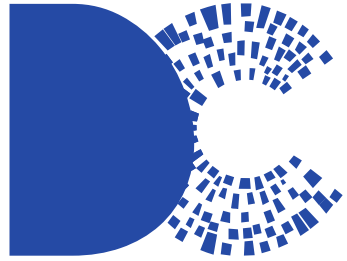
UNIVERSIDADE FEDERAL DE SERGIPE



DEPARTAMENTO DE COMPUTAÇÃO

Gerenciamento de interrupção

Sistemas Embarcados

Bruno Prado

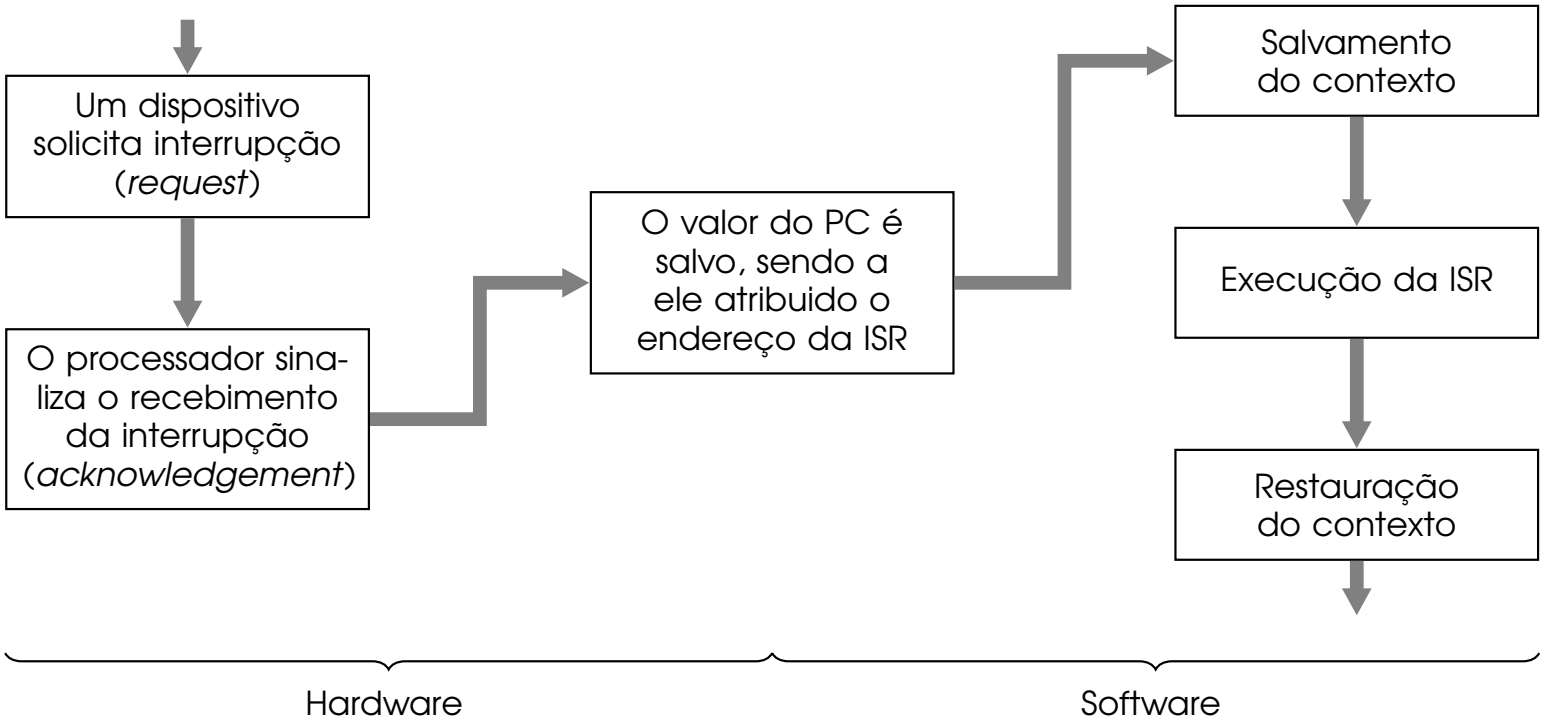
Departamento de Computação / UFS

Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)

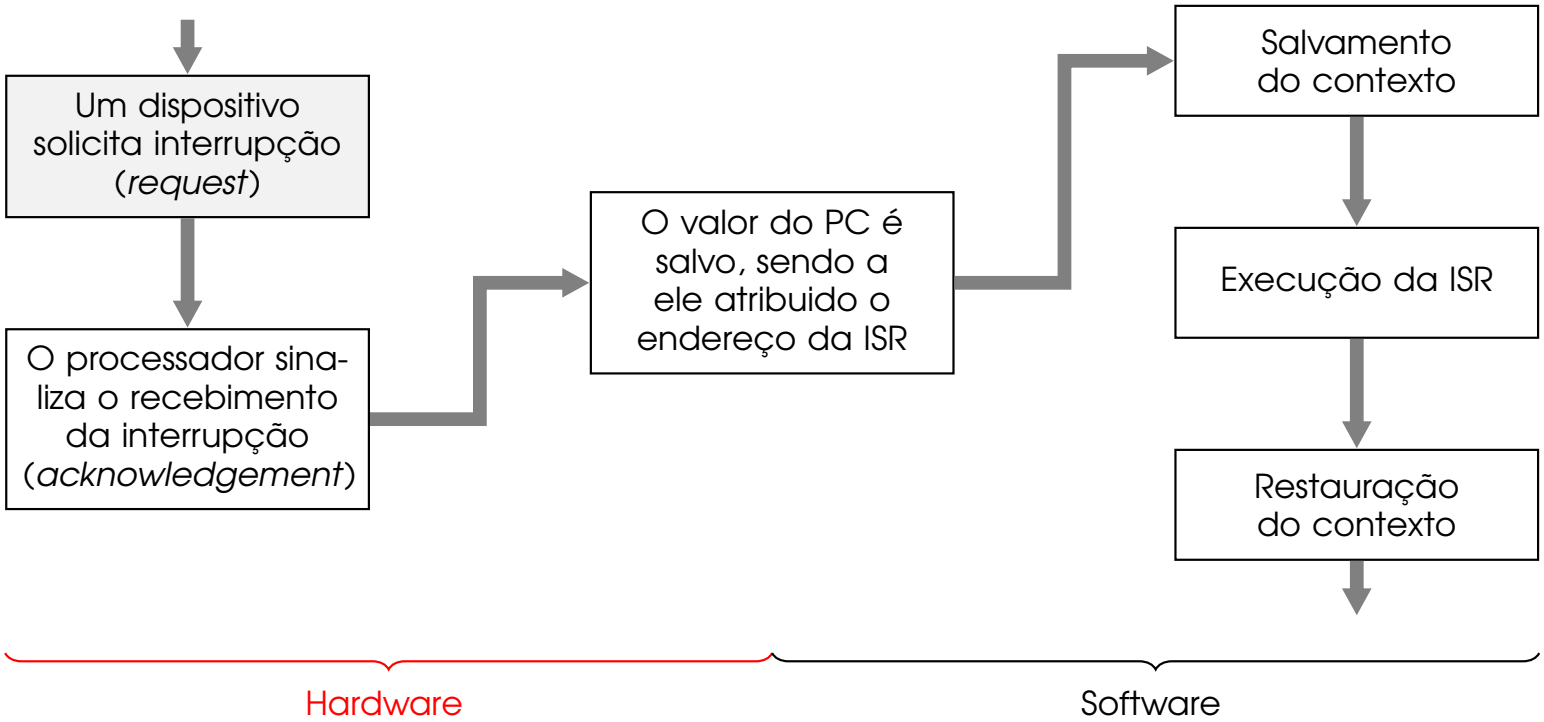
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



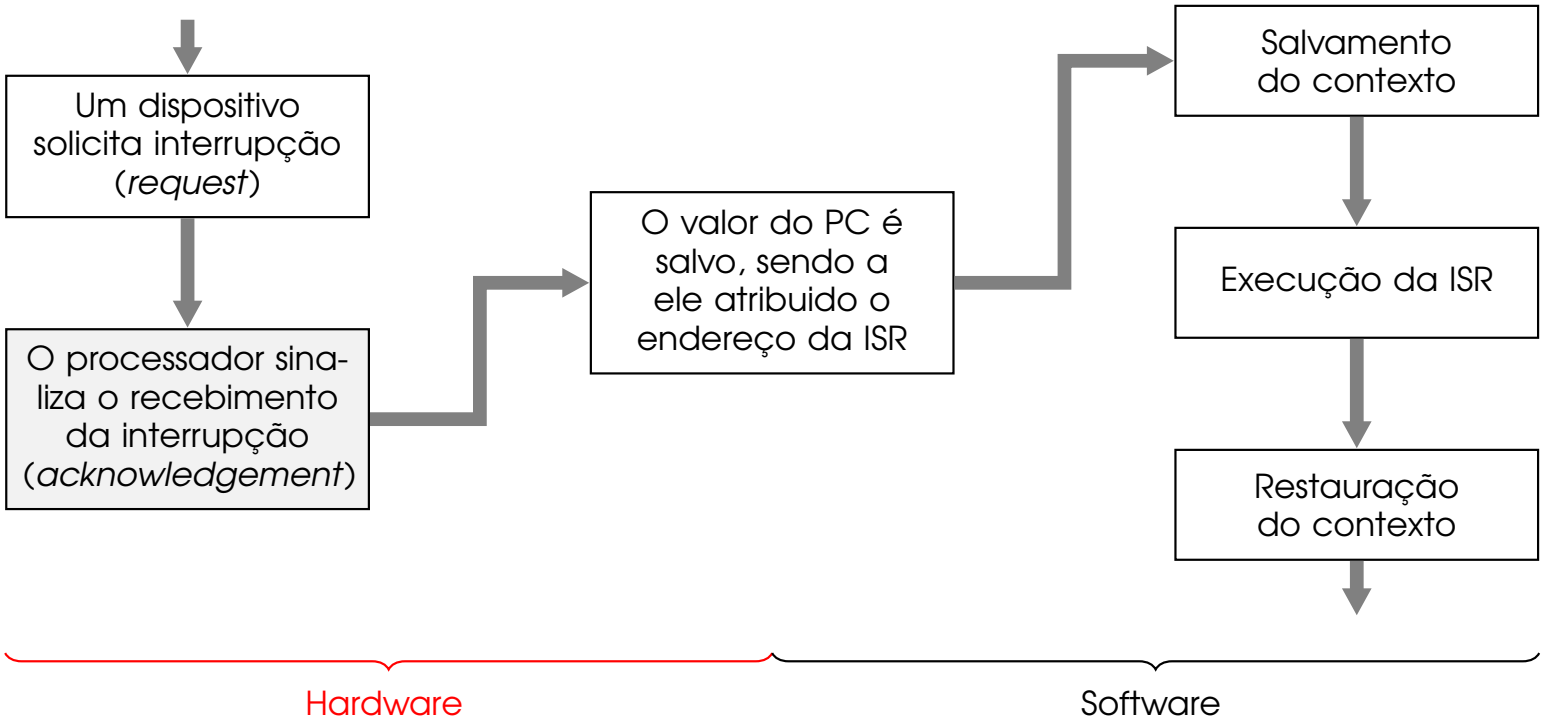
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



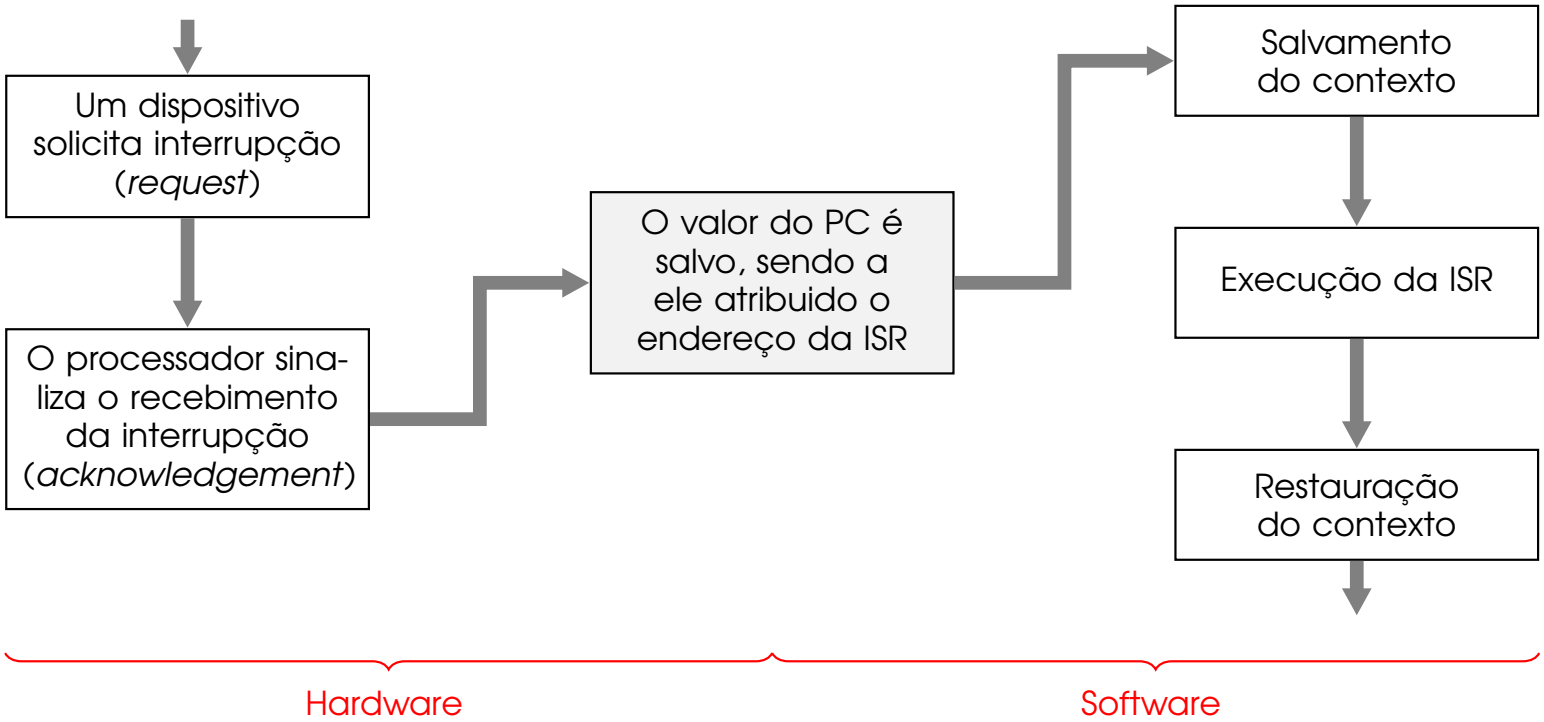
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



