INTERRUPT ED ECCEZIONI

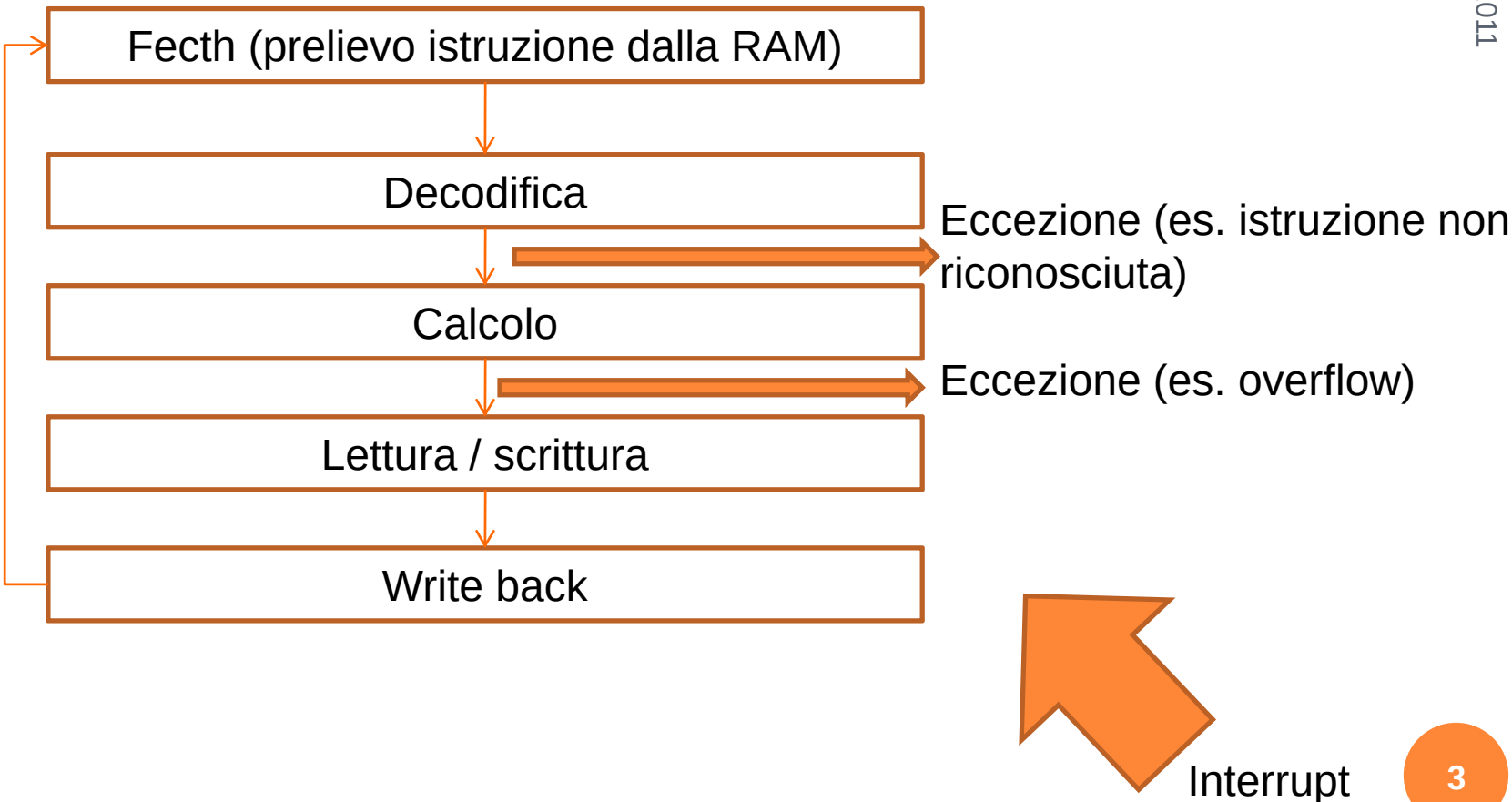
I. Frosio

15 April 2011

SOMMARIO

- Eccezioni ed interrupt – Esempi
- Gestione SW delle eccezioni: i dettagli

CICLO DI ESECUZIONE DI UN'ISTRUZIONE



ECCEZIONI ED INTERRUPT

