

ÍNDICE DAS TABELAS

Tabela 1.1 - Representação em binário e em decimal dos números de 0 a 15.....	10
Tabela 1.2 - Comparação entre características das pessoas e dos computadores	11
Tabela 1.3 - Fita cronológica dos factos mais relevantes na história dos computadores e dos processadores até Outubro de 2006.....	23
Tabela 2.1 - Características de algumas portas lógicas	33
Tabela 2.2 - Axiomas da álgebra de Boole.....	38
Tabela 2.3 - Exemplo de uma tabela de verdade de um sistema digital com 4 entradas (16 combinações de valores diferentes) e 2 saídas.....	39
Tabela 2.4 - Exemplo da Tabela 2.3, explicitando todos os termos mínimos possíveis. Para a função de uma saída só contam aqueles em que a saída vale 1	40
Tabela 2.5 - Número de entradas possível para um <i>multiplexer</i> para vários números de sinais de selecção.....	46
Tabela 2.6 - Tabela de verdade do decodificador de sete segmentos.....	50
Tabela 2.7 - Tabela de verdade do decodificador 1-de-4.....	51
Tabela 2.8 - Tabela de verdade do decodificador 1-de-8.....	51
Tabela 2.9 - Tabela de verdade da ROM.....	53
Tabela 2.10 - Tabela de estados do contador.....	67
Tabela 2.11 - Tabela de estados do microondas simples.....	75
Tabela 2.12 - Tabela de estados do semáforo simples.....	77
Tabela 2.13 - Tabela de estados do semáforo com botão para peões	80
Tabela 2.14 - Conteúdo da ROM no exemplo do semáforo simples.....	83
Tabela 2.15 - Comparação da forma como as pessoas e os computadores representam os números. Quando se esgotam as combinações dos símbolos, volta-se ao princípio com mais um símbolo.....	86
Tabela 2.16 - Decomposição em factores de números decimais e binários.....	86
Tabela 2.17 - Representação dos números 0 a 15 em decimal, binário e hexadecimal	88
Tabela 2.18 - Principais factores multiplicativos das potências de 2	89
Tabela 2.19 - Primeiras potências de 2.....	90

Tabela 2.20 - Exemplos de cálculo de potências de 2 com base em potências já conhecidas	91
Tabela 2.21 - Designações usuais para o tamanho (em <i>bits</i>) das representações de números e respectiva gama de valores representáveis.....	91
Tabela 2.22 - Representações de números positivos e negativos, em módulo e sinal e em complemento para 2.....	92
Tabela 2.23 - Extensão de um número binário com sinal (em complemento para 2).....	95
Tabela 2.24 - Tabela de verdade do somador elementar de 1 <i>bit</i> com transporte.....	98
Tabela 2.25 - Evolução dos registos do circuito de multiplicação no exemplo da Fig. 2.52, em que o multiplicador é 1110. O resultado final (produto) está nos registos P e M	103
Tabela 2.26 - Evolução dos registos do circuito de divisão no exemplo da Fig. 2.54, em que o divisor é 1011. O resultado final (resto e quociente) fica nos registos R e Q. O cinzento mostra a entrada sucessiva do quociente no registo Q.....	106
Tabela 2.27 - Contagem em binário e em código Gray	108
Tabela 3.1 - Operações básicas suportadas por uma RAM	115
Tabela 3.2 - Conjunto básico de operações	122
Tabela 3.3 - Funcionamento da ALU da Fig. 3.11. O <i>bit</i> 0 de SEL_ALU é o da direita (o de menor peso).....	126
Tabela 3.4 - Convenções da RTL	126
Tabela 3.5 - Operações básicas que o circuito da Fig. 3.10 permite fazer, com os valores dos sinais que controlam o circuito. A descrição aparece em texto e em RTL	127
Tabela 3.6 - Constantes usadas pelo Programa 3.1	133
