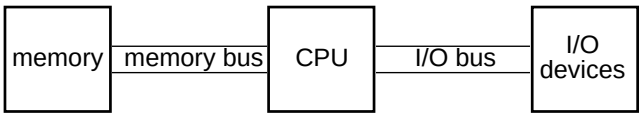
Organization of a Computer System

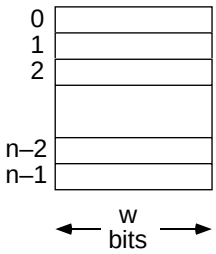
- A computer system is constructed from three types of components:
  - Processors (or *central processing units*, CPUs)
  - Memories
  - Input/Output (I/O) devices, including disks
- A typical computer system is organized as follows:



- Chapter 3 in Maccabe discusses memories in detail, but only briefly introduces processors and I/O devices
  - More on those topics in Chapters 9 & 12

Memory System Terminology

- A *memory system* is a collection of *memory elements* (*words*)
  - Each element is the same width (the *memory width*)
    - If each memory element is 8 bits (1 *byte*) wide, the memory is *byte addressable*
  - Each element is accessed through a unique *address*
    - Addresses are usually positive integers starting at 0
    - An address must be $\log_2 n$ bits wide to address n memory elements
- An n x w memory system:



- 1 k = 1024 = 2^{10}
  1M = 1024k = 2^{20}

Instruction and Data Bit Widths

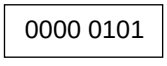
- Numbers for some common CPUs:

Intel 8086	varies	16
Intel 80286	varies	16
Intel 80386	varies	32
Intel Pentium	varies	32
Intel Pentium II	varies	32
Intel Merced ('99)	*varies*	*64*
Motorola 601	32	32
Motorola 604	32	32
HP PA-8000	64	64

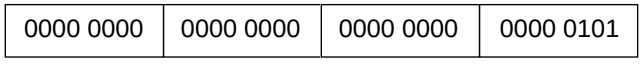
- Questions:
  - Why would the instruction width vary? Why would it stay constant? Which is better, and why?
  - Are there problems with different data widths on different machines?
  - Are bigger numbers generally better or worse? Why?

Bits & Bytes & Words

- A *bit* is one binary digit; a *byte* is 8 bits
- A *word* is the primary unit by which data is manipulated (even in a byte-addressable memory)
  - The word size varies from CPU to CPU, and is determined by the width of the datapath (ALUs, registers, etc.)
  - May be 8-bit, 16-bit, 32-bit, 64-bit, etc.
- How are these bits and bytes labeled?
  - How would I label the bits in this 8-bit value?

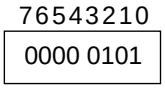


  - How would I label the bytes in this 4-byte word?

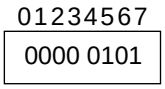


Big Endian vs. Little Endian

- Labeling right to left is called *little endian*



- Labeling left to right is called *big endian*



- Including labeling the bytes in a word, we have 4 possibilities:
  - Consistent little endian (Intel 80x86)
    bytes right to left, bits right to left
  - Consistent big endian (PDP11, TI 9900)
    bytes left to right, bits left to right
  - Inconsistent little endian (?)
    bytes right to left, bits left to right
  - Inconsistent big endian (Motorola 68000)
    bytes left to right, bits right to left

  Maccabe's book follows this convention…

Types of Memory Systems

- RAM — random access memory
  - Read/write memory would probably be a better name
  - Memory elements only store the data temporarily
  - SRAM — static RAM (speedy!)
    - Memory elements keep their value as long as the RAM has power
  - DRAM — dynamic RAM (dense!)
    - Memory elements lose their value unless the RAM is *refreshed* at regular intervals
- ROM — read-only memory
  - Memory elements store the data forever
  - ROM — programmed once, at factory
  - PROM — programmable ROM
  - EPROM — erasable programmable ROM

The Memory Hierarchy

- Hierarchy, cost, and performance:
  - CPU random access registers,
    64-1024 bytes, 1-10ns access,
    1 word block, 200 MB/s bandwidth
  - Cache random access SRAM,
    8-256 k bytes, 20ns access,
    16 word block, 8 MB/s bandwidth
  - Memory random access DRAM,
    8-64 M bytes, 60ns access,
    16 word block, 1 MB/s bandwidth
  - Disk direct access sectors,
    1- 10 G bytes, 10ms access,
    4 k word block, 1 MB/s bandwidth
  - Tape sequential access blocks,
    1 T bytes, 10ms-10s access,
    1 k word block, 1 MB/s bandwidth
- Main idea — keep frequently-accessed data where it can be accessed quickly