

Aula 9

- Transferência de informação por DMA
 - Configuração hardware
 - Passos de uma transferência
- Modos de funcionamento
 - Modo "bloco"
 - Modo "*burst*"
 - Modo "*cycle-stealing*"
- Configurações

José Luís Azevedo, Arnaldo Oliveira, Tomás Silva, Bernardo Cunha

Introdução

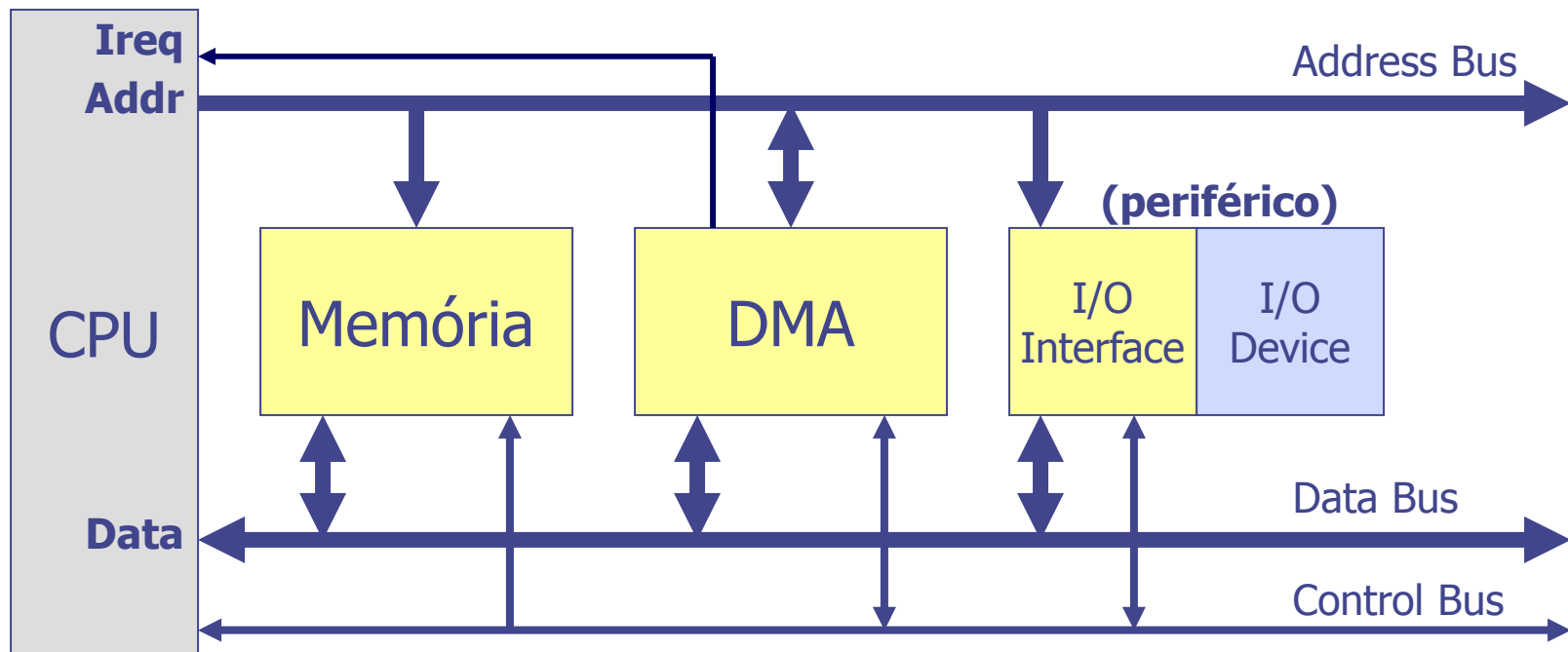
- O problema da (possível) longa latência apresentada pelos periféricos é adequadamente tratada pela técnica de E/S por interrupção
- Esta técnica não resolve, contudo, a questão da transferência a taxas elevadas, uma vez que o limite é sempre imposto pelo facto de o CPU ter que executar um programa para efetuar a transferência
- Transferir informação de um periférico para a memória implica, do ponto de vista temporal:
 - O tempo de latência de resposta à interrupção
 - O tempo para executar instruções de: leitura de uma *word* do módulo de E/S do periférico, de escrita dessa *word* no endereço destino da memória e de atualização dos endereços origem e destino
 - Para cada iteração, o tempo necessário para verificar se todos os dados foram já transferidos (envolve a leitura de um ou mais registos do módulo de E/S do periférico e a verificação dos bits de status necessários)

Introdução

- Acresce ainda o facto de que, durante o período de tempo em que o CPU está a realizar esta transferência, se encontra completamente ocupado a realizar esta tarefa, diminuindo assim a sua eficiência global na realização de outras tarefas
- A solução para o problema identificado é designada por **DMA** (do inglês *Direct Memory Access*) e consiste na transferência de informação do periférico diretamente para a memória, sem utilizar o processador
- **Exercício 1:** um programa para transferir dados de 32 bits de um periférico para a memória é implementado com um ciclo com 10 instruções. Admitindo que o CPU funciona a 100 MHz e que o programa em causa apresenta um CPI de 2, determine a taxa de transferência máxima, em Bytes/s, que se consegue obter (suponha um barramento de dados de 32 bits)
- **Exercício 2:** admita que, de uma forma não realista, o programa do exercício anterior era constituído por apenas 2 instruções, e o CPI era 1. Qual seria nesse caso a taxa de transferência?

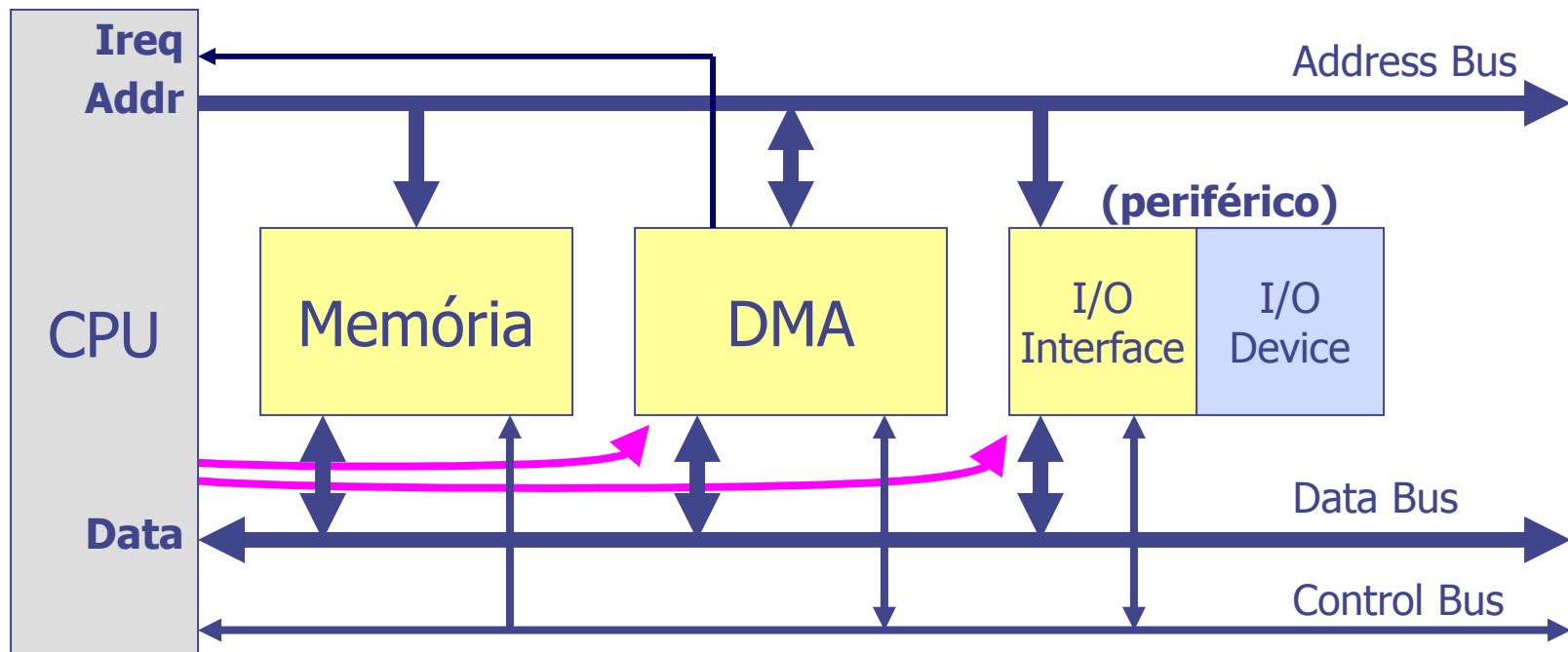
Transferência por Acesso Direto à Memória (DMA)

