CESAR16i

Mecanismo de Interrupção

Prof. Sérgio Cechin

Elementos de Chamadas de Subrotinas

- Mecanismo de chamada
  - Como é realizada a chamada da subrotina?
    - Ex: RAMSES x CESAR
- Endereço de destino
  - Onde está o endereço de destino?
- Salvamento de dados de retorno
  - Quais dados são salvos?
  - Onde são salvos?

O caso “JSR” do CESAR

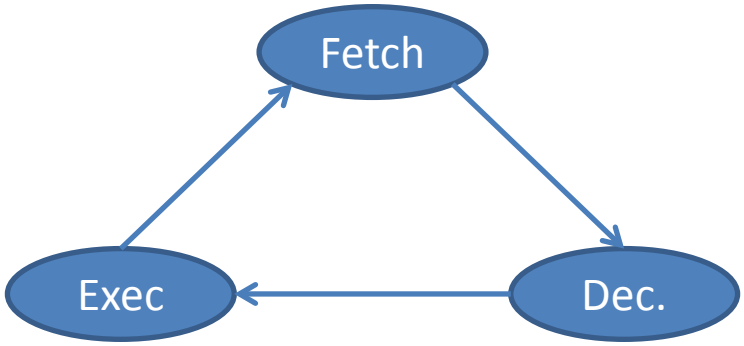
- **Mecanismo de chamada**:
  - Instrução JSR
- **Endereço de destino**:
  - Está no operando indicado pela instrução
- **Salvamento de dados**:
  - Salva o endereço de retorno
  - Coloca-o no topo da pilha

JSR – Operação

Programa Principal

Fluxo natural

```
...
mov     #VAR1,r0
inc     r1
dec     r2
mov     r0,VAR2
...
```



Interrupção

- Apresenta os mesmos elementos de uma chamada de subrotina.
- **Mecanismo de chamada**:
  - Acionamento de **sinal de hardware** no processador
  - O processador só reconhece o pedido de interrupção no final das instruções
- **Endereço de destino**:
  - Determinado pelo **vetor de interrupção**
  - São dois endereços da memória que contém o endereço da rotina de interrupção. É equivalente a:

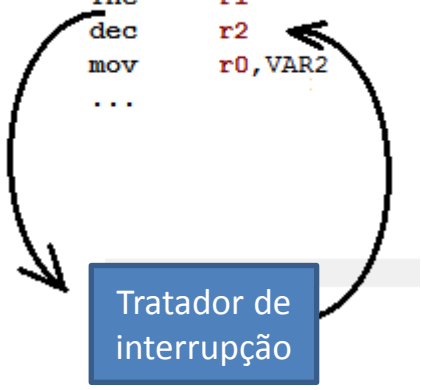
```
MOV     #ROTINA_DE_INTERRUPCAO, R0
JSR     (R0)
```

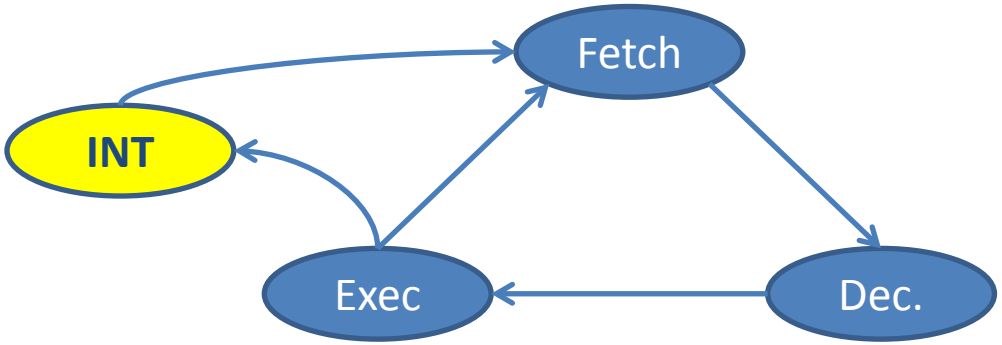
- **Salvamento de dados**:
  - Salva o endereço de retorno e os flags (códigos de condição NZCV)
    - Endereço de retorno: endereço da instrução que seria executada, caso não tivesse sido detectada uma interrupção
  - Coloca esses dados no topo da **pilha**

