Capítulo 2

Abstrações e Tecnologia Computacional

Neste capítulo discute as abstrações utilizadas em sistemas de computadores. O princípio da abstração é a forma encontrada pelos projetistas de *hardware* e *software* para tratar a complexidade dos sistemas computacionais. Além disso, apresenta algumas das tecnologias computacionais que vem sendo empregadas e que tornam mais fáceis a utilização destes sistemas.

2.1 Introdução

Nos últimos anos temos presenciado uma evolução rápida dos sistemas computacionais. Podemos dizer que os computadores levaram a civilização contemporânea a uma terceira revolução, à chamada revolução da informação. Antes a civilização presenciou a revolução agrícola e a industrial. A revolução da informação proporcionou um aumento significativo na disseminação de informações. Este fato aumentou a capacidade intelectual da humanidade com impacto direto nas ciências.

Atualmente os cientistas da computação têm trabalhado em várias linhas de investigação científica. Eles buscam viabilizar ferramentas mais eficientes para que os cientistas teóricos e experimentais possam explorar novas fronteiras do conhecimento nas mais diversas áreas como as Engenharias, a Astronomia, a Biologia, a Física, a Química entre outras.

O custo dos sistemas computacionais tem decrescido significativamente nos últimos anos. A redução do custo permite que a utilização dos computadores se multiplique. Hoje estes sistemas são viáveis para diversas aplicações onde antigamente o custo era proibitivo, como, por exemplo, nos caixas eletrônicos, nos automóveis ou *laptops*.

Os avanços verificados na tecnologia de *hardware* têm permitido que os especialistas criem programas extremamente úteis. Este fato pode ser comprovado no emprego dos computadores nas mais diversas atividades desenvolvidas pelas pessoas.

Os bons programadores têm como um dos seus principais objetivos o desempenho dos seus programas. Para os usuários a obtenção de resultados rápidos é um fator preponderante para o sucesso de qualquer *software*. Nos anos 60 e 70 a principal restrição para um bom desempenho dos computadores era o tamanho da memória. Assim, os programadores procuravam minimizar o espaço de memória utilizado pelo seu programa para torná-lo mais

rápido. Os avanços nas técnicas de projeto dos processadores e na tecnologia de fabricação das memórias reduziram muito a importância de programas pequenos.

Os programadores atuais para obter bom desempenho devem se familiarizar com os aspectos que substituíram o modelo de memória dos anos 60 e 70, como a natureza hierárquica dos sistemas de memória associado ao paralelismo no nível dos processadores. Os especialistas que desejam construir compiladores, sistemas operacionais, sistemas gerenciadores de base de dados ou programas aplicativos realmente eficientes e, portanto competitivos, devem ampliar seus conhecimentos sobre organização de computadores.

Nas próximas seções apresentamos os fundamentos que são utilizados nos demais capítulos. Introduzimos os conceitos básicos e as definições, além de destacar os principais componentes de *hardware* e *software* da máquina.

2.2 Níveis de Software Inferiores ao Programa do Usuário

A comunicação num sistema de computador eletrônico digital binário é realizada através de sinais elétricos. Os símbolos escolhidos para representar os dígitos binários são 0 e 1. Podemos visualizar a linguagem da máquina como um conjunto de números na base 2. Cada dígito binário também é conhecido como um *bit*.

Os computadores executam programas que são compostos por um conjunto de comandos que formam a solução de um problema. Cada comando individual é denominado **instrução**. As instruções nada mais são do que um conjunto de *bits* inteligíveis pelo computador e que podem ser associados a números. Por exemplo, os *bits*

1000110010100000

podem informar a um determinado processador para somar dois números.

Nos primeiros computadores os programadores tinham que escrever seus programas na linguagem nativa do computador. A prática de usar números binários era cansativa e propensa a erros. Este fato levou a invenção de uma notação mais próxima da maneira das pessoas pensarem. Inicialmente esta tradução era realizada manualmente para a linguagem binária. O uso da máquina para programar a própria máquina foi à forma encontrada pelos pioneiros da programação para traduzir programas escritos em uma linguagem simbólica para a linguagem da máquina. O primeiro deste programas foi denominado **montador**.

O montador traduzia uma versão simbólica de cada instrução para a sua versão correspondente em binário. Por exemplo, o programador poderia escrever **ADD A, B** sendo o montador responsável pela tarefa de traduzir esta instrução para os *bits* 1000110010100000. O nome desta linguagem simbólica, formada por mnemônicos, é **linguagem de montagem** ou **linguagem *assembly***.

