**The Memory System**

* A memory system is built using an array of memory cells
  + Each cell = 1 bit
  + Cells are grouped into addressable words
  + k address bits allow 2k unique addresses/words of memory
  + Address decoder decodes k address bits → selects one row of cells (i.e. one word of data) at the decoded address
  + Each column of cells has a sense/write circuit
    - Has 2 bit lines, read/write line, and chip select line
  + E.g. 1Kib x 1 = 1024 bits of memory with 1 bit of input/output
    - 32 rows x 32 columns
    - 10-bit address
      * First 5 bits → 5-to-32 decoder → row address
      * Second 5 bits → select lines for 32-to-1 output MUX/1-to-32 input deMUX → column address
      * One bit at row & column address is returned
* Memory latency – amount of time required for the transfer of a word of memory
* Memory bandwidth – number of bits/bytes that can be transferred/unit time
  + Depends on the access speed of data & the width of the communication channel
* **Random access memory** – access time to any memory location is the same
* Static RAM
  + Bits stored in latches
  + Data is retained as long as the power stays on (volatile)
  + Used for registers & high-speed caches
  + Costly
  + Memory cell latch – made of 2 inverters (4 transistors)
  + 2 switches open/close the connection to the 2 bit lines
  + 6 transistors in total – 6T SRAM
* Dynamic RAM
  + Bits are stored temporarily using capacitors
  + After a fixed duration, data is lost as the capacitors are discharged
    - Cells need to be refreshed periodically to retain the data
  + Used for large memory systems
  + Cheap
  + Charge threshold determines if capacitor holds a 1 or 0
  + During read
    - If capacitor charge is above threshold (1), recharge to full value
    - If charge is below threshold (0), drive bit line to ground to discharge completely
  + Asynchronous DRAM
    - Row address is placed first
      * RAS (row address strobe) signal is asserted
      * Row of cells is selected & cells are refreshed
    - Column address is then placed
      * CAS (column address strobe) signal is asserted
      * Desired byte is selected for read/write
    - No clock signal needed – only RAS & CAS (external signals)
    - CAS assertion needed for each column address – more delay
  + Fast page mode
    - Supports burst transfers in the same row by reusing the row address
    - Set row address at start of transfer, only reset column addresses for each cycle
  + Synchronous DRAM (SDRAM)
    - Built-in refresh circuitry using clock signal
    - Row & column selection are signalled by refresh counter
    - On read, contents of the entire row is loaded into sense/write amps
    - Data in the selected column → output register (buffer)
    - Buffer registers allow a new access to be made while data is still being transferred to/from the registers
    - SDRAM burst transfers
      * Column address is automatically incremented & consecutive words are selected
    - CAS assertion needed for only first column address – less delay
  + Double data rate SDRAM
    - Transfers data on both clock edges
    - Nearly doubles memory bandwidth for burst transfers
    - Organized into 2 banks of memory so that adjacent words in memory can be simultaneously accessed – switch between banks on each rising/falling clock edge
* Larger memory modules
  + Combine multiple chips
  + E.g. 2Mib x 32 bits – using 4 rows of 4 chips of 512Kib x 8 bits
    - 2Mib = 221 bits = 21 address bits
    - Bits 20 & 21 are decoded to select the row of chips (using chip select/CS line)
    - The other 19 bits are used to select row from each chip
    - Each chip returns 8 bits of data = 32 bits total
  + Columns = module output size (bit)/chip output size (bit)
  + Rows = module size (M)/chip size (M)
  + Types
    - Single in-line memory modules (SIMM)
    - Dual in-line memory modules (DIMM)
  + Memory controllers
    - Needed as an interface between SDRAM devices and the system bus
    - Issues appropriate CAS, RAS, CS etc. signals based on processor’s request
* **Read-only memory** – retains stored data when power is off (non-volatile)
  + Programmable ROM (PROM) – burn out fuses to indicate “1” bits; single-use
  + Erasable PROM (EPROM) – use transistors instead of fuses
  + Electrically erasable PROM (EEPROM) – memory is programmed
  + Flash memory – similar to EEPROM
* Direct memory access (DMA)
  + Program-controlled I/O requires the processor to be involved & interrupt state-saving etc.
