António José Estêvão Cabrita

**RESUMOS**

Table of Contents

[1. Introdução aos Computadores 2](#_Toc287301640)

[1.1 Parte 1 - Conceitos básicos das TIC 2](#_Toc287301641)

[1.1.1 Informação 2](#_Toc287301642)

[1.1.2 Números 3](#_Toc287301643)

[1.1.3 Representação binária 4](#_Toc287301644)

[1.1.4 Digital vs. Analógico 6](#_Toc287301645)

[1.1.5 Hardware e software 7](#_Toc287301646)

[1.1.6 . Definições de informação 7](#_Toc287301647)

[1.1.7 Breve História das Tecnologias de Informação e Comunicação 8](#_Toc287301648)

[2 Parte 2 - Estrutura e Funcionamento dos Computadores 11](#_Toc287301649)

[2.1 Organização de um computador 11](#_Toc287301650)

[2.2 Unidade Central de Processamento 12](#_Toc287301651)

[2.3 O “fetch cycle” 12](#_Toc287301652)

[2.4 As instruções do processador 13](#_Toc287301653)

[2.5 A memória 14](#_Toc287301654)

[2.6 Memória secundária 15](#_Toc287301655)

[2.7 Dispositivos de entrada e saída 16](#_Toc287301656)

[2.8 Conflitos de hardware 16](#_Toc287301657)

[3 - Sistemas Operativos 18](#_Toc287301658)

[3.1 A “shell” 19](#_Toc287301659)

[3.2 Tipos de SO 19](#_Toc287301660)

[3.3 Ficheiros 21](#_Toc287301661)

[3.4 Processos 23](#_Toc287301662)

[3.5 Gestão de memória 24](#_Toc287301663)

[3.6 Gestão de dispositivos 26](#_Toc287301664)

[3.7 “Boot strapping” 26](#_Toc287301665)

# Introdução aos Computadores

## Parte 1 - Conceitos básicos das TIC

### Informação

Os tipos de informação com que interagimos mais frequentemente nos sistemas informáticos são os seguintes:

• **Texto**: uma sequência de caracteres (símbolos) que codificam um sistema de escrita de uma língua humana.

• **Imagem**: um padrão bidimensional de luz e cor, que reflecte o que a nossa visão capta. Esse padrão pode ser estático, quando não se altera com o tempo, ou dinâmico, quando depende do passar do tempo – neste caso designa-se vídeo.

• **Som**: um padrão de vibração do ar que reflecte o que é captado pelo nosso sentido da audição.

Destes três tipos de informação, o que nos é menos imediato, o texto (temos que aprender a lê-lo), é também o mais fácil de representar num computador. Basta atribuir um código numérico a cada letra e a mais alguns símbolos adicionais (espaço em branco, pontuação, etc.) e representar o texto através da sequência de códigos correspondente à sequência de caracteres do texto. Por exemplo, usando o código ASCII (American Standard Code for Information Interchange).

Uma imagem é um padrão bidimensional de **luz** e **cor**, logo, a primeira dificuldade é a representação da luz e da cor. A luz é representada por um número que identifica a intensidade luminosa. O tom de cor pode ser representado pelas percentagens relativas de três cores básicas (vermelho, verde e azul nos monitores; cyan, magenta e amarelo nas impressoras). A disposição espacial da luz e da cor é representada por um sistema de coordenadas cartesianas, ou seja, cada ponto da imagem é localizado espacialmente por duas coordenadas numéricas (x e y), que representam a distância horizontal e vertical a um dos cantos da imagem, tomado como referência. Para representar o vídeo, há que adicionar uma terceira coordenada ao sistema: o tempo.

O tempo também é um parâmetro importante na representação do som, além dos parâmetros típicos das ondas sonoras, como amplitudes, frequências e comprimentos de onda.

Do exposto, concluímos que tudo o que precisamos para representar uma grande variedade de informação são números. As tecnologias de informação e comunicação limitam-se a processar e transmitir gigantescas quantidades de números.

### Números

Sabemos da matemática que há várias classes de números: naturais, inteiros, racionais, reais. Qual ou quais as que nos convêm para representar os tipos de informação que discutimos na secção anterior?

A quantidade de números reais também é infinita. O conceito de infinito é uma noção estranha: será que por serem também infinitos, existem tantos números reais como inteiros? Se assim fosse, poderíamos esperar que a representação dos números reais fosse semelhante à dos números inteiros. Mas, na realidade, não é isto que acontece. O matemático Georg Cantor demonstrou, em 1874, que existem muito mais números reais do que números inteiros (poderíamos dizer infinitamente mais…). Intuitivamente, este facto reflecte-se nos dois tipos de infinitos presentes nos números reais: o infinitamente grande e o infinitamente pequeno. Para efeitos de representação, o infinitamente grande é limitado pela **amplitude** da representação, ou seja, o maior número real representável. O infinitamente pequeno traduz-se na **precisão** da representação, ou seja, o número de casas decimais.

Os números reais podem ser representados no computador de duas formas distintas: a representação em **vírgula fixa** e a representação em **vírgula flutuante**. A representação em vírgula fixa é a mais natural: a parte inteira do número é representada por um grupo de dígitos de dimensão fixa (tal como nos números inteiros) e a parte decimal também. A figura 1 ilustra a representação em vírgula fixa.



**Figura 1: Representação em vírgula fixa**

No entanto, a representação quase universalmente usada é a representação em vírgula flutuante. Nesta representação consideram-se apenas os dígitos significativos (eliminam-se os zeros à esquerda e à direita), e guarda-se a posição da vírgula decimal. A sequência de dígitos significativos chama-se **mantissa** e a posição da vírgula é o **expoente**. Por exemplo, na representação do número 56000, a mantissa é 56 e o expoente é 3 (três casas à direita da mantissa). No caso de um número com casas decimais, por exemplo, 0,0037, a mantissa é 37 e o expoente é -4 (quatro casas à esquerda da mantissa). Outro exemplo: 7,5 – a mantissa é 75, o expoente é -1. A figura 2 ilustra esquematicamente a representação em vírgula flutuante.