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

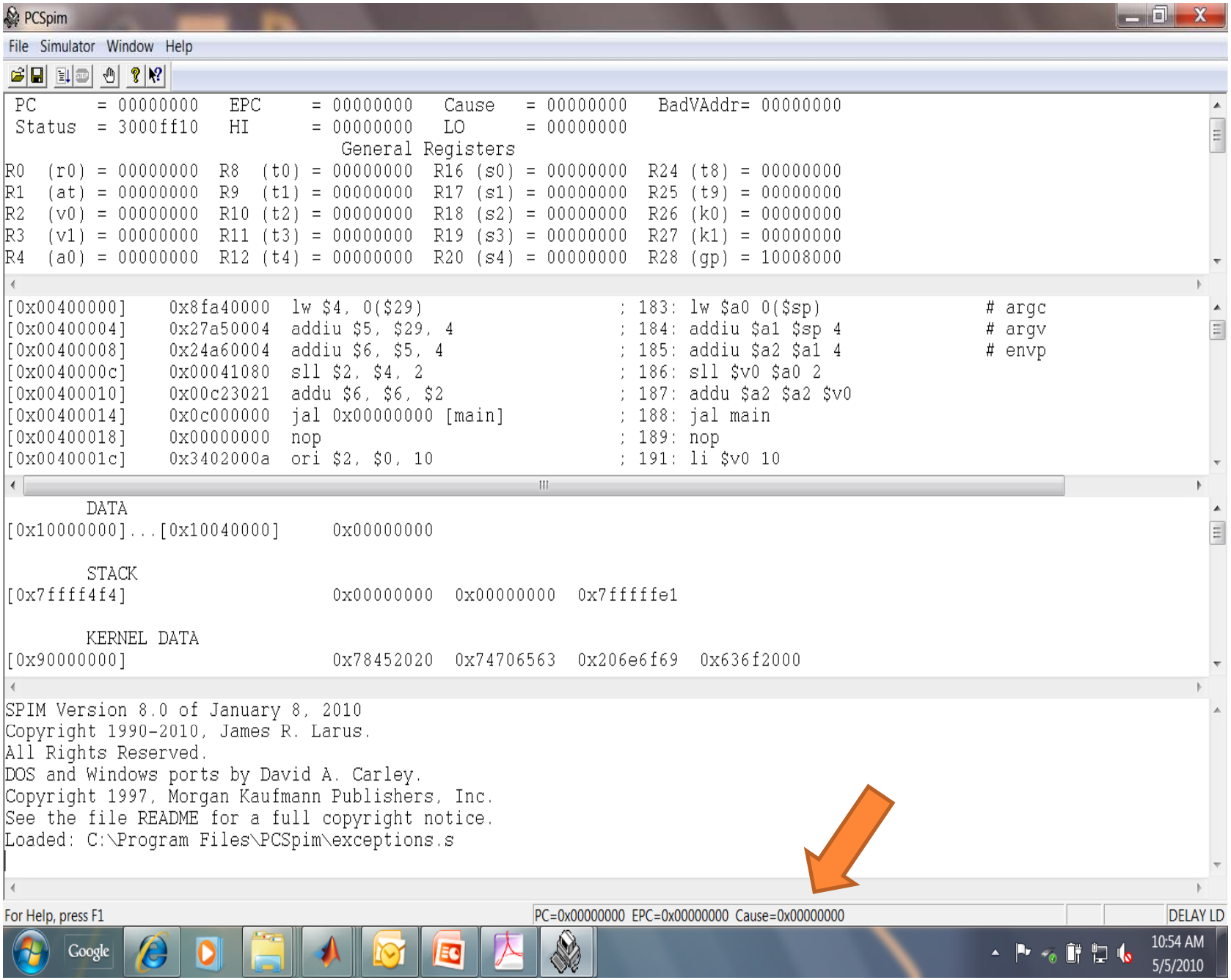
- Interrupt: esterno al processore (es. click del mouse), viene generalmente attesa la fine del ciclo dell'istruzione prima di servire l'interrupt.

