAM, porém mais rápida que o disco, para preencher a lacuna, mas fracassaram. Os desafiantes nunca levaram um produto ao mercado no momento certo. Quando um novo produto estava para ser entregue, as DRAMs e os discos continuavam a ter avanços rápidos, os custos caíam de modo correspondente e os produtos desafiantes ficavam imediatamente obsoletos.

A memória flash, porém, é um desafiante sério. Essa memória semicondutora é não volátil como os discos e tem aproximadamente a mesma largura de banda, mas a latência é 100 a 1.000 vezes mais rápida que o disco. Flash é popular em câmeras e players de música portáteis, pois vem com capacidades muito menores, é mais reforçado e mais eficiente em termos de potência do que os discos, apesar de o custo por gigabyte em 2008 ser cerca de 6 a 10 vezes maior que o de disco. Diferente dos discos e da DRAM, os bits da memória flash se desgastam após 100.000 a 1.000.000 de escritas. Assim, os sistemas de arquivo precisam registrar o número de escritas e ter uma estratégia para evitar desgastar o armazenamento, por exemplo, movendo dados populares. O Capítulo 6 descreve a memória flash com mais detalhes.

Embora os discos rígidos não sejam removíveis, há várias tecnologias de armazenamento em uso que incluem as seguintes:

- Os discos óticos, incluindo os *compact disks* (CDs) e *digital video disks* (DVDs), constituem a forma mais comum de armazenamento removível. O padrão de disco ótico Blu-Ray (BD) é o aparente sucessor do DVD.

- As placas de memória removíveis baseadas em FLASH geralmente são conectadas por uma conexão Universal Serial Bus (USB) e muitas vezes são usadas para transferir arquivos.
- A fita magnética fornece apenas acesso serial lento e tem sido usada para realização de backups de disco – uma função que, hoje, normalmente está sendo substituída por discos rígidos duplicados.

A tecnologia de disco ótico funciona de uma maneira completamente diferente da tecnologia de disco magnético. Em um CD, os dados são gravados em espiral, com os bits individuais gravados queimando-se pequenas cavidades – de aproximadamente 1 micron (10^{-6} metros) de diâmetro – na superfície do disco. O disco é lido emitindo-se um laser na superfície do CD e determinando-se, pelo exame da luz refletida, se existe uma cavidade ou uma superfície plana (reflexiva). Os DVDs usam o mesmo método de emissão de um feixe de laser em direção a uma série de cavidades e superfícies planas. Além disso, há diversas camadas em que o feixe de laser pode ser focalizado, e o tamanho de cada cavidade é muito menor, o que, em conjunto, representa um significativo aumento na capacidade. Blu-Ray utiliza lasers com menor comprimento de onda, o que reduz o tamanho dos bits e, portanto, aumenta a capacidade.

Os gravadores de disco ótico nos computadores pessoais usam um laser para criar as cavidades na camada de gravação na superfície do CD ou DVD. Esse processo de gravação é relativamente lento, levando de minutos (para um CD inteiro) a dezenas de minutos (para um DVD inteiro). Portanto, para grandes quantidades, é usada uma técnica diferente, chamada *impressão*, que custa apenas alguns centavos por disco ótico.

Os CDs e DVDs regraváveis usam uma superfície de gravação diferente que possui um material reflexivo cristalino; as cavidades não reflexivas são formadas de maneira semelhante ao CD ou DVD de gravação única. Para apagar o CD ou DVD, a superfície é aquecida e resfriada lentamente, permitindo um processo de recocimento para restaurar a camada de gravação da superfície à sua estrutura cristalina original. Esses discos regraváveis são mais caros do que os discos de gravação única; para os discos somente de leitura – usados para distribuir software, música ou filmes – os custos do disco e da gravação são muito menores.

Comunicação com outros computadores

Explicamos como podemos realizar entrada, processamento, exibição e armazenamento de dados, mas ainda falta um item que é encontrado nos computadores modernos: as redes de computadores. Exatamente como o processador mostrado na [Figura 1.4](#) está conectado à memória e aos dispositivos de E/S, as redes conectam computadores inteiros, permitindo que os usuários estendam a capacidade de computação incluindo a comunicação. As redes se tornaram tão comuns que, hoje, constituem o backbone (espinha dorsal) dos sistemas de computação atuais; uma máquina nova sem uma interface de rede opcional seria ridicularizada. Os computadores em rede possuem diversas vantagens importantes:

- *Comunicação*: as informações são trocadas entre computadores em altas velocidades.
- *Compartilhamento de recursos*: em vez de cada máquina ter seus próprios dispositivos de E/S, os dispositivos podem ser compartilhados pelos computadores que compõem a rede.
- *Acesso remoto*: conectando computadores por meio de longas distâncias, os usuários não precisam estar perto do computador que estão usando.

As redes variam em tamanho e desempenho, com o custo da comunicação aumentando de acordo com a velocidade de comunicação e a distância em que as informações viajam. Talvez o tipo de rede mais comum seja a *Ethernet*. Sua extensão é limitada em aproximadamente um quilômetro, transferindo até 10 gigabits por segundo. Sua extensão e velocidade tornam a Ethernet útil para conectar computadores no mesmo andar de um

rede local (LAN) Uma rede projetada para transportar dados dentro de uma área geograficamente restrita, em geral, dentro de um mesmo prédio.

rede remota (WAN) Uma rede estendida por centenas de quilômetros, que pode atravessar continentes.

prédio; portanto, esse é um exemplo do que é chamado genericamente de **rede local (LAN)**. As redes locais são interconectadas com switches que também podem fornecer serviços de roteamento e segurança. As **redes remotas (WAN)** atravessam continentes e são a espinha dorsal da Internet, que é o suporte da World Wide Web. Elas costumam ser baseadas em cabos de fibra ótica e são alugadas de empresas de telecomunicações.

As redes mudaram a cara da computação nos últimos 25 anos, por se tornarem muito mais comuns e aumentarem dramaticamente o desempenho. Na década de 1970, poucas pessoas tinham acesso ao correio eletrônico (e-mail). A Internet e a Web não existiam, e a remessa física de fitas magnéticas era o meio principal de transferir grandes quantidades de dados entre dois locais. Nessa década, as redes locais eram quase inexistentes e as poucas redes remotas existentes tinham capacidade limitada e acesso restrito.

À medida que a tecnologia de redes avançou, ela se tornou muito mais barata e obteve uma capacidade de transmissão muito mais alta. Por exemplo, a primeira tecnologia de rede local a ser padronizada, desenvolvida há cerca de 25 anos, foi uma versão da Ethernet que tinha uma capacidade máxima (também chamada de largura de banda) de 10 milhões de bits por segundo, normalmente compartilhada por dezenas, se não centenas, de computadores. Hoje, a tecnologia de rede local oferece uma capacidade de transmissão de 100 milhões de bits por segundo até 10 gigabits por segundo, em geral compartilhada por, no máximo, alguns computadores. A tecnologia de comunicação ótica permitiu um crescimento semelhante na capacidade das redes remotas de centenas de kilobits até gigabits, e de centenas de computadores conectados a uma rede mundial até milhões de computadores conectados. Essa combinação do dramático aumento no emprego das redes e os aumentos em sua capacidade tornaram a tecnologia de redes o ponto central para a revolução da informação nos últimos 25 anos.

Recentemente, outra inovação na tecnologia de redes está reformulando a maneira como os computadores se comunicam. A tecnologia sem fio se tornou amplamente utilizada e a maioria dos laptops hoje incorpora essa tecnologia. A capacidade de criar um rádio com a mesma tecnologia de semicondutor de baixo custo (CMOS) usada para memória e microprocessadores permitiu uma significativa melhoria no preço, levando a uma explosão no consumo. As tecnologias sem fio disponíveis atualmente, conhecidas pelo padrão IEEE 802.11, permitem velocidades de transmissão de 1 a quase 100 milhões de bits por segundo. A tecnologia sem fio é um pouco diferente das redes baseadas em fios, já que todos os usuários em uma área próxima compartilham as ondas aéreas.

Verifique você mesmo

- A DRAM e o armazenamento de disco diferem significativamente. Descreva a principal diferença quanto a cada um dos seguintes aspectos: volatilidade, tempo de acesso e custo.

Tecnologias para construção de processadores e memórias

Os processadores e a memória melhoraram em uma velocidade espantosa porque os projetistas de computadores, durante muito tempo, abraçaram o que havia de mais moderno na tecnologia eletrônica a fim de tentar vencer a corrida para projetar um computador melhor. A [Figura 1.11](#) mostra as tecnologias usadas ao longo do tempo, com uma estimativa do desempenho relativo por custo unitário para cada tecnologia. A [Seção 1.7](#) explora a tecnologia que impulsionou a indústria da computação desde 1975 e continuará

Ano	Tecnologia usada nos computadores	Desempenho relativo/custo unitário
1951	Válvula	1
1965	Transistor	35
1975	Círculo Integrado	900
1995	Círculo VLSI (Very Large Scale Integrated)	2.400.000
2005	Círculo ULSI (Ultra Large Scale Integrated)	6.200.000.000

FIGURA 1.11 Desempenho relativo por custo unitário das tecnologias usadas nos computadores ao longo do tempo. Fonte: Computer Museum, Boston, com o ano de 2005 estimado pelos autores. Ver [Seção 1.10](#) no site.

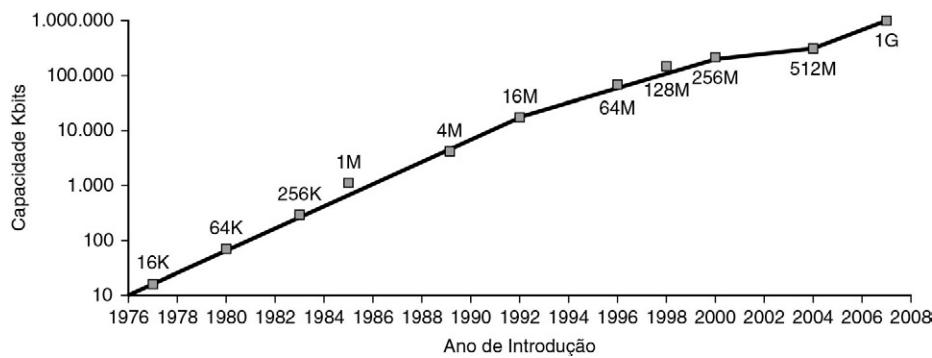


FIGURA 1.12 Crescimento da capacidade por chip de DRAM ao longo do tempo. O eixo y é medido em Kbits, em que $K = 1024 (2^{10})$. A indústria de DRAM quadruplicou a capacidade a cada quase três anos, um aumento de 60% por ano, durante 20 anos. Nos últimos anos, essa taxa diminuiu um pouco e está próximo do dobro a cada dois anos.

a impulsioná-la no futuro previsível. Como essa tecnologia esboça o que os computadores serão capazes de fazer e a velocidade com que irão evoluir, acreditamos que todos os profissionais de computação devem estar familiarizados com os fundamentos dos circuitos integrados.

Um **transistor** é simplesmente uma chave liga/desliga controlada por eletricidade. O *circuito integrado* (*CI*) combinou dezenas a centenas de transistores em um único chip. Para descrever o incrível aumento no número de transistores de centenas para milhões, o adjetivo *escala muito grande* é acrescentado ao termo, criando a abreviação **VLSI** (de **Very Large Scale Integrated**).

Essa taxa de integração crescente tem se mantido notavelmente estável. A [Figura 1.12](#) mostra o crescimento na capacidade da DRAM desde 1977. Durante 20 anos, a indústria quadruplicou consistentemente a capacidade a cada três anos, resultando em um aumento de mais de 16.000 vezes! Esse aumento no número de transistores para um circuito integrado é popularmente conhecido como a Lei de Moore, que diz que a capacidade em transistores dobra a cada 18 a 24 meses. A Lei de Moore resultou de uma previsão desse crescimento na capacidade do circuito integrado feita por Gordon Moore, um dos fundadores da Intel durante a década de 1960.

Sustentar essa taxa de progresso por quase 40 anos exigiu incríveis inovações nas técnicas de fabricação. Na Seção 1.7, discutimos como os circuitos integrados são fabricados.

válvula Um componente eletrônico, predecessor do transistor, que consiste em um tubo de vidro oco de aproximadamente 5 a 10 centímetros de comprimento do qual o máximo de ar foi removido e que usa um feixe de elétrons para transferir dados.

transistor Uma chave liga/desliga controlada por um sinal elétrico.

circuito Very Large Scale Integrated (VLSI) Um dispositivo com centenas de milhares a milhões de transistores.

1.4

Desempenho

Avaliar o desempenho dos computadores pode ser desafiador. A escala e a complexidade dos sistemas de software modernos, junto com a grande variedade de técnicas de melhoria de desempenho empregadas por projetistas de hardware, tornaram a avaliação do desempenho muito mais difícil.

Ao tentar escolher entre diferentes computadores, o desempenho é um atributo importante. Comparar e avaliar com precisão diferentes computadores é crítico para compradores e por consequência, também para os projetistas. O pessoal que vende computadores também sabe disso. Normalmente, os vendedores desejam que você veja seu computador da melhor maneira possível, não importa se isso reflete ou não as necessidades da aplicação do comprador. Logo, ao escolher um computador, é importante entender como medir melhor o desempenho e as limitações das medições de desempenho.

O restante desta seção descreve diferentes maneiras como o desempenho pode ser determinado; depois, descrevemos as métricas para avaliar o desempenho do ponto de vista

de um usuário do computador e um projetista. Também analisamos como essas métricas estão relacionadas e apresentamos a equação clássica de desempenho do processador, que usaremos no decorrer do texto.

Definindo o desempenho

Quando dizemos que um computador tem melhor desempenho que outro, o que queremos dizer? Embora essa pergunta possa parecer simples, uma analogia com aviões de passageiros mostra como a questão de desempenho pode ser sutil. A [Figura 1.13](#) mostra alguns aviões de passageiros típicos, juntamente com a velocidade de cruzeiro, alcance e capacidade. Se você quisesse saber qual dos aviões nessa tabela tem o melhor desempenho, primeiro precisaríamos definir o desempenho. Por exemplo, considerando diferentes medidas de desempenho, vemos que o avião com a maior velocidade de cruzeiro é o Concorde, o avião com o maior alcance é o DC-8, e o avião com a maior capacidade é o 747.

Aeronave	Capacidade de passageiros	Alcance do voo (milhas)	Velocidade do voo (mi/h)	Taxa de passageiros (passageiros × mi/h)
Boeing 777	375	4.630	610	228.750
Boeing 747	470	4.150	610	286.700
BAC/Sud Concorde	132	4.000	1.350	178.200
Douglas DC-8-50	146	8.720	544	79.424

FIGURA 1.13 A capacidade, alcance e velocidade de uma série de aviões comerciais. A última coluna mostra a taxa com que o avião transporta passageiros, que é a capacidade vezes a velocidade de voo (ignorando o alcance e os tempos de decolagem e pouso).

Vamos supor que o desempenho seja definido em termos de velocidade. Isso ainda deixa duas definições possíveis. Você poderia definir o avião mais rápido como aquele com a velocidade de voo mais alta, levando um único passageiro de um ponto para outro com o menor tempo. Porém, se você estivesse interessado em transportar 450 passageiros de um ponto para outro, o 747 certamente seria o mais rápido, como mostra a última coluna da figura. De modo semelhante, podemos definir o desempenho do computador de diferentes maneiras.

