Respostas _Teoricas_2024

1. O espaço de endereçamento num sistema computacional refere-se à gama de endereços de memória que o sistema pode aceder. Esses mesmos endereços são normalmente usados para identificar e localizar os bytes de dados na memória do computador.
2. Decodificação de endereço.
3. a) Cada endereço contém 1 byte;
   b) Cada endereço contém 1 bit;
   c) Cada endereço contém 1 word (4 bytes).
4. a) Data Bus é o barramento onde os dados caminham quer venham do CPU ou para o CPU;
   b) Address Bus é o barramento que identifica qual o endereço ao qual o CPU quer aceder;
   c) Control Bus é o barramento que indica qual a operação ou instrução que está/quer que seja feita.
5. Address Bus.
6. Control Bus.

7. O cross-compiler é o compilador que traduz o código produzido pelo host em código executável para o microcontrolador.
8. Bootloader é um programa que reside sempre no microcontrolador (permanentemente na memória) e que apenas disponibiliza funções básicas de transferência e execução de programas. Não disponibiliza debug e lê informação do porto de comunicação e escreve na memória Flash.
9. a) A arquitetura de um microcontrolador é muito menos complexa do que a arquitetura de um sistema computacional;
   b) A frequência de trabalho de um microcontrolador é bastante inferior ao da frequência de um CPU de um sistema computacional;
   c) A disponibilização de periféricos de um microcontrolador é bastante maior do que de um sistema computacional;
   d) O custo de um microcontrolador é inferior ao sistema computacional;
   e) A energia consumida por um microcontrolador é inferior ao sistema computacional;
   f) Relativamente aos campos de aplicação, o microcontrolador uma vez que é usado para tarefas específicas, tem um menor número de aplicações.
10. É um sistema computacional especializado que executa apenas tarefas pré-definidas e com recursos disponíveis limitados. Tem um custo inferior a um sistema computacional. Pode ser implementado com base num microcontrolador e tem diversos exemplos de aplicação.
11. O microcontrolador é baseado numa arquitetura de Harvard, em que existem dois espaços de endereçamento independentes (um para o programa e outro para dados), o programa não pode ler dados da memória de instruções e o CPU não pode ler instruções da memória de dados. Em termos de memória o sistema usa a memória Flash e a memória RAM.

12. a) Configura o módulo de operação dos pinos I/O como entrada ou saída;
    b) É usado para escrever os valores dos pinos (portos) configurados como saída;

c) É usado para ler os valores dos pinos (portos) configurados como entrada.

13. a) Não;
    b) Sim;
    c) RD_PORT.
14. a) Sim;
    b) RD_TRIS.
15. a) Saída uma vez que está a 0;
    b) Escrita de um valor no porto de saída.
16. a) Entrada uma vez que está a 1;
    b) Leitura de um valor no porto de entrada.
17. Um flip-flop.
18. Tem o objetivo de permitir a leitura do estado atual dos pinos de entrada de forma estável e sincronizada.
19. .

20. A sua designação é I/O Module.
21. São realizadas duas tarefas, polling e interrupção. Polling consiste no facto do CPU ter de esperar que o periférico esteja disponível para a troca de informações sendo que esta espera é efetuada num ciclo de verificação da informação de status do periférico. A interrupção em que o CPU está a fazer as suas tarefas até que recebe a sinalização do periférico a dizer que está pronto para a troca de informação, assim que o CPU estiver pronto ele assume o controlo desta troca.
22. Polling.
23. A parte que interage entre o CPU e o periférico.
24. a) Porta Tri-State;
    b) De modo a diminuir o ruído.

