Week 1 – Number Systems

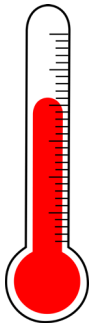
Section 1.1-1.3

Analog vs Digital

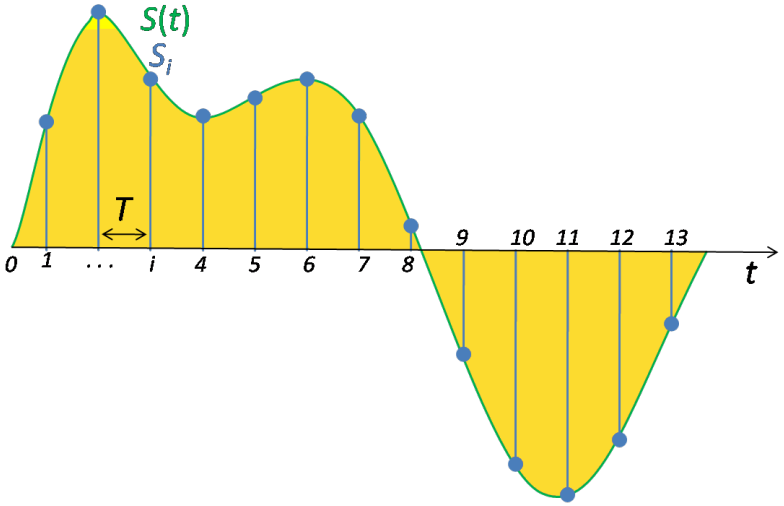
Section 1.1 – 1.2

Analog vs Digital

- Analog: system can take on a “continuous” set of values



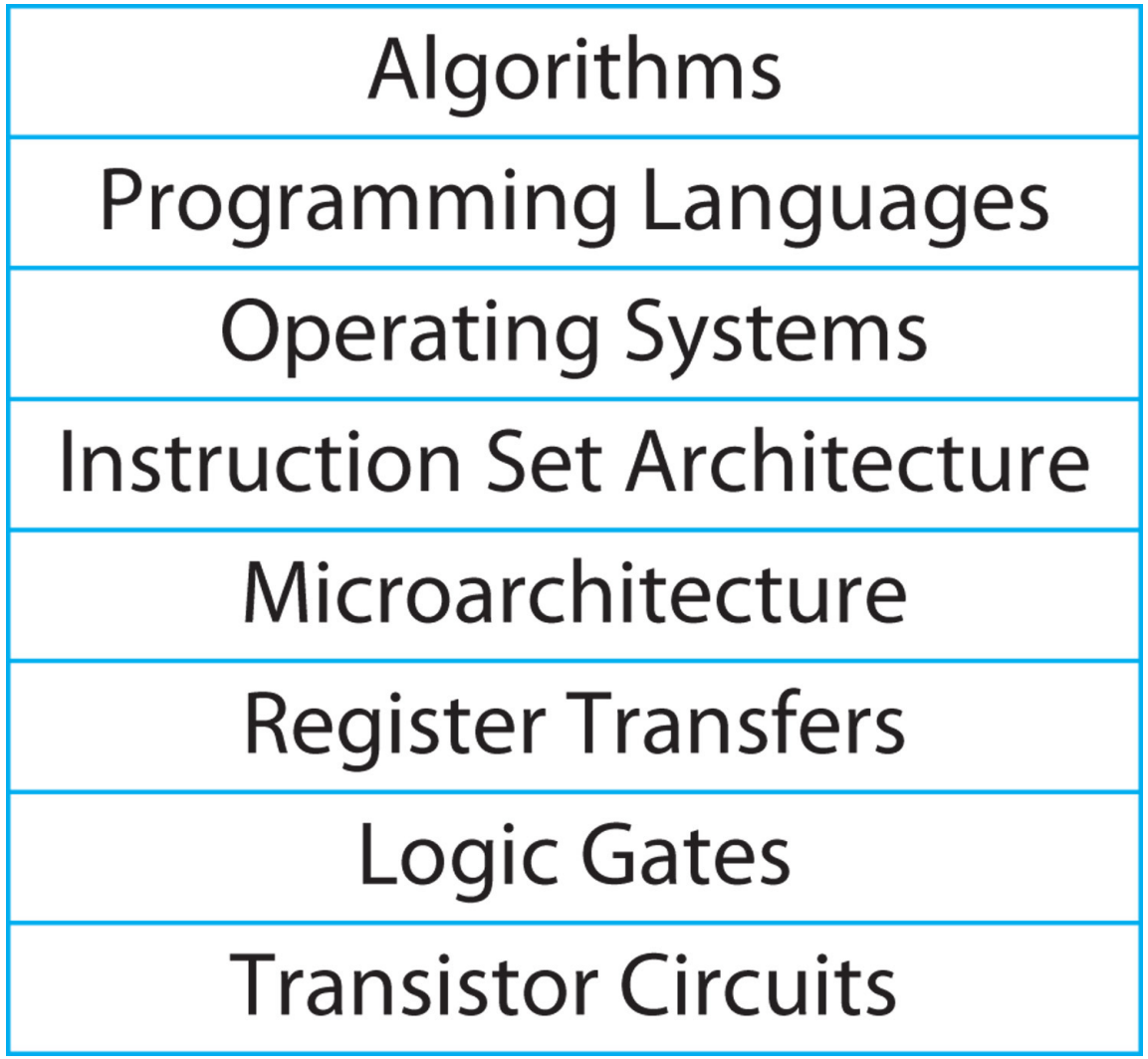
- Digital: system can take on fixed or “discrete” set of values



