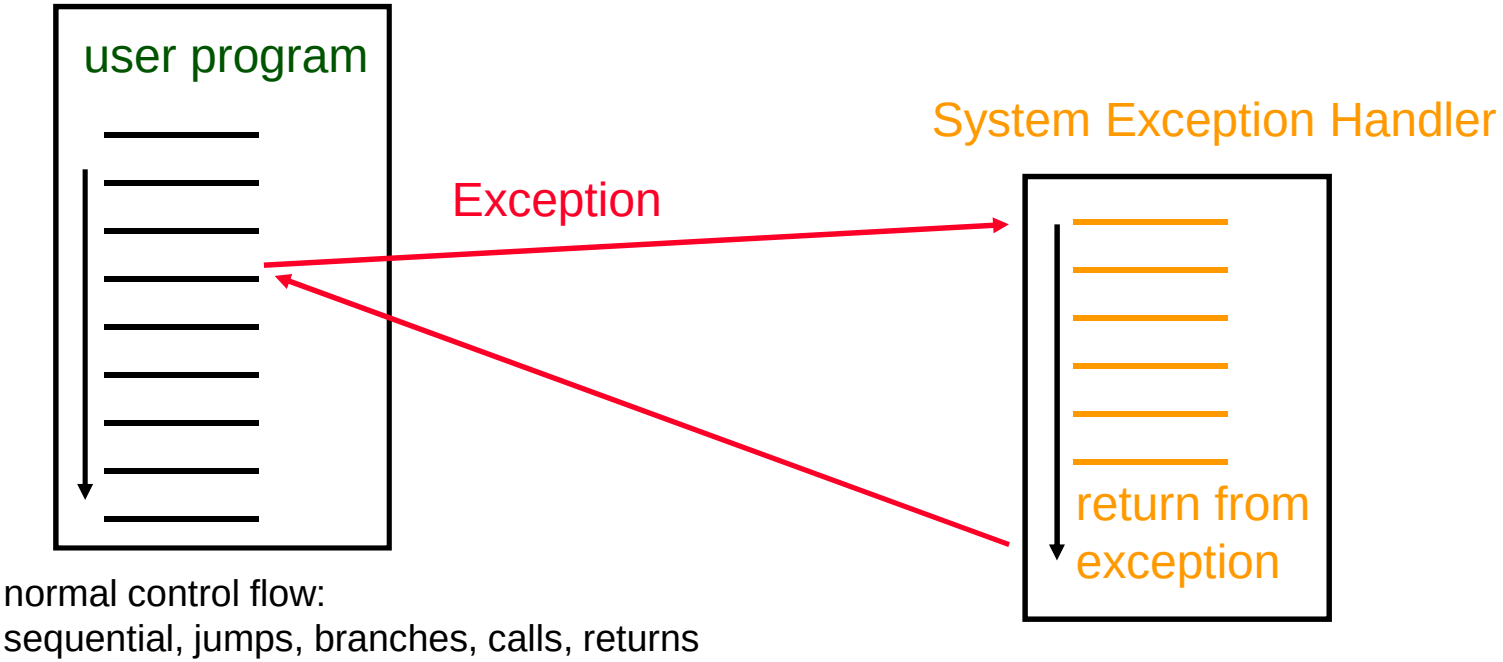
Lezione 9
Eccezioni e interrupt

http://www.dii.unisi.it/~giorgi/didattica/arcal1

Eccezioni



- **Eccezione = trasferimento non-previsto del flusso di controllo**
  - **Il sistema gestisce l'eccezione eseguendo una routine "agganciata" ad una eccezione di quel tipo**
    - **E' necessario memorizzare l'indirizzo dell'istruzione che ha generato l'eccezione (es. divisione per zero)**
  - **Successivamente il controllo deve tornare al programma utente**
  - **E' necessario salvare e ripristinare lo stato del programma utente**

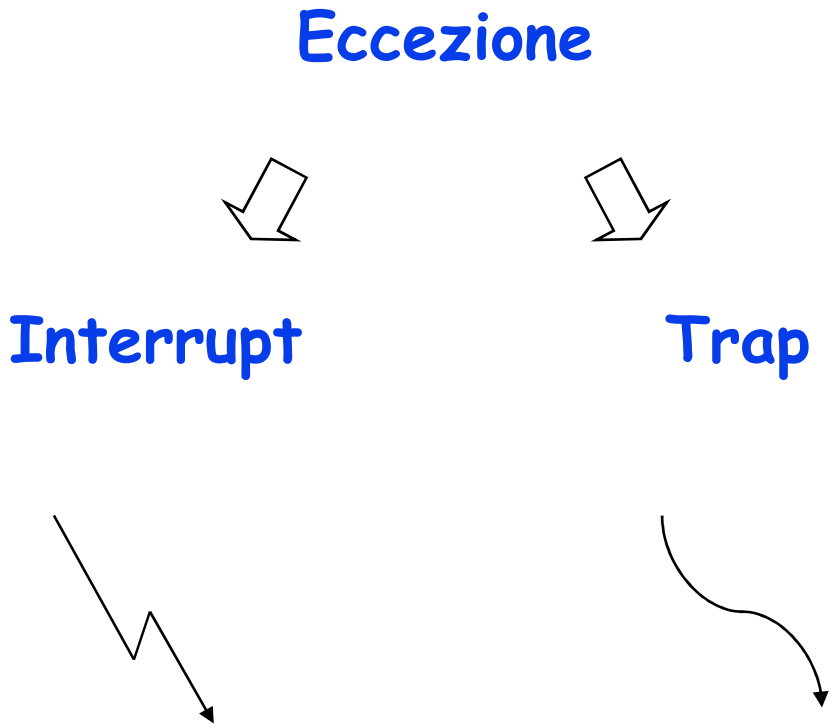
Che ne e' dell'istruzione che era in esecuzione?

