CSS 422 Hardware and Computer Organization

Introduction to computer architecture

Professor: Yang Peng

The slides are re-produced by the courtesy of
Dr. Arnie Berger and Dr. Wooyoung Kim

Topic

- Introduction to computer hardware and architecture
  - Chapter 1 by Berger (Available online)
  - Chapter 1 by Null (NOT available online)

- Number system
  - Chapter 1 by Berger (Available online)
  - Chapter 2.1 by Null (NOT available online)

Computer Architecture
Computer Organization

Computer Architecture

- Attributes of a system **visible to the programmer**
- Have a direct impact on the logical execution of a program

Architectural attributes include:

- Instruction set, number of bits used to represent various data types, I/O mechanisms, techniques for addressing memory, etc.

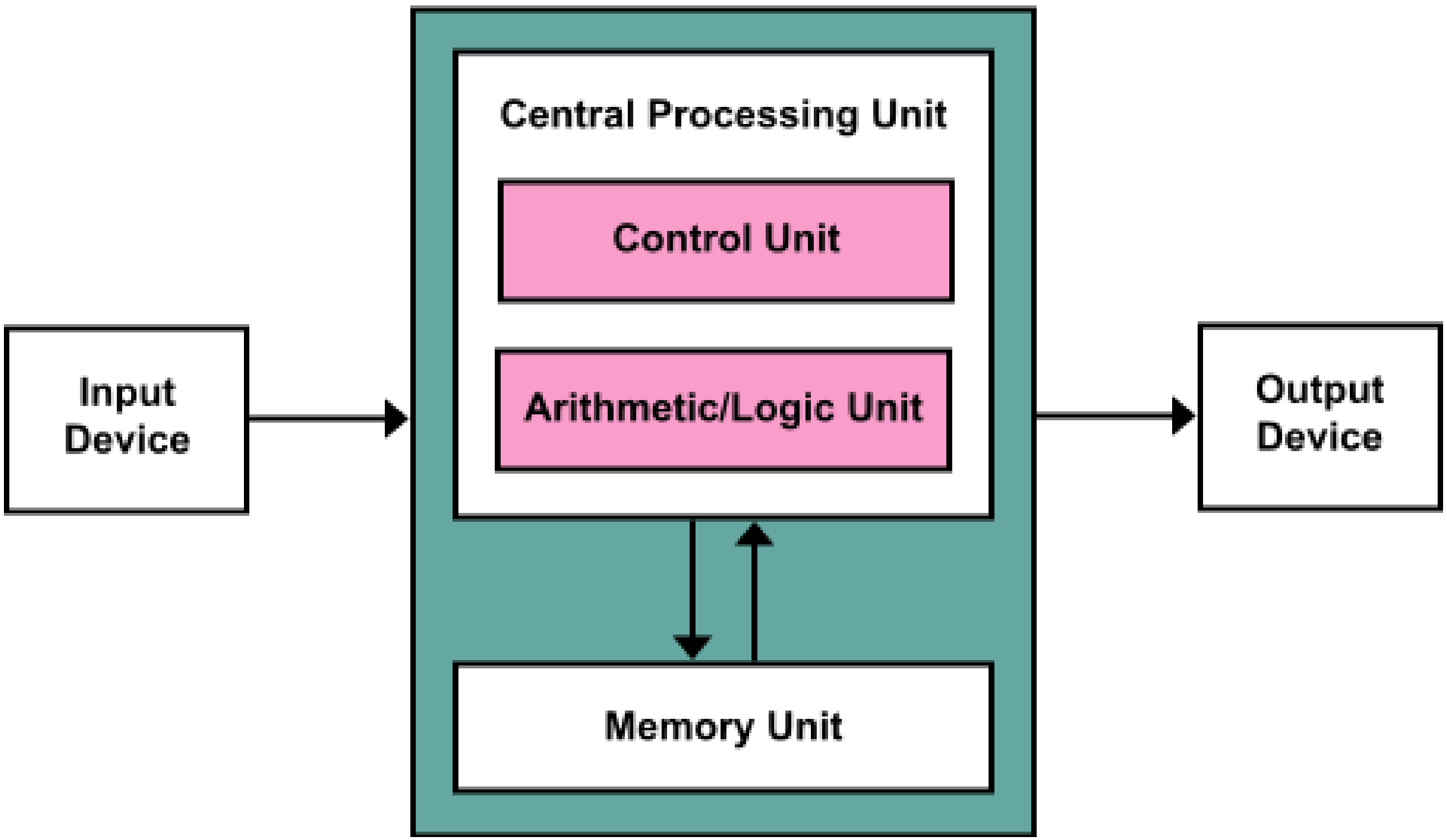
Computer Organization

- Hardware details **transparent to the programmer**
- The operational units and their interconnections that realize the architecture