Interrupção

Programa Principal

```
...
mov     #VAR1,r0
inc     r1
dec     r2
mov     r0,VAR2
...
```





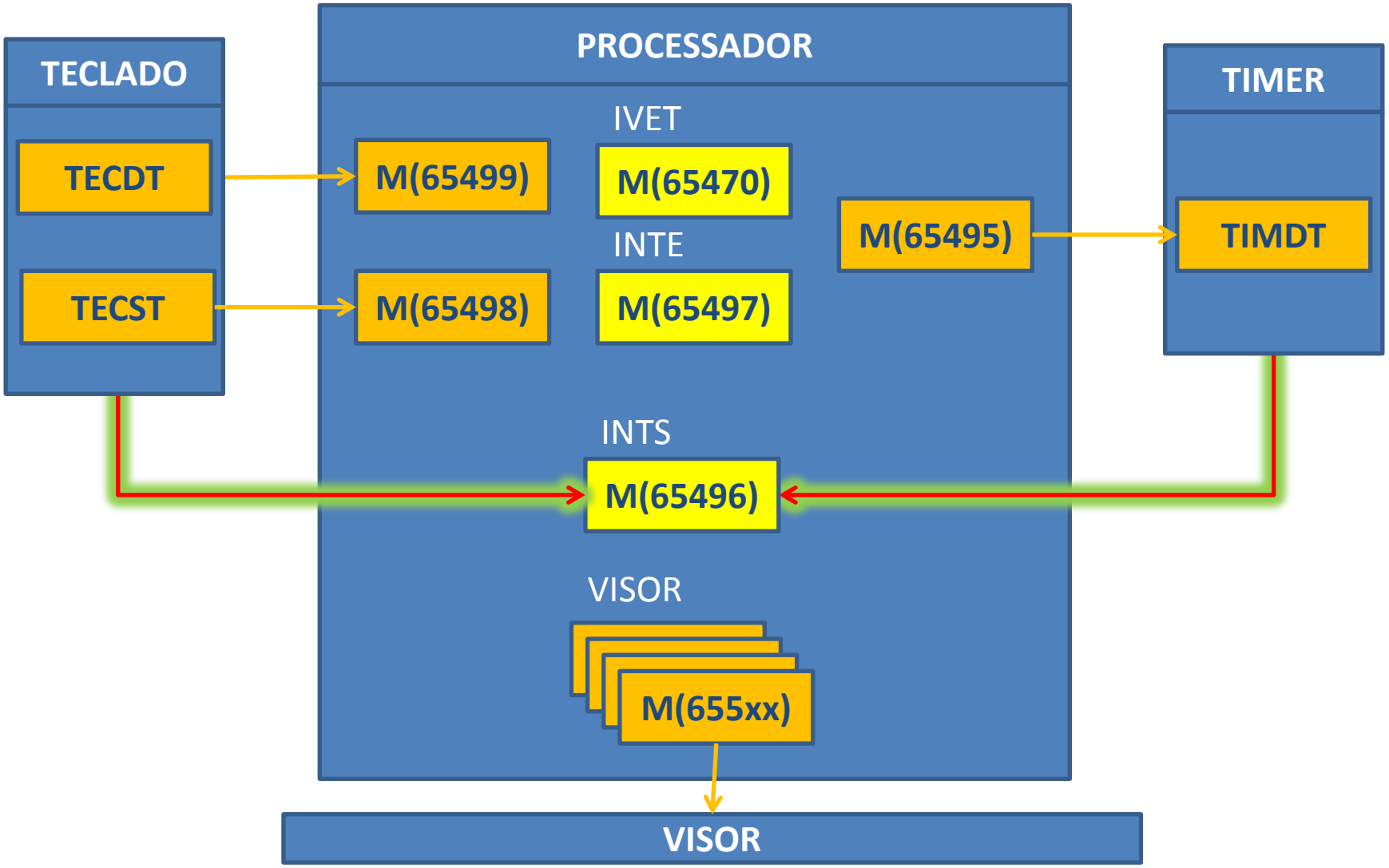
É como se houvesse um JSR entre o “inc” e o “dec”

```
...
mov     #VAR1,r0
inc     r1
jsr     isr
dec     r2
mov     r0,VAR2
...
```

