

## Parte Teórica

## Cotações:

1– 0,5; 2– 0,5; 3– 2,5; 4– 2,5; 5a– 1,5 b– 0,5; 6a– 0,5 b– 1,5 c– 2,5; 7a– 0,5 b– 2 c– 2; 8a– 2, 8b– 1

**Nota: 1 e 2 descontados 0,5 por cada resposta errada (resposta no enunciado)**

1. O I2C é um bus:
  - a. Síncrono
  - b. Assíncrono
2. Qual o tipo de memória usado nos SSD:
  - a. SRAM
  - b. Flash Memory
  - c. EPROM
3. Como funciona um sistema de interrupções vetorizadas? Quais as diferenças relativamente a interrupções não-vetorizadas? Descreva o sistema de interrupções do PIC32.

Num sistema de interrupções vetorizadas quando um pedido de interrupção de um dispositivo é aceite, o controlador de interrupções coloca no bus o *IT vector* correspondente ao dispositivo. O *IT vector* é usado para indexar a tabela em memória com os endereços de entrada das subrotinas de tratamento das interrupções (*IT Handlers* ou *ISRs*), iniciando-se de imediato a execução da subrotina de tratamento das interrupções do dispositivo em causa.

Num sistema de interrupções não-vetorizadas quando um pedido de interrupção de um dispositivo é aceite, o controle é transferido para a subrotina genérica de tratamento das interrupções, que começa por identificar qual o periférico cujo pedido de interrupção foi aceite. No PIC32 o sistema de interrupções tem dois modos de funcionamento possíveis designados por “*Single Vector Mode*” e “*Multi-Vector Mode*”, correspondendo o *Single Vector Mode* ao funcionamento não-vetorizado do sistema de interrupções. Nos trabalhos de laboratório usou-se *Multi-Vector Mode*, isto é, interrupções vetorizadas.

O controlador de interrupções do PIC32 suporta até 256 fontes de interrupção (IRQs) provenientes dos diferentes módulos integrados no chip e de até 5 fontes de interrupção externas. O programador pode associar a cada fonte de interrupção um de 7 níveis de prioridade, e dentro de cada nível de prioridade pode ainda estabelecer diferentes prioridades de segundo nível (*Subpriority* de 0 a 3).

4. *Carrier Sense Multiple Access* é usado por Ethernet e por CAN. Explique em que consiste. Porque é que CAN usa uma estratégia diferente da usada por Ethernet na resolução de conflitos no acesso ao bus?

*Carrier Sense Multiple Access* (CSMA) é um protocolo de transmissão num meio ao qual estão ligados múltiplos dispositivos (*Multiple Access*). *Carrier Sense* significa que cada um dos dispositivos ligados à rede monitora (“*sense*”) continuamente o estado do bus, e só quando ele está livre tenta transmitir.

*Carrier Sense Multiple Access /Collision Detect* (CSMA/CD) é o protocolo usado pelas redes Ethernet. Qualquer nó que “*senses*” a linha livre pode iniciar a transmissão de uma *frame* (trama). Se um outro dispositivo tenta simultaneamente transmitir diz-se que ocorre uma colisão e as frames são canceladas. Os nós envolvidos esperam então um tempo aleatório e tentam de novo até conseguirem enviar as suas *frames*, não existindo a garantia de um máximo para a transmissão das *frames*, o que torna as redes Ethernet inadequadas para aplicações “tempo real”.

As redes CAN usam *Carrier Sense Multiple Access /Collision Avoidance* (CSMA/CA) em lugar de CSMA/CD. Em CAN o bus a que se ligam os nós tem uma lógica “*wired AND*”, isto é, quando um nó transmite zero impõe o nível zero no bus, independentemente do que os outros nós estejam a transmitir. Diz-se assim que o “0” é dominante e o “1” recessivo.

## Parte Teórica

Em CAN cada frame (trama), se inicia (imediatamente após o *start bit*) pelo campo identificador da mensagem. É este campo que vai decidir a arbitragem entre os nós que tentam simultaneamente transmitir. Como o identificador é transmitido “MSB first”, a mensagem com o menor identificador (i.e. aquela em que primeiro é transmitido um 0) é aquela que ganha a arbitragem, podendo prosseguir a transmissão da respetiva trama enquanto todos os outros nós desistem de transmitir, só o tentando de novo após essa transmissão terminar. Há assim a garantia de que em cada momento a transmissão da mensagem mais prioritária não é atrasada pela competição com mensagens de menor prioridade.