- Transferência de dados entre a memória e um periférico sem a intervenção do CPU. Um periférico especializado (DMA – DMA Controller) efetua essa transferência



Transferência por Acesso Direto à Memória (DMA)

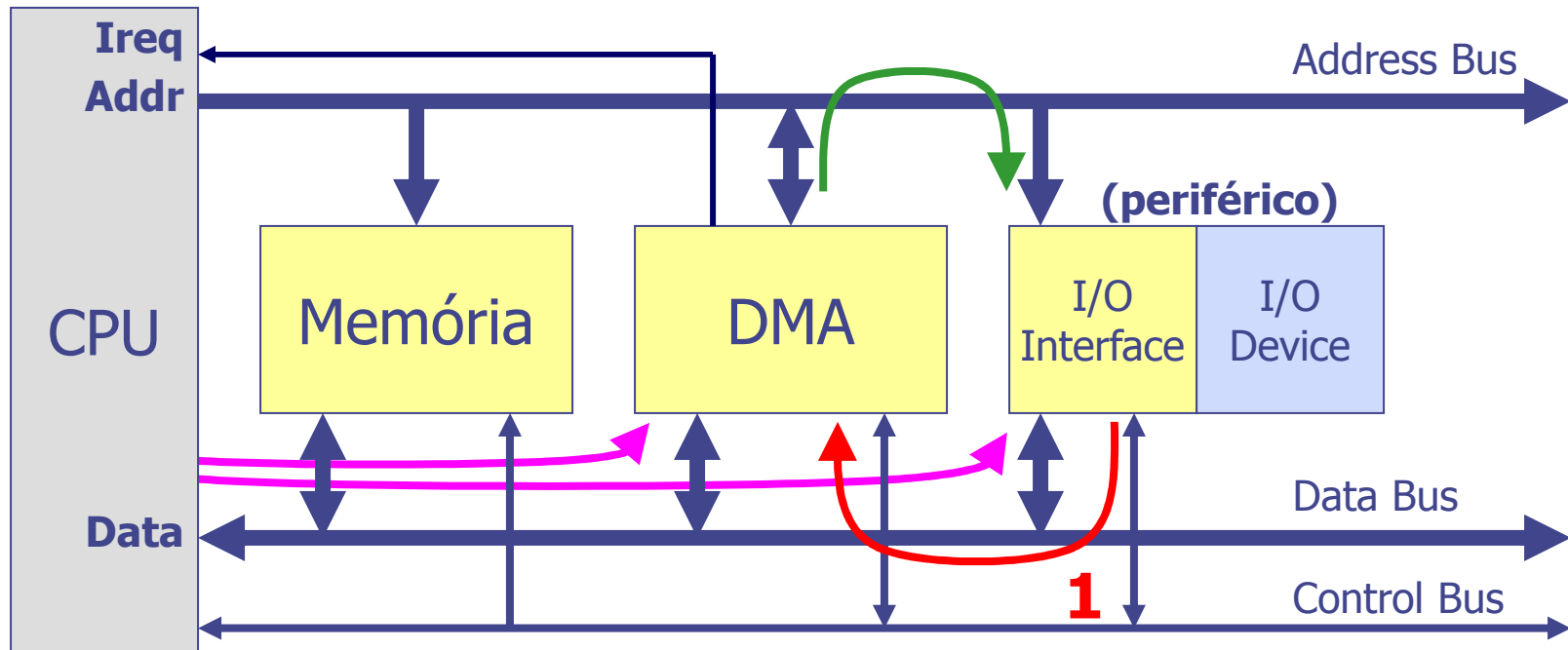
- Transferência de dados entre a memória e um periférico sem a intervenção do CPU. Um periférico especializado (DMA – DMA Controller) efetua essa transferência



- CPU programa o DMA e o periférico para efetuar uma transferência entre o mesmo e a memória

Transferência por Acesso Direto à Memória (DMA)

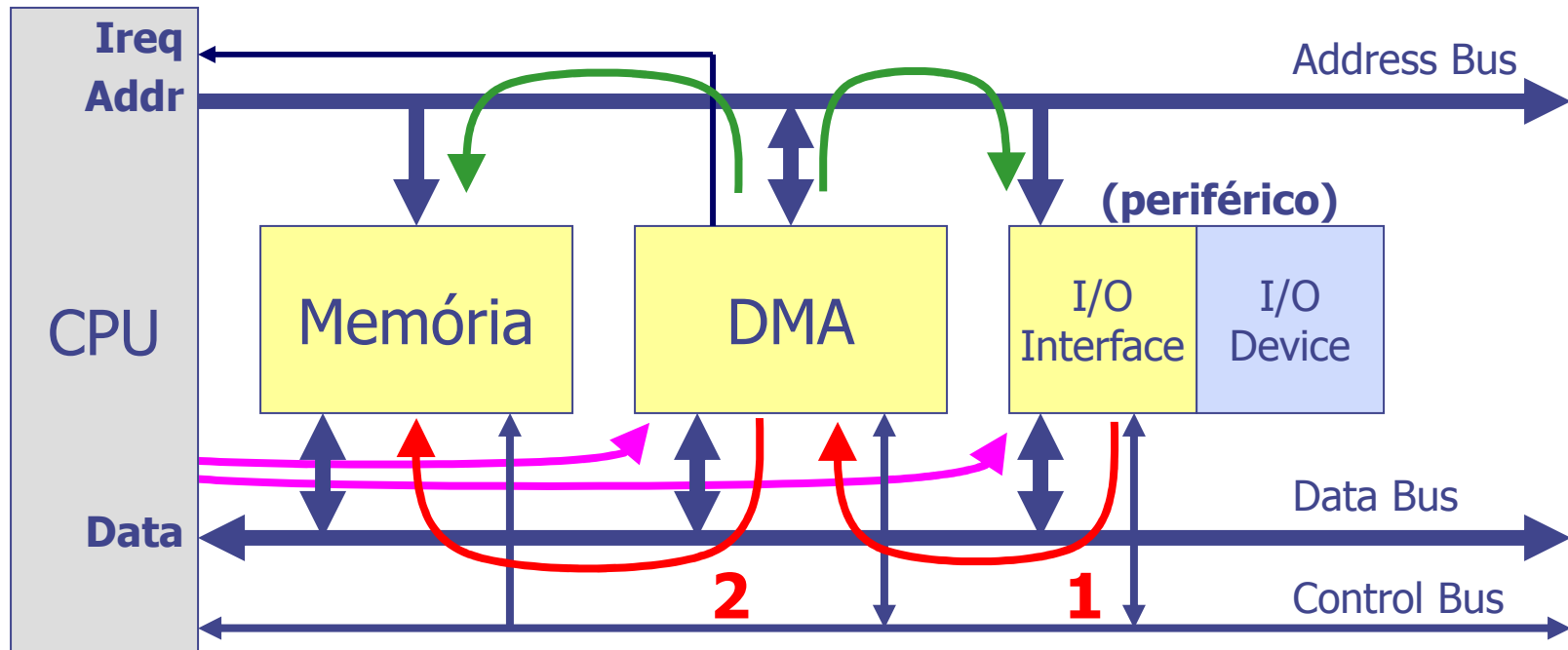
- Transferência de dados entre a memória e um periférico sem a intervenção do CPU. Um periférico especializado (DMA – DMA Controller) efetua essa transferência



