14 - (DMA) Trabalhe por mim !

Rafael Corsi - rafael.corsi@insper.edu.br

Maio - 2017

Introdução

Direct Memory Access (DMA)

Figura 1: Linha 4 CPTM

Ações de um software

Quais são as ações mais comuns de um software ?

Ações de um software

Quais são as ações mais comuns de um software ?

- Esperar por dados (ou ações)
- Processar dados
- Movimentar dados
  - Receber/ transferir

Movimentar dados, origens e destinos.

Movimentar dados, origens e destinos.

- Memória –> Memória
- Memória –> Periférico
- Periférico –> Memória

Memória -> Memória : memcpy

```
memcpy(void *p_src, void *p_dst, int nWords)
```

- r0 : p_src
- r1 : p_dst
- r2 : nWords

Implementação em assembler :

```
wordcopy
    LDR  r3, [r0], #4 ; load a word from the source
                      ; (post increment r0 in 4)
    STR  r3, [r1], #4 ; store it to the destination
                      ; (post increment r1 in 4)
    SUBS r2, r2, #1   ; decrement the counter
    BNE  wordcopy     ; ... copy more
```

Memória –> Memória

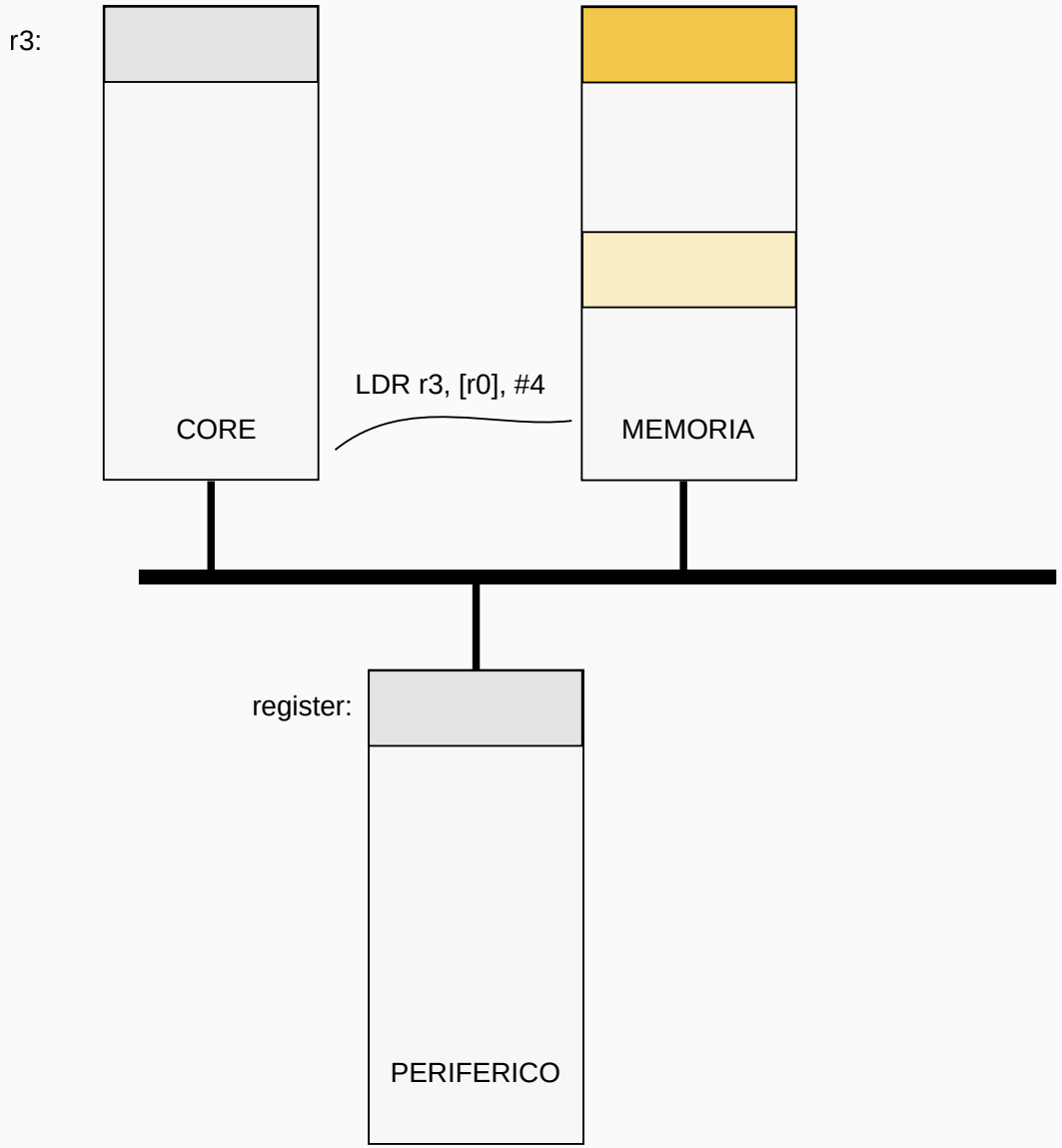


Figura 2: Load

Memória –> Memória

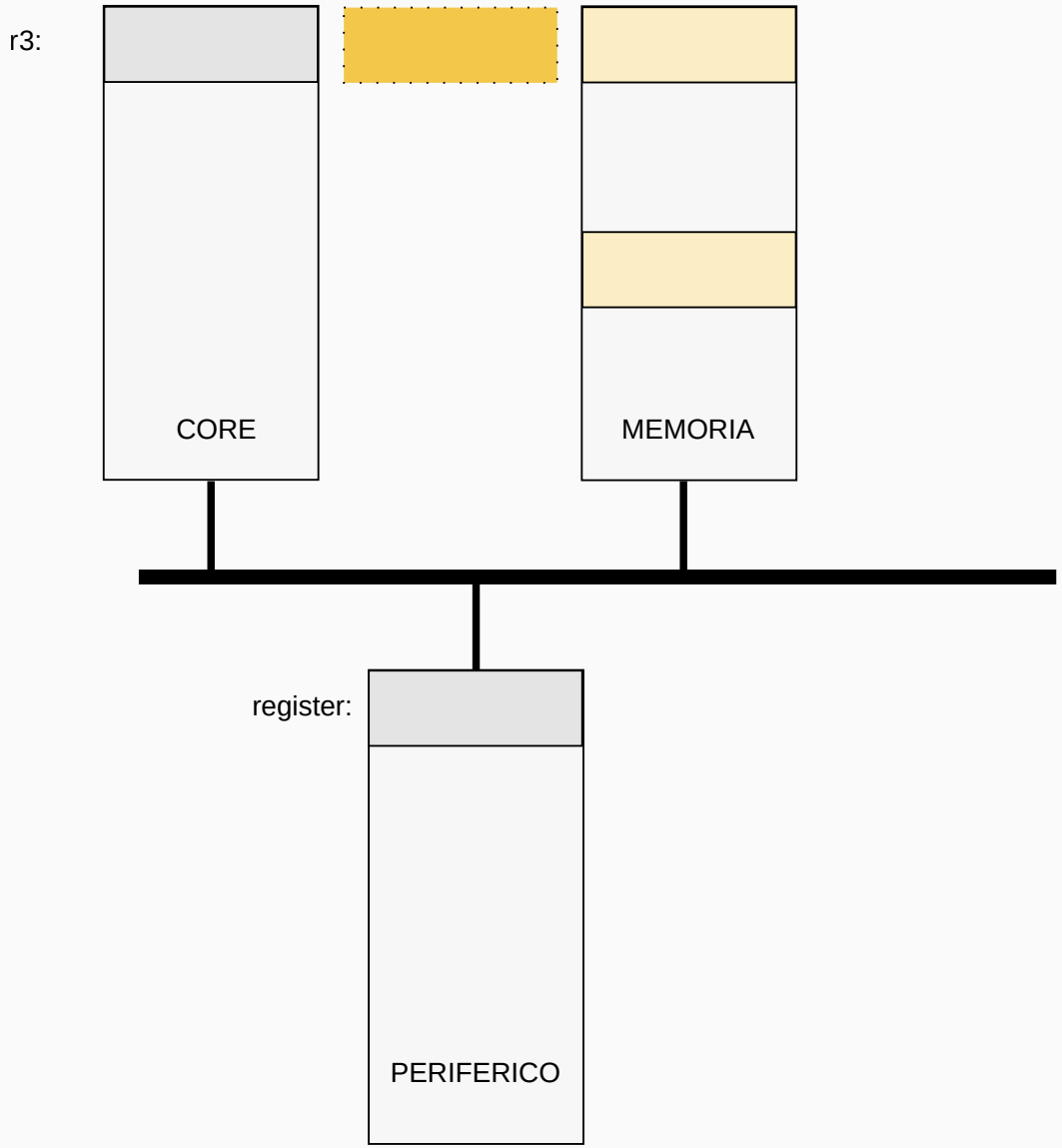


Figura 3: Transfering

Memória –> Memória

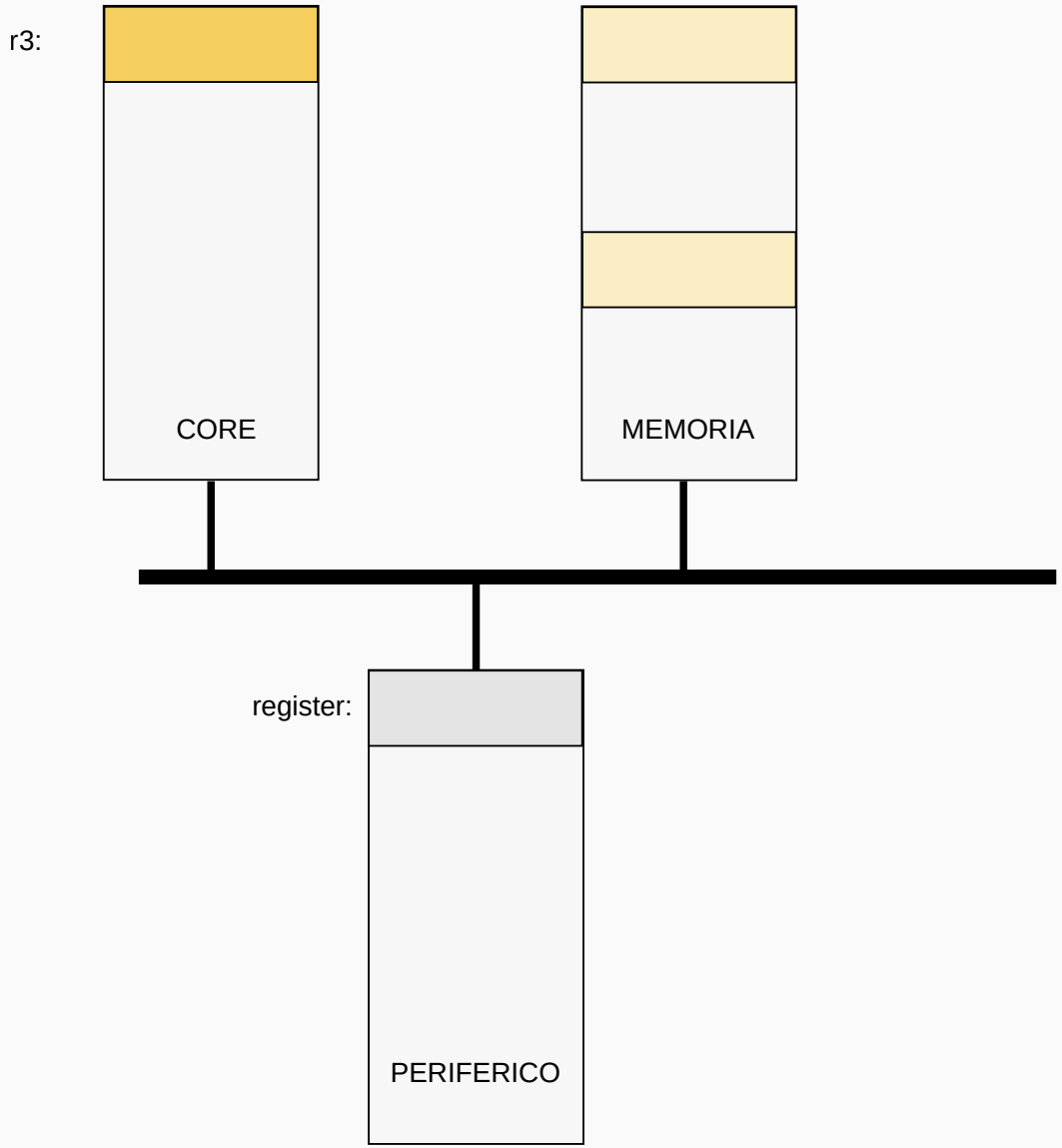