Apesar de muitas mudanças e melhorias, as linguagens de montagem estão longe da facilidade desejada pelos cientistas. A linguagem de montagem obriga o programador a escrever uma linha para cada instrução a ser executada pela máquina. Isto força o programador a raciocinar como a máquina.

Este pensamento de raciocinar num nível mais baixo inspirou uma questão simples. Se podemos traduzir uma linguagem de montagem para instruções binárias, de maneira a tornar mais simples o trabalho dos programadores, porque não traduzir uma notação de mais alto nível para uma notação binária?

Apesar de ser uma tarefa mais complexa projetar e construir este tradutor, ela é viável. Os programadores de hoje têm uma alta produtividade devido ao projeto deste tradutor de alto nível denominado **compilador**. Os compiladores são programas que aceitam uma notação mais natural muito próxima da nossa linguagem. A linguagem que eles compilam são denominadas **linguagem de alto nível**. Seu uso permite que um programador escreva a seguinte expressão A + B e o compilador a expresse na linguagem de montagem **ADD A, B**. Por sua vez, o montador deve traduzir este comando para a instrução binária 1000110010100000.

As linguagens de alto nível oferecem várias vantagens aos programadores. Primeiro, elas permitem que os programadores raciocinem de uma forma mais natural, mesmo usando palavras em língua inglesa e notações algébricas. Além disso, as linguagens de alto nível são projetadas de acordo com o tipo de programa a ser escrito. Por exemplo, a linguagem Fortran foi projetada para realizar cálculos científicos, o Cobol para processar dados comerciais, o Lisp para manipular símbolos e assim por diante.

A segunda vantagem do uso das linguagens de alto nível é o aumento da produtividade dos programadores. O tempo gasto para desenvolver programas melhora na razão inversa da quantidade de linhas de código necessárias para expressar uma idéia. Outra vantagem é o fato de que programas escritos em tais linguagens são independentes do computador no qual foram desenvolvidos. Isto porque os compiladores e montadores, em geral, podem traduzir linguagens de alto nível para instruções de qualquer máquina.

A reutilização de programas já escritos é muito mais vantajosa do que escrever novamente um programa desde o princípio. Por isso, os programadores compartilham as rotinas potencialmente mais utilizadas, organizando-as em **bibliotecas**. Uma das primeiras bibliotecas a ser idealizada continha rotinas de entrada e saída de dados. Ela continha rotinas, por exemplo, para controlar uma impressora, um disco, fitas entre outros dispositivos.

A experiência mostrou que um conjunto de programas poderia ser mais eficientemente executado se houvesse um programa separado responsável por supervisionar a execução dos demais. Ele seria responsável por iniciar o processamento do próximo programa numa fila, evitando a intervenção humana para realizar esta tarefa. Estes tipos de

programas, denominados **supervisores**, juntamente com as rotinas de entrada e saída formam a base do que hoje chamamos de **sistema operacional**. Os sistemas operacionais são programas que controlam e coordenam os recursos físicos da máquina, além de implementarem a interface entre os serviços do sistema e o usuário.

Os *softwares* tendem a ser classificados por seu uso. Aqueles que implementam serviços para tornar a máquina mais atrativa, serviços estes utilizados por outros programas, são denominados **softwares do sistema**. Como exemplo destes *softwares* podemos citar os sistemas operacionais, os compiladores e montadores. Já os *softwares* de aplicação ou simplesmente aplicativos são programas desenvolvidos para simples usuários da máquina, como por exemplo, as planilhas eletrônicas ou os processadores de texto.

2.3 Componentes do Computador

A seção 2.2 apresentou alguns dos conceitos básicos envolvidos no nível do *software*. Esta seção aborda um pouco sobre o *hardware* dos computadores. A Figura 2.1 ilustra uma visão simplificada dos níveis hierárquicos do *hardware* e do *software*, mostrados de forma clássica, como anéis concêntricos, construídos a partir do núcleo do *hardware* em direção a camada do *software* mais próxima do usuário.

