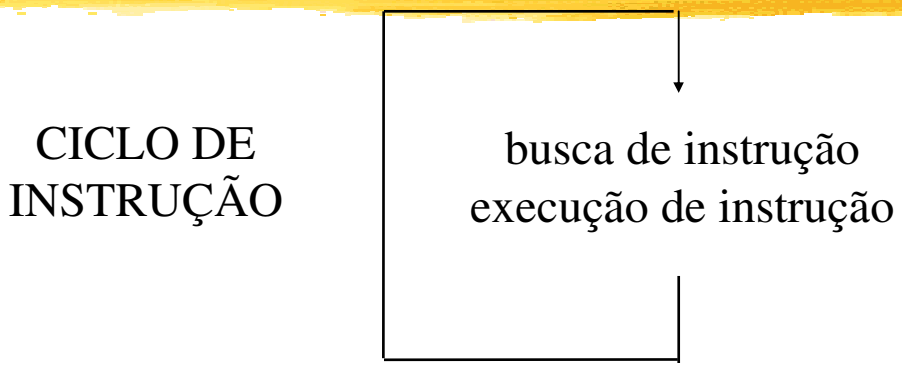
7. Interrupção
(capítulo 6 do livro texto)

Execução de um programa

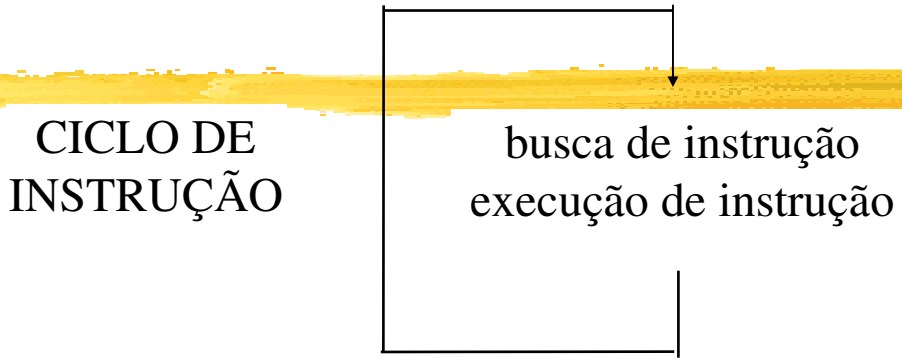


Interrupção: é a quebra da sequência de operação de forma a atender eventos especiais, tais como:

- entrada de dados atendendo à solicitação de periférico
- atendimento de sinal de sensor
- sinal interno gerado pelo relógio do sistema

Interrupção

Execução de um programa

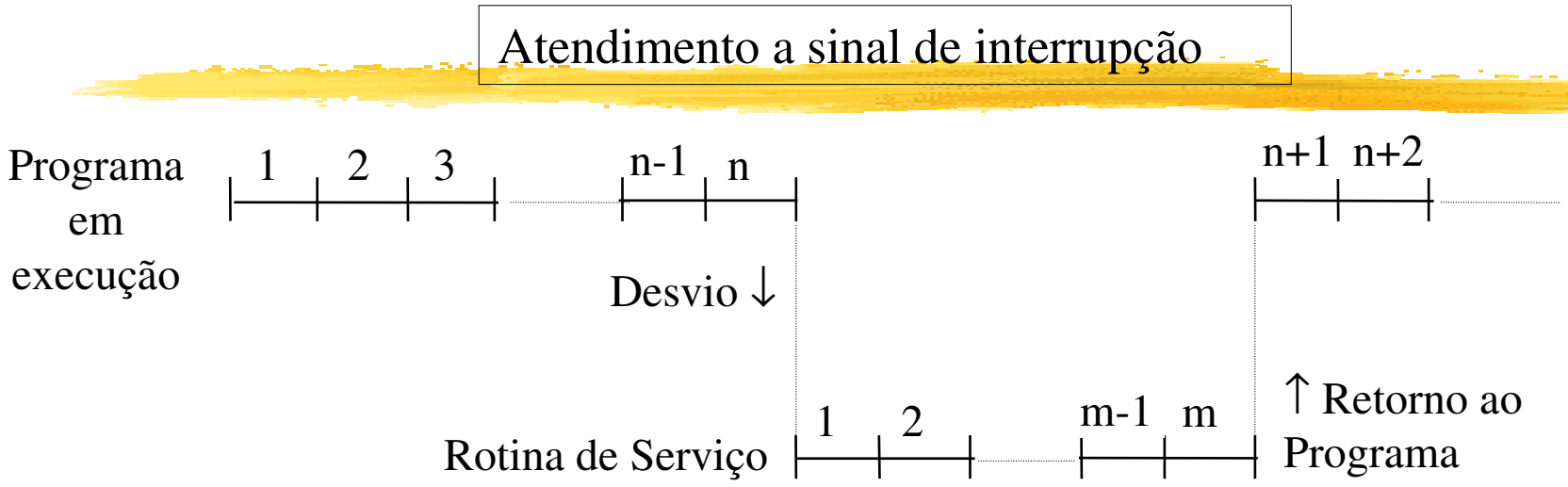


Interrupção: é a quebra da sequência de operação de forma a atender eventos especiais, tais como:

. entrada de dados atendendo à solicitação de periférico
. atendimento de sinal de sensor
. sinal interno gerado pelo relógio do sistema

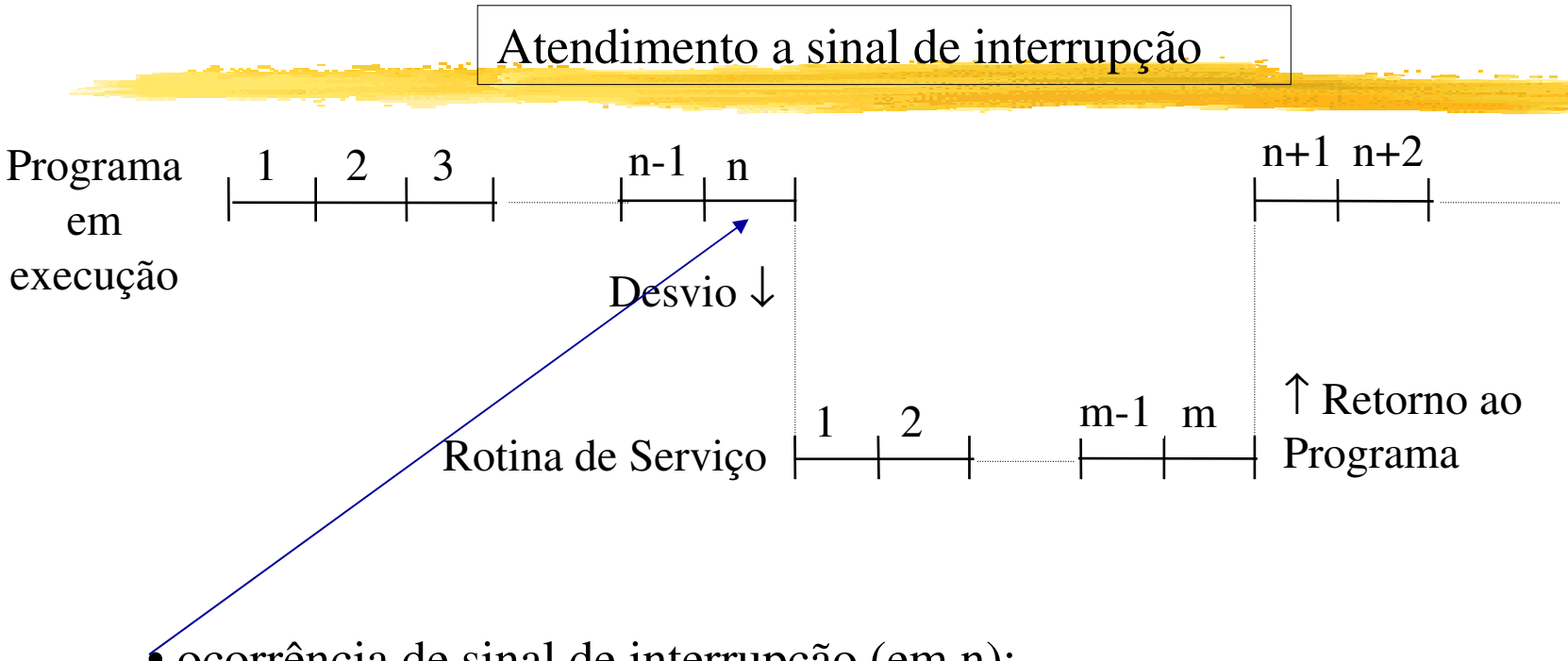
via sinal de interrupção

7.1 Mecanismo de Interrupção



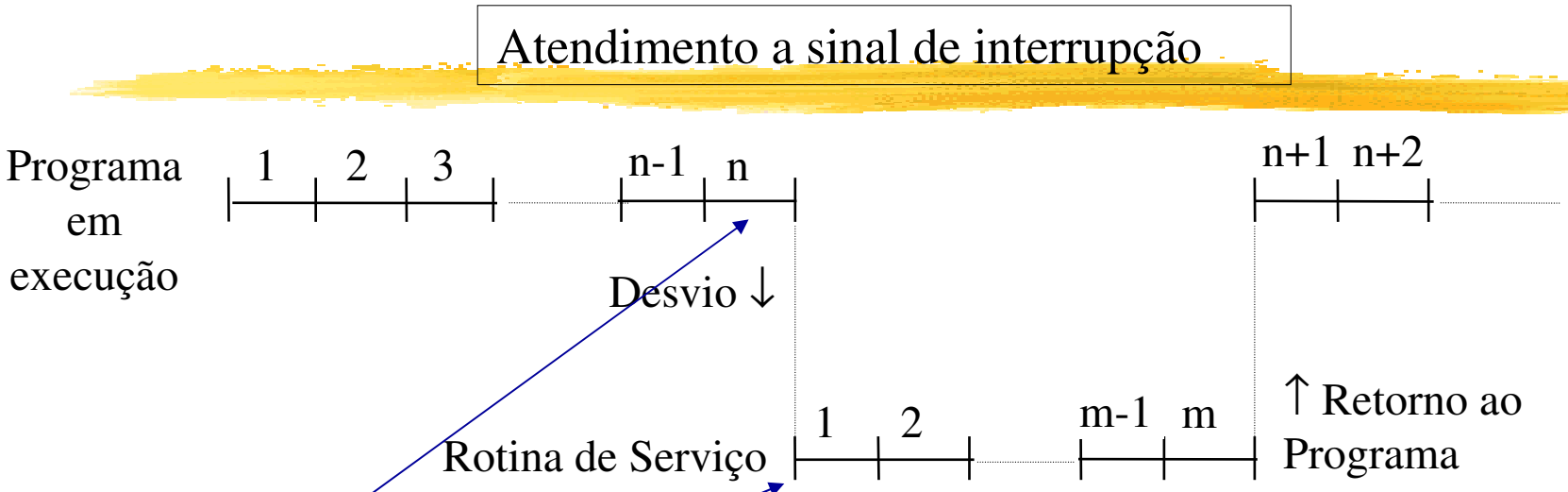
- ocorrência de sinal de interrupção (em n);
- atendimento da interrupção via execução da Rotina de Serviço;
- execução retorna ao programa (após m).

