# Eccezioni

Esercitazione

Architettura degli elaboratori

# Eccezioni

Variazioni rispetto alle condizioni normali di funzionamento della CPU

- Asincrone – indipendenti dall'esecuzione del programma in corso – e.g. provocate dalle periferiche di I/O
- Sincrone
  - Richieste dal programmatore – e.g. syscall, break
  - Involontarie: e.g. elaborazioni con valori non-ammissibili (Overflow aritmetico, generato dalla ALU durante la fase di execute) o accesso a indirizzi di memoria non corretti (e.g. a indirizzi di memoria non allineati o fuori dallo spazio del processo, generato dall'unità di controllo durante la fase di decodifica)

| 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 |

Conseguenze: cambiamento del normale flusso di esecuzione di un programma → in tutti i casi (in MIPS), il flusso di esecuzione passa ad una locazione di memoria prefissata (0x8000 0180), dove si trova il gestore delle eccezioni (*exception handler*)

# Registri Co-processore 0 per gestione eccezioni

- Nei processori MIPS, le informazioni necessarie per la gestione delle eccezioni (e interruzioni) sono trattate da una parte della CPU denominata **co-processore 0**
- Per accedere ai *registri del co-processore 0* sono utilizzate le istruzioni:
  - mfc0 – move from coprocessor 0
  - mtc0 – move to coprocessor 0
- N.B. in SPIM, sono presenti **solo alcuni** dei registri del co-processore 0:

| 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 |

# Registro Cause e possibili cause di eccezione

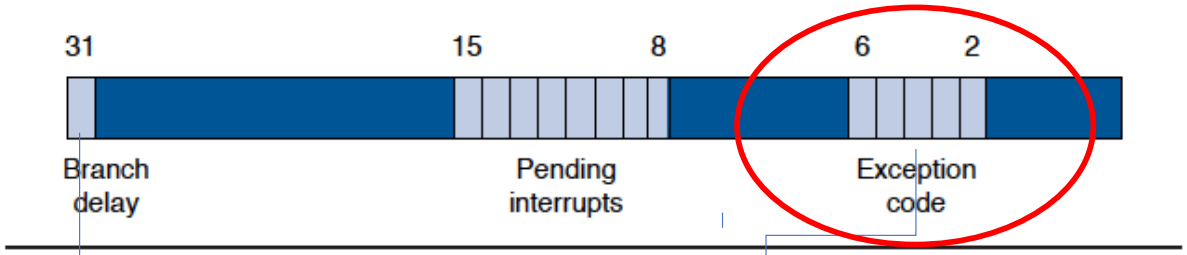



**FIGURE A.7.2 The Cause register.**

1 se l'eccezione è avvenuta durante esecuzione di un'istruzione del branch delay slot

| 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 |

# Esercizio

- Quale eccezione è stata sollevata se il registro Cause contiene il valore 0x0030?

# Soluzione

- Quale eccezione è stata sollevata se il registro Cause contiene il valore 0x0030?

0x0030 = 0000 0000 0011 0000

bit[6-2] = 01100

Codice eccezione 12 → Arithmetic Overflow

# Gestione delle eccezioni

Per tutti i tipi di eccezione, a livello **hardware** è necessario:

- Salvataggio in **EPC** dell'indirizzo dell'istruzione che ha causato l'eccezione (PC-4)
- Aggiornamento di PC con l'indirizzo **0x80000180**
- Scrittura nel registro **Cause** (nei 5 bit da bit_2 a bit_6) del codice dell'eccezione

Avverrà poi, a carico del **software**

- Esecuzione della procedura di gestore delle eccezioni – **exception handler** (collocata nello spazio kernel text a partire dalla locazione 0x80000180)
- Ripristino dello stato che si aveva al momento dell'eccezione (sarà svolta dal gestore dell'eccezione con la richiesta di esecuzione dell'istruzione **eret**)

