

ÍNDICE DAS FIGURAS

Fig. 1.1 - O computador como sistema de processamento de informação	5
Fig. 1.2 - Estrutura básica de um computador	7
Fig. 1.3 - Arquitectura básica de um computador	7
Fig. 1.4 - Processo de transformar um conceito num programa que um computador saiba executar. Parte do percurso é automático, mas a parte mais difícil ainda é manual ..13	
Fig. 1.5 - Lei de Moore verificada pela evolução do número de transístores dos microprocessadores produzidos pela Intel.....	24
Fig. 2.1 - Amplificador de sinais analógicos. (a) – Com um transístor e uma resistência; (b) – O transístor comporta-se como uma resistência variável, controlada pelo sinal de entrada.....	28
Fig. 2.2 - (a) – Circuito inversor; (b) – Função de transferência	30
Fig. 2.3 - Circuito inversor binário. (a) – Com 0 à entrada e 1 à saída; (b) – Situação inversa.....	31
Fig. 2.4 - Comportamento de dois inversores. (a) – Circuito; (b) – Diagrama temporal simplificado; (c) – Diagrama temporal detalhado	31
Fig. 2.5 - Implementação simplificada de duas portas lógicas. (a) – NAND; (b) – NOR ..35	
Fig. 2.6 - Exemplos de mapas de Karnaugh de 4 entradas	41
Fig. 2.7 - Mapas de Karnaugh das funções Z_1 e Z_2 da Tabela 2.4	43
Fig. 2.8 - Logigrama do circuito correspondente às saídas Z_1 e Z_2 na Fig. 2.7.....	44
Fig. 2.9 - Circuito equivalente de um <i>multiplexer</i> , exemplificado para quatro entradas e dois sinais de selecção. A saída pode ser feita igual a qualquer das entradas, mas apenas a uma de cada vez (de acordo com os sinais de selecção).....	46
Fig. 2.10 - <i>Multiplexers</i> de 2, 4, 8 e 16 entradas, com 1, 2, 3 e 4 sinais de selecção, respectivamente	47
Fig. 2.11 - Logigrama e tabela de verdade dos <i>multiplexers</i> da Fig. 2.10. (a) – Duas entradas; (b) – Quatro entradas.....	47
Fig. 2.12 - Utilização de <i>multiplexers</i> para implementar as funções Z_1 e Z_2 da Tabela 2.3	48
Fig. 2.13 - Mostrador (<i>display</i>) de sete segmentos e algumas das combinações possíveis	49

Fig. 2.14 - Ligação do decodificador ao mostrador de sete segmentos.....	50
Fig. 2.15 - Exemplos de representação. (a) – Decodificador 1-de-4; (b) – Decodificador 1-de-8.....	51
Fig. 2.16 - Estrutura interna de uma ROM (<i>Read Only Memory</i>). Cada OR tem tantas entradas quantas as saídas do decodificador	52
Fig. 2.17 - Trinco SR com NORs. (a) – Circuito; (b) – Tabela de estados.....	55
Fig. 2.18 - Trinco SR com NANDs. (a) – Circuito; (b) – Tabela de estados.....	56
Fig. 2.19 - Trinco D. (a) – Circuito; (b) – Tabela de estados	57
Fig. 2.20 - Bâscula D activada pelo flanco (<i>edge-triggered</i>) ascendente	58
Fig. 2.21 - Representação dos trincos e bâsculas D. (a) – Trinco; (b) – Bâscula D activada no flanco ascendente de C; (c) – Bâscula D activada no flanco descendente de C...58	
Fig. 2.22 - Exemplo de comportamento dos trincos e bâsculas D.....	59
Fig. 2.23 - Bâscula D em circuito de transição alternada (<i>toggle</i>). (a) – Circuito; (b) – Diagrama temporal	59
Fig. 2.24 - Bâscula D com entradas para forçar estados de 0 (CL – <i>Clear</i>) e 1 (PR – <i>Preset</i>).....	60
Fig. 2.25 - Registos. (a) – Trinco (<i>latch</i>) de 8 <i>bits</i> ; (b) – Registo com bâsculas de 8 <i>bits</i> , activado pelo flanco ascendente do relógio e com sinal de inicialização a 0; (c) – Representação do trinco de 8 <i>bits</i> ; (d) – Representação do registo de 8 <i>bits</i>	61
Fig. 2.26 - Utilização de um registo com bâsculas quando as entradas do registo dependem das próprias saídas.....	62
Fig. 2.27 - Utilização de saídas com capacidade de três estados (<i>tristate</i>).....	63
Fig. 2.28 - Registo de 8 <i>bits</i> com saídas <i>tristate</i> . (a) – Versão mais detalhada; (b) – Representação do conjunto a usar em diagramas mais complexos.....	65