Se você estivesse rodando um programa em dois computadores desktop diferentes, diria que o mais rápido é o computador que termina o trabalho primeiro. Se estivesse gerenciando um centro de dados com diversos servidores rodando tarefas submetidas por muitos usuários, você diria que o computador mais rápido é aquele que completasse o máximo de tarefas durante um dia. Como um usuário de computador individual, você está interessado em reduzir o **tempo de resposta** — o tempo entre o início e o término de uma tarefa — também conhecido como **tempo de execução**. Os gerentes de centro de dados normalmente estão interessados em aumentar o **throughput** ou **largura de banda** — a quantidade total de trabalho realizado em determinado tempo. Logo, na maioria dos casos, ainda precisaremos de diferentes métricas de desempenho, além de diferentes conjuntos de aplicações para avaliar computadores embutidos e de desktop, que são mais voltados para o tempo de resposta, contra servidores, que são mais voltados para o throughput.

tempo de resposta Também chamado tempo de execução. O tempo total exigido para o computador completar uma tarefa, incluindo acessos ao disco, acessos à memória, atividades de E/S, overhead do sistema operacional, tempo de execução de CPU e assim por diante.

throughput Também chamado largura de banda. Outra medida de desempenho, é o número de tarefas completadas por unidade de tempo.

Throughput e tempo de resposta

As mudanças a seguir em um sistema de computador aumentam o throughput, diminuem o tempo de resposta ou ambos?

1. Substituir o processador em um computador por uma versão mais rápida.

EXEMPLO

2. Acrescentar processadores adicionais a um sistema que utiliza múltiplos processadores para tarefas separadas — por exemplo, busca na World Wide Web.

Diminuir o tempo de resposta quase sempre melhora o throughput. Logo, no caso 1, o tempo de resposta e o throughput são melhorados. No caso 2, ninguém realiza o trabalho mais rapidamente, de modo que somente o throughput aumenta.

RESPOSTA

Porém, se a demanda para processamento no segundo caso fosse quase tão grande quanto o throughput, o sistema poderia forçar as solicitações a se enfileirarem. Nesse caso, aumentar o throughput também poderia melhorar o tempo de resposta, pois poderia reduzir o tempo de espera na fila. Assim, em muitos sistemas de computadores reais, mudar o tempo de execução ou o throughput normalmente afeta o outro.

Na discussão sobre o desempenho dos computadores, vamos nos preocupar principalmente com o tempo de resposta nos primeiros capítulos. Para maximizar o desempenho, queremos minimizar o tempo de resposta ou o tempo de execução para alguma tarefa. Assim, podemos relacionar desempenho e tempo de execução para o computador X:

$$\text{Desempenho}_X = \frac{1}{\text{Tempo de execução}_X}$$

Isso significa que, para dois computadores X e Y, se o desempenho de X for maior que o desempenho de Y, temos

$$\text{Desempenho}_X > \text{Desempenho}_Y$$

$$\frac{1}{\text{Tempo de execução}_X} > \frac{1}{\text{Tempo de execução}_Y}$$

$$\text{Tempo de execução}_Y > \text{Tempo de execução}_X$$

Ou seja, o tempo de execução em Y é maior que o de X, se X for mais rápido que Y.

Na discussão de um projeto de computador, normalmente queremos relacionar o desempenho de dois computadores diferentes quantitativamente. Usaremos a frase “X é n vezes mais rápido que Y” — ou, de modo equivalente, “X tem n vezes a velocidade de Y” — para indicar

$$\frac{\text{Desempenho}_X}{\text{Desempenho}_Y} = n$$

Se X for n vezes mais rápido que Y, então o tempo de execução em Y é n vezes maior do que em X:

$$\frac{\text{Desempenho}_X}{\text{Desempenho}_Y} = \frac{\text{Tempo de execução}_Y}{\text{Tempo de execução}_X} = n$$

Desempenho relativo

Se o computador A executa um programa em 10 segundos e o computador B executa o mesmo programa em 15 segundos, o quanto A é mais rápido que B?

EXEMPLO

RESPOSTA

Sabemos que A é n vezes mais rápido que B se

$$\frac{\text{Desempenho}_A}{\text{Desempenho}_B} = \frac{\text{Tempo de execução}_B}{\text{Tempo de execução}_A} = n$$

Assim, a razão de desempenho é

$$\frac{15}{10} = 1,5$$

e A, portanto, é 1,5 vez mais rápido que B.

No exemplo anterior, também poderíamos dizer que o computador B é 1,5 vez mais lento que o computador A, pois

$$\frac{\text{Desempenho}_A}{\text{Desempenho}_B} = 1,5$$

significando que

$$\frac{\text{Desempenho}_A}{1,5} = \text{Desempenho}_B$$

Para simplificar, normalmente usaremos a terminologia mais rápido que quando tentamos comparar computadores quantitativamente. Como o desempenho e o tempo de execução são recíprocos, aumentar o desempenho requer diminuir o tempo de execução. Para evitar a confusão em potencial entre os termos aumentar e diminuir, normalmente dizemos “melhorar o desempenho” ou “melhorar o tempo de execução” quando queremos dizer “aumentar o desempenho” e “diminuir o tempo de execução”.

Medindo o desempenho

O tempo é a medida de desempenho do computador: o computador que realiza a mesma quantidade de trabalho no menor tempo é o mais rápido. O tempo de execução do programa é medido em segundos por programa. Porém, o tempo pode ser definido de diferentes maneiras, dependendo do que contamos. A definição mais clara de tempo é chamada de tempo do relógio, tempo de resposta ou tempo decorrido. Esses termos significam o tempo total para completar uma tarefa, incluindo acessos ao disco, acessos à memória, atividades de entrada/saída (E/S), overhead do sistema operacional — tudo.

Contudo, os computadores normalmente são compartilhados e um processador pode trabalhar em vários programas simultaneamente. Nesses casos, o sistema pode tentar otimizar o throughput em vez de tentar minimizar o tempo decorrido para um programa. Logo, normalmente queremos distinguir entre o tempo decorrido e o tempo que o processador está trabalhando em nosso favor. Tempo de execução de CPU, ou simplesmente tempo de CPU, que reconhece essa distinção, é o tempo que a CPU gasta computando para essa tarefa, e não inclui o tempo gasto esperando pela E/S ou pela execução de outros programas. (Lembre-se, porém, de que o tempo de resposta experimentado pelo usuário será o tempo decorrido do programa, e não o tempo de CPU.) O tempo de CPU pode ser dividido ainda mais no tempo de CPU gasto no programa, chamado tempo de CPU do usuário, e o tempo de CPU gasto no sistema operacional, realizando tarefas em favor do programa, chamado tempo de CPU do sistema. A diferenciação entre o tempo de CPU do sistema e do usuário é difícil de se realizar com precisão, pois normalmente é difícil atribuir a responsabilidade pelas atividades do sistema operacional a um programa do usuário em vez do outro, e por causa das diferenças de funcionalidade entre os sistemas operacionais.

tempo de execução de CPU

Também chamado tempo de CPU. O tempo real que a CPU gasta calculando para uma tarefa específica.

tempo de CPU do usuário

O tempo de CPU gasto em um programa propriamente dito.

tempo de CPU do sistema

O tempo de CPU gasto no sistema operacional realizando tarefas em favor do programa.

Por uma questão de consistência, mantemos uma distinção entre o desempenho baseado no tempo decorrido e baseado no tempo de execução da CPU. Usaremos o termo desempenho do sistema para nos referirmos ao tempo decorrido em um sistema não carregado e desempenho da CPU para nos referirmos ao tempo de CPU do usuário. Vamos focalizar o desempenho da CPU neste capítulo, embora nossas discussões de como resumir o desempenho possam ser aplicadas às medições de tempo decorrido ou tempo de CPU.

Diferentes aplicações são sensíveis a diferentes aspectos do desempenho de um sistema de computador. Muitas aplicações, especialmente aquelas rodando em servidores, dependem muito do desempenho da E/S, que, por sua vez, conta com o hardware e o software. O tempo decorrido total medido por um relógio comum é a medida de interesse. Em alguns ambientes de aplicação, o usuário pode se importar com o throughput, tempo de resposta ou uma combinação complexa dos dois (por exemplo, o throughput máximo com o tempo de resposta no pior caso). Para melhorar o desempenho de um programa, deve-se ter uma definição clara de qual métrica de desempenho interessa e depois prosseguir para procurar gargalos de desempenho medindo a execução do programa e procurando os prováveis gargalos. Nos próximos capítulos, vamos descrever como procurar gargalos e melhorar o desempenho em diversas partes do sistema.

Embora, como usuários de computador, nos importemos com o tempo, quando examinamos os detalhes de um computador, é conveniente pensar sobre o desempenho em outras métricas. Em particular, os projetistas de comunicação podem querer pensar a respeito de um computador usando uma medida que se relaciona à velocidade com que o hardware pode realizar suas funções básicas. Quase todos os computadores são construídos usando-se um clock que determina quando os eventos ocorrem no hardware. Esses intervalos de tempo discretos são chamados de ciclos de clock (ou batidas, batidas de clock, períodos de clock, clocks, ciclos). Os projetistas referem-se à extensão de um período de clock como o tempo para um ciclo de clock completo (por exemplo, 250 picosegundos, ou 250 ps) e como a taxa de clock (por exemplo, 4 gigahertz, ou 4 GHz), que é o inverso do período de clock. Na próxima subseção, formalizaremos o relacionamento entre os ciclos de clock do projetista de hardware e os segundos do usuário do computador.

1. Suponha que saibamos que uma aplicação que usa um cliente de desktop e um servidor remoto seja limitada pelo desempenho da rede. Para as mudanças a seguir, indique se somente o throughput melhora, o tempo de resposta e o throughput melhoram, ou nenhum destes melhora.
 - a. Um canal de rede extra é acrescentado entre o cliente e o servidor, aumentando o throughput total da rede e reduzindo o atraso para obter o acesso à rede (pois agora existem dois canais).
 - b. O software de rede é melhorado, reduzindo assim o atraso na comunicação da rede, mas não aumentando o throughput.
 - c. Mais memória é acrescentada ao computador.
2. O desempenho do computador C é quatro vezes mais rápido que o desempenho do computador B, que executa determinada aplicação em 28 segundos. Quanto tempo o computador C levará para executar essa aplicação?

Desempenho da CPU e seus fatores

Usuários e projetistas normalmente examinam o desempenho usando diferentes métricas. Se pudéssemos relacionar essas diferentes métricas, poderíamos determinar o efeito de uma mudança de projeto sobre o desempenho experimentado pelo usuário. Como estamos

Entendendo o desempenho do programa

ciclo de clock Também chamado batida, batida de clock, período de clock, clock, ciclo. O tempo para um período de clock, normalmente do clock do processador, que trabalha a uma taxa constante.

período de clock A extensão de cada ciclo de clock.

Verifique você mesmo

interessados no desempenho da CPU neste ponto, a medida de desempenho final é o tempo de execução da CPU. Uma fórmula simples relaciona as métricas mais básicas (ciclos de clock e tempo do ciclo de clock) ao tempo da CPU:

$$\frac{\text{Tempo de execução da CPU}}{\text{para um programa}} = \frac{\text{Ciclos de clock da CPU para um programa}}{\text{para um programa}} \times \text{Tempo do ciclo de clock}$$

Como alternativa, como a taxa de clock e o tempo do ciclo de clock são inversos,

$$\frac{\text{Tempo de execução da CPU}}{\text{para um programa}} = \frac{\text{Ciclos de clock da CPU para um programa}}{\text{Taxa de clock}}$$

Essa fórmula deixa claro que o projetista de hardware pode melhorar o desempenho reduzindo o número de ciclos de clock exigidos para um programa ou o tamanho do ciclo de clock. Conforme veremos em outros capítulos, os projetistas normalmente têm de escolher entre o número de ciclos de clock necessários para um programa e a extensão de cada ciclo. Muitas técnicas que diminuem o número de ciclos de clock podem também aumentar o tempo do ciclo de clock.

EXEMPLO

Melhorando o desempenho

Nosso programa favorito executa em 10 segundos no computador A, que tem um clock de 2 GHz. Estamos tentando ajudar um projetista de computador a montar um computador B, que executará esse programa em 6 segundos. O projetista determinou que é possível haver um aumento substancial na taxa de clock, mas esse aumento afetará o restante do projeto da CPU, fazendo com que o computador B exija 1,2 vez a quantidade de ciclos de clock do computador A para esse programa. Que taxa de clock o projetista deve ter como alvo?

RESPOSTA

Vamos primeiro achar o número de ciclos de clock exigidos para o programa em A:

$$\text{Tempo de CPU}_A = \frac{\text{Ciclos de clock de CPU}_A}{\text{Taxa de clock}_A}$$

$$10 \text{ segundos} = \frac{\text{Ciclos de clock de CPU}_A}{2 \times 10^9 \frac{\text{ciclos}}{\text{segundo}}}$$

$$\text{Ciclos de clock de CPU}_A = 10 \text{ segundos} \times 2 \times 10^9 \frac{\text{ciclos}}{\text{segundo}} = 20 \times 10^9 \text{ ciclos}$$

O tempo de CPU para B pode ser encontrado por meio desta equação:

$$\text{Tempo de CPU}_B = \frac{1,2 \times \text{Ciclos de CPU}_A}{\text{Taxa de clock}_B}$$

$$6 \text{ segundos} = \frac{1,2 \times 20 \times 10^9 \text{ ciclos}}{\text{Taxa de clock}_B}$$

$$\text{Taxa de clock}_B = \frac{1,2 \times 20 \times 10^9 \text{ ciclos}}{6 \text{ segundos}} = \frac{0,2 \times 20 \times 10^9 \text{ ciclos}}{\text{segundo}} = \frac{4 \times 10^9 \text{ ciclos}}{\text{segundo}} = 4 \text{ GHz}$$

Para executar o programa em 6 segundos, B deverá ter o dobro da taxa de clock de A.

Desempenho da instrução

Essas equações de desempenho não incluíram qualquer referência ao número de instruções necessárias para o programa. (Veremos como são as instruções que compõem um programa no próximo capítulo.) Porém, como o compilador claramente gera instruções para executar, e o computador teve de rodá-las para executar o programa, o tempo de execução deverá depender do número de instruções em um programa. Um modo de pensar a respeito do tempo de execução é que ele é igual ao número de instruções executadas multiplicado pelo tempo médio por instrução. Portanto, o número de ciclos de clock exigido para um programa pode ser escrito como

$$\text{Ciclos de clock de CPU} = \text{Instruções para um programa} \times \frac{\text{Ciclos de clock médios}}{\text{por instrução}}$$

O termo ciclos de clock por instrução, que é o número médio de ciclos de clock que cada instrução leva para executar, normalmente é abreviado como CPI. Como diferentes instruções podem exigir diferentes quantidades de tempo, dependendo do que elas fazem, CPI é uma média de todas as instruções executadas no programa. CPI oferece um modo de comparar duas implementações diferentes da mesma arquitetura do conjunto de instruções, pois o número de instruções executadas para um programa, logicamente, será o mesmo.

ciclos de clock por instruções (CPI) Número médio de ciclos de clock por instrução para um programa ou fragmento de programa.

Usando a equação de desempenho

Suponha que tenhamos duas implementações da mesma arquitetura de conjunto de instruções. O computador A tem um tempo de ciclo de clock de 250 ps e um CPI de 2,0 para algum programa, e o computador B tem um tempo de ciclo de clock de 500 ps e um CPI de 1,2 para o mesmo programa. Qual computador é mais rápido para esse programa e por quanto?

EXEMPLO

Sabemos que cada computador executa o mesmo número de instruções para o programa; vamos chamar esse número de I. Primeiro, encontramos o número de ciclos de clock do processador para cada computador:

$$\text{Ciclos de clock de CPU}_A = I \times 2,0$$

$$\text{Ciclos de clock de CPU}_B = I \times 1,2$$

RESPOSTA

Agora, podemos calcular o tempo de CPU para cada computador:

$$\begin{aligned}\text{Tempo de CPU}_A &= \text{Ciclos de clock de CPU}_A \times \text{Tempo de ciclo de clock} \\ &= I \times 2,0 \times 250 \text{ ps} = 500 \times I \text{ ps}\end{aligned}$$

De modo semelhante, para B:

$$\text{Tempo de CPU}_B = I \times 1,2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

Claramente, o computador A é mais rápido. A quantidade mais rápida é dada pela razão dos tempos de execução:

$$\frac{\text{Desempenho da CPU}_A}{\text{Desempenho da CPU}_B} = \frac{\text{Tempo de execução}_B}{\text{Tempo de execução}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1,2$$

Podemos concluir que o computador A é 1,2 vez mais rápido que o computador B para esse programa.