**Figura 2: Representação em vírgula flutuante**

A representação em vírgula flutuante tem a vantagem de não desperdiçar dígitos que representam zero e, dessa forma, aumenta a gama de valores que podem ser representados com os mesmos dígitos. A amplitude da representação em vírgula flutuante corresponde ao número de dígitos usados no expoente e a precisão corresponde ao número de dígitos da mantissa.

O grande inconveniente dos números em vírgula flutuante é a complexidade das operações e a imprecisão dos resultados. Por exemplo, um resultado que devia ser zero pode ser, após uma operação com vírgula flutuante, algo como 0,00000193. Pequenos erros na precisão dos números podem levar a resultados inesperados. Foi o que aconteceu em 1994, quando o fabricante Intel lançou o seu novo microprocessador Pentium, que tinha um erro (bug) nas operações de vírgula flutuante.

### Representação binária

Do exposto nas secções anteriores, concluímos que os números inteiros e reais são a base da representação da informação nos sistemas informáticos. Por sua vez, estes números são representados, com certas limitações, por sequências de dígitos. Cada dígito é um de vários símbolos que representam valores numéricos pequenos: no sistema decimal há dez dígitos. No entanto, os computadores electrónicos utilizam apenas dois dígitos (representação binária). O número de dígitos de um sistema de numeração designa-se por **base**: o sistema decimal é de base 10 e o sistema binário é de base 2.

Pensemos no processo de contagem na base 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. A seguir ao 9 vem 10, isto é, voltamos a usar o dígito 0 mas juntamos-lhe um 1 à esquerda. Quando chegamos a 19, voltamos a 0 mas temos de incrementar o dígito da esquerda: 20. E assim sucessivamente. Quando chegamos a 99, temos de juntar mais um dígito à esquerda: 100. Etc, etc… Na representação binária é a mesma coisa mas só temos dois dígitos. Vejamos como se processa a contagem: 0, 1. Como não há mais dígitos, temos de voltar ao 0, juntando-lhe um 1 à esquerda: 10. Não se deve confundir este número binário que representa dois com o número decimal dez (representado por 10 no sistema decimal). Continuando, temos 11 e depois 100, já que se esgotaram os dígitos em ambas as posições. Depois vem 101, 110, 111, 1000 e assim sucessivamente. A tabela 1 mostra a numeração binária e decimal correspondente para os 10 primeiros números naturais.

**Tabela 1: Numeração binária e decimal**



Cada dígito de uma representação binária é denominado **bit** (abreviatura de binary digit – dígito binário). Se dispusermos de N bits, é possível representar 2x2x…x2 = 2N números diferentes. Um grupo de 8 bits é designado por **byte**. No entanto há uma diferença em relação ao sistema decimal: enquanto que um quilograma são 1000 gramas, um quilobyte são 1024 bytes. A razão desta diferença é que se torna muito mais conveniente usar uma potência de conveniente usar potências de 10 (1000 = 103) no sistema decimal.

As potências de 2 têm um papel essencial na determinação do valor de um número binário. Enquanto que no sistema decimal temos a casa das unidades, das dezenas, das centenas, dos milhares, etc., no sistema binário temos a casa das unidades, dos 2, dos 4, dos 8, dos 16, etc. Estes números são as sucessivas potências de 2 (1 = 20, 2 = 21, 4 = 22, 8 = 23, 16 = 24) e correspondem ao “peso” de cada dígito binário. Suponhamos que queremos determinar quanto vale o número binário 10011011. Em primeiro lugar, atribuamos a cada dígito o peso correspondente:



Agora, tal como no sistema decimal, basta multiplicar o valor de cada dígito pelo respectivo peso e somar tudo. É claro que no sistema decimal, já está tudo feito, pelo que sabemos intuitivamente quanto vale, por exemplo, 1799 (=1x1000 + 7x100 + 9x90 + 9x1), mas no sistema binário, temos de fazer as contas. A tarefa é, no entanto, simplificada, já que basta somar os pesos dos dígitos 1 (multiplicar um dígito 0 pelo seu peso dá 0). Assim, para o exemplo acima, o resultado é:

**128 + 16 + 8 + 2 + 1 = 155**

O número binário 10011011 vale, então, 155. Como passámos de uma representação em base 2 para uma representação em base 10 (que é mais natural para nós, seres humanos), este procedimento chama-se **conversão de um número binário para decimal**.

A operação inversa, a **conversão de um número decimal para binário** é ligeiramente mais complicada: exige uma série de divisões sucessivas por 2. Vejamos, por exemplo, como se converte o número 232 em binário. Temos de dividir o número por 2 e olhar para o resto da divisão.

O resto é 0, pelo que o dígito binário das unidades é 0. Continuemos a dividir o quociente que obtivemos.

Agora obtivemos 1, pelo que o dígito binário seguinte (o de peso 2) é 1. Continuando as divisões sucessivas por 2.

O processo termina quando chegamos a um quociente 0. O número binário que pretendemos é, finalmente, 11101010, que é a sequência, por ordem inversa, dos restos obtidos.

Como vimos, em última análise, toda a informação é representada num computador por uma sequência enorme de zeros e uns. Para além das bases 2 e 10, são importantes outras bases de numeração como 8 e 16, pela sua facilidade de conversão de e para a base 2. De facto, para converter um número na base 2 para uma base 2N basta agrupar cada N dígitos do número binário e substituí-los pelo correspondente dígito em base 2N. Por serem muito usadas, estas bases recebem nomes especiais – a base 8 é **octal** e a base 16 é **hexadecimal**.

