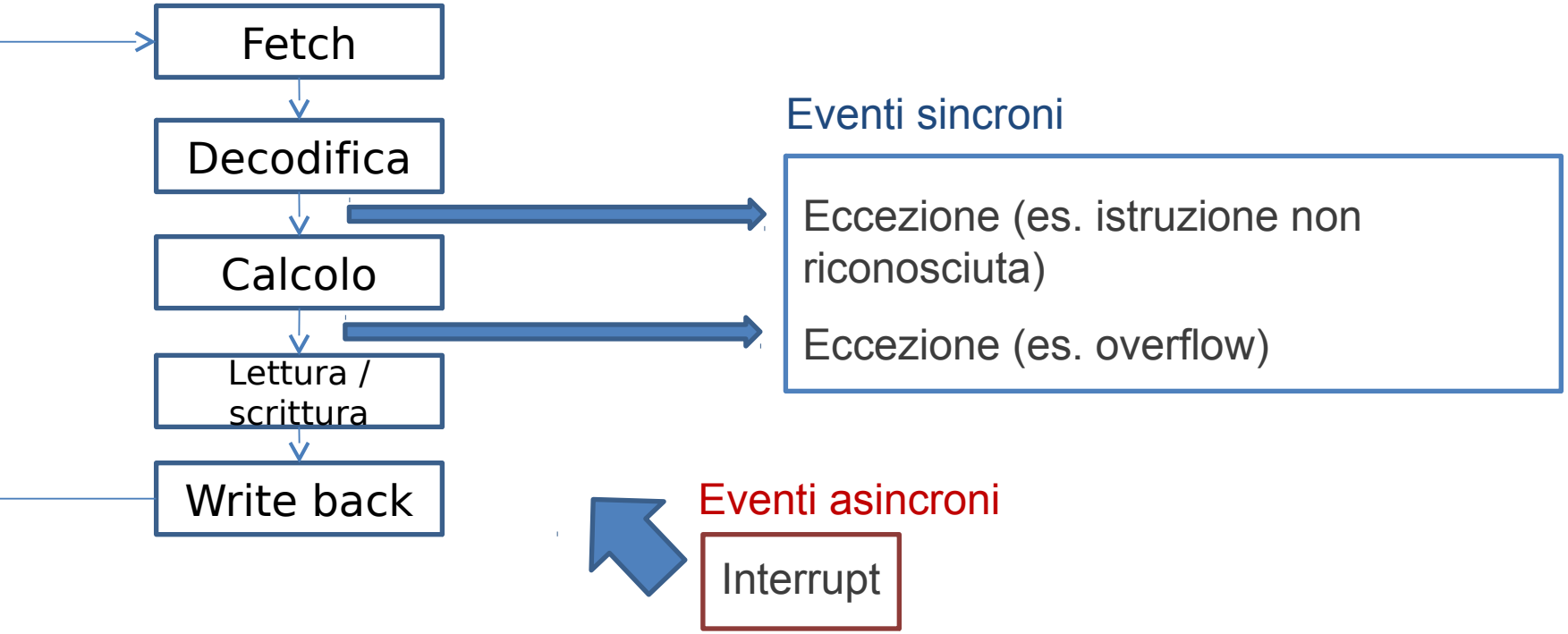
Eccezioni

Università degli studi di Milano
matteo.re@unimi.it
http://homes.di.unimi.it/re/

Eccezioni

- Un’eccezione è un cambiamento **inaspettato** di flusso



Eccezioni

- Eccezione: interna al processore, modifica il flusso di esecuzione di un'istruzione.

- Interrupt: esterno al processore (es. click del mouse), viene generalmente attesa la fine del ciclo dell'istruzione prima di servire l'interrupt.

Type of event	From where?	MIPS terminology
I/O device request	External	Interrupt
Invoke the operating system from user program	Internal	Exception
Arithmetic overflow	Internal	Exception
Using an undefined instruction	Internal	Exception
Hardware malfunctions	Either	Exception or interrupt