Tabela 3.7 - Casos de utilização de constantes e valores dos sinais relevantes que permitem que as constantes sejam levadas até aos recursos adequados.....	138
Tabela 3.8 - Conteúdo da memória de instruções e uso da memória de dados no caso do Programa 3.2. Os valores entre parênteses são o número de <i>bits</i> que cada sinal ocupa, num total de 16 <i>bits</i> (largura da memória de instruções)	141
Tabela 3.9 - Evolução do registo A do processador e da memória de dados ao longo da execução do Programa 3.2 para N=4. Os rectângulos a cinzento indicam os endereços de instruções por onde passa o controlo em cada iteração.....	142
Tabela 3.10 - Resumo dos sinais que cada célula da memória de instruções deve conter, juntamente com o número de <i>bits</i> necessário para cada sinal.....	145
Tabela 3.11 - Conteúdo da ROM de descodificação de instruções. “x” representa um <i>bit</i> cujo valor é indiferente	149

Tabela 3.12 - Conjunto de instruções suportada pelo PEPE-8 e que constituem a sua linguagem <i>assembly</i>	151
Tabela 3.13 - Correspondência de instruções e endereços entre o algoritmo em RTL e o programa em linguagem <i>assembly</i> do Programa 3.4.....	159
Tabela 3.14 - Evolução do algoritmo de contagem de <i>bits</i> a 1 em Valor. Os <i>bits</i> a negrito são os correspondentes à posição em teste em cada iteração.....	161
Tabela 3.15 - Efeito de um acesso do processador, em leitura e em escrita, num determinado dispositivo (memória ou periférico).....	167
Tabela 3.16 - Mapa de endereços de dados do circuito da Fig. 3.26.....	172
Tabela 3.17 - Comparação das características das várias soluções de implementação de sistemas de controlo.....	176
Tabela 4.1 - Mapa de endereços.....	192
Tabela 4.2 - Mapa de utilização da memória.....	192
Tabela 4.3 - Acessos possíveis à memória no caso de processadores de 16 <i>bits</i> (Fig. 4.4a) e 32 <i>bits</i> (Fig. 4.4b) com capacidade de endereçamento de <i>byte</i>	195
Tabela 4.4 - Mapa de endereços com endereçamento por <i>byte</i>	197
Tabela 4.5 - Mapa de utilização da memória com endereçamento de <i>byte</i>	197
Tabela 4.6 - Codificação das instruções mais comuns por tipos de operandos.....	199
Tabela 4.7 - <i>Bits</i> de estado mais comuns.....	204
Tabela 4.8 - Duas formas de encarar um número binário: só valores positivos e com sinal (em complemento para 2).....	205
Tabela 4.9 - Exemplos de afectação dos <i>bits</i> de estado pela operação de adição.....	206
Tabela 4.10 - Notação RTL usada para descrever as instruções do PEPE.....	210
Tabela 4.11 - Instruções de salto do PEPE.....	212
Tabela 4.12 - Combinações possíveis dos operandos numa transferência de dados.....	215
Tabela 4.13 - Instruções de transferência de dados entre registos.....	216
Tabela 4.14 - Efeito das instruções de transferência de registos. Cada linha mostra o novo conteúdo dos registos após a execução da instrução respectiva.....	217
Tabela 4.15 - Instruções de transferência de uma constante para um registo.....	218
Tabela 4.16 - Extensão de um valor binário com sinal de 8 <i>bits</i> para 16 e 32 <i>bits</i>	219
Tabela 4.17 - Inicialização de registos com constantes de 16 <i>bits</i>	220
Tabela 4.18 - Acesso aos campos das fichas pela soma do endereço base de cada ficha com o índice de cada campo da ficha.....	226

Tabela 4.19 - Instruções de acesso à memória em 16 <i>bits</i>	228
Tabela 4.20 - Instruções de acesso à memória em 8 <i>bits</i>	229
