Universidade Tecnológica Federal do Paraná – UTFPR
Bacharelado em Ciência da Computação

BCC33B – Arquitetura e Organização de Computadores

Prof. Rogério A. Gonçalves

rogerioag@utfpr.edu.br

Aula 011

Aula de Hoje:

- **Interrupções**
- **Tratamento de Exceções**

Interrupção

Eventos Externos:

Exemplo: ocorrência de um alarme externo

Requer atenção imediata do processador

Como atender a solicitação do evento externo?

Sub-Rotina:

Poderia ser escrito um código para atender a ocorrência do evento.

MAS, como NOTIFICAR o programa principal sobre a ocorrência do evento??

2 Soluções para notificar o processador:

Varredura ou Polling

Interrupção

Interrupção

Polling:

Programa principal verifica (“varre”) periodicamente se há ocorrência de evento externo.

Por exemplo: Um sinal de alarme pode ativar um bit numa porta de entrada.

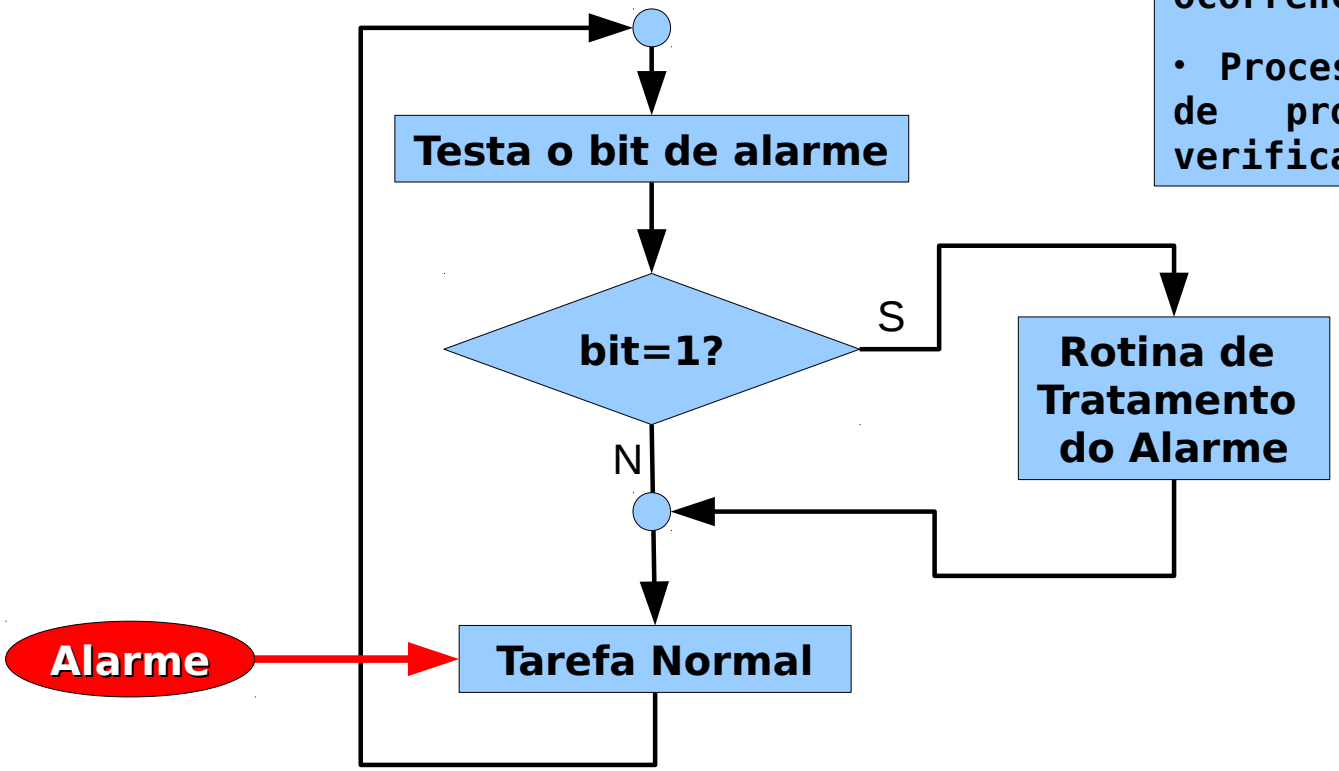
O programa principal verifica periodicamente a porta de entrada para testar o bit de aviso de ocorrência de alarme.

Se ocorrer o evento o processador desvia a execução para a sub-rotina correspondente.

Problema com Polling (próximo slide)!

Interrupção

Polling

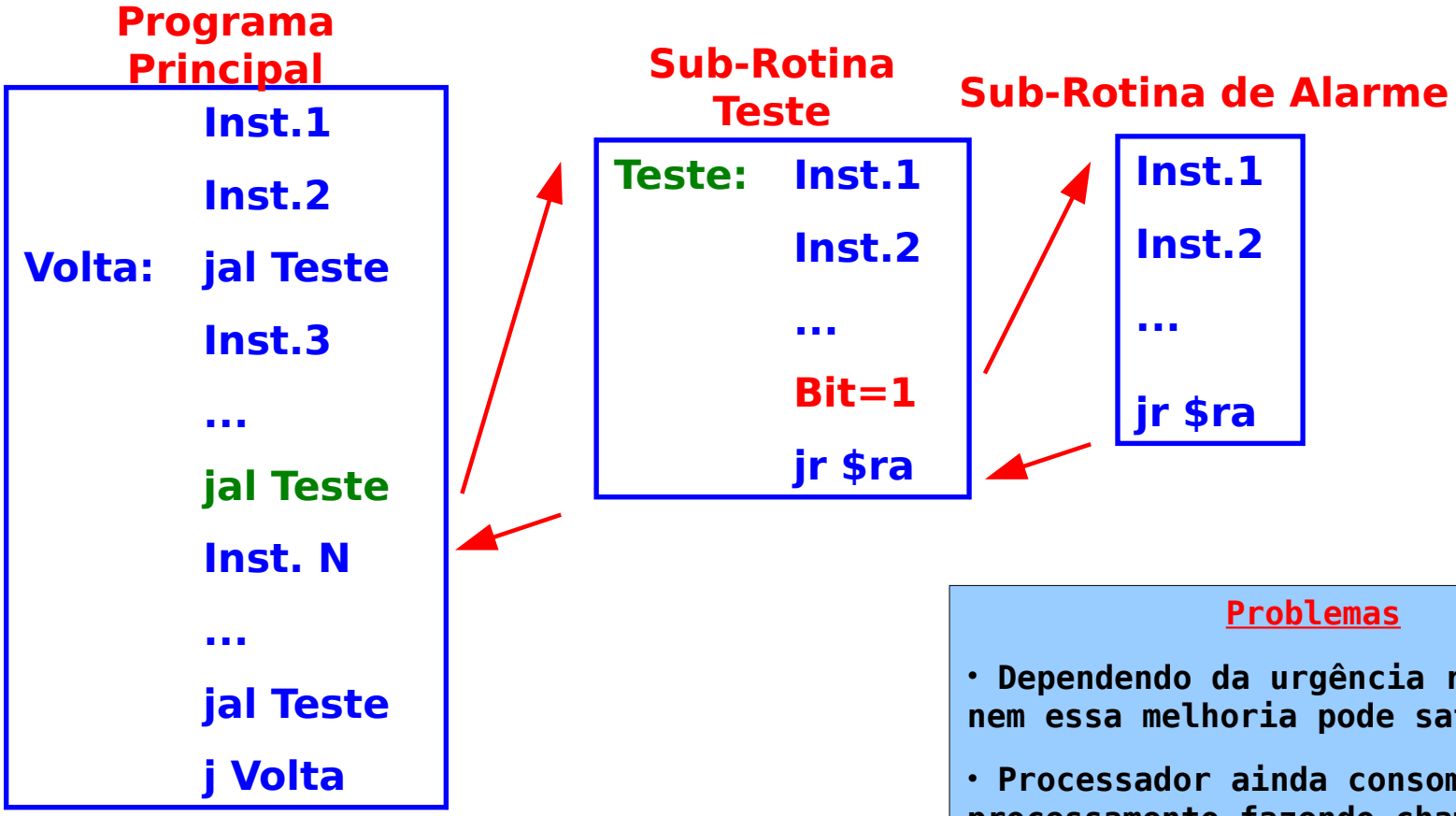


Problemas

- Dependendo do instante em que ocorre o Alarme, pode decorrer um tempo longo até que seja verificada a ocorrência.
- Processador consome tempo de processamento fazendo verificações periódicas

Interrupção

Uso de sub-rotinas no programa principal para tentar diminuir o tempo de resposta ao sinal de Alarme

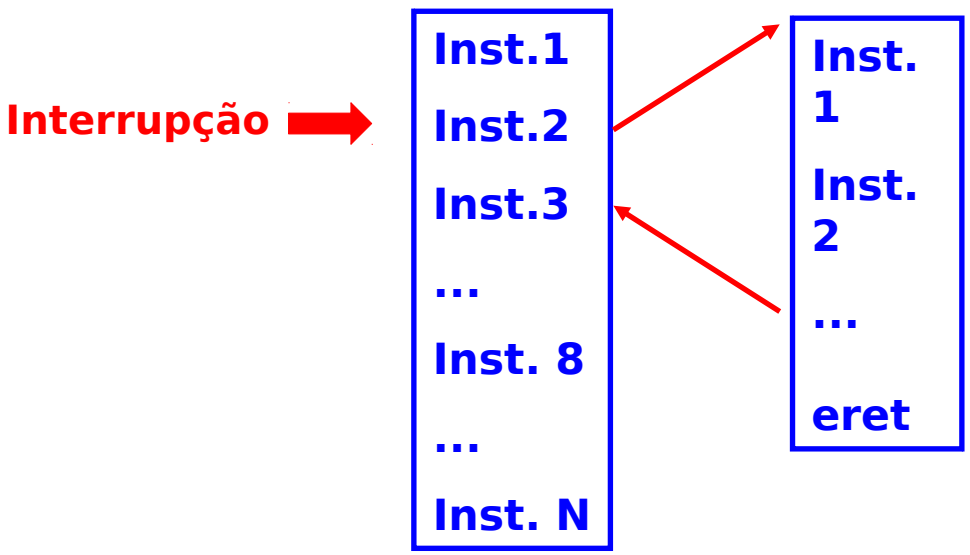


Problemas

- Dependendo da urgência requerida, nem essa melhoria pode satisfazer
- Processador ainda consome tempo de processamento fazendo chamadas de sub-rotinas periodicamente

Interrupção

<u>Alternativa Viável</u>: uso de interrupção para notificar processador



Evento externo notifica o processador em uma das entradas de interrupção

Processador suspende o processamento do programa principal e desvia a execução para a Rotina de Tratamento da Interrupção

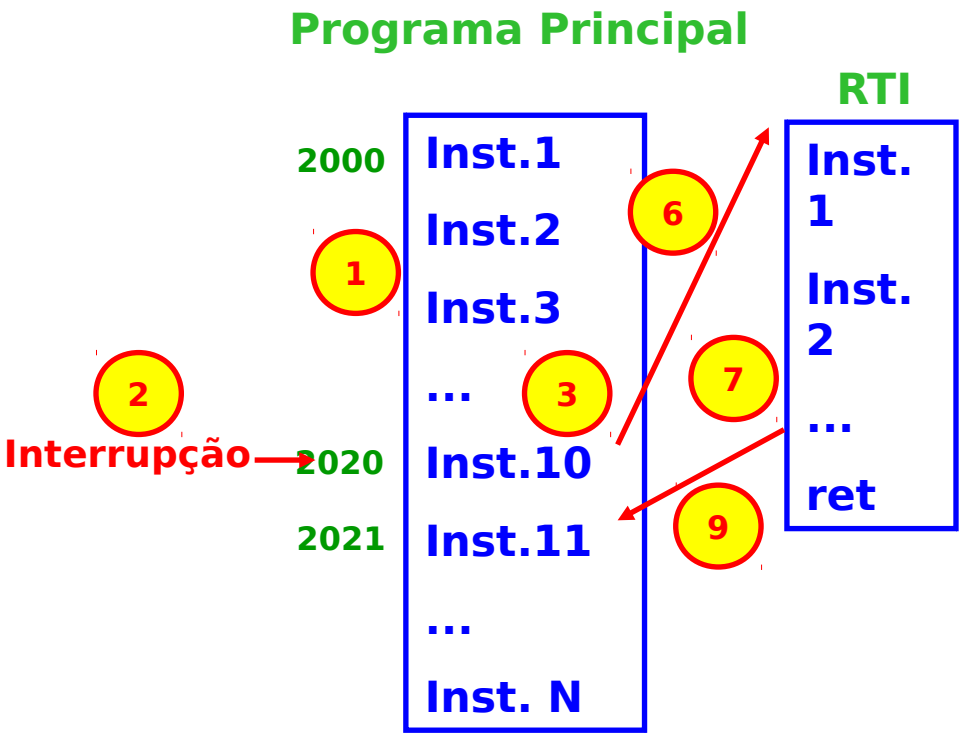
Após execução da Rotina o processador retoma a execução do programa principal a partir de onde parou

<u>Vantagem:</u>

- **Permite que os eventos sejam atendidos rapidamente**
- **Processador não precisa ficar fazendo verificações periódicas**

Interrupção vetorizada

Processamento da Interrupção



1. Processador executando prog. princ.
2. Recebe Interrupção em 2020
3. Termina de executar instr. em 2020
4. Salva PC (2021) na Pilha
5. Carrega PC com endereço da RTI
6. Desvia execução para RTI
7. RTI é executada até RET
8. Extrai PC da Pilha
9. Retorna execução a partir da última instrução após interrupção

Para sistemas que usam vetor de interrupção

É passado ao processador o código da interrupção e então desvia para o endereço da rotina de tratamento de interrupção (RTI)

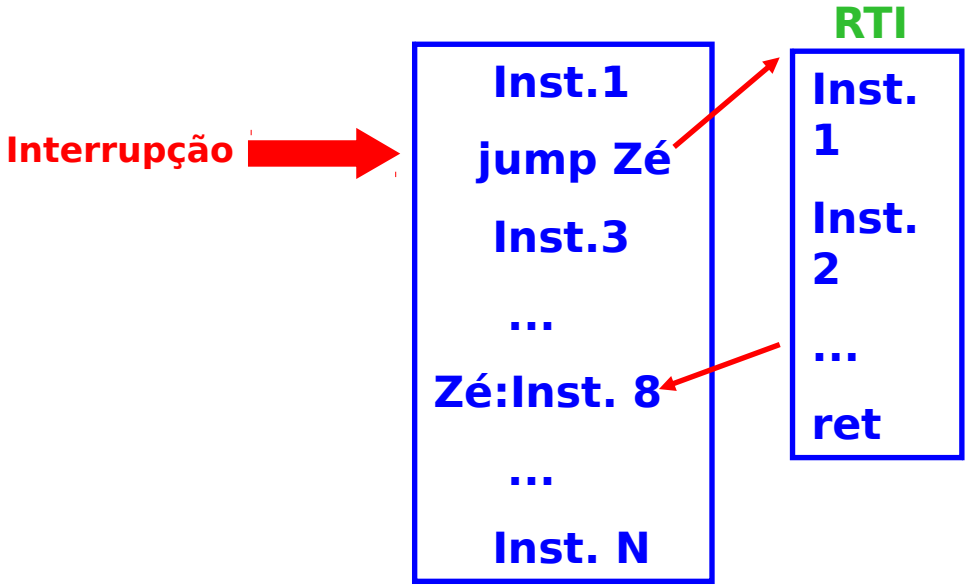
Interrupção

Se no momento da interrupção o processador estiver executando uma instrução de salto?

Se no momento da interrupção o processador estiver executando uma instrução de chamada de procedimento?

Interrupção

Ocorrência de interrupção durante JUMP



1) Termina de executar a instrução de salto (jump) (carrega PC com endereço "Alvo")

2) Salva PC atual ("Alvo") na Pilha

3) Desvia para RTI

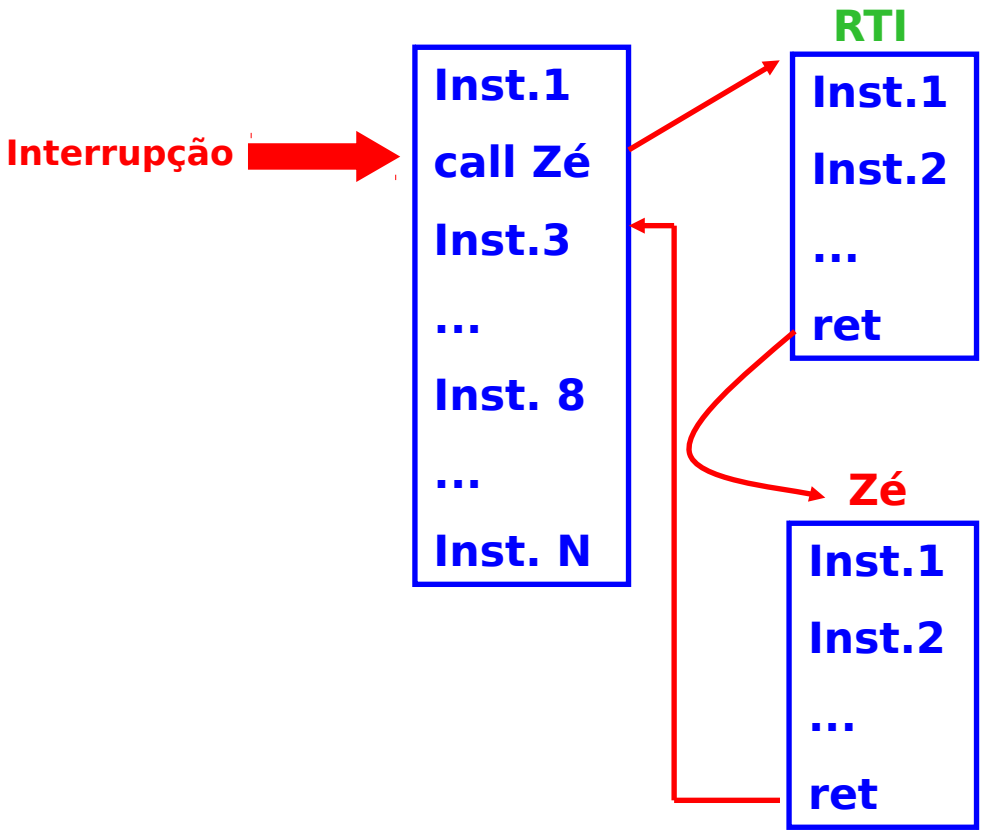
4) Executa RTI até retorno

5) Extrai PC da Pilha

6) Retoma execução a partir do endereço "Alvo"

Interrupção

Ocorrência de interrupção durante CALL



- Salva PC atual (instrução seguinte ao CALL) na Pilha
- Salva endereço da Sub-Rotina na Pilha
- Executa RTI até o retorno
- Extrai endereço da Sub-Rotina da Pilha e carrega no PC
- Executa Sub-Rotina até retorno
- Extrai PC da Pilha
- Retoma execução a partir da instrução seguinte à chamada.

MIPS

- **Tratadores de Exceção *(Exception handlers)***
  - **Também conhecidos como:**
    - *trap handlers*
    - *interrupt handlers*
- **O mesmo mecanismo serve aos três tipos, mesmo sendo distintos.**
- **Exceções são causadas por condições excepcionais que ocorrem em tempo de execução.**
  - **Ex.: referência a um endereço de memória inválido.**
- ***Traps* são causadas por instruções para esse propósito diretamente no código.**
- **Interrupções são causadas por dispositivos externos.**

Exceção X Interrupção

- **Exceção:** um evento inesperado (exceção) de dentro do processador.
  - Por exemplo: *overflow* aritmético
- **Interrupção:** evento inesperado, mas de fora do processador.
  - Por exemplo: comunicação com dispositivo de entrada e saída.
- Algumas arquiteturas não distinguem, usam o termo interrupção para todas as exceções.

Exceções (Interrupções)

- **Chamada de procedimento, não prevista no código, para um *exception handler (tratador de exceção)***
- **Causado por:**
  - **Hardware, também chamado *interrupção*, exemplo:**
    - » teclado
  - **Software, também chamado de *traps*, exemplo:**
    - » instrução não definida
    - » Overflow aritmético
- **Quando uma exceção ocorre, o processador:**
  - **Registra a causa da exceção**
  - **Desvia a execução para exception handler no endereço de instrução 0x80000180, conhecido como endereço de exceção.**
  - **É o ponto de entrada do Sistema Operacional que irá tratar a exceção.**
  - **Retorna ao programa.**

Registradores de Exceção

- São registradores de *status* e não fazem parte do *register file*. São de 32 bits também.
  - **`Cause`**
    - » Registra a causa da exceção
    - » O bit menos significativo codifica a origem de exceções: instrução indefinida = 0 e overflow aritmético = 1.
  - **`EPC` (Exception PC)**
    - » Registra o PC onde ocorreu a exceção
    - » Endereço da instrução afetada.

- **`EPC` e `Cause`: parte do Coprocessador 0**
- Move from Coprocessor 0
  - **`mfc0 $t0, EPC`**
  - Move o conteúdo de **`EPC`** para **`$t0`**

Registradores de Coprocessador 0

Nome do registrador	Número do registrador	Uso
BadVAddr	8	endereço de memória em que ocorreu uma referência de memória problemática
Count	9	temporizador
Compare	11	valor comparado com o temporizador que causa interrupção quando combinam
Status	12	máscara de interrupções e bits de habilitação
Cause	13	tipo de exceção e bits de interrupções pendentes
EPC	14	endereço da instrução que causou a exceção
Config	16	configuração da máquina

Exceções

- O Processador salva a causa e o PC em `Cause` e `EPC`
- Processador desvia para o *exception handler* (0x80000180)
- Exception handler:
  - Salva os registradores na pilha
  - Lê o registrador `Cause`

    ```
    mfc0 Cause, $t0
    ```
  - Trata a exceção
  - Restaura os registradores
  - Retorna ao programa

    ```
    mfc0 EPC, $k0
    jr $k0
    ```

Gerando interrupções via software (Traps)

- **O conjunto de instruções do MIPS inclui um número de instruções que condicionalmente disparam uma exceção do tipo trap com base em valores de dois registradores ou de uma constante e um registrador:**
- *trap if equal:*
  - teq, teqi
- *trap if not equal:*
  - tne, tnei
- *trap if greater than or equal:*
  - tge, tgeu, tgei, tgeiu
- *trap if less than:*
  - tlt, tltu, tlti, tltiu

Quando exceções acontecem

- **No Coprocessor 0:**
  - registrador $12 (status) bit 1 é setado
  - registrador $13 (cause) bits 2-6 são setados com o código do tipo de exceção.
  - registrador $14 (epc) é setado para o endereço da instrução que disparou a exceção.
- **Se a exceção foi causada por um endereço de memória inválido, o registrador $8 (vaddr) do Coprocessador 0 é setado para o endereço inválido.**
- **O fluxo de execução salta para a endereço 0x800000180 localizado da memória de instruções. Este endereço no segmento de código do kernel (kernel text segment) (.ktext directive) é o padrão de localização do tratador de exceções no MIPS32.**

Registradores Status e Cause

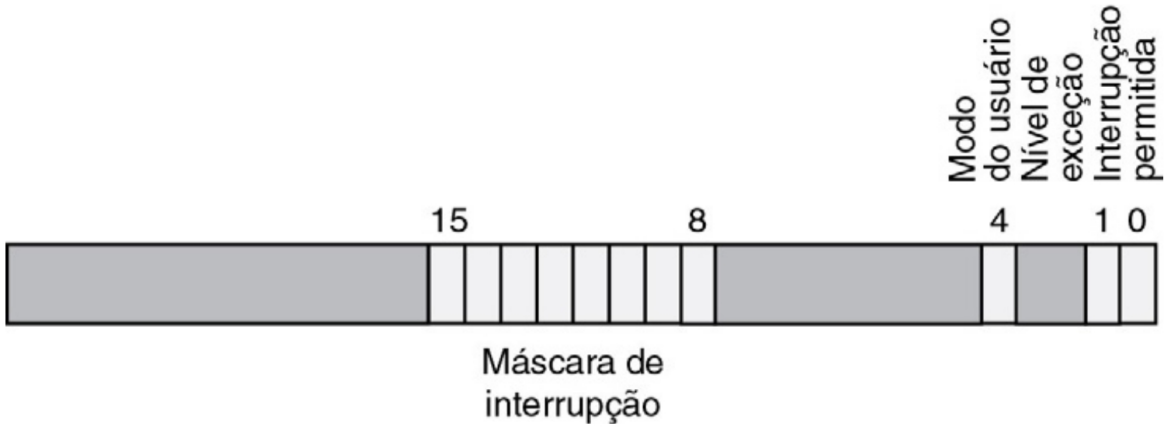


FIGURA B.7.1 O registrador Status.

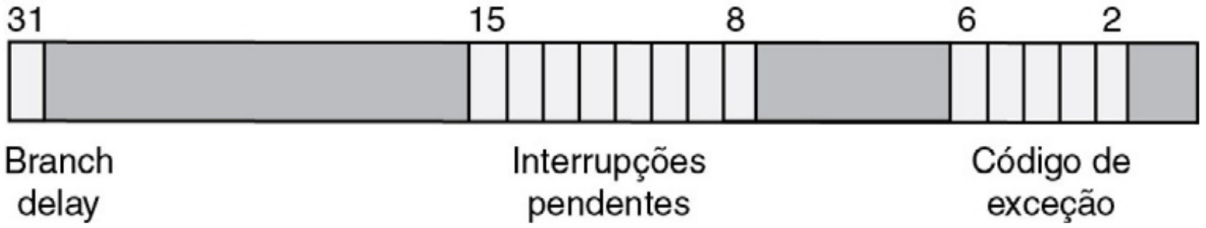


FIGURA B.7.2 O registrador Cause.