A equação clássica de desempenho da CPU

contador de instrução O número de instruções executadas pelo programa.

Agora, podemos escrever essa equação de desempenho básica em termos do contador de instrução (o número de instruções executadas pelo programa), CPI e tempo de ciclo do clock:

$$\text{Tempo de CPU} = \text{Contador de instrução} \times \text{CPI} \times \text{Tempo de ciclo de clock}$$

ou então, como a taxa de clock é o inverso do tempo de ciclo de clock:

$$\text{Tempo de CPU} = \frac{\text{Contador de instrução} \times \text{CPI}}{\text{Taxa de clock}}$$

Essas fórmulas são particularmente úteis porque separam os três fatores principais que afetam o desempenho. Podemos usá-las para comparar duas implementações diferentes ou para avaliar uma alternativa de projeto se soubermos seu impacto sobre esses três parâmetros.

EXEMPLO

Comparando segmentos de código

Um projetista de compilador está tentando decidir entre duas sequências de código para determinado computador. Os projetistas de hardware forneceram os seguintes fatos:

	CPI para cada classe de instrução		
	A	B	C
CPI	1	2	3

Para determinada instrução na linguagem de alto nível, o escritor do compilador está considerando duas sequências de código que exigem as seguintes contagens de instruções:

Sequência de código	Contagens de instruções para cada classe de instrução		
	A	B	C
1	2	1	2
2	4	1	1

Qual sequência de código executa mais instruções? Qual será mais rápida? Qual é o CPI para cada sequência?

A sequência 1 executa $2 + 1 + 2 = 5$ instruções. A sequência 2 executa $4 + 1 + 1 = 6$ instruções. Portanto, a sequência 1 executa menos instruções.

Podemos usar a equação para ciclos de clock de CPU com base na contagem de instruções e CPI a fim de descobrir o número total de ciclos de clock para cada sequência:

$$\text{Ciclos de clock de CPU} = \sum_{i=1}^n (\text{CPI}_i \times C_i)$$

Isso gera

$$\text{Ciclos de clock de CPU}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ ciclos}$$

$$\text{Ciclos de clock de CPU}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ ciclos}$$

Assim, a sequência de código 2 é mais rápida, embora execute uma instrução extra. Como a sequência de código 2 leva menos ciclos de clock em geral, mas tem mais instruções, ela deverá ter um CPI menor. Os valores de CPI podem ser calculados por

$$\text{CPI} = \frac{\text{Ciclos de clock de CPU}}{\text{Contagem de instruções}}$$

$$\text{CPI}_1 = \frac{\text{Ciclos de clock de CPU}_1}{\text{Contagem de instruções}_1} = \frac{10}{5} = 2,0$$

$$\text{CPI}_2 = \frac{\text{Ciclos de clock de CPU}_2}{\text{Contagem de instruções}_2} = \frac{9}{6} = 1,5$$

RESPOSTA

A Figura 1.14 mostra as medições básicas em diferentes níveis no computador e o que está sendo medido em cada caso. Podemos ver como esses fatores são combinados para gerar um tempo de execução medido em segundos por programa:

$$\text{Tempo} = \frac{\text{Segundos}}{\text{Programa}} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Ciclos de clock}}{\text{Instrução}} \times \frac{\text{Segundos}}{\text{Ciclo de clock}}$$

Lembre-se sempre de que a única medida completa e confiável do desempenho do computador é o tempo. Por exemplo, mudar o conjunto de instruções para reduzir a contagem de instruções pode levar a uma organização com um tempo de ciclo de clock menor ou CPI maior, que compensa a melhoria na contagem de instruções. De modo semelhante, como o CPI depende do tipo das instruções executadas, o código que executa o menor número de instruções pode não ser o mais rápido.

Colocando EM perspectiva

Componentes do desempenho	Unidades de medida
Tempo de execução de CPU para um programa	Segundos para o programa
Contagem de instruções	Instruções executadas para o programa
Ciclos de clock por instrução (CPI)	Número médio de ciclos de clock por instrução
Tempo do ciclo de clock	Segundos por ciclo de clock

FIGURA 1.14 Os componentes básicos do desempenho e como cada um é medido.

Como determinar o valor desses fatores na equação de desempenho? Podemos medir o tempo de execução da CPU rodando o programa, e o tempo do ciclo de clock normalmente é publicado como parte da documentação de um computador. A contagem de instruções e o CPI podem ser mais difíceis de se obter. Naturalmente, se soubermos a taxa de clock e o tempo de execução da CPU, só precisamos da contagem de instruções ou do CPI para determinar o outro.

Podemos medir a contagem de instruções usando ferramentas de software que determinam o perfil da execução ou usando um simulador da arquitetura. Como alternativa, podemos usar contadores de hardware, que estão incluídos na maioria dos processadores, para registrar uma série de medidas, incluindo o número de instruções executadas, o CPI médio e, frequentemente, as origens da perda de desempenho. Como a contagem de instruções depende da arquitetura, mas não da implementação exata, podemos medir a contagem de instruções sem conhecer todos os detalhes da implementação. Porém, o CPI depende de diversos detalhes de projeto no computador, incluindo o sistema de memória e a estrutura do processador (conforme veremos nos Capítulos 4 e 5), além da mistura de tipos de instruções executados em uma aplicação. Assim, o CPI varia por aplicação, bem como entre implementações com o mesmo conjunto de instruções.

O exemplo anterior mostra o perigo de usar apenas um fator (contagem de instruções) para avaliar o desempenho. Ao comparar dois computadores, você precisa examinar todos os três componentes, que se combinam para formar o tempo de execução. Se alguns dos fatores forem idênticos, como a taxa de clock no exemplo anterior, o desempenho pode ser determinado comparando-se todos os fatores não idênticos. Como o CPI varia por mix de instruções, tanto a contagem de instruções quanto o CPI precisam ser comparados, mesmo que as taxas de clock sejam idênticas. Vários exercícios ao final deste capítulo lhe pedem para avaliar uma série de melhorias de computador e compilador, que afetam a taxa de clock, CPI e contagem de instruções. Na Seção 1.8, examinaremos uma medida de desempenho comum, que não incorpora todos os termos e, portanto, pode ser enganosa.

mix de instruções Uma medida da frequência dinâmica das instruções por um ou muitos programas.

Entendendo o desempenho do programa

O desempenho de um programa depende do algoritmo, da linguagem, do compilador, da arquitetura e do hardware real. A tabela a seguir resume como esses componentes afetam os fatores na equação de desempenho da CPU.

Componente de hardware ou software	Afeta o quê?	Como?
Algoritmo	Contagem de instruções, possivelmente CPI	O algoritmo determina o número de instruções do programa-fonte executadas e, portanto, o número de instruções de processador executadas. O algoritmo também pode afetar o CPI, favorecendo instruções mais lentas ou mais rápidas. Por exemplo, se o algoritmo utiliza mais operações de ponto flutuante, ele tenderá a ter um CPI mais alto.
Linguagem de programação	Contagem de instruções, CPI	A linguagem de programação certamente afeta a contagem de instruções, pois as instruções na linguagem são traduzidas para instruções de processador, o que determina a contagem de instruções. A linguagem também pode afetar o CPI por causa dos seus recursos; por exemplo, uma linguagem com um suporte intenso para abstração de dados (por exemplo, Java) exigirá chamadas indiretas, que usarão instruções de CPI mais alto.

Componente de hardware ou software	Afeta o quê?	Como?
Compilador	Contagem de instruções, CPI	A eficiência do compilador afeta a contagem de instruções e os ciclos médios por instruções, pois o compilador determina a tradução das instruções da linguagem-fonte para instruções do computador. O papel do compilador pode ser muito complexo e afetar o CPI de maneiras complexas.
Arquitetura do conjunto de instruções	Contagem de instruções, taxa de clock, CPI	A arquitetura do conjunto de instruções afeta todos os três aspectos do desempenho da CPU, pois afeta as instruções necessárias para uma função, o custo em ciclos de cada instrução e a taxa de clock geral do processador.

Detalhamento: Embora você possa esperar que o CPI mínimo seja 1,0, conforme veremos no Capítulo 4, alguns processadores buscam e executam múltiplas instruções por ciclo de clock. Para refletir essa técnica, alguns projetistas invertem o CPI para falar sobre IPC, ou instruções por ciclo de clock. Se um processador executa em média duas instruções por ciclo de clock, então ele tem um IPC de 2 e, portanto, um CPI de 0,5.

Determinada aplicação escrita em Java roda por 15 segundos em um processador de desktop. Um novo compilador Java é lançado, exigindo apenas 60% das instruções do compilador antigo. Infelizmente, isso aumenta o CPI por 1,1. Com que velocidade podemos esperar que a aplicação rode usando esse novo compilador? Escolha a resposta certa a partir das três opções a seguir:

- a. $\frac{15 \times 0,6}{1,1} = 8,2 \text{ seg}$
- b. $15 \times 0,6 \times 1,1 = 9,9 \text{ seg}$
- c. $\frac{15 \times 1,1}{0,6} = 27,5 \text{ seg}$

Verifique você mesmo

1.5

A barreira da potência

A Figura 1.15 mostra o aumento na taxa de clock e na potência de oito gerações de microprocessadores Intel durante 25 anos. Tanto a taxa de clock quanto a potência aumentaram rapidamente durante décadas e depois se estabilizaram recentemente. O motivo pelo qual elas cresceram juntas é que estão correlacionadas e o motivo para o seu recuo recente é que chegamos ao limite de potência prático para o resfriamento dos microprocessadores comuns.

A tecnologia dominante para circuitos integrados é denominada Complementary Metal Oxide Semiconductor (CMOS). Para CMOS, a principal fonte de dissipação de potência é a chamada potência dinâmica — ou seja, a potência que é consumida durante a comutação. A dissipação da potência dinâmica depende da carga capacitiva de cada transistor, da tensão elétrica aplicada e da frequência com que o transistor é comutado:

$$\text{Potência} = \text{Carga capacitiva} \times \text{Tensão elétrica}^2 \times \text{Frequência comutada}$$

A frequência comutada é uma função da taxa de clock. A carga capacitiva por transistor é uma função do número de transistores conectados a uma saída (chamado de fanout) e da tecnologia, que determina a capacidade dos fios e dos transistores.

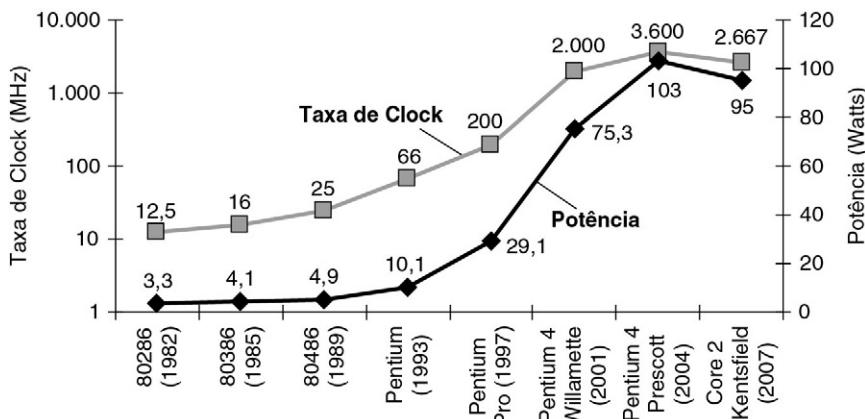


FIGURA 1.15 Taxa de clock e potência para microprocessadores Intel x86 durante oito gerações e 25 anos. O Pentium 4 fez um salto dramático na taxa de clock e potência, porém menor em desempenho. Os problemas térmicos do Prescott levaram ao abandono da linha Pentium 4. A linha Core 2 retorna a uma pipeline mais simples, com menores taxas de clock e múltiplos processadores por chip.

Como as taxas de clock poderiam crescer por um fator de 1.000 enquanto a potência crescia por um fator apenas de 30? A potência pode ser diminuída reduzindo-se a tensão elétrica, o que ocorreu a cada nova geração da tecnologia, e a potência é uma função da tensão elétrica ao quadrado. Normalmente, a tensão elétrica foi reduzida em 15% por geração. Em 20 anos, as tensões passaram de 5V para 1V, motivo pelo qual o aumento na potência é de apenas 30 vezes.

Potência relativa

EXEMPLO

Suponha que tenhamos desenvolvido um novo processador, mais simples, que tem 85% da carga capacitiva do processador mais antigo e mais complexo. Além do mais, considere que ele tenha tensão ajustável, de modo que pode reduzir a tensão em 15% em comparação com o processador B, o que resulta em um encolhimento de 15% na frequência. Qual é o impacto sobre a potência dinâmica?

$$\frac{\text{Potência}_{\text{nova}}}{\text{Potência}_{\text{antiga}}} = \frac{\langle \text{Carga capacitiva} \times 0,85 \rangle \times \langle \text{Tensão} \times 0,85 \rangle^2 \times \langle \text{Frequência comutada} \times 0,85 \rangle}{\text{Carga capacitiva} \times \text{Tensão}^2 \times \text{Frequência comutada}}$$

Assim, a razão de potência é

$$0,85^4 = 0,52$$

Logo, o novo processador usa cerca de metade da potência do processador antigo.

O problema hoje é que reduzir ainda mais a tensão parece causar muito vazamento nos transistores, como torneiras de água que não conseguem ser completamente fechadas. Até mesmo hoje, cerca de 40% do consumo de potência é decorrente de vazamentos. Se os transistores começassem a vazar mais, o processo inteiro poderia se tornar incontrolável.

Para tentar resolver o problema de potência, os projetistas já conectaram grandes dispositivos a fim de aumentar o resfriamento e depois desligaram partes do chip que não são usadas em determinado ciclo de clock. Embora existam muitas maneiras mais dispendiosas de resfriar os chips e, portanto, aumentar a potência para, digamos, 300 watts, essas técnicas são muito caras para computadores de desktop.

RESPOSTA



Como os projetistas de computador bateram contra a barreira da potência, eles precisaram de uma nova maneira de prosseguir e escolheram um caminho diferente do modo como projetaram microprocessadores nos seus primeiros 30 anos.

Detalhamento: Embora a potência dinâmica seja a principal fonte de dissipação de potência na CMOS, a dissipação de potência estática ocorre devido à corrente de vazamento que flui mesmo quando um transistor está desligado. Conforme já mencionamos, o vazamento normalmente é responsável por 40% do consumo de potência em 2008. Assim, aumentar o número de transistores aumenta a dissipação de potência, mesmo que os transistores estejam sempre desligados. Diversas técnicas de projeto e inovações de tecnologia estão sendo implantadas para controlar o vazamento, mas é difícil reduzir mais a tensão.

1.6

Mudança de mares: Passando de processadores para multiprocessadores

O limite de potência forçou uma mudança dramática no projeto dos microprocessadores. A Figura 1.16 mostra a melhoria no tempo de resposta dos programas para microprocessadores de desktop ao longo dos anos. Desde 2002, a taxa reduziu de um fator de 1,5 por ano para um fator de menos de 1,2 por ano.

Em vez de continuar diminuindo o tempo de resposta de um único programa executando num único processador, em 2006 todas as empresas de desktop e servidor estavam usando microprocessadores com múltiplos processadores por chip, em que o benefício normalmente está mais no throughput do que no tempo de resposta. Para reduzir a confusão entre as palavras processador e microprocessador, as empresas se referem aos processadores como “cores” (ou núcleos), e esses microprocessadores são chamados genericamente de microprocessadores “multicore” (ou múltiplos núcleos). Logo, um microprocessador “quadcore” é um chip que contém quatro processadores, ou quatro núcleos.