25. a) Relativamente ao sistema de interrupções do PIC32, podemos ter dois modos de interrupções, as interrupções geradas por software onde o CPU usa polling para saber quando a interrupção é gerada ou então interrupção por hardware;
    b) A interrupção por hardware gera um sinal que um dos periféricos gera para sinalizar o CPU de que existe um pedido de interrupção;
    c) Cada periférico tem atribuído a si uma prioridade fixa ou seja, no caso de haver 2 ou mais linhas de interrupção ativas simultaneamente, o CPU atende em 1º lugar a de mais alta prioridade.
26. a) Prólogo é o conjunto de instruções que salvam os registos necessários antes de entrar na função de atendimento à interrupção;
    b) Epílogo é o conjunto de instruções de que repõem os valores corretos dos registos guardados antes de ser iniciado o atendimento à interrupção.
27. O overhead é o número de ciclos de relógio que o CPU gasta a fazer o epílogo e o prólogo de cada rotina de serviço à interrupção. Existe para saber quantas instruções é que a RSI tem verdadeiramente para atender e tratar da interrupção.
28. Overhead = 20 ciclos de relógio / 2 CPI = 10 instruções

ISR (instruction service routine) = 40 instruções
Rotina = Overhead + ISR = 50 instruções
5 MIPS = 5M
Frequência = 5M / 50 = 10^5 rotinas = 100kHz

29. Rotina * Frequência = MIPS (instruções por segundo)
    Overhead = 75 / 2.5 = 30 instruções
    Frequência = 200 kHz
    16 MIPS = 16M
    (ISR + 30) * 200k = 16M ⇔
    ⇔ 200k ISR + 6M = 16M ⇔
    ⇔ 200k ISR = 10M ⇔
    ⇔ ISR = 50 instruções
30. Overhead = 75 / 3 = 25
    Frequência = 500kHz
    33.3 MIPS = 33.3M
    (ISR + 25) * 500k = 33.3M ⇔
    ⇔ 500k ISR + 12.5M = 33.3M ⇔
    ⇔ 500k ISR = 20.8M ⇔
    ⇔ ISR = 41.6 instruções
31. ISR = 70 instruções
    Frequência = 50kHz
    5 MIPS = 5M
    (70 + Overhead) * 50k = 5M ⇔
    ⇔ 3.5M + 50k Overhead = 5M ⇔
    ⇔ 50k Overhead = 1.5M ⇔
    ⇔ Overhead = 30 instruções
32. 40 MIPS = 40M
    Overhead = 40 / 2 = 20 instruções
    ISR = 20 instruções
    Rotina = 20 + 20 = 40 instruções
    40 * Frequência = 40M ⇔
    ⇔ Frequência = 1MHz
33. a) Os periféricos ficam ligados em cadeia, cada um tem um Interrupt Acknowledge IN e um Interrupt Acknowledge OUT. O OUT está ligado ao IN do próximo periférico, este apenas fica ativo se o periférico em questão não tiver gerado nenhum Interrupt. O primeiro a receber um Interrupt Acknowledge que tenha feito requerido a interrupção vai responder com o seu vetor, sendo esta a interrupção que vai ser executada ou seja, a ordem de colocação dos periféricos na cadeia, relativamente ao CPU, determina a sua prioridade;
    b) Interrupt Acknowledge.
34. Genericamente, o procedimento de identificação da fonte de interrupção num esquema de interrupções vetorizadas em que os periféricos estão organizados numa cadeia daisy chain é o seguinte:
    - Quando o CPU deteta o pedido de interrupção ("Ireq") e está em condições de o atender ativa o sinal "Interrupt Acknowledge", ("Inta");
    - O sinal "Inta" percorre a cadeia até ao periférico que gerou a interrupção;

- O periférico que gerou a interrupção coloca o seu identificador (vetor) no barramento de dados e bloqueia a propagação do sinal "Interrupt Acknowledge";
- O CPU lê o vetor e usa-o como índice da tabela que contém os endereços das RSI ou instruções de salto para as RSI.

