Computer Organization and Design (4th)
by Hennessy, Patterson
Chapter 6.10, Problem 1E

Step 1

If the highest priority interrupt is disabled and then the low-priority interrupt is disabled. Because the low-priority interrupts to stop them from interrupting handling of the current interrupt.

Step 2

The status register is saved to assure that any lower priority interrupts that have been detected are handled with the status register is restored the current interrupt.

Computer Organization and Design (4th)
by Hennessy, Patterson
Chapter 6.10, Problem 2E

Step 1

Assume the lower numbers have higher interrupt priorities

a)

Priority	Interrupt
1	Overheat
2	Power Down
3	Ethernet Controller Data

Step 2

b)

Priority	Interrupt
1	Overheat
2	Reboot
3	Mouse Controller

Computer Organization and Design (4th)
by Hennessy, Patterson
Chapter 6.10, Problem 3E

Step 1

a) Power Down: Jump to an emergency power down sequence and begin execution.

Overheat: Jump to an emergency power down sequence and begin execution.

Ethernet Controller Data: Jump to the Ethernet controller code and handle data input.

Step 2

b) Overheat: Jump to an emergency power down sequence and begin execution.

Reboot: jump to address zero and reinitialize the system.

Mouse Controller: Jump to the mouse controller code and handle input. Restore the program state and continue execution.

Computer Organization and Design (4th)
by Hennessy, Patterson
Chapter 6.10, Problem 4E

Step 1

If the interrupt enable bit of the Cause register is not set, all interrupts are disabled and no interrupts will be handled. By setting all bits to zero in the mask then all interrupts also will be disabled.

Computer Organization and Design (4th)
by Hennessy, Patterson
Chapter 6.10, Problem 5E

Step 1

The hardware support for saving and restoring program state prior to interrupt handling would help substantially. In particular, when an interrupt is handled that does not terminate execution, the running program must return to the point where the interrupt occurred. Handling this in the operating system is certainly feasible, but the only solution requires storing information on a stack or some other dedicated memory area. In some case, registers are dedicated to this task.

Step 2

Providing hardware support removes the burden from the operating system and program state need not be pulled from the CPU and put in memory. This is essentially the same has handling a function call, except that some interrupts do not allow the interrupted program to resume execution. Like an interrupt, a function must store program state information before jumping to its code. There are sophisticated activation record management protocols and frequently supporting hardware for many CPUs.

Computer Organization and Design (4th)
by Hennessy, Patterson
Chapter 6.10, Problem 6E

Step 1

The interrupt handler implements the priority interrupts. First higher priority interrupts handled and then lower priority interrupts are disabled when higher priority interrupt is being handled. Even though each interrupt causes a jump to its own vector, the interrupt system implementation is still handles the interrupt signals and both approaches have the same capabilities.