A Figura 1.17 mostra o número de processadores (núcleos), potência e taxas de clock de microprocessadores recentes. O plano de registro oficial para muitas empresas é dobrar o número de núcleos por microprocessador por geração de tecnologia de semicondutor, que é aproximadamente a cada dois anos (veja Capítulo 7).

No passado, os programadores podiam contar com inovações no hardware, na arquitetura e nos compiladores para dobrar o desempenho de seus programas a cada 18 meses sem ter de mudar uma linha de código. Hoje, para os programadores obterem uma melhoria significativa no tempo de resposta, eles precisam reescrever seus programas de modo que tirem proveito de múltiplos processadores. Além do mais, para obter o benefício histórico de rodar mais rapidamente nos microprocessadores mais novos, os programadores terão de continuar a melhorar o desempenho de seu código à medida que dobra o número de núcleos.

Para reforçar como os sistemas de software e hardware trabalham lado a lado, usamos uma seção especial, Interface hardware/software, no livro inteiro, com a primeira aparecendo logo em seguida. Essas seções resumem ideias importantes nessa interface crítica.

O paralelismo sempre foi fundamental para o desempenho na computação, mas normalmente esteve oculto. O Capítulo 4 explicará sobre o pipelining, uma técnica elegante que roda programas mais rapidamente sobrepondo a execução de instruções. Este é um exemplo de paralelismo em nível de instrução, em que a natureza paralela do hardware é retirada de modo que o programador e o compilador possam pensar no hardware como executando instruções sequencialmente.

Forçar os programadores a estarem cientes do hardware paralelo e reescrever explicitamente seus programas para serem paralelos foi a “terceira trilha” da arquitetura de

Até agora, a maior parte do software tem sido como música escrita para um solista; com a geração atual de chips, estamos adquirindo alguma experiência com duetos e quartetos e outros pequenos grupos; mas compor um trabalho para grande orquestra e coro é um tipo de desafio diferente.

Brian Hayes, *Computing in a Parallel Universe*, 2007.

Interface de hardware/software

computadores, pois empresas no passado, que dependiam dessa mudança no comportamento, fracassaram (veja Seção 7.14 no site). Do ponto de vista histórico, é surpreendente que a indústria inteira de TI tenha apostado seu futuro em que os programadores finalmente passarão com sucesso para a programação explicitamente paralela.

Por que tem sido tão difícil para os programadores escreverem programas explicitamente paralelos? O primeiro motivo é que a programação paralela é, por definição, programação de desempenho, o que aumenta a dificuldade da programação. Não apenas o programa precisa estar correto, solucionar um problema importante e oferecer uma interface útil às pessoas ou outros programas que o chamam, mas ele também precisa ser rápido. Caso contrário, se você não precisasse de desempenho, bastaria escrever um programa sequencial.

O segundo motivo é que rápido, para o hardware paralelo, significa que o programador precisa dividir uma aplicação de modo que cada processador tenha aproximadamente a mesma quantidade de coisas a fazer ao mesmo tempo, e que o overhead do escalonamento e coordenação não afasta os benefícios de desempenho em potencial do paralelismo.

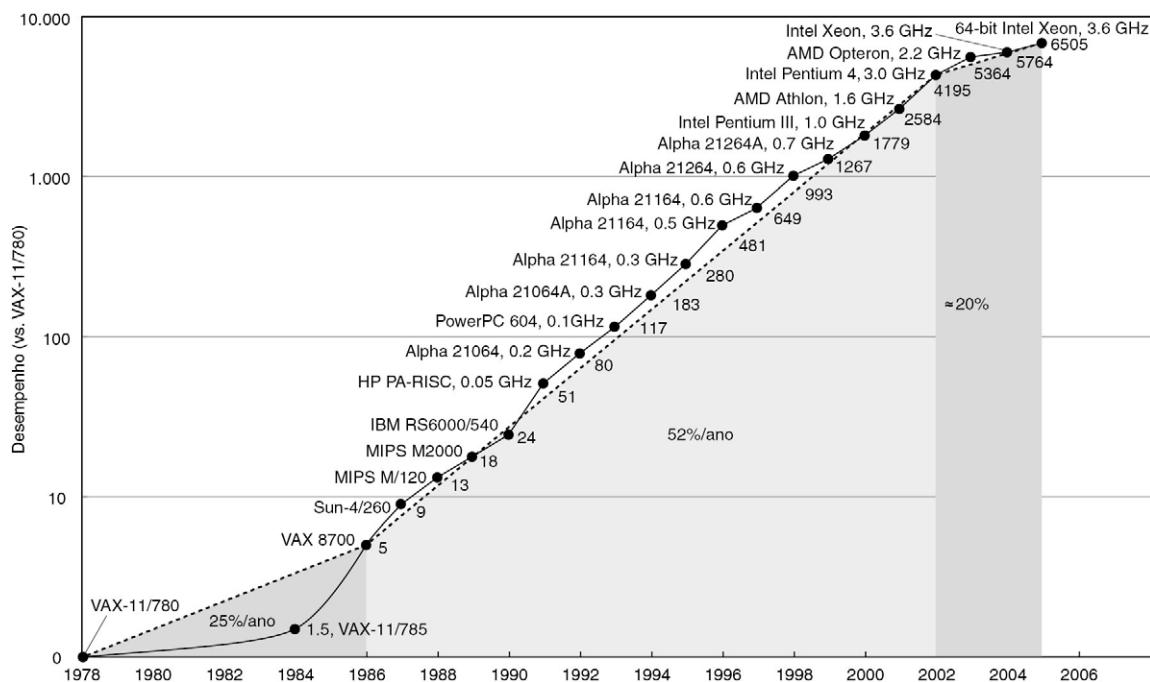


FIGURA 1.16 Crescimento do desempenho do processador desde meados da década de 1980. Este gráfico representa o desempenho relativo ao VAX 11/780 medido pelos benchmarks SPECInt (veja Seção 1.8). Antes de meados da década de 1980, o crescimento do desempenho do processador foi em grande parte controlado pela tecnologia e teve uma média de 25% por ano. O aumento no crescimento para cerca de 52% desde então é atribuído a ideias arquiteturais e organizacionais mais avançadas. Por volta de 2002, esse crescimento levou a uma diferença no desempenho por um fator de cerca de sete. O desempenho para cálculos orientados a ponto flutuante aumentou ainda mais rapidamente. Desde 2002, os limites de potência, o paralelismo disponível em nível de instrução e a latência de memória longa reduziram o desempenho recentemente, para cerca de 20% por ano.

Produto	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Núcleos por chip	4	4	2	8
Taxa de clock	2,5 GHz	~ 2,5 GHz ?	4,7 GHz	1,4 GHz
Potência do microprocessador	120 W	~ 100 W ?	~ 100 W ?	94 W

FIGURA 1.17 Número de núcleos por chip, taxa de clock e potência para os microprocessadores multicore em 2008.

Como uma analogia, suponha que a tarefa fosse escrever um artigo de jornal. Oito repórteres trabalhando no mesmo artigo poderiam potencialmente escrever um artigo oito vezes mais rápido. Para conseguir essa velocidade aumentada, seria preciso desmembrar a tarefa de modo que cada repórter tivesse algo para fazer ao mesmo tempo. Assim, temos de escalarizar as subtarefas. Se algo saísse errado e apenas um repórter levasse mais tempo do que os sete outros levaram, então o benefício de ter oito escritores seria diminuído. Assim, temos de balancear a carga por igual para obter o ganho de velocidade desejado. Outro perigo seria se os repórteres tivessem de gastar muito tempo falando uns com os outros para escrever suas seções. Você também se atrasaria se uma parte do artigo, como a conclusão, não pudesse ser escrita até que todas as outras partes fossem concluídas. Assim, deve-se ter o cuidado para reduzir o overhead de comunicação e sincronização. Para essa analogia e para a programação paralela, os desafios incluem escalarização, balanceamento de carga, tempo para sincronismo e overhead para comunicação entre as partes. Como você poderia imaginar, o desafio é ainda maior com mais repórteres de um artigo de jornal e mais processadores na programação paralela.

Para refletir essa mudança de mares no setor, os próximos cinco capítulos desta edição do livro possuem uma seção sobre as implicações da revolução paralela relacionadas a cada capítulo:

- *Capítulo 2, Seção 2.11: Paralelismo e instruções: sincronização.* Normalmente, tarefas paralelas independentes às vezes precisam ser coordenadas, como dizer quando elas completaram seu trabalho. Esse capítulo explica as instruções usadas por processadores multicore para sincronizar tarefas.
- *Capítulo 3, Seção 3.6: Paralelismo e aritmética de computador: Associatividade.* Em geral, os programadores paralelos começam de um programa sequencial funcionando. Uma questão natural para descobrir se sua versão paralela funciona é: “ela tem a mesma resposta?” Se não, uma conclusão lógica é que existem bugs na nova versão. Essa lógica considera que a aritmética do computador é associativa: você recebe a mesma soma quando adiciona um milhão de números, não importando a ordem. Esse capítulo explica que, embora essa lógica se mantenha para números inteiros, não serve para os números de ponto flutuante.
- *Capítulo 4, Seção 4.10: Paralelismo e paralelismo avançado em nível de instrução.* Dada a dificuldade da programação explicitamente paralela, um esforço tremendo foi investido na década de 1990 para que o hardware e o compilador revelassem o paralelismo implícito. Esse capítulo descreve algumas dessas técnicas agressivas, incluindo a busca e a execução simultâneas de múltiplas instruções e a estimativa dos resultados das decisões, com a execução especulativa das instruções.
- *Capítulo 5, Seção 5.8: Paralelismo e hierarquias de memória: coerência do cache.* Um modo de reduzir o custo da comunicação é fazer com que todos os processadores usem o mesmo espaço de endereço, de modo que qualquer processador possa ler ou gravar quaisquer dados. Visto que todos os processadores atuais utilizam caches para manter uma cópia temporária dos dados na memória mais rápida, mais próxima do processador, é fácil imaginar que a programação paralela seria ainda mais difícil se os caches associados a cada processador tivessem valores inconsistentes dos dados compartilhados. Esse capítulo descreve os mecanismos que mantêm coerentes os dados em todos os caches.
- *Capítulo 6, Seção 6.9: Paralelismo e E/S: Redundant Arrays of Inexpensive Disks.* Se você ignorar a entrada e saída nessa revolução paralela, a consequência não intencionada da programação paralela pode ser fazer com que seu programa paralelo gaste a maior parte do seu tempo esperando pela E/S. Esse capítulo descreve RAID, uma técnica para acelerar o desempenho dos acessos ao armazenamento. RAID assinala outro benefício em potencial do paralelismo: tendo muitas cópias dos recursos, o sistema pode continuar a fornecer serviço apesar de uma falha de um recurso. Logo, RAID pode melhorar tanto o desempenho quanto a disponibilidade da E/S.

Eu acreditava que [os computadores] seriam uma ideia universalmente aplicável, assim como os livros. Só não imaginava que se desenvolveriam tão rapidamente, pois não pensei que fôssemos capazes de colocar tantas peças em um chip quanto finalmente colocamos. O transistor apareceu inesperadamente. Tudo aconteceu muito mais rápido do que esperávamos.

J. Presper Eckert,
coinventor do Eniac, falando
em 1991

Além dessas seções, existe um capítulo inteiro sobre processamento paralelo. O Capítulo 7 entra em mais detalhes sobre os desafios da programação paralela; apresenta as duas técnicas contrastantes para a comunicação de endereçamento compartilhado e passagem explícita de mensagens; descreve o modelo restrito de paralelismo que é mais fácil de programar; discute a dificuldade do benchmarking de processadores paralelos; apresenta um novo modelo de desempenho simples para microprocessadores multicore e finalmente descreve e avalia quatro exemplos de microprocessadores multicore usando esse modelo.

A partir desta edição do livro, o Apêndice A descreve um componente de hardware cada vez mais comum, que está incluído com os computadores de desktop: a unidade de processamento de gráficos (Graphics Processing Unit – GPU). Inventada para acelerar os gráficos, as GPUs estão se tornando plataformas de programação por si sós. Como você poderia esperar, neste momento, as GPUs são altamente paralelas. O Apêndice A descreve a GPU NVIDIA e realça partes de seu ambiente de programação paralelo.

1.7

Vida real: Fabricação e benchmarking do AMD Opteron X4

Cada capítulo possui uma seção intitulada “Vida Real”, que associa os conceitos no livro com um computador que você pode usar em seu dia a dia. Essas seções abordam a tecnologia na qual se baseiam os computadores modernos. Nesta primeira “Vida Real”, veremos como os circuitos integrados são fabricados e como o desempenho e a potência são medidos com o AMD Opteron X4 como exemplo.

Vamos começar do início. A fabricação de um chip começa com o silício, uma substância encontrada na areia. Como não é um bom condutor de eletricidade, é chamado de semicondutor. Com um processo químico especial, é possível acrescentar ao silício materiais que permitem que minúsculas áreas se transformem em um entre três dispositivos:

- Excelentes condutores de eletricidade (usando fios microscópicos de cobre ou alumínio)
- Excelentes isolantes de eletricidade (como cobertura plástica ou vidro)
- Áreas que podem conduzir ou isolar sob condições especiais (como uma chave)

Os transistores se encaixam na última categoria. Um circuito VLSI, então, simplesmente consiste em bilhões de combinações de condutores, isolantes e chaves, fabricados em um único e pequeno pacote.

O processo de fabricação dos circuitos integrados é decisivo para o custo dos chips e, consequentemente, fundamental para os projetistas de computador. A Figura 1.18 mostra esse processo. O processo inicia com um lingote de cristal de silício, que se parece com uma salsicha gigante. Hoje, os lingotes possuem de 20 a 30cm de diâmetro e cerca de 30 a 60cm de comprimento. Um lingote é finamente fatiado em wafers de não mais que 0,25cm de espessura. Esses wafers passam por uma série de etapas de processamento, durante as quais são depositados padrões de elementos químicos em cada lâmina, criando os transistores, os condutores e os isolantes discutidos anteriormente. Os circuitos integrados de hoje contêm apenas uma camada de transistores, mas podem ter de dois a oito níveis de condutor de metal, separados por camadas de isolantes.

Uma única imperfeição microscópica no wafer propriamente dito ou em uma das dezenas de passos da aplicação dos padrões pode resultar na falha dessa área do wafer. Esses defeitos, como são chamados, tornam praticamente impossível fabricar um wafer perfeito. Para lidar com a imperfeição, várias estratégias têm sido usadas, mas a mais simples é colocar muitos componentes independentes em um único wafer. O wafer com os padrões é, então, cortado em seções individuais desses componentes, chamados dies, mais informalmente conhecidos como chips. A Figura 1.19 é uma fotografia de um wafer com mi-

silício Um elemento natural que é um semicondutor.

semicondutor Uma substância que não conduz bem a eletricidade.

lingote de cristal de silício Uma barra composta de um cristal de silício que possui entre 15 e 30cm de diâmetro e cerca de 30 a 60cm de comprimento.

wafer Uma fatia de um lingote de silício de não mais de 2,5mm de espessura, usada para criar chips.

defeito Uma imperfeição microscópica em um wafer ou nos passos da aplicação dos padrões que pode resultar na falha do die que contém esse defeito.

