

Capítulo 2

Conceitos de Computação e Computadores

O estudo de algoritmos e de lógica de programação é facilitado com o conhecimento do funcionamento do computador e de como ele executa seus programas. O objetivo deste capítulo é discutir os conceitos básicos de computação e computadores, introduzindo os conceitos importantes, tais como a organização básica de um computador, como os dados e as instruções são armazenados e como as instruções são executadas. A aplicação desses conceitos será feita a partir do Capítulo 3 e se estenderá por todo o livro.

2.1 Origens da computação

2.1.1 A necessidade de calcular

A capacidade do ser humano em realizar cálculos surgiu com sua habilidade de se comunicar com mais precisão. Inicialmente os primeiros humanoides comunicavam-se por um conjunto de grunhidos, tal como observado em outros animais. A evolução permitiu que houvesse um aprimoramento de suas capacidades cognitivas, levando dessa forma ao surgimento de vocabulários mais extensos e depois à elaboração de regras para a composição de frases com esses vocabulários. Assim, surgiram as primeiras linguagens.

Acompanhando esse processo evolutivo, houve o aparecimento da escrita. A escrita permitiu o registro de informações importantes, que poderiam ser compartilhadas com outros humanos, mesmo que seu autor estivesse ausente. De início, eram pinturas rupestres e marcas em cavernas e em ossos. Depois, apareceram os símbolos que representavam palavras utilizadas na comunicação. Por fim, houve o surgimento do alfabeto,

permitindo que a capacidade de comunicação fosse estendida e simplificada, pois agora não seria mais necessário decorar um número muito grande de símbolos, já que cada palavra pode ser formada a partir de um pequeno alfabeto.

É nesse contexto evolutivo da comunicação humana que surge a necessidade do homem em realizar cálculos. Como o desenvolvimento da capacidade de comunicação é resultado direto do processo de desenvolvimento de raciocínio, o ser humano passou a ter necessidade de controlar e proteger suas atividades primárias. Essas atividades traduziam-se principalmente no registro de suas transações comerciais, como a contagem de rebanhos, a troca de moedas e a divisão de terras, bem como a elaboração de calendários para determinar as estações do ano para a agricultura.

2.1.2 O desenvolvimento de sistemas de numeração

Para representar as quantidades envolvidas em computações, foi necessário o desenvolvimento de sistemas de numeração. No caso do ser humano, o sistema mais evidente é o decimal, em razão do uso de todos os seus dedos para computar (ainda aprendemos assim!). Dessa forma, surgiu a palavra *digitus*, que, traduzindo do latim, significa **dedo**. Em português essa palavra é conhecida como **dígito**.

O sistema decimal emprega a base 10, isto é, cada dedo representa um número no intervalo de 1 a 10. A grande deficiência do sistema decimal empregando as mãos é a capacidade de representar grandes números. Algumas pessoas tentaram explorar esse limite, como o monge beneditino Beda (673-735), que desenvolveu um método que permitia a descrição de números até 10.000, de acordo com a posição dos dedos e um método que possibilitava a contagem até 1.000.000, colocando a mão em várias partes do corpo.

Deve-se observar que nem sempre a base 10 foi utilizada por todos os povos. Os babilônios (2000 a.C.), por exemplo, empregavam o sistema sexagesimal (base 60), pois era baseado em um sistema de unidades de pesos e medidas adotado. Os maias (~ 0 a.C.) utilizavam um sistema vigesimal (base 20), que derivava da forma como calculavam seus calendários. Os gregos usavam um sistema misto, decimal e hexadecimal. Os romanos, um sistema decimal com símbolos especiais para os números 5, 50 e 500 etc.

Como se pode observar, a contagem utilizando apenas as mãos não é prática. Além do problema de representar grandes números, não é possível registrar os cálculos. Foi então necessário o desenvolvimento de símbolos para representar números escritos. Por exemplo, os egípcios (3500 a.C.) utilizavam um sistema de representação numérica com símbolos específicos para as potências de 10, como 1, 10, 100, 1.000, 10.000 etc. Nesse sistema, o número 1 era representado por um traço vertical; o número 10, por um osso de calcanhar invertido; o número 100, por um laço; o número 1.000, por uma flor de lótus; e o número 10.000, por um dedo dobrado. Esses símbolos estão indicados na Tabela 2.1.

Tabela 2.1 Alguns símbolos do sistema de numeração egípcio.

Número	Símbolo
1	
10	�
100	𓀃
1.000	𓁻
10.000	𓁻

Assim, um número como 23.523 primeiro deve ser decomposto em potências de 10 como $2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$. Daí, basta repetir o símbolo correspondente a cada potência de acordo com seu fator multiplicativo, conforme apontado na Figura 2.1.

**Figura 2.1** O número 23.523 em egípcio.

O sistema empregado pelos romanos era similar. Como já foi citado, eles usavam os símbolos para as potências de 10 e para os números 5, 50, 500 etc. A Tabela 2.2 exibe os símbolos de números utilizados pelos romanos.

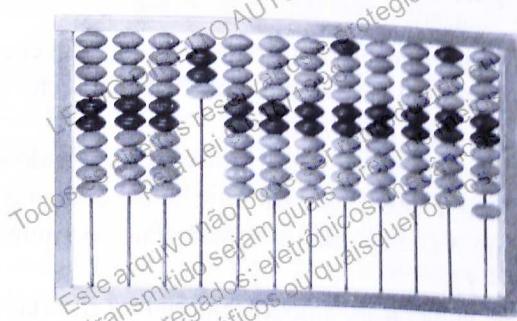
Os números começando com 4 e 9 eram formados utilizando-se uma abreviação subtrativa, isto é, 9 igual a 10 menos 1, que em romano era escrito como *IX* e 40 igual a 50 menos 10, escrito como *XL*. Seguindo esse raciocínio, um número como 1.979 era escrito assim: *MCMXXIX*.

Esse tipo de sistema de numeração, embora superior ao uso dos dedos para fazer a contagem, apresenta ainda alguns problemas, como realizar multiplicações. Os romanos, por exemplo, utilizavam-se de um *ábaco* para efetuar as operações aritméticas.

Tabela 2.2 Símbolos do sistema de numeração romano.

Número	Símbolo
1	<i>I</i>
5	<i>V</i>
10	<i>X</i>
50	<i>L</i>
100	<i>C</i>
500	<i>D</i>
1.000	<i>M</i>
10.000	\overline{X}
100.000	\overline{C}
1.000.000	\overline{M}

Os primeiros ábacos que se tem notícia datam de aproximadamente 1.000 anos antes de Cristo, sendo utilizados por babilônios e egípcios. Um ábaco é uma placa contendo sulcos (ábaco romano) ou uma esquadria de madeira contendo arames (ábaco japonês ou *soroban*) em que pedrinhas podiam ser movidas, representando dígitos de um número e que permitiam a realização de operações aritméticas (veja a Figura 2.2).

**Figura 2.2** Um ábaco típico.

Foi com o uso de pedrinhas para auxiliar nas contagens que surgiu o termo cálculo. O **cálculo**, palavra que dá origem ao verbo calcular, deriva da palavra latina *calculus* e está relacionada com a palavra grega *chalix*, ambas significando pedrinha ou seixo, que foram os primeiros objetos a auxiliar o homem em seus cálculos. Por sua vez, a **computação** significa o ato ou o efeito de **computar**, que é um verbo que exprime o ato de fazer contagem, de contar ou calcular.

O uso do ábaco é bem simples. Cada pedrinha na parte superior vale 5 e na parte inferior vale 1. As colunas (sulcos ou arames) indicam da direita para a esquerda as unidades, dezenas, centenas, milhares etc. Um dígito em sua posição é representado pelo número de pedrinhas que são deslocadas (para cima ou para baixo, de acordo com a convenção adotada) da sua posição original. Por exemplo, a Figura 2.2 representa o número 23.516.

É possível realizar todas as operações aritméticas com um ábaco (algumas podem requerer “truques” particulares). A adição, a operação mais simples de se executar com um ábaco, por exemplo, é feita da seguinte forma: primeiro, representa-se no ábaco o primeiro operando, da forma descrita anteriormente. Depois, sem apagar o primeiro operando, representa-se da mesma forma o segundo operando. Quando ocorre o estouro de uma coluna, é colocado o “vai-um” necessário na próxima. Interpretando o estado final do ábaco, tem-se o resultado.

O problema da utilização do ábaco é que cada passo apaga o precedente, de modo que para se verificar um resultado é necessário refazer o cálculo. No entanto, esse instrumento ainda é muito popular no Japão, onde, em 1946, foi utilizado para derrotar uma calculadora elétrica em uma competição.

Um sistema de numeração próximo ao que se utiliza atualmente foi inventado pelos chineses. Esse sistema emprega a posição dos números para indicar seu valor. Os chineses contavam os números de 1 a 9, representando-os por palitos convenientemente arranjados, conforme exibido na Tabela 2.3.

Os chineses realizavam as operações aritméticas com esses símbolos, colocando os palitos que representam os dígitos em um tabuleiro denominado *suan-phan*. Nesse tabuleiro, uma casa vazia representava 0 (ainda não havia o conceito de 0) e os palitos significando os dígitos eram colocados nas casas de acordo com a posição desse dígito no número. Por exemplo, a soma de 62.014 com 74.168 em um *suan-phan* é representada pela Figura 2.3.

Nessa figura, os números 62.014 e 74.168 estão representados respectivamente na terceira e na quarta linhas. Na quinta linha, tem-se o resultado (136.182) e, na segunda, aparecem os “vai-um” da soma.

Tabela 2.3 Símbolos do sistema de numeração chinês.

Número	Símbolo
1	
2	
3	
4	
5	
6	—
7	—
8	—
9	—

LEI DO DIREITO AUTORAL
Todos os direitos reservados e protegidos
pela Lei 9.610/1998.