5. A figura representa um diagrama temporal simplificado de um ciclo de leitura de uma DRAM. O tempo de acesso é o intervalo  $[t_1, t_2]$ . O tempo de restabelecimento é o intervalo  $[t_2, t_3]$ , durante o qual a DRAM tem de restaurar a carga antes de o processador possa efetuar um novo acesso. Assuma que:  $t_2 - t_1 = 25\text{ns}$  e  $t_3 - t_2 = 15\text{ns}$

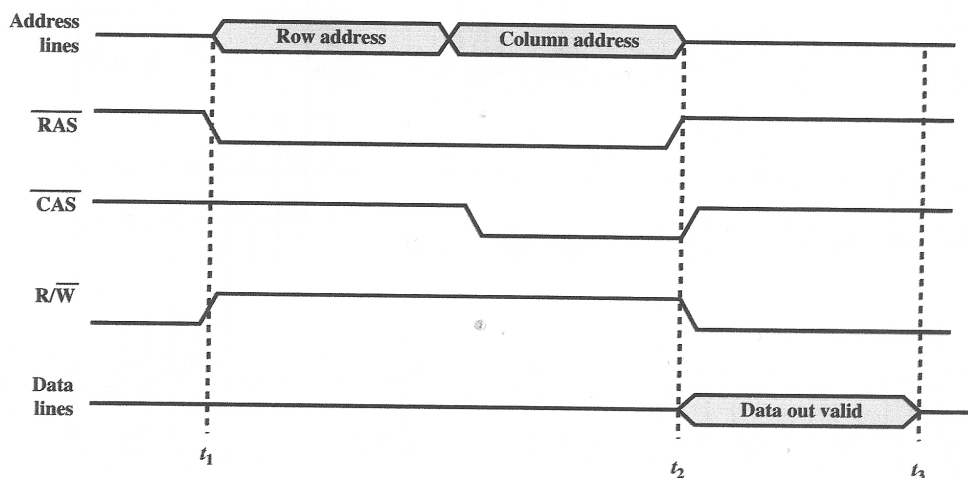
- a. Qual é o tempo de ciclo? Qual a máxima taxa de transferência que esta DRAM pode assegurar assumindo que se trata de uma 64Mx4?

$$T_{\text{ciclo}} = 25 + 15 = 40 \text{ ns};$$

$$\text{Transfer rate} = 4 * (1 / (40 * 10^{-9})) = 10^8 \text{ bits/s} = 100 \text{ Mbits/s}$$

- b. Se se construir uma memória x32 usando esta componente qual a taxa de transferência?

800 Mbits/s



6. Pretende-se projetar um sistema de memória que permita detetar erros nos bytes armazenados.
- a. Se apenas se pretender detetar erros de um bit, que solução adotaria?

Acrescentar-se-ia a cada byte um bit de paridade

- b. Se se pretender detetar erros e corrigir os erros de 1 bit, que código de deteção e correção de erros adotaria? O código de quantos bits adicionais para deteção e correção de erros seriam necessários por cada byte?

O código de Hamming. Seriam necessários 4 bits adicionais (C8, C4, C2, C1). Se para além de detetar e corrigir erros de 1 bit se pretendesse também detetar erros de 2 bits, acrescentar-se-ia aos 4 bits um bit de paridade.

- c. Para o byte 01101101 gere os bits de deteção e correção de erros. Mostre que o código identifica corretamente um erro no bit 4.

## Parte Teórica

Posição dos bits: 12 11 10 9 8 7 6 5 4 3 2 1  
M8 M7 M6 M5 C8 M4 M3 M2 C4 M1 C2 C1

$$C1 = M1 \text{ XOR } M2 \text{ XOR } M4 \text{ XOR } M5 \text{ XOR } M7 = 1 \text{ XOR } 0 \text{ XOR } 1 \text{ XOR } 0 \text{ XOR } 1 = 1$$

$$C2 = M1 \text{ XOR } M3 \text{ XOR } M4 \text{ XOR } M6 \text{ XOR } M7 = 1 \text{ XOR } 1 \text{ XOR } 1 \text{ XOR } 1 \text{ XOR } 1 = 1$$

$$C4 = M2 \text{ XOR } M3 \text{ XOR } M4 \text{ XOR } M8 = 0 \text{ XOR } 1 \text{ XOR } 1 \text{ XOR } 0 = 0$$

$$C8 = M5 \text{ XOR } M6 \text{ XOR } M7 \text{ XOR } M8 = 0 \text{ XOR } 1 \text{ XOR } 1 \text{ XOR } 0 = 0$$