Fig. 2.29 - Banco de registos	65
Fig. 2.30 - Contador com bâscula D em sequência (<i>ripple</i>), ligado a um decodificador de sete segmentos. (a) – Circuito; (b) – Diagrama temporal, ilustrando os sucessivos atrasos	66
Fig. 2.31 - Mapas de Karnaugh para simplificar as expressões das entradas das bâsculas do contador	68
Fig. 2.32 - Contador binário, ligado a um decodificador de sete segmentos	69
Fig. 2.33 - Contador binário com possibilidade de carregamento de um valor em paralelo	70
Fig. 2.34 - Exemplo de aplicação de um contador binário com carregamento em paralelo	70

Fig. 2.35 - Exemplo de aplicação de um contador binário de contagem decrescente com carregamento em paralelo. (a) – Circuito; (b) – Evolução temporal dos sinais.....	71
Fig. 2.36 - Registos de deslocamento. (a) – Para a direita; (b) – Para a esquerda.....	72
Fig. 2.37 - Registo de deslocamento para a direita com carregamento em paralelo.....	72
Fig. 2.38 - Modelos de máquinas de estado. (a) – Modelo de Moore; (b) – Modelo de Mealy.....	74
Fig. 2.39 - Diagrama de estados do microondas simples	75
Fig. 2.40 - Circuito do controlo do microondas simples	76
Fig. 2.41 - Diagrama de estados do semáforo simples	78
Fig. 2.42 - Diagrama de estados do semáforo com botão para peões.....	79
Fig. 2.43 - Esquema básico de uma máquina de estados microprogramada, com uma ROM (sem variáveis de entrada).....	81
Fig. 2.44 - Esquema básico de uma máquina de estados microprogramada, com uma ROM (com uma variável de entrada). (a) – versão mais genérica; (b) – com incremento de estado por omissão.....	84
Fig. 2.45 - Conversão de um número decimal em binário.....	87
Fig. 2.46 - Correspondência entre as representações de um número em binário e em hexadecimal.....	88
Fig. 2.47 - Conversão entre as representações de um número em decimal e em hexadecimal.....	89
Fig. 2.48 - Complemento para 2. (a) – Obtenção do simétrico de 42H; (b) – Soma de 42H com o seu simétrico; (c) – Simétrico de zero; (d) – Simétrico de -1	94
Fig. 2.49 - Estrutura de um somador de transporte em onda (<i>ripple</i>). (a) – Somador elementar de 1 <i>bit</i> ; (b) – Circuito interno do somador elementar; (c) – Somador de N <i>bits</i>	97
Fig. 2.50 - Circuito somador/subtractor. Com $F=0$, faz $S=X+Y$. Com $F=1$, faz $S=X-Y$..	99
Fig. 2.51 - Circuito somador/subtractor com detecção de excesso (<i>overflow</i>).....	100
Fig. 2.52 - Multiplicação de números binários sem sinal	101
Fig. 2.53 - Circuito (simplificado) para a multiplicação de números binários sem sinal	102
Fig. 2.54 - Divisão de números binários sem sinal.....	104
Fig. 2.55 - Circuito (simplificado) para a divisão de números binários sem sinal	104
Fig. 3.1 - Arquitectura básica de um computador	113
Fig. 3.2 - Esquema básico de uma RAM (<i>Random Access Memory</i>) com 16 células.....	114

Fig. 3.3 - O sinal <code>ESCR_A</code> memoriza no registo A o primeiro operando lido da memória	119
Fig. 3.4 - Escrita no registo A do resultado de uma soma, activando o sinal <code>ESCR_A</code>	119
Fig. 3.5 - A utilização de um <i>multiplexer</i> permite escolher, de acordo com o sinal <code>SEL_A</code> , a fonte do valor a escrever no registo A: ou o resultado de uma soma ou um valor lido da memória de dados	120
Fig. 3.6 - A ligação da saída do registo A à entrada da memória de dados permite guardar o valor deste registo na memória de dados (só quando se activa <code>WR</code>).....	121
Fig. 3.7 - Soma e subtracção de dois números binários de 8 <i>bits</i>	123
Fig. 3.8 - Equivalência entre subtracção e soma com o complemento para 2	123
Fig. 3.9 - Conjuncção (AND) e disjunção (OR) de dois números binários de 8 <i>bits</i>	124