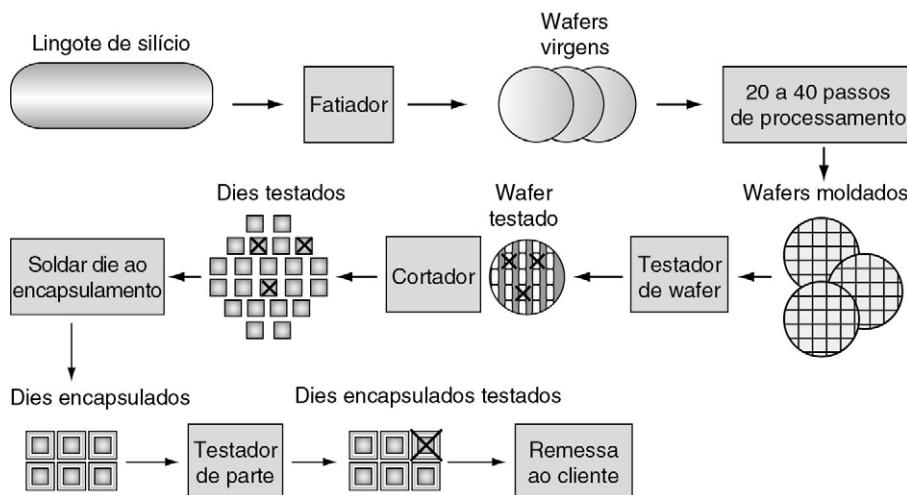


FIGURA 1.18 Processo de fabricação de um chip. Após ser fatiado de um lingote de silício, os wafers virgens passam por 20 a 40 passos para criar wafers com padrões (veja a Figura 1.19). Esses wafers com padrões são testados com um testador de wafers e é criado um mapa das partes boas. Depois, os wafers são divididos em dies (moldes) (veja a Figura 1.9). Nessa figura, um wafer produziu 20 dies, dos quais 17 passaram no teste. (X significa que o die está ruim.) O aproveitamento de dies bons nesse caso foi de 17/20, ou 85%. Esses dies bons são soldados a encapsulamentos e testados outra vez antes de serem remetidos para os clientes. Um die encapsulado ruim foi encontrado nesse teste final.

croprocessadores antes de serem cortados; anteriormente, a Figura 1.9 mostrou um die individual do microprocessador e seus principais componentes.

Cortar os wafers em seções permite descartar apenas aqueles dies que possuem falhas, em vez do wafer inteiro. Esse conceito é quantificado pelo aproveitamento de um processo, definido como a porcentagem de dies bons do número total de dies em um wafer.

O custo de um circuito integrado sobe rapidamente conforme aumenta o tamanho do die, em razão do aproveitamento mais baixo e do menor número de dies que pode caber em um wafer. Para reduzir o custo, um die grande normalmente é “encolhido” usando um processo da próxima geração, que incorpora tamanhos menores de transistores e de fios. Isso melhora o aproveitamento e o número de dies por wafer.

Tendo dies bons, eles são conectados aos pinos de entrada/saída de um encapsulamento usando um processo chamado soldagem. Essas peças encapsuladas são testadas uma última vez, já que podem ocorrer erros no encapsulamento, e são remetidas aos clientes.

Conforme já foi mencionado, outra limitação de projeto cada vez mais importante é o consumo de energia. O consumo é um problema por duas razões. Primeiro, a corrente precisa ser trazida para o chip e distribuída por toda sua área; os microprocessadores modernos usam centenas de pinos apenas para alimentação e aterramento! Da mesma forma, múltiplos níveis de interconexões são usados unicamente para distribuição de corrente e aterramento para as partes do chip. Segundo, a energia é dissipada como calor e precisa ser removida. Um AMD Opteron X4 modelo 2356 a 2,0 GHz produz 120 watts em 2008, que precisam ser removidos de um chip cuja área de superfície é de apenas 1cm²!

dies As seções retangulares individuais cortadas de um wafer, mas informalmente conhecidos como chips.

aproveitamento A porcentagem de dies bons do número total de dies em um wafer.

Detalhamento: O custo de um circuito integrado pode ser expresso em três equações simples:

$$\text{Custo por die} = \frac{\text{Custo por wafer}}{\text{Dies por wafer} \times \text{aproveitamento}}$$

$$\text{Dies por wafer} = \frac{\text{Área do wafer}}{\text{Área do die}}$$

$$\text{Aproveitamento} = \frac{1}{(1 + (\text{Defeitos por área} \times \text{Área do die} / 2))^2}$$

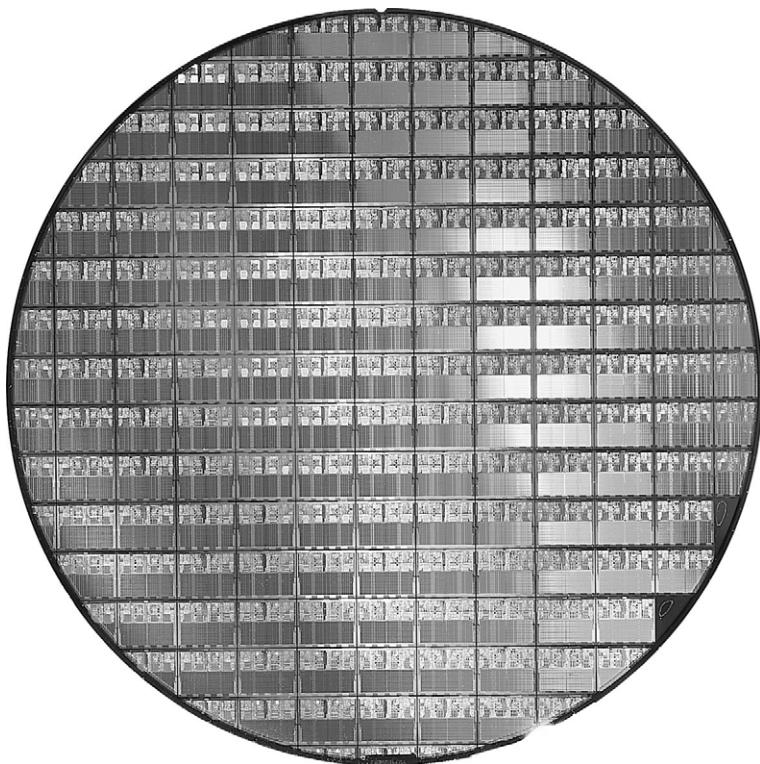


FIGURA 1.19 Um wafer de 300mm de diâmetro dos chips AMD Opteron X2, o predecessor dos chips Opteron X4 (Cortesia da AMD). O número de dies de Pentium por wafer em 100% de aproveitamento é 117. As várias dezenas de chips parcialmente arredondados nas bordas do wafer são inúteis; são incluídas porque é mais fácil criar as máscaras usadas para imprimir os padrões desejados ao silício. Esse die usa uma tecnologia de 90 nanômetros, o que significa que os menores transistores possuem um tamanho de aproximadamente 90 nm, embora normalmente sejam um pouco menores do que o tamanho real catalogado, que se refere ao tamanho dos transistores como “desenhados” versus o tamanho final fabricado.

A primeira equação é simples de se derivar. A segunda é uma aproximação, pois não subtrai a área perto da borda do wafer arredondado que não pode acomodar os dies retangulares (veja Figura 1.19). A equação final é baseada nas observações empíricas dos aproveitamentos nas fábricas de circuito integrado, com o expoente relacionado ao número de etapas de processamento crítico.

Logo, dependendo da taxa de defeito e do tamanho do die e wafer, os custos geralmente não são lineares em relação à área do die.

Benchmark de CPU SPEC

Um usuário de computador que executa os mesmos programas todos os dias seria o candidato perfeito para avaliar um novo computador. O conjunto de programas executados formaria uma carga de trabalho. Para avaliar dois sistemas, um usuário simplesmente compararia o tempo de execução da carga de trabalho nos dois computadores. A maioria dos usuários, porém, não está nessa situação. Em vez disso, eles precisam contar com outros métodos que medem o desempenho de um computador candidato, esperando que os métodos reflitam como o computador funcionará com a carga de trabalho do usuário. Essa alternativa normalmente é seguida pela avaliação do computador usando um conjunto de benchmarks — programas escolhidos especificamente para medir o desempenho. Os benchmarks formam uma carga de trabalho que o usuário acredita que irá prever o desempenho da carga de trabalho real.

O System Performance Evaluation Cooperative (SPEC) é um esforço com patrocínio e suporte de uma série de fornecedores de computador a fim de criar conjuntos padrão de benchmarks para sistemas de computador modernos. Em 1989, o SPEC criou originalmente

carga de trabalho Um conjunto de programas executados em um computador que é a coleção real das aplicações executadas por um usuário ou construídas a partir de programas reais para aproximar tal mistura. Uma carga de trabalho típica especifica o programa e as frequências relativas.

benchmark Um programa selecionado para uso na comparação do desempenho de computadores.

Descrição	Nome	Contagem de instruções × 10 ⁹	CPI	Tempo do ciclo de clock (seg × 10 ⁹)	Tempo de execução (seg)	Tempo de referência (seg)	SPECratio
Processamento de string interpretado	perl	2.118	0,75	0,4	637	9.770	15,3
Compactação de classificação em bloco	bzip2	2.389	0,85	0,4	817	9.650	11,8
Compilador C GNU	gcc	1.050	1,72	0,4	724	8.050	11,1
Otimização combinatória	mcf	336	10,00	0,4	1.345	9.120	6,8
Jogo Go (IA)	go	1.658	1,09	0,4	721	10.490	14,6
Pesquisa de sequência genética	hmmer	2.783	0,80	0,4	890	9.330	10,5
Jogo de xadrez (IA)	sjeng	2.176	0,96	0,4	837	12.100	14,5
Simulação de computador quântico	libquantum	1.623	1,61	0,4	1.047	20.720	19,8
Compactação de vídeo	h264avc	3.102	0,80	0,4	993	22.130	22,3
Biblioteca de simulação de evento discreto	omnetpp	587	2,94	0,4	690	6.250	9,1
Jogos/descoberta de caminho	astar	1.082	1,79	0,4	773	7.020	9,1
Análise XML	xalancbmk	1.058	2,70	0,4	1.143	6.900	6,0
Média geométrica							11,7

FIGURA 1.20 Benchmarks SPECINTC2006 executando no AMD Opteron X4 modelo 2356 (Barcelona). Conforme explica a equação na seção “A equação clássica de desempenho da CPU, anteriormente neste capítulo, o tempo de execução é o produto dos três fatores nesta tabela: contagem de instruções em bilhões, clocks por instrução (CPI) e tempo do ciclo de clock em nanosegundos. SPECratio é simplesmente o tempo de referência, que é fornecido pelo SPEC, dividido pelo tempo de execução medido. O único número mencionado como SPECINTC2006 é a média geométrica dos SPECratios. A Figura 5.40 mostra que mcf, libquantum, omnetpp e xalancbmk possuem CPIs relativamente altos, pois possuem taxas de perda de cache altas.

um conjunto de benchmark focalizando o desempenho do processador (agora chamado SPEC89), que evoluiu por cinco gerações. A mais recente é SPEC CPU2006, que consiste em um conjunto de 12 benchmarks de inteiros (CINT2006) e 17 benchmarks de ponto flutuante (CFP2006). Os benchmarks de inteiros variam desde parte de um compilador C até um programa de xadrez e uma simulação de computador quântico. Os benchmarks de ponto flutuante incluem códigos de grade estruturados para modelagem de elemento finito, códigos de método de partículas para dinâmica molecular e códigos de álgebra linear esparsa para dinâmica de fluidos.

A Figura 1.20 descreve os benchmarks de inteiros SPEC e seu tempo de execução no Opteron X4, mostrando os fatores que explicam o tempo de execução: contagem de instruções, CPI e tempo do ciclo de clock. Observe que o CPI varia por um fator de 13.

Para simplificar o marketing dos computadores, o SPEC decidiu informar um único número para resumir todos os 12 benchmarks de inteiros. As medidas do tempo de execução são primeiro normalizadas dividindo-se o tempo de execução em um processador de referência pelo tempo de execução no computador medido; essa normalização gera uma medida, chamada SPECratio, que tem a vantagem de que resultados numéricos maiores indicam desempenho melhor (ou seja, o SPECratio é o inverso do tempo de execução). Uma medição de resumo CINT2006 ou CFP2006 é obtida apanhando-se a média geométrica dos SPECratios.

Detalhamento: Ao comparar dois computadores usando SPECratios, use a média geométrica, de modo que ela informe a mesma resposta relativa não importa o computador utilizado para normalizar os resultados. Se calculássemos a média dos valores de tempo de execução normalizados com uma média aritmética, os resultados variariam dependendo do computador que escolhêssemos como referência.

A fórmula para a média geométrica é

$$\sqrt[n]{\prod_{i=1}^n \text{Razão do tempo de execução}_i}$$

Carga de destino %	Desempenho (ssj_ops)	Potência média (Watts)
100%	231.867	295
90%	211.282	286
80%	185.803	275
70%	163.427	265
60%	140.160	256
50%	118.324	246
40%	92.035	233
30%	70.500	222
20%	47.126	206
10%	23.066	180
0%	0	141
Soma geral	1.283.590	2.605
$\Sigma \text{ssj_ops} / \Sigma \text{potência}$ =		493

FIGURA 1.21 SPECpower_ssj2008 executando no AMD Opteron X4 2345 (Barcelona) a 2,3 GHz e soquete dual com 16 GB de DRAM DDR2-667 e um disco de 500 GB.

em que Razão do tempo de execução é o tempo de execução, normalizado ao computador de referência, para o iº programa de um total de n na carga de trabalho, e

$$\prod_{i=1}^n a_i \text{ significa o produto } a_1 \times a_2 \times \dots \times a_n$$

Benchmark de potência SPEC

Hoje, o SPEC oferece uma dúzia de conjuntos de benchmark diferentes, projetados para testar uma grande variedade de ambientes de computação usando aplicações reais e regras de execução e requisitos de relatório estritamente especificados. O mais recente é o SPECpower. Ele informa o consumo de potência dos servidores em diferentes níveis de carga de trabalho, dividido em incrementos de 10%, por um período de tempo. A Figura 1.21 mostra os resultados para um servidor usando o Barcelona.

SPECpower começou com o benchmark SPEC para aplicações comerciais em Java (SPECJBB2005), que exerce processadores, caches e memória principal, além da máquina virtual Java, compilador, coletor de lixo e partes do sistema operacional. O desempenho é medido no throughput e as unidades são operações de negócios por segundo. Mais uma vez, para simplificar o marketing dos computadores, o SPEC resume esses números em um único número, chamado “ssj_ops geral por Watt”. A fórmula para essa única métrica de resumo é

$$\text{ssj_ops geral por Watt} = \left(\sum_{i=0}^{10} \text{ssj_ops}_i \right) / \left(\sum_{i=0}^{10} \text{potência}_i \right)$$

em que ssj_opsi é o desempenho em cada incremento de 10% e potência_i é a potência consumida em cada nível de desempenho.

Verifique você mesmo

Um fator fundamental para determinar o custo de um circuito integrado é o volume de produção. Quais das seguintes afirmativas são razões para um chip fabricado com alto volume de produção custar menos?

1. Com altos volumes de produção, o processo de fabricação pode ser transformado em um projeto particular, aumentando o aproveitamento.
2. É menos trabalhoso projetar uma peça com alto volume de produção do que uma com baixo volume de produção.

3. Como as máscaras usadas para fabricar o chip são caras, o custo por chip é menor para volumes de produção mais altos.
4. Os custos de desenvolvimento de engenharia são altos e quase sempre independem do volume de produção; portanto, o custo de desenvolvimento por die é menor com peças de alto volume de produção.
5. Peças de alto volume de produção normalmente possuem dies menores do que as peças de baixo volume de produção e, portanto, têm um aproveitamento mais alto por wafer.

1.8

Falácia e armadilhas

A ciência deve começar com os mitos e com a análise dos mitos.

Sir Karl Popper,
The Philosophy of Science, 1957

A finalidade de uma seção de falácia e armadilhas, que será incluída em cada capítulo, é explicar alguns conceitos errôneos comuns que você pode encontrar. Chamamos esses equívocos de falácia. Quando estivermos discutindo uma falácia, tentaremos fornecer um contraexemplo. Também discutiremos armadilhas ou erros facilmente cometidos. Em geral, as armadilhas são generalizações de princípios verdadeiros em um contexto restrito. O propósito dessas seções é ajudar a evitar esses erros nas máquinas que você pode projetar ou usar. Falácia e armadilhas de custo/desempenho têm confundido muitos arquitetos de computador, incluindo nós. Consequentemente, esta seção não poupa exemplos relevantes. Vamos começar com uma armadilha que engana muitos projetistas e revela um relacionamento importante no projeto de computadores.