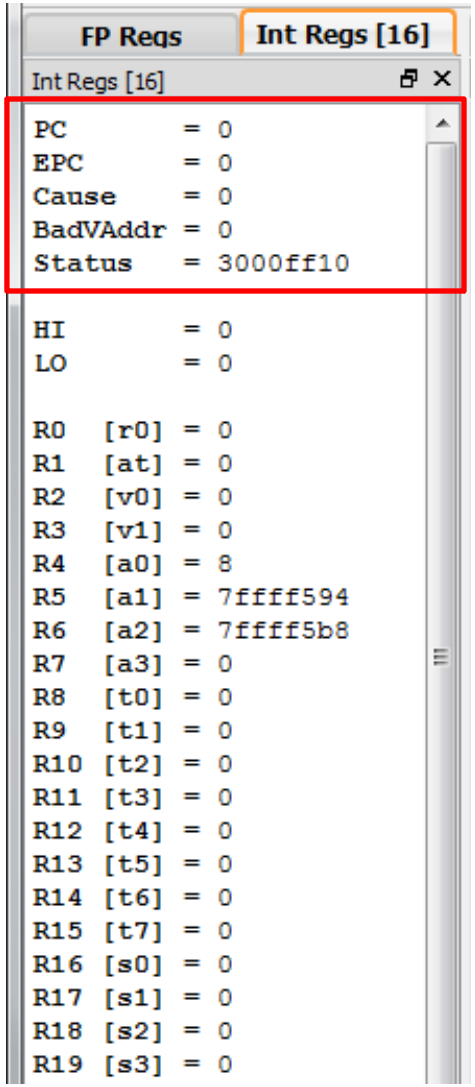
Gestione SW di un’eccezione

- la gestione delle eccezioni è affidata al *coprocessor 0* (che lavora in kernel mode)

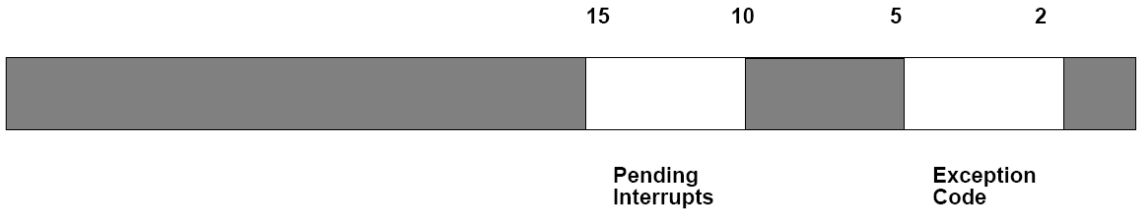
- nei registri di questo coprocessore troviamo le informazioni che il sw necessita per gestire l’eccezione

Register name	Register number	Usage
BadVAddr	8	memory address at which an offending memory reference occurred
Count	9	timer
Compare	11	value compared against timer that causes interrupt when they match
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	address of instruction that caused exception
Config	16	configuration of machine

I registri per la gestione di eccezioni



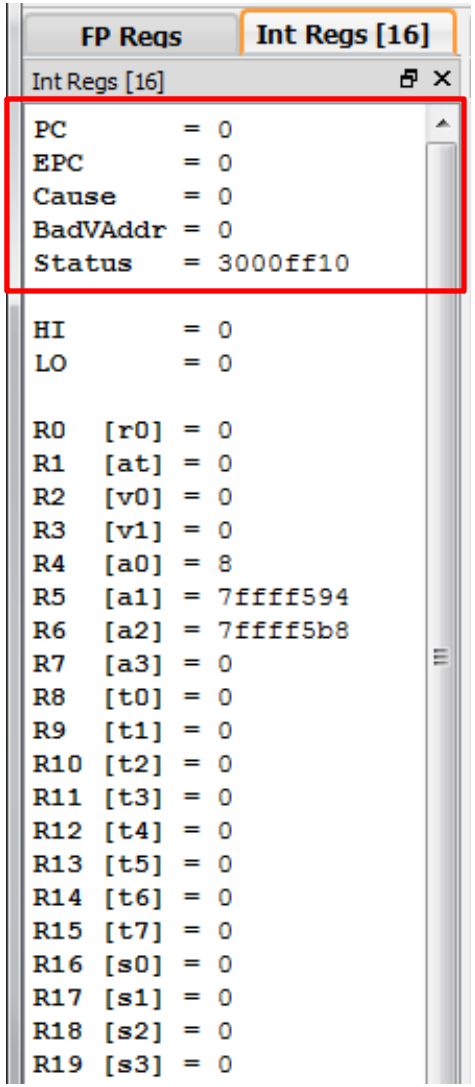
- Registro Cause: codifica la causa dell'eccezione attraverso un **Exception Code**. *(immagine riferita a SPIM)*