- L'architettura MIPS definisce l'istruzione che causa un'eccezione come una "istruzione senza effetto"

- Le eccezionidevono evitare assolutamente di modificare lo stato della macchina
  - Es. i contenuti dei registri $s… NON devono essere alterati

- La gestione delle eccezioni e' un aspetto complesso del microprocessore spesso trascurato
  - Potenzialmente puo' anche limitare le prestazioni

Classificazione Generale delle Eccezioni



Classificazione delle Eccezioni (2)

- Interrupt (interruzione)
  - Eccezione causata da eventi esterni
  - Asincrona rispetto all'esecuzione del programma
  - La sua gestione "accade" fra istruzioni di un altro programma
  - Il programma in esecuzione deve essere sospeso e poi riattivato
- Trap
  - Eccezione causata da eventi interni
    - Condizioni eccezionali (overflow, divisione per zero)
    - Errori del sistema (parita')
    - Gestione della memoria virtuale (page fault)
  - Sincrona rispetto all'esecuzione del programma
  - Il gestore dell'eccezione deve rimediare alla situazione
  - L'istruzione "colpevole" puo' essere riprovata o simulata e il programma puo' o continuare o essere bloccato (abort)

Classificazione usata nel MIPS

- **Ogni cambiamento inaspettato del flusso di controllo viene denominato "eccezione"**
  - **Non distinguo fra sorgenti esterne e interne del microprocessore**

Tipo di evento	*Da dove?*	*Terminologia MIPS*
Richiesta di I/O da dispositivo	Esterno	Interrupt
Chiamata a Sistema Operativo dal programma utente (System Call)	Interno	Eccezione
Overflow aritmetico	Interno	Eccezione
Uso di istruzione non definita	Interno	Eccezione
Malfunzionamento hardware	Interno	Eccezione
Malfunzionamento hardware	Esterno	Interrupt

Routine di Gestione dell'Eccezione (Exception Handler)

- Esistono vari metodi
  - Vettore di Interruzione (approccio tradizionale)
  - Tabella delle Routine di Gestione (approccio RISC)
  - Salto ad indirizzo fisso (approccio MIPS)

Sia 'causa' il numero di identificazione dell'eccezione

Vettore di Interruzione

- Memorizzo in una tabella gli indirizzi delle Routine di Gestione per ogni possibile causa
  - 370, 68000, Vax, 80x86, . . .
- Sia 'base' l'indirizzo base di tale tabella
- Lo stato viene salvato facendo ricorso allo stack
- Il valore del PC sara': PC ← MEM[base+causa*4]

