Repescagem - 2.º Teste de Introdução à Arquitetura de Computadores — IST – LEIC-T

1.º Semestre 2017/2018 — Duração: 60 minutos — 3 fevereiro 2018

NOME		NÚMERO	

1. (1 + 1 + 3 + 1 valores) Considere o seguinte programa em linguagem *assembly* do PEPE-16, que usa duas interrupções, embora ambas invoquem a mesma rotina (rot_int). Para facilitar, fornece-se a descrição interna das instruções CALL e RET e uma descrição sumária do mecanismo das interrupções.

CALL Etiqueta	SP ← SP-2 M[SP]←PC PC ← Endereço da Etiqueta
RET	PC ← M[SP] SP ← SP+2

Invocação de rotina de interrupção	Guarda endereço de retorno Guarda registo de estado (RE) Salta para rotina de interrupção
RFE	Repõe registo de estado (RE) Repõe endereço de retorno no PC

	Endereços				
		PLACE	2000H		
		N	EQU	15	
	2000H	contador:	WORD	0	
	2002H	tabela:	**WORD**	**0**	
	2004H		**WORD**	**rot_int**	
	2006H		**WORD**	**0**	
	2008H		**WORD**	**rot_int**	
Int		PLACE	0000H		
	0000H		**MOV**	SP, 1000H	
	0002H		MOV	**BTE, tabela**	
	0004H		MOV	R0, N	
	0006H		MOV	R1, contador	
	0008H		EI1		
	000AH		EI3		
	000CH		EI		
X	**000EH**	mais:	CALL	conta	
X	**0010H**		JMP	mais	
X	**0012H**	conta:	DI		; faz o contrário de EI
	0014H		PUSH	R0	
	0016H		MOV	R0, [R1]	
	0018H	soma1:	ADD	R0, 1	
	001AH		MOV	[R1], R0	
	001CH		EI		
X	**001EH**		POP	R0	
X	**0020H**		**RET**		
	0022H	rot_int:	**PUSH**	**R0**	; rotina usada pelas duas interrupções
	0024H		MOV	R0, 0	
	0026H		MOV	[R1], R0	
	0028H		**POP**	**R0**	
	002AH		**RFE**		

a) Preencha os endereços que faltam na coluna "Endereços" (preencha apenas as linhas em que tal faça sentido) e as instruções (ou parte delas) que faltam no programa. Considera-se que cada MOV com uma constante ocupa apenas uma palavra;

b) Na coluna mais à esquerda, "Int", coloque um "X" **apenas** nas linhas das instruções em que o PEPE, em vez de executar logo essa instrução, pode ir atender primeiro uma interrupção, se esta tiver entretanto sido pedida (num dos pinos relevantes) mas ainda não tiver sido atendida;

c) Acabe de preencher a tabela com informação sobre a sequência de todos os acessos de dados à memória feitos pelo programa, de leitura (L) ou escrita (E) durante a primeira iteração do programa principal (primeira passagem pela etiqueta "mais" e até lá voltar), assumindo que só a interrupção 3 é ativada nesse período, e precisamente durante a instrução com etiqueta "soma1".
Ignoram-se os acessos de busca de instrução. Use apenas as linhas que necessitar.
Quando o valor lido ou escrito (coluna da direita) for o registo de estado, use apenas RE como valor (uma vez que é difícil determinar o valor de todos os bits de estado).

Endereço da instrução ou n.º da interrupção que causa o acesso	Endereço acedido	L ou E	Valor lido ou escrito
000EH	0FFEH	E	0010H
0014H	0FFCH	E	15
0016H	2000H	L	0
001AH	2000H	E	1
3	0FFAH	E	001EH
3	0FF8H	E	RE
0022H	0FF6H	E	1
0026H	2000H	E	0
0028H	0FF6H	L	1
002AH	0FF8H	L	RE
002AH	0FFAH	L	001EH
001EH	0FFCH	L	15
0020H	0FFEH	L	0010H

d) Assumindo que cada iteração do programa principal demora 1 ms e que os períodos das interrupções 1 e 3 são de 100 ms e 150 ms, respetivamente, estime o valor mínimo e máximo que a variável "contador" poderá ter no momento em que a rotina de interrupção é invocada (medição efetuada ao longo de um tempo suficientemente longo). As interrupções não estão sincronizadas.