- ri

0	INT	External interrupt
4	ADDRL	Address error exception (load or instruction fetch)
5	ADDRS	Address error exception (store)
6	IBUS	Bus error on instruction fetch
7	DBUS	Bus error on data load or store
8	SYSCALL	Syscall exception
9	BKPT	Breakpoint exception
10	RI	Reserved instruction exception
12	OVF	Arithmetic overflow exception

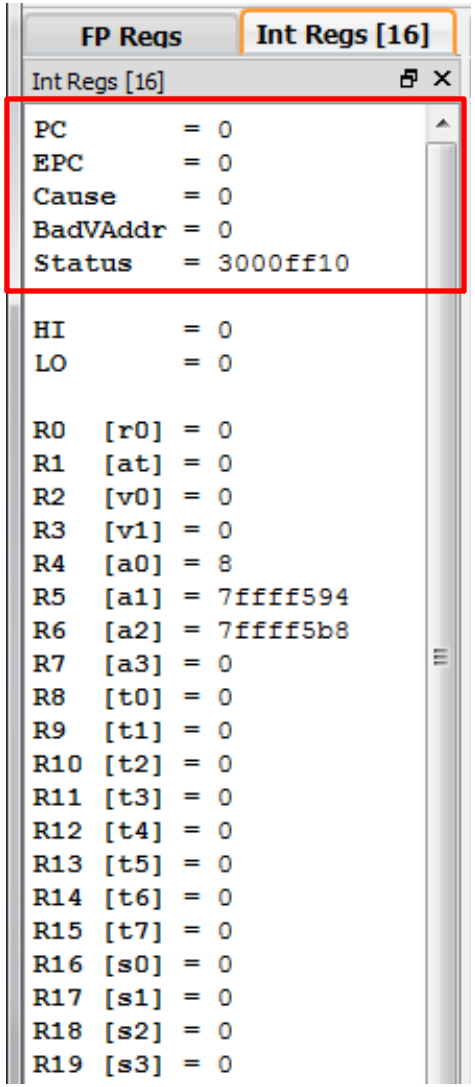
I registri per la gestione di eccezioni



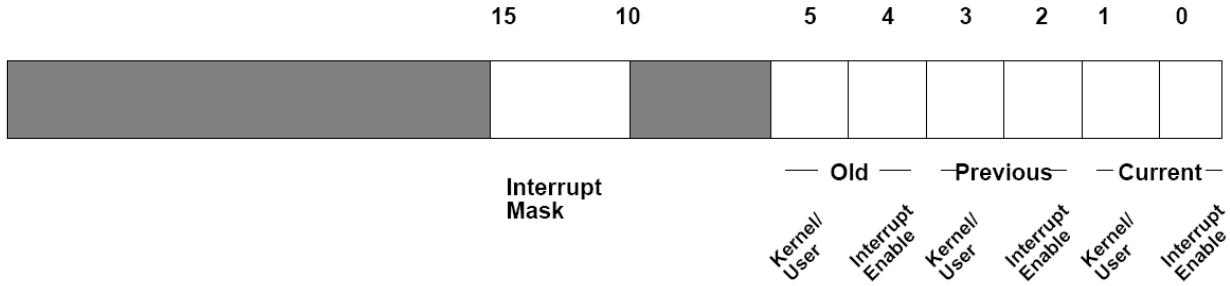
- Registro EPC: *exception program counter*, indirizzo della word in cui sta la *faulting instruction* (è analogo di linking in chiamata a procedura, perché non si usa il registro `$ra`?)

- Registro BadVaddr: indirizzo errato nel caso di una address exception (ad esempio se si effettua per errore un accesso non allineato a un indirizzo X, questo registro riporta X)

I registri per la gestione di eccezioni



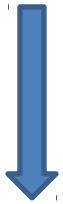
- Registro Status: interrupt mask e bit di controllo *(immagine riferita a SPIM)*:



- Interrupt mask [15-10]: i-esimo bit a 1 se interrupt abilitati a quel livello.
- I bit [5-0] tengono traccia dello stato del programma durante la gestione dell'eccezione:
- Lo stato è dato da due bit:
  - kernel/user: vale 1 se in modalità di esecuzione user, 0 se in modalità kernel;
  - interrupt enable: vale 1 se gli interrupt sono abilitati, 0 altrimenti.
- Quando viene sollevata l'eccezione lo stato *previous* viene copiato nello stato *old* (il cui valore viene quindi perso); lo stato *current* viene copiato nello stato *previous*; entrambi i bit dello stato *current* vengono settati a 0 (la gestione dell'eccezione prevede kernel mode e interrupt disabilitati).
- Al rientro dalla gestione dell'eccezione gli stati *old* e *previous* vengono copiati negli stati *previous* e *current*, rispettivamente; il programma riparte dallo stesso stato in cui era prima dell'eccezione.

