**Universidade de Aveiro** Arquitetura de Computadores II Questões teóricas

**INTRODUÇÃO**

1. Num sistema computacional, como definiria com as suas próprias palavras o conceito de espaço de endereçamento?

O espaço em bits ou bytes que estão disponíveis para representar um endreço do sistema

2. Quando, num sistema computacional, estamos a determinar, em função do endereço presente no barramento, qual o periférico ou memória que deve ser selecionada, estamos perante uma operação que é normalmente designada por?

Descodificação de endereços

3. Quando nos referimos ao tipo de organização de memória num sistema, o que significa dizer que este é:

*a. byte-addressable*

Cada endereço de contém 1 byte

*b. bit-addressable*

Cada endereço de contém 1 bit

*c. word-addressable*

Cada endereço de contém 1 word (4 bytes)

4. Identifique e descreva pelas suas palavras qual o papel, na arquitetura de um sistema computacional:

a. do *Data Bus*

Data bus é o barramento que os dados percorrem quer venham do CPU ou vão para o CPU

b. do *Address Bus*

Address Bus é o barramento que identifica qual o endereço ao qual o CPU quer aceder

c. do *Control Bus*

Control Bus é o barramento que indica qual a operação ou instrução que está/quer que seja feita

5. Na arquitetura de um sistema computacional, como designa o barramento que permite identificar o registo, na memória ou num periférico, do qual ou para o qual vai ocorrer uma transferência de informação.

Adress Bus

6. Na arquitetura de um sistema computacional, como designa o barramento que permite especificar o tipo de operação efetuada sobre a memória ou sobre um periférico.

Control Bus

**MICROCONTOLADORES (EMBEDED SYSTEMS)**

7. Um compilador-cruzado (*cross-compiler*) é o nome dado a um tipo específico de programa. Como descreveria, nas suas próprias palavras o que caracteriza este tipo de programa?

Um cross-compiler corre na máquina host mas traduz o software desenvolvido de forma a que seja executável na máquina target.

8. Identifique qual a função de um *bootloader* num sistema baseado em microcontrolador?

Bootloader é um programa que é executado no arranque do sistema do microcontrolador e que apenas disponibiliza funções básicas de transferência e execução de programas.

9. Quando falamos em microcontroladores, por oposição a um sistema computacional de uso geral, o que podemos afirmar:

a. Quanto aos principais aspetos da sua arquitetura interna

A arquitetura de um microcontrolador é muito menos complexa do que a arquitetura de um sistema computacional.

b. Quanto à sua frequência de trabalho

A frequência de trabalho de um microcontrolador é bastante inferior à frequência de um CPU de um sistema computacional.

c. Quanto à disponibilização de periféricos

A disponibilização de periféricos de um microcontrolador é bastante maior do que a de um sistema computacional.

d. Quanto ao custo

O custo de um microcontrolador é bastante inferior ao de um sistema computacional.

e. Quanto à energia consumida

O consumo de energia de um microcontrolador é bastante inferior de um sistema computacional.

f. Quanto aos seus campos de aplicação

Pode ser usado em tarefas específicas.

10. Como descreveria as principais caraterísticas de um sistema embebido?

Um sistema embebido é um sistema computacional especializado pensado para realizar uma tarefa específica ou para controlar um determinado dispositivo. Tem requisitos próprios e tarefas pré-definidas. Pode ser implementado num microcontrolador ou pode fazer parte de um sistema computacional mais complexo.

11. Um microcontrolador PIC32 usa uma arquitetura *pipelined* semelhante à estudada nas aulas de AC1. Descreva qual o modelo base da arquitetura usada e o tipo ou tipos de memória usadas pelo sistema.

O modelo base da arquitetura usada num microcontrolador PIC32 é uma Arquitetura Harvard e o sistema usa memórias voláteis (RAM) e não voláteis (ROM).

**MÓDULOS DE I/O**

12. Na arquitetura de um microcontrolador PIC32 qual a finalidade e funcionalidade dos seguintes registos:

a. TRIS:

Define o porto como porto de saída ou porto de entrada.

b. LAT:

Usado para escrever um valor num porto de saída.

c. PORT:

Usado para ler o valor de um porto de entrada.

13. Num porto de I/O do PIC32 (esquema abaixo), quando o porto se encontra configurado como porto de entrada,

a. Continua a ser possível escrever no registo de saída?

Sim

b. Continua a ser possível ler o valor que se encontra armazenado no registo LAT?

Sim

c. Se sim, qual o sinal que permite realizar essa leitura?

RD\_PORT

Uma imagem com diagrama, texto, Esquema, file

Descrição gerada automaticamente

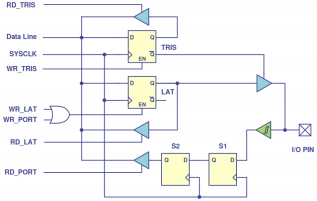
14. Num porto de I/O do PIC32 (esquema abaixo), se eu não souber qual a configuração do mesmo (saída ou entrada),

a. Será possível saber essa informação programaticamente?

