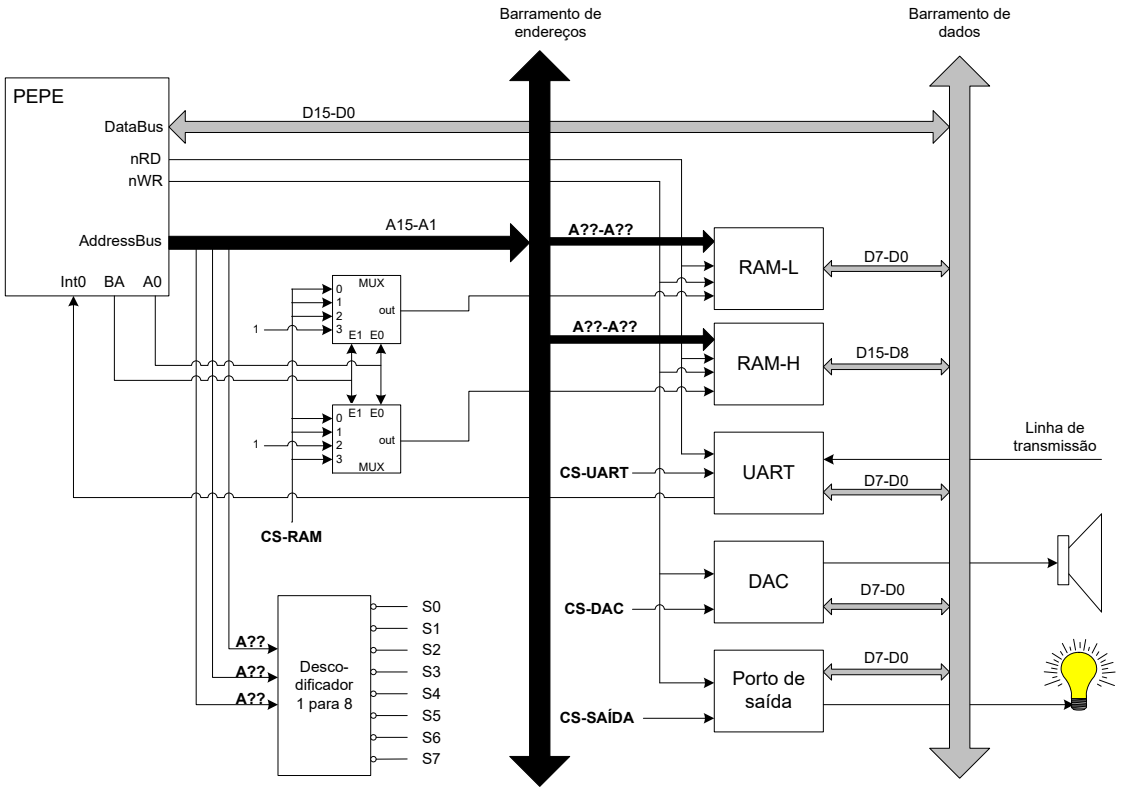

1 - A figura representa um sistema computacional baseado no processador PEPE, com 4 Kbytes de RAM e 3 periféricos de 8 bits. O objectivo básico deste circuito é receber continuamente (da linha de transmissão) bytes com som digitalizado, reproduzir esse som num altifalante e acender a lâmpada sempre que cada valor de som (cada byte) exceder um dado valor.

Todos os sinais relevantes para a descodificação de endereços (chip selects, nRD, nWR e saídas do descodificador de 1 para 8) são activos a 0. O sinal BA (Byte Addressing) indica se o acesso do PEPE à memória/periféricos é em byte (BA=1) ou em palavra (BA=0).

A RAM está dividida em dois módulos, RAM-L e RAM-H, com 8 bits de largura cada.