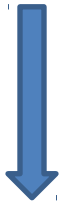
Esempi

```
main:

lui $t0, 0xffff
ori $t0, $t0, 0xffff
Jr $t0
addi $v0, $zero, 10
syscall
```



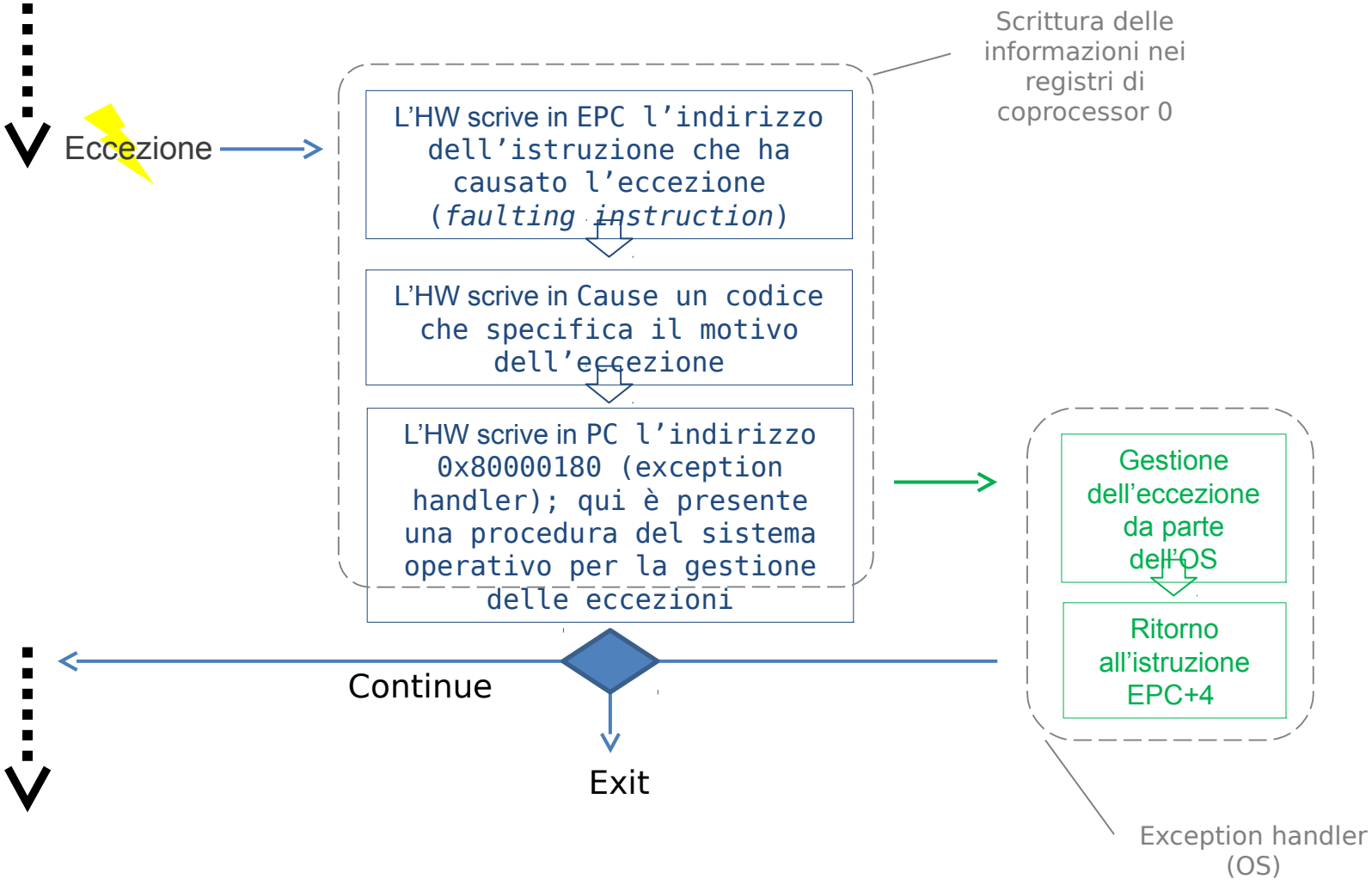
Bad Address in text read

```
main:

lui $t0, 0x8000
lui $t1, 0x8000
add $t2, $t0, $t1
addi $t3, $t2, -100
addi $v0, $zero, 10
syscall
```



Overflow

Gestione SW di un'eccezione



- Vediamo nel dettaglio come vengono svolte queste operazioni dal SW

Gestione SW di un’eccezione

- Come avviene la lettura e scrittura nei registri del coprocessore 0?