Figura 4: Register

Memória –> Memória

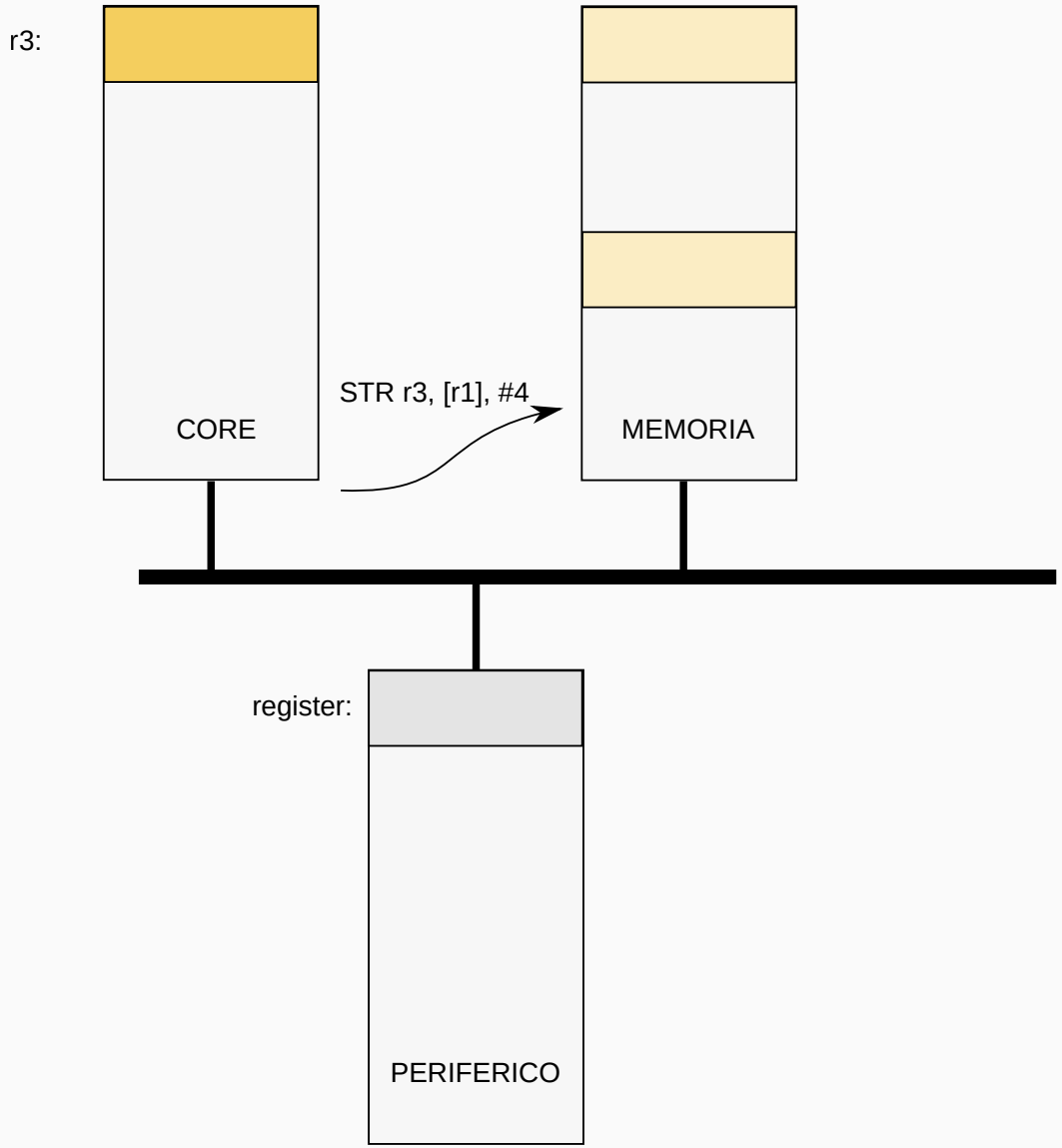


Figura 5: Load

Memória –> Memória

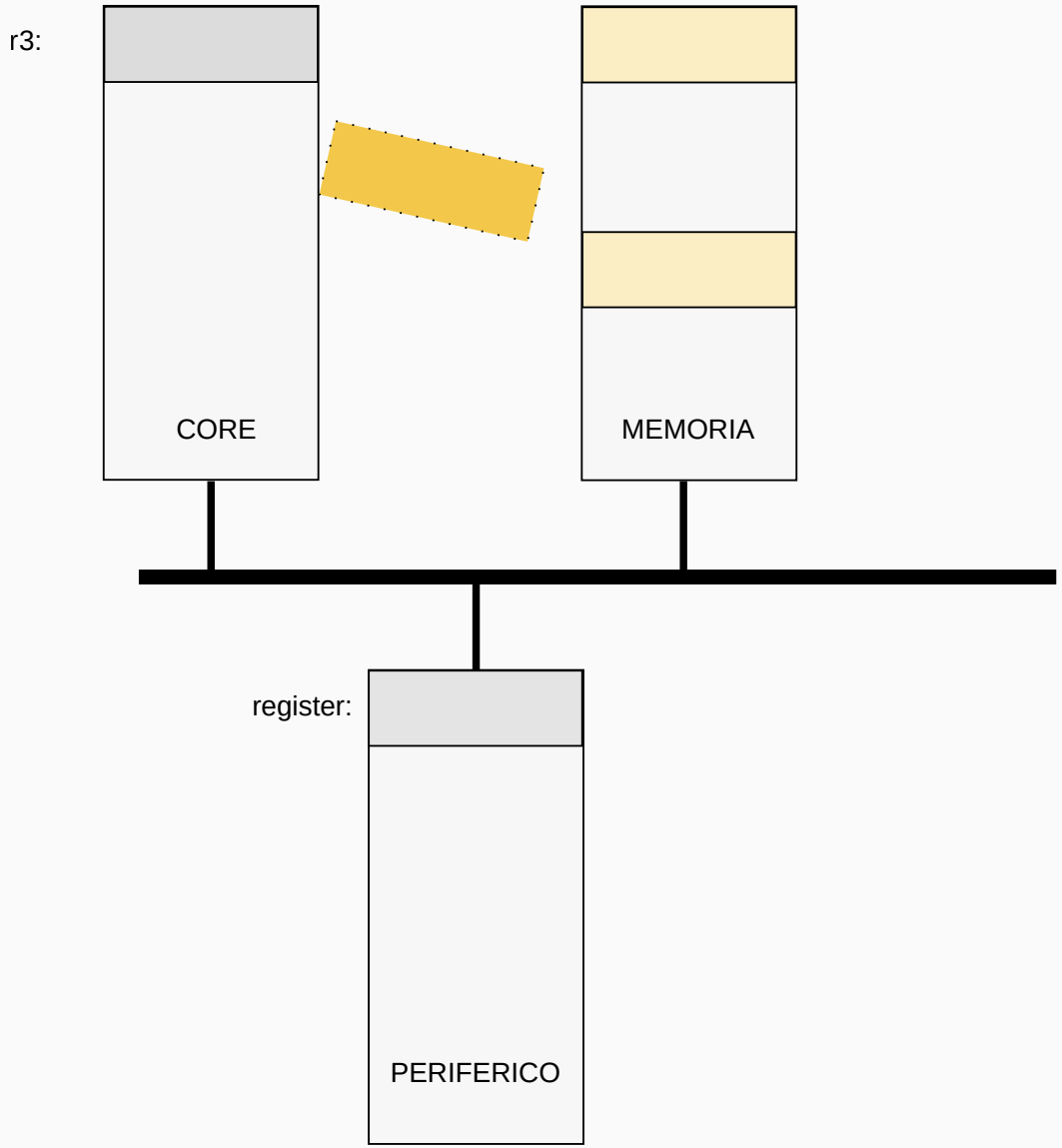


Figura 6: Loading

Memória –> Memória

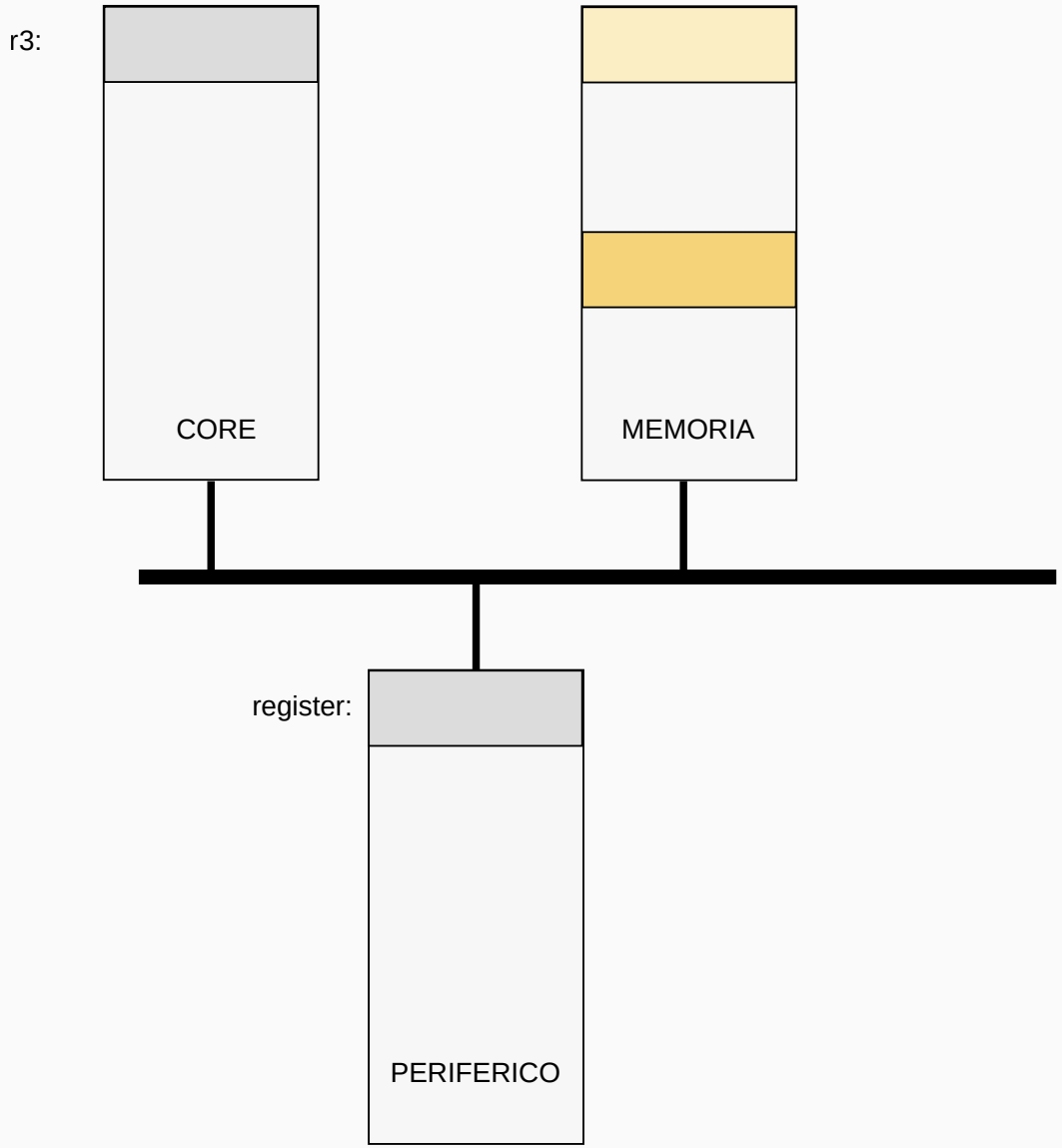


Figura 7: Memory

Memória para periférico e vice versa

UART

- Lembra do código 13 UART ?

Memória –> Periférico

```
/**
 *  Envia para o UART uma string
 */
uint32_t usart_putString(uint8_t *pstring){
  uint32_t i = 0 ;

  while(*(pstring + i)){
    usart_serial_putchar(USART0, *(pstring+i++));
    while(!uart_is_tx_empty(USART0)){};
  }

  return(i);
}
```

Periférico -> Memória

```
/*
 * Busca no UART uma string
 */
uint32_t usart_getString(uint8_t *pstring){
  uint32_t i = 0 ;

  usart_serial_getchar(USART0, (pstring+i));
  while(*(pstring+i) != '\n'){
    usart_serial_getchar(USART0, (pstring+(++i)));
  }
  *(pstring+i+1)= 0x00;
  return(i);

}
```

CORE

Nesses exemplos não existe manipulação de dados, não seria interessante que um existisse uma forma de transferência de dados que não dependesse diretamente do CORE ?

Para isso que existe o priférico :

- Direct Memmory Access (DMA)

Ele é um hardware dedicado para esse tipo de ação !