Armadilha: Esperar que a melhoria de um aspecto de um computador aumente o desempenho geral por uma quantidade proporcional ao tamanho da melhoria.

Essa melhoria tem visitado os projetistas de hardware e de software. Um problema de projeto simples ilustra isso muito bem. Suponha que um programa execute em 100 segundos em um computador, com operações de multiplicação responsáveis por 80 segundos desse tempo. Quanto terei de melhorar a velocidade da multiplicação se eu quiser que meu programa execute cinco vezes mais rápido?

O tempo de execução do programa depois de fazer a melhoria é dado pela seguinte equação simples, conhecida como lei de Amdahl:

Tempo de execução após o aprimoramento =

$$\frac{\text{Tempo de execução afetado pelo aprimoramento}}{\text{Quantidade de aprimoramento}} + \text{Tempo de execução não afetado}$$

Para este problema:

$$\text{Tempo de execução após o aprimoramento} = \frac{80 \text{ seg}}{n} + (100 - 80 \text{ segundos})$$

Como queremos que o desempenho seja cinco vezes mais rápido, o novo tempo de execução deverá ser 20 segundos, gerando

$$20 \text{ seg} = \frac{80 \text{ seg}}{n} + 20 \text{ seg}$$

$$0 = \frac{80 \text{ seg}}{n}$$

lei de Amdahl Uma regra indicando que a melhoria de desempenho possível com determinado aprimoramento é limitada pela quantidade com que o recurso aprimorado é utilizado. Essa é uma versão quantitativa da lei dos retornos decrescentes.

Ou seja, não existe quantidade pela qual podemos melhorar a multiplicação para conseguir um aumento quíntuplo no desempenho, se a multiplicação é responsável por apenas 80% da carga de trabalho.

A melhoria de desempenho possível com determinado aprimoramento é limitada pela quantidade com que o recurso aprimorado é utilizado. Esse conceito também gera o que chamamos de lei dos retornos decrescentes na vida diária.

Podemos usar a lei de Amdahl para estimar os aprimoramentos no desempenho quando sabemos o tempo consumido para alguma função e seu ganho de velocidade em potencial. A lei de Amdahl, junto com a equação de desempenho da CPU, é uma ferramenta prática para avaliar melhorias em potencial. A lei de Amdahl é explorada com mais detalhes nos exercícios.

Um tema comum no projeto do hardware é um corolário da lei de Amdahl: torne mais rápido o caso comum. Essa orientação simples nos faz lembrar que, em muitos casos, a frequência com que um evento ocorre pode ser muito mais alta do que a frequência de outro evento. A lei de Amdahl nos lembra que a oportunidade para melhoria é afetada por quanto tempo o evento consome. Assim, tornar o caso comum rápido tenderá a melhorar o desempenho mais do que otimizar o caso raro. Ironicamente, o caso comum normalmente é mais simples do que o caso raro, e, portanto, normalmente é mais fácil de melhorar.

A lei de Amdahl também é usada para se demonstrar limites práticos do número de processadores paralelos. Examinamos esse argumento na seção de Falácias e Armadilhas do Capítulo 7.

Falácia: Os computadores com pouca utilização demandam menos potência.

A eficiência de potência importa em baixas utilizações, pois as cargas de trabalho do servidor variam. A utilização de CPU para os servidores no Google, por exemplo, está entre 10% e 50% na maior parte do tempo e em 100% em menos de 1% do tempo. A Figura 1.22 mostra a potência para os servidores com os melhores resultados do SPECpower em 100% de carga, 50% de carga, 10% de carga e ocioso. Até mesmo os servidores com apenas 10% de utilização se queimam com cerca de dois terços de sua potência máxima.

Como as cargas de trabalho dos servidores variam, mas utilizam uma grande fração da potência máxima, Luiz Barroso e Urs Hözle [2007] argumentam que deveríamos reprojetar o hardware para alcançar a “computação proporcional à energia”. Se os servidores futuros usassem, digamos, 10% da potência máxima a 10% de carga de trabalho, poderíamos reduzir a conta de eletricidade dos centros de dados e nos tornarmos bons cidadãos corporativos em uma era de preocupação crescente com as emissões de CO₂.

Fabricante do servidor	Microprocessador	Total de núcleos/soquetes	Taxa de clock	Desempenho Máximo (ssj_ops)	Potência com 100% de carga	Potência com 50% de carga	50% de carga/100% de potência	Potência com 10% de carga	10% de carga/100% de potência	Potência ociosa ativa	Ocioso ativo/100% de potência
HP	Xeon E5440	8/2	3,0 GHz	308.022	269 W	227 W	84%	174 W	65%	160 W	59%
Dell	Xeon E5440	8/2	2,8 GHz	305.413	276 W	230 W	83%	173 W	63%	157 W	57%
Fujitsu Siemens	Xeon X3220	4/1	2,4 GHz	143.742	132 W	110 W	83%	85 W	65%	80 W	60%

FIGURA 1.22 Resultados do SPECPower para três servidores com o melhor ssj_ops geral por watt no quarto trimestre de 2007. O ssj_ops geral por watt dos três servidores são 698, 682 e 677, respectivamente. A memória para os dois primeiros servidores é de 16 GB e do último é 8 GB.

Armadilha: Usar um subconjunto da equação de desempenho como uma métrica de desempenho.

Já mostramos a falácia de prever o desempenho com base simplesmente na taxa de clock, ou na contagem de instruções ou no CPI. Outro erro comum é usar apenas dois dos três fatores para comparar o desempenho. Embora o uso de dois dos três fatores possa ser válido em um contexto limitado, o conceito facilmente também é mal utilizado. Sem dúvida, quase

todas as alternativas propostas para o uso do tempo como métrica de desempenho por fim levaram a afirmações enganosas, resultados distorcidos ou interpretações incorretas.

Uma alternativa ao tempo é o milhões de instruções por segundo (MIPS). Para determinado programa, o MIPS é simplesmente

$$\text{MIPS} = \frac{\text{Contagem de instruções}}{\text{Tempo de execução} \times 10^6}$$

Como MIPS é uma taxa de execução de instruções, MIPS especifica o desempenho inversamente ao tempo de execução; computadores mais rápidos possuem uma taxa de MIPS mais alta. A boa notícia sobre MIPS é que ele é fácil de entender e computadores mais rápidos significam um MIPS maior, que corresponde à intuição.

Existem três problemas com o uso do MIPS como uma medida para comparar computadores. Primeiro, MIPS especifica a taxa de execução de instruções, mas não leva em conta as capacidades das instruções. Não podemos comparar computadores com diferentes conjuntos de instruções usando MIPS, pois as contagens de instruções certamente serão diferentes. Segundo, MIPS varia entre os programas no mesmo computador; assim, um computador não pode ter uma única avaliação MIPS. Por exemplo, substituindo o tempo de execução, vemos o relacionamento entre MIPS, taxa de clock e CPI:

$$\text{MIPS} = \frac{\text{Contagem de instruções}}{\frac{\text{Contagem de instruções} \times \text{CPI}}{\text{Taxa de clock}} \times 10^6} = \frac{\text{Taxa de clock}}{\text{CPI} \times 10^6}$$

Lembre-se de que o CPI variou em 13× para SPEC2006 no Opteron X4, de modo que o MIPS também varia. Finalmente, e mais importante, se um novo programa executa mais instruções, mas cada instrução é mais rápida, o MIPS pode variar independentemente do desempenho!

Considere as seguintes medidas de desempenho para um programa:

Medida	Computador A	Computador B
Número de instruções	10 bilhões	8 bilhões
Taxa de clock	4 GHz	4 GHz
CPI	1,0	1,1

- Que computador tem a avaliação MIPS mais alta?
- Qual computador é mais rápido?

milhões de instruções por segundo (MIPS) Uma medida da velocidade de execução do programa baseada no número de milhões de instruções. MIPS é calculado como a contagem de instruções dividida pelo produto do tempo de execução e 106.

Verifique você mesmo

1.9

Comentários finais

Embora seja difícil prever exatamente o nível de custo/desempenho que os computadores terão no futuro, é seguro dizer que serão muito melhores do que são hoje. Para participar desses avanços, os projetistas e programadores de computador precisam entender várias questões.

Os projetistas de hardware e de software constroem sistemas computacionais em camadas hierárquicas; cada camada inferior oculta seus detalhes do nível acima. Esse princípio de abstração é fundamental para compreender os sistemas computacionais atuais, mas isso não significa que os projetistas podem se limitar a conhecer uma única tecnologia. Talvez o exemplo mais importante de abstração seja a interface entre hardware e software de baixo

Enquanto o Eniac é equipado com 18.000 válvulas e pesa 30 toneladas, os computadores no futuro poderão ter 1.000 válvulas e talvez pesar apenas 1,5 tonelada.

Popular Mechanics,
março de 1949

nível, chamada arquitetura do conjunto de instruções. Manter a arquitetura do conjunto de instruções como uma constante permite que muitas implementações dessa arquitetura – provavelmente variando em custo e desempenho – executem software idêntico. No lado negativo, a arquitetura pode impedir a introdução de inovações que exijam a mudança da interface.

Existe um método confiável para determinar e informar o desempenho usando o tempo de execução dos programas reais como métrica. Esse tempo de execução está relacionado a outras medições importantes que podemos fazer pela seguinte equação:

$$\frac{\text{Segundos}}{\text{Programa}} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Ciclos de clock}}{\text{Instrução}} \times \frac{\text{Segundos}}{\text{Ciclo de clock}}$$

Usaremos essa equação e seus fatores constituintes muitas vezes. Lembre-se, porém, de que individualmente os fatores não determinam o desempenho: somente o produto, que é igual ao tempo de execução, é uma medida confiável do desempenho.

Colocando em perspectiva

O tempo de execução é a única medida válida e incontestável do desempenho. Muitas outras métricas foram propostas e desapareceram. Às vezes, essas métricas possuem falhas desde o início, não refletindo o tempo de execução; outras vezes, uma métrica que é válida em um contexto limitado é estendida e usada além desse contexto ou sem o esclarecimento adicional necessário para torná-la válida.

As tecnologias vitais para os processadores modernos são os compiladores e o silício. De igual importância para uma compreensão da tecnologia de circuito integrado é o conhecimento das taxas de mudança tecnológica esperadas. Enquanto o silício impulsiona o rápido avanço do hardware, novas ideias na organização dos computadores melhoraram seu custo/desempenho. Duas das principais ideias são a exploração do paralelismo no programa, normalmente por meio de processadores múltiplos, e a exploração da localidade dos acessos a uma hierarquia de memória, em geral por meio de caches.

A potência substituiu a área do die como o recurso mais crítico do projeto de microprocessadores. Conservar energia enquanto se tenta aumentar o desempenho tem迫使ido o setor de hardware a passar para microprocessadores multicore, forçando, assim, o setor de software a passar para a programação do hardware em paralelo.

Os projetos de computadores sempre foram medidos pelo custo e desempenho, além de outros fatores importantes, como potência, confiabilidade, custo de proprietário e escalabilidade (ou facilidade de expansão). Embora este capítulo tenha focalizado o custo, o desempenho e a potência, os melhores projetos buscarão o equilíbrio apropriado para determinado mercado entre todos esses fatores.

Mapa para este livro

Na base dessas abstrações estão os cinco componentes clássicos de um computador: caminho de dados, controle, memória, entrada e saída (veja novamente a [Figura 1.4](#)). Esses cinco componentes também servem de estrutura para os demais capítulos do livro:

- Caminho de dados: Capítulos 3, 5, 7 e Apêndice A
- Controle: Capítulos 4, 7 e Apêndice A
- Memória: Capítulo 5
- Entrada: Capítulo 6
- Saída: Capítulo 6

Como dissemos, o Capítulo 4 descreve como os processadores exploram o paralelismo implícito; o Capítulo 7 descreve os microprocessadores multicore explicitamente paralelos, que estão no núcleo da revolução paralela; e o Apêndice A descreve o chip de processador gráfico altamente paralelo. O Capítulo 5 descreve como a hierarquia de memória explora a localidade. O Capítulo 2 descreve os conjuntos de instruções – a interface entre os compiladores e a máquina – e destaca o papel dos compiladores e das linguagens de programação ao usar os recursos do conjunto de instruções. O Apêndice B oferece uma referência para o conjunto de instruções do Capítulo 2. O Capítulo 3 descreve como os computadores realizam operações aritméticas. ☈ O Apêndice C, no site, apresenta o projeto lógico.

1.10

Perspectiva histórica e leitura adicional

Para cada capítulo, há uma seção dedicada à perspectiva histórica que pode ser encontrada no site que acompanha este livro. Podemos traçar o desenvolvimento de uma ideia por meio de uma série de máquinas ou descrever alguns projetos importantes; e fornecemos referências, caso você esteja interessado em pesquisar mais a fundo.

Esta perspectiva histórica desse capítulo fornece uma base para algumas das principais ideias apresentadas neste capítulo de abertura. Sua finalidade é apresentar a história humana por trás dos avanços tecnológicos e colocar as realizações dentro de seu contexto histórico. Entendendo o passado, você pode compreender melhor as forças que formarão a computação no futuro. Cada seção de perspectiva histórica no site termina com sugestões para leitura adicional, que também são coletadas separadamente no site na seção “Further Reading”. O restante da ☈ Seção 1.10 está no site.

Um campo ativo da ciência é como um imenso formigueiro; a pessoa quase desaparece na massa de mentes afundando umas sobre as outras, carregando informações de um lugar para outro, passando-as adiante na velocidade da luz.

Lewis Thomas, “Natural Science”, em *The Lives of a Cell*, 1974

1.11

Exercícios¹

A maioria dos exercícios nesta edição é projetada de modo que apresente uma descrição qualitativa com o apoio de uma tabela que oferece parâmetros quantitativos alternativos. Esses parâmetros são necessários para solucionar as perguntas que compreendem o exercício. Perguntas individuais podem ser solucionadas usando-se qualquer um ou todos os parâmetros — você decide quantos dos parâmetros deverão ser considerados para qualquer pergunta do exercício. Por exemplo, é possível dizer “complete a Pergunta 4.1.1 usando os parâmetros dados na linha A da tabela”. Como alternativa, os instrutores podem personalizar esses exercícios para criar novas soluções, substituindo os parâmetros indicados pelos seus próprios valores exclusivos.

As avaliações do tempo relativo à solução dos exercícios são mostradas entre colchetes após cada número de exercício. Em média, um exercício avaliado em [10] levará o dobro do tempo de um avaliado em [5]. As seções do texto, que devem ser lidas antes de resolver um exercício, serão indicadas entre sinais de maior e menor; por exemplo, <1.3> significa que você deve ler a Seção 1.3, “Sob as tampas”, para ajudar a resolver esse exercício.

Exercício 1.1

Encontre a palavra ou frase da seguinte lista que melhor corresponde à descrição nas questões a seguir. Use os números à esquerda das palavras na resposta. Cada resposta deve ser usada apenas uma vez.

¹ Contribuição de Javier Bruguera da Universidade de Santiago de Compostela.