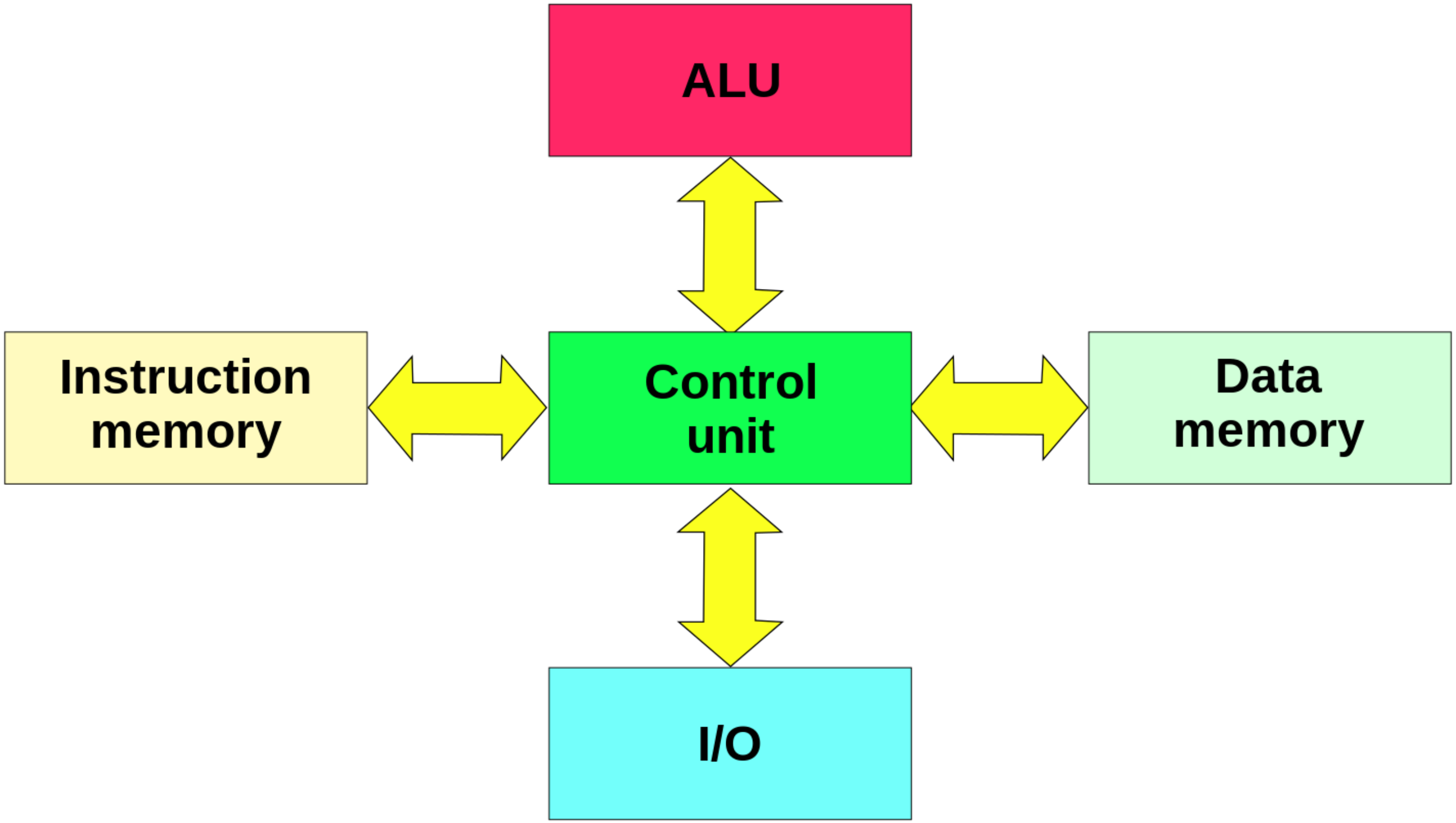
Organizational attributes include:

- Control signals, interfa between the computer peripherals, memory technology used, etc.