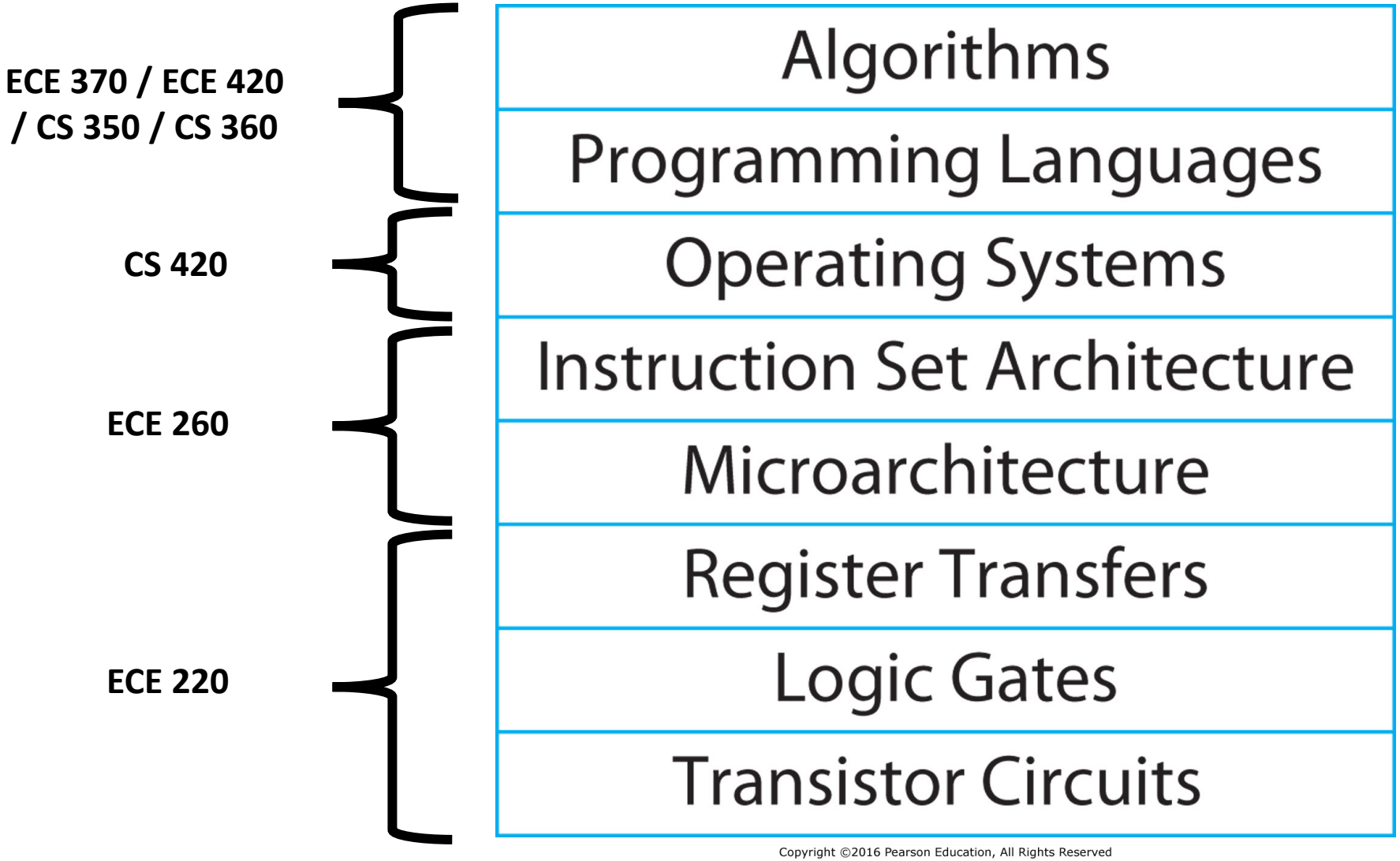
Digital Logic

- Computer systems are built on “Boolean” or digital logic
- All values in the system are either ‘1’ or ‘0’
- Because the system uses Boolean logic, certain properties can be established for each “layer” in the computer.