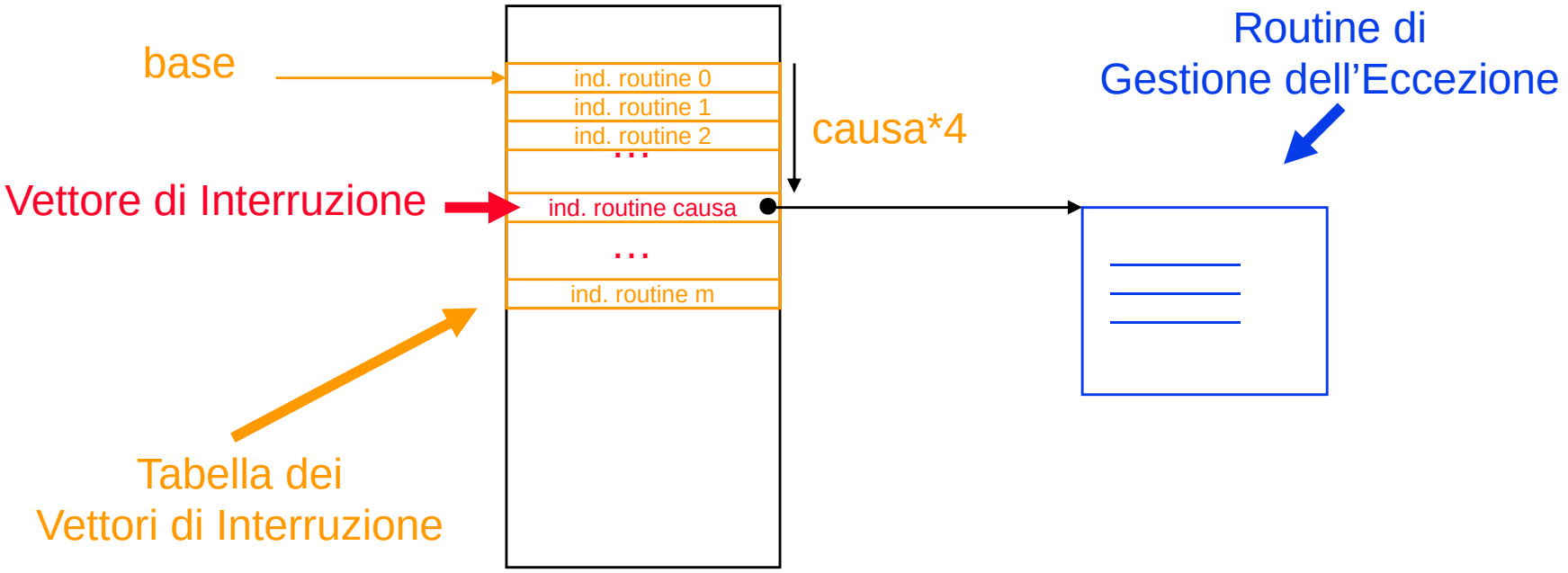
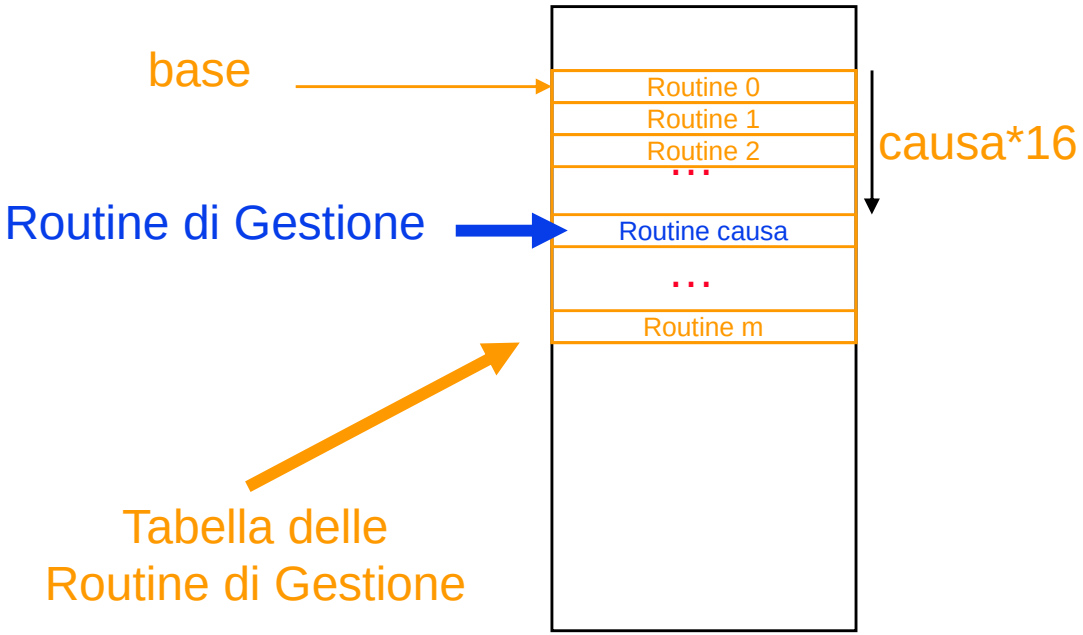


