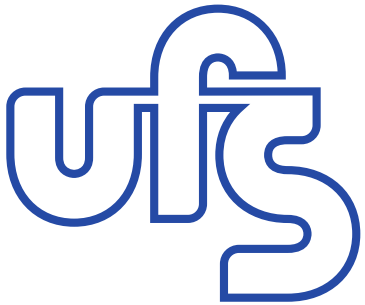
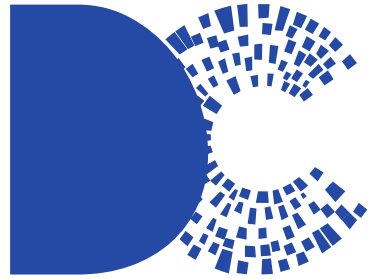


UNIVERSIDADE FEDERAL DE SERGIPE



DEPARTAMENTO DE COMPUTAÇÃO

Entrada e saída

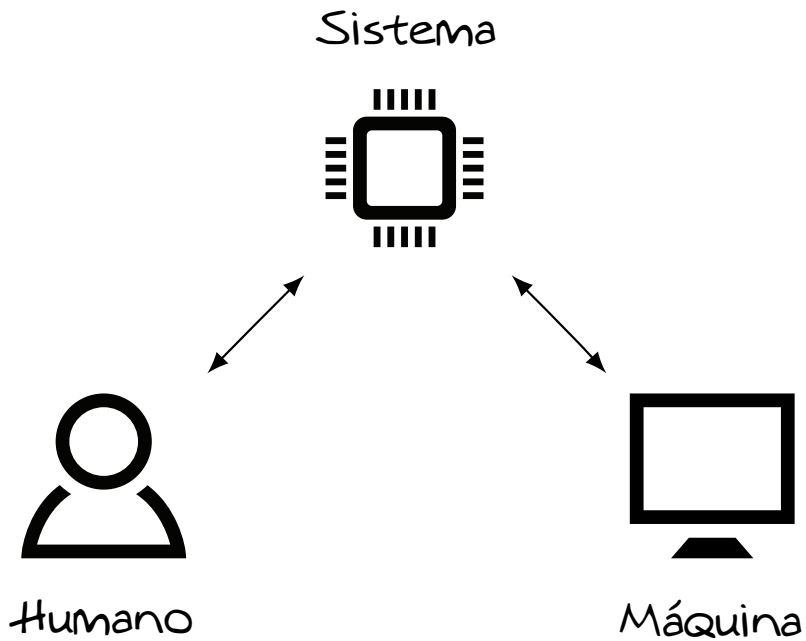
Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

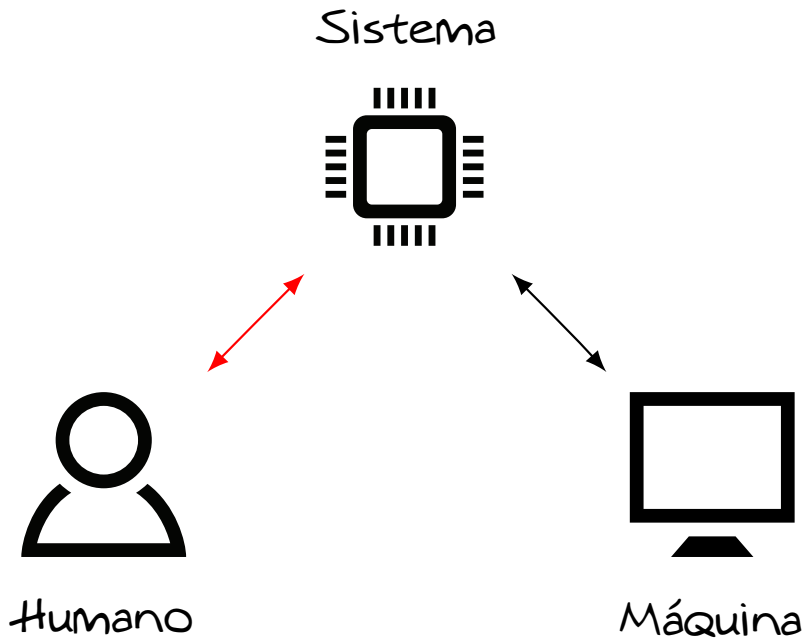
Introdução

- Papel dos dispositivos de entrada e saída (E/S)
  - Comunicação do sistema com o mundo exterior



Introdução

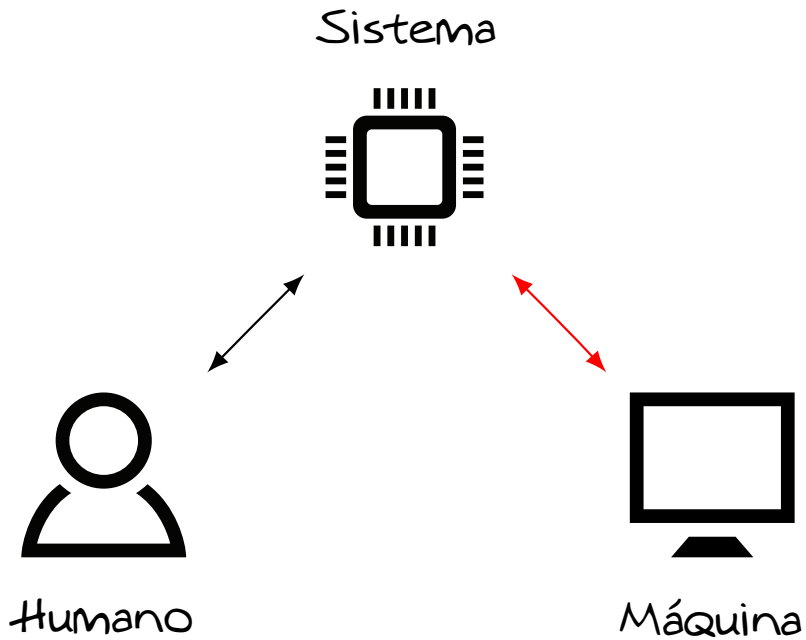
- Papel dos dispositivos de entrada e saída (E/S)
  - Comunicação do sistema com o mundo exterior



Interação com o usuário (humano)

Introdução

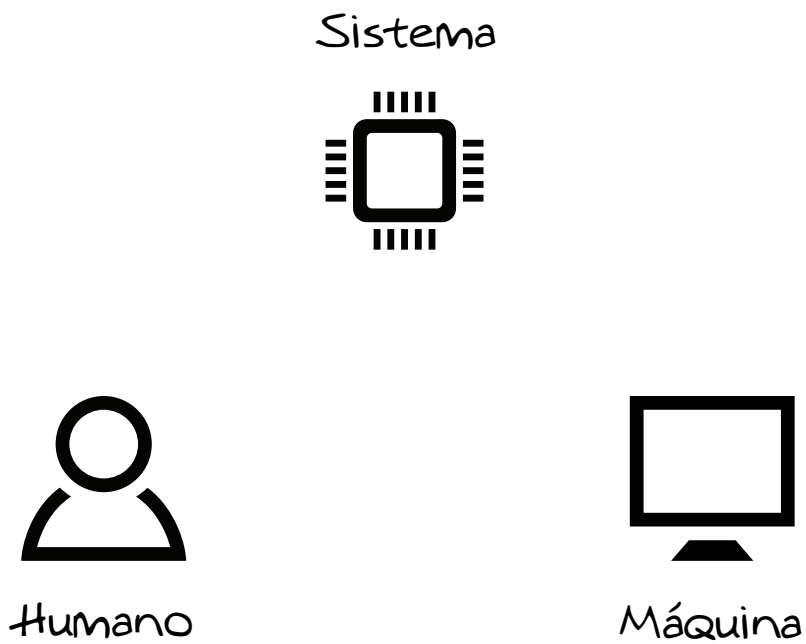
- Papel dos dispositivos de entrada e saída (E/S)
  - Comunicação do sistema com o mundo exterior



Interface com outros sistemas (computacional)

Introdução

- Papel dos dispositivos de entrada e saída (E/S)
  - Comunicação do sistema com o mundo exterior



Como seria utilizar um sistema sem dispositivos de E/S?

Introdução

- Classificação dos dispositivos de E/S

Introdução

- Classificação dos dispositivos de E/S
  - Comportamento
    - Somente escrita (impressora)
    - Somente leitura (teclado)
    - Escrita e leitura (disco rígido)

Introdução

- Classificação dos dispositivos de E/S
  - Comportamento
    - Somente escrita (impressora)
    - Somente leitura (teclado)
    - Escrita e leitura (disco rígido)
  - Interface
    - Humana (tela sensível ao toque)
    - Computacional (rede)

Introdução

- Classificação dos dispositivos de E/S
  - Comportamento
    - Somente escrita (impressora)
    - Somente leitura (teclado)
    - Escrita e leitura (disco rígido)
  - Interface
    - Humana (tela sensível ao toque)
    - Computacional (rede)
  - Taxa de dados
    - Define qual a taxa de transmissão de dados com a qual um determinado dispositivo é capaz de operar
    - Teclado possui uma taxa máxima de 10 bytes/s

Introdução

- Comparativo entre diferentes dispositivos

Dispositivo	Comportamento	Interface	Taxa de dados (Mbit/s)
Teclado	Entrada	Humano	0,0001
Tela	Saída	Humano	800 - 8.000
Placa de rede	Entrada/Saída	Sistema	100 - 1.000
Impressora	Saída	Sistema	3,2

Introdução

- Comparativo entre diferentes dispositivos

Dispositivo	Comportamento	Interface	Taxa de dados (Mbit/s)
Teclado	Entrada	Humano	0,0001
Tela	Saída	Humano	800 - 8.000
Placa de rede	Entrada/Saída	Sistema	100 - 1.000
Impressora	Saída	Sistema	3,2

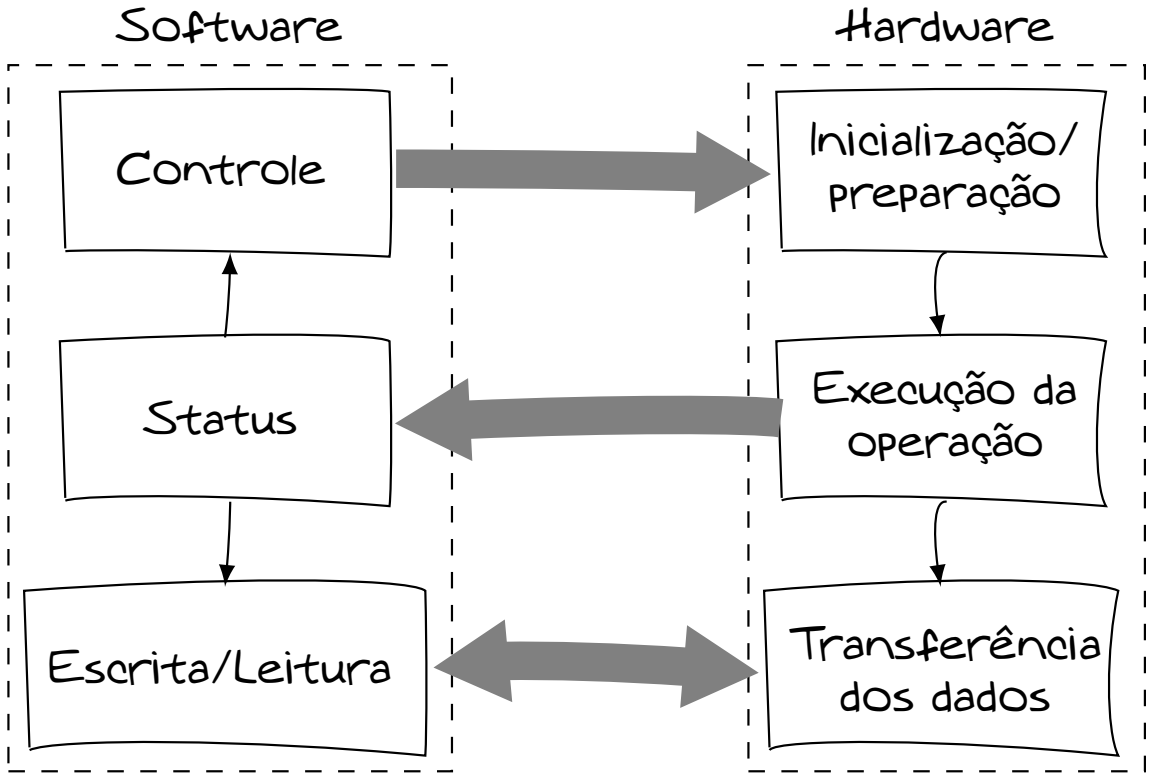
Para medição dos dados para armazenamento e transmissão é adotada a notação SI
(1 MB = 1.000.000 bytes)