- DMA lê um valor do periférico e armazena-o num registo interno

Transferência por Acesso Direto à Memória (DMA)

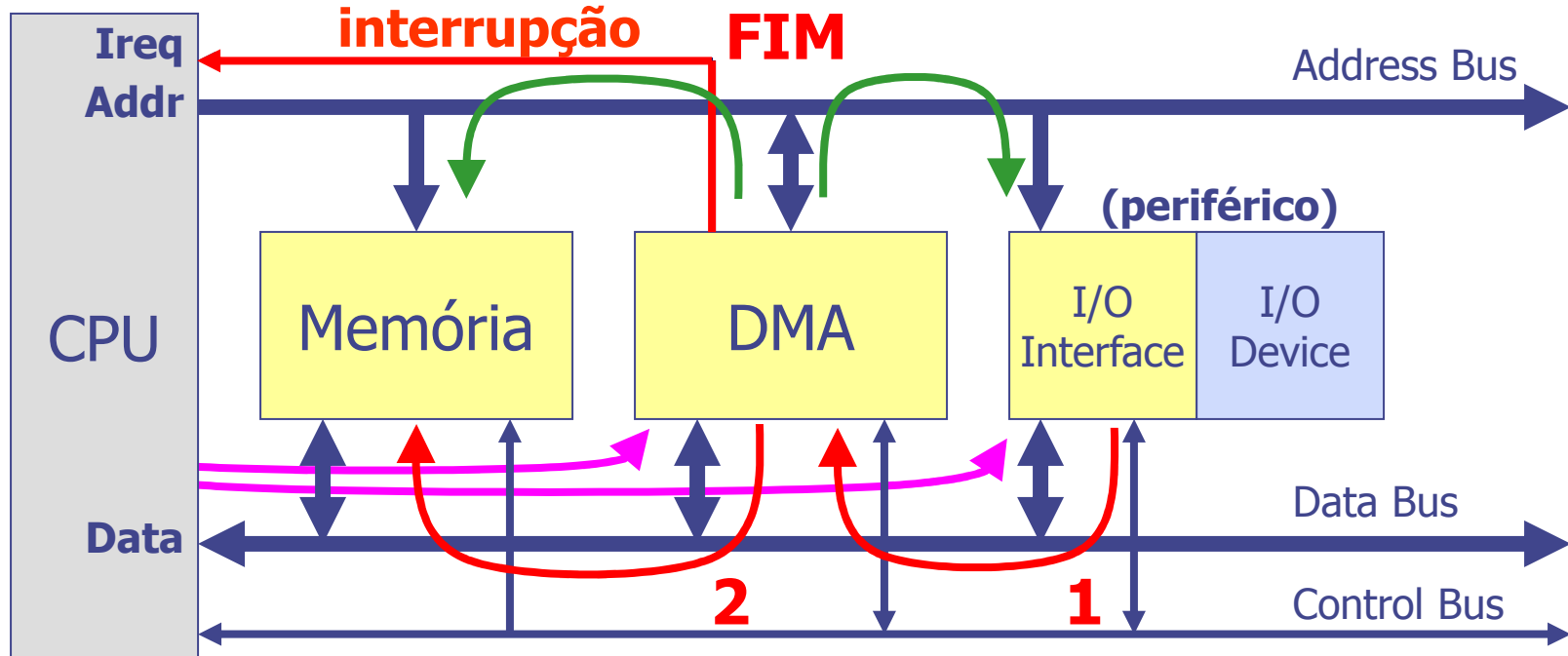
- Transferência de dados entre a memória e um periférico sem a intervenção do CPU. Um periférico especializado (DMA – DMA Controller) efetua essa transferência



- DMA copia o valor armazenado no seu registro interno para a memória

Transferência por Acesso Direto à Memória (DMA)

- Transferência de dados entre a memória e um periférico sem a intervenção do CPU. Um periférico especializado (DMA – DMA Controller) efetua essa transferência



- Quando o DMA termina a transferência de todas as palavras, gera uma interrupção
- A transferência é **exclusivamente feita por hardware** pelo DMA, i.e., não é executado qualquer programa

Controlador de DMA

- Um DMA é um periférico que, do ponto de vista do modelo de programação, é semelhante a qualquer outro periférico
- Disponibiliza um conjunto de registos internos a que o CPU acede para configurar uma transferência de dados, nomeadamente:
 - Endereço origem da informação (endereço de leitura)
 - Endereço destino da informação (endereço de escrita)
 - Quantidade de informação a transferir (nº de bytes/words)
- Pode também ter registos com informação sobre o estado de uma transferência
- Do ponto de vista da operação realizada, o DMA é um controlador especializado na transferência de dados (cópia), de forma autónoma, em hardware, após programação feita pelo CPU
- Durante a transferência, o DMA tem a capacidade de controlar os barramentos de endereços, dados e controlo, como se fosse um CPU

Controlador de DMA

- Para efetuar uma transferência de um bloco de dados de **n** palavras, o **DMA** realiza, no essencial, ao nível dos barramentos, as mesmas operações temporalmente sequenciais que seriam realizadas por um programa executado pelo CPU:
 - **Lê** uma palavra (*byte* ou *word*) do dispositivo fonte (do **source address**) para um registo interno – "**fetch**"
 - **Escreve** a palavra guardada no registo interno no passo anterior, no dispositivo destino (no **destination address**) – "**deposit**"
 - Incrementa *source address* e *destination address*
 - Incrementa o **número de palavras transferidas**
 - Se não transferiu a totalidade do bloco, repete desde o início
- Para realizar estas tarefas o DMA necessita de **controlar os barramentos** de **endereços** e de **dados** e ainda os sinais **rd** e **wr**
- Note-se que a capacidade do DMA para gerir os barramentos do sistema entra em conflito com capacidade idêntica do CPU