Tipo di evento	Provenienza	Terminologia MIPS
Richiesta di un dispositivo di I/O	Esterna	Interrupt
Chiamata al SO da parte di un programma	Interna	Eccezione
Overflow aritmetico	Interna	Eccezione
Uso di un'istruzione non definita	Interna	Eccezione
Malfunzionamento dell'hardware	Entrambe	Eccezione o Interruzione

RISPOSTA ALL'ECCEZIONE

- Interviene il sistema operativo.
- Risposta vettorializzata: salto ad un indirizzo diverso per ogni diverso tipo di eccezione.
- Tramite registro (causa): un unico entry point (0x80000180 in MIPS) per la gestione delle eccezioni; la causa dell'eccezione è nel registro causa.
- Valore del registro causa: 0x00000000 se non ci sono eccezioni, altrimenti contiene il codice dell'eccezione.
- NB: l'HW aggiorna automaticamente il registro causa al verificarsi dell'eccezione. La gestione dell'eccezione è SW (coordinamento HW / SW).

REGISTRO CAUSA IN PC-SPIM



ECCEZIONE: ESEMPIO PC-SPIM (ES. 5.1)

Si esegua il seguente programma:

```
main:
lui $t0, 0xffff                 # $t0=0xffff0000
ori $t0, $t0, 0xffff            # $t0=0xffffffff
jr      $t0                     # jump to 0xffffffff (exception)
addi $v0, $zero, 10
syscall
```

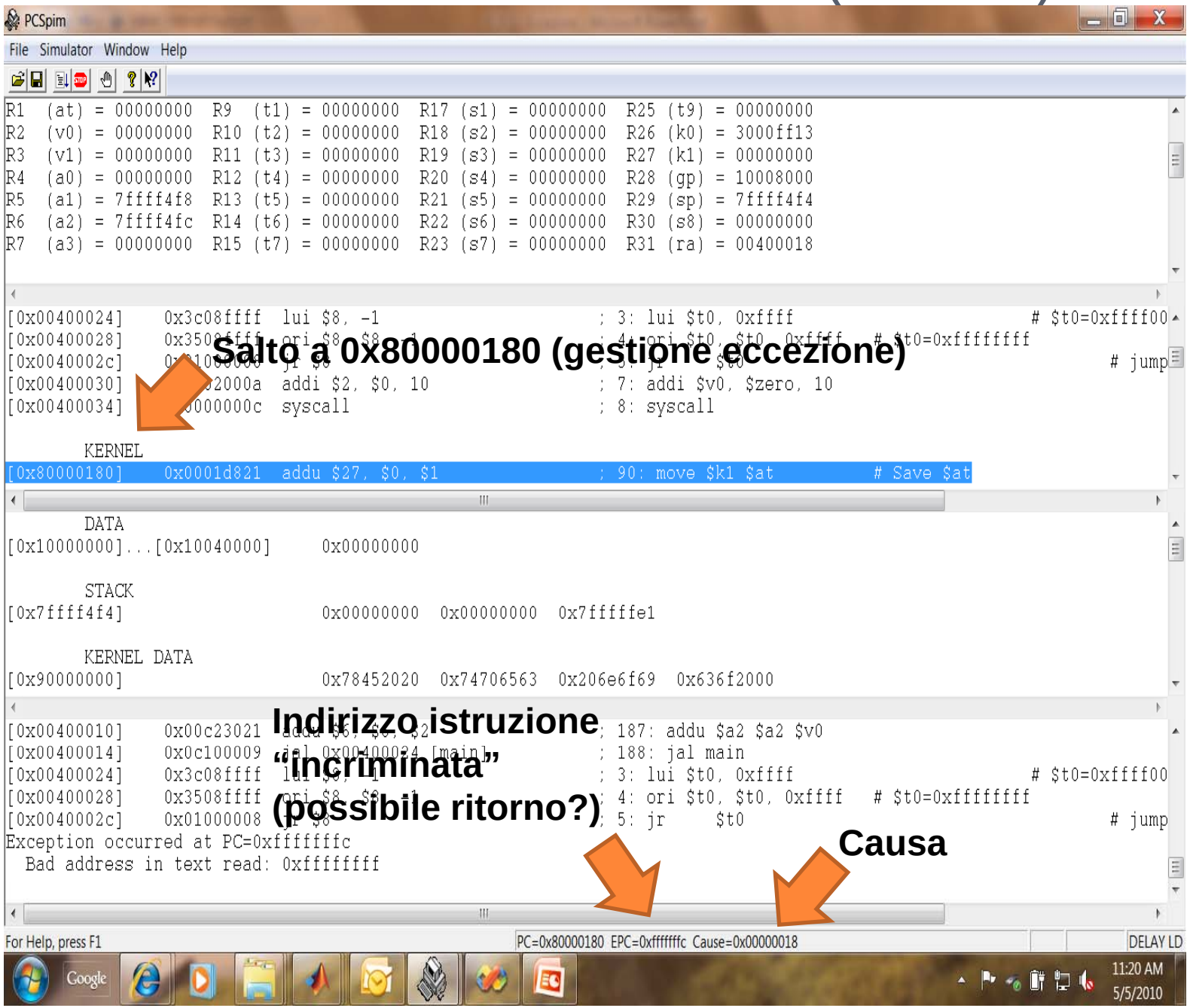
Si analizzi il comportamento del processore MIPS alla chiamata di jr.

