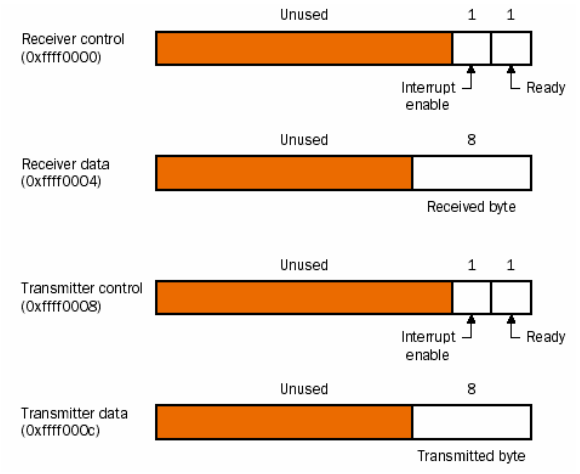
Gestione dell'I/O in SPIM

- SPIM simula un dispositivo di I/O
  - un terminale memory-mapped.
- Il dispositivo è formato da due unità indipendenti: un *receiver* ed un *transmitter*.
  - Il receiver legge caratteri dalla tastiera.
  - Il transmitter scrive caratteri sulla finestra di console.
  - Le due unità sono completamente indipendenti.
- Ogni dispositivo è mappato in memoria su due indirizzi
  - Control Register
  - Data Register

Dispositivi di I/O di SPIM



Dispositivi di I/O di SPIM

- I registri di controllo sono effettivamente di stato/controllo
- Lo stato si riduce a un bit di ready
- Il controllo permette di abilitare una gestione a interruzioni:
  - Il ricevitore genera un interrupt a livello 0 quando il bit ready è pari a 1
  - Il trasmettitore genera un interrupt a livello 1 quando il bit ready è pari a 1
- Per gestire l'interrupt I/O è necessario abilitare i corrispondenti livelli nel registro stato (vd. oltre)

Esempio: programmed I/O

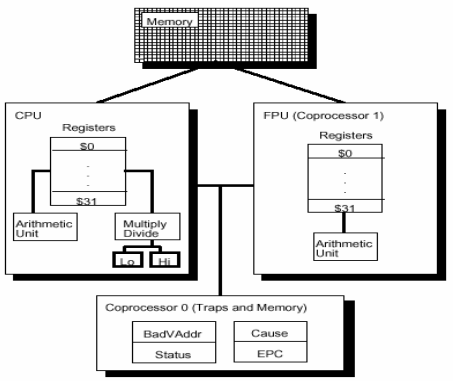
```
prtstr:
     li $t3,0xFFFF0008 # Carica in t3 l'ind. del contr. reg. del trasm.
     li $t4,0xFFFF000C # Carica in t4 l'ind. del data reg. del trasm.
     move      $t0,$a0        # Carica in t0 l'indirizzo della stringa
lchar:  lbu    $t1,0($t0)     # Carica in t1 il carattere corrente
     beqz      $t1,fine       # Carattere di fine stringa -> termina
# Operazione di output
busyw:  lw     $t2,0($t3)     # Carica in t2 il cont. del contr. reg.
     andi      $t5,$t2,1      # Controlla il bit ready
     beqz      $t5,busyw      # Torna in ciclo se ==0
     sb $t1,0($t4)     # Scrive il carattere nel data reg. del trasm.
     addi      $t0,$t0,1      # Incrementa il puntatore alla stringa
     b  lchar          # Continua il ciclo di scrittura dei caratteri
fine:   jr     $ra
```

Modi di esecuzione user / kernel

- Due modi di esecuzione:
  - User
  - kernel
- Per ognuno dei due modi di esecuzione sono previste apposite aree dati e codice
- In modo kernel vengono eseguite le istruzioni del sistema operativo ed, in particolare, vengono gestite le eccezioni.