###  Digital vs. Analógico

A informação representada desta forma diz-se **digital**, já que é constituída por uma série de dígitos binários. Os números reais têm uma característica, no entanto, que os distingue dos números inteiros: é impossível representar digitalmente, com exactidão, a maioria dos números reais – pensemos, por exemplo, no número π, com a sua sequência infinita de casas decimais que nunca se repete. Este facto tem a ver com a natureza infinita dos números reais, tal como foi descoberta por Cantor: amplitude infinita e precisão infinita.

As grandezas físicas medem-se recorrendo a números reais: os comprimentos são números reais de metros, os tempos são números reais de segundos. A informação veiculada por números inteiros é **discreta**, podendo ser representada de forma digital, enquanto que a informação veiculada por números reais é **contínua**, podendo ser representada apenas aproximadamente em forma digital. Um dispositivo que representa um número real de forma contínua diz-se **analógico**, por exemplo, um termómetro que indica a temperatura através do comprimento de uma coluna de mercúrio. Hoje em dia, também dispomos de termómetros digitais, que indicam o valor numérico da temperatura num visor, com uma precisão de uma ou duas casas decimais. As tecnologias que convertem informação analógica em digital e vice-versa são uma parte fulcral das modernas tecnologias de informação: a imagem num ecrã (analógica) resultou da conversão da informação digital guardada na memória da placa gráfica do computador; a placa de som converte o som captado analogicamente por um microfone em informação digital. Ao processo de conversão de informação analógica em informação digital chama-se **digitalização** ou **discretização**. Além da gravação digital de som, o processo de leitura de uma imagem num scanner também é um exemplo muito conhecido de digitalização.

###  Hardware e software

O hardware são os equipamentos físicos e o software são os programas que correm sobre o hardware. À luz dos conceitos introduzidos nas últimas secções, podemos ter uma perspectiva complementar: o software é, de facto, a informação digital manipulada por (e que manipula) os dispositivos físicos.

A questão essencial não é o hardware e o software serem conceitos distintos, mas sim o facto de serem conceptualmente independentes. Em princípio:

a) um pedaço de software (programa) com determinada funcionalidade pode correr em qualquer computador.

b) um computador pode correr qualquer tipo de programa.

O surgimento dos modernos computadores digitais permitiu cumprir o requisito b), já que são máquinas de carácter genérico com capacidade para executar qualquer processo automático (algoritmo) sob a forma de um programa.

As linguagens de programação, desenvolvidas desde os anos 50, respondem ao outro requisito, permitindo codificar um algoritmo de forma independente da máquina onde vai ser executado. Na prática, porém, existem dificuldades relacionadas com as especificidades de cada máquina, a que o programador não pode ser alheio. Estas especificidades fazem com que executar um mesmo programa em máquinas diferentes não seja trivial. Nos anos 90, com a expansão da Internet, foi desenvolvida a linguagem Java que permite a execução de um programa em várias máquinas, interpondo uma “máquina virtual” entre o software e o hardware propriamente dito. A máquina virtual Java não deixa de ser também um pedaço de software mas “esconde” os detalhes do hardware ao programador.

### . Definições de informação

Consideremos uma imagem em branco. A informação contida na imagem é quase nula. Do ponto de vista da representação em computador, no entanto, esta imagem irá ocupar o mesmo espaço que uma paisagem, por exemplo, pois ambas são representadas por uma matriz bidimensional de pontos coloridos. Temos a sensação que a imagem em branco está a desperdiçar espaço, já que não contém qualquer informação. É nesta ideia que se baseiam as definições de informação propostas por **Claude Shannon** em 1948 e por **Chaitin e Kolmogorov** nos anos 60. A teoria da informação de **Shannon** permite substituir qualquer sequência de dígitos por outra, mais reduzida, que contém exactamente a mesma informação: por exemplo, a imagem em branco pode ser substituída pela informação de um único ponto branco e pelo número de vezes que esse ponto é repetido na imagem. Os vulgares programas de compactação (WinZip, WinRar, Arj, etc.) baseiam-se na teoria de **Shannon** para reduzir o tamanho dos ficheiros, sem qualquer perda de informação.

A teoria da informação algorítmica, desenvolvida por **Chaitin** e, independentemente, por **Kolmogorov**, vai mais longe ao dizer que a informação contida numa sequência de dígitos é igual ao tamanho do menor programa de computador que consegue gerar essa sequência. O problema com esta definição é que não é possível construir automaticamente esse programa, tornando impraticável a construção de compactadores com base na teoria de **Chaitin**. No entanto, a definição de **Chaitin** é a mais interessante do ponto de vista matemático, já que propõe uma medida absoluta de informação, enquanto que a definição de **Shannon** mede apenas a informação média.

###  Breve História das Tecnologias de Informação e Comunicação

O computador digital, tal como o conhecemos, é uma invenção do século XX. No entanto, houve anteriormente projectos de dispositivos mecânicos para efectuar cálculos e computações. Notavelmente, **Schickard**, em 1623, **Pascal**, em 1642 e **Leibniz**, em 1671 construíram máquinas de calcular que tiveram algum sucesso.

Em 1834, **Charles Babbage**, professor de matemática em Cambridge, projectou uma máquina chamada “Analytical Engine”, onde introduziu as noções de unidade central de processamento e de memória. A máquina era programável mas, pela sua complexidade, **Babbage** não conseguiu obter financiamento para a sua construção.