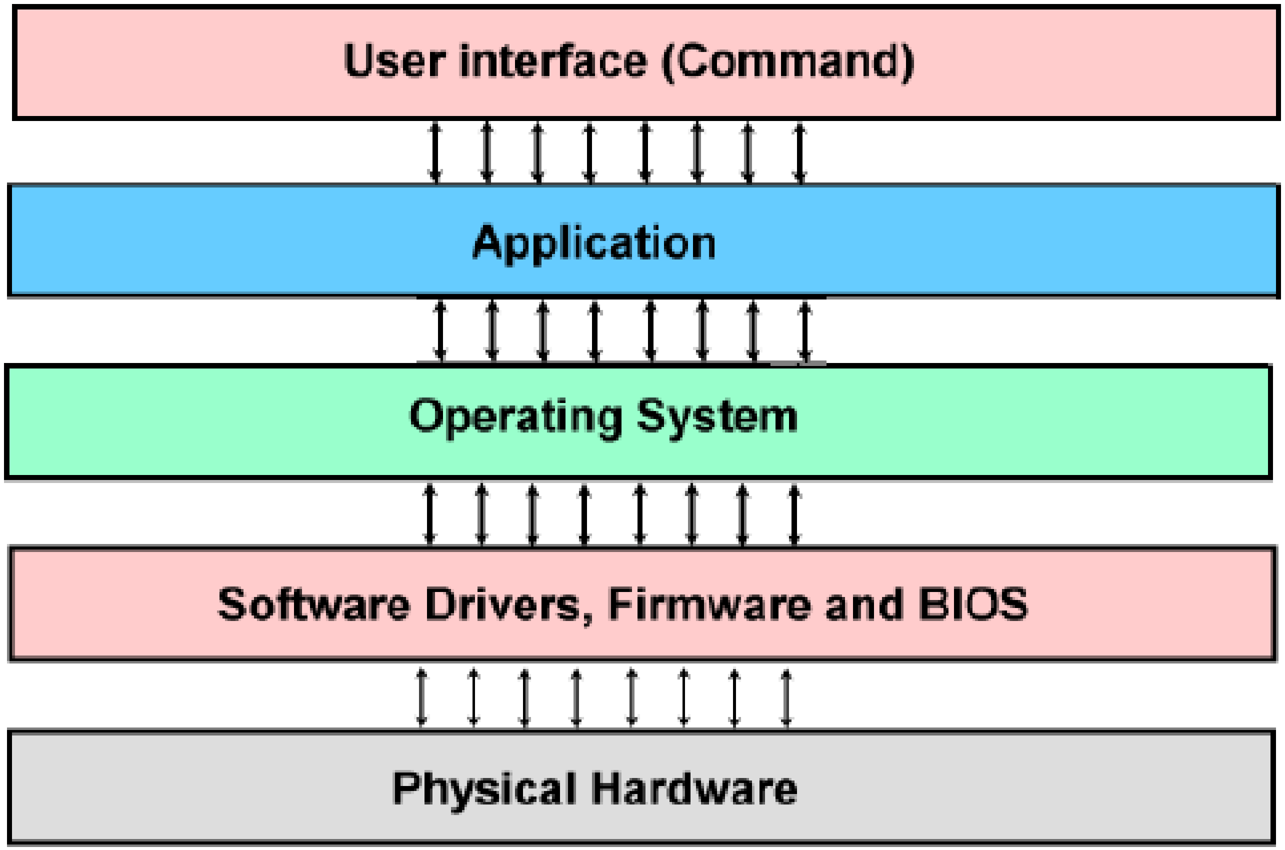
von Neumann Architecture (first / second generation)