Mecanismo de Interrupção



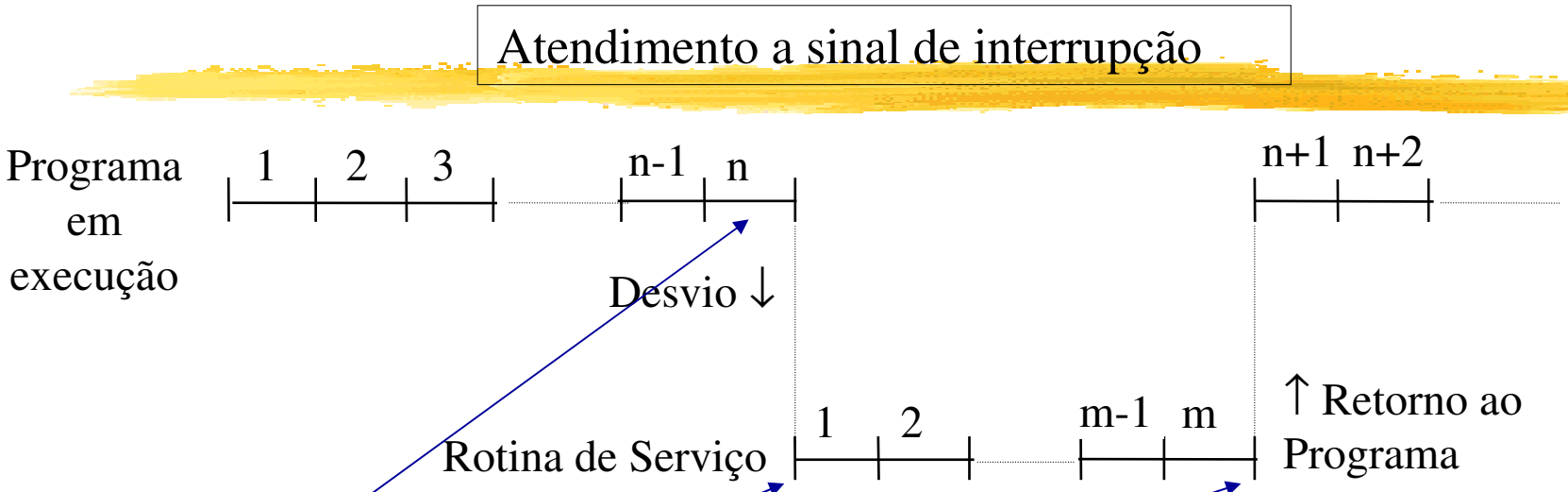
- ocorrência de sinal de interrupção (em n);
- atendimento da interrupção via execução da Rotina de Serviço;
- execução retorna ao programa (após m).

Mecanismo de Interrupção



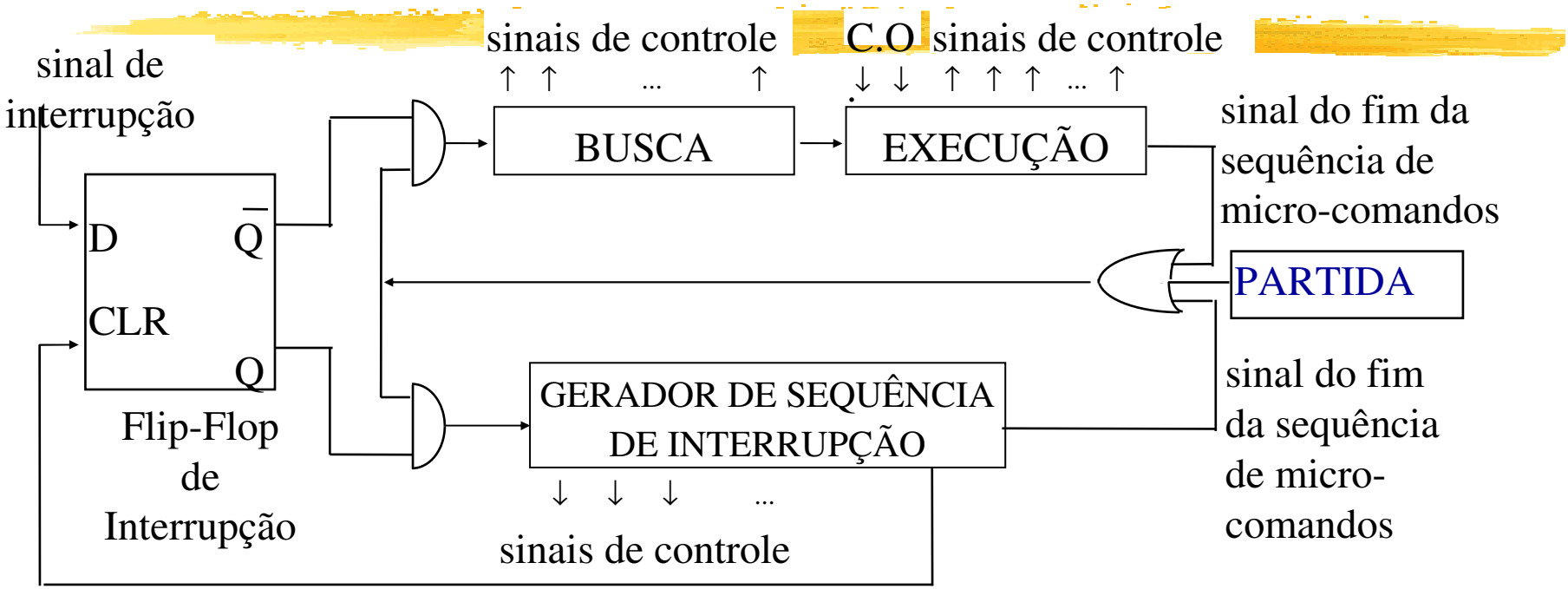
- ocorrência de sinal de interrupção (em n);
- atendimento da interrupção via execução da Rotina de Serviço;
- execução retorna ao programa (após m).

Mecanismo de Interrupção



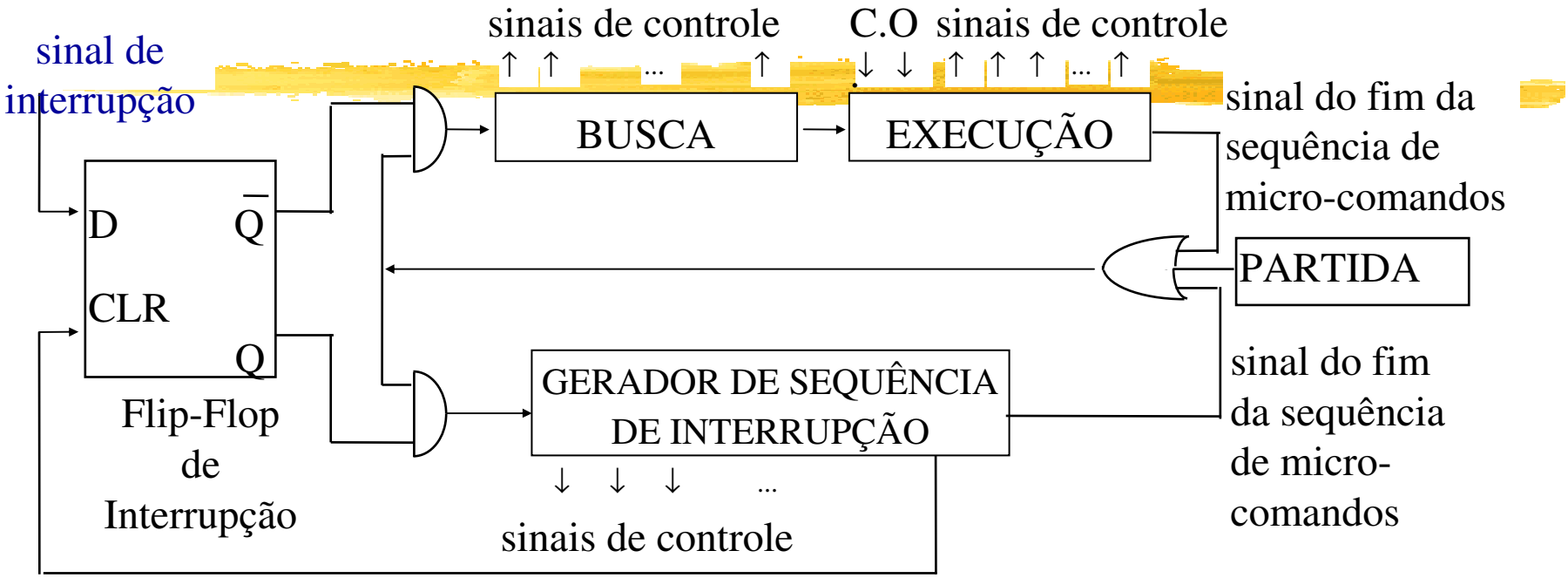
- ocorrência de sinal de interrupção (em n);
- atendimento da interrupção via execução da Rotina de Serviço;
- execução retorna ao programa (após m).

