Schedule…

Date	Day	Class No.	Title	Chapters	HW Due date	Lab Due date	Exam
5 Nov	Wed	19	Binary Numbers	13.1 – 13.2			
6 Nov	Thu						
7 Nov	Fri		Recitation		HW 8		
8 Nov	Sat						
9 Nov	Sun						
10 Nov	Mon	20	Exam Review			LAB 7	EXAM 2
11 Nov	Tue						
12 Nov	Wed	21	Boolean Algebra	13.2 – 13.3			

Numbered

Moses 1:33,35,37

33 And worlds without **number** have I created; and I also created them for mine own purpose; and by the Son I created them, which is mine Only Begotten.

37 And the Lord God spake unto Moses, saying: The heavens, they are many, and they cannot be **numbered** unto man; but they are **numbered** unto me, for they are mine

35 …all things are **numbered** unto me, for they are mine and I know them.

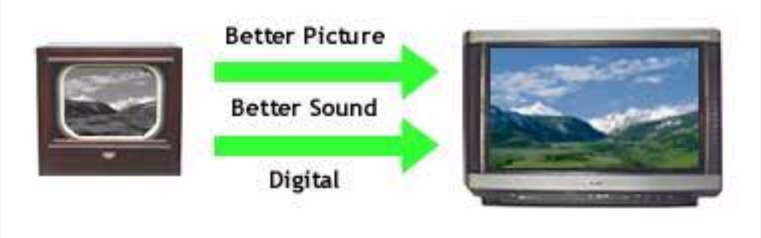
3 Nephi 18:31

31 …behold I know my sheep, and they are **numbered**.

Lecture 19 – Binary Numbers

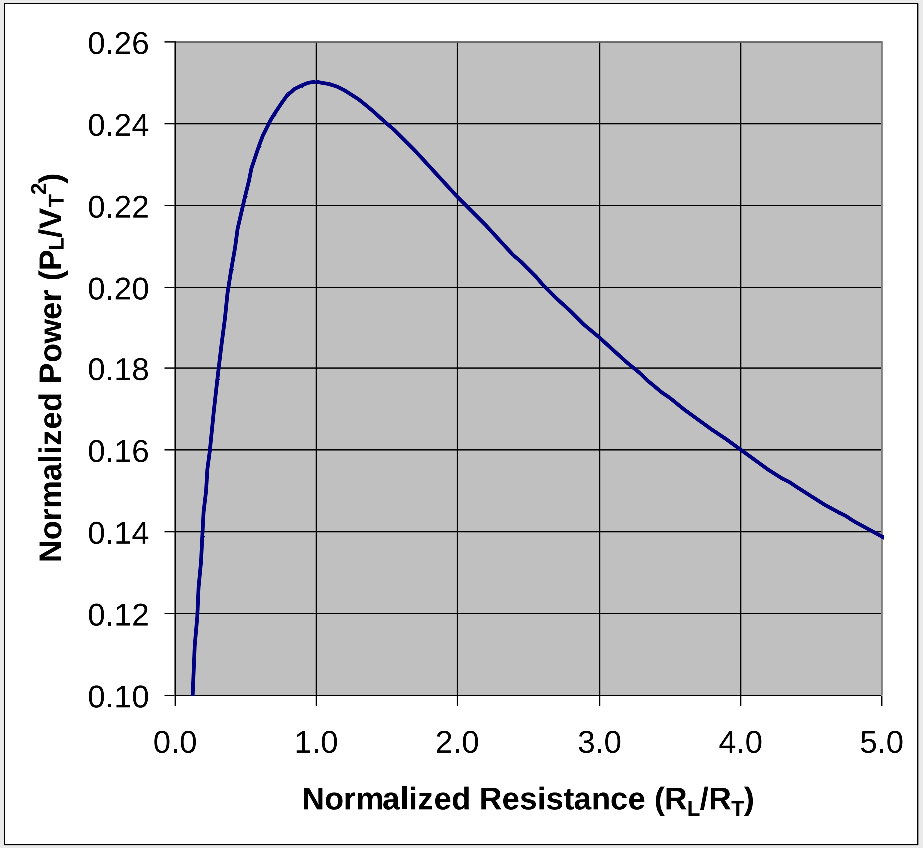
Digital vs. Analog

- Wristwatches (numbers vs. hands)
- LP's vs. CD's
- Rotary phone vs. Cell phone
- NTSC vs. HDTV
- Slide rule vs. calculator
- 737's vs. 777's