Gestione delle Eccezioni in SPIM

- La gestione delle eccezioni e delle interruzioni è effettuata dal coprocessore 0
- 4 registri mantengono tutte le informazioni necessarie alla gestione
- E' possibile accedere ai registri tramite le istruzioni:
  lwc0, swc0, mfc0, mtc0



Interruzioni ed eccezioni

- Esistono diverse categorie di “interruzioni”:
  - Eccezioni
  - Interruzioni
  - Chiamate di sistema
  - Segnale RESET

Eccezioni

- Corrispondono ad eventi “anormali”, tipicamente errori che impediscono la corretta esecuzione dell’istruzione in corso.
- Non sono mascherabili.
- Tipi di eccezioni:
  - ADDRL Address error in lettura (load o istr. Fetch)
  - ADDRS Address error in scrittura
  - IBUS Bus error in istr. fetch
  - DBUS Bus error in data
  - RI Reserved instruction
  - CPU Coprocessor inaccessibile
  - OVF Overflow

Gestione delle eccezioni

- All'atto di un'eccezione il processore:
  - Salva l'indirizzo dell'istruzione in EPC
  - Salva la configurazione presente del registro di stato
  - Passa in modo kernel e disabilita le interruzioni
  - Salta all'indirizzo 0x80000080

Interruzioni

- Le richieste di interruzione sono eventi asincroni provenienti in genere dalle periferiche.
- Il processore ha 6 linee di interruzione esterne che possono essere mascherate globalmente o singolarmente. L'attivazione di una di queste linee è una richiesta di interruzione.
- Le interruzioni sono segnalate nel registro Causa e sono gestite al termine dell'esecuzione dell'istruzione in corso, se non sono mascherate
- Il processore passa in modo kernel e salta al gestore di interruzioni dopo aver salvato l'indirizzo di ritorno.

Gestione delle interruzioni

- All'atto di un'interruzione il processore:
  - Salva l'indirizzo di ritorno (PC+4) in EPC
  - Salva la configurazione presente del registro di stato
  - Passa in modo kernel e disabilita le interruzioni
  - Salta all'indirizzo 0x80000080

Chiamate di sistema

- Due istruzioni per attivare chiamate di sistema, eseguibili in modo utente:
  - SYSCALL
  - BREAK
- L'istruzione SYSCALL permette ad un processo di chiedere un servizio al sistema, p. es. un'operazione di I/O
- Il codice identificativo del tipo di servizio richiesto e gli eventuali parametri devono essere preparati prima all'interno di registri generali
- L'istruzione BREAK è utilizzata più specificamente per inserire un punto di arresto all'interno di un programma
- In entrambi i casi, il processore passa in modo kernel e salta al gestore di interruzioni dopo aver salvato l'indirizzo di ritorno.

Gestione delle chiamate di sistema

- All'atto dell'esecuzione di una chiamata di sistema il processore:
  - Salva l'indirizzo di ritorno (PC+4) in EPC
  - Salva la configurazione presente del registro di stato
  - Passa in modo kernel e disabilita le interruzioni
  - Salta all'indirizzo 0x80000080

Reset

- Il processore possiede una linea RESET la cui attivazione produce il salto incondizionato alla routine di inizializzazione
- L'attivazione della linea RESET è simile ad una settima linea di interruzione esterna con le seguenti importanti differenze:
  - non è mascherabile
  - non è necessario salvare un indirizzo di ritorno
  - il codice per la gestione del reset si trova all'indirizzo 0xBFC00000
- All'atto dell'attivazione del RESET il processore
  - Passa in modo kernel e disabilita le interruzioni
  - Salta all'indirizzo 0xBFC00000

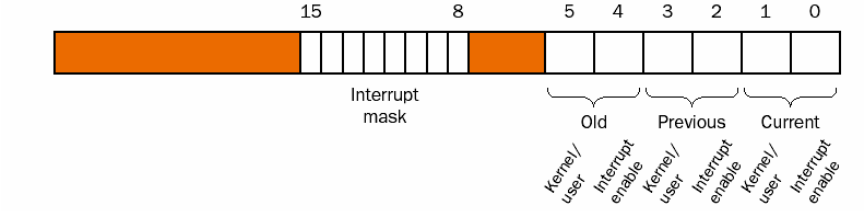
Registri del coprocessore 0

Register name	Register number	Usage
BadVAddr	8	register containing the memory address at which memory reference occurred
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	register containing address of instruction that caused exception

EPC contiene l'indirizzo di ritorno dal servizio all'interruzione

BadVAddr contiene l'indirizzo di memoria cui è stato fatto riferimento

Registro di Stato



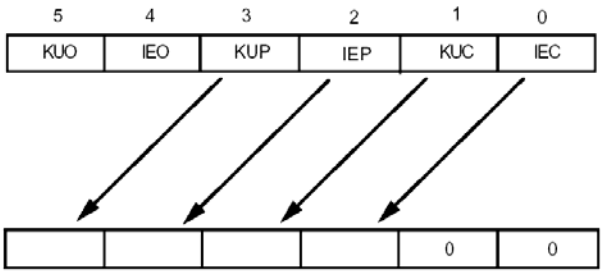
•L'interrupt mask contiene un bit per ciascuno degli 8 livelli possibili di interruzione: 6 hardware (10-15) e 2 software (8-9). Il livello è abilitato se il bit corrispondente è messo a 1.