Controlador de DMA

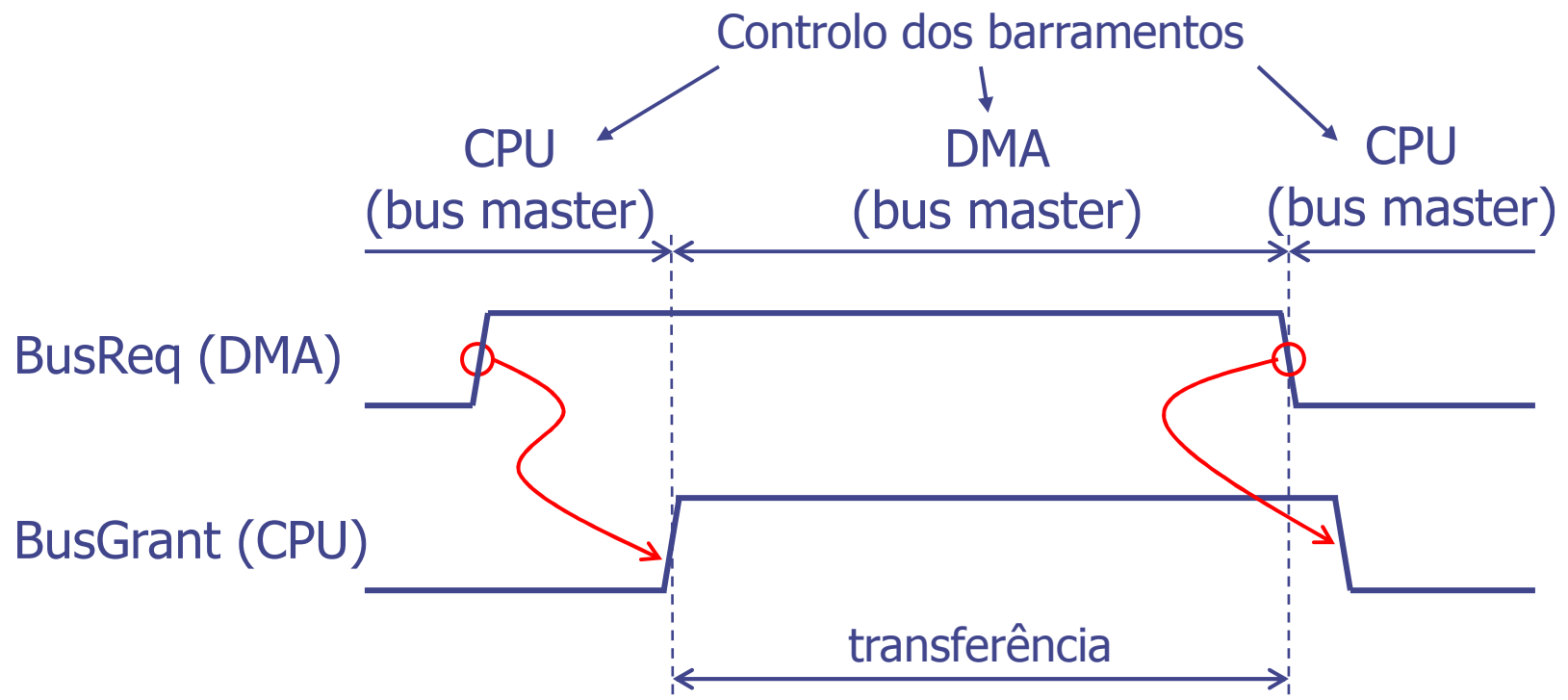
- Apenas um dos dois dispositivos, CPU e DMA, pode estar ativo no barramento de cada vez... Por defeito é o CPU
- Isto é, só pode haver, em cada instante, um "**bus master**" (só um dispositivo que é "bus master" pode controlar os barramentos)
- Quando o DMA necessita de aceder aos barramentos para realizar uma transferência, tem que primeiro ter autorização do CPU para ser o "bus master"
- O DMA só inicia a transferência de informação quando tem a confirmação, por parte do CPU, de que é "bus master"
- O controlo dos barramentos por parte do DMA é sempre temporário
- Quais são então os passos que o DMA tem que seguir para fazer uma transferência de dados ?

Controlador de DMA – passos para uma transferência

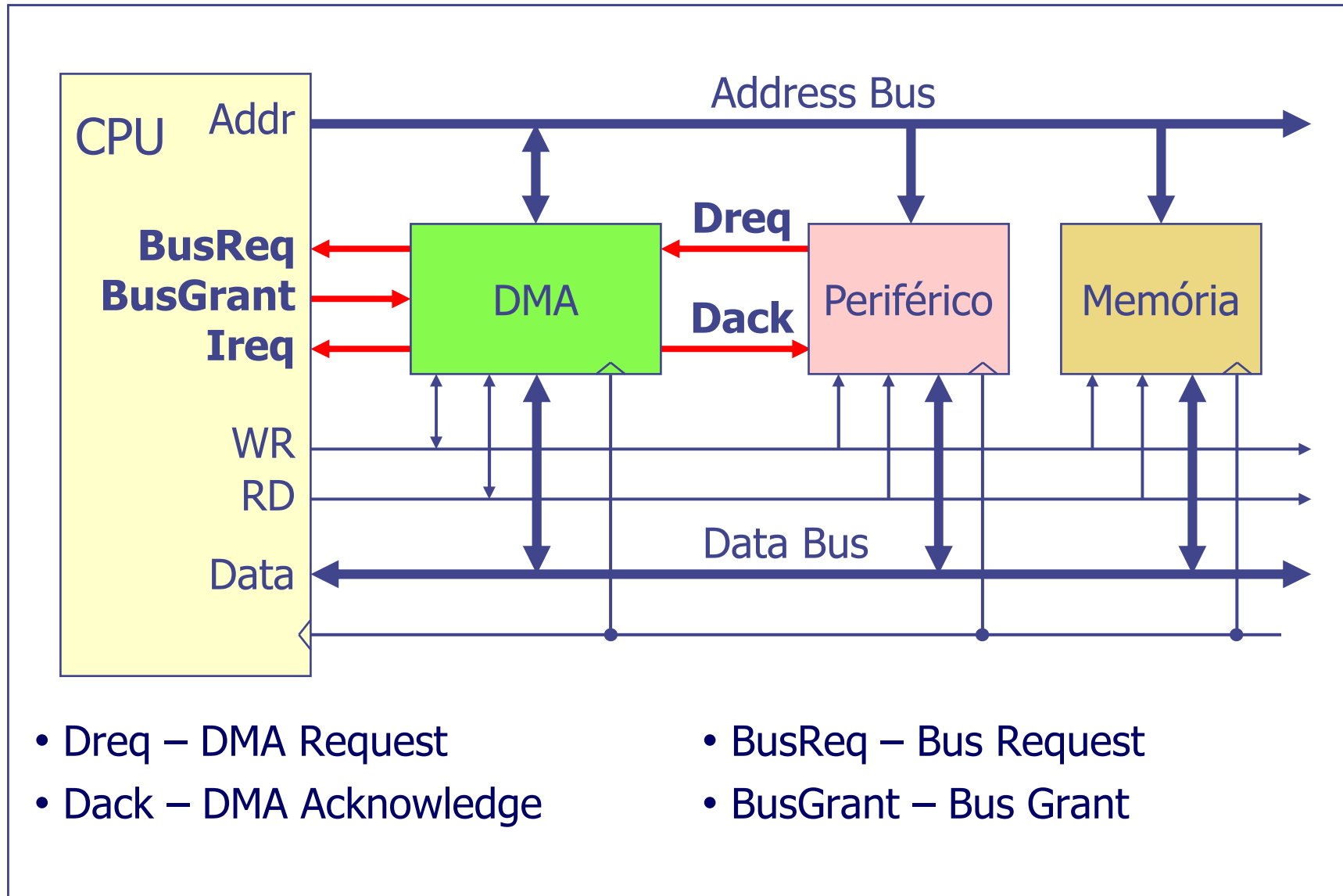
- DMA pede ao CPU para ser *bus master*
- Espera até ter a confirmação do CPU
- Efetua a transferência:
 - Lê uma palavra (*byte* ou *word*) do dispositivo fonte (do *source address*) para um registo interno
 - Escreve a palavra guardada no registo interno, no passo anterior, no dispositivo destino (no *destination address*)
 - Incrementa *source address* e *destination address*
 - Incrementa o número de palavras transferidas
 - Se não transferiu a totalidade do bloco, repete
- Retira o pedido para ser *bus master* libertando, simultaneamente, os barramentos (ou seja, as suas ligações aos barramentos de dados, de endereços e de controlo passam a funcionar como entradas)