Fig. 3.10 - Unidade de dados com uma unidade aritmética e lógica (ALU) em vez de um simples somador. O sinal <code>SEL_ALU</code> especifica a operação pretendida	125
Fig. 3.11 - Possível implementação da unidade aritmética e lógica (ALU) da Fig. 3.10	125
Fig. 3.12 - Especificação dos sinais de controlo pelas instruções na memória de instruções	129
Fig. 3.13 - Especificação da instrução a executar pelo contador de programa (PC – <i>Program Counter</i>). Embora rudimentar, este circuito já é o embrião do PEPE-8...	130
Fig. 3.14 - Hipótese de especificação de constantes nas instruções. O problema é que nem todas as instruções têm apenas uma constante. Esta não é uma boa solução	134
Fig. 3.15 - Circuito com suporte para constantes	137
Fig. 3.16 - Circuito com suporte para saltos (condicionais e incondicionais).....	139
Fig. 3.17 - Circuito completo do processador (PEPE-8, versão preliminar) e das memórias de instruções e de dados	143
Fig. 3.18 - Circuito completo do processador (PEPE-8, versão final), com descodificação dos <i>opcodes</i> das instruções por ROM.....	148
Fig. 3.19 - Sinais activos na instrução <code>LD valor</code>	154
Fig. 3.20 - Sinais activos na instrução <code>LD [endereço]</code>	155
Fig. 3.21 - Sinais activos na instrução <code>ST [endereço]</code>	155
Fig. 3.22 - Sinais activos na instrução <code>ADD valor</code>	156
Fig. 3.23 - Sinais activos na instrução <code>ADD [endereço]</code>	156
Fig. 3.24 - Sinais activos na instrução <code>JMP endereço</code>	157
Fig. 3.25 - Sinais activos na instrução <code>JZ endereço</code>	157

Fig. 3.26 - PEPE-8 com ligação às memórias e aos periféricos. As ligações mais importantes têm a sua largura em <i>bits</i> especificada. O <i>bit</i> de maior peso do endereço de dados indica se o acesso é à memória ou aos periféricos.....	165
Fig. 3.27 - Escrita na memória de dados, com os sinais activos destacados. O periférico de saída tem o sinal <i>WR</i> activado, mas não o sinal <i>CS</i> (que é activo a 0), pelo que os seus <i>bits</i> não são alterados	168
Fig. 3.28 - Escrita no periférico de saída, com os sinais activos destacados. A memória de dados tem <i>WR</i> activado, mas não <i>CS</i> (que é activo a 0), pelo que a célula endereçada não é alterada. O periférico de entrada tem <i>CS</i> activo mas não <i>RD</i> , pelo que não é acedido.....	169
Fig. 3.29 - Leitura da memória de dados, com os sinais activos destacados. <i>WR</i> da memória está inactivo (leitura). O periférico de entrada tem <i>RD</i> activo mas não <i>CS</i> , pelo que a sua saída de dados está inactiva, evitando conflitos com a memória.....	170
Fig. 3.30 - Leitura do periférico de entrada, com os sinais activos destacados. A memória tem a sua saída de dados inactiva, para evitar conflito com o periférico.....	170
Fig. 4.1 - Estrutura interna do PEPE-8	182
Fig. 4.2 - Processador com banco de registos.....	186
Fig. 4.3 - As <i>caches</i> permitem uma aparente separação das memórias de instruções e de dados. A interface de memória coordena as escritas e leituras à memória principal	189
Fig. 4.4 - Palavras com os <i>bytes</i> identificados por letras. (a) – De 16 <i>bits</i> ; (b) – De 32 <i>bits</i>	195
Fig. 4.5 - Registos do banco de registos do PEPE. Cada registo tem 16 <i>bits</i> , ou 2 <i>bytes</i> . A divisão vertical é meramente conceptual. Os cinco últimos registos têm algumas funcionalidades específicas, pelo que o seu uso deve ser controlado.....	202
Fig. 4.6 - Disposição dos <i>bits</i> de estado no RE (Registo de Estado).....	204
Fig. 4.7 - Circuito de base para exemplificar as instruções do PEPE.....	211
Fig. 4.8 - Acesso a tabelas em memória através de uma base (<i>R1</i>) e de um índice (<i>R2</i>) para obter o endereço do elemento pretendido. Também é possível aceder directamente, sem tabela (com <i>R3</i>).....	222
