1.

a. A gama de variação de um número em complemento para 2 é:

$-2^{k-1} \leq n \leq 2^{k-1} - 1$, em que **k** é o número de bits do número representado.

Assim sendo, para n = +60, tem que se verificar a condição

$+60 \leq 2^{k-1} - 1 \Rightarrow k \geq \log_2 (60 + 1) + 1 = 7$.

De igual modo, para n = -31, tem-se que,

$-2^{k-1} \leq -31 \Rightarrow k \geq \log_2 (31) + 1 = 6$.

Como se pretende representar ambos os números com o mesmo de bits, então o menor número de bits com que se consegue representar A e B é 7.

Deste modo $+60_{10}$ = 0 111100, em que o 0 sublinhado representa o sinal (positivo) do número. Por outro lado, para se achar a representação em complemento para 2 de –31, primeiro acha-se a sua representação de $+31_{10}$, que é 0 011111; para se achar o complemento para 2, faz-se 10000000 – 0011111 = 1 100001. Deste modo:

A = $+60_{10}$ = 0 111100, em complemento para 2

B = -31_{10} = 1 100001, em complemento para 2.

i. A + B

$$\begin{array}{r} 0111100 \\ +1100001 \\ \hline 10011101 \end{array}$$

em que o 1 mais à esquerda é ignorado, logo, A + B = 0011101 ($\Leftrightarrow +29_{10}$), o que dá o resultado correcto.

ii. A – B

Primeiro tem que se "transformar" – B em + (-B), o que equivale a ter

-B = 0011111 (-31_{10}).

Agora, soma-se,

$$\begin{array}{r} 0111100 \\ +0011111 \\ \hline 1011011 \end{array}$$

Neste caso a soma de dois números positivos (A e –B), dá, como resultado, um número negativo; tal justifica-se, pois A - B = $+91_{10}$, que não pode ser representado só com 7 bits em complemento para 2; diz-se, então, que o resultado está fora da gama de valores para 7 bits em complemento para 2, ou, mais correntemente, que ocorreu um erro de *overflow*.

b. A gama de variação de um número em complemento para 1 é:

$-(2^{k-1} - 1) \leq n \leq 2^{k-1} - 1$, em que **k** é o número de bits do número representado.

Assim sendo, para n = +98, tem que se verificar a condição

$+98 \leq 2^{k-1} - 1 \Rightarrow k \geq \log_2 (98 + 1) + 1 = 8$.

De igual modo, para n = -31, tem-se que,

$-(2^{k-1} - 1) \leq -55 \Rightarrow k \geq \log_2 (55 + 1) + 1 = 7$.

Como se pretende representar ambos os números com o mesmo de bits, então o menor número de bits com que se consegue representar C e D é 8.

Deste modo $+98_{10}$ = 0 1100010, em que o 0 sublinhado representa o sinal (positivo) do número. Por outro lado, para se achar a representação em complemento para 1 de –55, primeiro acha-se a sua representação de $+55_{10}$, que é 0 0110111; para se achar o complemento para 1, faz-se 11111111 – 00110111 = 1 1001000. Deste modo:

C = $+98_{10}$ = 0 1100010, em complemento para 1

D = -55_{10} = 1 1001000, em complemento para 1.

i. C + D

$$\begin{array}{r} 01100010 \\ +11001000 \\ \hline 100101010 \\ +1 \\ \hline 00101011 \end{array}$$

logo, C + D = 00101011 ($\Leftrightarrow +43_{10}$), o que dá o resultado correcto.

ii. D – C

Primeiro tem que se "transformar" – C em + (-C), o que equivale a ter

-C = 10011101 (-98_{10}).

Agora, soma-se,

$$\begin{array}{r} 11001000 \\ +10011101 \\ \hline 101100101 \\ +1 \\ \hline 01100111 \end{array}$$

Neste caso a soma de dois números negativos (D e –C), dá, como resultado, um número positivo; tal justifica-se, pois D - C = -153_{10}, que não pode ser representado só com 8 bits em complemento para 1; diz-se, então, que o resultado está fora da gama de valores para 8 bits em complemento para 1, ou, mais correntemente, que ocorreu um erro de *overflow*.