Durante o século XX, o computador electrónico surgiu como resultado da investigação em matemática e dos desenvolvimentos tecnológicos no campo da electrónica. No final do século XIX e no início do século XX, colocou-se a questão da formalização da matemática. Esta “máquina da verdade” acabou por se revelar uma utopia, ao nível da “máquina do movimento perpétuo” que muitos tinham tentado descobrir em séculos anteriores. Foi o matemático Kurt **Gödel** que frustrou as expectativas dos outros investigadores, ao demonstrar, em 1931, que é impossível formalizar qualquer teoria que envolva conjuntos infinitos: nem sequer a aritmética de números inteiros pode ser formalizada de forma completa. Este resultado deitou o sonho da “máquina da verdade” por terra, mas ironicamente, abriu o caminho para as bases teóricas do computador.

O matemático **Alan Turing** reformulou o teorema de **Gödel** em termos de um dispositivo teórico a que chamou “**máquina de Turing**”. Inicialmente, o dispositivo continha uma fita infinita com zeros e uns, e um programa, que dirigia as operações da máquina. Cada máquina tinha um programa diferente. A ideia revolucionária de **Turing** foi codificar o programa sob a forma de zeros e uns na fita. Desta forma, a máquina lia o programa na fita, interpretava-o e depois seguia as respectivas instruções. Deixou de haver necessidade de uma máquina específica para cada programa: uma única máquina executava qualquer programa que lhe fosse fornecido na fita. A esta máquina chamou-se **máquina universal de Turing** e constitui o modelo teórico dos computadores que usamos actualmente. **Turing** traduziu o resultado de **Gödel** em linguagem computacional da seguinte forma: não existe nenhum processo automático (leia-se programa) capaz de determinar se um programa da máquina universal de Turing termina ou não a sua execução.

As ideias de **Turing** foram desenvolvidas por **John von Neumann** que propôs a arquitectura em que os programas e os dados são guardados na mesma memória do computador. Os primeiros computadores baseados neste princípio foram construídos nos anos 40, usando a tecnologia das válvulas electromecânicas. A invenção do **transístor**, em 1947, proporcionou uma revolução na tecnologia dos computadores, dando origem à chamada segunda geração de computadores. As válvulas foram substituídas por transístores, componentes mais pequenos e mais fiáveis. A invenção que deu origem à geração seguinte (3ª) de computadores foi o **circuito integrado**: um componente que incorpora um circuito electrónico numa pequena pastilha de silício. Os primeiros computadores à base de circuitos integrados foram construídos nos anos 60, dando origem à proliferação de computadores verificada hoje em dia. Os anos 70 viram o nascimento do microprocessador, lançado pela companhia Intel e, em 1974 surge o primeiro computador de uso doméstico. Em 1981 a IBM lança o IBM PC, do qual derivam quase todos os computadores que temos hoje em cima das nossas secretárias.

Durante os anos 80 e 90, as tecnologias que mais contribuíram para a expansão das TIC foram os sistemas operativos de janelas e a Internet. Os sistemas operativos de janelas tornaram mais acessível e imediato o uso do computador: a Apple foi pioneira nesta área, introduzindo o sistema MacOs em 1984, que acompanhava o seu novo computador Macintosh. No ano seguinte, a Microsoft lançou a primeira versão do Windows, que evoluiu para o sistema operativo que é hoje usado na grande maioria dos computadores pessoais tipo PC.

A Internet foi montada em 1969 (na altura chamava-se ARPANET) e inicialmente ligava os computadores do departamento de defesa norte-americano. No final dos anos 70, a ARPANET foi libertada para uso civil e os primeiros utilizadores foram as grandes universidades. Só em meados dos anos 80 é que começaram a surgir os primeiros serviços de Internet que agora usamos: os nomes de domínio (e.g. microsoft.com ou univ-ab.pt) e os protocolos de e-mail e transferência de ficheiros.

A World Wide Web (WWW) é o serviço mais familiar da Internet e o responsável pela explosão da Internet nos anos 90. A WWW foi inventada em 1989 por **Tim Berners-Lee**, um físico do CERN (Centro Europeu de Pesquisa Nuclear), como forma de partilhar documentos hiper-ligados entre a comunidade de investigação em física.

Obviamente que estes desenvolvimentos de software não seriam possíveis sem o correspondente avanço do hardware: embora a ideia básica se mantenha desde os anos 40, os computadores tornaram-se muito mais rápidos, com muito mais capacidade de armazenamento de dados e programas, e mais compactos.

# Parte 2 - Estrutura e Funcionamento dos Computadores

## Organização de um computador

Os computadores são constituídos por três grandes subsistemas: a **unidade central de processamento** (CPU – central processing unit), a **memória** e os **dispositivos de entrada e saída** (I/O – input/output).

Para funcionar, o computador precisa de um motor que faça progredir o processamento da informação, de espaço para armazenamento dessa informação e de mecanismos para receber e transmitir informação de/para o exterior. Cada uma destas funções corresponde a um subsistema.. A figura 1 mostra os três sub-sistemas e os canais de transferência de informação entre eles. Cada canal de transferência de informação é genericamente designado por ***bus*** (barramento, nas traduções mais comuns para português).



**Figura 1 - Arquitectura do computador**

O CPU está encarregue da gestão da memória e do subsistema de I/O. No entanto, por vezes há interesse em que o subsistema de I/O aceda à memória sem passar pelo CPU. Por exemplo, o carregamento de informação de um CD para memória pode ser feito directamente. Para isso, criaram-se canais privilegiados de acesso à memória, chamados **DMA** (direct memory access), tendo em conta que a grande maioria das operações de um computador se resume a transferir informação de um lado para o outro. O DMA permite libertar o CPU para realizar tarefas menos “monótonas”, e melhorar o desempenho geral do sistema.