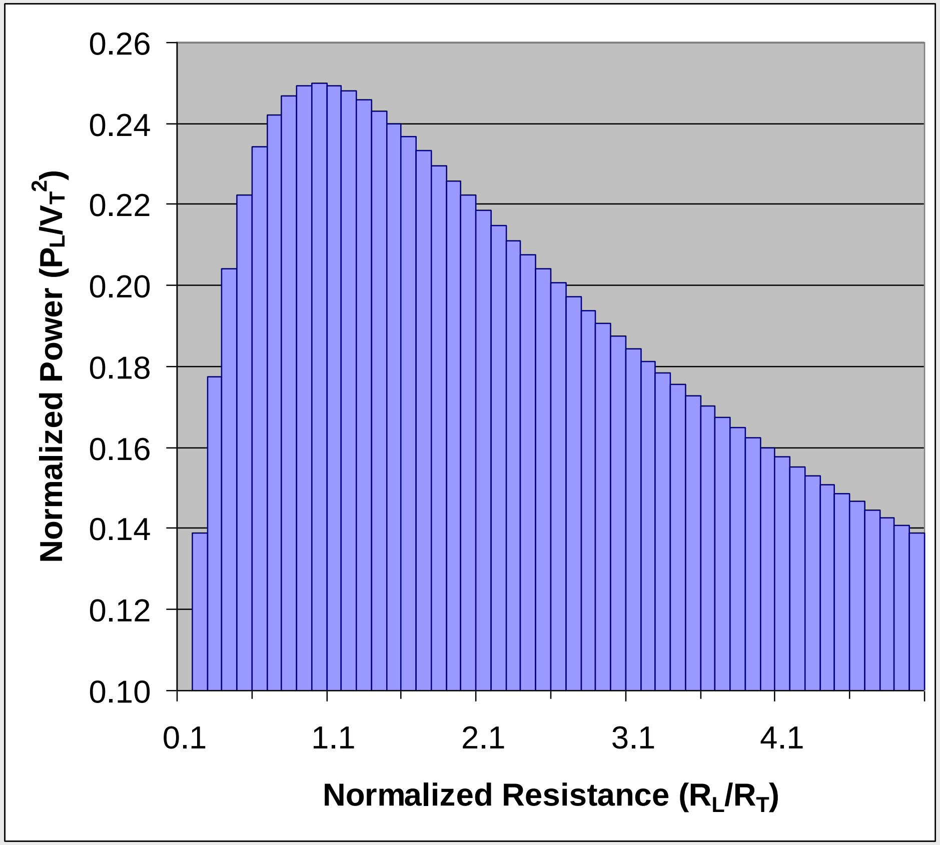


Digital vs. Analog

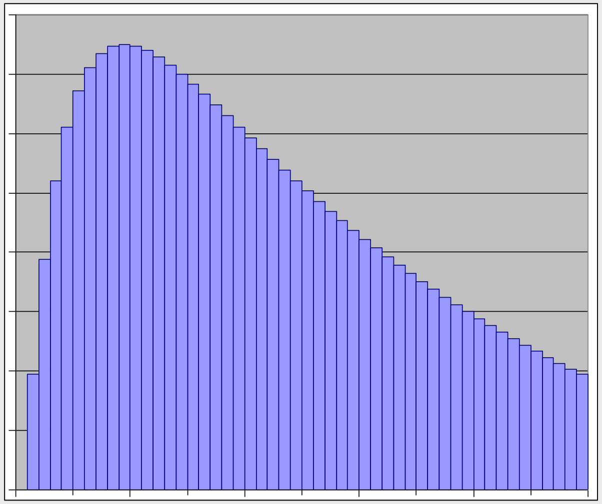
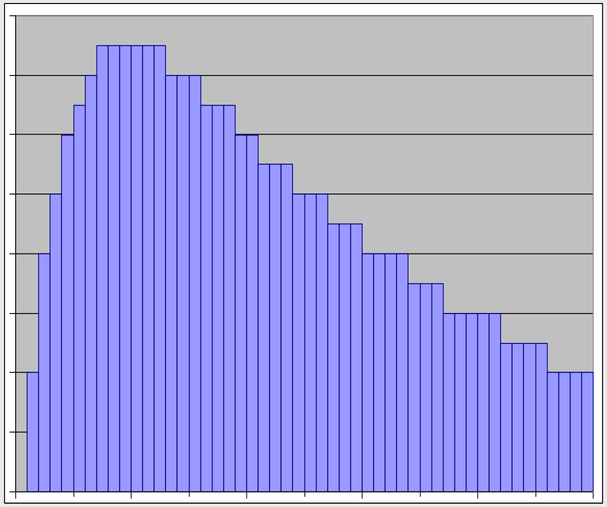
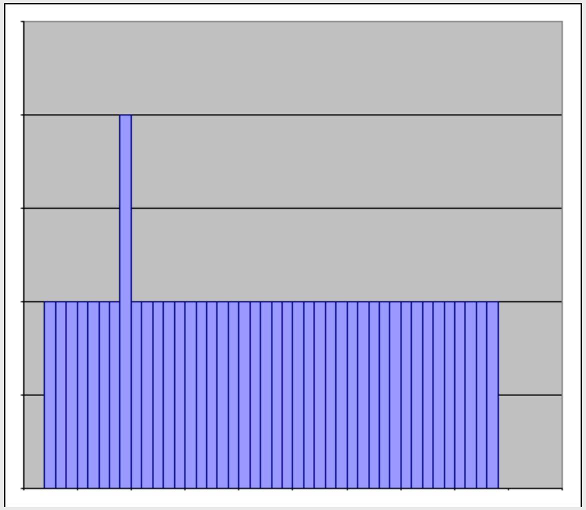
Analog