2.

a. Atendendo ao diagrama de estados, começa-se por se fazer uma tabela de transições:

Diagrama de estados:

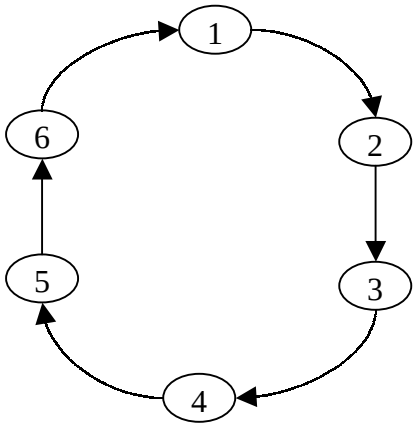
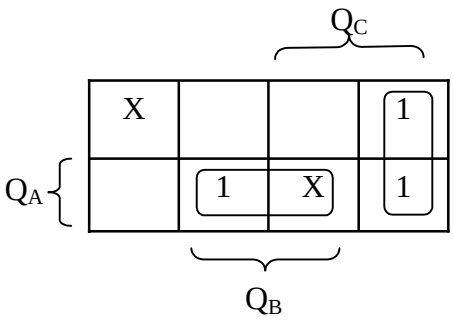


Tabela de transições:

Estado actual			Estado seguinte		
Q_C	Q_B	Q_A	Q^*_C	Q^*_B	Q^*_A
0	0	0	X	X	X
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	X	X	X

Simplificando, para Q^*_C, vem:



Donde

$$Q^*_C = Q_C \cdot \overline{Q_B} + Q_B \cdot Q_A$$

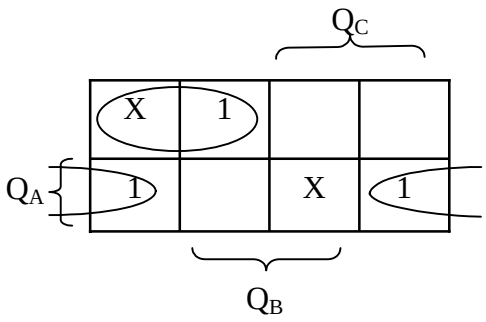
A passagem para flip-flop's tipo JK é dada pala seguinte relação:

$$Q^* = j_C \cdot \overline{Q} + \overline{k_C} \cdot Q \ ,$$

donde

$$Q^*_C = j_C \cdot \overline{Q_C} + \overline{k_C} \cdot Q_C = Q_C \cdot \overline{Q_B} + Q_B \cdot Q_A \cdot (Q_C + \overline{Q_C})$$
$$= Q_C \cdot \overline{Q_B} + Q_B \cdot Q_A \cdot Q_C + Q_B \cdot Q_A \cdot \overline{Q_C} = Q_C \cdot (\overline{Q_B} + Q_B \cdot Q_A) + Q_B \cdot Q_A \cdot \overline{Q_C}$$
$$= Q_C \cdot (\overline{Q_B} + Q_A) + Q_B \cdot Q_A \cdot \overline{Q_C} \Leftrightarrow$$
$$\Leftrightarrow \begin{cases} j_C = Q_B \cdot Q_A \\ \overline{k_C} = \overline{Q_B} + Q_A \Leftrightarrow k_C = Q_B \cdot \overline{Q_A} \end{cases}$$

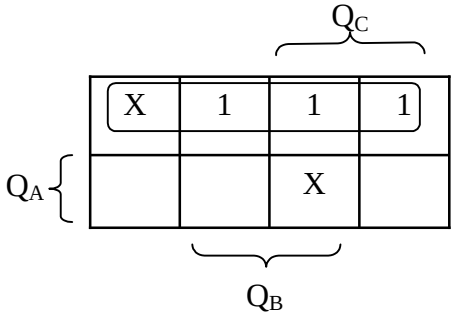
De igual modo, para Q^*_B, vem:



Donde

$$Q^*_B = \overline{Q_B} \cdot Q_A + \overline{Q_C} \cdot \overline{Q_A}$$
$$Q^*_B = j_B \cdot \overline{Q_B} + \overline{k_B} \cdot Q_B = \overline{Q_B} \cdot Q_A + \overline{Q_C} \cdot \overline{Q_A} \cdot (Q_B + \overline{Q_B})$$
$$= \overline{Q_B} \cdot Q_A + \overline{Q_C} \cdot \overline{Q_A} \cdot Q_B + \overline{Q_C} \cdot \overline{Q_A} \cdot \overline{Q_B} = \overline{Q_B} \cdot (Q_A + \overline{Q_C} \cdot \overline{Q_A}) + Q_B \cdot \overline{Q_A} \cdot \overline{Q_C}$$
$$= Q_B \cdot (Q_A + \overline{Q_C}) + Q_B \cdot \overline{Q_A} \cdot \overline{Q_C} \Leftrightarrow$$
$$\Leftrightarrow \begin{cases} j_B = Q_A + \overline{Q_C} \\ \overline{k_B} = \overline{Q_A} \cdot \overline{Q_C} \Leftrightarrow k_B = Q_A + Q_C \end{cases}$$

Por último, para Q^*_A, vem:



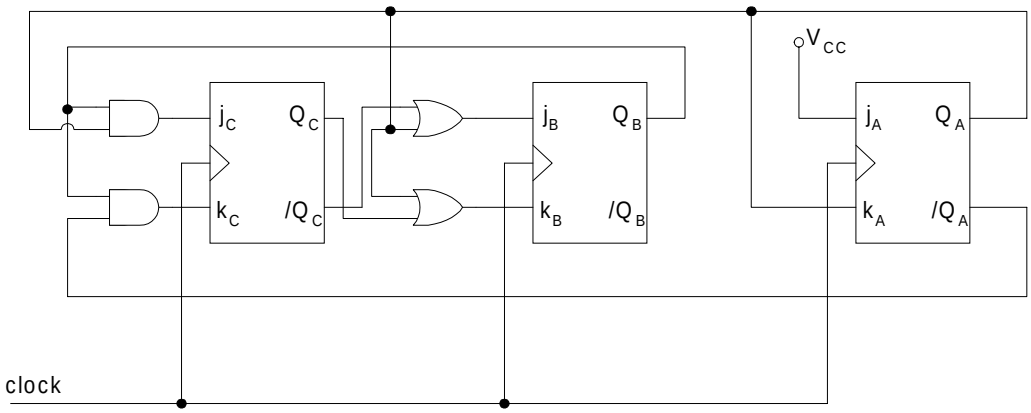
Donde

$$Q^*_A = \overline{Q_A}$$

donde

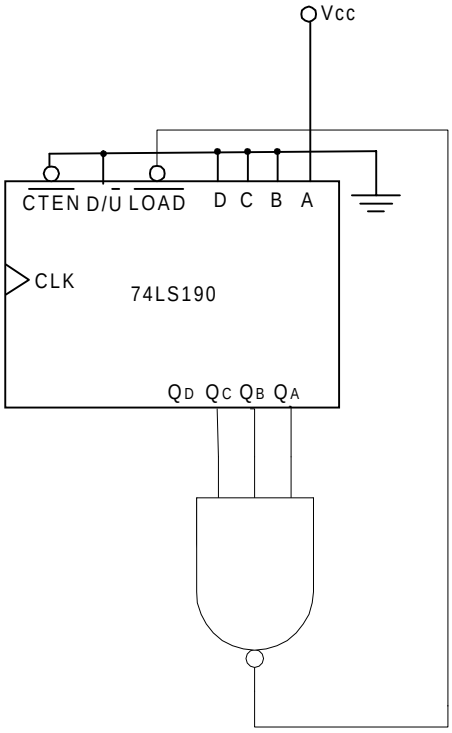
$$Q^*_A = j_A \cdot \overline{Q_A} + \overline{k_A} \cdot Q_A = 1 \cdot \overline{Q_A} + \overline{Q_A} \cdot Q_A \Leftrightarrow$$
$$\Leftrightarrow \begin{cases} j_A = 1 \\ \overline{k_A} = \overline{Q_A} \end{cases} \Leftrightarrow k_A = Q_A$$