Fig. 4.9 - Acesso a tabelas em memória através de uma base e de um índice constante (instrução <i>MOV R0, [R1+6]</i>)	226
Fig. 4.10 - Disposição de uma cadeia de caracteres (<i>string</i>) em memória, usando codificação ASCII. O valor 00H serve de terminador.....	229
Fig. 4.11 - Disposição dos caracteres de 8 <i>bits</i> na memória de 16 <i>bits</i>	230
Fig. 4.12 - Situação pretendida após execução do Programa 4.7	230

Fig. 4.13 - Situação pretendida após execução do Programa 4.8	234
Fig. 4.14 - Soma de um número em duas partes.....	239
Fig. 4.15 - Exemplos de instruções lógicas. Para além do resultado indica-se também o efeito nos <i>bits</i> de estado Z e N.....	245
Fig. 4.16 - Conversão de letras maiúsculas e minúsculas para maiúsculas	250
Fig. 4.17 - Efeito de uma máscara (0FH) com as várias operações lógicas	253
Fig. 4.18 - Cifra de caracteres (negação de alguns <i>bits</i>) com uma máscara XOR.....	257
Fig. 4.19 - Deslocamento linear de um <i>bit</i> (uma posição). Os registos são de 8 <i>bits</i> , mas o princípio é o mesmo para qualquer número de <i>bits</i> . Um deslocamento de N <i>bits</i> repete o deslocamento de um <i>bit</i> N vezes ($1 \leq N \leq 15$)	261
Fig. 4.20 - Deslocamentos lineares, lógicos e aritméticos, à esquerda e à direita. Os registos são de 8 <i>bits</i> , mas o princípio é o mesmo para qualquer número de <i>bits</i> ...	262
Fig. 4.21 - Compactação de dois dígitos em BCD num só <i>byte</i> a partir de dois dígitos em ASCII (38H e 32H, ou '8' e '2').....	264
Fig. 4.22 - Rotação de um <i>bit</i> (uma posição). Os registos são de 8 <i>bits</i> , mas o princípio é o mesmo para qualquer número de <i>bits</i> . Uma rotação de N <i>bits</i> repete a rotação de um <i>bit</i> N vezes ($1 \leq N \leq 15$).....	265
Fig. 4.23 - Rotações, com e sem transporte, à esquerda e à direita. Os registos são de 8 <i>bits</i> , mas o princípio é o mesmo para qualquer número de <i>bits</i>	265
Fig. 5.1 - Ilustração do algoritmo de ordenação de bolha para quatro números. Os rectângulos indicam os pares de números testados. Aqueles em que houve troca de números face à iteração anterior estão a cinzento.....	275
Fig. 5.2 - Elementos básicos de um fluxograma (S – Sim, N – Não).....	276
Fig. 5.3 - Fluxograma do algoritmo de ordenação por bolha	277
Fig. 5.4 - Chamadas de uma função. Os rectângulos são instruções, executados sequencialmente. Os rectângulos mais escuros são os de chamada da função, que retorna sempre para a instrução seguinte àquela que a chamou	305
Fig. 5.5 - Mecanismo de chamada/retorno de rotinas com endereço de retorno guardado em memória (na pilha), com a evolução do estado da pilha ao longo da chamada/retorno de algumas rotinas	321
Fig. 5.6 - Implementação da pilha, em várias situações: (a) – Pilha vazia; (b) – Pilha com três endereços de retorno; (c) – Pilha com seis endereços de retorno; (d) – Pilha cheia.....	323
Fig. 5.7 - Variantes da implementação da pilha, em várias situações: (a) a (d) – Pilha com funcionamento igual ao da Fig. 5.6 (usada no PEPE); (e) a (h) – Pilha com topo em baixo e SP a apontar para a primeira posição livre.....	331

Fig. 5.8 - Evolução da pilha ao longo da chamada da rotina da Tabela 5.25: (a) – Estado inicial; (b) – Após colocar parâmetros na pilha; (c) – Após o CALL; (d) – Após a rotina guardar os valores de registos na pilha; (e) – Após escrita do resultado (registo <i>w</i>) na pilha; (f) – Após repor os registos guardados na pilha; (g) – Após o RET (h) – Estado que a pilha deveria ter após o RET	348