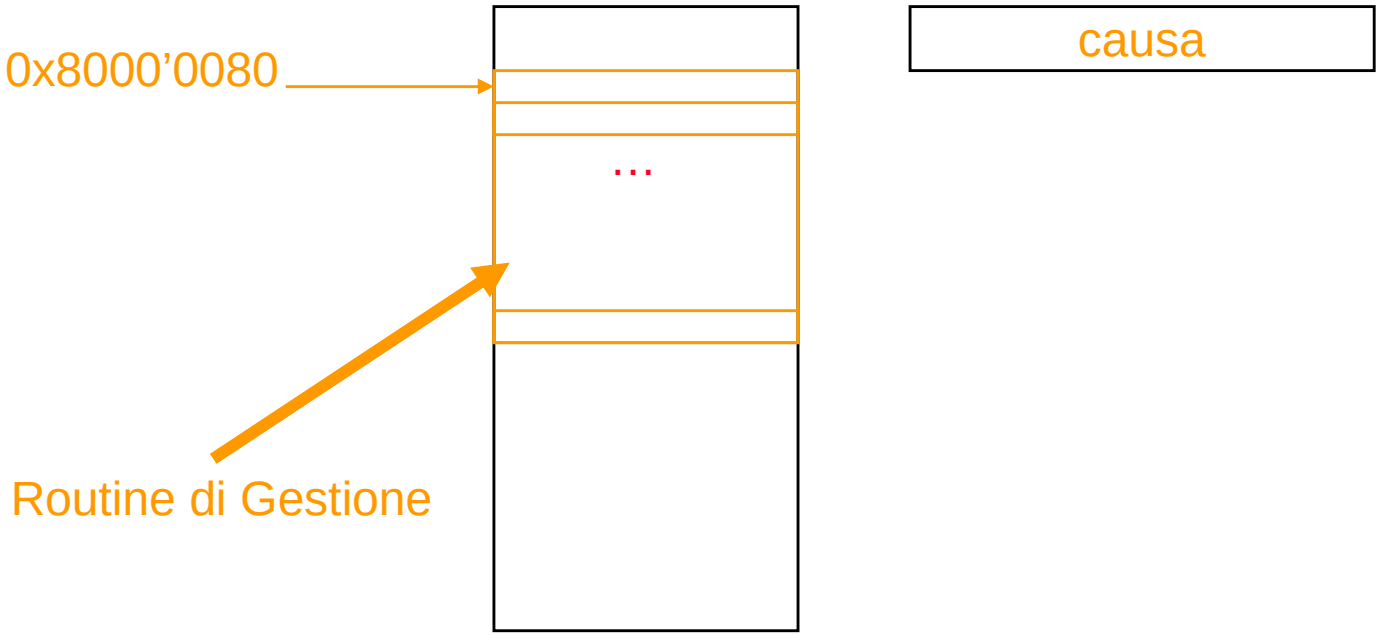
Tabella delle Routine di Gestione

- Memorizzo in una tabella direttamente le Routine di Gestione per ogni possibile causa
  - Sparc, PA, M88K, . . .
- Sia 'base' l'indirizzo base di tale tabella
- Il valore del PC sara': PC ← base+causa*16
- Salto senza far uso dello stack; nel codice iniziale ho piu' flessibilita'



Salto ad indirizzo fisso

- Salto sempre ad un indirizzo fisso che nel MIPS vale 0x8000'0080
- Il valore del PC sara': PC ← 0x8000'0080
- Salto senza far uso dello stack; nel codice della routine di gestione analizzo i vari casi
- In pratica ho una piccola tabella per gestire gli eventi:
  - RESET, Overflow, Istruzione Sconosciuta, TLB, …



Salvataggio dello stato durante un'eccezione

- Salvataggio sullo stack (Push)
  - Vax, 68k, 80x86
- Registri ausiliari (Shadow Registers)
  - M88k, ARM
  - Lo stato e' salvato in registri ausiliari sia visibili che non visibili direttamento all'utente (es. registri interni della pipeline)
- Salvataggio in registri speciali
  - MIPS
  - 4 Registri: EPC, BadVaddr, Status, Cause

Supporto alle gestione delle eccezioni nel MIPS ISA

- Vari registri a 32 bit:

- EPC
  contiene l'indirizzo dell'istruzione "colpevole"
- Status
  contiene i bit di mascheramento e i bit di abilitazione
- Cause
  i bit 5,4,3,2 di questo registro codificano le possibili sorgenti di eccezione, ad es.:
  - undefined instruction=0
  - arithmetic overflow=1
- BadVAddr (Bad Virtual Address)
  contiene l'indirizzo di memoria al guale si e' verificato un riferimento di memoria "sbagliato"

Dove si trovano tali registri

- EPC registro 14 coprocessore 0
- Status registro 12 coprocessore 0
- Cause registro 13 coprocessore 0
- BadVAddr registro 8 coprocessore 0

'coprocessore 0' non e' altro che una parte della CPU

- Al verificarsi di un'eccezione, la parte di controllo del microprocessore modifica: EPC, Status, Cause, BadVAddr, PC
  - Nota 1:
    - In fase di fetch il PC (oldPC) e' stato aggiornato PC+4 (newPC). Per poter identificare l'istruzione "colpevole" e' necessario far riferimento a oldPC anziche' a newPC. Quindi in EPC viene scritto oldPC.
  - Nota 2:
    - PC (newPC) viene (sovra-)scritto direttamente il nuovo valore (PC←0x8000'0080)

Status Register (1)

- Mask (8 bits) = 1 bit per ognuna di 6 sorgenti di eccezione hardware e 2 sorgenti di eccezione software
  - 1 => abilita quel tipo di eccezione
  - 0 => disabilita quel tipo di eccezione

- k = kernel/user bit (v. slide successiva)
  - 0 => ero nel kernel quando si e' verificata l'eccezione
  - 1 => ero nella modalita' utente
- e = interrupt enable bit
  - 0 => interrupt sono (erano) disabilitati
  - 1 => interrupt sono (erano) abilitati

- Al verificarsi di un'eccezione
  - i 4 bit LSB sono fatti scorrere verso sinistra di 2 posizioni
  - i nuovi 2 bit LSB sono azzerati; questo significa
    → esecuzione in modalita' kernel e con gli interrupt disabilitati