    - Too much overhead for often small data transfers
  + DMA controller handles memory accesses for the processor
    - Generates required control signals
    - Increments address for consecutive words
    - Processor only needs to provide size and direction of transfer
* Memory hierarchy
  + (Smaller & faster & more expensive)
  + In processor
    - Registers
    - L1 cache
    - L2 cache
  + Main memory
  + Secondary memory (i.e. disk)
  + (Larger & slower & less expensive)
* **Caches**
  + Make memory seem faster to the processor by taking advantage of the locality of references
    - Temporal locality – instruction/data used recently
    - Spatial locality – instruction/data located near recently used data
    - These data are more likely to be used again – keep them in caches
  + Cache block – set of multiple words of data transferred between caches & main memory
  + Cache hit – desired memory block is found in cache
  + Cache miss – desired memory block is not found in cache
  + Cache eviction – remove a block of data from cache
  + Cache dirty bit – a bit that identifies if the contents of a cache block have been changed
  + Processor ↔ cache ↔ main memory
  + Processor load/stores ⇒ read/writes to/from the cache
* Cache read
  + On cache hit
    - Read contents directly from cache
  + On cache miss
    - Load contents from main memory into cache, then read from cache
    - Load-through protocol – processor may read directly from main memory
* Cache write
  + On cache hit
    - Write-through protocol – cache block & corresponding memory block are updated simultaneously
    - Write-back protocol – updates cache block & flags the change using dirty bit
  + On cache miss
    - Write-through protocol – main memory is updated directly
    - Write-back protocol – updates main memory, loads block into appropriate cache block & sets dirty bit
* Mapping functions – determining where in the cache to load data from memory
  + Direct mapping
    - Memory block j is stored in the cache block j mod n, where n = # of blocks in cache
    - Blocks mapped to the same cache block can overwrite each other
    - Address is divided into word, block & tag fields
      * E.g. consider cache = 128 blocks \* 16 words
      * Memory = 4K blocks \* 16 words (16-bit address)
      * 4 word bits – chooses a word in a block (out of 16)
      * 7 block bits – chooses block position in cache (out of 128)
      * 5 tag bits – identifies the block (out of 4K/128 = 32 possible blocks of main memory that could be mapped to this cache position)
      * During read, the highest 5 bits are compared to determine if the desired word is already in the cache at this location (specified by the 7 bits)
        + i.e. match = cache hit
  + Fully associative
    - More flexible; memory block can be placed in any cache block
    - Address = 12 tag bits + 4 word/offset bits
    - All tags are compared to in parallel determine if cache hit/miss – “associative search”
    - Cache blocks are replaced only if cache is full – requires replacement algorithm to select which block to replace
  + Set-associative
    - Cache is divided into sets of blocks
    - Memory block j is mapped to the set j mod s, where s = # of sets in cache
    - Memory block can be placed in any blocks in its set
    - Reduces possibilities for block replacement (from direct mapping)
    - Reduces size of associative search (from fully associative)
    - Fully associative = set-associative with 1 set
    - E.g. set size k = 2
      * 4 word bits
      * 6 block bits – chooses set position (out of 128/2 = 64)
      * 6 tag bits – identifies block (out of 4K/128 \* 2 = 64)
  + Stale data
    - Valid bit determines if valid data is loaded in cache block
    - V = 0 – no cache hit even if tag matches (will not load stale data)
    - V = 1 – set only when block is loaded into this cache location
* Replacement algorithms
  + Least recently used (LRU) – high complexity & high performance
    - Uses additional bits to monitor age of cache blocks
    - On hit
      * Reset referenced block age = 0
      * Any other ages that are < referenced block’s age before resetting, age++
    - On miss
      * All ages++
      * If cache not full, set new block age = 0
      * If cache full, replace block with highest age and reset age = 0
    - Leverages temporal locality
  + First-in first-out (FIFO) – medium complexity & low performance
  + Random – low complexity & medium performance
* Cache performance
  + Hit rate h = cache hits/all attempted accesses
  + Miss rate m = 1 – h = cache misses/all attempted accesses
  + C = access time for a block from cache
  + M = access time for a block from main memory (miss penalty)
  + Average access time for a block = tavg = hC + (1 – h)M
  + Performance improvement due to cache = tavg(no cache)/tavg(cache) = M/(hC + mM)
  + With two caches: tavg = h1C1 + m1(h2C2 + m2M)
  + Improving hit rate
    - Larger cache – but more costly
    - Larger blocks – reduces # of misses; but increases miss penalty time
* **Virtual memory**
  + Physical memory may not have capacity for the entire address space
  + Large programs don’t fit in main memory, must be partially stored in secondary memory
  + Virtual memory loads parts of program from disk to main memory
  + Just as cache bridges the speed gap between processor & main memory, virtual memory bridges the size & speed gaps between main & secondary memories
  + Programs reference virtual addresses
  + Memory management unit (MMU)
    - Translates virtual addresses → physical addresses
    - Data is divided into fix-sized pages (larger than cache blocks)
    - Virtual address = vpn bits + offsets
      * Virtual page number (vpn) (higher bits) = which page
      * Offset (lower bits) = which word within a page
    - Page table – stored in main memory
      * Contains location of each page in memory
      * Page frame – area in main memory that holds one page
      * Page table base register – base address of page table (i.e. address of first page)
    - Base register + vpn = location of desired page
      * If page is not in memory, load into memory
    - Page frame is returned and replaces the vpn bits
      * Physical address = page frame + offset
  + Translation look-aside buffer (TLB)
    - Improves performance of address translation
      * Translation normally requires 2 main memory accesses
    - Analogous to a cache for page table entries
      * i.e. if hit, retrieve page frame from TLB
      * If miss, retrieve page frame from page table & update TLB
  + If page is in main memory, proceed access
  + If page is not in memory, i.e. address cannot be translate, MMU raises a page fault using an interrupt
    - Invokes OS to load desired page from disk into memory (using DMA)
    - Pages are replaced using LRU
* **Secondary storage – magnetic hard disks**
  + Platters mounted on a common spindle
  + Platters are covered in thin magnet film & spin at constant rate
  + Read/write heads consist of magnetic yoke & coil
    - Write – apply current thru coil → changes direction of magnetization of film
    - Read – changes in magnetization of film induce voltage in head and are sensed → control circuitry interprets as 0 or 1
    - Phase encoding – change in direction of magnetization within one clock cycle determines binary value
  + Disk is divided into concentric tracks
    - Each track divided radially into sectors
    - The stack of corresponding tracks on multiple disks is a cylinder
    - Data on each track is preceded by a sector header – contains information about sector addresses on the track
  + Only one surface & one track can be read at a time