Sim

b. se sim, qual o sinal de controlo que permite essa operação?

O sinal TRISx onde x corresponde à letra do porto (RD\_TRIS no esquema)



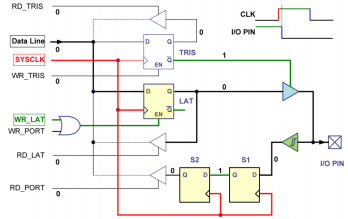
15. Considere que está a decorrer uma operação sobre um porto de I/O de um PIC32 (ver esquema abaixo – considere o instante assinalado no diagrama temporal).

a. O porto encontra-se configurado como entrada ou saída?

Como saída.

b. Identifique e descreva qual a operação que está em curso.

Está em curso uma operação de escrita do valor 0.

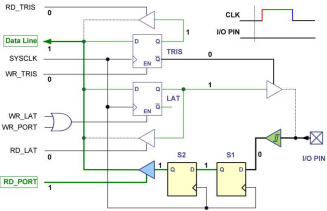


(2)

16. Repita o exercício anterior para o esquema apresentado abaixo

Porto configurado como entrada.

Operação de leitura do valor de entrada no porto.



17. Na implementação da parte de dados de um porto de saída, que tipo de dispositivo lógico deve ser usado para armazenar o valor transferido através do barramento de dados durante um ciclo de escrita: Uma latch.

18. Na implementação de um porto de I/O do PIC32, o registo PORT está associado a um conjunto de dois flip-flops D em série (*shift register* de dois andares). Qual o objetivo dessa implementação?

Sendo que a leitura do porto é feita de forma assíncrona relativamente ao clock do CPU, usa-se dois flip-flops D em série (*Shift register* de dois andares) de forma a sincronizar a leitura do porto com o clock do CPU com um atraso de até dois ciclos de relógio.

**NOÇÕES BÁSICAS SOBRE PERIFÉRICOS**

20. A descrição da funcionalidade de um dado dispositivo periférico, o seu conjunto de registos de dados, de controlo e de *status* é genericamente designada, no contexto de arquitetura de computadores, por uma designação específica. Qual essa designação?

É designada por I/O Module.

21. Quando, no acesso que o CPU faz a um módulo de E/S, é usada a técnica de entrada/saída de dados por software (programada), quais as tarefas que são realizadas pelo CPU?

O CPU espera que o periférico esteja disponível para a troca de informação através de um ciclo de verificação da informação de status do periférico (POLLING).

Uma vez que o CPU recebe a informação que o periférico está disponível para a troca de informação então o CPU começa esta troca até todos os dados terem sido transferidos.

22. O método de transferência de informação entre um CPU e um módulo de E/S (I/O), em que o programa executado no CPU é responsável por iniciar, monitorizar e controlar a transferência de informação, designa-se por:

Método de transferência por interrupção.

23. Quando nos referimos a um “Módulo de I/O”, estamos a referir-nos especificamente a que parte do periférico de que este módulo faz parte?

À parte que interage entre o CPU e o periférico.

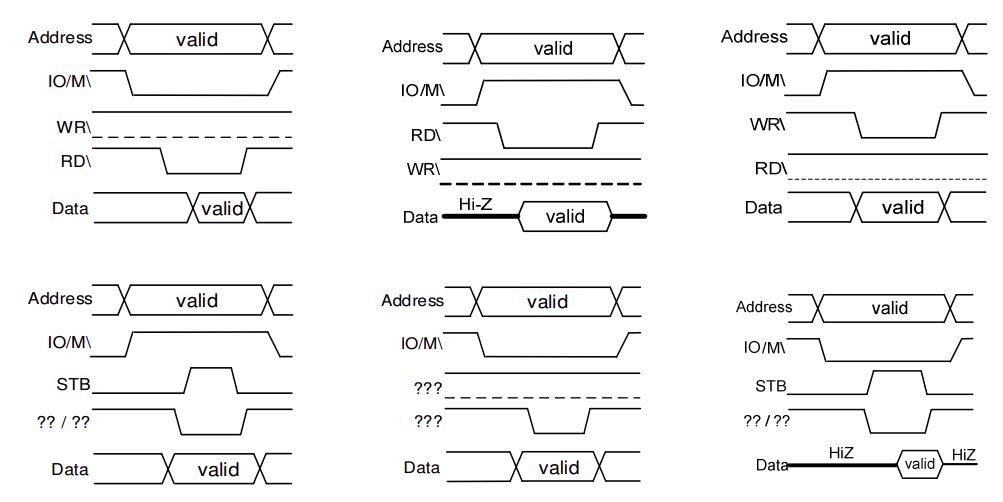
24. Na implementação da parte de dados de um porto de entrada de um módulo de I/O:

a. Que tipo de dispositivos lógicos devem ser usados na ligação ao barramento de dados?