Tabela 4.21 - Exemplos de como vencer as limitações da instrução MOV em termos de combinações de operandos. Poderiam ter sido usados outros registos quaisquer ...	235
Tabela 4.22 - Instruções aritméticas do PEPE, excepto multiplicação e divisão	236
Tabela 4.23 - Instruções aritméticas de multiplicação e divisão no PEPE	240
Tabela 4.24 - Instruções lógicas do PEPE	244
Tabela 4.25 - Codificação das letras em ASCII. As maiúsculas diferem das minúsculas apenas no <i>bit</i> 5	249
Tabela 4.26 - Utilização das instruções de manipulação de um só <i>bit</i> para alterar a codificação das letras em ASCII.....	251
Tabela 4.27 - Princípio de funcionamento das máscaras, em cada um dos seus <i>bits</i> , neutros e activos, para cada uma das operações lógicas.....	253
Tabela 4.28 - Valores dos registos e das células de memória ao longo da execução do Programa 4.19.....	256
Tabela 4.29 - Utilização da máscara XOR (3CH) para conseguir uma cifra muito simples dos caracteres (“computadores” transforma-se em “-SQLIH]XSNYO”)......	258
Tabela 4.30 - Instruções de deslocamento do PEPE.....	260
Tabela 4.31 - Modos de endereçamento mais comuns	267
Tabela 4.32 - Possíveis soluções para os modos de endereçamento não suportados pelo PEPE.....	269
Tabela 5.1 - Principais aspectos da linguagem C no Programa 5.1	280
Tabela 5.2 - Programa 5.1, em C, e as instruções correspondentes em linguagem <i>assembly</i>	282
Tabela 5.3 - Geração de endereços durante o processo de tradução de um programa em linguagem <i>assembly</i> para código-máquina. As etiquetas são opcionais. <i>Mw</i> e <i>Mb</i> significam acesso à memória em palavra e <i>byte</i> , respectivamente	289
Tabela 5.4 - Endereços atribuídos pelo <i>assembler</i> ao traduzir este programa <i>assembly</i> para código-máquina. Note-se o efeito das directivas <i>PLACE</i> no valor de <i>CE</i> (coluna da esquerda).....	290
Tabela 5.5 - Tabela de símbolos. Só os definidos com <i>EQU</i> têm o seu valor atribuído directamente pelo utilizador. Os símbolos de endereço (etiquetas) têm o seu valor atribuído pelo <i>assembler</i> , embora com base nas directivas <i>PLACE</i>	292
Tabela 5.6 - Programa 5.2, em C e em linguagem <i>assembly</i>	296

Tabela 5.7 - Conversão de alguns casos da instrução de atribuição em C para linguagem <i>assembly</i>	298
Tabela 5.8 - Programa 5.3, em C, e as instruções em linguagem <i>assembly</i> que lhe correspondem.....	310
Tabela 5.9 - Rotina <i>ordenaBolha</i> do Programa 5.4, em C, e as instruções em linguagem <i>assembly</i> que lhe correspondem	313
Tabela 5.10 - Mecanismo de chamada/retorno com o RL (registo de ligação).....	315
Tabela 5.11 - Mecanismo de chamada/retorno com o RL (registo de ligação) com mais do que um nível de chamada de rotinas.....	316
Tabela 5.12 - Programa 5.3 e as instruções em linguagem <i>assembly</i> que lhe correspondem, usando as instruções <i>CALLF</i> e <i>RETF</i> e o RL (registo de ligação). As instruções sublinhadas são as relevantes para o mecanismo de chamada/retorno...318	
Tabela 5.13 - Diferenças do mecanismo de chamada/retorno com RL (Tabela 5.12) e com pilha (Tabela 5.8) na implementação em <i>assembly</i> do Programa 5.3.....	319
Tabela 5.14 - Comparação entre os mecanismos de chamada/retorno de rotinas com endereço de retorno guardado em registo (RL) e em memória (pilha)	320
Tabela 5.15 - Analogia e diferenças entre o funcionamento da recepção de um supermercado e o mecanismos de chamada/retorno de rotinas	320