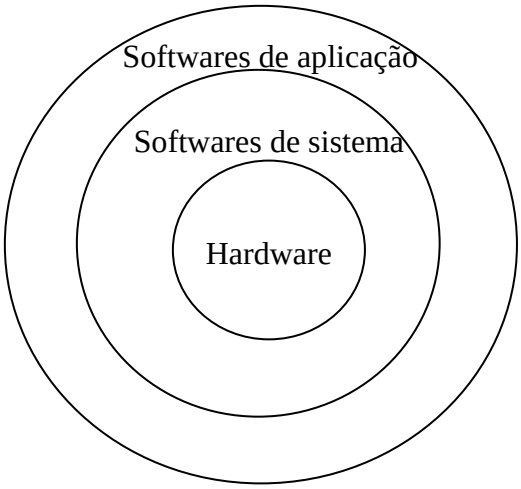

Figura 2.1. *Visão simplificada dos níveis hierárquicos do hardware e do software.*

Um computador típico dos dias de hoje possui vários dispositivos de entrada e saída como um teclado, vídeo, *mouse*, impressora e um gabinete contendo mais *hardware*. Segundo o livro de Patterson e Henessy, este *hardware* é composto por cinco componentes clássicos que são os dispositivos de entrada de dados, os dispositivos de saída de dados, a memória, o caminho de dados e a unidade de controle. Estes dois últimos componentes quando

combinados são chamados de processador. A Figura 2.2 ilustra uma possível organização interna do *chip* de um processador.

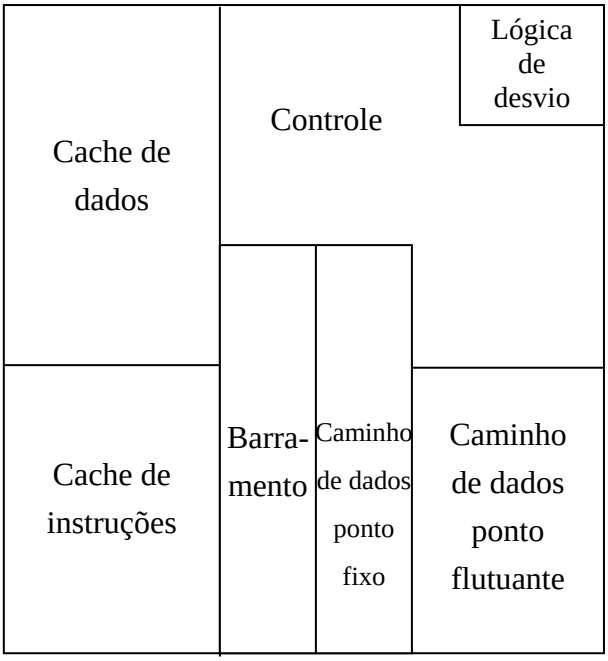

Figura 2.2. *Interior de um chip de processador.*

Um fato comum que ocorre tanto na descrição do *hardware* quanto na descrição do *software* é que sempre que aprofundamos a discussão aparecem mais detalhes sobre o assunto. Se olharmos pela perspectiva inversa, os detalhes do nível mais baixo são mantidos escondidos dos níveis mais altos. O uso dessas camadas ou **abstrações** é uma das técnicas mais importantes empregadas no projeto de sistemas computacionais sofisticados. Isto porque permite que modelos mais simples sejam utilizados nos níveis superiores.

Uma das abstrações mais importantes é a interface entre o *hardware* e o *software* no nível mais baixo. Em função de sua importância essa abstração recebeu o nome de **arquitetura do conjunto de instruções**. A arquitetura do conjunto de instruções de uma máquina inclui todos os pontos que o programador precisa conhecer para fazer com que a máquina trabalhe corretamente. Esta interface padrão permite que os projetistas de computadores pensem nas funções independentemente do *hardware* que as executa.

Os projetistas de computadores distinguem a arquitetura do conjunto de instruções da **implementação da arquitetura**. Esta interface abstrata possibilita que diferentes implementações de uma mesma arquitetura possam ser realizadas, com custo e desempenho variáveis e que executam o mesmo *software*.

Uma outra questão importante nos dias de hoje é a conexão entre computadores através das redes. As ligações em rede têm uma série de vantagens como a troca de informações com altas taxas de velocidade, o compartilhamento de recursos e acessos remotos.

As redes variam muito em relação ao seu tamanho e seu desempenho. Uma das redes de comunicação mais populares é a Ethernet. Sua tecnologia é apropriada para interconexão numa rede local. Isto é, ela é adequada, por exemplo, a distâncias como um andar de um edifício. Para redes de longa distância (entre cidades, continentes) são utilizadas transmissões via satélite, rádio ou fibras óticas.

2.4 Circuitos Integrados

O desenvolvimento da tecnologia de implementação dos componentes eletrônicos vem sustentando o crescimento da indústria de computação desde os anos 70. A tecnologia é que impõe limites ao que um computador pode realizar. Por exemplo, é a tecnologia que define a velocidade de processamento e a capacidade de armazenamento.