ESERCIZIO 5.1 - SOLUZIONE

- Il processore salta a 0xffffffff e carica l'istruzione di questa locazione di memoria;
- Non viene eseguita la fetch(l'indirizzo non contiene infatti alcuna istruzione) -> eccezione!
- Nel registro causa compare il codice di eccezione 0x00000018 (istruzione non può essere decodificata);
- Nel registro EPC compare l'indirizzo 0xfffffffc, per un eventuale ritorno alla riga di codice che ha generato l'eccezione. Questo è l'indirizzo della word che ha generato l'errore (così come supposto dall'HW).

- Exception 6 -> bus error on instruction fetch.

ECCEZIONE: ESEMPIO PC-SPIM (ES. 5.1)



ECCEZIONE: ESEMPIO PC-SPIM (ES. 5.1)

- Non è possibile effettuare jump a 0xffffffff...
- Nel registro EPC viene salvato l'indirizzo dell'istruzione che ha generato l'interruzione (in questo caso 0xfffffffc = 0xffffffff-3), nel caso in cui sia possibile un ritorno dall'eccezione alla normale esecuzione del codice.
- Il "rientro dall'eccezione porta ad una riga di codice non valida"! La procedura di gestione dell'eccezione termina quindi l'esecuzione del programma.

ECCEZIONE: ESERCIZIO ES. 5.2

- main:

- lui $t0, 0x8000 # $t0=0x80000000
- lui $t1, 0x8000 # $t1=0x80000000
- add $t2, $t0, $t1 # Overflow
- addi $t3, $t2, -100 # Next instruction

- addi $v0, $zero, 10
- syscall

- Si genera un overflow. Dopo avere gestito l'eccezione (info sulla console per l'utente), il programma riprende il suo flusso normale.

- Si analizzi il comportamento del registro cause e la gestione dell'eccezione da parte del sistema operativo.

ES. 5.2 - OSSERVAZIONI

- Nel registro cause compare il codice 0x00000030 -> overflow.
- C’è una salto all’indirizzo di ritorno, effettuato tenendo conto del valore di EPC, che fa continuare il flusso del programma.

SOMMARIO

- Eccezioni ed interrupt – Esempi
- Gestione SW delle eccezioni: i dettagli

GESTIONE SW DI UN'ECCEZIONE

- L'HW rileva un'eccezione;
- L'HW aggiorna il contenuto del registro EPC (Exception Program Counter) con l'indirizzo dell'istruzione che ha causato l'eccezione;
- L'HW aggiorna il registro Cause con un codice che specifichi il motivo per il quale si è verificata l'eccezione;
- ...

- Cosa sono i registro EPC e Cause?
- I registri EPC e Cause fanno parte, con altri registri, del cosidetto coprocessore 0...

REGISTRI DEL COPROCESSORE 0

Un insieme di registri speciali, denominati coprocessore 0, per la gestione (tra le altre cose) delle eccezioni.

Nome del registro	Numero del registro in coprocessore 0	Utilizzo
Bad/Addr	8	Registro contenente l'indirizzo di memoria a cui si è fatto riferimento (cf. "page fault").
Count	9	Timer (MIPS: 10ms).
Compare	11	Valore da comparare con un timer. Genera un interrupt.
Status	12	Maschera delle interruzioni e bit di abilitazione. Stato dei diversi livelli di priorità (6 HW e 2 SW).
Cause	13	Tipo dell'interruzione e bit delle interruzioni pendenti
EPC	14	Registro contenente l'indirizzo dell'istruzione che ha causato l'interruzione.

