Eccezioni

Esercitazione 8

Architettura degli elaboratori

2

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)

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

3

Gestione delle eccezioni in MIPS

Per tutti i tipi di eccezione, avviene:

- Salvataggio dell'indirizzo dell'istruzione che ha causato l'eccezione (in **EPC**)
- Cambiamento del normale flusso di esecuzione di un programma: aggiornamento di PC con il valore 0x80000180 (indirizzo dello spazio kernel text in cui ha inizio la procedura di gestore delle eccezioni – **exception handler**)

Qualora l'eccezione sia tale da poter riprendere l'esecuzione al termine della sua gestione, ***sarà necessario ripristinare lo stato al momento della rilevazione dell'eccezione***. In MIPS non è svolta e *deve essere realizzata come parte*, ovviamente finale, *del gestore delle eccezioni*

4

Co-processore 0 per gestione eccezioni

- Istruzioni MIPS
  - per accedere ai *registri del co-processore 0*:
    - mfc0 – move from coprocessor 0;
    - mtc0 – move to coprocessor 0
  - eret: restituisce il controllo al programma interrotto
- Registri del co-processore 0:
  - Riservati al coprocessore 0: $k0, $k1
  - set dei registri disponibili in SPIM

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

5

Registro Cause e possibili cause di eccezione

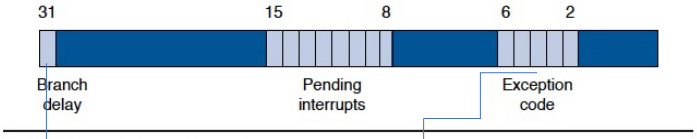


FIGURE A.7.2 The Cause register.

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

6

Exception Handler

- Porzione di codice assembly che contiene il codice che gestisce le eccezioni
- Si trova alla locazione fissa 0x80000180 in 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 TUTTI i registri macchina
- Ad esso sono riservati due registri $k0, $k1 che può utilizzare senza doverne preservare i contenuti
- Al termine del trattamento dell'eccezione, l'exception handler ritorna il controllo al programma interrotto (istruzione eret)
- MIPS (e quindi anche SPIM) rileva e tratta le eccezioni PRIMA del completamento dell'esecuzione della istruzione corrente:
  - se è una interruzione, dopo la sua gestione, l'esecuzione del programma deve riprendere **dall'istruzione** corrente (salvata nel registro EPC)
  - se si tratta di una **eccezione** ed è possibile proseguire (se non è avvenuto un errore irreparabile e non si tratta di una istruzione break), l'esecuzione deve riprendere dall'istruzione successiva (EPC+4)

7

simpleExcHandler.s

Il file **simpleExcHandler.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):

- sono salvati i registri usati al suo interno (a parte i registri riservati $k0 e $k1)
- è analizzato il contenuto del registro Cause ($13 del co-processore 0)
- è stampato un messaggio contenente il nome e codice dell'eccezione
- è ritornato il controllo al programma in cui è stata generata l'eccezione, dopo aver:
  - ripristinato i registri utilizzati (a parte i registri riservati $k0 e $k1)
  - aggiornato il contenuto di EPC:=EPC+4