Porta Tristate.

b. por que razão é fundamental usar esses dispositivos?

Serve para diminuir o ruido digital.

25. Os diagramas temporais que se seguem, nas várias figuras, representam operação de transferência de informação (leitura ou escrita) de/para dispositivos que podem estar mapeados em espaços de endereçamento de I/O ou de memória. Identifique, para cada diagrama qual o tipo de operação e espaço de endereçamento utilizado:

1- Leitura de Memoria, 2- Leitura de Periferico, 3- Escrita na memoria, 4 – Escrita no periférico, 5 -

(4)

**INTERRUPÇÕES**

27. No que respeita ao sistema de interrupções do PIC32 é (usado na placa DETPIC32):

a. Descreva sucintamente os dois modos de funcionamento do sistema de interrupções

Existem dois modos de funcionamento do sistema de interrupções, um por software e outro por hardware.

A interrupção por software apenas usa 1 entrada de interrupção e apenas uma única RSI para todas as fontes de interrupção.

Por outro lado, a interrupção por hardware usa apenas 1 entrada de interrupção, mas cada periférico possui um identificador único (vetor) e existe uma RSI para cada vetor de interrupção.

b. Do caso do sistema de interrupções feita por hardware, como descreveria sucintamente o seu funcionamento, ainda no caso da alínea anterior, o que determina a ordem pela qual as interrupções são servidas nos casos em que ocorram em simultânea a partir de mais do que uma fonte

Durante o processo de atendimento à interrupção, na fase de identificação da fonte, o periférico gerador da interrupção identifica-se através do seu vetor.

O vetor vai ser usado depois como índice de uma tabela que contém o endereço de cada uma das RSI, ou intruções de salto incondicional para as RSI.

O sistema de interrupções do PIC32 é baseado num módulo de gestão exterior ao CPU (controlador de interrupções). Este controlador funciona como um priority encoder que define a ordem pela qual as interrupções são executadas.

28. Numa RSI, qual o objetivo do conjunto de instruções designado por:

a. "prólogo"?

O prólogo é um conjunto de instruções que salvam os registos necessários antes de entrar na função de atendimento à interrupção.

b. "epílogo"?

O epílogo é um conjunto de instruções que repõem os valores corretos dos registos guardados antes de acabar o atendimento à interrupção

29. Descreva, por palavras suas, o que se entende por *overhead* da transferência de informação por interrupção e as razões que justificam esse overhead.

O overhead é o número de ciclos de relógio que o CPU gasta a fazer o epilogo e o prólogo da rotina de serviço à interrupção. Existe para saber quantas instruções é que a RSI tem verdadeiramente para atender e tratar da interrupção.

30. Considere um sistema baseado num CPU a funcionar a uma frequência de 10 MHz com uma taxa de execução de 5 MIPS (5x106 instruções por segundo, CPI = 2) que processa por interrupção eventos externos periódicos. Se o overheadtotal do atendimento à interrupção for de 20 ciclos de relógio, e a rotina de serviço à interrupção tiver 40 instruções, determine a máxima frequência a que esses eventos podem ocorrer para que todas as interrupções possam ser atendidas.

5 MIPS = 5 000 000 IPS

CPI = 2

Overhead = 20 ciclos de relógio = 20/2 instruções = 10 instruções

RSI = 40 + 10 = 50 instruções

f = 5 000 000/50 = 100 000 Hz

31. Considere um sistema baseado num CPU a funcionar a uma frequência de 40 MHz com uma taxa de execução de 16 MIPS (16x106 instruções por segundo, CPI = 2.5). Pretende-se processar por interrupção eventos externos periódicos que ocorrem a uma frequência de 200 kHz. Para cumprir este requisito e sabendo que o *overhead* total do atendimento a uma interrupção é 75 ciclos de relógio, calcule o número máximo de instruções máquina que a rotina de serviço à interrupção pode ter.

16 MIPS = 16 000 000 IPS

CPI = 2,5

Overhead = 75 ciclos de relógio = 75/2,5 = 30 instruções

(ISR + 30) \* 200 000 = 16 000 000 ⬄ 200k ISR + 6 000 000 = 16 000 000 ⬄ 200k ISR = 10 000 000 ⬄

ISR = 10 000 000/200 000 = 50 instruções

33. Considere um sistema baseado num CPU a funcionar a uma frequência de 10 MHz com uma taxa de execução de 5 MIPS (5x106 instruções por segundo, CPI = 2) que processa por interrupção eventos externos periódicos. A rotina de serviço à interrupção tem 70 instruções e verificou-se experimentalmente que a máxima frequência a que os eventos externos podem ocorrer é 50 kHz. Nestas condições determine, em ciclos de relógio, qual o valor máximo que pode que pode ser usado pelo *overhead* total do atendimento.

