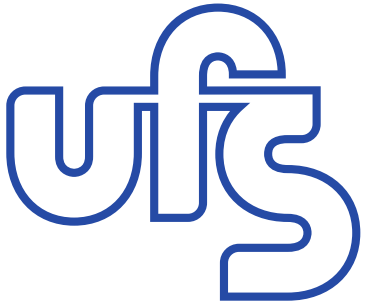
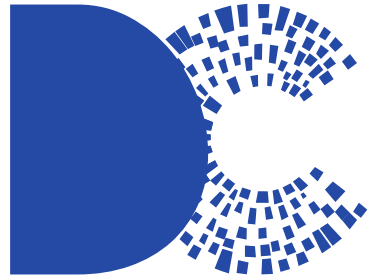




Entrada e saída

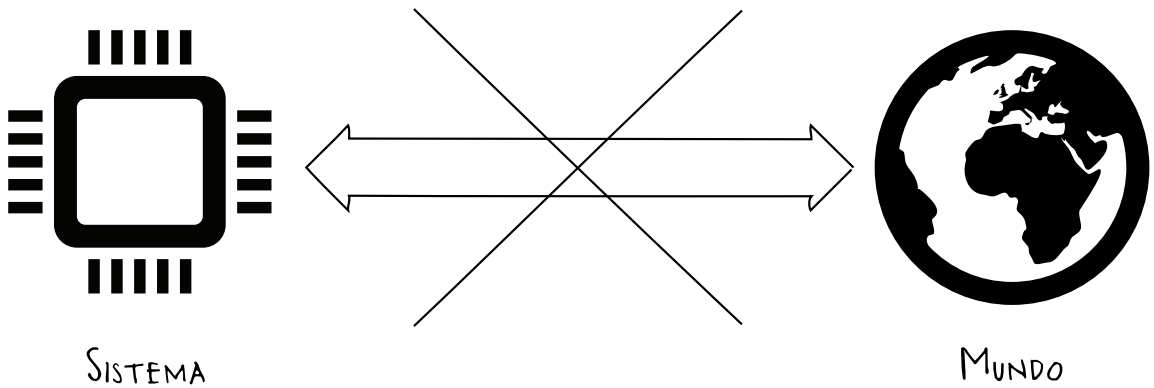
Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

Introdução

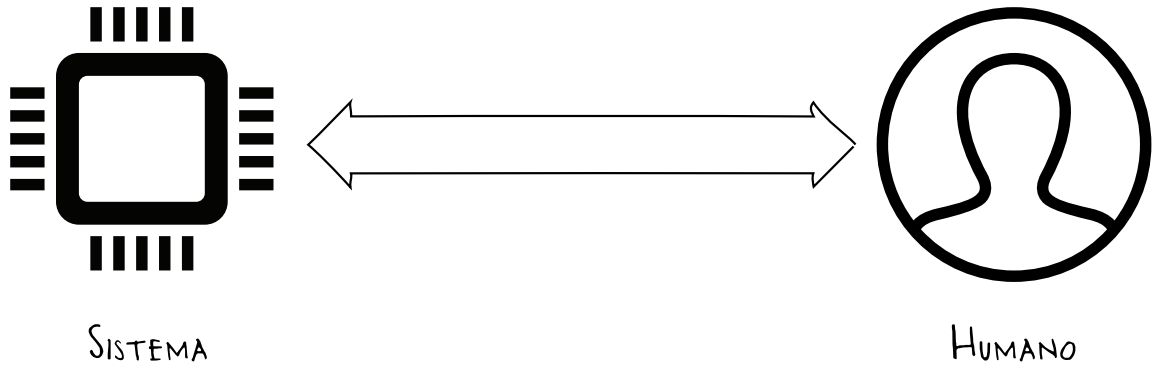
- Qual é a importância da entrada e saída (E/S)?
  - Comunicação sistema $\times$ mundo



Como seria utilizar um sistema sem E/S?

Introdução

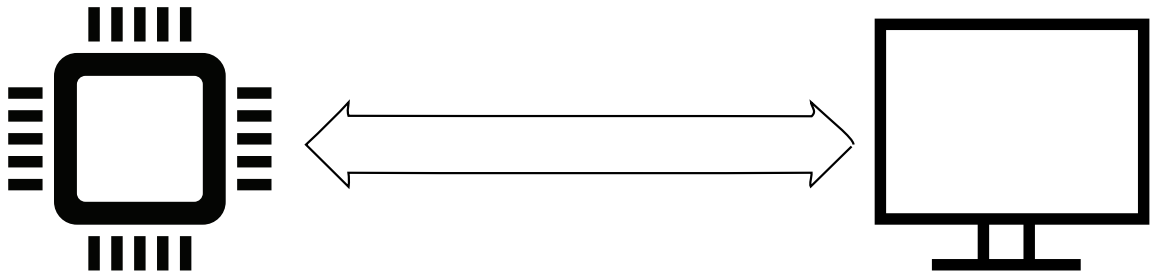
- Qual é a importância da entrada e saída (E/S)?
  - Comunicação sistema $\times$ mundo



Interação com o usuário (humano)

Introdução

- Qual é a importância da entrada e saída (E/S)?
  - Comunicação sistema $\times$ mundo



Interface com outros sistemas (computacional)

Introdução

- Classificação dos dispositivos de E/S

Dispositivo	Comportamento	Interface	Taxa de dados (Mbit/s)
Teclado	Entrada	Humano	0,0001
Tela	Saída	Humano	800 - 8000
Placa de rede	Entrada/Saída	Sistema	100 - 1000
Disco rígido	Entrada/Saída	Sistema	30 - 3500

Introdução

- Classificação dos dispositivos de E/S

Dispositivo	Comportamento	Interface	Taxa de dados (Mbit/s)
Teclado	Entrada	Humano	0,0001
Tela	Saída	Humano	800 - 8000
Placa de rede	Entrada/Saída	Sistema	100 - 1000
Disco rígido	Entrada/Saída	Sistema	30 - 3500

Para armazenamento e transmissão de dados é adotada a notação em base decimal (SI)

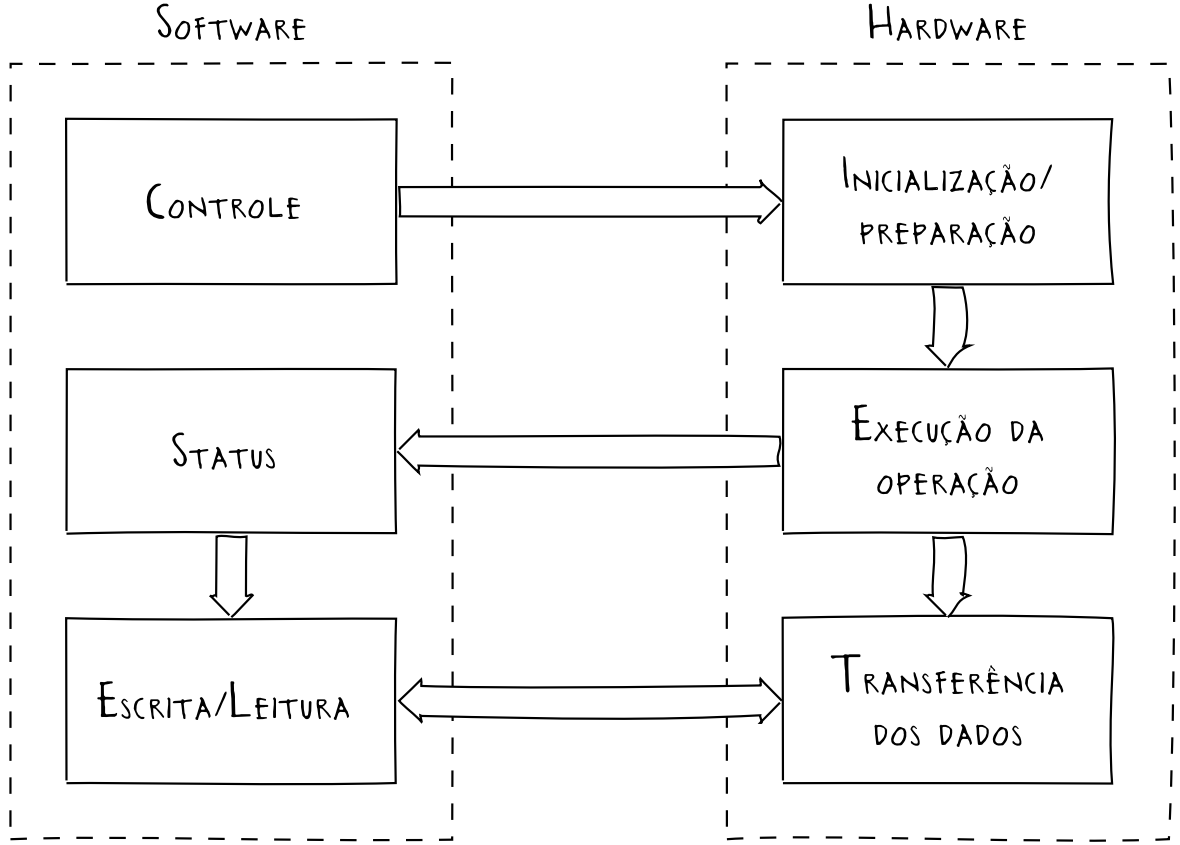
1 MB = 1.000.000 (10^6) bytes < 1 MiB = 1.048.576 (2^{20}) bytes

Introdução

- Tipos de operações de entrada e saída
  - Programada
  - Por interrupção
  - Acesso direto à memória (DMA)

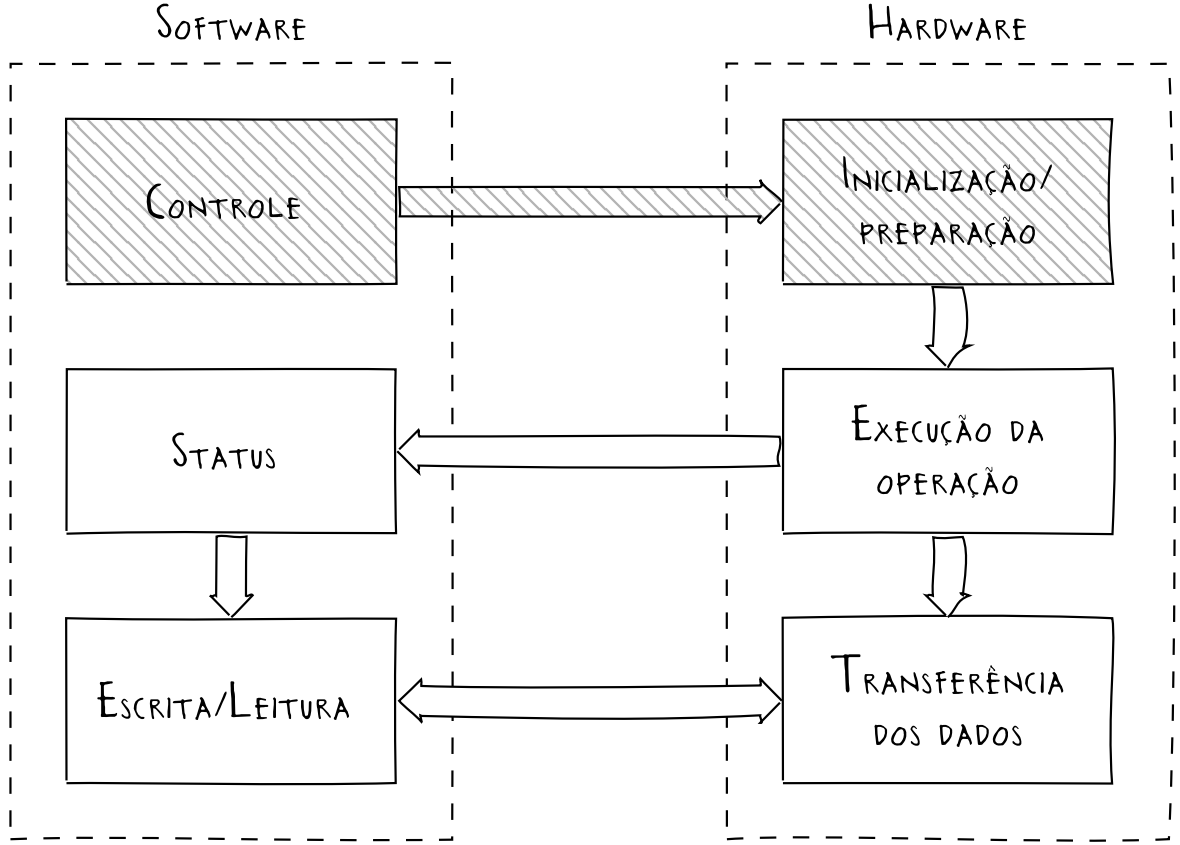
E/S programada

- As operações de E/S são realizadas pelo processador utilizando um gerenciador de dispositivo (*driver*)