8

Esempio – overflow aritmetico

Dopo aver verificato che nelle impostazioni di QTSPIM sia stato caricato il file **simpleExcHandler.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 e' 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
```

9

Esempio – overflow aritmetico (in QTSPIM)

- Quando avviene l'eccezione
  - appare un messaggio sulla Message Window "Exception occured …"; tale stampa è effettuata internamente al simulatore e NON dal simpleExcHandler.s che non ha ancora iniziato l'esecuzione
  - a PC è assegnato il valore 0x80000180 (indirizzo iniziale del gestore delle eccezioni)
  - a EPC è assegnato l'indirizzo della istruzione che ha provocato l'eccezione
  - Cause = 0x00000030
  - Status = 3000ff12
- Il messaggio scritto nella Message Window:
  Exception occurred at PC=0x00400030
  Arithmetic overflow

  [0040002c] 34090002 **ori $9, $0, 2** *; 6: li $t1, 2 # aggiungo abbastanza per superare il massimo positivo*
  [00400030] 01092020 **add $4, $8, $9** *; 7: add $a0, $t0, $t1 # cosa succede dopo aver eseguito questa istruzione?*
- Nella finestra di Console di QTSPIM è invece visualizzato il messaggio scritto dall'exception handler **simpleExcHandler.s**

10

simpleExcHandler.s

- MIPS (e quindi anche QTSPIM) rileva e tratta le eccezioni PRIMA del completamento dell'esecuzione della istruzione corrente:
  - se l'eccezione corrente era una interruzione, 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 e non si tratta di una istruzione break), l'esecuzione deve riprendere dall'istruzione successiva (EPC+4)

11

simpleExcHandler.s – 2

N.B. per QTSPIM dovremo usate 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          # kernel text segment (inizia a 0x80000180)
move  $k1, $at             # salvataggio $at (usato nelle pseudo-instruzioni
                           # dall'exception handler)
sw    $v0, save0           # salvataggio $v0 (usato da myExceptionHandler.s)
sw    $a0, save1           # salvataggio $a0 (usato da myExceptionHandler.s)
                           #non serve salvare $k0 e $k1
                           #perché per convenzione riservati

…
```

Salvataggio dello stato corrente dei registri

12

simpleExcHandler.s – 2

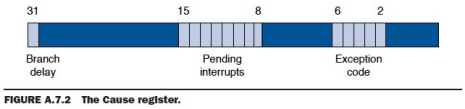


FIGURE A.7.2 The Cause register.

```
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)
…
```

13

simpleExcHandler.s – 3

```
#gestione interrupt
li    $v0, 4                    # Stampa Interrupt rilevato
la    $a0, __m3_                #__m3_: .asciiz " Interrupt\n"
syscall
# print "  Interrupt “
# Gestione dell'Interrupt XXXX non trattata in questo esempio
IntXXXX:
...
EndIntXXXX:
```

14

simpleExcHandler.s – 4

```
# Epilogo gestione interruzione
mtc0   $0, $13          # Si resetta il Cause register ($13)


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

                  # Preleva l'indirizzo di ritorno
                  # e salta all'indirizzo di ritorno
eret              # Return from exception handler
```

15

simpleExcHandler.s – 5

```
# Gestione delle eccezioni
Excp_ret:                   # Stampa le info sulla eccezione
 li     $v0, 4
 la     $a0, __m1_
 syscall                    # print "  Exception "

 li     $v0, 1
 srl    $a0, $k0, 2         # shift right logical per estrarre il Codice eccezione
 andi  $a0, $a0, 0x1f
 syscall                    # print Codice eccezione

 li     $v0, 4
 lw    $a0, __excp($k0)
 syscall                    # print stringa def eccezione

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

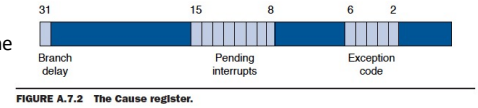


FIGURE A.7.2 The Cause register.

16

simpleExcHandler.s – 6

```
Excp_Aritm:
bne $k0, 0xc, End_Excp_Aritm    # non e' eccezione per overflow #aritmetico (num 12)
li    $v0, 4                    # syscall 4 (print_str)
la    $a0, __e12_               # eccezione overflow aritmetico
syscall
j      End_Excp_Hnd
End_Excp_Aritm:                 # Fine gestione overflow aritmetico
```

17

… per ogni eccezione gestita dall'handler … (ad esempio per Address Exception Load)