			—	—	—	—
	—	—	—	—	—	—————
	—	—	—	—	—————	—————
—	—	—	—	—	—————	—————
—	—	—	—	—————	—————	—————
—	—	—	—————	—————	—————	—————
—	—————	—————	—————	—————	—————	—————
—————	—————	—————	—————	—————	—————	—————

Este arquivo não pode ser reproduzido ou
transmitido sejam quais forem os meios,
eletrônicos ou mecânicos,
empregados.
fotográficos ou quaisquer outros.

Figura 2.3 O uso do *suan-phan* chinês.

Por fim, é importante mencionar a contribuição da Índia no desenvolvimento dos sistemas de numeração. Foram os hindus que primeiro registraram os cálculos em papéis (650 d.C.) e que inventaram o símbolo para o zero em sua escrita. O uso do zero permitiu que os cálculos pudessem ser representados no papel, sem a necessidade de deixar um espaço em branco.

A matemática hindu foi levada pelos árabes, que a disseminaram pelo Ocidente até a Espanha. Os árabes foram os responsáveis também pela representação numérica que se utiliza atualmente, com a criação do *algarismo*, da *tabuada* e da *álgebra*. Algarismo é uma palavra derivada do nome do matemático *Al-Khwarizmi*¹. Com as Cruzadas, foi possível introduzir esses conhecimentos no mundo ocidental, os quais são utilizados até hoje.

2.2 A evolução dos computadores

2.2.1 Geração zero – Computadores puramente mecânicos

Com o Iluminismo (século XVIII), principalmente após o surgimento do cálculo diferencial por Newton e Leibniz e da tentativa de definir uma linguagem matemática universal por Leibniz, que mais tarde convergiria para o que se conhece por *lógica simbólica*, surgiram os primeiros dispositivos mecânicos de cálculo.

A primeira calculadora portátil foi desenvolvida por John Napier em 1612 e chamava-se *ossos de Napier* (veja a Figura 2.4).

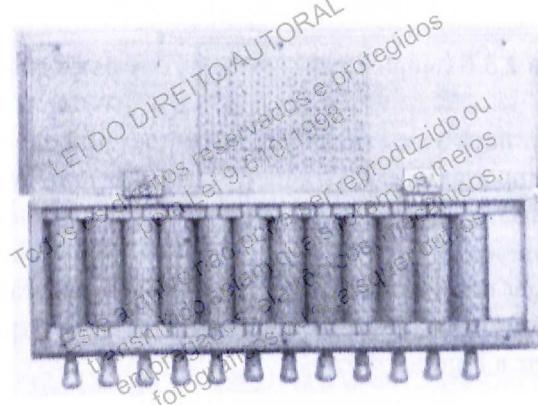


Figura 2.4 Os ossos de Napier.

¹A palavra algoritmo também deriva de seu nome.

Os ossos de Napier eram na verdade um conjunto de bastões para se realizar as multiplicações por meio de adições. Cada bastão continha a tabuada de um número e a multiplicação de um número x por um número y era realizada consultando-se a x -ésima linha do bastão correspondente ao número y . Por exemplo, para multiplicar o número 3 por 849 bastava procurar na quinta linha dos bastões correspondentes aos números 8, 4 e 9 o valor do resultado das multiplicações, deslocar à esquerda os valores encontrados de acordo com a sua posição no número (sua potência de dez) e então somar os resultados, conforme indicado na Figura 2.5. Assim, o valor de 3×849 é dado pela soma $2.400 + 120 + 27$, ou seja, 2.547.

Índice	8	4	9
1	0 8	0 4	0 9
2	1 6	0 8	1 8
3	2 4	1 2	2 7
4	3 2	1 6	3 6
5	4 0	2 0	4 5
6	4 8	2 4	5 4
7	5 6	2 8	6 3
8	6 4	3 2	7 2
9	7 2	3 6	8 1

Figura 2.5 Exemplo de utilização dos ossos de Napier.

Napier ainda foi o primeiro a escrever números com o símbolo de ponto como separador decimal e, mais importante, criou o conceito de *logaritmo*. William Oughtred, em 1622, deu origem, a partir de conceitos dos ossos de Napier, à primeira régua de cálculo.

Em 1642, o matemático Blaise Pascal criou uma máquina de somar, a *Pascaline*, inicialmente para auxiliar nos negócios de seu pai. A adição era realizada com o auxílio de engrenagens que giravam de acordo com um seletor de disco que permitia escolher um dígito desejado (veja à Figura 2.6).

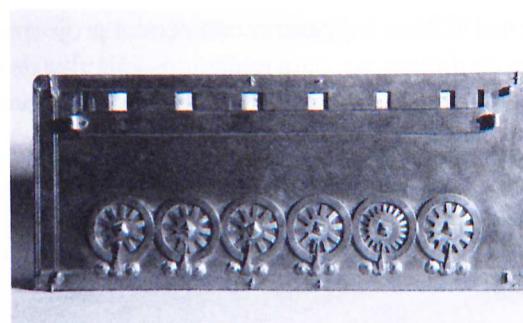


Figura 2.6 A *Pascaline* de Pascal.

O resultado era apresentado em um visor mecânico, acima dos seletores. Em 1673, utilizando uma engrenagem cilíndrica de passo, Leibniz inventou uma máquina que conseguia fazer multiplicações, por somas sucessivas que eram automaticamente realizadas pelas engrenagens.

Outro fator histórico para contribuir no desenvolvimento de dispositivos automáticos de cálculo foi a Revolução Industrial. Nesse contexto, destaca-se inicialmente Joseph-Marie Jacquard, que em 1801, na França, inventou uma máquina de tear automática (veja a Figura 2.7), cujos padrões eram fornecidos por cartões perfurados.

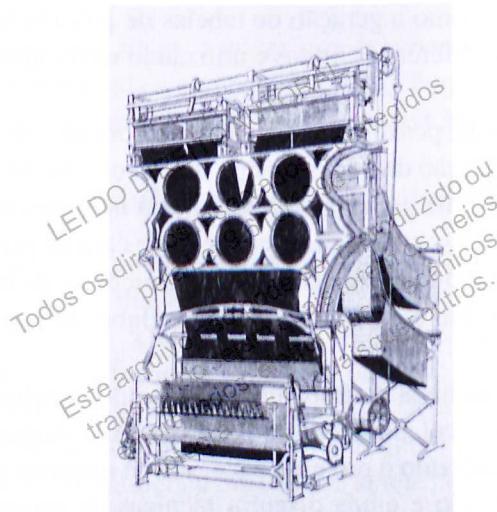


Figura 2.7 O tear automático de Jacquard.

Charles Babbage, em 1822, na Inglaterra, começou a projetar uma máquina a vapor programável, a *máquina de diferenças*, para realizar os cálculos de tabelas de navegação, que apresentavam sérias discrepâncias nas operações (veja a Figura 2.8).

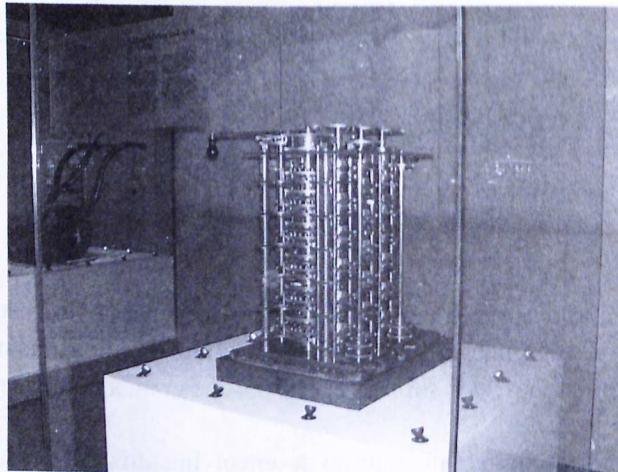


Figura 2.8 A máquina de diferenças de Babbage.

A máquina de diferenças era capaz de realizar somente as adições. No entanto, os cálculos mais sofisticados, como a geração de tabelas de polinômios, podiam ser feitos com o auxílio da técnica de diferenças finitas e utilizando um conjunto finito de posições de memória dessa máquina.

Dez anos depois, Babbage pensou em generalizar o conceito de sua máquina de diferenças para suportar a realização de qualquer tipo de cálculo, mesmo que ela não tivesse sido construída especificamente para isso. Denominada *máquina analítica*, tinha como princípio básico a programação; utilizando o conceito de cartões perfurados de Jacquard, a máquina seria alimentada com cartões contendo instruções e dados que seriam, então, processados pela máquina. Foi com esse projeto que Babbage ficou conhecido como o pai da computação.

Apesar de a máquina analítica de Babbage não ter sido concluída por descredito de seus financiadores, a grande colaboradora de Babbage, Ada Augusta King, a condessa de Lovelace, além de ter traduzido o projeto conceitual da máquina para a língua inglesa, propôs programas de exemplo e ainda discutiu técnicas de programação para aquela máquina. Ada Augusta tornou-se, então, a primeira programadora do mundo.

Outra contribuição contemporânea, mas de caráter matemático, foi o desenvolvimento de um sistema de lógica simbólica e de raciocínio feito por George Boole, em

1854. A *lógica booleana*, como ficou conhecida, é até hoje usada para o projeto de circuitos digitais utilizados nos computadores.

Chegando próximo ao século XX, em 1890, o norte-americano Hermann Hollerith projetou um equipamento para auxiliar na realização do censo daquele ano. A máquina, chamada de *tabulador eletromecânico*, processava automaticamente cartões perfurados, permitindo assim a contagem do número de habitantes (veja a Figura 2.9).



Figura 2.9 O tabulador eletromecânico de Hollerith.

A partir dessa máquina, surge o nome *processamento de dados*. Com o sucesso de sua máquina, Hollerith funda a companhia CTR (Computing-Tabulating-Recording), que, em 1924, passa a se chamar *International Business Machine* ou apenas IBM.

2.2.2 Primeira geração – Computadores a válvula e relé

Todos os direitos reservados e protegidos.
É proibida a reprodução ou
transmissão eletrônica de qualquer parte
deste documento sem autorização.