# FSM modificata per includere la gestione hardware di 2 eccezioni

**stato 10 - codice operativo non corrispondente a nessuna istruzione valida (fase decode)**

**stato 11 - overflow aritmetico (fase execute istruzioni aritm-logica)**

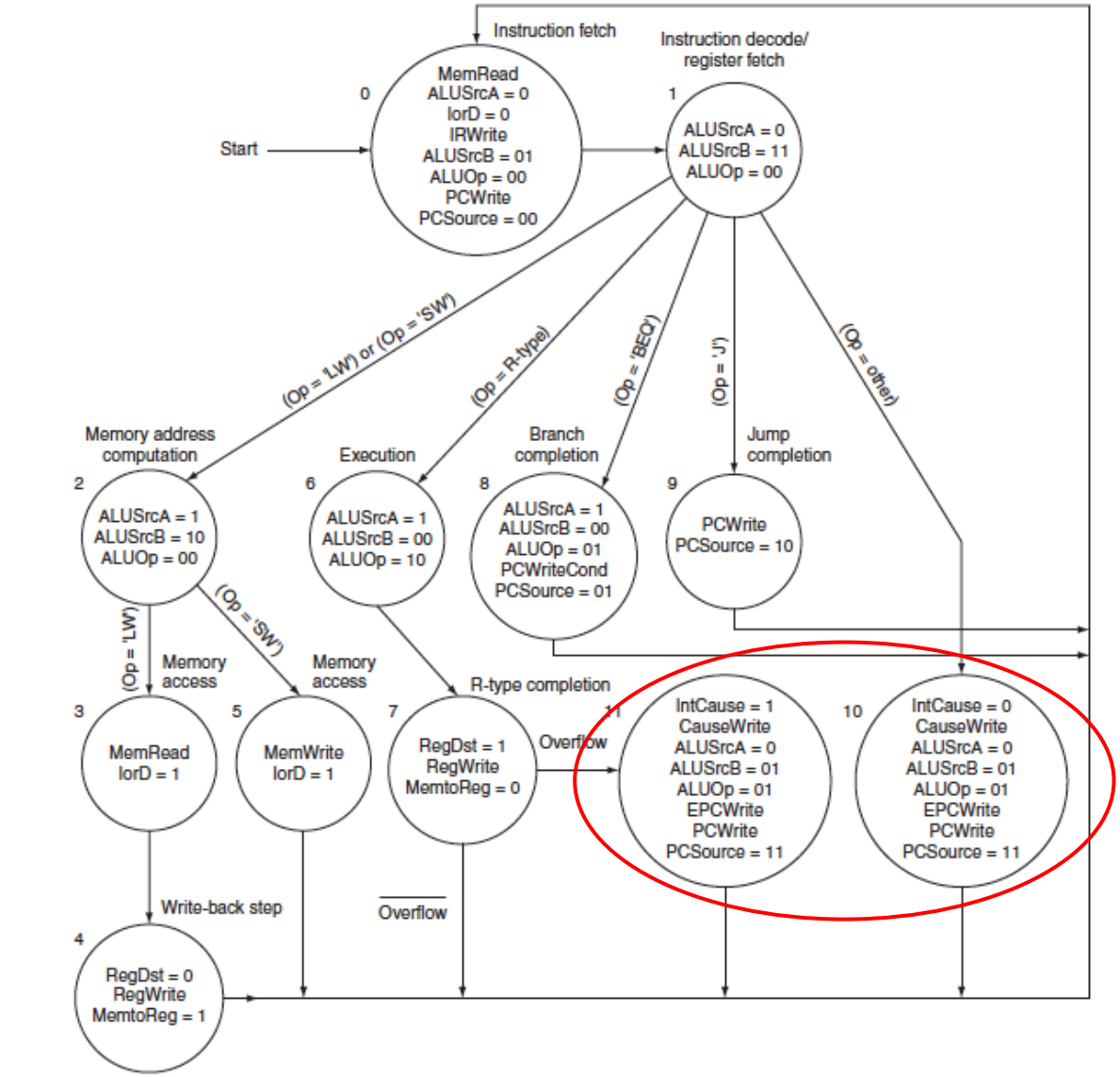
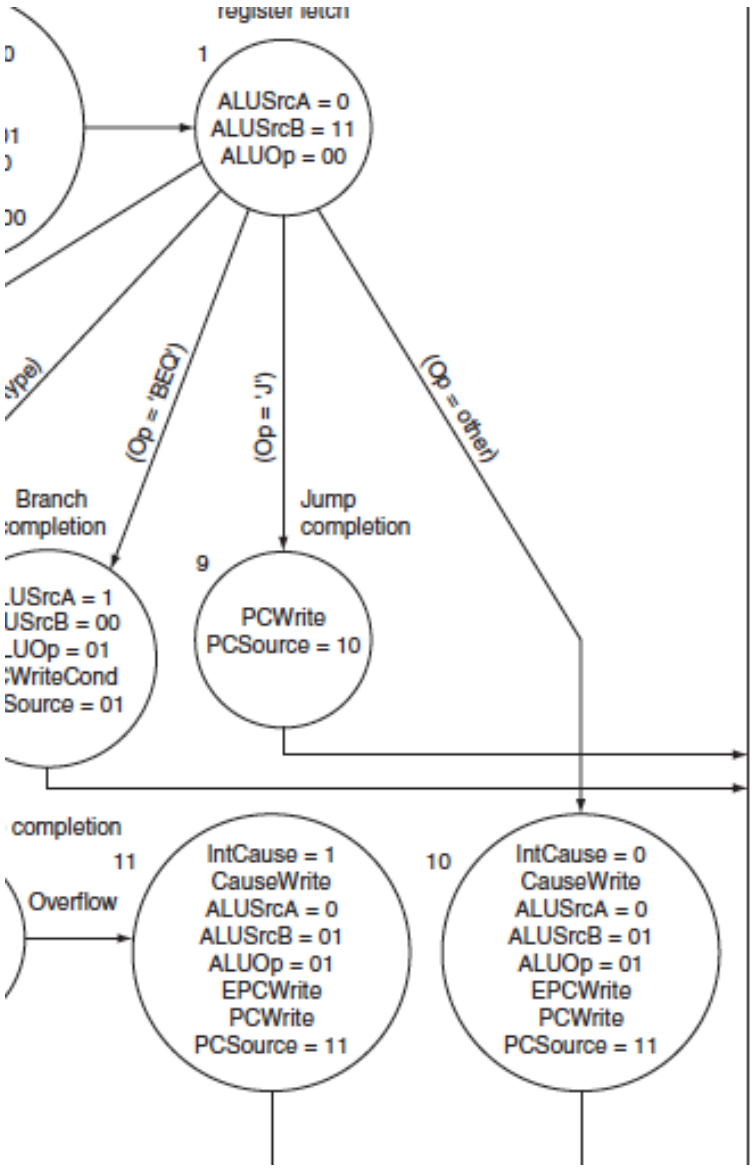
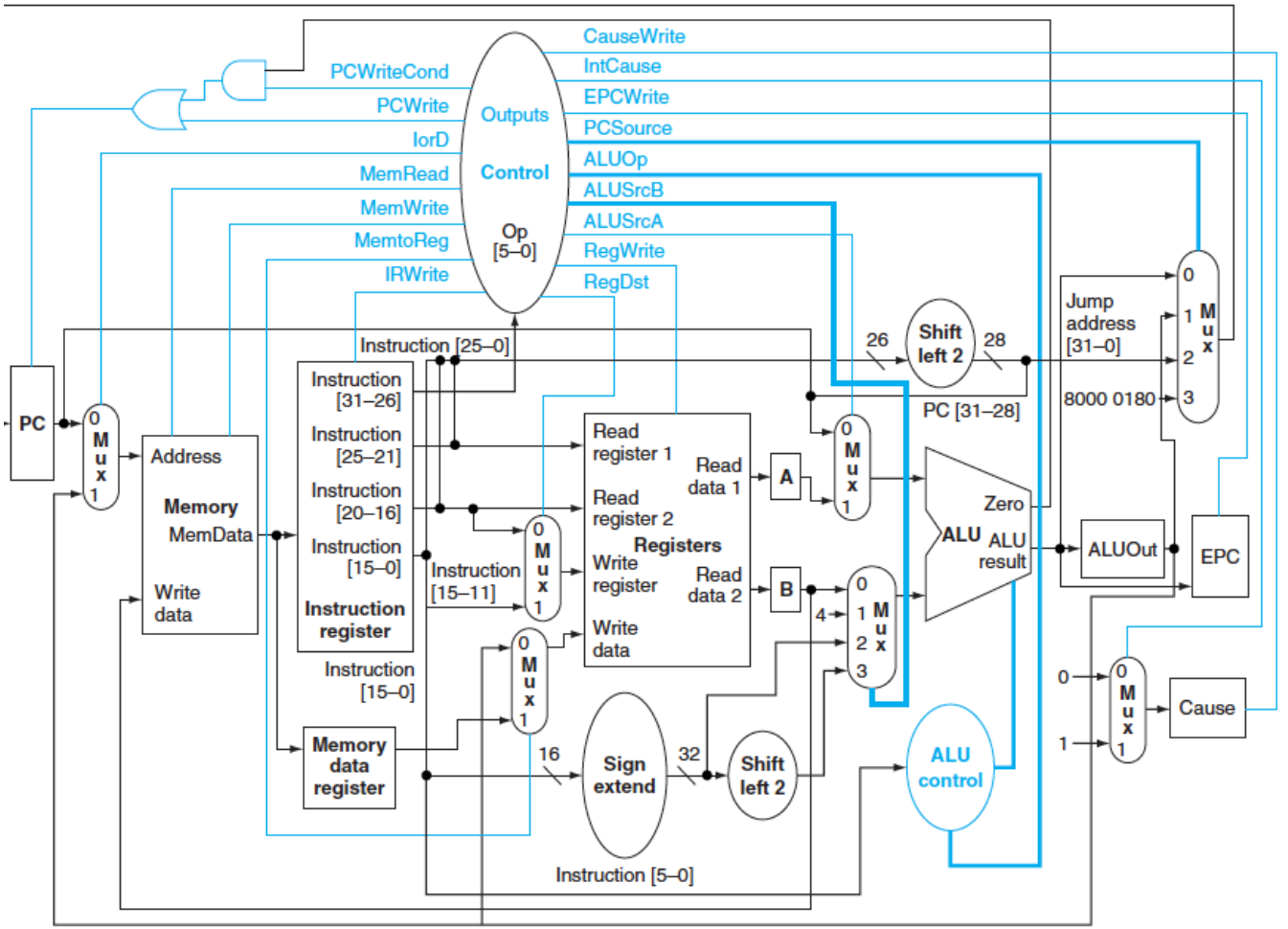