Arquitetura simplificada de controlador com tratamento de interrupção



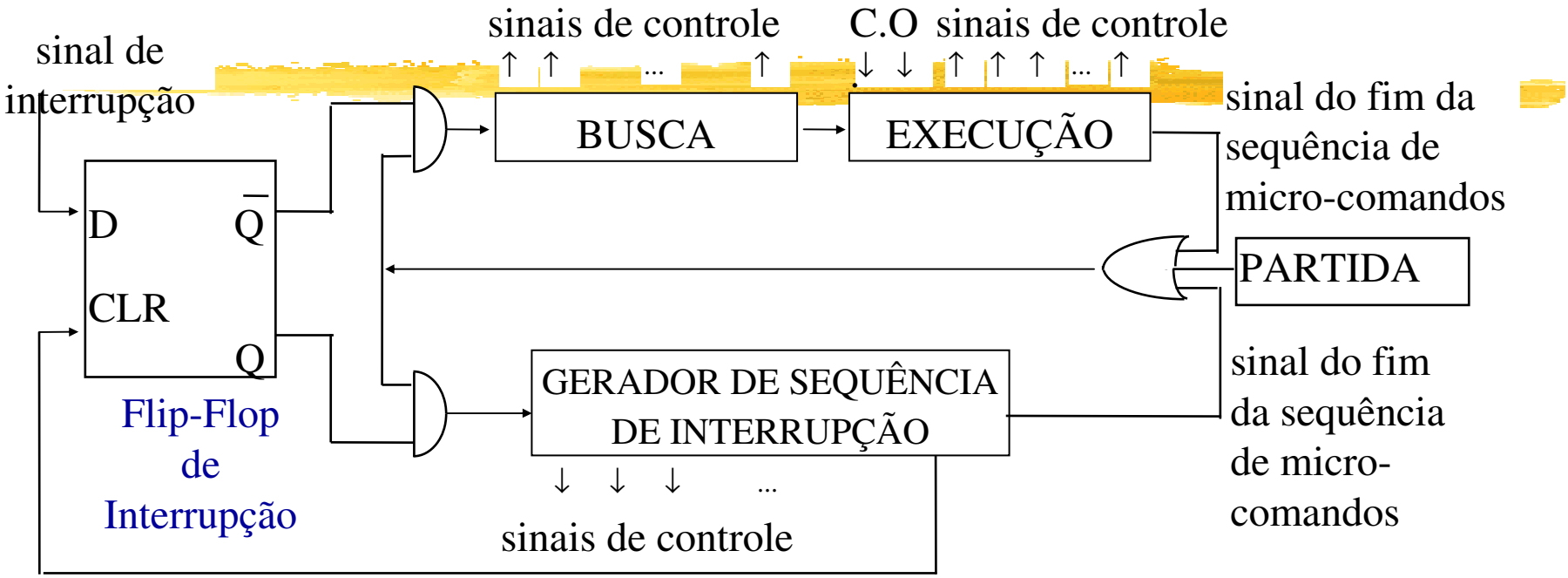
- dada a partida (**PARTIDA**) na máquina, o controlador gerará repetitivamente os controles da busca da instrução seguidos dos controles de execução da instrução específica, que dependem do Código de Operação (C.O.) da instrução.

Arquitetura simplificada de controlador com tratamento de interrupção



- gerado um sinal de interrupção, o *Flip-Flop de Interrupção* será setado e o controlador, ao iniciar um novo ciclo de instrução, passará a gerar os controles do *Gerador de Sequência de Interrupção*. O efeito do sinal de interrupção **só se inicia ao fim da execução da instrução corrente**.

Arquitetura simplificada de controlador com tratamento de interrupção



- após a execução da sequência de interrupção, a CPU retoma o seu modo de operação normal, visto que o Gerador de Sequência de Interrupção inclui, como último sinal de controle da sequência gerada, o microcomando para desativar o Flip-Flop de Interrupção.

Execução de programa com interrupção

CICLO DE INSTRUÇÃO COM TRATAMENTO DE SINAL DE INTERRUPÇÃO

busca de instrução
execução de instrução
IF sinal de interrupção
THEN sequência de interrupção

Interrupção é atendida **após** o término da execução da instrução corrente.

7.2 Interrupção – sequência de sinais

SINAIS DE CONTROLE DO GERADOR DE SEQUÊNCIA DE INTERRUPÇÃO

Atendimento de interrupção: os microcomandos (sinais de controle) gerados pelo Gerador de Sequência de Interrupção devem ativar microoperações para:

- armazenar o conteúdo do PC para posterior retorno à execução interrompida;
- armazenar o "estado" do programa, PSW, para restaurar, no retorno, a situação anterior à interrupção;
- desviar o controle de execução para a rotina de serviço.

Os microcomandos da sequência de interrupção equivalem aos microcomandos gerados na **fase de execução** das instruções:

```
PUSH  PC      ; armazena o conteúdo de PC
PUSH  PSW     ; armazena o estado corrente do processador
JUMP  RS      ; PC ← endereço da rotina de serviço (RS)
```

Interrupção – sequência de sinais cont.

ARMAZENAMENTO DO ESTADO CORRENTE ("FLAGS")

```
.....
        LOOP:   ADD   R1, R2  <------- interrupção
                JZ   LOOP
                ....
```

atendimento

```
PUSH   PC
PUSH   PSW
```

Interrupção – sequência de sinais cont.

RETORNO DA INTERRUPÇÃO

- no atendimento da interrupção, o conteúdo dos registradores PC e PSW são armazenados na pilha;
- no retorno da interrupção, as microoperações executadas pela instrução RTI deverão corresponder à fase de execução das seguintes instruções:

```
POP    PSW    ; restaura o estado do programa
POP    PC     ; retorno ao programa
```

- devemos notar que o retorno ao programa interrompido é *voluntário* (assim como em uma subrotina) por parte da rotina de serviço.

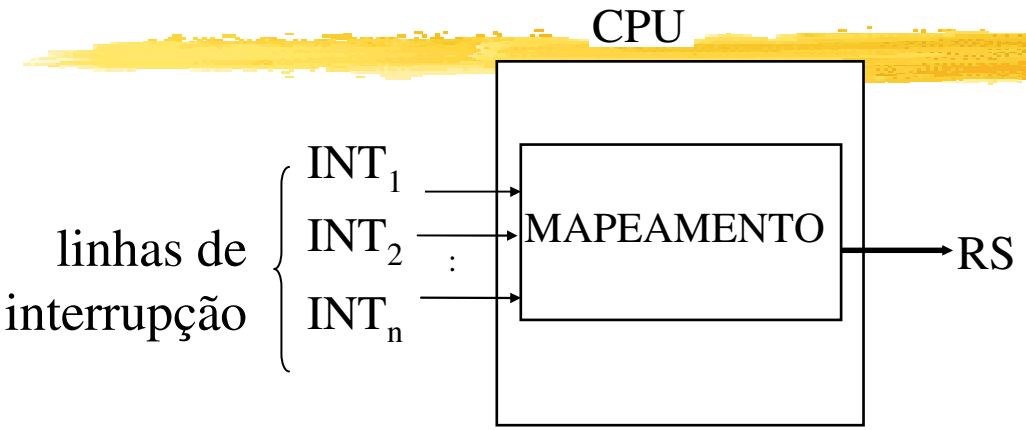
7.3 Endereço da Rotina de Interrupção

a cada evento ou dispositivo gerador de interrupção está associada uma linha de interrupção e existe uma rotina de serviço para tratar a interrupção.

geração do endereço

- por mapeamento interno à CPU
- por mapeamento externo à CPU
- por mapeamento misto
- por mecanismo de polling

Endereço da Rotina de Interrupção
mapeamento interno à CPU

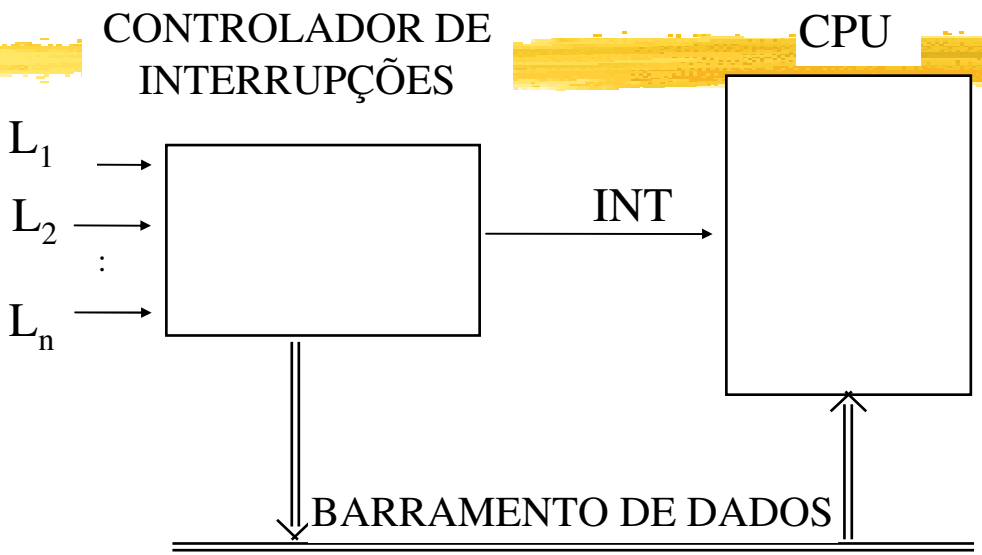


- o Gerador de Sequência de Interrupções deve prover sinais de controle para realizar o desvio para a rotina: (as microoperações ativadas pelo Gerador de Sequência de Interrupções equivalem à execução das seguintes instruções)

  PUSH PC
  PUSH PSW
  JUMP RS ; **R**otina de **S**erviço

- o *endereço de interrupção* associado a cada pino de interrupção é fornecido pelo manual do processador. O programador deve instalar a rotina de serviço de interrupção no *endereço de interrupção* associado ao pino de interrupção onde foi ligado o dispositivo.