Digital



Digital vs. Analog

Digital signals are limited to a **set of possible values** (**precision**)

Set of **10** different symbol values → **decimal**

Set of **2** different symbol values → **binary**

Number Representations

- **Decimal** means that we have **ten** digits to use in our representation (the **symbols** 0 through 9)

  Example: What is 3,546?

  three thousands + *five* hundreds + *four* tens + *six* ones.

  $3{,}546_{10} = 3\times10^3 + 5\times10^2 + 4\times10^1 + 6\times10^0$

- How about negative numbers?
  - We use two more **symbols** to distinguish positive and negative, namely, **+** and **–**.

Number Representations

<u>Example1</u>: What is **1011.101**?

Number Representations

<u>Example1</u>: What is **1011.101**?

- Depends on what **radix** or **base** we use
  - Decimal → base = 10 (digit set: {0, 1, 2, 3, 4, 5, 6, 7, 8 , 9})
  - Binary → base = 2 (digit set: {0, 1})
  - Hexadecimal → base = 16 (digit set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F})
  - Other? → base = r (digit set: {0, … r-1})

Etymologically Correct Base Names

- 2 - binary
- 3 - ternary
- 4 - quaternary
- 5 - quinary
- 6 - senary
- 7 - septenary
- 8 - octal
- 9 - nonary
- 10 - denary, although this is never used; instead decimal is the common term.
- 11 - undenary
- 12 - duodenary, although this is never used; duodecimal is the accepted word.
- 16 - senidenary, although this is never used; see the discussion in hexadecimal.
- 20 - vegesimal
- 60 - sexagesimal

Number Representations

<u>Example1</u>: What is 1011.101?

- Depends on what **radix** or **base** we use
  - Decimal → base = 10 (digit set: {0, 1, 2, 3, 4, 5, 6, 7, 8 , 9})
  - Binary → base = 2 (digit set: {0, 1})
  - Hexadecimal → base = 16 (digit set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E , F})
  - Other? → base = r (digit set: {0, … r-1})

For base 10

- $1011.101_{10} = 1\times10^3 + 0\times10^2 + 1\times10^1 + 1\times10^0 + 1\times10^{-1} + 0\times10^{-2} + 1\times10^{-3}$

For base 2

- $1011.101_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

For base r

- $1011.101_r = 1\times r^3 + 0\times r^2 + 1\times r^1 + 1\times r^0 + 1\times r^{-1} + 0\times r^{-2} + 1\times r^{-3}$

Binary Numbers

- **Binary** means that we have **two** digits to use in our representation:
  - the **symbols** `0` and `1`

Example: What is 1011_2?

- *one* eights + *zero* fours + *one* twos + *one* ones.
- $1011_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

Binary Numbers

<u>Bit</u>: a single binary symbol (i.e. a **0** or a **1**)

- Bits rely only on *approximate* physical values.
  - A logical '`1`' is a relatively high voltage (1.2V, 3.3V, 5V).
  - A logical '`0`' is a relatively low voltage (0V - 1V).

