Instituto Politécnico de Bragança

Escola Superior de Tecnologia e de Gestão

Departamento de Informática e Comunicações

Introdução aos Sistemas Informáticos

Apontamentos das Aulas Teóricas

Autores:	Nuno Gonçalves Rodrigues Paulo Alexandre Vara Alves
Adaptação:	Reis Lima Quarteu
Licenciaturas:	Engenharia Informática Engenharia Mecânica Engenharia Química Gestão e Engenharia Industrial
Docentes:	Reis Lima Quarteu Cândida Silva Lúcia Torrão Pedro Dias
Ano Lectivo:	2004/2005

Índice Geral

1 – Unidades Funcionais de um Computador

Um sistema informático é composto pelo *hardware* e pelo *software*:

- O ***hardware*** limita-se a um conjunto passivo de fios e componentes electrónicos que precisa de ordens (comandos e instruções) para desenvolver qualquer actividade.
- O ***software*** é a parte não física do sistema de computação, isto é, a parte lógica do sistema. Trata-se por isso de um conjunto alterável de instruções que permite aos componentes de Hardware do computador a realização de tarefas bem definidas.

O computador é um equipamento electromecânico que manipula informação representada sob a forma de dígitos binários (0 ou 1), mais conhecidos como *bits*.

- 8 *bits* = 1 *byte*[1];
- 1 *KByte* (*kilobyte*, KB) = 2^{10} = 1.024 *bytes*;
- 1 *MByte* (*megabyte*, MB) = 2^{20} = 1.024 x 1.024 = 1.048.576 *bytes*;
- 1 *GByte* (*gigabyte*, GB) = 2^{30} = 1.024 x 1.024 x 1.024 = 1.073.741.824 *bytes*.

Existe um conjunto de componentes básicos comuns aos diversos modelos de computadores, que são:

- Unidade Central de Processamento;
- Memória;
- Sistemas Periféricos (de entrada e de saída).

Estes componentes básicos, ou unidades funcionais, estão organizados tal como se representa na figura 1.1.

1 Também se pode encontrar a palavra "octeto" para designar um *byte*. Nesta sebenta, será usada a designação *"byte"*.

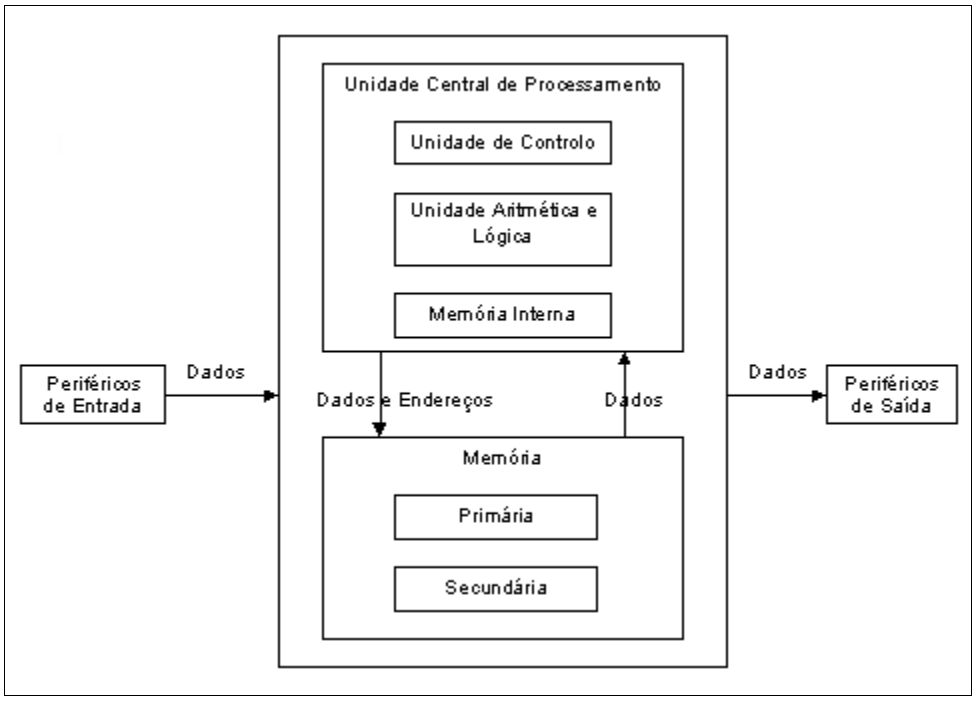


Figura 1.1: *Arquitectura geral de um sistema informático.*

1.1. A Unidade Central de Processamento

A Unidade Central de Processamento (também designada simplesmente como processador) ou UCP (ou CPU, sigla da expressão inglesa *"Central Processing Unit"*), é o componente responsável pela **execução das instruções dos programas** e pelo **controlo do sistema**. Para isso, a UCP é composta por três subcomponentes:

- Unidade Aritmética e Lógica;
- Unidade de Controlo;
- Memória Interna.

1.1.1. Unidade Aritmética e Lógica

A Unidade Aritmética e Lógica (UAL, também vulgarmente conhecida como ALU, sigla da expressão inglesa *"Arithmetic and Logic Unit"*) é constituída por diversos módulos, em que cada um é responsável por um tipo específico de operação do tipo aritmética ou lógica. São exemplos as operações de adição, comparação, multiplicação, divisão, deslocamento[2], etc.

2 A operação de deslocamento consiste em "deslocar" os dígitos de um número para a esquerda ou para a direita, completando com zeros à direita ou à esquerda (conforme a direcção do deslocamento). Por exemplo, a

1.1.2. Unidade de Controlo

A Unidade de Controlo (UC) é responsável por seleccionar as diferentes instruções segundo a ordem de execução, interpretá-las e activar as respectivas operações da máquina para que estas sejam executadas. Esta unidade também é responsável por efectuar um determinado número de controlos sobre as transferências de dados das unidades periféricas para a memória central e vice-versa, bem como sobre as transferências (internas à UCP) entre a memória central e a UAL. Por fim, a UC é ainda responsável por manter o controlo da sequência da execução das instruções do programa.

1.1.3. Memória Interna

A memória interna da UCP permite armazenar os valores dos operandos que são fornecidos à UAL, bem como os resultados das operações. Os seus tempos de acesso são muito baixos (trata-se de um tipo de memória extremamente rápida), mas é um tipo de memória muito cara: é por este motivo que a memória central dos computadores não é constituída por memória desse tipo.

A memória interna da UCP é constituída por um acumulador (uma área da memória interna onde os dados são colocados, ou "acumulados", como se fossem uma pilha de dados) e por vários registos (células de memória de acesso muito rápido, e que se destinam ao armazenamento temporário dos dados manipulados pela UCP).

Possui uma baixa capacidade: a última geração de processadores tem 1 MB de memória interna (ou cache interna).

1.2. A Memória

A memória do computador é o local onde se armazena a informação a processar e a informação processada. Encontra-se dividida em pequenos blocos sequenciais, em que cada um possui um endereço, através do qual se pode aceder a cada um dos blocos, para leitura ou para escrita.

1.2.1. Parâmetros de Classificação das Memórias

Os parâmetros utilizados para classificar as memórias são:

operação de deslocar o número 1101 de um dígito para a esquerda terá como resultado o número 1010.

- **Tempo de Acesso:** Tempo que a UCP demora a aceder à memória. É desejável que seja o menor possível.

- **Capacidade de Endereçamento:** Distinge qual é o menor valor endereçável na memória. No exemplo da Figura 1.2, o menor valor endereçável é composto por 4 *bytes*, o que corresponde a 32 *bits* (4 x 8 *bits*). Existem no entanto memórias em que é possível endereçar *bit* a *bit*.

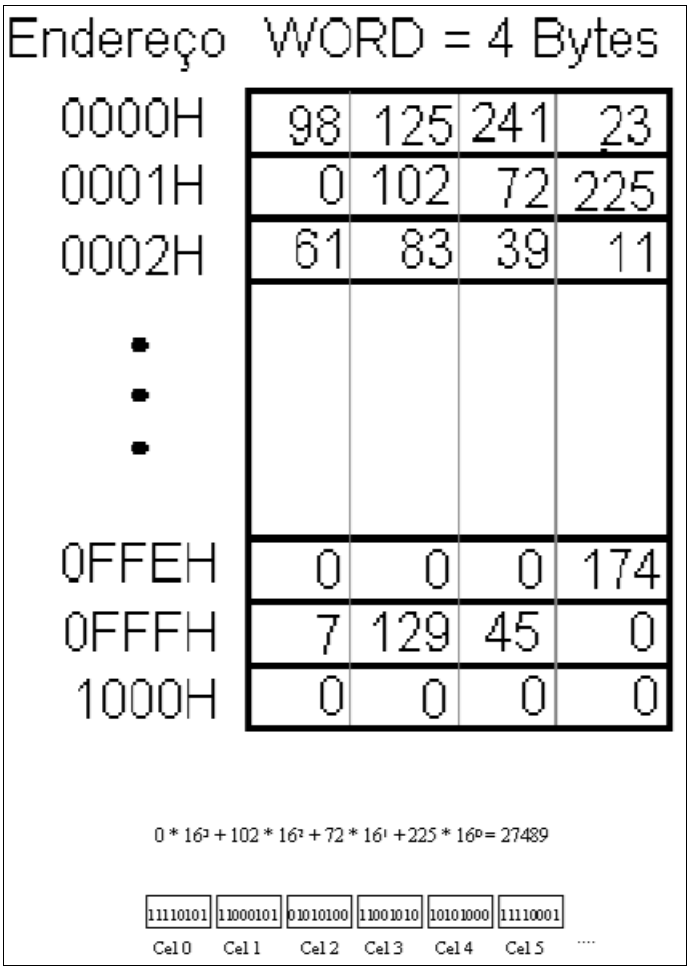


Figura 1.2: *Representação de uma memória cuja capacidade de endereçamento é de 4 bytes. Cada bloco de memória tem, por isso, 4 bytes, e são identificados por endereços únicos, representados em notação hexadecimal (indicada pela letra "H"), desde a posição 0000H até à posição 1000H. A notação hexadecimal será discutida na secção 2.1.3 desta sebenta.*

- **Tamanho da Memória:** Limita a quantidade de informação que é possível armazenar. É medido em *KBytes*, *MBytes* ou *GBytes*.

- **Tipo de Acesso:** Pode ser sequencial ou aleatório.

- **Acesso Sequencial:** Para aceder a uma determinada posição da memória, é necessário ler ou passar por todas as posições anteriores;
- **Acesso Aleatório:** O acesso pode ser feito directamente à posição que se pretende ler ou em que se pretende escrever.

- **Capacidade de Leitura e Escrita:** Todas as memórias permitem que se leia o seu conteúdo (excepto as que são intencionalmente protegidas), mas nem todas permitem que se escreva nelas.
- **Volatilidade:** É a falta de capacidade da memória para reter a informação indefinidamente, eventualmente por falta de corrente de alimentação. Diz-se que uma memória é volátil se todo a informação nela armazenada se perder em caso de falta de alimentação; uma memória será não volátil se a informação armazenada se mantiver, mesmo perante a falta de alimentação.

1.2.2. Memória Primária

Destina-se a armazenar a informação que se encontra a ser processada pelo computador. Pode ser de dois tipos: RAM e ROM.

Memória RAM (*Random Access Memory*) – Memória de Acesso Aleatório

- Permite leitura e escrita;
- Tempos de acesso muito rápidos e do tipo aleatório (isto é, qualquer posição pode ser acedida);
- Volátil (o seu conteúdo é perdido em caso de falha de corrente eléctrica);
- Como possui um preço relativamente elevado (256 *MBytes* custam cerca de 125€), não existe em grande quantidade na maioria dos computadores.

Memória ROM (*Read-Only Memory*) – Memória Apenas de Leitura

- Permite apenas leitura, pois o processo de escrita é realizado apenas uma única vez pelo fabricante;
- Tempo de acesso é baixo (semelhante ao da memória RAM);

- Não volátil (o seu conteúdo nunca se perde);
- Destina-se a armazenar uma parte do programa responsável pela gestão do sistema, que permite inicializar o computador, bem como a formatação e reconhecimento de determinados dispositivos que fazem parte do mesmo equipamento.

1.2.3. Memória Secundária

Destina-se a armazenar, de uma forma permanente, a informação que não se encontra a ser utilizada. Em geral, as unidades de memória secundária possuem grande capacidade de armazenamento. O seu tempo de acesso é bastante superior ao das memórias do tipo primária (isto é, as memórias secundárias são muito mais lentas do que as memórias primárias). Alguns dos tipos de memória secundária são os que se seguem.

Discos Rígidos

- Trata-se de um dos dispositivos mais comuns, que possui dimensões de armazenamento elevadas. Actualmente, os discos rígidos com maior capacidade podem ter mais de 200 *GBytes*.
- Preço por *MByte* muito inferior ao da memória RAM (200 *Gbytes* custam cerca de 125€);
- Permite leitura e escrita, e um acesso praticamente aleatório;
- Tempos de acesso elevados;
- Não volátil;
- Podem ser externos e portáveis.

Disquetes

- Em tudo semelhante aos discos rígidos, mas com capacidades de armazenamento muito menores (1,44 *MBytes*);
- Tempos de acesso superiores (são muito lentas);
- Portáveis;

- Começam a ser abandonadas pelos utilizadores e fabricantes.

CD-ROM

- Sigla da expressão inglesa *"Compact Disc - Read-Only Memory"*;
- Dispositivo que utiliza a tecnologia óptica, igual aos CDs de música;
- Permite armazenar grandes quantidades de informação (650 ou 700 MB), permanentemente;
- É imune a interferências electromagnéticas;
- Transportável;
- Permite leitura aleatória, com tempos de acesso que tendem a aproximar-se aos dos discos rígidos;
- Os CD-RW (*"Compact Disc - Rewritable"*) podem ser gravados mais do que uma vez;
- Os CD-R (*"Compact Disc - Recordable"*) apenas podem ser gravados uma única vez.

DVD

- Sigla da expressão inglesa *"Digital Versatile Disc"*;
- Dispositivo que também utiliza a tecnologia óptica para o armazenamento dos dados;
- Podem armazenar dados em um ou nos dois lados, utilizando uma ou duas camadas de cada lado;
- Possuem capacidades de armazenamento que variam entre 4,7 e mais de 17 *GBytes*;
- Transportável;
- Ideal para armazenar vídeos e músicas;
- Tempos de acesso semelhantes aos dos CDs.

Fitas Magnéticas

- É o processo que permite armazenar maiores quantidades de informação.

- É utilizado, entre outros tipos de organizações, por instituições bancárias, agências de seguros, ou outras organizações que tenham necessidade de salvaguardar gigantescas quantidades de dados.
- Os seus tempos de acesso são muito elevados, devido a ser um sistema ainda muito mecânico e de acesso sequencial.

1.2.4. Memória Cache

Trata-se de uma memória que é continuamente actualizada, de forma a conter a informação usada mais recentemente, reduzindo assim o tempo de acesso à memória principal ou ao disco enquanto um programa é executado.

Cache Interna

- Encontra-se dentro do processador, o que permite uma execução mais rápida das instruções.
- É controlada pela Unidade de Controlo da UCP.
- Memória muito rápida de onde são transferidos os dados directamente para os registos, imediatamente antes de esses dados serem processados pela UAL.
- Existe em vários níveis situados entre a memória principal e os componentes internos da UCP.
- Baixa capacidade de armazenamento (1 *Mbyte*).

1.2.5. Memória Virtual

A memória virtual é, na realidade, uma técnica usada pelos programadores de sistemas operativos para permitir a um computador trabalhar com mais memória RAM do aquela que se encontra instalada na máquina. Esta técnica consiste em passar parte da memória que nõ estiver a ser usada para o disco rígido.

Frequentemente, a memória virtual é gerida organizando a memória em porções de dimensão padronizada, chamadas **páginas**. A troca de páginas de dentro para fora da memória é designada de *"swapping"* ("acto de trocar", em inglês).

1.3. Os Sistemas Periféricos

Um sistema de computação tem que possuir meios de introduzir a informação. Os dispositivos responsáveis por captar esta informação designam-se por **Unidades de Entrada**. De igual modo, também existem dispositivos responsáveis por disponibilizar a informação após o seu processamento, que são as **Unidades de Saída**.

1.3.1. Dispositivos de Entrada

- Leitor de fita perfurada;
- Leitor de código de barras;
- Digitalizadores (*"scanners"*);
- Placas de aquisição de vídeo;
- Câmara de vídeo;
- Teclado;
- *Joystick*;
- Rato e *Trackball*;
- Monitor de toque (*"touchscreen"*);
- Microfone, etc.

1.3.2. Dispositivos de Saída

- Terminal (monitores);
- Placa gráfica;
- Perfuradora de cartão e de fita;
- Impressora;
- Traçador de gráficos e de curvas (*"plotters"*);

- Vídeo-projector, etc.

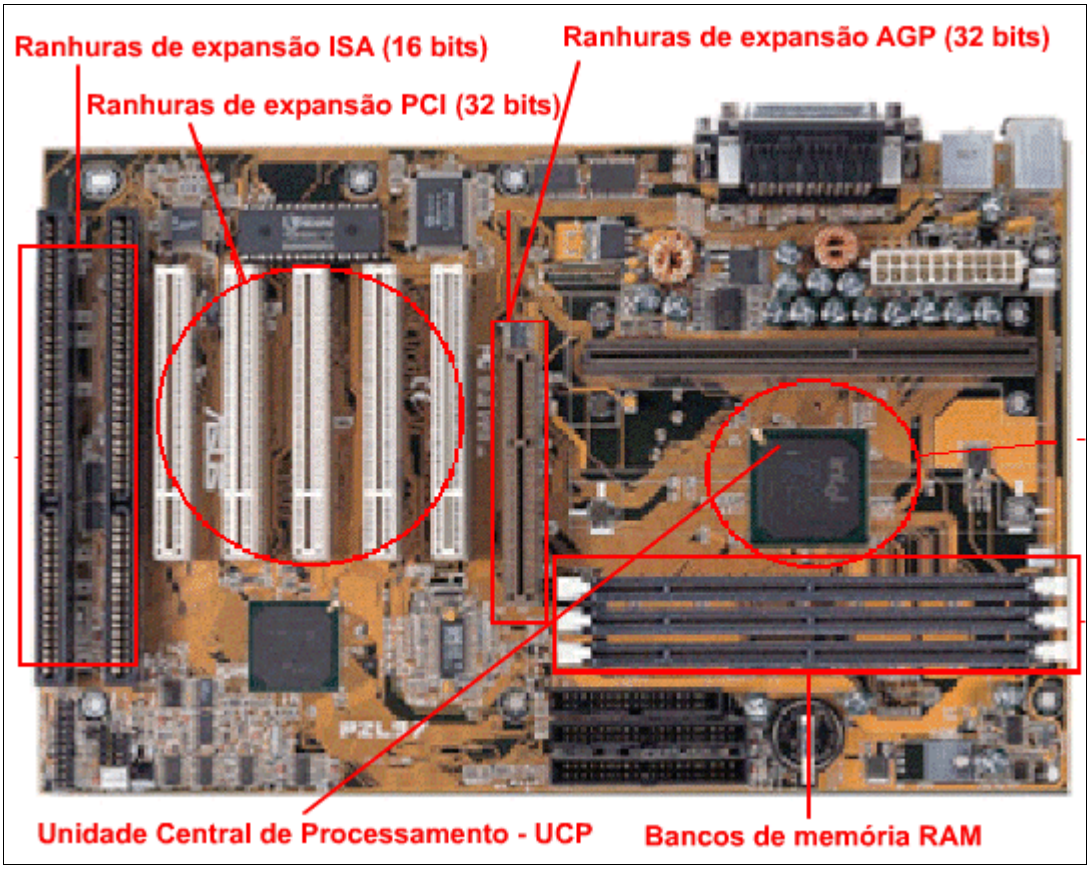


Figura 1.3: *Aspecto geral da placa-mãe (**"motherboard"**) de um computador, estando assinaladas as suas várias secções.*

1.3.3. Dispositivos de Entrada e de Saída

- Porta paralela;
- Porta USB;
- Portas série;
- Leitor/gravador de disquetes;
- Leitor/gravador de CD/DVD;
- Modem/Fax;
- Placa de rede;

- Placa de Som, etc.