Status Register (2): Modalita' User/Kernel

- Il calcolatore puo' gestire se stesso utilizzando due modalita' di esecuzione denominate classicamente
  - Modalita' User
  - Modalita' Kernel
- Il Sistema Operativo puo' essere visto come un programma speciale in esecuzione sul calcolatore in una modalita' privilegiata (modalita' kernel) in cui:
  - Tutte le risorse del calcolatore possono essere utilizzate direttamente
  - Al software utente vengono presentate risorse "virtuali" che sono piu' potenti delle risorse fisiche, es.:
    - si forniscono i "file" anziche' i "settori del disco"
    - si fornisce una memoria molto piu' grande (memoria virtuale)
  - Viene garantita la protezione fra i vari programmi utente o di piu' utenti
- Le eccezioni consentono al sistema di rispondere ad eventi che si verificano mentre un qualsiasi programma utente e' in esecuzione
  - Il Sistema Operativo entra in azione dal momento in cui inizia ad essere eseguita la routine di gestione delle eccezioni

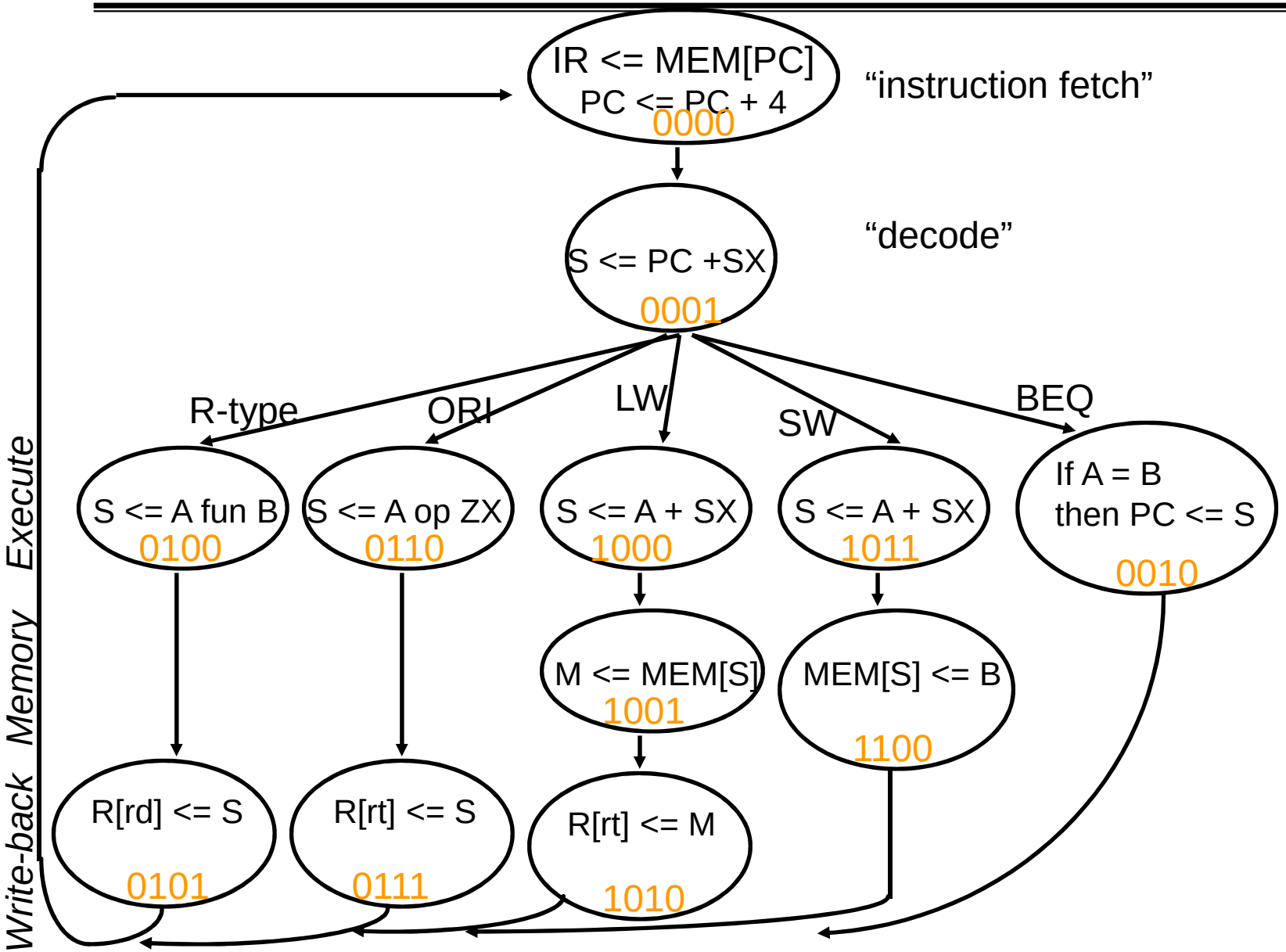
Cause Register

- Pending (8 bits) assegna 1 bit ad ognuna delle 6 sorgenti hardware e delle 2 sorgenti software: il bit vale 1 se l'eccezione si e' gia' verificata ma non e' stata ancora servita
  - Serve per gestire i casi in cui si puo' verificare piu' di un'eccezione contemporanamente (eccezioni annidate o "nested exceptions")
  - Utile per controllare se ci sono eccezioni nonostante si sia disabilitata la partenza della routine di eccezione per quella sorgente specifica (nello status Register)
- Code (4 bits) codifica la regione dell'eccezione
  - 00 (INT) → external interrupt
  - 04 (ADDRL) → address error exception (load or instr fetch)
  - 05 (ADDRS) → address error exception (store)
  - 06 (IBUS) → bus error on instruction fetch
  - 07 (DBUS) → bus error on data fetch
  - 08 (Syscall) → system call exception
  - 09 (BKPT) → breakpoint exception
  - 10 (RI) → reserved instruction exception
  - 12 (OVF) → arithmetic overflow exception