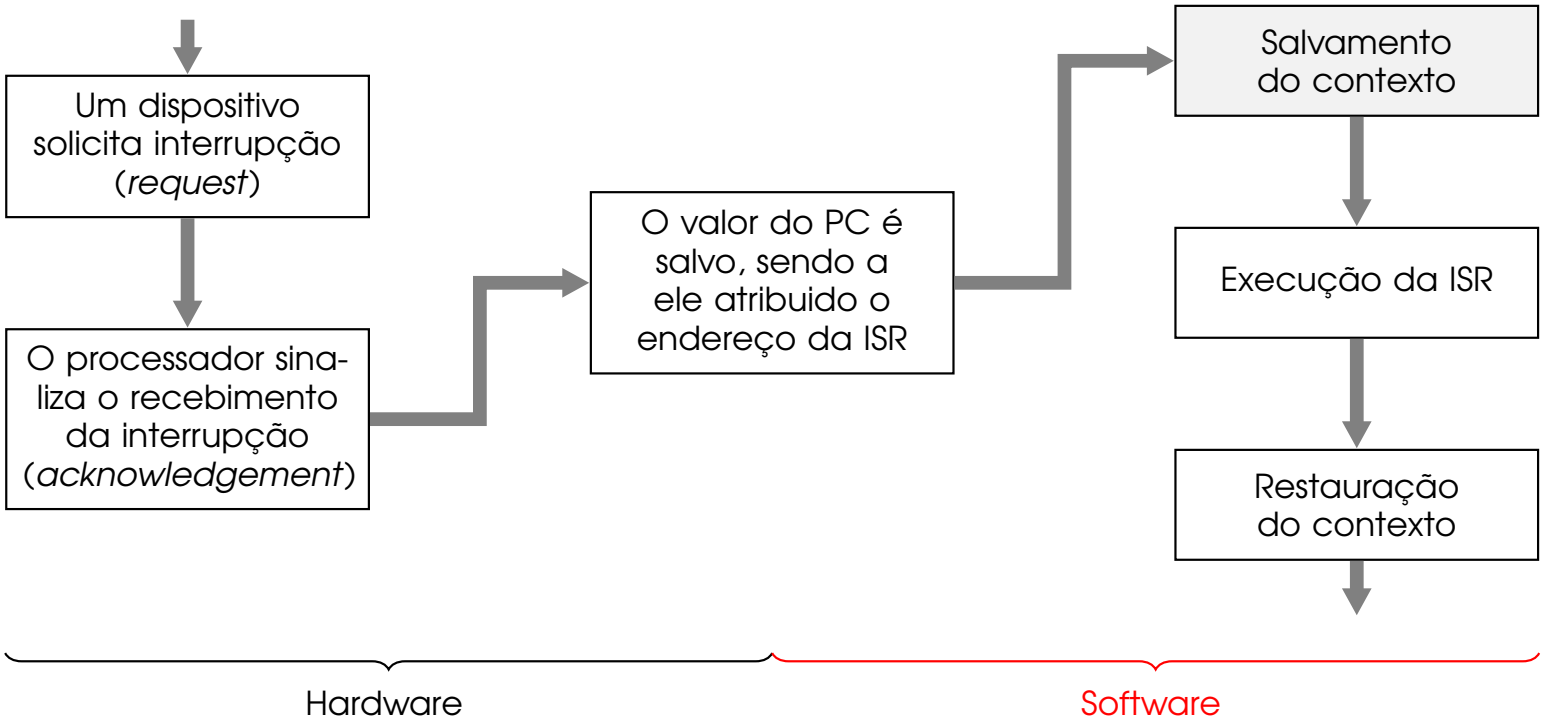
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



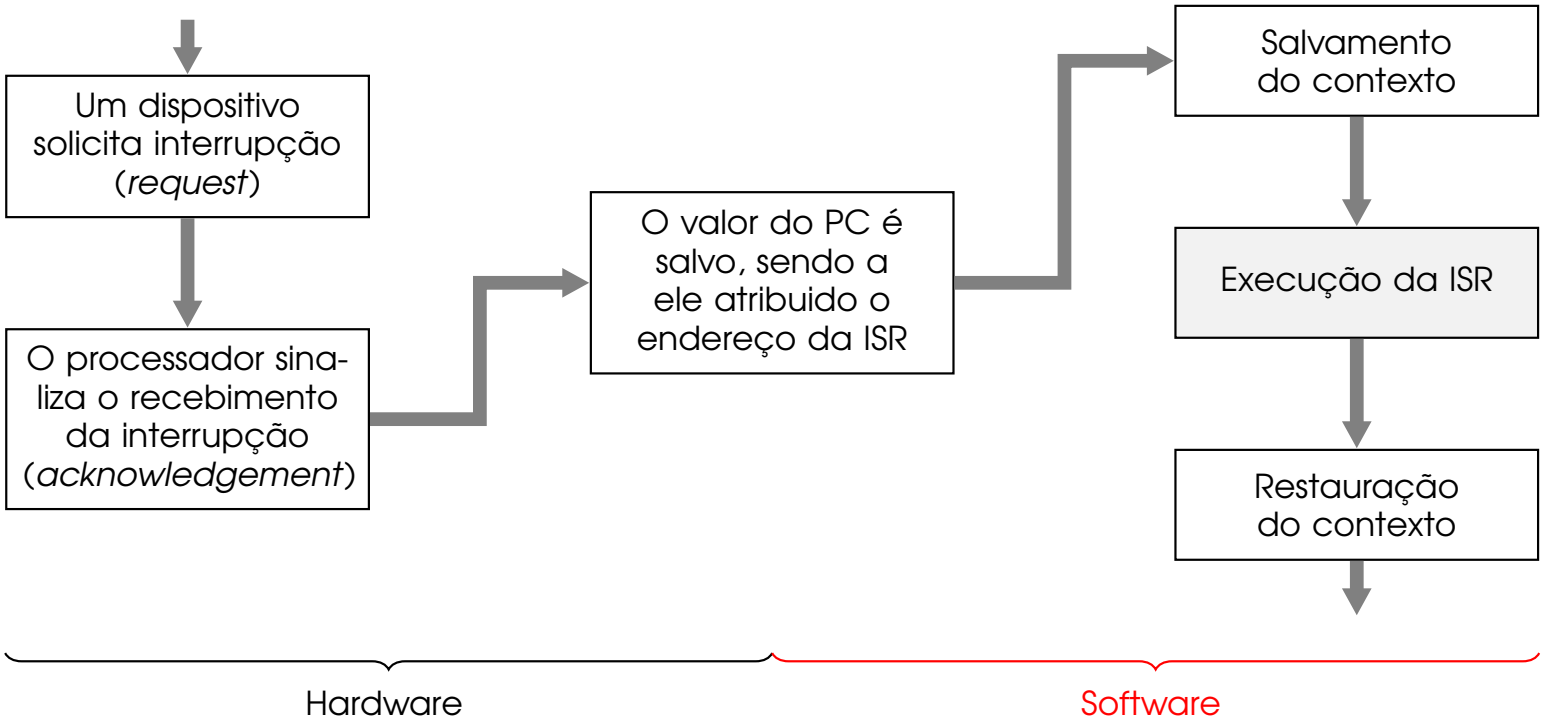
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



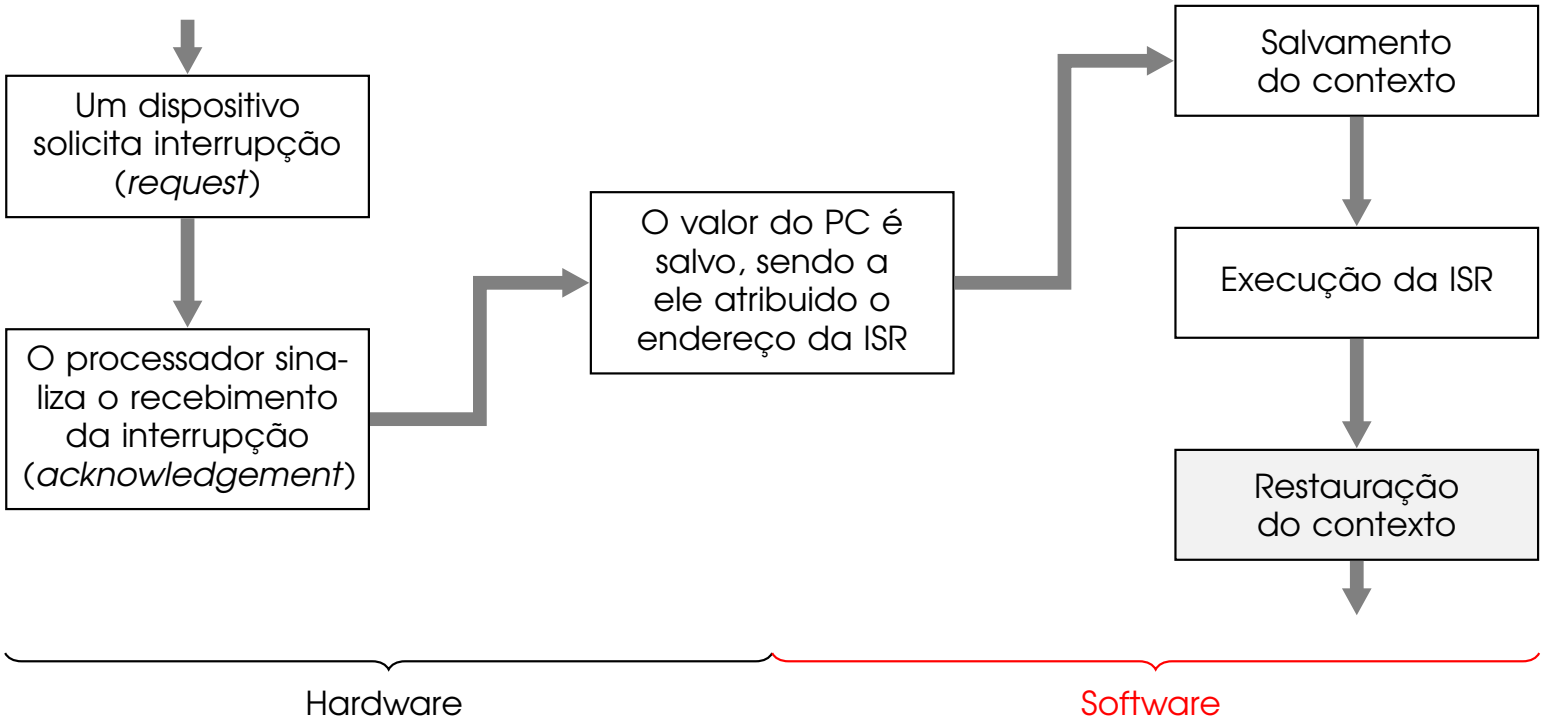
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



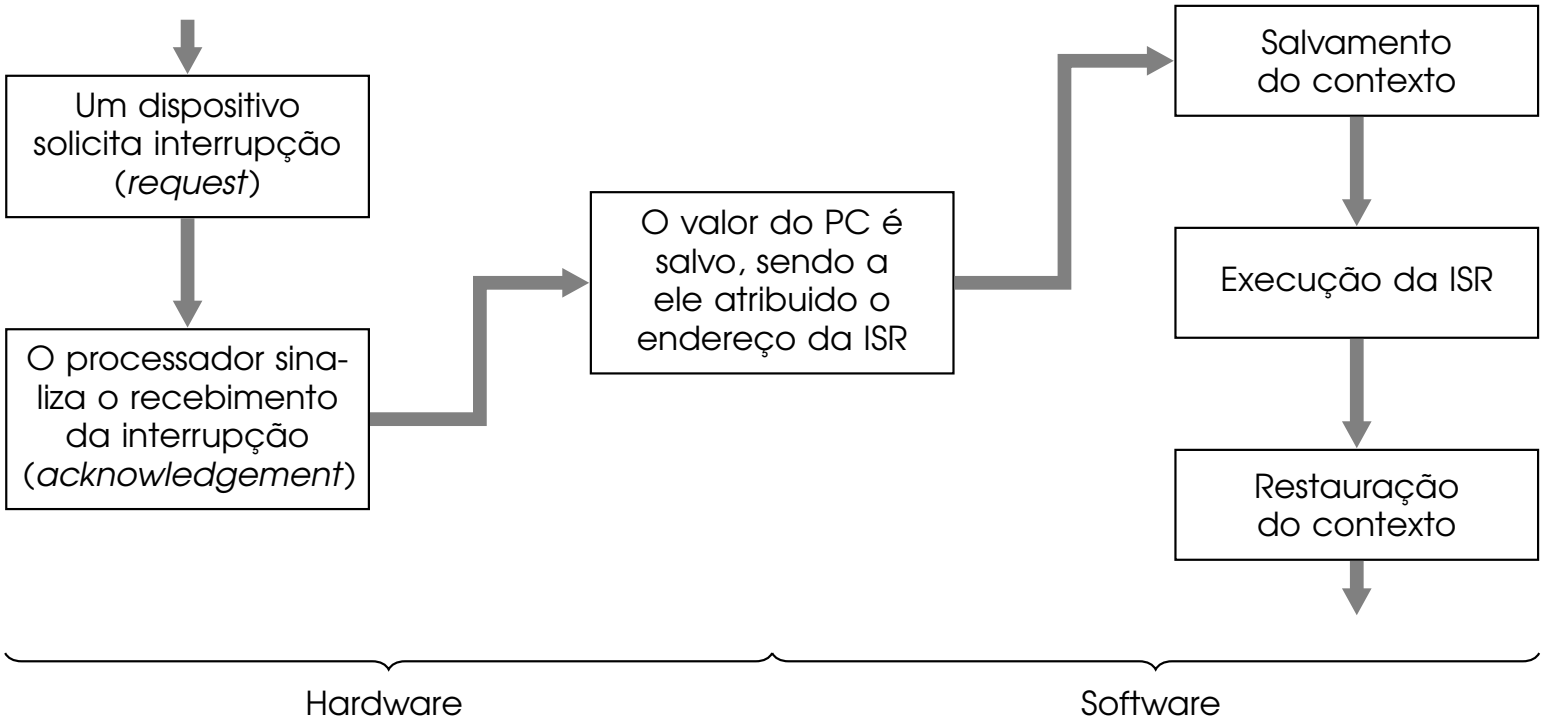
Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



Introdução

- O que é uma interrupção?
  - É um evento criado por um dispositivo de hardware (interrupção) ou gerado pela execução do software (exceção) que requisita a utilização de rotinas de tratamento de interrupção (ISR)



Introdução

- Por que utilizar interrupção é necessário?