DMA : Memória –> Memória

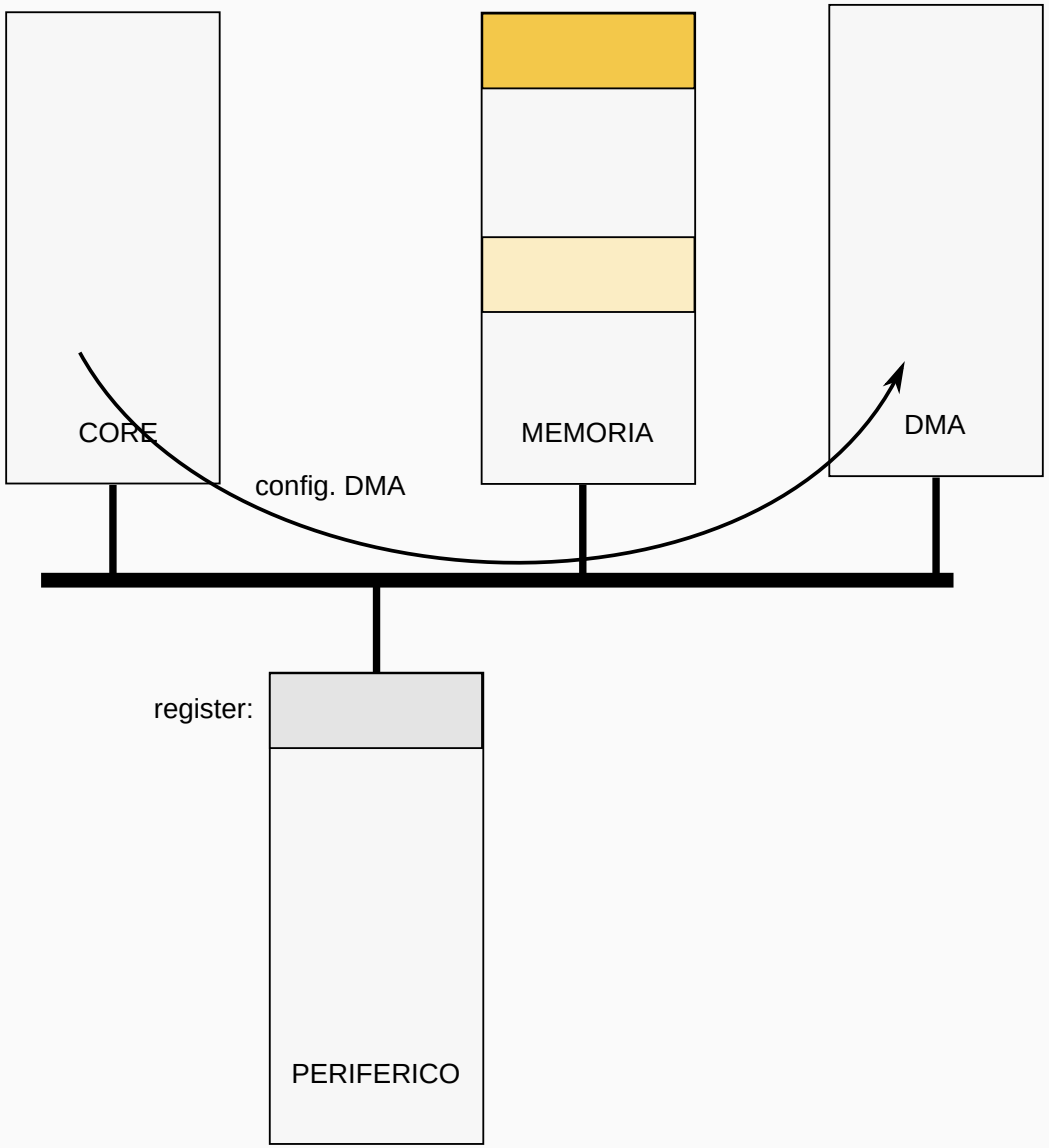


Figura 8: Configura DMA

DMA : Memória –> Memória

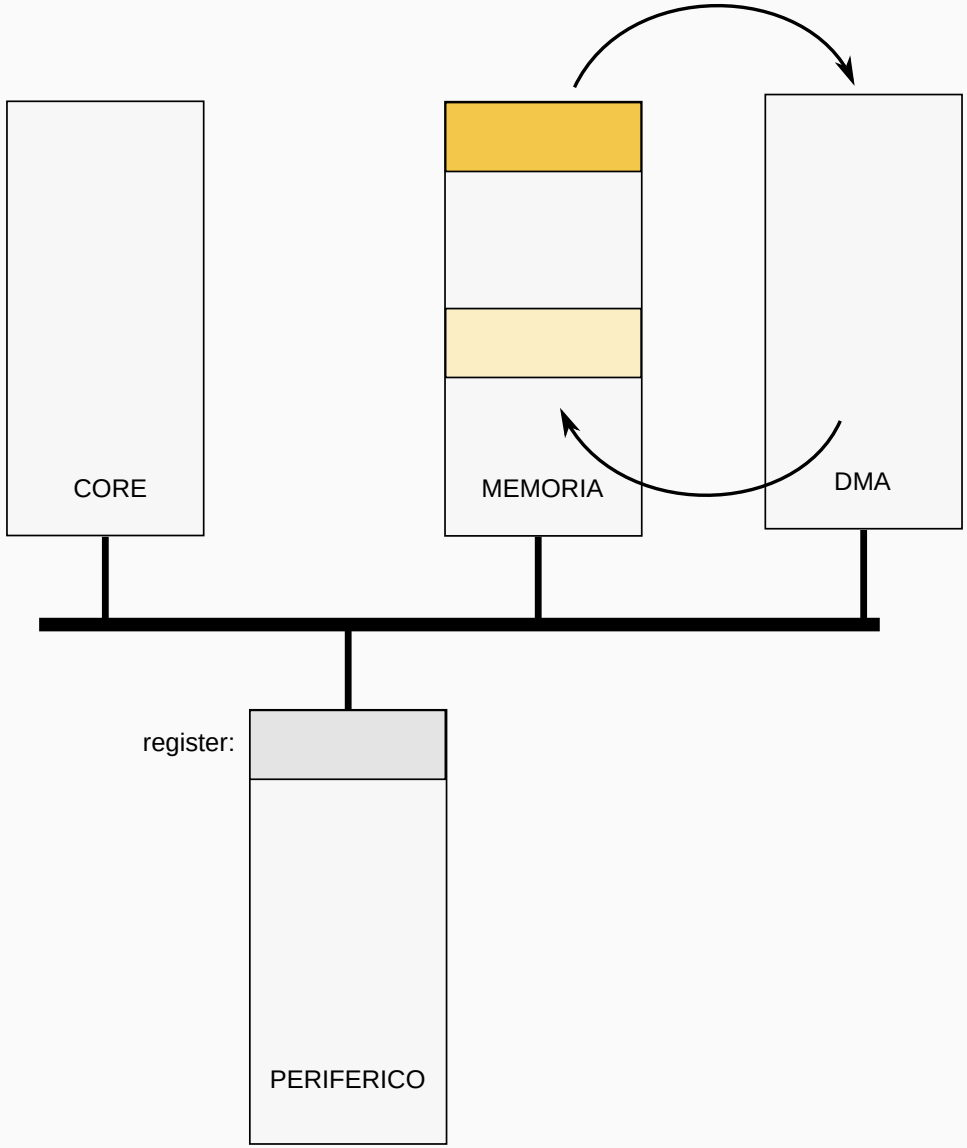


Figura 9: Transferência

DMA : Memória -> Periférico

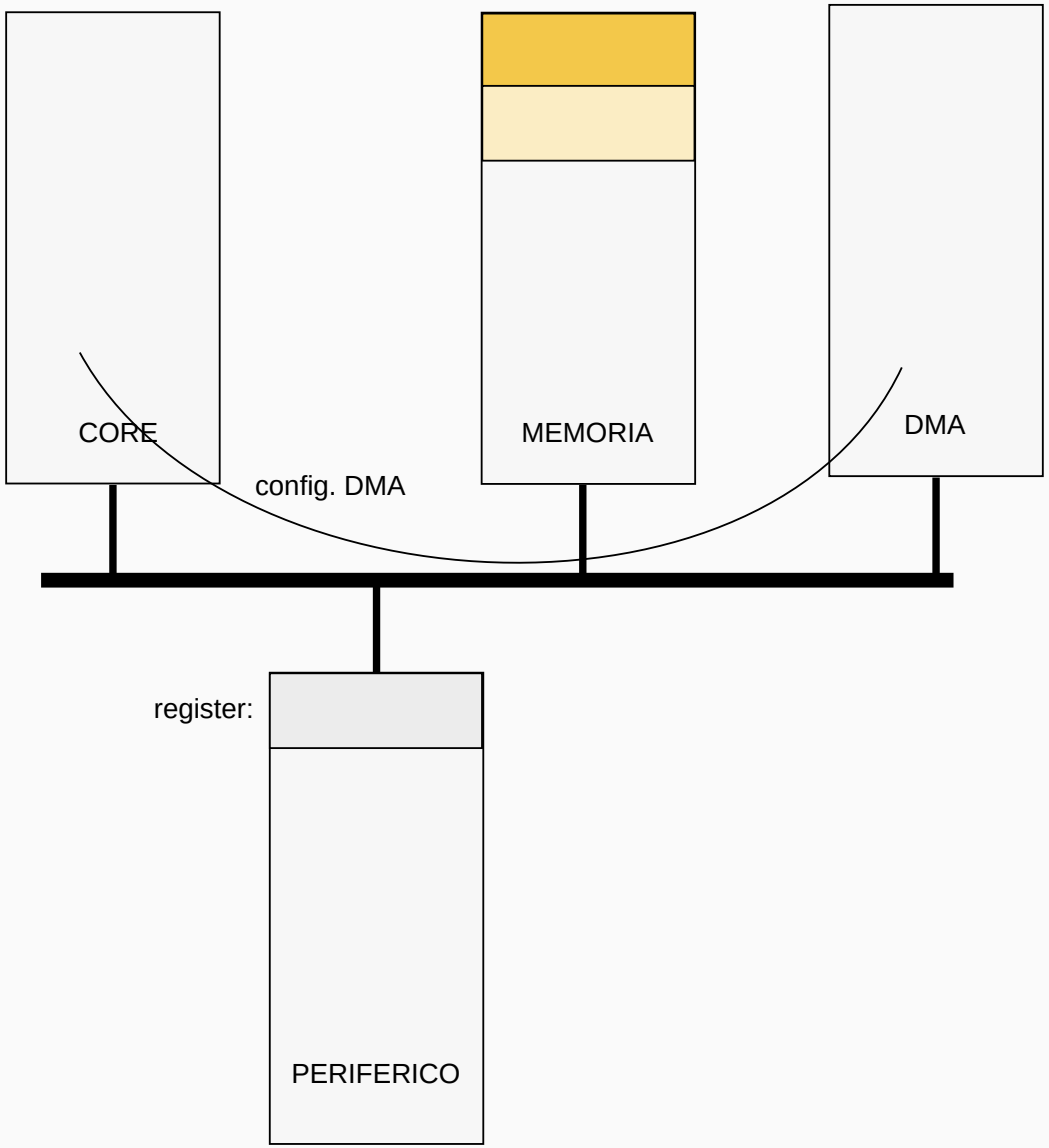


Figura 10: Configura DMA

DMA : Memória -> Periférico

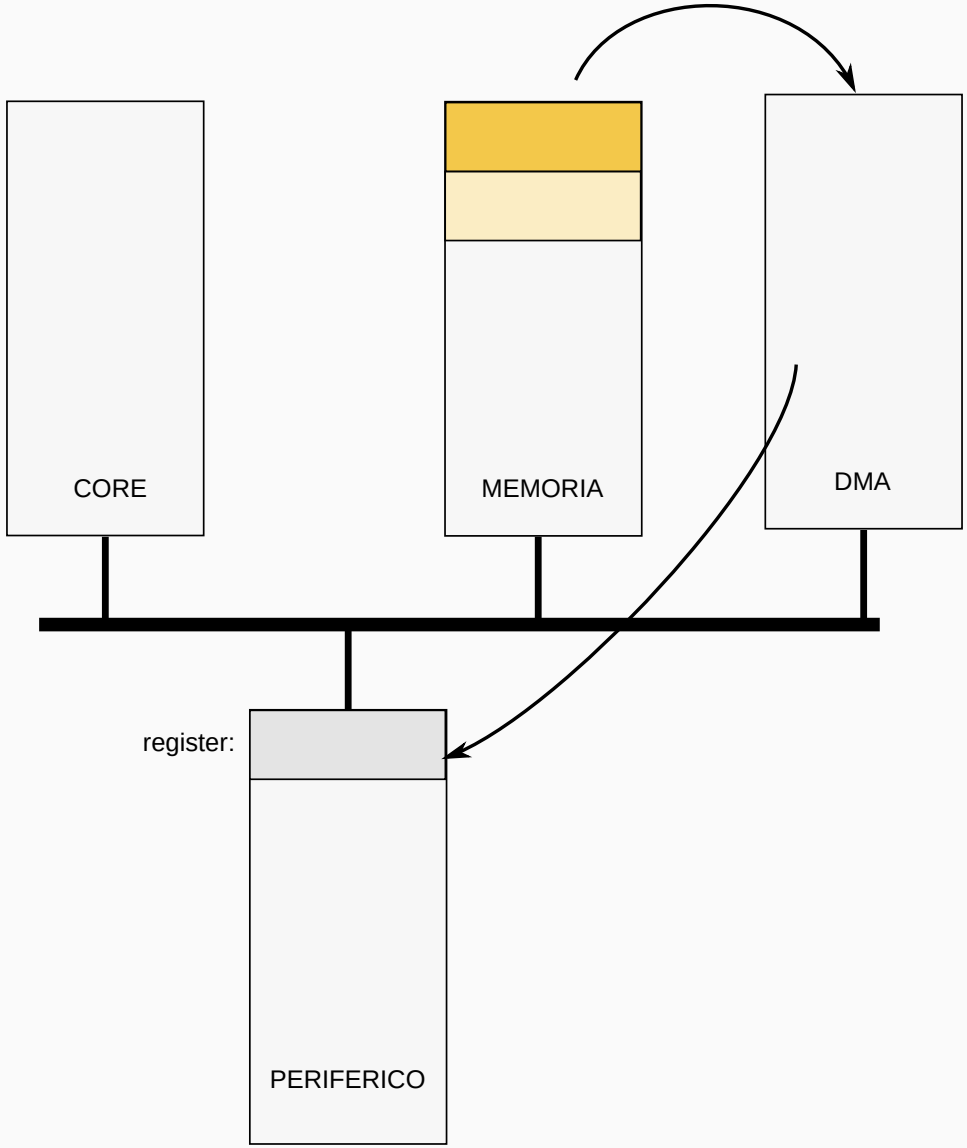