GESTIONE SW DI UN'ECCEZIONE

- L'HW rileva un'eccezione;
- L'HW aggiorna il contenuto del registro EPC (Exception Program Counter) con l'indirizzo dell'istruzione che ha causato l'eccezione;
- L'HW aggiorna il registro Cause con un codice che specifichi il motivo per il quale si è verificata l'eccezione;
- L'HW aggiorna il contenuto del registro PC al valore 0x80000180 (exception handler); qui è presente una procedura del sistema operativo per la gestione delle eccezioni;
- Gestione dell'eccezione da parte dell'OS
- Ritorno all'istruzione EPC+4...

- ... Come è possibile effettuare il ritorno all'istruzione EPC+4?
  - E' necessario utilizzare un jr <r>, ove r è un registro contenente l'indirizzo EPC+4...
  - Come è possibile copiare il contenuto di EPC in un registro generico?

LETTURA / SCRITTURA DEI REGISTRI DEL CORPOCESSORE 0

- **mfc0 $t0, $13** # copia dal registro $13 del coprocessore 0 (Cause register) al registro $t0
- **mtc0 $14, $t0** # copia dal registro $t0 al registro $14 (EPC) del coprocessore 0
- **lwc0 $13, 100($t3)** # carica una word dall'indirizzo 100($t3) al registro $13 del coprocessore 0
- **swc0 $13, 50($t2)** # salva la word nel registro $13 del corpocessore 0 in memoria

GESTIONE SW DI UN'ECCEZIONE

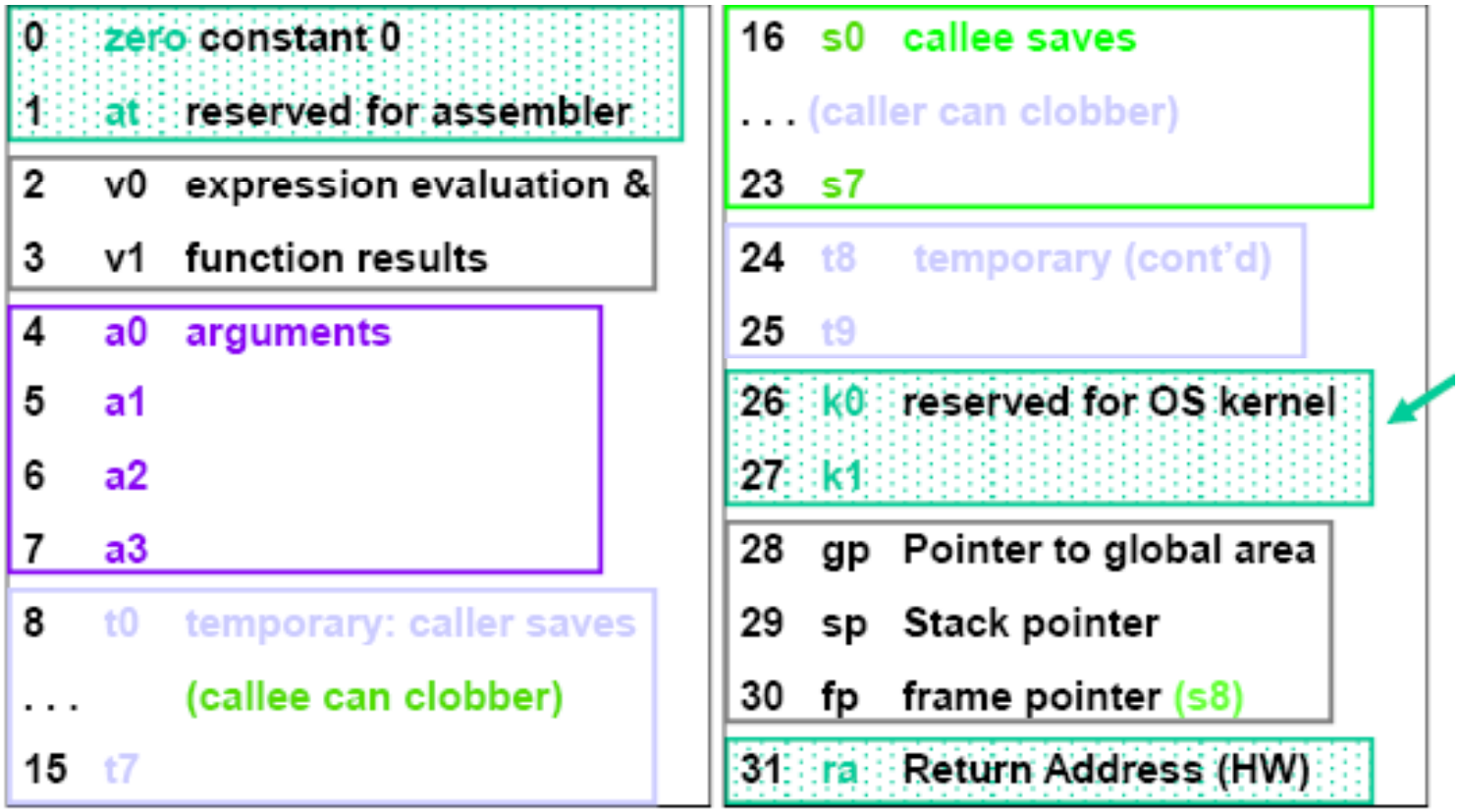
- L'HW rileva un'eccezione;
- L'HW aggiorna il contenuto del registro EPC (Exception Program Counter) con l'indirizzo dell'istruzione che ha causato l'eccezione;
- L'HW aggiorna il registro Cause con un codice che specifichi il motivo per il quale si è verificata l'eccezione;
- L'HW aggiorna il contenuto del registro PC al valore 0x80000180 (exception handler); qui è presente una procedura del sistema operativo per la gestione delle eccezioni;
- Gestione dell'eccezione da parte dell'OS
- Ritorno all'istruzione EPC+4...