E/S programada

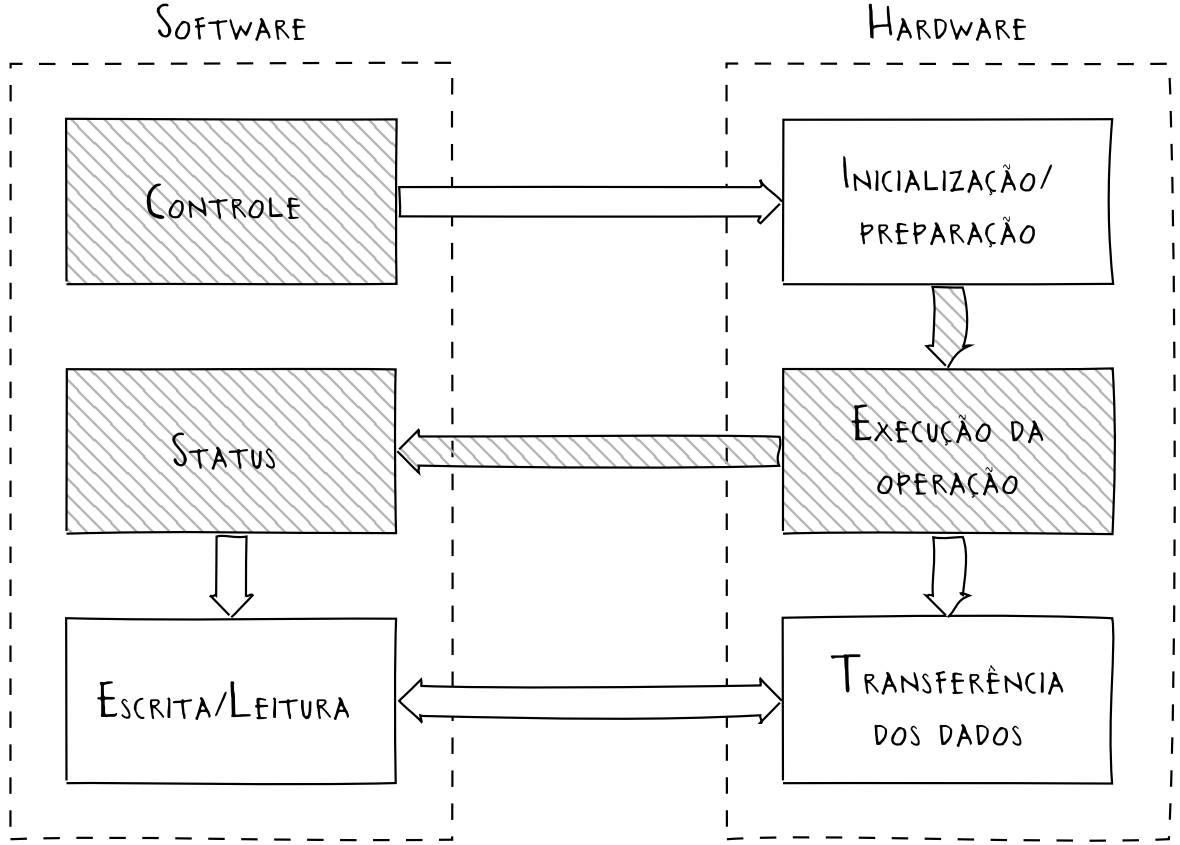
- As operações de E/S são realizadas pelo processador utilizando um gerenciador de dispositivo (*driver*)



Inicialização ou preparação do dispositivo para operação de E/S

E/S programada

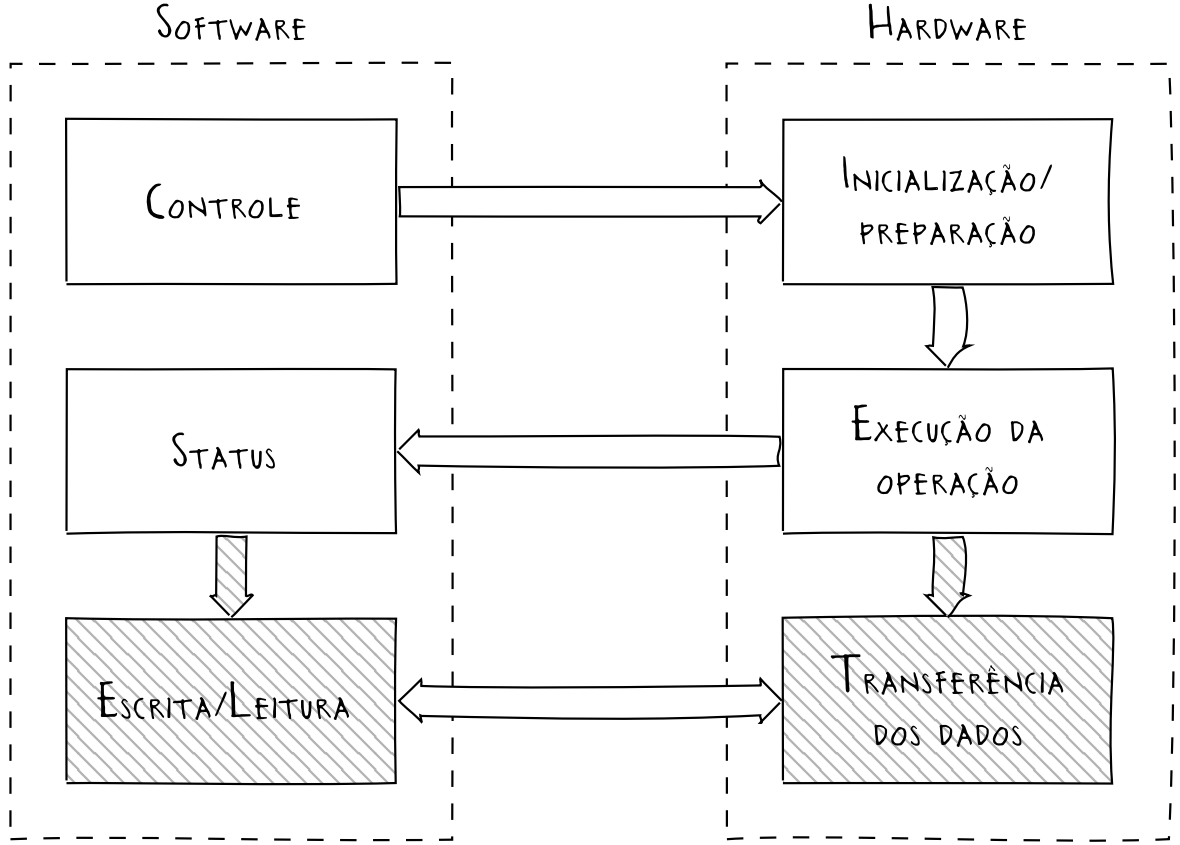
- As operações de E/S são realizadas pelo processador utilizando um gerenciador de dispositivo (*driver*)



O processador é mais rápido (geralmente)
do que o dispositivo (o controle precisa esperar por pendências)

E/S programada

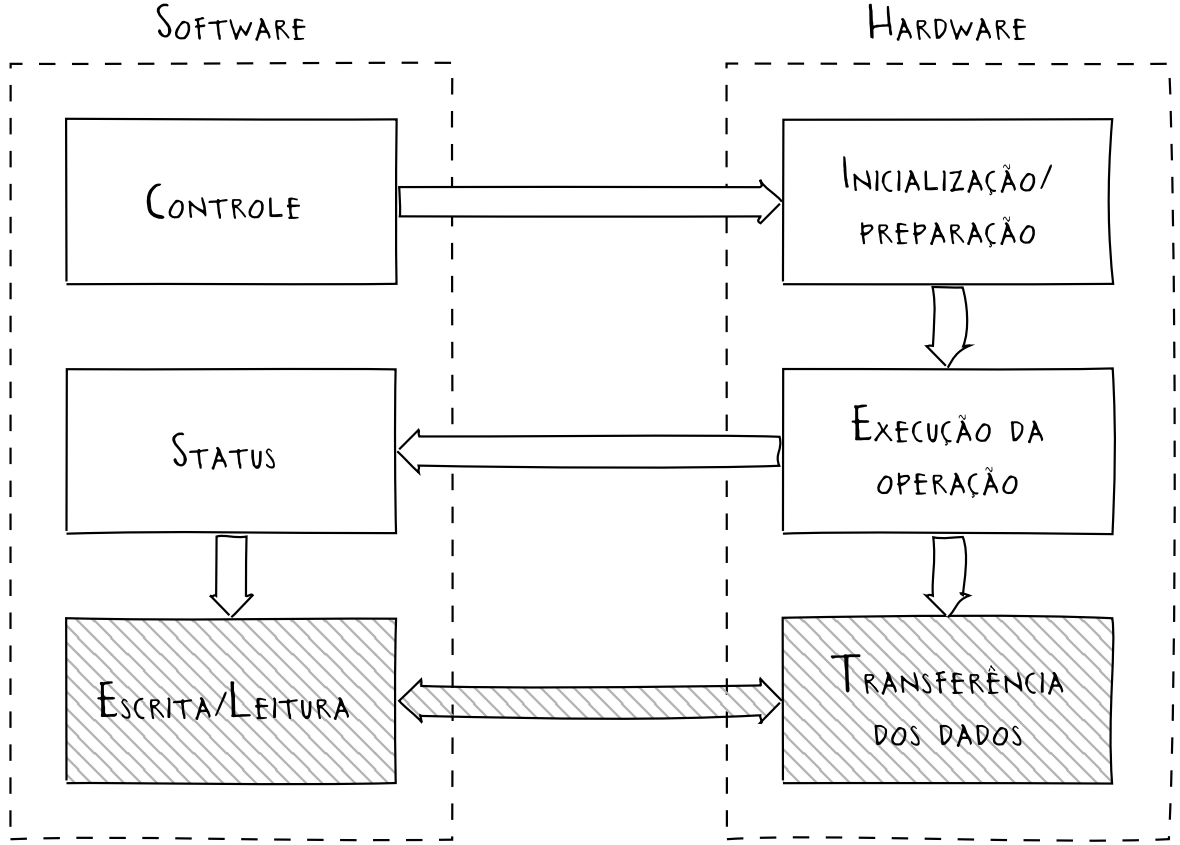
- As operações de E/S são realizadas pelo processador utilizando um gerenciador de dispositivo (*driver*)



Tanto o software como o hardware
se preparam para realizar a transferência dos dados