Introdução

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

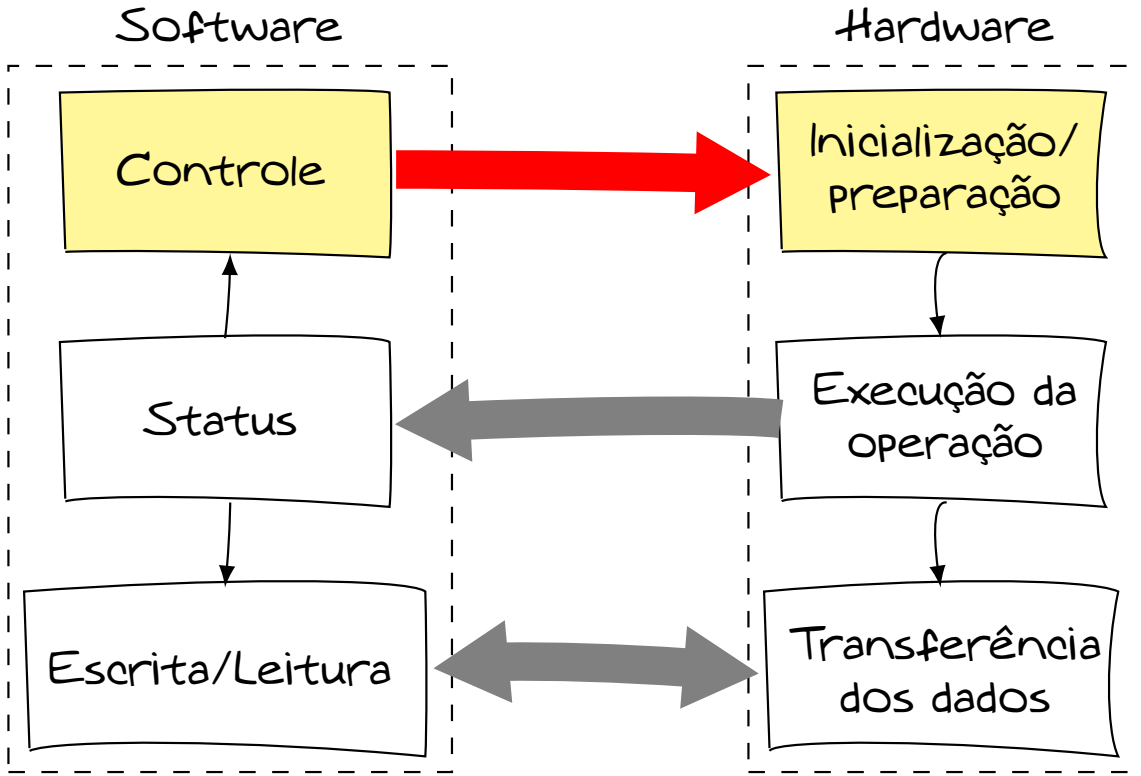
E/S programada

- O que é entrada e saída programada?
  - Todas as operações de E/S são realizadas diretamente pelo processador através do gerenciador de dispositivo (*driver*)



E/S programada

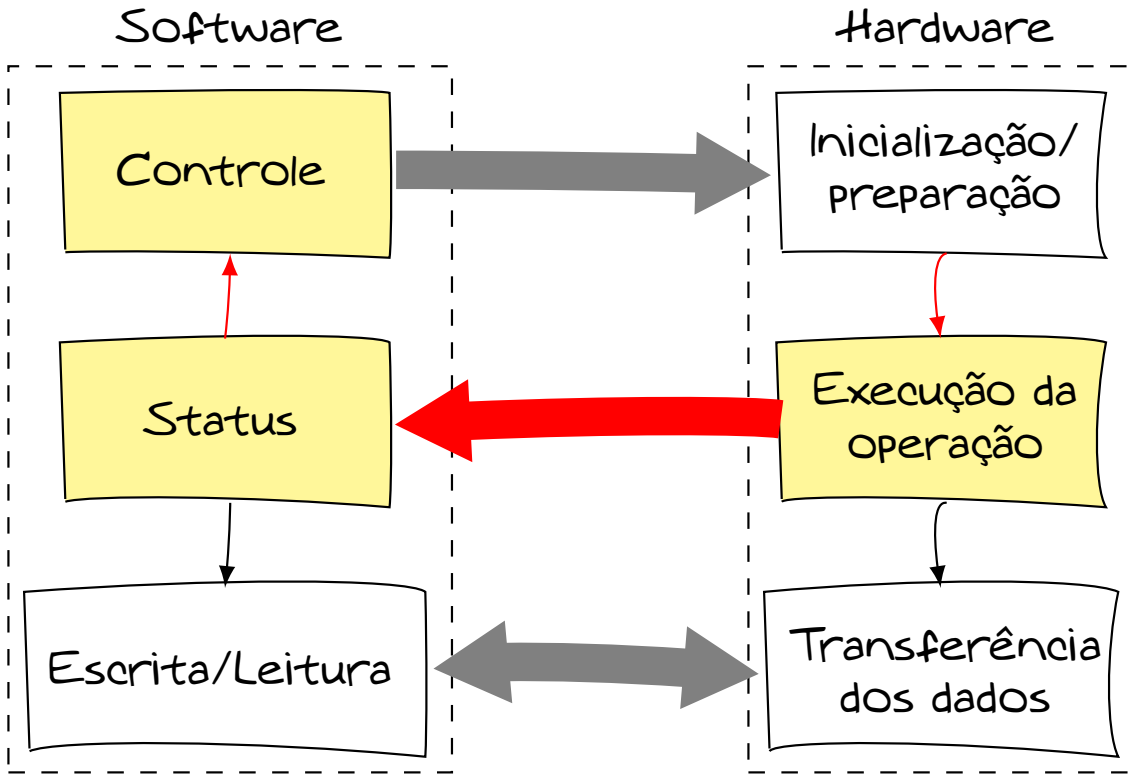
- O que é entrada e saída programada?
  - Todas as operações de E/S são realizadas diretamente pelo processador através do gerenciador de dispositivo (*driver*)



É feita a inicialização ou preparação do dispositivo para realização da operação de E/S

E/S programada

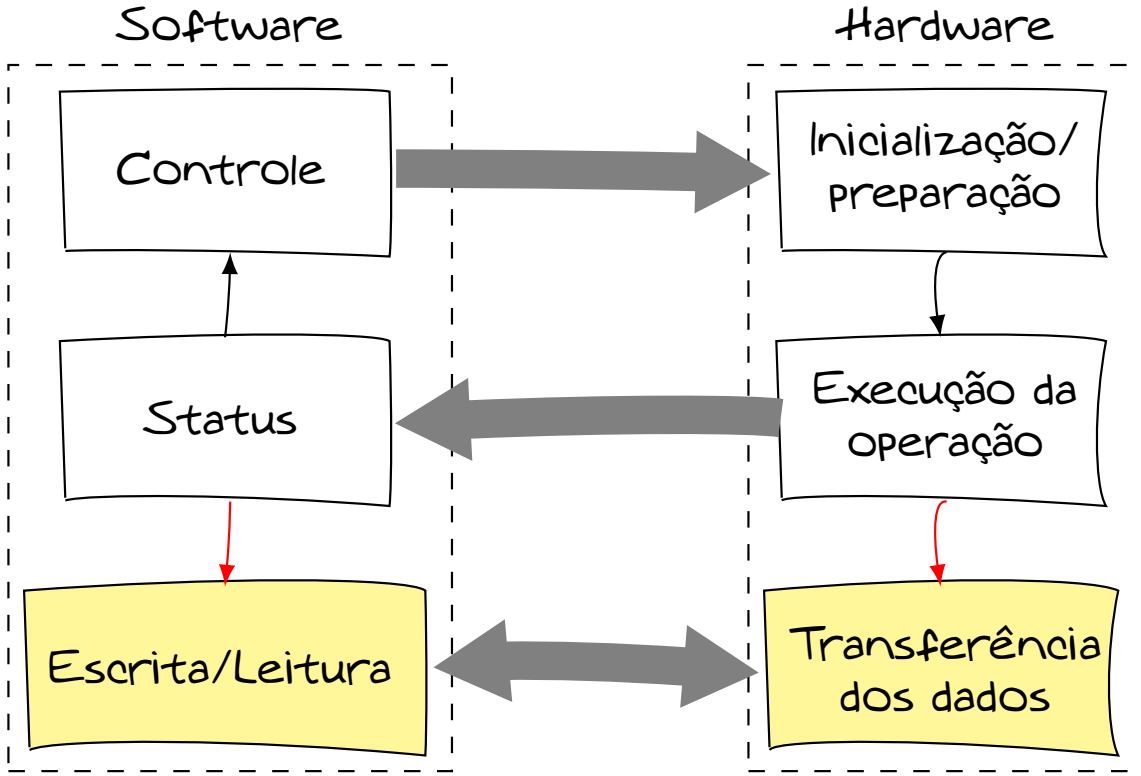
- O que é entrada e saída programada?
  - Todas as operações de E/S são realizadas diretamente pelo processador através do gerenciador de dispositivo (*driver*)



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

E/S programada

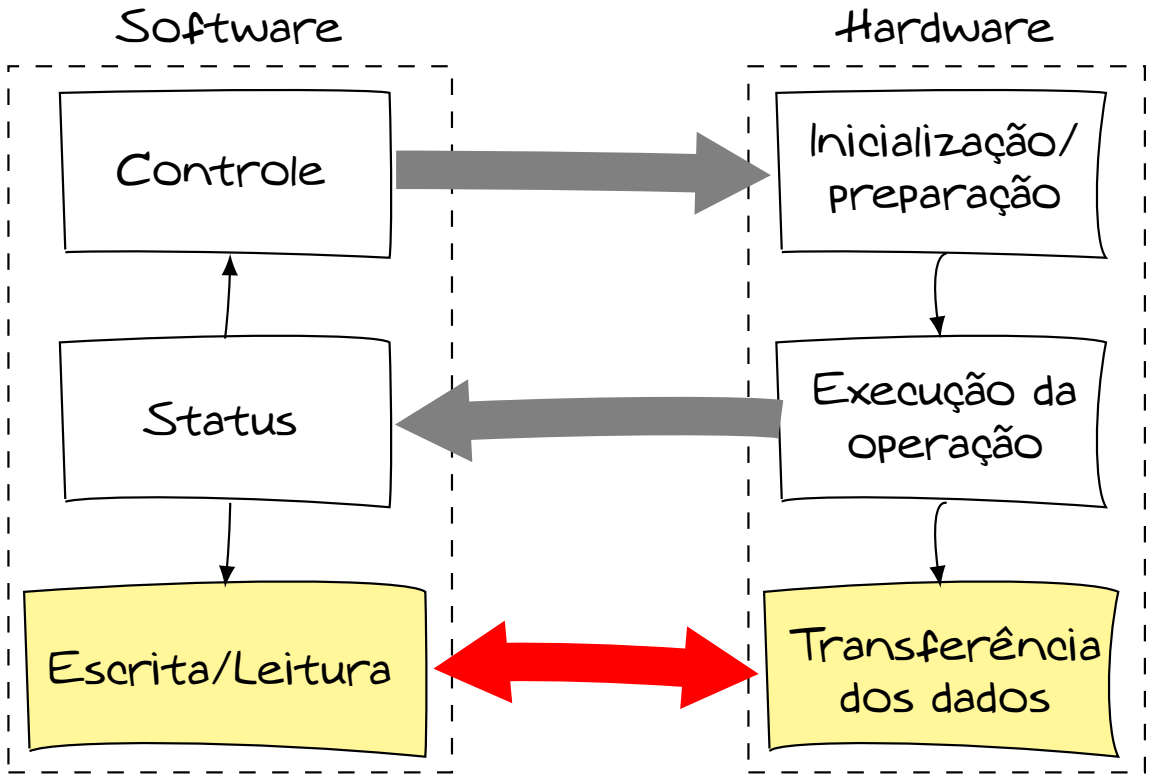
- O que é entrada e saída programada?
  - Todas as operações de E/S são realizadas diretamente pelo processador através do gerenciador de dispositivo (*driver*)



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

E/S programada

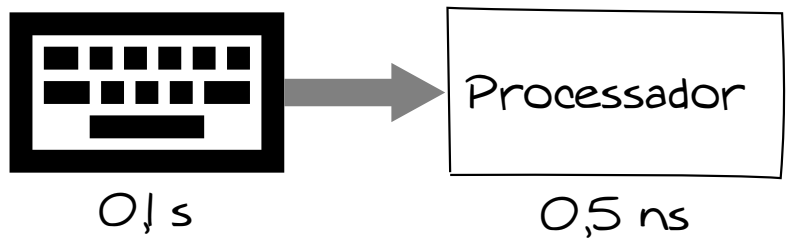
- O que é entrada e saída programada?
  - Todas as operações de E/S são realizadas diretamente pelo processador através do gerenciador de dispositivo (*driver*)