1.	mundos virtuais	14.	sistema operacional
2.	computadores desktop	15.	compilador
3.	servidores	16.	bit
4.	servidores inferiores	17.	instrução
5.	supercomputadores	18.	linguagem assembly
6.	terabyte	19.	linguagem de máquina
7.	petabyte	20.	C
8.	centros de dados	21.	assembler
9.	computadores embutidos	22.	linguagem de alto nível
10.	processadores multicore	23.	software do sistema
11.	VHDL	24.	software de aplicação
12.	RAM	25.	cobol
13.	CPU	26.	fortran

1.1.1 [2] <1.1> Computador usado para executar grandes problemas e normalmente acessado por meio de uma rede

1.1.2 [2] <1.1> 1015 ou 250 bytes

1.1.3 [2] <1.1> Computador composto de centenas a milhares de processadores e terabytes de memória

1.1.4 [2] <1.1> Aplicação atual da ficção científica que provavelmente estará disponível no futuro próximo

1.1.5 [2] <1.1> Um tipo de memória chamada memória de acesso aleatório

1.1.6 [2] <1.1> Parte de um computador, chamada unidade central de processamento

1.1.7 [2] <1.1> Milhares de processadores formando um grande cluster

1.1.8 [2] <1.1> Um microprocessador contendo vários processadores no mesmo chip

1.1.9 [2] <1.1> Computador desktop sem a tela ou teclado, normalmente acessado por uma rede

1.1.10 [2] <1.1> Atualmente a maior classe de computador que executa uma aplicação ou um conjunto de aplicações relacionadas

1.1.11 [2] <1.1> Linguagem especial usada para descrever componentes de hardware

1.1.12 [2] <1.2> Computador pessoal que oferece bom desempenho para usuários isolados a um baixo custo

1.1.13 [2] <1.2> Programa que traduz instruções na linguagem de alto nível para a linguagem assembly

1.1.14 [2] <1.2> Programa que traduz instruções simbólicas para instruções binárias

1.1.15 [2] <1.2> Linguagem de alto nível para processamento de dados comerciais

1.1.16 [2] <1.2> Linguagem binária que o processador pode entender

1.1.17 [2] <1.2> Comandos que os processadores entendem

1.1.18 [2] <1.2> Linguagem de alto nível para computação científica

1.1.19 [2] <1.2> Representação simbólica das instruções de máquina

1.1.20 [2] <1.2> Interface entre o programa do usuário e o hardware, oferecendo uma série de serviços e funções de supervisão

1.1.21 [2] <1.2> Software/programas desenvolvidos pelos usuários

1.1.22 [2] <1.2> Dígito binário (valor 0 ou 1)

1.1.23 [2] <1.2> Camada de software entre o software de aplicação e o hardware, que inclui o sistema operacional e os compiladores

1.1.24 [2] <1.2> Linguagem de alto nível usada para escrever software de aplicação e de sistemas

1.1.25 [2] <1.2> Linguagem portável, composta de palavras e expressões algébricas, que precisa ser traduzida para a linguagem assembly antes de ser executada em um computador

1.1.26 [2] <1.2> 1012 ou 240 bytes

Exercício 1.2

Considere as diferentes configurações mostradas na tabela seguinte.

	Configuração	Resolução	Memória Principal	Rede Ethernet
a.	1	640 x 480	2 GB	100 Mbit
	2	1280 x 1024	4 GB	1 Gbit
b.	1	1024 x 768	2 GB	100 Mbit
	2	2560 x 1600	4 GB	1 Gbit

1.2.1 [10] <1.3> Para uma tela colorida usando 8 bits para cada uma das cores primárias (vermelho, verde, azul) por pixel e com uma resolução de 1280×800 pixels, qual deve ser o tamanho (em bytes) do buffer de frame a fim de armazenar um frame?

1.2.2 [5] <1.3> Se um computador tem uma memória principal de 2 GB, quantos frames ele poderia armazenar, supondo que a memória não contém outra informação?

1.2.3 [5] <1.3> Se um arquivo de 256 KB for enviado por uma rede Ethernet, quanto tempo levará para chegar?

Para os problemas abaixo, utilize as informações da tabela abaixo para o tempo de acesso para cada tipo de memória.

	Cache	DRAM	Memória Flash	Disco magnético
a.	5 ns	50 ns	5 μ s	5 ms
b.	7 ns	70 ns	15 μ s	20 ms

1.2.4 [5] <1.3> Descubra quanto tempo é necessário para ler um arquivo de uma memória DRAM se a memória cache demora 2 microsegundos para isso.

1.2.5 [5] <1.3> Descubra quanto tempo é necessário para ler um arquivo de um disco magnético se a memória cache demora 2 microsegundos para isso.

1.2.6 [5] <1.3> Descubra quanto tempo é necessário para ler um arquivo de uma memória flash se a memória cache demora 2 microsegundos para isso.

Exercício 1.3

Considere os três diferentes processadores P1, P2 e P3 executando o mesmo conjunto de instruções com as taxas de clock e CPIs dadas na tabela a seguir.

	Processador	Taxa de clock	CPI
a.	P1	3 GHz	1,5
	P2	2,5 GHz	1,0
	P3	4 GHz	2,2
b.	P1	2 GHz	1,2
	P2	3 GHz	0,8
	P3	4 GHz	2,0

1.3.1 [5] <1.4> Qual processador possui o desempenho mais rápido expressado pelas instruções por segundo?

1.3.2 [10] <1.4> Se cada processador executa um programa em 10 segundos, encontre o número de ciclos e o número de instruções.

1.3.3 [10] <1.4> Ao tentar reduzir o tempo em 30%, a CPI aumenta em 20%. Qual a taxa de clock que deve ser utilizada para a redução de tempo?

Para os problemas abaixo, utilize as informações da tabela seguinte.

	Processador	Taxa de clock	Número de instruções	Tempo
a.	P1	3 GHz	20×10^9	7 s
	P2	2,5 GHz	30×10^9	10 s
	P3	4 GHz	90×10^9	9 s
b.	P1	2 GHz	20×10^9	5 s
	P2	3 GHz	30×10^9	8 s
	P3	4 GHz	25×10^9	7 s

1.3.4 [10] <1.4> Ache instruções por ciclos (IPC) para cada processador.

1.3.5 [5] <1.4> Ache a taxa de clock para P2 que reduz o tempo de execução para o P1.

1.3.6 [5] <1.4> Ache o número de instruções para P2 que reduz o tempo de execução para o P3.

Exercício 1.4

Considere duas implementações diferentes da mesma arquitetura do conjunto de instruções. Existem quatro classes de instruções: A, B, C e D. A taxa de clock e o CPI de cada implementação são dados na tabela a seguir.

Processador	Taxa de clock	CPI Classe A	CPI Classe B	CPI Classe C	CPI Classe D
P1	1,5 GHz	1	2	3	4
P2	2 GHz	2	2	2	2

1.4.1 [10] <1.4> Dado um programa com 106 instruções divididas em classes das seguintes formas: 10% classe A, 20% classe B, 50% classe C e 20% classe D, que implementação é mais rápida?

1.4.2 [5] <1.4> Qual é o CPI global para cada implementação?

1.4.3 [5] <1.4> Ache os ciclos de clock exigidos nos dois casos.

A tabela a seguir mostra o número de instruções para um programa.

Aritmética	Store	Load	Desvio	Total
500	50	100	50	700

1.4.4 [5] <1.4> Considerando que as instruções aritméticas levam 1 ciclo, load e store 5 ciclos e desvio 2 ciclos, qual é o tempo de execução do programa em um processador de 2 GHz?

1.4.5 [5] <1.4> Ache o CPI para o programa.

1.4.6 [10] <1.4> Se o número de instruções de carga puder ser reduzido pela metade, qual é o ganho de velocidade e o CPI?

Exercício 1.5

Considere duas implementações diferentes, P1 e P2, do mesmo conjunto de instruções. Existem cinco classes de instruções (A, B, C, D e E) no conjunto de instruções. A taxa de clock e o CPI de cada classe são dados a seguir.

		Taxa de clock	CPI Classe A	CPI Classe B	CPI Classe C	CPI Classe D	CPI Classe E
a.	P1	1,0 GHz	1	2	3	4	3
	P2	1,5 GHz	2	2	2	4	4
b.	P1	1,0 GHz	1	1	2	3	2
	P2	1,5 GHz	1	2	3	4	3

1.5.1 [5] <1.4> Suponha que o desempenho de pico seja definido como a taxa mais rápida que um computador pode executar qualquer sequência de instruções. Quais são os desempenhos de pico de P1 e P2 expressos em instruções por segundo?

1.5.2 [5] <1.4> Se o número de instruções executadas em um certo programa for dividido igualmente entre as classes de instruções exceto, para a classe A, que ocorre com o dobro da frequência das outras, qual computador é o mais rápido? O quanto ele é mais rápido?

1.5.3 [5] <1.4> Se o número de instruções executadas em um certo programa é dividido igualmente entre as classes de instruções, exceto para a classe E, que ocorre com o dobro da frequência das outras, qual computador é o mais rápido? O quanto ele é mais rápido?

A tabela a seguir mostra o desmembramento de tipo de instrução para diferentes programas. Usando esses dados, você estará explorando as opções de desempenho com diferentes mudanças feitas em um processador MIPS.

		Número de Instruções				
		Cálculo	Load	Store	Desvio	Total
a.	Programa 1	1000	400	100	50	1550
b.	Programa 4	1500	300	100	100	2000

1.5.4 [5] <1.4> Supondo que os cálculos usem 1 ciclo, instruções de load e store usem 10 ciclos e desvios usem 3 ciclos, ache o tempo de execução de cada programa em um processador MIPS de 3 GHz.

1.5.5 [5] <1.4> Supondo que os cálculos usem 1 ciclo, instruções de load e store usem 2 ciclos e desvios usem 3 ciclos, ache o tempo de execução de cada programa em um processador MIPS de 3 GHz.

1.5.6 [5] <1.4> Supondo que os cálculos usem 1 ciclo, instruções de load e store usem 2 ciclos e desvios usem 3 ciclos, qual é o ganho de velocidade de um programa se o número de instruções de cálculo puder ser reduzido pela metade?

Exercício 1.6

Os compiladores podem ter um impacto profundo sobre o desempenho de uma aplicação em determinado processador. Este problema explorará o impacto que os compiladores têm sobre o tempo de execução.

		CPI Classe A	CPI Classe B	CPI Classe C	CPI Classe D	CPI Classe E
a.	P1	1	2	3	4	5
	P2	3	3	3	5	5
b.	P1	1	2	3	4	5
	P2	2	2	2	2	6

1.6.1 [5] <1.4> Para o mesmo programa, dois compiladores diferentes são utilizados. Essa tabela mostra o tempo de execução dos dois programas compilados diferentes. Ache o CPI médio para cada programa dado que o processador tem um tempo de ciclo de clock de 1 nS.

1.6.2 [5] <1.4> Considere os CPIs médios encontrados em 1.6.1, mas que os programas compilados executem em dois processadores diferentes. Se os tempos de execução nos dois processadores forem os mesmos, o quanto mais rápido é o clock do processador rodando o código do compilador A versus o clock do processador rodando o código do compilador B?

1.6.3 [5] <1.4> Um novo compilador é desenvolvido, usando apenas 600 milhões de instruções, e tendo um CPI médio de 1,1. Qual é o ganho de velocidade do uso desse novo compilador versus o uso do Compilador A ou B no processador original de 1.6.1?

Considere duas implementações diferentes, P1 e P2, do mesmo conjunto de instruções. Existem cinco classes de instruções (A, B, C, D e E) no conjunto de instruções. P1 tem uma taxa de clock de 4 GHz, e P2 tem uma taxa de clock de 6 GHz. Os números médios de ciclos para cada classe de instruções para P1 e P2 são listados na tabela a seguir.

	Classe	CPI em P1	CPI em P2
a.	A	1	2
	B	2	2
	C	3	2
	D	4	4
	E	5	4
	Classe	CPI em P1	CPI em P2
b.	A	1	2
	B	1	2
	C	1	2
	D	4	4
	E	5	4

1.6.4 [5] <1.4> Suponha que o desempenho de pico seja definido como a taxa mais rápida que um computador pode executar qualquer sequência de instruções. Quais são os desempenhos de pico de P1 e P2 expressos em instruções por segundo?

1.6.5 [5] <1.4> Se o número de instruções executadas em um certo programa for dividido igualmente entre as classes de instruções no Problema 2.36.4, exceto para a classe A, que ocorre com o dobro da frequência das outras, o quão mais rápido é P2 em relação a P1?

1.6.6 [5] <1.4> Em que frequência, P2 tem o mesmo desempenho de P1 para o mix de instruções dado em 1.6.5?

Exercício 1.7

A tabela a seguir mostra o aumento na taxa de clock e potência de oito gerações de processadores Intel durante 28 anos.

Processador	Taxa de clock	Potência
80286 (1982)	12,5 MHz	3,3 W
80386 (1985)	16 MHz	4,1 W
80486 (1989)	25 MHz	4,9 W
Pentium (1993)	66 MHz	10,1 W
Pentium Pro (1997)	200 MHz	29,1 W
Pentium 4 Willamette (2001)	2 GHz	75,3 W
Pentium 4 Prescott (2004)	3,6 GHz	103 W
Core 2 Ketsfield (2007)	2,667 GHz	95 W

1.7.1 [5] <1.5> Qual é a média geométrica das razões entre as gerações consecutivas para taxa de clock e potência? (A média geométrica é descrita na Seção 1.7.)

1.7.2 [5] <1.5> Qual é a maior mudança relativa na taxa de clock e potência entre as gerações?

1.7.3 [5] <1.5> O quão maior é a taxa de clock e potência da última geração com relação à primeira geração?

Considere os valores a seguir para a voltagem em cada geração.

Processador	Tensão
80286 (1982)	5
80386 (1985)	5
80486 (1989)	5
Pentium (1993)	5
Pentium Pro (1997)	3,3
Pentium 4 Willamette (2001)	1,75
Pentium 4 Prescott (2004)	1,25
Core 2 Ketsfield (2007)	1,1

1.7.4 [5] <1.5> Ache a média das cargas capacitivas, considerando um consumo de energia estática desprezível.

1.7.5 [5] <1.5> Ache a maior mudança relativa em tensão entre as gerações.

1.7.6 [5] <1.5> Ache a média geométrica das razões de tensão nas gerações desde o Pentium.

Exercício 1.8

Suponha que tenhamos desenvolvido novas versões de um processador com as características a seguir.

	Versão	Tensão	Taxa de clock
a.	Versão 1	1,75 V	1,5 GHz
	Versão 2	1,2 V	2 GHz
b.	Versão 1	1,1 V	3 GHz
	Versão 2	0,8 V	4 GHz

1.8.1 [5] <1.5> Em quanto foi reduzida a carga capacitiva entre as versões se a potência dinâmica foi reduzida em 10%?

1.8.2 [5] <1.5> Em quanto foi reduzida a potência dinâmica se a carga capacitiva não muda?

1.8.3 [5] <1.5> Supondo que a carga capacitiva da versão 2 é 80% da carga capacitativa da versão 1, ache a voltagem para a versão 2 se a potência dinâmica da versão 2 for reduzida em 40% a partir da versão 1.

Supondo que as tendências da indústria mostrem que a geração de um novo processo se expanda da seguinte forma:

	Capacitância	Tensão	Taxa de clock	Área
a.	1	$1/2^{1/2}$	1,15	$1/2^{1/2}$
b.	1	$1/2^{1/4}$	1,2	$1/2^{1/4}$

1.8.4 [5] <1.5> Encontre o fator de expansão para a potência dinâmica.

1.8.5 [5] <1.5> Encontre a expansão da capacidade por área unitária.

1.8.6 [5] <1.5> Assumindo que um processador Core 2 com a taxa de clock de 2,667 GHz, um poder de consumo de 95 W e uma tensão de 1,1 V, encontre a tensão e a taxa de clock do processador para sua próxima geração de processamento.

Exercício 1.9

Embora a potência dinâmica seja a principal fonte de dissipação de energia na CMOS, a corrente de vazamento produz uma dissipação de potência estática $V \times I_{vazamento}$. Quanto menores as dimensões no chip, mais significativa é a potência estática. Considere os valores mostrados na tabela a seguir para a dissipação de potência estática e dinâmica para várias gerações de processadores.

	Tecnologia	Potência dinâmica (W)	Potência estática (W)	Tensão (V)
a.	180 nm	50	10	1,2
b.	70 nm	90	60	0,9

1.9.1 [5] <1.5> Ache a porcentagem da potência total dissipada compreendida por potência estática.