Tabela 5.16 - Instruções de chamada/retorno de rotinas com endereço de retorno na pilha	325
Tabela 5.17 - Programa com quatro rotinas, útil apenas para ilustrar o mecanismo de chamada/retorno. Este é o programa que serviu de base à Fig. 5.5	326
Tabela 5.18 - Evolução do estado de execução do programa da Tabela 5.17. Por motivos de clareza, nas quatro colunas da direita omitem-se os valores nas linhas em que a pilha ou o SP não estão envolvidos	328
Tabela 5.19 - Registos usados pela rotina <i>ordena</i> e significado dessa utilização.....	333
Tabela 5.20 - Instruções <i>PUSH</i> e <i>POP</i> . <i>Mw</i> significa acesso em palavra (16 <i>bits</i>).....	335
Tabela 5.21 - Equivalência funcional das instruções <i>CALL</i> e <i>RET</i> e sua relação com <i>JMP</i>	335
Tabela 5.22 - Evolução dos registos relevantes e da pilha após a execução de cada instrução do Programa 5.5	336
Tabela 5.23 - Comparação entre duas implementações da rotina <i>ordena</i> , sem e com guarda/recuperação dos registos usados pela rotina	339

Tabela 5.24 - Programa simples destinado apenas a ilustrar a passagem de parâmetros (por registos e pela pilha). A passagem por registos é notoriamente mais simples, e mesmo assim a passagem por pilha está errada (ver texto).....	346
Tabela 5.25 - Passagem de parâmetros e resultado pela pilha: princípio básico e duas implementações, simplista (mas errada) e correcta (embora simplificada).....	347
Tabela 5.26 - Esquema típico de chamada/retorno de uma rotina com mudança de contexto e passagem de parâmetros e resultado pela pilha.....	350
Tabela 5.27 - Versões iterativa (em cima) e recursiva da função <i>factorial</i> , em C, e as instruções em linguagem <i>assembly</i> que lhe correspondem	356
Tabela 5.28 - Execução temporal das instruções da versão recursiva da função <i>factorial</i> , em linguagem <i>assembly</i> . Apenas as alterações de valores dos registos estão indicadas (um valor mantém-se até ser alterado)	357
Tabela 5.29 - Cálculo de uma expressão em C e as instruções em linguagem <i>assembly</i> que lhe correspondem.....	360
Tabela 5.30 - Cálculo do endereço de um elemento na tabela linearizada, que é o mesmo (base + índice) seja esta encarada como uma só tabela ou como uma sequência de tabelas	368
Tabela 5.31 - Comparação das principais características das tabelas e das listas ligadas.....	381
Tabela 5.32 - Exemplo simples que ilustra o efeito de ausência de comentários, comentários inúteis e úteis.....	393
Tabela 5.33 - Exemplo de utilização de um cabeçalho.....	394
Tabela 6.1 - Operações suportadas pelos dispositivos e relação com os sinais do barramento de controlo	413
Tabela 6.2 - Mapa de endereços usado no circuito da Fig. 6.4.....	417
Tabela 6.3 - Evolução dos <i>bits</i> A14, A13 e A12 (do barramento de endereços) usados para seleccionar um dos sinais CS na saída do descodificador.....	420
Tabela 6.4 - Mapa de endereços usado no circuito da Fig. 6.4.....	423
Tabela 6.5 - Conteúdo da PROM1 para implementar o mapa de endereços da Tabela 6.4	427
Tabela 6.6 - Conteúdo da PROM2 para implementar o mapa de endereços da Tabela 6.4	428
Tabela 6.7 - Possibilidades de acesso à memória resultantes das quatro combinações dos sinais BA e A0	431
Tabela 6.8 - Mapa de endereços do circuito da Fig. 6.13, com endereçamento de <i>byte</i>	436

Tabela 6.9 - Valores na memória e nos registos após a execução das instruções anteriores, com os métodos <i>big-endian</i> e <i>little-endian</i>	439