E/S programada

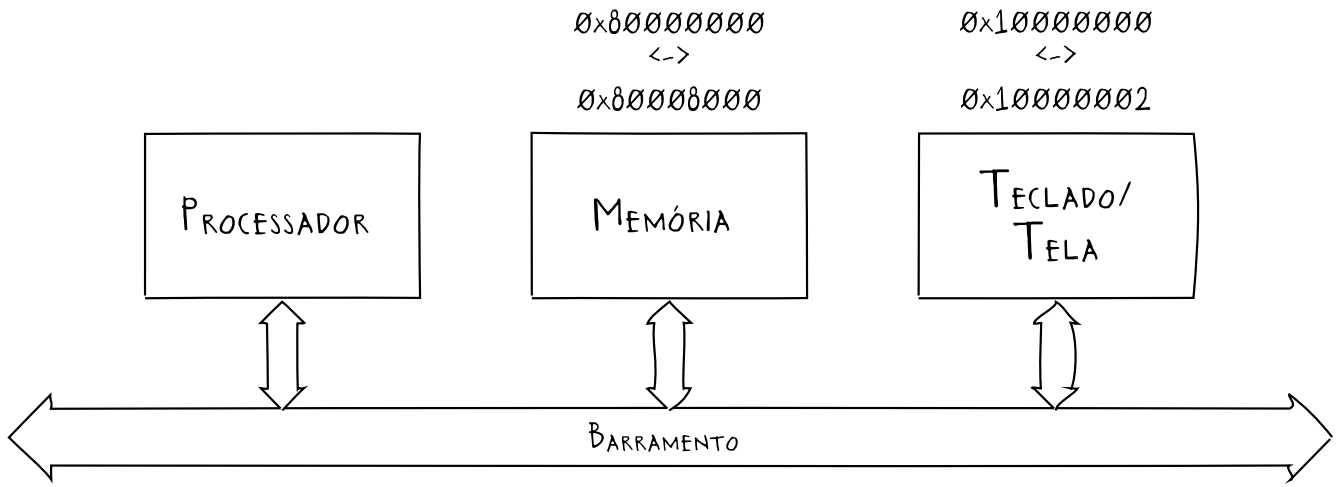
- As operações de E/S são realizadas pelo processador utilizando um gerenciador de dispositivo (*driver*)



Os dados são transferidos entre a memória de dados do software e os registradores do dispositivo

E/S programada

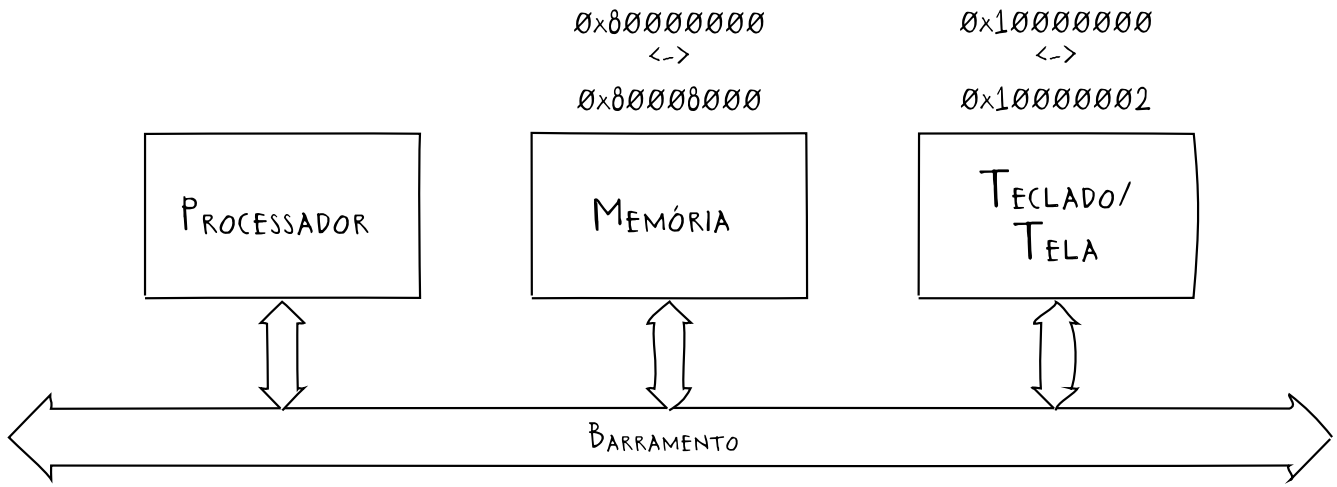
- E/S por mapeamento em memória



As operações de E/S são realizadas
através de instruções de acesso à memória

E/S programada

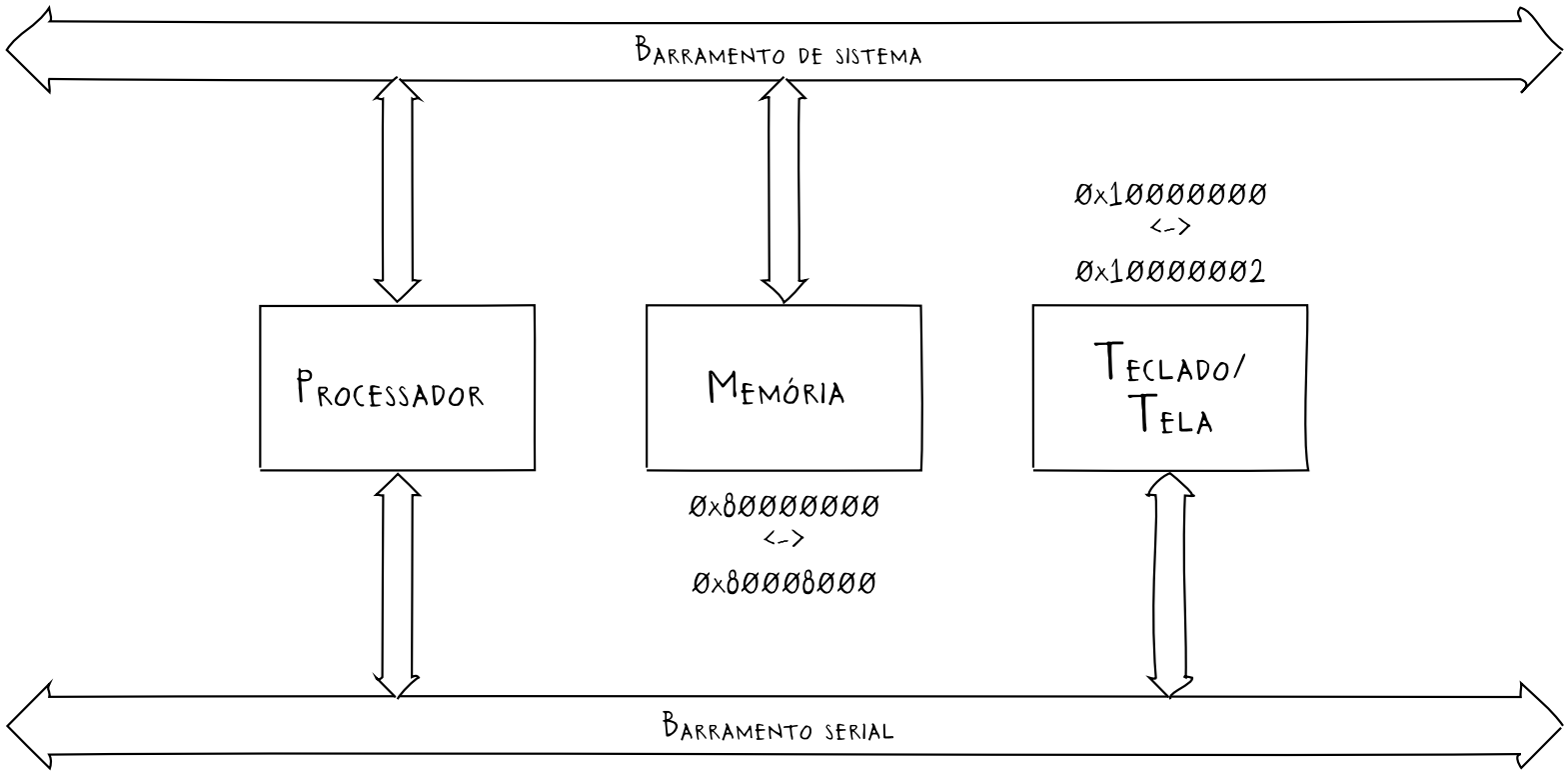
- E/S por mapeamento em memória



É preciso alocar ou mapear endereços
no espaço de memória para cada dispositivo

E/S programada

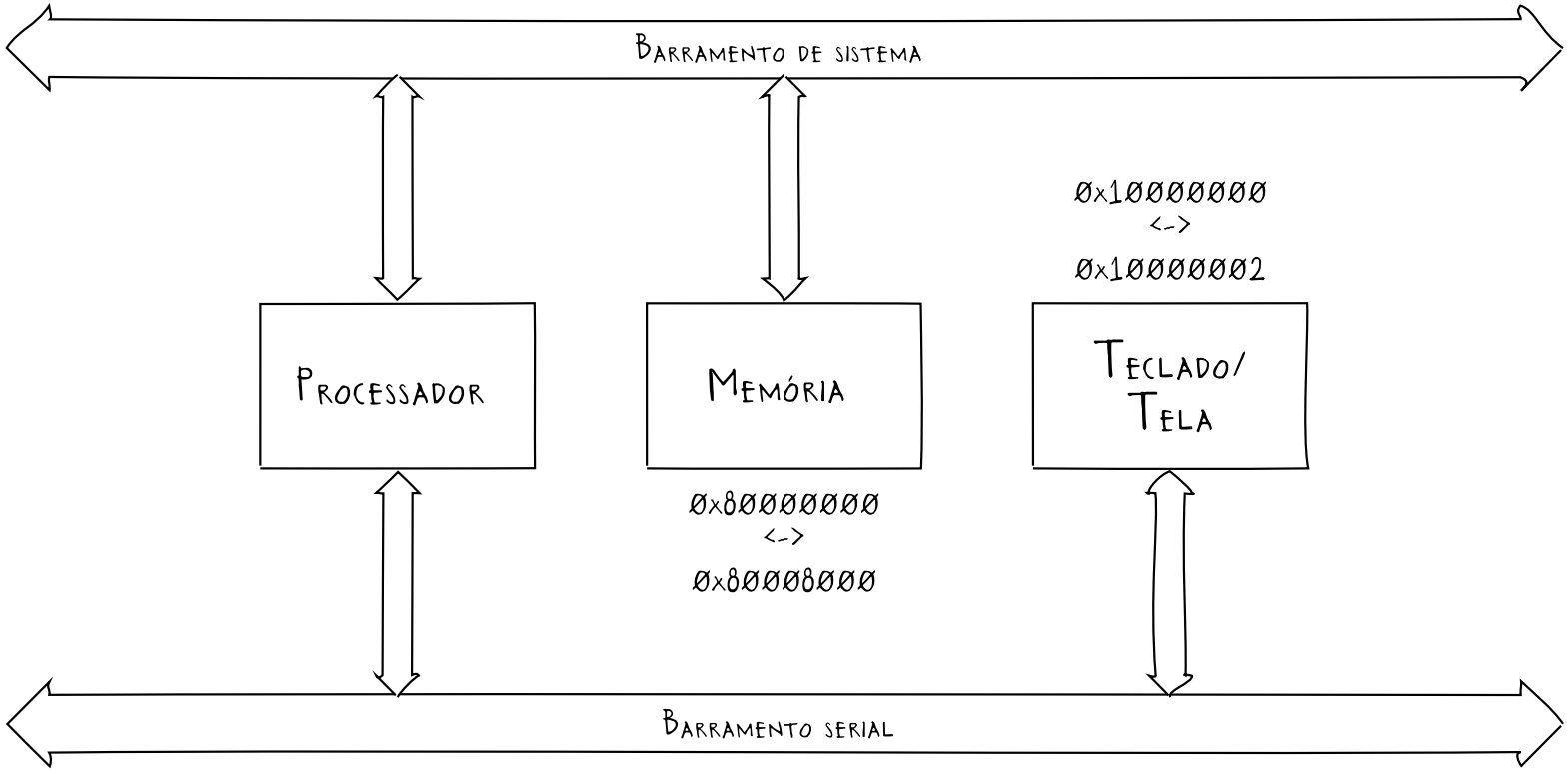
- E/S por instruções especializadas



As operações de E/S são realizadas através de instruções especializadas

E/S programada

- E/S por instruções especializadas



Com barramentos separados na plataforma,
é possível o endereçamento dedicado para E/S