Layers of Abstraction in Computing



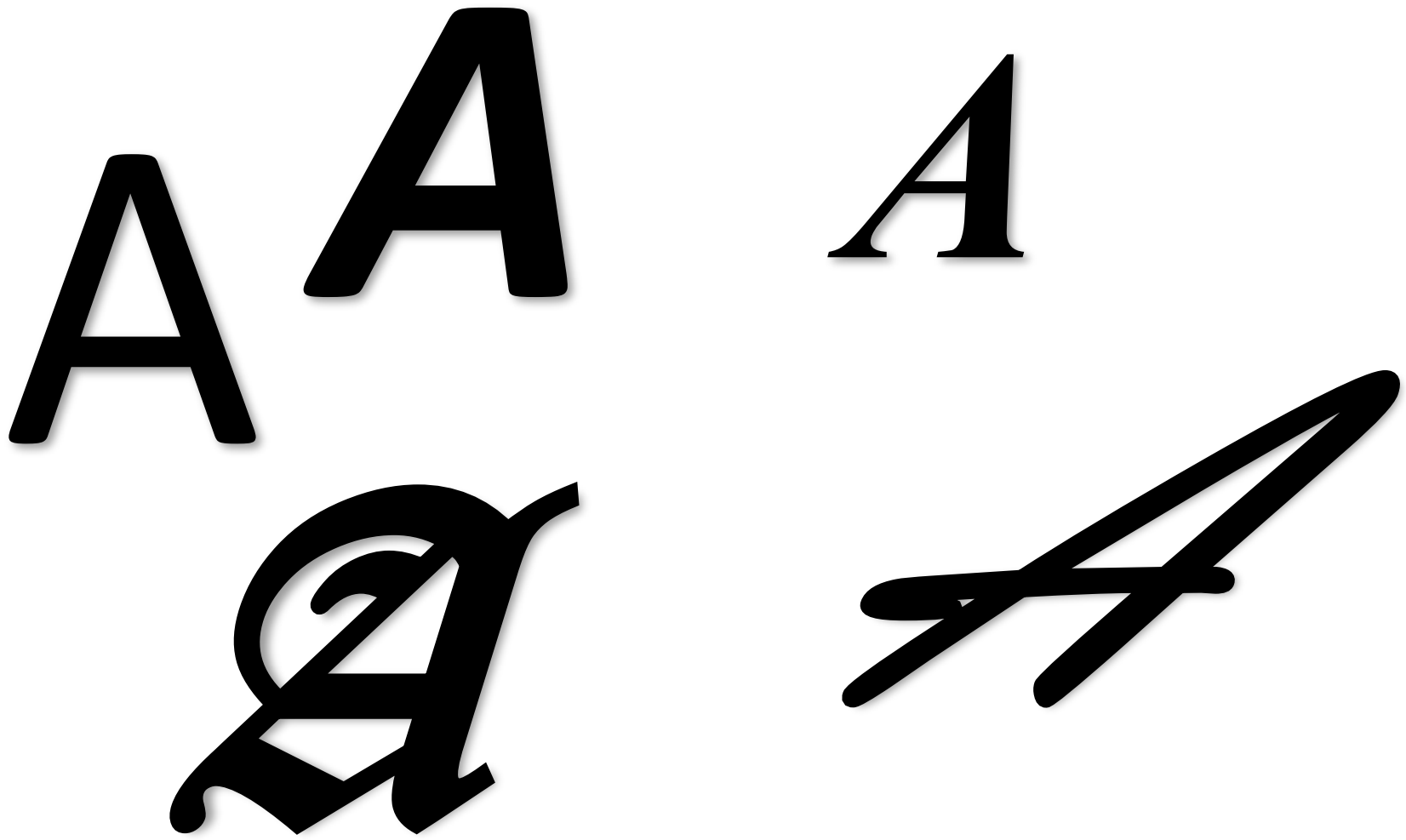
Layers of Abstraction in Computing



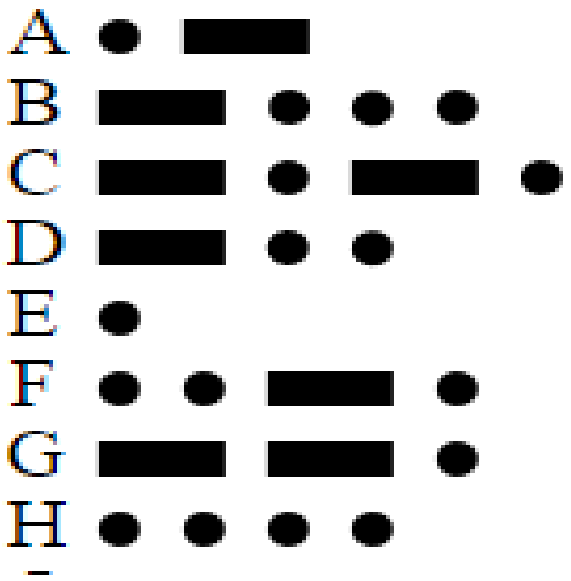
All information in a computing system must have some representation

Each representation can “appear” difference but represents the same underlying information

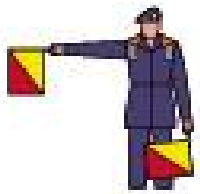
A



A



A (1)



B (2)

C (3)

D (4)

Binary and Encodings

- All information in a computer is stored as digital 1’s and 0’s
- Various encoding schemes and number systems to represent information
-

ASCII Encoding Table

97	61	141	&#97;	a
98	62	142	&#98;	b
99	63	143	&#99;	c
100	64	144	&#100;	d
101	65	145	&#101;	e
102	66	146	&#102;	f
103	67	147	&#103;	g
104	68	150	&#104;	h
105	69	151	&#105;	i
106	6A	152	&#106;	j
107	6B	153	&#107;	k
108	6C	154	&#108;	l