Fig. 5.9 - Evolução do contexto de uma rotina ao longo da sua chamada: (a) – Estado inicial; (b) – Após colocar parâmetros na pilha; (c) – Após o CALL; (d) – Após a rotina guardar os registos na pilha; (e) – Após escrita do resultado (registo <i>w</i>) na pilha; (f) – Após repor os registos guardados na pilha; (g) – Após o retorno; (h) – Estado que a pilha deveria ter após o retorno	351
Fig. 5.10 - Implementação de uma tabela de duas dimensões em memória. (a) – Tabela em duas dimensões; (b) – Tabela linearizada por linhas; (c) – Índice dos elementos nas tabelas linearizadas; (d) – Tabela linearizada por colunas	367
Fig. 5.11 - Organização típica da pilha e do montão	379
Fig. 5.12 - Exemplo de lista ligada. Tem de haver um registo ou célula de memória que aponte para a cabeça. (a) – Lista original; (b) – Após inserção de um novo elemento antes da cauda; (c) – Após remoção da <i>Ficha2</i> da lista	382
Fig. 5.13 - Uma possível implementação em memória do exemplo da Fig. 5.12, mostrando o valor dos apontadores	383
Fig. 5.14 - Ciclo de desenvolvimento de um programa	389
Fig. 5.15 - Ambientes de desenvolvimento de um programa: (a) – Para o próprio computador; (b) – Para outro computador(desenvolvimento cruzado – <i>cross development</i>)	396
Fig. 5.16 - Exemplo de um computador alvo usado num sistema de desenvolvimento cruzado	398
Fig. 5.17 - Exemplo de um microcontrolador, um computador completo num só circuito integrado	400
Fig. 6.1 - Barramentos do PEPE, ilustrando a ligação do processador aos dispositivos	410
Fig. 6.2 - Circuito da Fig. 6.1 corrigido em termos de barramento de controlo (os sinais RD e WR só devem ligar aos dispositivos que suportem a operação respectiva)	413
Fig. 6.3 - Descodificação de endereços, activando um dos sinais individuais de selecção de dispositivo para um determinado valor do barramento de endereços	416
Fig. 6.4 - Descodificação de endereços para implementar o mapa de endereços da Tabela 6.2	418
Fig. 6.5 - Representação gráfica do mapa de endereços da Tabela 6.2 e evolução dos quatro <i>bits</i> de maior peso do barramento de endereços. Note-se que o último endereço não é 64 K mas sim 64 K–1 (FFFFH)	419

Fig. 6.6 - Discriminação de todos os <i>bits</i> de endereço de forma a eliminar os espelhos na descodificação de endereços.....	422
Fig. 6.7 - Descodificação de um mapa de endereços irregular com descodificadores	424
Fig. 6.8 - Descodificação de um mapa de endereços irregular com PROMs	426
Fig. 6.9 - Descodificação de endereços. (a) – De palavra; (b) – De <i>byte</i> ; (c) Detalhe de geração dos sinais de selecção par e ímpar.....	429
Fig. 6.10 - Sistema da Fig. 6.4 mas agora com descodificação de endereços. O bloco SA (Selector de Acesso) é o descrito pela Fig. 6.9c	432
Fig. 6.11 - Descodificação de endereços de <i>byte</i> separada para os endereços pares e ímpares.....	433
Fig. 6.12 - Organização da memória. (a) – Modelo conceptual com endereçamento de <i>byte</i> (sequência linear de <i>bytes</i>); (b) – Modelo real, ilustrando a ligação ao barramento de dados do processador; (c) – <i>Bytes</i> que constituem cada palavra num acesso em 16 <i>bits</i> (sempre com endereço par).....	434
Fig. 6.13 - Ligação de periféricos de 8 e 16 <i>bits</i>	435
Fig. 6.14 - Métodos de ligar o processador à memória: (a, b, c) – <i>Big-endian</i> ; (d, e, f) – <i>Little-endian</i> (o processador foi omitido para não estar repetido).....	437
Fig. 6.15 - Disposição em memória da palavra de 32 <i>bits</i> 98 76 54 32H com um processador de 32 <i>bits</i> . (a) – <i>Big-endian</i> ; (b) – <i>Little-endian</i>	440
Fig. 6.16 - Alinhamento dos acessos à memória. (a) e (b) – Acessos individuais a um <i>byte</i> , par ou ímpar, sem problemas de alinhamento; (c) – Acesso a uma palavra alinhada; (d) – O acesso a um endereço ímpar (desalinhado) obriga a efectuar dois acessos para aceder correctamente à palavra inteira.....	443
