**Instruction Sets & Addressing Modes**

* **Instruction** – a command to CPU to perform a task
  + E.g. move data, perform arithmetic/logic operations, transfer control
  + Execution cycle of a computer
    - Fetch next instruction (store in IR) → decode instruction → fetch operands (if necessary) → perform operation → store result (if necessary)
  + Load – e.g. load R2, LOC – read contents from address LOC and store in register R2
  + Operations – e.g. ADD R4, R2, R3 – add contents of R2 & R3 and store result in R4
  + Store – e.g. store R4, LOC – copy contents of R4 into address LOC
  + One instruction may require many memory operations
    - E.g. fetching the instruction, fetching operands, storing the result
    - The memory device interface is a bottleneck – must be designed and used efficiently
* **RISC – reduced instruction set computers**
  + Each instruction implements a simple function; e.g. add, sub
  + Simpler for compilers to generate code
  + Instructions have fixed size (one word)
* **CISC – complex instruction set computers**
  + Each instruction can implement complex functionality; e.g. insertInQueue
  + Difficult for compilers to decipher
  + Instructions have variable sizes
* Instruction lengths depend on the address modes used
* Memory addressing
  + Instructions are not allowed to address individual bits in memory
  + **Address space** – the range of addresses
    - Depends on the width of the address signal; e.g. 24-bit address generates an address space of 224 locations
  + Memory systems are typically byte addressable
    - Each byte has unique address; adjacent words have addresses that differ by the # of bytes in a word
  + Word addressable system
    - Each word has unique address; adjacent words differ by 1
* Word & byte encoding
  + A 32-bit word can store a 32-bit int or 4 bytes (e.g. ASCII chars)
* Byte ordering
  + **Big-endian** – assigns lower byte addresses to more significant bytes
    - i.e. 0 address starts at MSB (left)
  + **Little-endian** – assigns higher byte addresses to less significant bytes
    - i.e. 0 address starts at LSB (right)
* **Aligned address** – begins at a byte address that is a multiple of the number of bytes in a word
  + E.g. a 4-byte word has aligned addresses at 0x0, 0x4, 0x8, 0xc, …
* **Instruction types**
* A computer must have instructions capable of performing 4 types of operations:
  + Data transfer b/t memory and processor registers
  + Arithmetic & logic operations
  + Program sequencing & control
  + I/O transfers (typically memory mapped; same as data transfer operations)
* **Instruction-set architecture (ISA)** – defines set of opcodes (machine language) and native command implemented by a processor