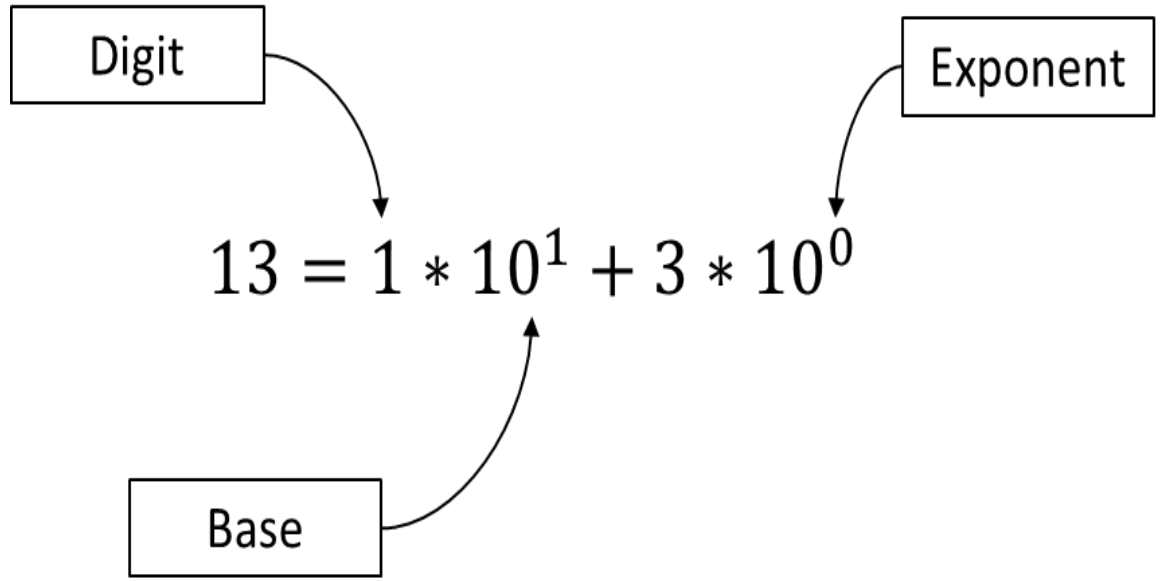
Number Systems

Section 1.3

Bases and Number Systems

- All number systems have a *base*, *digit*, and *exponent* representation

$$digit * \ base^{exp}$$



Expanded Representation

- All numbers are formed by summing each digit, base, and exponent

$$10524 = 1 * 10^4 + 0 * 10^3 + 5 * 10^2 + 2 * 10^1 + 4 * 10^0$$

- For any number, you should be able to identify the digit, position, and base.
  - Example: 47,586

Num.	4	7	5	8	6
Pos.	10^4	10^3	10^2	10^1	10^0

Example

- Write out the following values in an expanded form some each digit, base, and exponent summed together.
  - Hint: don’t forget the 0’s

- 54,087

- 1001

Example

- Write out the following values in an expanded form some each digit, base, and exponent summed together.
  - Hint: don’t forget the 0’s

- 54,087=$5 * 10^4 + 4 * 10^3 + 0 * 10^2 + 8 * 10^1 + 7 * 10^0$

- 1001=$1 * 10^3 + 0 * 10^2 + 0 * 10^1 + 1 * 10^0$

Base-10 and Other Systems

- Our decimal system is a Base-10 system. However other bases are possible. The Babylonians used base 60.

- $45_{10} = 63_7 = 6 * 7^1 + 3 * 7^0$

- $525_{10} = 1350_7 = 1 * 7^3 + 3 * 7^2 + 5 * 7^1 + 0 * 7^0$

Binary and Computing Systems

- Computer systems are built on a binary system of 1’ and 0’ values. Foundation of digital/Boolean logic.
- Only allowable digits in binary are {0,1}
- Binary is a Base-2 system. Later we will focus on Hexadecimal (base-16) and octal (base-8).
- Should memorize all base-2 powers up to 2^12
  - 2^0 = 0, 2^1 = 2, 2^2=4…etc.

Binary to Decimal Conversion

- Goal is to find a base-2 representation of a base-10 number.
  - $135_{10} = ?* 2^3 + ?* 2^2 + ?* 2^1 + ?* 2^0$
- The sequence of *bits* {0,1} is our binary representation of that decimal number.
  - $13_{10} = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 1101_2$
- Need sufficient number of “bits” to represent the decimal number.
  - N-bits allows representation of 2^N-1decimal value
  - 3-bits => 7; 4-bits => 15…etc.

Decimal to Binary: Subtraction Method

- Various methods from converting to decimal to binary

- Subtraction method:
  1. Select a number of bits (N) for your value. Keep running tally of your number.
  2. Select the highest binary position that is just less than the number
  3. If number > 2^position, then subtract 2^position from the number. Mark a 1. Otherwise Mark a 0
  4. Repeat (2) until running tally is 0.

Example Binary to Decimal Conversion: Subtraction Method

~~21~~
~~5~~
~~1~~

Bits:	1	0	1	0	1
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

~~13~~
~~5~~
~~1~~

Bits:	0	1	1	0	1
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

~~27~~
~~11~~
~~3~~
~~1~~

Bits:	1	1	0	1	1
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

Decimal to Binary: Addition Method

- Another method called the “addition” method is possible. Most prefer “subtraction” but use what you want.