Palavra armazenada: 011001100111

Palavra lida com erro no bit 4 (M5): 011101100111

Cálculo de Cs para a palavra lida: apenas se alteram os valores dos bits que dependem de M5

$$C1_{\text{calc}} = 0 \quad C1 \text{ XOR } C1_{\text{calc}} = 1$$

$$C2_{\text{calc}} = 1 \quad C2 \text{ XOR } C2_{\text{calc}} = 0$$

$$C4_{\text{calc}} = 0 \quad C4 \text{ XOR } C4_{\text{calc}} = 0$$

$$C8_{\text{calc}} = 1 \quad C8 \text{ XOR } C8_{\text{calc}} = 1$$

**Bit na posição 9 errado – erro em M5**

7. Um sistema de 32-bits com a memória byte-addressable tem uma cache direct-mapped de 4kBytes com blocos (linhas) de 4 palavras de 32 bits.

- a. Quantas linhas tem a cache? **Linha: 4 palavras -> 16 bytes; N° linhas =  $4 * 2^{10} / 16 = 2^8 = 256$**   
b. Quais os campos em que estão divididos os endereços da memória e qual o número de bits de cada um deles?

|     |    |       |   |        |   |
|-----|----|-------|---|--------|---|
| 31  | 12 | 11    | 4 | 3      | 0 |
| Tag |    | Index |   | Offset |   |

**Tag – os 20 bits mais significativos do endereço do bloco de memória armazenado na linha da cache**

**Index – endereço da linha da cache (8 bits)**

**Offset – 4 bits que indicam a posição do byte na linha. Quando Offset = XX00, os 2 bits mais significativos (XX) indicam a posição de uma palavra.**

- c. O conteúdo da posição de memória 101B2A5C foi transferido para a cache. Em que linha da cache e em que posição na linha se encontra? Quais os endereços das outras palavras armazenadas na mesma linha da cache? Qual o conteúdo do campo **Tag** correspondente a essa linha?

**Linha da cache =  $A5_{16} = 165$  Posição da palavra na linha:  $11_2 = 3$  Posição do byte:  $C_{16} = 12$**

**Endereços das outras palavras armazenadas na mesma linha: 101B2A50, 101B2A54, 101B2A58**

**Tag = 101B2**

8. Uma unidade de disco tem as seguintes características: *Seek time* = 10ms; Velocidade de rotação = 5400rpm; *Transfer Rate* = 8 MB/s.  
a. Calcule o tempo médio requerido para transferir 32kB do disco para a memória, assumindo que esses dados se encontram armazenados em sectores contíguos.

$$T_{\text{av}} = t_{\text{seek}} + t_{\text{rot}}/2 + t_{\text{data}} = 10 + 0,5 * (60/5400) * 10^3 + (32 * 2^{10} / 8 * 2^{20}) * 10^3 \text{ ms}$$

$$T_{\text{av}} \approx 10 + 5,56 + 0,03 = 15,59 \text{ ms}$$

- b. As transferências entre o disco e a memória fazem-se por DMA. Porquê?

**As transferências de informação de/para o disco envolvem não itens individuais (bytes ou palavras) mas blocos de dados (no mínimo o conteúdo de um sector do disco, cuja dimensão nas atuais unidades de disco é no mínimo de 512 Bytes e pode ir até 8kB), com velocidades de transferência**

## Parte Teórica

elevadas (8 MB/s no caso da unidade de disco aqui referida). Se o processador tivesse de efetuar essas transferências:

disco -> registo do processador -> memória na leitura

memória -> registo do processador -> disco na escrita

o tempo de processador gasto nas comunicações com o disco seria considerável. Para libertar o processador da tarefa da transferência de dados, o sistema de I/O dos computadores inclui um módulo (o controlador de DMA) com capacidade de endereçar a memória e de fazer a transferência de dados diretamente entre memória e os periféricos em que a transferência de dados se faz em blocos (*block devices*) e não caráter a caráter (*character devices*).

O controlador de DMA atua assim como um processador auxiliar controlado pelo processador central que lhe indica o endereço base do bloco de memória para/de onde transferir os dados e a dimensão do bloco de dados. Quando termina a transferência o controlador de DMA gera uma interrupção para informar o processador de que terminou a tarefa.

No caso do PIC32 como os módulos de I/O estão integrados no mesmo chip que o processador, os módulos que comunicam com *block devices* (Ethernet, CAN, High Speed USB,...) incorporam controlador de DMA próprio, existindo para além disso um controlador de DMA autónomo que permite assegurar a capacidade de acesso direto à memória de outros dispositivos externos.