Introdução

- Por que utilizar interrupção é necessário?
  - Evitar a espera do processador e sem reduzir a eficiência de execução das operações

Introdução

- Por que utilizar interrupção é necessário?
  - Evitar a espera do processador e sem reduzir a eficiência de execução das operações
  - Permitir o funcionamento assíncrono que evita a utilização de *polling* em dispositivos de E/S lentos

Introdução

- Por que utilizar interrupção é necessário?
  - Evitar a espera do processador e sem reduzir a eficiência de execução das operações
  - Permitir o funcionamento assíncrono que evita a utilização de *polling* em dispositivos de E/S lentos

```
1  // Biblioteca de E/S padrão
2  #include <stdio.h>
3  // Função principal
4  int main() {
5      // Variável de nome
6      char nome[50] = { 0 };
7      // Mensagem de pergunta
8      printf("Qual é o seu nome?\n");
9      // Leitura do teclado
10     scanf("%s", nome);
11     // Mensagem de resposta
12     printf("Olá %s!\n", nome);
13     // Retorno sem erros
14     return 0;
15 }
```

Introdução

- Por que utilizar interrupção é necessário?
  - Evitar a espera do processador e sem reduzir a eficiência de execução das operações
  - Permitir o funcionamento assíncrono que evita a utilização de *polling* em dispositivos de E/S lentos

```
1  // Biblioteca de E/S padrão
2  #include <stdio.h>
3  // Função principal
4  int main() {
5      // Variável de nome
6      char nome[50] = { 0 };
7      // Mensagem de pergunta
8      printf("Qual é o seu nome?\n");
9      // Leitura do teclado
10     scanf("%s", nome);
11     // Mensagem de resposta
12     printf("Olá %s!\n", nome);
13     // Retorno sem erros
14     return 0;
15 }
```

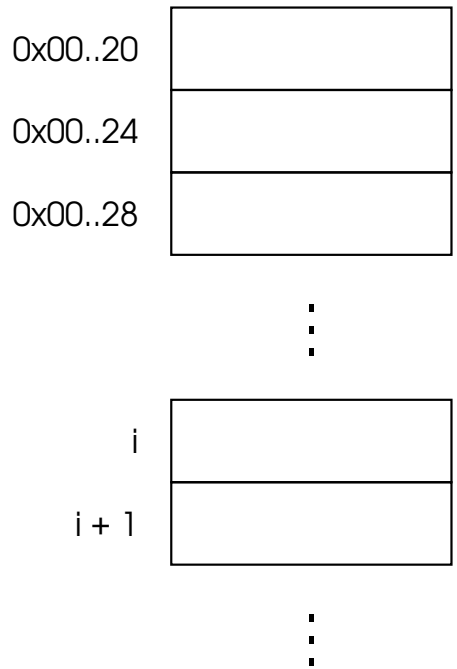
Introdução

- Comparativo entre tipos de interrupções

Hardware	Software
Assíncrona*	Síncrona*
Mascarável*	Não mascarável*
E/S em periféricos (envio ou recebimento de dados)	Execução de operações (instrução inválida ou divisão por zero)

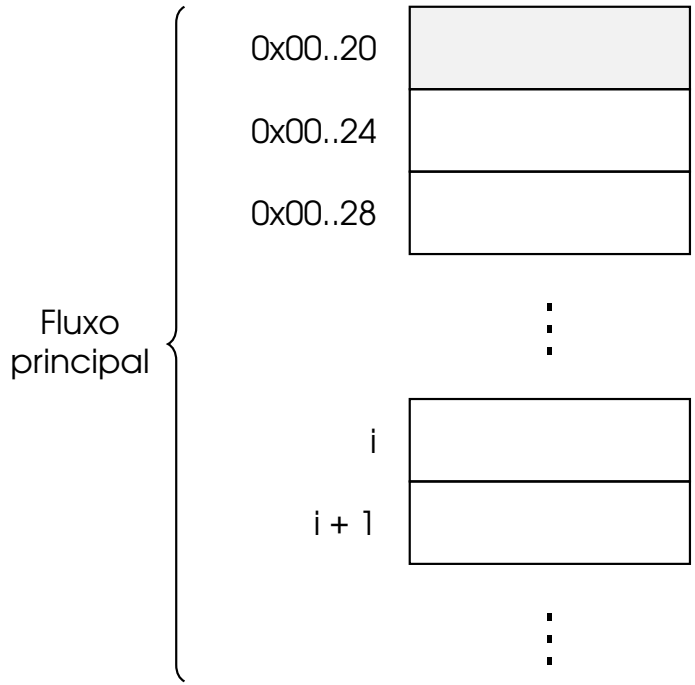
Interrupção

- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Interrupção

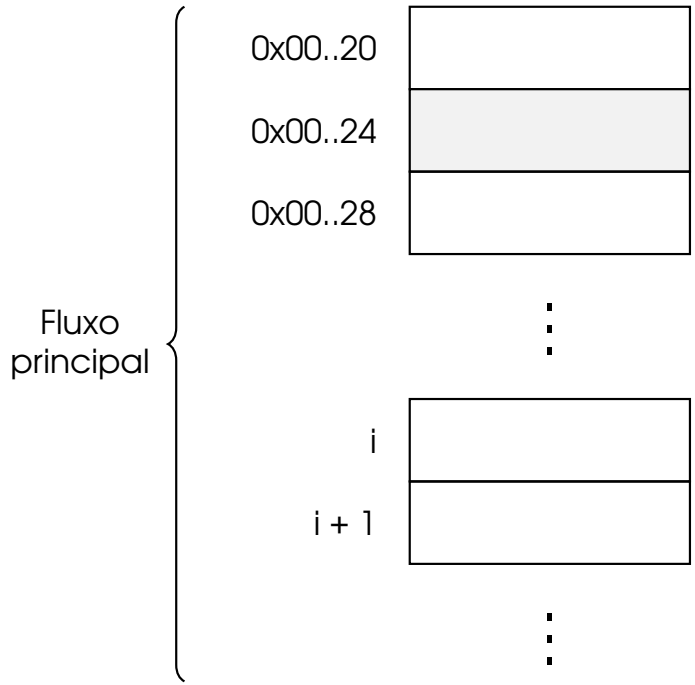
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo principal da aplicação

Interrupção

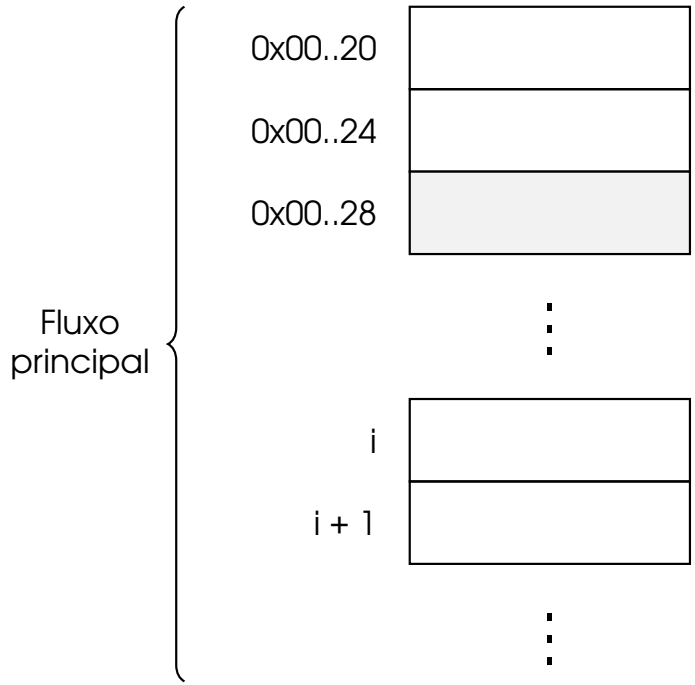
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo principal da aplicação

Interrupção

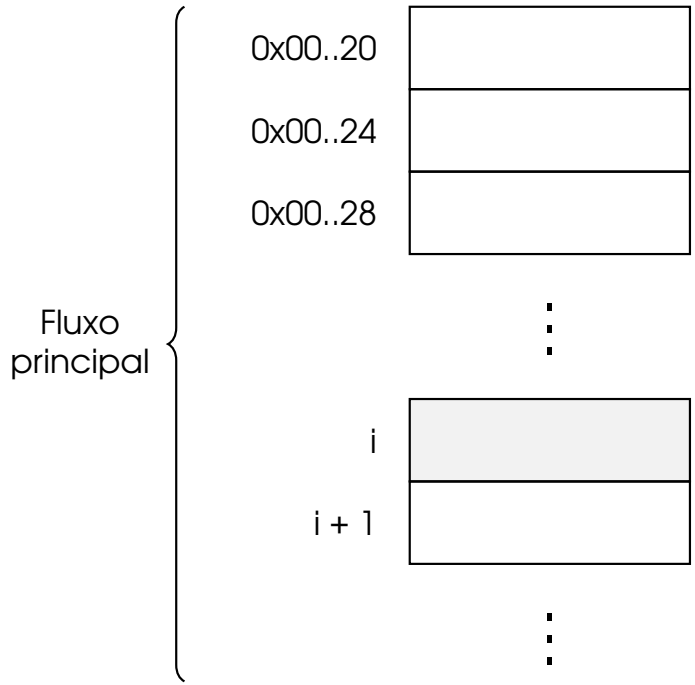
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo principal da aplicação

Interrupção

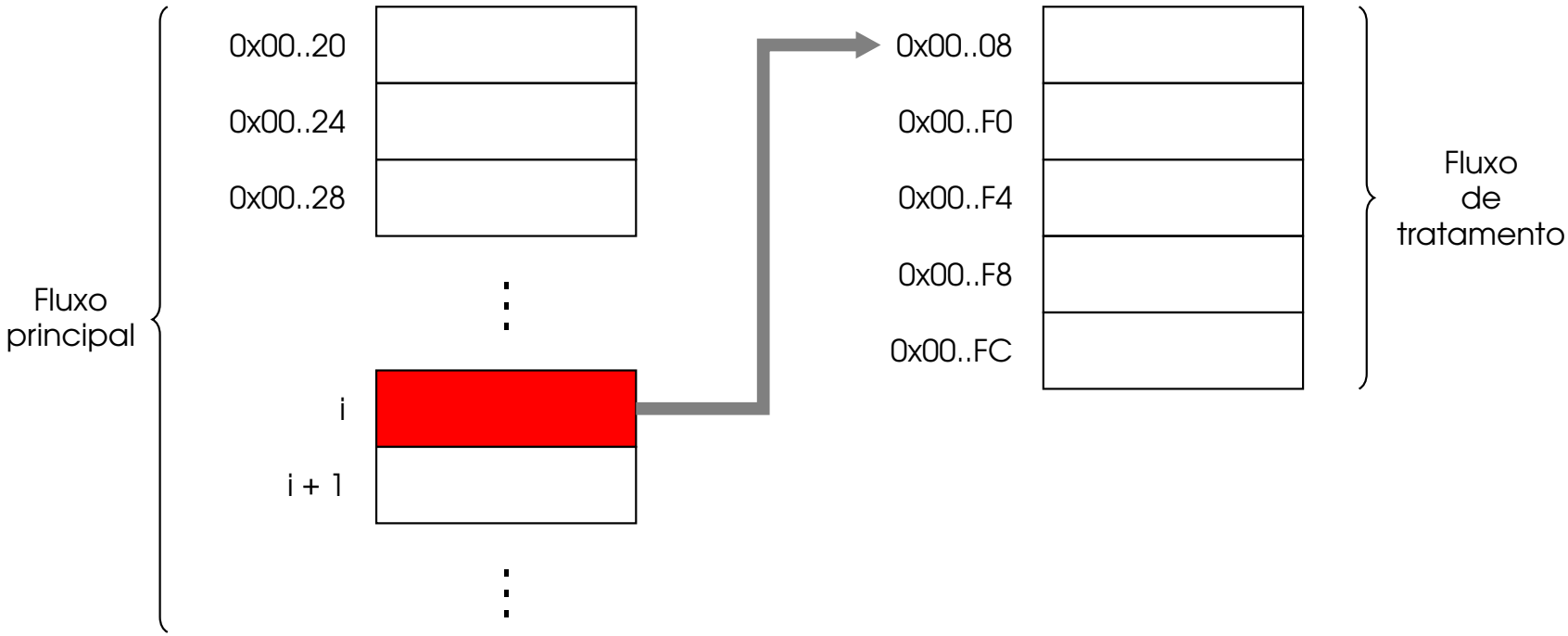
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo principal da aplicação

Interrupção

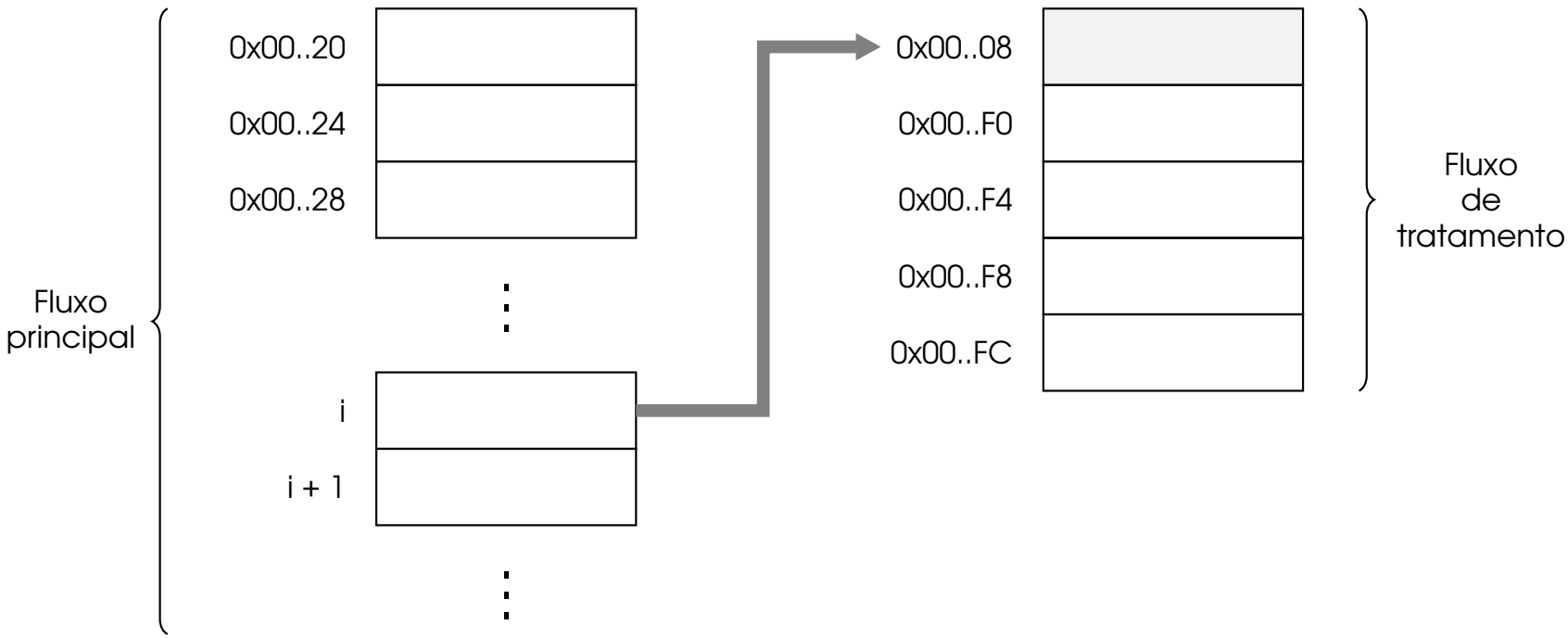
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Evento de interrupção gerado por
hardware ou software causa um desvio para ISR

Interrupção

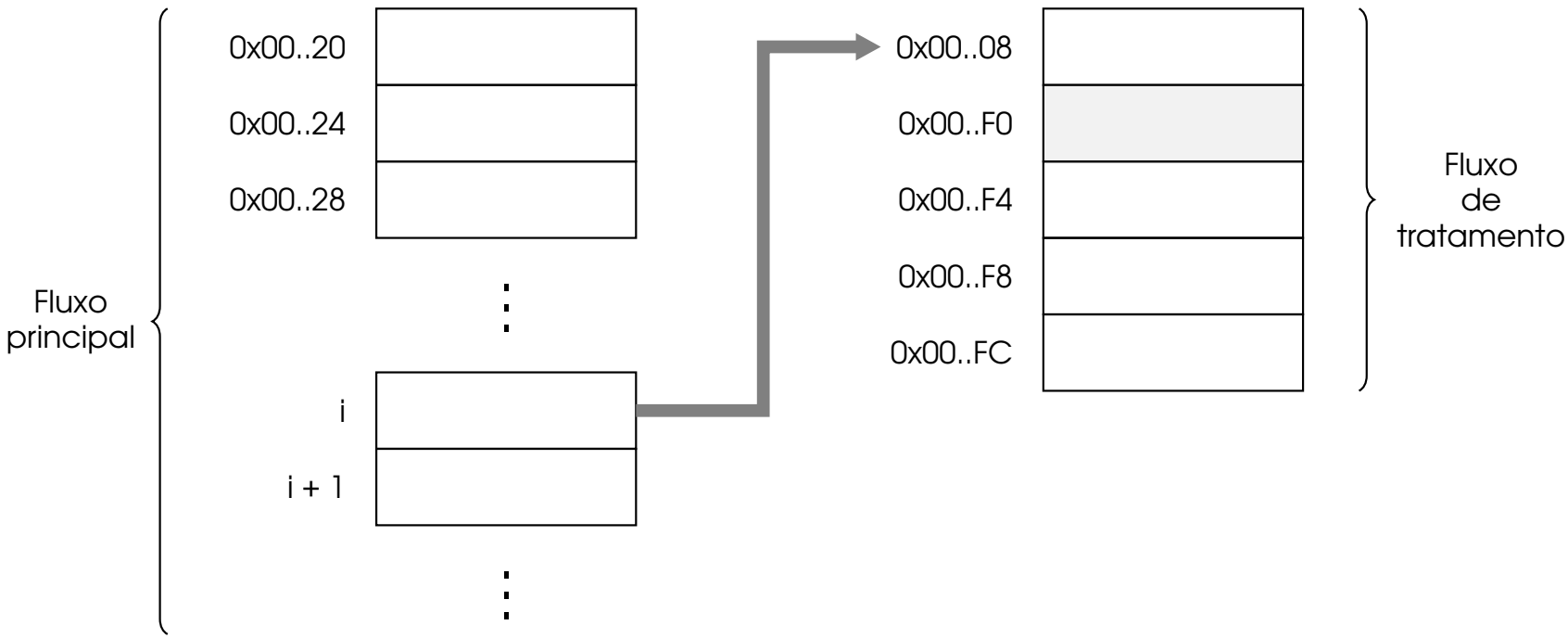
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo de tratamento da interrupção