1.4. Breve Análise Comparativa

Super Utilizador

- Pentium IV 3,2 GHz
- Placa Gráfica 128 MB DDR
- 1024 MB de Memória RAM
- 200 GB de Disco
- Monitor de 19" TFT
- Gravador e Leitor de DVD
- Placa de Som *Sound Blaster Digital* 5:1
- Colunas *Creative* 5:1

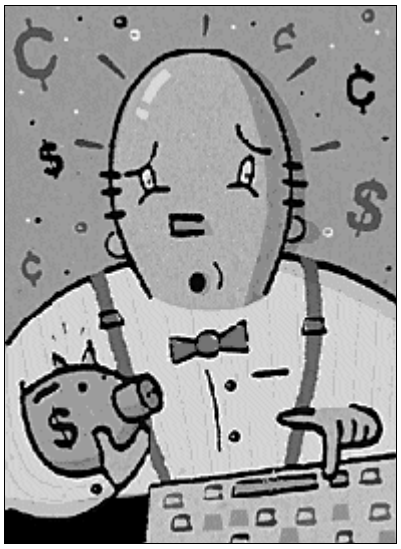
Cerca de 2.000 €

Utilizador médio

- Pentium IV 2,8 GHz
- Placa Gráfica AGP 64 MB
- 512 MB de Memória RAM
- 80 GB de Disco
- Monitor de 17"
- CD-ROM 52X
- Placa de Som 128 *bits*
- Colunas de 360 W

Cerca de 1.000 €

Utilizador poupado



- Intel Celeron 2 GHz
- Placa Gráfica AGP 32 MB
- 256 Mb de Memória RAM
- 60 GB de Disco
- Monitor de 15”

Cerca de 750 €

2 – Armazenamento de Dados

2.1. Armazenamento de Bits

Os computadores representam a informação através de conjuntos de dígitos binários aos quais se dá o nome de *bit* (palavra originária da expressão inglesa *"binary digit"*). Um *bit* é um dos dígitos 0 (zero) ou 1 (um). Cada *bit* é representado por um dispositivo electrónico que está "ligado" (um) ou "desligado" (zero).

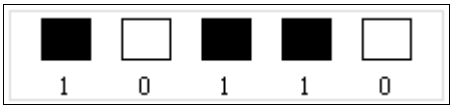


Figura 2.1: *Exemplo de código binário. O dígito binário 1 é representado por um quadrado preenchido, e o dígito binário 0 é representado por um quadrado não preenchido.*

2.1.1. Portas Lógicas

Operações Lógicas ou **Booleanas** são operações que apenas manipulam os valores lógicos **Verdadeiro** (1) e **Falso** (0). As operações de conjunção[3], disjunção[4] e disjunção exclusiva[5] (XOR, ou exclusivo) são operações lógicas efectuadas por circuitos digitais, e são semelhantes às operações de adição e multiplicação. Para dois valores dados de entrada é gerado um único valor de saída. Os únicos dígitos manipulados por estas operações são 0 e 1. A operação de negação[6] é outra operação lógica, onde o valor de saída é o oposto do valor da entrada.

3 Também conhecida como "E lógico", sendo representada pela palavra inglesa AND.

4 Também conhecida como "OU lógico", sendo representada pela palavra inglesa OR.

5 Também conhecida como "OU exclusivo", sendo representada pela palavra XOR.

6 Representada pela palavra inglesa NOT.

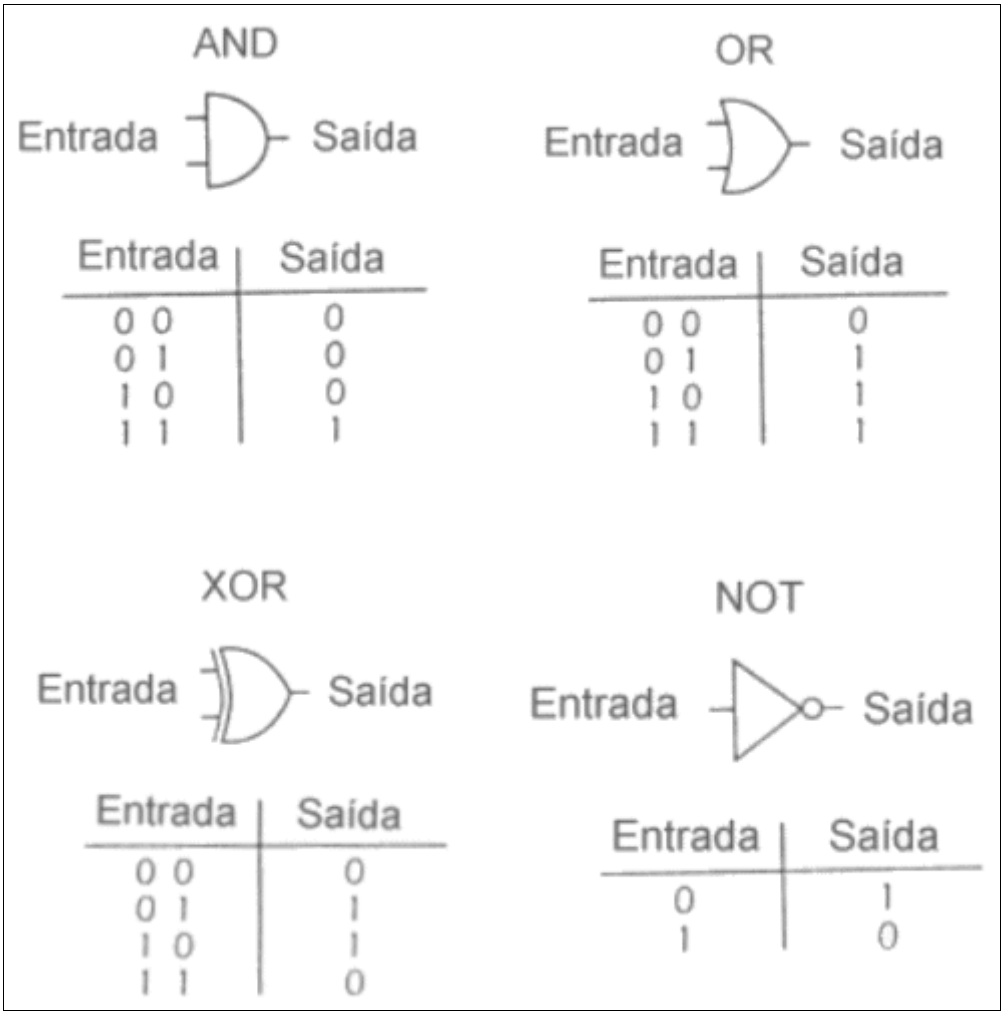


Figura 2.2: *Operações lógicas de conjunção (AND), disjunção (OR), disjunção exclusiva (XOR) e negação (NOT).*

Uma **porta lógica** é um dispositivo que produz uma saída lógica para uma determinada operação lógica de entrada. As portas podem ser construídas utilizando diversas tecnologias, tais como: circuitos electrónicos integrados ou dispositivos ópticos. As portas usadas nos computadores actuais usam pequenos circuitos integrados chamados *chips*. Nesses circuitos integrados, os digitos 0 e 1 são representados por valores de tensão eléctrica de 0 (zero) e 5 volts, respectivamente.

Os computadores são construídos através de blocos que contém portas lógicas. As portas podem ser usadas como blocos constituintes de circuitos com os quais os computadores são construídos, o que permite compreender o funcionamento interno de um computador sem ter necessidade de conhecer a arquitectura das portas.

2.1.2. Flip-Flops

Um *flip-flop* é um circuito em que o valor de saída oscila entre dois valores, mediante os valores de entrada introduzidos. É ideal para representar o armazenamento de um *bit* num computador. Por isso, as memórias internas dos computadores são construídas por blocos de circuitos *flip-flop*.

Um exemplo de um *flip-flop* encontra-se representado na figura 2.3. Este circuito é capaz de armazenar um *bit* da seguinte forma:

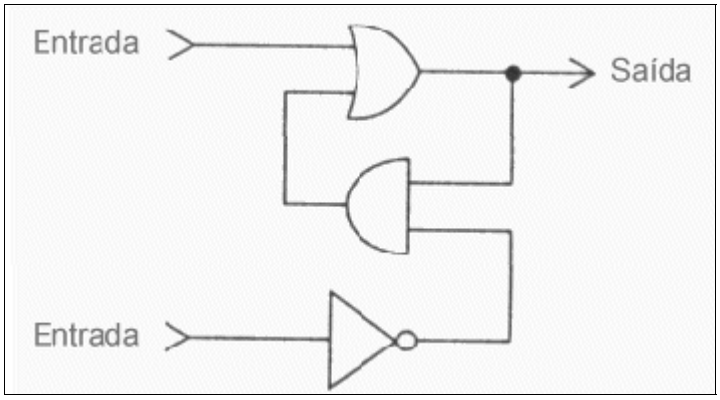


Figura 2.3: *Representação de um circuito lógico conhecido como flip-flop. Uma das principais características deste circuito é a sua capacidade para armazenar um bit.*

- Para manter o valor actual armazenado (representado por X na figura 2.4), basta colocar nas duas entradas do circuito o valor lógico 0 (zero);

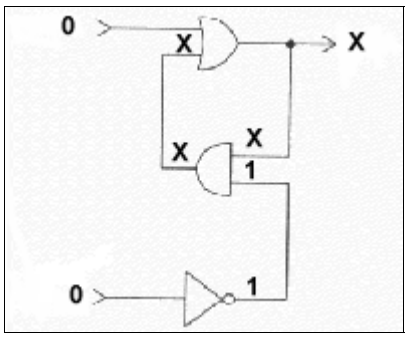


Figura 2.4: *Qualquer que seja o valor da saída deste flip-flop, ele permanecerá constante enquanto as suas duas entradas tiverem o valor lógico 0 (zero).*

- Para guardar o valor lógico 1 (um), basta colocar na entrada de cima o valor lógico 1 (figura 2.5);

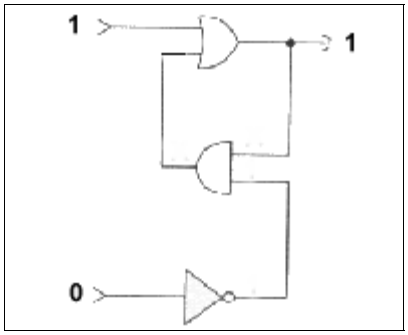


Figura 2.5: *Para colocar o valor lógico 1 (um) na saída deste flip-flop, deve-se colocar o valor lógico 1 na entrada de cima.*

- Para guardar o valor lógico 0 (zero), basta colocar na entrada de cima o valor lógico 0 (figura 2.6).

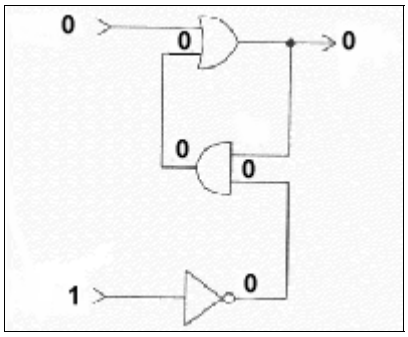


Figura 2.6: *Para colocar o valor lógico 0 (zero) na saída deste flip-flop, deve-se colocar o valor lógico 1 (um) na entrada de baixo.*

2.1.3. Notação Hexadecimal

Ao considerar a actividade interna de um computador, temos que lidar com conjuntos de *bits*, que podem ser mais ou menos longos.

A notação hexadecimal surgiu devido à dificuldade da mente humana em manipular conjuntos de *bits*, e aproveitando a vantagem de os computadores armazenarem sequências binárias em múltiplos de quatro.

A notação hexadecimal usa um símbolo para representar quatro *bits*, tal como se representa na tabela 2.1. Por exemplo, e de acordo com essa tabela, a sequência binária 1011 0101 é representada, em notação hexadecimal, como B5.

Decimal	Binário	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Tabela 2.1: *Tabela com as diferentes combinações binárias de 4 bits, os seus correspondentes símbolos hexadecimais e os seus correspondentes valores decimais.*

2.2. Codificação da Informação para Armazenamento

2.2.1. Representação de Símbolos

Para podermos guardar num computador textos com letras, números, pontuação, etc., foi necessário criar códigos de representação dos diferentes símbolos. Um dos mais conhecidos é o código ASCII, sigla da expressão inglesa *"American Standard Code for Information Interchange*" ("Código Padrão Americano para o Intercâmbio de Informação"), representado na tabela 2.2.

O código ASCII representa todos os caracteres do alfabeto, além de outros caracteres especiais, usando 7 *bits* (2^7 = 128 caracteres) para representar cada caracter. Por exemplo, a palavra "Ola" é codificada, segundo o código ASCII, como `1001111 1101100 1100001`.

```
O  →  4F  →  1001111
l  →  6C  →  1101100
a  →  61  →  1100001
```

Decimal	Hexadecimal	Caracter	Decimal	Hexadecimal	Caracter	Decimal	Hexadecimal	Caracter
0	00	NUL	43	2B	+	86	56	V
1	01	SOH	44	2C	,	87	57	W
2	02	STX	45	2D	-	88	58	X
3	03	ETX	46	2E	.	89	59	Y
4	04	EOT	47	2F	/	90	5A	Z
5	05	ENQ	48	30	0	91	5B	[
6	06	ACK	49	31	1	92	5C	\
7	07	BEL	50	32	2	93	5D	]
8	08	BS	51	33	3	94	5E	^
9	09	HT	52	34	4	95	5F	_
10	0A	LF	53	35	5	96	60	`
11	0B	VT	54	36	6	97	61	a
12	0C	FF	55	37	7	98	62	b
13	0D	CR	56	38	8	99	63	c
14	0E	SO	57	39	9	100	64	d
15	0F	SI	58	3A	:	101	65	e
16	10	DLE	59	3B	;	102	66	f
17	11	DC1	60	3C	<	103	67	g
18	12	DC2	61	3D	=	104	68	h
19	13	DC3	62	3E	>	105	69	i
20	14	DC4	63	3F	?	106	6A	j
21	15	NAK	64	40	@	107	6B	k
22	16	SYN	65	41	A	108	6C	l
23	17	ETB	66	42	B	109	6D	m
24	18	CAN	67	43	C	110	6E	n
25	19	EM	68	44	D	111	6F	o
26	1A	SUB	69	45	E	112	70	p
27	1B	ESC	70	46	F	113	71	q
28	1C	FS	71	47	G	114	72	r
29	1D	GS	72	48	H	115	73	s
30	1E	RS	73	49	I	116	74	t
31	1F	US	74	4A	J	117	75	u
32	20	Espaço	75	4B	K	118	76	v
33	21	!	76	4C	L	119	77	w
34	22	"	77	4D	M	120	78	x
35	23	#	78	4E	N	121	79	y

Decimal	Hexadecimal	Caracter	Decimal	Hexadecimal	Caracter	Decimal	Hexadecimal	Caracter
36	24	$	79	4F	O	122	7A	z
37	25	%	80	50	P	123	7B	{
38	26	&	81	51	Q	124	7C	|
39	27	'	82	52	R	125	7D	}
40	28	(	83	53	S	126	7E	~
41	29	)	84	54	T	127	7F	DEL
42	2A	*	85	55	U			

Tabela 2.2: *Tabela ASCII.*

2.2.2. Representação de Números

Para representar valores numéricos recorrendo ao código ASCII, seria necessário 1 *byte* (8 *bits*) por cada símbolo. Usando a representação binária (de base 2), pode-se representar o mesmo número com menos *bits*.

Usando a notação binária, num *byte* podemos armazenar inteiros entre 0 e 255 (00000000 e 11111111, respectivamente). Usando 16 *bits* (2 *bytes*), podemos representar valores numéricos entre 0 e 65.535, enquanto que, se usarmos a codificação ASCII, apenas seria possível representar valores numéricos entre 0 e 99.

Dada uma representação binária (com n *bits*) de um valor numérico, é possível determinar esse valor através da seguinte expressão: $\mathbf{valor\ decimal = a_n * 2^n + a_{n-1} * 2^{n-1} + \ldots + a_1 * 2^1 + a_0 * 2^0}$, em que a_n, a_{n-1}, ..., a_1, a_0, representam cada um dos *bits* da representação binária (ver figura 2.7).

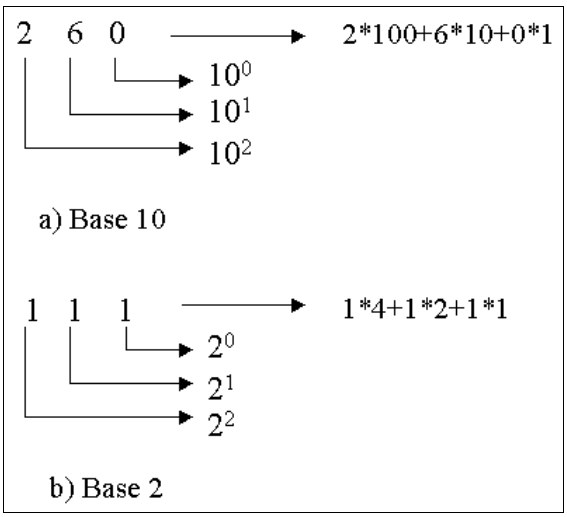


Figura 2.7: *Determinação do valor numérico de representações numéricas de base 10 e de base 2.*

O método de conversão de um número inteiro positivo para binário é conhecido como o "método das divisões sucessivas", e é composto por 3 passos, representados na figura 2.8:

1. Dividir o número por 2;
2. Dividir o quociente sucessivamente por 2 até que seja igual a zero;
3. O número em binário obtido corresponde aos valores dos restos das divisões, lidos da direita para a esquerda (ou de cima para baixo).

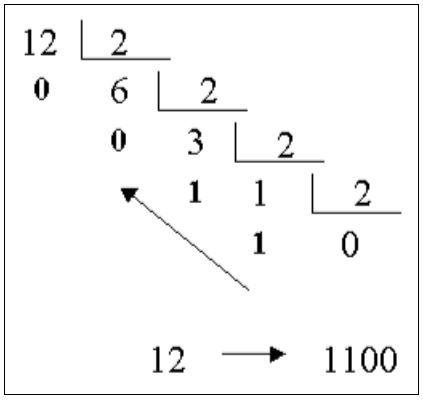


Figura 2.8: *Conversão de decimal para binário, através do método das divisões sucessivas.*

2.3. O Sistema Binário

2.3.1. Adição Binária

2	4	8	8
+ 3	+ 5	+ 9	+ 10
5	9	17	18

Figura 2.9: *Adição decimal.*

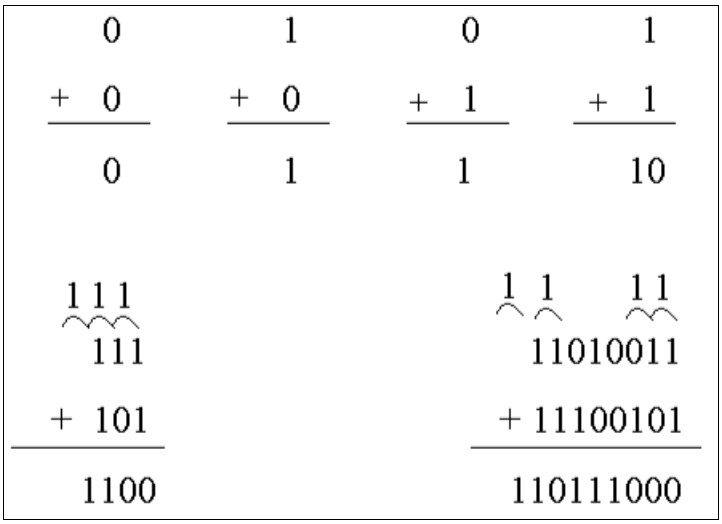


Figura 2.10: *Adição binária.*

2.3.2. Fracções Binárias

Dada uma representação binária com uma parte fraccionária (com n *bits* à esquerda da vírgula decimal, e m *bits* à direita da vírgula decimal) de um valor numérico fraccionário, é possível determinar esse valor através da seguinte expressão: **valor decimal = $a_n * 2^n + a_{n-1} * 2^{n-1} + \ldots + a_1 * 2^1 + a_o * 2^o + a_{-1} * 2^{-1} + \ldots + a_{-m} * 2^{-m}$**, em que a_n, a_{n-1}, ..., a_1, a_o, representam cada um dos *bits* da parte interia da representação binária, e em que a_{-1}, ..., a_{-m}, representam cada um dos *bits* da parte fraccionária da representação binária (ver figura 2.11).

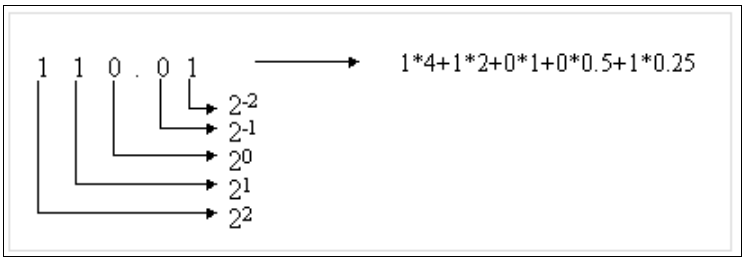


Figura 2.11: *Conversão de fracções binárias para decimal.*

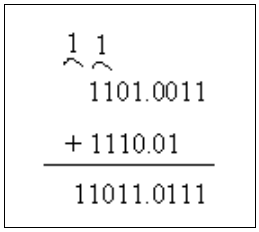


Figura 2.12: *Adição de números fraccionários binários.*

2.4. Armazenamento de Inteiros

2.4.1. Complemento para Dois

Para determinar um valor negativo pelo complemento para 2, é necessário negar *bit* a *bit* e, em seguida, somar 1. Por exemplo, vamos determinar o representação binária do simétrico[7] do número 96, representado em base decimal.