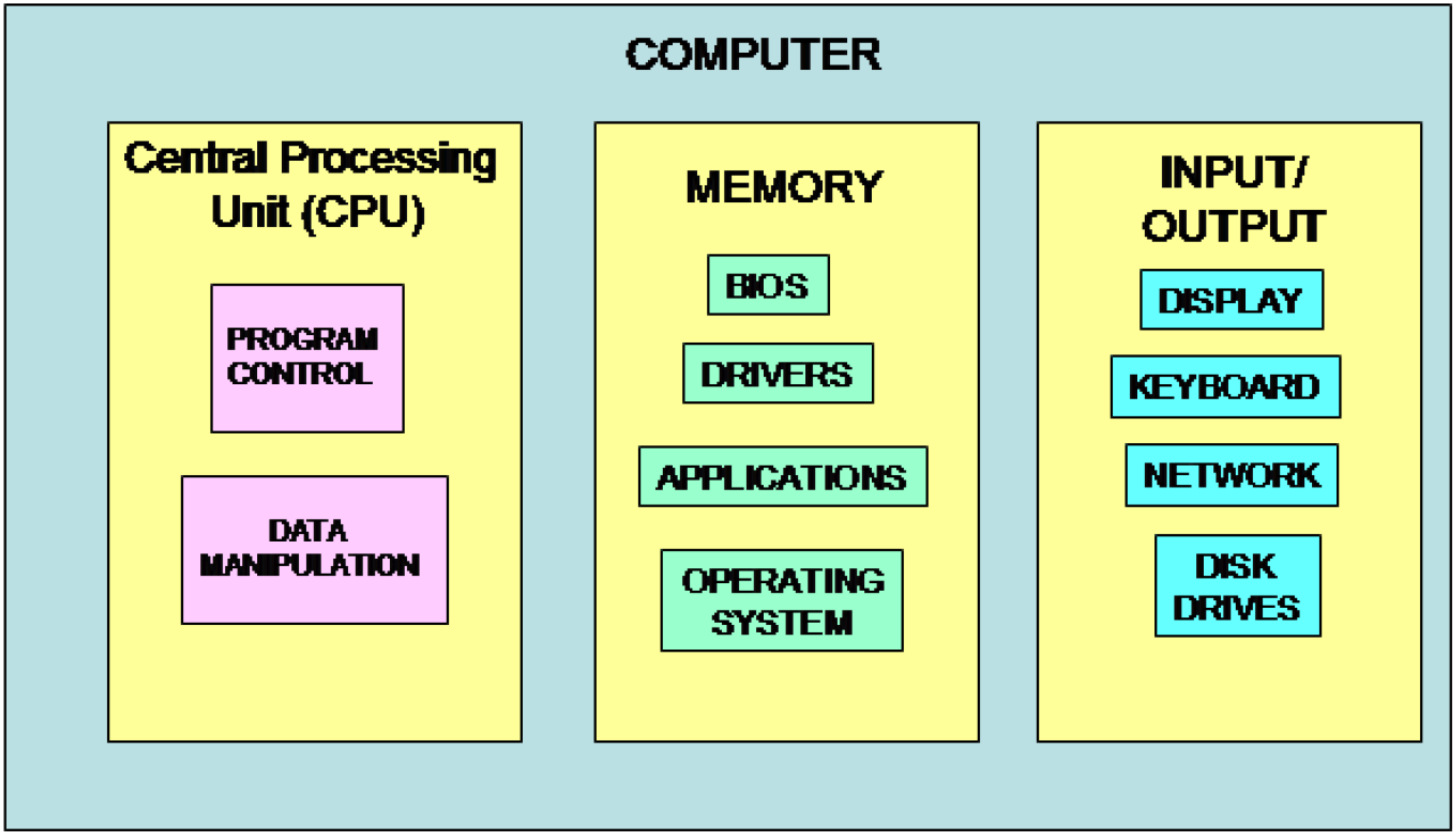
Harvard Architecture



Software Designer's View of Today's Computer



Hardware Designer's View of Today's Computer



The Digital Computer

- Machine to carry out instructions
  - A program
- Instructions are simple
  - Add numbers (*no subtraction actually*)
  - Check if a number is zero
  - Copy data between memory locations
- Primitive instructions in machine language

Data Representation in Computers

Question:

How can we quickly, cost effectively and accurately transmit, receive, store and manipulate numbers in a computer?