Interrupção

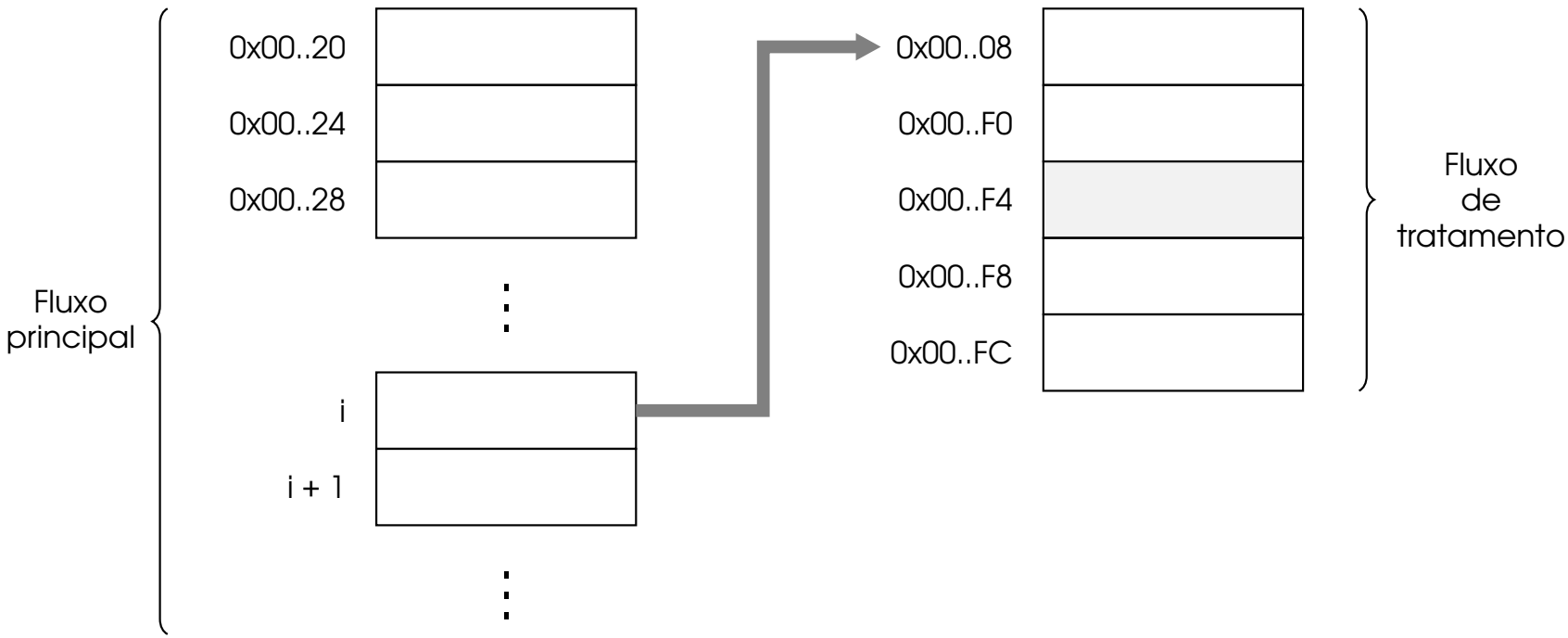
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo de tratamento da interrupção

Interrupção

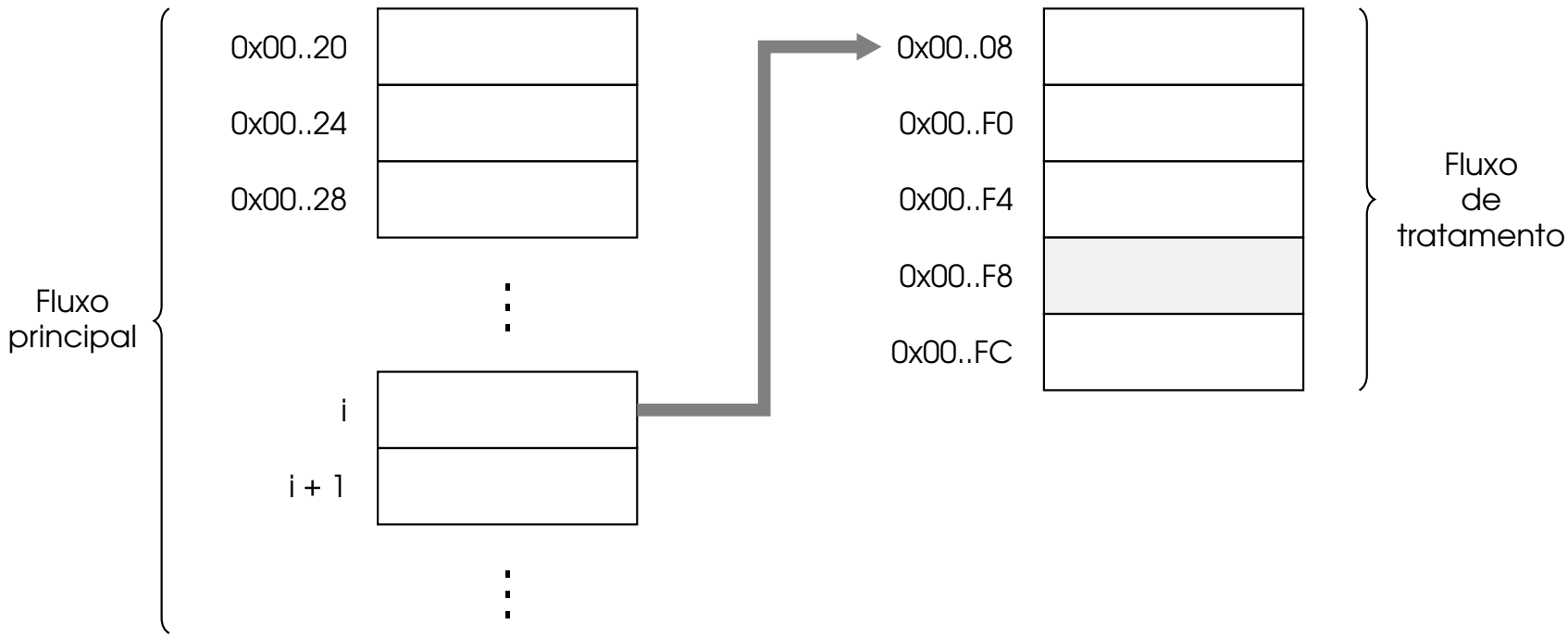
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo de tratamento da interrupção

Interrupção

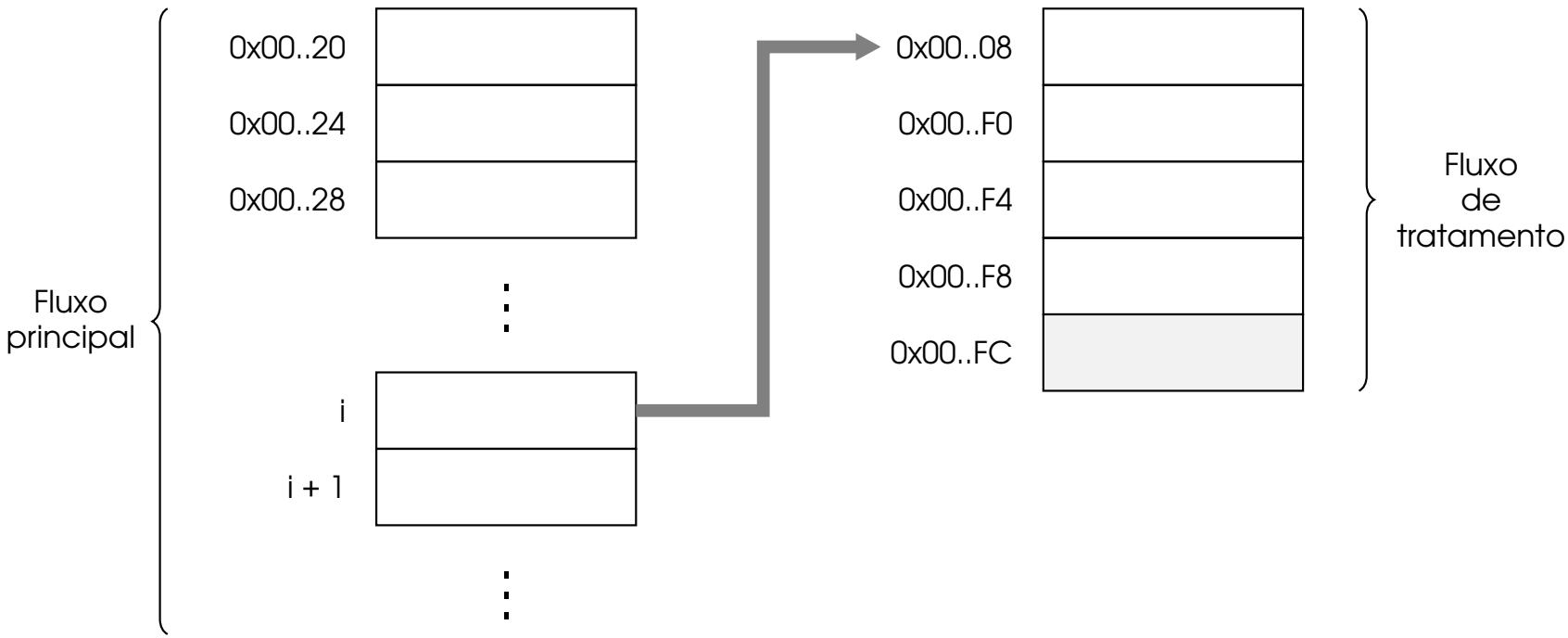
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo de tratamento da interrupção

Interrupção

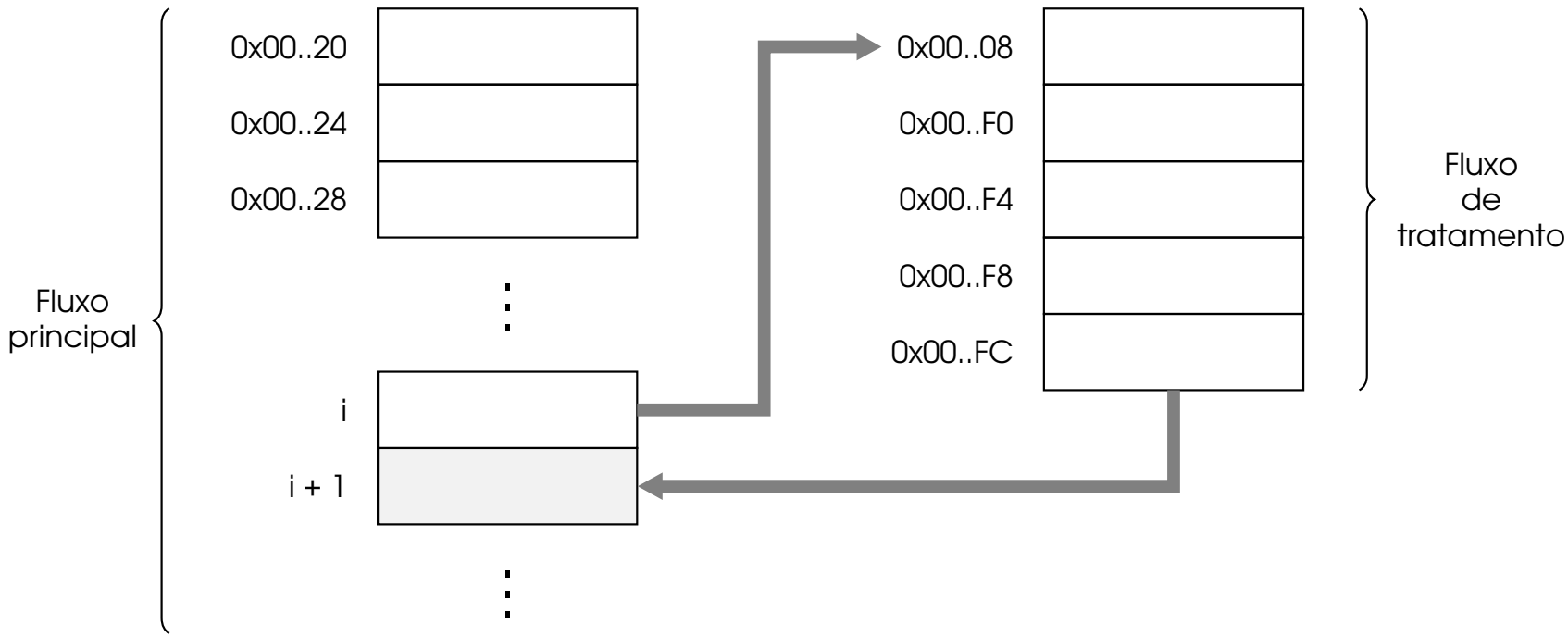
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Execução do fluxo de tratamento da interrupção

Interrupção

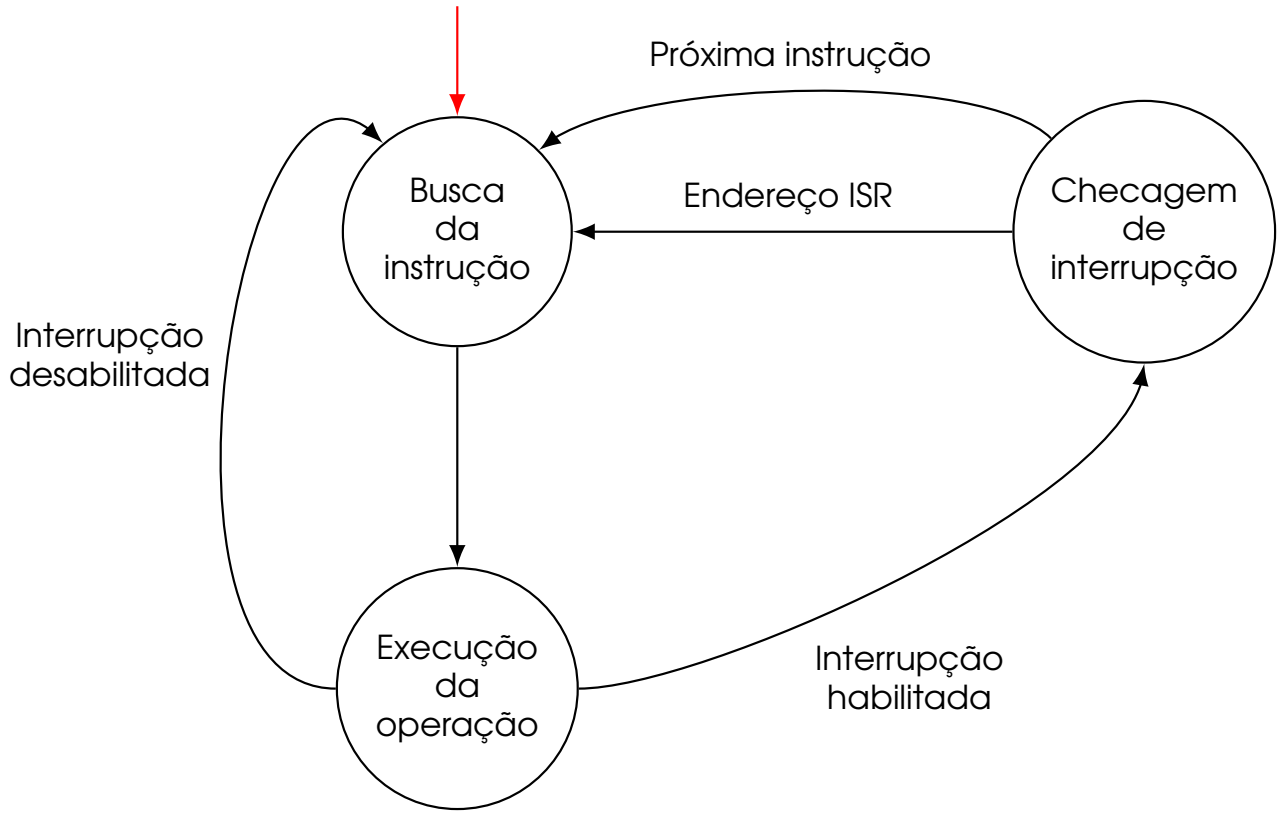
- Controle de fluxo para interrupção
  - Interrupção gerada durante execução da instrução *i*