$+96_{(10)} = 01100000_{(2)}$

Para se obter a representação binária do valor simétrico de 96 usando a técnica do complemento para 2, nega-se a representação binária de 96, *bit* a *bit*, e obtém-se:

$10011111_{(2)}$

Por fim, deve-se somar 1 à representação binária obtida no passo anterior:

```
10011111
+       1
--------
10100000 → Complemento para 2
```

Existe um outro processo de aplicação do complemento para 2 para passar um número de positivo para negativo: copia-se os dígitos da direita para a esquerda até se encontrar um 1. A partir daí, troca-se os zeros por uns e os uns por zeros.

O primeiro *bit* indica o sinal do número binário: se for zero, o número é positivo; se for um, o número é negativo.

7 O simétrico de um número e o número com o mesmo valor, mas de sinal contrário. Por exemplo, o simétrico de 96 é -96, e vice-versa.

Binário (3 bits)	Decimal	Binário (4 bits)	Decimal
011	3	0111	7
010	2	0110	6
001	1	0101	5
000	0	0100	4
111	-1	0011	3
110	-2	0010	2
101	-3	0001	1
100	-4	0000	0
		1111	-1
		1110	-2
		1101	-3
		1100	-4
		1011	-5
		1010	-6
		1001	-7
		1000	-8

Tabela 2.3: *Complemento para dois usando 3 e 4 bits.*

3 – Manipulação de Dados

3.1. Conceito de Programa

Um **programa** é um conjunto de instruções que indica a um computador o que este deve fazer. Os programas permitem a um computador funcionar e contém geralmente uma lista de **variáveis** que representam números, texto, gráficos, e outros tipos de informações. Um programa contém também uma série de **procedimentos** que dizem ao computador o que fazer com tais variáveis.

Os programadores usam **linguagens de programação** para criar programas, mas todos os programas escritos nessas linguagens têm de ser traduzidos para linguagem máquina antes de o computador poder executar esses programas.

O processador de um computador funciona apenas com uma linguagem especifica, chamada de **código** ou **linguagem máquina**. Esta linguagem é constituída exclusivamente por dígitos binários (zeros e uns). Ainda que seja possível fazer um programa directamente em código máquina, tal tarefa é extremamente morosa e muito sujeita a erros.

Assim, foram desenvolvidas linguagens de programação de **baixo** e **alto nível**, consoante estão mais próximas do código máquina ou da linguagem humana, respectivamente. É nessas linguagens que são escritos os programas que os computadores executam; esses programas são designados como **programas fonte** ou **código fonte**.

Os **programas tradutores** têm, então, a função de converter um programa escrito numa linguagem de programação (programa fonte) para código máquina (figura 3.1).

As linguagens de mais baixo nível são as linguagens de montagem, mais conhecidas como linguagens ***assembly***. Um programa escrito numa dessas linguagem necessita de um tradutor para que as suas instruções sejam convertidas para código máquina; a esse tradutor chama-se **assemblador**.

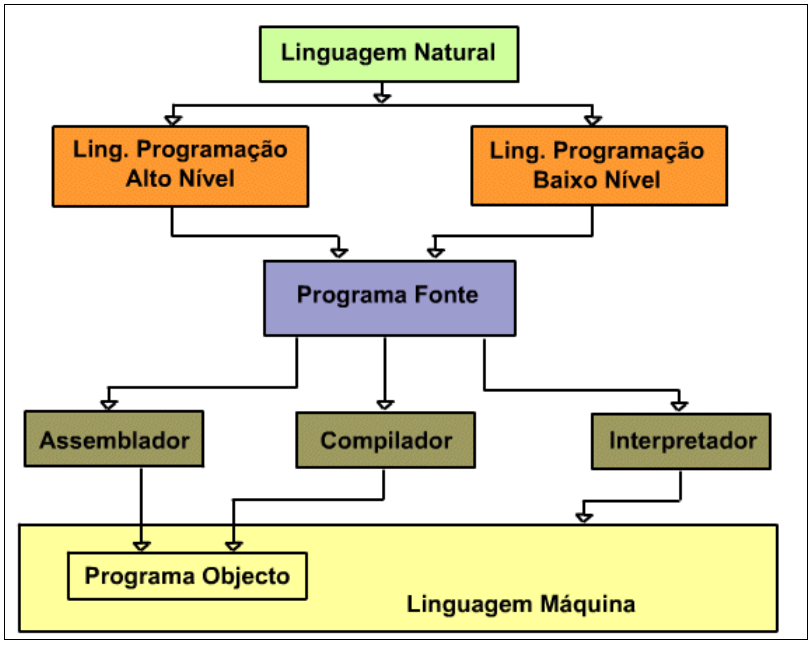


Figura 3.1: *Representação dos vários tipos de interpretação de um programa escrito numa linguagem de programação.*

As linguagens de alto nível – Pascal, Fortran, C, etc. – requerem igualmente programas tradutores. A tradução destas linguagens de alto nível para linguagem máquina pode efectuar-se por dois processos diferentes:

1. **Compilação**: neste processo, o tradutor (ao qual se dá o nome de **compilador**) analisa todo o código fonte e, se o programa não contiver nenhum erro de sintaxe[8], converte o código fonte em código máquina que poderá ser executado directamente pela UCP.

2. **Interpretação**: neste processo, o tradutor (ao qual se dá o nome de **interpretador**) analisa uma linha do programa fonte de cada vez; se a linha não contiver nenhum erro de sintaxe, o interpretador traduz essa linha em código máquina que poderá ser executado directamente pela UCP.

Para traduzir um programa para código máquina, são necessários os seguintes passos:

1. O tradutor é carregado para a memória central através do sistema operativo;

2. O programa fonte é introduzido no computador, ou por entrada directa (teclado) ou a partir da memória secundária (disco, disquete, CD, etc.);

8 Sintaxe é o conjunto de regras de escrita a que todos os programas escritos numa determinada linguagem de programação terão de obedecer para que o programa possa ser executado.

3. O tradutor gera o **programa objecto** (isto é, o programa que contém o código-máquina pronto a ser executado) que fica gravado em memória; ao mesmo tempo, se forem detectados erros de sintaxe, serão enviadas ao programador mensagens informativas;

4. O programa objecto está em condições de ser executado.

3.2. Execução de Programas

Um computador executa um programa guardado na memória copiando as instruções desse programa para a Unidade de Controlo (UC) da UCP, uma de cada vez. A ordem pela qual as instruções são extraídas da memória corresponde à ordem pela qual se encontram armazenadas na memória, excepto se houver uma instrução de salto (isto é, uma instrução que altera a ordem normal de execução de um programa, através da indicação explícita da próxima instrução que deverá ser executada).

A UC contém dois registos essenciais para a execução de um programa, representados na figura 3.2[9]:

1. O **contador de programa** (*"program counter"*, mais conhecida pela sigla **PC**) contém o endereço da próxima instrução a ser executada. É uma forma expedita de se saber qual a próxima instrução em qualquer parte do programa;

2. O **registo de instruções** (*"instruction register"*, mais conhecida pela sigla **IR**) é usado para guardar a instrução que está a ser executada.

9 Para se compreender o processo de execução de um programa pela UCP de um computador, é necessário compreender a arquitectura interna da UCP. Assim, recorde-se que a UCP é constituída por uma unidade de controlo (UC), uma unidade aritmética e lógica e por registos internos que formam a sua memória interna. A UC possui ainda dois registos especiais, o Contador de Programa (PC) e o Registo de Instrução (IR), que permitem controlar a execução dos programas armazenados na memória central do computador. A UCP está fisicamente ligada a essa memória através de um canal de fios condutores, chamado barramento, através do qual os dados são transmitidos. Por fim, a memória central está dividida em células, todas com o mesmo tamanho. Cada célula possui um endereço que a identifica univocamente.

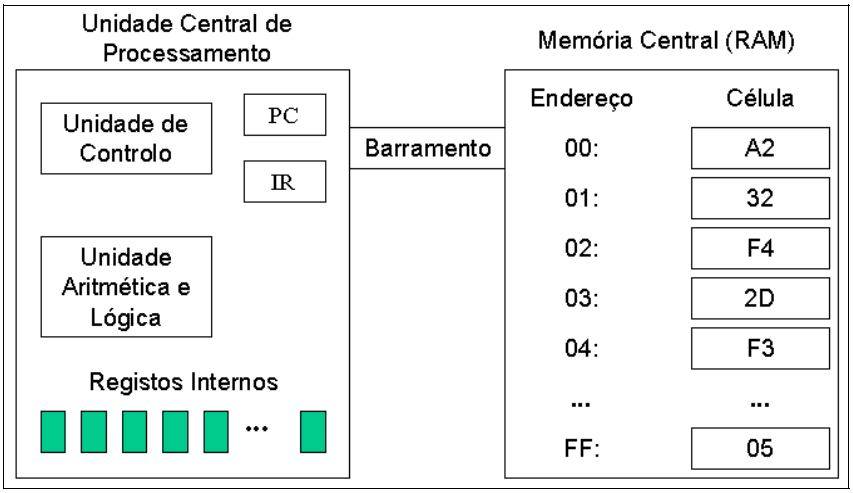


Figura 3.2: *Arquitectura típica de uma unidade central de processamento.*

A UC guarda no registo IR a instrução que está armazenada na posição de memória cujo endereço está guardado no registo PC. Em seguida, a UC executa a operação correspondente à instrução guardada no IR. Entretanto, o PC já foi actualizado com o endereço da próxima instrução a ser executada. Em geral, a próxima instrução a ser executada é a que se encontra imediatamente a seguir à instrução actualmente em execução. Contudo, essa ordem de execução pode ser alterada por instruções de salto. Este processo repete-se para todas as instruções de um programa, até que a instrução que indica o fim da execução de um programa seja executada. A este ciclo de operações dá-se o nome de **ciclo da máquina**, e está representado na figura 3.3.

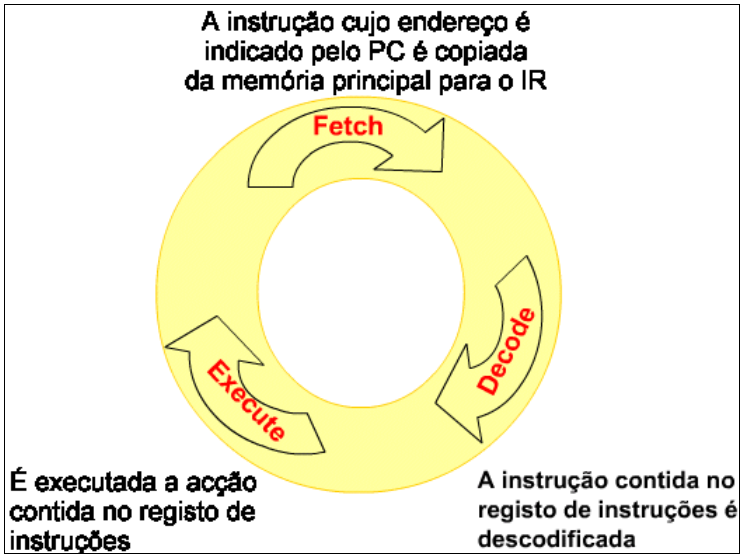


Figura 3.3: *O ciclo da máquina.*

3.3. Comunicação Computador-Periférico

A comunicação entre o CPU e um periférico é normalmente manipulada por um dispositivo chamado **controlador**. Os controladores de periféricos têm como principal função a conversão dos dados utilizados pela UCP para os formatos utilizados internamente por cada periférico controlado, e vice-versa. Os controladores são ligados ao mesmo barramento que liga a UCP à memória principal (figura 3.4).

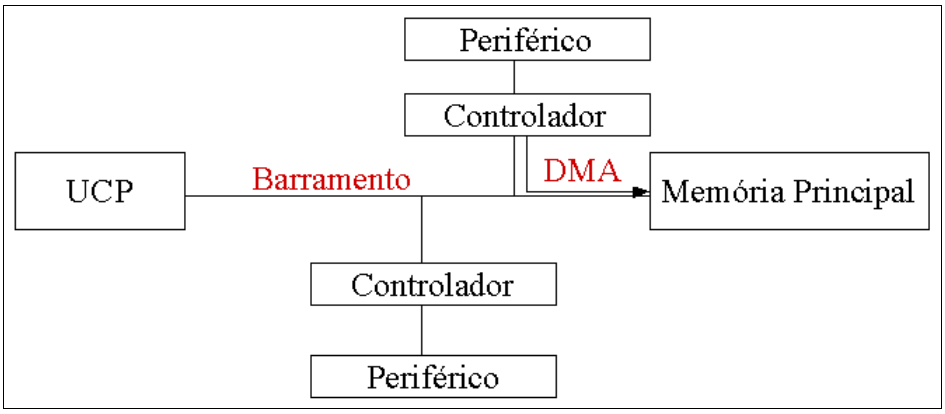


Figura 3.4: *A comunicação entre a UCP de um computador e um periférico é feita através de um controlador, que é ligado ao mesmo barramento que liga a UCP à memória central desse computador. A capacidade que um controlador tem para aceder directamente à memória central é conhecida como Acesso Directo à Memória ("Direct Access Memory", cuja sigla é DMA).*

Alguns controladores têm a capacidade de ler e escrever na memória central do computador sempre que o processador não estiver a utilizar o barramento. Essa capacidade de um controlador de periférico aceder directamente à memória central é conhecida como **acesso directo à memória** (*"direct access memory"*, representada pela sigla DMA), também representada na figura 3.4. Esta funcionalidade contribui para aumentar o desempenho global do sistema, mas também ajuda a aumentar em muito a sua complexidade.

3.3.1. Comunicação Série

- É transmitido um *bit* de cada vez;
- A transmissão é lenta;
- É necessário apenas uma linha para a transmissão dos dados;
- Permite maiores distâncias (até 15 metros);

- Usando um modem[10], pode-se comunicar para qualquer parte do mundo através da rede telefónica.

3.3.2. Comunicação Paralela

- Todos os *bits* de uma combinação binária, com um tamanho pré-determinado, são transmitidos ao mesmo tempo;
- Utiliza várias linhas de transmissão de dados, uma para cada *bit* transmitido;
- A transmissão é bastante mais rápida que a comunicação série;
- Transmite unicamente a curtas distâncias.

10 Um ***modem*** (contracção da expressão "modulador-demodulador", que representa algo que converte, ou **modula**, um sinal digital num sinal analógico, e que posteriormente reconverte, ou **demodula**, esse sinal analógico no sinal digital inicial) é um aparelho electrónico que converte os sinais digitais utilizados por um computador nos sinais analógicos que podem ser transmitidos através da rede telefónica, e vice-versa.

4 – Sistemas Operativos

4.1. A Evolução dos Sistemas Operativos

4.1.1. Classificação do Software

Designa-se por *software* a parte lógica do sistema informático. O *software* pode ser classificado de duas maneiras, representadas pela figura 4.1:

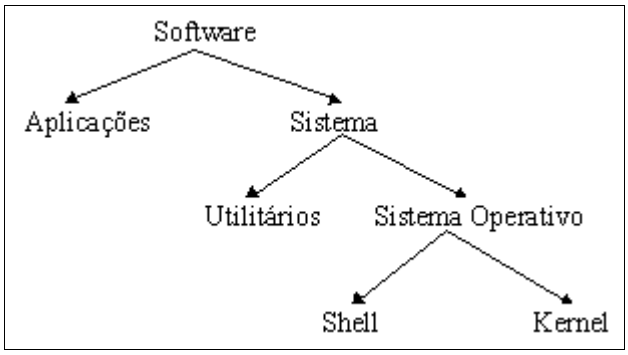


Figura 4.1: *Classificação do software.*

1. ***Software* de base ou de sistema:** Conjunto de pequenos programas ou procedimentos que fazem a gestão de recursos e operações de base de um sistema informático. Inclui-se aqui o núcleo do sistema operativo e um conjunto de programas auxiliares que normalmente o acompanham.

2. ***Software* de aplicações:** Conjunto de programas que desempenham tarefas específicas para a utilização do computador. Alguns exemplos: compiladores, interpretadores, utilitários, bibliotecas, gestores de base de dados, processadores de texto, folhas de cálculo, programas para a construção de páginas *web*, etc.

4.1.2. Tipos de Processamento

Processamento em Lotes (*"Batch"*)

No inicio da computação, todas as tarefas eram processadas em lotes: as tarefas a executar eram inseridas no computador, armazenadas internamente, e processadas sequencialmente. Os lotes correspondiam aos conjuntos de cartões perfurados que continham o código máquina relativo à tarefa que se queria executar.

Os utilizadores tinham que aguardar pelo fim do processamento dos lotes em fila de espera para que o seu lote pudesse ser processado.

Processamento Interactivo

É executado tanto em terminal partilhado com outros utilizadores (chamado também de **consola**), como num computador pessoal. O utilizador e o computador interagem durante a sessão. No entanto, o utilizador continua a ter de esperar pelo fim do processamento das tarefas de outros utilizadores para que as suas tarefas possam ser processadas.

Partilha do Tempo (*"Timesharing"*)

É uma forma de processamento interactivo no qual muitos utilizadores podem utilizar um único computador simultaneamente. Como o computador opera muito mais rápido que o utilizador no terminal, o utilizador tem a sensação de que tem o computador apenas à sua disposição.

4.2. Arquitectura

Um sistema operativo é um conjunto integrado de rotinas especializadas, utilizadas para controlar a generalidade dos recursos e funções de um computador. Um sistema operativo também tem como objectivo principal criar um ambiente de trabalho para o utilizador e servir de intermediário (através de uma interface, mais ou menos amigável) entre o *hardware* do computador e o utilizador, tal como é representado na figura 4.2.

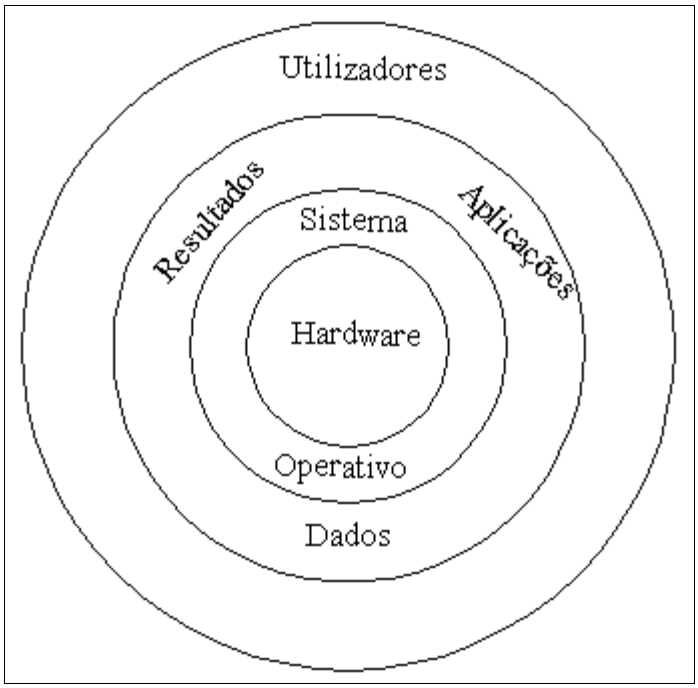


Figura 4.2: *Arquitectura de um sistema operativo.*

Cabe assim ao sistema operativo entrar em contacto com o BIOS (*"Basic Input Output System"*, expressão inglesa que significa "Sistema Básico de Entrada e Saída") para poder gerir o funcionamento básico do sistema computadorizado. O BIOS tem a responsabilidade de conhecer os complexos mecanismos de comunicação entre as diferentes partes do sistema informático (o processador, as memórias e as outras componentes, periféricos, etc.). Em geral, o BIOS é definido pelo fabricante do computador e encontra-se na memória ROM da máquina.

As principais funções de um sistema operativo são:

- **Gestão da Memória:** Alocar zonas de memória a cada um dos programas em execução, e garantir que um programa em execução não acede à zona de memória de um outro programa em execução.
- **Gestão da Unidade Central de Processamento:** Decidir qual o programa que, a cada momento, o processador deve executar.
- **Controlo dos Periféricos:** Controlar e coordenar as actividades dos equipamentos periféricos.
- **Carregamento de Programas:** Transferir da memória externa para a memória interna os programas que deverão ser executados.

- **Preparação da execução:** Confirmar que os ficheiros e recursos de *hardware* estão disponíveis para utilização antes do início da execução de um programa.
- **Finalização da execução:** Libertar a zona de memória e demais recursos de h*ardware* que estavam afectos ao programa.
- **Gestão da fila de espera:** Assegurar a continuidade das tarefas a realizar em cada momento.