- ... Come è possibile effettuare il ritorno all'istruzione EPC+4?
  - Copio il contenuto di EPC in un registro (e aggiungo 4) quale indirizzo di ritorno...
  - mfc0 $t0, $14 # $t0 = EPC
  - addi $t0, $t0, 4 # $t0 +=4

GESTIONE SW DI UN'ECCEZIONE

- ... Come è possibile effettuare il ritorno all'istruzione EPC+4?
  - Copio il contenuto di EPC in un registro (e aggiungo 4) quale indirizzo di ritorno...
  - mfc0 $t0, $14 # $t0 = EPC
  - addi $t0, $t0, 4 # $t0 +=4
- Ma in questo modo alteriamo il registro $t0...
- Non possiamo utilizzare lo stack per salvare $t0 perchè lo stack potrebbe essere corrotto...
- Il registro $ra non è a disposizione...
- Sono necessari dei registri dedicati alla gestione delle eccezioni!

REGISTRI PER LA GESTIONE DELLE ECCEZIONI



GESTIONE SW DI UN'ECCEZIONE

- L'HW rileva un'eccezione;
- L'HW aggiorna il contenuto del registro EPC (Exception Program Counter) con l'indirizzo dell'istruzione che ha causato l'eccezione;
- L'HW aggiorna il registro Cause con un codice che specifichi il motivo per il quale si è verificata l'eccezione;
- L'HW aggiorna il contenuto del registro PC al valore 0x80000180 (exception handler); qui è presente una procedura del sistema operativo per la gestione delle eccezioni;
- Gestione dell'eccezione da parte dell'OS
- Ritorno all'istruzione EPC+4...

- ... Come è possibile effettuare il ritorno all'istruzione EPC+4?
  - Copio il contenuto di EPC in un registro (e aggiungo 4) quale indirizzo di ritorno...
  - mfc0 $k0, $14 # $k0 = EPC
  - addi $k0, $k0, 4 # $k0 +=4

RISPOSTA AD UN'ECCEZIONE

- 1) Prologo
- 2) Risposta all'eccezione (exception handler)
- 3) Epilogo

RISPOSTA AD UN'ECCEZIONE: PROLOGO

1. salvataggio in EPC del PC corrente (HW);
2. caricamento nei bit 2-5 del registro causa, $13, del codice dell'eccezione vericatasi (HW);
3. salvataggio dello stato;
4. caricamento nel PC del valore 0x80000180 (HW).