A UART (Universal Asynchronous Receiver Transmitter) é um periférico que suporta comunicação série assíncrona, com bit de paridade e 2 stop bits. Neste sistema, a UART é usada apenas para recepção. Quando a UART acaba de receber um byte, gera uma interrupção ao processador, que deve fazer um acesso de leitura à UART, lendo esse byte.

O DAC (Digital to Analog Converter) é um periférico que gera um sinal analógico com uma tensão proporcional ao valor do byte que o processador escrever nele. Assim, a sequência de bytes recebidos na UART pode ser transformada em som analógico, que depois de amplificado será ouvido num altifalante.

O porto de saída não é mais do que um simples periférico que memoriza o byte que o processador escrever nele. O bit de menor peso deste porto liga a um circuito adequado para controlar uma lâmpada. Se o bit for 1 a lâmpada acende, se for 0 a lâmpada apaga-se.

a) - Quantos bytes tem cada RAM de capacidade? Justifique. Sugestão: releia o início da pergunta 1.

2 K bytes (metade da capacidade total da RAM). Uma RAM tem os bytes com endereço par, a outra os bytes com endereço ímpar.

b) - Pretende-se que os vários dispositivos estejam acessíveis nos seguintes endereços (poderão também ser acessíveis noutros, mas não pode haver mais do que um dispositivo mapeado em cada endereço possível).

Dispositivo	**Endereço(s) em que deve estar acessível**
RAM	A partir de 0000H
UART	6500H
DAC	7500H
Porto de saída	2200H

Preencha a tabela seguinte (completando coisas que no desenho não estão completamente especificadas) para que tal aconteça.

O que é preciso ligar	**Nome do sinal ou dos bits onde liga**
Que bits ligam às RAMs (A??-A??)	***A11- A1***
Que bits ligam ao descodificador (A??)	***A14, A13 e A12***
Onde liga o sinal CS-RAM?	***S0***
Onde liga o sinal CS-UART?	***S6***
Onde liga o sinal CS-DAC?	***S7***
Onde liga o sinal CS-SAÍDA?	***S2***

Apresente uma breve justificação

O bit A0 é usado para discriminar qual das RAMs, par ou ímpar. Cada RAM tem 2 Kbytes, pelo que precisa de 11 bits, A1 a A11. As RAMs são os dispositivos com mais endereços, logo são elas que determinam o número de endereços em que cada saída do descodificador está activa. Assim, os bits que ligam ao descodificador são os seguintes ao de maior peso que liga às RAMs, ou A12, A13 e A14. Cada saída do descodificador está activa durante 1000H endereços, pelo que as saídas a que os CS ligam correspondem ao dígito hexadecimal de maior peso dos endereços dos vários dispositivos.

c) - Imagine que executa a instrução **MOVB R1, [R2]** estando já a funcionar o esquema de endereçamento da alínea b). Indique qual o valor dos seguintes sinais durante o acesso e supondo que R2 = 6FFEH.

Sinal	**Valor**
A0	***0***
BA	***1***
nRD	***0***
nWR	***1***
CS_RAM	***1***
CS_UART	***0***
CS_DAC	***1***
CS_SAÍDA	***1***

2 - Imagine que o emissor, do outro lado da linha de transmissão, produz bytes a um ritmo de 20.000 por segundo.

a) - Qual o ritmo mínimo, em bits/segundo, a que a UART deve receber os bits vindos do emissor para que todos os bytes de som consigam ser transmitidos? Justifique. Sugestão: veja as características da UART (indicadas atrás).

Cada byte demora 12 bits a ser transmitido (start bit, 8 bits do byte, bit de paridade, 2 stop bits). Logo, têm de ser transmitidos no mínimo 240.000 bits/segundo (12 × 20.000).

b) - De quanto em quanto tempo é que o PEPE recebe uma interrupção? Justifique.