<u>Byte</u>: a sequence of 8 bits

Binary Numbers

- Numbers are represented by a sequence of bits:
  - A collection of two bits has four possible values or **states**: 00, 01, 10, 11
  - A collection of three bits has eight possible **states**:
    000, 001, 010, 011, 100, 101, 110, 111
  - ***A collection of n bits has 2^n possible states.***
- By using groups of bits, we can achieve high **precision**.
  - 8 bits → number of states: 256.
  - 16 bits → number of states: 65,536
  - 32 bits → number of states: 4,294,967,296
  - 64 bits → number of states: 18,446,744,073,709,550,000

Data Types

- Bits alone don't give information – they must be interpreted
  - Data types are what interpret bits

<u>Example</u>: interpret the following bits

0100 1000 0100 0101   0101 1000 0100 0001

  - The **integers**: 18501_{10} and 22593_{10}?
  - The **characters**: H E X A ?
  - The **floating-point number**: 202081.015625_{10}?
  - Other?

Data Types

Unsigned integers

- 0, 1, 2, 3, 4, …

Signed integers

- …, -3, -2, -1, 0, 1, 2, 3, …

Floating point numbers

- PI = 3.14159 x 10^0

Characters

- ‘0’, ‘1’, ‘2’, … , ‘a’, ‘b’, ‘c’, … , ‘A’, ‘B’, ‘C’, … , ‘@’, ‘#’, …

Unsigned Integers

- Weighted positional notation
  - "3" is worth 300, because of its **position**, while "9" is only worth 9

329

10^2 10^1 10^0

3x100 + 2x10 + 9x1 = 329

Most Significant Bit (**MSB**)

101

Least Significant Bit (**LSB**)

2^2 2^1 2^0

1x4 + 0x2 + 1x1 = 5

- What do these **unsigned** binary numbers represent?

`0000`	`1111`	`0001`	`0111`	`1011`
`0110`	`1010`	`1000`	`1100`	`1001`

Unsigned Integers

Example2: What numbers can be represented with 3 bits?

Unsigned Integers

Example2: What numbers can be represented with 3 bits?

2^2	2^1	2^0	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Binary Arithmetic

- Base-2 addition – just like base-10!
  - add from right to left, propagating carry

```
  10010        10010        1111
+  1001      +  1011      +    1
  -----        -----       -----
  11011        11101       10000
```

carry

```
  10111
+   111
  -----
  11110
```

Subtraction, multiplication, division,…

Signed Binary Integers

- Determine the range of values for n bits

n bits → 2^n distinct values

½ for negative values

½ for positive values

$$\frac{2^n}{2} = 2^{n-1}$$

$$\frac{2^n}{2} = 2^{n-1}$$

$$-2^{n-1} \xrightarrow{?} 2^{n-1}$$

BUT – need a symbol for **zero**

Signed Binary Integers

3 common representations for **signed** integers:

1. Sign magnitude
2. 1's compliment
3. 2's compliment

Most common for computers

Sign-Magnitude

Range: $-\left(2^{n-1}-1\right) \longrightarrow \left(2^{n-1}-1\right)$

Representations

- 01111_{binary} => $15_{decimal}$
- 11111 => -15
- 00000 => 0
- 10000 => -0

The **MSB** encodes the sign:

`0` = +

`1` = –

Problem

- Difficult addition/subtraction
  - check signs
  - convert to positive
  - use adder or subtractor as required
- How to add two sign-magnitude numbers?
  - Ex: 1 + (-4)

1's Complement

Range: $-\left(2^{n-1}-1\right) \longrightarrow \left(2^{n-1}-1\right)$

Representations

- 00110_{binary} => $6_{decimal}$
- 11001 => -6
- 00000 => 0
- 11111 => -0

Problem

- Difficult addition/subtraction
  - no need to check signs as before
  - cumbersome logic circuits
    - *end-around-carry*
- How to add to one's complement numbers?
  - Ex: 4 + (-3)

To negate a number, Invert it, bit-by-bit.

MSB still encodes the sign:

0 = +

1 = –

Two’s Complement

- Problems with sign-magnitude and 1’s complement
  - two representations of zero (+0 and –0)
  - arithmetic circuits are complex
- ***Two’s complement*** representation developed to make circuits easy for arithmetic.
  - only one representation for zero
  - just **ADD** the two numbers to get the right answer (regardless of sign)

Two's Complement

Range: $\boxed{-\left(2^{n-1}\right) \longrightarrow \left(2^{n-1}-1\right)}$

Representation:

- If number is **positive** or **zero**,
  - normal binary representation, zeroes in upper bit(s)
- If number is **negative**,
  - start with positive number
  - flip every bit (i.e., take the one's complement)
  - then add one

MSB still encodes the sign:
`0` = +
`1` = –

```
  00101   (5)            01001   (9)
  11010   (1's comp)     10110   (1's comp)
+     1                +     1
  -----                  -----
  11011   (-5)           10111   (-9)
```

Two's Complement

Example3: What is **0110101**$_2$ in decimal?