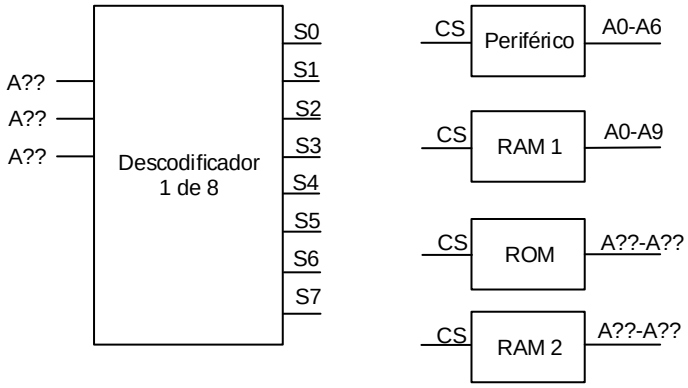
mínimo **0** Máximo **100**

2. (2 valores) Numa transmissão de dados por um barramento série assíncrono, com bit de paridade e 2 stop bits, a um ritmo de transmissão de 1000 bits/seg, indique o tempo mínimo para transmitir 30.000 bytes de dados. Justifique.

Para transmitir um byte, para além dos 8 bits de dados é preciso enviar um start bit, um bit de paridade e dois stop bits, ou seja, 12 bits no total.

Logo, para enviar 30.000 bytes é preciso enviar 360.000 bits. A 1000 bits por segundo, o tempo mínimo é 360 segundos, ou 6 minutos.

3. (3 valores) Considere o seguinte sistema de descodificação de endereços utilizado por um processador de bus de dados de 8 bits e bus de endereços de 16 bits. Preencha a informação em falta sobre o descodificador e cada dispositivo (bits de endereço a que liga, capacidade, saída do descodificador a que deve ligar e o endereço de fim da gama de endereços em que esse dispositivo está ativo, não considerando endereços de acesso repetido - espelhos).



Bits de endereço a o descodificador deve ligar **A11 .. A13**

Dispositivo	Bits de endereço	Capacidade (bytes) (decimal)	Saída do descodificador	Início (hexadecimal)	Fim (hexadecimal)
Periférico	A0-A6	**128**	**S3**	1800H	**187FH**
RAM 1	A0-A9	**1 K**	**S1**	0800H	**0BFFH**
ROM	**A0-A10**	2 K	**S4**	2000H	**27FFH**
RAM 2	**A0-A8**	512	**S0**	0000H	**01FFH**

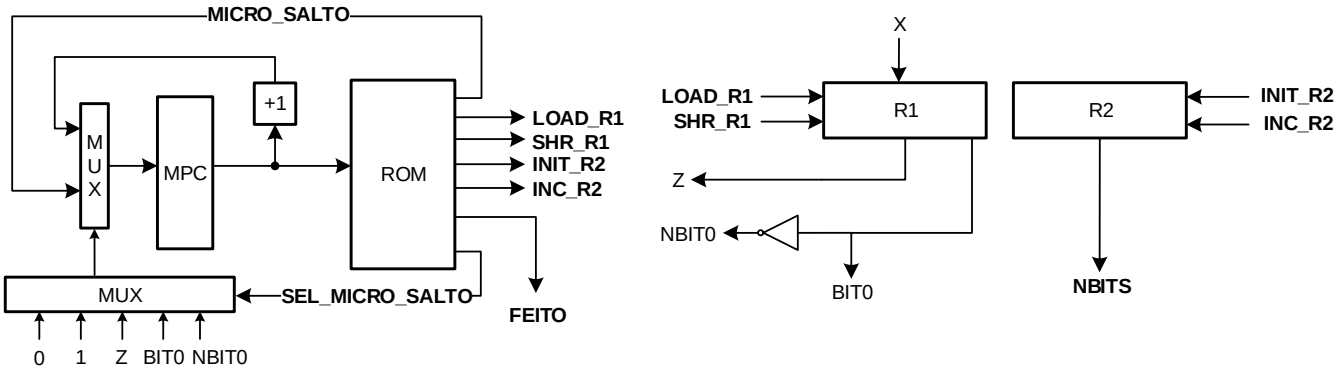
4. (2 valores) Considere a seguinte a tabela de Karnaugh, relativa a uma função de quatro entradas e uma saída. Preencha a tabela de verdade que lhe deu origem e simplifique a respetiva função, escrevendo no retângulo a expressão algébrica mais simplificada que lhe é equivalente.