Interruzioni Precise/Imprecise

- Precisa → lo stato della macchina viene conservato, come se il programma avesse eseguito fino all'istruzione "colpevole"
  - Posizione adottata nei calcolatori IBM
  - Vantaggio: il codice di sistema potra' essere trasportato senza problemi su una implementazione diversa della stessa architettura
  - Svantaggio: difficile da implementare nel caso di pipeline, out-of-order execution, …
- Imprecisa → e' il software di sistema che deve occuparsi di scoprire in che stato il sistema si trova e quindi prendere le opportune decisioni
  - Posizione adottata nei primi processori MIPS (i piu' recenti usano le interruzioni PRECISE)
  - Svantaggio: le routine di gestione delle interruzioni debbono essere probabilmente riscritte se si cambia microprocessore MIPS
  - Vantaggio: snellisce l'hardware del microprocessore nei casi piu' frequenti
- Il motivo principale per cui in alcuni processori si evita l'uso degli interrupt precisi e' legato essenzialmente a obiettivi di prestazioni elevate
  - Chiaramente questo puo' mettere in crisi: sviluppatori del software di sistema, utenti, il mercato…

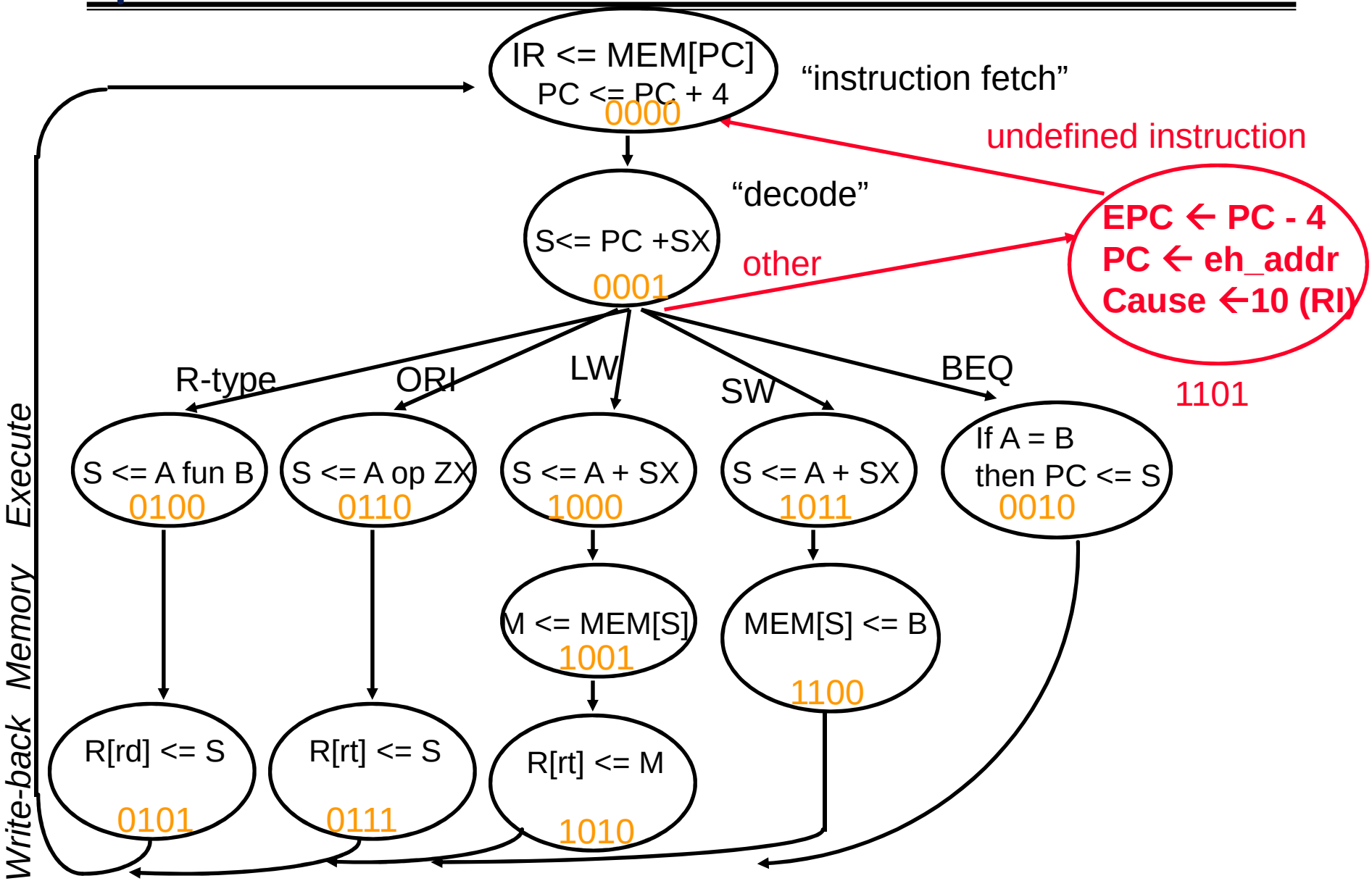
Macchina a Stati Finiti che descrive il nostro MIPS