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

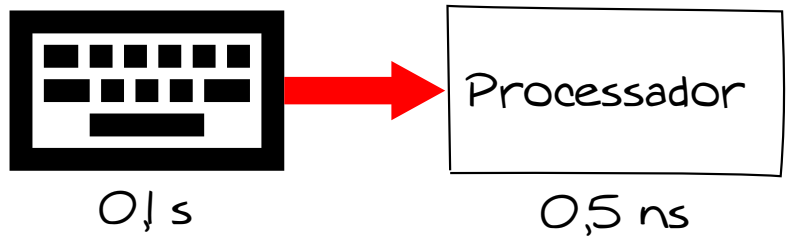
E/S programada

- Leitura do teclado com E/S programada
  - Processador
    - Dispositivo de entrada e saída
    - Frequência de operação de 2 GHz
  - Teclado
    - Dispositivo de entrada
    - Taxa de dados de 10 bytes/s



E/S programada

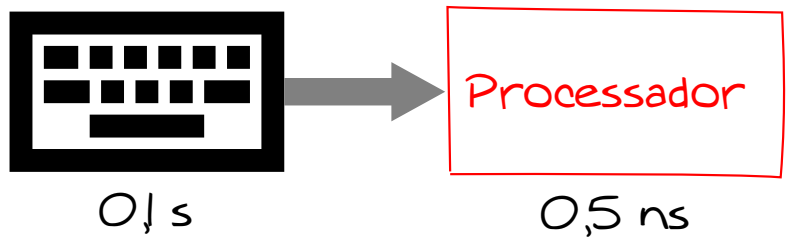
- Leitura do teclado com E/S programada
  - Processador
    - Dispositivo de entrada e saída
    - Frequência de operação de 2 GHz
  - Teclado
    - Dispositivo de entrada
    - Taxa de dados de 10 bytes/s



Em sua velocidade máxima de operação, o teclado é capaz de enviar 1 byte a cada 0,1 segundo

E/S programada

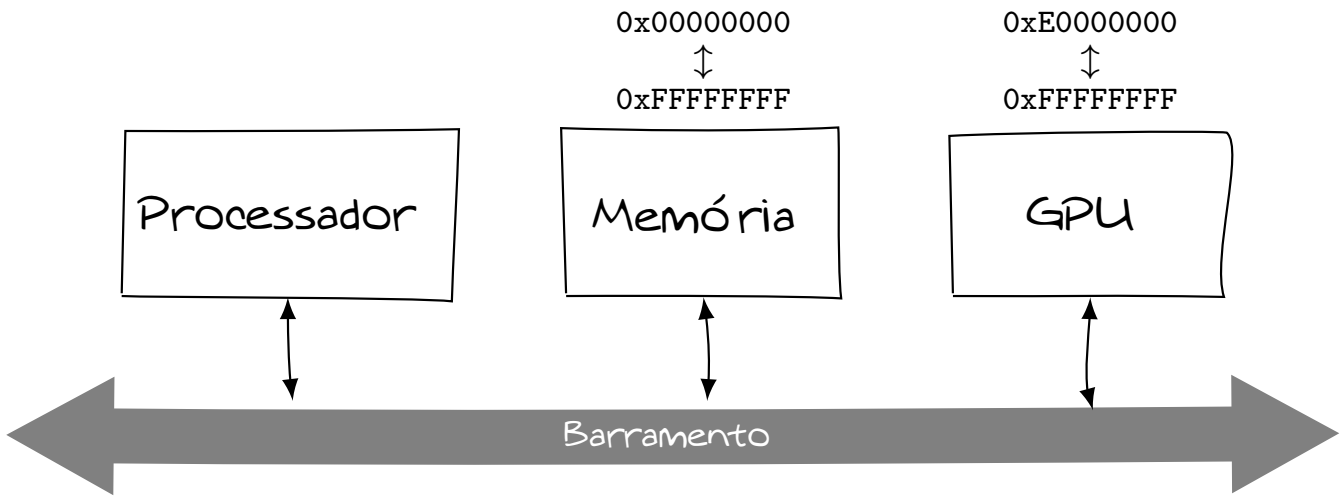
- Leitura do teclado com E/S programada
  - Processador
    - Dispositivo de entrada e saída
    - Frequência de operação de 2 GHz
  - Teclado
    - Dispositivo de entrada
    - Taxa de dados de 10 bytes/s



Durante a espera de 0,1 segundo, o processador poderia ter executado 200 milhões de instruções, mas está aguardando o dado ser recebido pelo teclado

E/S programada

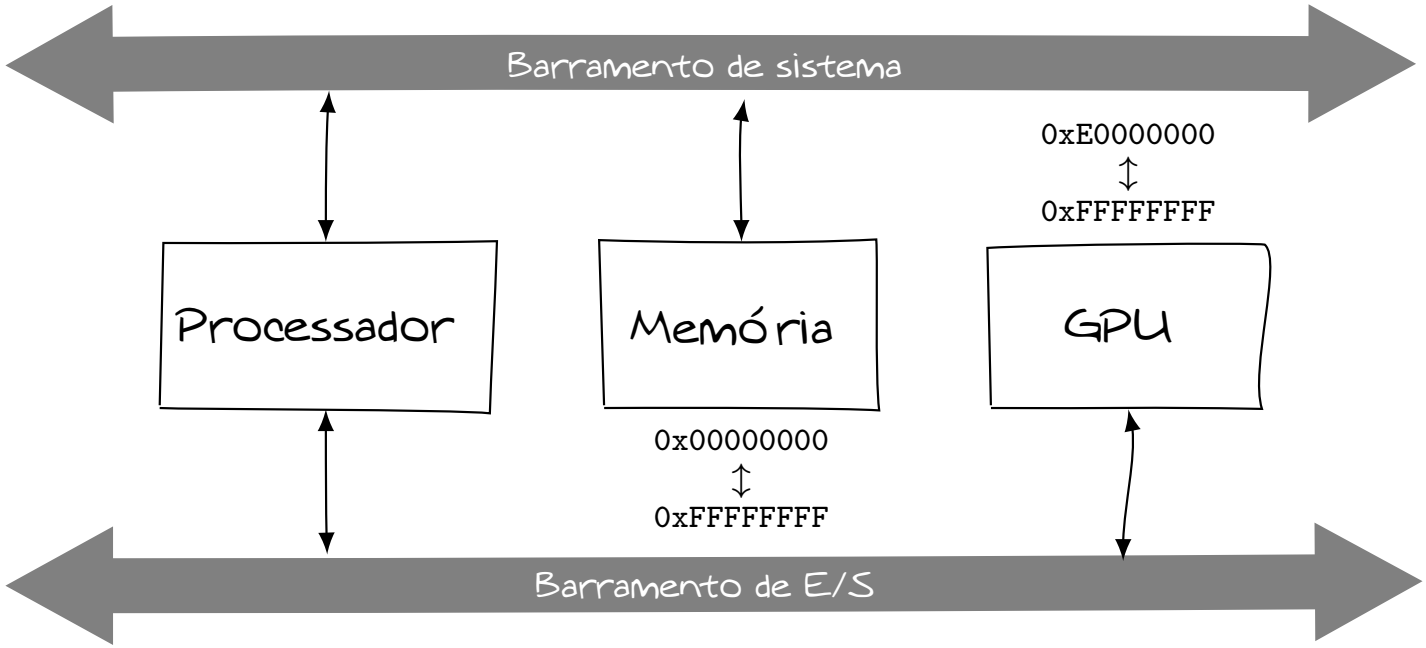
- E/S por mapeamento em memória
  - Operações realizadas com acessos à memória



Unidade de processamento gráfico (GPU)
com memória e endereçamento compartilhado

E/S programada

- E/S por instruções especializadas
  - Instruções especializadas para acesso aos dispositivos



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

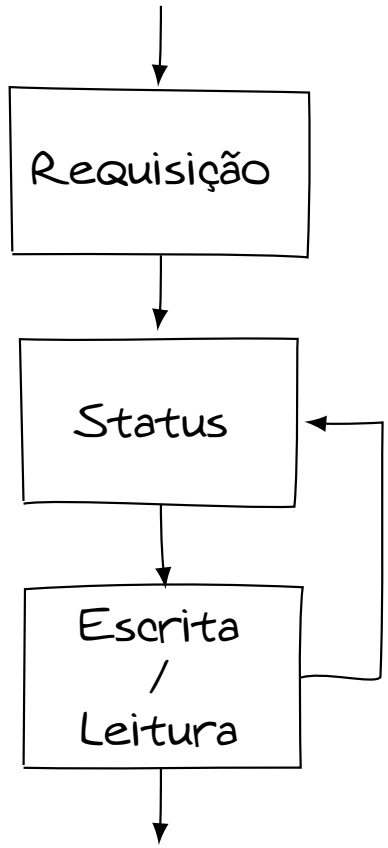
Exemplo

- Traduza o código fonte abaixo para linguagem de montagem e execute o seu comportamento
  - Um contador está mapeado em 0x80808080, sendo decrementado até zero

```
1  // Tipos inteiros de tamanho fixo
2  #include <stdint.h>
3  // Ponteiro para dispositivo
4  volatile uint32_t* const contador = (volatile
       uint32_t*)(0x80808080);
5  // Função principal
6  int main() {
7      // Ajustando contador para 100
8      (*contador) = 0x80000064;
9      // Repete enquanto maior do que zero
10     while((*contador) > 0);
11     // Retorno sem erros
12     return 0;
13 }
```

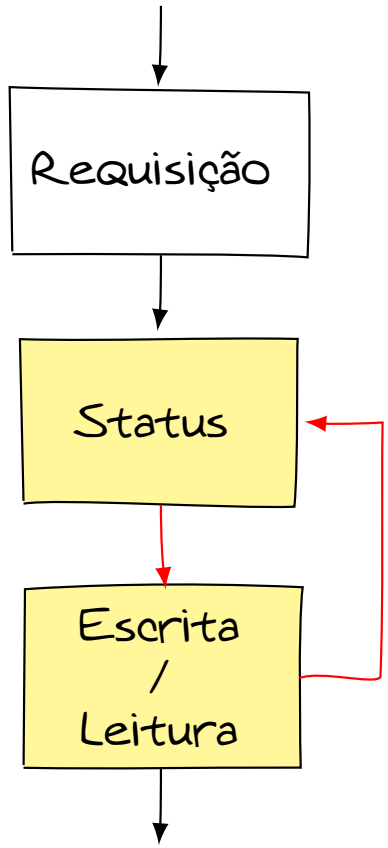
E/S por interrupção

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



E/S por interrupção

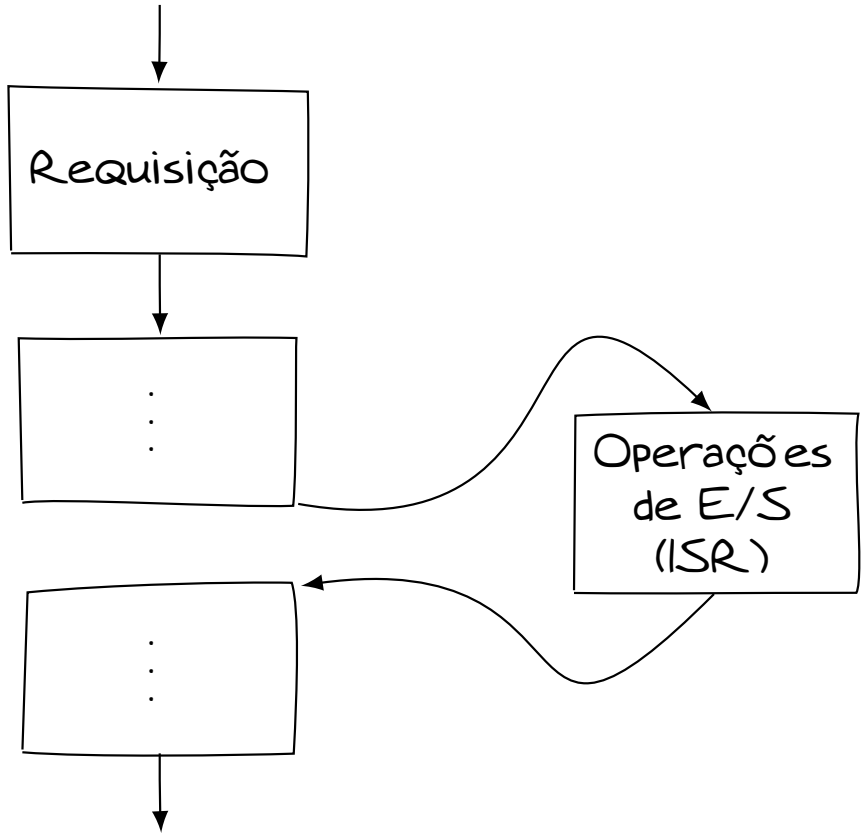
- A E/S programada pode impor ao processador uma longa espera durante a transferência de dados