**FIGURE 5.40 This shows the finite state machine with the additions to handle exception detection.** States 10 and 11 are th

N.B. Schema fig. 5.39 (pag 344) relativo a SOLA eccezione Istruzione non valida
Per l'eccezione di Overflow dovremmo avere l'uscita di overflow dalla ALU

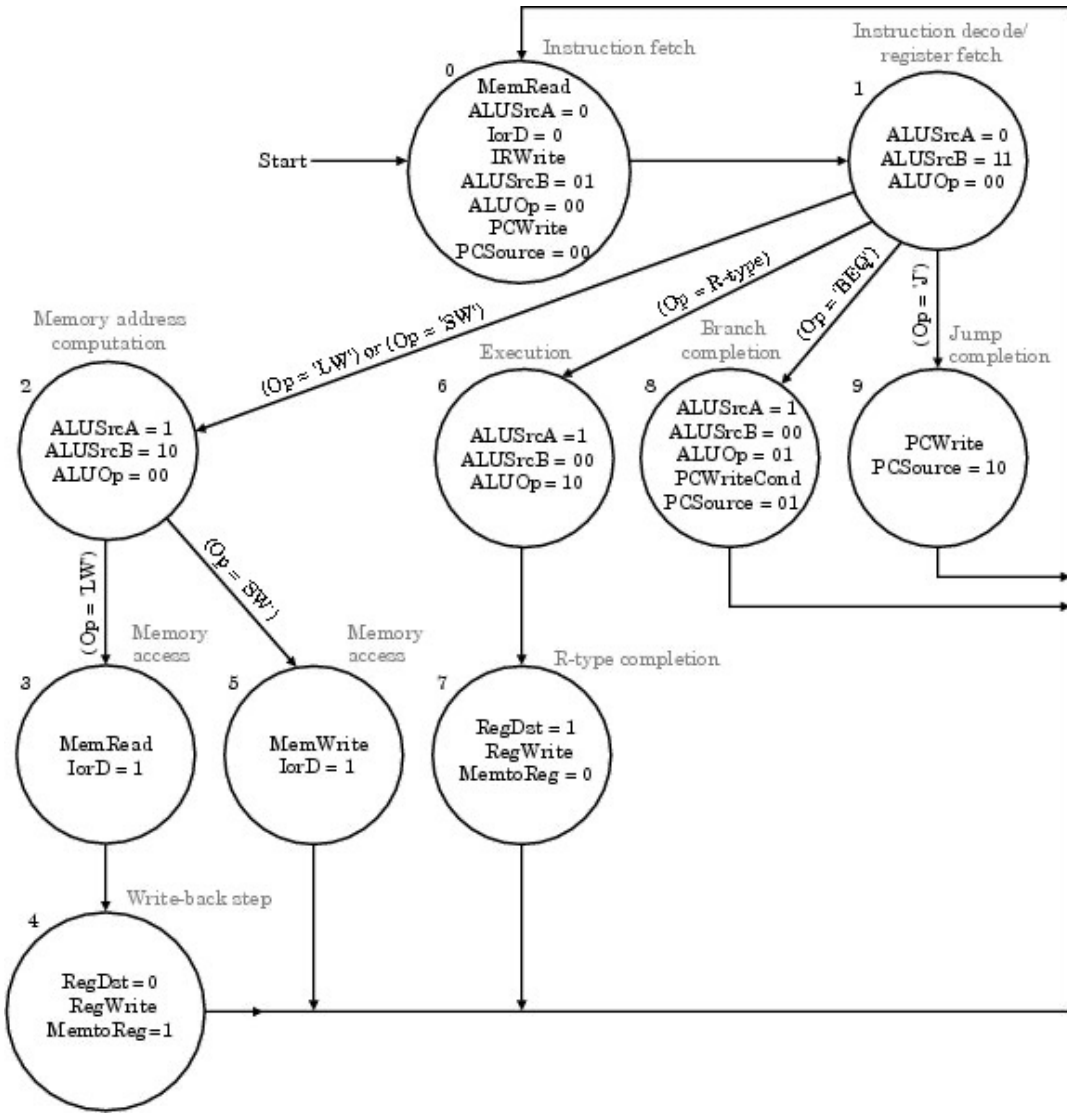



**GURE 5.39 The multicycle datapath with the addition needed to implement exceptions.** The specific additions include the Cause

# Esercizio

Facendo riferimento alla FSM dell'automona di controllo del datapath multiciclo, si indichi tra le risposte la descrizione corretta della/delle righe della funzione di transizione di stato associata allo **stato 2**



# Esercizio – possibili risposte

- 2 righe: la **prima** con in input solo i bit Op per LW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 3, la **seconda** con in input 1 per il valore di Overflow e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 5
- 2 righe, entrambe con 1 per l'input Overflow: la **prima** con in input i bit Op per LW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 3, la **seconda** con in input i bit Op per SW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 5
- 2 righe, entrambe con irrilevante l'input Overflow: la **prima** con in input i bit Op per LW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 3, la **seconda** con in input i bit Op per SW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 5
- 2 righe entrambe con irrilevante l'input Overflow: la **prima** con in input solo i bit Op per LW, e in output i bit che codificano lo stato 3, la **seconda** con in input solo i bit Op per SW, e in output i bit che codificano lo stato 5
- Nessuna delle altre è corretta

# Esercizio – soluzione

- 2 righe: la **prima** con in input solo i bit Op per LW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 3, la **seconda** con in input 1 per il valore di Overflow e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 5