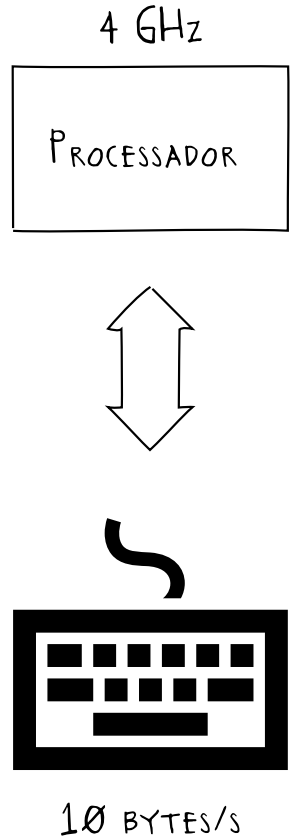
E/S programada

- Realizando E/S programa do teclado mapeado em memória no endereço 0x10000000

```
54  # Função principal
55  .global main
56  main:
…       ...
60      # Carregando endereço do teclado
61      li a0, 0x10000000
62      # Leitura da tecla
63      espera:
64          lb t0, 0(a0)
65          beq t0, zero, espera
…       ...
77      # Ajustando valor de retorno para zero (sucesso)
78      mv a0, zero
79      # Retornando da chamada
80      ret
```

E/S programada

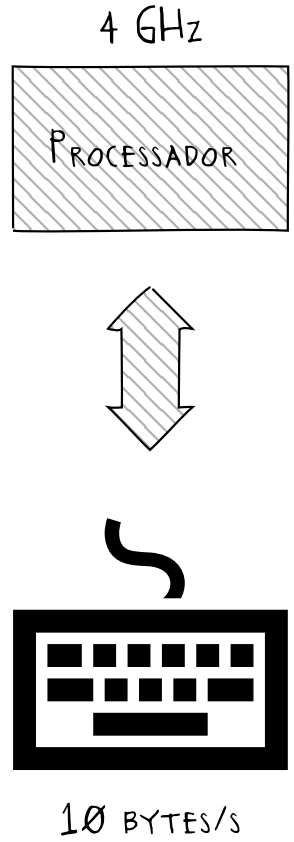
- Leitura do teclado com E/S programada



O teclado é capaz de enviar 1 byte
a cada 0,1 segundo (velocidade máxima)

E/S programada

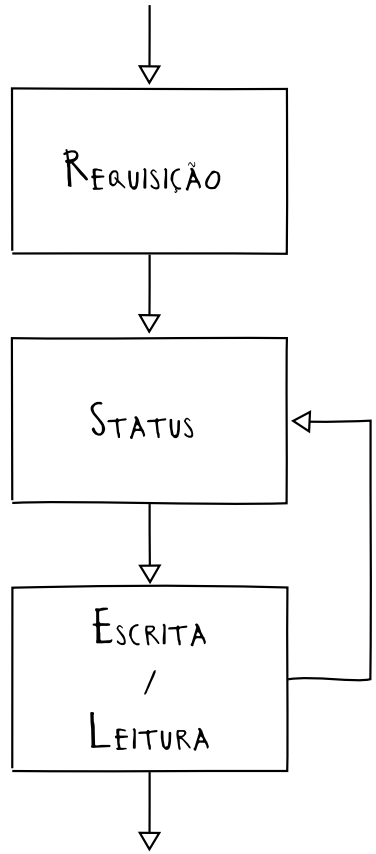
- Leitura do teclado com E/S programada



Durante esta espera de 0,1 segundo,
o processador poderia ter executado
400 milhões de instruções

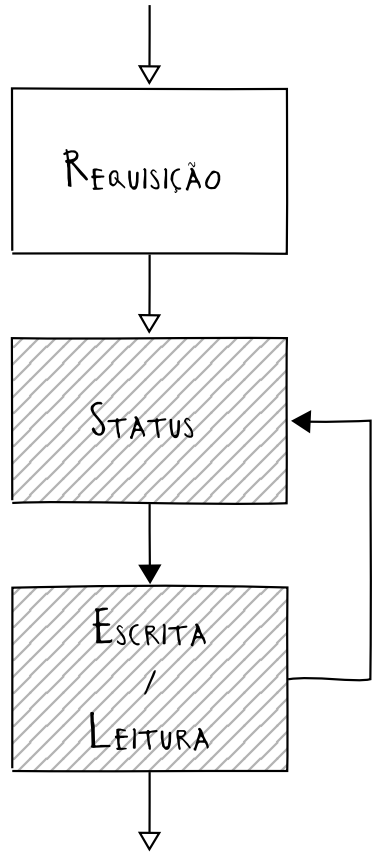
E/S por interrupção

- A E/S programada pode impor ao processador uma longa espera durante a transferência de dados



E/S por interrupção

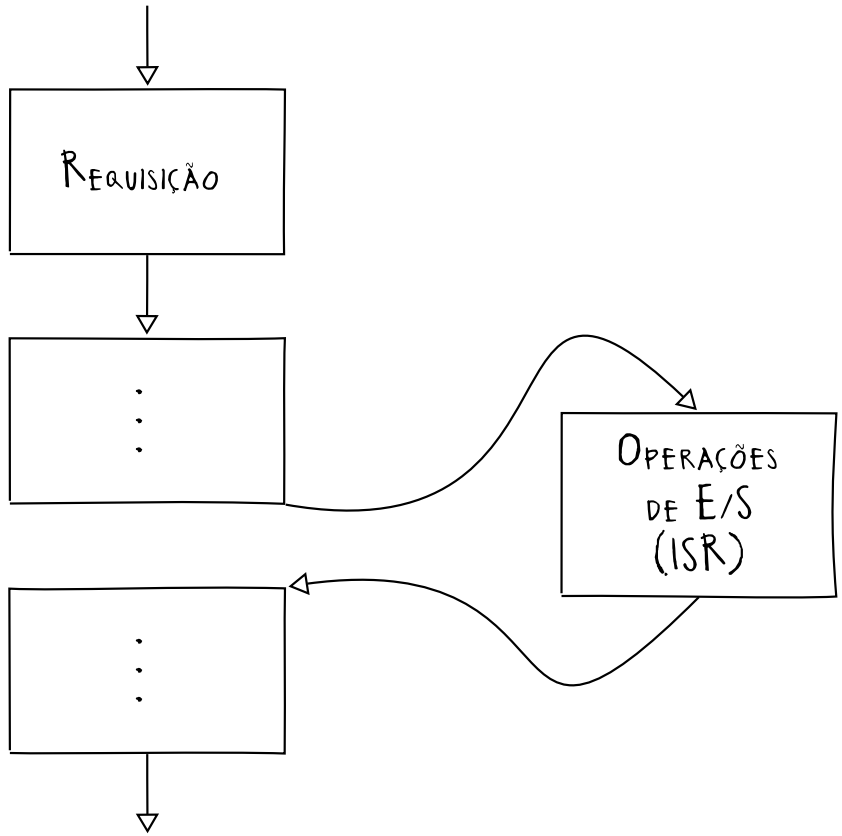
- A E/S programada pode impor ao processador uma longa espera durante a transferência de dados



Acesso bloqueante por polling

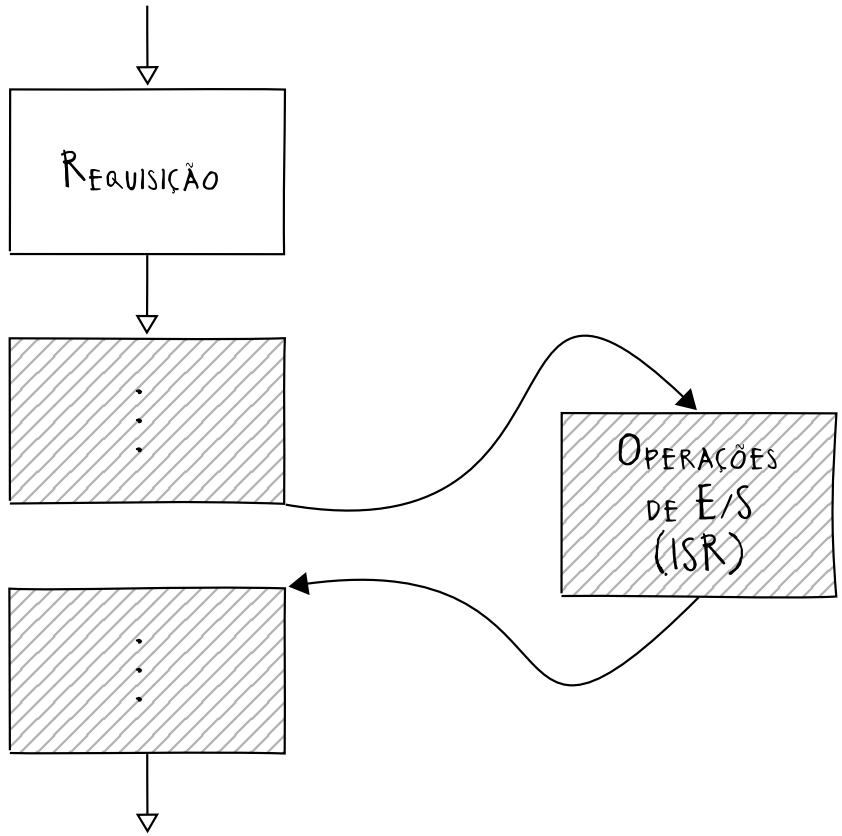
E/S por interrupção

- O processador faz a requisição de operação de E/S para o dispositivo, sem bloquear o fluxo de execução do software



E/S por interrupção

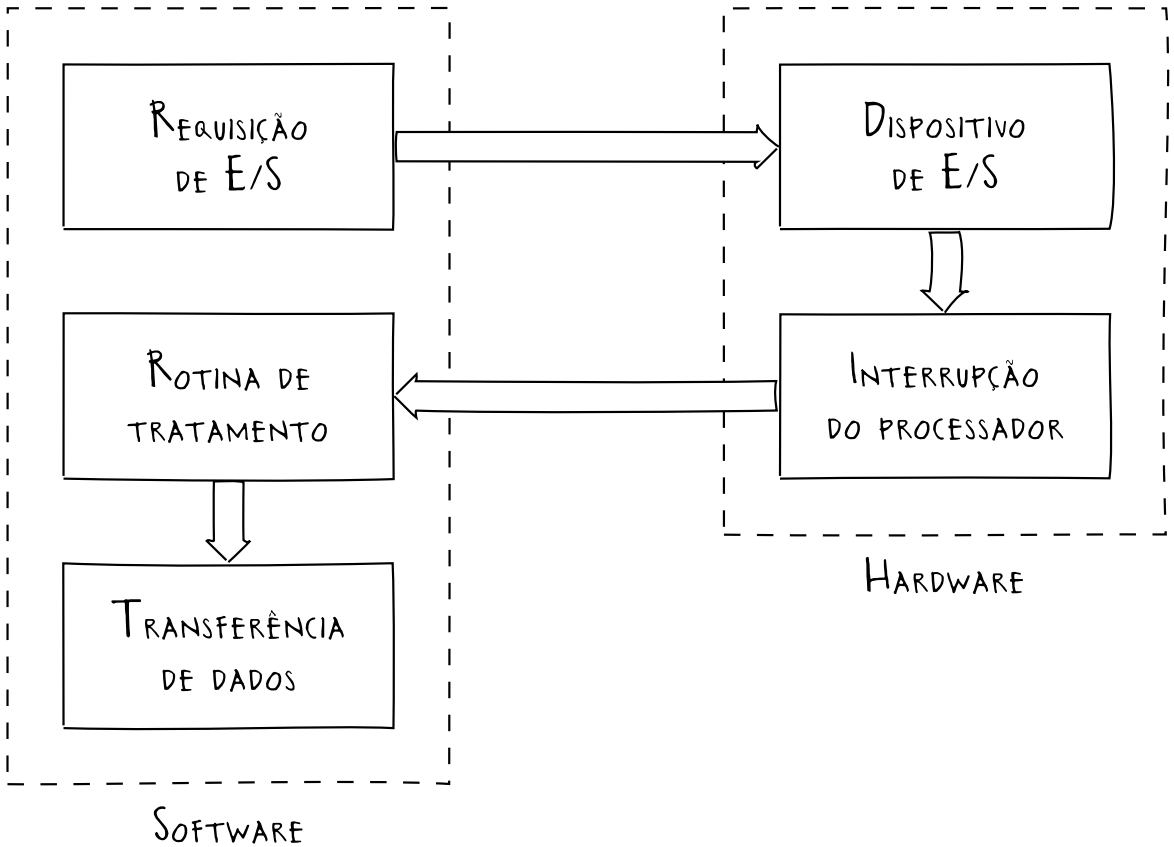
- O processador faz a requisição de operação de E/S para o dispositivo, sem bloquear o fluxo de execução do software