4.2.1. Shell

A ***"shell"*** ("concha", em inglês) é um programa que fornece uma **interface** entre o sistema operativo e o utilizador. A função da *shell* é esconder as complexidades internas do computador, tornando mais acessível a utilização de um equipamento informático por vários tipos de utilizadores.

Através da *shell*, os utilizadores comunicam ao computador as tarefas que eles pretendem ver executadas (figura 4.3).

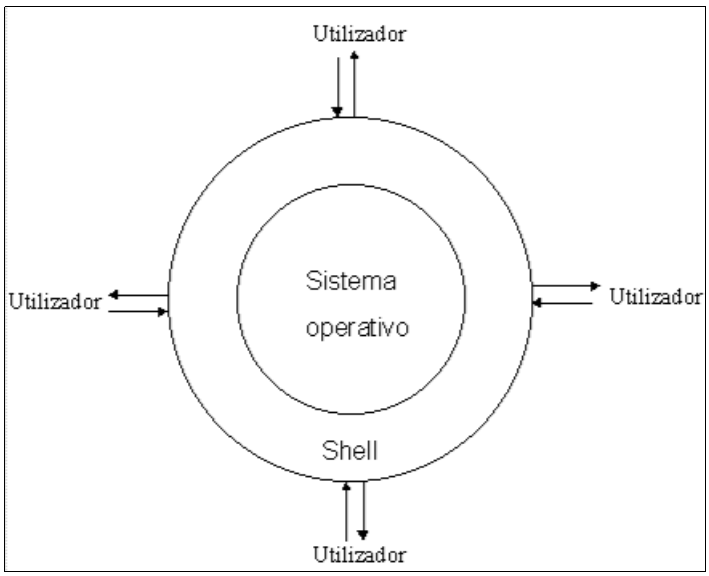


Figura 4.3: *A shell de um sistema operativo oferece uma interface através da qual os utilizadores podem utilizar o computador sem necessitarem de conhecer as suas complexidades internas.*

As *shells* mais modernas oferecem uma interface gráfica com a qual os objectos (ficheiros, programas, e outros dispositivos) podem ser manipulados com dispositivos móveis como, por exemplo, o rato.

4.2.2. Núcelo

O núcleo (também conhecido pela palavra inglesa *"kernel"*) é a parte interna do sistema operativo, a qual contém componentes de software que efectuam as tarefas correspondentes às funções básicas do sistema operativo (por exemplo, gestão de ficheiros, gestão dos periféricos, gestão de memória, etc.).

Esta é a parte de um sistema operativo com a qual o utilizador interage através da interface oferecida pela *shell* do sistema operativo. Esta é também a componente que actua directamente sobre o *hardware* da máquina.

4.2.3. Arranque do Computador

Quando se liga um computador, vendo o caso particular dos PC's, pode verificar-se uma das duas situações seguintes:

1. O computador tem um disco já preparado com o sistema operativo, o qual é transmitido automaticamente no arranque à UCP;

2. O computador não têm disco (ou este não têm o sistema operativo) e então o arranque do sistema terá de fazer-se a partir de um suporte físico (uma disquete, um CD, etc.) que contenha o sistema operativo.

Após a leitura do código máquina do sistema operativo, esse código é armazenado numa parte da memória RAM. Desta forma, uma parte do sistema operativo fica em condições de controlar globalmente o funcionamento do sistema.

4.3. Coordenação da Actividade da Máquina

Um **processo** é a parte activa de um programa, caracterizando o **estado** actual de uma actividade (instrução actual, conteúdo dos registos, conteúdo da memória, etc.). Pode-se dizer que um processo define o caminho de uma actividade em execução.

Um programa pode estar associado a vários processos ao mesmo tempo. Além disso, um processo pode pertencer à execução de uma aplicação ou de programas utilitários, bem como a porções do sistema operativo.

Os processos vão concorrer, uns contra os outros, pelo acesso à UCP, aos periféricos, memória e dados. Para evitar esta "guerra", cabe ao sistema operativo coordenar a execução dos processos, determinando, em cada momento, que processo pode aceder a que recurso. Neste contexto, coordenar significa que:

- Cada processo tem os recursos de que necessita em cada instante;
- Os processos não interferem entre si;
- Os processos são capazes de trocar informações entre si.

Em cada instante, um processo poderá estar em um de três estados:

1. Em execução;
2. Pronto a executar (à espera que a UCP possa executar as suas instruções);
3. À espera de algum outro recurso do *hardware*.

O sistema operativo divide o tempo em pequenos intervalos (de mais ou menos 50 milissegundos de duração), e permite que um determinado processo possa ser executado pela UCP durante esse intervalo de tempo. Ao fim desse intervalo, a execução do processo é interrompida, o seu estado é salvaguardado, o sistema operativo determina qual o próximo processo será executado pela UCP, esse processo volta a poder usufruir do recurso UCP, e assim por diante. Esta sequência de passos é ilustrada pela figura 4.4.

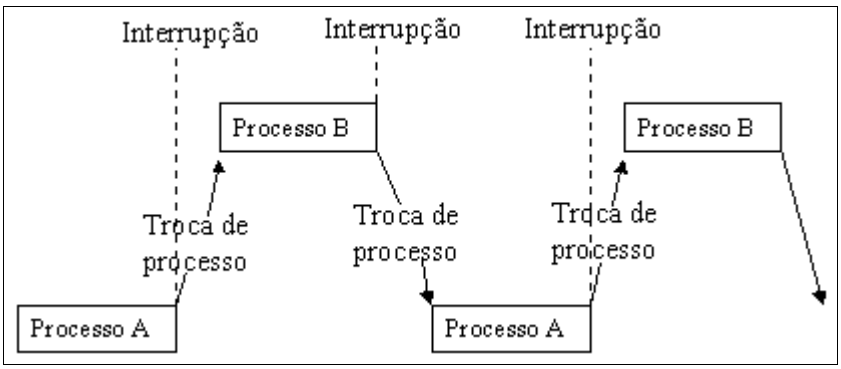


Figura 4.4: *Coordenação, por parte do sistema operativo, dos processos em execução num computador.*

4.4. Exemplos de Sistemas Operativos

4.4.1. MS-DOS – *Microsoft Disk Operating Sistem*

O sistema operativo MS-DOS é uma versão desenvolvida pela empresa norte-americana *Microsoft* do sistema operativo DOS, o qual partilha muitas das funções do sistema operativo PC DOS, da IBM, lançado em 1981. Trata-se de um sistema operativo que controla operações de entrada e saída, periféricos e outras funções de processador. A sua *shell* resume-se a uma linha de comandos, e é implementada através do ficheiro **command.com**.

As principais características deste sistema operativo são as seguintes:

- **Monotarefa:** Permite que apenas um programa seja executado de cada vez;
- **Monoutilizador:** O ambiente de trabalho é único, sendo só possível um único utilizador trabalhar numa mesma máquina de cada vez.

4.4.2. Unix

Trata-se de um sistema operativo que foi desenvolvido pelos Laboratórios Bell, da empresa norte-americana AT&T, no início da década de 1970. Foi escrito na linguagem de programação C, uma linguagem muito popular também no mundo dos PC baseados em processadores da Intel e nos sistemas operativos da Microsoft.

Cada utilizador de um sistema informático baseado em Unix tem uma **conta** identificada por um **nome de utilizador** (*"login"*) e autenticada através de uma **palavra-passe**, cabendo ao **administrador** do sistema gerir as contas dos utilizadores e todos os recursos computacionais colocados à disposição desses utilizadores.

Existem actualmente muitas variantes deste sistema operativo, algumas delas são comerciais (Solaris, Digital Unix, Ultrix, HP-UX, etc), e outras são gratuitas (Linux, FreeBSD, etc.).

As principais características deste sistema operativo são as seguintes:

- **Multitarefa:** Permite a execução simultânea de mais do que um processo;
- **Multi-utilizador:** Permite ter mais do que um terminal ou estação de trabalho ligado ao computador.

4.4.3. Windows 95, 98 e ME

Trata-se uma família de sistemas operativos de 32 *bits*, mas que são capazes de executar também aplicações de apenas 16 *bits* (baseadas na versão anterior desta família, o Windows 3.1) e aplicações baseadas em MS-DOS. Foram lançados pela Microsoft nos anos de 1995 (Windows 95), 1998 (Windows 98) e 2000 (Windows ME).

Estas versões do sistema operativo Windows não necessitam do MS-DOS para poderem ser executados, tornando assim este último sistema operativo completamente obsoleto.

Baseiam-se numa ambiente de trabalho (*shell*) gráfico, proporcionando ao utilizador uma interface amigável baseado em janelas, menus e ícones.

Uma das suas características mais importantes é o facto de ser multitarefa, cabendo assim ao sistema operativo a função de atribuir a cada processo em execução um determinado tempo de processador. Desta forma, o acesso de cada aplicação à UCP é gerido de uma forma mais racional.

4.4.4. Windows NT 4, 2000 e XP

Trata-se de uma família de sistemas operativos de 32 *bits*, porque executam instruções em blocos de 32 *bits*. As suas interfaces gráficas são muito semelhantes (ou, em alguns casos, iguais) às da família de sistemas operativos analisada no subcapítulo anterior.

Os sistemas operativos **Windows NT Server** e **Windows 2000 Server** são sistemas operativos servidores, isto é, foram especialmente desenvolvidos para serem utilizados para a partilha de recursos a todos os computadores de uma rede. Por outro lado, os sistemas operativos **Windows NT Workstation**, **Windows 2000 Professional** e **Windows XP** são especialmente vocacionados para a utilização em estações de trabalho.

5 – Redes de Computadores

5.1. Perspectiva Histórica

Inicialmente os computadores funcionavam, em geral, de forma isolada, comunicando para o exterior apenas através da rede telefónica ou de redes proprietárias. Actualmente, muitas vezes parece que não é possível trabalhar em condições satisfatórias se ocorrer uma qualquer falha, mesmo que muito pequena, nas comunicações entre os computadores de uma rede. Os sistemas informáticos deixaram de ser predominantemente centralizados para darem lugar a um modelo de operação em rede, em que as diversas componentes do sistema encontram-se distribuídas por uma área geográfica mais ou menos extensa.

A disponibilidade quase imediata da informação é um benefício que cada vez menos se dispensa, e que gera novas necessidades, impulsionando o desenvolvimento tecnológico. Assiste-se, assim, a uma convergência crescente entre a indústria dos computadores e a área das comunicações.

5.2. Principais Características das Redes

Uma **rede de computadores** pode ser definida como uma colecção de computadores autónomos interligados através de um meio físico de transmissão, e que são capazes de trocarem informação. Algumas utilizações típicas de uma rede de computadores são as seguintes:

- Partilha de recursos (*hardware* ou *software*);
- Aumento da fiabilidade global do sistema informático através da redundância (isto é, a salvaguarda dos recursos mais críticos do sistema em mais do que uma localização física);
- Economia: melhor relação entre o custo e o desempenho do que os sistemas informáticos baseados em grandes computadores centrais (mais conhecidos pela sua

designação em inglês, *"mainframes"*), o que dá origem a uma melhor rentabilização dos recursos;

- Escalabilidade: possibilidade de incrementar o desempenho global do sistema graças à facilidade com que se adicionam novos subsistemas (por exemplo, novos computadores clientes ou servidores);
- Meio de comunicação: por exemplo, correio electrónico (*e-mail*), videoconferência, trabalho cooperativo, acesso a bases de dados remotas, comércio e banca electrónica, lazer, etc.

As redes de computadores podem ser classificadas de diversas formas:

- Topologia Física:
  - Anel (figura 5.1);

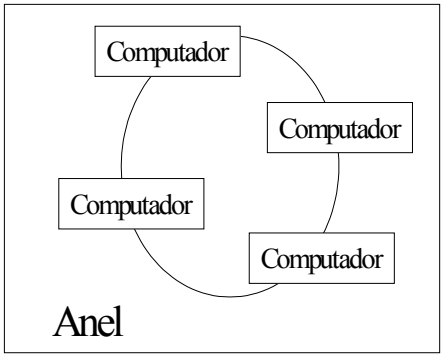


Figura 5.1: *Rede com topologia física do tipo anel.*

  - Estrela (figura 5.2);

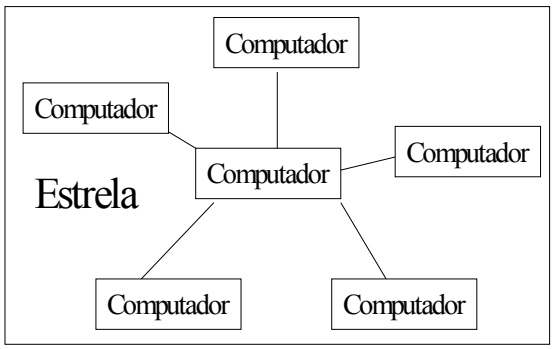


Figura 5.2: *Rede com topologia física do tipo estrela.*

- Barramento (figura 5.3);

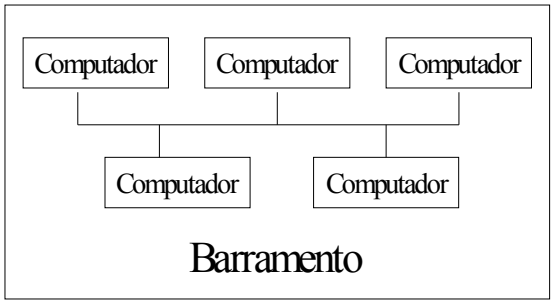


Figura 5.3: *Rede com topologia física do tipo barramento.*

- Irregular ou emaranhado (figura 5.4).

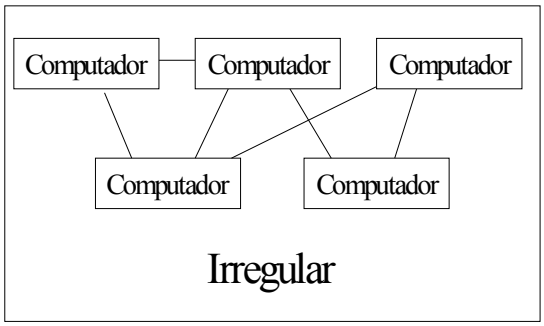


Figura 5.4: *Rede com topologia física do tipo irregular ou emaranhado.*

- Topologia Lógica:
  - Barramento (exemplo: IEEE 802.3 – *Ethernet*, a tecnologia de rede mais usada em redes locais);
  - Anel (exemplos: IEEE 802.4 - *Token Bus*, IEEE 802.5 - *Token Ring*).
- Área de Cobertura:
  - Redes Locais (*"Local Area Network"*, mais conhecida pela sua sigla LAN): *Ethernet*, *Fast-Ethernet*, *Token Ring*, etc.
  - Redes Metropolitanas (*"Metropolitan Area Network"*; sigla MAN): CATV, Anéis DQDB, etc.
  - Redes de Área Alargada (*"Wide Area Network"*, sigla WAN): Internet, VPN's, etc.

- Propriedade:
  - Redes públicas;
  - Redes privadas.

5.3. Protocolos de Rede

Para que os programas que residem em computadores diferentes possam comunicar uns com os outros, é necessário que existam um conjunto de regras que devem ser respeitadas por todos para que as comunicações possam ser efectuadas. Essas regras são designadas de **protocolos**.

Um protocolo pode ser definido como um conjunto de convensões ou regras, as quais são mutuamente aceites por duas entidades ou sistemas, e as quais regem a comunicação entre aquelas entidades ou sistemas, definindo aspectos como a sintaxe, a semântica e a temporização das trocas de dados.

A implementação de protocolos é mais facilitada se as diversas tarefas envolvidas forem individualizadas, conforme a sua natureza, em módulos separados. Desta forma, a **arquitectura** de uma rede de computadores segue, tipicamente, uma organização por **camadas** ou **níveis**, em que cada camada oferece serviços à camada imediatamente acima, escondendo–lhe os detalhes de implementação.

Alternativamente, uma arquitectura pode ser vista como uma **pilha de protocolos**, em que cada protocolo se enquadra num determinado nível dessa arquitectura.

5.3.1. O Modelo de Referência OSI

Trata-se de um modelo proposto pela ISO (*International Standards Organization*, "Organização de Padrões Internacionais"), com o objectivo de promover a padronização do domínio das comunicações por computador. A designação OSI (*Open Systems Interconnection*, "Interligação de Sistemas Abertos") refere-se à abertura dos sistemas à comunicação com outros sistemas, por oposição a soluções proprietárias.

A informação que se pretende transmitir é dividida em **pacotes**. No computador-origem, cada camada (começando na camada superior) acrescenta mais informações a cada pacote. No computador-destino, os pacotes são "desembrulhados" por cada uma das camadas. As

informações adicionais que se acrescentam aos pacote permitem verificar a sua integridade quando chegarem ao destino.

Na camada física, os pacotes são enviados através do meio físico de transmissão até ao destino. Durante o seu caminho entre as máquinas de origem e de destino, os pacotes poderão passar por computadores intermédios. Nestas máquinas, os pacotes são recebidos mas apenas desembrulhados até à camada de rede, para que a integridade da informação em trânsito possa ser verificada. Estas máquinas intermédias voltam a enviar os pacotes, pelo meio físico de transmissão até ao seu destino. O modelo OSI está representado na figura 5.5. Note-se também que uma camada de um determinado computador apenas comunica com a camada correspondente de outro computador, tal como também se encontra representado naquela figura.

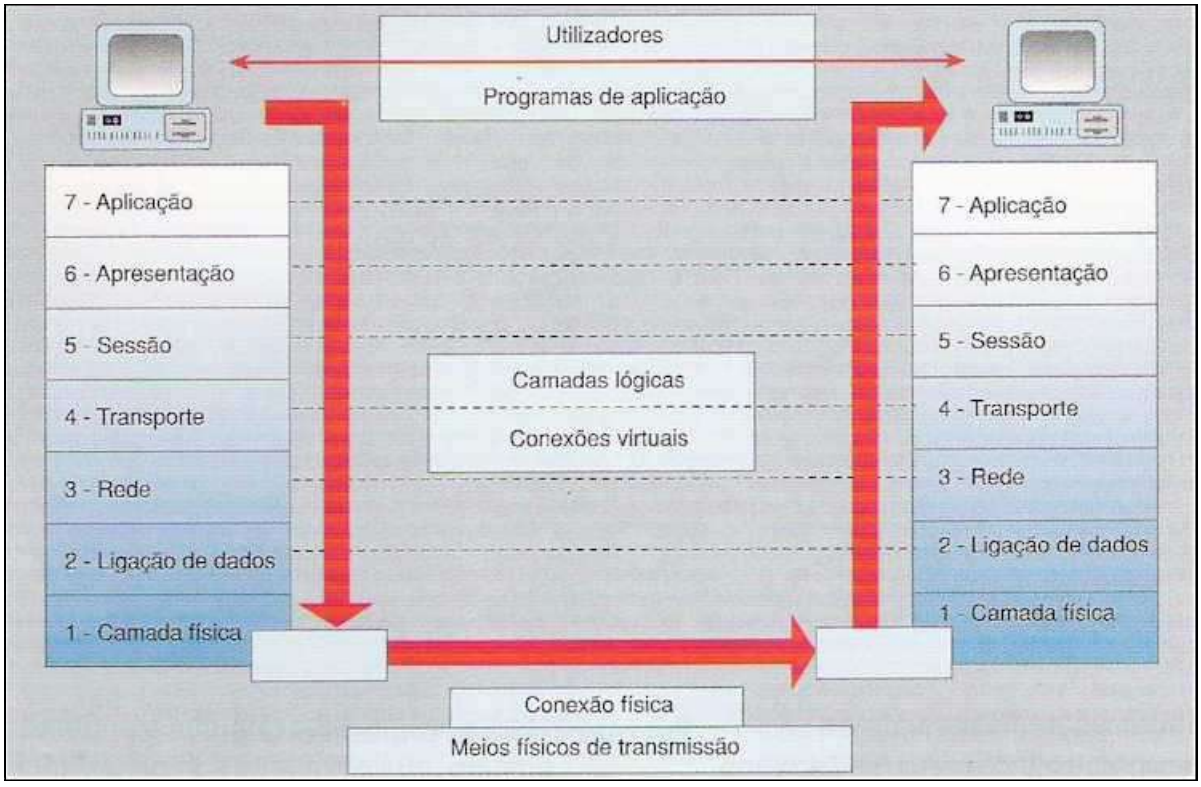


Figura 5.5: *Modelo de referência OSI. Segundo este modelo, as mensagens geradas num computador (emissor) têm de atravessar sete camadas de protocolos até chegarem ao canal de transmissão; no computador destinatário (receptor), as mensagens voltam a atravessar as mesmas sete camadas no sentido inverso, para chegarem, finalmente, ao software de aplicação do receptor. Cada camada, num computador emissor, comunica (virtualmente) com a camada que lhe corresponde no computador receptor. (Azul, 1997)*

As camadas do modelo de referência OSI são as seguintes:

- **Camada Física:**
  - Interacção directa com o meio físico de transmissão;
  - Transformação da representação lógica da informação em símbolos físicos (impulsos eléctricos, sinais ópticos, etc.).
- **Camada de Ligação de Dados:**
  - Preparação dos pacotes para serem enviados;
  - Controlo do acesso aos meios físicos de transmissão;
  - Controlo do fluxo dos pacotes entre os nós da rede;
  - Controlo de erros (retransmissão de pacotes).
- **Camada de Rede:**
  - Estabelecimento da rota mais adequada;
  - Transferência de informação entre nós da rede.
- **Camada de Transporte:**
  - Transferência de informação entre os sistemas inicial e final;
  - Serviço fiável: controlo de fluxo, de erros e de ordenação dos pacotes.
- **Camada de Sessão:**
  - Estabelece e mantém o intercâmbio de dados entre emissor e receptor durante uma sessão de comunicação.
- **Camada de Apresentação:**
  - Codifica e descodifica os dados ao nível do seu formato visual;
  - Converte formatos de sistemas diferentes.
- **Camada de Aplicação:**
  - Estabelece a interface entre o *software* de aplicação e as camadas inferiores.

