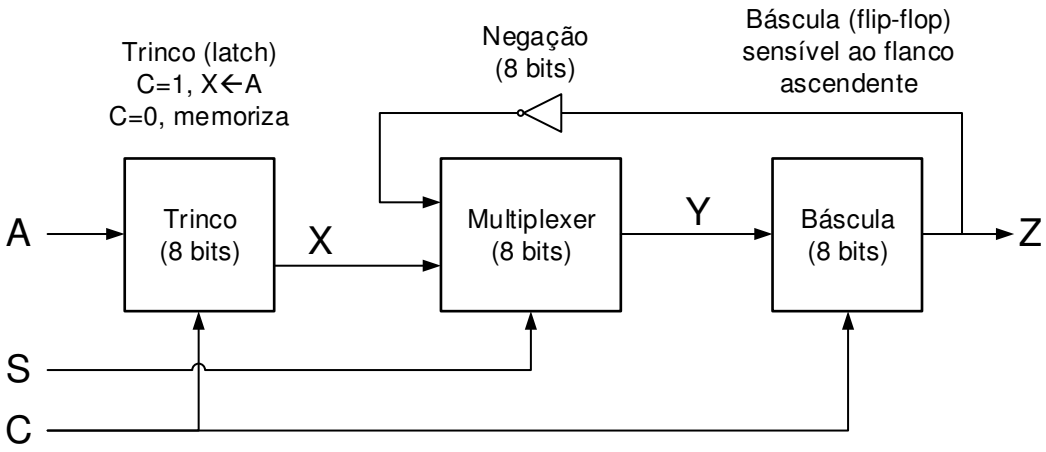
Repescagem - 1.º Teste de Introdução à Arquitetura de Computadores — IST – LEIC-T

1.º Semestre 2015/2016 — Duração: 60 minutos — 18 janeiro 2016

NOME		NÚMERO	

1. (3 valores) Considere o seguinte circuito, em que os sinais A, 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=1 seleciona a entrada X). A negação é na realidade um conjunto de 8 negações (negam todos os 8 bits do barramento). Assumindo que os sinais A, 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	27H		6BH		54H	A1H			4FH		8EH		
C	0	1	1	0	0	0	1	0	0	1	1	0	1
S	0	0	0	0	0	1	1	1	1	1	1	0	0
X	93H	**27H**	**6BH**	**6BH**	**6BH**	**6BH**	**A1H**	**A1H**	**A1H**	**4FH**	**8EH**	**8EH**	**8EH**
Y	**89H**	**76H**	**76H**	**76H**	**76H**	**6BH**	**A1H**	**A1H**	**A1H**	**4FH**	**8EH**	**5EH**	**A1H**
Z	76H	**89H**	**89H**	**89H**	**89H**	**89H**	**6BH**	**6BH**	**6BH**	**A1H**	**A1H**	**A1H**	**5EH**

2. (2 valores) Considere o número decimal −2838. Represente-o em notação de complemento para 2, em hexadecimal com 16 e 32 bits.

				F	**4**	**E**	**A**	H
F	**F**	**F**	**F**	**F**	**4**	**E**	**A**	H

3. (2 + 1 + 1 valores) Considere o número hexadecimal FC38H.

   a) Converta este número para decimal, considerando que está representado em notação de complemento para 2 com 16 bits.

   -968 decimal

   b) Represente-o agora em binário, com o número mínimo de bits necessário na notação de complemento para 2 (deixe em branco as casas que não precisar).