Os primeiros computadores eletrônicos começaram a surgir na década de 1930. Entre 1935 e 1938, Konrad Zuse, em Berlim, projetou e construiu uma série de máquinas eletromecânicas baseadas em relés. Um relé é um dispositivo que, se excitado por uma corrente elétrica, é capaz de fechar um contato, servindo assim como uma chave “liga-desliga”. As máquinas reconhecidas como Z-1, Z-2, Z-3 e Z-4 (veja a Figura 2.10) só foram conhecidas fora da Alemanha após o término da Segunda Guerra Mundial. Essas máquinas utilizavam aritmética binária e já apresentavam uma organização interna similar à dos computadores modernos.

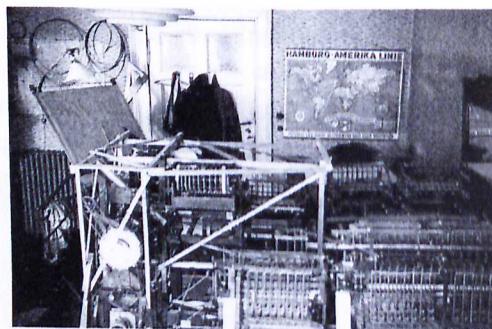


Figura 2.10 O computador Z-1 de Zuse.

Paralelamente, nos Estados Unidos, entre 1936 e 1939, John Vincent Atanasoff e John Berry desenvolveram uma máquina baseada em válvulas, denominada *ABC*, com o propósito de resolver conjuntos de equações lineares da Física (Figura 2.11).

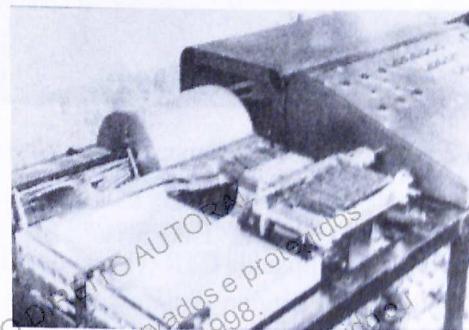


Figura 2.11 O computador ABC de Atanasoff e Berry.

A válvula é um dispositivo puramente eletrônico que, como o relé, funciona como uma chave, porém com velocidade dez mil vezes mais rápida. Da mesma forma que as máquinas de Zuse, a máquina ABC apresentava alguns conceitos encontrados em computadores modernos, como uma unidade aritmética e uma unidade de memória re-generativa, e operava com aritmética binária.

Nessa época, uma grande contribuição teórica foi dada pelo matemático inglês Alan Turing. Ele definiu o conceito intitulado *Máquina Universal de Turing*, estabelecendo um dispositivo teórico capaz de executar qualquer algoritmo descrito, definindo assim as bases para o estudo da computabilidade: um algoritmo computável é aquele que pode ser executado por uma máquina de Turing.

No período da Segunda Guerra Mundial, o computador torna-se uma ferramenta necessária para auxiliar no cálculo de tabelas de balística para canhões navais e artilharia antiaérea. Nos Estados Unidos, destaca-se nesse período o computador eletromecânico Harvard Mark-I de 1944 (veja a Figura 2.12), concebido por Howard Aiken e implementado pela IBM como ASCC (Automatic Sequence Control Calculator).

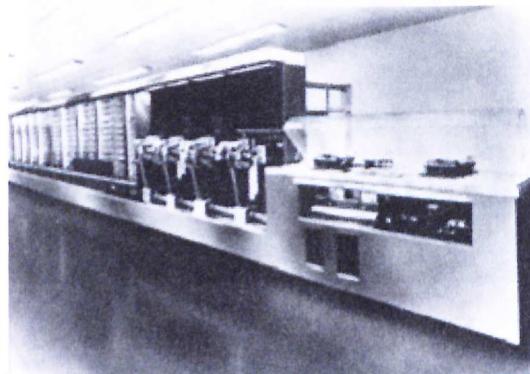


Figura 2.12 O computador Harvard Mark-I de Aiken.

Essa máquina não possuía o conceito de programa armazenado: o programa era “carregado” por meio de uma fita perfurada, executando as instruções durante sua leitura. Ocupava 120 m², continha milhares de relés e conseguia multiplicar números de dez dígitos em três segundos.

Na Inglaterra, outro grande problema era a decifração de códigos secretos alemães. Um projeto secreto de computador denominado *Colossus* (veja a Figura 2.13), foi desenvolvido entre 1940 e 1944 com o intuito de auxiliar na quebra de códigos da máquina alemã Enigma e foi revelado somente em 1970.

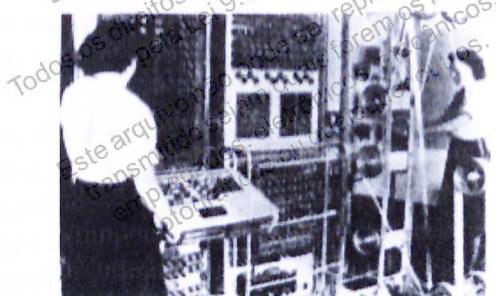


Figura 2.13 O computador britânico Colossus.

Em 1946 é apresentado o computador Eniac (Electronic Numeric Integrator and Calculator), cujo desenvolvimento iniciou-se em 1943, liderado por J. Presper Eckert e John Mauchly (veja a Figura 2.14).

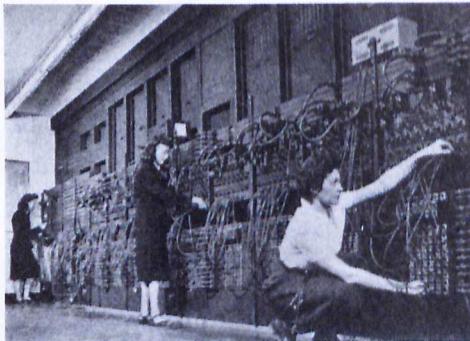


Figura 2.14 O computador Eniac.

O Eniac continha 18 mil válvulas, pesava 30 toneladas e era capaz de realizar 5 mil adições e subtrações e 300 multiplicações por segundo. Possuía uma memória pequena e seus programas eram configurados por cabos, o que tornava complexa a tarefa de programar essa máquina.

Em 1945, nos Estados Unidos, o matemático húngaro John von Neumann, consultor do projeto Eniac, propôs uma arquitetura que seria seguida por todas as gerações de computadores (mais detalhes na Seção 2.4). Ele propôs o conceito de *programa armazenado*, ou seja, a memória do computador armazenaria tanto as instruções a ser executadas quanto os dados a ser processados. Dessa forma, as instruções poderiam ser facilmente modificadas sem a necessidade de alterar as ligações com os cabos ou outros dispositivos. Outro benefício desse conceito é que tanto as instruções quanto os dados seriam armazenados segundo uma única representação, de modo que as instruções seriam executadas da mesma forma que os dados, permitindo, assim, modificações automáticas dessas instruções.

Essa arquitetura dividia o computador em *unidade central de processamento, memória principal e dispositivos de entrada e saída*, e ficou conhecida como a *arquitetura de Von Neumann*, sendo utilizada inicialmente no computador Edvac (Electronic Discrete Variable Computer), que se tornou operacional em 1951 (veja a Figura 2.15), no computador IAS (Institute for Advanced Study, da Universidade de Princeton), de 1952 (veja a Figura 2.16) e no computador Edsac (Electronic Delay Storage Automatic Calculator) da Universidade de Cambridge, em 1949 (veja a Figura 2.17).

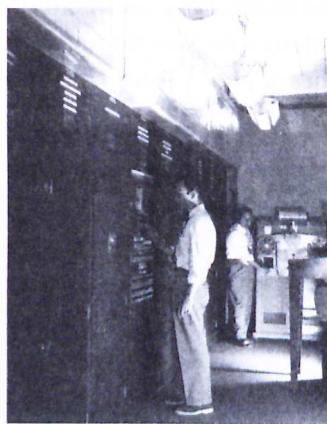


Figura 2.15 O computador Edvac.

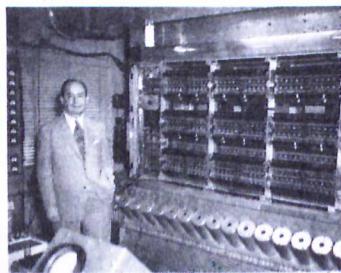


Figura 2.16 John von Neumann e o computador IAS.



Figura 2.17 O computador Edsac.

Esses desenvolvimentos desencadearam, no início dos anos 50, a criação de outros computadores baseados na arquitetura de Von Neumann. Destes, o primeiro computador que obteve sucesso comercial nessa época foi o Univac (Universal Automatic Computer), de 1951 (veja a Figura 2.18).



Figura 2.18 O computador Univac.

Em vez de válvulas, o Univac empregava diodos de cristal, que tornava sua velocidade superior aos contemporâneos valvulados. Além disso, foi o primeiro computador a contar com unidades de equipamentos periféricos independentes, como teletipos e impressoras, além de uma sofisticada unidade de armazenamento em fita.

A IBM demorou um pouco mais para lançar outros computadores após o ASCC. Em 1953, a empresa lançou o modelo 701, para processamento científico. Em 1956, o modelo 704, que tinha o dobro de memória do 701 e contava com hardware para realização de cálculos com ponto flutuante, e, em 1958, seu último computador valvulado, o modelo 709, similar ao 704 (veja a Figura 2.19).



Figura 2.19 O computador IBM 709.

2.2.3 Segunda geração – Computadores transistorizados

Com a invenção do *transistor* em 1947 por William Shockley, John Bardeen e Walter Brattain, foi possível revolucionar a construção de computadores, oferecendo mais confiabilidade e velocidade do que as válvulas. Um transistor é um dispositivo semicondutor, isto é, conduz corrente elétrica de acordo com uma tensão aplicada e por isso pode ser utilizado como uma chave, como a válvula e o relé. Em comparação com a válvula e o relé, o transistor é mais confiável, menor e mais rápido. No entanto, essa tecnologia demoraria dez anos até figurar em um computador.