Modifica del MIPS per poter gestire le eccezioni

- Supponiamo di partire dal processore "base" (che non gestisce eccezioni) e di voler gestire eccezioni: es. nel caso di istruzione indefinita.
- L'eccezione "Undefined Instruction" viene riconosciuta nel momento in cui, trovandomi nello stato 0001, non ho uno stato successivo in cui andare quando ho un op-code sconosciuto
  - Questo si puo' implementare definendo un nuovo stato (stato #13 o 1101) per tutti gli op-code diversi da lw, sw, 0 (R-type), jmp, beq, e ori
  - Nella figura e' mostrato simbolicamente con "other" per indicare che l'op-code non e' nessuno di quelli noti uscenti dallo stato #1

Implementazione del meccanismo di eccezione per una istruzione non definita

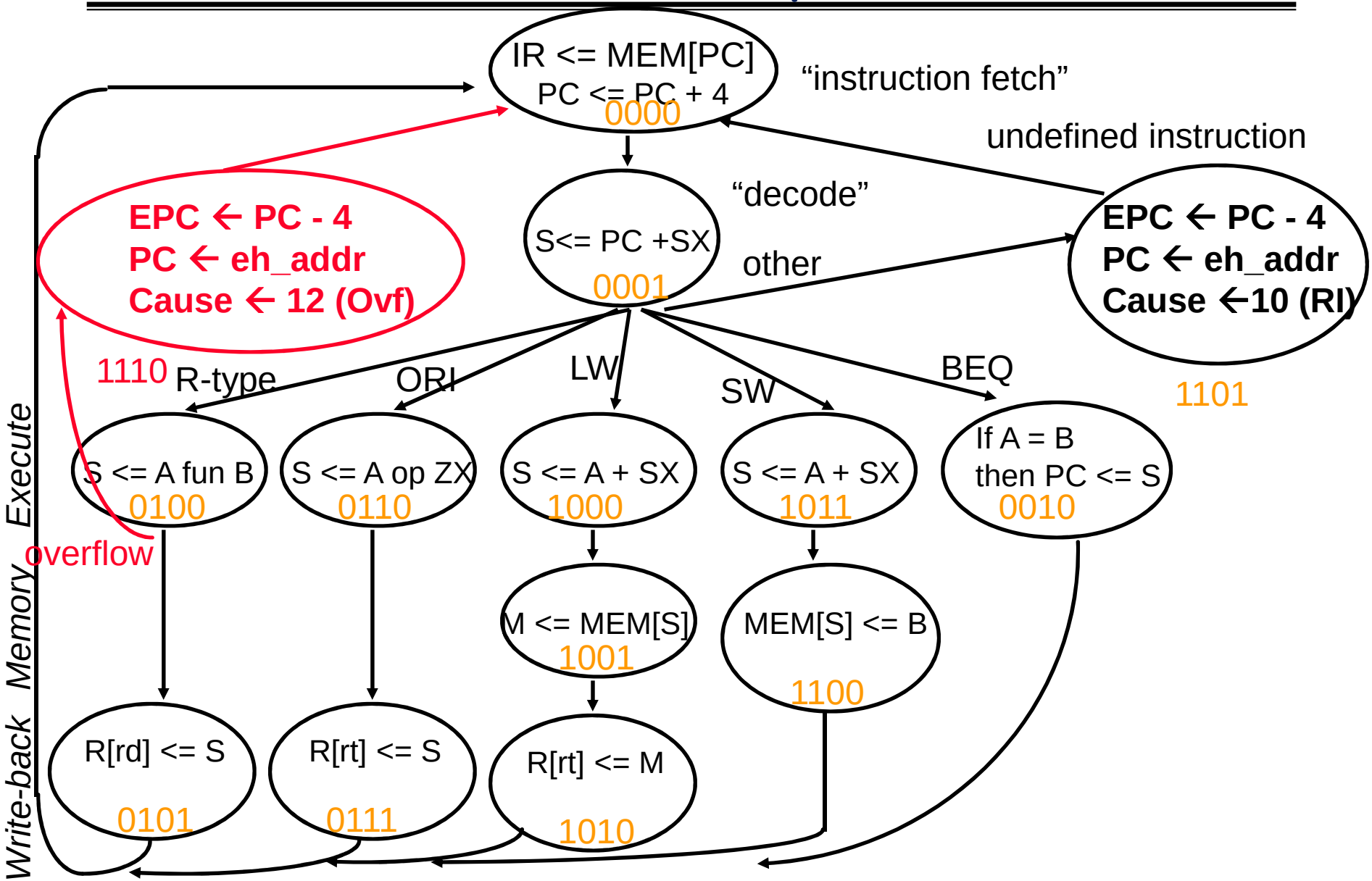


Implementazione del meccanismo di eccezione per overflow aritmetico

- L'eccezione "Arithmetic overflow"
  - Puo' essere implementata con un ulteriore stato #14