- 2 righe, entrambe con 1 per l'input Overflow: la **prima** con in input i bit Op per LW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 3, la **seconda** con in input i bit Op per SW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 5
- 2 righe, entrambe con irrilevante l'input Overflow: la **prima** con in input i bit Op per LW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 3, la **seconda** con in input i bit Op per SW e i bit che codificano lo stato 2, e in output i bit che codificano lo stato 5
- 2 righe entrambe con irrilevante l'input Overflow: la **prima** con in input solo i bit Op per LW, e in output i bit che codificano lo stato 3, la **seconda** con in input solo i bit Op per SW, e in output i bit che codificano lo stato 5
- Nessuna delle altre è corretta

# Gestione software delle eccezioni
# Exception Handler

- Porzione di codice (assembly) che contiene le istruzioni per **gestire le eccezioni**
- Si trova alla locazione fissa **0x80000180** dell'area ktext riservata al sistema operativo
- Dal momento che viene eseguito dopo l'interruzione di un programma utente in un punto qualsiasi, è tenuto a **preservare il contenuto di TUTTI i registri** macchina
- Ad esso sono **riservati due registri $k0, $k1** che può utilizzare senza doverne preservare il contenuto
- Al termine del trattamento dell'eccezione, l'exception handler ritorna il controllo al programma interrotto (tramite istruzione **eret**)
- MIPS rileva e tratta le eccezioni **PRIMA** del completamento dell'esecuzione della istruzione corrente:
  - se l'eccezione corrente era una interruzione, dopo la sua gestione, l'esecuzione del programma deve riprendere dall'istruzione corrente (salvata nel registro EPC)
  - se si tratta di un altro tipo di eccezione ed è possibile proseguire (se non è avvenuto un errore irreparabile), l'esecuzione deve riprendere **dall'istruzione successiva (EPC+4)**

# myExceptionHandler.s

Il file **myExceptionHandler.s** contiene un esempio di codice assembly per la gestione delle eccezioni in cui sono gestite alcune eccezioni di tipo sincrono (ovvero, NON interrupt generati da periferica esterna).