Endereço da Rotina de Interrupção
mapeamento externo à CPU



- o Gerador de Sequência de Interrupção deve prover sinais de controle para realizar a leitura do *endereço de interrupção* gerado pelo Controlador de Interrupções: (microperações equivalentes às instruções)

```
PUSH  PC
PUSH  PSW
IN    CONTRL    ; lê o endereço dado pelo controlador de interrupções
JUMP  RS        ; rotina de serviço (no caso depende de CONTRL)
```

Endereço da Rotina de Interrupção
mapeamento misto e polling

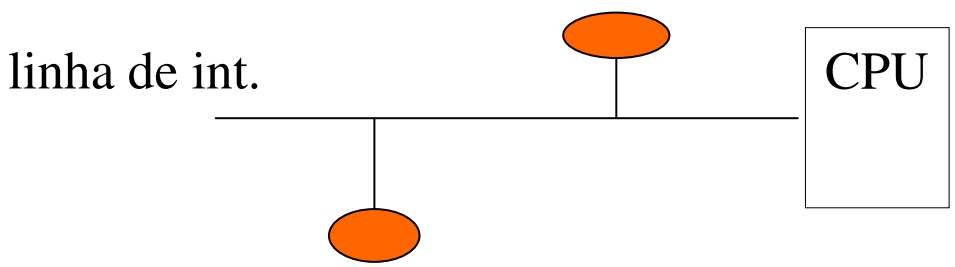
Misto

arquiteturas normalmente oferecem a abordagem externa, mantendo porém um segundo pino de interrupção para mapeamento interno.

Polling

lido o "estado" da interface do dispositivo, a rotina de serviço utiliza o mecanismo de "*polling*" que equivale a um CASE em linguagem de alto nível

```
RS:     IN      STATUS1
        Jcond   RS1
        IN      STATUS2
        Jcond   RS2
                :
                :
        IN      STATUSn
        Jcond   RSn
```

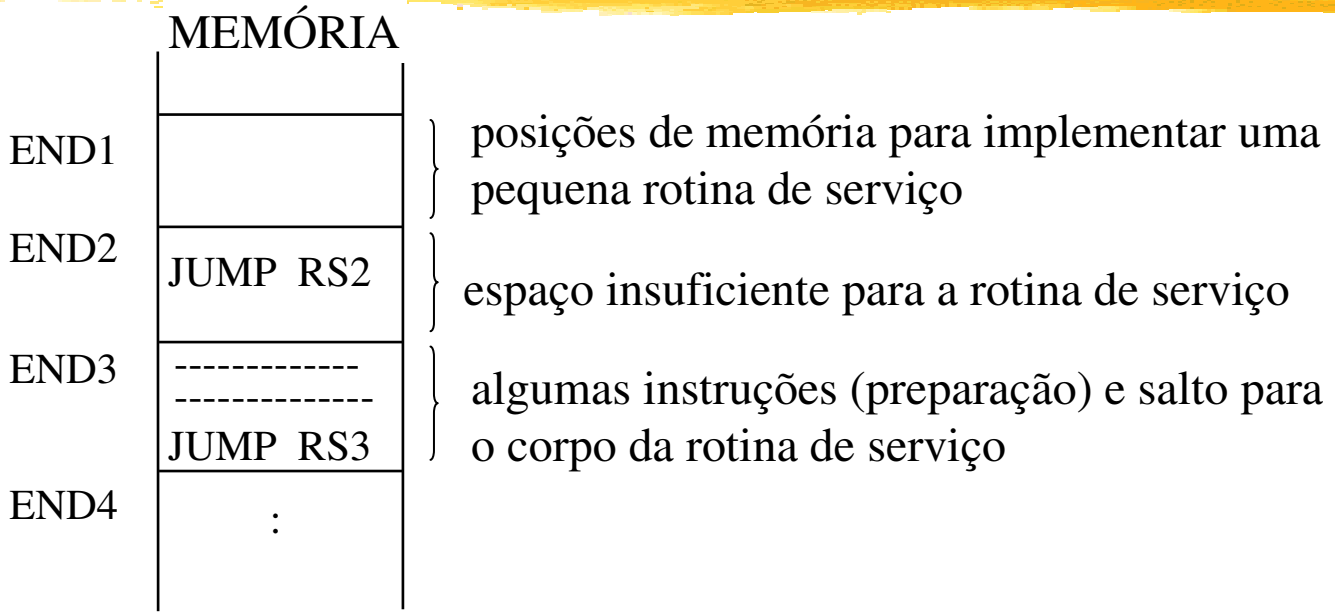


7.4 Organização dos endereços de interrupção

apresentamos aqui como os endereços **mapeados** são organizados

- em posições não contíguas de memória
- em vetores
- em vetores com definição de estado

Organização endereços de interrupção em posições não contíguas de memória



Nesta organização o mapeamento gera *endereços de interrupção* para posições não contíguas de memória existindo, desta forma, um pequeno conjunto de palavras de memória entre os *endereços de interrupção* que será utilizado pela rotina de serviço de interrupção ou para uma instrução de desvio para esta rotina.

Organização endereços de interrupção em posições não contíguas de memória – cont.

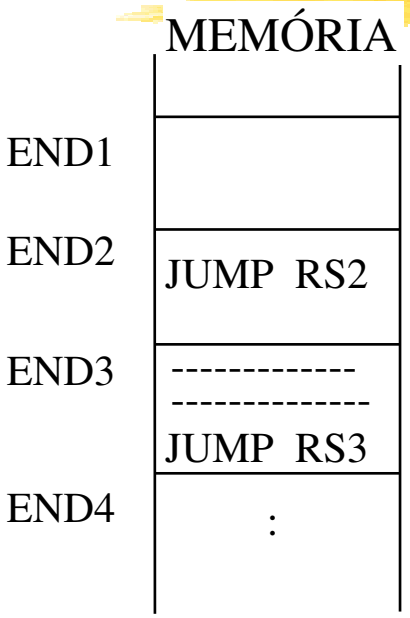
SINAIS DO GERADOR DE SEQUÊNCIA DE INTERRUPÇÃO

Neste caso as microoperações ativadas pelo Gerador de Sequência de Interrupção equivalem às microoperações das fases de execução das seguintes instruções:

```
PUSH   PC
PUSH   PSW
JUMP   ENDi            ; PC ← ENDi
```

sendo ENDi o endereço gerado pelo mapeamento da CPU.

Organização endereços de interrupção em posições não contíguas de memória – cont.



Exemplo de instalação das rotinas de serviço:

```
ORG     END1
--------------
--------------
RTI
ORG     END2
JUMP    RS2
ORG     END3
--------------
--------------
JUMP    RS3
```

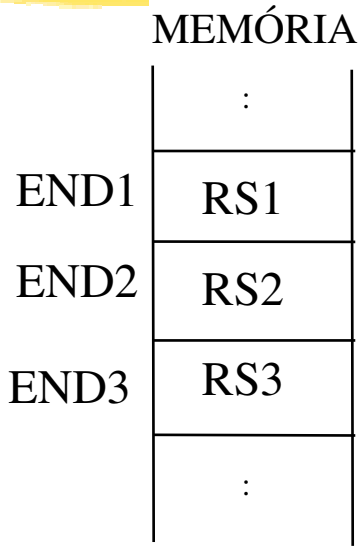
instruções da rotina de serviço de interrupção RS1

desvio para rotina RS2

parte do código da rotina aqui e parte a partir da posição RS3

ORG – é uma pseudo-instrução que define o endereço da próxima posição

Organização endereços de interrupção em vetores (interrupção vetorizada)



END_i - endereço de interrupção associado à linha i

RS_i - endereço da rotina de serviço associado à linha i

Nesta organização, utilizada pela maioria das arquiteturas modernas, o mapeamento gera endereços em posições contíguas de memória. Isto implica não haver espaço para a instalação da rotina de serviço.

Os *endereços de interrupção*, gerados pelo mapeamento, devem conter apenas os **endereços das rotinas de serviço** apropriadas, não havendo espaço sequer para a instrução de salto (JUMP), que é implícita.

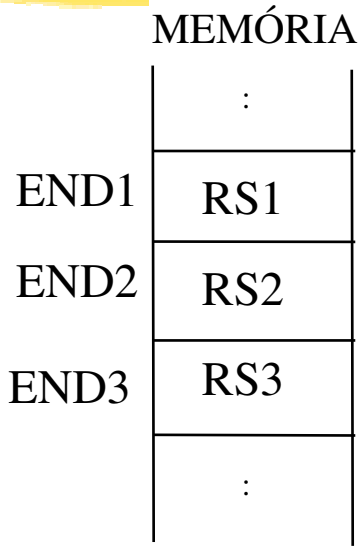
Organização endereços de interrupção em vetores (interrupção vetorizada) – cont.

SINAIS DO GERADOR DE SEQUÊNCIA DE INTERRUPÇÃO

Para este tipo de organização, as microoperações ativadas pelo Gerador de Sequência de Interrupção equivalem às microoperações das fases de execução das seguintes instruções:

PUSH PC
PUSH PSW
JUMP (ENDi) ; PC ← (ENDi) ≡ RSi

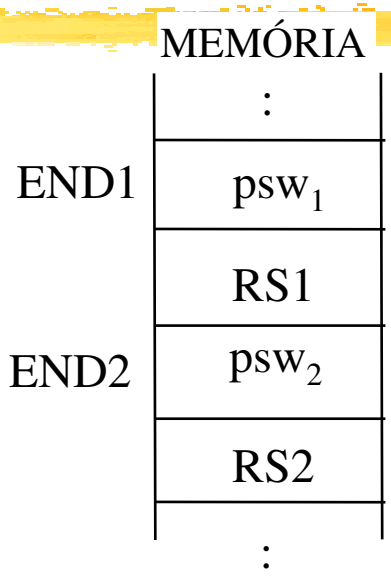
Organização endereços de interrupção em vetores (interrupção vetorizada) – cont.



