Chapter 1

- **Learning objectives:**
    - Describe the basic elements of a computer system and their interrelationship.
    - Explain the steps taken by a processor to execute an instruction.
    - Understand the concept of interrupts and how and why a processor uses interrupts.
    - List and describe the levels of a typical computer memory hierarchy.
    - Explain the basic characteristics of multiprocessor and multicore organizations.
    - Discuss the concept of locality and analyze the performance of a multilevel memory hierarchy.
    - Understand the operation of a stack and its use to support procedure call and return.

- **Instruction cycle**
    - Fetch next instruction (address pointed to by PC) place into IR (via MAR & MBR)
    - Execute instruction (instruction contains opcode & target memory address)
    - (*If interrupts enabled*) check for interrupts; if present, initiate interrupt handler
- **Interrupts**
    - Handle interrupt:
        - Save PC, PSW & general register values onto control stack
        - Increment stack pointer accordingly
        - Load new PC from interrupt code
    - Return from interrupt:
        - Restore PC, PSW & general register values from control stack
        - Restore stack pointer
    - Multiple interrupts
        - Sequential processing – disable interrupts during interrupt
            - Con: no priority
        - Nested processing – high priority call can interrupt low priority interrupt call
- Programmed I/O – processor must poll I/O device for results
- Interrupt-driven I/O – uses interrupts
- **Caching**
    - Useful because of locality of reference
    - Main memory contains many blocks (size = K words)
    - Cache contains lines (size = K words) – much fewer than the # of blocks in memory
    - When a word (RA) is looked up, check if it's in cache
        - If yes = hit; return RA
        - If no = miss; copy block in main memory that contains RA into a line in cache
    - Design strategies:
        - Cache size
        - Block size
        - Mapping function (where in cache to place new blocks)
        - Replacement algorithm (e.g. least recently used/LRU)
        - Write policy (when to update changes in cache to memory)

        - Number of cache levels

- **Direct memory access**
    - More efficient for bulk data transfers
    - Processor delegates I/O operation to DMA module; it only specifies:
        - Read or write
        - Address of I/O device
        - Starting address & size of read/write

<u>Chapter 2</u>

- **Learning objectives:**
    - Summarize, at a top level, the key functions of an operating system (OS).
    - Discuss the evolution of operating systems for early simple batch systems to modern complex systems.
    - Give a brief explanation of each of the major achievements in OS research, as defined in Section 2.3.
    - Discuss the key design areas that have been instrumental in the development of modern operating systems.
    - Define and discuss virtual machines and virtualization.
    - Understand the OS design issues raised by the introduction of multiprocessor and multicore organization.
    - Understand the basic structure of Windows 7.
    - Describe the essential elements of a traditional UNIX system.
    - Explain the new features found in modern UNIX systems.
    - Discuss Linux and its relationship to UNIX.