No caso dos computadores pessoais que usamos quotidianamente, o CPU e a memória encontram-se na chamada “**motherboard**”, uma placa de circuitos integrados que constitui o elemento mais central do computador. Geralmente, as “motherboards” actuais já permitem a substituição do CPU e várias configurações de memória através da inserção de placas mais pequenas em suportes apropriados. Estes suportes estão ligados a um *bus* de acesso ao CPU. As placas de expansão mais comuns são a **placa gráfica** e a **placa de som**.

## Unidade Central de Processamento

A unidade central de processamento (CPU) ou processador é o motor de um computador. Há quem diga, mais poeticamente, que é o “coração” ou o “cérebro” do computador.

O processador é essencialmente uma máquina de estados que executa instruções a um determinado ritmo. O estado do processador em cada momento é determinado por um conjunto de registos, que mais não são do que pequenos espaços de armazenamento de dados (à semelhança da memória, mas internos ao processador). Em cada passo, o processador lê a próxima instrução e executa-a, modifica os seus registos e passa para o estado seguinte.

O ritmo a que o processador executa instruções é determinado pela sua **velocidade**, que se mede em ciclos por segundo. Um ciclo por segundo é uma unidade a que se chama **hertz** (símbolo Hz). Quando se adquire um computador novo, uma das características que os vendedores gostam de apontar é a velocidade do processador, que actualmente andam na ordem dos poucos gigahertz (1 GHz = 1000 milhões de hertz). A razão por que se mede a velocidade em ciclos por segundo e não em instruções por segundo é que as instruções de um processador têm durações de execução diferentes, consoante a sua complexidade, enquanto que um ciclo tem sempre uma duração fixa. A velocidade do processador é imposta exteriormente por um cristal que emite impulsos eléctricos a um ritmo constante – o mecanismo de geração desses impulsos chama-se o **relógio** (clock). Às vezes é possível aumentar a velocidade do processador aumentando o ritmo do relógio (processo chamado *overclocking*.

## O “fetch cycle”

O processador tem um modo de funcionamento muito simples e que pode ser ilustrado pela seguinte “receita”:

Passo 1: ler a próxima instrução

Passo 2: descodificar a instrução lida

Passo 3: executar a instrução

Passo 4: voltar ao passo 1

Este procedimento chama-se “fetch cycle” e é tudo o que o processador faz. Parece incrivelmente simples, especialmente se tivermos em conta que apenas existem duas ou três centenas de instruções diferentes.

Como é que surge então toda a diversidade de funções que um computador exibe? O segredo está na sequência específica de instruções que é fornecida ao computador, sequência essa a que chamamos **programa**. Este aspecto é análogo à diversidade de mensagens que se conseguem obter com apenas 26 letras.

## As instruções do processador

Um **programa** de computador é constituído por uma sequência de instruções, cuidadosamente preparada por um programador. Um programa é visto por um utilizador como uma caixa fechada onde entram dados por uma extremidade e saem os resultados pretendidos pela outra extremidade, como se ilustra na figura 2:



**Figura 2 - O programa**

Com esta perspectiva, não é difícil compreender o tipo de instruções que um processador executa:

- **instruções de transferência de informação**: servem para copiar e mover dados de um lado para o outro: da memória para os registos e vice-versa; dos registos para o subsistema de I/O e vice-versa.

- **instruções de transformação de informação**: são de quatro tipos:

* **Operações aritméticas**: efectuam cálculos numéricos sobre os dados
* **Operações lógicas**: aplicam operadores da lógica aos dados (e, ou, negação lógica)
* **Deslocação e rotação**: alteram a sequência dos bits de um registo ou posição de memória.
* **Comparações**: efectuam comparações (menor, maior, igualdade) sobre os dados

- **instruções condicionais**: conferem capacidade de decisão aos programas permitindo executar uma instrução A se uma determinada condição for verdadeira ou uma instrução B caso contrário. As condições são normalmente baseadas no resultado de uma instrução de comparação.

- **instruções de salto e chamada**: permitem alterar a ordem de execução das instruções saltando para outro ponto do programa. São muitas vezes associadas às instruções condicionais, quando é preciso executar um grupo de várias instruções sob determinada condição.

- **outras instruções**: entrar em modos especiais de funcionamento do processador, parar o processador, não fazer nada durante algum tempo, etc.

A seguinte sequência exemplifica um programa (traduzido para português) que pode ser executado num CPU:

* *copiar* conteúdo da posição 2412 da memória para o registo A
* *somar* 6 ao registo A
* *comparar* o conteúdo do registo A com o número 10
* *se* for menor, *saltar* para a posição <reprovado> do programa
* *transferir* a frase “aprovado” para o ecrã
* - <reprovado>: transferir a frase “reprovado” para o écrã

Os dados entram na posição2412 da memória e os resultados saem pelo écrã.

Deixa-se como exercício para o leitor a categorização de cada uma das instruções que aparecem neste programa.

## A memória

A utilização que fizemos acima do termo “posição” para indicar o local onde se situam tanto dados como instruções do programa não é acidental, mas reflecte o duplo papel da memória num sistema informático: o de armazenar dados e o de armazenar programas.

No início da história dos computadores, os dados e os programas eram guardados em memórias diferentes; havia a memória de dados e a memória de programas. A introdução da memória comum para guardar dados e programas constituiu uma mudança muito importante na forma de organizar a informação no computador. A esta nova organização chama-se arquitectura de **Von Neumann**, e todos os computadores actuais se baseiam nela.

A memória de um computador não é mais do que uma sequência de “compartimentos” identificados pela sua “posição” e que contêm informação (ver figura 3). Cada compartimento contém uma quantidade fixa de bits (8, 16, 32 ou 64, conforme o computador). Independentemente do tamanho do compartimento, a capacidade da memória mede-se em bytes (isto é, grupos de 8 bits), sendo mais comum actualmente o uso dos seus múltiplos (kilobytes - Kb, megabytes - Mb, gigabytes - Gb).