Quando a requisição está pronta,
é gerado um evento de interrupção (trap)
para executar as operações de E/S

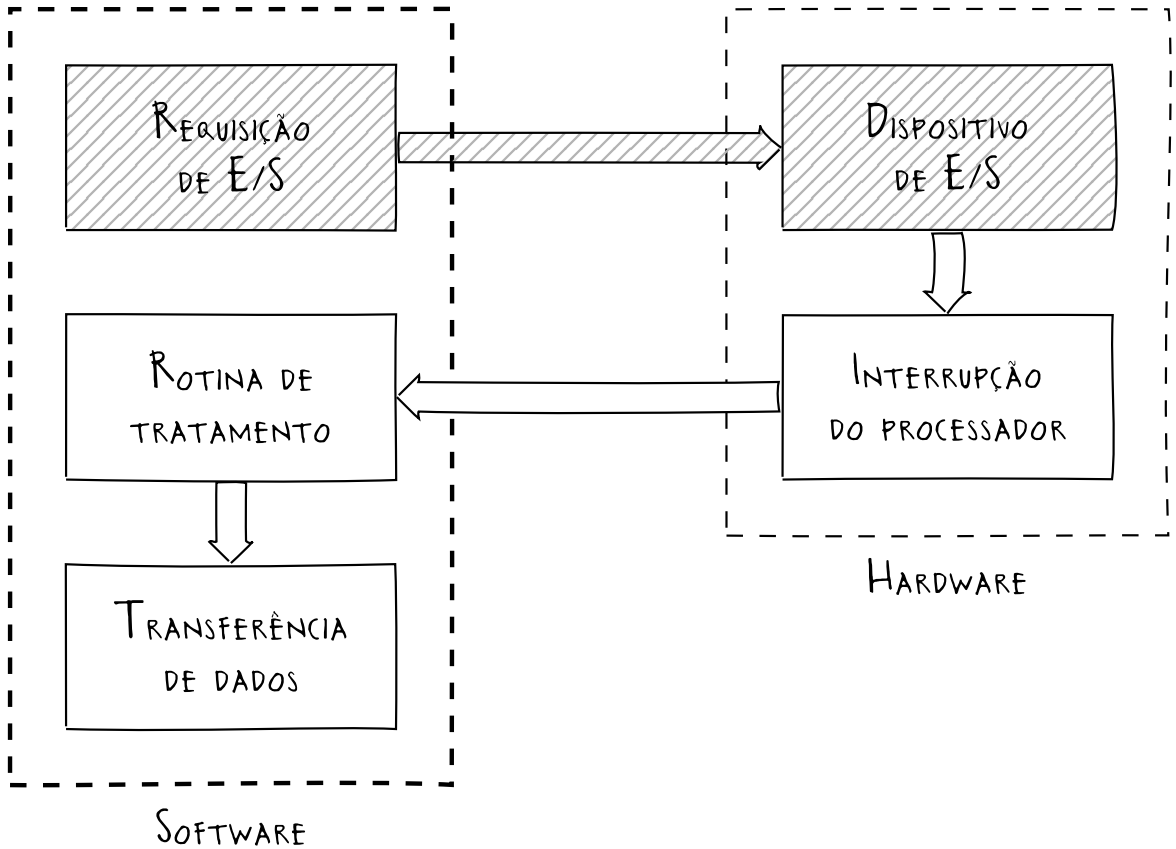
E/S por interrupção

- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



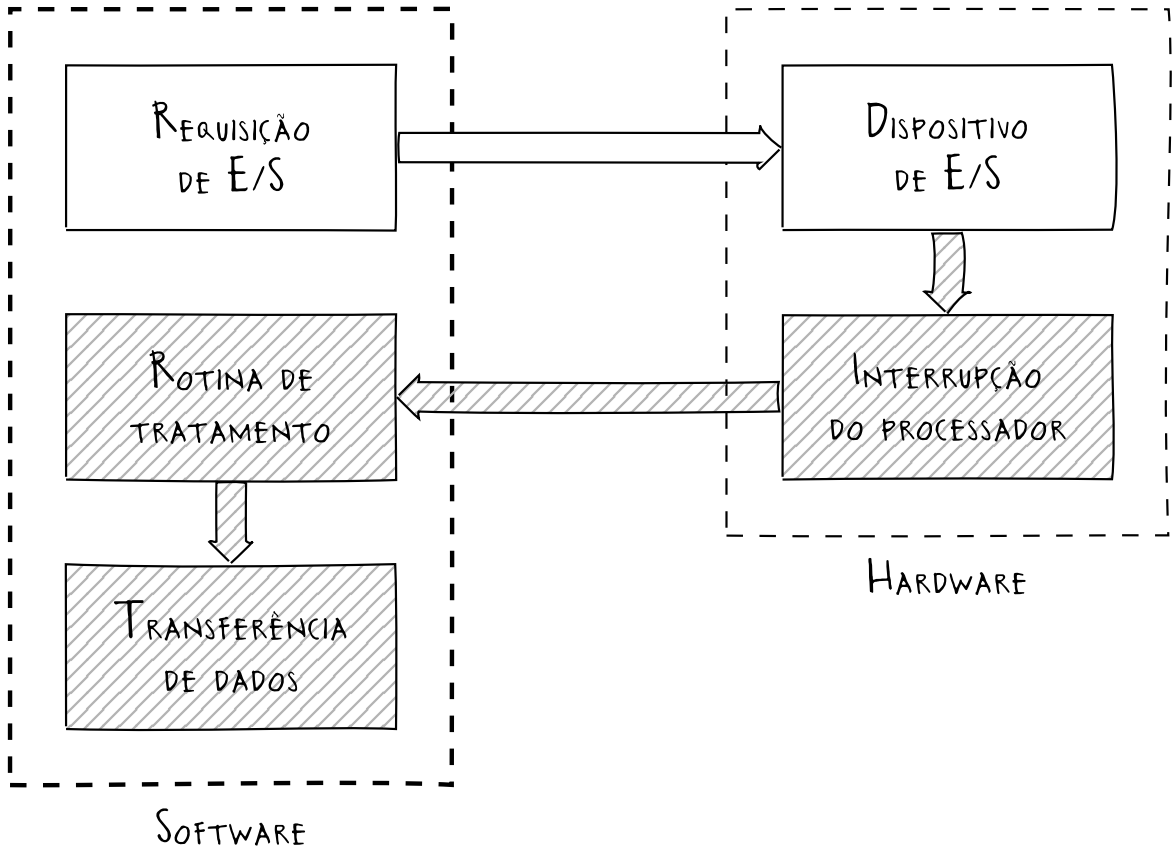
E/S por interrupção

- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



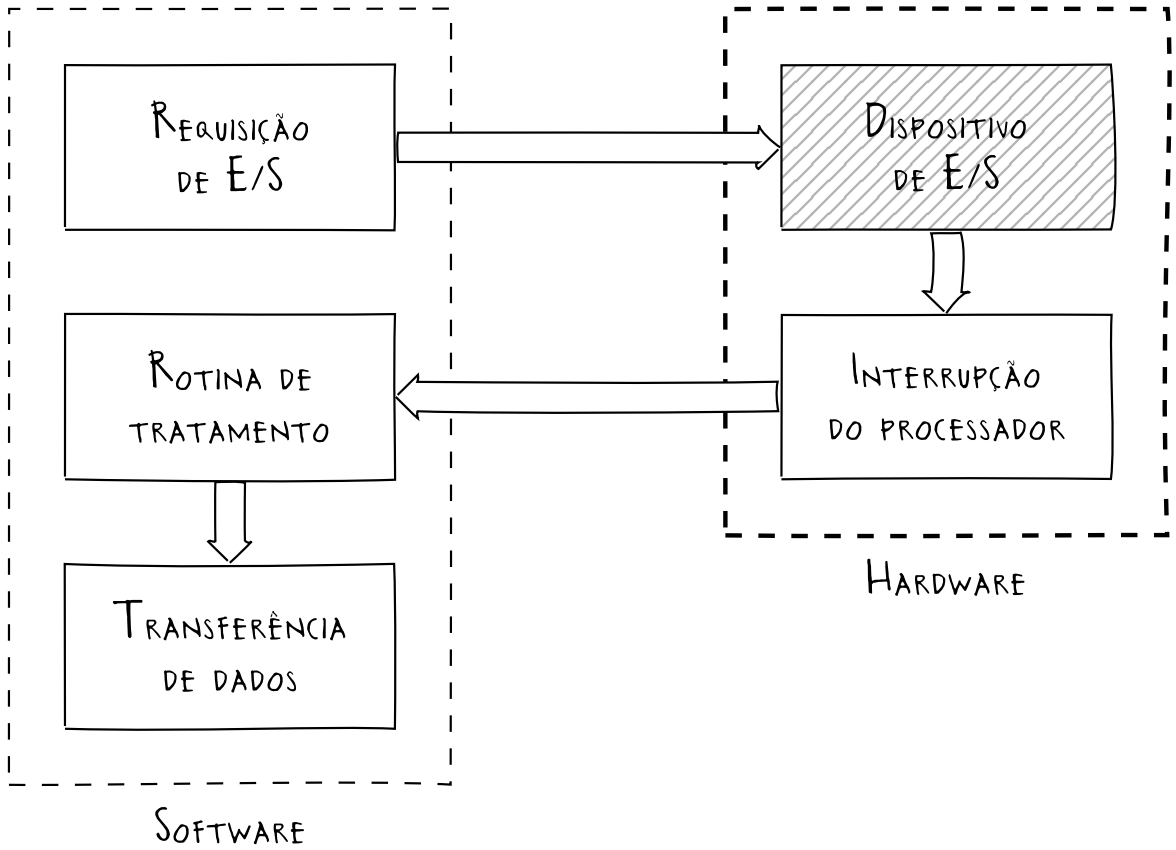
E/S por interrupção

- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



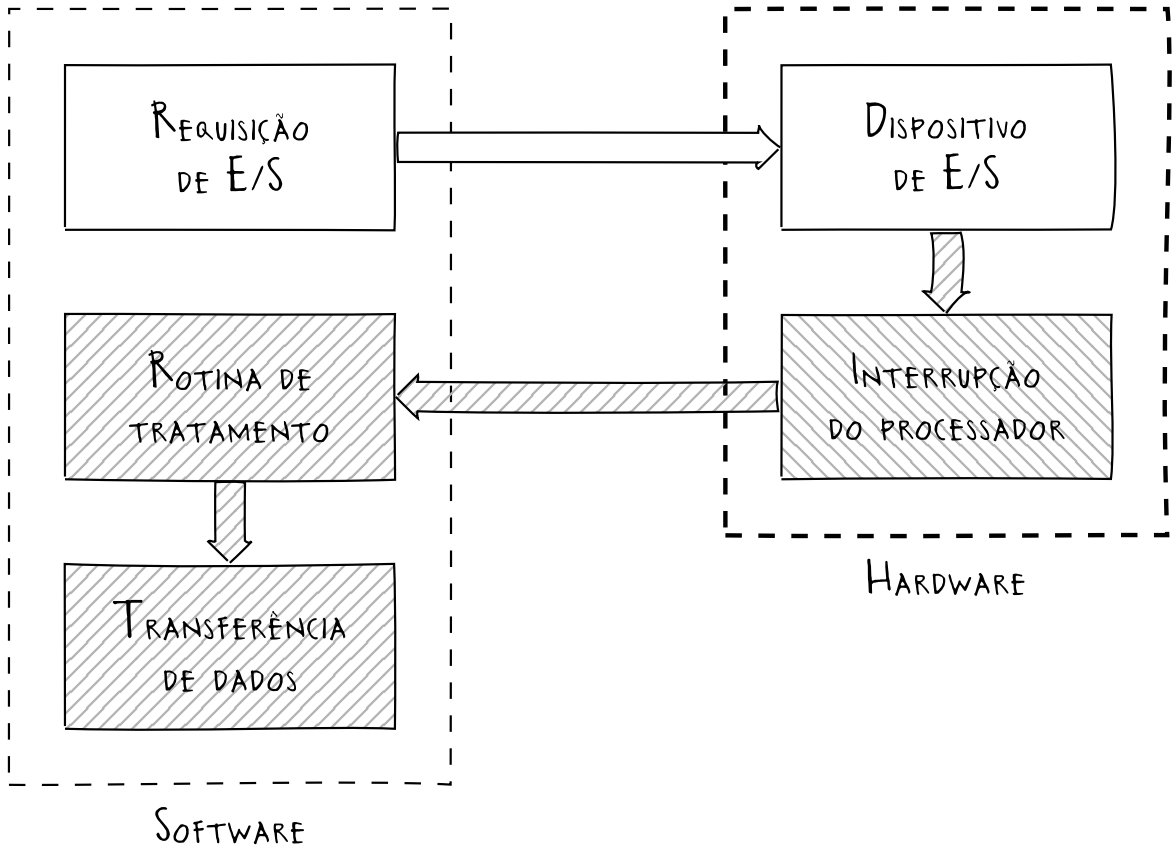
E/S por interrupção

- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



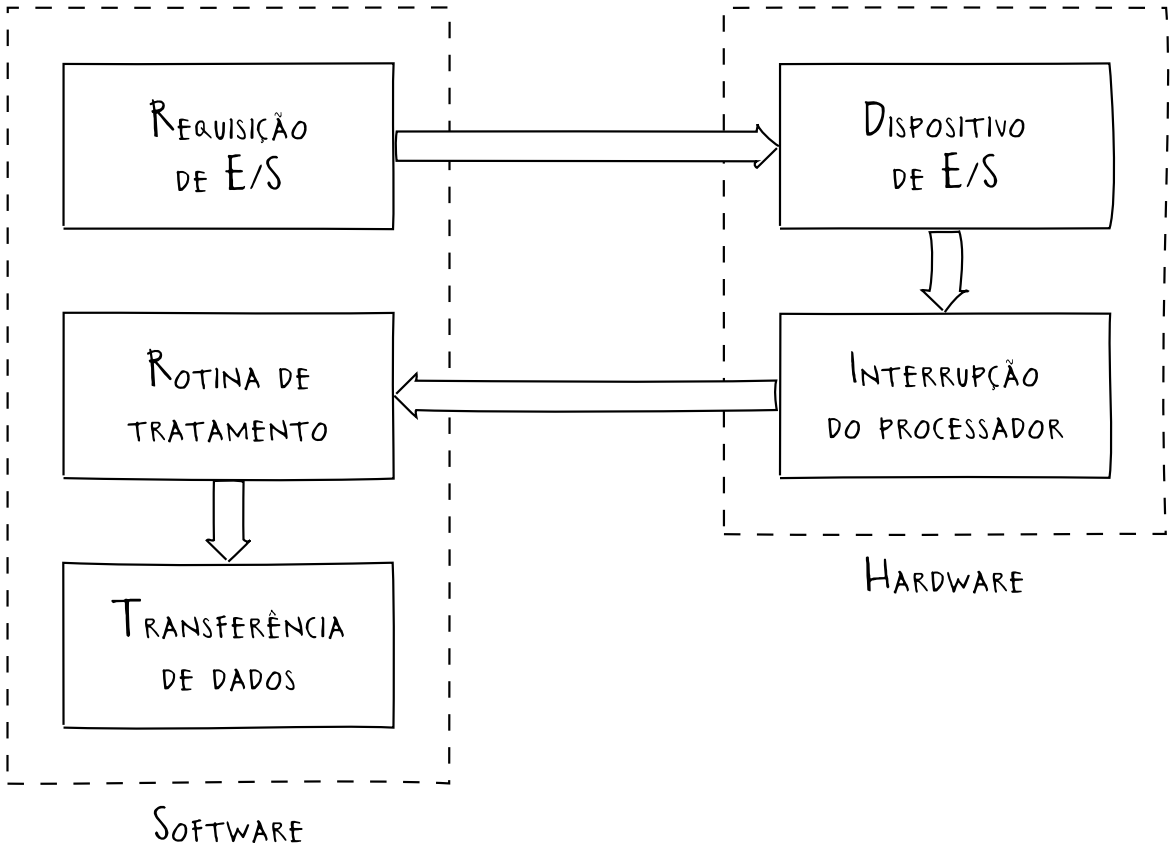
E/S por interrupção

- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



E/S por interrupção

- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)

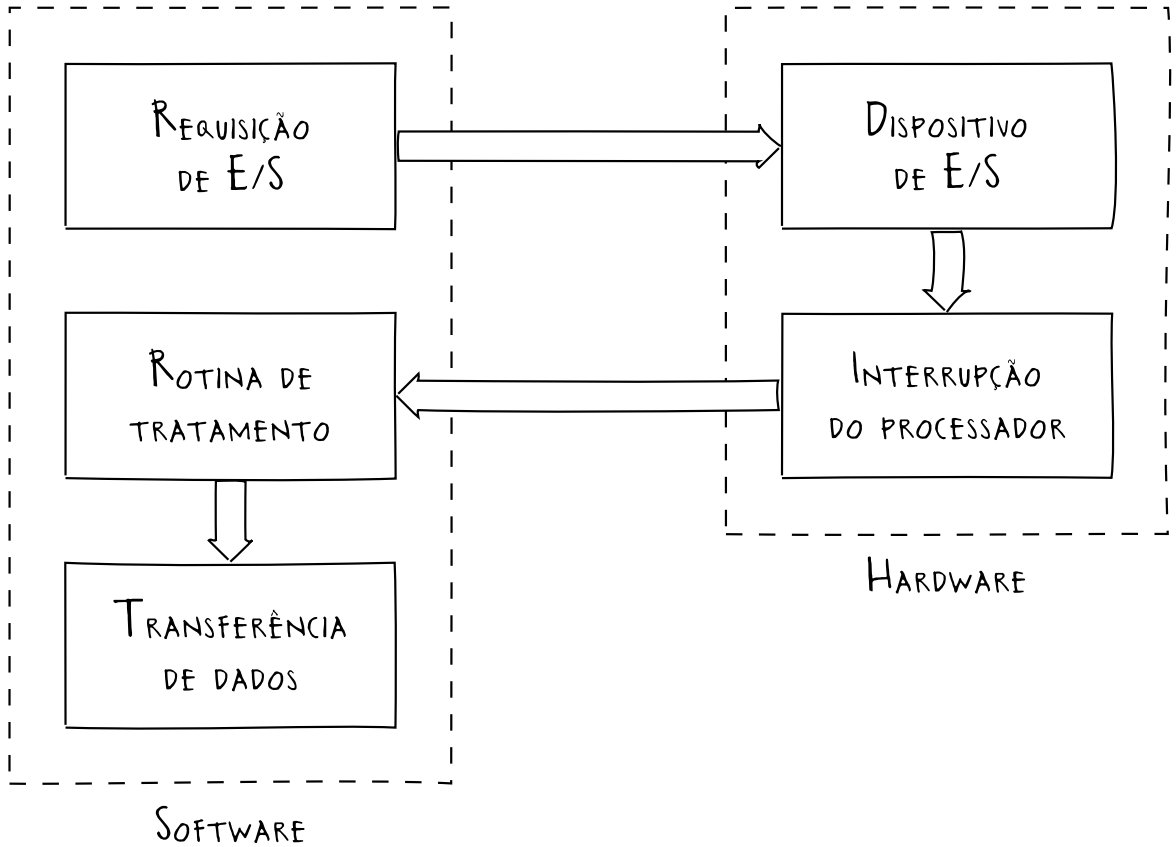


✓ Elimina o processo de espera do processador

✓ Reduz a quantidade de informação transmitida

E/S por interrupção

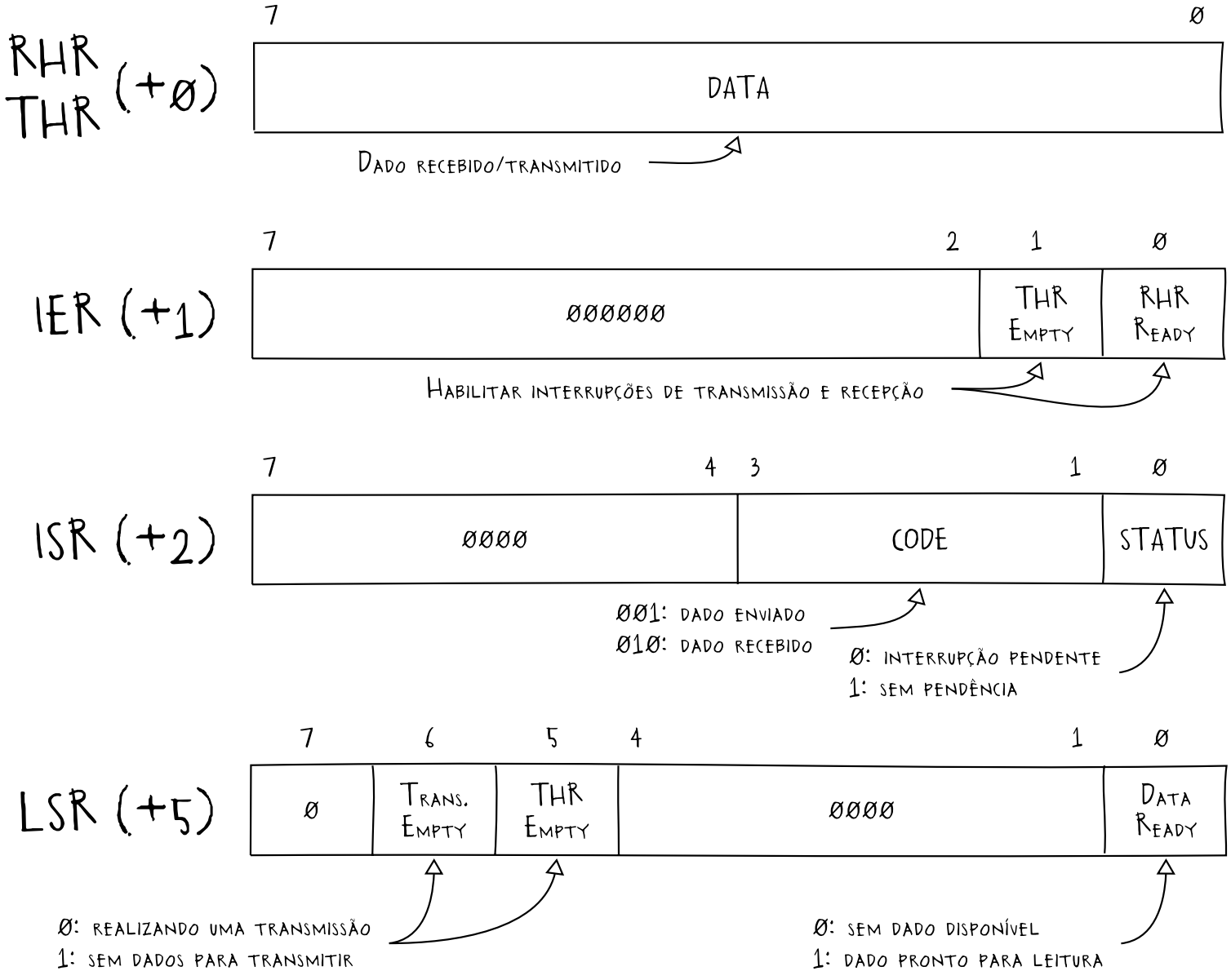
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