Acesso bloqueante por *polling*

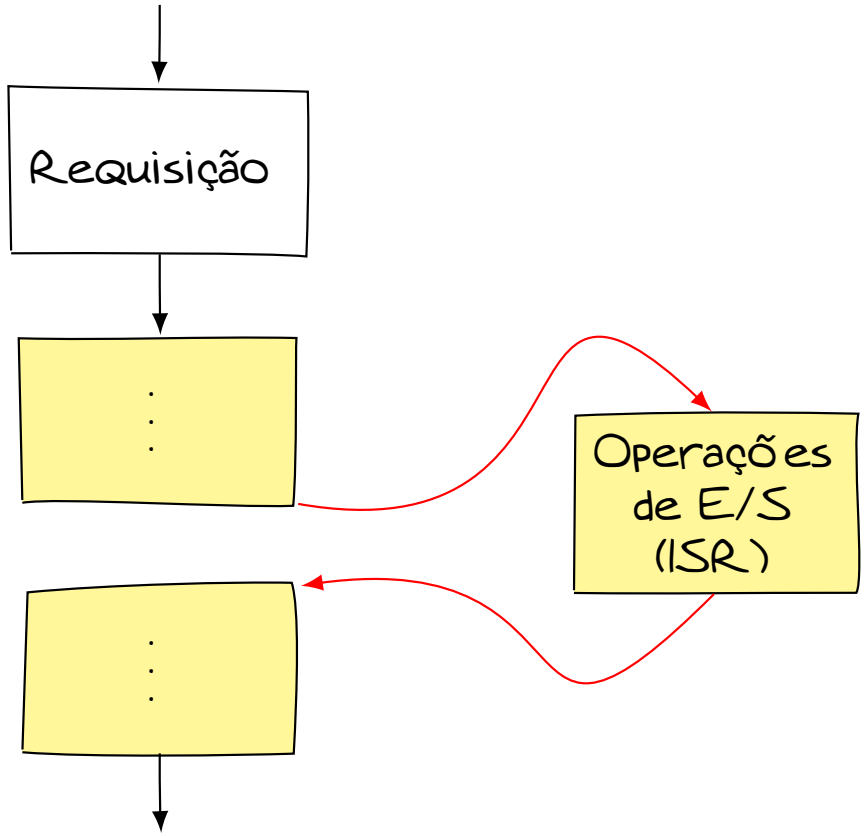
E/S por interrupção

- O processador faz a requisição de operação de E/S para o dispositivo e continua seu fluxo de execução



E/S por interrupção

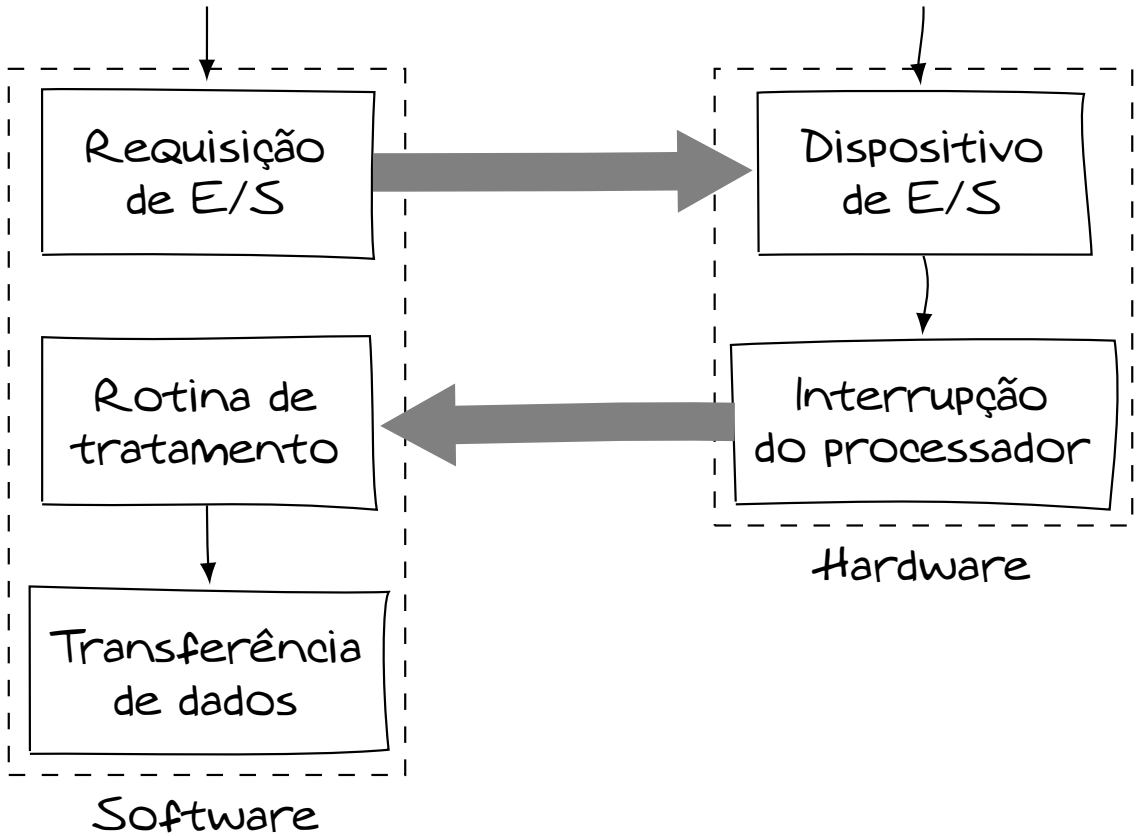
- O processador faz a requisição de operação de E/S para o dispositivo e continua seu fluxo de execução



Quando a requisição está pronta, é gerada uma interrupção que executa as operações de E/S (ISR)

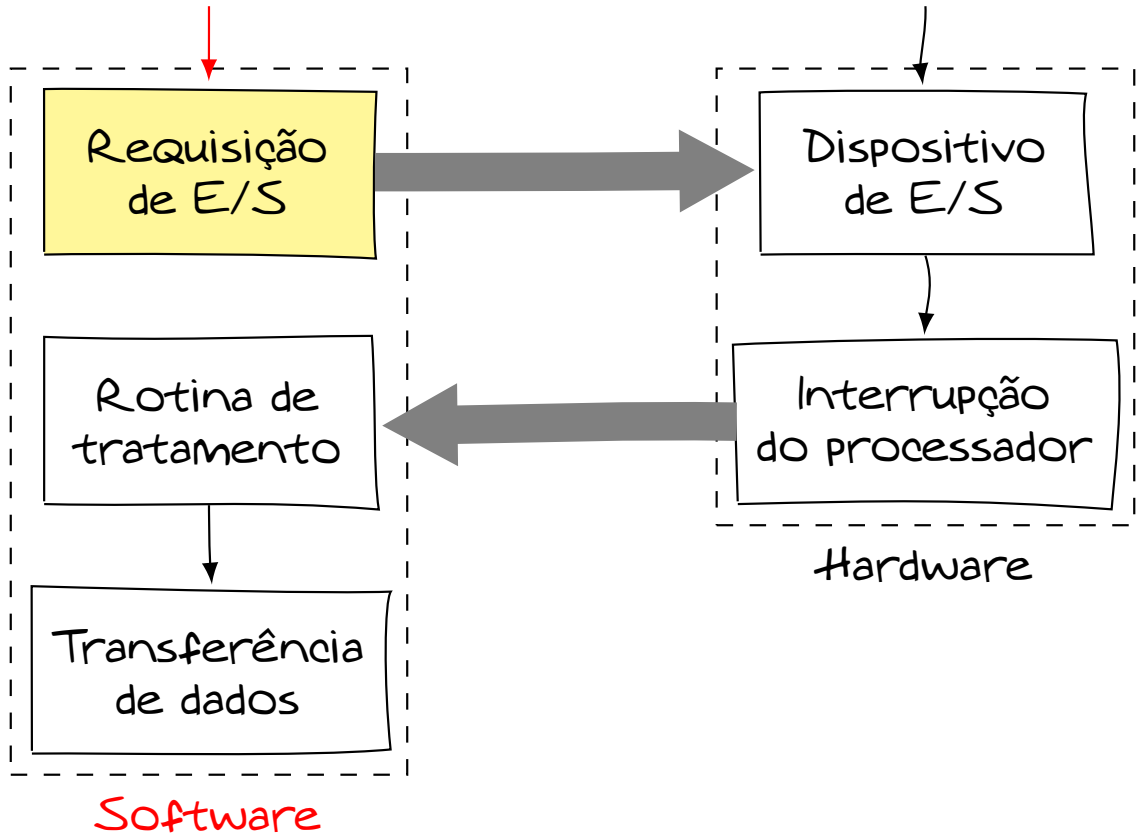
E/S por interrupção

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



E/S por interrupção

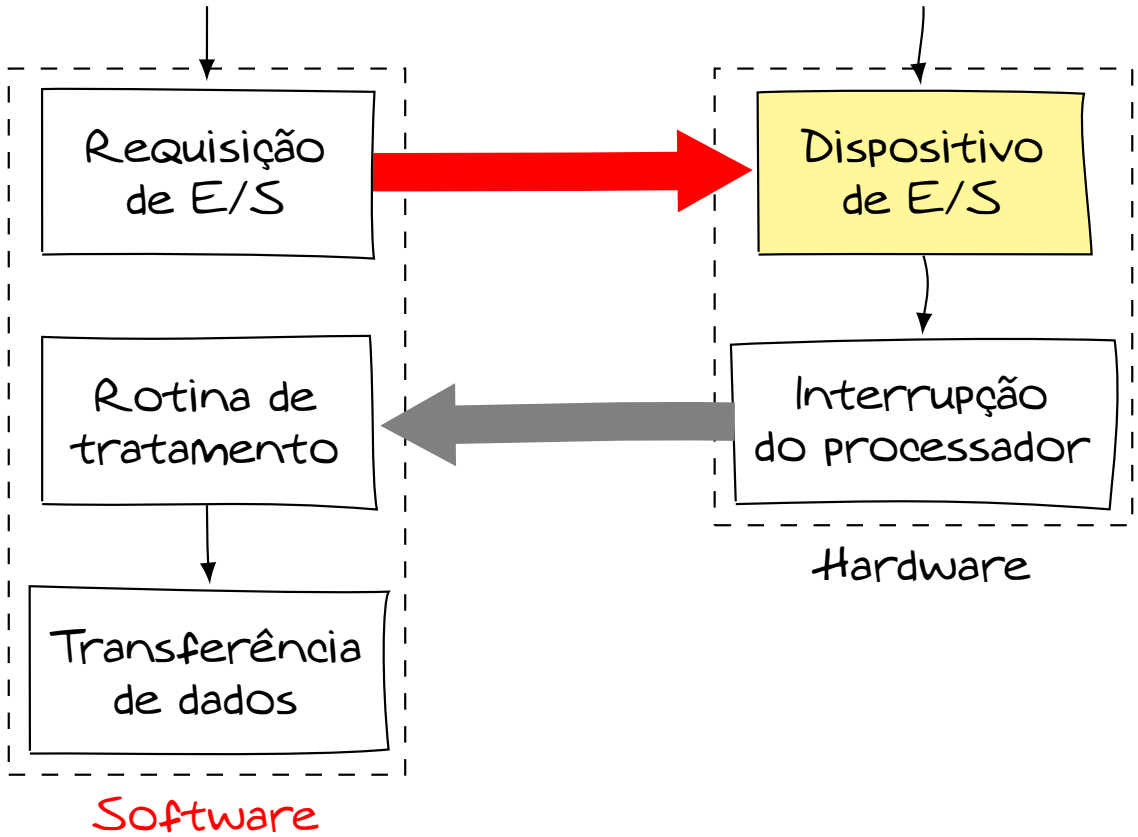
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de software

E/S por interrupção

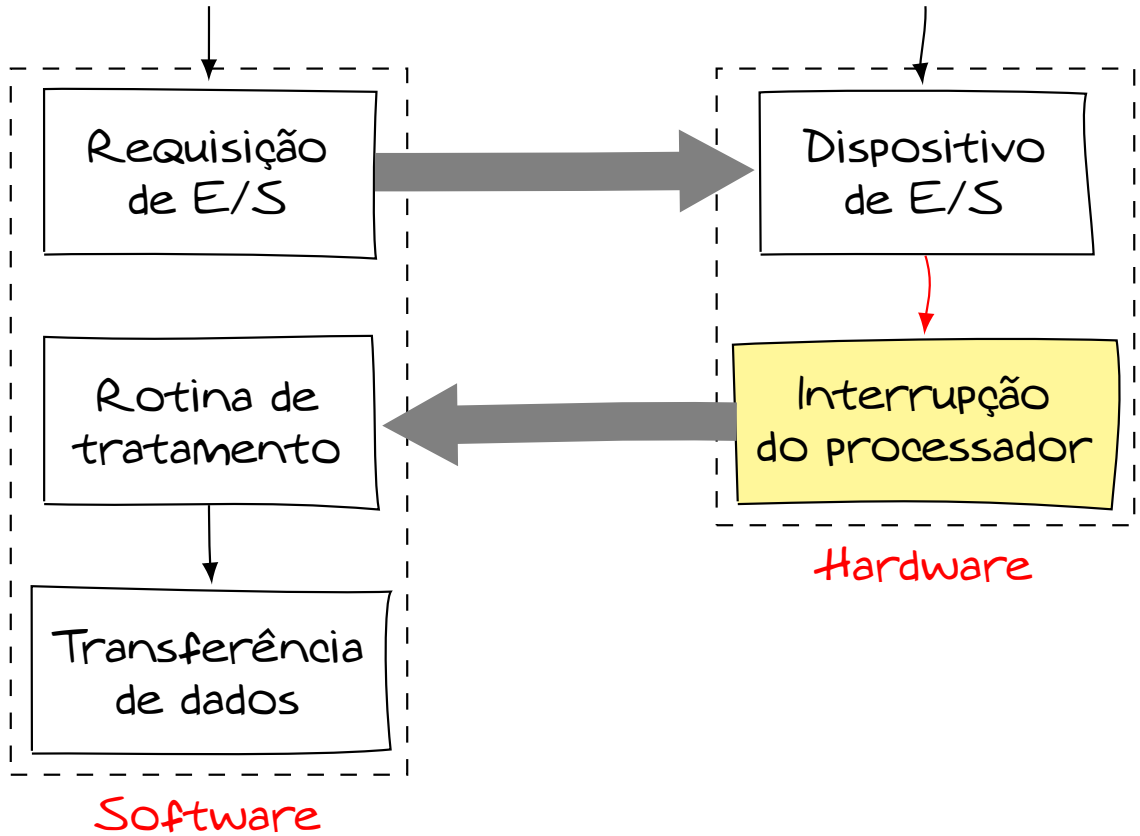
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de software