Causa da Exceção

Número	Nome	Causa da exceção
00	Int	interrupção (hardware)
04	AdEL	exceção de erro de endereço (load ou busca de instrução)
05	AdES	exceção de erro de endereço (store)
06	IBE	erro de barramento na busca da instrução
07	DBE	erro de barramento no load ou store de dados
08	Sys	exceção de syscall
09	Bp	exceção de breakpoint
10	RI	exceção de instrução reservada
11	CpU	coprocessador não implementado
12	Ov	exceção de overflow aritmético
13	Tr	trap
15	FPE	ponto flutuante

Exceções

Exception	Cause
Hardware Interrupt	0x00000000
System Call (syscall)	0x00000020
Breakpoint / Divide by 0	0x00000024
Undefined Instruction	0x00000028
Arithmetic Overflow	0x00000030

No MARS

- **Existem três maneiras de incluir um tratador de exceções em um programa MIPS:**
  - Escrever a rotina do tratador de exceções no mesmo arquivo do programa.
  - Escrever um tratador em um arquivo separado e armazená-lo no mesmo diretório do programa e selecioná-lo em Settings/Assemble all files in directory.
  - Escrever um tratador em um arquivo separado, armazená-lo em algum diretório e então abri-lo em “Settings/Exception Handler...".

No MARS

- **Se não há instruções no endereço do tratador, 0x800000180**
  - **O MARS irá terminar o programa MIPS com uma mensagem de erro apropriada.**
- **O tratador de exceções pode retornar o controle para o programa usando a instrução *`eret`*.**
- **Isto irá colocar o valor do registrador EPC ($14) no PC, então tenha certeza de incrementar o $14 em 4 antes de retornar, pulando a instrução que causou a exceção;**

No MARS

- **As instruções *mfc0* e *mtc0* são usadas para ler e escrever nos registradores do Coprocessador 0.**
- **Os bits 8-15 do registrador Cause ($13) podem ser usados para indicar interrupções pendentes.**
- **Atualmente eles são usados somente para o teclado e o *Display*, onde bit 8 representa a interrupção do teclado e o bit 9 representa a interrupção do *display*.**

Tipos de Exceções

- **Declarados no MARS:**
  - **ADDRESS_EXCEPTION_LOAD (4),**
  - **ADDRESS_EXCEPTION_STORE (5),**
  - **SYSCALL_EXCEPTION (8),**
  - **BREAKPOINT_EXCEPTION (9),**
  - **RESERVED_INSTRUCTION_EXCEPTION (10),**
  - **ARITHMETIC_OVERFLOW_EXCEPTION (12),**
  - **TRAP_EXCEPTION(13),**
  - **DIVIDE_BY_ZERO_EXCEPTION (15),**
  - **FLOATING_POINT_OVERFLOW (16),**
  - **FLOATING_POINT_UNDERFLOW (17).**
- **Um tratador de exceções deve primeiro salvar o conteúdo dos registradores de propósito geral e depois de sua execução restaurá-los.**

Exemplo

```
   .text
main:
   teqi $t0, 0          # dispara um trap porque $t0 contem 0.
   li   $v0, 10         # Após o retorno do exception handler, sai do programa
   syscall              # término normal.

# Trap handler in the standard MIPS32 kernel text segment

   .ktext 0x80000180
   move $k0,$v0     # Salva o valor de $v0
   move $k1,$a0     # Salva o valor de $a0
   la   $a0, msg    # endereço da string a ser impressa.
   li   $v0, 4      # chamada de sistema para imprimir string.
   syscall
   move $v0,$k0     # restaura $v0
   move $a0,$k1     # restaura $a0
   mfc0 $k0,$14     # Coprocessor 0 register $14 tem o endereço da instrução que fez o trap.
   addi $k0,$k0,4   # Adiciona 4 para ir para o endereço da próxima instrução.
   mtc0 $k0,$14     # Store o novo endereço de volta em $14
   eret             # Error return; set PC para o valor de $14
   .kdata
msg:
   .asciiz "Trap generated"
```

Resumo da Aula de Hoje

Tópicos mais importantes:

Linguagem Assembly

Microprocessador MIPS

Exceções e Interrupções