- Addition method:
  1. Select a number of bits (N) for your value. Keep running tally called *sum*.
  2. Select the highest binary position that is just less than the number
  3. If 2^pos > *sum*, mark 0. Else, mark 1 and add 2^pos to *sum*
  4. Repeat (2) until sum is desired value.

Example Binary to Decimal Conversion: Addition Method

30

~~16~~
~~24~~
~~28~~
~~30~~

Bits:	1	1	1	1	0
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

17

~~16~~
~~17~~

Bits:	1	0	0	0	1
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

6

~~4~~
~~6~~

Bits:	0	0	1	1	0
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

Practice

- Convert the following decimal values into binary:

- 25_{10}
- 11_{10}
- 718_{10}

Practice

- Convert the following decimal values into binary:

- $25_{10} = 11001_2$
- $11_{10} = 1011_2$
- $718_{10} = 10\ 1100\ 1110_2$

MSB and LSB

- In a binary representation there are two “special” bits called the *most significant bit (MSB)* and the *least significant bit* (LSB).

- Not there either is more/less important, just that this bit carries the largest/smallest value

$$718_{10} = 10\ 1100\ 1110_2$$

Binary to Decimal Conversion

- Reverse process from Decimal to Binary. Sum the values of each position.

23 = 16 + 4 + 2 +1

Bits:	1	0	1	1	1
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

18 = 16 + 2

Bits:	1	0	0	1	0
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

4 = 4

Bits:	0	0	1	0	0
Pos.	4	3	2	1	0
Value.	16	8	4	2	1

Example Binary to Decimal

- With a friend, convert the following values from binary into decimal

- $01\ 0101_2$
- $10\ 1111_2$
- $1\ 0111\ 0111_2$

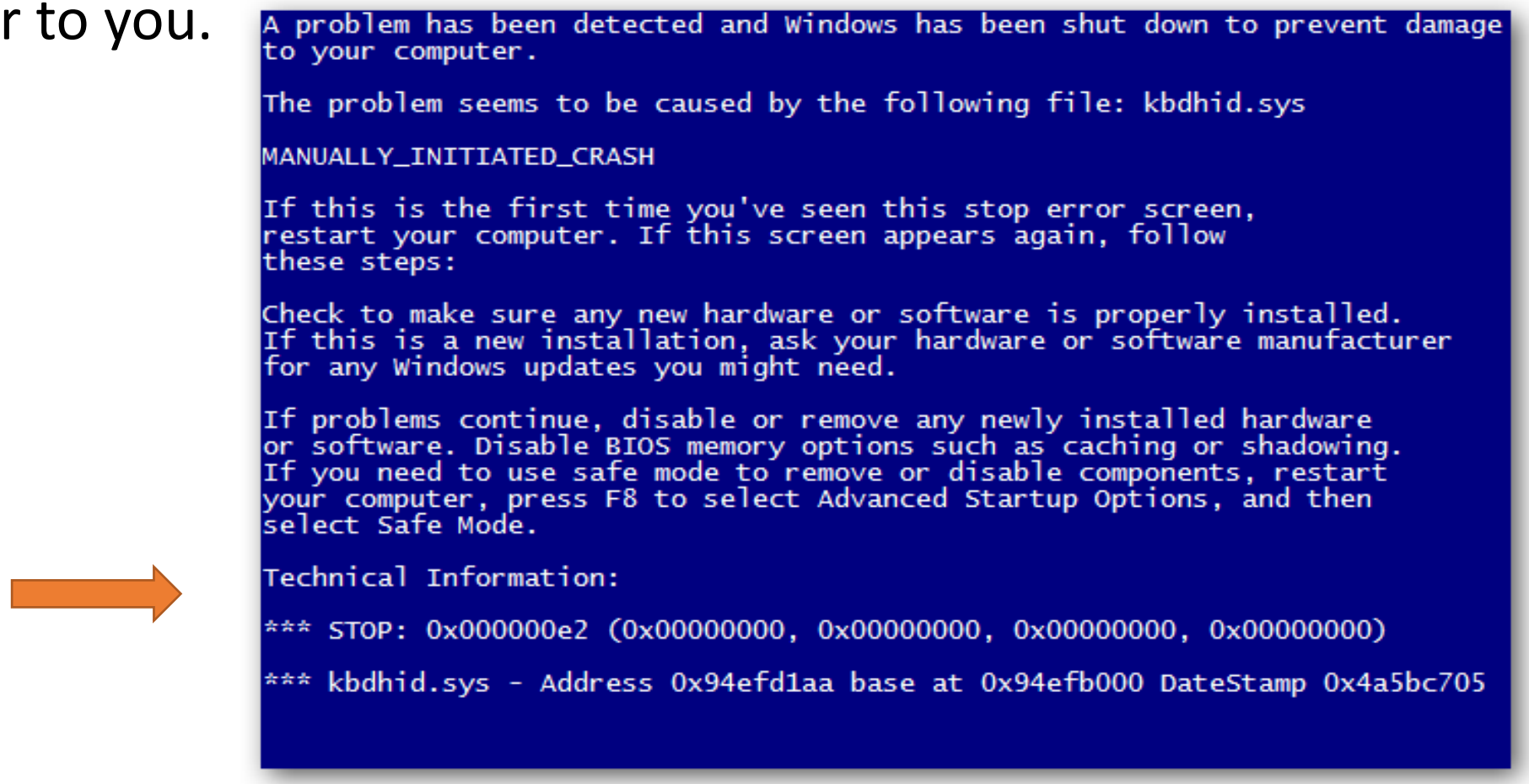
Example Binary to Decimal

- With a friend, convert the following values from binary into decimal

- $01\ 0101_2 = 16 + 4 + 1 = 21_{10}$
- $10\ 1111_2 = 32+8+4+2+1=47_{10}$