Fig. 6.17 - Ligação de um dispositivo a um barramento de dados e respectivo controlo, com recurso a uma interface <i>tristate</i> caso o dispositivo possa ser lido. (a) – Periférico só de saída; (b) – Periférico só de entrada; (c) – Periférico de entrada/saída; (d) – Memória.....	445
Fig. 6.18 - Ciclo de leitura no acesso à memória/periféricos. O primeiro ciclo acede à memória em modo de <i>byte</i> (BA=1)	447
Fig. 6.19 - Ciclos de escrita no acesso à memória/periféricos. O primeiro ciclo acede à memória em modo de <i>byte</i> (BA=1)	449
Fig. 6.20 - Os ciclos de acesso à memória/periféricos em leitura e escrita podem suceder-se de forma contínua.....	450
Fig. 6.21 - Tempos mais relevantes a respeitar nos ciclos de acesso à memória/periféricos	451

Fig. 6.22 - Prolongamento do tempo de acesso num ciclo de leitura quando o pino <i>WAIT</i> do processador é activado.....	455
Fig. 6.23 - Princípio básico das interrupções. O programa prossegue normalmente, sem se preocupar se alguma tecla foi carregada. O tratamento da tecla é automático....	458
Fig. 6.24 - Uso do registo <i>BTE</i> , da Tabela de Excepções e do número da excepção que ocorreu para determinar o endereço da rotina que a atende.....	459
Fig. 6.25 - Pinos de interrupção do <i>PEPE</i>	461
Fig. 6.26 - Ligação de um interruptor a um dos pinos de interrupção do <i>PEPE</i> . (a) – Resistência de <i>pull-up</i> ; (b) – Resistência de <i>pull-down</i>	461
Fig. 6.27 - <i>Bits</i> de controlo das interrupções no <i>RE</i> (Registo de Estado)	462
Fig. 6.28 - Exemplos que ilustram o funcionamento das interrupções.....	465
Fig. 6.29 - Ciclo de processamento das instruções, incluindo detecção e tratamento das interrupções. Note-se a execução da instrução <i>RFE</i>	468
Fig. 6.30 - Controlo da intensidade luminosa de uma lâmpada de incandescência com recurso a interrupções. O <i>triac</i> é um interruptor electrónico que permite ligar e desligar a lâmpada	469
Fig. 6.31 - Circuito de controlo da intensidade luminosa de uma lâmpada usando um <i>triac</i> comandado por um microprocessador. Por simplicidade, a memória está omitida.....	471
Fig. 6.32 - Controlador de interrupções programável (<i>PIC</i>) e coexistência entre pinos de interrupção simples e vectorizados.....	474
Fig. 6.33 - Ciclo de leitura do vector do controlador de interrupções programável (<i>PIC</i>)	476
Fig. 6.34 - Registo de Estado (<i>RE</i>) incluindo os <i>bits</i> <i>TD</i> e <i>TV</i> , que controlam as excepções <i>DIV0</i> e <i>EXCESSO</i> , respectivamente	479
Fig. 6.35 - Com movimento de rotação do disco e linear do braço consegue posicionar-se a cabeça magnética em qualquer ponto do disco. (a) – Posição de repouso; (b) – Braço posicionado num cilindro intermédio.....	482
Fig. 6.36 - Organização dos dados do disco. (a) – Conjunto de pistas concêntricas; (b) – Uma única pista, a que é possível aceder com o braço fixo; (c) – Pista com sectores marcados	483
Fig. 6.37 - Estrutura de uma interface gráfica e geração das imagens no monitor	485
Fig. 6.38 - Principais métodos de arbítrio. (a) – Em cadeia; (b) – Centralizado paralelo; (c) – Conflito na identificação; (d) – Colisão nos dados	486
Fig. 6.39 - Comunicação paralela assíncrona. (a) – Ligação entre emissor e receptor; (b) – Sequência temporal dos sinais	489

Fig. 6.40 - Comunicação série assíncrona. (a) – Emissor (o tempo de repouso pode variar entre 0 e infinito); (b) – Recuperação dos <i>bytes</i> no receptor por amostragem do sinal; (c) – Receptor com frequência de recepção ligeiramente inferior à frequência de emissão	492
Fig. 6.41 - Envolvente (início e fim) de cada pacote, com indicação do número de <i>bits</i> gastos antes e depois do pacote propriamente dito	494