Retorno ao fluxo anterior de execução

Interrupção

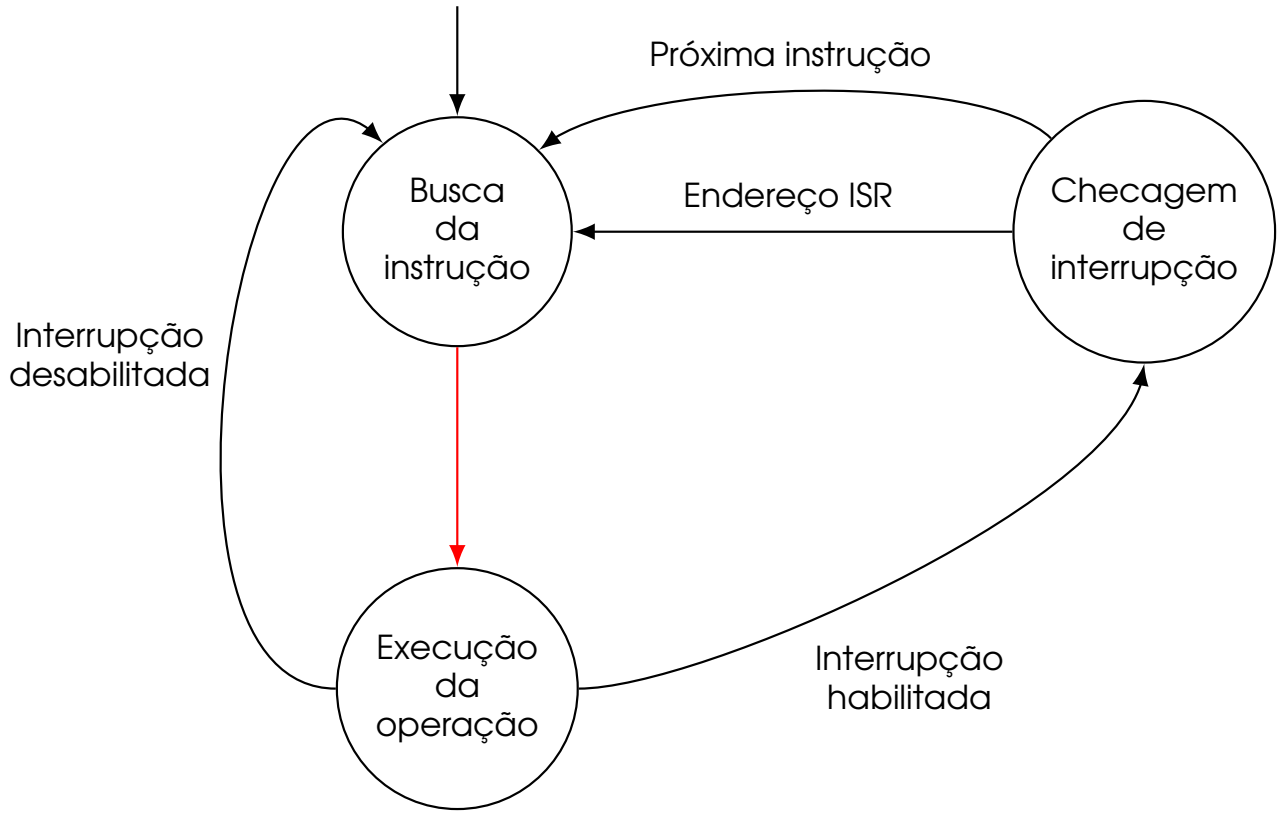
- Controle de fluxo para interrupção
  - Máquina de estados



Busca da próxima instrução

Interrupção

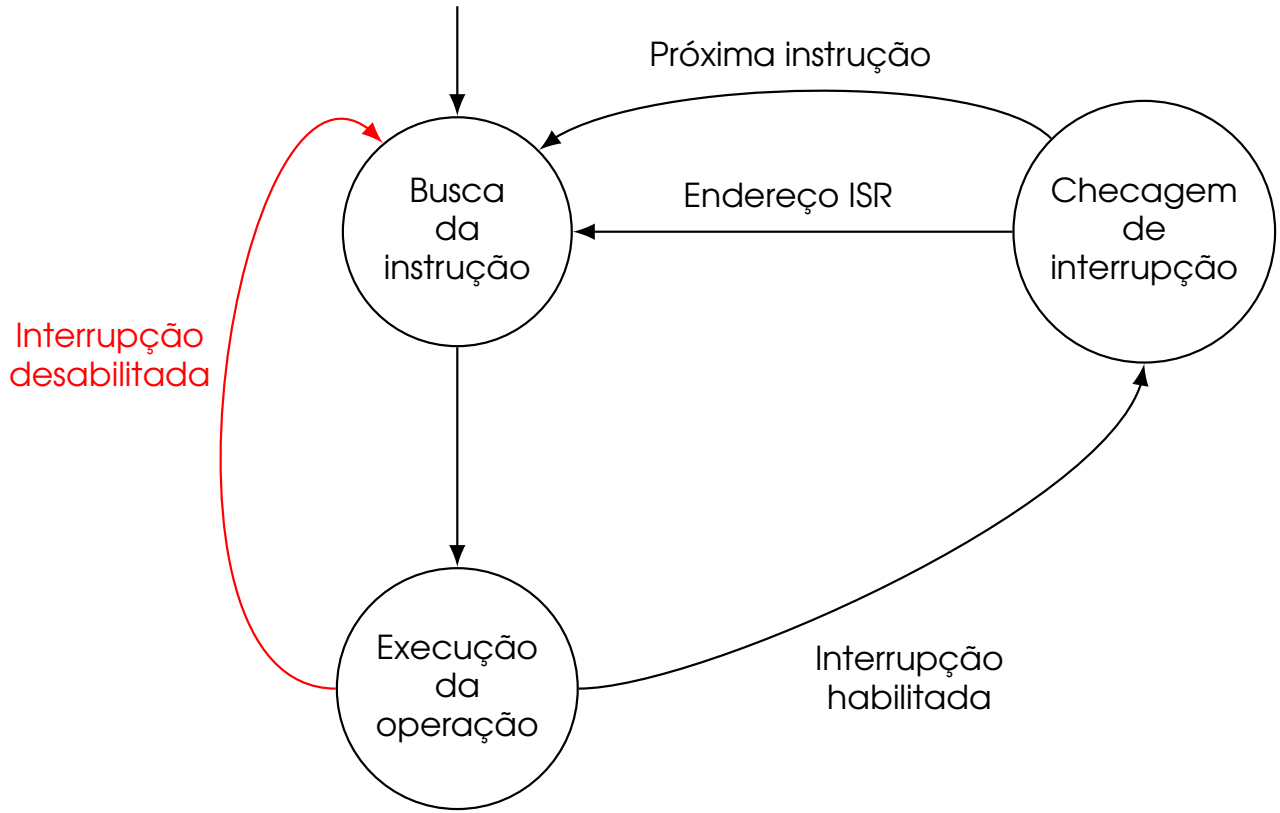
- Controle de fluxo para interrupção
  - Máquina de estados



Execução da operação

Interrupção

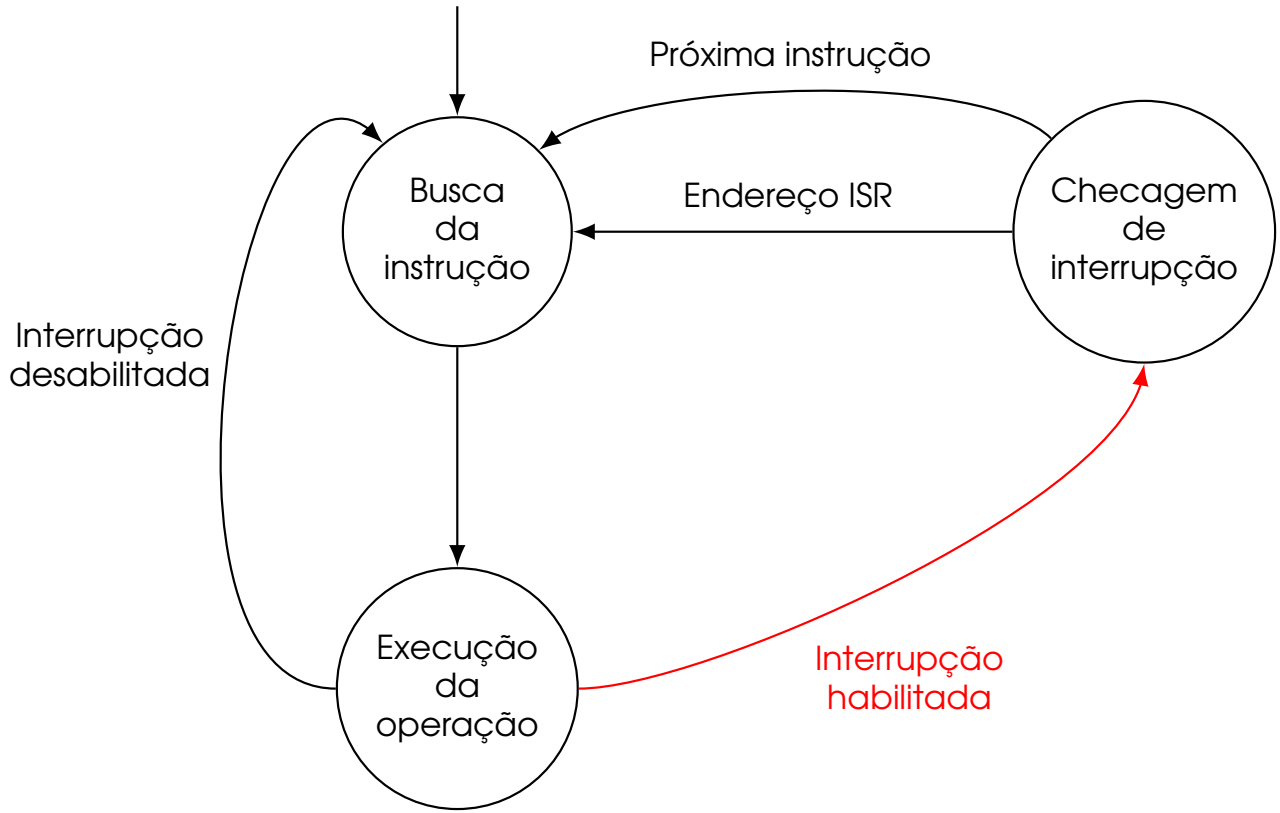
- Controle de fluxo para interrupção
  - Máquina de estados



Interrupções mascaráveis ficam pendentes

Interrupção

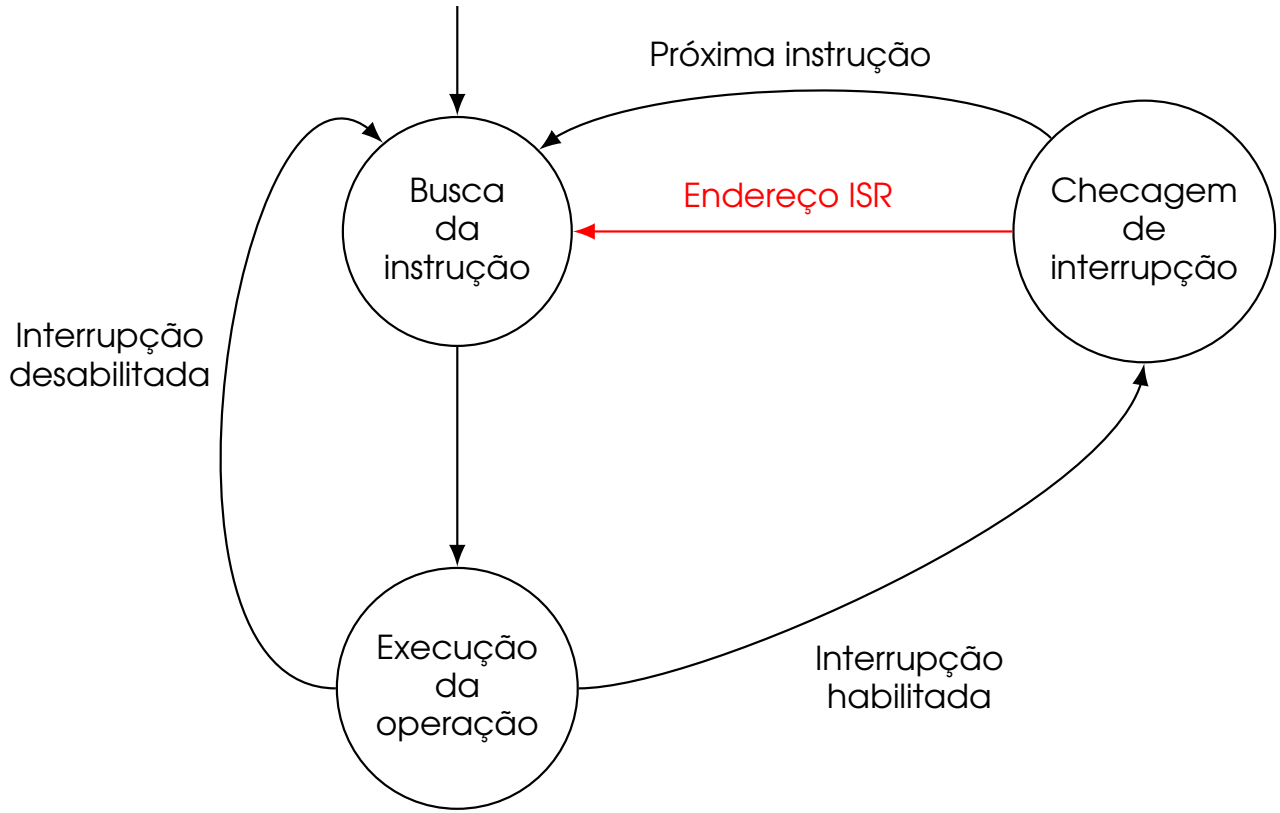
- Controle de fluxo para interrupção
  - Máquina de estados