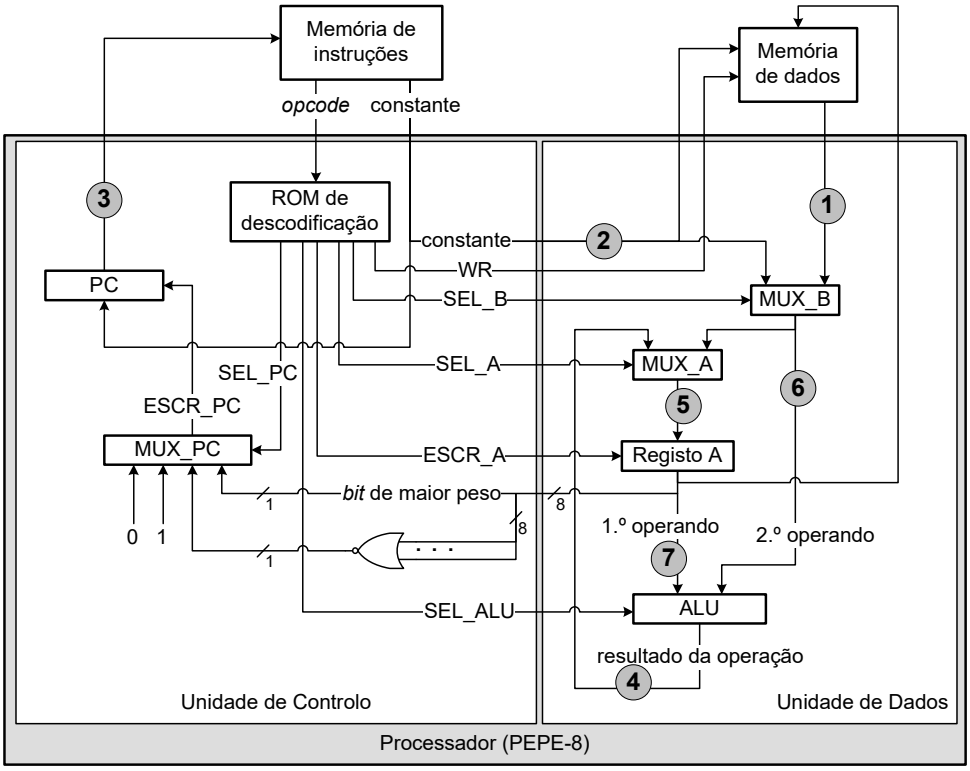
					1	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**

binário

   c) Indique agora, em decimal, qual o maior número que consegue representar na notação de complemento para 2, com esse número de bits.

   1023 decimal

4. (3 + 1 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) Cada registo é formado por 8 básculas D ativas no flanco ascendente do relógio. Suponha que o processador está a executar o programa na tabela da esquerda e neste momento o relógio está a 0 e a instrução selecionada é a ADD 31H. Os sinais de controlo preparam todos os valores para serem memorizados nos registos quando o sinal de relógio passar de 0 para 1 (o que executa a instrução ADD). Preencha a tabela seguinte, com os valores (em hexadecimal) dos sinais referenciados com os números, antes e depois de o relógio mudar de 0 para 1. Coloque X se um dado valor não puder ser determinado.

Endereço	Instrução	RTL
59H	LD 2EH	A ← 2EH
5AH	ADD 31H	A ← A + 31H
5BH	LD 45H	A ← 45H

→

	Sinais						
Relógio	1	2	3	4	5	6	7
0	**X**	**31H**	**5AH**	**5FH**	**5FH**	**31H**	**2EH**
1	**X**	**45H**	**5BH**	**A4H**	**45H**	**45H**	**5FH**

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 momento imediatamente anterior à execução da instrução ADD 31H (passagem do relógio de 0 para 1).

Sinal	**Valor numérico, Ativo/inativo, ou qual indicação selecionada**
Constante	**31H**
WR	**Inativo (não escreve na memória)**
SEL_B	**Esquerda**
SEL_A	**Esquerda**
ESCR_A	**Ativo (vai escrever no registo A)**
SEL_ALU	**Soma**
SEL_PC	**0 (não salta)**

5. (2 + 3 + 2 valores) Pretende-se fazer um programa que calcule o fatorial de um número N de forma recursiva (com uma rotina que se chama a ela própria). O objetivo é definir a rotina fatorial (N) como o produto de N por fatorial (N-1). Em *assembly* do PEPE-16, N é uma constante e o valor de N! deve ser guardado na variável "resultado".