10 MHz = 10 000 000 Hz

5 MIPS = 5 000 000 IPS

CPI = 2

RSI = 70 instruções

f = 50 000

(RSI + Overhead) \* 50 000 = 5 000 000 ⬄ 70 + Overhead = 100 ⬄ Overhead = 100 – 70 ⬄ Overhead = 30 instruções = 60 ciclos de relógio

34. Considere agora um sistema baseado num CPU a funcionar a uma frequência de 80 MHz com uma taxa de execução de 40 MIPS (40x106 instruções por segundo, i.e. CPI = 2) que processa, por interrupção, eventos externos periódicos. Se o *overhead* total do atendimento à interrupção for de 40 **ciclos de relógio**, e a rotina de serviço à interrupção tiver 20instruções, determine a máxima frequência a que esses eventos podem ocorrer para que todas as interrupções possam ser atendidas.

80 MHz = 80 000 000

40 MIPS = 40 000 000

CPI = 2

Overhead = 40 ciclos de relógio = 40/2 = 20 instruções

RSI = 20 instruções

(RSI + Overhead) \* f = 40 000 000 ⬄40 \* f = 40 000 000 ⬄ f = 40 000 000/40 ⬄ f = 1 000 000 Hz

35. Descreva sucintamente, para o caso de um sistema de interrupções vetorizadas com prioridade estabelecida por “*daisy chain*”:

a. Como é estabelecida a prioridade de resposta a interrupções simultâneas

A ordem de colocação dos periféricos na cadeia, relativamente ao CPU, determina a sua prioridade.

b. Como se designa o sinal por hardware que permite estabelecer e assegurar o funcionamento do sistema em “*daisy chain*”

Interrupt Acknowledge

36. Descreva, sucintamente, as fases temporais de atendimento a uma interrupção num sistema de interrupções vetorizadas.

O CPU deteta o pedido de interrupção e quando está em condições de atender o pedido ativa um sinal que percorre os periféricos até encontrar o periférico que gerou a interrupção.

O periférico que gerou a interrupção coloca o seu identificador (vetor) no barramento de dados.

O CPU lê esse vetor e usa-o como índice da tabela que contém os endereços das RSI ou as intruções de salto para as RSI.

37. Como designaria a organização de um sistema de atendimento a interrupções em que a identificação, pelo CPU, do periférico gerador da interrupção é feita por hardware, num ciclo de *interrupt acknowledge* durante o qual o periférico gerador da interrupção coloca o seu vetor no barramento de dados.

Daisy Chain

38. Descreva, sucintamente, o funcionamento de um sistema de interrupções baseado em "identificação da fonte por software"

O CPU vai ler o registo status de cada um dos periféricos para determinar qual dos periféricos requisitou a interrupção.

**DMA**

39. Descreva a sequência de operações para que possa ocorrer uma transferência por DMA, em modo bloco, quando o controlador de DMA pretende dar início a uma transferência

1. O CPU envia um comando ao controlador do disco (DiskCtrl): leitura de um dado setor, número de palavras.

2. O CPU programa o DMA: endereço inicial da zona de dados a transferir (Controlador do disco), endereço inicial da zona destino (Memória), número de palavras a transferir

3. O CPU pode continuar com outras tarefas

4. Quando o DiskCtrl tiver lido a informação pedida para a sua zona de memória interna, ativa o sinal Dreq do DMA (sinalizando dessa forma o DMA de que a informação está pronta para ser transferida)

5. O DMA ativa o sinal BusReq, pedindo autorização ao CPU para ser bus master, e fica à espera...

6. Logo que possa, o CPU coloca os seus barramentos em alta impedância e ativa o sinal BusGrant (o que significa que o DMA passou a ser o bus master)

7. O DMA ativa o sinal Dack e o DiskCtrl, em resposta, desativa o sinal Dreq

8. O DMA efetua a transferência

9. Quando o DMA termina a transferência

10. O CPU quando deteta a desativação do BusReq desativa também o sinal BusGrant e pode novamente usar os barramentos

11. Logo que possa, o CPU atende a interrupção gerada pelo DMA

40. Qual a operação que, tipicamente, um controlador de DMA executa quando conclui um processo de transferência de informação enquanto *bus master*.

Gerar uma interrupção para ser atendida pelo CPU

41. Descreva, sucintamente, qual a finalidade do sinal *busgrant* num sistema que suporte transferência de dados por DMA, quem gera esse sinal e em que circunstâncias

O busgrant é um sinal que tem a finalidade de alertar ao DMA que o CPU já acabou as suas tarefas e que o DMA pode assumir o controlo do Bus Master

42. Descreva, sucintamente, qual a sequência de eventos que ocorrem numa transferência por DMA, em modo *cycle stealing*, quando o controlador de DMA pretende dar início a uma transferência elementar.

No modo cycle stealing o DMA aproveita os ciclos de relogio em que o CPU não necessita do acesso ao Bus Master para ir transferindo parcialmente os seus dados

43. Descreva, sucintamente, qual a diferença entres os modos de operação “bloco” e “burst” de um controlador de DMA.

Bloco - O DMA assume o controlo dos barramentos até todos os dados terem sido transferidos

Burst - O DMA transfere até atingir o número de palavras pré-programado ou até o periférico não ter mais informação pronta para ser transferida.