É feita a checagem por interrupções pendentes

Interrupção

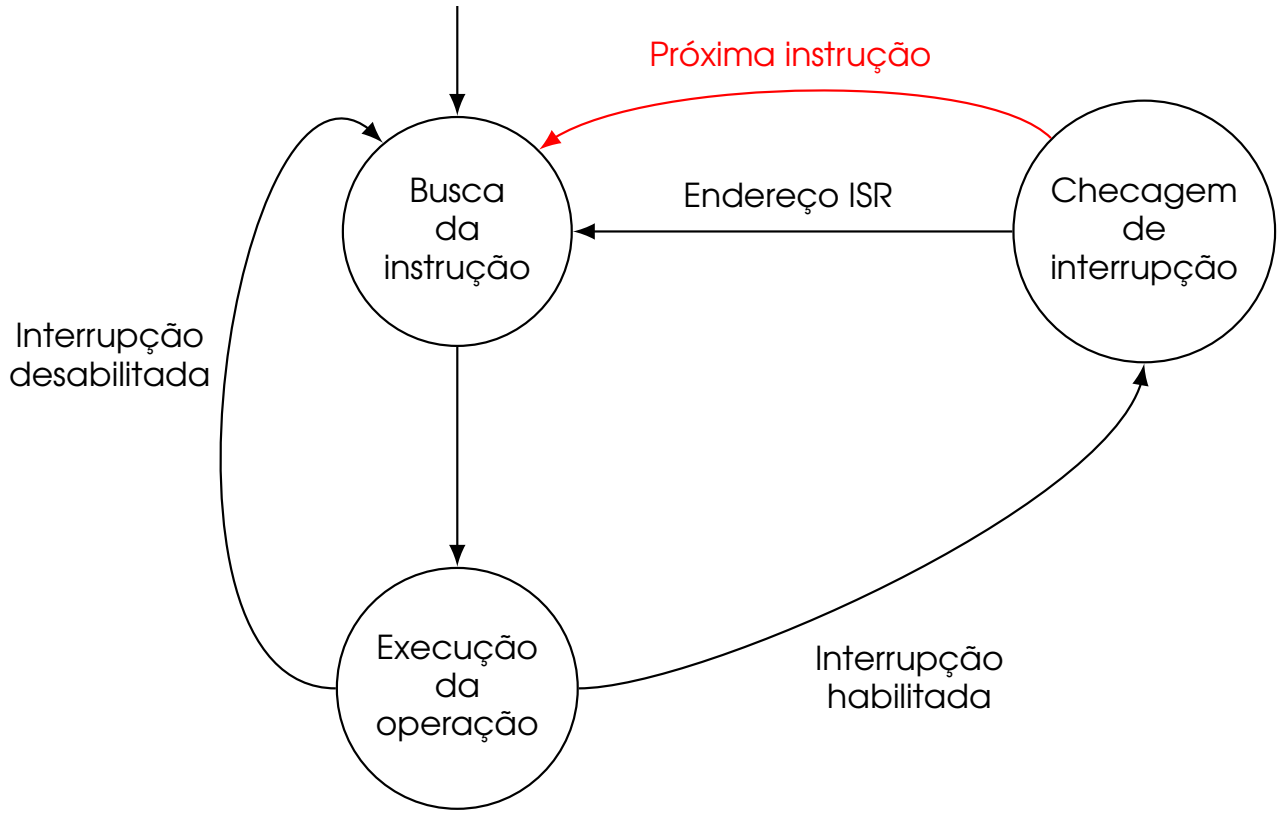
- Controle de fluxo para interrupção
  - Máquina de estados



O endereço da ISR é atribuído ao PC, caso exista alguma interrupção pendente

Interrupção

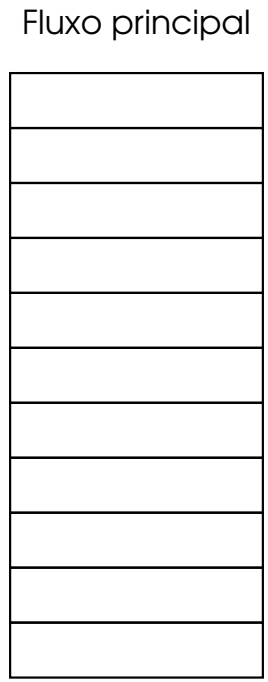
- Controle de fluxo para interrupção
  - Máquina de estados



Com nenhuma interrupção pendente,
a próxima instrução é buscada

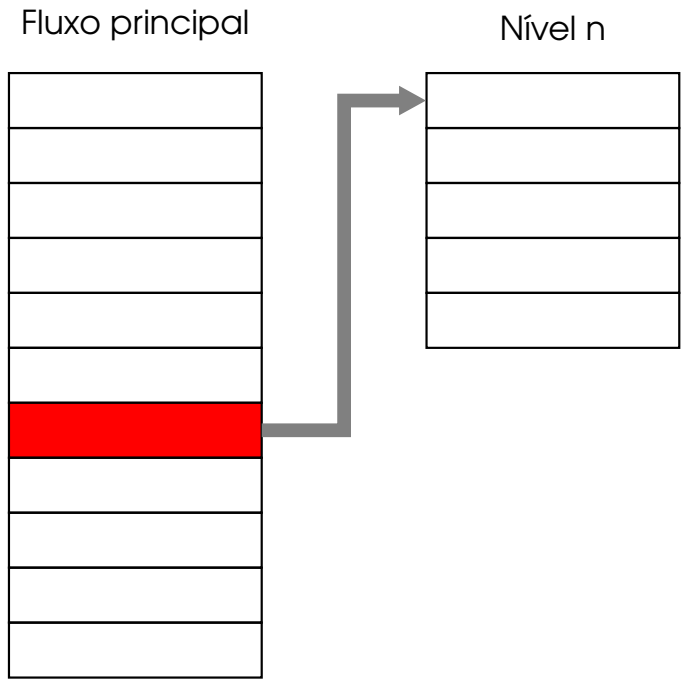
Interrupção

- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Interrupção

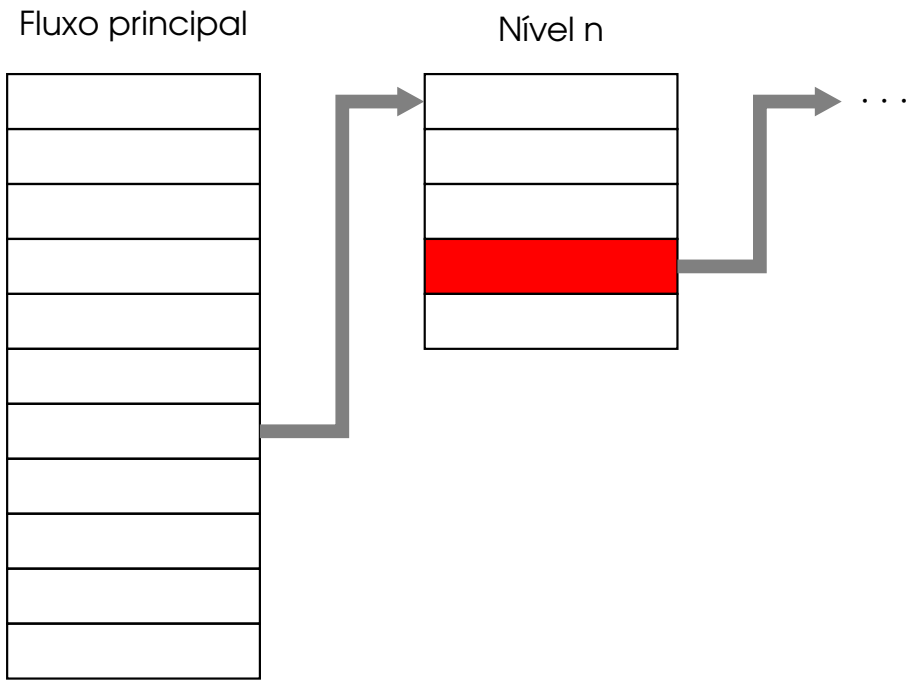
- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível
possuem maior prioridade (−3 - máxima e *n* - mínima)

Interrupção

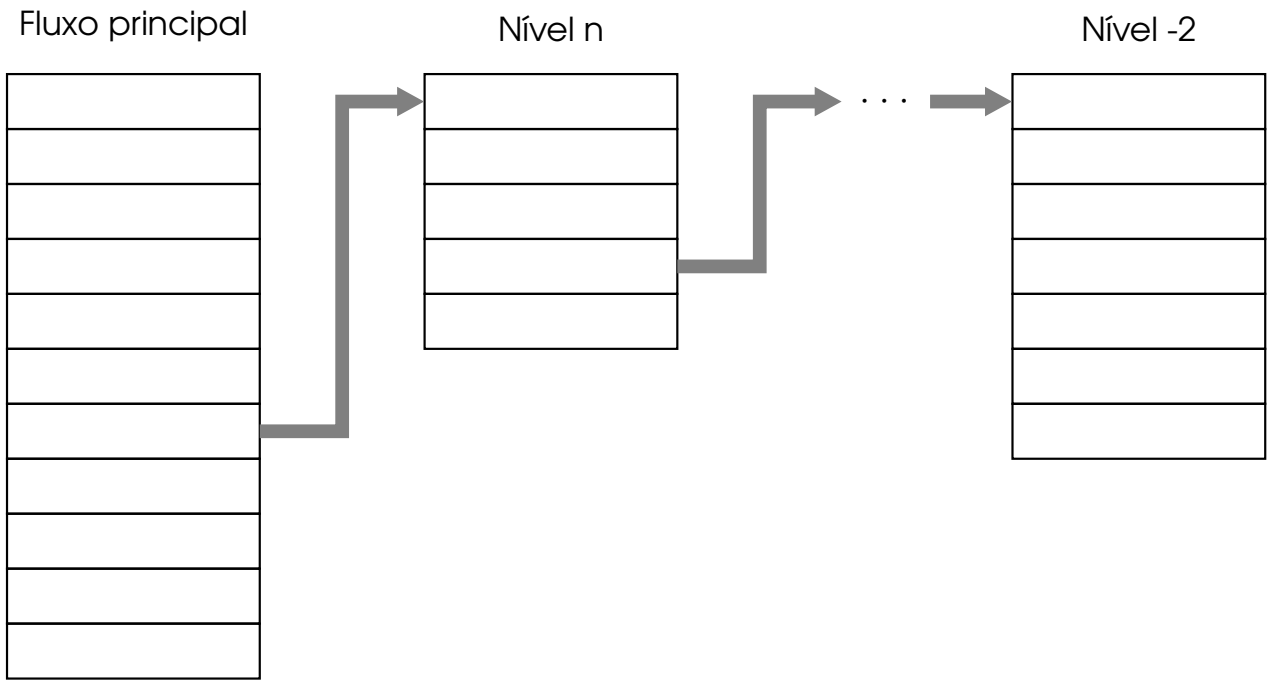
- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível
possuem maior prioridade (−3 - máxima e *n* - mínima)

Interrupção

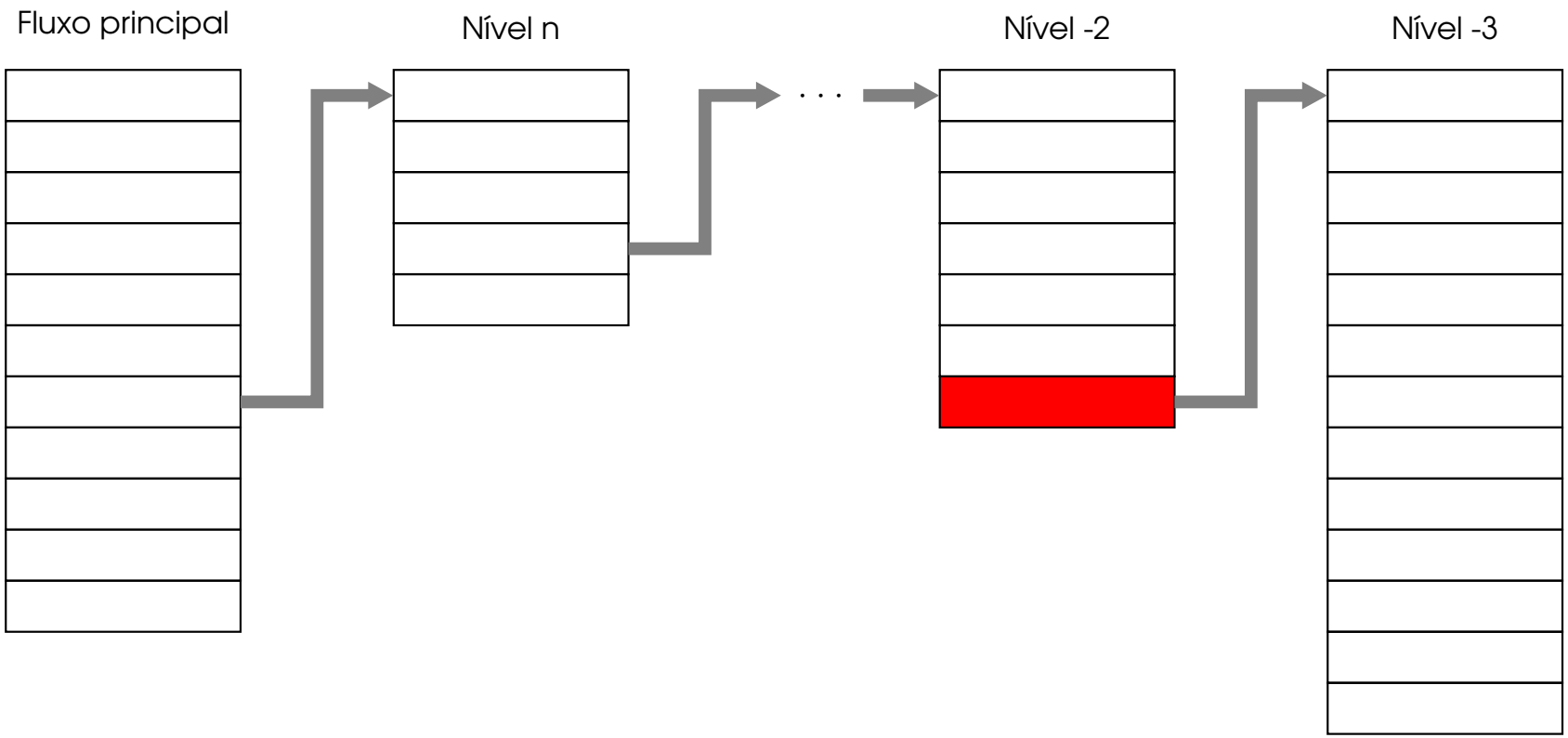
- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível
possuem maior prioridade (-3 - máxima e n - mínima)