O primeiro computador transistorizado da história foi o *TX-0*, construído no Massachusetts Institute of Technology, em 1957, para servir como base de testes para o computador *TX-2*. Apesar de o *TX-2* não ter obtido êxito, um engenheiro do MIT, chamado Ken Olsen, fundou uma companhia para manufaturar uma versão comercial do *TX-0*, a DEC (Digital Equipment Company), que, em 1961, lançou o PDP-1, o primeiro minicomputador comercial. O sucesso do PDP-1 fez com que a DEC lançasse o PDP-8 em 1965 e depois outras famílias baseadas nesse modelo até 1990.

A IBM, por sua vez, lançou os modelos 7090 e 7094, que eram as versões transistorizadas baseadas no modelo 709. Apesar de serem muito mais rápidos que seu contemporâneo da DEC, o PDP-1, eram muito mais caros. A IBM então criou o modelo 1401, mais barato e tão rápido quanto os modelos 7090 e 7094, sendo destinado principalmente a aplicações comerciais (veja a Figura 2.20).



Figura 2.20 O computador IBM 1401.

O ápice dos computadores transistorizados foi o CDC-6600, um supercomputador criado pela CDC (Control Data Corporation) em 1964 (veja a Figura 2.21).



Figura 2.21 O computador CDC-6600.

O CDC-6600 distingua-se dos demais de sua época por descarregar o processamento da CPU pelo uso de pequenos computadores auxiliares que tratavam de outras tarefas como entrada e saída de dados e gerenciamento de tarefas de forma paralela. Dessa forma, conseguia executar até dez instruções simultaneamente.

2.2.4 Terceira geração – Computadores com circuitos integrados

A evolução natural do transistor foi o surgimento do circuito integrado em 1958, criado por Robert Noyce. Um circuito integrado pode conter dezenas de transistores, executando desde funções lógicas simples até as funções mais complexas. A vantagem está no pequeno espaço ocupado, robustez a interferências elétricas e baixo consumo.

A pioneira no uso de circuitos integrados em computadores foi a IBM, que estabeleceu uma linha de computadores – o sistema 360 – para substituir os tipos 7094 e 1401 (veja a Figura 2.22).

O sistema 360 era vendido em diversos modelos para atender às exigências de custo e desempenho e foi lançado em 1965. Foi o primeiro computador de propósito geral produzido, atendendo tanto a processamento científico quanto a comercial, e as versões posteriores baseadas nessa arquitetura são utilizadas até hoje.

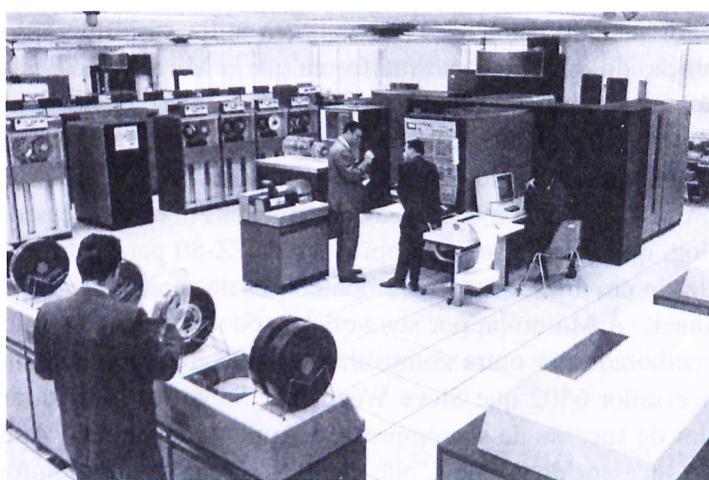


Figura 2.22 O computador IBM 360.

O primeiro minicomputador com circuitos integrados foi o PDP-11 da DEC, uma evolução do PDP-8. Essa máquina obteve um grande sucesso de vendas, sendo adotada principalmente por universidades.

2.2.5 Quarta geração – Computadores com *chips* VLSI

A quarta geração é marcada pelos microprocessadores. Um microprocessador é um dispositivo eletrônico encapsulado em um *chip* que possui internamente uma unidade de controle, uma unidade lógico-aritmética e uma memória interna, englobando as unidades funcionais básicas de um computador (leia a Seção 2.4). O primeiro microprocessador que surgiu foi o Intel 4004, de 1971, originalmente desenvolvido para uma empresa japonesa de calculadoras. O Intel 4004 era um microprocessador de 4 bits, isto é, poderia utilizar até 2^4 posições de memória.

Rapidamente a Intel percebeu que o 4004 poderia ser usado em outros projetos e então decidiu lançar um microprocessador mais poderoso, o 8008 de 1972. Foi com esse microprocessador que surgiu o primeiro microcomputador do mundo, na França, o *Micral* de 1973, que não obteve êxito. O Micral era programado diretamente com números binários, não possuindo nenhum periférico de entrada ou saída. As versões posteriores contavam com um software montador (*assembler*) desenvolvido por Philippe Kahn, que mais tarde fundaria a Borland International.

Com a popularização do microprocessador, surgiram diversos *kits* que podiam ser comprados em lojas e montados em casa. O mais famoso desses foi o *Altair 8800*,

que, em 1974, era vendido por US\$ 439 e também utilizava o microprocessador Intel 8080, uma evolução do 8008. Da mesma forma que o Micral, não possuía periféricos de entrada, nem de saída, mas contou depois com um interpretador Basic, escrito por Bill Gates e Paul Allen, de uma pequena empresa (na época) denominada Microsoft. Surgiram, assim, os primeiros microcomputadores.

Outras empresas também começaram a construir seus microprocessadores. Em 1974 é fundada a Zilog, que lança o seu microprocessador Z-80 para concorrer com a Intel. O Z-80 foi utilizado em diversos microcomputadores de sucesso, por exemplo, o TRS-80 da Radio Shack. A Motorola, por sua vez, lançou nessa época o microprocessador 6500, que foi melhorado por outra companhia, a MOS Technology, lançando o 6502. Foi o microprocessador 6502 que Steve Wozniak e Steve Jobs utilizaram no primeiro microcomputador de sucesso de sua empresa, a Apple Computer: o Apple II de 1976. O Apple II foi o sucessor do Apple I, que em 1975 já usava um monitor de TV como dispositivo de saída, um teclado para efetuar a entrada de dados e uma unidade de cassete para armazenar programas e dados. Foi para o Apple II que foram escritos os primeiros softwares utilitários para um microcomputador: a planilha Visicalc e o editor de textos Wordstar.

Em 1981, a IBM decidiu investir em microcomputadores com o lançamento do IBM-PC (Personal Computer). Em vez de desenvolver todo o projeto, a IBM resolveu montá-lo a partir de peças e software fornecidos por terceiros e ainda disponibilizou todo o seu projeto para empresas interessadas nele. O primeiro PC era baseado no microprocessador Intel 8088, de 16 bits e velocidade de *clock* de 4.77 MHz. Possuía 16 K de memória RAM padrão, expansível até 256 K, um ou dois acionadores de disquete de 160 K e monitor opcional. O preço inicial era de US\$ 1.565, correspondente a aproximadamente US\$ 4 mil atualmente. A disponibilidade do projeto do PC fez com que outros fabricantes iniciassem a construção massiva de PCs, o que tornou esse equipamento popular e mais vendido até hoje, embora possua concorrentes à altura ou mesmo superiores.

A tecnologia VLSI (Very Large Scale Integration), que surgiu na década de 1980, permitiu que milhões de transistores pudessem ser encapsulados em uma única pastilha, também denominada *chip*. Dessa forma, foi possível criar pastilhas mais complexas e poderosas, reduzindo ainda mais o tamanho dos computadores e aumentando sua velocidade e processamento. Assim, a Intel ampliou sua família de microprocessadores, lançando depois do Intel 8088 os microprocessadores Intel 80286, Intel 386, Intel 486, Pentium, Pentium MMX, Pentium II, Pentium III e então o Pentium IV, sendo que o projeto deste último, após diversas atualizações e modificações, vem servindo de base para os *chips* atuais da Intel.

É claro que não se deve esquecer do outro grande concorrente do IBM-PC, o Apple Macintosh. A Apple, apesar de ter sido a pioneira na popularização dos microcomputadores, perdeu a liderança assim que a IBM abriu o projeto de seu IBM-PC. O Apple II original foi substituído pelo Macintosh em 1984, que, depois de sucessivas versões, tornou-se uma família de máquinas, abrangendo desde *notebooks* até máquinas poderosas para o trabalho em rede. A linha Macintosh atual da Apple também é baseada nos *chips* Intel.

A história do desenvolvimento do computador é incompleta sem a história do desenvolvimento do software. Reservou-se para o Apêndice A um resumo com os fatos históricos relevantes da história do computador e de seu software.

2.3 A representação da informação em um computador

2.3.1 A eletrônica digital do computador

Os circuitos eletrônicos de um computador moderno operam com sinais de dois níveis distintos ou *binário*. A razão de se utilizar circuitos que operam sob a forma binária e não decimal está no fato de que essa solução é simples e de baixo custo de implementação. Além disso, com circuitos digitais binários e utilizando-se dos resultados da *lógica booleana* (veremos sua aplicação em programação no Capítulo 3), é possível implementar em hardware qualquer tipo de função lógica, permitindo, assim, a construção de diversos tipos de circuitos empregados em um computador, como registradores, memórias, microcontroladores e o próprio microprocessador.

O ingrediente básico das pastilhas (*chips*) utilizadas no projeto dos circuitos do computador e que permite o chaveamento entre dois níveis lógicos é o *transistor*. Um transistor é um componente eletrônico criado a partir de um material *semicondutor*, isto é, possui a propriedade de conduzir a corrente elétrica somente após uma tensão conveniente ter sido aplicada em seus terminais. Dessa maneira, uma das aplicações do transistor é a de servir como uma chave “liga–desliga”, conforme ilustrado na Figura 2.23.

