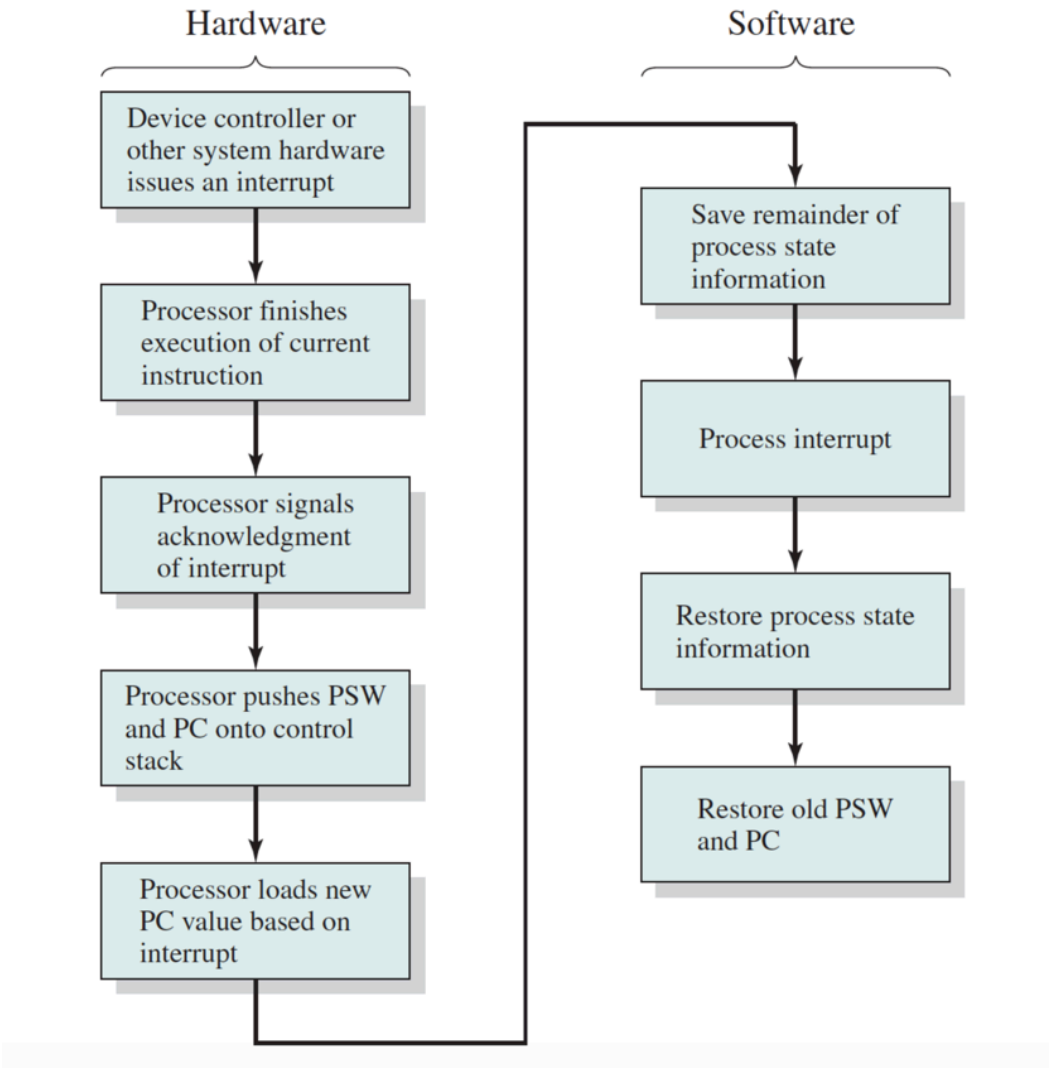
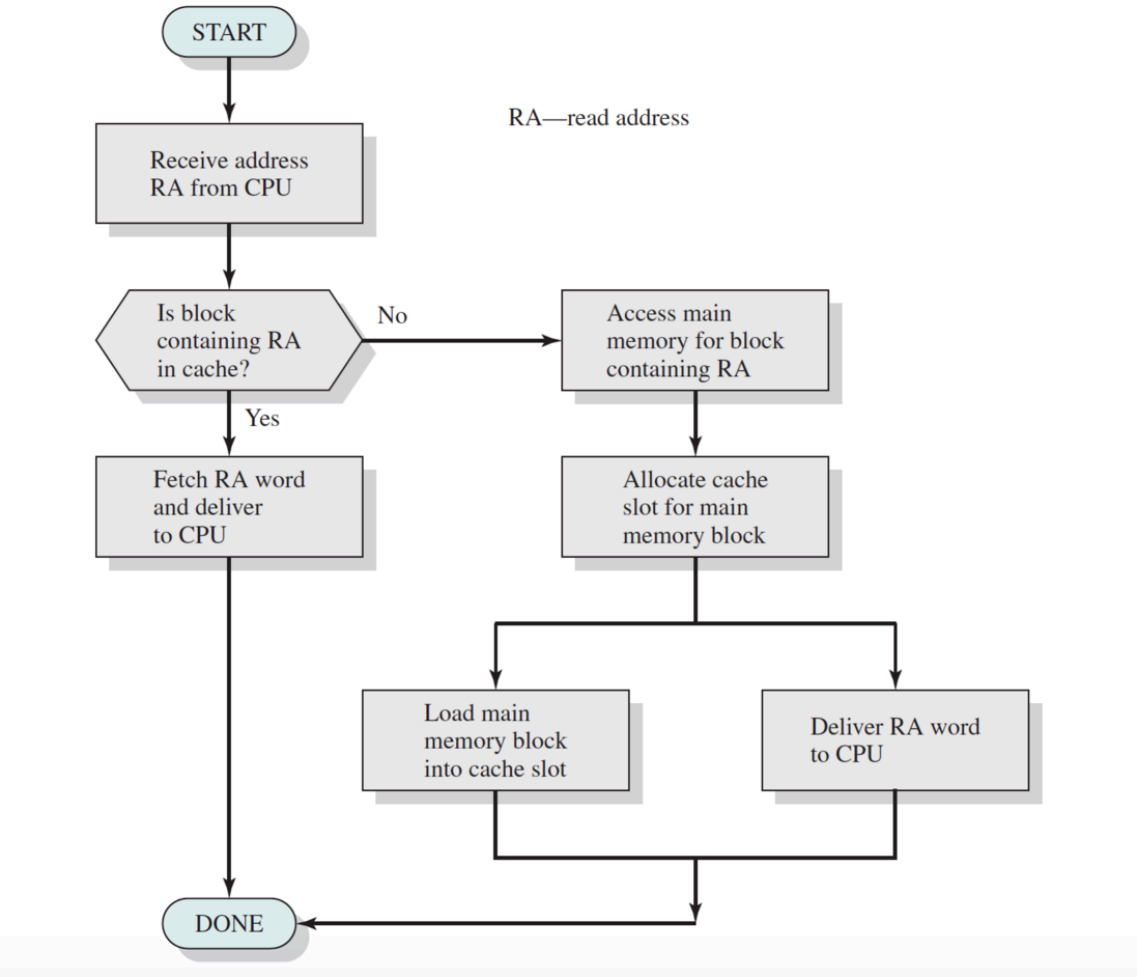
Chapter 1