Interrupção

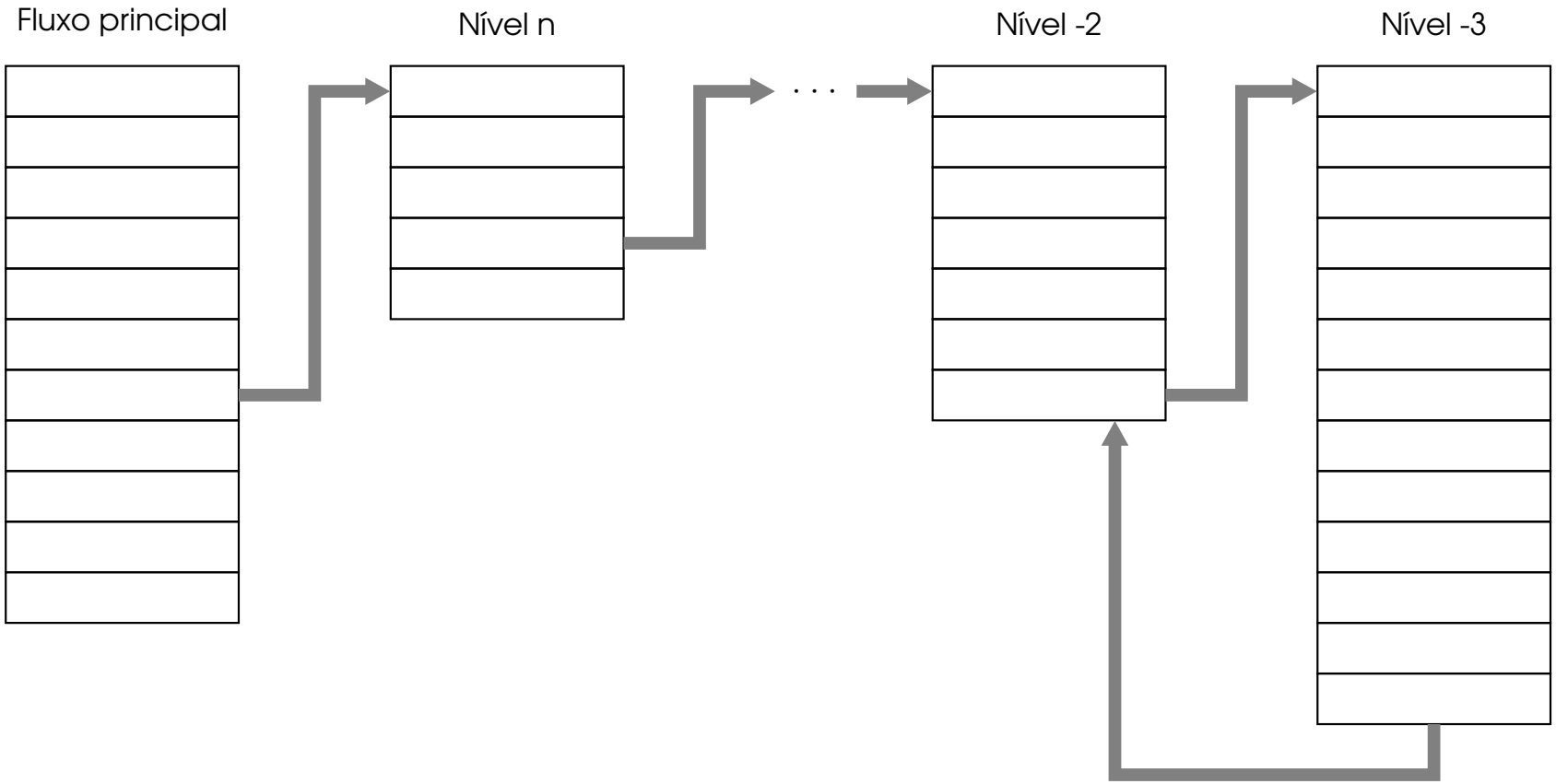
- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível possuem maior prioridade (−3 - máxima e *n* - mínima)

Interrupção

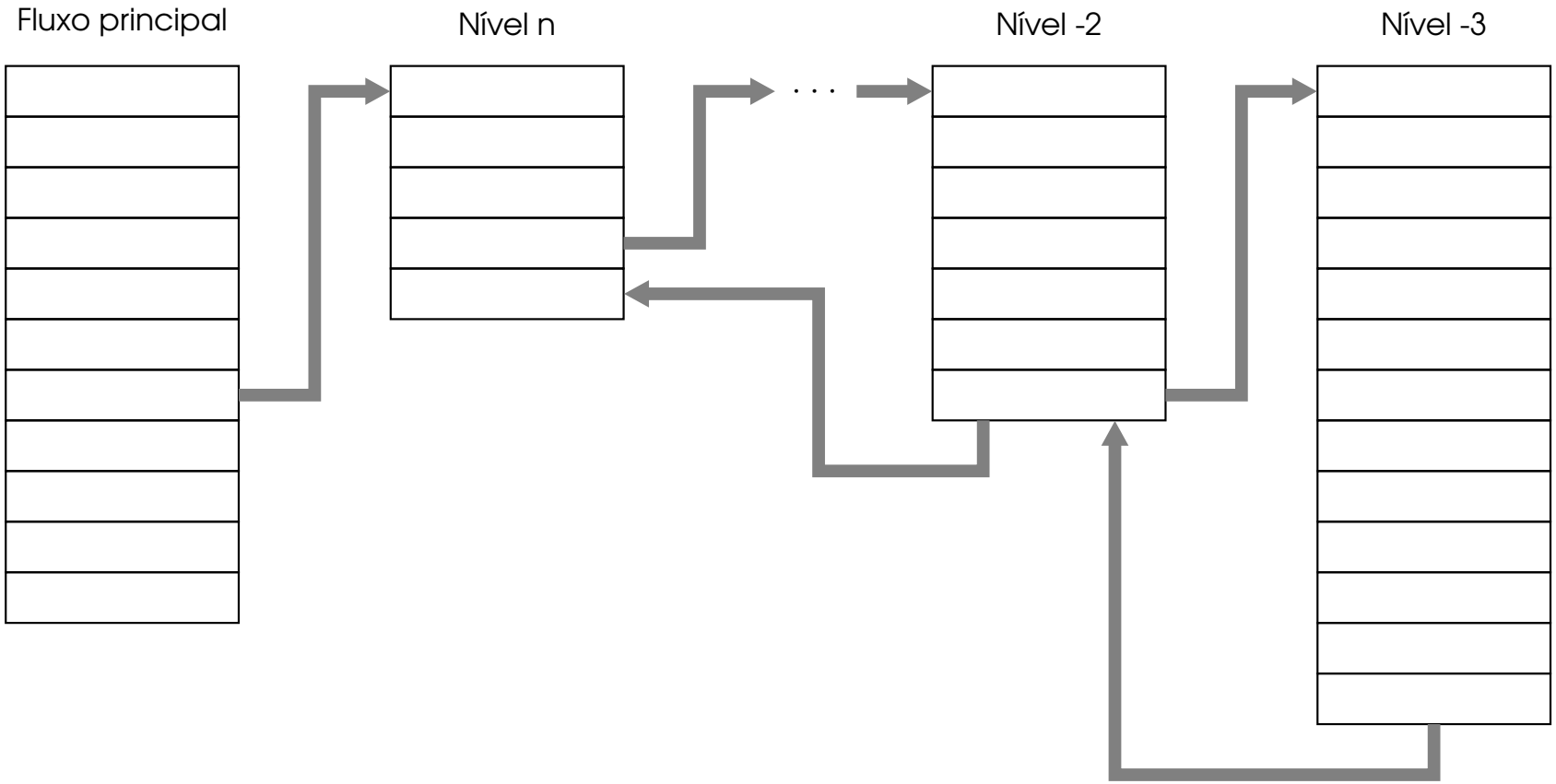
- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível possuem maior prioridade (−3 - máxima e *n* - mínima)

Interrupção

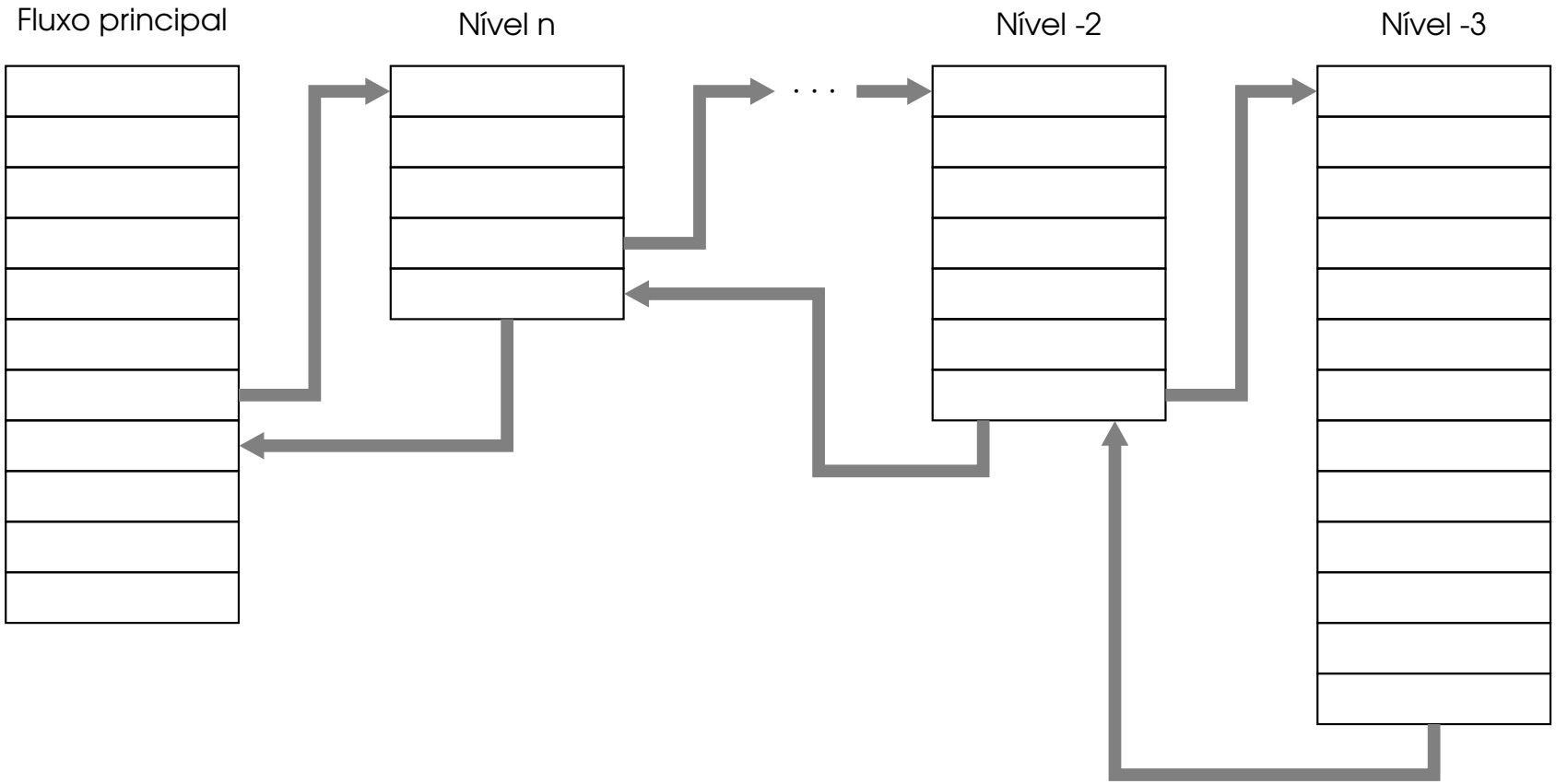
- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível possuem maior prioridade (−3 - máxima e *n* - mínima)

Interrupção

- Priorização das interrupções
  - Como as requisições são organizadas em uma fila de prioridade, é possível o aninhamento das ISRs



Geralmente os menores valores de nível possuem maior prioridade (−3 - máxima e *n* - mínima)

Interrupção

- Tabela de vetor de interrupção (software)

Prioridade	Tipo	Endereço
-	Topo da pilha	0x00000000
-3	Reset	0x00000004
-2	NMI	0x00000008
-1	Hard fault	0x0000000C
0*	Memory management fault	0x00000010
1*	Bus fault	0x00000014
2*	Usage fault	0x00000018
-	Reservado	$0x00..1C \leftrightarrow 0x00..2B$
3*	SVCall	0x0000002C
4*	Debug monitor	0x00000030
-	Reservado	0x00000034
5*	PendSV	0x00000038
6*	SysTick	0x0000003C

Interrupção

- Tabela de vetor de interrupção (hardware)

Prioridade	Tipo	Endereço
7*	Window Watchdog interrupt	0x00000040
8*	PVD through EXTI Line detection interrupt	0x00000044
9*	Tamper interrupt	0x00000048
10*	RTC global interrupt	0x0000004C
11*	Flash global interrupt	0x00000050
12*	RCC global interrupt	0x00000054
13*	EXTI Line0 interrupt	0x00000058
14*	EXTI Line1 interrupt	0x0000005C
15*	EXTI Line2 interrupt	0x00000060
16*	EXTI Line3 interrupt	0x00000064
17*	EXTI Line4 interrupt	0x00000068
⋮	⋮	⋮
74*	USB On The Go FS global interrupt	0x0000014C

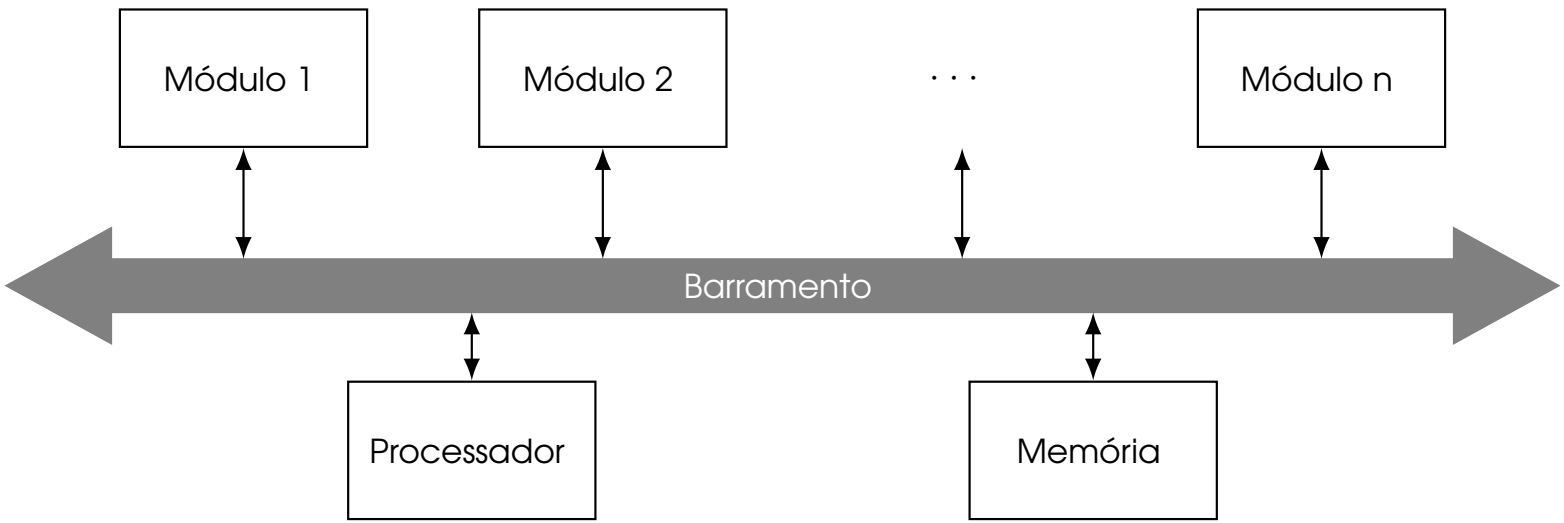
Interrupção

- Tabela de vetor de interrupção

```
 1  // Tipos inteiros de tamanho fixo
 2  #include <stdint.h>
...
53  // Declaração de topo da pilha e de funções
54  extern char _estack;
55  int main();
56  void SysTick();
57  void botao();
58  // Tabela de vetor de interrupção
59  uint32_t (* const vector_table[]) __attribute__
        ((section(".text.vector_table"))) = {
60      (uint32_t*)(&_estack), // Topo da pilha
61      (uint32_t*)(main),     // Reset
62      0,                     // NMI
...
72      (uint32_t*)(SysTick),  // SysTick
...
79      (uint32_t*)(botao),    // EXTI Line0 interrupt
...
84  };
...
```