Destas equações resulta o circuito seguinte:

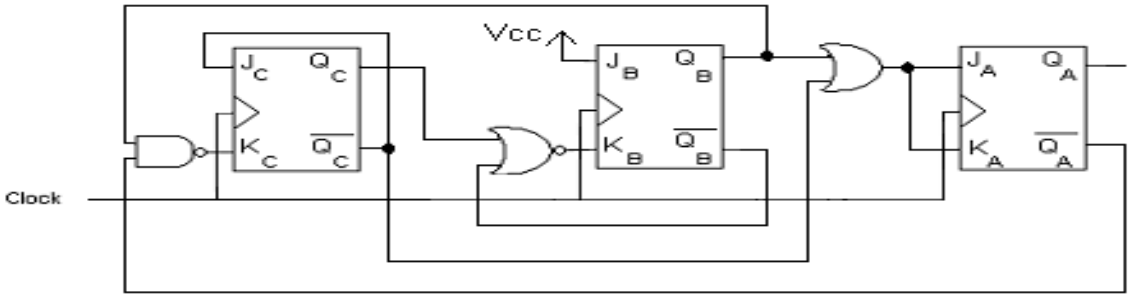


b. O contador 74LS190 tem uma entrada /LOAD assíncrona, que, sempre que é activada, coloca, na saída do contador, o valor das entradas D, C, B e A, correspondendo a entrada D ao bit mais significativo e a entrada A ao bit menos significativo. Deste modo, pretende-se que sempre que termine o estado 6 (ou seja, mal se inicie o estado 7) o contador seja colocado no estado 1.
Para isso, por um lado, é necessário detectar o estado 7; tal pode ser feito com uma porta AND de 3 entradas ligadas às saídas Q_C, Q_B e Q_A; quando estas saídas forem simultaneamente 1 (correspondente ao número $111_2 = 7_{10}$), então está-se perante o estado 7 (na realidade quando $Q_C Q_B Q_A = 111$ pode corresponder a duas situações, dependendo de Q_D; se este for 0 então $0111_2 = 7_{10}$; se for 1 então $1111_2 = 15_{10}$; contudo esta última hipótese nunca ocorrerá, se o circuito for bem projectado, pois o maior valor que o contador pode atingir é 7, logo é dispensável a ligação a Q_D).
Há, também, a referir, que a entrada /LOAD é activa ao nível lógico baixo, logo, para activar esta entrada, o AND referido não terá os efeitos pretendidos, pois a sua saída será 1 quando as suas entradas forem 111;

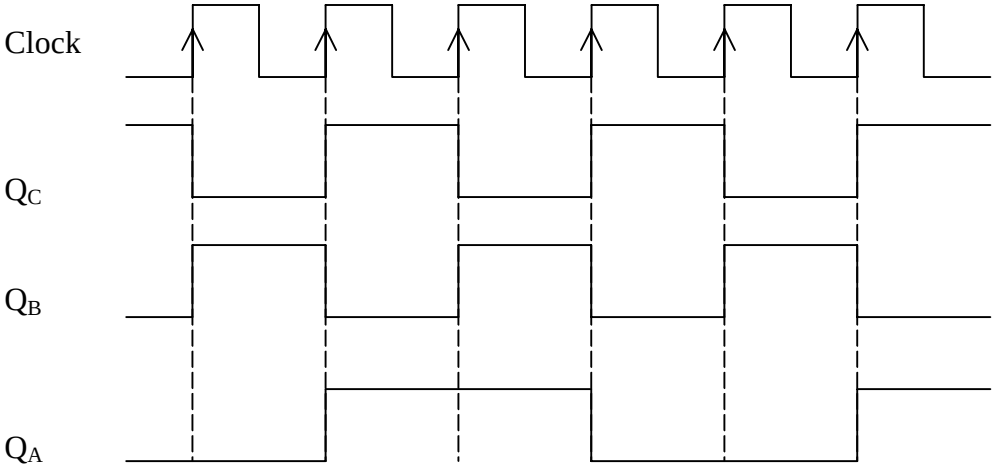
como se pretende que a saída seja 0 quando as entradas forem 111, utiliza-se um NAND.
Por outro lado, sempre que a entrada /LOAD é activada é necessário que o valor das entradas A, B, C, e D representem o estado 1; para tal, é necessário ligar a entrada A a V_{CC} e as restantes entradas à massa, de modo a termos 0001 nas entradas do contador sempre que a entrada /LOAD for activada.
Duas considerações finais: a entrada D/U tem que estar ligada à massa, de modo a que seja feita uma contagem no modo ascendente; a entrada CTEN também tem de estar ligada à massa para permitir que a contagem seja feita.
Tendo tudo isto em consideração, a implementação do sistema referido utilizando um contador 74LS190 seria a seguinte:



3. O circuito apresentado é o seguinte:



O diagrama temporal deste circuito será o seguinte:

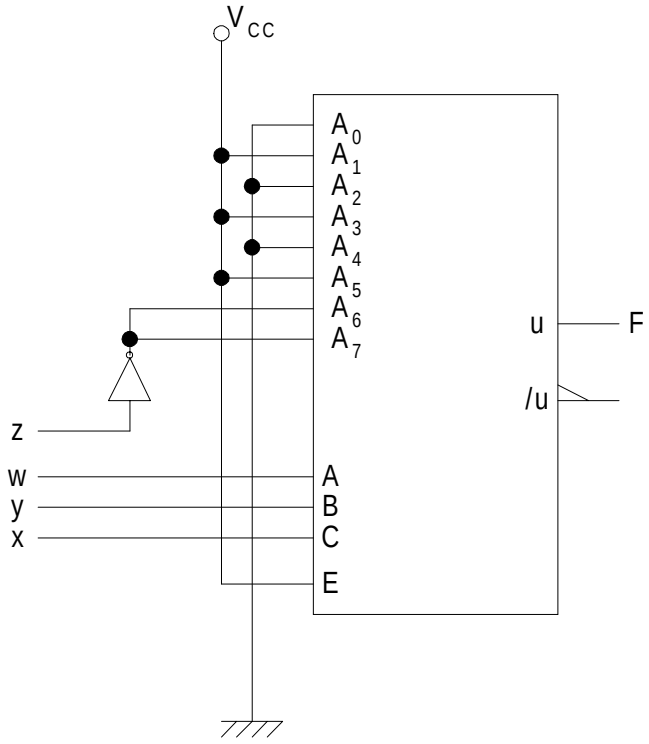


4.

a. O multiplexer da figura contém 3 entradas de controlo; ora, como se pode implementar qualquer função lógica com **n + 1** variáveis lógicas com um multiplexer de **n** entradas de controlo e como a função especificada tem 4 variáveis lógicas, basta 1 multiplexer para a implementar.
Para implementar a função lógica especificada, convém, em primeiro lugar, expandir a tabela de verdade apresentada para que todas as combinações possíveis estejam devidamente representadas e ordenadas. Assim sendo, a tabela de verdade expandida seria:

x	**y**	**w**	**z**	**F**	
0	0	0	0	0	0
0	0	0	1	0	
0	0	1	0	1	1
0	0	1	1	1	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	0	0
1	0	0	1	0	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	1	/z
1	1	0	1	0	
1	1	1	0	1	/z
1	1	1	1	0	

Daqui se tira a implementação directa da função, através dos valores especificados à direita da tabela. A entrada de controlo **C** corresponde à entrada mais significativa (variável **x**) e a entrada **A** corresponde à entrada menos significativa (variável **w**).
A entrada **E** corresponde ao *enable* do multiplexer e, como é activa ao nível lógico alto, deve estar ligada a V_{CC}.
Assim sendo, o circuito que implementaria a função especificada seria o seguinte:



b. As vantagens dos multiplexers, face às portas lógicas convencionais, são as seguintes:
   - Implementação directa de uma função lógica, sem necessidade de simplificação da função
   - Circuitos digitais menos "confusos"
   - Ocupam menos espaço em casos onde se utilizam muitas e vários tipos de portas lógicas (portas do tipo AND, OR; etc.)