```
Excp_AdEL:
  bne $a0, 0x4, End_Excp_AdEL #se il codice eccezione NON è 0x4=0100, vado oltre
  mfc0 $a0, $8                #copio in $a0, l'indirizzo di memoria dell'istruzione
                                che ha generato quest'eccezione (NB. registro $8 del coprocessore 0
                                contiene questo valore)
andi   $a0, $a0, 0x3          # E' allineato alla word? Uno degli ultimi 2 bit è a 1?
beq    $a0, $0, End_Excp_Hnd  # nel caso di word allineata, vado alla fine del gestore
 li      $v0, 10              # nel caso di «Bad address», EXIT (codice 10)
 syscall
 j       End_Excp_Hnd
End_Excp_AdEL:          # Fine  gestione "Bad address"
```

18

simpleExcHandler.s – 7

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

 lw     $v0, save0          # Si ripristina i registri salvati
 lw     $a0, save1
.set noat
 move  $at, $k1
.set at
                            # Si 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)
```

N.B. eret – ritorna alla istruzione specificata in EPC
- Eccezioni: si deve incrementare EPC di 4 e salvare il nuovo valore in EPC nel coprocessore
- Interruzioni: non si deve eseguire nessuna modifica a EPC

19

Esercizio - Modifica all'ExceptionHandler per la gestione di ulteriori eccezioni

- E.g. Break
- E.g. Address Exception Load (store)
- …

20

FSM modificata per includere gestione di 2 eccezioni

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

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

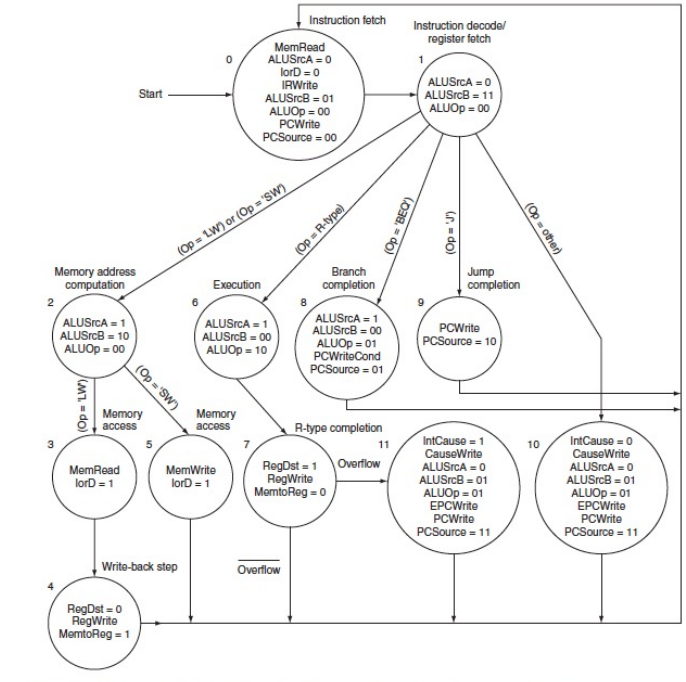
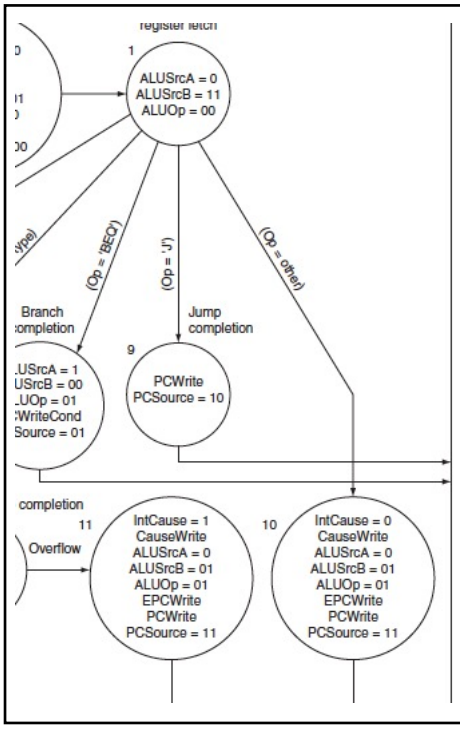