   a) O programador Chico Esperto implementou o programa da forma indicada a seguir. Preencha os endereços do lado esquerdo e a tabela do lado direito com informação sobre os primeiros acessos à memória realizados pelo programa. Preencha apenas os espaços que forem relevantes. Considera-se que todos os MOVs ocupam apenas uma palavra. 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	
	N	EQU	4
1000H	resultado:	WORD	0
1002H	pilha:	TABLE	100H
1202H	fim_pilha:		
	PLACE	0	
0000H		MOV	SP, fim_pilha
0002H		MOV	R1, N
0004H		MOV	R2, resultado
0006H		CALL	fatorial
0008H		MOV	[R2],R1
000AH	fim:	JMP	fim
	; fatorial - Calcula o fatorial de N		
	; Entrada – R1: N		
	; Saída – R1: N!		
000CH	fatorial:	PUSH	R2
000EH		MOV	R2, R1
0010H		SUB	R1, 1
0012H		CALL	fatorial
0014H		MUL	R1, R2 ; (N-1)! * N
0016H		RET	

Endereço da instrução executada	Endereço acedido	L ou E	Valor lido ou escrito
0006H	**1200H**	**E**	**0008H**
000CH	**11FEH**	**E**	**1000H**
0012H	**11FCH**	**E**	**0014H**
000CH	**11FAH**	**E**	**0004H**
0012H	**11F8H**	**E**	**0014H**
000CH	**11F6H**	**E**	**0003H**
0012H	**11F4H**	**E**	**0014H**
000CH	**11F2H**	**E**	**0002H**
0012H	**11F0H**	**E**	**0014H**
000CH	**11EEH**	**E**	**0001H**
0012H	**11ECH**	**E**	**0014H**

b) O Chico Esperto ficou admirado por o seu programa não funcionar. Outro programador reescreveu o programa da forma indicada em baixo. Preencha de novo os endereços do lado esquerdo e a tabela do lado direito com informação sobre os acessos à memória realizados pelo programa. Preencha apenas os espaços que forem relevantes. Considera-se que todos os MOVs ocupam apenas uma palavra.

Endereços			
	PLACE	1000H	
	N	EQU	4
1000H	resultado:	WORD	0
1002H	pilha:	TABLE	100H
1200H	fim_pilha:		
	PLACE	0	
0000H		MOV	SP, fim_pilha
0002H		MOV	R1, N
0004H		MOV	R2, resultado
0006H		CALL	fatorial
0008H		MOV	[R2],R1
000AH	fim:	JMP	fim
	; fatorial - Calcula o fatorial de N		
	; Entrada – R1: N		
	; Saída – R1: N!		
000CH	fatorial:	PUSH	R2
000EH		CMP	R1, 1
0010H		JLE	acabou
0012H		MOV	R2, R1
0014H		SUB	R1, 1
0016H		CALL	fatorial
0018H		MUL	R1, R2 ; (N-1)! * N
001AH		JMP	sai
001CH	acabou:	MOV	R1, 1
001EH	sai:	POP	R2
0020H		RET	

Endereço da instrução executada	Endereço acedido	L ou E	Valor lido ou escrito
0006H	**1200H**	**E**	**0008H**
000CH	**11FEH**	**E**	**1000H**
0016H	**11FCH**	**E**	**0018H**
000CH	**11FAH**	**E**	**0004H**
0016H	**11F8H**	**E**	**0018H**
000CH	**11F6H**	**E**	**0003H**
0016H	**11F4H**	**E**	**0018H**
000CH	**11F2H**	**E**	**0002H**
001EH	**11F2H**	**L**	**0002H**
0020H	**11F4H**	**L**	**0018H**
001EH	**11F6H**	**L**	**0003H**
0020H	**11F8H**	**L**	**0018H**
001EH	**11FAH**	**L**	**0004H**
0020H	**11FCH**	**L**	**0018H**
001EH	**11FEH**	**L**	**1000H**
0020H	**1200H**	**L**	**0008H**
0008H	**1000H**	**E**	**24**

c) O novo programa já funciona? Compare os dois e/ou explique que erros o Chico Esperto cometeu.

O novo programa já funciona. O Chico Esperto cometeu dois erros básicos:

1 – A rotina fatorial, recursiva, não tem condição de paragem da recursividade. Assim, chama-se a ela própria de forma infinita e nunca termina (até que o programa deixe de funcionar, por utilização da pilha para além da sua área reservada. Chega mesmo a escrever nos endereços do programa!)

2 – O PUSH R2 não tem o correspondente POP R2 (embora neste caso o programa nunca lá chegue)

O novo programa já tem estes dois problemas resolvidos.