Tabela 6.10 - Explicação dos tempos mais relevantes nos ciclos de acesso à memória/periféricos e restrições a cumprir	452
Tabela 6.11 - Principais características dos mecanismos das excepções e das rotinas normais	459
Tabela 6.12 - Hipóteses de efectuar um pedido de interrupção.....	461
Tabela 6.13 - Formato dos 8 <i>bits</i> de menor peso do RCN (Registo de Configuração do Núcleo).....	462
Tabela 6.14 - Operações a efectuar durante o ciclo completo do atendimento de uma interrupção.....	466
Tabela 6.15 - Operações elementares envolvidas numa instrução <i>SWE (Software Exception)</i>	478
Tabela 6.16 - Algumas das excepções predefinidas (em <i>hardware</i>) no PEPE.....	479
Tabela 6.17 - Comparação de algumas características de dois protocolos de comunicação série: RS232 e USB, nas suas várias versões.....	491
Tabela 6.18 - Evolução das características dos principais processadores usados nos PCs (até Outubro de 2006).....	519
Tabela 6.19 - Características de alguns microcontroladores disponíveis no mercado	534
Tabela 6.20 - Factores que influenciam o tempo de execução dos programas (considerando apenas o tempo de execução no processador).....	544
Tabela 7.1 - Codificação dos <i>bits</i> de selecção <i>SEL_RE</i> , que indicam quais os <i>bits</i> de estado que podem ser alterados directamente (sem ser pelas operações normais de escrita num registo qualquer).....	570
Tabela 7.2 - Geração de constantes de 16 <i>bits</i>	571
Tabela 7.3 - Definição da codificação das operações da ALU e de alguns sinais internos. “X” significa “tanto faz”.....	572
Tabela 7.4 - Características dos circuitos de entrada na unidade de excepções	577
Tabela 7.5 - Condições que determinam o novo valor do MPC em cada ciclo de relógio (caixa “Controlo MPC” na Fig. 7.6). “X” significa “tanto faz”	579
Tabela 7.6 - Microprograma (simbólico) e microcódigo (binário) para o circuito da Fig. 7.7. Os números por baixo dos sinais indicam o número de <i>bits</i> que cada sinal tem	583

- Tabela 7.7 - Microprogramação simbólica. A ROM de microcódigo é gerada a partir de uma tabela como esta (que representa apenas algumas microinstruções e sinais a controlar). O simulador inclui o microcódigo completo.....585
- Tabela 7.8 - Microinstruções que tratam do atendimento de excepções e de duas instruções que não existem no PEPE de base mas podem ser definidas, alterando a ROM do microcódigo, e usadas em programas de linguagem *assembly*.....587
- Tabela 7.9 - Exemplos de mapeamento entre instruções e sequências de microinstruções, a usar na ROM de mapeamento. Apenas algumas instruções estão representadas..589
- Tabela 7.10 - Adequação da microprogramação da Tabela 7.7 para processamento com cadeia de estágios. A última microinstrução de uma instrução faz o mapeamento da primeira microinstrução da próxima instrução, que já percorreu a cadeia de estágios de instruções. As microinstruções *m_BSC1* e *m_BSC2* desapareceram (deixa de ser preciso gastar tempo com a busca explícita das instruções), tal como o sinal *ESCR_RI*. As últimas três colunas foram alteradas e o resto da tabela manteve-se sem alterações.....597
- Tabela 7.11 - Adequação da microprogramação da Tabela 7.8 para processamento com cadeia de estágios. O salto para a busca de instruções (*m_BSC1*) foi substituído pelo mapeamento (com o sinal *MAP*) da próxima instrução. A microinstrução *m_SUM2* tem um mapeamento condicional. O resto da tabela manteve-se sem alterações....598