44. Considere um controlador de DMA não dedicado, a funcionar em modo bloco, em que um *bus cycle* é realizado em 1 ciclo de relógio. Calcule o tempo necessário para efetuar a transferência de um bloco de dados para as seguintes condições:

a. Controlador de 32 bits, frequência de funcionamento do DMA de 500MHz, bloco de 512 words de 32 bits

T = 2ns

Tempo necessário = 512 \* 2 \* T \* 1(32 para 32bits) = 2048 ns

b. Controlador de 16 bits, frequência de funcionamento do DMA de 1GHz, bloco de 512 words de 32 bits

T = 1ns

Palavras = 512, precisa de 2 ciclos para transferir 32bits

Tempo = 512 \* 2 \* T \* 2 (16 para 32 bits) = 2048 ns

c. Controlador de 16 bits, frequência de funcionamento do DMA de 1GHz, bloco de 512 words de 16 bits

T = 1ns

Tempo = 512 \* 2 \* T \* 1 (16 para 16 bits) = 1024ns

d. Controlador de 16 bits, frequência de funcionamento do DMA de 500MHz, bloco de 2Kwords de 16 bits

T = 2ns, 16bits para 16 bits,

Tempo = 2048 \* 2 \* T \* 1 = 2048 \* 4 = 8196 ns

45. Volte a resolver o problema anterior considerando agora que um *bus cycle* é realizado em 2 ciclos de relógio e para as seguintes condições:

a. Controlador de 32 bits, frequência de funcionamento do DMA de 1GHz, bloco de 1K words de 32 bits

T = 1ns

Tempo = 1024 \* 2 \* 2T = 4096 ns

b. Controlador de 16 bits, frequência de funcionamento do DMA de 500MHz, bloco de 2K words de 32 bits

T = 2ns

Tempo = 2048 \* 2 \* 2T \* 2(16 para 32bits) = 32 768 ns

c. Controlador de 16 bits, frequência de funcionamento do DMA de 1GHz, bloco de 256 words de 32 bits

T = 1ns

Tempo = 256 \* 2 \* 2T \* 2(16 para 32bits) = 2048ns

d. controlador de 16 bits, frequência de funcionamento do DMA de 500MHz, bloco de 2Kwords de 16 bits

T = 2ns

Tempo = 2048 \* 2 \* 2T \* 1(16 para 16bits) = 16 384ns

46. Resolva as duas primeiras alíneas do problema anterior considerando agora que o controlador é dedicado.

É só dividir por 2 todos os resultados

47. Considere agora um controlador de DMA não dedicado, a funcionar em modo *cycle-stealing*, em que um *bus cycle* é realizado em 2 ciclos de relógio e o tempo mínimo entre operações elementares é 1 ciclo de relógio. Calcule o tempo mínimo necessário para efetuar a transferência de um bloco de dados para as seguintes condições:

a. Controlador de 32 bits, frequência de funcionamento do DMA de 250 MHz, bloco de 512 words de 32 bits

T = 4ns, (32 para 32bits)

Tempo = 512 \* (2T + T + 2T + T) = 122884 ns = 12,28 us

b. Controlador de 16 bits, frequência de funcionamento do DMA de 1GHz, bloco de 512 words de 32 bits

T = 1ns, (16 para 32bits)

Tempo = 512 \* 2 \* (2T + T + 2T + T) = 6144ns

c. Controlador de 16 bits, frequência de funcionamento do DMA de 500MHz, bloco de 2Kwords de 16 bits

T = 2ns, (16 para 16bits)

Tempo = 2000 \* (2T + T + 2T + T) = 24us

48. Determine o número de *bus cycles* necessários para efetuar uma transferência por um controlador de DMA dedicado a funcionar em modo bloco, dadas as seguintes condições:

a. Controlador de 32 bits, frequência de funcionamento do DMA de 500 MHz, bloco de 512 words de 32 bits

T = 2ns, (32 para 32bits)

Tempo = 512 \* T = 1024 ns

b. Controlador de 16 bits, frequência de funcionamento do DMA de 1GHz, bloco de 4K words de 32 bits

T = 1ns, (16 para 32bits)

Tempo = 4k \* T \* 2 = 8000 ns

c. Controlador de 16 bits, frequência de funcionamento do DMA de 1GHz, bloco de 512 words de 32 bits

T = 1ns, (16 para 32bits)

Tempo = 512 \* T \* 2 = 1024 ns

d. Controlador de 16 bits, frequência de funcionamento do DMA de 500MHz, bloco de 1Kwords de 16 bits

T = 2ns, (16 para 16bits)

Tempo = 1k \* T = 2048 ns

49. Determine a taxa de transferência de pico (expressa em Bytes/s) de um DMA não dedicado de 32 bits (i.e. com barramento de dados de 32 bits), a funcionar a 100 MHz em modo "*cycle-stealing*". Suponha ainda que são necessários 2 ciclos de relógio (equivalente a 2 \*TBC) para efetuar uma operação de leitura ou escrita. desse DMA. O tempo mínimo entre operações elementares deverá ser de 1TBC