Controlador de DMA

- Para se tornar um "bus master", o DMA:
 1. ativa o sinal "**BusReq**" (*Bus Request*) e
 2. espera pela ativação do sinal "**BusGrant**" (CPU ativa *Bus Grant* quando está em condições de libertar os barramentos)



DMA – Hardware

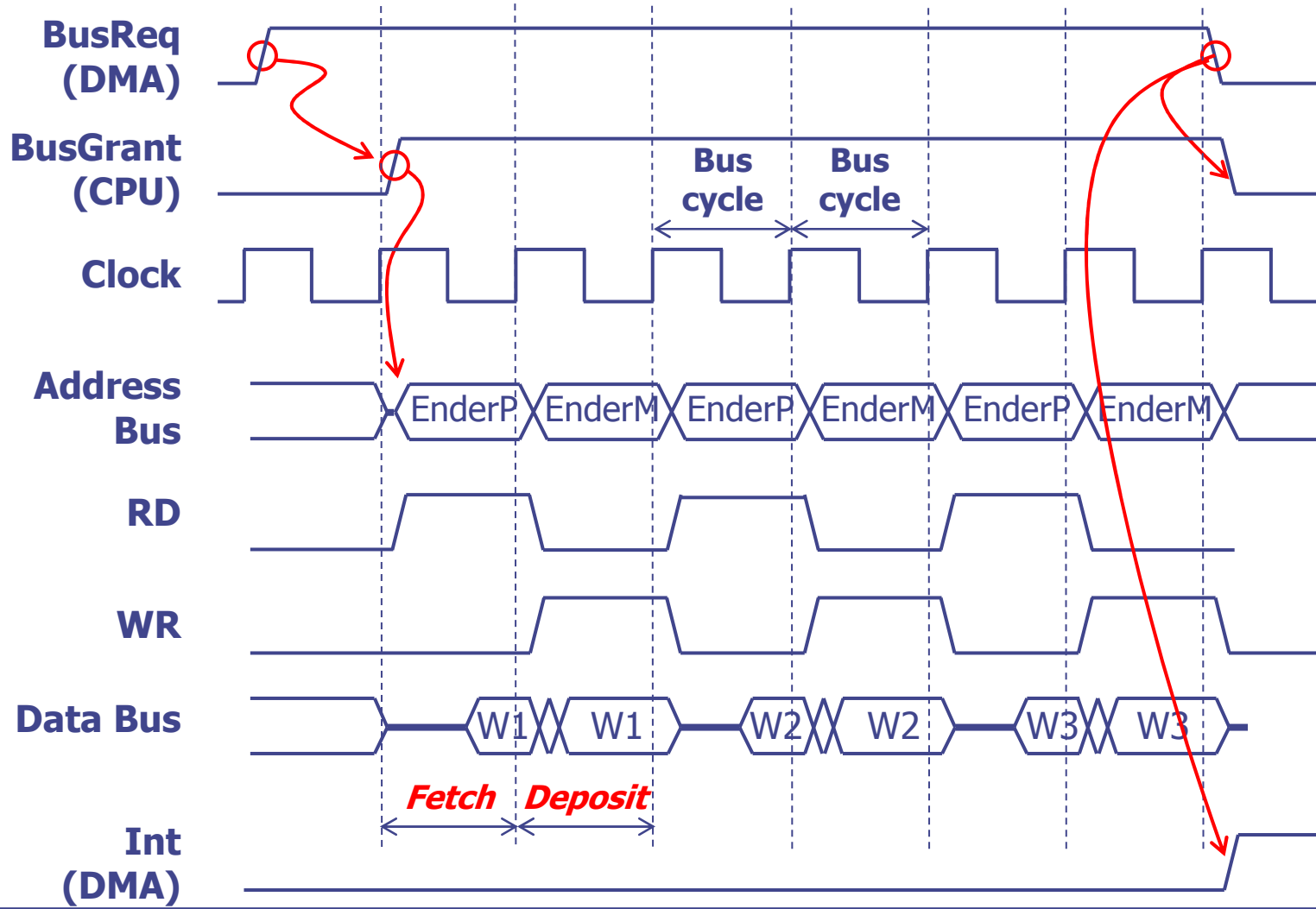


DMA – Modos de operação

- **Bloco** – o DMA assume o controlo dos barramentos até todos os dados terem sido transferidos
- **Burst** (rajada)
 - O DMA transfere até atingir o número de palavras pré-programado ou até o periférico não ter mais informação pronta para ser transferida. Se não foi transferida a totalidade da informação:
 - O periférico pode desativar o sinal Dreq o que leva o DMA a desativar o sinal BusReq e a libertar os barramentos
 - Logo que o periférico ative de novo o sinal Dreq o DMA volta a ativar o sinal BusReq e, logo que seja "bus master", continua no ponto onde interrompeu
- **Cycle Stealing**
 - O DMA assume o controlo dos barramentos durante 1 *bus cycle* e liberta-os de seguida ("rouba" 1 *bus-cycle* ao CPU) - transfere parcialmente 1 palavra (*fetch* ou *deposit*)
 - O CPU só liberta os barramentos nos ciclos em que não acede à memória (por exemplo, no estágio MEM de uma instrução aritmética na arquitetura MIPS, *pipelined*)
 - A transferência é mais lenta, mas o impacto do DMA no desempenho do CPU é nulo - o DMA aproveita os ciclos que não são, de qualquer modo, usados pelo CPU

Modo Bloco (exemplo)

- Transferência de 3 *words* de um periférico para a memória



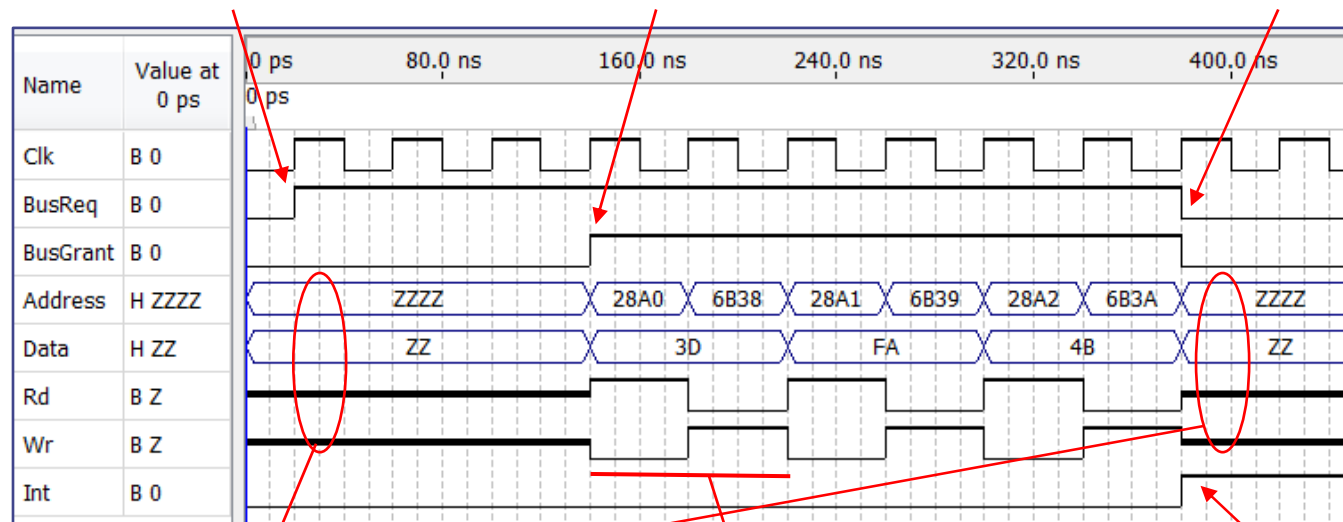
Modo Bloco (exemplo)

