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

1.º Semestre 2017/2018 — Duração: 60 minutos — 11 dezembro 2017

NOME		NÚMERO	

1. (1 + 1 + 3 + 1 valores) Considere o seguinte programa em linguagem *assembly* do PEPE-16, que usa duas interrupções. 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

Int	Endereços				
		N	EQU	5	
		PLACE	1000H		
		VALOR	EQU	53FH	
	1000H	Z:	WORD	0	
	1002H	tabela:	**WORD**	**0**	
	1004H		**WORD**	**0**	
	1006H		**WORD**	**B**	
	1008H		**WORD**	**A**	
Int		PLACE	0000H		
	0000H		**MOV**	SP, 2000H	
	0002H		MOV	**BTE, tabela**	
	0004H		MOV	R1, N	; durante esta instrução, todos os pinos
					; de interrupção são ativados (0 --> 1)
	0006H		MOV	R0, VALOR	
	0008H		EI2		
	000AH		EI3		
	000CH		**CALL**	itera	
X	**000EH**	fim:	JMP	fim	
	0010H	itera:	**PUSH**	**R1**	
	0012H	maisUm:	SHR	R0, 1	; desloca para a direita
	0014H		SUB	R1, 1	
	0016H		JNZ	maisUm	
	0018H		EI		
X	**001AH**		MOV	R1, Z	
X	**001CH**		MOV	[R1], R0	
X	**001EH**		POP	R1	
X	**0020H**		RET		
	0022H	A:	SHL	R0, 2	; desloca para a esquerda
	0024H		RFE		
	0026H	B:	SUB	R0, R1	; rotina da interrupção 2
	0028H		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), assumindo que todos os pinos de interrupção do PEPE são ativados (passam de 0 para 1) durante a instrução MOV R1, Z. 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
000CH	1FFEH	E	000EH
0010H	1FFCH	E	5
2	1FFAH	E	001CH
2	1FF8H	E	RE
0028H	1FF8H	L	RE
0028H	1FFAH	L	001CH
3	1FFAH	E	001CH
3	1FF8H	E	RE
0024H	1FF8H	L	RE
0024H	1FFAH	L	001CH
001CH	1000H	E	00A4H
001EH	1FFCH	L	5
0020H	1FFEH	L	000EH

d) Indique os valores (hexadecimal, 16 bits) de R0 e R1 no fim do programa.

R0 [00A4] H R1 [5] H

2. (2 valores) Considere que no seu portátil consegue executar um jogo com 30 frames/seg, em que cada frame gasta 20% do seu tempo em CPU e 80% em GPU (processador gráfico). Decide comprar um portátil novo (vem aí o Natal...), em que a CPU e a GPU são 20% e 45% mais rápidas que no seu portátil antigo. Ou seja, demoram cerca de 80% e 55%, respetivamente, do tempo que demoravam a fazer uma dada tarefa. Estime quantos frames/seg é que o portátil novo conseguirá atingir. Justifique.

No portátil atual, cada frame demora Tf:

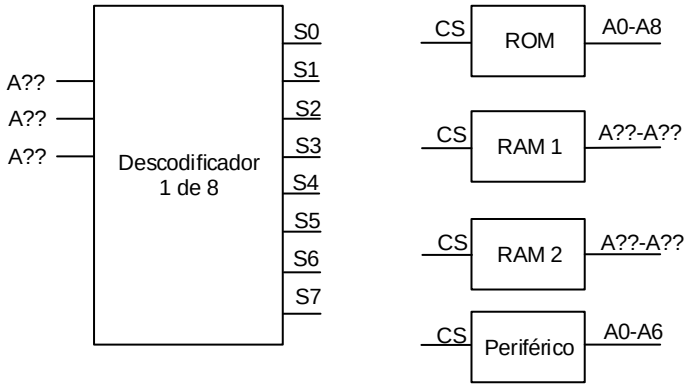
Tf = 0,2*Tf + 0,8*Tf

No portátil novo, cada frame demorará, para o mesmo jogo:

Tfnovo = (0,2*0,8)*Tf + (0,8*0,55)*Tf = 0,6 Tf

Logo, o novo tempo de cada frame é 0,6 do antigo, pelo que a melhoria frames/seg é de 1/0,6 sobre 30 frames/seg. Com o novo portátil, deverá ser possível atingir os 30/0,6 = 50 frames/seg.