T = 1/100Mhz = 0,01 \* 10 ^-6 = 10 \*10^-9 = 10ns

1TBC = 1Ciclo

Na transferência de 4Bytes -> 2TBC + 1c + 2TBC + 1c = 60ns

1Byte = 15ns

F = 66.6 MHz

50. Resolva o problema anterior, mas considerando agora as seguintes condições:

a. Controlador de 32 bits, frequência do DMA de 120 MHz, 1TBC = 2 ciclos de relógio, tempo mínimo entre operações elementares de **2 TBC**

T = 8.3ns

4Bytes -> 2T + 4T + 2T + 4T ~ 100 ns

1Byte -> 100/4 = 25ns; F = 40MHz

b. Controlador de 32 bits, frequência do DMA de 80 MHz, 1TBC = 2 ciclos de relógio, tempo mínimo entre operações elementares de **3 TBC**

T = 1/80MHz = 12.5 ns

4Bytes -> 2T + 6T + 2T + 6T = 200ns

1Byte -> 200/4 = 50ns; F = 20MHz

c. Controlador de 16 bits, frequência do DMA de 120 MHz, 1TBC = 2 ciclos de relógio, tempo mínimo entre operações elementares de **2 TBC**

T = 8.3ns

2Bytes -> 2T + 4T + 2T + 4T ~ 100ns

1Byte -> 100/2 = 50ns; F = 25MHz

d. Controlador de 16 bits, frequência do DMA de 200 MHz, 1TBC = 1 ciclos de relógio, tempo mínimo entre operações elementares de **1 TBC**

T = 5ns

2Bytes -> T + T + T + T ~ 20ns

1Byte -> 20/2 = 10ns; F = 100MHz

51. Suponha um DMA não dedicado de 32 bits (i.e. com barramento de dados de 32 bits), a funcionar a 100 MHz. Suponha ainda que são necessários 2 ciclos de relógio para efetuar uma operação de leitura ou escrita (i.e. 1 "bus cycle" é constituído por 2 ciclos de relógio).

a. Determine a taxa de transferência desse DMA (expressa em Bytes/s), supondo um funcionamento em modo bloco.

T = 1/100MHz = 10ns

4 Bytes -> 2T = 20 ns

1Byte -> 20/4 = 5ns; F = 200MHz

b. determine a taxa de transferência de pico desse DMA (expressa em Bytes/s), supondo um funcionamento em modo "cycle-stealing" e um tempo mínimo entre operações elementares de 1 ciclo de relógio ("fetch", 1T mínimo, "deposit", 1T mínimo).

T = 1/100MHz = 10ns

4 Bytes -> T + T + T + T= 40 ns

1Byte -> 40/4 = 10 ns; F = 100MHz

**TIMERS**

52. Considere um *timer* em que a relação entre as frequências de entrada e de saída é uma constante k configurável. Se colocar dois desses timers em cascata (i.e., ligados em série) com constantes de divisão k1 e k2, determine a expressão algébrica que estabelece a relação entre a frequência à entrada do primeiro (fin) e a frequência à saída do segundo (fout).

Fin1/k1 = fout1

Fout1 = fin2

fin2/k2 = fout2

Fin1 = k1\*k2 fout2

(7)

53. Descreva, por palavras suas o que se entende por *Duty Cycle* de um sinal digital periódico. Dê alguns exemplos em que a manipulação dinâmica deste valor pode ser usado em aplicações práticas.

O duty cycle é a razão entre o tempo em que um sinal está ativo e o tempo total do sinal. O duty cycle pode ser usado praticamente para diminuir intensidades de por exemplo luzes e rotações de uma ventoinha.

54. Considere um timer em que a relação entre as frequências de entrada e de saída é dada por (**k+1**) em que **k** é uma constante configurável. Determine o período do sinal de saída para os valores seguintes: Fin/Fout = k+1 => Fout = Fin/k+1

a. Frequência de entrada do *timer* for 20MHz e **k** = 1999

Fout = 20MHz/2000 = 10kHz -> T = 1/10kHz = 100 us

b. Frequência de entrada do *timer* for 40MHz e **k** = 1249

Fout = 40MHz/1250 = 32kHz -> T = 31.25us

c. Frequência de entrada do *timer* for 80MHz e **k** = 32767

Fout = 80MHz/32768 = 2441Hz -> T = 409us

d. Frequência de entrada do *timer* for 2MHz e **k** = 1023

Fout = 80MHz/32768 = 1953Hz -> T = 539us

55. Alguns dos timers que estudou e utilizou têm, como último andar do temporizador, um divisor por dois. Descreva, sucintamente, qual a razão e finalidade desse divisor por dois.