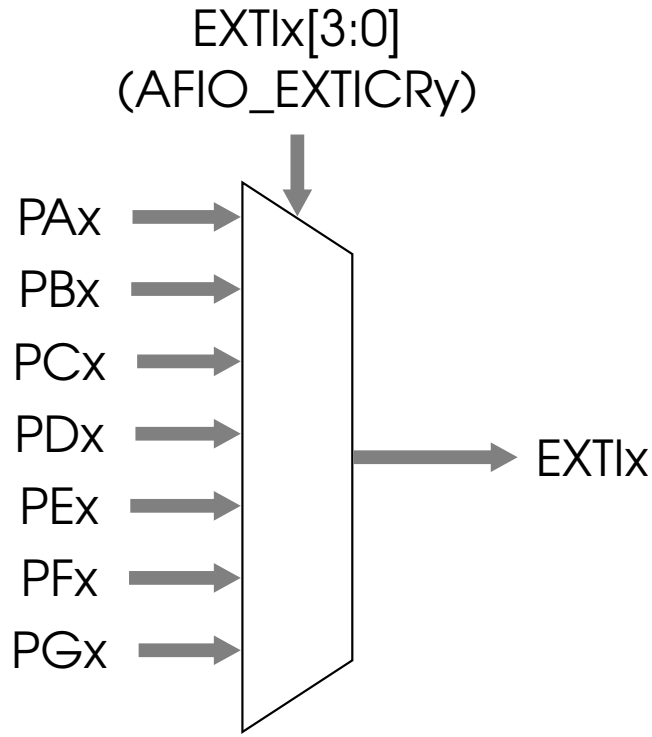
Interrupção

- Eventos de interrupção de hardware
  - São requisições assíncronas de periféricos de E/S da plataforma que solicitam do processador a execução de rotinas para realizar a transferência de dados ou para realizar ações pré-definidas pelo programador



Interrupção

- Mapeamento de interrupção externa (EXTI)

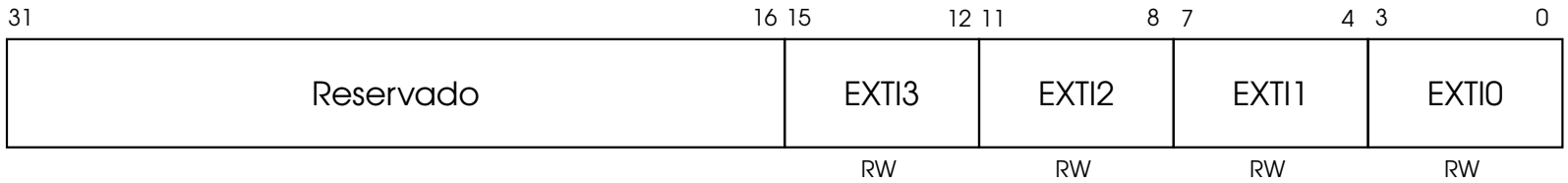


$x = [0, 15]$
$y = [0, 4]$

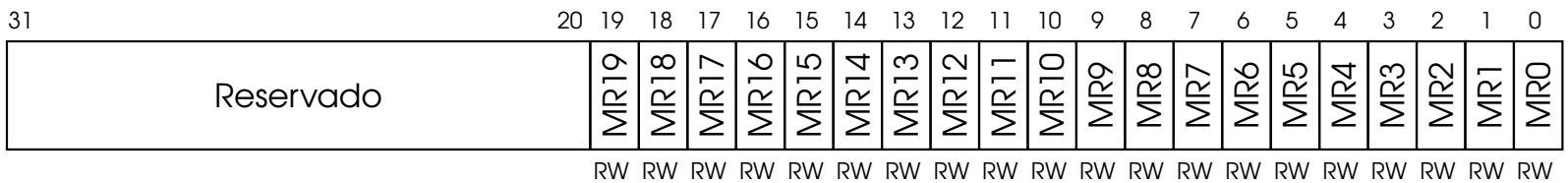
- As linhas 16 até 19 estão associadas aos eventos de saída de PVD, alarme de RTC, despertar (*wake up*) de USB e Ethernet, respectivamente

Interrupção

- Registrador de configuração de EXTI 1 (AFIO_EXTICR1)
  - *Endereço* = 0x40010008
  - *Valor inicial* = 0x0000

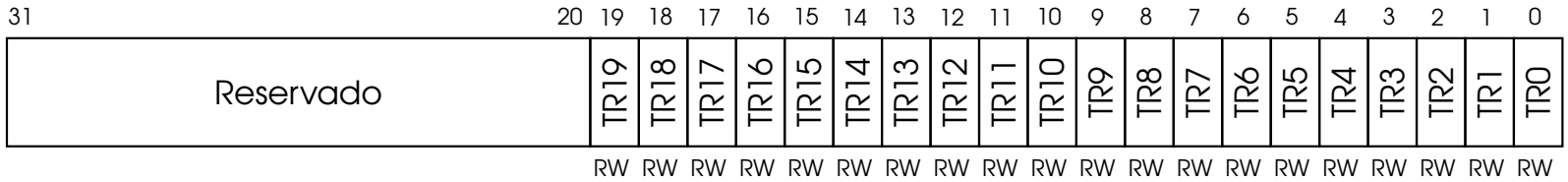


- Registrador de máscara de interrupção (EXTI_IMR)
  - *Endereço* = 0x40010400
  - *Valor inicial* = 0x00000000

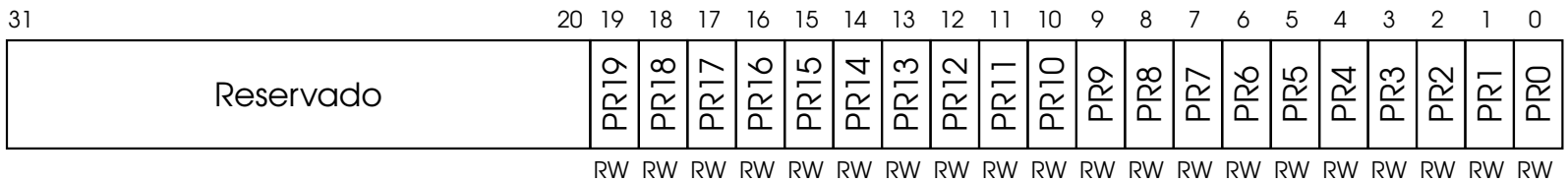


Interrupção

- Registrador de gatilho de descida (EXTI_FTSR)
  - *Endereço* = 0*x*4001040*C*
  - *Valor inicial* = 0x00000000

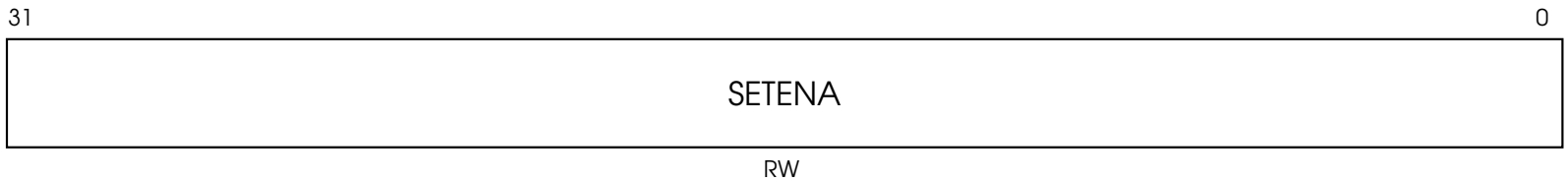


- Registrador de pendência de interrupção (EXTI_PR)
  - *Endereço* = 0*x*40010414
  - *Valor inicial* = 0*xXXXXXXXX*

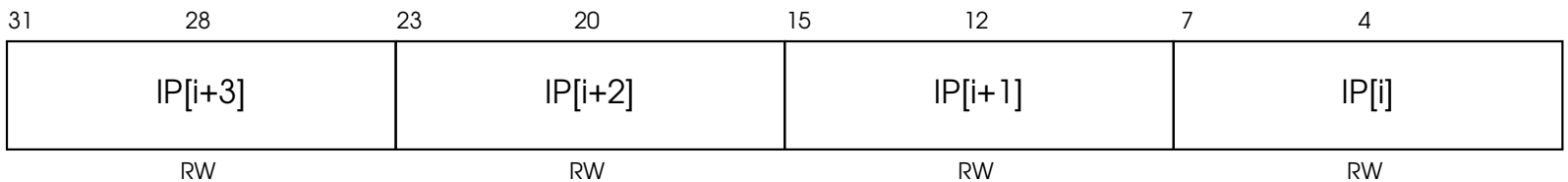


Interrupção

- Registrador de ativação de interrupção (NVIC_ISER0)
  - *Endereço* = 0*xE*000*E*100
  - *Valor inicial* = 0x00000000



- Registrador de prioridade de interrupção (NVIC_IPR)
  - *Endereço* = 0*xE*000*E*400 (*IP*[0] até *IP*[67])
  - *Valor inicial* = 0x00000000



Interrupção

- Definições de endereços de registradores

```
1   // Tipos inteiros de tamanho fixo
2   #include <stdint.h>
... ...
33  // Definição de registradores de AFIO, EXTI e NVIC
34  #define AFIO_EXTICR1 (0x40010008)
35  #define EXTI_IMR     (0x40010400)
36  #define EXTI_FTSR    (0x4001040C)
37  #define EXTI_PR      (0x40010414)
38  #define NVIC_ISER0   (0xE000E100)
39  #define NVIC_IPR     (0xE000E400)
... ...
```

Interrupção

- Configuração para associação de PA0 para EXTI0

```
1   // Tipos inteiros de tamanho fixo
2   #include <stdint.h>
... ...
163 // Configurar pino PA0 para interrupção em EXTI0
164 void configurar_PA0_EXTI0() {
165     // EXTI0 <-> GPIO A
166     (*EXTICR1) &= 0xFFF0;
167     // Interrupção por gatilho de borda negativa
168     (*FTSR) |= 1;
169     // Desativar mascaramento
170     (*IMR) |= 1;
171     // Habilitando interrupção
172     (*ISER0) |= (1 << 6);
173     // Ajustando prioridade para nível 13
174     IPR[6] = 0xD0;
175 }
... ...
```

Interrupção

- Rotina de tratamento de interrupção para EXTI0

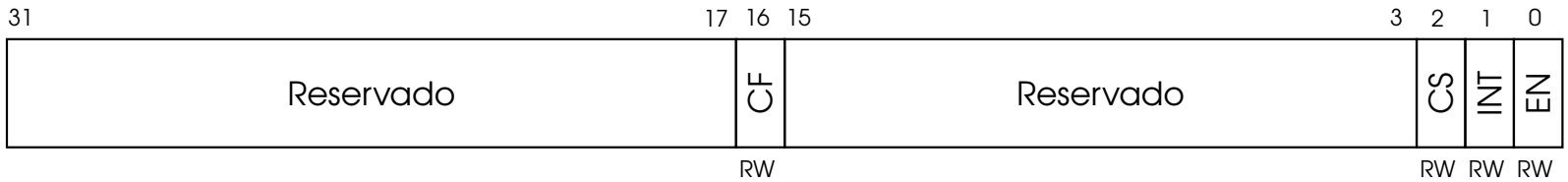
```
 1  // Tipos inteiros de tamanho fixo
 2  #include <stdint.h>
...
118 // Função de botão
119 void botao() {
...
136     // Checando registrador de pendência
137     if((*PR) & 1) {
138         // Limpando pendência no bit 0
139         (*PR) = 1;
140     }
141 }
...
```

Interrupção

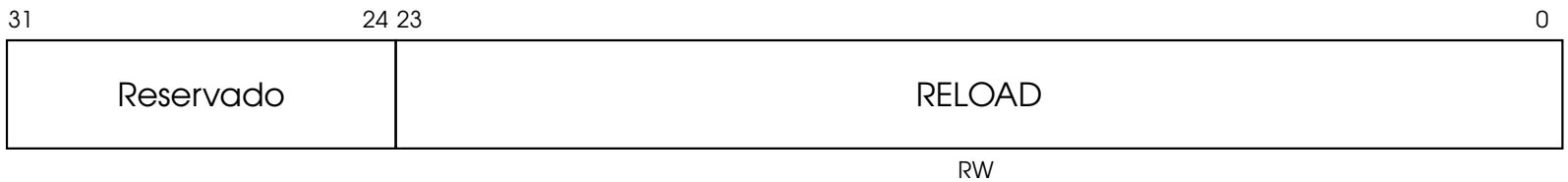
- Eventos de interrupção de software
  - Podem ser gerados explicitamente por instruções de interrupção ou implicitamente por exceções decorrentes de operações realizadas
  - A interrupção de software *SysTick* pode ser programada para gerar interrupções em um intervalo de tempo configurável pelo programador

Interrupção

- Registrador de controle e status (SYST_CSR)
  - *Endereço* = 0*xE*000*E*010
  - *Valor inicial* = 0x00000004



- Registrador de recarga (SYST_RVR)
  - *Endereço* = 0*xE*000*E*014
  - *Valor inicial* = 0*xXXXXXXXX*



Interrupção

- Configuração do *SysTick*

```
 1  // Tipos inteiros de tamanho fixo
 2  #include <stdint.h>
...
26  // Definição de registradores de SysTick
27  #define SYST_CSR            (0xE000E010)
28  #define SYST_RVR            (0xE000E014)
29  // Definição de campos de SYST_CSR
30  #define SYST_CSR_CLKSOURCE (2)
31  #define SYST_CSR_TICKINT    (1)
32  #define SYST_CSR_ENABLE     (0)
...
156 // Configurar SysTick
157 void configurar_SysTick() {
158     // Configuração do SysTick para 1 segundo (8 Mhz)
159     (*RVR) = 8000000;
160     // Habilitando SysTick com clock interno e interrupção
161     (*CSR) |= (1 << SYST_CSR_CLKSOURCE) | (1 <<
            SYST_CSR_TICKINT) | (1 << SYST_CSR_ENABLE);
162 }
...
```

Interrupção

- Rotina de tratamento de interrupção para *SysTick*

```
  1 // Tipos inteiros de tamanho fixo
  2 #include <stdint.h>
... ...
113 // Função SysTick
114 void SysTick() {
115     // Invertendo valor do pino PC13 (LED)
116     (*PC_ODR) ^= (1 << PC13_ODR);
117 }
... ...
```

Exercício

- Estude e reproduza os experimentos vistos nesta aula
  - Analise as configurações possíveis para os dispositivos de E/S, como modo de operação ou taxa de transferência, além de entender o funcionamento de outros dispositivos, como ADC/DAC e Timer
  - Implemente uma versão equivalente do exemplo fornecido utilizando o *framework* Arduino e faça um comparativo de utilização de memória
  - Pesquise por ferramentas para simulação do software embarcado baseado em plataformas ARM

Exercício

- Implemente um sistema de controle semafórico com sincronização por comunicação serial
  - Considere o cenário com um par sincronizados de semáforos, cada um com seu botão de pedestre, que permite o controle adaptativo do fluxo de pessoas e veículos em um cruzamento de duas vias com temporização padrão de 5 segundos para amarelo e 15 segundos para verde e vermelho
    - Quando o pedestre pressiona o botão, é feito o envio de $'B'$ e ocorre a mudança do sinal verde para amarelo ou prorrogação do vermelho por 15 segundos
    - A transição de estado envia $'R'$ (vermelho), $'G'$ (verde), $'Y'$ (amarelo) ou $'S'$ (sincronização)
    - No processo de sincronização, o sinal fica piscando em amarelo até receber uma mensagem com estado válido do outro semáforo
    - É enviado a contagem de tempo restante em segundos para cada estado, exceto quando estiver em sincronização

Exercício

- Para fins de compatibilidade entre o hardware da placa (*blue pill*, STM32F103) e o simulador QEMU (*netduino2*, STM32F205), realize os ajustes necessários de portabilidade do código fonte
  - A comunicação serial deve ser feita com taxa de transferência de 9600 bps e gerar interrupção para recebimento dos dados
  - Utilize um dos *timers* suportados (não utilizar o *systick*) para gerar interrupções com intervalo de 1 segundo e com prioridade inferior à interrupção da serial