50 microsegundos (1/20.000), pois cada byte recebido origina uma interrupção por parte da UART.

c) - Suponha que enquanto os bytes com o som estão a ser recebidos, o processador está a correr outro programa, demorando 1 minuto a executar nestas condições. Se o ritmo de produção de bytes do emissor se mantiver mas aumentarmos o ritmo de transmissão dos bits para o dobro, o que é que sucede ao tempo de execução do programa: mantém-se, sobe ou desce? Justifique.

Aumentar o ritmo de transmissão (cada bit demora menos tempo a ser transmitido) mantendo o número de bytes enviados por segundo apenas faz com que o tempo de intervalo entre dois bytes consecutivos aumente. A cadência de interrupções e o processamento associado do programa mantêm-se. Logo, o tempo de execução do programa mantém-se.

3 - Bem, nada funciona sem o programa adequado.

a) - Quando um byte é recebido, é preciso escrevê-lo no DAC. Também é preciso verificar ver se o valor do byte está acima ou abaixo de um dado limiar, indicado pela constante simbólica LIMITE (assume-se que foi previamente definida). Se byte >= LIMITE, deve ser ligada a lâmpada, para o que basta colocar o bit de menor peso do porto de saída a 1. Se byte < LIMITE, a lâmpada deve ser apagada.. Programe a rotina adequada para implementar esta funcionalidade no contexto deste problema (não é preciso usar todas as linhas).

Etiquetas	Instruções	Comentários
rot_int0:		*Rotina de interrupção (invocada sempre que a UART recebe um byte)*
	PUSH R1	*Salva registos usados*
	PUSH R2	
	MOV R2, 6500H	*Endereço da UART*
	MOVB R1, [R2]	*Lê o byte que chegou*
	MOV R2, 7500H	*Endereço do DAC*
	MOVB [R2], R1	*Escreve o byte que chegou no DAC*
	MOV R2, LIMITE	*Valor limite para não acender a lâmpada*
	CMP R1, R2	*Já chegou ao limite?*
	JLT apaga	
acende:	*MOV R1, 1*	*Sim, acende a lâmpada*
	JMP lampada	
apaga:	*MOV R1, 0*	*Não, apaga a lâmpada*
lâmpada:	*MOV R2, 2200H*	*Endereço do periférico da lâmpada*
	MOVB [R2], R1	*Actualiza o estado da lâmpada*
	POP R2	*Repõe valor dos registos*
	POP R1	
	RFE	*Retorna da interrupção*

b) - Também é preciso fazer um programa principal. O programador até já tinha feito algumas coisas, mas deixou cair ao chão a pen-drive onde tinha guardado o programa principal e as instruções ficaram por uma ordem aleatória. Houve mesmo linhas do ficheiro (ou partes de linha) que se perderam! Nomeadamente, os comentários perderam-se todos. O que foi possível ler da pen-drive (sabe-se lá em que estado) é o seguinte:

```
        EI
ciclo:  CALL    processo1
        WORD    rot_int0
PLACE   0000H
        EI0
        JMP     ciclo
        MOV     SP,
CALL    processo2
        MOV     BTE,
```

Sabendo que as instruções não ocupam endereços da RAM além de 200H, refaça o programa de modo a funcionar correctamente (não é preciso usar todas as linhas).

Etiquetas	Instruções		Comentários
PLACE	*200H*		
int0	*WORD*	*rot_int0*	*Tabela de endereços das rotinas de interrupções*
PLACE	*0000H*		*Endereço de arranque do PEPE após reset*
inicio:	*MOV*	*SP, 1000H*	*Inicialização do SP com o endereço imediatamente após a pilha*
	MOV	*BTE, int0*	*Inicialização do registo BTE com o endereço da tabela de interrupções*
	EI0		*Permite interrupções do tipo 0*
	EI		*Permite interrupções*
ciclo:	*CALL*	*processo1*	*Chama o processo 1*
	CALL	*processo2*	*Chama o processo 2*
	JMP	*ciclo*	*Volta ao ciclo principal*