What is it's 2's complement?

Two's Complement

<u>Example3</u>: What is $\mathbf{0110101}_2$ in decimal?

What is it's 2's complement?

$$\mathbf{0110101}_2 = 0\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$

$$= 53_{10}$$

Two's Complement

Example3: What is $\mathbf{0110101}_2$ in decimal?
What is it's 2's complement?

$$\mathbf{0110101}_2 = 0\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$
$$= 53_{10}$$

```
  0110101        (53)
  1001010   (1's comp)
+       1
  -------
  1001011       (-53)
```

2's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

$$\mathtt{0110} = 2^2 + 2^1 = 6$$

$$\mathtt{1110} = -2^3 + 2^2 + 2^1 = -2$$

$$\boxed{-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0}$$

2's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

```
0110 = 2² + 2¹ = 6
1110 = -2³ + 2² + 2¹ = -2
```

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^{1} \quad 2^{0}$$

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^{1} \quad 2^{0}$$

–MSB

2's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

`0110 = 2² + 2¹ = 6`

$0110 = 2^2 + 2^1 = 6$

$1110 = -2^3 + 2^2 + 2^1 = -2$

$$\boxed{-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0}$$

$$\boxed{-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0}$$

–MSB + remaining bits

2's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

`0110 = 2`^{2}` + 2`^{1}` = 6`

`1110 = -2`^{3}` + 2`^{2}` + 2`^{1}` = -2`

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0$$

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0$$

–MSB + remaining bits

`11111111`

2's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

```
0110 = 2^2 + 2^1 = 6
1110 = -2^3 + 2^2 + 2^1 = -2
```

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^{1} \quad 2^{0}$$

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^{1} \quad 2^{0}$$

–MSB + remaining bits

11111111

2's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

$$0110 = 2^2 + 2^1 = 6$$

$$1110 = -2^3 + 2^2 + 2^1 = -2$$

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0$$

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0$$

–MSB + remaining bits

$$11111111 = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

$$= -1$$

Two's Complement

- Positional number representation with a twist
  - **MSB** has a *negative* weight

```
0110 = 2² + 2¹ = 6
1110 = -2³ + 2² + 2¹ = -2
```

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0$$

$$-2^{n-1} \quad 2^{n-2} \quad \cdots \quad 2^1 \quad 2^0$$

–MSB + remaining bits = –1

$11111111 = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -1$

Two's Complement Shortcut

- To take the two's complement of a number:
  1. copy bits from right to left until (and including) the first **`1`**
  2. flip remaining bits to the left

```
  011010000
  100101111   (1's comp)
+         1
  ---------
  100110000
```

```
 011010000
(flip)   (copy)
 100110000
```

Two’s Complement Negation

- To negate a number, invert all the bits and add 1 (or use shortcut)

Number	Decimal Value	Negated Binary Value
0110	6	1010
0111	7	1001
0000	0	0000
1111	-1	0001
0100	4	1100
1000	-8	1000 (??)

Decimal to Binary Conversion

Positive numbers

❶ start with empty result

❷ if decimal number is odd, prepend '`1`' to result
else prepend '`0`'

❸ divide number by 2, throw away fractional part (INTEGER divide)

❹ if number is non-zero, go back to ❷ else you are done

Negative numbers

- do above for **positive version** of number and **negate result.**

Decimal to Binary Conversion

Number	Binary Value
5	0101
6	0110
123	01111011
35	00100011
-35	11011101
1007	01111101111

Hexadecimal Notation

- Binary is hard to read and write by hand
- Hexadecimal is a common alternative
  - 16 digits are **0123456789ABCDEF**

Binary	Hex	Dec
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

```
0100 0111 1000 1111 = 0x478F
1101 1110 1010 1101 = 0xDEAD
1011 1110 1110 1111 = 0xBEEF
1010 0101 1010 0101 = 0xA5A5
```

0x is a common prefix for writing numbers which means *hexadecimal*

1. Separate binary code into groups of 4 bits (starting from the right)
2. Translate each group into a single hex digit

Binary to Hex Conversion

- Every four bits is a hex digit.
  - start grouping from right-hand side

011	1010	1000	1111	0100	1101	0111
⇓	⇓	⇓	⇓	⇓	⇓	⇓
3	**A**	**8**	**F**	**4**	**D**	**7**

This is not a new machine representation, just a convenient way to write the number.