- Exemplo de uma transferência de 3 bytes:
 - Endereço de início na origem: 0x28A0, [0x28A0, 0x28A2]
 - Endereço de início no destino: 0x6B38, [0x6B38, 0x6B3A]

DMA ativa BusReq

CPU responde com BusGrant

DMA desativa BusReq

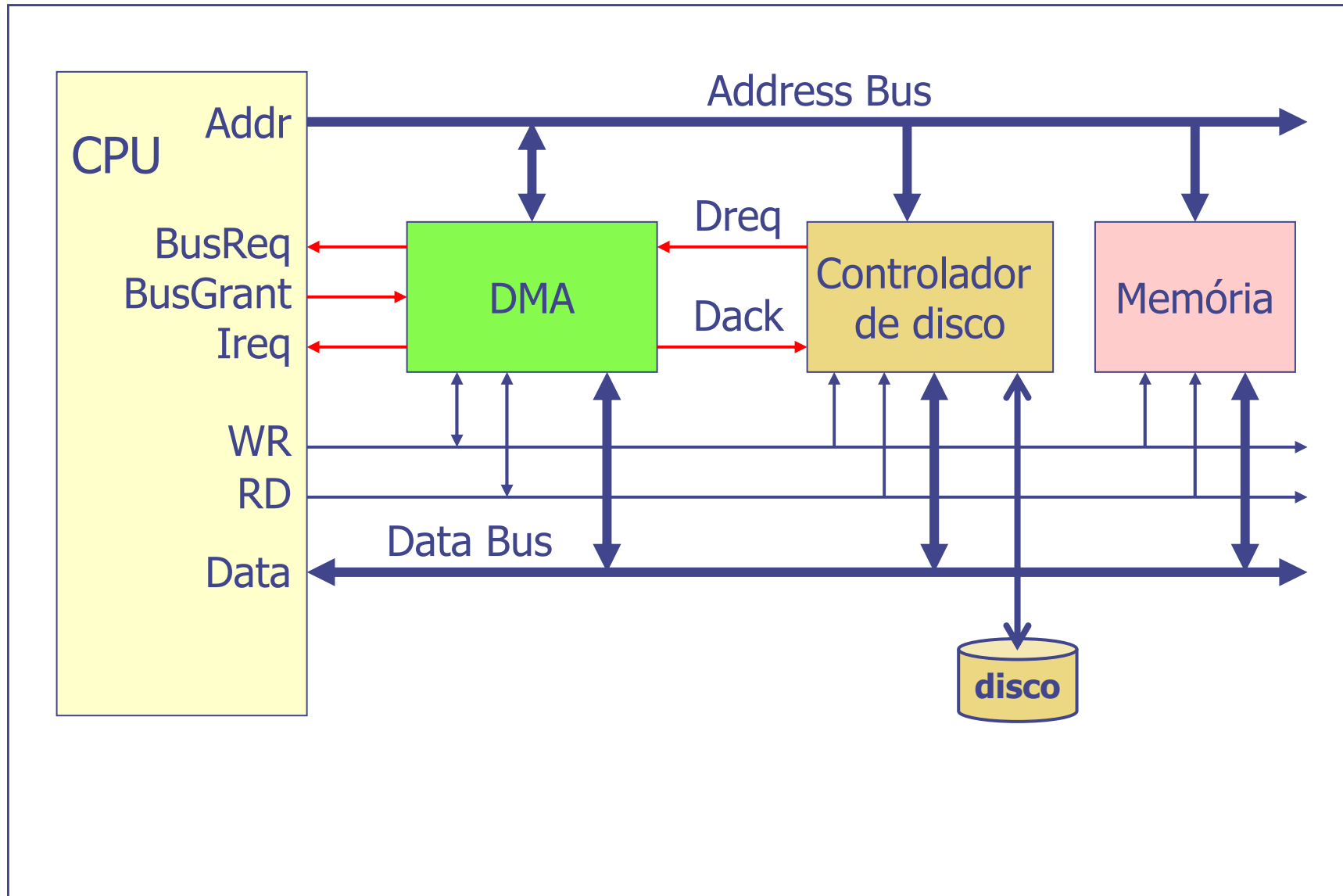


Barramentos do DMA em alta impedância

Transferência do 1º byte

DMA gera interrupção no fim da transferência

DMA – Hardware (exemplo)



Modo Bloco – Exemplo de uma transferência por DMA

1. O CPU envia um comando ao controlador do disco (DiskCtrl): leitura de um dado sector, 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**

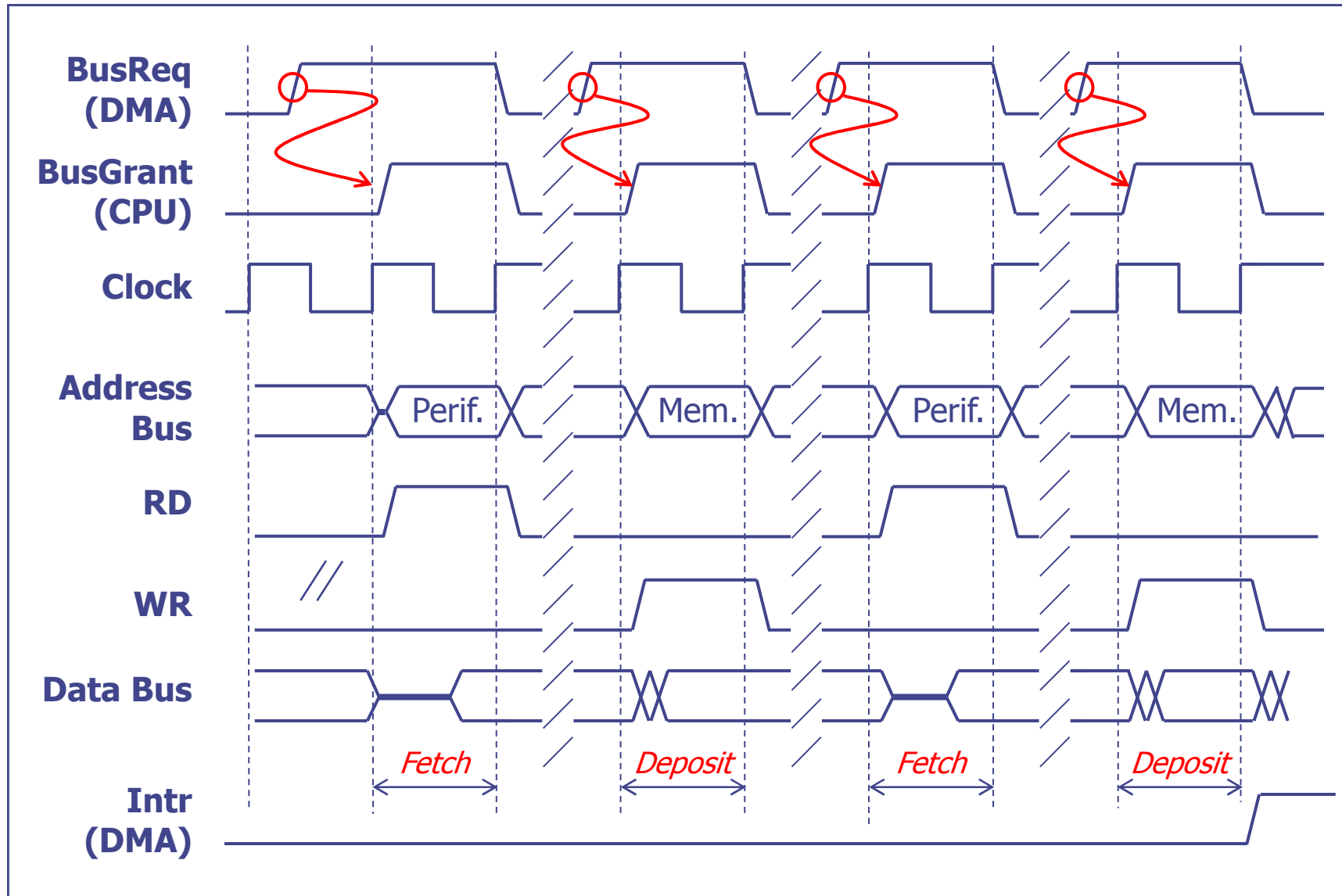
Modo Bloco – Exemplo de uma transferência por DMA