Figura 11: Preparando

DMA : Memória -> Periférico

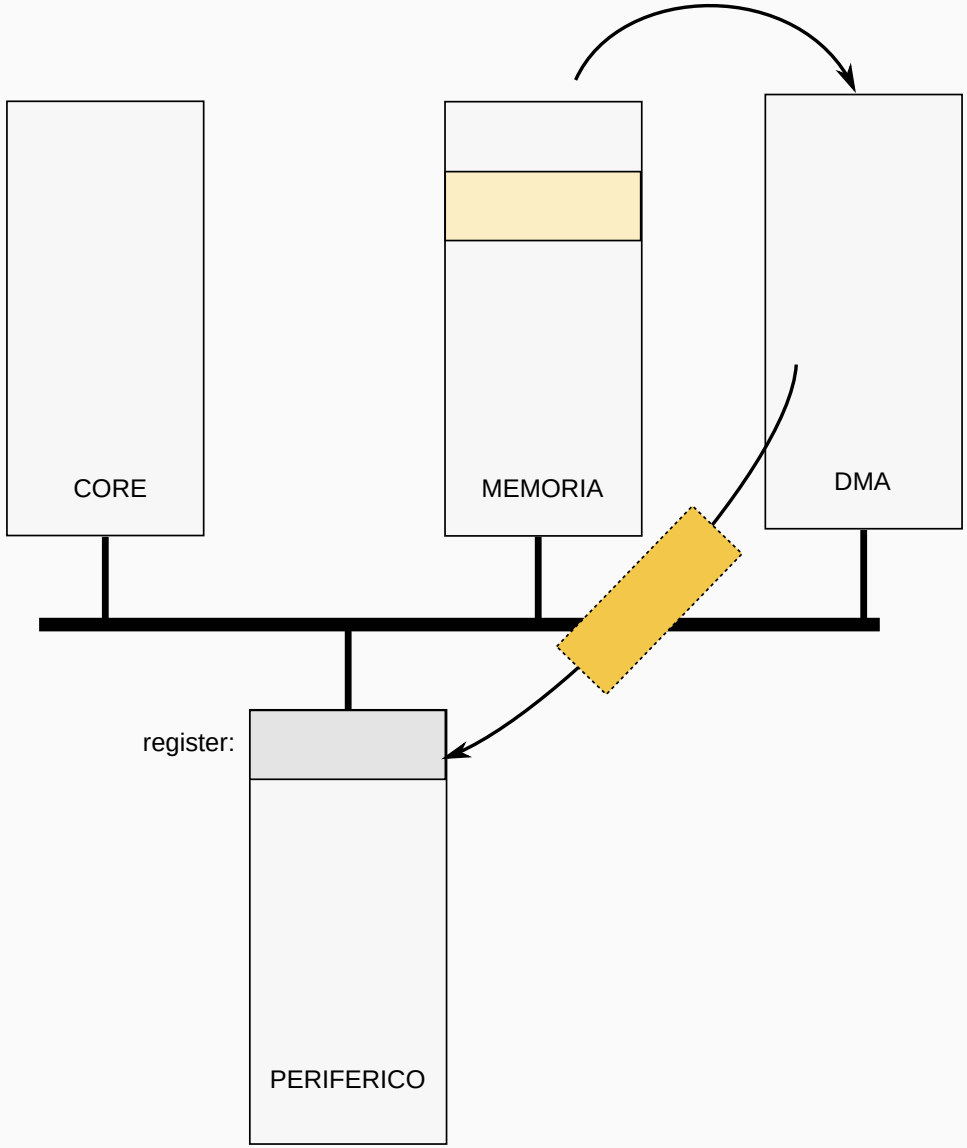


Figura 12: Carregando

DMA : Memória -> Periférico

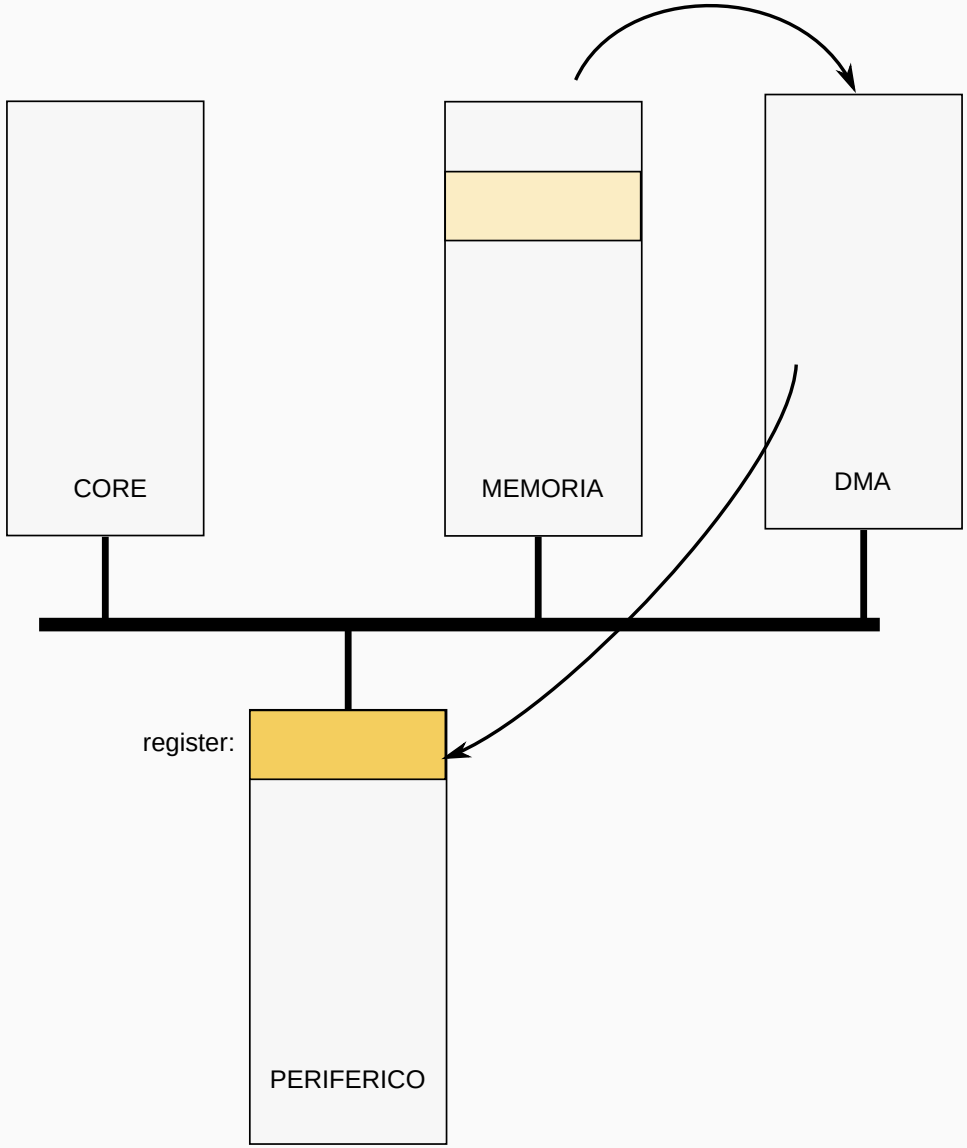


Figura 13: Transferido

DMA : Memória -> Periférico

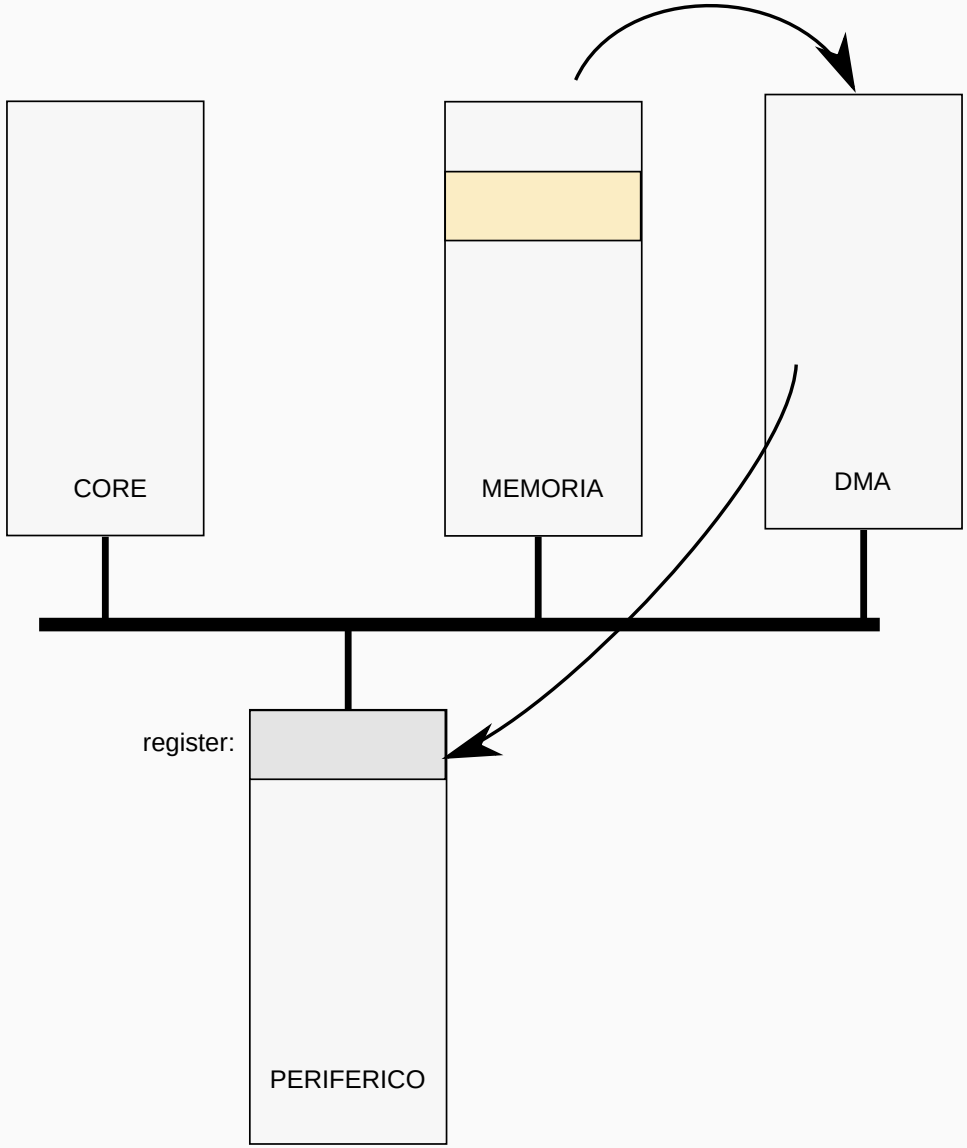


Figura 14: Preparando