1.9.2 [5] <1.5> Se a potência total dissipada for reduzida em 10% enquanto mantém a estática para a taxa de potência total do problema 1.9.1, quanto a tensão deve ser reduzida para que a corrente de vazamento continue igual?

1.9.3 [5] <1.5> Determine a razão entre potência estática e potência dinâmica para cada tecnologia.

Considere agora a dissipação de potência dinâmica de diferentes versões de um processador para três diferentes tensões dadas na tabela seguinte.

	1,2 V	1,0 V	0,8 V
a.	75 W	60 W	35 W
b.	62 W	50 W	30 W

1.9.4 [5] <1.5> Determine a potência estática para cada versão em 0,8 V, considerando uma razão entre potência estática e dinâmica de 0,6.

1.9.5 [5] <1.5> Determine a dissipação das potências estática e dinâmica utilizando as taxas obtidas no problema 1.9.1.

1.9.6 [10] <1.5> Determine a média geométrica das variações de potência entre as versões.

Exercício 1.10

A tabela a seguir mostra o desmembramento de tipo de instrução de determinada aplicação executada em 1, 2, 4 ou 8 processadores. Usando esses dados, você estará explorando o ganho de velocidade das aplicações em processadores paralelos.

	Processadores	# Instruções por processador			CPI		
		Aritmética	Load/Store	Desvio	Aritmética	Load/Store	Desvio
a.	1	2560	1280	256	1	4	2
	2	1280	640	128	1	4	2
	4	640	320	64	1	4	2
	8	320	160	32	1	4	2

	Processadores	# Instruções por processador			CPI		
		Aritmética	Load/Store	Desvio	Aritmética	Load/Store	Desvio
b.	1	2560	1280	256	1	4	2
	2	1350	800	128	1	6	2
	4	800	600	64	1	9	2
	8	600	500	32	1	13	2

1.10.1 [5] <1.4, 1.6> A tabela apresentada mostra o número de instruções exigido por processador para completar um programa em um multiprocessador com 1, 2, 4 ou 8 processadores. Qual é o número total de instruções executadas por processador? Qual é o número agregado de instruções executadas por todos os processadores?

1.10.2 [5] <1.4, 1.6> Dados os valores de CPI à direita da tabela, ache o tempo de execução total para esse programa em 1, 2, 4 e 8 processadores. Considere que cada processador tem uma frequência de clock de 2 GHz.

1.10.3 [5] <1.4, 1.6> Se o CPI das instruções aritméticas fosse dobrado, qual seria o impacto sobre o tempo de execução do programa em 1, 2, 4 ou 8 processadores?

A tabela a seguir mostra o número de instruções por núcleo de processador em um processador multicore, além do CPI médio para executar o programa em 1, 2, 4 ou 8 núcleos.

Usando esses dados, você estará explorando o ganho de velocidade das aplicações em processadores multicore.

	Núcleos por processador	Instruções por núcleo	CPI médio
a.	1	1,00E + 10	1,2
	2	5,00E + 09	1,3
	4	2,50E + 09	1,5
	8	1,25E + 09	1,8
	Núcleos por processador	Instruções por núcleo	CPI médio
b.	1	1,00E + 10	1,2
	2	5,00E + 09	1,2
	4	2,50E + 09	1,2
	8	1,25E + 09	1,2

1.10.4 [10] <1.4, 1.6> Considerando uma frequência de clock de 3 GHz, qual é o tempo de execução do programa usando 1, 2, 4 ou 8 núcleos?

1.10.5 [10] <1.5, 1.6> Suponha que o consumo de potência do núcleo de um processador possa ser descrito pela equação a seguir

$$\text{Potência} = \frac{5\text{mA}}{\text{MHz}} \text{Tensão}^2$$

em que a tensão de operação do processador é descrita pela equação a seguir

$$\text{Tensão} = \frac{1}{5} \text{Frequência} + 0,4$$

com a frequência medida em GHz. Assim, a 5 GHz, a tensão seria 1,4 V. Ache o consumo de potência do programa executando em 1, 2, 4 e 8 núcleos, supondo que cada núcleo esteja operando a uma frequência de clock de 3 GHz. De modo semelhante, ache o consumo de potência do programa executando em 1, 2, 4 ou 8 núcleos supondo que cada núcleo esteja operando a 500 MHz.

1.10.6 [10] <1.5, 1.6> Ao utilizar um único núcleo, encontre o CPI requerido para o núcleo conseguir o tempo de execução igual ao tempo obtido ao utilizar o número de núcleos da tabela acima (vezes de execução do problema 1.10.4). Repare que o número de instruções deve ser o número agregado de instruções executadas entre todos os núcleos.

Exercício 1.11

A tabela a seguir mostra os dados de manufatura para diversos processadores.

	Diâmetro do wafer	Dies por wafer	Defeitos por área unitária	Custo por wafer
a.	15 cm	84	0,020 defeitos/cm ²	12
b.	20 cm	100	0,031 defeitos/cm ²	25

1.11.1 [10] <1.7> Ache o aproveitamento.

1.11.2 [5] <1.7> Ache o custo por die.

1.11.3 [10] <1.7> Se o número de dies por wafer for aumentado em 10% e os defeitos por unidade de área aumentar em 15%, ache a área do die e o aproveitamento.

Suponha que, com a evolução da tecnologia de manufatura dos dispositivos eletrônicos, o aproveitamento varie como mostra a tabela a seguir.

	T1	T2	T3	T4
aproveitamento	0,85	0,89	0,92	0,95

1.11.4 [10] <1.7> Ache o número de defeitos por unidade de área para cada tecnologia, dada uma área de die de 200 mm².

1.11.5 [5] <1.7> Represente graficamente a variação do aproveitamento junto com a variação dos defeitos por área unitária.

Exercício 1.12

A tabela a seguir mostra os resultados para os programas de benchmark SPE CPU2006 rodando em um AMD Barcelona.

	Nome	Contagem instruções × 109	Tempo de execução (seg)	Tempo de referência (seg)
a.	bzip2	2389	750	9650
b.	go	1658	700	10.4090

1.12.1 [5] <1.7> Descubra o CPI se o tempo de ciclo de clock for 0,333 ns.

1.12.2 [5] <1.7> Ache a razão SPEC.

1.12.3 [5] <1.7> Para esses dois benchmarks, encontre a média geométrica.

A tabela a seguir mostra dados para outros benchmarks.

	Nome	CPI	Taxa de clock	SPECratio
a.	libquantum	1,61	4 GHz	19,8
b.	astar	1,79	4 GHz	9,1

1.12.4 [5] <1.7> Ache o aumento em tempo de CPU se o número de instruções do benchmark for aumentado em 10% sem afetar o CPI.

1.12.5 [5] <1.7> Encontre o aumento no tempo de CPU se o número de instruções do benchmark aumentar em 10% e o CPI for aumentado em 5%.

1.12.6 [5] <1.7> Ache a mudança na SPECratio para a mudança descrita em 1.12.5.

Exercício 1.13

Suponha que estejamos desenvolvendo uma nova versão do processador AMD Barcelona com uma taxa de clock de 4 GHz. Acrescentamos algumas instruções ao conjunto de instruções, de modo que o número de instruções foi reduzido em 15% a partir dos valores mostrados para cada benchmark no Exercício 1.12. Os tempos de execução obtidos aparecem na tabela a seguir.

	Nome	Tempo de execução (seg)	Tempo de referência (seg)	SPECratio
a.	bzip2	700	9.650	13,7
b.	go	620	10.490	16,9

1.13.1 [10] <1.8> Ache o novo CPI.

1.13.2 [10] <1.8> Em geral, esses valores de CPI são maiores que aqueles obtidos nos exercícios anteriores para os mesmos benchmarks. Isso é decorrente principalmente da taxa de clock usada nos dois casos, 3 GHz e 4 GHz. Determine se o aumento no CPI é semelhante ao da taxa de clock. Se eles forem diferentes, por que isso ocorre?

1.13.3 [5] <1.8> Por quanto o tempo de CPU foi reduzido?

A tabela a seguir mostra dados para outros benchmarks.

	Nome	Tempo de execução (seg)	CPI	Taxa de clock
a.	libquantum	960	1,61	3 GHz
b.	astar	690	1,79	3 GHz

1.13.4 [10] <1.8> Se o tempo de execução for reduzido por outros 10% sem afetar o CPI e com uma taxa de clock de 4 GHz, determine o número de instruções.

1.13.5 [10] <1.8> Determine a taxa de clock exigida para gerar uma redução de mais 10% no tempo de CPU enquanto mantém o número de instruções e CPI inalterados.

1.13.6 [10] <1.8> Determine a taxa de clock se o CPI for reduzido em 15% e o tempo de CPU em 20% enquanto o número de instruções for inalterado.

Exercício 1.14

A [Seção 1.8](#) cita como armadilha a utilização de um subconjunto da equação de desempenho como uma métrica de desempenho. Para ilustrar isso, considere os dados a seguir para a execução de determinada sequência de 106 instruções em diferentes processadores.

Processador	Taxa de clock	CPI
P1	4 GHz	1,25
P2	3 GHz	0,75

1.14.1 [5] <1.8> Uma falácia comum é considerar o computador com a maior taxa de clock como tendo o maior desempenho. Verifique se isso é verdade para P1 e P2.

1.14.2 [10] <1.8> Outra falácia é considerar que o processador executando o maior número de instruções precisará de um tempo de CPU maior. Considerando que o processador P1 está executando uma sequência de 106 instruções e que o CPI dos processadores P1 e P2 não muda, determine o número de instruções que P2 pode executar ao mesmo tempo em que P1 precisa para executar 106 instruções.

1.14.3 [10] <1.8> Uma falácia comum é usar milhões de instruções por segundo (MIPS) para comparar o desempenho de dois processadores diferentes e considerar que o processador com o maior valor de MIPS tem o maior desempenho. Verifique se isso é verdade para P1 e P2.

Outro valor de desempenho comum é milhões de operações de ponto flutuante por segundo (MFLOPS), definido como

$$\text{MFLOPS} = \frac{\text{Nº de operações de PF}}{(\text{Tempo de execução} \times 10^6)}$$

mas esse valor tem os mesmos problemas do MIPS. Considere os programas na tabela a seguir, rodando nos dois processadores a seguir.

	Processador	Cont. instruções	Número de instruções			CPI			Taxa de clock
			L/S	PF	Desvio	L/S	PF	Desvio	
a.	P1	1 x 106	50%	40%	10%	0,75	1,0	1,5	4 GHz
	P2	5 x 106	40%	40%	20%	1,25	0,8	1,25	3 GHz
b.	P1	5 x 106	30%	30%	40%	1,5	1,0	2,0	4 GHz
	P2	2 x 106	40%	30%	30%	1,25	1,0	2,5	3 GHz

1.14.4 [10] <1.8> Encontre os valores de MFLOPS para os programas.

1.14.5 [10] <1.8> Ache os valores de MIPS para os programas.

1.14.6 [10] <1.8> Ache o desempenho para os programas e compare com MIPS e MFLOPS.

Exercício 1.15

Outra armadilha citada na [Seção 1.8](#) é esperar aprimorar o desempenho geral de um computador melhorando apenas um aspecto do computador. Isso pode ser verdade, mas nem sempre é. Considere um computador rodando programas com os tempos de CPU mostrados na tabela a seguir.

	Instruções PF	Instruções INT	Instruções L/S	Instruções desvio		Tempo total
a.	70 s	85 s	55 s	40 s	250 s	
b.	40 s	90 s	60 s	20 s	210 s	

1.15.1 [5] <1.8> Em quanto é reduzido o tempo total se o tempo para as operações de PF for reduzido em 20%?

1.15.2 [5] <1.8> Em quanto o tempo para operações INT é reduzido se o tempo total for reduzido em 20%?

1.15.3 [5] <1.8> O tempo total pode ser reduzido em 20% reduzindo-se apenas o tempo para as instruções de desvio?

A tabela a seguir mostra o desmembramento de tipo de instrução por processador de determinada aplicação executada em diferentes números de processadores.

	Processadores	Instruções PF	Instruções INT	Instruções L/S	Instruções desvio	CPI (PF)	CPI (INT)	CPI (L/S)	CPI (Desvio)
a.	2	280 x 106	1000 x 106	640 x 106	128 x 106	1	1	4	2
b.	16	50 x 106	110 x 106	80 x 106	16 x 106	1	1	4	2

Considere que cada processador tenha uma taxa de clock de 2 GHz.

1.15.4 [10] <1.8> Por quanto devemos melhorar o CPI das instruções de PF se quisermos que o programa execute duas vezes mais rápido?

1.15.5 [10] <1.8> Por quanto devemos melhorar o CPI das instruções de L/S se quisermos que o programa execute duas vezes mais rápido?

1.15.6 [5] <1.8> Por quanto o tempo de execução do programa é melhorado se o CPI das instruções de INT e PF for reduzido em 40% e o CPI das instruções de L/S e desvio for reduzido em 30%?

Exercício 1.16

Outra armadilha, relacionada à execução dos programas em sistemas multiprocessadores, é esperar melhoria no desempenho aprimorando apenas o tempo de execução de parte das rotinas. A tabela a seguir mostra o tempo de execução de cinco rotinas de um programa rodando em diferentes quantidades de processadores.

	Nº processadores	Rotina A (ms)	Rotina B (ms)	Rotina C (ms)	Rotina D (ms)	Rotina E (ms)
a.	4	12	45	6	36	3
b.	32	2	7	1	6	2

1.16.1 [10] <1.8> Ache o tempo de execução total e em quanto ele é reduzido se o tempo das rotinas A, C e E for melhorado em 15%.

1.16.2 [10] <1.8> Em quanto o tempo total é reduzido se a rotina B for melhorada em 10%?

1.16.3 [10] <1.8> Em quanto o tempo total é reduzido se a rotina D for melhorada em 10%?

O tempo de execução em um sistema multiprocessador pode ser dividido em tempo de computação para as rotinas mais tempo de roteamento gasto enviando dados de um processador para outro. Considere o tempo de execução e o tempo de roteamento dados na tabela a seguir. Nesse caso, o tempo de roteamento é um componente importante do tempo total.

Nº processadores	Rotina A (ms)	Rotina B (ms)	Rotina C (ms)	Rotina D (ms)	Rotina E (ms)	Roteamento (ms)
2	40	78	9	70	4	11
4	29	60	4	36	2	13
8	15	45	3	19	3	17
16	7	35	1	11	2	22
32	4	23	1	6	1	23
64	2	12	0,5	3	1	26

1.16.4 [10] <1.8> Para cada duplicação do número de processadores determine a razão do tempo de computação novo para antigo e a razão do tempo de roteamento novo para antigo.

1.16.5 [5] <1.8> Usando a média geométrica das razões extrapole para descobrir o tempo de computação e o tempo de roteamento em um sistema com 128 processadores.

1.16.6 [10] <1.8> Ache o tempo de computação e o tempo de roteamento para um sistema com um processador.

§1.1: Questões para discussão: muitas respostas são aceitáveis.

§1.3: Memória em disco: não volátil, tempo de acesso longo (milissegundos), e custo de US\$0,20 a US\$2,00/GB. Memória usando semicondutores: volátil, tempo de acesso curto (nanossegundos) e custo de US\$20 a US\$75/GB.

§1.4: 1.a: ambos, b: latência, c: nenhum. 2,7 segundos.

§1.4: b.

§1.7: 1, 3 e 4 são motivos válidos. A resposta 5 geralmente pode ser verdadeira, pois o alto volume pode tornar o investimento extra para reduzir o tamanho do die em, digamos, 10%, uma boa decisão econômica, mas isso não precisa ser verdadeiro.

§1.8: 53: a. Computador A tem a maior avaliação MIPS. b. Computador B é mais rápido.

**Respostas das
Seções “Verifique
você mesmo”**