- Generally 4 types of instructions:
  - Processor ↔ memory
  - Processor ↔ I/O
  - Data processing
  - Control
- **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:
    - Finish execution of current instruction & signal acknowledgement
    - 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



- **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



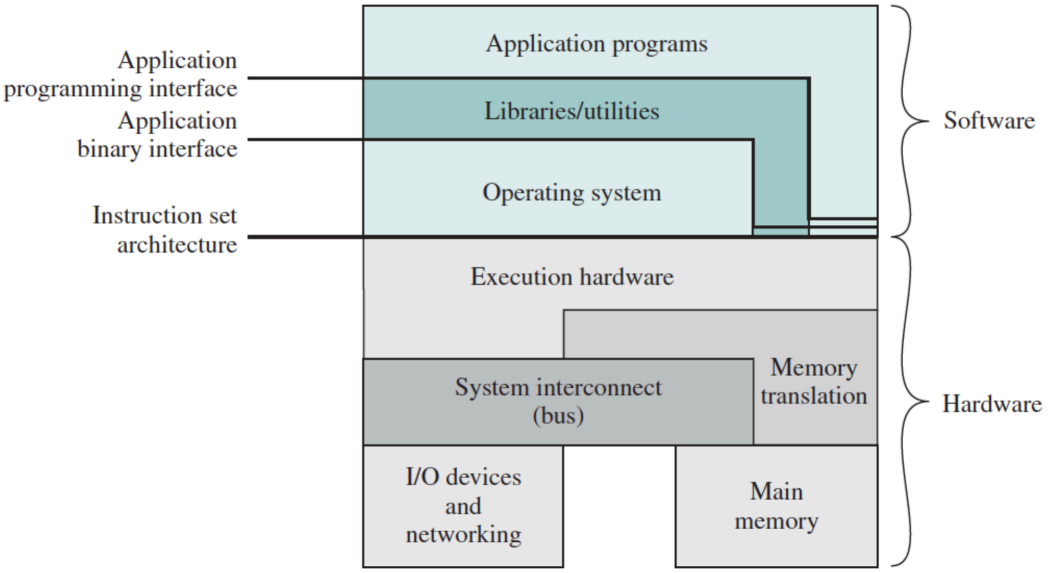
- Programmed I/O – processor must poll I/O device for results
- Interrupt-driven I/O – uses interrupts
- Both – processor must manage each I/O transfer
- **Direct memory access**
  - More efficient for bulk data transfers
  - Processor delegates I/O operation to DMA module – as opposed to reading/writing words through processor
  - Only specifies:
    - Read or write
    - Address of I/O device
    - Starting address
    - Size (# of words)
- **Symmetric multiprocessor (SMP)**
  - ≥ 2 similar processors of comparable capability
  - Share the same memory and I/O access
  - All processors can perform the same functions (symmetric)
  - Controlled by an integrated OS that provides interaction between processors
  - Advantages of SMP:

        - Performance
        - Availability (redundancy against failures)
        - Incremental growth (adding more processors)
        - Scaling (vendors can offer a range of products)
    - Disadvantage:
        - Each processor has private cache – each cache invalidation has to happen in multiple places
- Multicore processor
    - Multiple processors on the same silicon chip

Chapter 2

- **Objectives of an operating system:**
    - Convenience (as a user/computer interface)
    - Efficiency (as a resource manager)
    - Ability to evolve
- An OS provides services for:
    - Program development
    - Program execution
    - Access to I/O
    - Access to files
    - Control of system access
    - Error detection & response
    - Accounting & usage statistics
- Kernel – portion of OS that's in main memory
- Key interfaces in a computer system:
    - Instruction set architecture (ISA)
        - Hardware vs. software; set of machine language instructions
    - Application binary interface (ABI)
        - OS vs. libraries & utilities; interface for system calls
    - Application programming interface (API)
        - Libraries & utilities vs. user programs; interface for high-level language library calls

- The OS is a control mechanism that often gives control away for the processor to do "useful work", and then has control returned to it by the processor

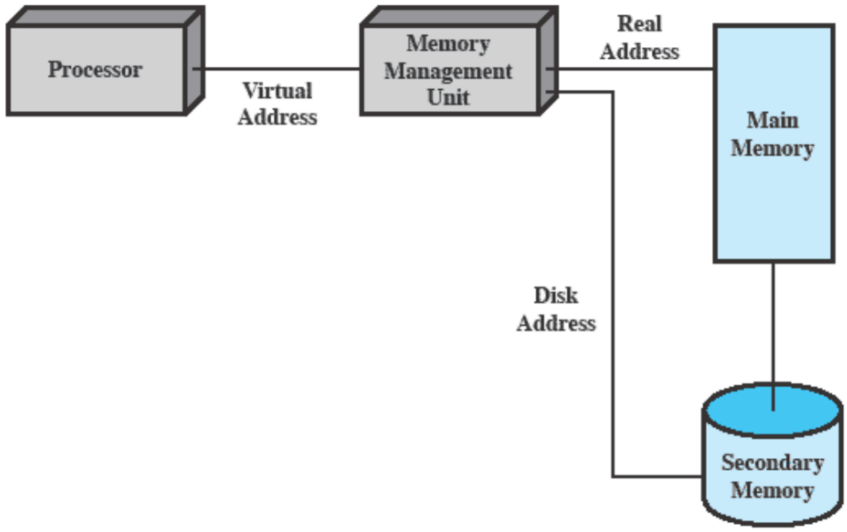


- **Evolution of the OS**
  - Serial processing
    - Each job is run one at a time and one after another
    - Disadvantages:
      - Manual scheduling results in processing time wasted
      - Setup time associated with each job takes too long
  - Batch OS
    - Monitor – software that stays in main memory & controls the sequence of events
    - User doesn't have direct access to processor
    - Jobs are batched together sequentially and executed
    - Processor control is returned to monitor after every job is done
    - Job control language gives special instructions to the monitor
    - Hardware features:
      - Memory protection
      - Timer
      - Privileged instructions (kernel mode vs. user mode)
      - Interrupts
  - Uniprogramming vs. multiprogramming
    - Uni – process only one job at a given time
    - Multi - Processor can run other jobs while waiting on I/O for a particular job
  - Time sharing
    - Share processor time among many users
    - Time slicing – use system clock to interrupt and reassign processor control to different users

- Too fast for humans to notice

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

- Major achievements:
  - Process
    - A program in execution; or
    - An entity that can be assigned to and executed on a processor; or
    - A unit of activity with thread of execution, a state, and an associated set of resources
  - Memory management
    - Process isolation
    - Automatic allocation and management
    - Support of modular programming
    - Protection and access control
    - Long-term storage
  - Information protection & security
    - Availability
    - Confidentiality
    - Data integrity
    - Authenticity
  - Scheduling & resource management
    - Fairness
    - Differential responsiveness
    - Efficiency
- Virtual memory



- **Fault tolerance**
  - Reliability = probability of correct operation up to time *t*
  - Mean time to failure = average uptime
  - Mean time to repair = average downtime