 # copiare dal registro $13 del coprocessore 0 (Cause register) al registro $t0

mfc0 $t0, $13

 # copiare dal registro $t0 al registro $14 del coprocessore 0 (EPC)

mtc0 $14, $t0

 # load word dall’indirizzo 100($t3) al registro $13 del coprocessore 0

lwc0 **$13,** 100($t3)

 # store word dal registro $13 del coprocessore 0 in memoria

swc0 **$13**, 50($t2)

Gestione SW di un'eccezione

- L'exception handler non deve alterare lo stato corrente di registri e memoria
  - ha a disposizione due registri riservati `$k0` e `$k1`
  - nel caso debba fare register spilling, non userà lo stack ma la apposita area `.kdata`

Nome	Numero	Utilizzo	Preservato durante le chiamate
$zero	0	costante zero	*Riservato MIPS*
$at	1	riservato per l'assemblatore	*Riservato Compiler*
$v0-$v1	2-3	valori di ritorno di una procedura	No
$a0-$a3	4-7	argomenti di una procedura	No
$t0-$t7	8-15	registri temporanei (non salvati)	No
$s0-$s7	16-23	registri salvati	Si
$t8-$t9	24-25	registri temporanei (non salvati)	No
$k0-$k1	26-27	gestione delle eccezioni	*Riservato OS*
$gp	28	puntatore alla global area (dati)	Si
$sp	29	stack pointer	Si
$s8	30	registro salvato (fp)	Si
$ra	31	indirizzo di ritorno	No

Gestione SW di un’eccezione

- Suddividiamo logicamente la gestione di un’eccezione nelle tre fasi operative di *prologo*, *corpo della procedura* ed *epilogo*;

- *Prologo*: stampa le informazioni relative all’eccezione sollevata;

- *Corpo*: valuta la possibilità di ripristinare il flusso di esecuzione e, nel caso di un interrupt, esegue una procedura specifica per la sua gestione;

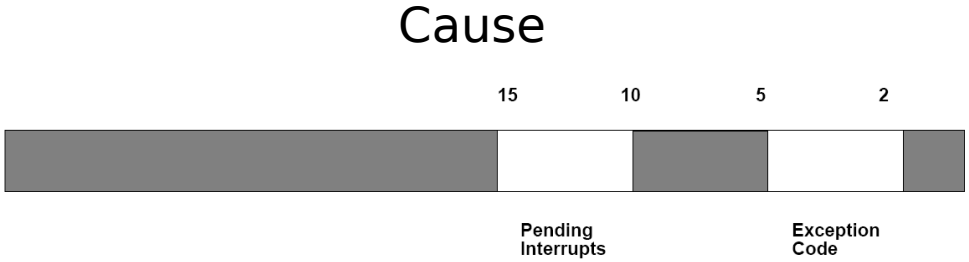
- *Epilogo*: ripristina lo stato del processore e dei registri, fa riprendere l’esecuzione dall’istruzione successiva a quella che ha causato l’eccezione.

Exception Handler (preambolo)

```
.kdata
__m1_:  .asciiz "  Exception "
__m2_:  .asciiz " occurred and
ignored\n"
__e0_:   .asciiz "  [Interrupt] “

[...]

s1:  .word 0
s2:  .word 0
```

Exception Handler (prologo-1)

0	INT	External interrupt
4	ADDRL	Address error exception (load or instruction fetch)
5	ADDRS	Address error exception (store)
6	IBUS	Bus error on instruction fetch
7	DBUS	Bus error on data load or store
8	SYSCALL	Syscall exception
9	BKPT	Breakpoint exception
10	RI	Reserved instruction exception
12	OVF	Arithmetic overflow exception