A	B	C	D	Z
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**1**

		CD			
		00	01	11	10
AB	00	1	1		1
	01	1	1	1	
	11		1	1	
	10	1			1

Z = $\overline{A}\,\overline{C} + BD + \overline{B}\,\overline{D}$

5. (1,5 + 0,5 valores) Pretende-se construir um circuito microprogramado que conte o número de bits a 1 num dado número binário, X. O algoritmo é simples: se o bit 0 (menor peso) do valor X for 1, incrementa-se um contador. Segue-se um deslocamento de um bit à direita, repetindo-se estes dois passos até o valor a testar ser zero. O diagrama seguinte descreve o circuito. O registo R1 recebe o número cujos bits a 1 se devem contar. O registo tem uma saída (Z) que vale 1 quando tem o valor zero e outra (BIT0) que é igual ao bit 0 (o de menor peso) de R1. Quando o algoritmo termina (R1 fica a zero), a saída FEITO deve ficar ativa. A saída NBITS (valor de R2) indica o número de bits a 1 em X e poderá ser lida por um outro circuito.



a) Preencha a tabela seguinte com os valores necessários para implementar a funcionalidade descrita. Indique apenas os sinais ativos em cada ciclo de relógio e deixe em branco as restantes células.

Endereço na ROM	Microinstruções	FEITO	LOAD_R1	SHR_R1	INIT_R2	INC_R2	SEL_MICRO_SALTO	MICRO_SALTO
0	R1 ← X R2 ← 0		SIM		SIM			
1	(R1 == 0): MPC ← 6						Z	6
2	(BIT0 == 0): MPC ← 4						NBIT0	4
3	R2 ← R2 + 1					SIM		
4	R1 ← R1 >> 1			SIM				
5	MPC ← 1						1	1
6	FEITO ← 1 MPC ← 6	SIM					1	6

b) Quantos bits de largura deve ter no mínimo a ROM de microprograma? **11**

6. (1,5 + 1,5 valores) Suponha que a *cache* de um processador com 16 bits de endereço, endereçamento de byte, é de mapeamento direto, com uma capacidade de 512 palavras (blocos de 4 palavras).

a) Indique o número de bits de cada um dos campos em que o endereço se divide para acesso à *cache*.

Etiqueta	**6**
Índice	**7**
Palavra dentro do bloco	**2**
Byte dentro da palavra	**1**

b) Na execução de instruções do tipo MOV R1, [R2] num dado programa, verificou-se que nuns casos o valor de R1 demorava 3 ns a obter, noutros 20 ns, e que em média demorava 6,4 ns. Qual a taxa de sucesso (*hit rate*) da *cache* com este programa?

80 %

7. (2 valores) Considere um processador com 24 bits de endereço, endereçamento de byte e suporte para memória virtual com páginas de 4K bytes. Assuma que a memória física tem uma capacidade de 1 Mbyte e que a TLB é uma *cache* totalmente associativa de 4 entradas, cujo conteúdo é numa dada altura o indicado na tabela da esquerda. Acabe de preencher as outras duas tabelas para este exemplo concreto.

Posição da TLB	**Bit validade**	**Nº de página virtual**	**Nº de página física**
0	1	23BH	68H
1	0	7BAH	3AH
2	1	158H	3AH
3	1	2B5H	2AH

Dimensão do espaço virtual	**16 M**
Número de páginas virtuais	**4 K**
Número de páginas físicas	**256**

Endereço virtual	**Endereço físico**
158B6CH	3AB6CH
23BE4AH	**68E4AH**
2B5813H	2A813 H