ISRC -- Resolução do Teste 2 – 8 de Janeiro de 2010

Pergunta 1

Apresenta-se uma das soluções possíveis; as linhas a itálico já estavam preenchidas no enunciado

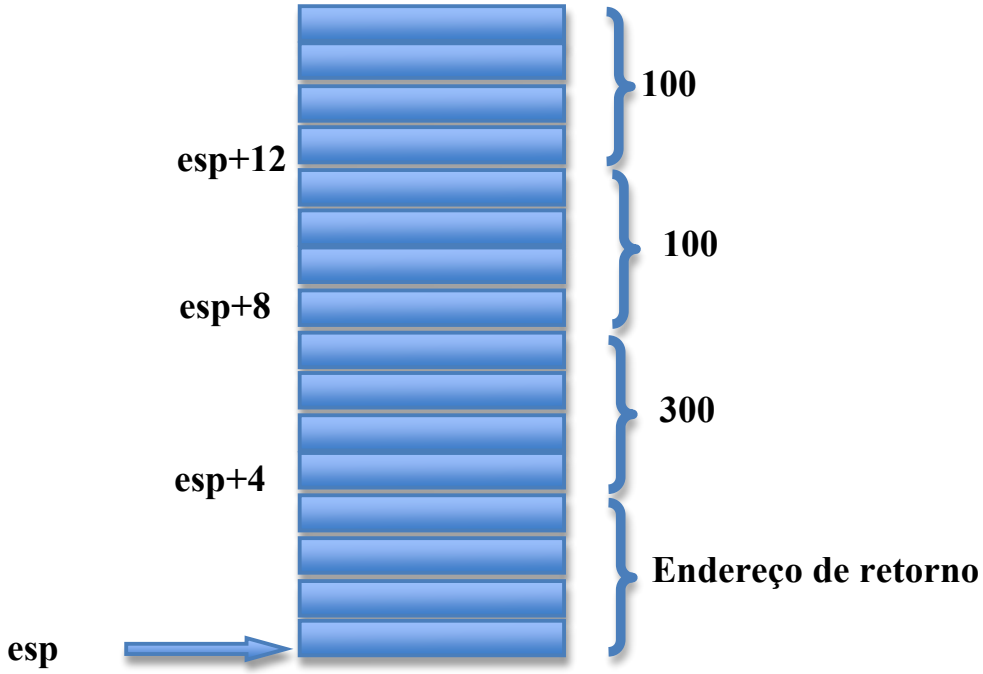
Etiqueta	***Instrução***	***Comentário***
Loop,	*Input*	--
	Skipcond 800	--
	Jump End	Se o número introduzido é 0, termina
	Store Temp	Guarda o nº introduzido na posição de memória Temp
	Load Max	Carrega o maior valor introduzido até ao momento
	Sub Temp	Acumulador = Acumulador – conteúdo de Temp
	Skipcond 000	Se o acumulador é < 0 o valor introduzido é maior do que o máximo corrente; é preciso saltar a instrução seguinte para guardar este valor como novo máximo
	Jump Loop	Não era um novo máximo ...
	Load Temp	Recupera o valor introduzido
	Store Max	Guarda em Max o valor introduzido
	Jmp Loop	Vai ler mais um valor
End,	*Load Max*	--
	Output	--
	Halt	--
Max,	*dec 0*	--
Temp,	*dec 0*	--

Pergunta 2

a) As primeiras instruções preenchem eax e ebx de acordo com a tabela seguinte

Etiqueta	**Instrução**	**eax**	**ebx**	**Comentários**
_start:	mov eax, 0x200	0x200	?	
	mov ebx, 0x100	0x200	0x100	
	mov ecx, 0x300	0x200	0x100	
	sub eax, ebx	0x100	0x100	eax = eax - ebx
	jl lab1	0x100	0x100	Salta se eax(200)<ebx(100).Falso
	push eax	0x100	0x100	Empilha o valor 0x100
lab1:	push ebx	0x100	0x100	Empilha o valor 0x100
	push ecx	0x100	0x100	Empilha o valor 0x300
	call myfunc	0x100	0x100	Empilha endereço de retorno e vai para myfunc

Quando se entra na rotina myfunc o “stack” tem o seguinte conteúdo:



Quando se vai executar a função myfunc:

Etiqueta	Instrução	eax	ebx	Comentário
myfunc	mov eax, [esp+8]	0x100	0x100	Ver figura
	mov ebx, [esp+4]	0x100	0x300	Ver figura
	sub ebx, eax	0x100	0x200	ebx = ebx (0x300) – eax (0x100)
	mov [temp], eax	0x100	0x200	
	ret	0x100	0x200	