Non posso salvare lo stato (il contenuto dei registri del register file) in stack (perchè lo stack potrebbe essere causa dell'eccezione), salvo perciò in $k0, $k1 e in .kdata. i valori necessari.

Il segmento di dati .kdata svolge il ruolo dello stack per la procedura di risposta ad interrupt ed eccezioni.

RISPOSTA AD UN'ECCEZIONE: EPILOGO

1. Pulire il cause register e salvarlo;
2. Settare a 0 il bit di interrupt corrispondente e salvare lo status register;
3. Ripristino dello stato;
4. Caricamento nel PC del valore EPC + 4.

EXPECPTION HANDLER: PC-SPIM (1)


```
0x80000180:
move $k1 $at            # Save $at
sw $v0 s1               # Not re-entrant and we can't trust $sp
sw $a0 s2               # But we need to use these registers
```

Salva $at, $v0, $a0 nel segmento .kdata

```
mfc0 $k0 $13            # Cause register
```

Cause in $k0

```
srl $a0 $k0 2           # Extract ExcCode Field
andi $a0 $a0 0x1f
li $v0 4                # syscall 4 (print_str)
la $a0 __m1_
syscall
li $v0 1                # syscall 1 (print_int)
srl $a0 $k0 2           # Extract ExcCode Field
andi $a0 $a0 0x1f
Syscall
li $v0 4                # syscall 4 (print_str)
```

srl -> Shift right logical by a constant number of bits

Stampa info per l'utente

EXPECPTION HANDLER: PC-SPIM (CONT. 2)


```
andi $a0 $k0 0x3c
lw $a0 __excp($a0)
nop
syscall
```

Stampa info per l'utente

Se PC non è corrotto

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

```
Nop
mfc0 $a0 $14                # EPC
```

$a0 = EPC

```
andi $a0 $a0 0x3  # Is EPC word-aligned?
beq $a0 0 ok_pc
```

Se EPC non è corrotto

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

Gestione "speciale" per (E)PC corrotto [Exit]

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

Stampa info per l'utente

EXPECPTION HANDLER: PC-SPIM

```
srl $a0 $k0 2          # Extract ExcCode Field
```

$a0 = Exception code

```
andi $a0 $a0 0x1f
bne $a0 0 ret          # 0 means exception was an interrupt
nop
```

Se è un interrupt...

```
ret: mfc0 $k0 $14      # Bump EPC register
addiu $k0 $k0 4        # Skip faulting instruction
mtc0 $k0 $14
```

$k0 = EPC +4
EPC = $k0

```
lw $v0 s1              # Restore other registers
lw $a0 s2
move $at $k1           # Restore $at
```

Ripristina valori originali dei registri

```
mtc0 $0 $13            # Clear Cause register
mfc0 $k0 $12           # Set Status register
ori  $k0 0x1           # Interrupts enabled
mtc0 $k0 $12
```

Clear

```
eret
```

Ritorno

RISPOSTA AD UN'ECCEZIONE: PROLOGO

- ***Exception handler (0x80000180):***
  - ***stampa un messaggio solo se si verifica un'eccezione.***
  - ***Il messaggio conterrà l'indirizzo (in $a0) ed il codice dell'eccezione (in $a1).***
  - ***Non fa nulla se si verifica un interrupt.***

- Occorrerà identificare se si tratta di un'eccezione (allo scopo controllo se i bit che identificano le eccezioni sono tutti a 0 nel registro causa).

- Inserirò quindi in $a0 il codice dell'eccezione ed in $a1 l'indirizzo (dovrò quindi prima salvare il contenuto di $a0 e $a1 in .kdata).