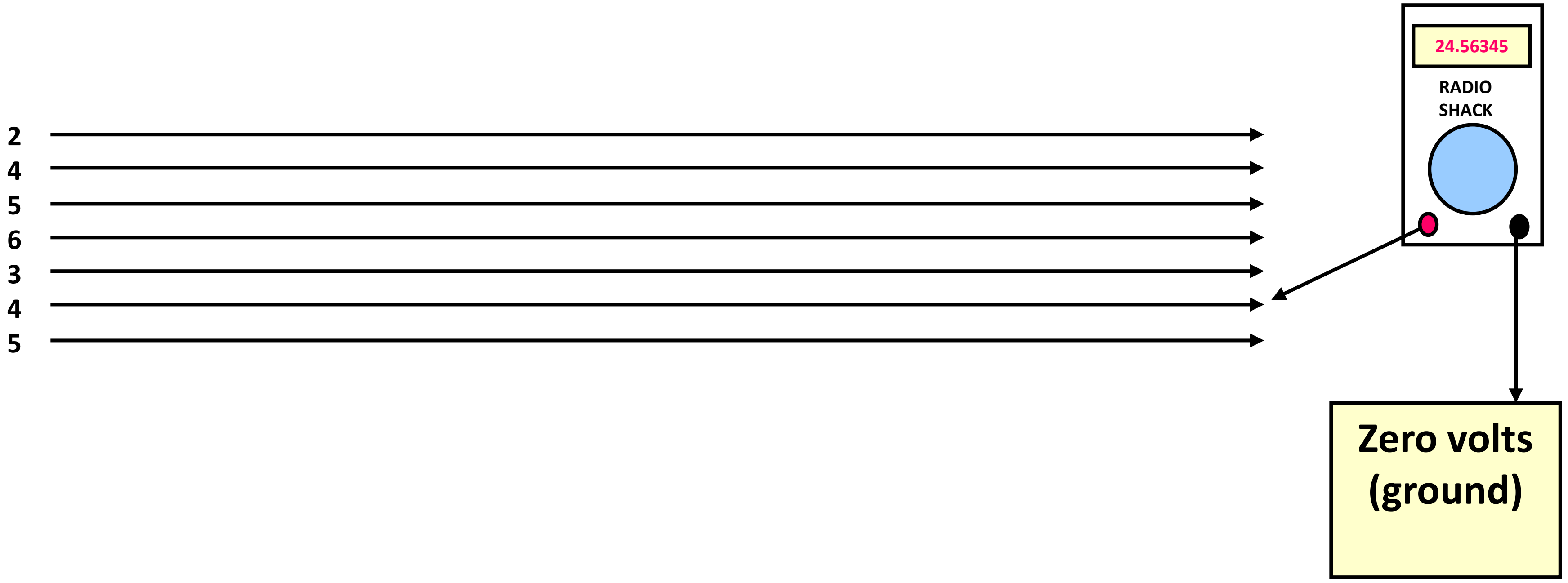
Possible Approach #1

- Represent the data value **as a voltage or current** along a **single** electrical conductor (signal trace) or wire

- **Problems:**
  - **Measuring large numbers is difficult, slow and expensive!**
  - **How do you represent +/- 32,673,102,093?**

Possible Approach #2

- Represent the data value **as a voltage or current** along **multiple** electrical conductors
- Let **each wire represent one decade of the number**
- Only need to **divide up the voltage on each wire into 10 steps**
  - 0 V to 9 volts



Comments on Approach #2

- **This is better than the first approach**
  - Only need to worry about 10 discrete signal levels
- However, **modern electronics are not sufficiently fast enough to make this a viable solution, and**
- It can have **considerable "slop" between values** before it causes problems
  - What if the second wire gives 4.2 V, or 4.5 V?