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

37. No modo bloco, o CPU envia um comando ao controlador do disco (DiskCtrl) para leitura de um dado setor, número de palavras. Em seguida, o CPU programa o DMA, indicando o endereço inicial da zona de dados a transferir (no controlador do disco), o endereço inicial da zona de destino (na memória) e o número de palavras a transferir. Após programar o DMA, o CPU pode prosseguir com as outras tarefas.
    Quando o DiskCtrl conclui a leitura da informação solicitada e a armazena na zona de memória interna, ele ativa o sinal Dreq do DMA, indicando que a informação está pronta para ser transferida. O DMA, por sua vez, solicita autorização ao CPU para se tornar o bus master, ativando o sinal BusReq e aguardando.
    Assim que possível, a CPU coloca os seus barramentos em alta impedância e concede permissão ao DMA para assumir o controlo, ativando o sinal BusGrant. O DMA, então, ativa o sinal Dack, e o DiskCtrl responde desativando o sinal Dreq.
    O DMA inicia a transferência, lendo do endereço de origem para um registo interno e escrevendo do registo interno para o endereço de destino, enquanto incrementa endereços e o número de palavras transferidas. Durante esta fazer, a CPU fica impedida de aceder à memória de dados.
    Quando o DMA conclui a transferência, ele desativa o sinal Dack, cede o controlo dos barramentos (de dados, endereços e controlo) para entradas, desativa o sinal BusReq e ativa o sinal de interrupção.
    A CPU, ao detetar a desativação do BusReq, desativa o sinal BusGrant e retoma o uso dos barramentos. Em seguida, assim que possível, a CPU atende à interrupção gerada elo DMA.
38. Gerar uma interrupção para ser atendida pelo CPU.
39. O sinal BusGrant tem a finalidade de alertar o DMA que o CPU já acabou as suas tarefas e conceder-lhe (ao DMA) autorização para se tornar o bus master e controlar os barramentos do sistema durante uma transferência de dados.
40. No modo cycle Stealing o DMA aproveita os ciclos de relógio em que o CPU necessita do acesso ao bus master para ir transferindo parcialmente os seus dados.
41. 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.
42. DMA não dedicado, a funcionar em modo bloco, em que o bus cycle é realizado em 1 ciclo de relógio. Calcular o tempo necessário para efetuar a transferência de um bloco de dados.
    a) Bloco de Words = 512 words
    T = 1 / Frequência = 1 / 500M = 2 ns
    Bits das Words = 32 bits
    Bits do Controlador = 32 bits
    Tempo Necessário = Bloco de Words * 2 * T * (Bits das Words / Bits do Controlador) = 512 * 2 * 2 * (32 / 32) = 2048 * 1 = 2048 ns

b) Bloco de Words = 512 words
T = 1 / Frequência = 1 / 1G = 1 ns
Bits das Words = 32 bits
Bits do Controlador = 16 bits
Tempo Necessário = Bloco de Words * 2 * T * (Bits das Words / Bits do Controlador) = 512 * 2 * 1 * (32 / 16) = 1024 * 2 = 2048 ns
c) Tempo Necessário = 1024 ns
d) Tempo Necessário = 8192 ns

43. DMA não dedicado, a funcionar em modo bloco, em que o bus cycle é realizado em 2 ciclos de relógio.
a) Bloco de Words = 1024 words
T = 1 / Frequência = 1 / 1G = 1 ns
Bits das Words = 32 bits
Bits do Controlador = 32 bits
Tempo Necessário = Bloco de Words * 2 * 2T * (Bits das Words / Bits do Controlador) = 1024 * 2 * (2 * 1) * (32 / 32) = 4096 * 1 = 4096 ns
b) Tempo Necessário = 32768 ns
c) Tempo Necessário = 2048 ns
d) Tempo Necessário = 16384 ns
44. DMA dedicado, a funcionar em modo bloco, em que o bus cycle é realizado em 1 e 2 ciclos de relógio.
Apenas será necessário dividir por dois os resultados obtidos anteriormente.
45. DMA não dedicado, a funcionar em modo cycle-stealing, em que o bus cycle é realizado em 2 ciclos de relógio e o tempo mínimo entre operações elementares é 1 ciclo de relógio. Calcular o tempo mínimo necessário para efetuar a transferência de um bloco de dados.
a) Bloco de Words = 512 words
T = 1 / Frequência = 1 / 250M = 4 ns
Bits das Words = 32 bits
Bits do Controlador = 32 bits
Tempo Mínimo = Bloco de Words * (Bits das Words / Bits do Controlador) * (2T + T + 2T + T) = 512 * (32 / 32) * (2 * 4 + 4 + 2 * 4 + 4) = 122884 ns = 12,28 us
b) Tempo Mínimo = 6144 ns
c) Tempo Mínimo = 24 us
46. Determinar o número de bus cycles necessários para efetuar uma transferência por um controlador de DMA dedicado a funcionar em modo bloco.
47. T = 1 / 100 M = 10ns
1 T_{BC} = 1 Ciclo
2 T_{BC} + 1 T_{BC} + 2 T_{BC} + 1 T_{BC} = 60 ns
60 ns / 4 = 15 ns
Frequência = 1 / 15 ns = 66.6 MHz
48. a) T = 1 / Frequência = 1 / 120M = 8.3 ns
2 T_{BC} + 4 T_{BC} + 2 T_{BC} + 4 T_{BC} = 100 ns
100 ns / (32 / 8) = 25 ns
Frequência = 1 / 25 ns = 40 MHz
b) T = 1 / Frequência = 1 / 80 M = 12.5 ns
2 T_{BC} + 6 T_{BC} + 2 T_{BC} + 6 T_{BC} = 200 ns
200 ns / (32 / 8) = 50 ns
Frequência = 1 / 50 ns = 20 MHz