Exemplo de instalação das rotinas de serviço:

```
        ORG     END1
        ADR     RS1     ; pseudo-instr.: posição =  end.RS1
        ORG     END2
        ADR     RS2
        ORG     END3
        ADR     RS3
        ------------------
        ORG     5700    ; exemplo de endereço
RS1:    ------------------
        ------------------
        RTI
        ORG     2500
RS2:    ------------------
        RTI
        ORG     3240
RS3:    ------------------
        RTI
```

Organização endereços de interrupção em vetores com PSW (estado)



psw_i - "estado" da rotina de serviço associado à linha i

RS_i - endereço da rotina de serviço associado à linha i

Em algumas arquiteturas, além do endereço da rotina de serviço, podemos armazenar o *estado da rotina de serviço* pswi no *Vetor de Interrupção*.

O *estado da rotina de serviço* é carregado no PSW no atendimento de interrupção definindo o "estado" para a execução da rotina de serviço.

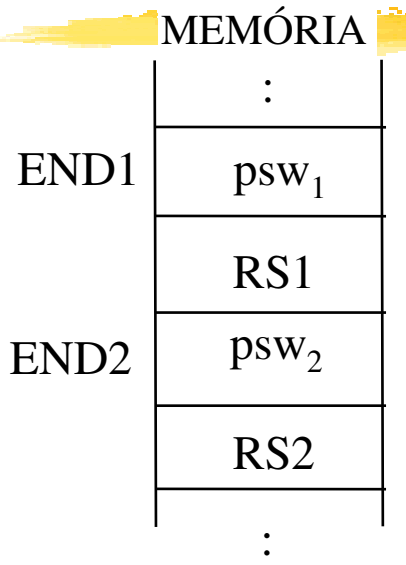
Organização endereços de interrupção em vetores com PSW (estado) cont.

SINAIS DO GERADOR DE SEQUÊNCIA DE INTERRUPÇÃO

As microoperações ativadas pelo gerador de interrupção equivalem às das fases de execução das seguintes instruções:

```
PUSH   PC
PUSH   PSW
MOV    ENDi, PSW        ; PSW ← (ENDi) ≡ pswi
JUMP   (ENDi + 1)       ; PC ← (ENDi + 1) ≡ RSi
```

Organização endereços de interrupção em vetores com PSW (estado) cont.



Exemplo de instalação das rotinas de serviço:

```
ORG   END1
DW    psw1     ; psw1 é um valor
ADR   RS1
ORG   END2
DW    psw2
ADR   RS2
```

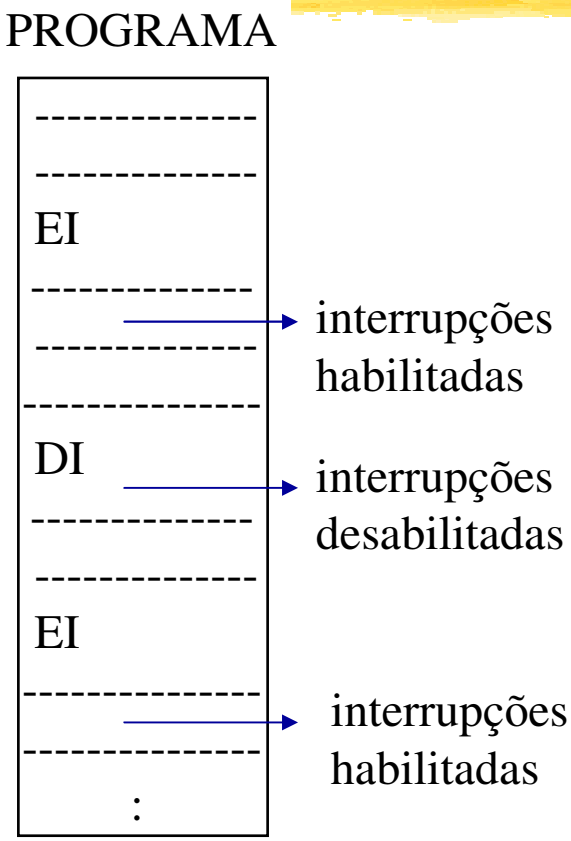
DW – define word: uma pseudo-instrução, indica ao tradutor que a posição deve ser carregada (para execução) com o valor definido.

7.5 Atendimento de interrupções
controle do atendimento

- por flag de interrupção
- por mascaramento de linhas de interrupção
    - com controlador de interrupção interno à CPU
    - com controlador de interrupção externo à CPU
- por mascaramento por nível de prioridade

Atendimento de interrupções - cont.

controle do atendimento por flag de interrupção



Através do *Flag de Interrupção* todo o sistema de interrupção é habilitado ou desabilitado.

O controle do sistema de interrupção faz-se por meio de instruções que manipulam o valor do *Flag de Interrupção*: **EI** ("Enable Interrupt")
DI ("Disable Interrupt").

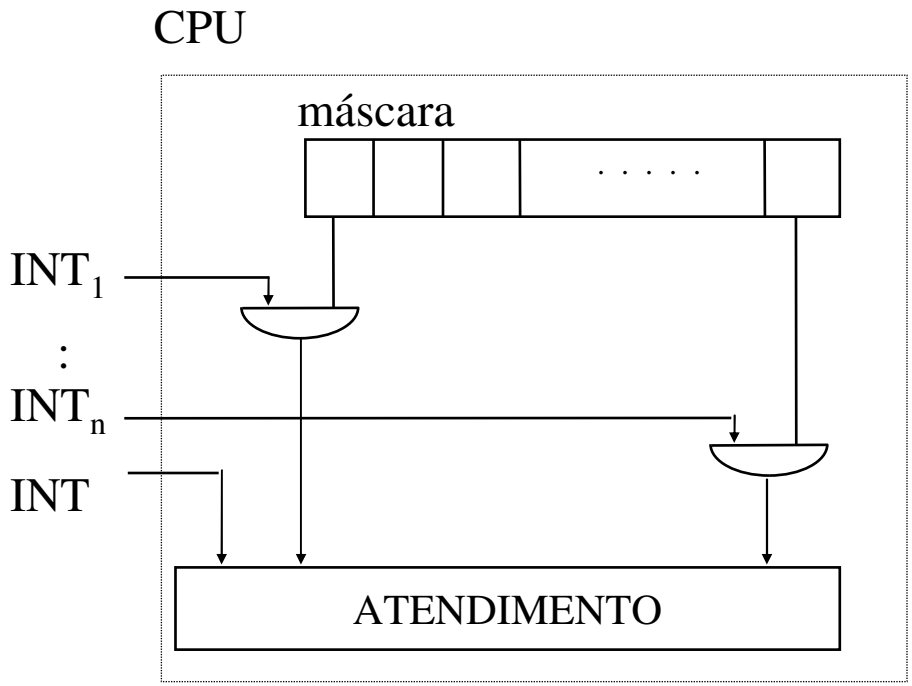
EI habilita o atendimento de sinais de interrupção **após** a execução da **instrução seguinte** à instrução EI.

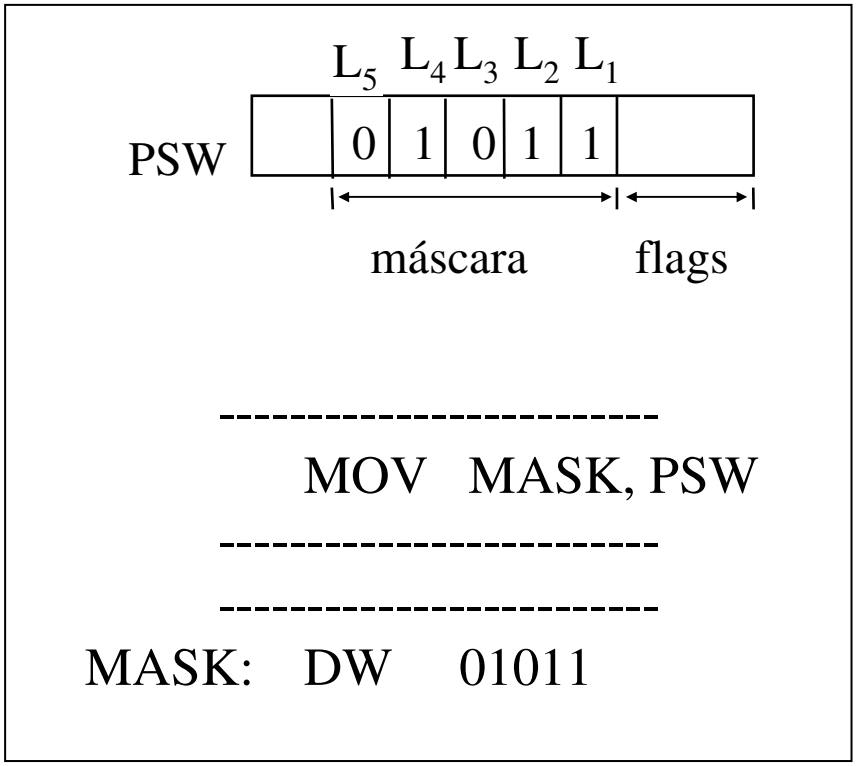
DI desabilita o atendimento de sinais de interrupção **imediatamente** após a sua execução.