**Figura 3 - A memória**

A posição de um compartimento é designada tecnicamente por **endereço**. A comunicação entre o processador e a memória é sempre feita nos seguintes termos: “dá-me os dados que estão no endereço x” ou “coloca os dados y no endereço z”. A informação que circula entre o processador e a memória é, portanto, de três tipos:

* endereços
* dados
* informação de controlo

A informação de controlo corresponde ao tipo de operação que o processador quer executar na memória: ler dados ou escrever dados. Os três tipos de informação são transmitidos em simultâneo, pelo que o bus que liga o processador à memória está dividido em três partes: o **bus de endereços**, o **bus de dados** e o **bus de controlo**.

* $bus endereços \left(bits\right)=Log\_{2}2^{y} = y Log\_{2}2=y x 1=y$
* $2^{y}(bits)=\frac{Memória total}{arquitectura}$
* **Bus dados = “Arquitectura” = 8, 16, 32 ou 64 bits**

Adiante veremos que o bus de controlo pode ter funções adicionais.

Exemplo: Quantos bits têm de ter o bus de endereço mais o bus de dados que ligam o processador a uma memória de 32 bits, se a memória tiver uma capacidade de 1 quilobyte?

1kb = 1024x8 = 2^10 x 2^3 =2^13 bits

32 bits = 2^5

Bus dados = arquitectura = 32 bits = 2^5

Memória total / arquitectura = 2^13 / 2^5 = 2^8

Bus endereços + bus dados = 8 + 32 = 40 bits

## Memória secundária

A memória de que falámos até agora designa-se por memória principal e tem a vantagem de ser acessível com grande rapidez pelo processador. Infelizmente, como é constituída por circuitos electrónicos, só cumpre a sua função (a de memorizar os dados) enquanto for alimentada por corrente eléctrica.

Para colmatar este inconveniente, existem dispositivos de **memória secundária** (o disco, as disquetes, os CDs, os DVDs) que armazenam dados de forma mais duradoura. Infelizmente, também têm um aspecto menos bom: o acesso processa-se de uma forma muito mais lenta. O computador normalmente usa a memória principal e só ocasionalmente transfere dados de/para a memória secundária.

Dos vários dispositivos de memória secundária, o **disco duro** é o mais importante. Tem uma capacidade muito grande (medida em Gb) e uma velocidade de acesso mais rápida que os outros dispositivos de memória secundária.

As disquetes podem ser transportadas de computador para computador. O inconveniente das disquetes é que têm mantido a mesma capacidade (máximo: 1,44Mb). Por essa razão têm sido usados outros dispositivos como as “zip drives” e as “pen disk” que cumprem a mesma função mas têm capacidades muito maiores. Até os CDs e DVDs (graváveis e regraváveis) são utilizados para transferir informação entre computadores, apesar da maior lentidão de acesso. Os CDs e DVDs são preferíveis quando se pretende guardar informação de forma permanente (por exemplo, um trabalho acabado) uma vez que têm uma durabilidade bastante maior que as disquetes e os outros dispositivos e não são influenciáveis por campos magnéticos.

## Dispositivos de entrada e saída

Os dispositivos de entrada e saída são controlados pelo processador da mesma forma que a memória, ou seja, via um bus de endereços, um bus de dados e um bus de controlo. Na realidade, são os mesmos buses de acesso à memória, e existe uma linha do bus de controlo que indica se o processador está a aceder à memória ou a um dispositivo de I/O.

Cada dispositivo está associado a um endereço onde o computador lê ou escreve informação.

Existe, no entanto, uma diferença fundamental entre a memória e os dispositivos de entrada/saída. Enquanto que a memória é passiva, ou seja, não toma nunca a iniciativa de comunicar com o processador, os dispositivos de entrada e saída podem ser activos, iniciando processos de comunicação com o processador. A forma que os dispositivos têm para fazer esta indicação é através de um pedido de interrupção. Um **pedido de interrupção** (*interrupt request*) é efectuado por um canal especial que liga um dispositivo de entrada/saída ao processador, a que se chama linha de **IRQ** (interrupt request). Quando o processador recebe um pedido de interrupção, pára temporariamente o que estava a fazer e executa um pequeno programa que atende o pedido.

## Conflitos de hardware

Por vezes, ao instalar um novo componente de hardware no computador, surgem conflitos com outros dispositivos que impedem que o hardware funcione correctamente.

Estes conflitos podem surgir quando há coincidência nas seguintes especificações para dispositivos diferentes:

* pedidos de interrupção (IRQ)
* endereços dos dispositivos de entrada/saída
* canais de DMA
* memória principal usada pelos dispositivos

Por exemplo, se dois dispositivos tiverem a mesma linha de IRQ, poderá surgir um conflito. No entanto, nem sempre isso acontece. Há casos em que dois dispositivos podem perfeitamente partilhar a mesma linha de IRQ sem haver problemas. Em geral, o sistema operativo sabe informar se dois dispositivos estão em conflito ou não.

Se houver conflito, a forma de o resolver é atribuir outra linha de IRQ ao dispositivo em questão (ou endereço de I/O, ou canal de DMA ou seja o que for que está a causar o conflito). A resolução nem sempre é pacífica, já que em alguns casos, os dispositivos têm parâmetros definidos de fábrica que impedem que sejam alterados.

# - Sistemas Operativos

O **sistema operativo** (SO) é a primeira camada de software de um sistema informático. Todo o restante **software de aplicação** está dependente do SO.