**Questo** gestore delle eccezioni:

- salva i registri usati al suo interno (a parte i registri riservati $k0 e $k1)
- analizza il contenuto del registro Cause ($13 del co-processore 0)
- *stampa un messaggio contenente il nome e codice dell'eccezione*
- ripristina i registri utilizzati (a parte i registri riservati $k0 e $k1)
- aggiorna il contenuto di EPC:=EPC+4
- ritorna il controllo al programma in cui è stata generata l'eccezione

# Esempio – overflow aritmetico

Dopo aver verificato che nelle impostazioni di QTSPIM sia stato caricato il file myExceptionHandler.s come gestore delle eccezioni

Caricare ed eseguire il seguente programma che: Genera un overflow aritmetico provando ad addizionare "con segno" due numeri positivi la cui somma NON è rappresentabile in complemento a 2

```
.text
main:
li   $t0, 0x7fffffff        # massimo positivo in complemento a 2
li   $t1, 2                 # aggiungo abbastanza per superare il massimo positivo
add  $a0, $t0, $t1          # istruzione che genera l'eccezione di overflow
li   $v0, 1                 # stampa il risultato dell'operazione aritmetica
syscall
jr   $ra                    # fine
```

# Esempio – overflow aritmetico (in QTSPIM)

- Quando avviene l'eccezione sulla Message Window appare un messaggio "Exception occured at PC=0x00400034 Arithmetic overflow" – tale stampa è effettuata internamente al simulatore e NON da myExceptionHandler.s che non ha ancora iniziato l'esecuzione
- PC ha valore 0x80000180 (indirizzo iniziale del gestore delle eccezioni)
- EPC ha valore 0x400034 (indirizzo della istruzione che ha causato l'eccezione)
- Cause ha valore 0x00000030 (**N.B. Cause[6-2] = $01100_2 = 12_{10}$**)
- Con l'esecuzione di myExceptionHandler.s, nella finestra di Console (output di QTSPIM) è visualizzato il messaggio scritto dall'exception handler myExceptionHandler.s

# myExceptionHandler.s – 1

N.B. per QTSPIM dovremo usare la direttiva .set noat (e poi .set at) per abilitare la possibilità di utilizzare il registro $at

```
.kdata
save0: word 0
save1: word 0

.ktext 0x80000180
move  $k1, $at

sw    $v0, save0
sw    $a0, save1


...
```

```
# kernel text segment (inizia a 0x80000180)
# salvataggio $at (usato nelle pseudo-instruzioni
# dall'exception handler)
# salvataggio $v0 (usato da myExceptionHandler.s)
# salvataggio $a0 (usato da myExceptionHandler.s)
#non serve salvare $k0 e $k1
#perché per convenzione riservati
```

