

1.º Teste de Introdução à Arquitetura de Computadores

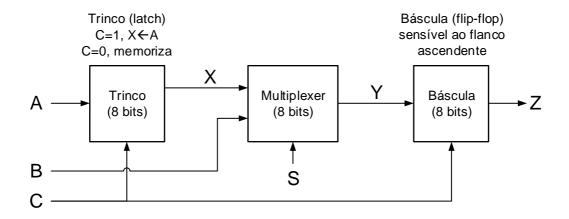
1.° Semestre 2015/2016

Duração: 60 minutos

IST – LEIC-Taguspark 29 outubro 2015

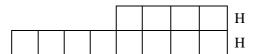
NOME	NÚMERO	

1. (3 valores) Considere o seguinte circuito, em que os sinais A, B, X, Y e Z são barramentos de 8 bits, C é o *clock* (tanto do trinco como da báscula) e S é o sinal de seleção do *multiplexer* (S=0 seleciona a entrada X). Assumindo que os sinais A, B, C e S evoluem ao longo do tempo da forma indicada na tabela seguinte, acabe de preencher o resto da tabela (escreva todas as células, mesmo que o valor se mantenha).



A	21	Н	4A	Н	73H		9CH		A4	H		58H	
В	ЕЗН	29	Н	67	Ή	C8H		85	Н	B1H			
С	1	0	0	1	1	1	0	1	1	0	0	1	0
S	0	0	0	0	1	1	1	1	0	0	0	0	0
X													
Y													
Z	25H												

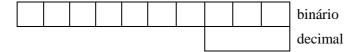
2. (2 valores) Considere o número decimal -2574. Represente-o em notação de complemento para 2, em hexadecimal com 16 e 32 bits.



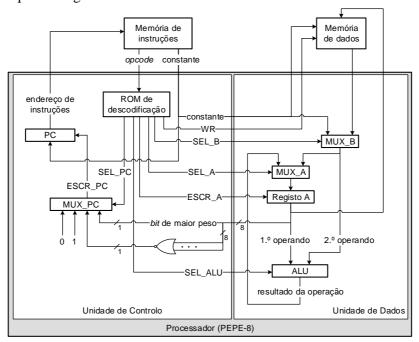
3. (2 valores) Converta para decimal o número A75H, considerando que este número está representado em notação de complemento para 2 com 12 bits.



4. (2 valores) Indique, em binário e em decimal, qual o maior número que consegue representar em notação de complemento para 2, com 10 bits.



5. (2+2 valores) A figura seguinte representa o diagrama de blocos básico do PEPE-8, processador de 8 bits, bem como as memórias a que está ligado.



a) Justifique a necessidade/utilidade da existência do multiplexer A.

b) Na tabela seguinte estão referidos os sinais usados para comandar quer a Unidade de Dados quer a Unidade de Controlo. Preencha esta tabela, especificando para cada sinal qual a indicação concreta que fornece no caso de o PEPE-8 estar a executar a instrução ADD [40H] (ou, em notação RTL, A ← A + M[40H]).

Sinal	Valor numérico, ou Qual indicação selecionada, ou Não interessa (sinal inativo)
Constante	
WR	
SEL_B	
SEL_A	
ESCR_A	
SEL_ALU	
SEL_PC	

6. (3+4 valores) Considere o seguinte programa em linguagem *assembly* do PEPE-16. Para facilitar, fornece-se a descrição interna das instruções CALL e RET.

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

Endereços **PLACE** 2000H ; início da zona de dados **INDICE EQU** 05H TABLE 20H tabela: WORD 0 pos_bit: ; variável para guardar o resultado **PLACE** ; início da zona de código 0HSP, 1000H ; inicializa todos os elementos da tabela a 78H 0010H MOV R2, INDICE 0012H **CALL** elemento ; trata o elemento n.º INDICE 0014H MOV R1, pos_bit 0016H MOV [R1], R0 ; guarda resultado na variável 0018H fim: JMP fim ; Obtém em R0 a posição do bit a 1 com maior peso no elemento da tabela ; com índice dado por R2 (exemplo: se o elemento for 37H, R0 ficará com 5) 001AH elemento: **R**1 001CH R2 001EH **R**3 MOV 0020H R3, tabela ; base da tabela 0022H SHL 0024H MOV R1, [R3+R2]; obtém elemento da tabela 0026H CALL ; obtém posição do bit de maior peso a 1 posicao 0028H 002AH 002CH RET 002EH ; Obtém em R0 a posição do bit a 1 com maior peso no valor de R1 posicao: 0030H **PUSH R**1 0032H MOV R0, -1 ; inicializa a -1 porque soma logo a seguir 0034H | maisUm: R0, 1 ; conta mais uma posição ADD 0036H ; desloca R1 à direita SHR R1, 1 0038H maisUm 003AH 003CH

a) Preencha os <u>endereços que faltam</u> (lado esquerdo, preencha apenas as linhas em que tal faça sentido) e os <u>espaços no programa</u>.

Considera-se que cada MOV com uma constante ocupa apenas uma palavra.

b) Acabe de preencher a tabela com informação sobre os acessos de dados à memória feitos pelo programa, de leitura (L) ou escrita (E). <u>Use apenas as linhas que necessitar</u>.

Considere que todos os registos estão a 0000H antes de o programa começar.

Note que no início o programa inicializa todos os elementos da tabela a 78H.

Endereço da instrução que faz o acesso	Endereço acedido	L ou E	Valor lido ou escrito
•			