FIGURE 5.40 This shows the finite state machine with the additions to handle exception detection. States 10 and 11 are th

21



N.B. Schema fig. 5.39 (pag 344) relativo a SOLA eccezione Istruzione non valida
La gestione dell'eccezione aritmetica richiede il collegamneto dell'uscita overflow della ALU con la control unit e, nella versione riportata sul libro, la sua memorizzazione in un registro temporaneo

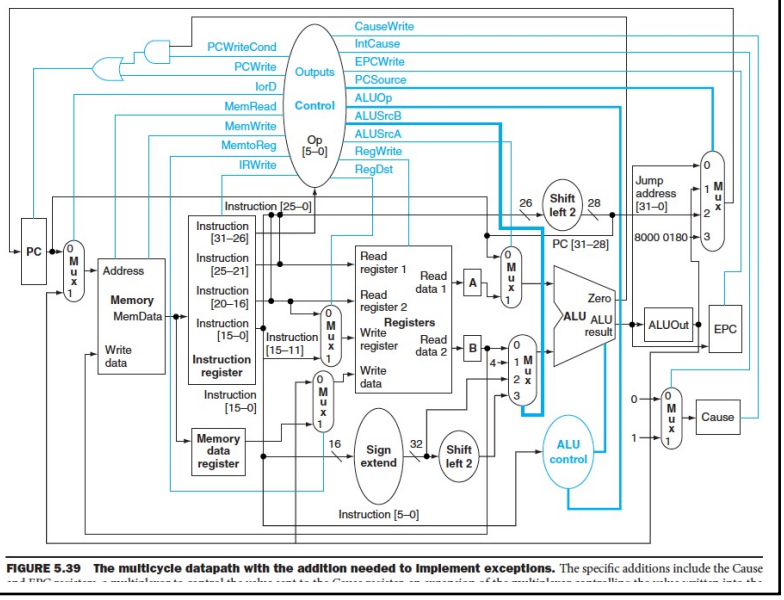


FIGURE 5.39 The multicycle datapath with the addition needed to implement exceptions. The specific additions include the Cause

Esercizio – modifica datapath per overflow

- Modicare lo schema del **datapath multiciclo esteso per la gestione delle eccezioni** in modo da gestire l'eccezione generata da overflow aritmetico
- Soluzione (diversa da quella descritta dal libro di testo, disponibile nel video Laboratorio Eccezioni - soluzione overflow) prevede che
  - next_state(stato6) → 11
  - non next_state(stato7) → 11