   As desvantagens dos multiplexers, face às portas lógicas convencionais, são as seguintes:
   - Para um número elevado de variáveis lógicas é preferível simplificar o circuito e como tal recorrer às portas convencionais que poderão ser em número consideravelmente menor, limitando, deste modo, a sua gama de aplicações práticas
   - O custo dos multiplexers é maior quando comparados com o uso de CI's com um só tipo de portas

5.

a. Em primeiro lugar definem-se quais as variáveis de entrada e de saída do sistema e qual o seu significado.
Assim sendo, tem-se que as variáveis de entrada são:

PF { 1 – todas as portas estão fechadas
   { 0 – há portas abertas

SP { 1 – pressão nivelada
   { 0 – pressão desnivelada

De notar que estas variáveis têm os valores lógicos especificados no enunciado do problema, pelo que não haveria qualquer alternativa a esta solução.
As variáveis de saída não têm qualquer especificação pelo que se pode atribuir qualquer lógico a qualquer das situações. As variáveis de saída são:

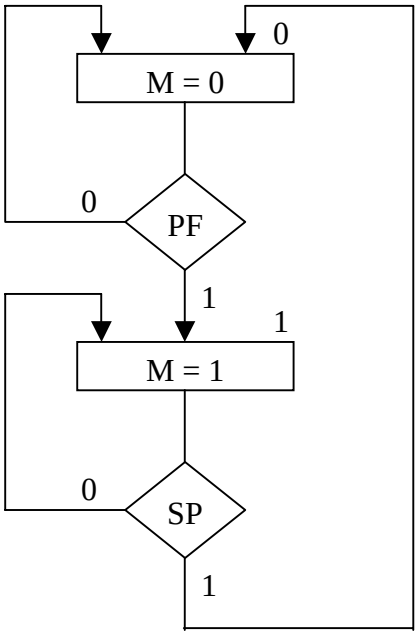
M { 1 – motor em funcionamento (a nivelar a pressão)
  { 0 – motor desligado

De seguida definem-se os estados do sistema. Neste caso há dois estados descritos de seguida:

Estado 0: as portas da câmara estão abertas e como tal o motor está desligado

Estado 1: as portas da câmara estão fechadas e o motor está em funcionamento, ou seja, está a nivelar a pressão entre o exterior e o interior da câmara.

Tendo isto em consideração, o fluxograma do sistema tem a seguinte forma:

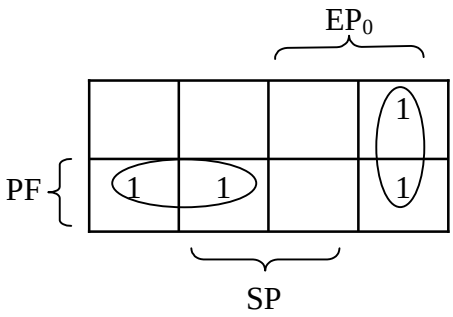


b. Atendendo ao fluxograma constrói-se a seguinte tabela de verdade:

EP_0	SP	PF	PE_0	M
0	X	0	0	0
0	X	1	1	0
1	0	X	1	1
1	1	X	0	1

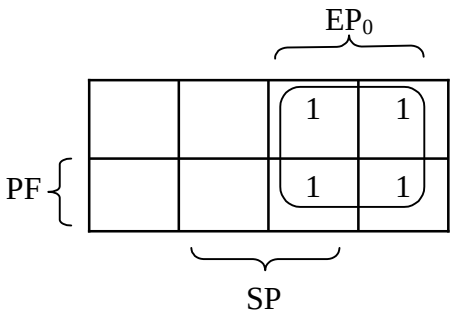
c. Para simplificar as saídas, recorre-se, por exemplo, a um mapa de Karnaugh para cada uma delas. Assim sendo, vem:

Para PE_0:



$$PE_0 = EP_0 \cdot \overline{SP} + \overline{EP_0} \cdot PF$$

Para M:



$M = EP_0$

d. Com as simplificações efectuadas na alínea anterior, tem-se que um circuito que implementaria o problema especificado, seria o seguinte: