

Tratamento de exceções

Eventos excepcionais em tempo de execução

- **Fluxo de execução é sequencial (default)**
 - Alterado explicitamente pelo programador
 - » Instruções de desvio
- **Há alterações implícitas do fluxo de execução**
 - Podem ser causadas por eventos excepcionais
 - » Em tempo de execução
- **Eventos excepcionais**
 - Causam mudança **inesperada** na sequência do programa
 - Terminologia: exceção e/ou interrupção
 - Causa: anomalia ou mecanismo de suporte ao S.O.

Exceções e interrupções

- **Exceção**

- **Evento inesperado ocorrido dentro da CPU**

- » **Overflow aritmético**
 - » **Divisão por zero**
 - » **Instrução indefinida**
 - » **Chamada de sistema**

- **Interrupção**

- **Evento inesperado ocorrido fora da CPU**

- » **Requisição a partir de dispositivo de E/S**
 - » **Disfunção no HW**

Exceções e interrupções

- **Exceção**

- Evento inesperado ocorrido dentro da CPU

- » **Overflow aritmético**
 - » **Divisão por zero**
 - » **Instrução indefinida**
 - » **Chamada de sistema**

Anomalia

**Mecanismo de
suporte ao S.O.**

- **Interrupção**

- Evento inesperado ocorrido fora da CPU

- » **Requisição a partir de dispositivo de E/S**
 - » **Disfunção no HW**

Uso de exceções e interrupções

- **Interrupções**
 - **Geralmente para transferir dados de/para E/S**
 - » **Exceto quando sinalizam disfunção de hardware**
- **Exceções**
 - **Sinalizar e/ou corrigir anomalia**
 - **Abortar o programa (se anomalia irrecuperável)**
 - **Invocar rotinas do sistema operacional**

Uso de exceções e interrupções

- **Interrupções**
 - Geralmente para transferir dados de/para E/S
 - » Exceto quando sinalizam disfunção de hardware
 - **Serão abordadas no estudo de E/S**
- **Exceções**
 - Sinalizar e/ou corrigir anomalia
 - Abortar o programa (se anomalia irrecuperável)
 - Invocar rotinas do sistema operacional
 - **Serão abordadas nesta aula**

Como tratar exceções ?



Tratador de exceções ("exception handler")



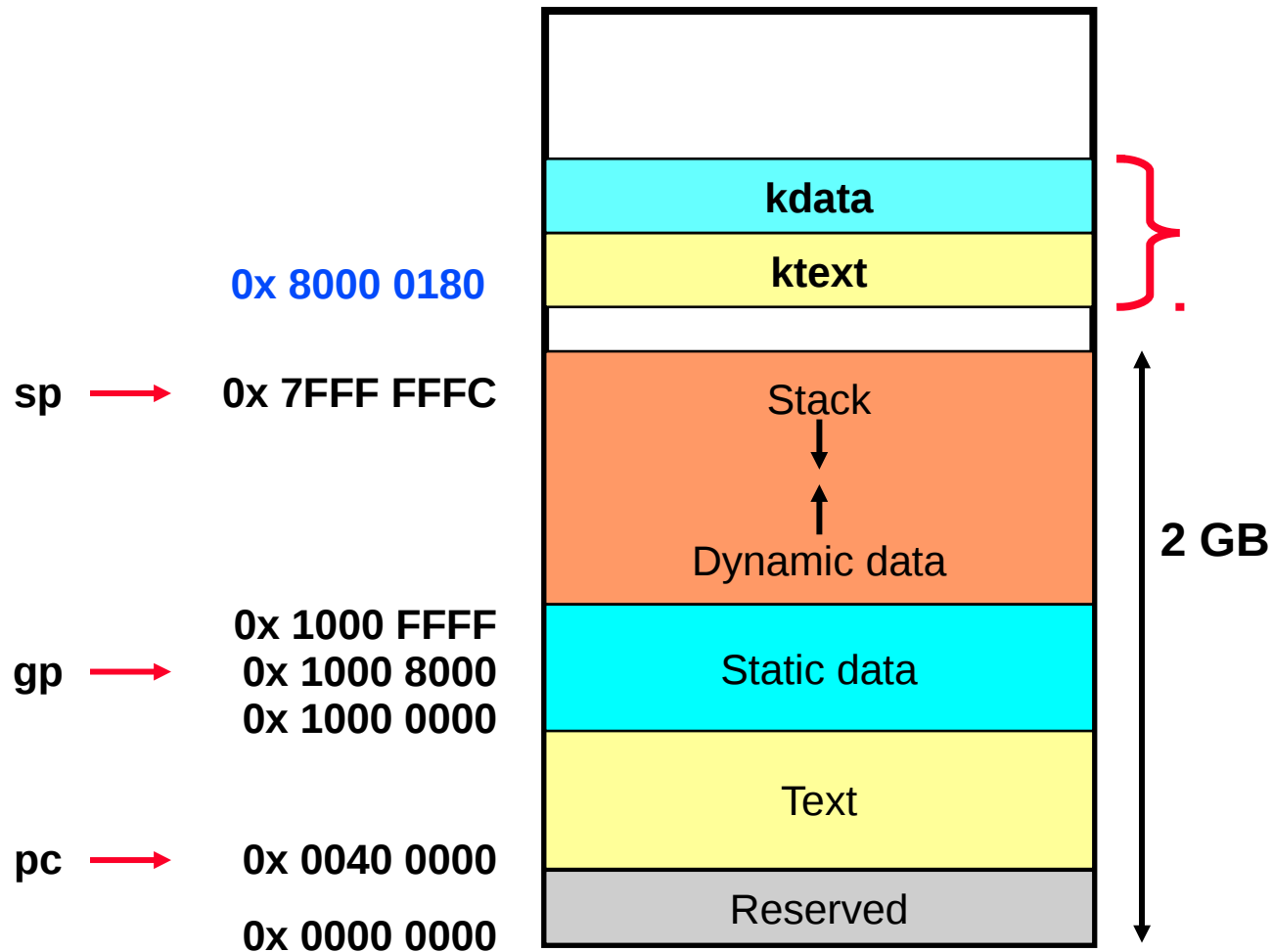
Exceções: suporte de HW (1)

- **Deteccção**
 - Overflow (ALU), instrução indefinida (unidade de controle)
- **Retorno ao fluxo normal (eventual)**
 - Registrador adicional: **EPC**
 - » “Exception program counter”
- **Sinalização ao S.O.**
 - Avisar que houve exceção é insuficiente
 - » Eventos distintos requerem tratamentos distintos
 - » Causa da exceção é codificada
 - Registrador adicional: **Cause**

Etapas de um tratador de exceções

- **Salvamento de contexto**
 - Registradores usados pelo tratador
- **Decodificação da causa**
 - Leitura e interpretação de **Cause**
- **Ação corretiva ou sinalizadora**
 - Chama uma rotina diferente para cada tipo de exceção
- **Restauração de contexto**
 - Registradores usados pelo tratador
- **Retorno ao fluxo de execução**
 - Reinicia execução em **EPC** ou **EPC+4**

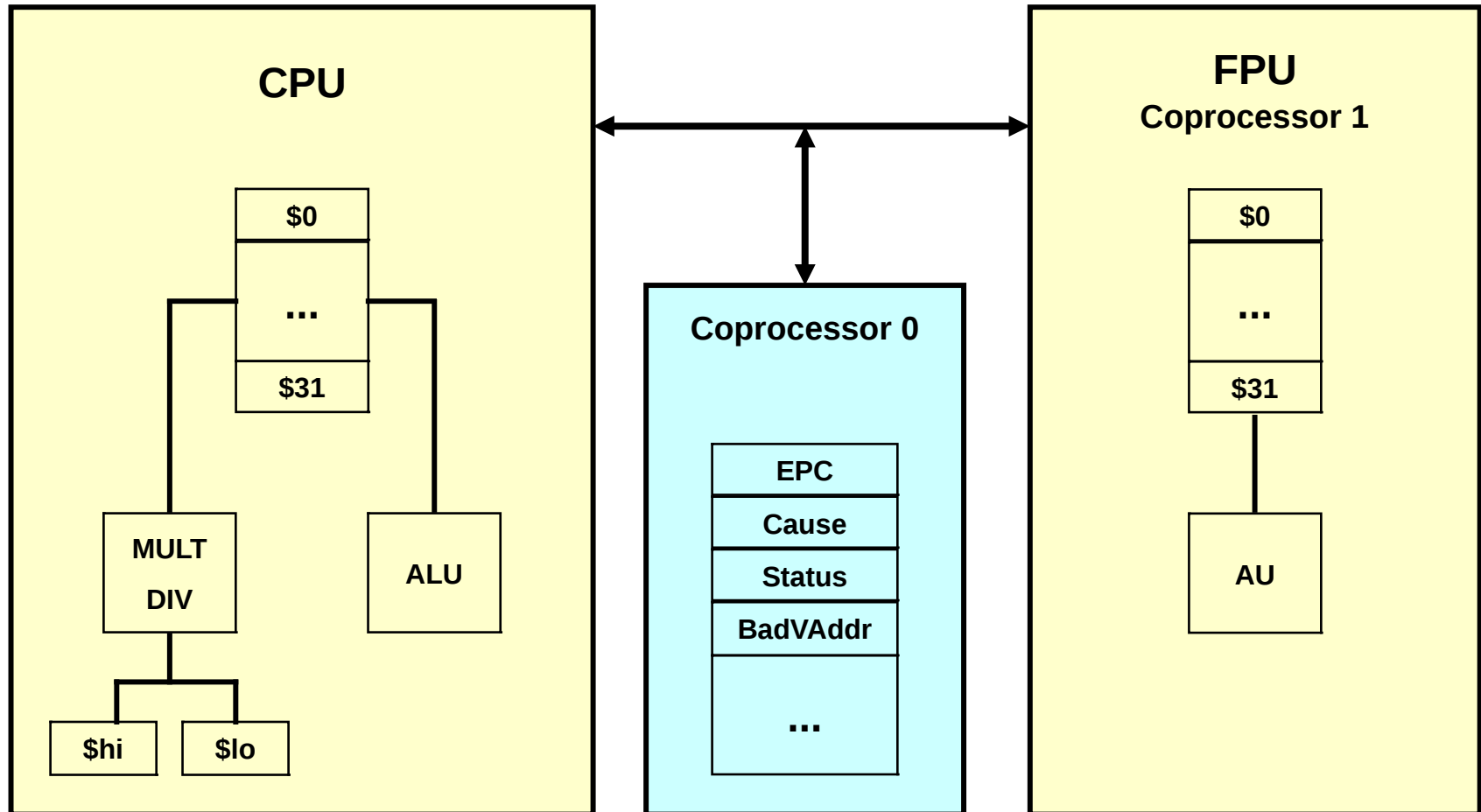
Onde reside o tratador para MIPS?



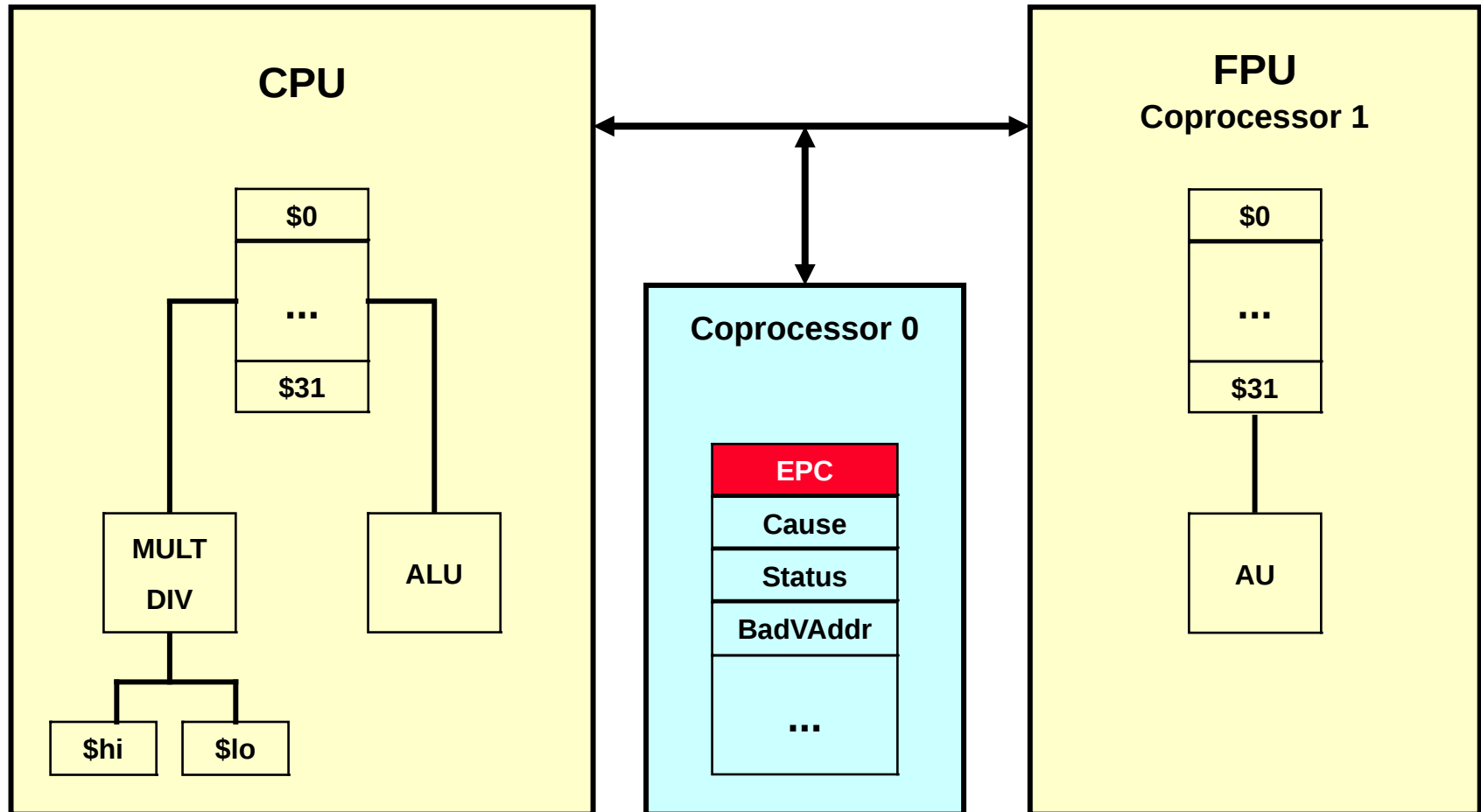
Exceções: suporte de HW (2)

- **Administração de mais de um evento**
 - Quando evento excepcional sob tratamento
 - Outros eventos não devem interrompê-lo
 - Registrador adicional: **Status**
- **Suporte a eventos específicos**
 - Exceções geradas no subsistema de memória
 - » Desalinhamento, violação de endereço reservado
 - Registrador adicional: **BadVAddr**

MIPS: decomposição da ISA



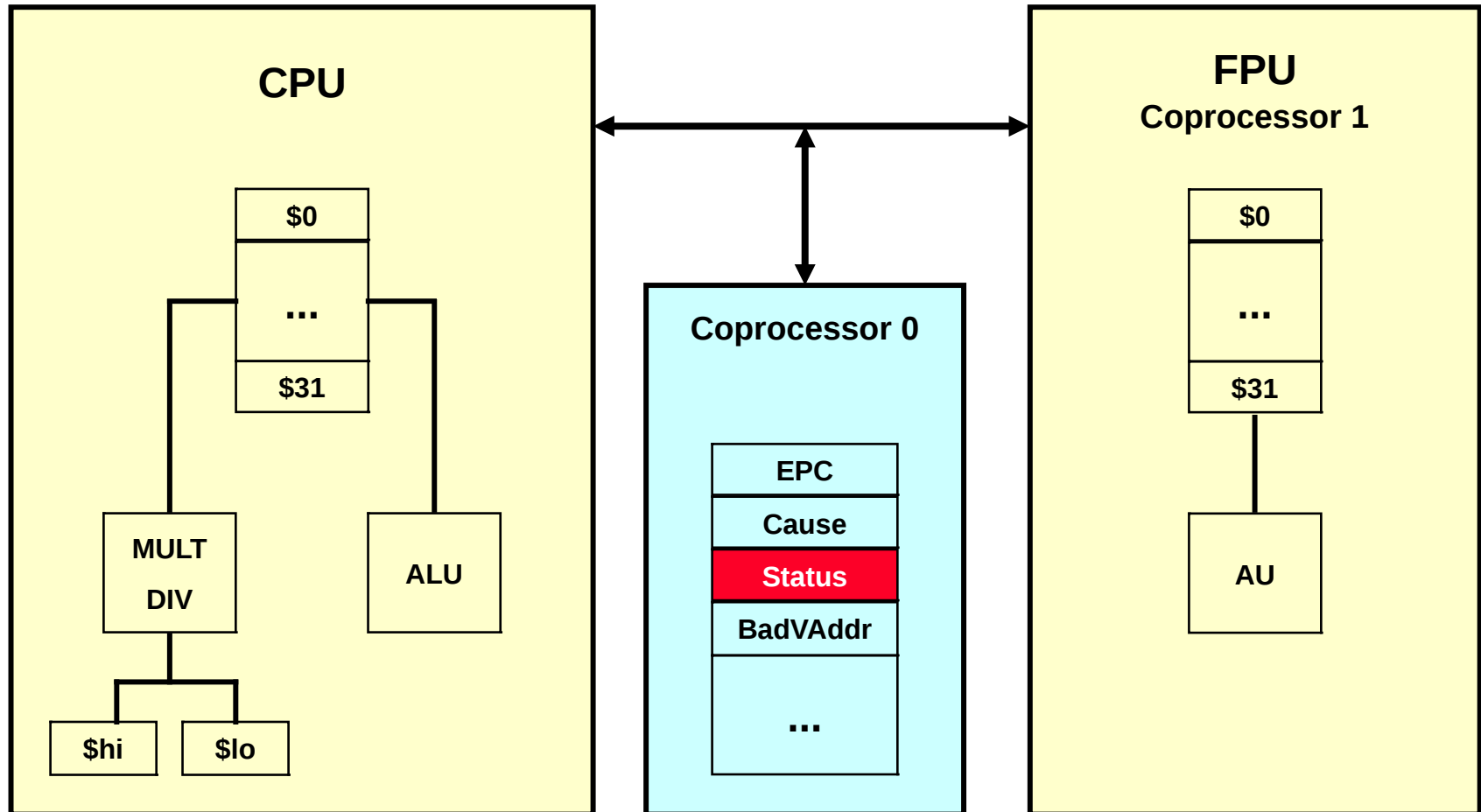
MIPS: decomposição da ISA



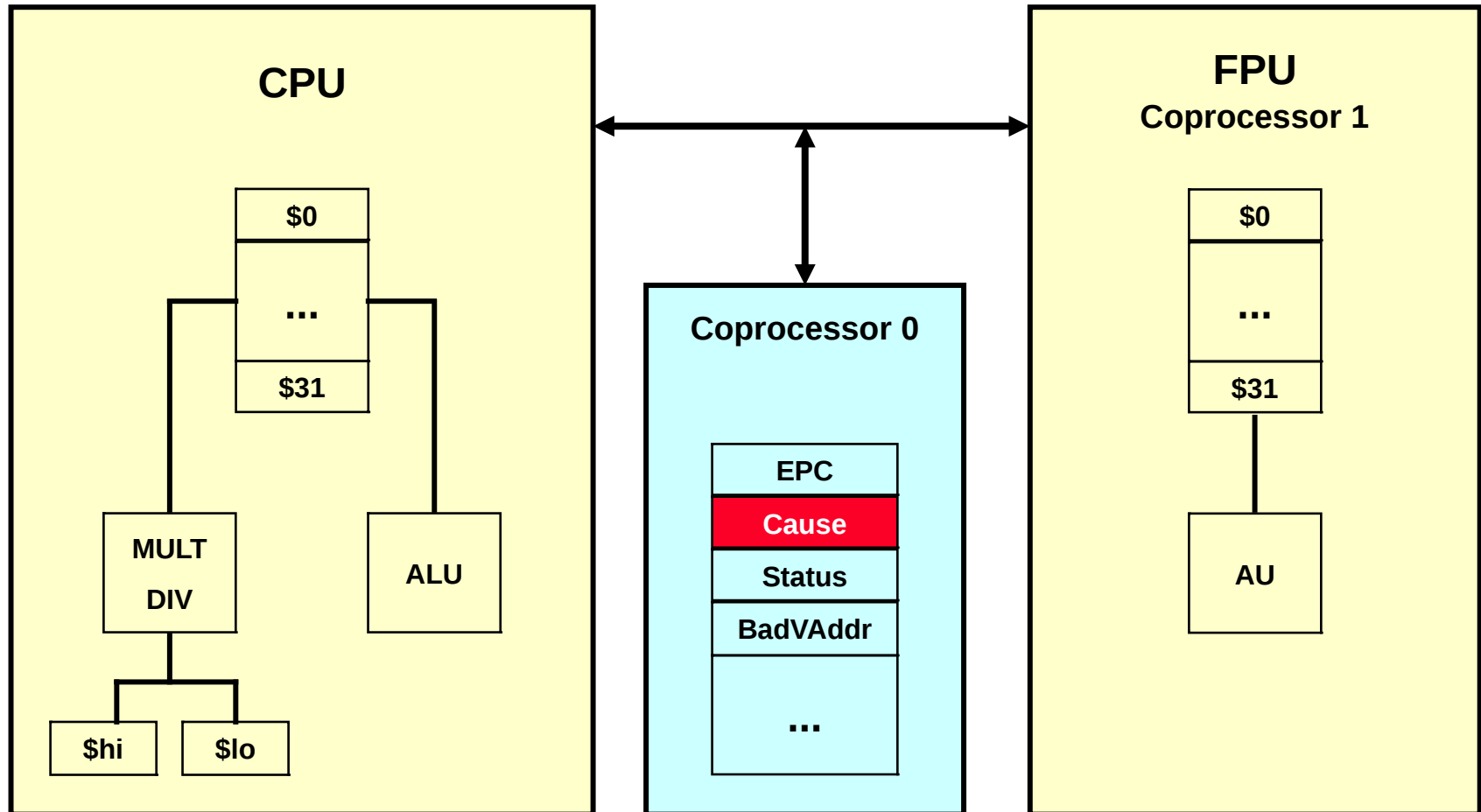
Endereço da instrução que causou exceção

Luiz C. V. dos Santos, INE/CTC/UFSC

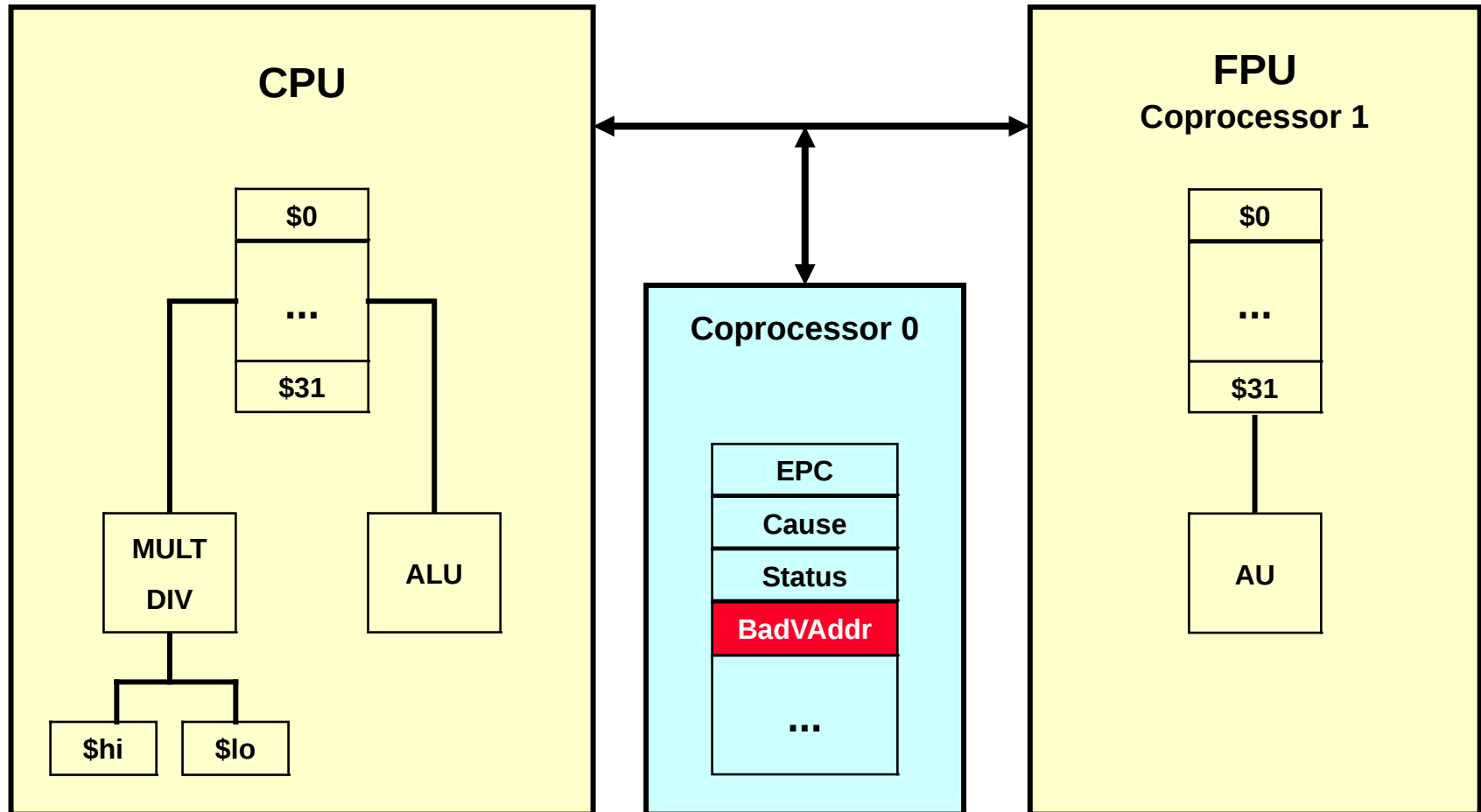
MIPS: decomposição da ISA



MIPS: decomposição da ISA



MIPS: decomposição da ISA

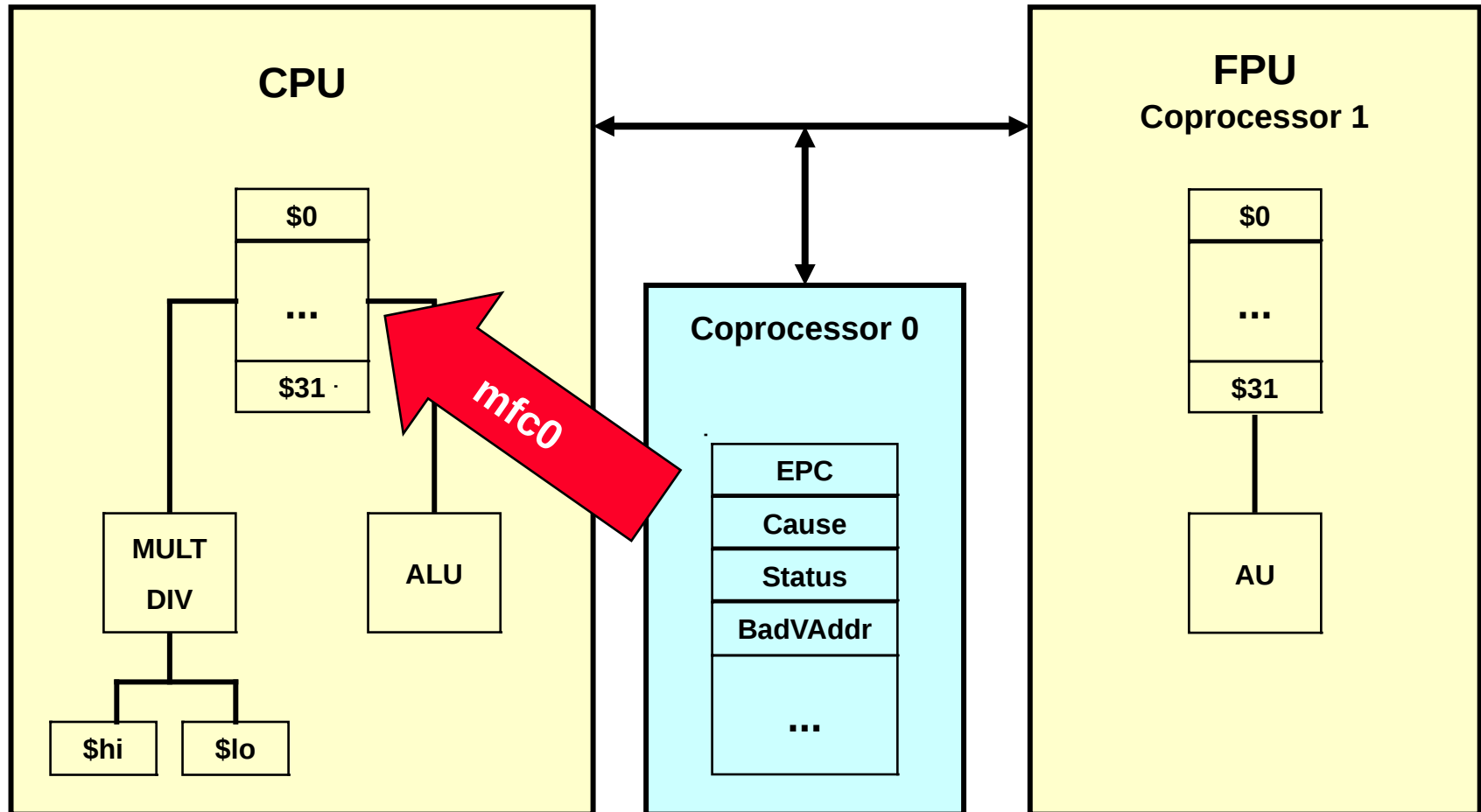


Endereço que violou restrição de uso da memória

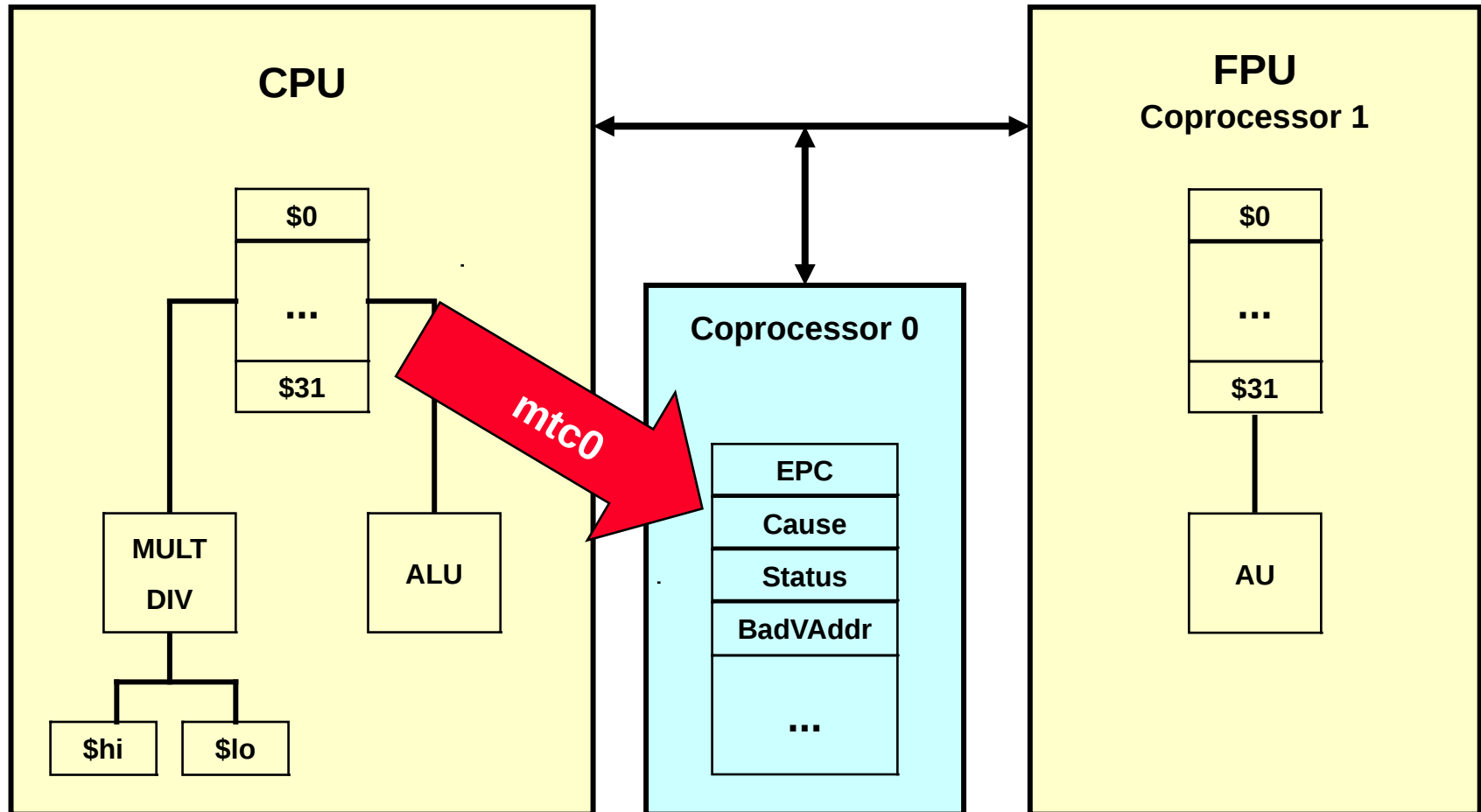
Luiz C. V. dos Santos, INE/CTC/UFSC

INE 5411, exception-handling, slide 17

MIPS: decomposição da ISA



MIPS: decomposição da ISA



Exceção: registradores de suporte

Nome	Número	Uso
BadVAddr	8	Endereço que violou restrição de uso memória
Status	12	Bits de habilitação de exceção e interrupção
Cause	13	Tipo de exceção e interrupções pendentes
EPC	14	Endereço da instrução que causou exceção

mfc0 \$k0, \$14 # \$k0 ← EPC (“move from c0”)

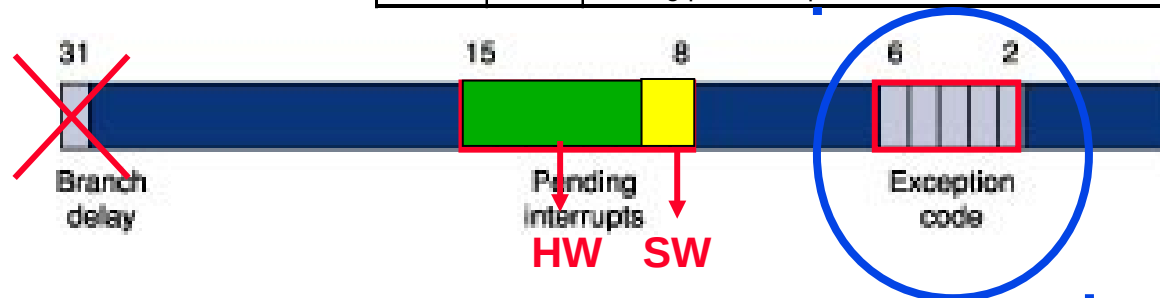
mtc0 \$k0, \$12 # Status ← \$k0 (“move to c0”)

eret # PC ← EPC (“exception return”)