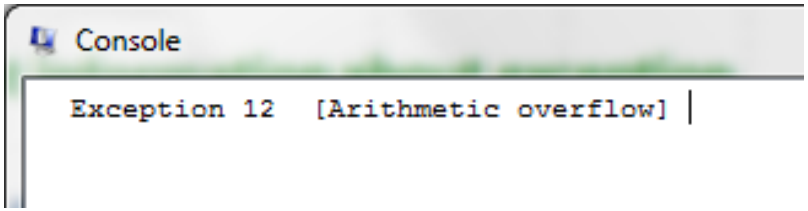

```
move $k1 $at      # salvo il registro  $at
.set at
sw $v0 s1         # salviamo in kdata i registri $v0 e $a0
sw $a0 s2         #

mfc0 $k0 $13# Cause register
srl $a0 $k0 2 # Extract ExcCode Field
andi $a0 $a0 0xf
```

Esempio: Estrazione di ExcCode con Aritmetic Overflow (codice 12):

(1) Cause Register: 00000000000000000000000000 01100 00 *(in Spim vedremo 110000)*
(2) Shift a destra di 2 bit: 00 00000000000000000000000000 01100
(3) AND con 0x1F (11111): 00 00000000000000000000000000 01100

Exception Handler (prologo-2)

```
# Print information about exception.

li $v0 4          # syscall 4 (print_str)
la $a0 __m1_
syscall

li $v0 1          # syscall 1 (print_int)
srl $a0 $k0 2     # Extract ExcCode Fie
andi $a0 $a0 0x1f
syscall

li $v0 4          # syscall 4 (print_str)
andi $a0 $k0 0x3c
lw $a0 __excp($a0)
nop
syscall
```



Exception Handler (corpo-3)

- Prima cosa da fare: controllare se è possibile riprendere l'esecuzione, altrimenti eseguire exit.

- Se è proprio un errore di fetch, controllo se in EPC ho un indirizzo valido (sintatticamente)

```
bne $k0 0x18 ok_pc    # Bad PC exception requires special checks
```

- Se la causa **non** è *«bus error on instruction fetch»* (codice 6 (0x18)) allora $pc è valido, in caso contrario in $pc c'è un valore errato e sono necessari controlli aggiuntivi

```
mfc0 $a0 $14 # copio EPC in $a0
andi $a0 $a0 0x3 # controllo che sia allineato (l'and con 0…011 deve essere 0…0)
beq $a0 0 ok_pc
```

- Nel caso in cui anche EPC non sia valido, non mi resta che fare exit (l'esecuzione non può continuare dopo l'eccezione).

```
li $v0 10          # Exit on really bad PC
syscall
```

Exception Handler (corpo-4)

- Controllo se c’è stato un interrupt. In quel caso verrà gestito con del codice specifico.

```
ok_pc:
li $v0 4          # syscall 4 (print_str)
la $a0 __m2_
syscall

srl $a0 $k0 2  # Extract ExcCode Field
andi $a0 $a0 0xf
bne $a0 0 ret  # se $a0 = 0 allora era un interrupt

# Interrupt-specific code
```

- Se non è un interrupt passo all’epilogo della gestione (ret).

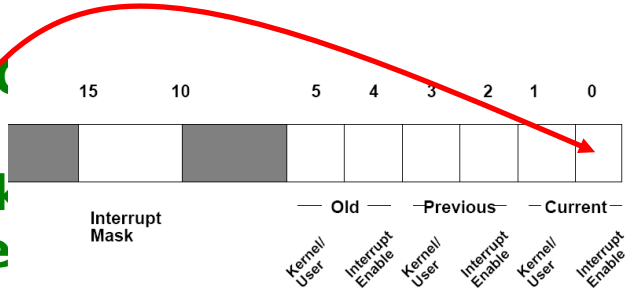
Exception Handler (epilogo-5)

- Calcolo l'indirizzo da cui riprendere l'esecuzione e lo copio in EPC

```
ret:
# ritorno da eccezione (no interrupt)
mfc0 $k0 $14      # copio valore di EPC
addiu $k0 $k0 4   # salto la "faulting instruction"
mtc0 $k0 $14      # aggiorno EPC
```

- Ripristino i registri e lo stato del processore: $v0, $a0, $at, e i registri Cause e Status

```
lw $v0 s1         # ripristino i registri che ho salvato in
kdata
lw $a0 s2

mtc0 $0 $13   # rest del registro C

mfc0 $k0 $12 # copio Status in $k
ori  $k0 0x1     # Interrupts enable
mtc0 $k0 $12  # aggiorno Status

eret          # Ritorno da eccezione (MIPS32)
```



Esempio

- Cosa succede quando facciamo una jump a 0xffffffff (esempio nelle slides precedenti)?