c) T = 1 / Frequência = 1 / 120 M = 8.3 ns
$2\ T_{BC} + 4\ T_{BC} + 2\ T_{BC} + 4\ T_{BC} = 100$ ns
100 ns / (16 / 8) = 50 ns
Frequência = 1 / 50 ns = 20 MHz
d) T = 1 / Frequência = 1 / 200 M = 5 ns
$1\ T_{BC} + 1\ T_{BC} + 1\ T_{BC} + 1\ T_{BC} = 20$ ns
20 ns / (16 / 8) = 10 ns
Frequência = 1 / 10 ns = 100 MHz

49. DMAC não dedicado de 32 bits, a funcionar a 100 MHz. São necessários 2 ciclos de relógio para efetuar uma operação de leitura ou escrita (isto é 1 bus cycle é constituído por 2 ciclos de relógio).
    a) Determinar a taxa de transferência desse DMAC, em modo bloco.
    T = 1 / 100 M = 10 ns
    2T = 20 ns
    20 / (32 / 8) = 5 ns
    Frequência = 1 / 5ns = 200 MHz
    b) Determinar a taxa de transferência desse DMAC, em modo cycle-stealing e um tempo mínimo entre operações elementares de 1 ciclo de relógio ("fetch", 1T mínimo, "deposit", 1T mínimo).
    T = 1 / 100 M = 10 ns
    T + T + T + T = 40 ns
    40 / (32 / 8) = 10 ns
    Frequência = 1 / 10ns = 100 MHz
    c) Os resultados serão os mesmo.

50. $f1_{out} = f1_{in} / k_1$
    $f1_{out} = f2_{in}$
    $f2_{out} = f2_{in} / k2$
    $f1_{in} = k1 * k2 * f2_{out}$
51. O duty cycle (ciclo de trabalho) de um sinal digital periódico é a proporção do tempo durante o qual o sinal está num estado ativo (normalmente alto ou 1) em relação ao período total do sinal. Em outras palavras, é a fração do período em que o sinal está no seu estado ativo. Por exemplo, se um sinal tem um período de 10 milissegundos e está ativo por 2 milissegundos dentro desse período, então o ciclo de trabalho desse sinal é de 20%. Alguns exemplos de aplicação são o controlo de velocidade de motores, o controlo de potência, a modulação de largura de pulso (pwm), etc.
52. $f_{in} / f_{out} = k + 1$
    $f_{out} = f_{in} / (k + 1)$
    a) f_{out} = 20M / (1999 + 1) = 10kHz
    T = 1 / 10k = 100 us
    b) f_{out} = 40M / (1249 + 1) = 32kHz
    T = 1 / 32k = 31.25 us
    c) T = 409 us
    d) T = 539 us
53. O divisor por dois é utilizado para dividir a frequência de saída pela metade fazendo com que o duty cycle seja de 50%.
54. Período do sinal de saída:
    $f_{out(prescaler)} = PBCLK / k_{prescaler}$ ⇔

⇔ $f_{out(prescaler)}$ = 20M / 4 = 5 MHz

f_{out} = $f_{out(prescaler)}$ / (PR1 + 1) ⇔

⇔ f_{out} = 5M / (2499 + 1) = 2000 Hz

T = 1 / f_{out} ⇔

⇔ T = 1 / 2000 = 500 us

Duty Cycle:

OCK = (duty cycle * (PR + 1)) / 100 ⇔

⇔ 834 = (duty cycle * (2499 + 1)) / 100 ⇔

⇔ 83400 = duty cycle * 2500 ⇔

⇔ duty cycle = 83400 / 2500 ⇔

⇔ duty cycle = 33.36%

55. a) $k_{prescaler}$ = PBCLK / ((PR2 + 1) * f_{out}) = 50M / ((65535 + 1) * 85) = 50M / 5570560 = 9

    Como $k_{prescaler}$ tem de ser uma potência de 2 o valor acima mais próximo é 16.

    b) f_{out} = (PBCLK / $k_{prescaler}$) / (PR2 + 1) ⇔

    ⇔ 85 = (50M / 16) / (PR2 + 1) ⇔

    ⇔ 85 * 16 * (PR2 + 1) = 50M ⇔

    ⇔ PR2 = (50M / (85 * 16)) – 1

    ⇔ PR2 = 36764
56. Mesma coisa no entanto como o prescaler apenas pode ter os valores de 1, 8, 64 e 256 temos que o $k_{prescaler}$ = 64 e fazendo as contas, o PR2 vai ser 9191.
57. a) $k_{prescaler}$ = PBCLK / ((PR2 + 1) * f_{out}) = 40M / ((65535 + 1) * 100) = 40M / 6553600 = 6

    Como $k_{prescaler}$ tem de ser uma potência de 2 o valor acima mais próximo é 8.

    b) PR3 = (PBCLK / (f_{out} * $k_{prescaler}$)) - 1 = (40M / (100 * 8)) – 1 = 49999

    OC5RS = ((PR3+ 1) * duty cycle) / 100 = ((49999 + 1) * 25) / 100 = 12500

    c) PWM = $\log_2$(PR3 + 1) = $\log_2$(49999 + 1) = 15 bits
58. PR é o PRx obviamente e o OCK o OCxRS também.

    PR = (PBCLK / (f_{out} * $k_{prescaler}$)) – 1 = (20M / (200 * 8)) – 1 = 12499

    OCK = ((PR+ 1) * duty cycle) / 100 = ((12499 + 1) * 25) / 100 = 3125
59. Para que f_{out} tenha 15 ms a sua frequência tem de ser de 1 / 0,015 = 66.6 Hz

    $k_{prescaler}$ = PBCLK / ((PR1 + 1) * f_{out}) = 20M / ((65535 + 1) * 66.6) = 4.5

    Como o prescaler de Timer1 apenas pode ter os valores de 1, 8, 64 e 256 temos que o $k_{prescaler}$ = 8.

    PR1 = (PBCLK / (f_{out} * $k_{prescaler}$)) – 1 = (20M / (66.6 * 8)) – 1 = 37537
60. f_{in} / f_{out} = k + 1

    f_{out} = f_{in} / (k + 1)

    a) f_{out} = 1 / 0,005 = 200 Hz

    200 = 20M / (k + 1) ⇔

    ⇔ k = (20M / 200) – 1 = 99999

    b) f_{out} = 1 / 0,001 = 1 kHz

    k = (25M / 1k) – 1 = 24999

    c) k = 999999
61. a) T = 1 / 100k = 10 us

    $10 * 2^{16}$ = 655360 us

    b) T = 1 / 20k = 50 us

    $50 * 2^{10}$ = 51200 us

    c) T = 1 / 50M = 20 us

    $20 * 2^{24}$ = 335544320 us
62. T = 1 / 100k = 10 us

10 us = 10 * 10^{-6} s

170 ms = 170 * 10^{-3} s

10 * 10^{-6} * 2^{N} = 170 * 10^{-3} ⇔

⇔ N = 14

63. T = 1 / 250k = 4 us

4 us = 4 * 10^{-6} s

480 ms = 480 * 10^{-3} s

4 * 10^{-6} * 2^{N} = 480 * 10^{-3} ⇔

⇔ N = 17

64. f_{in} * 2^{16} = 150 * 10^{-3} ⇔

⇔ f_{in} = 2.3 * 10^{-6}

T = 1 / (2.3 * 10^{-6}) = 434783 Hz = 435 kHz