Atendimento de interrupções - cont.

controle do atendimento por mascaramento

Controlador de Interrupção interno à CPU



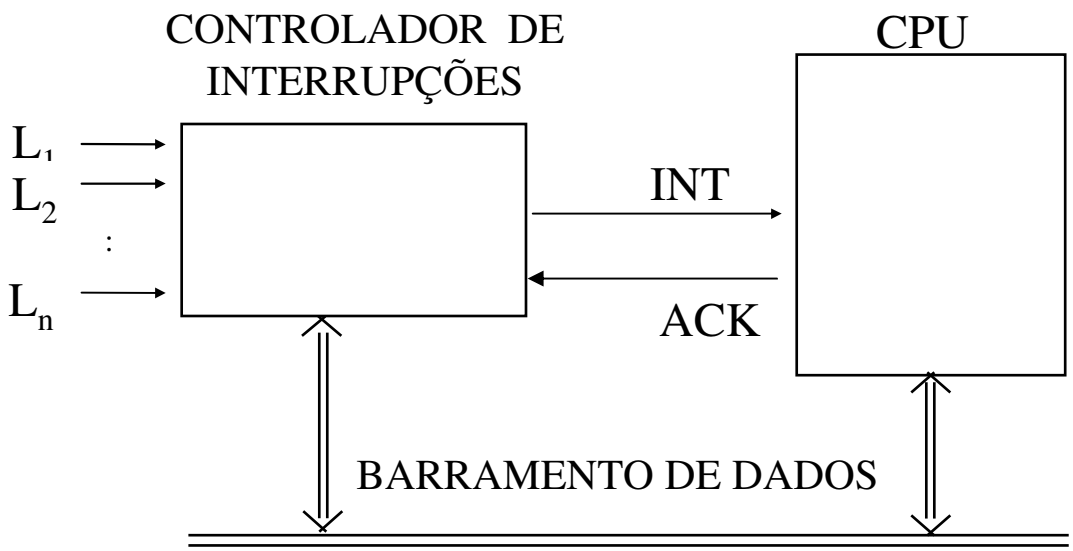


Com o carregamento automático do estado no PSW pela sequência de interrupção, o psw_i da rotina de serviço determina a máscara durante o processamento da rotina de serviço. Terminada a rotina de serviço o psw original é restaurado.

Atendimento de interrupções - cont.

controle do atendimento por mascaramento

Controlador de Interrupção externo à CPU

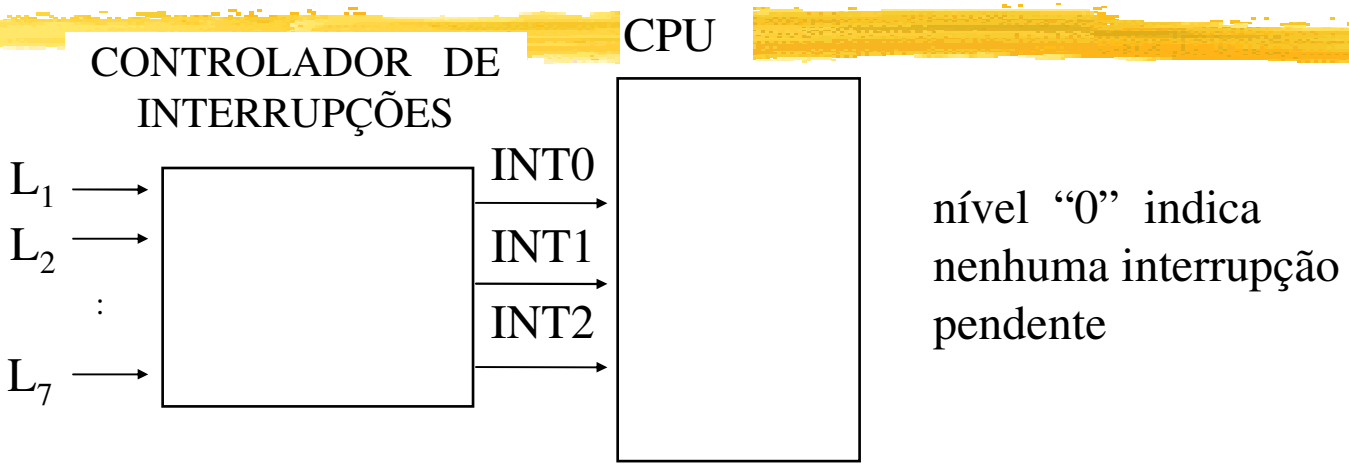


A máscara deve ser, previamente, enviada ao Controlador de Interrupções através de uma instrução de saída (escrita), endereçando o registrador de máscara do Controlador de Interrupções.

Exemplo: MOV #01011, CONTRL ; CONTRL é o endereço do
; reg. de máscara do Controlador de
; Interrupções externo.

Atendimento de interrupções - cont.

controle do atendimento por nível

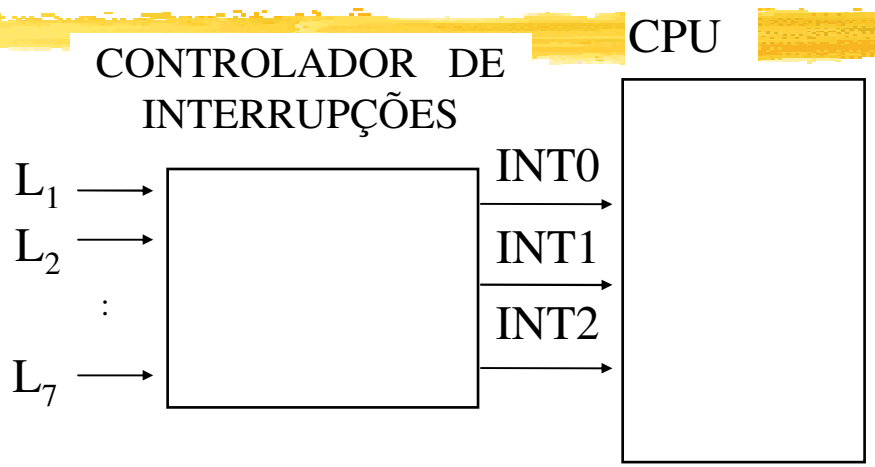


O exemplo mostra uma arquitetura com mascaramento por nível de prioridade, onde a CPU possui 3 pinos de interrupção que recebem o nível de interrupção codificado de 0 a 7.

Este nível de interrupção, neste caso, é gerado por um *Controlador de Interrupção* externo à CPU que controla 7 linhas de interrupção. O nível "0" indica que não existe sinal de interrupção. Os níveis de 1 a 7 estão associados às linhas L1 a L7 de interrupção.

Atendimento de interrupções - cont.

controle do atendimento por nível



Nível "0" indica nenhuma interrupção pendente

O mecanismo é implementado atribuindo um nível à própria CPU, ou seja, ao programa em execução. Sinais de interrupção cujo nível seja *igual* ou *inferior* ao nível da CPU não serão atendidos. Com este mecanismo, o *nível da CPU* é utilizado como máscara de interrupção.

O *nível da CPU* é também definido como um campo do PSW:

```
        -------
         MOV   NIVEL, PSW
        -------
NIVEL:  DW    101
```

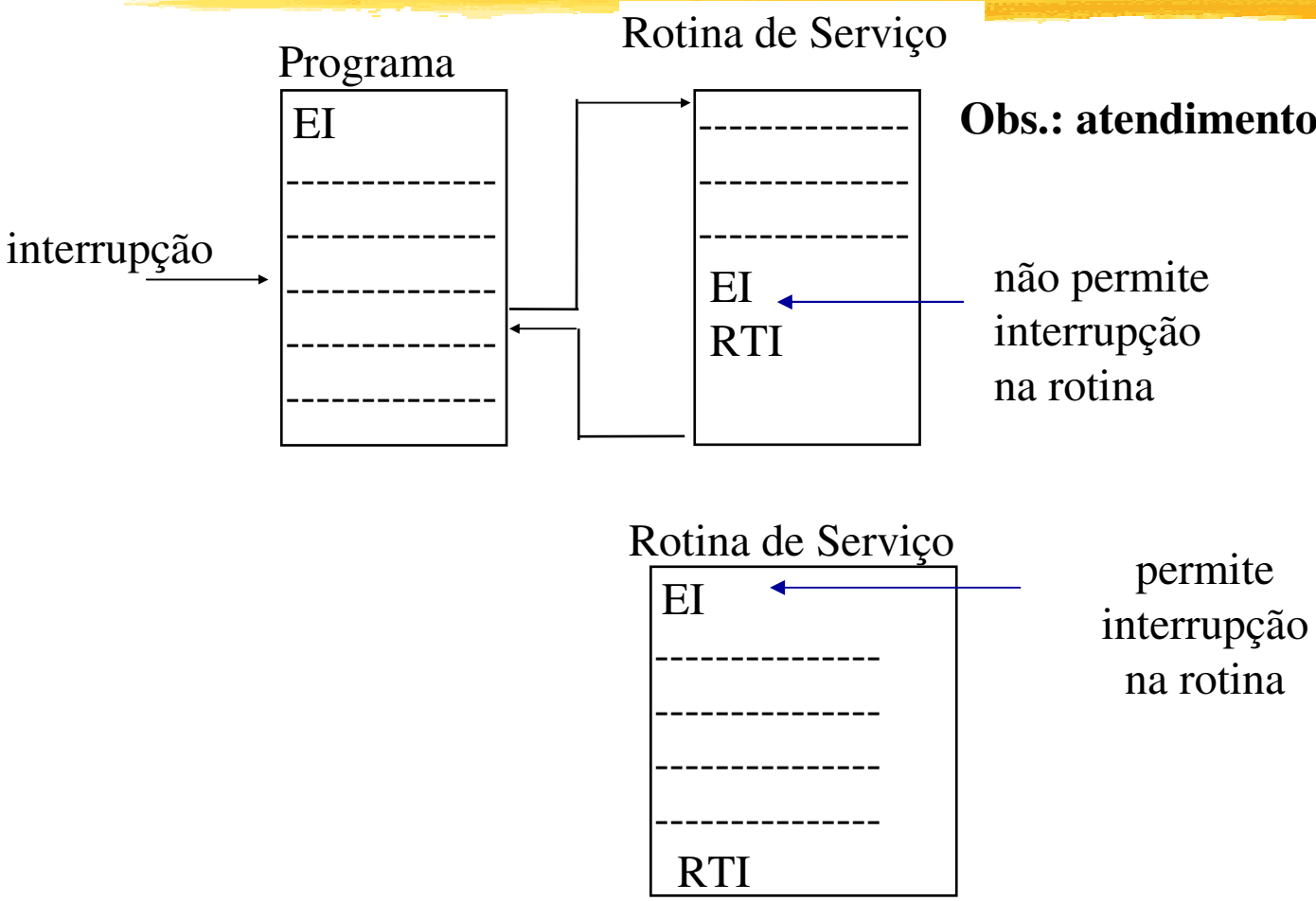
7.6 Interrupção de rotina de serviço

Durante a execução pela CPU de uma rotina de serviço de interrupção, pode ocorrer outro sinal de interrupção na mesma linha associada à rotina de interrupção ou em outra linha de interrupção.