- **Better approach?**
- Hint: Electronics are ***really good*** at switching things on and off very fast
  - Modern transistors (electronic switches and amplifiers) can switch a signal on or off in 10's of picoseconds (trillionths of a second)

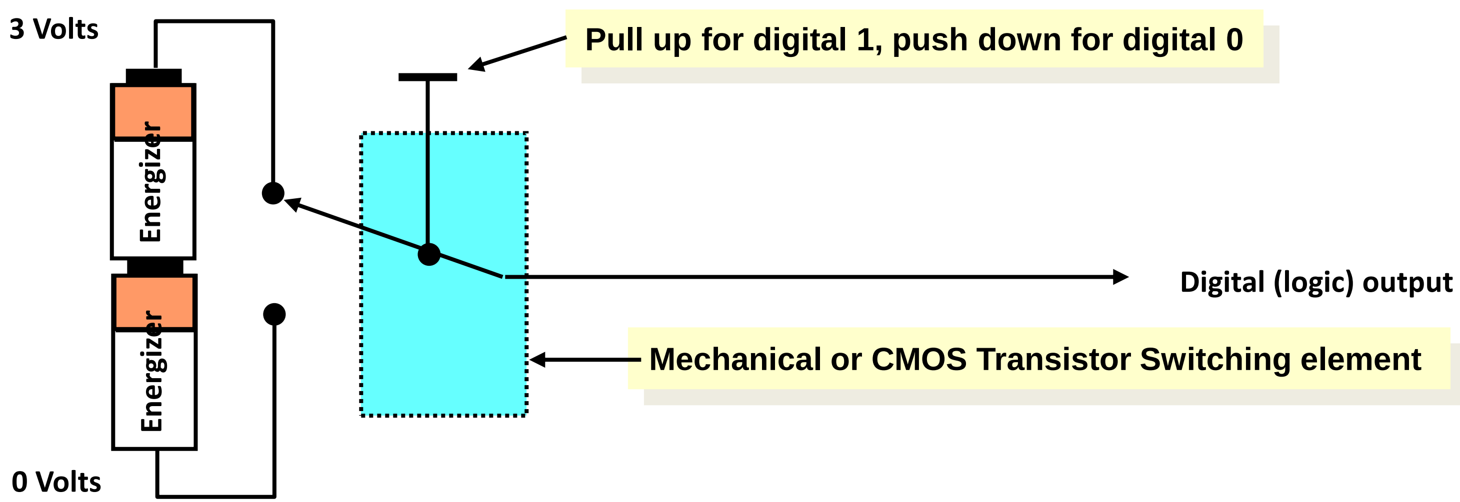
Possible Approach #3

- Represent the data value **as a voltage or current** along **multiple, parallel,** electrical conductors
- Let **each wire represent one power of 2 of the number (2^0 ~ 2^N)**
- Only need to **divide the voltage on each wire into 2 possible steps**
  - 0 V "no volts" or "some volts" greater than zero (on or off)
- **Can have lots of "slop" between values**

2^0	→	on	→	1
2^1	→	off	→	0
2^2	→	on	→	1
2^3	→	off	→	0
2^4	→	off	→	0
2^5	→	on	→	1
2^6	→	on	→	1
2^7	→	on	→	1
2^8	→	off	→	0
2^9	→	off	→	0
2^{10}	→	off	→	0
2^{11}	→	on	→	1
2^{12}	→	on	→	1
2^{13}	→	on	→	1
2^{14}	→	on	→	1
2^{15}	→	off	→	0

Comments on Approach #3

- Using **transistors** as electronic, high-speed on/off switches is a very efficient way to accurately **send signals at high speed**
- Each signal on a wire is either “on” or “off”
  - An “on” signal means that **some voltage is present** (~3 volts or greater)
  - An “off” signal means that the **voltage is mostly absent** (< 0.4 volts)
- Each wire or signal trace represents either the number 0 (no voltage) or the number 1 (some voltage)
- Imagine that each electronic device is like a mechanical switch that can **quickly switch the voltage on a wire** between 0 volts and 3 volts



Binary Number System

- Since we are switching between two voltage levels, our number system has only 2 digits, 1 or 0: a ***binary*** number system
- The arithmetic and logical operations on a set of binary number is called ***Boolean Algebra***
- From approach #3, what is the number:
  - off on on on on off off off on on on off off on off on ?
  - 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 ?
  - Answer: 30949