DMA : Memória -> Periférico

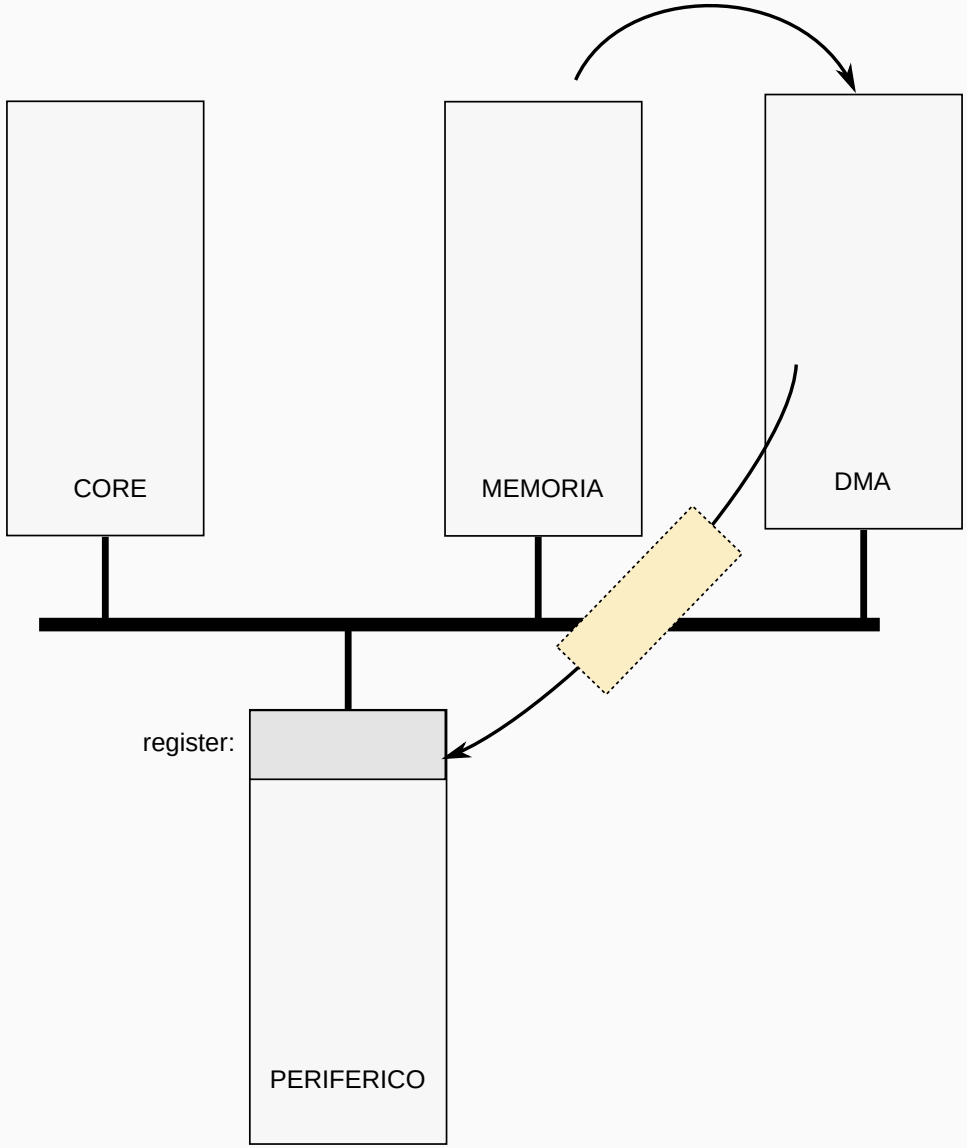


Figura 15: Carregando

DMA : Memória -> Periférico

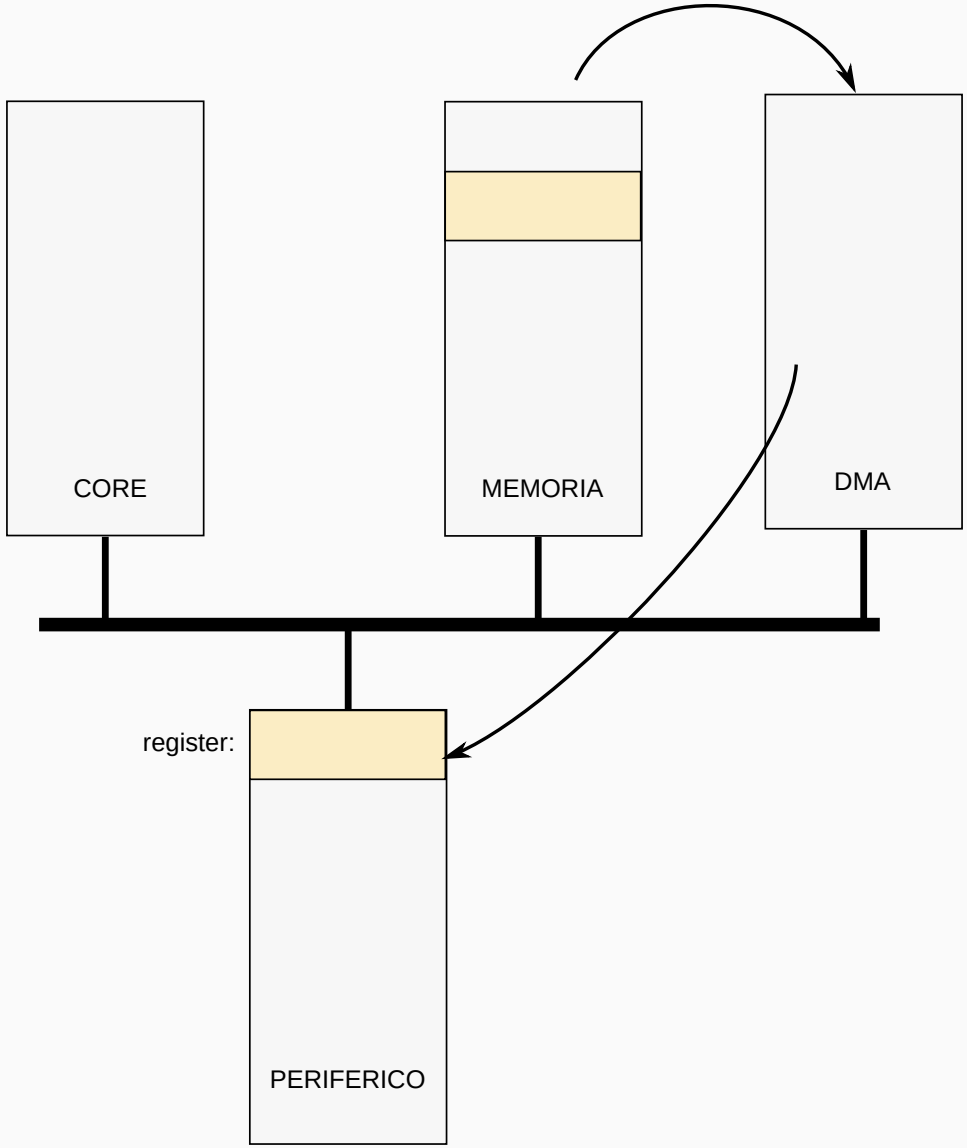
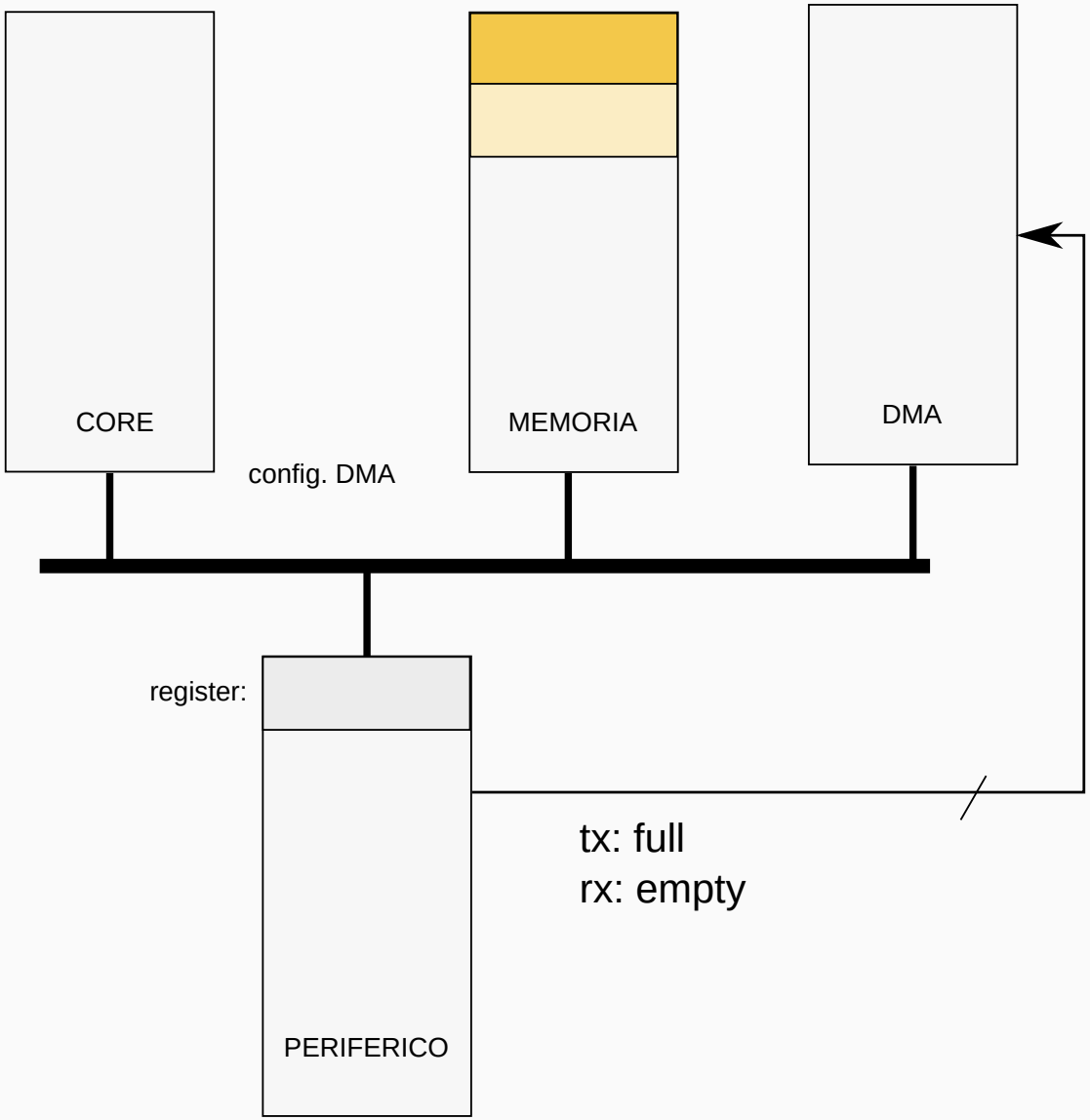


Figura 16: Transferindo

Mas como o DMA sabe que pode transferir um novo dado ?

Mas como o DMA sabe que pode transferir um novo dado ?