O conteúdo da posição *temp* quando a rotina termina é 0x100

b)

Etiqueta	Instrução	Comentário
myfunc:	mov ebx,[esp+4]	ebx recebe o parâmetro de entrada
	mov eax, 0	eax vai acumular o valor
l1:	add eax, ebx	Mais uma soma
	dec ebx	Passa ao valor inferior
	jnz l1	Se ainda não é 0 continuar a acumular
	ret	

Pergunta 3

a) As instruções privilegiadas são aquelas que o CPU só executa se estiver em modo supervisor, Exemplos são a manipulação do sistema de interrupções, as instruções de entrada saída, as que manipulam o hardware de protecção de memória.

b) Quando está a ser executado código do sistema o CPU está em modo supervisor; quando está a ser executado o código de um programa lançado por um utilizador o CPU está em modo utilizador. Esta organização impede que o código utilizador execute instruções privilegiadas; se isso fosse possível o utilizador poderia comprometer o bom funcionamento do sistema, por exemplo desligando as interrupções.

c) Ao executar a instrução que é uma interrupção por software, o PC é carregado com um valor que está guardado no vector de interrupções. Esse valor aponta para o código do sistema que analisa e executa o pedido feito pelo utilizador. Antes de se executar a interrupção por software o CPU está em modo utilizador; quando ocorre a interrupção por software, o CPU muda para modo supervisor, o que corresponde ao facto de passar a executar código do sistema.

Pergunta 4

1	Elem. Neutro
2	Complemento ou inversa
3	Distributiva
4	Idempotência
5	Associativa
6	Comutativa
7	Distributiva
8	Complemento ou inversa
9	Elem. Neutro

Pergunta 5 Apresenta-se uma das soluções possíveis

((xy')+(x'z)+(yz)')'	Expressão negada
(xy')'(x'y)'(yz)''	Lei de Morgan
(x'+y'')(x''+y')(yz)	Lei de Morgan + Dupla negação
(x'+y)(x+y')(yz)	Dupla negação (2x)
(x'x+x'y'+yx+yy')(yz)	Distributiva
(0+x'y'+yx+0)(yz)	Complemento
(x'y'+yx)(yz)	Elem. Neutro
x'y'yz+yxyz	Distributiva
yxyz	Complemento
xyz	Idempotência

Pergunta 6

wx \ yz	00	01	11	10
00	1	X	1	X
01		1		1
11				
10		1		1

Resp1: F=w'x'+(w xor x)(y xor z)

Resp2: F=w'x'+w'y'z+w'yz'+x'y'z+x'yz'

Pergunta 7

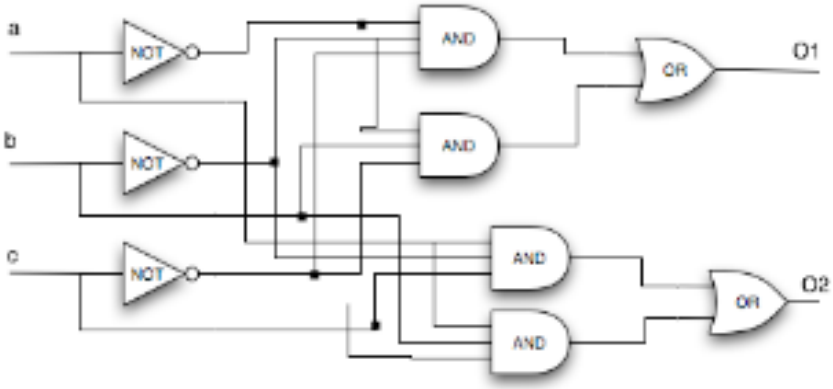
a)

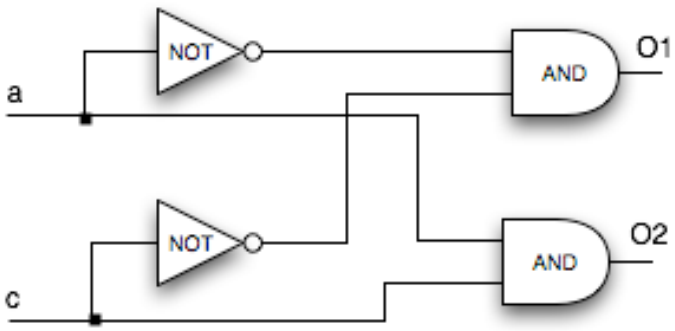
Complem2	a	b	c	O1	O2
0	0	0	0	1	0
+1	0	0	1	0	0
+2	0	1	0	1	0
+3	0	1	1	0	0
-4	1	0	0	0	0
-3	1	0	1	0	1
-2	1	1	0	0	0
-1	1	1	1	0	1

O1= a'b'c'+a'bc'= a'c'(b'+b)=a'c'

O2= ab'c+abc= ac(b'+b)=ac

b) Apresentam-se duas versões do circuito, uma é antes da simplificação e outra depois





Pergunta 8

a) A = descodificador
B = half adder
C = full adder

b) para f0=1; f1=0 a operação realizada é OR (OU lógico) entre entradas A e B, resultado em C. Apenas o terceiro AND do descodificador terá a saída a 1, o que permite, ao entrar nas portas AND que ligam às portas OR, realizar A0 OR B0 e A1 OR B1. As saídas dessas portas AND seguem para o output, respectivamente C0 e C1.