- How did I get this?
  - $0 \times 2^{15} = 0$ $\quad 1 \times 2^{14} = 16{,}384$ $\quad 1 \times 2^{13} = 8{,}192$
  - $1 \times 2^{12} = 4{,}096$ $\quad 1 \times 2^{11} = 2{,}048$ $\quad 0 \times 2^{10} = 0$
  - $0 \times 2^{9} = 0$ $\quad 0 \times 2^{8} = 0$ $\quad 1 \times 2^{7} = 128$
  - $1 \times 2^{6} = 64$ $\quad 1 \times 2^{5} = 32$ $\quad 0 \times 2^{4} = 0$
  - $0 \times 2^{3} = 0$ $\quad 1 \times 2^{2} = 4$ $\quad 0 \times 2^{1} = 0$
  - $1 \times 2^{0} = 1$

16,384 + 8,192 + 4,096 + 2,048 + 128 + 64 +32 + 4 + 1 = 30949

Number Systems

- We count in the decimal system because we have 10 fingers
  - There is nothing unique about counting in decimal
  - We would count in octal (base 8) if we had 8 fingers
- The ***BASE (Radix)*** of a number system is just the number of distinct digits in that system
  - Computer systems are naturally binary (base 2)
  - Common number systems used with computational devices:
  - Base 2: 0,1 : Binary
  - Base 8: 0,1,2,3,4,5,6,7 : Octal
  - Base 10: 0,1,2,3,4,5,6,7,8,9 : Decimal
  - Base 16: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F : Hexadecimal

Binary, Octal and Hexadecimal

- We use the binary number system to represent numbers and logical operations in a computer
- Reading and writing binary numbers is tedious and error-prone because the numbers can be very long
- Octal and hexadecimal are ways to simplify the representation of numbers to make them easier to understand and manipulate
- For example:
  - 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 = 30,949 in decimal
  - 0 111 100 011 100 101 = 074345 in octal
  - 0111 1000 1110 0101 = 78E5 in hexadecimal
- Notice how hexadecimal is the most compact way to represent the number
- Notice how the binary numbers are grouped together in octal (by 3) and hexadecimal (by 4)
- As you'll see, we convert between binary, octal and hexadecimal be **changing how the binary numbers are grouped** together

Converting from Decimal to Other Bases

- Algorithm:
  - Example: Convert 73,503 to base 17

	quotient		remainder
17	73,503		
17	4,323	………	12 (C)
17	254	………	5
17	14	………	16 (G)
17	0	………	14 (E)

Therefore, the answer is $EG5C_{17}$.

Why this algorithm?

EG5C = { E * 17^2 + G * 17 + 5}*17 + C

= {{ E * 17 + G} * 17 +5} * 17 + C

Converting from Decimal to Other Bases

- Algorithm:
  - Example: Convert $EG5C_{17}$ To Decimal

EG5C = E * 17^3 + G * 17^2 + 5 * 17^1 + C *17^0

= {{ E * 17 + G} * 17 +5} * 17 + C

.

.

.