- Existem sinais de controle dos periféricos para o DMA



XDMA

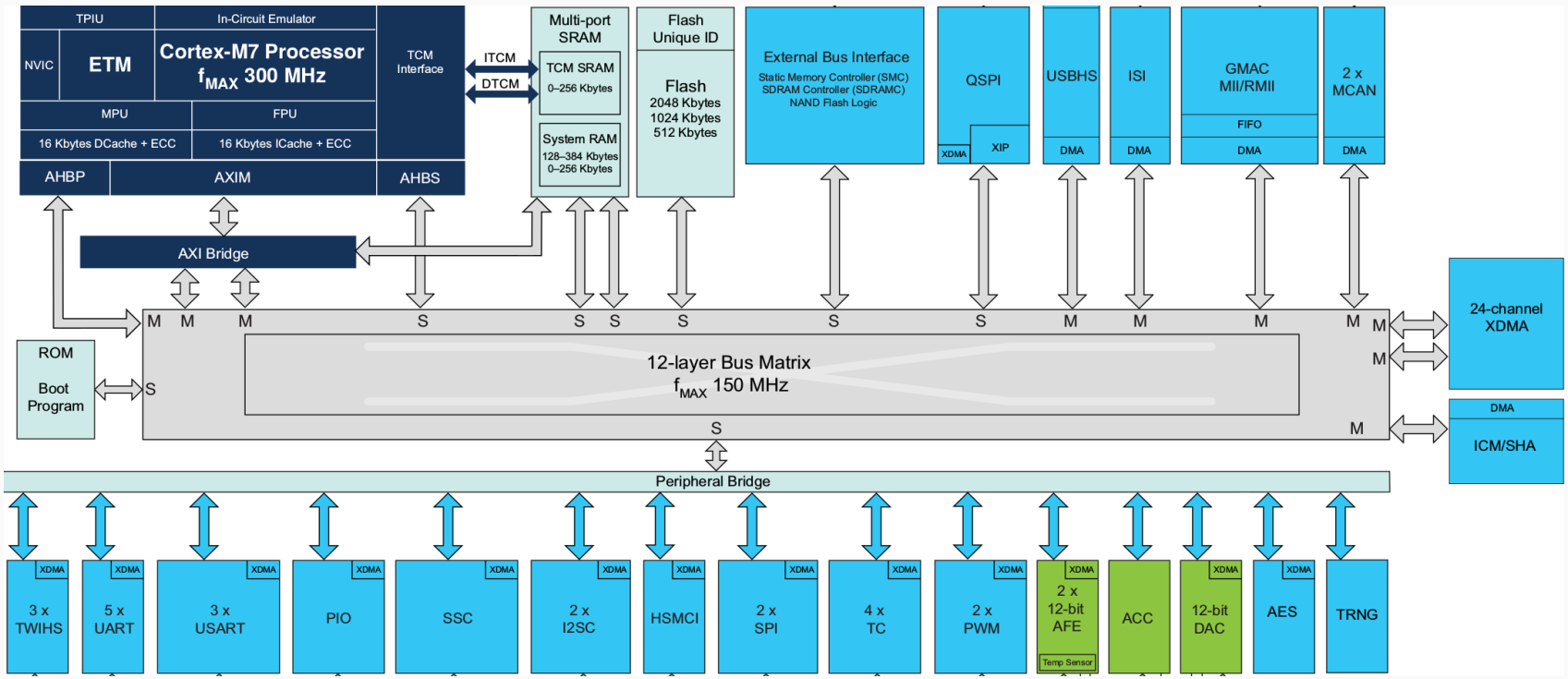


Figura 18: SAME70

SAME70

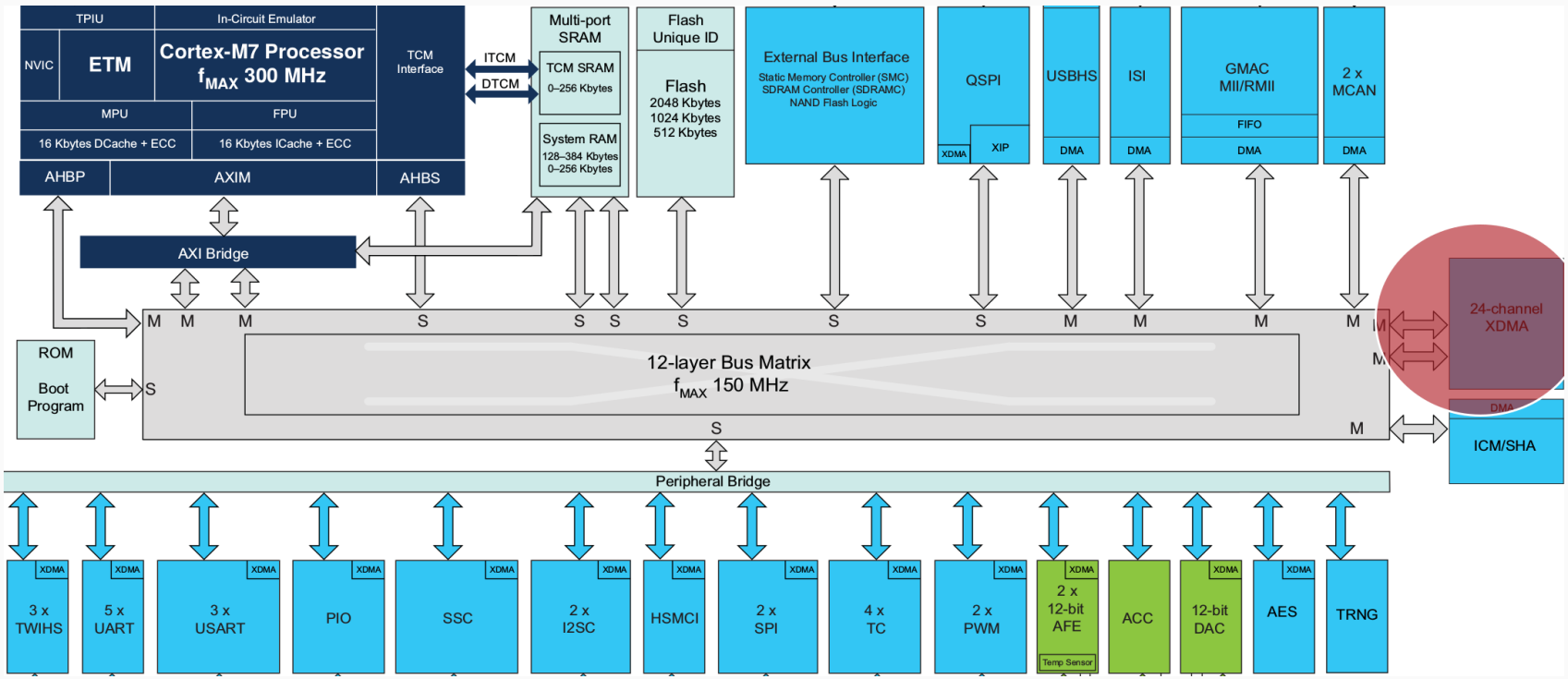


Figura 19: SAME70

XDMAC - DMA Controller (XDMAC)

The DMA Controller (XDMAC) is a AHB-protocol central direct memory access controller. It performs peripheral data transfer and memory move operations over one or two bus ports through the unidirectional communication channel. Each channel is fully programmable and provides both peripheral or memory-to-memory transfer. The channel features are configurable at implementation.

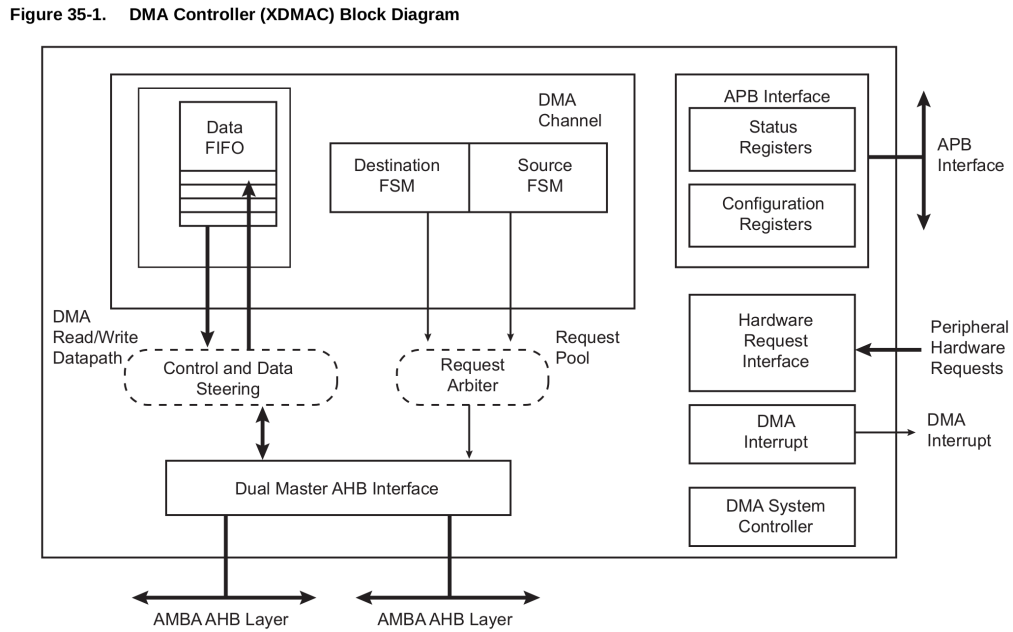


Figura 20: XDMA

XDMA - Operações

- O DMA do SAME70 possui 24 canais
- Cada ação de transferência é configurada via um canal (24 DMA Channels)
- Suporta endereçamento fixo ou auto incrementado
- 3.1kbytes embedded FIFO