- Tabela 7.12 - Exemplo de evolução da cadeia de estágios de instruções. “X” refere-se a valores anteriores que são irrelevantes para este exemplo. A zona cinzenta em T5 e T6 indica o esvaziamento da cadeia com simulação de NOPS602
- Tabela 7.13 - Evolução detalhada ao longo do tempo e do espaço das instruções nos estágios do PEPE. A instrução *ADD*, a negrito, exemplifica a passagem de uma instrução. A zona cinzenta ilustra o esvaziamento de todos os estágios, devido a um salto. Note-se a paragem nos registos *RI* e *RMI* (operandos) durante a execução das microinstruções de mesma instrução.605
- Tabela 7.14 - Exemplo de evolução do estado de uma *cache* de mapeamento directo. Os blocos acedidos estão representados a cinzento e os alterados a negrito. As células em branco representam as linhas da *cache* vazias (*bit* de validade inactivo)624
- Tabela 7.15 - Exemplo de evolução do estado de uma *cache* de mapeamento associativo. Os blocos acedidos estão representados a cinzento e os alterados a negrito. As células em branco representam as linhas da *cache* vazias (*bit* de validade inactivo)626
- Tabela 7.16 - Exemplo de evolução do estado de uma *cache* de mapeamento associativo de duas vias. Os blocos acedidos estão representados a cinzento e os alterados a negrito. As células em branco representam as linhas da *cache* vazias (*bit* de validade inactivo).....629

Tabela 7.17 - Hierarquia de memórias de um computador, principais características de cada nível e comparação com um exemplo da vida real (biblioteca)	639
Tabela 7.18 - Informação típica contida numa entrada da tabela de páginas	646
Tabela 7.19 - Formato das entradas na tabela de páginas.....	655
Tabela 7.20 - Comparação entre as principais características dos trincos lógicos e as dos semáforos binários.....	672
Tabela A.1 - Pinos do módulo PEPE.....	690
Tabela A.2 - Descrição dos <i>bits</i> do RE (Registo de Estado).....	691
Tabela A.3 - Registos auxiliares do PEPE.....	692
Tabela A.4 - Formato do RCN (Registo de Configuração do Núcleo)	692
Tabela A.5 - <i>Bits</i> do RCCD (Registo de Configuração da <i>Cache</i> de Dados). Uma escrita neste registo invalida todo o conteúdo da <i>cache</i>	693
Tabela A.6 - <i>Bits</i> do RCCI (Registo de Configuração da <i>Cache</i> de Instruções). Uma escrita neste registo invalida todo o conteúdo da <i>cache</i>	694
Tabela A.7 - <i>Bits</i> do RCMV (Registo de Configuração da Memória Virtual)	694
Tabela A.8 - Exceções predefinidas do PEPE	696
Tabela A.9 - Descrição detalhada das instruções do PEPE	701
Tabela B.1 - Pinos do módulo CREPE.....	706
Tabela B.2 - Registos auxiliares do CREPE.....	707
Tabela B.3 - Formato do RCN (Registo de Configuração do Núcleo) do CREPE.....	708
Tabela B.4 - Formato do RCP (Registo de Configuração dos Portos de entrada/saída) ..	708
Tabela B.5 - Formato do RCT (Registo de Configuração dos Temporizadores)	710
Tabela B.6 - Formato do RCU (Registo de Configuração das UARTs).....	711
Tabela B.7 - Formas de controlar a emissão e recepção de <i>bytes</i> na UART 1. Para a UART 2, é só substituir o 1 pelo 2 nos nomes	712
Tabela B.8 - Formato do REP (Registo de Estado dos Periféricos).....	713
Tabela B.9 - Exceções predefinidas do CREPE	714
Tabela D.1 - Características da representação dos números em vírgula flutuante na norma IEEE 754.....	731
Tabela E.1 - Codificação ASCII.....	735