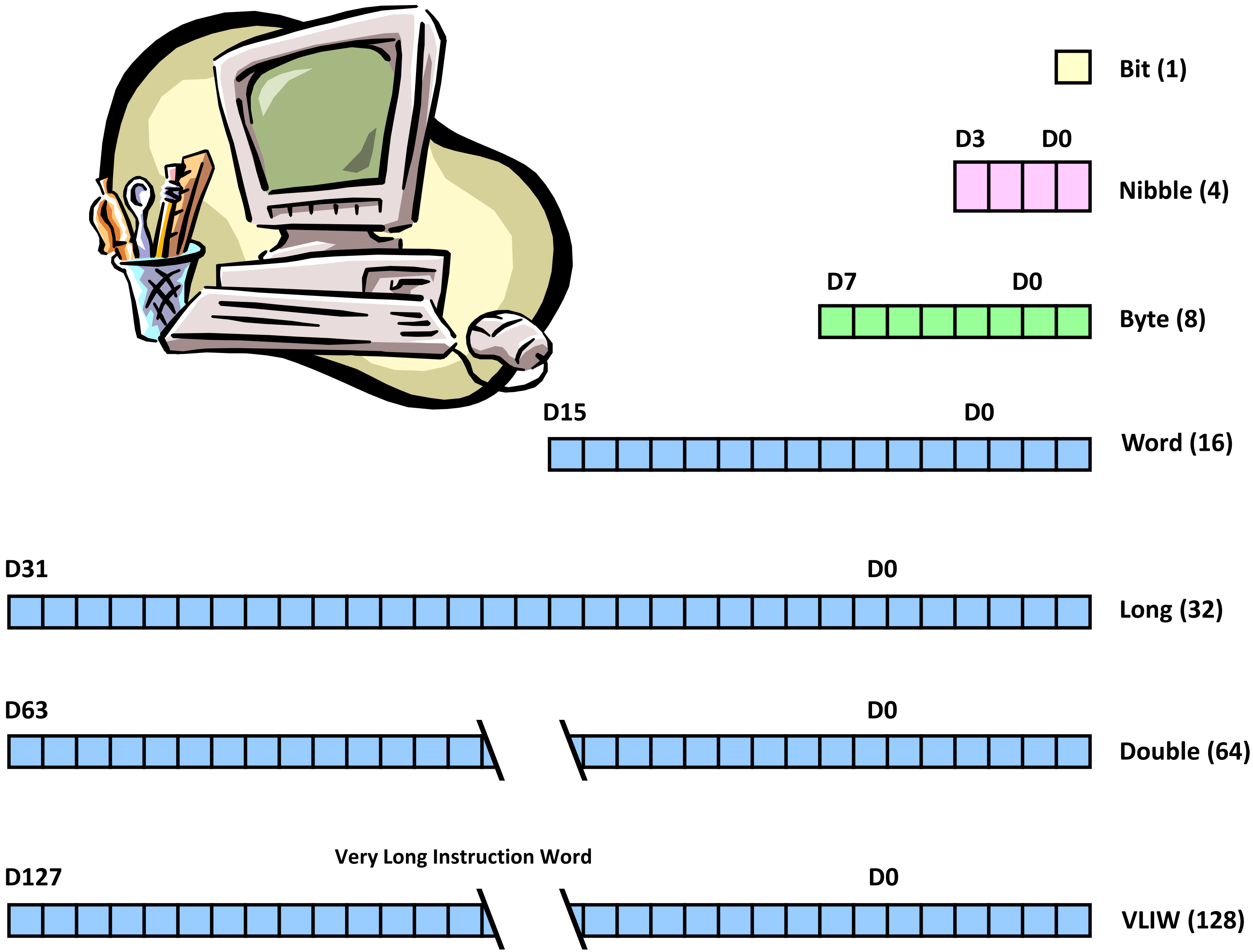
= 73,503

Let’s work on a 16-bit number

- Binary: 0101111111010111
- Octal: 0 101 111 111 010 111 = 057727 (group by threes)
- Hex: 0101 1111 1101 0111 = 5FD7 (group by fours)
- Decimal: ???

- Exercise: Convert $5DE37A05_{16}$ to Octal

Bits, Bytes, Nibbles, Words, etc.



Engineering Notation

- In order to represent very large or very small numbers, we usually resort to ***scientific notation:***
    - For example: *Avogadro's Number = 6.022 x 10* ^{23}
        - Mantissa = 6.022, exponent = 23
- It is common in engineering to use a shorthand version of scientific notation
- Replacement values

TERA = 10^{12} (T)	**PICO = 10^{-12} (p)**
GIGA = 10^{9} (G)	**NANO = 10^{-9} (n)**
MEGA = 10^{6} (M)	**MICRO = 10^{-6} (μ)**
KILO = 10^{3} (K)	**MILLI = 10^{-3} (m)**

Computers and Numbers

- In the **digital world**
  - 1K means 1,024, or 2^{10}
  - 1M means 1,048576, or 2^{20}
  - 1G means 1,073,741,824, or 2^{30}
- Example
  - 512 megabytes of memory really means 512 x (2^{20}) bytes, or 2^{29} bytes of memory
- In general, ***anything to do with the size in bytes*** uses ***computer-speak*** K,M,G
  - Anything else, such as clock speed or time, use standard units