5.3.2. A Pilha Protocolar TCP/IP

Trata-se de um conjunto de protocolos que resultou da investigação levada a cabo no âmbito da ARPANET, uma rede de comutação de pacotes financiada pela agência governamental norte–americana DARPA (*Defense Advanced Research Projects Agency*) em finais da década de 1960 e durante a década de 1970.

A pilha TCP/IP compreende 4 camadas, conceptualmente semelhantes às camadas correspondentes do modelo OSI (figura 5.6). Esta pilha é constituída por um conjunto bastante alargado de protocolos, posicionados ao longo das diferentes camadas desta arquitectura:

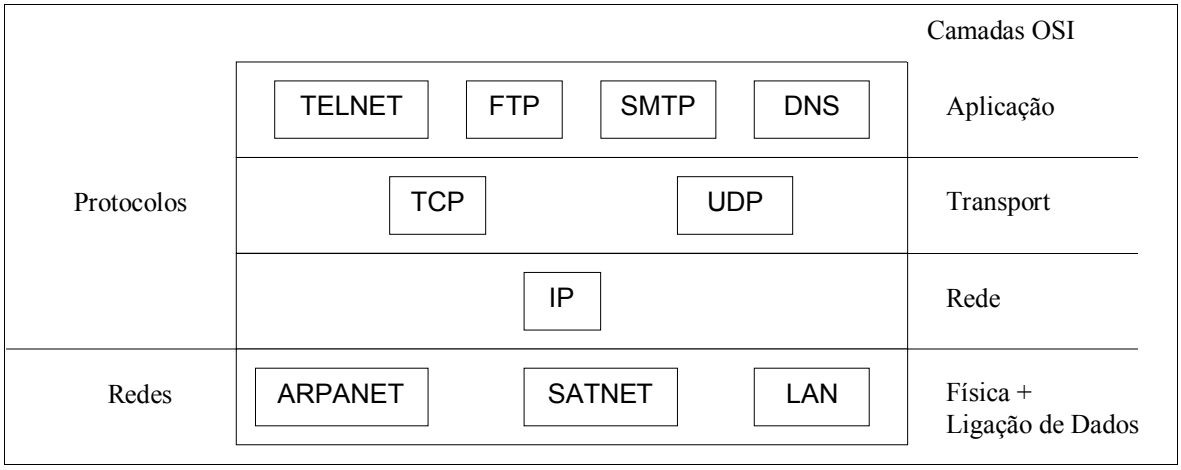


Figura 5.6: *A pilha protocolar TCP/IP. As camadas protocolares desta pilha correspondem a uma ou mais camadas do modelo de referência OSI.*

- **IP** (*"Internet Protocol"*): Oferece o serviço de rede;
- **TCP** (*"Transmission Control Protocol"*): Oferece o serviço de transporte orientado à conexão;
- **UDP** (*"User Datagram[11] Protocol"*): Oferece o serviço de transporte não orientado à conexão;
- **DNS** (*"Domain Name System"*): Oferece o serviço de resolução de nomes;
- **SMTP** (*"Simple Mail Transfer Protocol"*): Permite a transmissão de mensagens do serviço de correio electrónico;

11 Nos primórdios do desenvolvimento da Internet, "datagrama" era o nome que se dava às partições das mensagens enviadas de um computador a outro. Hoje em dia, esta designação caiu em desuso, tendo sido substituída pela palavra "pacote".

- **FTP** (*"File Transfer Protocol"*): Oferece um serviço de transferência de ficheiros;
- **Telnet**: Oferece o serviço de terminal virtual;
- **SNMP** (*"Simple Network Manegement Protocol"*): Fornece mecanismos de gestão remota de redes TCP/IP;
- **HTTP** (*"Hypertext Transer Protocol"*): Fornece um serviço de transferência de ficheiros hipertexto.

Endereços, Nomes e Domínios

Para que um computador possa ser identificado univocamente numa rede TCP/IP (por exemplo, a Internet), então tem que possuir um endereço numérico, designado por **endereço IP**. Esse endereço é formado por quatro campos, separados entre si por um ponto ("."). Por exemplo, `193.136.195.221` é o endereço IP do servidor dos alunos do IPB. Contudo, como é mais fácil para a mente humana identificar um computador por um nome do que por um número, surgiu a necessidade de converter os endereços IP em **nomes**, e vice-versa. O serviço responsável por essa conversão é o DNS, que estabelece a relação entre o nome do computador e o endereço IP correspondente. Por exemplo, o nome a que corresponde o endereço IP dado no exemplo anterior é *phobos.alunos.ipb.pt*.

Os nomes são formados por duas partes: uma que identifica o computador (no nosso exemplo, o nome do computador é *phobos*) e outra que identifica a sua localização (no nosso exemplo, a máquina phobos localiza-se no **domínio**[12] *alunos.ipb.pt*). A primeira parte dos nomes é definida livremente pelos utilizadores. A segunda parte é atribuída segundo o **Sistema de Domínio de Nomes** (*Domain Name System*, cuja sigla é DNS), criado em 1984. De acordo com o DNS, os nomes têm sempre a seguinte estrutura hierárquica (da direita para a esquerda): *nome.subdomínio(s).domínio*

O domínio de topo (o mais à direita) é normalmente formado pelos códigos de nome de país (de acordo com o padrão ISO 3166), excepto nos EUA[13], onde existem vários domínios de topo, formados por três digitos (**gov**, **mil**, **com**, **edu**, **org**, **net**, etc.[14]). De acordo com o padrão ISO 3166, o domínio de topo de todos os nomes dos computadores portugueses directamente ligados à Internet é **pt**.

12 Em termos gerais, pode-se considerar que um domínio é uma rede usada por uma organização.

13 Na realidade, os EUA possuem um domínio de topo definido pelo padrão ISO 3166: **us**. Na prática, porém, esse domínio não é muito utilizado.

Abaixo do domínio de topo, podem existir um ou vários subdomínios, que representam organizações, departamentos de organizações, etc. Por exemplo, todos os computadores do Instituto Politécnico de Bragança terminam o seu nome em **.ipb.pt**, os da Escola Superior de Tecnologia e de Gestão em **.estig.ipb.pt**, etc.

14 Estes domínios de topo são destinados (pelo menos, em teoria) a computadores pertencentes a redes de organizações governamentais, militares, comerciais, educativas, organizações genéricas e organizações relacionadas com a Internet, respectivamente.

6 – Tecnologias Multimédia

6.1. Introdução à Multimédia[15]

A palavra "multimédia" designa as várias formas de apresentação da informação: texto, imagem, animação, vídeo, som, gráficos, realidade virtual, etc. Em informática, o conceito de **multimédia** representa os sistemas de tratamento de informação e as aplicações que manipulam as várias formas de apresentação dessa informação.

As **aplicações multimédia** são desenvolvidas para cada vez mais finalidades, desde as actividades recreativas (jogos), passando pelas actividades de carácter educativo (programas didácticos, livros ou enciclopédias multimédia, etc.), até às mais avançadas utilizações das novas tecnologias (simulação de voo, produção pós-vídeo, etc.).

Os principais campos de utilização de multimédia são os seguintes:

- Criação de interfaces agradáveis entre o utilizador e a informação armazenada em computador;
- Divulgação de informação: livros, revistas, enciclopédias, catálogos, demonstrações, etc.;
- Aplicações destinadas à aprendizagem ou ao treino assistidos por computador;
- Concepção e apresentação de protótipos de mecanismos tecnológicos, obras de engenharia ou de arquitectura, etc.;
- Quiosques de informação, ou seja, dispositivos de apresentação de informação, que interagem com o utilizador, normalmente, através de ecrã de toque (*"touch screen"*).

Uma **aplicação multimédia** diz-se **interactiva** quando permite a intervenção do utilizador no controlo do desenrolar das operações. Isso é feito, normalmente, através da inserção de elementos gráficos especiais no ecrã da aplicação (palavras assinaladas a uma cor

15 Este subcapítulo é baseado em Azul (1997), volume 2, páginas 393 a 395.

diferente, botões, ícones ou zonas especiais), a partir dos quais o utilizador pode, através de um clique do rato ou outras formas de introdução de dados, aceder a caixas com informação suplementar, a outras secções do documento ou a outros documentos. A esses elementos gráficos especiais dá-se o nome de **hiperligação**.

Este modo de funcionamento deriva do conceito de **hipertexto**, já existente nos módulos de ajuda de muitas aplicações de *software* e que passou a constituir um dos pilares do funcionamento do sistema WWW (*"World Wide Web"*) da Internet.

Quando se combina o estilo de um documento em hipertexto com multimédia, passamos a ter aquilo que alguns autores designam por **hipermédia**.

O que temos realmente no sistema WWW da Internet é um gigantesco e sofisticado sistema de hipermédia: as páginas da *Web* apresentam-nos documentos de hipertexto, incluindo imagens fixas e animadas, assim como, frequentemente, som e vídeo.

Actualmente, a maioria das aplicações multimédia são interactivas e funcionam em estilo de hipertexto ou de hipermédia, pelo que pode-se afirmar que o conceito de multimédia engloba também os de interactividade e hipermédia.

Em geral, uma aplicação multimédia requer grande quantidade de espaço de armazenamento. Desta forma, o CD-ROM tornou-se no meio favorito para o armazenamento e a distribuição deste tipo de aplicações, porque permite guardar uma grande quantidade de informação (entre 650 e 700 MB). No entanto, o desenvolvimento crescente das funcionalidades oferecidas pelas aplicações multimédia mais comuns começa a tornar insuficiente o espaço de armazenamento disponível num CD-ROM vulgar. Por isso, os fabricantes de aplicações multimédia começam a utilizar o DVD como o suporte de eleição para o armazenamento e a distribuição dos seus produtos, já que os DVDs podem oferecer espaços de armazenamento com até cerca de 17 GB de capacidade máxima.

6.2. A Internet

6.2.1. A Evolução da Internet

A Internet pode ser definida como um conjunto de redes de computadores interligadas à escala mundial. Na Internet estão interligados vários tipos de redes, quer quanto à cobertura geográfica (redes locais, redes de longa distância, etc.), quer quanto aos equipamentos que as

compõem e à tecnologia utilizada. Para que isso seja possível, é necessário que existam regras comuns, designadas protocolos.

O conjunto (ou família) de protocolos utilizados na Internet é o **TCP/IP** (*Transmission Control Protocol/Internet Protocol*). Tal como já foi discutido no capítulo anterior (cap. 5 – Redes de Computadores), a família TCP/IP é composta por diversos protocolos, dos quais se destacam (para além dos dois – TCP e IP – que dão o nome à família) o DNS, o SMTP, FTP, Telnet, HTTP e NNTP.

A primeira antepassada da Internet nasceu, em 1969, de um projecto do Departmento de Defesa dos EUA. Essa rede chamava-se ARPANET (*Advanced Research Projects Agency Network*) e tinha como objectivo a interligação de computadores utilizados em centros de investigação com fins militares. Com esse projecto, pretendia-se desenvolver uma plataforma informática distribuída onde a informação pudesse ser armazenada em segurança (de forma redundante), e que pudesse continuar a funcionar mesmo se alguma parte da rede estivesse impossibilitada de trabalhar (por exemplo, em caso de um ataque nuclear; note-se que, naquela época, se estava em plena Guerra Fria entre os EUA e a URSS).

Foi no início da década de 1980, mais precisamente em 1983, com a adopção dos protocolos TCP/IP por parte da ARPANET[16], e com a criação da rede científica CSNET (*Computer Science Network*) e a sua ligação à ARPANET, que surgiu a verdadeira Internet.

Ao longo da década de 1980, o ritmo de crescimento da Internet continuou a aumentar, tornando necessária a existência e funcionamento de **estruturas de coordenação e de cooperação** entre o cada vez maior número de redes e operadores que a integravam. Assim, logo em 1983, foi criado o *Internet Activities Board* (IAB, agora designado *Internet Architecture Board*[17]), dentro do qual se criariam, em 1989, o *Internet Engineering Task Force* (IETF[18]) e o *Internet Research Task Force* (IRTF[19]).

16 Nesse mesmo ano, a componente estritamente militar da ARPANET separou-se e formou a rede MILNET (*Military Network*). Assim, a partir dessa altura, a ARPANET tornou-se uma rede essencialmente civil.

17 O IAB controla o desenvolvimento de padrões e protocolos para a Internet e actua como interface entre as várias entidades de desenvolvimento de padrões. URL: http://www.iab.org/.

18 O IETF procura soluções para problemas técnicos e operacionais da Internet e desenvolve padrões e protocolos. Os seus grupos de trabalho são constituídos unicamente por voluntários. URL: http://www.ietf.org/.

19 O IRTF promove a pesquisa relacionada com a evolução futura da Internet, através da criação de pequenos Grupos de Pesquisa (*Research Groups*) dedicados a projectos de longo prazo em áreas relativas a protocolos,

Em 1989, a Internet era constituída por mais de 100.000 computadores (*"hosts"*) ligados em rede. Mas é durante a primeira metade da década de 1990 que se regista a grande popularização da Internet, graças ao desenvolvimento de novos serviços mais amigáveis e eficientes (como o *Gopher*[20] e a WWW).

No início de 1996, estimava-se que a Internet devia contar com cerca de 9,5 milhões de computadores e mais de 30 milhões de utilizadores (39 milhões de utilizadores de correio electrónico e 26 milhões de utilizadores do conjunto de serviços Internet, segundo o *Third MIDS Internet Demographic Survey*, de Outubro de 1995).

6.2.2. Os Serviços da Internet

WWW – World Wide Web

A World Wide Web nasceu em 1989 no CERN (Centro Europeu de Física de Partículas), tendo como seu mentor Tim Berners-Lee, investigador daquela instituição. Surge inicialmente como uma forma mais fácil e visualmente mais atractiva de disponibilização de informação na Internet. Destacava-se de outros serviços (nomeadamente o *Gopher*), pela utilização de interfaces gráficos, tornando-se assim mais atraente, funcional e simples de utilizar.

A WWW é um sistema de informação hipertexto e hipermédia, distribuído, cooperativo e heterogéneo. O hipertexto oferece ligações de palavras-chave ou frases, com outras partes do mesmo texto, ou com documentos exteriores. Uma vez que os sistemas actuais também permitem manipular imagens, gráficos, sons e outros tipos de dados, podem-se estabelecer ligações hipertextuais entre esses diferentes formatos de informação, o que nos leva ao conceito de hipermédia, já discutido neste capítulo.

Além dos meios de comunicação utilizados pela Internet, a WWW assenta em quatro instrumentos fundamentais para o seu funcionamento:

- Protocolo HTTP (*Hypertext Transfer Protocol*);

aplicações, arquitecturas e tecnologias utilizadas na Internet. URL: http://www.irtf.org/.

20 O *Gopher* é um serviço da Internet, anterior à WWW, cujo objectivo é o de organizar e visualizar ficheiros alojados em servidores ligados à Internet. A informação apresentada é organizada de forma hierárquica e sob a forma de menus. Este serviço contribuiu em muito para o aumento da facilidade de utilização da Internet. Contudo, tornou-se completamente obsoleto perante a maior flexibilidade dos documentos hipertexto sobre os quais a WWW se baseia. Foi desenvolvido pela Universidade de Minnesota, EUA.

- Linguagem HTML (*Hypertext Markup Language*);
- *Software* para o servidor WWW (por exemplo, *Apache* e *Microsoft Internet Information Server*, etc.);
- *Software* cliente para a visualização de documentos, designado por navegador ou *"browser"* (por exemplo, *Microsoft Internet Explorer*, *Netscape Navigator*, *Mozilla*, *Opera*, etc.).

O protocolo HTTP define as regras de transferência dos ficheiros HTML entre o servidor WWW onde esses ficheiros estão armazenadas e o *browser* cliente (localizado numa máquina "cliente") que requisita esses ficheiros. Este protocolo utiliza o conceito de URLs (*"Uniform Resource Locator*", Localizador Uniforme de Recursos) para localizar recursos e informação na Internet. Um **URL** é uma espécie de identificador único, em toda a Internet, de um determinado recurso; a sua sintaxe segue o seguinte esquema: *protocolo://máquina [/caminho/ficheiro]*. As várias componentes de um URL podem ser assim descritas:

- Protocolo: indica qual o protocolo que será utilizado para aceder o recurso; pode ser um dos seguintes: http, https, ftp, news, gopher, entre outros;
- Máquina: nome ou endereço IP do computador onde se encontra armazenado o recurso[21];
- Caminho: (opcional) caminho completo das directorias, existentes na máquina, que devem ser percorridas até se chegar ao recurso;
- Ficheiro: (opcional) nome do ficheiro onde se encontra o recurso pretendido[22].

Um exemplo de URL é: http://www.ipb.pt/~reis.quarteu/isi2004.htm. O protocolo é o *http*, a máquina é designada por *www.ipb.pt*, o caminho é *~reis.quarteu*, e o ficheiro tem o nome de *isi2004.htm*.

21 Num URL, a parte correspondente à máquina também pode conter um número que é designado "porta" (isto é, um ponto lógico de ligação), e que representa um programa servidor especial existente na máquina, com o qual o cliente deseja interagir.

22 A parte correspondente ao ficheiro de um URL também pode conter mais dados que permitirão definir melhor o recurso que se quer aceder. Em geral, esses dados são listas de pares formados por atributos e os seus valores, os quais são utilizados para aceder a recursos gerados dinamicamente pelo servidor *web*. A sua discussão ultrapassamo âmbito desta disciplina.

FTP – Transferência de Ficheiros

O **FTP** (*File Transfer Protocol*, "Protocolo de Transferência de Ficheiros") é um protocolo que permite a transferência de ficheiros entre computadores numa rede TCP/IP. Ao mesmo tempo, é um dos serviços mais utilizados na Internet, permitindo o acesso a computadores e aos seus ficheiros.

Para se estabelecer a ligação a um computador através de FTP, é necessário utilizar um programa cliente de FTP (exemplos: *CuteFTP*, *WS_FTP*, etc.) para estabelecer uma ligação com um computador servidor de FTP. Para que a ligação FTP seja completamente estabelecida, o utilizador deverá indicar a sua identificação perante o sistema (*"login"*) e autenticá-la mediante uma palavra-passe.

A existência de servidores de ficheiros de acesso público depende da existência de um mecanismo que permita o acesso a qualquer pessoa, dispensando uma identificação específica para cada potencial utilizador. Neste contexto, criou-se o serviço **FTP anónimo**, baseado numa identificação anónima[23], que pode ser utilizada por todos, e numa autenticação baseada no endereço de correio electrónico do utilizador. É claro que um utilizador que se identifique como "anónimo" não terá os mesmo privilégios no acesso aos recursos disponibilizados pelo servidor FTP do que outros utilizadores "conhecidos" pela máquina.

Correio Electrónico

O correio electrónico é uma aplicação distribuída que permite a troca de mensagens entre pessoas, através de uma rede de computadores. Basicamente, funciona de forma análoga aos serviços postais, com a diferença da mensagem ser transportada entre o computador do remetente e o computador do destinatário pela infraestrutura de rede.

As vantagens do correio electrónico em relação ao correio tradicional são a sua velocidade e a flexibilidade da sua utilização: é possível enviar por correio electrónico tudo o que possa ser interpretado por um computador.

Para se poder receber ou enviar correio, é necessário ter um endereço de correio electrónico. Este endereço tem o seguinte formato: *identificador@máquina*. A 1.ª parte é a identificação do utilizador, ou *login*, correspondente a uma área de trabalho do utilizador dentro de uma máquina ligada à Internet; a 2.ª parte é a identificação da máquina onde ficará armazenado o

23 Em geral, a identificação anónima faz-se introduzindo a palavra *"anonymous"* como o identificador (*login*) do utilizador.

correio enviado ou recebido por esse utilizador. A separação entre estes dois componentes é feita pelo caracter @ (lê-se "arroba", ou *"at"* em inglês). Um exemplo de um endereço de correio electrónico é reis.quarteu@ipb.pt.