Nessa figura, em (a) o transistor é representado por um dispositivo de três terminais, conhecidos como *coletor*, *base* e *emissor*. Ao se aplicar uma tensão alta (V_H) em sua base, o transistor permite a condução de corrente elétrica entre o coletor e o emissor, fechando o circuito e produzindo como saída (V_O) uma tensão próxima de zero. Por outro lado, se for aplicada em sua base uma tensão convenientemente baixa ($V_L < V_H$), permitirá a passagem de uma corrente elétrica muito baixa entre os terminais coletor-emissor que, para fins práticos, se comporta como um circuito aberto.

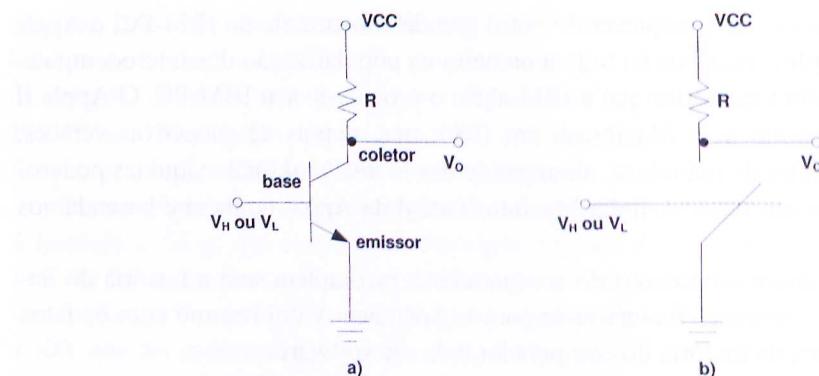


Figura 2.23 O transistor como chave.

Esse funcionamento pode ser descrito de forma simplificada como em (b). Nesse caso, o transistor é representado por uma chave controlada por tensão: se a tensão de entrada for alta, a resposta será um nível baixo; e se a tensão de entrada for baixa, a resposta será um nível alto. Convencionou-se chamar de nível alto o símbolo 1 e de nível baixo, 0. Esse exemplo ainda demonstra uma função lógica, o *inversor*: se a entrada for 1, a saída será 0; se a entrada for 0, a saída será 1.

Outras funções lógicas podem ser definidas da mesma forma, agrupando os transistores de forma conveniente. Não se intenciona aqui prolongar mais esse assunto. De qualquer maneira, lembre-se de que todos os dados armazenados e processados em um computador são traduzidos em sinais elétricos binários, ou seja, em um conjunto finito de 0s e 1s. Isso conduz ao conceito de *bit*.

2.3.2 Conceitos de bits e seus múltiplos

A palavra bit² ou *binary digit* representa de forma lógica um estado “ligado/desligado” ou *binário* existente em dispositivos eletrônicos digitais dos circuitos de um computador, como em registradores e memórias, por exemplo. Convencionou-se que um bit “ligado” é representado pelo símbolo 1 e “desligado”, por zero.

O uso e a manipulação de números binários é similar ao dos números decimais. Aplica-se o mesmo conceito do sistema decimal, em que a posição de cada dígito de um número representa a potência da base (nesse caso, 2) que ele está figurado.

²A palavra bit, na realidade, surgiu na teoria matemática da informação, significando uma unidade básica de informação cuja incerteza é de 50%.

Por exemplo, o número 10101_2 binário pode ser entendido como³:

$$10101_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 21_{10}$$

Dessa forma, o número binário 10101_2 corresponde ao número decimal 21_{10} . De forma semelhante, pode-se converter um número decimal em binário, dividindo esse número sucessivamente por dois, até que o quociente seja zero. O número binário correspondente é obtido lendo-se os restos das divisões, da última para a primeira. Por exemplo, para converter o número 25_{10} em seu correspondente binário, utilizamos a seguinte sequência de operações:

$$\begin{array}{r} 25 \\ -24 \quad \boxed{2} \\ \hline 1 \quad \boxed{-12} \\ \hline 0 \quad \boxed{6} \\ \hline 0 \quad \boxed{-6} \\ \hline 0 \quad \boxed{3} \\ \hline 0 \quad \boxed{-2} \\ \hline 1 \quad \boxed{1} \\ \hline 1 \quad \boxed{-0} \\ \hline 1 \end{array}$$

Assim, o número binário correspondente ao decimal 25_{10} é 11001_2 (confira).

Embora a unidade fundamental de informação do computador seja o bit, na prática utilizamos seus múltiplos, como o **byte**. Um byte representa o mesmo que oito bits e para fins de programação é o menor dado que se pode manipular diretamente. Os múltiplos do byte também são utilizados para representar as quantidades manipuladas e, geralmente, são o **kilobyte (KB)**, o **megabyte (MB)** e o **gigabyte (GB)**. Note que essas quantidades não são potências de dez e sim de dois, conforme ilustrado na Tabela 2.4.

Tabela 2.4 Os múltiplos do byte

Nome	Valor
Quilobyte (KB)	$1.024 \cdot (2^{10})$ bytes
Megabyte (MB)	$1.048.576 \cdot (2^{20})$ bytes
Gigabyte (GB)	$1.073.741.824 \cdot (2^{30})$ bytes

Os tipos de informação manipulados pelo computador durante a execução de um programa são os dados e as instruções que operam sobre esses dados. Na memória são sempre representados por bits. Dentre os tipos de dados mais conhecidos temos

³Para separar números binários de decimais, se anotará um índice indicando qual é sua base, 2 ou 10.

os *caracteres*, as *cadeias de caracteres*, as *imagens* e os *sons* que serão discutidos nas seções a seguir. A representação binária das instruções é assunto da Seção 2.4.

2.3.3 Caracteres e cadeias de caracteres

Os caracteres são símbolos digitados pelo usuário durante a execução de seu programa ou que ainda podem ser constantes presentes no texto do programa. Envolvem as letras maiúsculas e minúsculas (A, B, C, ..., Z, a, b, c, ..., z), os números decimais (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), símbolos especiais e de operação (+, -, *, /, SP (espaço), ", # etc.) e símbolos de controle (DEL (apagar), EOF (fim de arquivo), CR (retorno de carro) etc.) que não são visíveis na tela, mas que sinalizam alguma operação realizada no computador.

No texto, para se diferenciar um caractere de uma variável, o caractere será denotado pelo símbolo que o representa delimitado por *apóstrofes*, como, por exemplo, o caractere 'A', o caractere 'a', o caractere '1', e assim por diante.

É importante observar que o caractere '1' não representa o mesmo que o número inteiro 1. O anterior representa o símbolo '1', que foi, por exemplo, digitado pelo usuário em um programa enquanto o último representa um número inteiro, que pode ser manipulado em operações aritméticas.

Os caracteres são representados como números binários dentro do computador por meio de uma *tabela de caracteres*, que codifica os caracteres em números binários apropriados. Existem dois grandes sistemas para representar os conjuntos de caracteres: o EBCDIC (*Extended Binary Coded Decimal Interchange Code*) e o ASCII (*American Standard Code for Information Interchange*). O EBCDIC surgiu com o lançamento do computador IBM-360 na década de 60, e o ASCII foi definido para ser um padrão para a indústria de computadores e é o mais utilizado atualmente, em virtude de sua adoção pelos fabricantes de microcomputadores.

O conjunto de códigos ASCII original (128 símbolos) está descrito na Tabela 2.5. Para facilitar sua compreensão, os códigos dos caracteres estão representados apenas na base decimal.

Observe que, a partir dessa tabela, o código do caractere 'A' é 65, do caractere 'a', 97 e que as letras tanto maiúsculas quanto minúsculas estão representadas de forma contígua, isto é, o código de 'B' é 66, de 'b', 98 e assim por diante. Nessa tabela, percebe-se, também, a diferença entre o caractere '1' do número inteiro 1: o anterior vale 49_{10} (ou 110001_2) e o último é o próprio inteiro 1_{10} (ou 1_2).

As cadeias de caracteres representam uma sequência de caracteres que em um programa podem representar mensagens e textos. Uma cadeia de caracteres é também conhecida como *string*, que, traduzido do inglês, significa *sequência de elementos*. Na

memória do computador, uma cadeia de caracteres é representada pela sequência correspondente de seus códigos binários.

Tabela 2.5 A tabela de códigos ASCII.

Decimal	Caractere	Decimal	Caractere	Decimal	Caractere	Decimal	Caractere
0	NUL	32		64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(72	H	104	h
10	HT	41)	73	I	105	i
11	LF	42	*	74	J	106	j
12	VT	43	+	75	K	107	k
13	FF	44	,	76	L	108	l
14	CR	45	-	77	M	109	m
15	SO	46	.	78	N	110	n
16	SI	47	/	79	O	111	o
17	DLE	48	0	80	P	112	p
18	DC1	49	1	81	Q	113	q
19	DC2	50	2	82	R	114	r
20	DC3	51	3	83	S	115	s
21	DC4	52	4	84	T	116	t
22	NAK	53	5	85	U	117	u
23	SYN	54	6	86	V	118	v
24	ETB	55	7	87	W	119	w
25	CAN	56	8	88	X	120	x
26	EM	57	9	89	Y	121	y
27	SUB	58	:	90	Z	122	z
28	ESC	59	;	91		123	(
29	FS	60	?	92		124)
30	GS	61	>	93		125	~
31	RS	62	<	94		126	DEL
	US	63		95			

Considerando, por exemplo, a cadeia de caracteres:

A carta de programar.

Aqui o espaço em branco está escrito como _ para melhor visualização. Na memória do computador, essa cadeia de caracteres seria representada por números binários conforme a Tabela 2.6.

Tabela 2.6 Codificação de uma cadeia de caracteres.