- ✗ Processo de entrada e saída mais complexo
- ✗ Necessidade de um controlador de interrupção

E/S por interrupção

- UART 16550



E/S por interrupção

- Função principal

```
106 # Função principal
107 .global main
108 main:
... ...
112     # Ajustando endereço da UART
113     li a0, 0x10000000
114     # Ajustando endereços de prioridade, pendência,
            habilitação e limiar/reinvidicação/finalização
            do PLIC
115     li a1, 0x0c000000
116     li a2, 0x0c001000
117     li a3, 0x0c002000
118     li a4, 0x0c200000
119     # Obtendo ponteiro da string de números
120     la a5, numbers
... ...
```

E/S por interrupção

- Configuração da UART 16550

```
96  # Configuração da UART
97  uart_configuration:
98      # Checando status de interrupção
99      lb t0, 2(a0)
100     # Habilitando interrupção de transmissão
101     li t0, 2
102     sb t0, 1(a0)
103     # Retornando da chamada
104     ret
```

E/S por interrupção

- Configuração do PLIC

```
48  # Configuração do PLIC
49  plic_configuration:
50      # Ajustando prioridade da UART para 1 (fonte 10)
51      li t0, 1
52      sw t0, 40(a1)
53      # Checando interrupção pendente do PLIC UART (bit
            10)
54      lw t0, 0(a2)
55      # Habilitando a interrupção do PLIC UART (bit 10)
56      li t0, 0x400
57      sw t0, 0(a3)
58      # Checando o limiar de prioridade do contexto 0 do
            PLIC
59      lw t0, 0(a4)
60      # Retornando da chamada
61      ret
```

E/S por interrupção

- Envio de dados pela UART

```
63  # Enviar dados
64  send_data:
65      # Laço de envio de dados
66      send_data_loop:
67          # Lendo byte da string
68          lb t0, 0(a5)
69          # Enviando byte para UART
70          sb t0, 0(a0)
71          # Verificando se tem dado disponível
72          lb t0, 5(a0)
73          andi t0, t0, 1
74          # Checando se string terminou
75          bne t0, zero, send_data_loop
76      # Final do envio de dados
77      send_data_end:
78          # Retorno da chamada
79          ret
```

E/S por interrupção

- Rotina de tratamento da interrupção externa

```
24  # Tratamento da interrupção externa
25  .global external_interruption_handler
26  external_interruption_handler:
…       ...
32      # Obtendo causa do evento
33      csrr a0, mcause
34      # Reivindicando interrupção pendente
35      lw t0, 4(a4)
36      # Incrementando índice da string
37      addi a5, a5, 1
38      # Finalizando interrupção pendente
39      sw t0, 4(a4)
…       ...
45      # Retornando do evento
46      mret
```

Acesso direto à memória

- Mesmo com a utilização de E/S por interrupção, o processador ainda é responsável pela transferência dos dados entre os dispositivos e a memória

Acesso direto à memória

- Mesmo com a utilização de E/S por interrupção, o processador ainda é responsável pela transferência dos dados entre os dispositivos e a memória

  - ✗ A velocidade de transferência depende do dispositivo
  - ✗ Todo o processo é gerenciado pelo processador

Acesso direto à memória

- Para a transferência de grandes quantidades de dados, a técnica de acesso direto à memória ou *Direct Memory Access* (DMA) é a mais eficiente
  - O processador configura o DMA com informações sobre os endereços de origem e de destino, além da quantidade de dados que será transferida

Acesso direto à memória

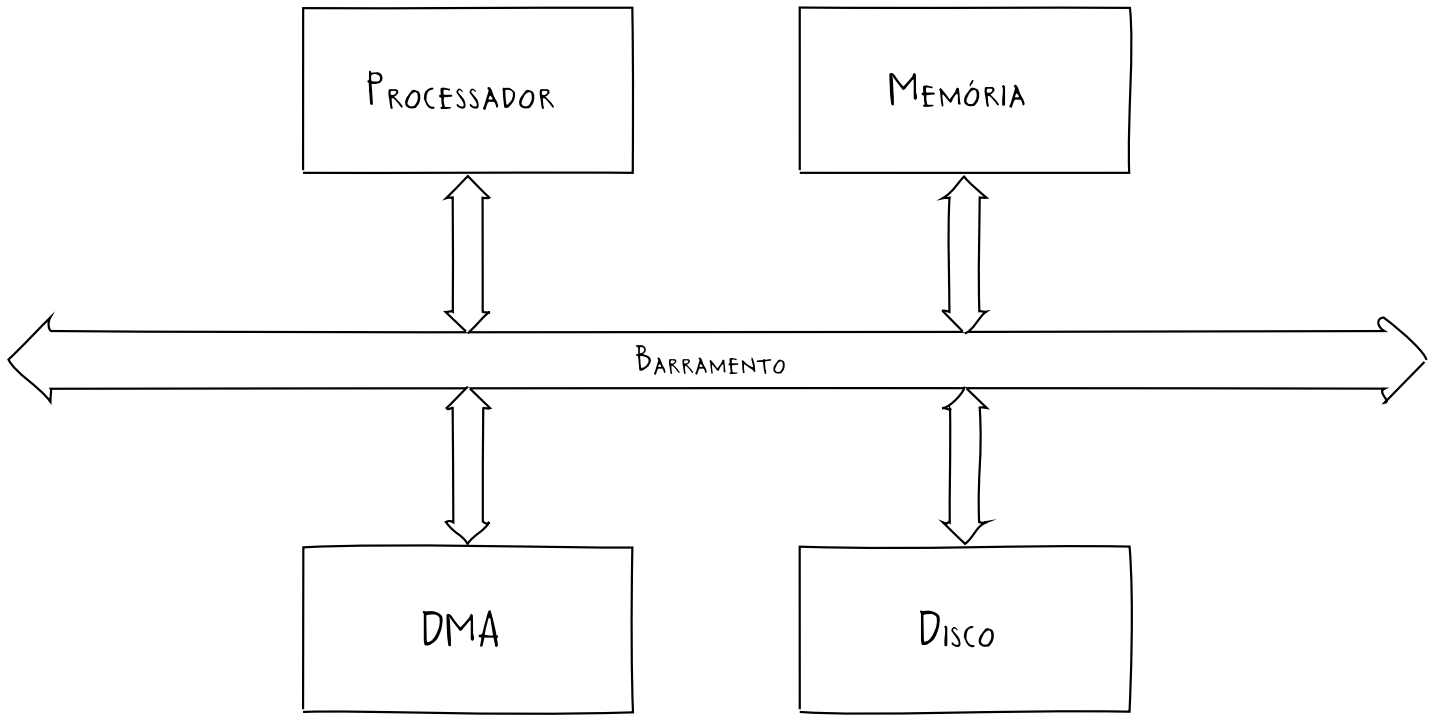
- Para a transferência de grandes quantidades de dados, a técnica de acesso direto à memória ou *Direct Memory Access* (DMA) é a mais eficiente
  - O processador configura o DMA com informações sobre os endereços de origem e de destino, além da quantidade de dados que será transferida