Salvataggio dello stato corrente dei registri

# myExceptionHandler.s– 2

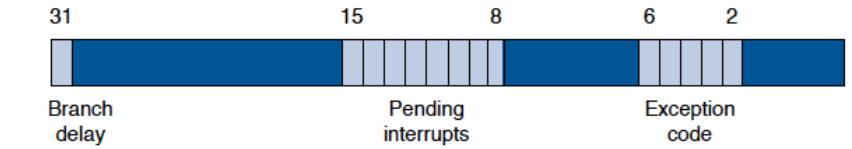



**FIGURE A.7.2 The Cause register.**

| Number | Name | Cause of excepti |
|---|---|---|
| 0 | Int | interrupt (hardware) |
| 4 | AdEL | address error except |
| 5 | AdES | address error except |
| 6 | IBE | bus error on instruct |
| 7 | DBE | bus error on data loa |
| 8 | Sys | syscall exception |
| 9 | Bp | breakpoint exceptior |
| 10 | RI | reserved instruction |
| 11 | CpU | coprocessor unimple |
| 12 | Ov | arithmetic overflow e |
| 13 | Tr | trap |
| 15 | FPE | floating point |

```
mfc0  $k0, $13          # copia nel registro $k0 della CPU
                        #il registro $13 (Cause register) del coprocessor 0
andi  $a0, $k0, 0x007c  # riconosce e distingue tra eccezioni e interrupt
                        # Exception mask: 0x007c = 0000 0000 0111 1100
bgtz  $a0, Excp_ret     # salta a Excp_ret se e' una eccezione, cioe' non è un
                        # interrupt da periferica (XXXX XXXX X000 00XX)
#se non e' un'eccezione, cioe' e' un interrupt
li    $v0, 4            # Stampa stringa Interrupt rilevato
la    $a0, __m3_        #__m3_: .asciiz " Interrupt\n"
syscall                 # print "  Interrupt "
IntXXXX: … # Gestione dell'Interrupt XXXX non trattata in questo esempio di exception handler
EndIntXXXX: #vedi prox slide
```

# myExceptionHandler.s – 3

```
EndIntXXXX:                # Epilogo gestione interruzione
mtc0   $0, $13             # Reset del Cause register ($13)

# Ripristino dei registri salvati
  lw       $v0, save0
  lw       $a0, save1
  move   $at, $k1

eret                # Ritorno al programma principale dall'exception handler
                    # Preleva l'indirizzo di ritorno e salta all'indirizzo di ritorno
```

# myExceptionHandler.s – 4

```
# Gestione delle eccezioni. Questo gestore semplicemente stampa le info sulla eccezione
Excp_ret:
  li      $v0, 4                  #stampa stringa rilevata eccezione
  la      $a0, __m1_              #__m1_: .asciiz "  Exception "
  syscall                          # print "  Exception "

  li      $v0, 1                  # print_integer per stampare il codice dell'eccezione rilevata
  srl     $a0, $k0, 2             # shift right logical e and con 0x1f per estrarre il Codice eccezione
  andi  $a0, $a0, 0x1f
  syscall                         # print Codice eccezione

  li      $v0, 4                  #stampa stringa specifica eccezione rilevata
  lw    $a0, __excp($k0)          # __excp($k0)=__e12_: .asciiz              "  [Arithmetic overflow] "
  syscall                         # print stringa con nome dell'eccezione

  srl     $k0, $k0, 2             # estrae il Codice eccezione perche' era in $a0 che e' stato sovracritto
 andi $k0, $k0, 0x1f
```

# myExceptionHandler.s – 5

```
#caso: eccezione Overflow aritmetico
Excp_Aritm:
bne $k0, 0xc, End_Excp_Aritm        # se non e' eccezione per overflow
                                    #aritmetico (codice 12) fa branch alla fine
#qui potrei avere altre istruzioni di gestione dell'eccezione. Qui facciamo solo la stampa
li   $v0, 4                         # syscall 4 (print_str)
la   $a0, __e12_                    # __e12_: .asciiz "[Arithmetic overflow] "
syscall
j     End_Excp_Hnd

End_Excp_Aritm:                     # Fine gestione overflow aritmetico
```

# myExceptionHandler.s – 6

```
Excp_AdEL:      #caso: eccezione Address Error in Load
 bne $k0, 0x4, End_Excp_Hnd          #se il codice l'eccezione NON e' 4=0x4=0100, vado oltre
                                     #(poiche' sono qui gestite solo 2 eccezioni, vado alla fine del gestore)
 li    $v0, 4                        # syscall 4 (print_str)
 la    $a0, __e4_                    # __e4_: .asciiz "  [Address error in inst/data fetch] at address : ”
 syscall
 li $v0, 1
 mfc0 $a0, $8                        #copio in $a0, l'indirizzo di memoria in cui un'istruzione ha generato
                                     #quest'eccezione (contenuto del registro $8 del coprocessore 0)
 syscall
 j End_Excp_Hnd
```

# myExceptionHandler.s – 7

```
End_Excp_Hnd:                # Epilogo del gestore delle Eccezioni
  mtc0    $0, $13            # Si resetta il Cause register

  lw      $v0, save0         # Si ripristinano i registri salvati
  lw      $a0, save1
  move  $at, $k1

# Preleva l'indirizzo di ritorno e salta all'indirizzo di ritorno
  mfc0   $k0, $14            # carica EPC in $k0
  addi    $k0, $k0, 4        # Incrementa di 4 per ritornare alla istruzione successiva
  mtc0   $k0, $14

  eret              # Return from exception handler (all'istruzione indicata da EPC)
```

- Per ulteriori esercizi, si vedano le risorse in e-learning relative al Laboratorio – Eccezioni