E/S por interrupção

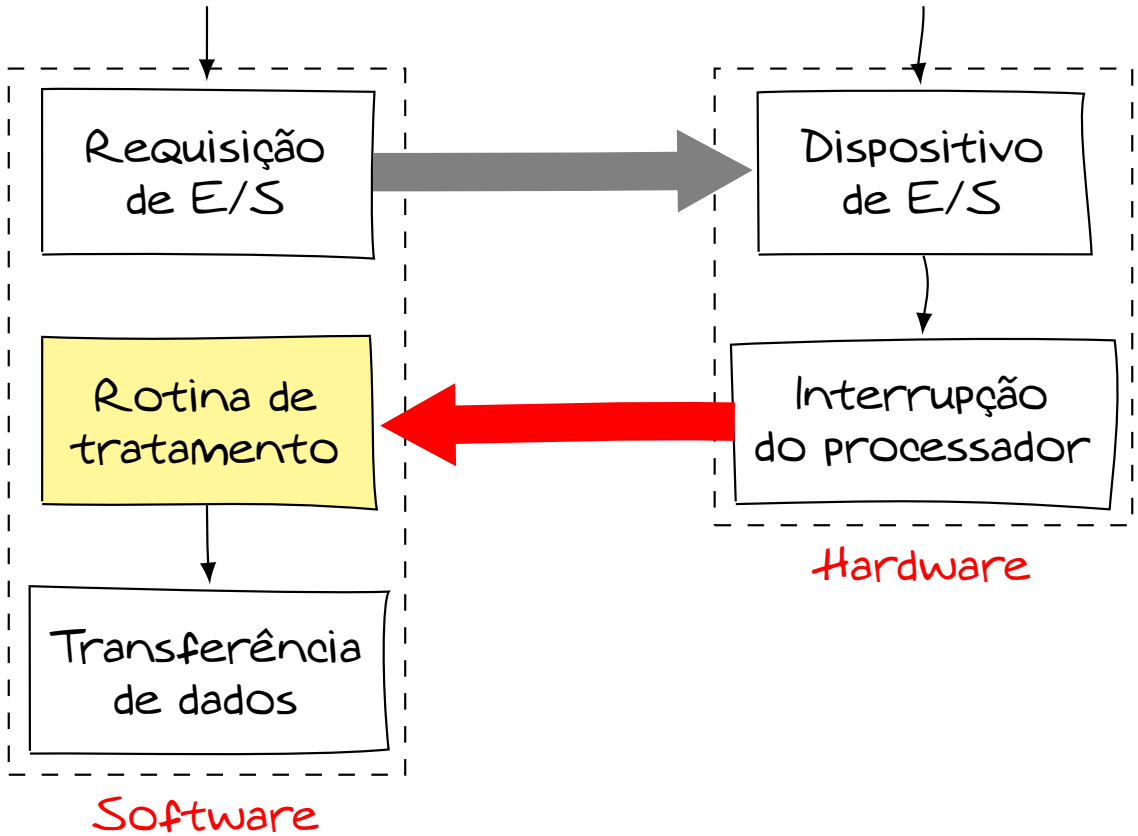
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de software

E/S por interrupção

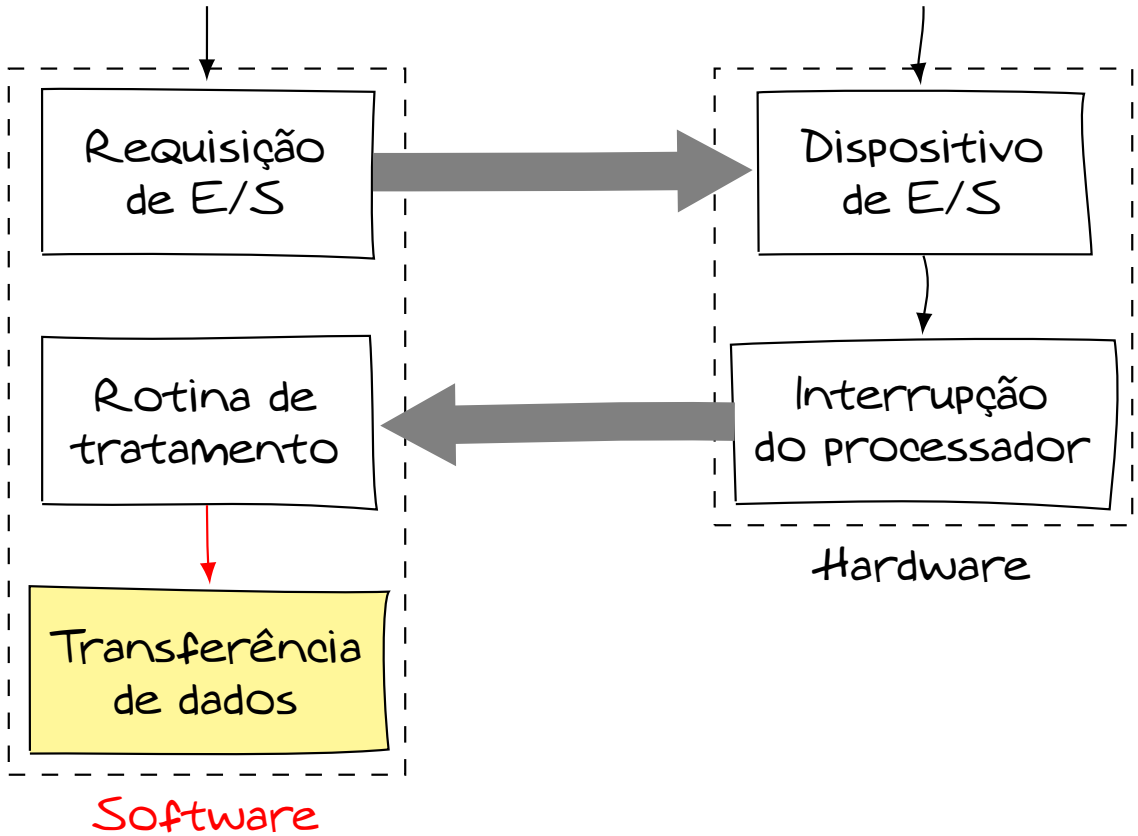
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de software

E/S por interrupção

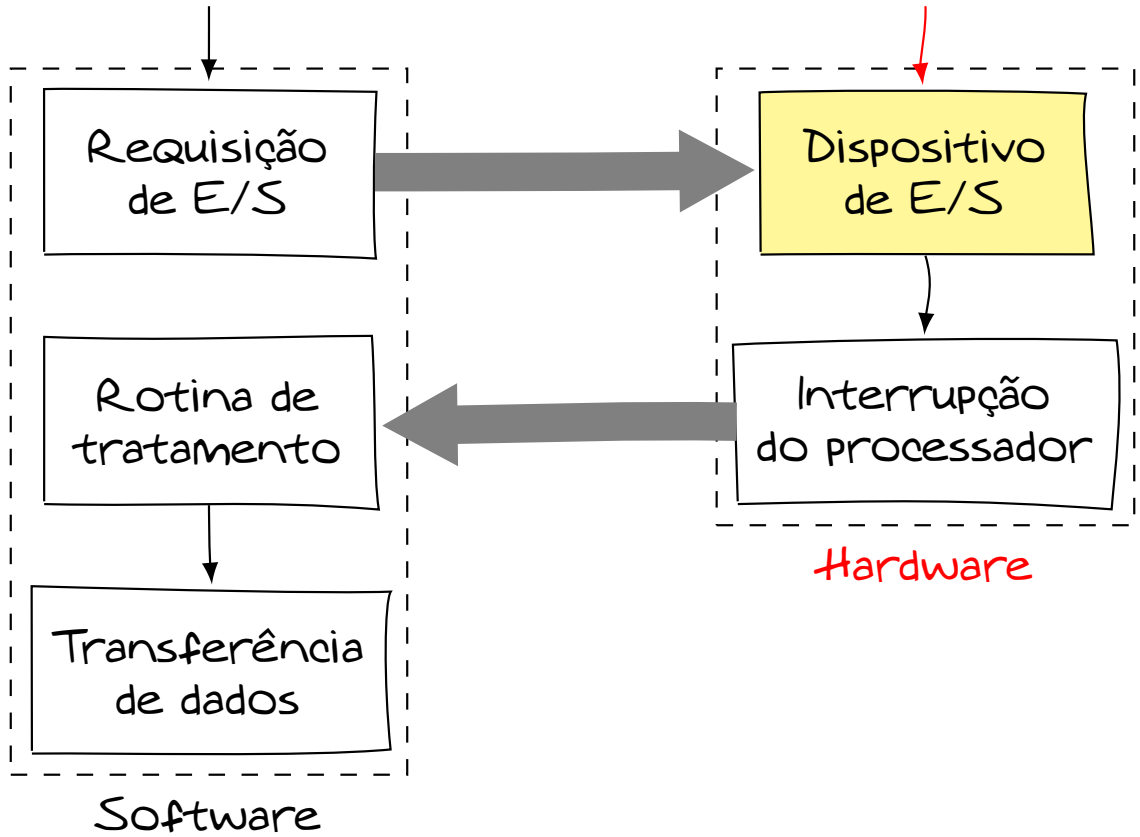
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de software

E/S por interrupção

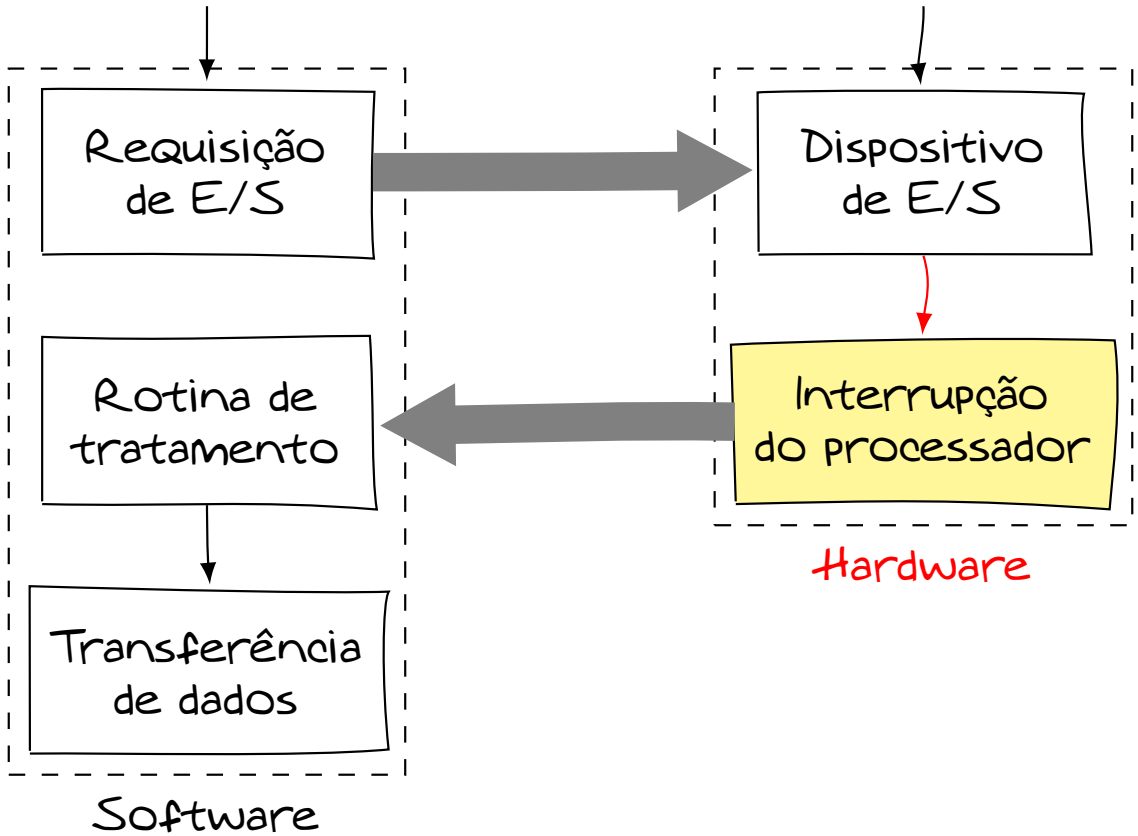
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de hardware