Pergunta 9

a) Edge triggered (activado na fronteira). O estado do circuito só muda na fonteira entre estados do relógio. Neste circuito, tal acontece apenas na subida do sinal do relógio, nunca quando o relógio está em qualquer outro estado, mesmo que as entradas J e K mudem.

b) Q = 1

c) Não. No intervalo de t2, K varia de 1 para 0 e novamente para 1, mas essas alterações nunca ocorrem na subida do estado do relógio, daí não terem qualquer efeito no circuito (é edge triggered na subida). Quando se dá a subida do sinal do relógio, K já está novamente a 1.

Pergunta 10

Por favor ver a figura da questão 10 do teste 2.
A tabela com a evolução dos FFs ao longo do tempo, está descrita em baixo.
Têm-se em conta os seguintes pressupostos:
-- X' = NOT(X).
-- A figura representa um circuito sequencial síncrono, i.e. ambos os FFs mudam ao "mesmo tempo"/são sensíveis à mesma transição de relógio, e neste caso, podemos considerar que, por exemplo, são ambos "edge-triggered" com transição de relógio 0->1.
-- (J1,K1) representa as entradas do FF1, e Q1 a saída deste FF. O mesmo se aplica a (J0,K0) e Q0 em relação a FF0.
-- O instante (t0+1) representa a situação em que já ocorreu uma transição de relógio à qual os FFs são sensíveis – a passagem do tempo t0 ao tempo (t0+1) corresponde a um ciclo do relógio. Assim, o estado das entradas dos FFs em t0 (i.e. J1 e K1, e J0 e K0), determina qual vai ser o estado dos FFs em (t0+1). O mesmo se passa sobre (t0+2) em relação a (t0+1), etc.
-- Assim que o estado dos FFs muda, considera-se que a propagação pelos circuitos combinatórios associados é "instantânea", pelo que eles estabilizam, de acordo com a álgebra de Boole, antes da próxima transição de relógio. Ou seja, os valores das entradas dos FFs deste circuito em particular são definidas pelas expressões Booleanas:

J0 = Inp
K0 = Inp' + Q1'
J1 = Inp.Q0
K1 = Inp' + Q0
-- consequentemente, depois de calcular (J1,K1) de acordo com as expressões acima, i.e. quais os valores de (J1,K1) em t0, estes vão então definir o valor de Q1 em (t0+1);
-- do mesmo modo, o cálculo de (J0,K0) em t0 vai definir o valor de Q0 em (t0+1)

-- Como se afirma que no instante t0 (i.e. no ciclo de relógio corrente, i.e. antes da próxima transição de relógio à qual os FFs são sensíveis) Inp passa de zero a um, temos que em t0 Inp=1.
Assim sendo, dados os valores Q1=0 e Q=0 no tempo t0, bem como Inp=1, e considerando a "propagação instantânea", os valores das entradas dos FFs em t0 são: J1=0, K1=0, J0=1, K0=1.

Temos que a Tabela com evolução de ambos os FFs é:

	t0	**t0+1**	**t0+2**	**t0+3**
Inp	1	1	1	1
J1	0	1	0	1
K1	0	1	0	1
Q1	0	0	1	1
Q1'	1	1	0	0
Inp = J0	1	1	1	1
K0	1	1	0	0
Q0	0	1	0	1
Q0'	1	0	1	0

Como é sabido, o próximo estado de um circuito sequencial depende do estado anterior e das entradas, se existirem e, de acordo com o que foi dito acima, as entradas J, K dos FFs resultam da propagação "instantânea" dos valores do estado do circuito durante um determinado ciclo do relógio (i.e. antes da próxima transição significativa). Assim, e como se pode ver observando, por exemplo, a coluna da tabela t0+1, temos

J1= Inp and Q0 = 1 and 1 = 1,

logo, J1 é um em t0.
Observando a tabela, pode-se então concluir que o valor de Q1 e Q0 ao fim de três ciclos do relógio é Q1=1 e Q0=1.