Fontes de Interrupção – CESAR16i

- Geradas por alguns dos periféricos
  - Teclado
  - Timer
- O Visor não gera interrupção

Arquitetura Completa



Endereços Especiais

```
; Área de Periféricos
                org     HFF80
; Acesso em 16 bits
                daw     [31]
IVET:           dw      0       ; Vetor de interrupção
; Acesso em 8 bits
                                ; HFFC0
                dab     [23]    ; Reservado para uso futuro
TIMDT:          db      0       ; Timer Base Time
INTS:           db      0       ; INTERRUPT STATUS
INTE:           db      0       ; INTERRUPT ENABLE
TECST:          db      0       ; Status do teclado
TECDT:          db      0       ; Dado do teclado
VISOR:          dab     [36]    ; Portas de acesso ao visor
```

Registradores Especiais – Interrupção

- IVET – Vetor de interrupção
- INTE – Controle de habilitação da interrupção
- INTS – Controle do estado (*status*) da interrupção

IVET – Vetor de Interrupção

- Contém o endereço da rotina de tratamento das interrupções.
- Dois bytes em formato BigEndian (padrão de 16 bits do CESAR)
- Deve ser carregado antes que as interrupções possam acontecer
  - Escrever o endereço da ISR no IVET
  - ISR = *Interrupt Service Routine*

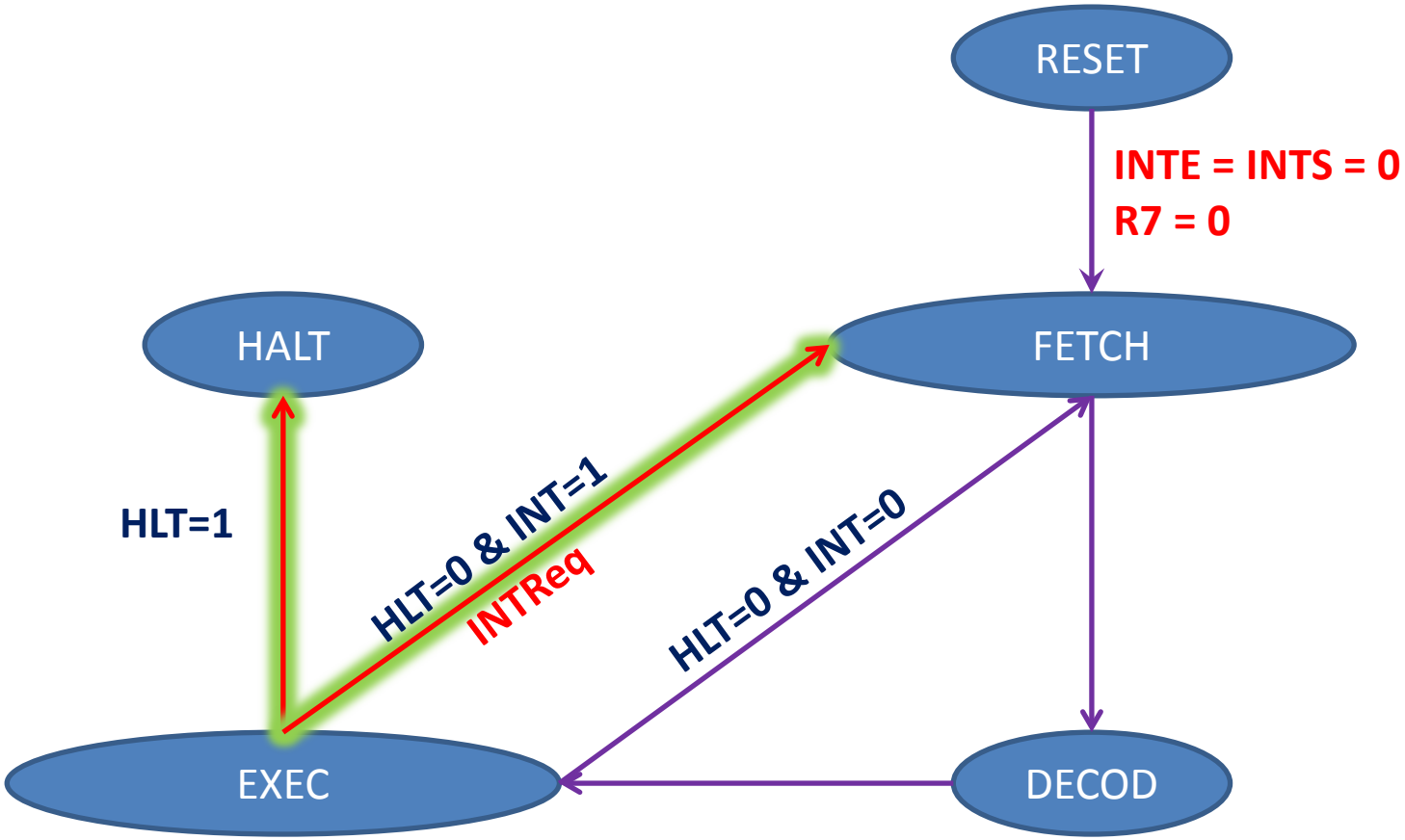
Interrupt Enable – INTE

- Usado para definir quais as interrupções estão habilitadas.
- Esse endereço é lido/escrito no formato “byte”
- Apenas três de seus bits têm significado. Os restantes devem ser mantidos em “0”
  - Bit 7: IE – *Interrupt Enable*
    - Controle geral das interrupções.
  - Bit 1: IEStec – *Interrupt Enable Source 1: Teclado*
    - Indica que a interrupção de teclado está habilitada.
  - Bit 0: IEStim – *Interrupt Enable Source 0: Timer*
    - Indica que a interrupção de timer está habilitada.

Interrupt Status – INTS

- Indica a situação dos pedidos de interrupção.
- Esse endereço é lido/escrito no formato “byte”
- Apenas três de seus bits têm significado. Os restantes devem ser mantidos em “0”
  - Bit 7: IP – *Interrupt Pending*
    - Indica que está executando o tratador de interrupção.
    - Em geral, só deve estar ligado quando executando a ISR
  - Bit 1: IPStec – Interrupt Pending Source 1: Teclado
    - Indica que a interrupção foi gerada pelo acionamento de uma tecla.
  - Bit 0: IPStim – Interrupt Pendind Source 0: Timer
    - Indica que a interrupção foi gerada pelo timer.
- Os bits IPStec e IPStim devem ser usados pela ISR para identificar a origem da interrupção (teclado ou timer)

Máquina de Estados com Interrupção



HLT: executada uma instrução de HLT
INT: reconhecida uma interrupção

Condições e Ações

- Condições
  - INT= IE && ((IEStec && IPStec) || (IEStim && IPStim))
  - HLT= A última instrução executada foi um HLT
- Ações
  - INTReg
    - Coloca R7 na pilha (2 bytes em formato *BigEndian*)
    - Coloca IP e os flags NZCV na pilha (2 bytes)
      - Bit 7 do byte mais significativo de INTS: bit IP (*Interrupt pending*)
      - Bits [3:0] do byte menos significativo: bits NZCV
    - Faz IP=1 e IE=0
    - Coloca IVET (vetor de interrupção) em R7

Retorno da Interrupção

- Um RTS não serve para retornar da interrupção
  - O RTS tira da pilha apenas o endereço de retorno
  - Mas, ao reconhecer uma interrupção, também escreve-se na pilha os bits IP, N, Z, V e C
- É necessário um “retorno” específico para a interrupção: RTI (ReTurn from Interrupt)
  - Retorna IP e os flags NZCV da pilha
  - Retorna R7 da pilha
  - IE = 1

Interrupção no Simulador

- No simulador, as interrupções só estão ativas quando rodando um programa
  - Portanto, durante a execução passo-a-passo não são reconhecidas as interrupções
    - Nesse caso, elas são geradas pelo hardware do periférico mas são “perdidas”, pois o software não é interrompido
- Isso vale para as interrupções de TIMER e de TECLADO

INT – Programação adequada (Inicialização)

- Inicialização do *stack* (R6)
- Programação do timer, se houver (TIMDT)
- Programação das interrupções
  - Vetor de interrupção (IVET)
  - Habilitação das interrupções (INTE)
- Outros:
  - Teclado: limpar o TECST
  - INTS: garantir que não exista nenhuma interrupção ativa

INT – Programação adequada (Operação da ISR)

- Salvar registradores usados na ISR
- Testar o bit adequado do INTS, para identificar a fonte da interrupção
  - Executar processamento adequado
  - Desligar bit do INTS
- Retornar registradores
- RTI (e não RTS)

INT – Interrupção de ações

- A interrupção pode acontecer a qualquer momento
  - Isso pode ser problemático
- Exemplo: soma de variáveis de 32 bits

```
;
; VAR3 = VAR1 + VAR2
mov     VAR1,r0
add     VAR2,r0
mov     r0,VAR3

mov     VAR1+2,r0
adc     r0
add     VAR2+2,r0
mov     r0,VAR3+2
```

O que vai acontecer se:
1. Entrar uma INT entre a **primeira** e a **segunda** instrução
2. E a interrupção não salvar o R0

O que vai acontecer se:
1. Entrar uma INT entre a **terceira** e a **quarta** instrução
2. E a interrupção alterar o valor de VAR1

INT – Grupo de Variáveis

- Ao programar usando interrupção, as variáveis podem ser classificadas em **três grupos**
  - Variáveis usadas apenas no **programa principal (PP)**
    - São as variáveis do programa principal
  - Variáveis usadas apenas na **interrupção (ISR)**
    - São as variáveis da interrupção
  - Variáveis usadas em **ambos**
    - São as variáveis de comunicação entre o PP e a ISR
- As variáveis dos dois primeiros grupos podem ser tratadas da forma tradicional
- Mas, deve-se ter cuidados especiais com as variáveis do terceiro grupo
  - Essas variáveis também são chamadas de “variáveis compartilhadas”

Variáveis Compartilhadas

- Variáveis em registradores
- Variáveis em memória
  - A forma de acesso a essas variáveis determina a correção da operação
  - As formas de acesso são
    - Atômica
      - Não pode ser interrompida
    - Não atômica
      - Pode ser interrompida

Registradores

- O programa principal (PP) e a interrupção (ISR) podem utilizar todos os registradores do processador
- Os registradores R6 e R7 são, sempre, compartilhados entre o PP e a ISR
  - O R7 apenas indica a próxima instrução
  - O R6 indica onde na pilha os dados foram salvos
- Os registradores R0 até R5 podem ter usos diferentes no PP e na ISR
  - Então, deve-se garantir que não sejam “misturados”

Uso dos Registradores

- Existem duas formas de evitar a mistura do conteúdo dos registradores
  - Separar os registradores entre o PP e a ISR
    - Ex: o PP vai usar R0, R1, R2 e R3 e a ISR vai usar R4 e R5
  - Salvar os registradores na memória, sempre que for reconhecida uma interrupção
    - Como não se sabe quando vai ocorrer a interrupção, essa tarefa cabe à própria ISR
    - Deve-se salvar no início da ISR e retornar no final da ISR
- Salvamento dos registradores na memória
  - Em área de memória específica para essa finalidade
  - Na pilha

Acesso à Variáveis na Memória (Acesso atômico)

- A interrupção NÃO CONSEGUE interromper esses acessos
  - Lembrar que as interrupções só são reconhecidas no final da execução das instruções
- Exemplo: implementar A = A + B, onde A e B são variáveis de 16 bits

ADD B, A

Acesso à Variáveis na Memória (Acesso não-atômico)

- A interrupção PODE interromper esses acessos
- Exemplo: implementar A = A + B, onde A e B são variáveis de 16 bits

```
MOV   A, R0
ADD   B, R0
MOV   R0, A
```

Exemplo 2

- Exemplo: implementar A = A + B, onde A e B são variáveis de **32 bits**
- Única solução possível no CESAR