Para se obter um relógio com duty cycle de 50%

56. Considere um timer e uma unidade OC como o da figura abaixo (semelhante aos dos PIC32). Admita que a frequência do relógio TCLK é de 20MHz, que o fator de divisão do *prescaler* é 4, que o valor armazenado em PR é 2499 e que o valor em OCK é 834. Determine o período do sinal de saída e o respetivo *duty cycle*.

OC = (dutycycle \* (PR+1))/100

834 = (duty \* 2500)/100 (=) 83400 = duty\*2500 (=) duty = 83400/2500 ~ 33%

PR = (Fin/k/Fout) – 1 (=) 2500 = 5M/Fout (=) Fout = 5M/2500 = 2kHz

|  |
| --- |
|  |

57. Pretende-se gerar um sinal com uma frequência de 85 Hz. Usando o Timer T2 e supondo PBCLK = 50 MHz:

a. Calcule o valor mínimo da constante de divisão a aplicar ao prescaler e indique qual o valor efetivo dessa constante

K = ceil( 50Mhz/(65536 \* fout)) = ceil(50Mhz/5M) ~ 10 => K = 16

b. Calcule o valor da constante PR2

PR2 = (50Mhz/16/85) – 1 = 36 740,…. ~ 36741

58. Repita o exercício anterior, supondo que se está a usar o Timer T1

K = 64, PR1 = 9191

59. Pretende-se gerar um sinal com uma frequência de 100 Hz e 25% de "duty-cycle". Usando o módulo "output compare" OC5 e como base de tempo o Timer T3 e supondo ainda PBCLK = 40 MHz:

a. Determine o valor efetivo da constante de prescaler que maximiza a resolução do sinal PWM

K = ceil(40Mhz/(65536 \* fout) ) = ceil(40Mhz/6,5M) ~ 6,.. => K = 8

b. Determine o valor das constantes PR3 e OC5RS

PR3 = (40Mhz/8/100) – 1 = (5M/100) - 1 = 50k – 1 = 49999

OC5RS = (25 \* 50000) /100 = 25 \* 500 = 12500

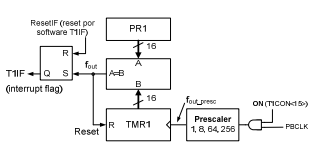
c. Determine a resolução do sinal de PWM obtido

log2(5M/100) = log2(50k) ~ 15bits

60. Considere ainda um timer como o da figura acima (semelhante aos dos PIC32) com a sua saída ligada uma unidade OC. Admita que a frequência do relógio TCLK é de 20MHz e que o fator de divisão do prescaler é 8. Determine quais os valores que deverá colocar nos registos em PR e OCK para obter na saída O1 um sinal periódico com uma frequência de 200Hz e um duty cycle de 25%:

PR = (20Mhz/8/200) – 1 = (100k/8) -1 = 12500 – 1 = 12499

OCK = (25 \* 12500) /100 = 3125

61. Considere um timer do tipo A do PIC32 (semelhante ao da figura) e um PBCLK = 20MHz. Determine o fator divisão do prescaler e o valor a colocar em PR1 para que o período de fout seja de 15ms, com a melhor precisão possível:

F = 1/15ms = 66Hz

K = ceil(20M/(65536\*66)) = ceil(20M/4M) ~ 5 => K = 8

PR1 = (20M/8/66) – 1 = 37878

62. Considere um timer com reset síncrono em que a relação entre as frequências de entrada e de saída é dada por (k+1), sendo k uma constante configurável. Determine o valor de k para as seguintes condições:

Fin / Fout = k+1

a. Frequência de entrada do timer for 20MHz e período do sinal à saída de 5ms

Fout = 1/5ms = 200Hz

K = (20M / 200) – 1 = 100k – 1 = 99999

b. Frequência de entrada do timer for 25MHz e período do sinal à saída de 1ms

Fout = 1/1ms = 1kHz

K = (25M/1k) – 1 = 25k – 1 = 24999

c. Frequência de entrada do timer for 40MHz e período do sinal à saída de 250ms

Fout = 1/1ms = 4Hz

K = (40M/4) – 1 = 1M – 1 = 999999

63. Calcule qual o tempo máximo entre resets ao um sistema de watchdog timer que gera um sinal de reset ao processador sempre que a contagem atinge o valor máximo. Admita as seguintes condições:

a. Frequência de entrada de 100 kHz, contador crescente de 16 bits

T = 10us; 10 \* 2^16 us = 655360us

b. Frequência de entrada de 20 kHz, contador crescente de 10 bits

T = 50us; 50 \* 2^10 us = 51200 us

c. Frequência de entrada de 50MHz, contador crescente de 24 bits

T = 20us; 20 \* 2^24us = 335544320 us

64. Um determinado microcontrolador disponibiliza um watchdog timer com uma frequência de entrada de 100 kHz. O programa em execução faz, por software, um reset ao watchdog timer com uma periodicidade que pode variar entre [10ms … 170ms]. Determine o número mínimo de bits do contador do watchdog timer por forma a que este nunca gere um reset ao processador.