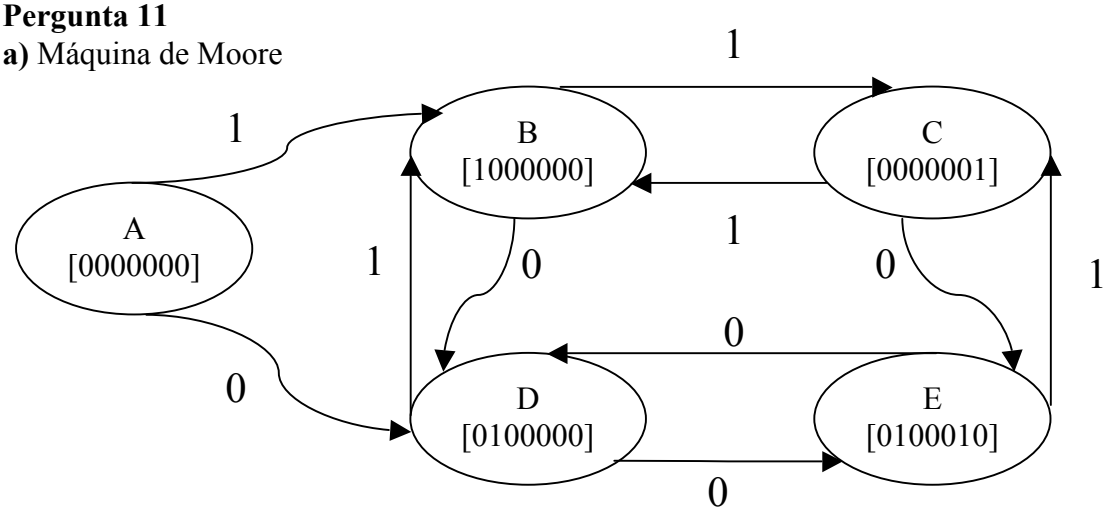
Outra maneira de registar a evolução dos FFs, por ventura mais rápida, é utilizando a tabela de transição de estados do FF tipo JK. Ou seja, considerando que, quando o estado (Q) de um FF tipo JK é zero, só interessa olhar para a sua entrada J, para se saber qual o estado seguinte desse FF; do mesmo modo, quando o estado de um FF tipo JK é um, basta olhar para a entrada K, para se saber qual o estado seguinte desse FF. Assim, pode-se construir a tabela:

	$Q^n1\ Q^n0$	**Inp**	**$Q^{n+1}1\ Q^{n+1}0$**	**J1 K1**	**J0 K0**
t0	0 0	1	0 1	0 X	1 X
t0+1	0 1	1	1 0	1 X	X 1
t0+2	1 0	1	1 1	X 0	1 X
t0+3	1 1	1	0 1	X 1	X 0

Obviamente, o cálculo de J1, K1, J0 ou K0, faz-se de acordo com as expressões Booleanas acima definas.

Pergunta 11

a) Máquina de Moore



Nota: na máquina de Moore, define-se, em cada estado, qual a saída/*output* da máquina nesse estado. Neste caso, a máquina em questão tem uma entrada, e 7 saídas, às quais correspondem as funções que definem o valor de cada um dos segmentos do LED de 7 segmentos. Assim, [1000000] pretende significar que o segmento sA está a um (i.e. a função para esta saída deve ser 1 nesse estado particular) e que os restantes estão a zero; [0000001] pretende significar que o segmento sG está a 1 e todos os outros têm de estar a zero, etc.

b) Número de FFs (JK ou D) necessários:

Em geral, a determinação do número de FFs necessários é em função do número de estados nos quais a máquina a implementar pode estar, uma vez que é necessária uma codificação única para cada estado que o permita distinguir dos outros estados. A memória da máquina (implementada através de FFs, permitindo cada um guardar/representar 1 bit) guarda então a codificação de um estado (de entre os que são possíveis para um problema particular). Relembra-se ainda que, cada estado permite identificar/definir, dado o valor (zero ou um) da(s) entrada(s) -- ou até pode nem existir nenhuma entrada no circuito -- qual deve ser o estado seguinte para o qual a máquina deve transitar.

A regra geral é então que o número k de FFs necessários tem de ser **maior ou igual** que o $\mathbf{log_2(n)}$ em que **n** é o número de estados (uma vez que k FFs permitem representar/codificar $2^k >= n$ estados).

Assim, para o exemplo particular acima, e para representar todos os estados, incluindo o inicial, são necessários 3 FFs. No entanto, por simplificação, poder-se-ia ignorar o estado inicial A (sendo este considerado como o estado em que o circuito não está em execução), e seriam então apenas necessários 2 FFs para representar os restantes estados.