O atendimento a este sinal de interrupção será controlado pelos mesmos mecanismos de habilitação e desabilitação de interrupções vistos (flag de interrupção, mascaramento e nível).

Interrupção de rotina de serviço - cont.

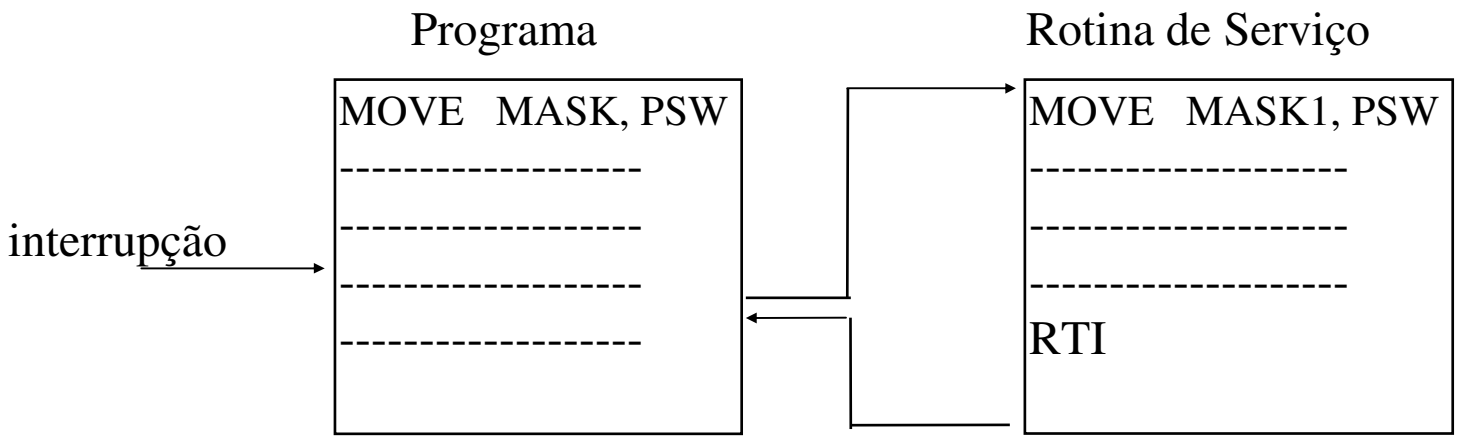
Via flag de interrupção



Obs.: atendimento de int. "gera" DI

Interrupção de rotina de serviço - cont.

Via máscara e máscara de nível usando vetor de interrupção.
Vetor sem psw

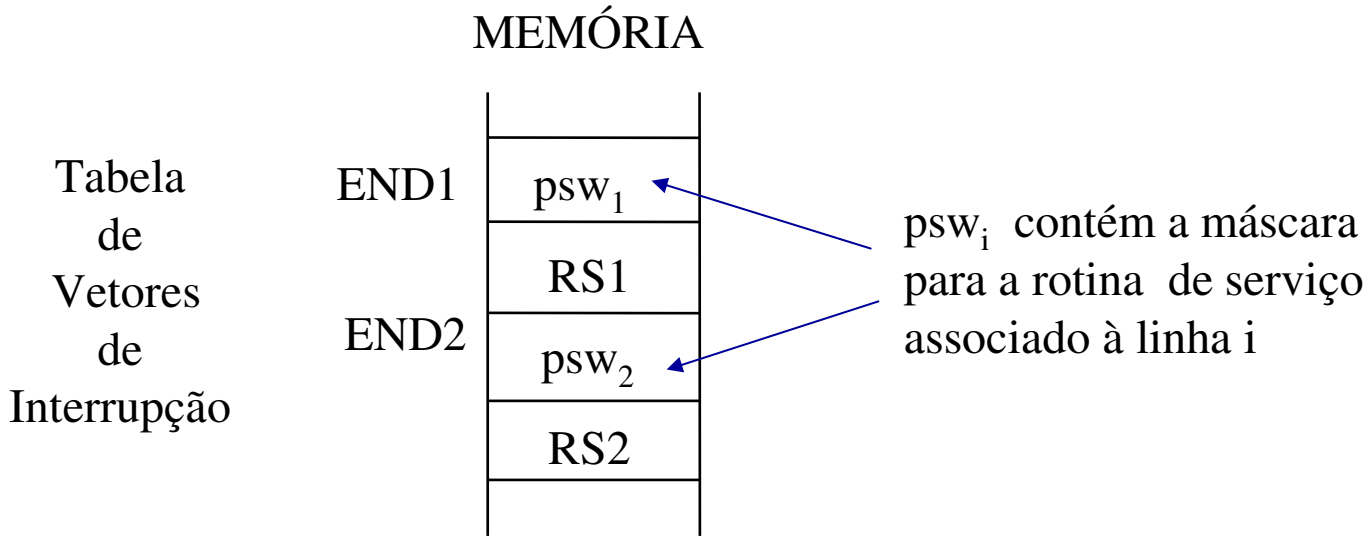


MASK - define os sinais de interrupção habilitados durante a execução de PROGRAMA.

MASK1- define os sinais de interrupção habilitados durante a execução da ROTINA DE SERVIÇO

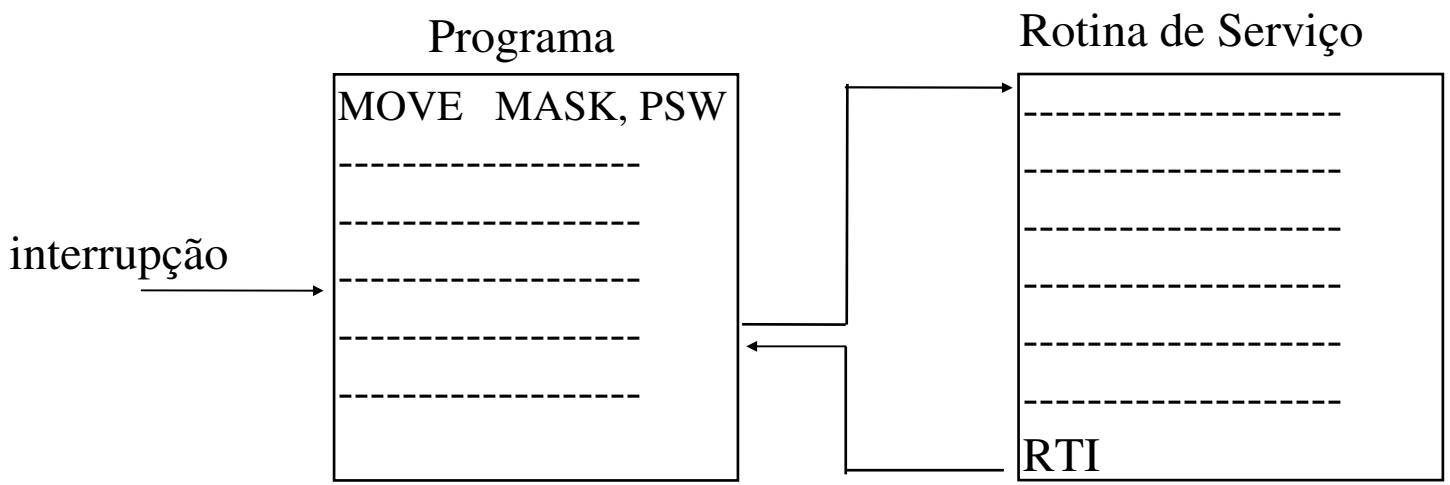
Interrupção de rotina de serviço - cont.

Via máscara e máscara de nível usando vetor de interrupção.
Vetor com psw



Interrupção de rotina de serviço - cont.

Via máscara e máscara de nível usando vetor de interrupção.
Vetor com psw – cont.



MASK - define os sinais de interrupção habilitados durante a execução de PROGRAMA.

MASK1 - que define os sinais de interrupção habilitados durante a execução da ROTINA DE SERVIÇO, está definido no Vetor de Interrupção e é carregado automaticamente.

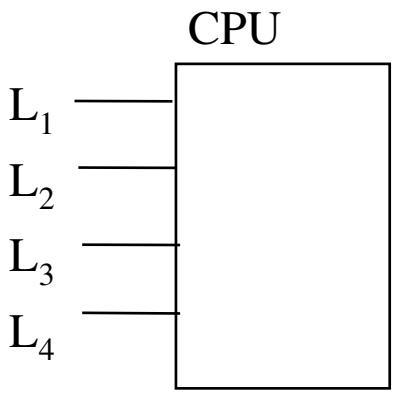
7.7 Prioridade no atendimento de Interrupção (ocorrência "simultânea" de interrupções)

- Fixa
- por Varredura
- Prioridade para dispositivos conectados à mesma linha

Prioridade no atendimento de Interrupção (cont.)

Prioridade Fixa

Em arquiteturas com o controlador de interrupções interno à CPU, a prioridade de atendimento de interrupções está ligada às linhas de interrupção e é **fixa**, não havendo a possibilidade de modificar a prioridade entre as linhas. Desta forma, os dispositivos deverão estar ligados às linhas de interrupção conforme a sua prioridade de atendimento.



PRIORIDADE

$L_1 > L_2 > L_3 > L_4$

o manual do processador fornecerá a informação sobre a prioridade das linhas de interrupção.

Prioridade no atendimento de Interrupção (cont.)