Soluzione – modifica datapath per gestione overflow

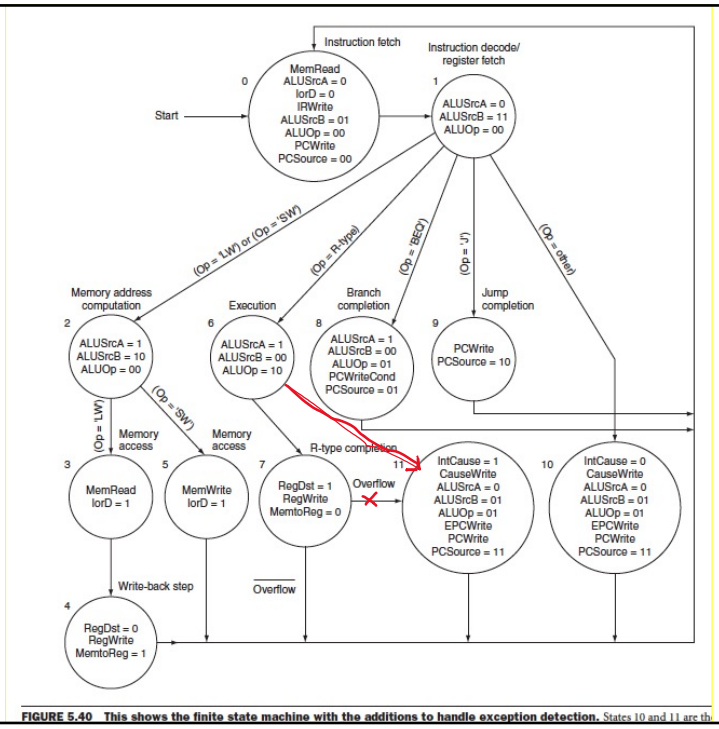


FIGURE 5.40 This shows the finite state machine with the additions to handle exception detection. States 10 and 11 are th

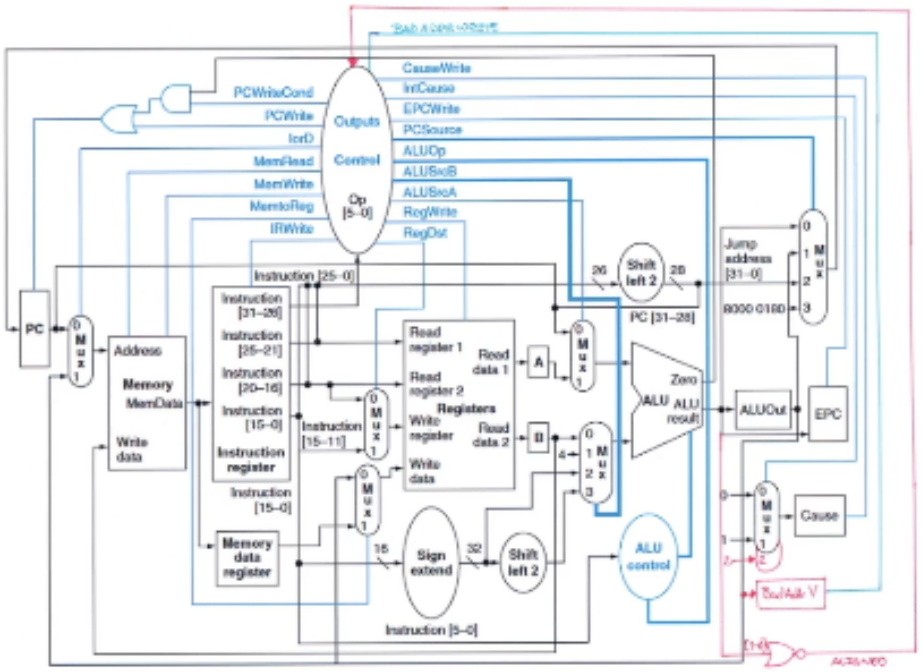
Esercizio – modifica datapath per lw non allineato

- Modicare lo schema del **datapath multiciclo esteso per la gestione delle eccezioni** e la relativa FSM in modo da gestire un'eccezione in caso si tenti di effettuare una **load word da un indirizzo non allineato**

Suggerimento: un possibile modo per verificare che un indirizzo sia allineato è controllare che gli ultimi due bit siano entrambi 0

