Interrupt Mechanisms in the 74xx PowerPC Architecture

Porting Plan 9 to the PowerPC Architecture

Ajay Surie
Adam Wolbach

Definitions

- MSR_{FOO} FOO bit of MSR
- SRRx Save/Restore Register X
- [y, z) Memory, spanning y to z (not including z)

Interrupt Classes

- Four Classes of System-Caused Interrupts
  - System Reset, Machine Check
    - Not maskable
  - External, Decrementer (Timer)
    - Maskable, taken if MSR_{EE} bit is set to 1
- Two Classes of Instruction-Caused Interrupts
  - Precise: System calls, most exceptions
  - Imprecise: Floating-Point Enabled Exception
    - No guarantees with knowing which instruction actually caused the exception

Interrupt Vectors

- A vector is a region in main memory containing the initial sequence of instructions to be executed upon taking an interrupt
  - Vector location unique to each type of interrupt
  - 256 bytes / 64 instructions allotted per vector
    - Enough to do some register manipulation and call an operating system's handler function
    - Not a concrete rule
  - [0x0, 0x3000) used for vectors in main memory
    - [0x0, 0x1000) used for architecture-defined interrupts
    - [0x1000, 0x3000) are implementation-specific

Outline of Interrupt Processing

- An interrupt can only occur when it has a higher priority level than any currently occurring interrupt
- SRR0 loaded with instruction address depending on the type of interrupt
  - Generally, tries to identify culprit, or next to execute
- Important bits of MSR (0, 5:9, 16:31) saved in SRR1
  - Bits 1:4 and 10:15 contain interrupt-specific information
- $MSR_{IR,\ DR,PR}$ set to 0
  - Virtualization off, kernel mode
- MSR_{RI} set if interrupt is “recoverable”

Interrupt “Ordering” and Program State

- System Reset and Machine Check interrupts are not “ordered”
  - Can occur at any time
  - Program state may be lost
- All other interrupts are “ordered”
  - Only one interrupt is reported at same time
  - When it is processed, no program state is lost
- Save/Restore Register 0 and 1 (SRR0/1)
  - Used in the saving of context

Important Bits in the MSR

- IP[25]: Interrupt Prefix
  - Controls the prefix of where interrupt vectors are stored in real memory (0xfffff000 if set, 0x0 if not)
- RI[30]: Recoverable Interrupt
  - If this is set on an interrupt, state can be salvaged
  - Hardware determines if state is salvageable

Plan 9 Interrupt Handling Overview

- All exception vectors contain an instruction sequence that calls trapvec(SB) to handle state saves / mode changes
- On an interrupt, virtualization is disabled
- The kernel determines whether a stack switch is necessary
  - This can be accomplished by determining the mode in which the interrupt occurred, stored in SRR1
- After registers are saved, virtualization is renabled and the kernel determines the appropriate handler to run

Plan 9

- Vector contains instruction sequence to an assembly routine that handles the interrupt
- If the interrupt was in user mode, find the wrapper routine

System Reset Interrupt

- Vector location: 0x100 (RA), 256 bytes
- Can be hardware or software generated
- SRR0 set to EA of instruction that would have executed next without this interrupt
- SRR1's interrupt info set to 0, MSR copied
- "Implementations can provide a means for software to distinguish between power-on Reset and other types of System Reset"
- Can be recovered from if $MSR_{RI} = 1$

Machine Check Exception

- Vector location: 0x200 (RA), 256 bytes
- Enabled if MSR_{ME} = 1 when exception hit
  - If MSR_{ME} = 0, machine enters Checkstop state
- Caused by hardware dying, temperature problem, or possibly by referencing a nonexistent RA
  - I think; implementation definitely processor-specific though
- SRR0 set on “best effort” basis to the instruction executing when the exception hit
- SRR1 set to processor-specific value
- If storage registers are valid, MSR_{RI} set to 1 and resumption of execution can occur

External Interrupts

- Vector location: 0x500 (RA), 256 bytes
- Generic for all external hardware interrupts: keyboard, mouse, etc, but not timer
- Occurs when MSR_{EE} = 1 and an external interrupt exception is presented to CPU
- SRR0 contains next instruction to execute, as if no interrupt had occurred
- SRR1 set as outlined

Decrementer (Timer) Interrupt

- Vector location: 0x900 (RA), 256 bytes
- Decrementer is a 32-bit register that acts as a countdown timer, causing an interrupt after passing through zero
  - Frequency is processor-specific
  - Interesting: Speculative execution can possibly read decrementer in advance of actual execution, getting old value; fixed with an isync before decrementer reads
- Occurs when MSR_{EE} = 1 and a decrementer exception is presented to CPU
- SRR0 contains next instruction to execute, as if no interrupt had occurred

Plan 9 Clock / Timer

- Decrementer used to maintain ticks since boot
- The timer is board specific and is handled as an external interrupt
  - Causes a context switch every 10 ms

System Calls

- Vector location: 0xC00 (RA), 256 bytes
- Occurs when system call instruction executes
  - Determining which system call is to be executed is something that is handled by the operating system
    - In Plan 9, R3 contains the number of the system call intended for execution
- SRR0 set to address of instruction after SC
- SRR1's interrupt info set to 0, MSR copied

Plan 9 System Calls

- System calls are all mostly machine independent (except fork, exec, etc.)
- A generic system call handler validates user stack state, etc.
- R3 contains the number of the system call to be executed
- After the system call executes, the kernel places the return value in R3, and restores the user mode state

Instruction Storage Interrupt

- Vector location: 0x400 (RA), 256 bytes
- Occurs on an instruction fetch when an EA cannot be translated, EA is in a direct-store segment, or a violation of storage protection
- SRR0 holds faulting instruction's EA
- $SRR1_1$ set if it was a hashed translation miss
- $SRR1_3$ set if it was a direct-store segment
- $SRR1_4$ set if storage access not permitted
- $SRR1_{10}$ set if segment table failed to find a translation

Data Storage Interrupt

- Vector location: 0x300 (RA), 256 bytes
- Occurs on direct-store errors with external devices, EA translation failures on data loads or stores, or a violation of storage protection
- SRR0 set to faulting instruction's EA
- Data Storage Interrupt Status Register holds information specific to DSI type
- Data Address Register set to the EA of the data access that failed

Less Interesting Interrupts

- Alignment Interrupts
  - Load/Store not aligned to size of data type
- Program
  - Illegal Instruction, Not privileged
- Trace
  - If enabled, occurs after every non-rfi instruction
- Several Floating-Point Exceptions
  - Divide-by-zero, etc.

Returning From Interruption (IRET)

- To return to normal execution, the following needs to occur
  - MSR_{RI} set to 0
  - SRR0/1 possibly set to values to be used by rfi
  - Execute rfi instruction
    - SRR1 copied into MSR
    - SRR0 copied into Next Instruction Address Register
  - Normal execution resumes

Precise / Imprecise Interrupts

- Upon taking a precise interrupt:
  - SRR0 points to instruction causing the exception or some instruction a known distance after it, depending on the interrupt's type
    - Guaranteed that all previous instructions have completed, and no subsequent instructions have begun processing on this processor
- Upon taking an imprecise interrupt:
  - SRR0 points to some unknown instruction, either at or after the instruction causing the interrupt
- All instruction interrupts are precise, except for floating-point enabled exceptions

Bibliography

- The book