Prioridade por Varredura

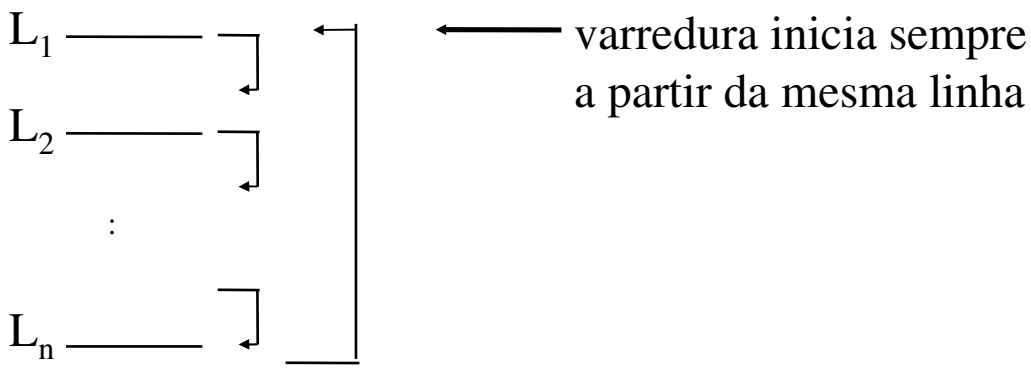
Quando a arquitetura necessita de um controlador de interrupções externo, geralmente é possível programar este controlador para duas diferentes situações de controle de prioridade:

- prioridade fixa associada às linhas de interrupção (analogamente ao anterior)
- linhas de interrupção sem prioridade (necessário realizar uma *Varredura* das linhas).

O mecanismo de *Varredura* é implementado por um hardware específico que percorre as linhas de interrupção até identificar a linha que gerou a interrupção.

Prioridade no atendimento de Interrupção (cont.)

Prioridade por Varredura – a partir da 1. linha



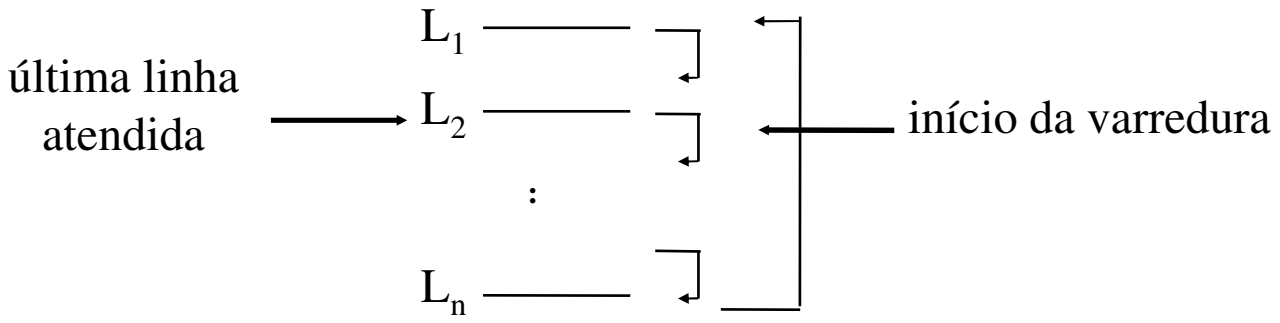
Neste caso, a ordem de varredura estabelece a prioridade.

Observemos que com a varredura a partir da primeira linha temos uma prioridade implícita onde :

PRIORIDADE L1 > L2 > L3 > ... > Ln

Prioridade no atendimento de Interrupção (cont.)

Prioridade por Varredura – a partir da linha da última interrupção



Neste caso, não existe prioridade entre as linhas de interrupção.

Iniciando a varredura a partir da última linha em que ocorreu o sinal de interrupção temos, a cada atendimento, uma prioridade diferente associada às linhas de interrupção.

No exemplo: PRIORIDADE L2 > L3 > ... > Ln > L1

Prioridade no atendimento de Interrupção (cont.)

Prioridade entre dispositivos conectados à mesma linha

Quando temos dois ou mais dispositivos ligados à mesma linha de interrupção é necessário implementar um *Polling* na rotina de serviço de interrupção, de forma a identificar qual dispositivo gerou esta interrupção.

Desta forma, a sequência de inspeção dos dispositivos na rotina de serviço, implicará na implementação, em software, de uma varredura que se inicia sempre a partir de uma mesma linha de interrupção.

Prioridade: L1 > L2 > ... > Ln

```
RS:     IN      status_1
        Jcond   RS1
        IN      status_2
        Jcond   RS2
          :
          :
        IN      status_n
        Jcond   RSn
          :
```

7.8 Exceções Internas

Exceções internas são geradas pela própria CPU não estando assim associadas a eventos externos.

São provocadas por condições de exceção durante a execução de instruções pela CPU como por exemplo:

- overflow,
- divisão por zero,
- instrução inválida ou não existente,
- tentativa de acesso a posições privilegiadas de memória,

etc.

No atendimento das exceções internas também existem endereços associados às exceções na Tabela de Vetores de Exceção.

A geração do endereço da exceção (endereço na tabela) é interna à CPU.

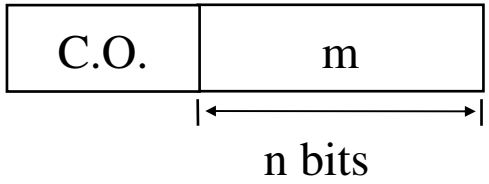
7.9 Exceções programadas

Usadas como apoio às fases de desenvolvimento e testes de programas.

São instruções que emulam a ocorrência de interrupções. Podem ser consideradas *interrupções síncronas*, pois sua ocorrência é determinada pelos pontos do programa onde as instruções estão colocadas.

Também utilizam endereços na Tabela de Vetores de Exceção.

A geração do endereço da exceção é explicitado na própria instrução:



com este formato é possível distinguir 2^n endereços diferentes na Tabela de Vetores de Exceção.

7.10 Questões explicativas

QUESTÃO 1 - Como acessar as rotinas de serviço uma vez que não existe a "chamada" com o endereço da rotina de serviço ?

RESPOSTA

Sensores estão ligados a "linhas de interrupção" que estão conectadas aos "pinos de interrupção" da pastilha do processador. Por construção interna, o sinal em um pino de interrupção gera, no processador, um salto equivalente a uma instrução JUMP para uma posição específica da memória.

Existe desta forma, por construção, um mapeamento entre uma "linha de interrupção" e uma posição de memória. Estas posições de memória (associadas a interrupções) são chamadas de "*reservadas*", pois só devem ser utilizadas para o atendimento de interrupção.

Questões explicativas

QUESTÃO 2 - O mapeamento entre as linhas de interrupção e posições reservadas de memória é sempre um para um ?

RESPOSTA

Nem sempre; nas arquiteturas modernas, uma linha de interrupção pode ser associada a diversas posições reservadas de memória para o atendimento de diferentes eventos geradores de sinal de interrupção.

Neste caso, o mapeamento deve ser feito externamente por um hardware, denominado Controlador de Interrupções, que envia ao processador o endereço de memória onde deve estar a rotina de serviço correspondente ao evento que gerou o sinal de interrupção.

Questões explicativas

QUESTÃO 3 - O que fazer quando temos mais sensores (geradores de sinais de interrupção) do que linhas de interrupção?

RESPOSTA

Podemos conectar mais de um sensor a cada linha de interrupção; porém, um mapeamento dispositivo-rotina de serviço deve ser efetuado, de forma a garantir que o pedido de interrupção seja atendido pela rotina apropriada.

Uma forma de implementar este mapeamento é, ao início da rotina, realizar a leitura de uma informação enviada pelo dispositivo identificando-se. Neste caso podemos dizer que temos diversas rotinas de serviço acessadas a partir de uma mesma posição de memória; a rotina correta é acionada através de uma construção CASE que identifica o sensor que gerou a interrupção.

Questões explicativas

QUESTÃO 4 - Como estão organizados os endereços de interrupção nas posições reservadas de memória ?

RESPOSTA

As posições reservadas de memória para os endereços de interrupção dependem do processador. Estes endereços de interrupção podem estar em posições não contíguas de memória, havendo espaço para o armazenamento da rotina de serviço; ou então em posições contíguas de memória, onde há espaço somente para indicar um salto para uma outra posição de memória onde a rotina de serviço é instalada.

Questões explicativas

QUESTÃO 5 - Existe a possibilidade de habilitar ou desabilitar interrupções ?

RESPOSTA

Sim, foram descritos diversos mecanismos existentes nas diversas arquiteturas de processadores ou Controladores de Interrupção.

A possibilidade de desabilitar interrupções pode ser muito importante quando, por exemplo, o processador está executando um algoritmo de tempo crítico não devendo, portanto, ser interrompido. Da mesma forma, após a execução deste algoritmo de tempo crítico, deve ser possível habilitar novamente as interrupções.

Questões explicativas

QUESTÃO 6 - Existe interrupção dentro de interrupção, ou seja, uma interrupção pode ocorrer durante a execução de uma rotina de serviço ?

RESPOSTA

Sim, é possível interrupção dentro de interrupção.

Foi discutido como utilizar o mecanismo de habilitação/desabilitação de interrupções para controlar interrupções dentro de rotinas de serviço de interrupções.

Questões explicativas

QUESTÃO 7 - O que ocorre se os sinais em duas ou mais linhas de interrupção estiverem ativos simultaneamente ?

RESPOSTA

Existem diversos mecanismos de Prioridade no Atendimento de Interrupção que foram detalhados.

Estes mecanismos são geralmente implementados em hardware podendo, no entanto, também ser implementados por programa.

Questões explicativas

QUESTÃO 8 - Existem somente exceções geradas externamente à CPU ?

RESPOSTA

Não; existem também as chamadas exceções internas que ocorrem devido a algum problema interno ao processador como a divisão por zero, instrução inválida, etc. Seu tratamento é feito pelo mesmo mecanismo que as interrupções (exceções externas), somente diferindo pelo tipo de evento que dispara a ativação.

Além disso, existem instruções que emulam, de uma certa forma, a ocorrência de interrupções, sendo normalmente utilizadas nas fases de desenvolvimento e testes de programas.

7.11 Exercícios

1. Resolva os exercícios 16 a 23 do livro texto.