XDMA - Canais

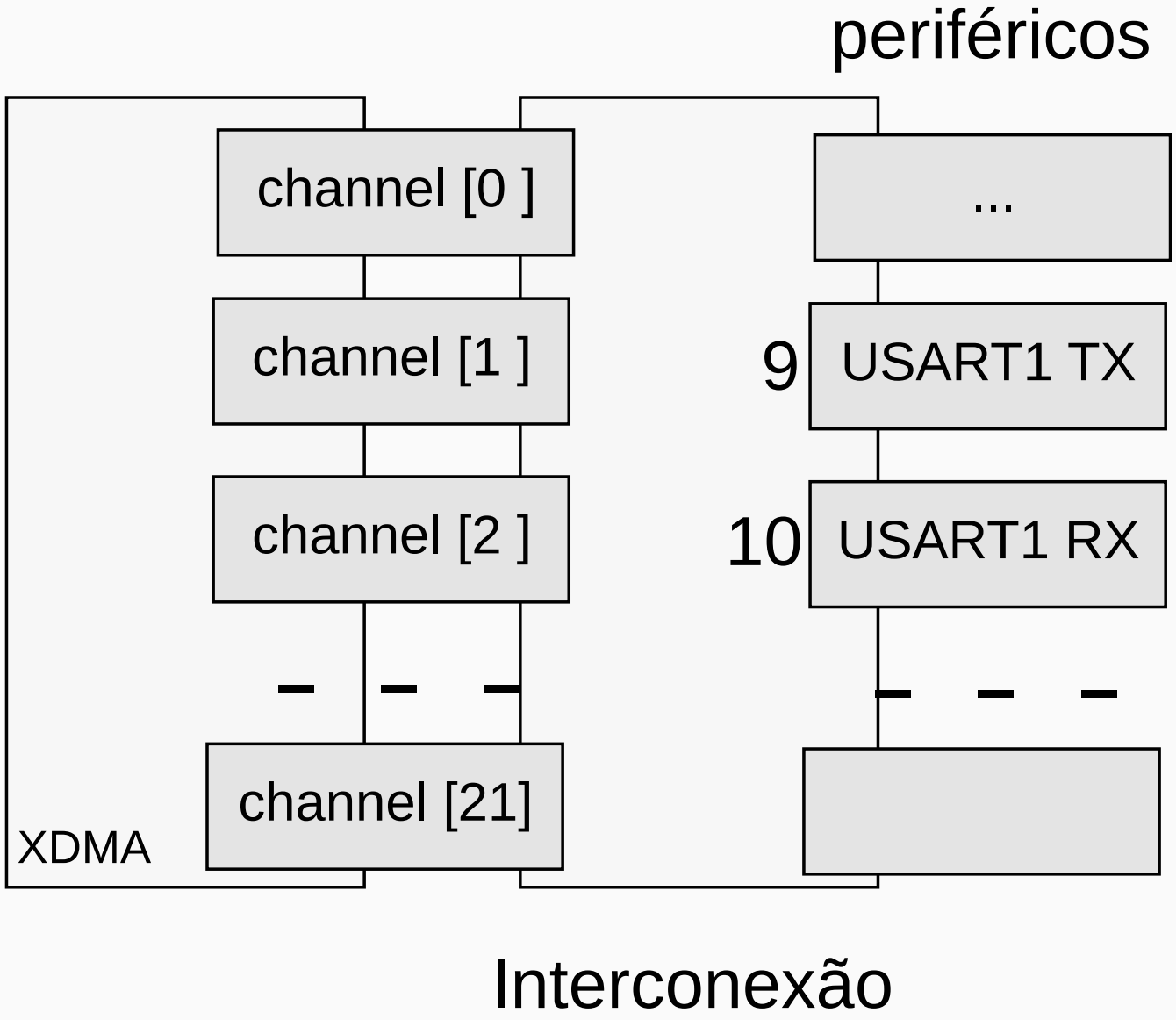


Figura 21: XDMA

XDMA - Canais

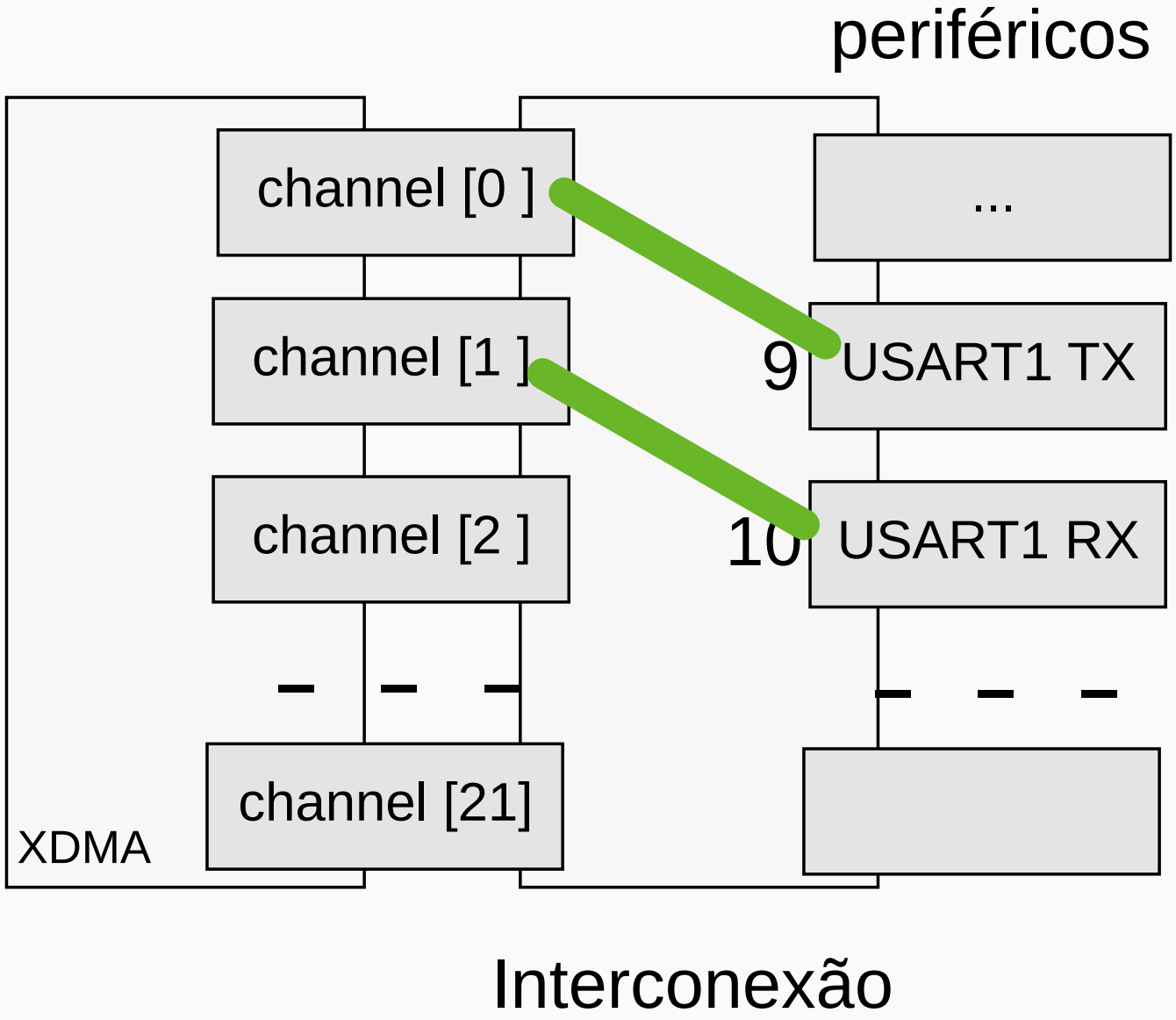


Figura 22: XDMA

XDMA - Matriz de Periféricos

Para utilizar o DMA com um periférico, deve-se conectar um canal ao canal do periférico em questão, seguindo a tabela 35.1 (pg. 480):

Table 35-1. Peripheral Hardware Requests

Peripheral Name	Transfer Type	HW Interface Number (XDMAC_CC.PERID)
HSMCI	Transmit/Receive	0
SPI0	Transmit	1
SPI0	Receive	2
SPI1	Transmit	3
SPI1	Receive	4
QSPI	Transmit	5
QSPI	Receive	6
USART0	Transmit	7
USART0	Receive	8
USART1	Transmit	9
USART1	Receive	10
USART2	Transmit	11
USART2	Receive	12

Figura 23: XDMA

USART

USART