A anatomia do registrador **cause**

Number	Name	Cause of exception
0	Int	Interrupt (hardware)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	IBE	Bus error on instruction fetch
7	DBE	Bus error on data load or store
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved instruction exception
11	CpU	Coprocessor unimplemented
12	Ov	Arithmetic overflow exception
13	Tr	Trap
15	FPE	Floating point exception

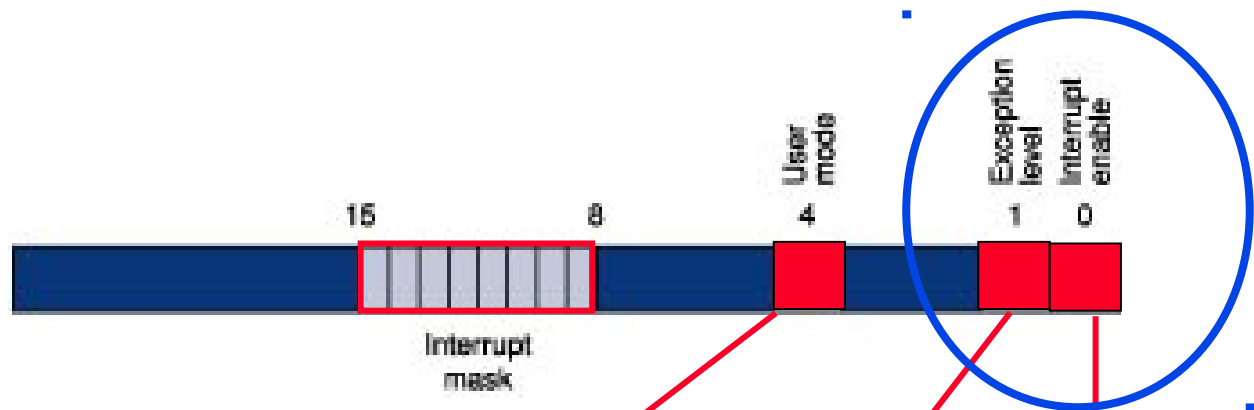
Cause



Lidos pelo tratador para identificar o
tipo de exceção

A anatomia do registrador status

Status



Atualizados pelo tratador
imediatamente antes de
retornar:

Bit 1 (EXL): será desativado (“0”)

Bit 0 (IE): será ativado (“1”)

0: user
1: kernel

0: no exception
1: exception occurred

(EPC não atualizado se
nova exceção ocorrer)

0: intr disabled
1: intr enabled

Tratador de exceção: exemplo

.ktext 0x80000180

.kdata

**\$a0 e \$a1 serão usados pelo tratador, mas
não podem ser preservados em pilha**

Tratador de exceção: exemplo

.ktext 0x80000180

.kdata

save0: .word 0

#Space for saving

save1: .word 0

#used registers

Tratador de exceção: exemplo

.ktext 0x80000180

sw \$a0, save0 #Save context
sw \$a1, save1 #

.kdata

save0: .word 0 #Space for saving
save1: .word 0 #used registers

Pseudo-instruções usam \$at, que precisa ser preservado, mas não na memória

Tratador de exceção: exemplo

.ktext 0x80000180

move \$k1, \$at **#Allow pseudo-intr.**

sw \$a0, save0 **#Save context**

sw \$a1, save1 **#**

.kdata

save0: .word 0 **#Space for saving**

save1: .word 0 **#used registers**

Tratador de exceção: exemplo

.ktext 0x80000180

move \$k1, \$at **#Allow pseudo-intr.**

sw \$a0, save0 **#Save context**

sw \$a1, save1 **#**

mfc0 \$k0, \$13 **#Read Cause**

.kdata

save0: .word 0 **#Space for saving**

save1: .word 0 **#used registers**

Tratador de exceção: exemplo

.ktext 0x80000180

move \$k1, \$at **#Allow pseudo-intr.**

sw \$a0, save0 **#Save context**

sw \$a1, save1 **#**

mfc0 \$k0, \$13 **#Read Cause**

srl \$a0, \$k0, 2 **#Extract ExCode**

andi \$a0, \$a0, 0xf

.kdata

save0: .word 0 **#Space for saving**

save1: .word 0 **#used registers**

**Nosso tratador vai manipular só exceções,
mas não interrupções (ExCode = 0)**

Tratador de exceção: exemplo

```
.ktext 0x80000180
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      #Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
```

done:

```
.kdata
save0: .word 0      #Space for saving
save1: .word 0      #used registers
```

Nosso tratador vai apenas sinalizar a exceção, imprimindo mensagem de erro

Tratador de exceção: exemplo

```
.ktext 0x80000180
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      #Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
move $a0, $k0       #Cause as parameter
mfc0 $a1, $14       #EPC as parameter
jal print_exp       #print error message
```

```
.kdata
save0: .word 0      #Space for saving
save1: .word 0      #used registers
```

**Como anomalia não é corrigida, instrução
que a gerou não será re-executada**

Tratador de exceção: exemplo

.ktext 0x80000180

```
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      # Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
move $a0, $k0       #Cause as parameter
mfc0 $a1, $14       #EPC as parameter
jal print_exp       #print error message
```

done:

```
mfc0 $k0, $14      #Read EPC
addiu $k0, $k0, 4  #Point to next
mtc0 $k0, $14      #Update EPC
```

.kdata

```
save0: .word 0     #Space for saving
save1: .word 0     #used registers
```