•Il bit 0 (IEC) gestisce l'abilitazione generale delle interruzioni (0=disabilitate).

•Il bit 1 (KUC) segnala se il processore è in modo kernel (0) o utente (1)

•I bit 2-5 realizzano uno stack di profondità due per i bit IE e KU.

Registro di Stato

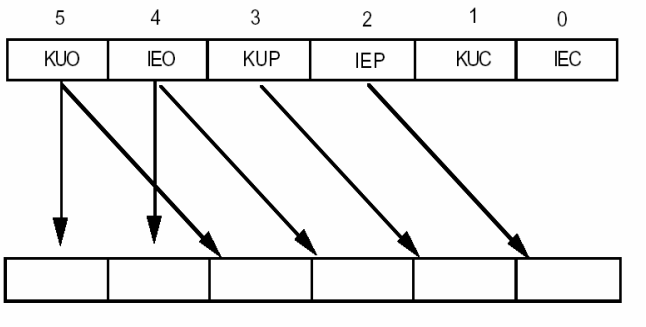


All'atto dell'interruzione, i 6 bit sono scalati a sinistra di due posizioni. I 2 bit meno significativi sono posti a 00 → modo kernel, interruzioni disabilitate.

Ritorno da un'interruzione

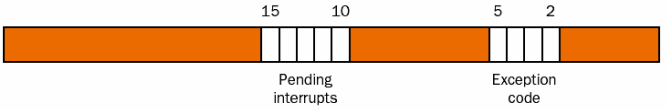
- Prima di riprendere l'esecuzione di un programma che ha effettuato una chiamata di sistema o è stato interrotto è necessario ripristinare il registro di Stato
- Il ripristino viene effettuato grazie ad un'istruzione RFE che rimette il registro di Stato nelle condizioni precedenti all'interruzione.

Registro di Stato



Al termine del servizio all'interruzione, viene ripristinato il registro di stato tramite un'istruzione RFE.

Registro Causa



I bit 10-15 corrispondono ai 6 livelli di interrupt hardware: se un bit è a 1, c'è un'interruzione non ancora servita a quel livello.

I bit 2-5 descrivono la causa dell'eccezione secondo la codifica:

Number	Name	Description
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
11	CPU	Coprocessore non disponibile

Istruzioni per la gestione di eccezioni

- **rfe** (return from exception)
  - Ripristina il registro Stato, ma non ritorna al programma interrotto
- **break** *code* (es. break 3)
  - Genera una eccezione di breakpoint parametrizzata da *code*. *code*=1 è riservato al debugger. Il code si estrae dall'istruzione indirizzata da EPC
- **syscall**
  - Genera una system call. Il codice del servizio è contenuto in $v0 ($2)
- **nop** (no operation)
  - Istruzione senza alcun effetto

Fasi del servizio delle eccezioni

- In seguito al verificarsi di un'eccezione
  - L'istruzione in esecuzione, e tutte quelle successive in esecuzione nella pipeline, sono abortite
  - Le informazioni utili alla gestione e al ripristino dell'esecuzione del processo interrotto sono salvate nei registri del coprocessore 0
  - La modalità di esecuzione del processore commuta a kernel
  - Il controllo è trasferito a un ***exception handler*** posto all'indirizzo **0x80000080** (kernel), che esamina la causa dell'eccezione e la gestisce

Il Gestore delle eccezioni

```
.ktext 0x80000080
sw $a0, save0   # Handler is not re-entrant and can't use
sw $a1, save1   # stack to save $a0, $a1
                # Don't need to save $k0/$k1
```

```
mfc0    $k0, $13        # Move Cause into $k0
mfc0    $k1, $14        # Move EPC into $k1

sgt     $v0, $k0, 0x44 # Ignore interrupts
bgtz    $v0, done

mov     $a0, $k0        # Move Cause into $a0
mov     $a1, $k1        # Move EPC into $a1
jal     print_excp      # Print exception error message
```


Il Gestore delle eccezioni

```
done:
        lw    $a0, save0
        lw    $a1, save1
        addiu $k1, $k1, 4  # Do not reexecute
                           # faulting instruction
        rfe                # Restore interrupt state
        jr    $k1

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