8. O DMA efetua a transferência: i) lê do endereço-origem para um registo interno e ii) escreve do registo interno para o endereço-destino (incrementa endereços e número de palavras transferidas). Durante esta fase o CPU está impedido de aceder à memória de dados
9. Quando o DMA termina a transferência:
 - Desativa o sinal **Dack**
 - Deixa de controlar os barramentos, isto é, os barramentos de dados, de endereços e de controlo passam a ser entradas
 - Desativa o sinal **BusReq**
 - Ativa o sinal de interrupção
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

Modo "Cycle-Stealing"

- Numa transferência em modo "*cycle-stealing*", o DMA efetua a seguinte sequência de operações:
 - 1. Torna-se bus master**
 - 2. Fetch:* lê 1 palavra do dispositivo fonte (do *source address*)
 - 3. Liberta os barramentos**
 4. Espera durante um tempo fixo pré-determinado (Ex. 1T)
 - 5. Torna-se bus master**
 - 6. Deposit:* escreve 1 palavra no dispositivo destino (no *destination address*)
 - 7. Liberta os barramentos**
 8. Incrementa *source address* e *destination address*
 9. Incrementa o nº de bytes/words transferidos
 10. Espera durante um tempo fixo pré-determinado (Ex. 1T)
 11. Se não transferiu a totalidade do bloco, repete desde 1
 12. Após a transferência da totalidade dos dados, ativa o sinal de interrupção

Modo "Cycle-Stealing" (exemplo)

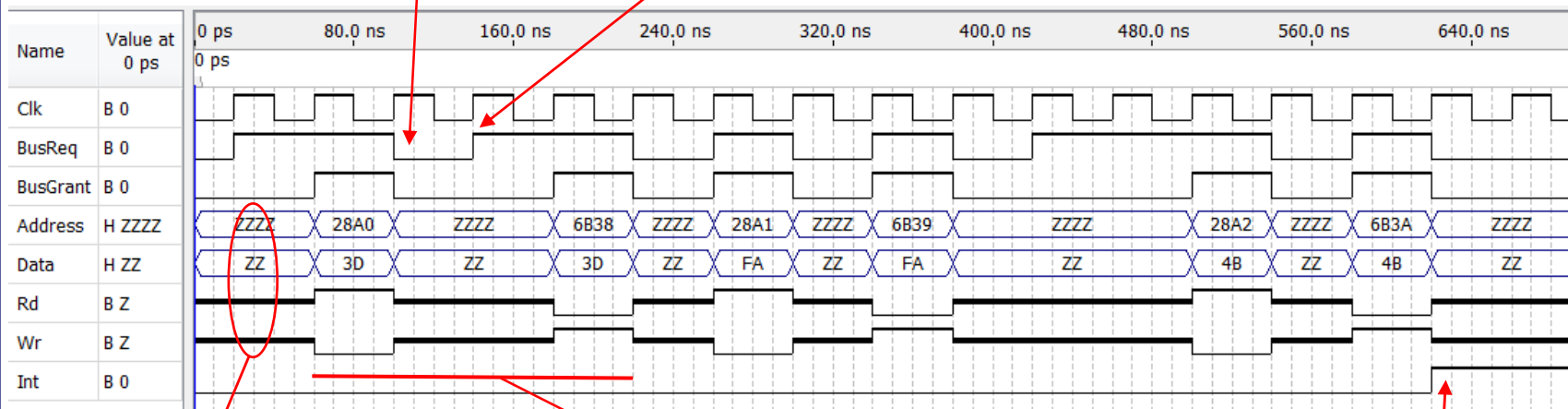


Modo "Cycle-Stealing" (exemplo)

- Exemplo de uma transferência de 3 bytes:
 - Endereço de início na origem: 0x28A0, [0x28A0, 0x28A2]
 - Endereço de início no destino: 0x6B38, [0x6B38, 0x6B3A]