Um transistor é um componente eletrônico controlado por eletricidade. Antigamente os circuitos integrados eram compostos por dezenas ou centenas de transistores em um único ***chip***. Num curto intervalo de tempo, a mesma área de um *chip* passou a integrar vários milhões de transistores dando origem aos chamados circuitos *very large-scale integrated* (*VLSI*).

O processo de fabricação de um *chip* tem início com o **silício** que é um elemento químico encontrado na areia. Ele é um elemento **semicondutor** porque não é um bom condutor de eletricidade nem um bom isolante. Quando combinado com outros elementos químicos especiais o silício pode adquirir excelente condutividade, ser completamente isolante, ou conduzir ou isolar a eletricidade sob certas condições especiais atuando como uma chave. Os transistores pertencem a esta última categoria. Portanto, um circuito *VSLI* é um conjunto formado por milhões de combinações de condutores, isolantes e chaves.

O processo de fabricação dos circuitos integrados é importante para a determinação do custo final de um *chip*. Ele determina também o custo final de um computador. A Figura 2.3 mostra o processo de fabricação de um *chip*.

A figura mostra uma barra de silício que após ser fatiada transforma-se num *waffer* virgem. Para gravar os circuitos são necessários de 20 a 30 passos de processamento. Os *waffers* já como padrão gravado são divididos em *chips* por um separador. Depois de testados estes *chips* são encapsulados, novamente testados e disponibilizados para consumo.

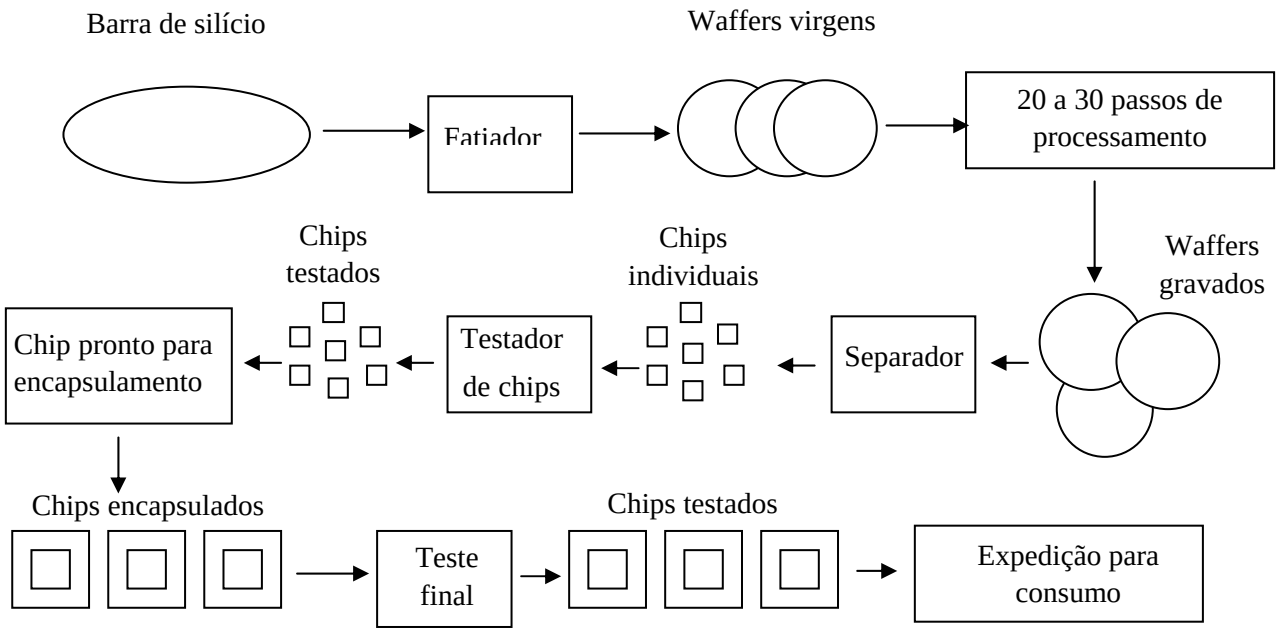

Figura 2.3. *Processo de fabricação de um chip.*

2.5 Considerações Finais

Apesar de ser difícil prever o nível da relação custo/desempenho que os computadores terão no futuro, é seguro apostar na premissa de que esta relação será melhor do que a possuímos hoje. Para aproveitar estes avanços os projetistas de computadores e os programadores de software devem entender de uma grande variedade de assuntos.

Tanto os projetistas de *hardware* quanto os projetistas de software constroem seus sistemas baseados numa arquitetura com níveis hierárquicos. O entendimento da abstração é de importância fundamental para a compreensão dos sistemas computacionais da atualidade.