Lecture Notes 15.14 ECS 154A
2018-08-29 Interrupts, memory Summer Session II 2018

1. More on interrupts
    a. Last time – discussing changes we need to make to support interrupts
        i. Modify our original program order of Fetch, Decode, Execute
            1. Add Check for interrupts to the beginning or end (CFDE or FDEC)
        ii. Add a place in memory to store the ISR code / instructions
            1. Need to protect this place in memory from being modified by any user processes
            2. ISRs tend to be privileged to handle data from hard drive, caches, so on
                a. If not protected, malicious programs can modify ISR
                b. Modified code would run any time interrupt is handled by OS
        iii. Support different types of interrupts
            1. Have different interrupts for keyboard versus hard drive
        iv. Need to be able to enable and disable interrupts
            1. What happens if we keep getting interrupted when handling an interrupt?
            2. Would never get anything done, would keep getting interrupted
        v. Need to know what to load into PC
            1. Interrupt changes program flow, as mentioned earlier
        vi. Need to add an Interrupt Service Routine (ISR)
            1. Routine is the address that gets loaded into the PC
            2. Handles the interrupt
        vii. Need a Return from Interrupt (RTI) instruction
            1. Once ISR is done, restore original state and go back to where we left off

2. Classification of interrupts
    a. Examples of possible interrupts
        i. Power failure
        ii. Arithmetic overflow
        iii. I/O device request
        iv. OS call (system call or syscall)
        v. Page fault (in virtual memory)
    b. How to classify interrupts
        i. By timing (with the clock)
            1. Synchronous (deterministic) – function of program and memory state
                a. Overflows, page faults
            2. Asynchronous (nondeterministic) – external device, or hardware malfunction
                a. Printer ready, bus error
        ii. Type of user
            1. User request – from a user program (OS / system call)
            2. Coerced – from the OS or hardware (page fault, protection violation)
        iii. Masking
            1. User maskable – can be temporarily ignored (overflow, user-set breakpoint)
            2. Non-maskable – must be handled (power failure, page fault, reset button)
        iv. Location (or time) in instruction
            1. Within an instruction – must be dealt with to finish instruction (page fault)
            2. Between instructions – not part of an instruction (I/O device request, OS call)
        v. Result
            1. Resume - transparently return to user process (page fault, I/O request)
            2. Terminate – give up and die (protection violation, power failure)

3. Memory overview
    a. Terminology
        i. Word – usually the size of an int, 32 bits
            1. Bytes are a fixed size, words are not!
        ii. Addressable units – usually bytes, but can be words
            1. 2^A = number of addressable units, where A is the number of bits in an address
        iii. Unit of transfer – number of bits written out or read into memory at a time
            1. Same idea as bus width

4. Characteristics of memory systems (incomplete list)
    a. Physical type
        i. Need two well-defined states in the medium to differentiate 0 and 1
        ii. Semiconductor – flip-flops, capacitors, so on
        iii. Magnetic surface – stored using magnetism (like hard drives)
        iv. Optical – CDs, Blu-ray, and the like
    b. Volatility
        i. Non-volatile – retains information when power is off (hard drives)
        ii. Volatile – loses information when power is cut (registers, flip flops, RAM)