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)
ROM	A0-A8	**512**	**S6**	3000H	**31FFH**
RAM 1	**A0-A9**	1 K	**S2**	1000H	**13FFH**
RAM 2	**A0-A10**	2 K	**S0**	0000H	**07FFH**
Periférico	A0-A6	**128**	**S4**	2000H	**207FH**

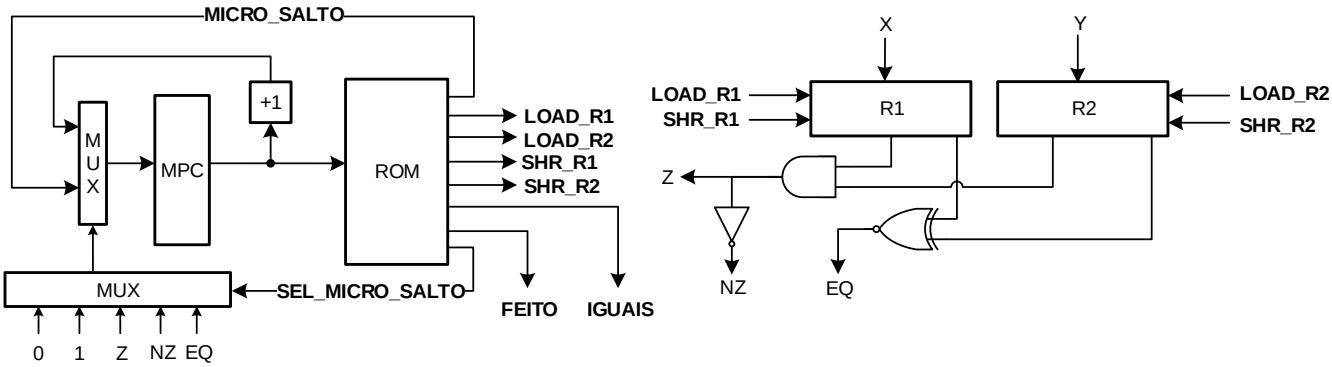
4. (2 valores) Considere a seguinte a tabela de Karnaugh, relativa a uma função de quatro entradas e uma saída. Preencha a tabela de verdade que lhe deu origem e simplifique a respetiva função, escrevendo no retângulo a expressão algébrica mais simplificada que lhe é equivalente.

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

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

Z = $A\bar{B} + AD + BC + \bar{B}D$

5. (1,5 + 0,5 valores) Pretende-se construir um circuito microprogramado que compare dois números binários, X e Y. O algoritmo de comparação é simples: comparam-se os bits 0 (menor peso) de cada um, seguido de um deslocamento de um bit à direita, repetindo-se estes dois passos até os registos serem zero. O diagrama seguinte descreve o circuito. Os registos R1 e R2 recebem os dois números a comparar. Cada registo tem uma saída que vale 1 quando o registo tem o valor zero. Os bits 0 de cada registo entram numa equivalência (ou-exclusivo negado), que faz EQ = 1 quando os dois bits são iguais. Quando o algoritmo termina (deteção de bits diferentes ou ambos os registos ficam a zero), a saída FEITO deve ficar ativa e a saída IGUAIS deve ficar ativa se os números forem iguais, inativa se forem diferentes.



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	LOAD_R1	SHR_R1	LOAD_R2	SHR_R2	FEITO	IGUAIS	SEL_MICRO_SALTO	MICRO_SALTO
0	R1 ← X R2 ← Y	SIM		SIM					
1	(R1(0) == R2(0)): MPC ← 3							EQ	3
2	FEITO ← 1 IGUAIS ← 0 MPC ← 2					SIM		1	2
3	R1 ← R1 >> 1		SIM						
4	R2 ← R2 >> 1				SIM				
5	(R1 != 0 ou R2 != 0): MPC ← 1							NZ	1
6	FEITO ← 1 IGUAIS ← 1 MPC ← 6					SIM	SIM	1	6

b) Quantos bits deve ter no mínimo o sinal SEL_MICRO_SALTO? **3**

6. (1,5 + 1,5 valores) Considere uma cache de dados de mapeamento direto, com blocos de 8 palavras cada, 4 bits de etiqueta, num processador com 16 bits de endereço com endereçamento de byte (uma palavra = dois endereços).

a) Quantos blocos tem esta cache? **256**

b) Suponha que o tempo de acesso em caso de *hit* e de *miss* é de 4 ns e 30 ns, respetivamente. Se o tempo médio de acesso for 9,2 ns, qual é a *hit rate* média?

80 %

7. (2 valores) Imagine um processador com endereçamento de byte, capaz de endereçar um espaço virtual de 00000H até FFFFFH, enquanto o espaço de endereçamento físico vai de 0000H até FFFFH. As páginas virtuais têm uma dimensão de 100H bytes. A TLB é totalmente associativa de 8 entradas e tem atualmente o conteúdo da tabela seguinte (algumas posições estão vazias, isto é, não inicializadas).

Posição da TLB	Bit validade	N.º página virtual (hexadecimal)	N.º página física (hexadecimal)
0	0	3B9	0F
1	1	207	31
2	0	2A0	3E
3	1	1EF	F1
4	1	3B9	03
5	0	0C3	1D
6	1	2A0	1B
7	0	A25	0C

Preencha a tabela seguinte para este computador e para **este conteúdo** da TLB.

Dimensão do espaço virtual (em bytes)	**1 M**
Dimensão do espaço físico (em bytes)	**64 K**
N.º páginas físicas	**256**
Endereço virtual que corresponde ao endereço físico F103H	**1EF03H**
Endereço físico que corresponde ao endereço virtual 3B90FH	**030FH**