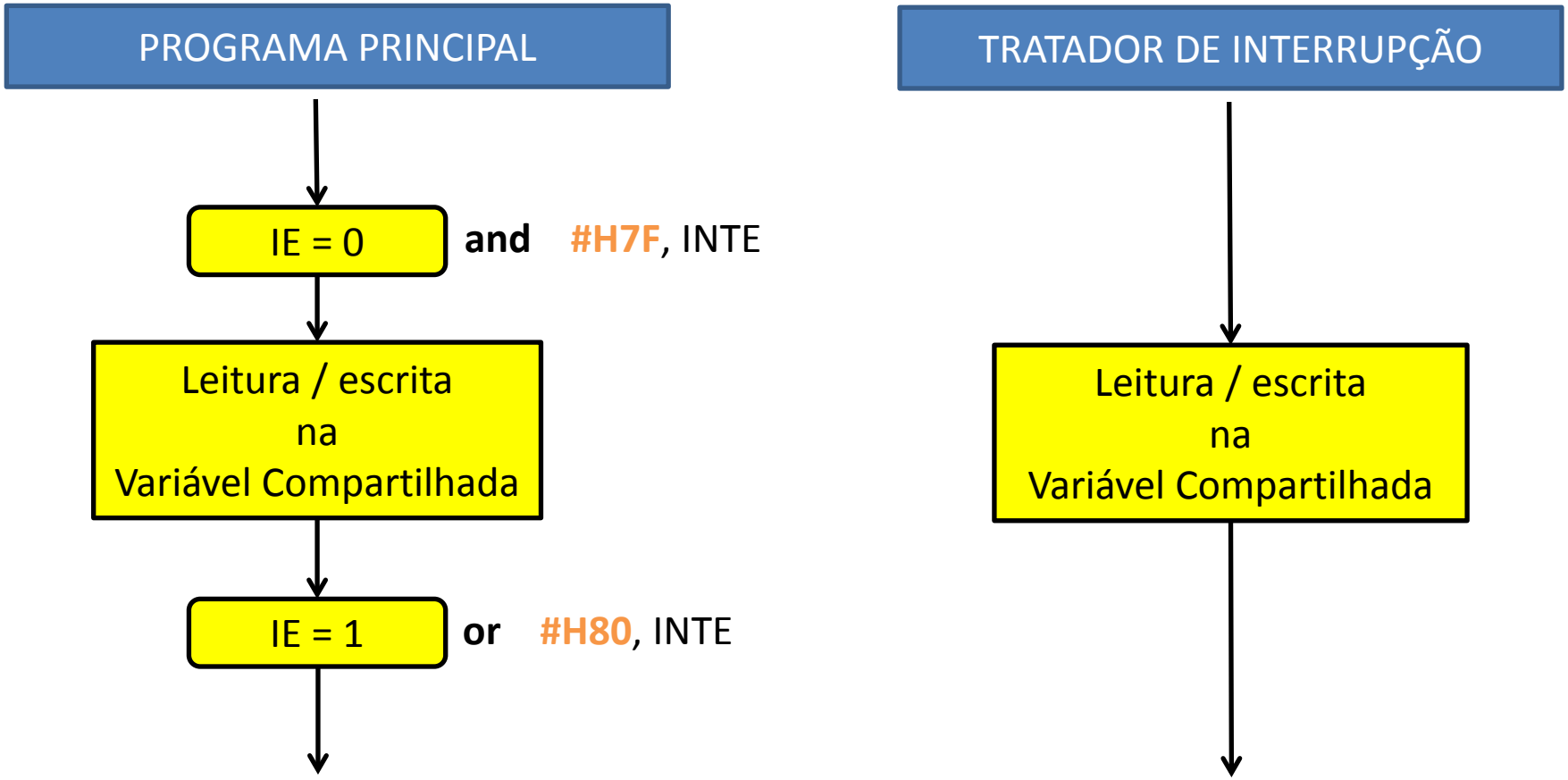
```
ADD   B, A
ADD   B+2, A+2
ADC   B
```

- Essa implementação PODE ser interrompida!

Acesso Não-Atômico – Solução

- Solução: impedir a interrupção nos momentos críticos
- Existem dois mecanismos
  - Controle de acesso (será visto em Sistemas Operacionais)
  - Desabilitação da interrupção
    - As instruções que acessam as variáveis compartilhadas serão executadas com as interrupções desabilitadas

Acesso Não Atômico



CESAR16i

Mecanismo de Interrupção