Com a finalidade de garantir a privacidade das mensagens, o serviço de correio electrónico pode ser protegido pela existência de uma palavra-passe associada a cada área de trabalho.

Existem vários protocolos que se encarregam de efectuar a transmissão de mensagens de correio electrónico. São eles:

- **SMTP** (*Simple Mail Transfer Protocol*): Efectua a interacção entre os servidores de correio electrónico. Este protocolo é o responsável pelo envio das mensagens de correio electrónico desde as suas origens até aos seus destinos.
- **POP3** (*Post Office Protocol,* versão 3): Permite exportar o correio electrónico armazenado em servidores SMTP para uma máquina cliente, na qual o correio será lido; actua entre o servidor de correio e o cliente. Este protocolo exige que todo o correio electrónico de um utilizador seja descarregado para a máquina cliente antes que as mensagens possam er lidas.
- **IMAP** (*Internet Message Access Protocol*): pretende ser o substituto do POP3, incorporando novas funções, tais como:
  - Ler o cabeçalho ou o remetente de uma mensagem antes de decidir se essa mensagem será descarregada para a máquina cliente;
  - Criar, alterar e apagar pastas no servidor IMAP;
  - Mover ou copiar mensagens entre as pastas de um servidor IMAP;
  - Apagar mensagens no próprio servidor, etc.

USENET – Grupos de Notícias

Os **grupos de notícias**, também conhecidos como fóruns de discussão *USENET News*, ou *newsgroups*, são grupos de mensagens públicas, organizadas hierarquicamente por temas, que circulam na Internet, podendo ser acedidas por qualquer utilizador desta rede.

No topo da hierarquia dos grupos de notícias estão os grandes temas, como **comp.*** (grupos relacionados com computadores), **sci.*** (grupos relacionados com ciência e investigação),

soc.* (grupos relacionados com cultura e questões sociais), **rec.*** (grupos relacionados com actividades recreativas), **alt.*** (grupos relacionados com assuntos diversos), etc. Dentro de cada um destes temas, existe uma grande variedade de subtemas, formando no seu conjunto uma hierarquia em árvore, que facilita a escolha e localização dos temas que nos podem interessar.

Deve-se referir ainda a existência de uma hierarquia **pt.***, que contém grupos de discussão relacionados com Portugal. Existem nesta hierarquia diversos grupos, entre os quais o *pt.geral*, *pt.politica*, *pt.mercado*, *pt.news*, *pt.ensino*, etc.

Da mesma forma que a *WWW* utiliza o protocolo HTTP, os grupos de notícias utilizam o protocolo **NNTP** (*Network News Transfer Protocol*), que tem como função o transporte das notícias através da Internet.

O funcionamento da USENET pode ser assim descrito:

- As notícias que circulam na USENET são armazenadas em servidores de notícias, os quais comunicam entre si para proceder à divulgação das mesmas.
- Esta comunicação é mantida com um conjunto de servidores vizinhos, que procedem à divulgação das mensagens usando um algoritmo de inundação (*"flooding"* em inglês; um servidor de notícias envia as novas mensagens que devem ser divulgadas para todos os outros servidores de notícias directamente ligados àquele servidor, com a excepção do servidor de onde foram recebidas essas novas notícias).
- São definidos em cada servidor quais os grupos de notícias que podem ser trocados com cada vizinho, assim como quais os grupos que podem ser recebidos.
- As mensagens apresentam um formato idêntico ao formato das mensagens de correio electrónico, à excepção do destinatário. No correio electrónico, as mensagens são dirigidas normalmente a um conjunto de um ou mais utilizadores singulares, enquanto na USENET são dirigidas a um grupo de notícias.

Telnet

O serviço *telnet* oferece um serviço de terminal virtual. Permite que através de um programa instalado numa máquina cliente, qualquer utilizador possa usar os recursos de um servidor, tal como se estivesse na sua própria máquina.

6.3. Pesquisa na World Wide Web

Existem dezenas de ferramentas de pesquisa na *World Wide Web*. Esses índices, catálogos e bases de dados são constituídos de duas formas básicas:

1. **Robots**, que recolhem e indexam automaticamente o texto integral (exemplos: Alta Vista, HotBot, InfoSeek, Lycos, OpenText, WebCrawler, etc.) ou apenas partes (Inktomi, WWW Worm, etc.) das páginas da *Web*. O funcionamento dos robots apenas é possível porque a *Web* é baseada em documentos hipertexto que se ligam uns aos outros. Este facto permite que, a partir de um único documento com hiperligações para outros documentos, se possa "partir à descoberta" de muitos outros documentos.

2. Construção e manutenção de **índices** ou **directórios de classificação** por intervenção humana (exemplos: SAPO, Yahoo, etc.), onde são pesquisáveis os títulos e uma breve descrição dos documentos indexados, e o esquema de classificação ou organização dos documentos.

Os termos a utilizar nas pesquisas devem ter em conta estes dois tipos de instrumentos. Assim, e de um modo geral, na pesquisa de bases de dados geradas a partir de robots (especialmente se esses robots indexam o texto integral dos documentos), pode-se utilizar termos mais específicos e em diferentes línguas (de acordo com as utilizadas nos documentos que procuramos). Por outro lado, na pesquisa de índices ou directórios, deve-se usar termos mais genéricos, que correspondam aos temas e subtemas da classificação utilizada, ou então a palavras conhecidas dos títulos dos documentos procurados. Finalmente, podem ainda ser utilizadas algumas ferramentas que interrogam simultaneamente diversos índices e bases de dados.

Alguns portais portugueses onde é possível pesquisar documentos *Web* são os seguintes:

- http://www.sapo.pt/
- http://www.clix.pt/
- http://www.aeiou.pt/
- http://www.netindex.pt/
- http://www.iol.pt/

Alguns portais estrangeiros onde é possível pesquisar documentos *Web* são os seguintes:

- http://www.google.com/
- http://www.yahoo.com/
- http://www.altavista.com/
- http://www.lycos.com/
- http://www.hotbot.com/
- http://www.excite.com/
- http://dmoz.org/
- http://www.searcheurope.com/
- http://www.webcrawler.com/

Alguns portais onde é possível efectuar pesquisas múltiplas sobre bases de dados e índices de documentos *Web* são os seguintes:

- http://www.search.com/
- http://www.globito.com/

6.4. Introdução à Linguagem HTML

A linguagem HTML é utilizada para a descrição de documentos hipertexto em vários sistemas, nomeadamente na WWW, o sistema de hipertexto mais popular na Internet. É constituída por uma conjunto de marcas (*tags*), que permitem a formatação de texto, imagens, som e animações em navegadores ou *browsers*. A actual versão do HTML recomendada pelo W3C (*World Wide Web Consortium*[24]) é o HTML 4.

Um pequeno guia de iniciação ao HTML pode ser encontrado neste endereço: http://www.w3.org/MarkUp/Guide (em inglês).

24 O W3C é a organização responsável por estabelecer os padrões das tecnologias que se apoiam na WWW. URL: http://www.w3.org/.

6.4.1. Principais Marcas do HTML 4

Neste subcapítulo, apresentar-se-á uma pequena lista das marcas mais comuns da linguagem HTML. Algumas destas marcas possuem mais atributos cujos valores podem também ser definidos pelo autor das páginas HTML. Apenas serão apresentados os atributos mais relevantes para cada marca. Por fim, este subcapítulo não pretende substituir nenhum livro sobre a linguagem HTML: apenas visa ser uma espécie de guia de referência rápida para os alunos desta disciplina. Maiores detalhes sobre a utilização das marcas aqui listadas deverão ser obtidos em livros sobre o tema, alguns dos quais aparecem referenciados na bibliografia desta sebenta.

Dentro de uma marca, a ordem dos seus atributos é irrelevante.

Por fim, deve-se referir como se faz a inserção de comentários num documento HTML. Um comentário (texto que não deverá ser visualizado num *browser*, apenas deverá aparecer no código HTML do documento) deverá ser colocado entre as marcas **<!--** e **-->**. Um exemplo da sua utilização é o que se segue:

```
<!-- Isto é um comentário. -->
```

HTML

Função: Identifica um documento HTML.

Sintaxe: `<HTML> ... </HTML>`

Exemplo:

```
<HTML>
    <BODY>
          Isto é um exemplo.
    </BODY>
</HTML>
```

HEAD

Função: Identifica o cabeçalho do documento. Deve vir entre as marcas <HTML> e </HTML>.

Sintaxe: `<HEAD> ... </HEAD>`

Exemplo:

```
<HTML>
    <HEAD> ... </HEAD>
    ...
</HTML>
```

TITLE

Função: Define o título do documento, o qual aparece na barra de título da janela do *browser* quando a página estiver a ser visualizada.

Sintaxe: `<TITLE> ... </TITLE>`

Exemplo:

```
<HTML>
    <HEAD>
          <TITLE>Título do documento</TITLE>
    </HEAD>
    ...
</HTML>
```

BODY

Função: Identifica a parte principal (o corpo) do documento. Deve vir entre as marcas <HTML> e </HTML>.

Sintaxe: `<BODY ...> ... </BODY>`

Atributos mais importantes:

- **BACKGROUND**: Define o endereço (URL) de uma imagem que será usada como imagem de fundo.
- **BGCOLOR**: Define uma cor de fundo.
- **TEXT**: Define a cor do texto.

Exemplo:

```
<BODY BGCOLOR="white" TEXT="black">
    O documento terá um fundo de cor branca, sob um texto de cor preta.
</BODY>
```

H1 ... H6

Função: Definem títulos e subtítulos, com diferentes tamanhos de letra.

Sintaxe: `<H1 ...> ... </H1>`

Atributos mais importantes:

- **ALIGN**: Define o alinhamento do texto. Pode assumir 4 valores:
  - **left**: alinhamento à esquerda;
  - **center**: alinhamento ao centro;
  - **right**: alinhamento à direita;
  - **justify**: alinhamento justificado (à esquerda e à direita).

Exemplo:

```
<H1 ALIGN="left">Título de nível 1</H1>
<H2 ALIGN="right">Título de nível 2</H2>
<H3 ALIGN="center">Título de nível 3</H3>
<H4 ALIGN="justify">Título de nível 4</H4>
<H5 ALIGN="left">Título de nível 5</H5>
<H6 ALIGN="right">Título de nível 6</H6>
```

P

Função: Define um parágrafo.

Sintaxe: `<P ...> ... </P>`

Atributos mais importantes:

- **ALIGN**: Define o alinhamento do texto. Pode assumir os mesmos valores do atributo **ALIGN** das marcas **H1** a **H6**.

Exemplo:

```
<P ALIGN="justify">Isto é um parágrafo.</P>
```

B, I e U

Função: Apresenta o texto envolvido com os estilos negrito, itálico ou sublinhado, respectivamente.

Sintaxe: `<B> ... </B>`
`<I> ... </I>`
`<U> ... </U>`

Exemplo:

```
<B>Texto em negrito</B>
<I>Texto em itálico</I>
<U>Texto sublinhado</U>
```

BR

Função: Insere uma quebra de linha.

Sintaxe: `<BR>`

Exemplo:

```
<P ALIGN="left">
    Isto é uma linha.
    <BR>
    Isto é outra linha, dentro do mesmo parágrafo.
</P>
```

HR

Função: Insere uma linha horizontal separadora.

Sintaxe: `<HR ...>`

Atributos mais importantes:

- **ALIGN**: Define o alinhamento da linha em relação ao interior da janela do *browser*. Pode assumir 3 valores:
  - **left**: alinhamento à esquerda;
  - **center**: alinhamento ao centro;

  - **right**: alinhamento à direita.

- **NOSHADE**: Se definido, indica que a linha não deverá apresentar o efeito de sombra.
- **SIZE**: Define a largura da linha. Expresso em pixeis.
- **WIDTH**: Define o comprimento da linha. Expresso em pixeis ou como uma percentagem.

Exemplo:

```
<P ALIGN="left">
    Isto é uma linha.
    <HR NOSHADE
        ALIGN="center"
        SIZE="8"
        WIDTH="50%">
    Isto é outra linha, dentro do mesmo parágrafo, e separada por uma
    linha horizontal.
</P>
```

A

Função:

1. Define uma âncora, isto é, um ponto de um documento hipertexto que pode ser referenciado por outros documentos (por outras palavras menos técnicas, para onde se pode "saltar" a partir de outros documentos).
2. Define uma hiperligação, isto é, uma área do documento que permite aceder um outro documento, ou um outro ponto dentro do mesmo documento, sempre que o utilizador clicar o botão esquerdo do rato dentro dessa área.

Sintaxe: `<A ...> ... </A>`

Atributos mais importantes:

- **HREF**: Define o URL que poderá ser acedido através da hiperligação.
- **NAME**: Define o nome da âncora.

Exemplo:

```
<A NAME="frio">Um dia muito frio</A>
<!-- Âncora no texto "Um dia muito frio" -->
...
<A HREF="#frio">Salta para o dia muito frio.</A>
<!-- Hiperligação para a âncora anterior -->
...
<A HREF="http://www.estig.ipb.pt/">Salta para a ESTiG.</A>
<!-- Hiperligação para um documento externo -->
```

IMG

Função: Insere uma imagem.

Sintaxe: `<IMG ...>`

Atributos mais importantes:

- **ALIGN**: Define o alinhamento da imagem. Pode assumir 5 valores:
  - **bottom**: alinha a parte inferior da imagem com a linha de texto próxima à imagem;
  - **midde**: alinha verticalmente o centro da imagem com a linha de texto próxima à imagem;
  - **top**: alinha a parte superior da imagem com a linha de texto próxima à imagem;
  - **left**: alinhamento à esquerda;
  - **right**: alinhamento à direita.
- **ALT**: Texto alternativo à imagem. Em geral, este texto é visualizado pelo navegador quando o cursor do rato permanece imóvel sobre a imagem durante um determinado intervalo de tempo (em geral, com a duração de uns poucos segundos).
- **BORDER**: Define a largura da linha que envolve a imagem. Expresso em pixeis. Se este valor for igual a zero, então a figura não terá nenhum bordo à sua volta.
- **HEIGHT**: Define a altura da imagem. Expresso em pixeis.

- **SRC**: Define o endereço (URL) da imagem que se quer inserir no documento.
- **WIDTH**: Define a largura da imagem. Expresso em pixeis.

Exemplo:

```
<!-- Imagem associada a uma hiperligação -->
<A HREF="http://www.estig.ip.pt/">
    <IMG SRC="estig.gif"
         ALIGN="right"
         BORDER="1"
         ALT="Logótipo da ESTiG">
</A>
```

OL

Função: Define uma lista numerada e ordenada. Cada item da lista deverá ser definido através da marca <LI>.

Sintaxe:

```
<OL ...>
    <!-- Itens da lista -->
</OL>
```

Atributos mais importantes:

- **START**: Indica qual o valor inicial da lista.
- **TYPE**: Define o tipo de lista numerada. Pode assumir 5 valores:
  - **A**: define uma lista numerada com letras maiúsculas;
  - **a**: define uma lista numerada com letras minúsculas;
  - **I**: define uma lista numerada com números romanos maiúsculos;
  - **i**: define uma lista numerada com números romanos minúsculos;
  - **1**: define uma lista numerada com números normais.

Exemplo:

```
<P ALIGN="left">Exemplo de uma lista numerada com números romanos
minúsculos.</P>
```

```
<OL START="1"
    TYPE="i">
      <!-- Itens -->
</OL>
```

UL

Função: Define uma lista não numerada. Cada item da lista deverá ser definido através da marca <LI>.

Sintaxe: `<UL ...>`

```
          <!-- Itens da lista -->
      </UL>
```

Atributos mais importantes:

- **TYPE**: Define o tipo de marcador dos itens da lista não numerada. Pode assumir 3 valores:
  - **circle**: define um círculo preenchido como marcador;
  - **disc**: define um círculo não preenchido como marcador;
  - **square**: define um quadrado não preenchido como marcador.

Exemplo:

```
<P ALIGN="left">Exemplo de uma lista não numerada com quadrados como
marcadores.</P>
<UL TYPE="square">
      <!-- Itens -->
</UL>
```

LI

Função: Define um item de uma lista numerada ou não numerada.

Sintaxe: `<LI> ... </LI>`

Exemplo:

```
<P ALIGN="left">Exemplo de uma lista numerada com 3 itens.</P>
```

```
<OL>
      <LI>1.º item</LI>
      <LI>2.º item</LI>
      <LI>3.º item</LI>
</OL>
```

TABLE

Função: Define uma tabela. Cada linha da tabela deverá ser definida através da marca <TR>. Cada célula de uma linha deverá ser definida através da marca <TD>.

Sintaxe:
```
<TABLE ...>
    <!-- Linhas da tabela -->
        <!-- Células da linha -->
    <!-- Fim da linha -->
</TABLE>
```

Atributos mais importantes:

- **ALIGN**: Define o alinhamento da tabela em relação à área interna de visualização do *browser*. Pode assumir os mesmos valores do atributo **ALIGN** da marca **HR**.
- **BGCOLOR**: Define a cor de fundo da tabela.
- **BORDER**: Define a largura das linhas que separam as células da tabela. Expresso em pixeis. Se este valor for igual a zero, então a tabela não terá nenhuma linha a separar as suas células.
- **CELLPADDING**: Define a distância entre o conteúdo de uma célula e os seus limites. Expresso em pixeis.
- **CELLSPACING**: Define a distância entre os limites de duas células adjacentes. Expresso em pixeis.
- **WIDTH**: Define a largura da tabela. Expresso em pixeis ou em percentagem da largura da área interna de visualização do *browser*.

Exemplo:

```
<!-- Ver exemplo da marca TD -->
```

TR

Função: Define uma linha de uma tabela. Cada célula de uma linha deverá ser definida através da marca <TD>.

Sintaxe:
```
<TABLE ...>
    <TR ...>    <!-- 1.ª linha da tabela -->
        <!-- Células da 1.ª linha -->
    </TR>
    <!-- Mais linhas da tabela -->
</TABLE>
```

Atributos mais importantes:

- **ALIGN**: Define o alinhamento horizontal do conteúdo das células da linha da tabela. Pode assumir os mesmos valores do atributo **ALIGN** das marcas **H1** a **H6**.
- **VALIGN**: Define o alinhamento vertical do conteúdo das células da linha da tabela. Pode assumir 3 valores[25]:
  - **top**: alinhamento em cima;
  - **middle**: alinhamento ao meio;
  - **bottom**: alinhamento em baixo.

Exemplo:

```
<!-- Ver exemplo da marca TD -->
```

TD

Função: Define uma célula de uma tabela.

Sintaxe:
```
<TABLE ...>
    <TR ...>    <!-- 1.ª linha da tabela -->
        <TD ...> ... </TD>
        <!-- Mais células da 1.ª linha da tabela -->
    </TR>
    <!-- Mais linhas da tabela -->
```

25 Na realidade, o atributo **valign** da marca **TR** pode assumir um 4.º valor, **baseline**, o qual não será discutido nesta sebenta.

```
</TABLE>
```

Atributos mais importantes:

- **ALIGN**: Define o alinhamento horizontal do conteúdo da célula. Pode assumir os mesmos valores que o atrinuto **ALIGN** da marca **TR**.
- **BGCOLOR**: Define a cor de fundo da célula.
- **COLSPAN**: Indica o número de colunas da tabela que a célula deverá ocupar.
- **ROWSPAN**: Indica o número de linhas da tabela que a célula deverá ocupar.
- **VALIGN**: Define o alinhamento vertical do conteúdo da célula. Pode assumir os mesmos valores que o atrinuto **VALIGN** da marca **TR**.
- **WIDTH**: Define a largura da célula. Expresso em pixeis ou em percentagem da largura da tabela.

Exemplo:

```
<TABLE ALIGN="center" BORDER=”1”>
    <TR ALIGN="center">
          <TD>Linha 1, Coluna 1</TD>
          <TD>Linha 1, Coluna 2</TD>
          <TD>Linha 1, Coluna 3</TD>
    </TR>
    <TR ALIGN="center">
          <TD>Linha 2, Coluna 1</TD>
          <TD COLSPAN="2">Linha 2, Colunas 2 e 3</TD>
    </TR>
</TABLE>
```

TH

Função: Define uma célula especial de uma tabela. Essa célula pode actuar visualmente como um título de uma coluna ou de uma linha, porque o seu conteúdo aparece automaticamente centrado e ao texto é aplicado o estilo negrito.

Sintaxe:
```
<TABLE ...>
          <TR ...>    <!-- 1.ª linha da tabela -->
                <TH ...> ... </TH>
```

```
            <!-- Estas células serão os títulos das colunas -->
            <!-- Podem haver tantas linhas ou tants colunas de
            títulos quanto se queira -->
        </TR>
        <!-- Mais linhas da tabela -->
    </TABLE>
```

Atributos mais importantes: Os mesmos da marca **TD**.