- $1\ 0111\ 0111_2 = 256 + 64 +32 + 16 + 4 + 2 + 1 = 375_{10}$

Base-8 and Base-16 Systems

- Other representations of binary systems are possible. Some may be familiar to you.



Base-16 Hexadecimal

- Takes binary representation of a number and groups into four bits
- Re-encodes each four bits as a number or letter [0-9,A-F].
- Allows for easier expression of large binary values

Hexadecimal / Binary Table

Decimal	Hex	Binary
0	0x0	0000
1	0x1	0001
2	0x2	0010
3	0x3	0011
4	0x4	0100
5	0x5	0101
6	0x6	0110
7	0x7	0111

Decimal	Hex	Binary
8	0x8	1000
9	0x9	1001
10	0xA	1010
11	0xB	1011
12	0xC	1100
13	0xD	1101
14	0xE	1110
15	0xF	1111

Binary to Hexadecimal Example

- To “convert” a binary value to hex, simple group the bits and replace with hex digit

- 1101 1100 1111 0001 => 0xDCF1
- 0001 1010 0110 0011 => 0x1A63

Hexadecimal to Binary Example

- Opposite process, however leave space for 0’s!!

- 0x24AF3 => 0010 0100 1010 1111 0011

- 0x24AF3 ≠ 0010 0100 1010 1111 11

Base – 8 Octal

- Another representation you may encounter is *Octal*. Similar to hex but groups bits into three's.

Decimal	Hex	Binary	Octal
0	0x0	0000	0
1	0x1	0001	1
2	0x2	0010	2
3	0x3	0011	3
4	0x4	0100	4
5	0x5	0101	5
6	0x6	0110	6
7	0x7	0111	7

Decimal	Hex	Binary	Octal
8	0x8	1000	10
9	0x9	1001	11
10	0xA	1010	12
11	0xB	1011	13
12	0xC	1100	14
13	0xD	1101	15
14	0xE	1110	16
15	0xF	1111	17

- Never encountered personally, but exists somewhere

Definitions and Rules

- Each digit of a binary value is a *bit*. One *byte* is 8-bits.
- Length/size of binary values determined by *bit-width*. 3-bit value, 7-bit value…etc.
  - 00010 is five bits; 010 is three bits
- An n-bit value can represent $[0 : 2^n - 1]$ values
- Computing systems have various *word sizes* based upon the largest register bit width. (Modern systems are 64-bit words).