**Cause e Status precisam ser preparados
para novas exceções**

Tratador de exceção: exemplo

.ktext 0x80000180

```
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      #Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
move $a0, $k0       #Cause as parameter
mfc0 $a1, $14       #EPC as parameter
jal print_exp       #print error message
```

done:

```
mfc0 $k0, $14      #Read EPC
addiu $k0, $k0, 4  #Point to next
mtc0 $k0, $14      #Update EPC
mtc0 $0, $13       #Clear Cause
```

.kdata

```
save0: .word 0     #Space for saving
save1: .word 0     #used registers
```


Tratador de exceção: exemplo

.ktext 0x80000180

```
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      #Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
move $a0, $k0       #Cause as parameter
mfc0 $a1, $14       #EPC as parameter
jal print_exp       #print error message
```

done:

```
mfc0 $k0, $14      #Read EPC
addiu $k0, $k0, 4  #Point to next
mtc0 $k0, $14      #Update EPC
mtc0 $0, $13       #Clear Cause
mfc0 $k0, $12      #Read Status
```

.kdata

```
save0: .word 0     #Space for saving
save1: .word 0     #used registers
```

Tratador de exceção: exemplo

.ktext 0x80000180

```
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      #Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
move $a0, $k0       #Cause as parameter
mfc0 $a1, $14       #EPC as parameter
jal print_exp       #print error message
```

done:

```
mfc0 $k0, $14      #Read EPC
addiu $k0, $k0, 4  #Point to next
mtc0 $k0, $14      #Update EPC
mtc0 $0, $13       #Clear Cause
mfc0 $k0, $12      #Read Status
ori $k0, $k0, 0x1  #Set IE bit
mtc0 $k0, $12      #Update Status
```

.kdata

```
save0: .word 0     #Space for saving
save1: .word 0     #used registers
```

O contexto precisa ser restaurado, antes de se retornar ao fluxo normal de execução

Tratador de exceção: exemplo

.ktext 0x80000180

```
move $k1, $at      #Allow pseudo-intr.
sw $a0, save0      #Save context
sw $a1, save1      #
mfc0 $k0, $13      #Read Cause
srl $a0, $k0, 2     #Extract ExCode
andi $a0, $a0, 0xf
beqz $a0, done      #Branch if interrupt
move $a0, $k0       #Cause as parameter
mfc0 $a1, $14       #EPC as parameter
jal print_exp       #print error message
```

done:

```
mfc0 $k0, $14      #Read EPC
addiu $k0, $k0, 4  #Point to next
mtc0 $k0, $14      #Update EPC
mtc0 $0, $13       #Clear Cause
mfc0 $k0, $12      #Read Status
ori $k0, $k0, 0x1  #Set IE bit
mtc0 $k0, $12      #Update Status
lw $a0, save0      #Restore context
lw $a1, save1
move $at, $k1
```

.kdata

```
save0: .word 0     #Space for saving
save1: .word 0     #used registers
```

**Finalmente, pode-se retornar com segurança
ao fluxo normal de execução, deixando para
desativar EXL na última instrução**

Tratador de exceção: exemplo

.ktext 0x80000180

move \$k1, \$at	#Allow pseudo-intr.
sw \$a0, save0	#Save context
sw \$a1, save1	#
mfc0 \$k0, \$13	#Read Cause
srl \$a0, \$k0, 2	#Extract ExCode
andi \$a0, \$a0, 0xf	
beqz \$a0, done	#Branch if interrupt
move \$a0, \$k0	#Cause as parameter
mfc0 \$a1, \$14	#EPC as parameter
jal print_exp	#print error message

done:

mfc0 \$k0, \$14	#Read EPC
addiu \$k0, \$k0, 4	#Point to next
mtc0 \$k0, \$14	#Update EPC
mtc0 \$0, \$13	#Clear Cause
mfc0 \$k0, \$12	#Read Status
ori \$k0, \$k0, 0x1	#Set IE bit
mtc0 \$k0, \$12	#Update Status
lw \$a0, save0	#Restore context
lw \$a1, save1	
move \$at, \$k1	

eret

Clear EXL & return

.kdata

save0: .word 0

#Space for saving

save1: .word 0

#used registers

Tratador de exceção: exemplo

.ktext 0x80000180

move \$k1, \$at	#Allow pseudo-intr.
sw \$a0, save0	#Save context
sw \$a1, save1	#
mfc0 \$k0, \$13	#Read Cause
srl \$a0, \$k0, 2	#Extract ExCode
andi \$a0, \$a0, 0xf	
beqz \$a0, done	#Branch if interrupt
move \$a0, \$k0	#Cause as parameter
mfc0 \$a1, \$14	#EPC as parameter
jal print_exp	#print error message

done:

mfc0 \$k0, \$14	#Read EPC
addiu \$k0, \$k0, 4	#Point to next
mtc0 \$k0, \$14	#Update EPC
mtc0 \$0, \$13	#Clear Cause
mfc0 \$k0, \$12	#Read Status
ori \$k0, \$k0, 0x1	#Set IE bit
mtc0 \$k0, \$12	#Update Status
lw \$a0, save0	#Restore context
lw \$a1, save1	
move \$at, \$k1	
eret	#Clear EXL & return

.kdata

save0: .word 0	#Space for saving
save1: .word 0	#used registers

Generalização:

na prática, **switch(ExCode)** redireciona
tratador para rotinas específicas de
tratamento de cada exceção

Conclusões

- **Tratamento de exceções**
 - Parte do núcleo (“**kernel**”) do sistema operacional
 - Requer registradores de **uso específico**
 - » Exemplos: EPC, Cause, Status, BadVAddr, ...
 - Particularidade do MIPS
 - » Instruções especiais: mtc0, mfc0, eret
 - » Registradores reservados para o “kernel”: \$**k0**, \$**k1**
- **A estrutura de um tratador de exceções**
 - Similar à do exemplo estudado
- **A generalização dessa estrutura permitir suporte a:**
 - Outras exceções
 - Rotinas de serviço (“device drivers”)