E/S por interrupção

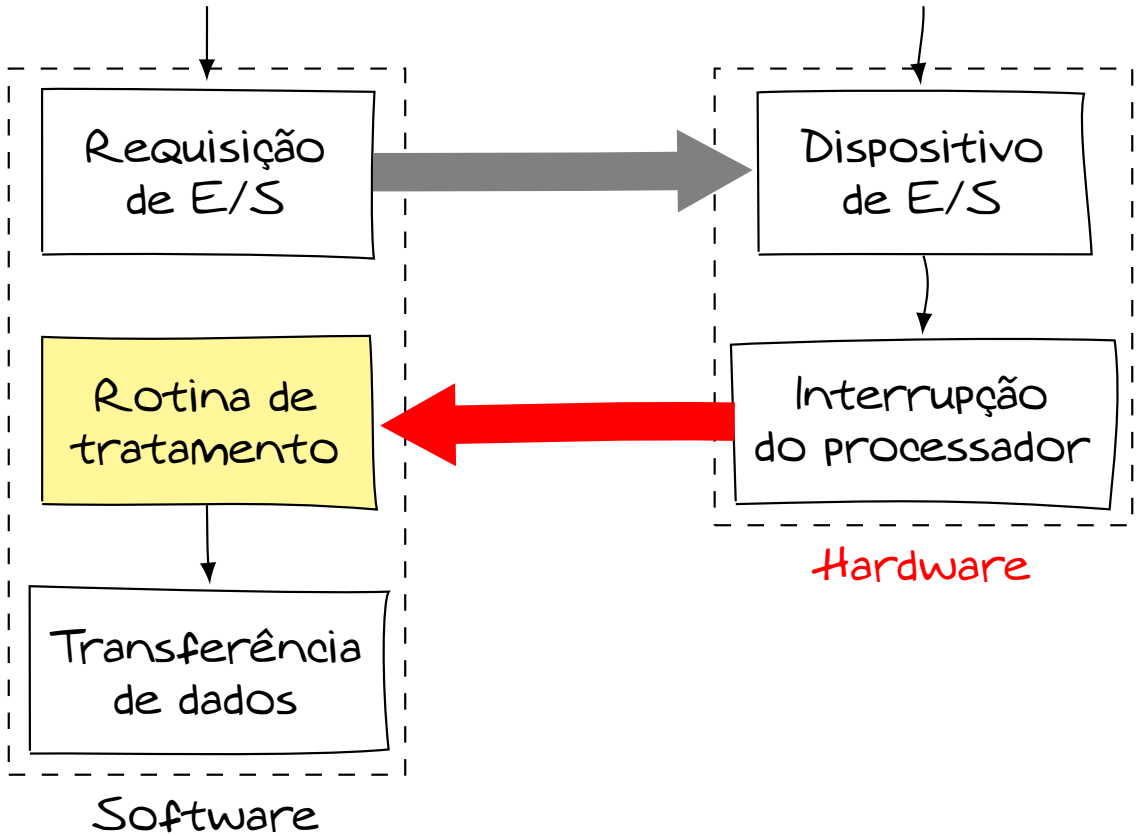
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de hardware

E/S por interrupção

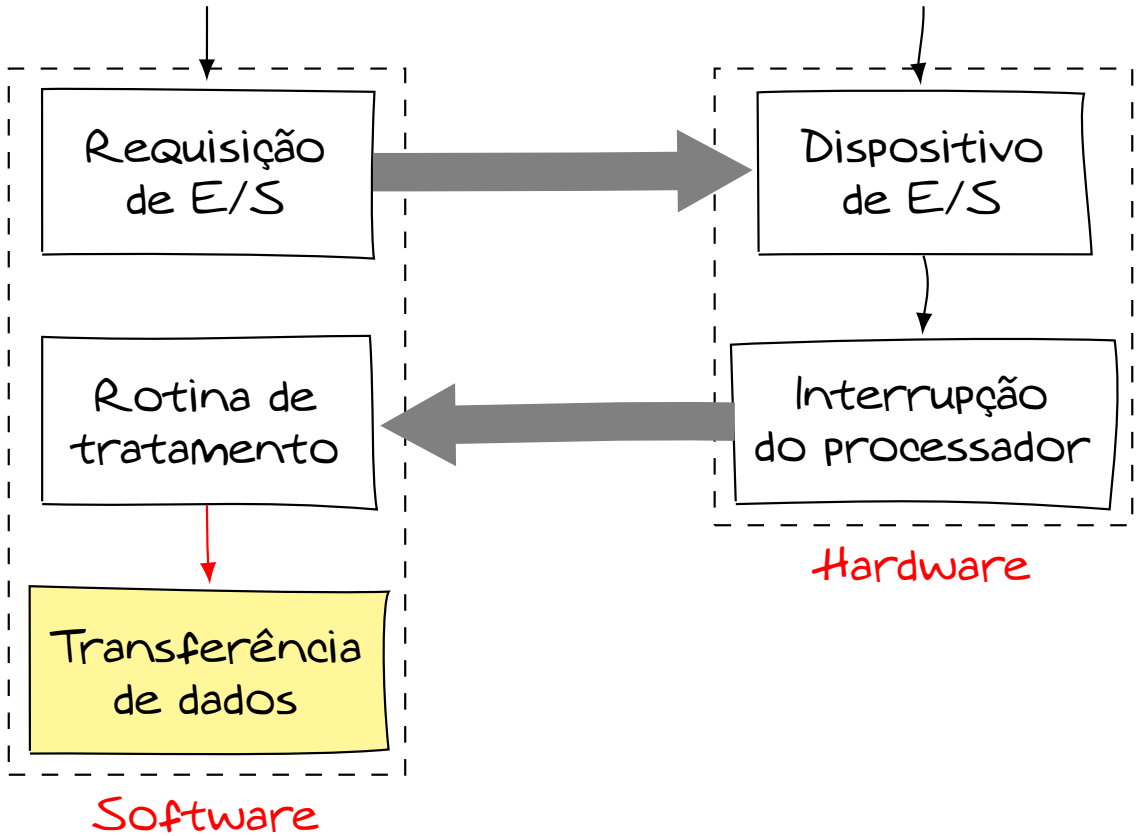
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de hardware

E/S por interrupção

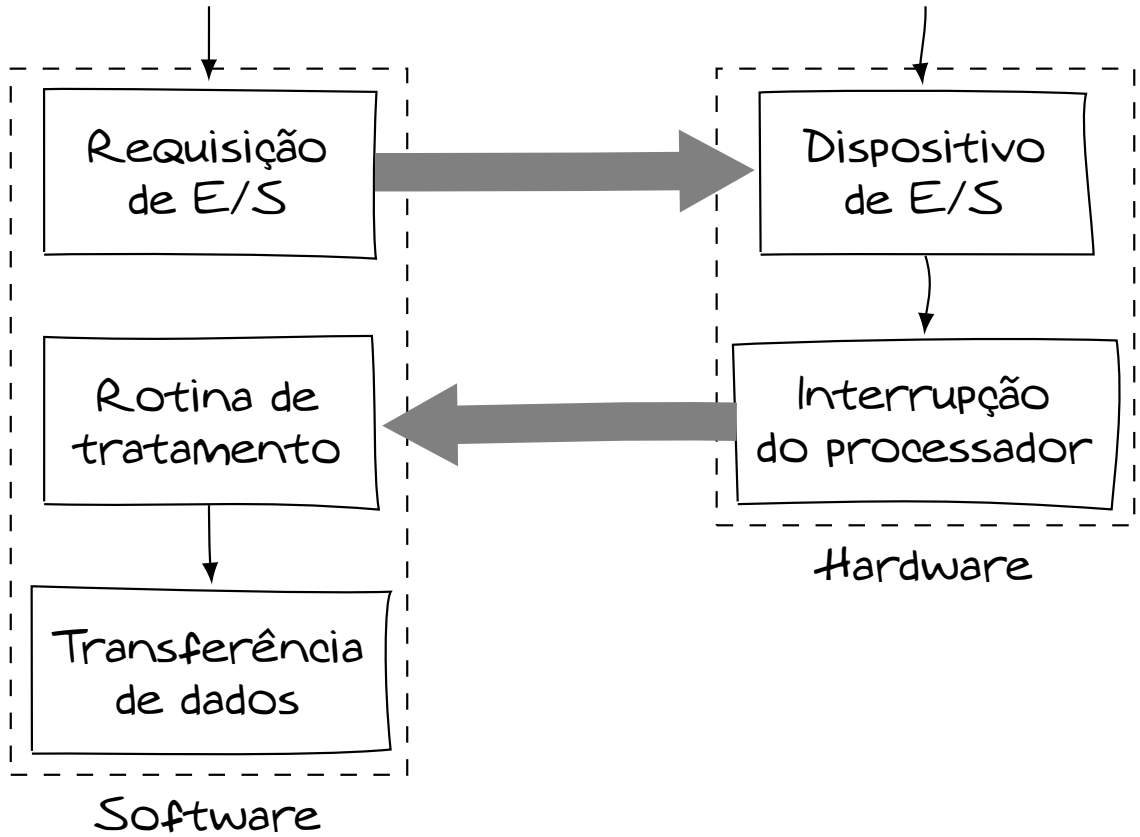
- O fluxo de execução pode ser iniciado pelo software (requisição) ou pelo hardware (interrupção)



Fluxo de hardware

E/S por interrupção

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

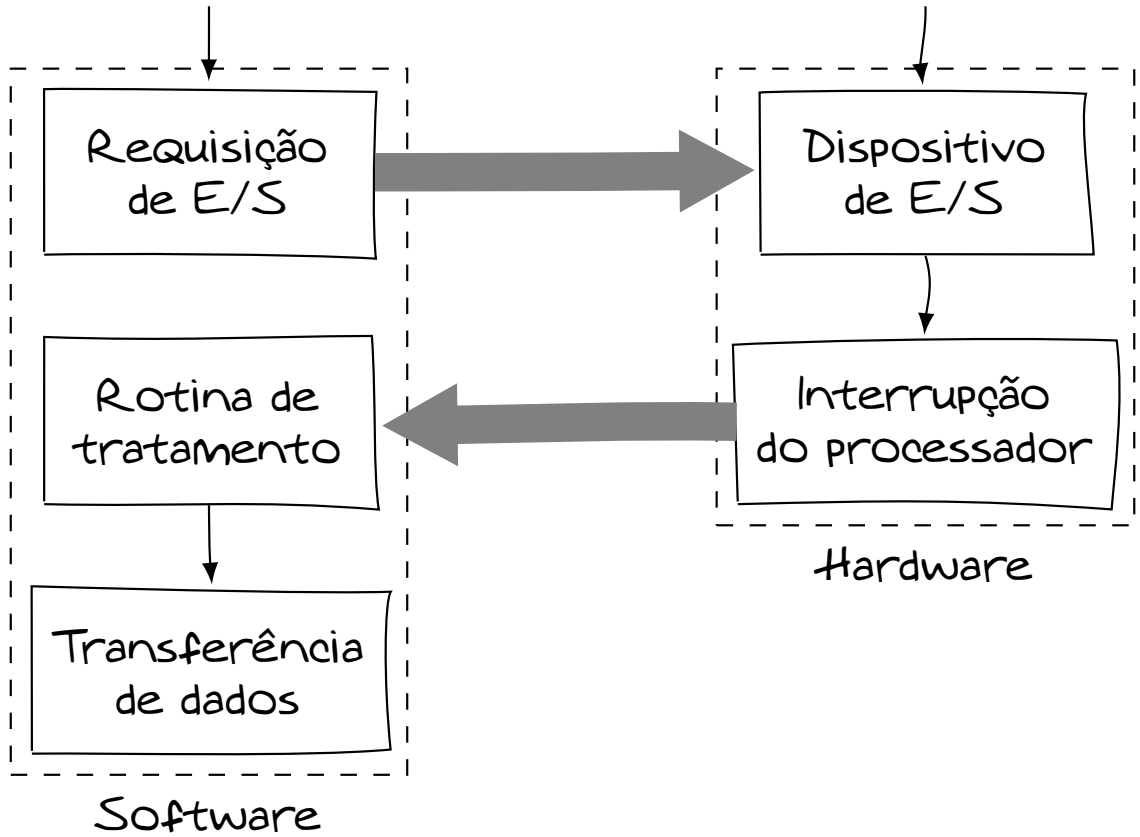


✓ Elimina o processo de espera do processador

✓ 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

- Código em C de contador com interrupção

```
1   // Tipos inteiros de tamanho fixo
2   #include <stdint.h>
3   // Ponteiro para dispositivo
4   volatile uint32_t* const contador = (volatile
        uint32_t*)(0x80808080);
5   // Rotina de tratamento de interrupção
6   void isr_hw1(uint32_t codigo) {
7       // Checagem de código de interrupção
8       if(codigo == 0xE1AC04DA) {
9           // Interrupção de software 100
10          int_sw(100);
11      }
12  }
... ...
```

