Interrupts and Exceptions

Studying Assignment: A.7
Reading Assignment: 3.1-3.2, A.10

CPU Modes and Address Spaces

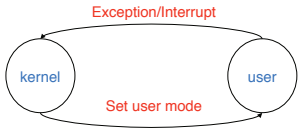
There are two processor (CPU) modes of operation:
- Kernel (Supervisor) Mode and
- User Mode

The processor is in Kernel Mode when CPU mode bit in Status register is set to zero. The processor enters Kernel Mode at power-up, or as result of an interrupt, exception, or error. The processor leaves Kernel Mode and enters User Mode when the CPU mode bit is set to one (by some instruction).

Memory address space is divided in two ranges (simplified):
- User address space
  – addresses in the range [0 – $7FFFFFFF_{16}$]
- Kernel address space
  – addresses in the range [80000000_{16} – $FFFFFFFF_{16}$]

Dual-Mode of CPU Operation

- *CPU mode bit* indicates the current CPU mode: 0 (=kernel) or 1 (=user).
- When an interrupt occurs, CPU hardware switches to the kernel mode.
- Switching to user mode (from kernel mode) done by setting CPU mode bit (by an instruction).



Privileged instructions can be executed only in kernel mode.

MIPS Privilege Instructions

With CPU in User Mode, the program in execution has access only to the CPU and FPU registers, while when CPU operates in Kernel Mode, the program has access to the full capabilities of processor including CP0 registers.

Privileged instructions can not be executed when the processor is in User mode, they can be executed only when the processor is in Kernel mode.

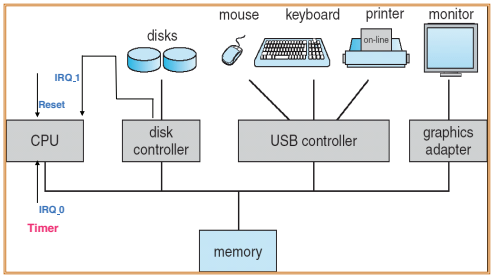
Examples of MIPS privileged instructions:
- any instruction that accesses Kernel address space
- all instructions that access any of CP0 registers, e.g. move word from CPU register to CP0 register

Interrupts Classes

There are two classes of interrupts:

1. Interrupts caused by external signals:
- Reset: A signal asserted on the appropriate pin;
- Hardware Interrupts: Interrupt requests made via asserting signal on any of special external pins.
- Interrupt request lines – IRQ's.
- Usually used by I/O controllers to signal normal I/O completion or a variety of error conditions. These interrupts also called I/O interrupts.
- Hardware interrupts can be masked by setting appropriate bits in Status register;

Interrupts by External Signals



There are also interrupts caused by hardware failure, such as power failure and memory parity error.

Interrupts Classes (continued)

- 2. Interrupts as result of instruction execution (these are also called exceptions); There are two types of exceptions:

  Type A caused by problems during instruction execution:

  - Address Error: a reference to a nonexistent or illegal memory address;
  - Reserved Instruction: An instruction with undefined opcode field or a privileged instruction in User mode;
  - Integer Overflow: An integer instruction results in a 2's complement overflow;
  - Floating Point Error: e.g. divide by zero, overflow, and underflow;

  Type B caused by special instructions:

  - MIPS processors: Syscall instruction executed;
  - Intel processors: INT n instruction executed;

Hardware Interrupt Processing in General

- Hardware interrupt processing always saves the address of the interrupted instruction:
  - In MIPS architecture, EPC register gets that address
  - Many architectures use a system stack (in memory) to save that address
- Hardware interrupt processing should transfer control to the appropriate interrupt service routine (software):
  - In MIPS architecture, the interrupt service routine must start at memory address 80000180_{16}, and then based on the content of Cause register appropriate (software) processing is performed,
  - In many architectures, new content into PC comes from one of special memory locations (*interrupt vectors*), which should contain addresses of service routines. Each interrupt has its specific vector address, built in the hardware as a part of ISA.

  It is responsibility of an operating system to load interrupt handling routines (software) starting at the appropriate memory locations, or to load those special memories with addresses of interrupt routines.
- Hardware interrupt processing always sets CPU into kernel mode.

MIPS Interrupt Processing

- When any of the interrupts previously listed occurs, hardware should perform some predefined (by its ISA) tasks.
- Here we describe in some level of details how MIPS processor processes interrupts.
- MIPS does hardware interrupt processing in three steps.

Step 1. (Saving content of PC)

- EPC register gets a value equal to either:
  - the address of a faulty instruction if the instruction itself caused problems (e.g. address error, reserved instruction) or hardware malfunctioning detected (e.g. memory parity error),
  - the address of the next instructions which would have been executed in all other cases, i.e. for interrupts caused by external causes or by those interrupt causing instructions.

MIPS Interrupt Processing (continued)

Step 2. (PC gets new value and interrupt cause code is saved)

- PC ← 80000180_{16}

  Thus, the next instruction is fetched from location 80000180_{16}
- Cause register ← a code of the interrupt

  Each interrupt has its code, e.g.:
  - hardware interrupt = 0
  - illegal memory address (load/fetch or store) = 4 or 5
  - bus error (fetch or load/store)= 6 or 7
  - syscall instruction execution = 8
  - illegal op-code, i.e. reserved or undefined op-code= 10
  - integer overflow = 12
  - any floating point exception = 15

Step 3. (Mode of CPU operation set to kernel mode)

- CPU mode bit ← 0;