- L'Unita' di Controllo di un microprocessore reale deve inoltre
  - Gestire tutti gli eventi che si possono generare a causa di eccezioni e le varie interazioni fra istruzioni in modo tale che la logica di controllo rimanga semplice e veloce

Modification to the Control Specification



Lettura/Scrittura di EPC, Cause, Status, BadVAddr

- mtc0 xx, $2 ➔ xx ← $2
  move to coprocessor 0 from register
- mfc0 $2, xx ➔ $2 ← xx
  move from coprocessor 0 into register

- lwc0 xx, 100($2) ➔ xx ← MEM[$2+100]
  load word into coprocessor 0 from memory
- swc0 xx, 100($2) ➔ MEM[$2+100] ← xx
  store word from coprocessor 0 into memory

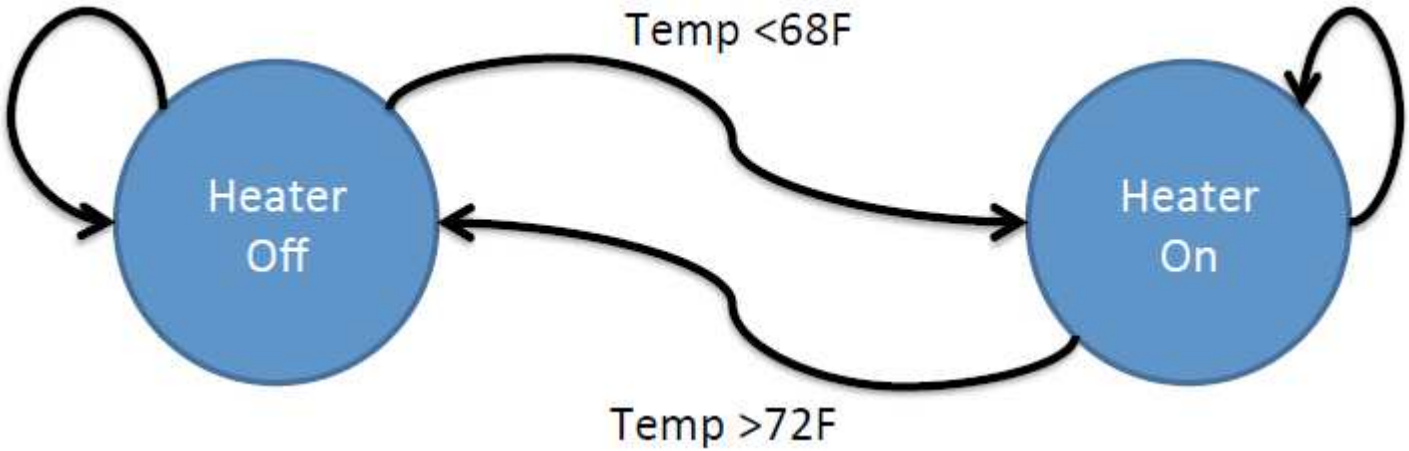
- xx indica uno dei registri speciali CAUSE, EPC, …

Gestore delle Interruzioni

```
.ktext 0x80000080
mfc0  $k0, $13          # $k0 ← Cause  (Cop0 reg #13)
mfc0  $k1, $14          # $k1 ← EPC    (Cop0 reg #14)

andi  $k0, $k0, 0x3C    # prelevo il tipo dell'eccezione (bit 5..2)
beq   $k0, $0,  fatto   # ignora le interruzioni (ma non le eccezioni)

sw    $a0, save0        # ipotesi: gestore non rientrante
sw    $a1, save1        # necessario evitare uso stack (puo' essere la causa…)
add   $a0, $k0, $0
add   $a1, $k1, $0
jal   gestione          # routine di gestione eccezione, param. input $a0,$a1
lw    $a0, save0
lw    $a1, save1

fatto:
addiu $k1, $k1, 4       # istr. succ. a quella che ha causato l'eccez.
rfe                     # restore from exc.: ripristina mask e (k,e) in Status
jr    $k1               # SALTA all'istr.succ. a quella che ha causato l'eccez

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

Macchine a Stati Finiti

- Un gruppo di stati e di transizioni



- Ci si sposta da uno stato all'altro seguendo una linea di transazione SE la condizione dalla transizione e' soddisfatta

Macchine a Stati Finiti (2)

- Nel campo dell'architettura dei calcolatori per le FSM:
  - Le transizioni tipicamente avvengono su fronti del clock
  - Sono complete
    - Tutti gli stati definiscono transizioni per tutti gli input
  - Sono deterministiche
- Possono essere trasformate in reti logiche dette Reti Sequenziali
  - Lo vedremo prossimamente.