Main Points

• Kernel control transfer
  – Interrupt: how do we resume execution after an interrupt as if the interrupt hadn’t happened?
  – System call: how do we execute a procedure called from an application, but implemented in the kernel?
  – Upcall: how do we deliver an event to user level?

• Concurrency introduction
  – More in section and on Friday
Interrupt Mechanics

• Processor saves any user level state
  – MIPS: special registers to hold user PC, SP
  – x86: hardware puts these on kernel stack
• Processor jumps to first instruction in handler
• Handler saves remaining registers
  – Any registers it will clobber (depends on compiler)
  – Floating point if necessary (not in OS/161)
• Handler runs on kernel stack, with interrupts disabled, must be short