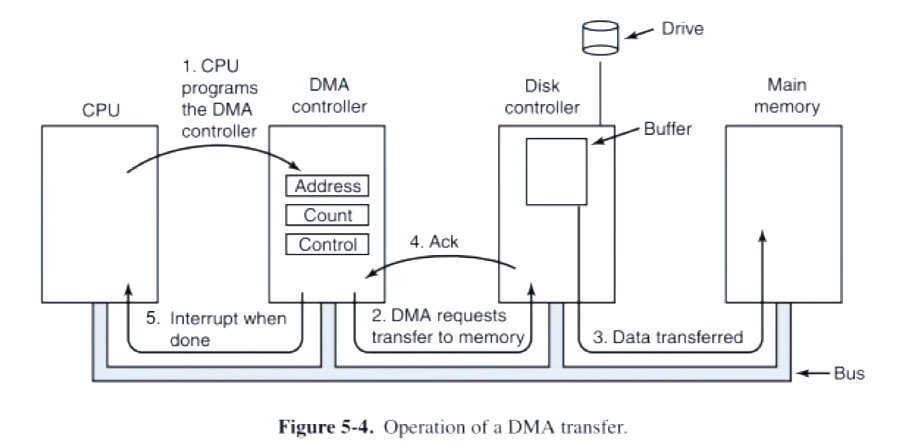
CS61C – 11/22/16

* Memory Mapped I/O : Device control/data registers mapped to CPU address space
* CPU synchronizes with I/O device:’
  + Polling
  + Interrupts
* Programed I/O:
  + CPU execs lw/sw instructions for all data movement to/from devices
  + Takes up great amounts of time
  + CPU spends time doing 2 things:
    - Getting data from device to main memory
    - Using data to compute
* Introduces DMA instead of Programmed I/O: Direct Memory Access
  + Not ideal because
    - CPU has to execute all transfers, could be doing other work
    - Device speeds don’t align well with SPU speeds (CPU faster)
    - Energy cost of using beefy general-purpose CPU where simpler hardware would suffice
  + Allows I/O devices to directly read/write main memory
    - CPU does not have to wait for their read/writes and do it itself
  + New Hardware: DMA Engine
  + DMA Engine contains registers written by CPU:
    - Memory address to place data
    - # of bytes
    - I/O device #, direction of transfer
    - Unit of transfer, amount to transfer per burst



* + Incoming Data from DMA to CPU
    - Receive interrupt from device
    - CPU take interrupt, begins transfer
      * Instructs DMA engine/ device to place data @ certain address
    - Device/DMA engine handle the transfer
      * CPU is free to execute other things
    - Upon completion, Device/DMA engine interrupt the CPU again
  + Outgoing Data
    - CPU decides to initiate transfer, confirms that external device is ready
    - CPU begins transfer
      * Instruct DMA engine/device that data is available @ certain address
    - Device/DMA engine handle the transfer
      * CPU is free to execute other things
    - Device/DMA engine interrupt the CPU again to signal completion
* Great Ideas: Computer Memory Hierarchy
  + Magnetic Disk
    - Kind of computer memory
      * Information stored by magnetizing ferrite material on surface of rotating disk
        + Similar to tape recorder except digital rather than analog data
    - Type of non-volatile storage
      * Retains its value without applying power to disk
    - Two types of magnetic disk
      * Hard Disk Drives (HDD) – faster, more dense, non-removable.
      * Floppy disks – slower, less, dense, removable (now replaced by USB “flash drives”).
    - Purpose in computer systems (Hard Drive):
      * Working file system + long-term backup for files
      * Secondary “backing store” for main-memory. Large, inexpensive, slow level in the memory hierarchy (virtual memory).
    - It is slow because of Disk Access Time
      * Disk access time = seek time + rotation time + transfer time + controller overhead
        + Seek time = time to position the head assembly at the proper cylinder

Average number of tracks to move arm

Number of tracks/3

Seek time = number of tracks moved \* time to move across one track

* + - * + Rotation time = time for the disk to rotate to the point where the first sectors of the block to access reach the head

Average distance of sector from head

½ time of rotation

* + - * + Transfer Time = time taken by the sectors of the block and any gaps between them to rotate past the head.
  + Flash Memory/SSD Technology
    - NMOS transistor with an additional conductor between gate and source/drain which “Traps” electrons.
    - Has a limited number of program-erase cycles
* Network: talking to the Outside World
  + Originally sharing I/O devices between computers
    - eg pinters
  + then communicating between computers
    - eg. file transfer protocol
  + then communicating between people
    - eg. email
  + then communicating between networks of computers
    - eg. File sharing, www, …
* shared vs. Switch-based networks
  + shared vs switched:
    - shared: 1 at a time
    - switched : pairs (“point-to-point” connections) communicate at the same time
  + Aggregate bandwidth (BW) in switched network is many times that a shared:
    - Point-to-point faster since no arbitration, simpler interface
* Networks:
  + Links connecting switches and/or routers to each other and to computers or devices
  + Ability to name the components and to route packets of information – messages – from a source to a destination
  + Layering, redundancy, protocols, and encapsulation as means of abstraction
* Software Protocol to Send and Receive
  + Divides data into packets that have header (address, destination, source, lenth), Payload (data), trailer (checksum)
  + SW send steps
    - Application copies data to OS buffer
    - OS calculates checksum, starts timer
    - OS sends data to network interface HW and says start
  + SW Receive steps
    - OS copies data from network interface HW to OS buffer
    - OS calculates checksum, if OK, send ACK; if not, delete message (sender resends when timer expires)
    - If OK, OAS copies data to user address space, and signals application to continue
* Protocol for Networks of Networks
  + Multiple layers
  + Use abstraction to cope with complexity of communication
  + Hierarchy of layers:
    - Application (chat client, game, etc)
    - Transport (TCP, UDP)
    - Network (IP)
    - Data Link Layer (Ethernet)
    - Physical Link (copper wireless, etc.)
* Protocol Family Concept
  + Protocol: packet structure and control commands to manage communication
  + Protocol Families (suites): a set of cooperating protocols that implement the network stack
  + Key to protocol families is that communication occurs logically at the same level of the protocol, called peer-to-peer, but is implemented via services at the next lower level
  + Encapsulation: carry higher level information within lower level “envelope”
* Semantic content->Identity->Location
* Concept
  + Message gets mapped with header and trailer and then another packaging of header trailer
  + Receiving end unpacks one layer by one layer until message
* Most popular protocol for network of networks
  + Transmission Control Protocol/ Internet Protocol (TCP/IP)
    - Application send message
    - TCP breaks into 64KiB segments, add 20B header
    - IP adds 20B header, sends to network
    - If Ethernet, broken into 1500Bpackets with headers, trailers
* Storage Attachment Evolution
  + Direct Attachment
    - Host connects directly to Disk Interface (DI) and memory
  + Network Server Attachment
    - Host goes through Network Interface (NI) to LAN and to Network File Server then to memory
      * r/w on file level
    - Also possible to not go through Network File Server, but through Network-attached Storage(NAS)
      * r/w on disk drive level
  + channel attached storage
    - host connects to disk storage subsystem
* LUN = logical unit
* Storage class memory