Caractere	A	á	a	r	t
Decimal	65	32	97	114	116
Binário	01000001	00100000	01100001	01110010	01110100

Caractere	e	á	d	e	á
Decimal	101	32	100	101	32
Binário	01100101	00100000	01100100	01100101	00100000

Caractere	p	r	ó	g	r
Decimal	112	114	111	103	114
Binário	01110000	01110010	01101111	01100111	01110010

Caractere	a	m	a	r	.
Decimal	97	109	97	114	46
Binário	01100001	01101101	01100001	01110010	00101110

Note que a representação binária dos caracteres desse exemplo ocupa sempre oito bits. Embora com um conjunto de 127 elementos seja necessário somente sete bits para codificá-los⁴ ($2^7 = 128$), pelo fato de que o byte é a menor unidade de informação que pode ser manipulada em um programa, cada caractere dessa tabela ocupa de fato um byte. A evolução da tabela ASCII original é a tabela **ASCII estendida**, possuindo 256 elementos e cada elemento ainda ocupa um byte⁵. Então, para fins práticos, se assumirá que um caractere ocupa um byte de memória.

2.3.4 Imagens

As imagens no computador são versões digitalizadas de imagens reais ou sintetizadas por algum software gráfico. Uma imagem, conforme percebida por nosso cérebro, é o resultado da interação das ondas eletromagnéticas componentes de uma fonte de luz refletida por uma imagem e que, ao alcançar células fotossensíveis denominadas *células-cone* existentes no fundo da retina, produzem e enviam sinais ao cérebro que então “formam” a imagem que se está vendo.

Existem três tipos de células-cone: aquelas que são sensíveis às frequências das cores

⁴A razão é que existem dois valores possíveis para ocupar cada posição de um bit, assim, com sete bits obtém-se $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^7 = 128$ valores distintos.

⁵Isso não vale para esquemas de codificação tipo Unicode que propõem o uso de bytes múltiplos para representar qualquer caractere ou símbolo de qualquer língua do planeta.

verde, azul e vermelha. Dessa forma justifica-se um modelo de cor que é amplamente utilizado por artistas e também pela maior parte dos programas gráficos: o modelo RGB (RED-GREEN-BLUE). Nesse modelo, qualquer cor pode ser formada pela adição de intensidades adequadas de vermelho, verde e azul. Por esse motivo, essas três cores são denominadas *cores aditivas primárias*.

Voltando-se ao computador, o responsável pela geração das cores que são percebidas em um monitor de vídeo é o *subsistema de vídeo*. O subsistema de vídeo é normalmente uma placa “espetada” na placa-mãe do computador ou um circuito embutido na mesma e se conecta ao monitor de vídeo por um cabo de vídeo.

A placa de vídeo dita o *modo de exibição* que se obterá no monitor. No modo *texto*, as informações a serem exibidas são organizadas por desenhos de caracteres e dividem a tela em uma matriz ocupada tipicamente por 80 colunas e 25 linhas. Nesse modo, os programas exibem suas informações escrevendo os caracteres em posições dentro do espaço de 80×25 posições possíveis. Um exemplo de programa que utiliza esse modo é o programa *Edit*, um editor de textos distribuído juntamente ao sistema operacional Microsoft Windows e que é executado no ambiente MS-DOS, conforme a Figura 2.24.



Figura 2.24 Tela em modo texto do programa *Edit*.

Cada caractere no modo texto é por si só representado por uma matriz de pontos – *pixels* ou *picture elements* –, que, organizados de forma conveniente, formam os caracteres que conhecemos. No modo texto, os atributos de cor para os caracteres são de apenas dois tipos: a cor de fundo (*background color*) e a cor do caractere (*foreground color*). Quando se alteram esses atributos de cor no modo texto, alteram-se a cor de todos os pixels correspondentes no caractere em questão. No modo texto, não é possível alterar a cor de um pixel individualmente.

Já no modo *gráfico*, a imagem percebida é formada por pixels que são individualmente acessados. Dessa forma, todas as informações visuais desenhadas na tela do computador são organizadas em uma grande matriz de pixels, cada um com sua própria cor. O tamanho dessa matriz depende da *resolução* da placa de vídeo. A resolução indica o número máximo de pixels que se visualizará, em linhas e colunas, e o produto desses números fornece a quantidade total de pixels que podem ser exibidos na tela.

A resolução de vídeo é uma propriedade da placa de vídeo que está instalada no computador. Assim, quando se fala em uma resolução de vídeo de 800×600 significa que a placa de vídeo em questão é capaz de exibir 800 vezes 600 pixels, ou 480.000 pixels. Um exemplo de uma tela em modo gráfico é representado pelas janelas dos aplicativos do sistema operacional Windows, nesse caso pelo programa de pintura *Paint*, conforme observado na Figura 2.25.

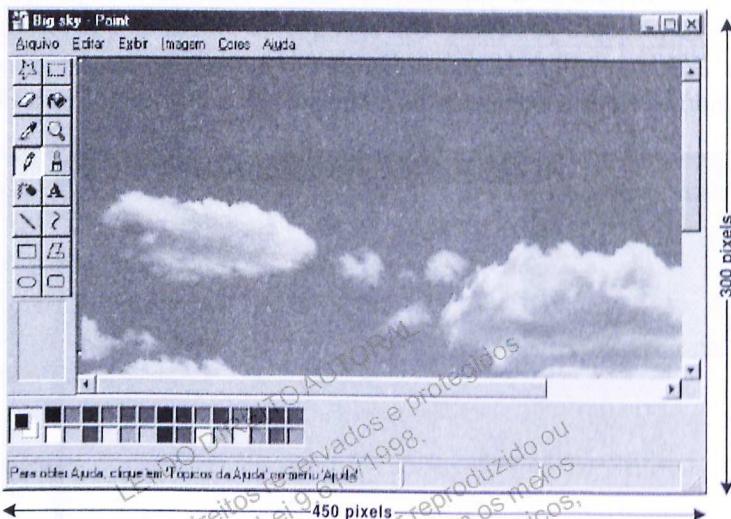


Figura 2.25 Tela em modo gráfico do aplicativo *Paint* do Windows.

No modo gráfico, emprega-se tipicamente a codificação da informação de cor, utilizando o modelo RGB já citado. Nesse caso, as cores possíveis são criadas a partir de quantidades de vermelho, verde e azul convenientes. Para o computador, esses números são representados por quantidades inteiras, no formato binário. O número total de cores que se pode apresentar no modo gráfico depende de um parâmetro do adaptador gráfico conhecido como *profundidade* de cor. A profundidade de cor é o número de bits utilizado para representar as cores oferecidas pelo adaptador gráfico.

Assim, ao comprar um adaptador gráfico, pode-se observar em sua especificação esse número. Atualmente é comum as placas com 32 bits de profundidade de cor, que são divididos nos matizes vermelho, verde e azul. Logo, uma cor de cada pixel no modo gráfico é representado por três números binários com oito bits cada (um byte, portanto), e cada um desses bytes representa a intensidade de vermelho, verde e azul na cor.

A quantidade de informação associada a uma imagem depende do seu tamanho em bits e da profundidade de cor utilizada. Uma imagem como a da Figura 2.25, se representada com uma informação de cor de 32 bits, vai ocupar $450 \times 350 \times 32 / 8$ bytes = 1.920.000 bytes ou ainda 1.8 MBytes. Observe a necessidade de compressão de dados.

Portanto, seja no modo texto ou no modo gráfico, as informações desenhadas na tela do computador são representadas por dois atributos:

- Localização (coluna, linha do caractere em modo texto; coluna, linha do pixel em modo gráfico).
- Cor (atributos de fundo/frente em modo texto; cor RGB pixel em modo gráfico).

E essas informações deverão ser armazenadas como números binários na memória do computador, segundo algum esquema de codificação proposto pelo sistema operacional e dispositivo gráfico. Por exemplo, considere uma imagem de 4×4 pixels, conforme a Figura 2.26.

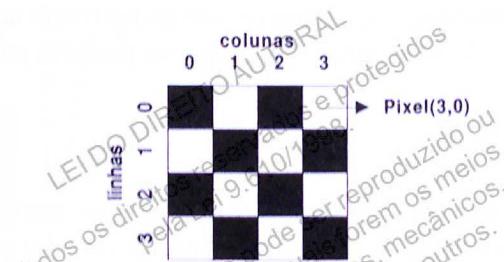


Figura 2.26 Exemplo de uma imagem.

Essa imagem contém somente duas cores: preto e branco. Presume-se que essas cores sejam representadas em uma placa com 32 bits de profundidade de cor e que possuam os seguintes códigos binários:

- Preto: 00000000000000000000000000000000.
- Branco: 11111111111111111111111111111111.

Tabela 2.7 Codificação de uma imagem.

Pixel	Código
(0,0)	00000000000000000000000000000000
(0,1)	11111111111111111111111111111111
(0,2)	00000000000000000000000000000000
(0,3)	11111111111111111111111111111111
(1,0)	11111111111111111111111111111111
(1,1)	00000000000000000000000000000000
(1,2)	11111111111111111111111111111111
(1,3)	00000000000000000000000000000000
(2,0)	00000000000000000000000000000000
(2,1)	11111111111111111111111111111111
(2,2)	00000000000000000000000000000000
(2,3)	11111111111111111111111111111111
(3,0)	11111111111111111111111111111111
(3,1)	00000000000000000000000000000000
(3,2)	11111111111111111111111111111111
(3,3)	00000000000000000000000000000000

Na memória do computador, essas informações seriam armazenadas de forma contígua, como indicado na Tabela 2.7.

Nesse exemplo, considerou-se que as matrizes de dados foram armazenadas segundo o armazenamento contíguo de suas linhas (poderiam ser também por suas colunas).

2.3.5 Sons

Da mesma forma que as imagens, os sons são informações analógicas, contínuas no tempo e amplitude. A forma de onda de um som (música ou fala) pode ser considerada como a soma de diversas formas de onda senoidais, cada uma com uma *frequência* e *amplitude* particular. Um exemplo de um sinal de som analógico está na Figura 2.27.

O problema do armazenamento de informações de som no computador é análogo ao das imagens. Por ser uma forma de onda contínua, seu armazenamento na forma como é encontrada na natureza é inviável no computador. A solução encontrada e aplicada por todos os subsistemas de som do computador foi a *amostragem digital do sinal*.