- Per il corpo della procedura dovrò copiare i registri da coprocessore 0 a register file per poterne elaborare il contenuto.

```
.ktext 0x80000180
# Prologo della procedura, inizia dall'entry point

la $k0, save0          # Handler is not re-entrant and can't use
sw $a0, 0($k0)         # stack to save $a0, $a1 (data required are
sw $a1, 4($k0)         # stored in kernel data segment)

.kdata
save0: .word 0, 0
```

$k0, $k1 (e $at) sono considerati registri temporanei riservati al SO. Non devono quindi essere salvati in .kdata

RISPOSTA AD UN'ECCEZIONE: CORPO DELLA PROCEDURA

- # Corpo della procedura

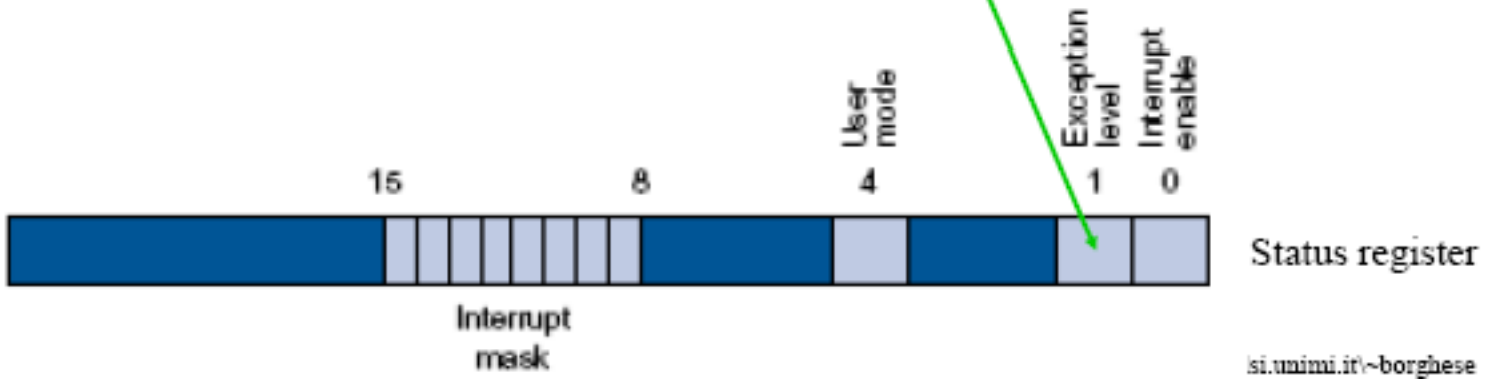
```
mfc0 $k0, $13              # Move Cause into $k0 of register file
srl $k0, $k0, 2            # Prepare ExcepCode field at position 0
andi $a0, $k0, 0x1f        # Extract ExcCode field (5 bits)and prepare it for printing (insert in a0)
beq $a0, $zero, epilogo    # Branch if ExcepCode=0: not an exception
mfc0 $k0, $12              # Status register into $k0
andi $k0, $k0, 0xfffe      # Mask all interrupts
ori $k0, $k0, 0x2          # Interrupt at exception level
mtco $12, $k0
mfc0 $a1, $14               # Move EPC into $a1
jal print_excp             # Print exception error message.
                           # Parameters are in $a0, $a1
```

- *Quando arriva un interrupt, imposta il suo bit di interrupt corrispondente nel registro Causa, anche se nello status register il bit corrispondente nella maschera è disabilitato.*
- *Quando un interrupt è pending, interromperà il processore, quando il suo bit corrispondente nella sua maschera nello status register è stato abilitato.*

RISPOSTA AD UN'ECCEZIONE: EPILOGO

```
        mtc0 $13, $0          # Clear Cause register
        mfc0 $k0, $12         # To fix Status register (bring status in k0)
        andi $k0, 0xfffd      # Clear EXL bit (Exception level, mask bit)
                              # in status register
        ori $k0, 0x1          # Enable interrupts in status register
        mtc0 $12, $k0         # Restore status register

Epilogo: mfc0 $k1, $14        # Bump EPC into $k0
        addiu $k0, $k1, 4     # Do not reexecute faulting instruction:
                              # EPC = EPC + 4
        mtc0 $14, $k1         # Write EPC + 4 in EPC
        la $k0, save0         #
        lw $a0, 0($k0)        # Restore previously saved registers
        lw $a1, 4($k0)
        jr $k1                # Return to EPC address (eret)
```



ESERCIZIO 5.4

- Si analizzi il codice exception.s;
- Si identifichi la porzione di codice ove è possibile inserire la gestione degli interrupt (es. ctrl+C).