E/S por interrupção

- Código em C de contador com interrupção

```
 1  // Tipos inteiros de tamanho fixo
 2  #include <stdint.h>
...
13  // Função principal
14  int main() {
15      // Ajustando endereço de ISR de hardware 1
16      int_hw1(&isr_hw1);
17      // Ajustando contador para 100
18      (*contador) = 0x80000064;
19      // Laço infinito
20      while(1);
21      // Retorno sem erros
22      return 0;
23  }
```

E/S por interrupção

- Código de montagem de contador com interrupção

```
1   // Segmento de código
2   .text
3       // Tabela de vetor de interrupção
4       init:
5           bun main
6           int 0
7           int 0
8           int 0
9           bun isr_hw1
10          .align 5
11      // Rotina de tratamento de interrupção
12      isr_hw1:
13          // Checagem de código de interrupção
14          mov r1, cr
15          l32 r2, [codigo]
16          cmp r1, r2
17          bne 1
18          // Interrupção de software 100
19          int 100
20          // Retorno de interrupção
21          reti
...
...
```

E/S por interrupção

- Código de montagem de contador com interrupção

```
1   // Segmento de código
2   .text
... ...
22      // Função principal
23      main:
24          // SP = 32 KiB
25          mov sp, 0x7FFC
26          // Habilitando interrupcao (IE = 1)
27          sbr sr[1]
28          // R1 = valor do contador
29          l32 r1, [valor]
30          // R2 = endereço do contador
31          l32 r2, [watchdog]
32          sra r2, r2, 2
33          // Watchdog = R1
34          s32 [r2], r1
35          // Laço infinito
36          bun -1
37          // Finalizacao de execucao
38          int 0
... ...
```

E/S por interrupção

- Código de montagem de contador com interrupção

```
1   // Segmento de código
2   .text
...
39  // Segmento de dados
40  .data
41      // Código de interrupção
42      codigo:
43          .4byte 0xE1AC04DA
44      // Valor do contador
45      valor:
46          .4byte 0x80000064
47      // Endereço do dispositivo
48      watchdog:
49          .4byte 0x80808080
```

Acesso direto à memória

- Apesar da E/S por interrupção ser mais eficiente, o processador ainda é responsável pela transferência dos dados entre os dispositivos e a memória

Acesso direto à memória

- Apesar da E/S por interrupção ser mais eficiente, o processador ainda é responsável pela transferência dos dados entre os dispositivos e a memória

✗ A velocidade de transferência é limitada pela capacidade de processamento do dispositivo

Acesso direto à memória

- Apesar da E/S por interrupção ser mais eficiente, o processador ainda é responsável pela transferência dos dados entre os dispositivos e a memória

✗ A velocidade de transferência é limitada pela capacidade de processamento 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 dos dados, além da quantidade 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 dos dados, além da quantidade que será transferida

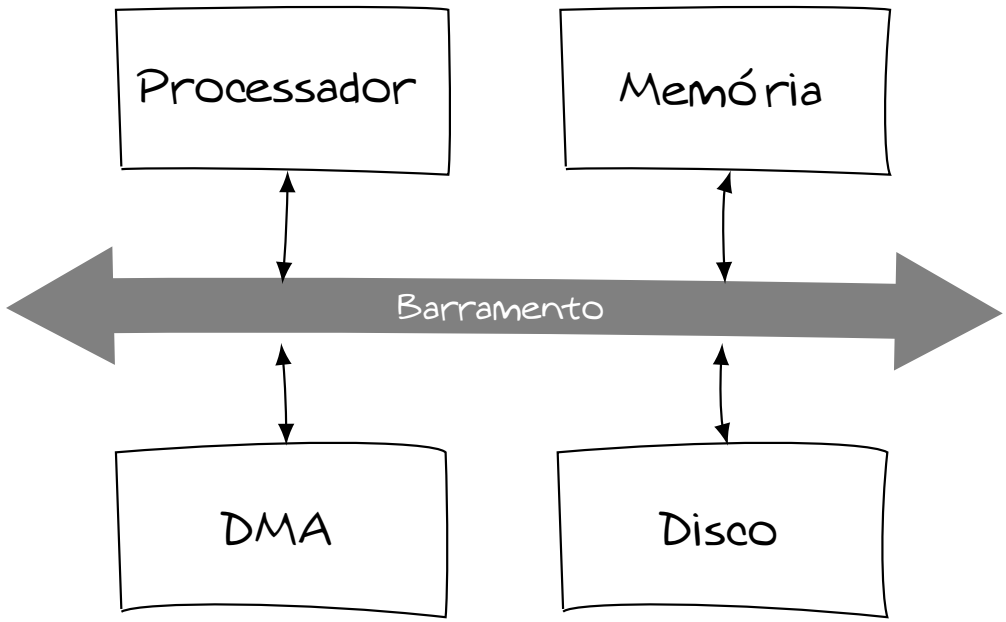
    ✓ É dedicado para controle e transferência de dados entre os dispositivos da plataforma

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 dos dados, além da quantidade que será transferida
    - ✓ É dedicado para controle e transferência de dados entre os dispositivos da plataforma
    - ✓ A transferência não depende do processador

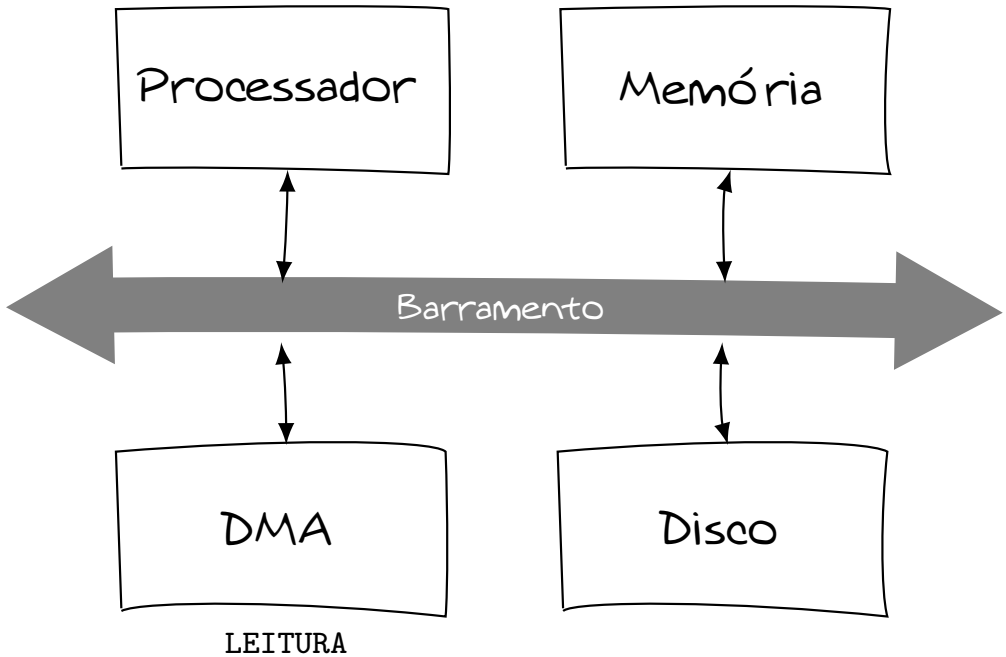
Acesso direto à memória

- Configuração do DMA



Acesso direto à memória

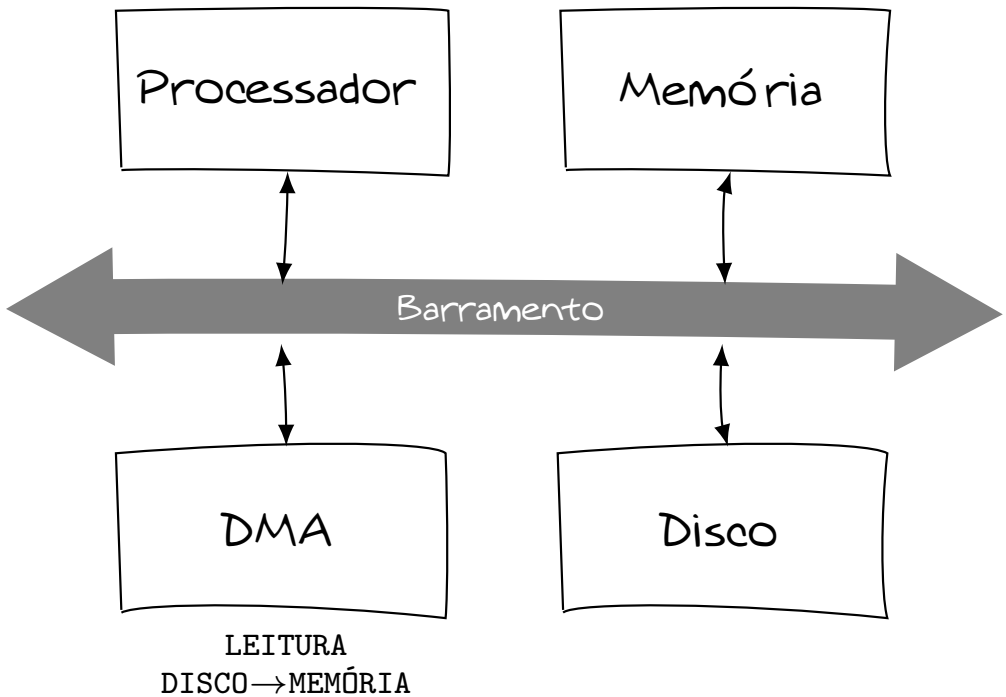
- Configuração do DMA



Definição da operação (escrita ou leitura)

