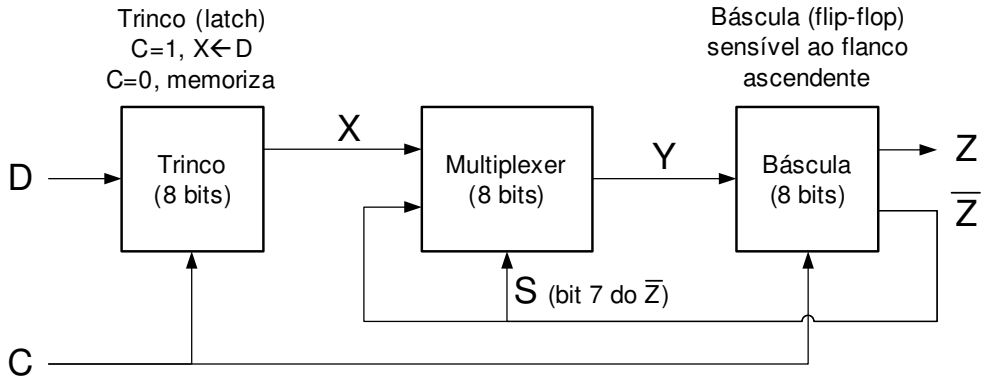
1.º Teste de Introdução à Arquitetura de Computadores — IST – LEIC-Taguspark

1.º Semestre 2016/2017 — Duração: 60 minutos — 24 outubro 2016

NOME		NÚMERO	

1. (3 valores) Considere o seguinte circuito, em que os sinais D, 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 D e C 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). Todos os valores de 8 bits estão representados em hexadecimal (não é preciso colocar o H).



D	35		7B		53		4E		B6		7D		
C	0	1	1	0	0	1	1	0	0	1	1	0	1
S													
X	B2												
Y													
Z	39												
$\overline{Z}$													

2. (2 + 2 + 2 valores) Suponha que quer somar os valores -2532 (em decimal) e C4AH (em hexadecimal, notação de complemento para 2 com 12 bits).

   a) Represente o primeiro operando em binário, 16 bits, notação de complemento para 2.

   binário

   b) Represente o segundo operando em binário, 16 bits, notação de complemento para 2.

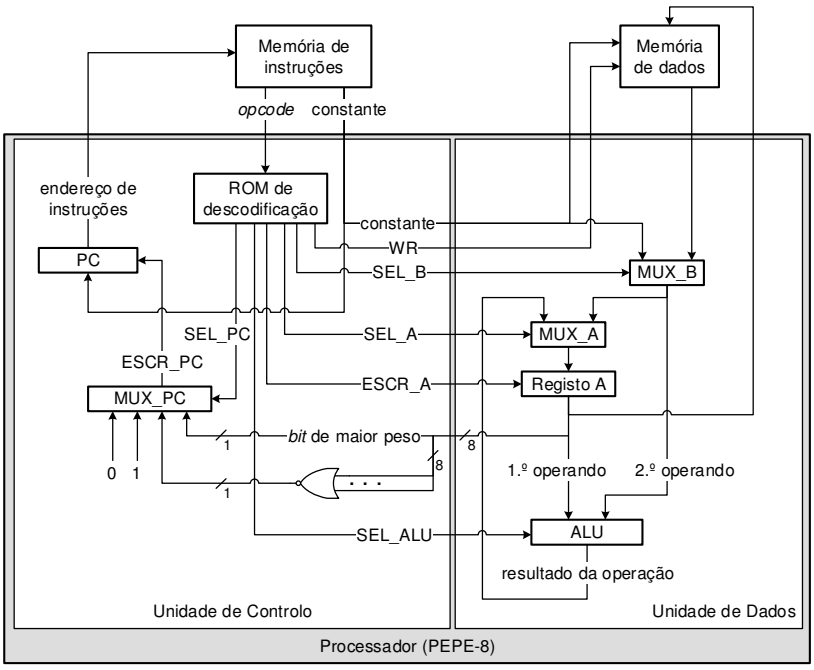
   binário

   c) Some os dois números e represente a soma em binário, mas apenas com o número mínimo de bits para o valor ser representado corretamente em notação de complemento para 2 (se for menor que 16, deixe os bits desnecessários em branco). Indique também qual é esse número mínimo de bits necessário.

   binário

   nº mínimo de bits

3. (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) 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 <u>instrução ADD 3FH</u>. Para cada sinal, use a indicação que for mais conveniente:

- Ativo / Não ativo;
- Um valor numérico;
- Uma indicação simples que especifique a opção a selecionar (ex: esquerda / direita);
- Um simples traço horizontal, ou uma cruz (não interessa para esta instrução).

Constante	WR	SEL_A	SEL_B	ESCR_A	SEL_ALU	SEL_PC

b) Idem, para a <u>instrução ST [5AH]</u>.

Constante	WR	SEL_A	SEL_B	ESCR_A	SEL_ALU	SEL_PC

4. (2 + 1 + 4 valores) Considere o seguinte programa em linguagem *assembly* do PEPE-16, que manipula os elementos de uma tabela. Para facilitar, fornece-se a descrição interna das instruções CALL e RET.

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

Endereços				
	PLACE	1000H		
	SIZE	EQU	3	; nº de elementos da tabela
	tabela:	WORD	5	; tabela
		WORD	9	
		WORD	23	
	resultado:	WORD	0	; variável para guardar o resultado
	PLACE	0000H		
			SP, 2000H	
		MOV	R1, tabela	
		MOV	R2, SIZE	
		CALL	X	
		MOV	R1, resultado	
		MOV	[R1], R3	
	fim:	JMP	fim	
	X:		Y	; X: argumentos R1, R2 - retorna R3
		DIV	R3, R2	; R3 = R3/R2
		RET		
	Y:	PUSH		; Y: argumentos R1, R2 - retorna R3
		PUSH		
		PUSH		
		MOV	R3, 0	
	ciclo:	MOV	R4, [R1]	
		ADD	R3, R4	
		ADD	R1,	
		SUB	R2,	
		JNZ	ciclo	

a) Preencha os endereços que faltam (lado esquerdo, preencha apenas as linhas em que tal faça sentido) e os espaços no programa. Considera-se que cada MOV com uma constante ocupa apenas uma palavra.

b) Indique qual a funcionalidade matemática das rotinas (em relação à tabela):

Rotina X	
Rotina Y	

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

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

Endereço em que está a instrução que faz o acesso	**Endereço acedido**	**L ou E**	**Valor lido ou escrito**