Fig. 6.42 - Enchimento de <i>bits</i> e codificação NRZI	496
Fig. 6.43 - Topologia física do USB.....	497
Fig. 6.44 - Topologia lógica do USB.....	499
Fig. 6.45 - Formatos dos pacotes mais relevantes do protocolo do USB. Cada campo tem indicado em cima o número de <i>bits</i> que ocupa. (a) – Símbolo; (b) – Dados; (c) – Aperto-de-mão; (d) – Início de quadro	501
Fig. 6.46 - Exemplos de utilização do tempo do USB. Os pacotes a cinzento são enviados pelo dispositivo e os restantes pelo concentrador. (a) – Quadro com duas transacções, ficando o resto do quadro não usado; (b) – Quadro totalmente vazio.....	502
Fig. 6.47 - Arquitectura do sistema de periféricos. (a) – Sistema simples com periféricos ligados directamente ao barramento do sistema; (b) – Sistema de periféricos com barramentos hierárquicos e interfaces para ligar ao barramento de sistema.....	504
Fig. 6.48 - Circuito simples para teste da espera activa.....	506
Fig. 6.49 - Circuito simples para teste da espera não activa (por interrupções)	507
Fig. 6.50 - Transferência de dados por DMA (<i>Direct Memory Access</i>).....	510
Fig. 6.51 - Arquitectura de entradas/saídas. (a) – Com controlador de DMA; (b) – Com processador de entradas/saídas (que inclui controlador de DMA)	513
Fig. 6.52 - Arquitectura original do PC (PC/XT).....	516
Fig. 6.53 - Arquitectura do PC com PCI e o seu <i>chipset</i>	529
Fig. 6.54 - Evolução da arquitectura do PC, com porto AGP.....	529
Fig. 6.55 - <i>Chipset</i> baseado em concentradores (<i>hubs</i>)	530
Fig. 6.56 - Arquitecturas genéricas. (a) – Do PEPE ; (b) – Do CREPE	535
Fig. 6.57 - Arquitectura de um computador baseado no PEPE	558
Fig. 7.1 - Diagrama geral do PEPE.....	562
Fig. 7.2 - Diagrama geral do núcleo do PEPE, com ênfase no caminho de dados, e ligações às <i>caches</i>	565
Fig. 7.3 - Diagrama de blocos do banco de registos do PEPE.....	569
Fig. 7.4 - Diagrama de blocos da ALU do PEPE	573

Fig. 7.5 - Diagrama de blocos simplificado da unidade de excepções. Apenas estão representadas algumas excepções, a título exemplificativo.....	576
Fig. 7.6 - Inserção da Unidade de Controlo no diagrama geral do núcleo do PEPE	578
Fig. 7.7 - Circuito microprogramado para deslocar um registo de N bits	582
Fig. 7.8 - Formato da instrução máquina para as novas instruções definidas. (a) – ADDM [Rd], k; (b) – SUM Rc, [Rs], Rd.....	588
Fig. 7.9 - Princípio do processamento em estágios. (a) – Processamento sequencial (uma só instrução de cada vez); (b) – Cadeia de estágios, com unidades especializadas e registos interestágio; (c) – Processamento em estágios	592
Fig. 7.10 - Efeito das bolhas na cadeia de estágios. Nos instantes T2, T4 e T5 nenhuma instrução fica pronta, perdendo-se eficiência	594
Fig. 7.11 - Estrutura das cadeias de estágios no PEPE. A cadeia de estágios de microinstruções pode evoluir várias vezes durante um estágio da cadeia de instruções	595
Fig. 7.12 - Arquitectura geral do núcleo do PEPE com cadeias de estágios	600
Fig. 7.13 - Evolução das cadeias de estágios com o programa Programa 7.1 da página 602, evidenciando as dependências de dados. Nas instruções com mais de uma microinstrução, só a primeira tem estágio BI e D, este último substituído pelo BM nas microinstruções seguintes.....	610
Fig. 7.14 - Uso de um relógio de duas fases, permitindo leitura após escrita no mesmo ciclo de relógio. (a) – Diagrama temporal; (b) – Registo interestágio; (c) – Registo do banco de registos	611
Fig. 7.15 - Detalhe da cadeia de estágios de microinstruções com os <i>multiplexers</i> para antecipação de dados. Esta figura é apenas ilustrativa, pois o PEPE não suporta esta característica, que exige um circuito de detecção de conflitos	612