Exemplo:

```
<TABLE BORDER="1" WIDTH="50%">
    <TR>
        <TH>Título 1</TH>
        <TH>Título 2</TH>
        <TH>Título 3</TH>
    </TR>
    <TR>
        <TD>Linha 1, Coluna 1</TD>
        <TD>Linha 1, Coluna 2</TD>
        <TD>Linha 1, Coluna 3</TD>
    </TR>
</TABLE>
```

6.5. Técnicas de Construção e de Gestão de Sítios WWW

O poder de comunicação e a facilidade de utilização da WWW são as principais razões da sua grande divulgação. Como consequência, hoje em dia praticamente todas as empresas estão a recorrer a este meio de publicidade e informação. Por isso, a construção de sítios na Internet é um mercado muito florescente e que movimenta muito dinheiro.

Para construir páginas HTML, deve-se ter em conta alguns aspectos:

1. As páginas devem ter um aspecto atraente, por forma a que as pessoas a voltem a visitar.

2. As páginas devem ser bastante interactivas; para isso, deve-se recorrer à programação em linguagens de programação adequadas, como, por exemplo, o *JavaScript*.

3. As imagens até 256 cores (8 *bits*) devem ter o formato GIF .

4. As imagens com 65.536 cores (16 *bits*) devem ter o formato JPEG.

5. A escolha entre o formato GIF e JPEG deve ser um compromisso entre a qualidade e o espaço ocupado pela imagem.

6. As imagens devem ocupar pouca memória; imagens com um tamanho igual ou superior a 20 KB já é considerada uma imagem pesada.

7. Dever-se-á usar compressão para que as imagens ocupem menos espaço, embora a qualidade seja pior.

8. Deve-se reduzir ao máximo as dimensões (altura e largura) das imagens.

9. Deve-se ter em atenção as incompatibilidades entre versões diferentes do HTML e entre *browsers*.

6.6. Exemplo de uma Página HTML

```
<html>
    <head>
          <title>ISI</title>
    </head>

    <body bgcolor="white">
          <h1 align="center">Introdução aos Sistemas Informáticos</h1>

          <p align="center">
                <img src="ISI.jpg"
                     alt="Computador">
          </p>

          <table align="center"
                 border="1"
                 cellpadding="5"
                 cellspacing="0">
                <tr>
                      <th>Componente</th>
                      <th>Itens</th>
                      <th>Valor</th>
                </tr>
```

```
<tr align="center">
      <td rowspan="3">Avaliação Contínua</td>
      <td>Prática</td>
      <td>20 %</td>
</tr>

<tr align="center">
      <td>Teórica</td>
      <td>10 %</td>
</tr>

<tr align="center">
      <td>Trabalho Prático</td>
      <td>10 %</td>
</tr>

<tr align="center">
      <td rowspan="2">Exame</td>
      <td>Com avaliação contínua:</td>
      <td>60 %</td>
</tr>

<tr align="center">
      <td>Sem avaliação contínua:</td>
      <td>100 %</td>
</tr>

<tr>
      <td align="center"
          valign="top">Docentes:</td>
      <td colspan="2">
            <ul>
                  <li>
                        Reis Quarteu (<a
                        href="mailto:reis.quarteu@ipb.pt">
                        reis.quarteu@ipb.pt</a>)
                  </li>
                  <li>
                        Cândida Silva (<a
                        href="mailto:cesilva@ipb.pt">cesil
                        va@ipb.pt</a>)
                  </li>
```

```
                                    <li>
                                          Lúcia Torrão (<a
                                          href="mailto:luciatorrao@ipb.pt">l
                                          uciatorrao@ipb.pt</a>)
                                    </li>
                                    <li>
                                          Pedro Dias (<a
                                          href="mailto:pdias@ipb.pt">pdias@i
                                          pb.pt</a>)
                                    </li>
                              </ul>
                        </td>
                  </tr>
            </table>
      </body>
</html>
```

ISI - Mozilla

File Edit View Go Bookmarks Tools Window Help

Search

Introdução aos Sistemas Informáticos

Componente	Itens	Valor
Avaliação Continua	Prática	20 %
	Teórica	10 %
	Trabalho Prático	10 %
Exame	Com avaliação continua:	60 %
	Sem avaliação continua:	100 %
Docentes:	• Reis Quarteu (reis.quarteu@ipb.pt) • Cândida Silva (cesilva@ipb.pt) • Lúcia Torrão (luciatorrao@ipb.pt) • Pedro Dias (pdias@ipb.pt)	

Figura 6.1: *Exemplo de uma página definida com a linguagem HTML.*

7 – Bases de Dados

7.1. Tópicos Gerais

Uma base de dados (BD) é um sistema de organização e de armazenamento de informação. Esse sistema de organização baseia-se num modelo conceptual. Um modelo de dados é um conjunto de especificações teóricas que estabelecem os princípios e as regras que presidem à organização dos dados. O modelo mais conhecido (e o mais amplamente utilizado) de organização de informação em bases de dados é o **modelo relacional**.

Para que uma base de dados (criada de acordo com um determinado modelo teórico) possa ser implementada numa determinada plataforma de *hardware* e de *software*, torna-se necessário recorrer a um conjunto de programas, cuja missão é a de organizar e gerir essa informação nos computadores. Esse conjunto de programas é designado por **Sistema de Gestão de Bases de Dados** (SGBD).

Os SGBDs possuem uma arquitectura baseada em três níveis distintos:

- **Nível Físico:** Define as formas de armazenamento dos dados em ficheiros e nos diferentes suportes físicos (discos, CDs, unidades de cópias de segurança, etc.);
- **Nível Conceptual:** Define a organização da informação em tabelas e relações;
- **Nível de Visualização:** Define as interfaces que apresentam a informação da BD.

Alguns dos SGBDs mais utilizados são os seguintes:

- *Microsoft Access* (http://office.microsoft.com/access/);
- *Microsoft SQL Server* (http://msdn.microsoft.com/sql/);
- *Oracle* (http://www.oracle.com/);
- *MySQL* (http://www.mysql.org/).

Alguns exemplos de organizações que necessitam de bases de dados são: bancos, instituições de ensino (escolas, institutos politécnicos, universidades, centros de formação profissional, institutos de línguas, etc.), agências seguradoras, Segurança Social, Finanças, hospitais, tribunais, bibliotecas, demais instituições governamentais, empresas, etc. Ou seja, praticamente todo o tipo de organização necessita de um SGBD para armazenar e gerir a informação produzida pelas suas actividades diárias.

7.2. Estrutura de uma Base de Dados

Neste subcapítulo, será analisada a estrutura de uma base de dados relacional, tomando como referência o SGBD *Microsoft Access*.

7.2.1. Tabelas

As tabelas constituem os objectos fundamentais de uma base de dados. É nas tabelas que a informação contida numa base de dados é efectivamente armazenada. Do ponto de vista lógico, uma tabela é uma estrutura rectangular, formado por uma ou mais linhas e por uma ou mais colunas. As linhas de uma tabela são designadas por **registos** e as colunas são designadas por **campos** ou **atributos**. Os campos são caracterizados por poderem armazenar dados de um determinado **tipo**.

As definições seguintes são fundamentais para se poder compreender o modelo relacional de bases de dados:

- **Tabela:** um conjunto de registos, organizados em campos, que representam algo de relevante para um determinado sistema ou problema.
- **Registo:** conjunto de campos cujos valores caracterizam alguma entidade, real ou abstracta, de um determinado sistema ou problema.
- **Campo:** Característicam relevante de todos os registos de uma tabela.
- **Relacionamento** ou **relação:** Ligação entre duas tabelas através da qual um registo de uma tabela se relaciona com um conjunto de registos da outra tabela.

Tipos de Dados

Define-se **tipo de dados** como o conjunto de dados que um campo pode tomar. Serão aqui apresentados apenas os tipos de dados mais importantes disponibilizados pelo SGBD *Microsoft Access*.

- **Texto:** Pode conter texto formado por quaisquer caracteres. Pode conter até 255 caracteres.
- **Memo:** É usado para textos extensos, até um limite de aproximadamente 64.000 caracteres.
- **Número:** Destina-se a armazenar informação numérica, podendo ser definido como:
  - **Byte:** Admite números inteiros entre 0 e 255; ocupa apenas 1 *byte*.
  - **Número Inteiro:** Admite números inteiros entre -32.768 e 32.767; ocupa 2 *bytes*.
  - **Número Inteiro Longo:** Admite números inteiros entre -2.147.483.648 e 2.147.483.647; ocupa 4 *bytes*.
  - **Simples:** Oferece uma precisão de 7 casas decimais e permite representar valores numéricos negativos compreendidos entre $-3,402823 \times 10^{38}$ e $-1,401298 \times 10^{-45}$, e valores numéricos positivos compreendidos entre $1,401298 \times 10^{-45}$ e $3,402823 \times 10^{38}$; ocupa 4 *bytes*.
  - **Duplo:** Oferece uma precisão de 15 casas decimais e permite representar valores numéricos negativos compreendidos entre $-1,79769313486231 \times 10^{308}$ e $-4,94065645841247 \times 10^{-324}$, e valores numéricos positivos compreendidos entre $4,94065645841247 \times 10^{-324}$ e $1,79769313486231 \times 10^{308}$; ocupa 8 *bytes*.
- **Data/Hora:** Armazena valores numéricos que representam datas e horas compreendidas entre os anos 100 e 9.999; ocupa 8 *bytes*.
- **Moeda:** Permite armazenar valores monetários e numéricos que necessitem de uma precisão de até 4 casas decimais; ocupa 8 *bytes*.
- **Numeração Automática:** Utiliza-se quando se pretende que o SGBD numere automaticamente os novos registos inseridos na BD; ocupa 4 *bytes*.

- **Sim/Não:** Campo que apenas admite os valores lógicos verdadeiro ou falso; ocupa apenas 1 *bit*.

Chave Primária

Para garantir que não existem dois registos exactamente iguais numa mesma tabela, é necessário definir um conjunto de um ou mais campos cujos valores sejam garantidamente diferentes em todas as linhas da tabela; esse conjunto de campos é designado por **chave primária**.

Assim, uma definição de chave primária é esta: trata-se de um conjunto de campos de uma tabela cujos valores identificam unicamente um registo dessa tabela.

Por exemplo, numa tabela de alunos, o número de aluno poderá ser a chave primária, já que identifica univocamente cada aluno. Por outro lado, o nome não poderá ser uma chave primária, já que poderão haver mais de um aluno com o mesmo nome.

7.2.2. Consultas

Uma consulta é um objecto da base de dados, constituído por um conjunto de comandos que permitem extrair informação das tabelas ou actuar sobre a informação existente, modificando-a (pode-se alterar informação já existente, acrescentar nova informação, ou eliminar informação).

As consultas são definidas através da linguagem SQL (*"Structured Query Language"*, Linguagem de Perguntas Estruturadas), uma linguagem padrão e específica para a consulta de informação armazenada em bases de dados relacionais.

7.2.3. Formulários e Relatórios

Um formulário é fundamentalmente um objecto da interface gráfica do SGBD Microsoft Access. A criação de formulários permite desenhar uma interface mais agradável e intuitiva, utilizando uma variedade de controlos que tornam mais fácil a tarefa do utilizador.

Um relatório é uma forma de apresentar a informação contida numa tabela, ou obtida através de uma consulta, em formato apropriado à impressão em papel. Um relatório pode conter campos, texto, imagens e cálculos estatísticos.

7.3. Normalização

Uma base de dados tem de ser projectada de modo que a informação não contenha dados redundantes. A **normalização** de uma base de dados é um processo que consiste na reestruturação da informação em diversas tabelas de tal forma que não exista redundância nos dados armazenados. O processo de normalização de uma base de dados é geralmente efectuado seguindo um conjunto de regras, designadas por **formas normais**:

- **1.ª Forma Normal (1FN):** Todos os grupos de campos com valores repetidos dentro de uma tabela devem ser retirados dessa tabela e colocados numa nova tabela.
- **2.ª Forma Normal (2FN):** Todos os campos não pertencentes à chave primária de uma tabela devem depender apenas de toda a chave primária, e não apenas de parte da chave.
- **3.ª Forma Normal (3FN):** Não deve haver dependências entre campos não pertencentes à chave primária de uma tabela.

As figuras seguintes apresentam um exemplo de um processo de normalização. Inicialmente, toda a informação da base de dados está concentrada numa única tabela (figura 7.1). Pode-se verificar que, para cada aluno, o valor do campo Instituição apenas depende do valor do campo Curso (isto é, sabendo-se o curso, é possível saber qual a instituição onde esse curso é leccionado). Assim, o processo de normalização começa por separar os dados referentes às instituições e colocá-los numa nova tabela (figura 7.2). O campo ID dessa nova tabela é do tipo Numeração Automática e permite que o nome de cada instituição apenas exista uma única vez na base de dados, contribuindo assim para a redução da redundância dos dados. O próximo passo consiste em separar os dados referentes aos cursos e colocá-los numa segunda nova tabela (figura 7.3). Também nessa tabela surge um campo ID, do tipo Numeração Automática, devido às mesmas razões apontadas para a tabela de instituições. Contudo, o tipo de dados do campo Instituição da tabela Cursos passou a ser numérico. A razão para isso é que esse campo é uma **chave externa**[26] utilizada para estabelecer a relação entre a tabela Cursos e a tabela Instituições, devendo, por isso, ser do mesmo tipo de dados do campo correspondente à chave primária da tabela Instituições. Por fim, o tipo de dados do campo Curso da tabela Alunos também passou a ser numérico (figura 7.4), porque esse campo é a chave externa que permite estabelecer a relação entre a tabela Alunos e a tabela Cursos. O esquema da base de dados normalizada encontra-se representado na figura 7.5.

26 Uma **chave externa** é um conjunto de campos (em geral, apenas um campo) de uma tabela que são usados para referenciar um registo existente numa outra tabela. Também são conhecidas como "chave estrangeira".

Alunos : Tabela

	Nome	Número	Curso	Instituição
▶	Rui Silva	2223	Engenharia Mecânica	ESTiG
	José Rodrigues	3200	Engenharia Química	ESTiG
	Antonieta da Silva	3445	Engenharia Química	ESTiG
	António Sala Pires	4356	Engenharia Mecânica	ESTiG
	Manuel Cardoso	6566	Engenharia Informática	ESTiG
	Manuela Pires	7689	Engenharia Informática	ESTiG
	Victor dos Santos	9889	Produção Agrícola	ESA
*				

Figura 7.1: *Base de dados não normalizada.*

Instituições : Tabela

	ID	Instituição
▶	1	ESA
	2	ESTiG
*	ca)	

Figura 7.2: *Tabela de instituições.*

Cursos : Tabela

	ID	Curso	Instituição
▶	1	Engenharia Informática	2
	2	Engenharia Mecânica	2
	3	Engenharia Química	2
	4	Produção Agrícola	1
*	ca)		

Figura 7.3: *Tabela de cursos.*

Alunos : Tabela

	Nome	Número	Curso
▶	Antonieta da Silva	3445	3
	António Sala Pires	4356	2
	José Rodrigues	3200	3
	Manuel Cardoso	6566	1
	Manuela Pires	7689	1
	Rui Silva	2223	2
	Victor dos Santos	9889	4
*			

Figura 7.4: *Tabela de alunos.*

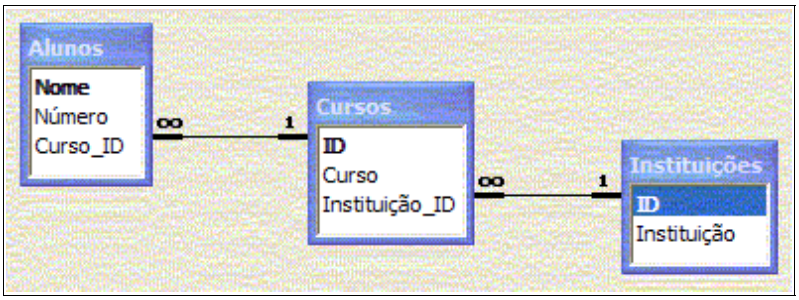


Figura 7.5: *Base de dados normalizada. Os relacionamentos são representados por linhas que unem as tabelas através de determinados campos. Note-se as multiplicidades das relações indicadas sobre cada linha de relacionamento.*

7.4. Manutenção da Integridade das Bases de Dados

A manutenção da integridade de uma base de dados é fundamental para o seu bom funcionamento, devendo ser assegurada pelo SGBD ou pelas aplicações que interagem com os dados.

Uma base de dados relacional deve assegurar dois tipos de integridade:

- **Integridade de Entidade:** Os valores dos atributos que correspondem à chave primária de uma entidade não podem ser nulos nem iguais a outros existentes na tabela.
- **Integridade Referencial:** Um valor de uma chave externa tem obrigatoriamente de existir como valor de um campo da chave primária da tabela relacionada com aquela chave externa.

8 – Algoritmos

8.1. Conceito de Algoritmo

Um algoritmo pode ser definido como uma sequência ordenada e não ambígua de passos elementares que levam à solução de um dado problema. Por exemplo, as indicações dadas para se chegar até uma determinada rua constituem um algoritmo para se encontrar essa rua. De igual modo, uma receita de cozinha é também uma forma muito conhecida de algoritmo.

O uso de uma linguagem algorítmica é fundamental porque, depois de desenvolvido o algoritmo, o programa poderá ser implementado em qualquer linguagem de programação. Para isso, é necessário que os algoritmos cumpram algumas regras. Assim, os passos de um algoritmo devem ser simples e não ambíguos, devendo estar cuidadosamente ordenados.

8.2. Descoberta Algorítmica

O desenvolvimento de um algoritmo consiste na subdivisão de um problema em problemas de menor dimensão, até se chegar a uma solução. Como exemplo, vamos elaborar um algoritmo para realizar uma tarefa bastante familiar: a substituição de uma lâmpada queimada. Numa primeira abordagem, a operação básica pode ser expressa em apenas dois passos simples:

1. Remova a lâmpada queimada;
2. Coloque a nova lâmpada.

Apesar de esta solução parecer resolver o problema, de facto estes passos não são suficientemente claros para uma máquina como, por exemplo, um robot, pois cada um dos passos envolve alguns pressupostos que um robot não conhece. Assim, o simples passo "remova a lâmpada queimada" pode ser expandido nos seguintes passos:

1. Posicione a escada debaixo da lâmpada queimada;
2. Suba a escada até que a lâmpada possa ser alcançada;
3. Rode a lâmpada queimada no sentido anti-horário até que ela se solte.

Existe a necessidade da introdução de diversos passos intermédios por forma a descrever em pormenor cada passo necessário a dar:

1. Posicione a escada debaixo da lâmpada queimada;
2. Seleccione uma nova lâmpada para a substituição;
3. Se a potência não for a mesma da queimada, repita:
    i. Deite fora a lâmpada seleccionada;
    ii. Seleccione uma lâmpada nova.
4. Repita até que a lâmpada possa ser alcançada:
    i. Suba um degrau na escada.
5. Repita até que a lâmpada fique solta:
    i. Rode a lâmpada no sentido anti-horário.
6. Posicione a nova lâmpada;
7. Repita até que a lâmpada esteja colocada:
    i. Rode a lâmpada no sentido horário.
8. Repita até alcançar o chão:
    i. Desça um degrau na escada.

Este processo é designado de **descoberta algorítmica** ou **aproximação algorítmica**, que consiste em decompor o problema em subproblemas, por forma a se chegar a uma solução. Após se ter desenvolvido o algoritmo, a implementação (numa linguagem de programação) é um simples acto de tradução, deixando de ser de raciocínio (figura 8.1). Por outro lado, a tentativa de solucionar um problema através da implementação directa de um programa escrito numa linguagem de programação, sem antes passar pela determinação de um algoritmo capaz de resolver o problema, pode-se revelar muito complicada.