O processo de amostragem digital de sinais analógicos, em particular o sinal de som, funciona da seguinte forma: o sinal de som é capturado via microfone ou saída de algum dispositivo analógico conectado à placa de som do computador. Em seguida, esse

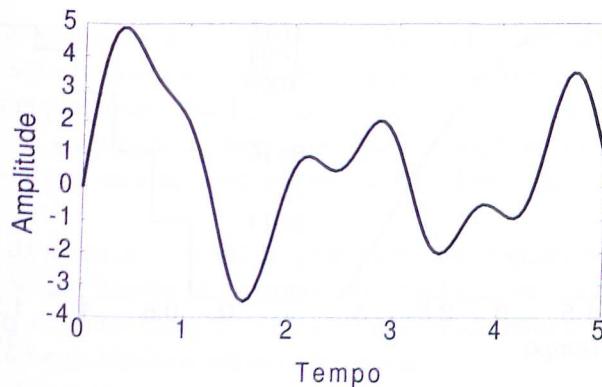


Figura 2.27 Exemplo de uma forma de onda de som.

sinal é *amostrado*, isto é, são coletadas amostras periódicas dele (sua amplitude). Isso é feito por um dispositivo conversor analógico-digital. Na sequência, a cada amplitude amostrada é atribuído um valor binário correspondente, sendo esse processo denominado *quantificação*. Por fim, esse sinal amostrado e quantificado é compactado e então armazenado de forma conveniente.

Por consequência desse processo de “discretização”, a forma de onda resultante apresenta um “serrilhado”, conforme exibido na Figura 2.28.



Figura 2.28 Exemplo de uma forma de onda de som após amostragem.

O processo de quantificação fornece os códigos binários para cada nível discreto. Um exemplo disso é exibido na Figura 2.29.

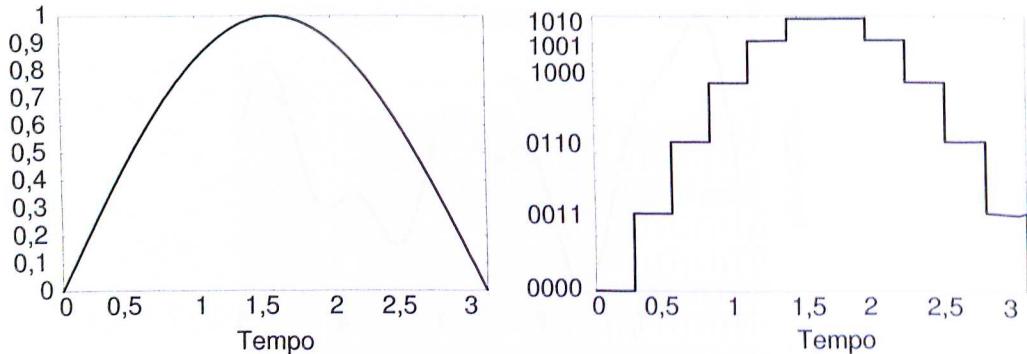


Figura 2.29 Exemplo de uma forma de onda de som após quantificação.

Nessa figura, um trecho ampliado da função $\sin(x)$ no intervalo a é amostrado e quantificado por números de quatro bits a uma taxa de amostragem de três amostras por unidade de tempo. A representação desse sinal no computador poderia ser conforme a Tabela 2.8.

Tabela 2.8 Codificação de um sinal de som.

Amostra	Código
0	0000
1	0011
2	0110
3	1000
4	1001
5	1010
6	1010
7	1001
8	1000
9	0110
10	0011

LEI DO DIREITO AUTORAL
Todos os direitos reservados.
Pela Lei nº 9.610/98.
Este arquivo não pode ser reproduzido ou
transmitido de nenhuma maneira, meios
fotográficos ou eletrônicos, quer outros.

Após ter sido armazenado, o som pode ser novamente reproduzido. Para tanto, aplica-se o caminho inverso: a partir das amostras existentes, o sinal é recomposto e filtrado para remover o efeito do “serrilhado”. Por fim é enviado ao amplificador de saída da placa de som.

A qualidade do som a ser armazenado depende da frequência de amostragem empregada e da resolução da placa de som. Sabe-se pelo critério de Nyquist que a taxa de amostragem mínima para que o sinal possa ser recuperado novamente na forma analógica é de duas vezes a maior frequência existente no sinal original. Para a percepção do ouvido humano, uma taxa de amostragem de 44,1 kHz é suficiente para as falas e músicas.

Já a resolução da placa de som indica quantos níveis de quantificação são possíveis. Por exemplo, uma Sound Blaster 16 emprega uma resolução máxima de 16 bits, ou seja, 2^{16} níveis possíveis de quantificação. Um som amostrado com essa placa é, portanto, representado por números binários inteiros de 16 bits.

Um som de qualidade demanda espaço de armazenamento. Considerando uma taxa de amostragem de 44,1 kHz e uma resolução de 16 bits, um sinal de som de 60 segundos ocupará $44,1 \times 10^3 \times 60 \times 16$ bits, ou seja 42.336.000 bits ou 5 Mbytes! Da mesma forma que as imagens, os sons no computador devem ser comprimidos por algum padrão existente de compressão (por exemplo, MP3).

2.4 A arquitetura de um computador

A arquitetura de um computador representa a maneira na qual seus componentes estão organizados. Da história da computação (Seção 2.2), consagraram-se diversos modelos de arquitetura. Nesta seção será descrito o mais famoso e utilizado deles: o modelo de arquitetura de *Von Neumann*.

A arquitetura de *Von Neumann* surgiu em virtude de seu criador, John von Neumann, que, em 1946, definiu as unidades funcionais básicas para o projeto do computador IAS (veja a Seção 2.2.2). Essas unidades funcionais (ou componentes) estão ilustradas na Figura 2.30 e são:

- UCP (Unidade Central de Processamento) ou CPU (Central Processing Unit): é representada atualmente por um microprocessador e sua função é executar programas armazenados na memória principal. A memória principal ou ainda RAM, (*random access memory*), representa a memória volátil (apagada quando sem energia), utilizada para armazenar instruções e dados de um programa.
- Caminhos (*bus*) de dados: representam uma fiação (impressa na placa-mãe) e seu propósito é transmitir dados, que podem ser endereços de áreas da memória, dados de um programa armazenados na memória e sinais de controle para/de outros componentes externos, como, por exemplo, dispositivos de entrada/saída.

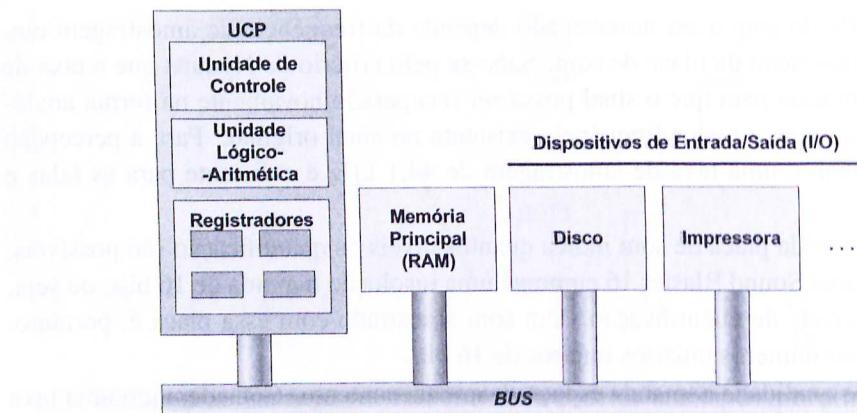


Figura 2.30 Organização típica de um computador.

- Dispositivos de entrada e saída ou dispositivos de I/O (*input/output*): representam interfaces para os dispositivos de entrada e saída, como, por exemplo, mouse, teclado, monitor, disco rígido, entre outros.

Esses componentes são representados por um conjunto de dispositivos eletrônicos digitais, encapsulados em *chips* (pastilhas) e em cartões de expansão que são respectivamente soldados e encaixados a uma placa especial denominada *placa-mãe* (ou ainda *motherboard* ou *mainboard*).

2.5 O funcionamento da UCP na execução dos programas

A UCP é a responsável pela execução dos programas e pelo controle dos outros componentes, como os periféricos de entrada e saída. A UCP é representada nos computadores pessoais pelo *microprocessador*. Simplificando, a UCP é internamente composta pelas seguintes unidades funcionais:

- uma unidade de controle (UC): responsável pela *busca* de instruções na memória principal;
- uma unidade lógico-aritmética (ULA): responsável pela execução de operações aritméticas e lógicas e pela *execução* das instruções provindas da memória principal;
- um conjunto de registradores: representa uma pequena memória (por exemplo, 32 registradores) que serve para armazenar resultados temporários e algumas infor-

mações de controle. Os registradores possuem normalmente um tamanho fixo em bits (por exemplo, 32 bits) e, por estarem situados internamente à UCP, são muito mais rápidos que a memória principal.

A organização interna de uma UCP é representada principalmente pelo seu *caminho de dados (data path)*. Esse caminho indica o fluxo de dados que ocorre internamente na UCP. A Figura 2.31 exibe uma simplificação dessa organização.

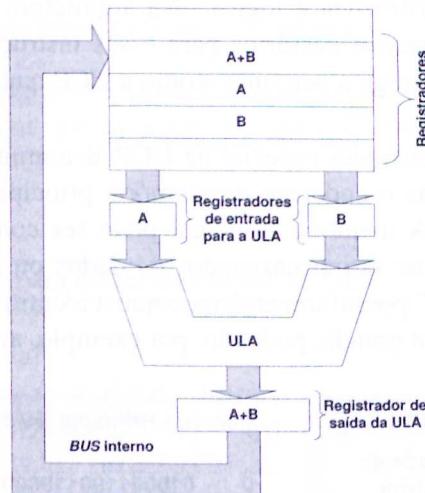


Figura 2.31 Caminho de dados de uma UCP.