Acesso direto à memória

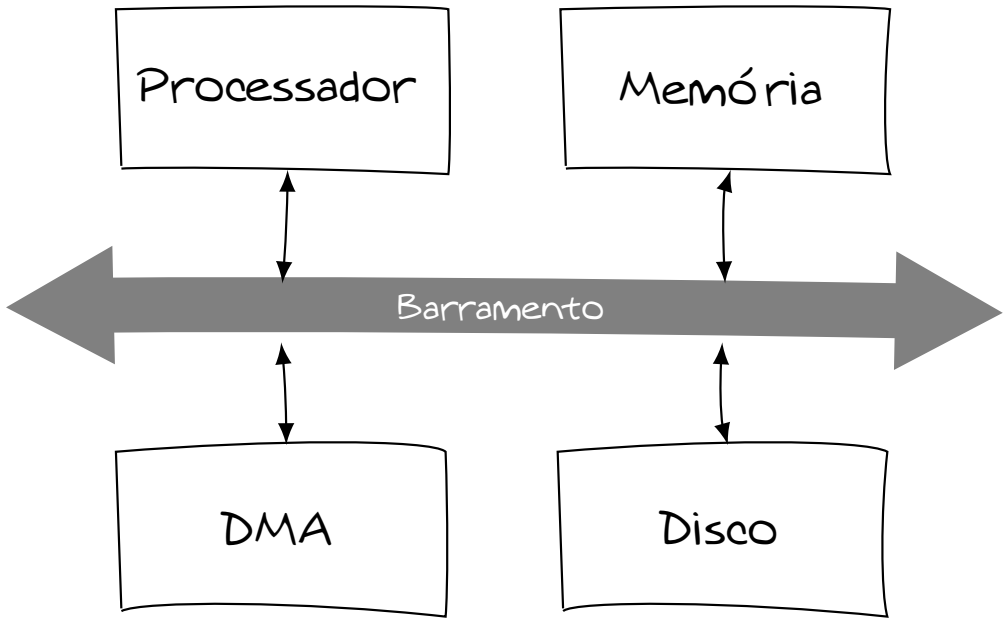
- Configuração do DMA



Endereços de origem e de destino dos dispositivos

Acesso direto à memória

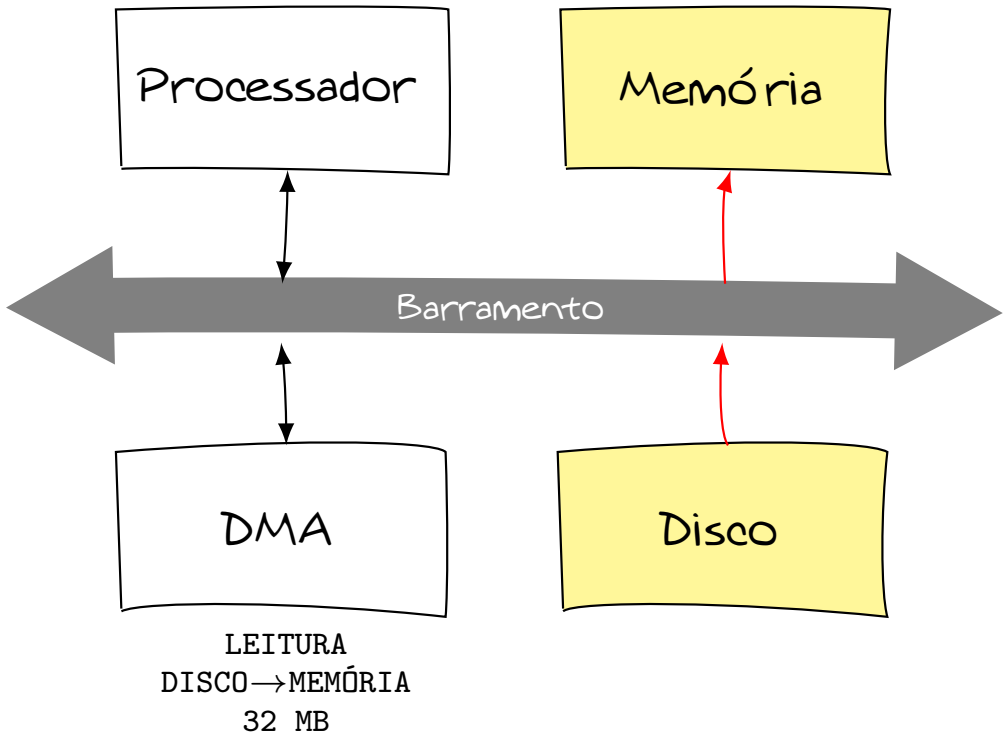
- Configuração do DMA



Quantidades de dados que serão transferidos

Acesso direto à memória

- Configuração do DMA



Transferência dos dados

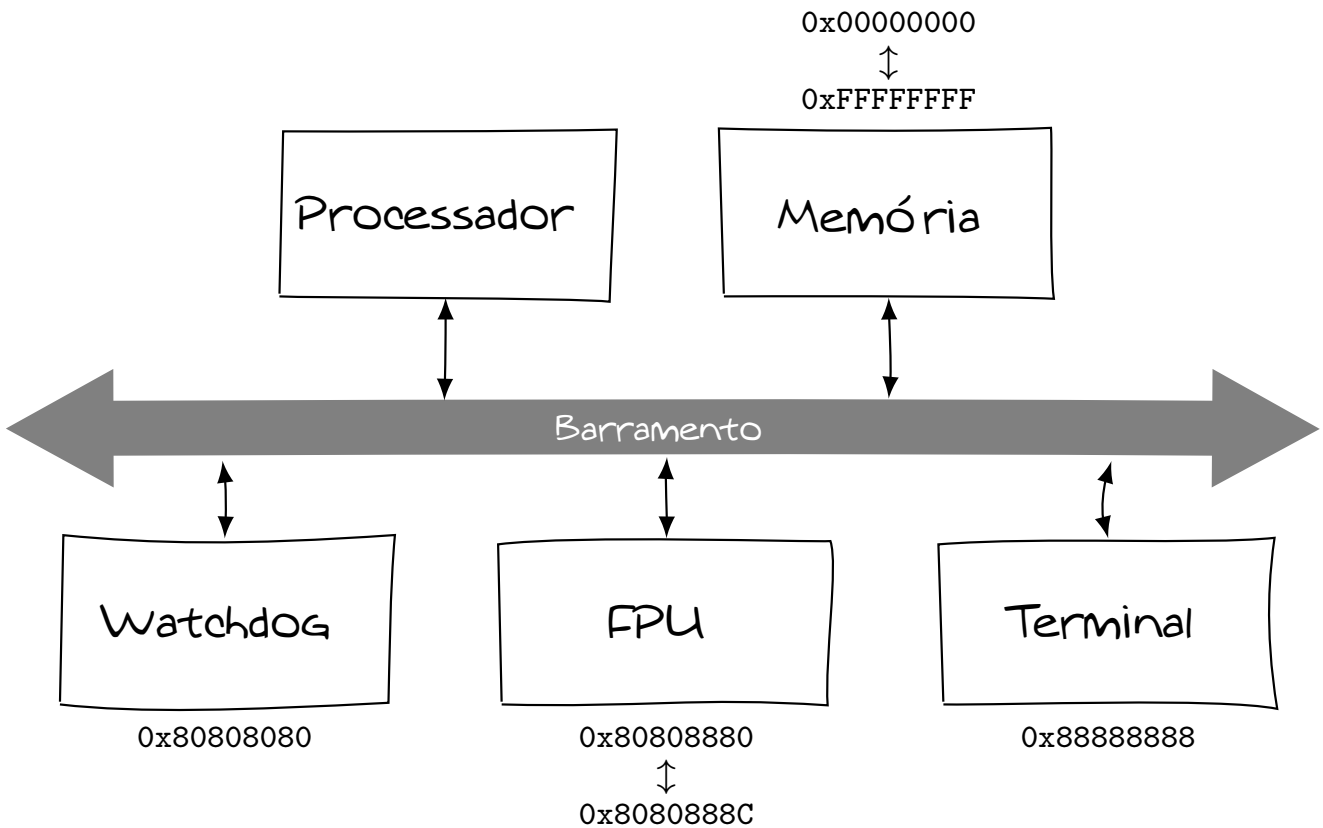
Acesso direto à memória

- 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
  - Temporizador (Watchdog)
  - Unidade de Ponto Flutuante (FPU)
  - Interface serial de texto (Terminal)



Exercício

- Registrador do temporizador (Watchdog)
  - Interrupção de hardware 1 com código 0*xE*1*AC*04*DA*

EN	Operação
0	Desativado
1	Ativado

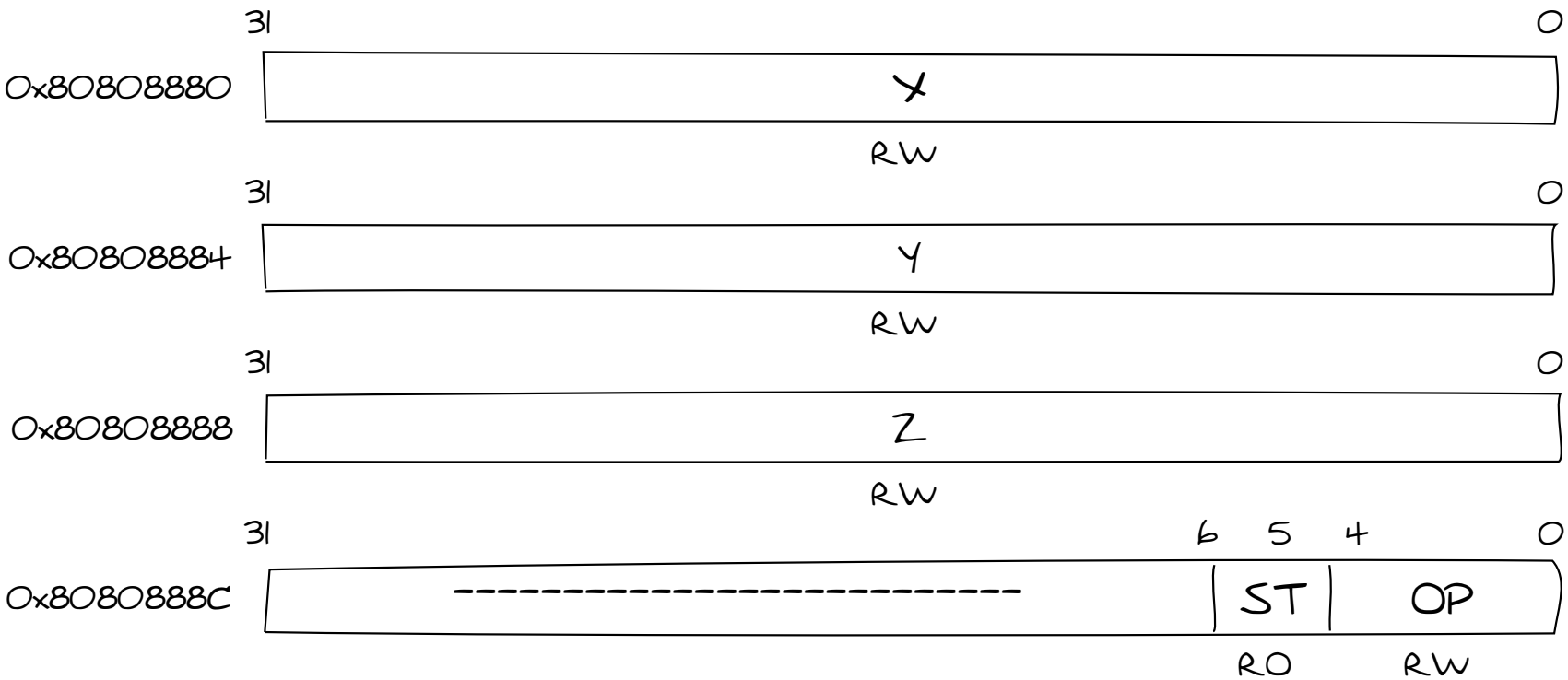
COUNTER	Operação
0000..0000	Tempo esgotado
0000..0001	*COUNTER* − −
⋮	⋮
1111...1110	*COUNTER* − −
1111...1111	*COUNTER* − −

Exercício

- Especificação do temporizador (Watchdog)
  - Caso o temporizador esteja habilitado ($EN = 1$), em cada ciclo de instrução o contador é decrementado até atingir o valor 0 (tempo esgotado)
  - Quando o tempo for esgotado, a operação do temporizador é desabilitada ($EN = 0$)
  - A interrupção fica pendente quando a interrupção está desabilitada ($IE = 0$), sendo ser sobrescrita pela última interrupção do temporizador

Exercício

- Registradores da Unidade de Ponto Flutuante (FPU)
  - Interrupções de hardware 2 (erro em operação), 3 (operações com tempo variável) e 4 (operações de tempo constante) com código 0x01*EEE*754



Exercício

- Registradores da Unidade de Ponto Flutuante (FPU)
  - Descrição dos campos de controle

ST	Status
0	Pronto
1	Erro

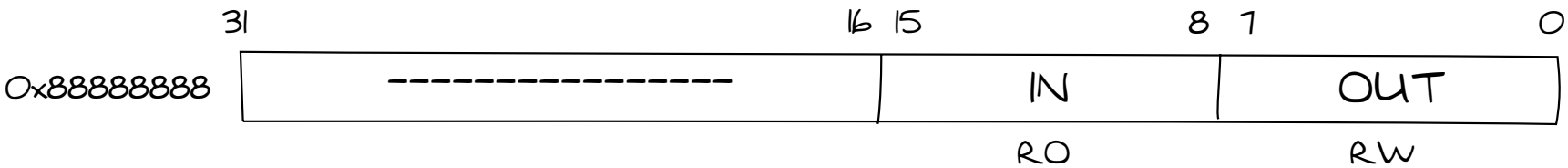
OP	Operação
00000	Sem operação
00001	Adição $Z = X + Y$
00010	Subtração $Z = X - Y$
00011	Multiplicação $Z = X \times Y$
00100	Divisão $Z = X \div Y$
00101	Atribuição $X = Z$
00110	Atribuição $Y = Z$
00111	Teto $\lceil Z \rceil$
01000	Piso $\lfloor Z \rfloor$
01001	Arredondamento $\|\|Z\|\|$

Exercício

- Especificação da Unidade de Ponto Flutuante (FPU)
  - O campo de status é somente leitura (RO) e seu valor só é alterado após a realização de alguma operação, sendo que somente a divisão por zero e o código de operação inválido ativam este campo
  - Os parâmetros X e Y são codificados como números inteiros pelo processador, mas através das operações de atribuição são codificados no formato IEEE 754
    - A quantidade de ciclos de cada operação é calculada pela expressão $|exp(X) - exp(Y)| + 1$ que depende da diferença dos valores de expoentes entre os parâmetros X e Y
    - Para as operações de atribuição, teto, piso e arredondamento a quantidade de ciclos é 1, ou seja, é constante independente das entradas utilizadas
  - As operações aritméticas de adição, subtração, multiplicação e divisão são armazenadas em Z em formato IEEE 754, mas as funções de teto, piso e arredondamento retornam um valor inteiro

Exercício

- Registrador da interface serial de texto (Terminal)
  - Não suporta interrupção
  - Escrita e leitura sequencial de bytes



Exercício

- Especificação da interface serial de texto (Terminal)
  - A sequência de bytes lidas em *IN* é dependente da aplicação, com suas regras sendo descritas pelo formato de entrada utilizado e pelo tamanho dos dados requisitados em cada operação
  - Após cada operação de escrita, o campo *OUT* recebe o byte que fica armazenado no registrador, permitindo a realização de operação de leitura
  - Para organizar a impressão do texto em codificação ASCII, somente no final da simulação são exibidos todos os caracteres escritos durante a execução

Exercício

- Ordenação crescente de vetor de números
  - O vetor de entrada possui 100 números aleatórios com 32 bits de tamanho sem sinal que deverá ser lido sequencialmente byte por byte da interface serial de texto (*IN*) na codificação *big-endian*

```
[START OF SIMULATION]
.
.
.
[TERMINAL]
Input numbers:
99 98 97 ... 2 1 0
Sorted numbers:
0 1 2 ... 97 98 99
[END OF SIMULATION]
```