Os SO mais antigos não obrigavam as aplicações a dependerem do SO, e muitos programas manipulavam directamente o hardware do computador. Este tipo de comportamento não é aconselhado, principalmente por duas razões. Primeiro, porque o hardware varia muito de máquina para máquina e não é viável escrever n versões de um mesmo programa para acomodar diversos tipos de hardware. Por outro lado, a manipulação directa dos recursos de hardware pode levar a que programas construídos com fins maliciosos possam afectar directamente esses recursos.

Os sistemas operativos cumprem de forma eficaz as suas duas funções essenciais, que são as seguintes:

1. **Disponibilizar uma máquina virtual, que é uma extensão da máquina física.**
2. **Gerir os recursos do sistema.**

Além de cumprirem estas funções, os actuais SO não permitem que um programa aceda directamente ao hardware nem que faça uma gestão própria dos recursos.

Nas funções do SO, a disponibilização de uma máquina virtual facilita a tarefa do programador, na medida em que este não tem de se preocupar com os detalhes de funcionamento de cada pedaço de hardware, nem com a gestão da memória ou do disco.

## A “shell”

Nos SO actuais a shell é essencialmente uma interface gráfica, em que os objectos são manipulados através de uma representação pictórica (ícones) no ecrã. No entanto, a interface com o utilizador não tem de ser gráfica

A shell é uma componente do sistema operativo que é independente das restantes componentes, no sentido em que é possível substituí-la por outra, mais conveniente.

## Tipos de SO

Existem muitos tipos de SO e estamos mais habituados aos sistemas operativos utilizados nos computadores pessoais, embora existam sistemas operativos específicos para computadores de bolso (PDAs), telemóveis, redes de computadores, dispositivos com restrições críticas de tempo (real-time), etc.

Os sistemas operativos são classificados quanto ao número de utilizadores que comportam, em **monoutilizador** e **multiutilizador**. Os sistemas monoutilizador estão preparados para suportar apenas um utilizador e não promovem a separação efectiva dos dados e processos de vários utilizadores.

Os sistemas multiutilizador estão construídos para suportar vários utilizadores e efectivamente separam os ficheiros e programas que cada utilizador executa. Garantem a protecção dos dados de cada utilizador, impedindo que outros utilizadores consultem e/ou alterem os respectivos ficheiros, e permitem que cada utilizador execute apenas os programas para os quais possui autorização.

Nos sistemas multiutilizador, existe um utilizador com mais privilégios do que os outros, o chamado **super-utilizador**, que tem autorização para executar programas que configuram e alteram o próprio sistema operativo, que gerem os recursos do sistema, e que definem as características dos outros utilizadores.

Independentemente do número de utilizadores, um SO é classificado quanto ao número de programas que pode executar ao mesmo tempo, em **monoprocessamento** e **multiprocessamento**. Um SO multiprocessamento pode ter em execução vários programas simultaneamente. No entanto, um CPU funciona sequencialmente, isto é, uma instrução de cada vez. Como é que um SO em multiprocessamento pode executar vários programas ao mesmo tempo? Há basicamente duas formas de o fazer: ou existem vários CPUs e os programas são distribuídos pelos CPUs ou só há um e terá de haver uma partilha do tempo que o CPU dedica a cada programa. Esta segunda solução designa-se por “**time-sharing**”.

## Ficheiros

O subsistema que gere os ficheiros chama-se **sistema de ficheiros** (filesystem). Tal como a shell, os actuais SOs disponibilizam vários sistemas de ficheiros, embora só seja possível escolher um para cada disco no processo de instalação do SO. Exemplos de sistemas de ficheiros são o NTFS (New Technology File System), o FAT32, ext2fs, o HFS (Macintosh Hierarchical File System) ou o nfs (network file system). Muitas vezes um mesmo sistema de ficheiros é usado em sistemas operativos diferentes.

Os ficheiros em disco são normalmente arrumados em **directorias** ou **pastas**, que se organizam numa estrutura em árvore. Uma cadeia de pastas é designada por **caminho** (path), e normalmente representada pelos nomes das pastas separadas por / ou \.

Compete ao sistema de ficheiros gerir esta estrutura de pastas e ficheiros. A estrutura é virtual, não correspondendo à forma como está organizado o disco, fisicamente. O disco é, tal como a memória central, uma sequência linear de espaços para guardar bytes, acessíveis através de um endereço numérico.

Esta gestão automática por parte do sistema de ficheiros pode gerar uma situação de alguma desarrumação no disco. A esta condição chama-se **fragmentação** do disco. O desempenho do sistema de ficheiros degrada-se nesta situação, e a solução é utilizar um programa, chamado desfragmentador.

## Processos

Um **processo** contém informação sobre a posição corrente no programa a ser executado, bem como sobre os conteúdos dos registos e posições de memória a serem utilizados pelo programa. Isto é, um processo contém informação sobre o estado de execução do programa.

É possível co-existirem vários processos a correrem o mesmo programa, uma vez que se distinguem pelo respectivo estado de execução. Num sistema multiprocessamento, em que os vários processos têm de partilhar o CPU, o sistema operativo atribui, rotativamente, um período de tempo a cada processo, findo o qual guarda o seu estado de execução e passa ao processo seguinte na fila. Quando chega novamente a vez do primeiro processo, o SO repõe o estado de execução guardado e volta a colocar o processo em funcionamento. Num sistema suficientemente rápido, o método rotativo transmite a ilusão de que todos os processos correm ao mesmo tempo. Esta ilusão é formalizada na noção de **processador virtual**, em que cada processo corre no seu próprio processador, embora este na realidade corresponda a uma fatia de tempo do processador real.

Cada processo utiliza recursos do sistema, como a memória, ficheiros em disco, periféricos, que, num sistema multiprocessamento podem estar a ser utilizados simultaneamente por outros processos.