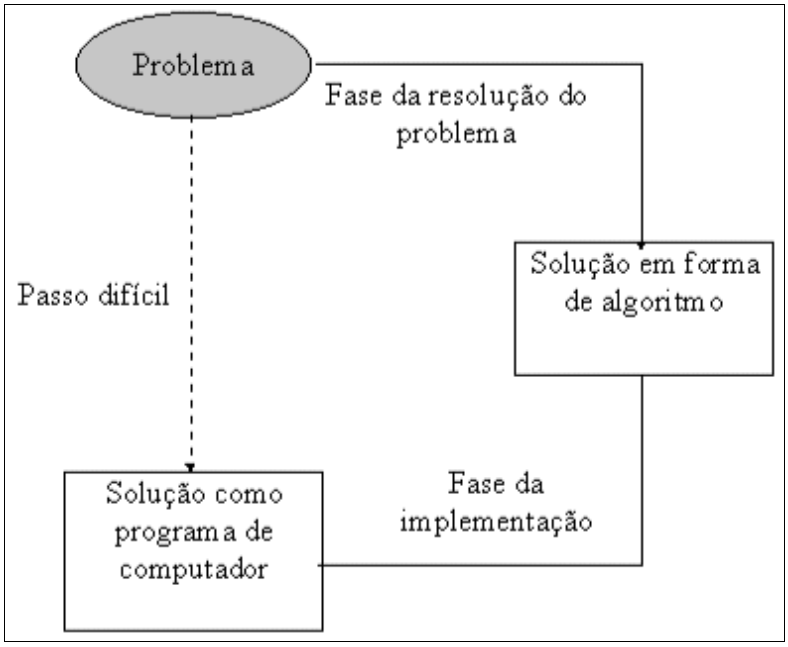


Figura 8.1: *Processo de descoberta (ou aproximação) algorítmica.*

8.3. Representação Algorítmica

8.3.1. Dados de Tipo Simples

Os dados de tipo simples podem ser:

- Inteiros;
- Reais;
- Lógicos ou booleanos;

- Cadeias de caracteres ou alfanuméricas[27] (*"strings"*).

8.3.2. Variáveis e Expressões

É possível manipular os dados dos tipos referidos no subcapítulo anterior através da combinação das operações fundamentais, para formar expressões mais elaboradas. Na matemática, o conceito é muito familiar. Por exemplo, se identificarmos a hipotenusa de um triângulo rectângulo por *a*, e os seus catetos por *b* e *c*, o Teorema de Pitágoras dá-nos a seguinte relação entre os três lados: $a^2 = b^2 + c^2$.

O conceito de **variável** é um dos mais importantes no domínio da algoritmia e programação. Uma variável representa uma posição de memória que armazena valores de um determinado tipo de dados, e que podem ser lidos e alterados por um programa. Uma variável pode receber muitos valores diferentes, mas, num dado instante, ela contém apenas um valor. É da responsabilidade do programador definir o nome das variáveis de um programa. O nome de uma variável deve representar convenientemente os valores que essa variável deverá armazenar.

8.3.3. Operação de Atribuição

A operação de atribuição é a forma de especificar que a uma dada variável será dado um valor. O operador de atribuição é uma pequena seta a apontar para a esquerda (←). Por exemplo:

- A ← 6 (o valor numérico 6 é atribuído à variável A);
- B ← – 31 (o valor numérico -31 é atribuído à variável B);
- C ← verdadeiro (o valor lógico verdadeiro é atribuído à variável C);
- D ← "cadeia de caracteres" (a cadeia de caracteres indicada é atribuída à variável D).

O lado direito da instrução de atribuição pode ser qualquer expressão. Uma expressão é uma combinação de variáveis, constantes e operadores. O resultado da avaliação de uma expressão no lado direito da instrução de atribuição é o valor que é atribuído à variável indicada no lado esquerdo dessa instrução. Deve-se ter em conta que a expressão deve gerar

27 Alguns exemplos de cadeias de caracteres são: "Instituto Politécnico", "automóvel", "Ano 2000" e "23". As cadeias alfanuméricas devem ser delimitadas com aspas (' " ').

um valor cujo tipo de dados seja compatível com o tipo de dados da variável à qual aquele valor será atribuído.

Um exemplo de uma sequência possível de operações é dado a seguir. Este exemplo manipula 3 variáveis cujos nomes são *numero1*, *numero2* e *resultado*. A soma dos valores das variáveis *numero1* e *numero2* é atribuído à variável *resultado*.

```
numero1 ← 13
numero2 ← 8
resultado ← numero1 + numero2
```

8.3.4. Entrada e Saída de Dados

Os dois comandos que permitem manusear as entradas e saídas de dados são:

1. O comando **Leia** permite ler valores, atribuindo-os às variáveis indicadas;
2. O comando **Escreva** permite mostrar os resultados ao utilizador.

A forma do comando **Leia** é, por exemplo, a seguinte:

```
Leia (A, B, C)
```

A forma do comando **Escreva** é, por exemplo, a seguinte:

```
Escreva ("A = ", A, " B =" , B, "C = ", C)
```

Um exemplo de um algoritmo mais elaborado e que utiliza o comando de saída de dados é dado a seguir:

```
nota1 ← 15
nota2 ← 13
nota3 ← 8
media ← (nota1 + nota2 + nota3) / 3
Escreva ("A média das notas é: ", media)
```

Se pretendêssemos que fosse o utilizador a fornecer os dados para as notas, então os 3 primeiros comandos de atribuição deveriam ser substituídos por um único comando de entrada de dados:

```
Leia (nota1, nota2, nota3)
```

8.4. Estruturas de Controlo

8.4.1. Estruturas de Selecção Simples

Se uma determinada condição (representada por uma expressão lógica) for verdadeira, então um conjunto de instruções será executado. Em alternativa, pode-se também indicar um conjunto de instruções que apenas será executado se a condição anterior for falsa.

Se *condição*
Então *alternativa verdadeira*
Fim de Se

Se *condição*
Então *alternativa verdadeira*
Senão *alternativa falsa*
Fim de Se

Exemplo:

```
Se A > B
Então escreva (A)
Senão escreva (B)
Fim de Se
```

8.4.2. Estrutura de Selecção Múltipla

Considere-se, como exemplo deste tipo de estrutura de selecção, um programa que apresente o menu de opções representado abaixo:

```
____________________
Menu
1 - Pára robot
2 - Arranca robot
3 - Sair
____________________
```

Um algoritmo capaz de representar o funcionamento deste programa é o seguinte:

```
Se opcao = 1
Então pára robot
```

```
Senão Se opcao = 2
      Então arranca robot
      Senão sair
      Fim de Se
Fim de Se
```

As estruturas de selecção múltipla também podem ser implementadas com o auxílio do comando **Caso**. Desta forma, o algoritmo anterior pode ser reescrito tal como se mostra a seguir:

```
Caso opcao de
"1" : pára robot
"2" : arranca robot
"3" : sair
Fim de Caso
```

8.5. Estruturas Iterativas

8.5.1. Estruturas de Repetição Condicional

Este tipo de estrutura permite que um bloco de instruções seja executado enquanto uma determinada condição for verdadeira. Quando essa condição passar a ser falsa, a execução do algoritmo prossegue na instrução imediatamente a seguir à estrutura repetitiva.

Enquanto *condição*
    Bloco de instruções
Fim de Enquanto

Como exemplo, considere-se um algoritmo de um programa que permite calcular o factorial de um número inteiro:

```
Leia (N)
factorial ← 1
contador ← 1

Enquanto contador < N
    contador ← contador + 1
    factorial ← factorial * contador
Fim de Enquanto
```

```
Escreva ("Factorial de ", N, " = ", factorial)
Fim
```

8.5.2. Estrutura de Repetição Incondicional

Este tipo de estrutura permite que um bloco de instruções seja executado um determinado número de vezes. Quando todas as iterações da estrutura repetitiva tiverem sido executadas, a execução do algoritmo prossegue na instrução imediatamente a seguir ao fim da estrutura repetitiva.

Para *var–controlo* ← valor_inicial **até** valor_final **repita**
    Bloco de instruções
Fim de Para

Como exemplo, considere-se um algoritmo que escreve o valor que a variável contadora do ciclo toma em cada iteração da estrutura de repetição incondicional.

```
Para contador ← 1 até N repita
    Escreva ("Contador = ", contador)
Fim de Para
```

9 – Linguagens de Programação

9.1. Perspectiva Histórica

As linguagens de programação têm como função descrever todas as operações que devem ser efectuadas por um computador, e que são necessárias para a resolução de um determinado problema. Existe uma grande variedade de linguagens de programação, divididas tipicamente em duas grandes categorias:

1. Linguagens de baixo nível;
2. Linguagens de alto nível.

Alguns exemplos de **linguagens de baixo nível** são os vários códigos-máquina e as diversas linguagens *assembly*. O vocabulário deste tipo de linguagens é muito elementar e completamente dependente do *hardware* que executa os programas escritos nestas linguagens. Como consequência, a formulação dos problemas é extremamente complicada e muito sujeita a erros. Em compensação, a execução dos programas é muito rápida, já que os programas são directamente escritos na linguagem compreendida pela UCP.

Por outro lado, as **linguagens de alto nível** caracterizam-se por possuírem um vocabulário adequado à expressão de problemas de grande complexidade, o que facilita a tarefa do programador, uma vez que os programas são escritos com o auxílio de linguagens muito mais próximas da linguagem humana, em claro contraste com as linguagens de baixo nível. Outra característica muito importante deste tipo de linguagens é a sua independência em relação ao *hardware* onde os programas serão executados: o mesmo código fonte pode ser executado em diferentes UCPs. Essas vantagens têm um inconveniente, já que tornam a tradução e a execução dos programas mais lentas: as linguagens de alto nível não são compreensíveis para a UCP, pelo que torna-se absolutamente necessário que os programas escritos nessas linguagens sejam traduzidos para código máquina. Além disso, o próprio processo de tradução dos programas gera código máquina não optimizado. Alguns exemplos de linguagens de alto nível são: BASIC, Fortran, COBOL, C, Pascal, Java, Prolog, etc.

9.2. Conceitos Tradicionais de Programação em Pascal

Um programa escrito na linguagem de programação Pascal é composto por um **cabeçalho** e por um **corpo de programa**, que inclui a zona de **declarações** e a zona de **instruções**, tal como é esquematizado a seguir:

- Cabeçalho do programa;
- Zona de declarações:
  - Declaração de **constantes**;
  - Declaração de **tipos de dados**;
  - Declaração de **variáveis**;
  - Declaração de **subprogramas**.
- Zona de instruções.

Um exemplo de um programa simples em Pascal, que apenas lê um valor em escudos e o converte num valor em euros, é listado abaixo:

```
program ConverteEscudosParaEuros(input, output);
uses WinCrt;
const
    taxa_conversao = 200.482;
var
    escudos, euros: real;
begin
    Write('Escreva a quantidade em escudos: ');
    ReadLn(escudos);
    euros := escudos / taxa_conversao;
    WriteLn('A quantidade em euros é ', euros:3:2, ' Euros');
end.
```

9.2.1. Tipos de Dados

Os tipos de dados pré-definidos da linguagem de programação Pascal são:

- Valores numéricos:

- **integer:** representa números inteiros compreendidos entre -32.768 e 32.767; ocupa 2 *bytes*.

- **real:** Oferece uma precisão de 11 ou 12 dígitos significativos e permite representar valores numéricos negativos compreendidos entre $-1,7 \times 10^{38}$ e $-2,9 \times 10^{-39}$, e valores numéricos positivos compreendidos entre $2,9 \times 10^{-39}$ e $1,7 \times 10^{38}$; ocupa 6 *bytes*.

- Valores lógicos ou booleanos:

  - **boolean:** apenas admite os valores **true** (verdadeiro) e **false** (falso).

- Valores que representam caracteres ASCII:

  - **char:** apenas admite um caracter, delimitado por apóstrofos ("pelicas"); exemplos: 'A', 'a', '?', '5', '+', etc.

- Cadeias de caracteres:

  - **string:** admite cadeias alfanuméricas com, no máximo e na maior parte dos compiladores Pascal, até 255 caracteres; as cadeias devem ser delimitadas por apóstrofos; exemplos: 'ISI', 'Pascal', 'aula n.º 1', etc.

9.2.2. Constantes

As constantes são representações de valores que não são alterados durante a execução do programa. Elas possuem um identificador (ou nome) e um valor, e são definidas na zona declarativa de um programa, tal como se indica:

```
const identificador = valor;
```

O tipo de dados da constante encontra-se implícito no valor. Por exemplo, no programa da conversão de valores em escudos para euros, foi definida uma constante chamada **taxa_conversao**, à qual foi atribuído um valor igual a 200,482. Ora, como esse valor é numérico com parte fraccionária, então o tipo de dados que foi automaticamente associado àquela constante foi o tipo **real**.

9.2.3. Variáveis

As variáveis são representações do conteúdo de posições (ou blocos) da memória do computador onde se armazenam valores que podem ser alterados durante a execução do programa. Tal como as constantes, as variáveis possuem um identificador, um valor e um tipo de dados. Também são declaradas na zona declarativa, desta forma:

```
var identificador: tipo_de_dados;
```

Um valor é atribuído a uma variável apenas durante a execução do programa.

9.2.4. Expressões

Uma expressão é um grupo de operandos (isto é, valores utilizados nas operações) agrupados por operadores (isto é, símbolos que representam operações), constituindo formas algébricas, aritméticas ou lógicas que representam um valor de um determinado tipo de dados.

As expressões numéricas representam um valor numérico real ou inteiro. Por exemplo, considere a expressão `(A * B – 2 * C) / 3` em que os identificadores *A*, *B* e *C* representam variáveis ou constantes numéricas.

De forma semelhante, as expressões booleanas representam um valor lógico: **true** (verdadeiro) ou **false** (falso). Por exemplo, considere a expressão `(2 * X) > (Y / 5)`, em que os identificadores *X* e *Y* representam variáveis ou constantes numéricas, embora a expressão represente um valor lógico, retornado pelo operador comparativo > (maior que).

9.2.5. Instruções

Uma instrução é uma ordem que obriga o computador a executar um conjunto de acções determinadas. Em Pascal, diz-se que uma instrução composta é um bloco de instruções delimitado pelas palavras-chave **begin** e **end**.

```
begin
    {sequência de instruções}
end;
```

Instrução de Atribuição

A atribuição de um valor a uma variável é feita com o auxílio do operador de atribuição, **:=**, desta forma:

```
variável := valor;
```

O valor atribuído pode ser uma constante, o valor de uma outra variável, ou o valor de uma expressão. Em quaisquer dos casos, o valor atribuído tem de ser de um tipo de dados compatível com o tipo de dados da variável.

Entrada de Dados

A entrada de dados introduzidos pelo utilizador através do teclado faz-se através do comando **ReadLn**. Este comando guarda os dados lidos nas variáveis indicadas na instrução. Deve-se ter em atenção que a ordem da entrada de dados tem de corresponder à ordem das variáveis na lista que acompanha os comandos.

Quando o utilizador estiver a introduzir dados numéricos, estes devem ser teclados na mesma linha e separados por espaços.

Um exemplo de utilização do comando **ReadLn** é dado a seguir:

```
ReadLn(a, b, c);
```

Visualização de Dados

Os dados produzidos por um programa podem ser visualizados no monitor de um computador através da utilização do comando **WriteLn**. Este comando envia para o monitor o valor de cada variável, expressão ou constante, pela ordem em que estas figuram são indicadas na lista que acompanha aquele comando. Depois de apresentar todos os valores, o comando **WriteLn** provoca uma mudança de linha no monitor. Um comando alternativo, muito semelhante ao **WriteLn**, mas que não provoca a mudança de linha final, é o comando **Write**.

Um exemplo de utilização do comando **WriteLn** é o que se segue:

```
WriteLn('a = ', a , 'b = ', b);
```

9.3. Implementação de um Programa em Pascal

Deve-se referir que, no fim de cada linha de instrução de um programa escrito em Pascal, se deve colocar um ponto e vírgula ";". Contudo, esse delimitador não é necessário quando a instrução precede o fim de um bloco de instruções (marcado pela palavra-chave **end**).

Como exemplo de aplicação, vamos escrever em Pascal o programa que calcula o factorial de um número. Esse programa foi já elaborado em linguagem algorítmica no capítulo anterior.

```
program CalcularFactorial(input, output);
uses WinCrt;
var
    N, factorial, contador: integer;
begin
    Write('Introduza o número: ');
    ReadLn(N);

    factorial := 1;
    contador := 1;

    while contador < N do
    begin
          contador := contador + 1;
          factorial := factorial * contador
    end;

    WriteLn('Factorial de ' , N, ' = ' , factorial)
end.
```

Bibliografia Recomendada

- Quarteu, R.; Sora, D. (2003): ***Estrutura e Funcionamento dos Computadores***. ESTiG/IPB.
- Rocha, N. P., Ramos, F. M. S., Oliveira, J. L. (1997): ***Introdução à Informática***. Universidade de Aveiro (004.01/ROC/INT).
- Velloso, F. C. (1994): ***Informática – Conceitos Básicos***. Editora Campus (004.01/VEL/INF).
- Nabais, C. (1993): ***Iniciação à Informática***. Editorial Presença (004.2.2/NAB/INI).
- Nascimento, A. J.; Heller, J. L. (1990): ***Introdução à Informática***. McGraw-Hill (004.01-1/NAS/INT).
- Matos, J. A. (1994): ***Compêndio de Informática***. Edições Graficria (004.3-1/MAT/COM).
- Azul, A. A. (1997): ***Introdução às Tecnologias de Informação (1 e 2)***. Porto Editora.
- Brookshear, J. G. (1997): ***Computer Science: an Overview***. Addison-Wesley Longman (004/BRO/COM).
- Sampaio, A. (2002): ***Hardware para Profissionais***. FCA (004.3/SAM/HAR).
- Gouveia, J., Magalhães, A. (2002): ***Hardware para PCs e Redes – Curso Completo***. FCA (004.3/GOU/HAR).
- Guimarães, A., Lages, N. (1994): ***Algoritmos e Estruturas de Dados***. LTC (004.6/GUI/ALG).
- Perry, G. (1999): ***Aprenda em 24 Horas Programação***. Editora Campus (004.43/PER/APR).
- Carriço, J. A. (1996): ***Desenho de Bases de Dados***. CTI – Centro de Tecnologias de Informação (004.65/CAR/DES).
- Kent, P. (1995): ***Guia Incrível da Internet***. Makron Books (004.6/KEN/GUI).

- Crumlish, C. (1995): ***Explorando a Internet***. Makron Books (004.7/CRU/EXP).
- Ramalho, J. A. (1996): ***Iniciando em HTML***. Makron Books (004.43/RAM/INI).
- Evans, T. (1996): ***HTML Simples e Rápido***. Makron Books (004.43-1/EVA/HTM).
- Raggett, D. (2002): ***Getting started with HTML***. URL: http://www.w3.org/MarkUp/Guide/. Visitado em 5 de Outubro de 2004.
- Quarteu, R. (2003): ***O Sistema Operativo Linux***. ESTiG/IPB.
- Pereira, P. (2000): ***Linux: Curso Completo***. FCA (004.45/PER/LIN).
- Trezentos, P.; Cardoso, A. (2000): ***Fundamental do Linux***. FCA (004.45/TRE/FUN).
- Sousa, S. (1996): ***Domine a 110 % Access 7 para Windows***. FCA (004.65-1/SOU/DOM).

Mais informações sobre a disciplina poderão ser obtidas no endereço web:

http://www.ipb.pt/~reis.quarteu/

Avaliação da Disciplina

A avaliação está estruturada em três componentes, válidas para todas as épocas:

- **Avaliação contínua**, ao longo das aulas da disciplina:
  - **Componente teórica**: realização de fichas de avaliação sobre a matéria abordada nas aulas teóricas (cotação: **10%** da nota final);
  - **Componente prática**: realização de fichas de avaliação sobre a matéria abordada nas aulas práticas (cotação: **20%** da nota final);
  - Se o aluno não realizar uma ficha, considerar-se-á, para efeitos de classificação final, que a classificação obtida pelo aluno nessa ficha é de **0 (zero)** valores;
  - Das fichas realizadas em cada componente (teórica e prática), não serão consideradas as 2 piores classificações de cada aluno para a classificação final ;
  - **Bonificação:** Os alunos que obtenham uma classificação final maior ou igual a **8,5** valores e menor que **9,5** valores, e que realizem, pelo menos, **80%** do total das fichas teóricas e práticas, terão direito a uma bonificação de **1 (um)** valor na classificação final;
- **Trabalho prático** (cotação: **10%** da nota final).
- **Exame escrito**, englobando toda a matéria (cotação: **60%** da nota final):
  - Para os alunos que não se submetam à avaliação contínua e que não realizem o trabalho prático, o exame escrito incluirá um conjunto de questões adicionais, de tal forma que a sua cotação passa a ser de **100%** da nota final;
  - Os alunos que tenham-se submetido à avaliação contínua e à realização do trabalho prático poderão escolher realizar o exame escrito na sua totalidade (cotação de **100%** da nota final); nesta situação, as notas obtidas com a avaliação contínua e com o trabalho prático não serão consideradas.

Docentes da Disciplina

Docente	Gabinete	Correio Electrónico	Horário de Atendimento
Reis Quarteu	37	reis.quarteu@ipb.pt	4.ª feira: 14 – 15 horas 5.ª feira: 10 – 12 horas 6.ª feira: 10 – 11 horas
Cândida Silva	94	cesilva@ipb.pt	2.ª feira: 16 – 18 horas 3.ª feira: 14 – 16 horas
Lúcia Torrão	68	luciatorrao@ipb.pt	2.ª feira: 15 – 17 horas 3.ª feira: 10 – 12 horas
Pedro Dias	37	pdias@ipb.pt	3.ª feira: 14 – 17 horas