T = 1/100k = 10us

10\*10^-6 \* 2^N = 170\*10^-3 (=) N = 13,... ⬄ N = 14

66. O programa em execução num microcontrolador faz, por software, um reset ao watchdog timer com uma periodicidade que pode variar entre [50ms … 150ms]. O watchdog desse microcontrolador usa um gerador de relógio próprio e um contador binário de 16 bits que, ao chegar ao fim de contagem, gera um reset por hardware ao microcontrolador. Dadas estas condições, e por forma a que o sistema de supervisão funcione adequadamente, determine qua a máxima frequência de relógio aplicada na entrada do watchdog.

T \* 2^16 > 150 \* 10^-3 ⬄ T > 150ms / 65536 ⬄ T > 2,3 us

F = 1/2,3us = 435 kHz

**NOÇÕES BÁSICAS DE BARRAMENTOS / DESCODIFICAÇÃO DE ENDEREÇOS**

67. Para um barramento de endereço como o indicado abaixo, que seleciona blocos de memória com 1Kbyte, suponha que no descodificador apenas se consideram os bits A15, A13 e A11, com os valores 1, 0 e 0, respetivamente.

a. Apresente a expressão lógica que implementa este descodificador:

i. Em lógica positiva

A15 . A13\ . A11\

ii. Em lógica negativa

A15\ + A13 + A11

b. indique os endereços inicial e final da gama-base descodificada e de todas as réplicas

1X0X0X



Temos: A14, A12, A10 como don’t cares, logo temos 2^3 replicas

1000 00 1000 01 1100 00 1100 01 …

[0x8000, 0x83FF] [0x8400, 0x87FF] [0xC000, 0xC3FF] [0xC400, 0xC7FF]

68. Considere o exemplo de um espaço de endereçamento indicado na figura abaixo, em que os blocos de memória têm uma dimensão de 4Kbyte. Admita agora não vamos descodificar os bits A14 e A12 do bloco dos 4 bits mais significativos, resultando na expressão CS\ = A15 + A13\

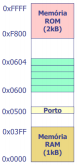
a. Determine as gamas do espaço de endereçamento de 16 bits ocupadas pela memória.

1000 1001 1100 1101

[0x8000, 0x8FFF] [0x9000, 0x9FFF] [0xC000, 0xCFFF] [0xD000, 0xDFFF]

b. Determine os endereços possíveis para aceder à 15ª posição da memória.

0x800E e 0x900E e 0xC00E e 0xD00E

69. Escreva as equações lógicas dos 4 descodificadores necessários para a geração dos sinais de seleção para cada um dos dispositivos identificados na figura ao lado.

1º [0x0000, 0x03FF] 0000 0000 CS= A15\. A14\. A13\. A12\. A11\. A10\

2º [0x0500] 0000 0101 CS= A15\. A14\. A13\. A12\. A11\. A10. A9\ . A8\

3º [0x0600, 0x0604] 0000 0110

4º [0xF800, 0xFFFF] 1111 1000

71. Para um barramento de endereço de 20 bits, semelhante ao indicado na figura, pretende-se gerar os sinais de seleção para 4 memórias de 8 kByte, a mapear em gamas de endereços consecutivas, de modo a formar um conjunto de 32 kByte. O endereço inicial deve ser configurável. Para um espaço de endereçamento de 20 bits:



1. Indique o número de bits dos campos K, S e R, supondo descodificação total

S = 2, R = 13, K = 5

c. Indique os endereços inicial e final para a primeira, segunda e última gamas de endereços possíveis de serem descodificadas

Supondo que a combinação de K bits é 00000

Primeira gama: S = 00

0000 000 0 0000 0000 0000 = 0x00000

0000 000 1 1111 1111 1111 = 0x01FFF

[0x00000, 0x01FFF]

Segunda gama: S = 01

[0x02000, 0x03FFF]

Última gama: S = 11

[0x06000, 0x07FFF]

e. Suponha que o endereço 0x3AC45 é um endereço válido para aceder ao conjunto de 32k. Indique os endereços inicial e final da gama que inclui este endereço. Indique os endereços inicial e final da memória de 8KByte à qual está atribuído este endereço

0x3AC45 = 0011 1010 1100 0100 0101, K = 111, S = 01

Endereço inicial e final da gama (S = 01) : [0x3A000, 0x3BFFF]

Endereço inicial e final: [0x38C45, 0x38D45]

(10)

72. Pretende-se gerar os sinais de seleção para os seguintes 4 dispositivos: 1 porto de saída de 1 byte, 1 memória RAM de 1 kByte (byte-addressable), 1 memória ROM de 2 kByte (byte-addressable), 1 periférico com 5 registos de 1 byte cada um. O espaço de endereçamento a considerar é de 20 bits.

b. Especifique a dimensão de todos os barramentos e quais os bits que são usados.

S tem 2bits, R tem 11 bits, K tem 7 bits