Fig. 7.16 - Resolução dos conflitos de dados da Fig. 7.13 com atrasos e troca de instruções independentes (assumindo que os <i>bits</i> de estado não têm influência nas instruções seguintes).....	613
Fig. 7.17 - Esquema básico de ligação das <i>caches</i>	618
Fig. 7.18 - Organização básica de uma <i>cache</i>	620
Fig. 7.19 - <i>Cache</i> de mapeamento directo	622
Fig. 7.20 - Mapeamento dos blocos na <i>cache</i> no mapeamento directo. Nesta figura, os elementos de memória representados são blocos e não células individuais	623
Fig. 7.21 - Estrutura de uma <i>cache</i> com mapeamento associativo	625
Fig. 7.22 - <i>Cache</i> de mapeamento associativo de duas vias. Consiste de duas <i>caches</i> de mapeamento directo, com procura simultânea (associativa) nas duas <i>caches</i>	627

Fig. 7.23 - Variantes da associatividade com oito blocos (apenas estão representadas as etiquetas). (a) – Mapeamento directo (só uma via); (b) – Mapeamento associativo com duas vias; (c) – Mapeamento associativo com quatro vias; (d) – Mapeamento totalmente associativo (um só conjunto)	628
Fig. 7.24 - Evolução do subsistema de <i>caches</i> . (a) – Sem <i>cache</i> ; (b) – <i>Cache</i> externa; (c) – <i>Cache</i> interna; (d) – <i>Cache</i> interna (L1) e externa (L2); (e) – Separação da <i>cache</i> L1 em dados e instruções; (f) – Integração da <i>cache</i> L2 no processador; (g) – <i>Cache</i> L3 externa; (h) – Integração da <i>cache</i> L3 no processador	633
Fig. 7.25 - Conceito de memória virtual.....	640
Fig. 7.26 - Tradução de endereços numa memória virtual paginada.....	643
Fig. 7.27 - Tradução de endereços com tabela de páginas multinível paginada.....	647
Fig. 7.28 - Uso da tabela de páginas multinível para aceder à palavra pretendida.....	648
Fig. 7.29 - Integração da TLB na tradução de endereços. Se o número de página virtual constar da TLB, o acesso ao número de página físico é imediato, senão tem de se percorrer a tabela de páginas	649
Fig. 7.30 - Integração dos mecanismos de memória virtual e <i>caches</i>	652
Fig. 7.31 - Multiprogramação: despacho e partilha de tempo por vários processos.....	660
Fig. 7.32 - Modelo básico de estados dos processos. Só pode haver um processo em Execução. Os processos no estado Executável constam de uma lista e os que estão no estado Bloqueado constam de outra. Mudança de estado implica passagem para outra lista	661
Fig. 7.33 - Interferência na sequência temporal do processamento dos processos causada pela multiprogramação quando mais do que um processo usa a mesma informação	666
Fig. 7.34 - Um semáforo é essencialmente constituído por uma variável inteira e por uma lista dos contextos dos processos bloqueados neste semáforo (que pode estar vazia)	670
Fig. 7.35 - Divisor microprogramado	684
Fig. A.1 - Disposição dos vários <i>bits</i> no RE (Registo de Estado).....	690
Fig. B.1 - Esquema de blocos e ligações do CREPE relevantes para o Programa B.1....	715
Fig. B.2 - Fluxograma do Programa B.1	716
Fig. C.1 - Circuito a desenhar no SIMAC	720
Fig. C.2 - Circuito da Fig. C.1, desenhado no SIMAC.....	721
Fig. C.3 - Interfaces dos módulos da Fig. C.2	724
Fig. C.4 - Interface do CREPE	725

Fig. D.1 - Formatos dos números em vírgula flutuante definidos pela norma IEEE 754. (a) – Precisão simples (32 <i>bits</i>); (b) – Precisão dupla (64 <i>bits</i>).....	729
Fig. D.2 - Número -12,375 representado em vírgula flutuante. (a) – Precisão simples (32 <i>bits</i>); (b) – Precisão dupla (64 <i>bits</i>).....	730
Fig. D.3 - Tipos de valores do IEEE 754.....	732
Fig. D.4 - Estrutura (simplificada) da multiplicação em vírgula flutuante.....	733
Fig. D.5 - Estrutura (simplificada) da adição em vírgula flutuante	733