✓ É dedicado para controle e transferência de dados

✓ A transferência não depende do processador

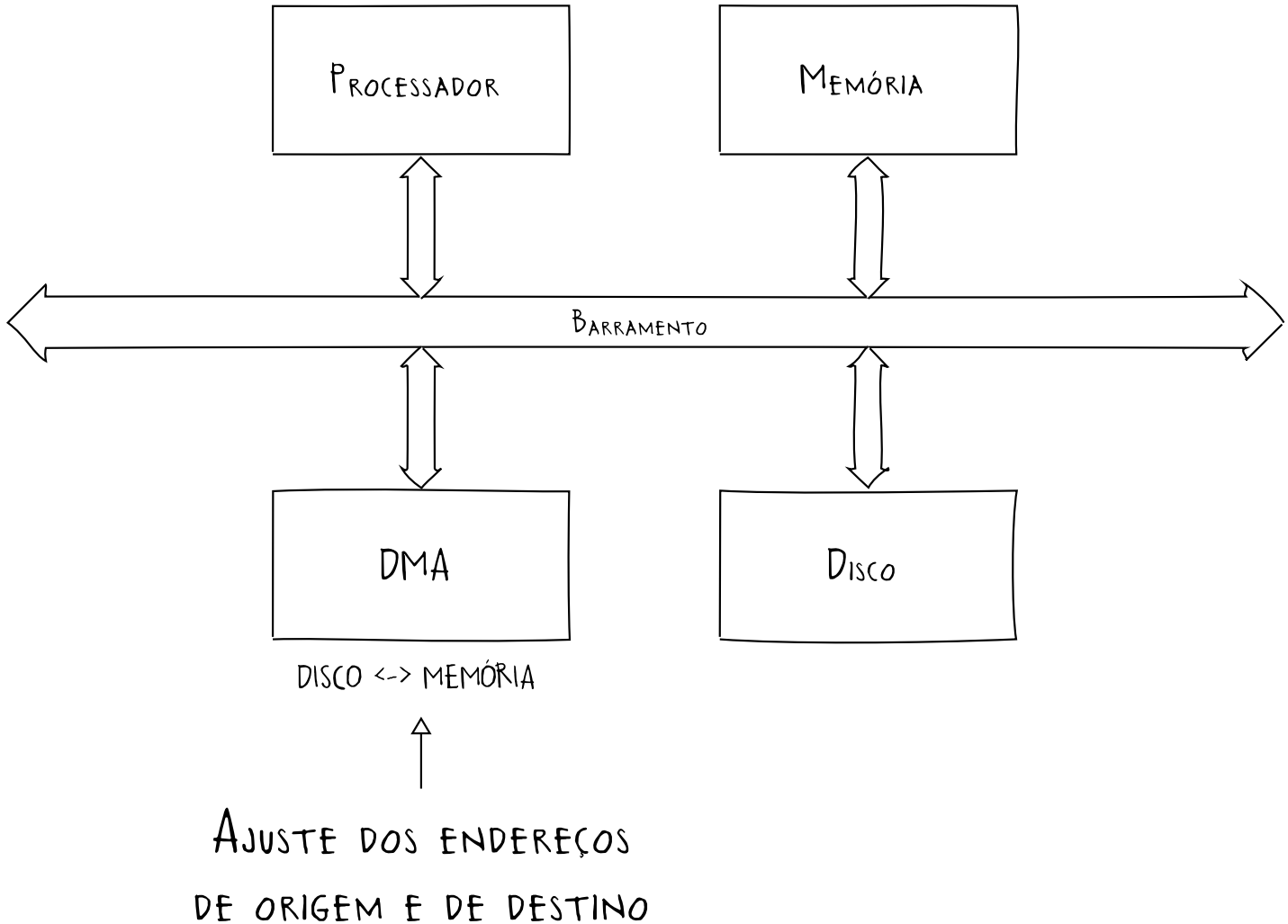
Acesso direto à memória

- Configuração do DMA



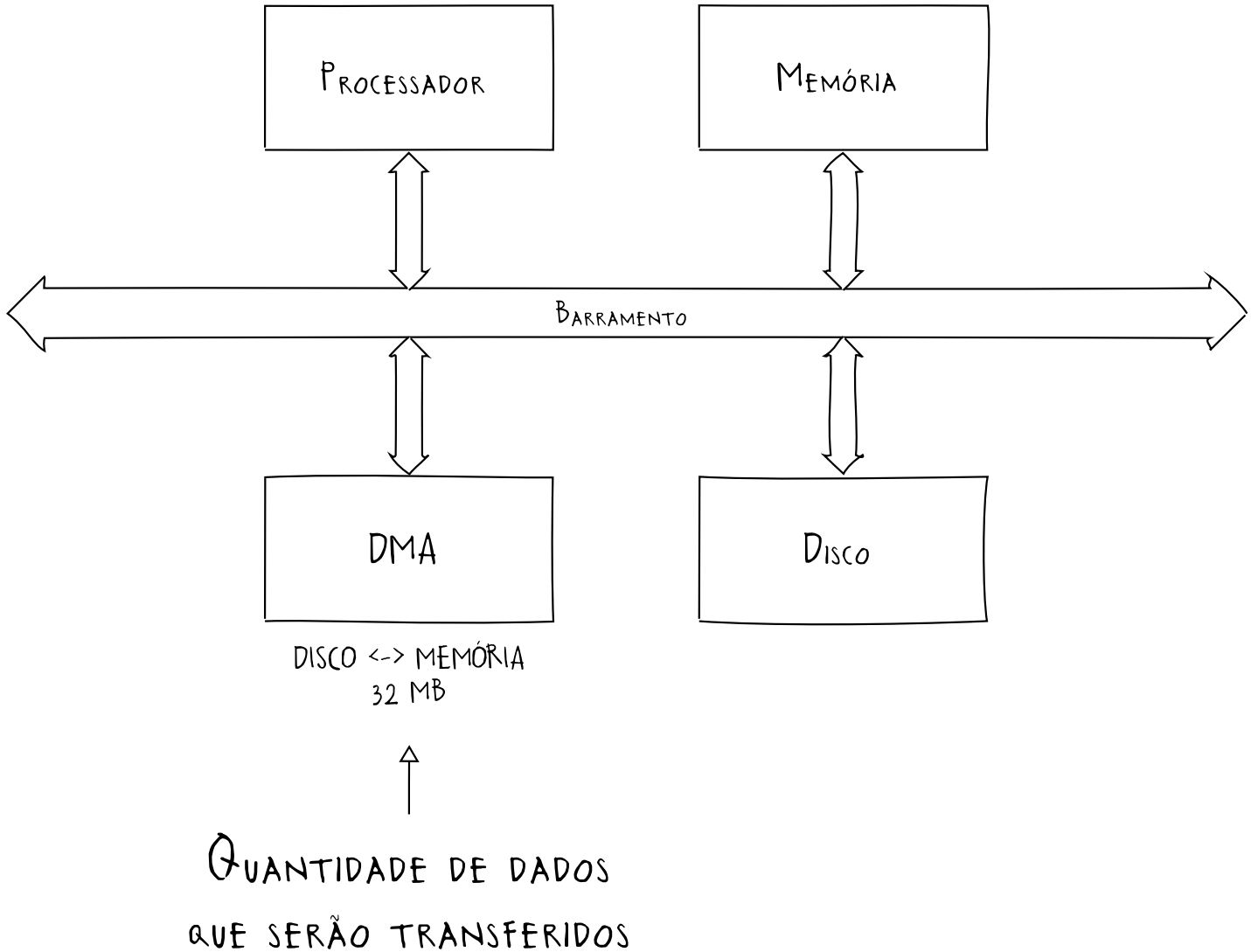
Acesso direto à memória

- Configuração do DMA



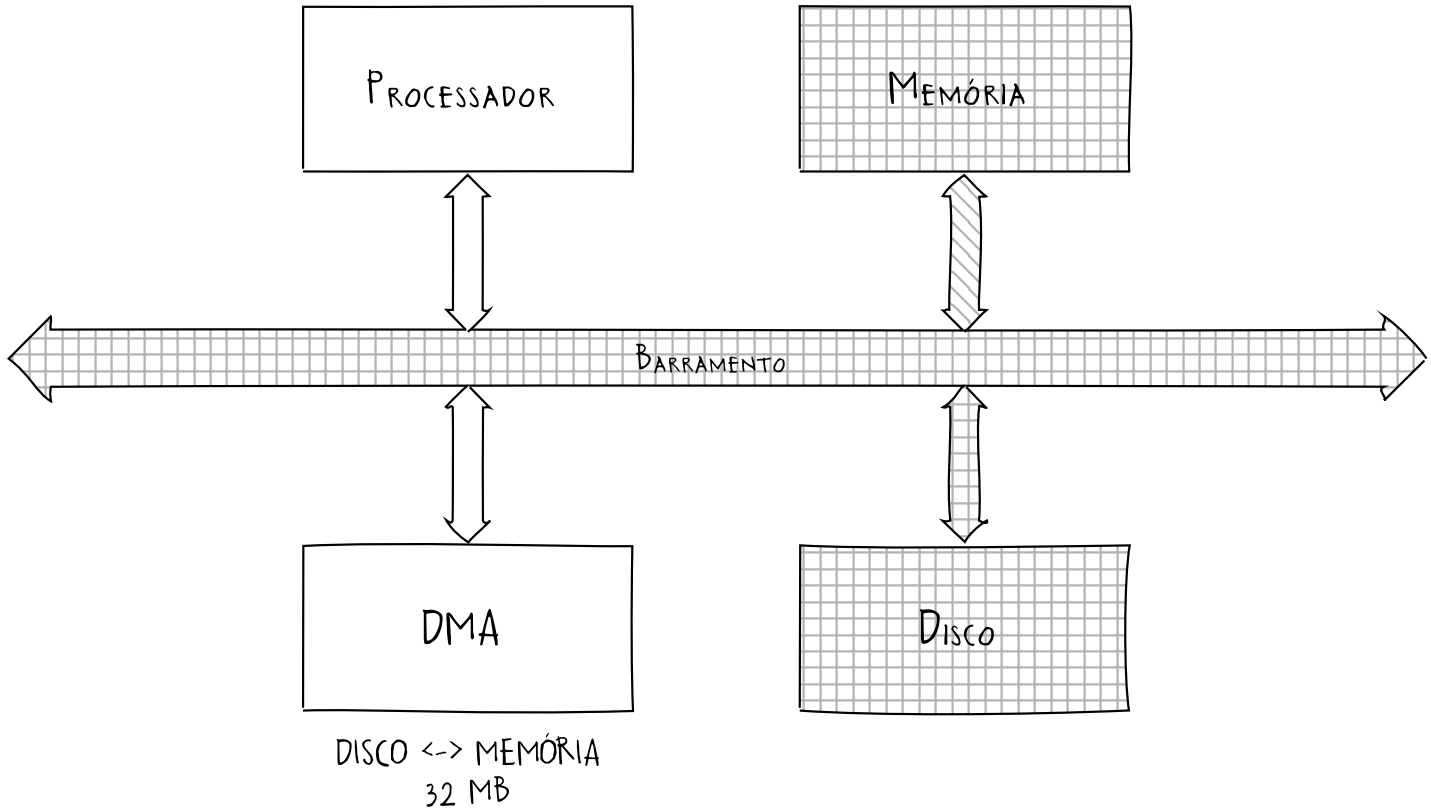
Acesso direto à memória

- Configuração do DMA



Acesso direto à memória

- Configuração do DMA



Transferência dos dados,
sem ocupar o processador

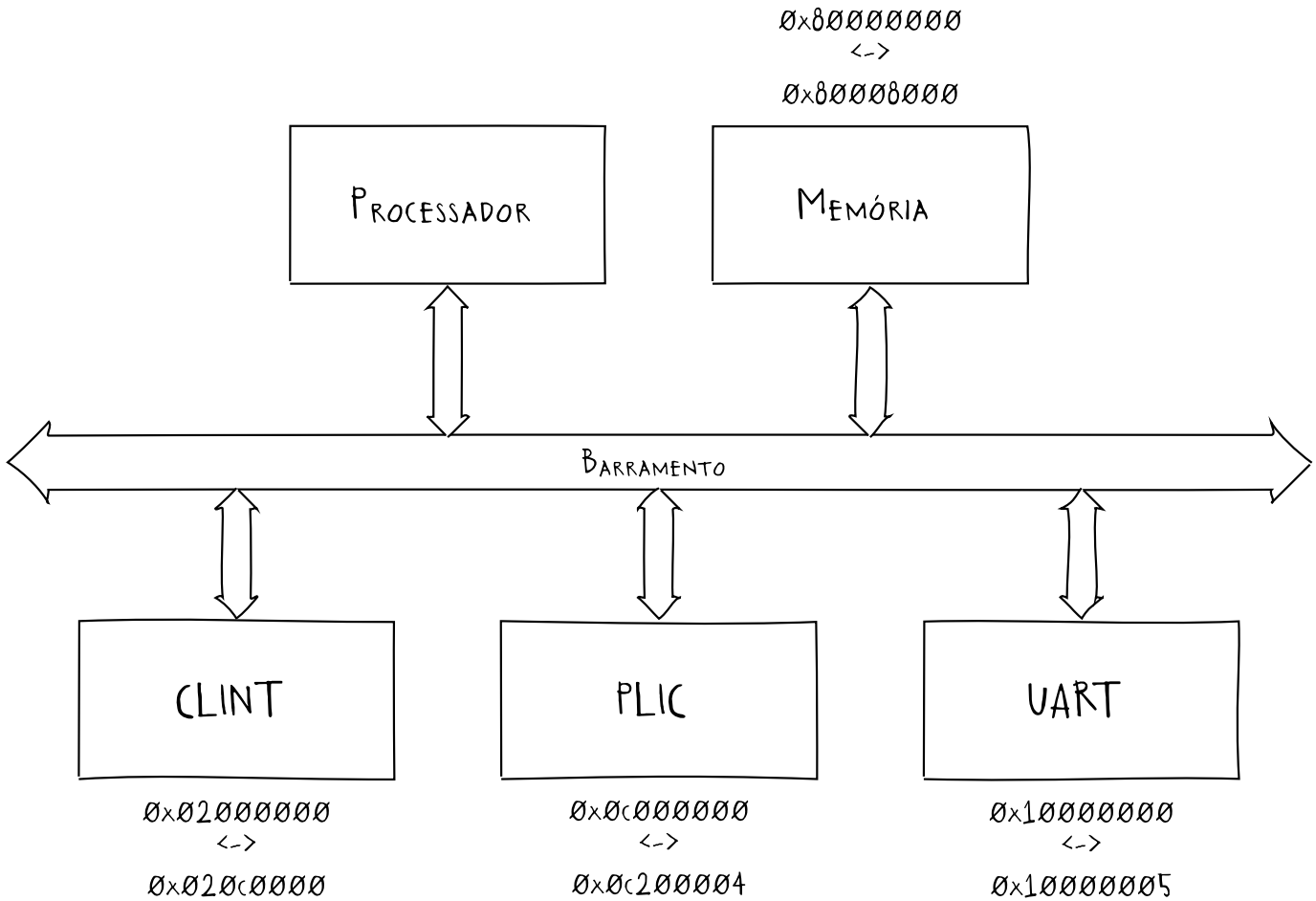
Acesso direto à memória

- Análise comparativa de desempenho de E/S

Tipo de E/S	Polling	Tamanho	Controlada pelo processador
Programada	Sim	Palavra	Sim
Por interrupção	Não	Palavra	Sim
DMA	Não	Bloco	Não

Exercício

- Implemente os seguintes dispositivos de E/S mapeados em memória, com suporte para interrupção
  - *Core Local INTerruptor* (CLINT)
  - *Platform-Level Interrupt Controller* (PLIC)
  - *Universal Asynchronous Receiver/Transmitter* (UART)



Exercício

- Implemente um software em linguagem de montagem (*assembly*) que realize a ordenação crescente de uma sequência de números com sinal e tamanho de 32 bits
  - Os dados serão lidos e escritos byte por byte em codificação ASCII, através de operações de E/S no dispositivo de comunicação serial (UART)
  - No primeiro campo será informada a quantidade de números que serão ordenados no vetor, com um valor máximo de 1000 e os números separados por espaço

```
10
5 13 -1 3 0 2 1 8 1 -2
```

```
-2,-1,0,1,1,2,3,5,8,13
```