– The USART supports the connection to the DMA Controller, which enables data transfers to the transmitter and from the receiver. The DMAC provides chained buffer management without any intervention of the processor. *

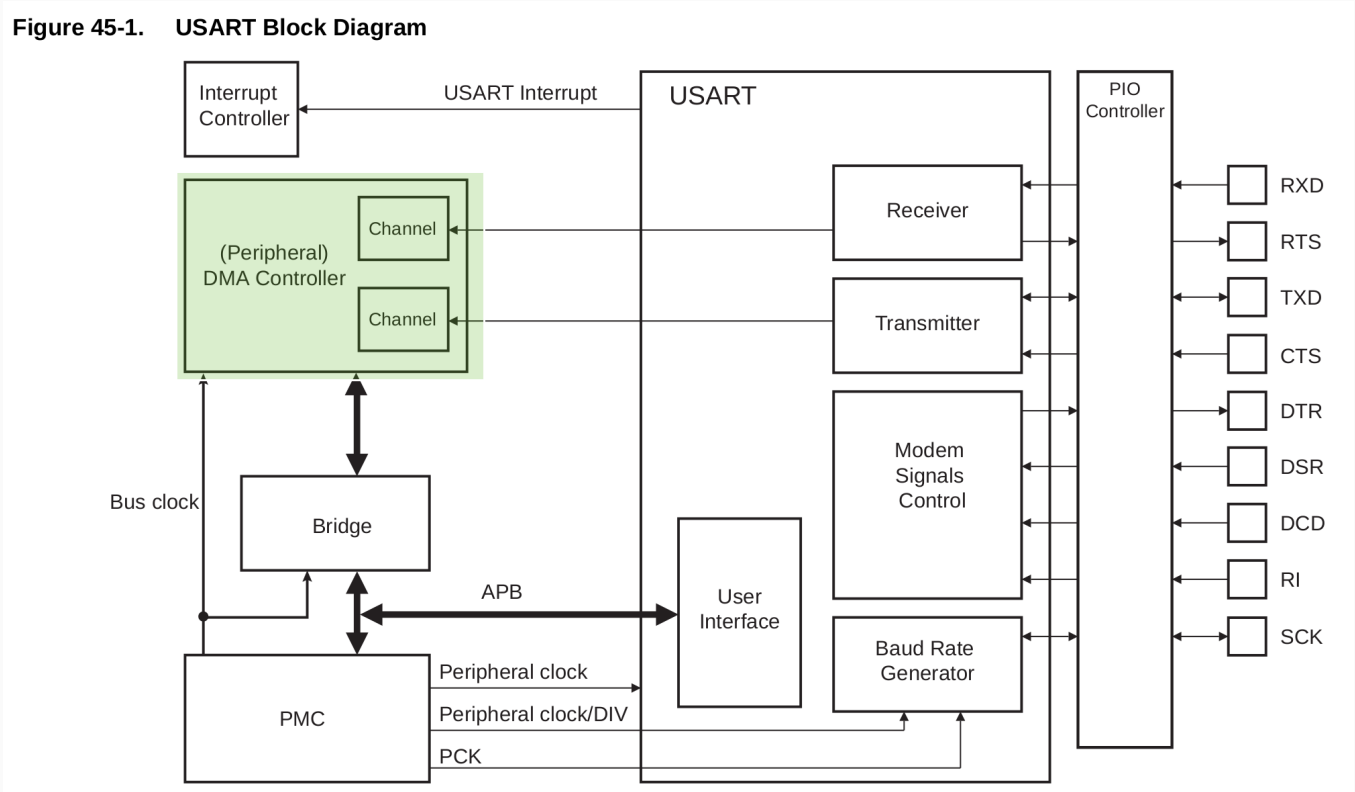


Figura 24: USART

Código

Código

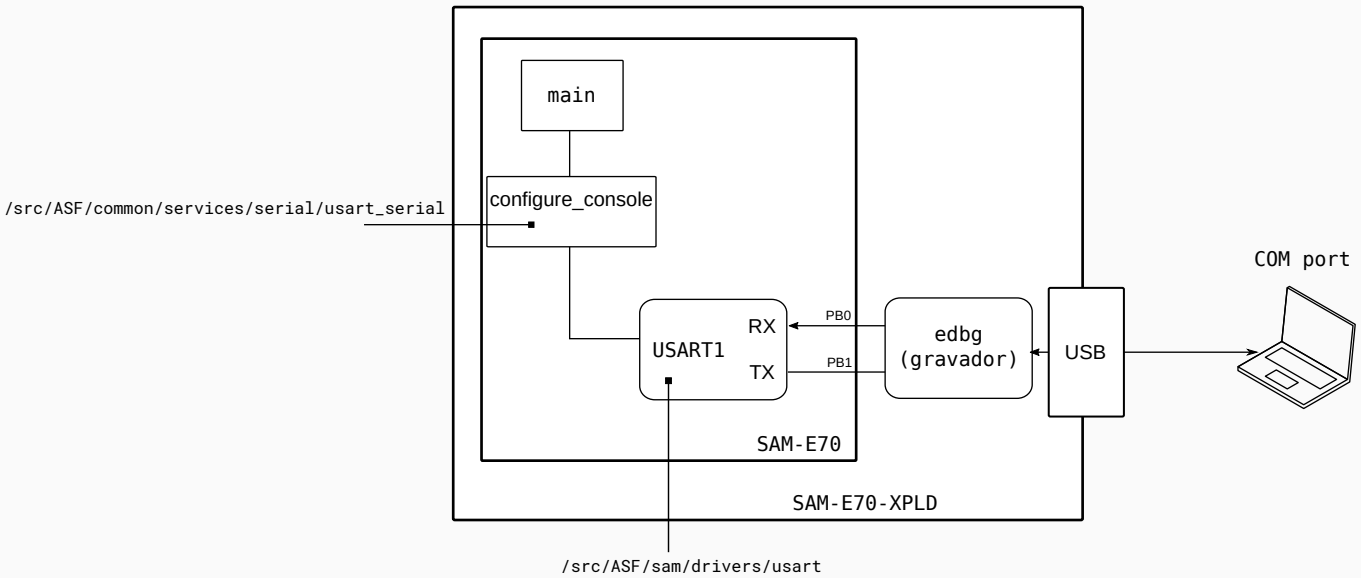


Figura 25: USART

Objetivos

- Automatizar a transferência de dados entre memória e USART via DMAC
- Modificar o código 13-UART e inserir o DMA (vide nota de aula)
- Deve-se, utilizando o DMA:
  - Quando apertado o botão da placa (SW0) inicializar a transferência de uma string para UART via DMA
    - tx_buffer
  - Uma vez transmitido a sting, inicializar a recepção de uma outra string com o número igual de caracteres
    - rx_buffer
    - com a string recebida, conferir se o dado transmitido é igual ao recebido e informar no terminal.