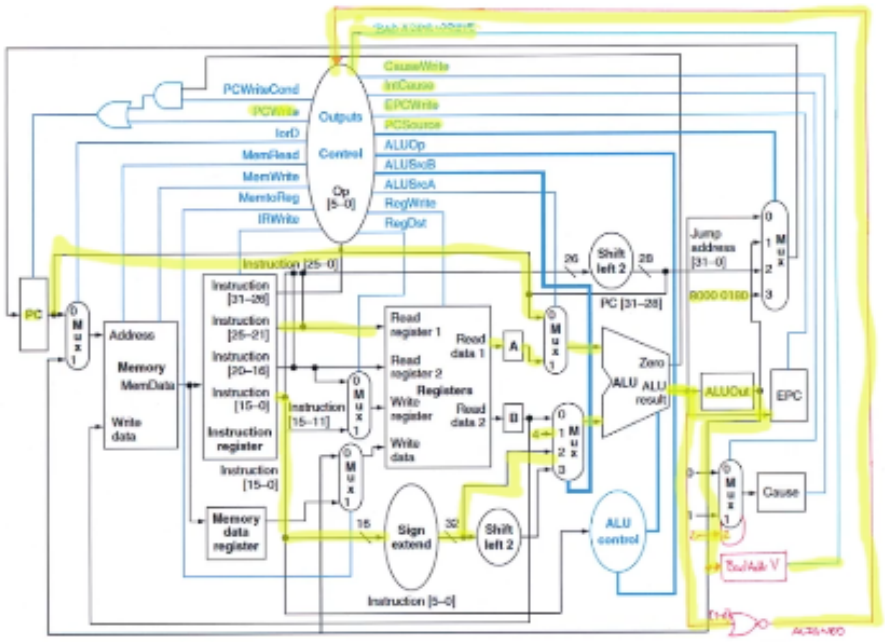
La soluzione è disponibile nel video Laboratorio Eccezioni - soluzione lw non allineata

Soluzione 1/3 – modifica datapath per lw non allineato



26

Soluzione 2/3 – modifica datapath per lw non allineato



27

Soluzione 3/3 – modifica datapath per lw non allineato

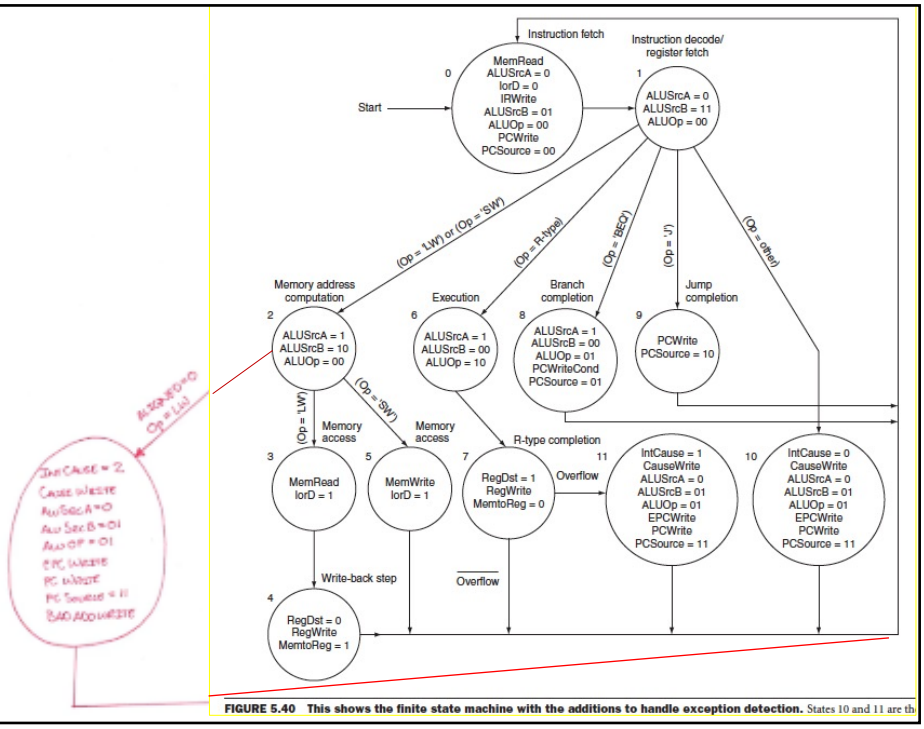


FIGURE 5.40 This shows the finite state machine with the additions to handle exception detection. States 10 and 11 are th

28

- Per ulteriori esercizi, si veda Laboratorio 7 – gestione Eccezioni

29