O funcionamento do caminho de dados pode ser ilustrado com a operação de adição de dois números inteiros, representados simbolicamente pelas variáveis A e B . Supondo que esses dois números tenham sido armazenados em dois registradores da UCP, no momento da execução da instrução de soma, eles são transferidos para os registradores especiais da ULA, chamados de *registradores de entrada* (veja a Figura 2.31). Os circuitos internos da ULA entram em ação e realizam a soma desses números, sendo armazenados em outro registrador especial da ULA, denominado *registraror de saída*. Por fim, esse número resultante é armazenado em algum outro registrador da UCP, de acordo com a instrução de soma realizada.

Cabe aqui uma observação. Nota-se que a operação descrita não é realizada de uma única vez. Compete aos circuitos da ULA a correta execução dessas etapas de acordo com um *microcódigo* definido em seus circuitos que, por sua vez, é sincronizado por um sinal de *relógio*. Esse sinal de relógio é uma fração do relógio nominal do microprocessador. Por exemplo, um microprocessador com um relógio de 3,0 GHz significa que uma

onda quadrada com período de $1/(3,0 \times 10^9)$ segundos será utilizada como referência para os circuitos digitais do computador.

Esse ciclo de execução e posterior armazenamento do resultado é intitulado *ciclo de caminho de dados* e pode-se afirmar que quanto mais rápido esse ciclo ocorrer, mais rápido é o processador que o abriga.

As instruções que são executadas pela ULA dependem do conjunto de instruções definidos para a UCP em questão. Essas instruções em uma UCP típica devem abrigar pelo menos as dos tipos aritmética e lógica. Na arquitetura de Von Neumann, tanto as instruções quanto os dados necessários para essas instruções são armazenados na memória principal. Assim, surge a pergunta: como a UCP realiza o processo de busca e execução das instruções?

Primeiro, existe um registrador especial na UCP, denominado de *contador de programa* ou CP, que armazena o endereço da memória principal que contém a próxima instrução a ser buscada. A memória principal pode ser considerada como um conjunto de “caixinhas” em que são armazenados os dados ou as instruções (veja a Figura 2.32). Cada “caixinha” possui um endereço que a identifica. É claro que o número de “caixinhas” pode ser bem grande, podendo, por exemplo, armazenar 256 Mbytes.

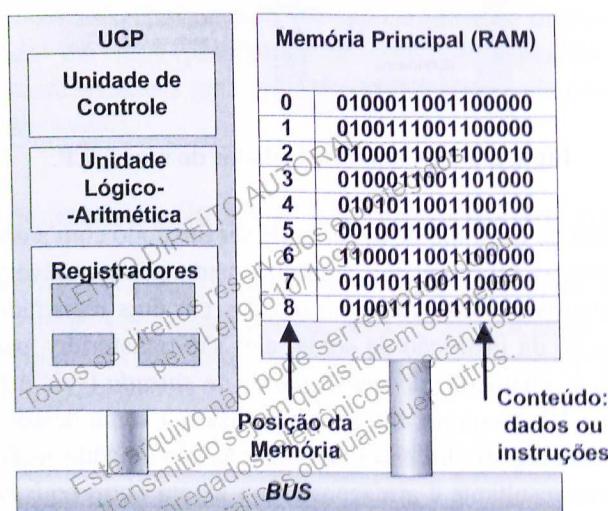


Figura 2.32 Memória principal do computador.

O funcionamento inicia-se da seguinte forma: a instrução armazenada no endereço apontado pelo registrador CP é buscada e armazenada em outro registrador especial, chamado *registrador de instrução* ou RI. Essa unidade de bits transferida da memória

para o registrador RI é nomeada como *palavra* de máquina. A largura da palavra de máquina depende da UCP, sendo típicos os valores de 8, 16, 32 e 64 bits.

Após a transferência da instrução para o registrador RI, o contador de programa (CP) é incrementado para apontar para a próxima instrução. A instrução presente no registrador RI é então decodificada para se saber qual o tipo de operação que representa. Se a instrução necessitar acessar um valor da memória, o endereço desse dado é armazenado em um registrador. Com o endereço armazenado em RI, o dado é buscado e armazenado em um registrador, que, por fim, é utilizado na execução da instrução propriamente dita.

A execução de um programa é, portanto, aquela repetitiva do parágrafo anterior até que uma instrução especial que indica o seu fim seja obtida. Esse funcionamento é resumido pelo Algoritmo 2.1.

Algoritmo 2.1 Algoritmo para o funcionamento da UCP.

Início

CP \leftarrow *endereco_inicial*

Enquanto *tipo_instrucao* \neq PARE **Faça**

RI \leftarrow TransferirMemoria(*CP*)

CP \leftarrow *CP* + 1

tipo_instrucao \leftarrow DecodificarInstrucao(*RI*)

endereco_dado \leftarrow PegarEnderecoDado(*RI*, *tipo_instrucao*)

Se *endereco_dado* \geq 0 **Então**

dado \leftarrow TransferirMemoria(*endereco*)

Fim Se

Executar(*tipo_instrucao*, *dado*)

Fim Enquanto

Fim

LADO DIREITO AUTOR
Todos os direitos reservados e protegidos
pela Lei 9.610/1998.
Este arquivo não pode ser reproduzido ou
transmitido sejam quais forem os meios,
fotográficos ou quaisquer outros.

Falta ainda determinar como as instruções serão representadas. Essa é uma decisão do projetista da UCP. Por exemplo, em uma UCP com palavra de memória de 16 bits, as instruções poderiam ser representadas conforme a Figura 2.33. Nessa figura, o código da instrução é representado pelos bits 12 a 15 e o operando (endereço a ser utilizado pela instrução) pelos bits restantes. Assim, a instrução aritmética de soma poderia ser representada pelo código 0010 e assim por diante.

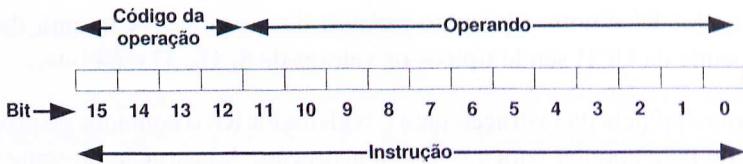


Figura 2.33 Exemplo de organização de instrução.

2.6 O projeto lógico na construção de programas

Por ser um texto introdutório de algoritmos e lógica de programação, a discussão a respeito de detalhes das instruções em nível de máquina termina por aqui. No entanto, deve ser explicado o processo de construção de um programa e reforçar o papel do estudo de algoritmos e lógica de programação nesse processo.

Um programa é para o computador um conjunto de instruções de máquina armazenadas na memória. Porém, normalmente essas instruções são geradas indiretamente, via arquivo texto contendo essas mesmas instruções em código de montagem (*assembly*), que são instruções mnemônicas, como ADD, MOV e outras mais fáceis de se lembrar que simples sequências de zeros e uns.

Mesmo assim, o uso de uma linguagem de montagem não é produtiva no sentido de criar programas em tempo hábil. Seu uso é destinado principalmente à programação de software de sistema (sistema operacional, por exemplo) e para softwares que operarão em tempo real, tais como drivers de vídeo, sistemas de controle industrial e outros.

Grande parte dos programas é na realidade escrita em linguagens de alto nível, que possuem instruções mais compreensíveis ao ser humano, como Pascal, Delphi, C, Java e C++. O processo de construção de um programa com essas linguagens segue as etapas ilustradas na Figura 2.34, que complementam as discussões feitas no Capítulo 1.



Figura 2.34 Etapas no desenvolvimento de um programa.

O processo de construção de um programa é iniciado pelas *ideias* que se tem a respeito do problema a ser resolvido. Depois, em uma etapa de planejamento, é realizado o *projeto lógico* do programa, assunto dos próximos capítulos deste livro. Essa etapa é crucial, pois é aqui que se definirá a lógica do programa em si e se ele servirá ou não como solução a um problema apresentado.

As etapas a seguir dependem da linguagem de programação que será utilizada. Considerando uma linguagem de programação *X*, a ideia nessa etapa é traduzir o projeto lógico para essa linguagem. Isso é feito com o conhecimento da equivalência das instruções definidas no projeto lógico com as instruções reais, disponibilizadas pela linguagem *X*.

Feito isso, o texto contendo as instruções do programa na linguagem *X* é submetido a um programa especial, denominado *compilador*. A tarefa do compilador é traduzir as instruções da linguagem *X* para aquelas de máquina do processador destino. O resultado é um programa executável (conhecido pela extensão .EXE no mundo DOS/Windows), que pode ser então colocado na memória pelo sistema operacional em questão e finalmente executado.

O projeto lógico, ponto de partida nesse processo, representa o programa em seu nível mais alto. Neste, os algoritmos que serão implementados são representados por gráficos, tais como fluxogramas, ou textos, como pseudolinguagem (como o Portugol), e são independentes de uma linguagem de programação. Daí se extraem algumas vantagens:

- Por serem independentes de uma linguagem de programação específica, os fluxogramas e o pseudocódigo podem ser reutilizados para definir programas que poderão ser implementados depois com qualquer linguagem de programação.
- O fluxograma e o pseudocódigo são ferramentas fáceis de aprender e mais fáceis de testar e verificar do que um programa escrito em uma linguagem de programação particular, pois não adicionam detalhes específicos dessas linguagens.
- Possuindo um projeto lógico verificado e testado, tornam-se mínimas as chances de escrever um programa com erros em uma linguagem de programação particular.

Todos os direitos reservados e protegidos
pela Lei 9.610/98.

Este arquivo não pode ser reproduzido
transmitido sejam quais forem os meios
empregados: eletrônicos, mecânicos,
fotográficos ou quaisquer outros.

LEI DO DIREITO AUTORAL

Todos os direitos reservados e protegidos
pela Lei 9.610/1998.

Este arquivo não pode ser reproduzido ou
transmitido sejam quais forem os meios
empregados: eletrônicos, mecânicos,
fotográficos ou quaisquer outros.