DMA desativa BusReq

DMA reativa BusReq

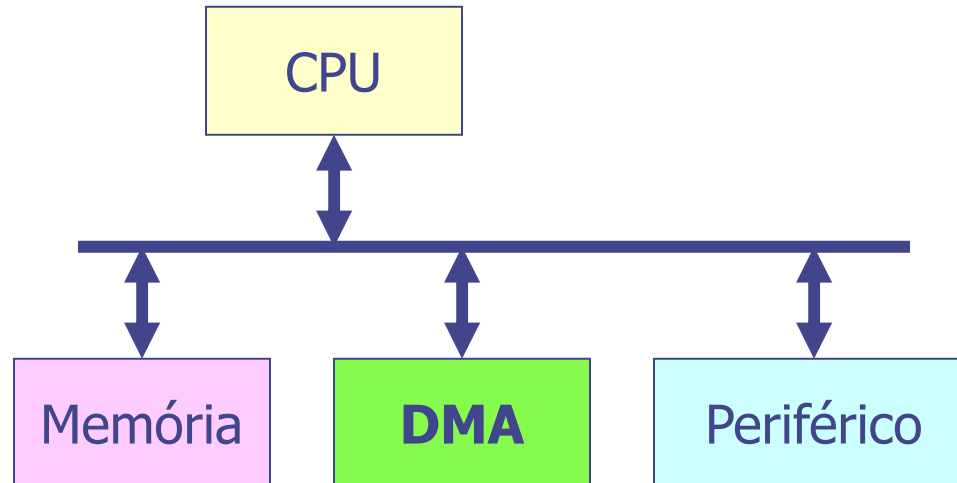


Barramentos do DMA em alta impedância

Transferência do 1º byte

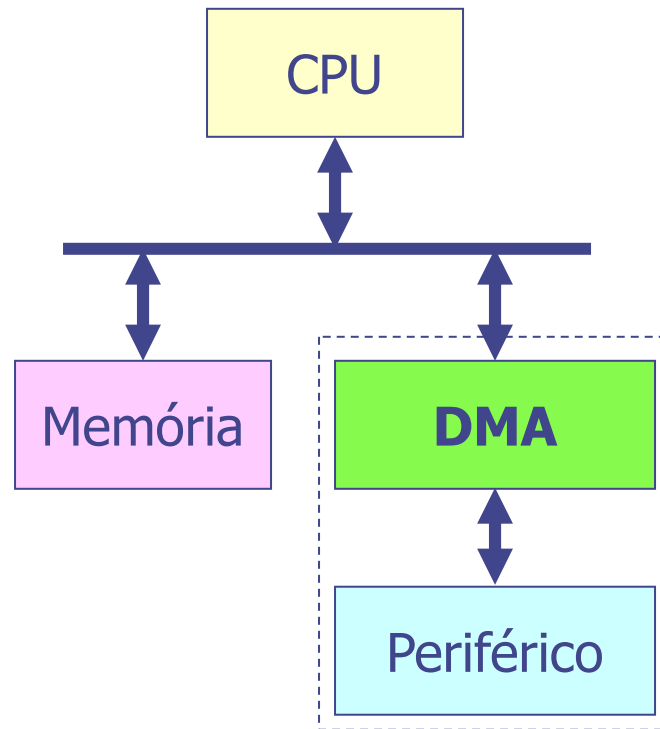
DMA gera interrupção no fim da transferência

Configurações – Canal de DMA



- O DMA fornece um serviço de transferência genérico
- Pode ser usado para transferir informação:
 - de I/O para memória
 - de memória para I/O
 - de memória para memória
- Para a transferência de 1 palavra o barramento é usado 2 vezes (2 "*bus cycles*" por palavra)
- Em modo "*cycle stealing*", por cada palavra transferida, o CPU liberta os barramentos 2 vezes

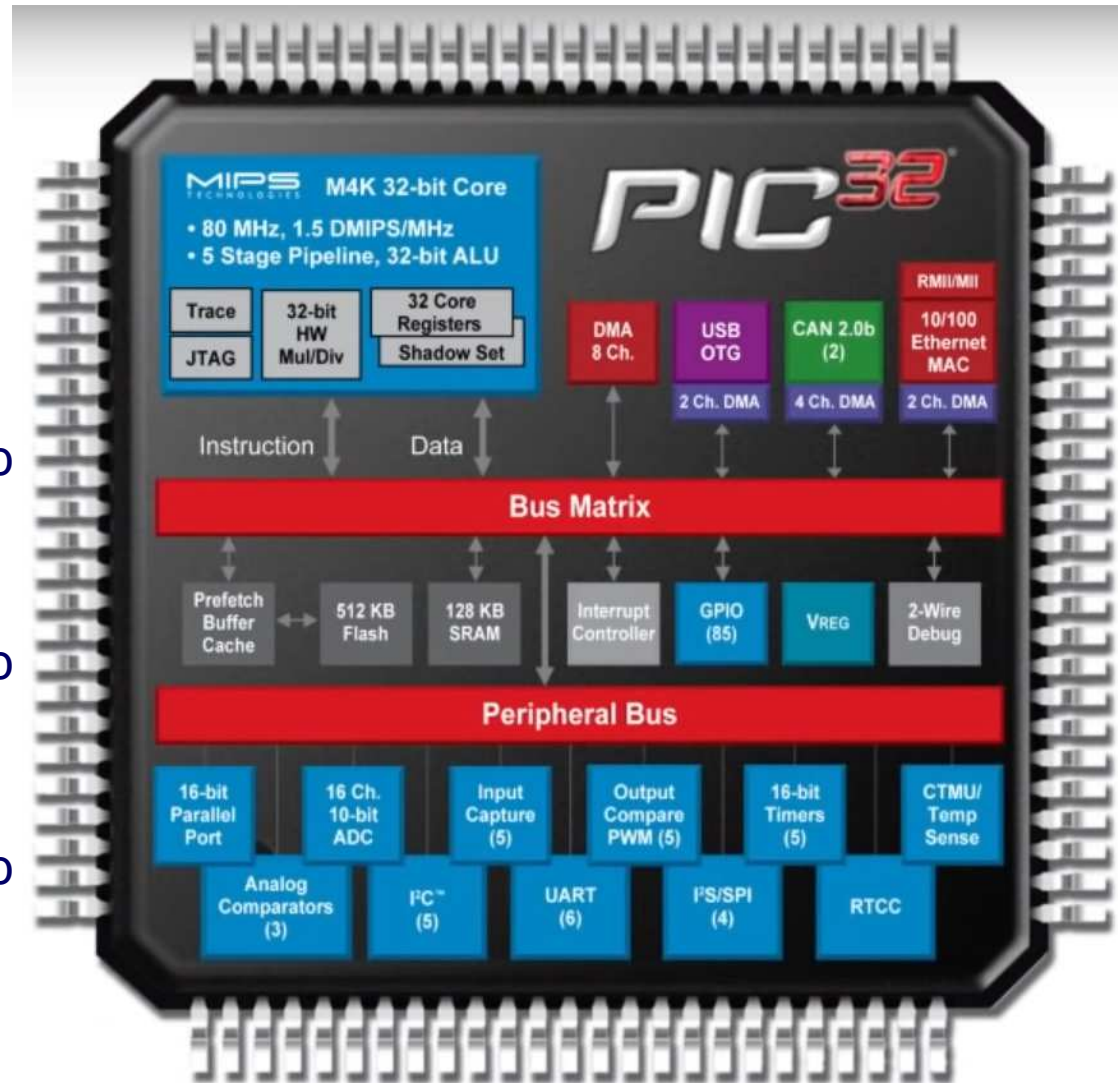
Configurações – DMA dedicado



- Um periférico tem o seu próprio DMA (**DMA dedicado**)
- Para a transferência completa de 1 palavra o barramento é usado apenas 1 vez (1 "*bus cycle*" por palavra):
 - DMA → memória ou memória → DMA

Controladores de DMA no PIC32

- Controlador DMA de 8 canais (transferência memória/periférico e memória/memória)
- Controlador dedicado no módulo USB
- Controlador dedicado no módulo CAN
- Controlador dedicado no módulo Ethernet



Exercícios

- 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).
- **Exercício 1** – Determine a taxa de transferência desse DMA (expressa em Bytes/s), supondo um funcionamento em modo bloco.
- **Exercício 2** – 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).
- **Exercício 3** – Repita os exercícios 1 e 2 supondo um DMA dedicado com as características referidas anteriormente.

Exercícios

- **Exercício 4** – Admita uma arquitetura em que o ciclo de barramento (bT) é igual $2ns$. Suponha que o CPU programou um controlador de DMA em modo "*cycle-stealing*" para ter um tempo de espera entre "*fetch*" e "*deposit*" igual a $1*bT$ e um tempo de espera entre o "*deposit*" e o próximo "*fetch*" igual a $2*bT$. Sabendo que o barramento de dados é de 16bits, determine a taxa de transferência de pico desse DMA expresso em Bytes/s.
- **Exercício 5** – Admita agora que, para as mesmas condições indicadas no exercício anterior, o tempo médio de resposta a um pedido de *BusReq* é, para um ciclo completo de transferência, de $2.5*bT$. Qual seria, nesse caso, a taxa média de transferência nesse período (expressa em palavras/s).
- **Exercício 6** – Repita os exercícios 4 e 5 supondo um DMA dedicado com as características referidas anteriormente.