Os processos também podem comunicar entre si e partilhar alguns recursos. A partilha de recursos pode originar conflitos entre os processos, a que se chamam problemas de **concorrência**. A resolução deste tipo de problemas é da responsabilidade não só do sistema operativo mas também do programador. Um dos mecanismos à disposição do programador para evitar um conflito é o **semáforo**. Quando um programa pretende utilizar um recurso de forma exclusiva, deve assinalar com um semáforo que mais ninguém deve executar operações sobre esse recurso. O sistema operativo disponibiliza e verifica os semáforos, garantindo a **exclusão mútua** de processos na região crítica.

Um outro problema que resulta da **concorrência** é o chamado “**deadlock**”, ou o impasse perante recursos bloqueados. Uma forma de resolver um deadlock é eliminar um dos processos, permitindo ao outro prosseguir. Mais tarde, o processo eliminado será reiniciado. Outra forma, que é utilizada no caso das impressoras, é o “**spooling**”. Neste método, não é permitido a um processo bloquear a impressora, mas a informação a imprimir deve ser enviada a um processo especial (o spooler) que guarda essa informação em disco até a impressora estar disponível. O processo que pretende imprimir pode continuar a sua execução, ficando a impressão a cargo do spooler.

## Gestão de memória

É a velocidade de acesso da memória central que faz com que seja a forma mais eficaz de temporariamente guardar a informação associada a um programa enquanto este está a ser executado.

A componente do sistema operativo que se encarrega de gerir este compromisso entre o uso de memória central e secundária é o **gestor de memória**.

Nos sistemas multiprocessamento, em que vários processos possuem informação para alojar na memória central, é natural que por vezes não haja memória suficiente para todos os processos. O gestor de memória coopera com o gestor de processos no sentido de guardar em memória a informação associada aos processos em execução, copiando temporariamente em disco (memória secundária) a informação associada a processos que não estão em execução. Este processo de cópia é designado por “**swapping**” e o espaço em disco que guarda a informação designa-se por ficheiro de “**swap**”. O swapping liberta espaço na memória para alojar informação para outro processo entretanto posto em execução. Se um sistema tem muitos processos, o desempenho degrada-se quando a memória é pouca, devido às frequentes cópias de informação entre a memória e o disco e vice-versa. Esta degradação traduz-se numa maior lentidão do funcionamento geral do sistema, mas o processo de swapping tem o mérito de manter o sistema em funcionamento, mesmo com pouca memória.

Outro problema com a gestão de memória surge quando um único programa necessita de mais memória do que aquela que está disponível. O sistema operativo, através do gestor de memória, inclui um mecanismo de **memória virtual** que permite que tal programa seja executado. A memória virtual é uma abstracção que faz com que qualquer programa tenha a ilusão de ter a memória suficiente para funcionar. Na realidade, o que o sistema operativo faz é manter parte da informação associada ao programa em memória central e outra parte em disco.

A técnica mais usada de memória virtual é a **paginação**, através da qual toda a informação associada a um programa é dividida em partes relativamente pequenas a que se chamam **páginas**. Existem várias estratégias mas uma minimamente eficiente guarda em memória as páginas que são mais frequentemente usadas enquanto que no disco ficam as que só ocasionalmente são acedidas.

## Gestão de dispositivos

Sendo o sistema operativo a camada de software que faz a ponte entre o hardware e o sofware de aplicação propriamente dito, resta saber como é que são tratados os dispositivos de entrada e saída.

Dada a enorme variedade de dispositivos de entrada e saída que um computador tem de controlar não é nada prático que o sistema operativo instalado num computador tenha em conta todas as possibilidades de periféricos que podem estar ligados.

Este problema é resolvido por recurso a componentes do SO que se designam por **gestores de dispositivos** (device drivers), tipicamente um para cada dispositivo a usar no sistema.

Os gestores de dispositivos têm um papel importante a cumprir na função 1 do SO (disponibilizar uma máquina virtual), porque apresentam ao programador uma funcionalidade uniforme para cada tipo de dispositivo. Por exemplo, um programador não precisa de conhecer os detalhes da impressora que vai ser usada, e só tem de considerar o “dispositivo virtual” fornecido pelo gestor de dispositivos da impressora. Além de facilitar o trabalho do programador, este método tem a grande vantagem de não ser necessário alterar o programa para que funcione com outra impressora: basta modificar o gestor de dispositivos da impressora, fornecido pelo respectivo fabricante.

##  “Boot strapping”

O sistema operativo controla todo o hardware, gere os processos e coordena as actividades do computador. Como é que o próprio SO é lançado inicialmente?

O sistema operativo é uma colecção de programas, muitos dos quais têm de ser lançados quando se liga o computador. Estes programas residem em disco, e têm de ser carregados em memória antes de serem executados, como qualquer outro programa.

A solução existe num tipo de memória de sigla ROM (read-only memory). Esta memória, por oposição à memória de tipo RAM, de que temos falado até agora, não perde a sua informação quando a energia eléctrica é desligada. De facto, a memória ROM é gravada de fábrica e o seu conteúdo é fixo. É esta ROM que contém o primeiro programa a ser executado pelo computador, a que se chama programa de “**bootstrapping**” ou, abreviadamente, programa de “boot”: é um pequeno programa que, após consulta de uma tabela situada no disco rígido, carrega o sistema operativo para a memória RAM. A tabela consultada pelo programa de boot (designada nos PCs por MBR – master boot record) contém informação sobre os sistemas operativos disponíveis e onde se situam. O MBR permite que o utilizador possa escolher entre vários sistemas operativos instalados no disco rígido.

Num computador com um SO Windows e um SO Linux é uma instalação com a designação de “dual boot”.