## Schedule...

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

Normalized Power ($P_L/V_T^2$) vs. Normalized Resistance ($R_L/R_T$)

Normalized Power ($P_L/V_T^2$) vs. Normalized Resistance ($R_L/R_T$)
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.

\[3,546_{10} = 3 \times 10^3 + 5 \times 10^2 + 4 \times 10^1 + 6 \times 10^0\]

- How about negative numbers?

- We use two more symbols to distinguish positive and negative, namely, + and −.
Example 1: What is $1011.101$?
Example 1: 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

**Example 1:** What is 1011.101?

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

**For base 10**

\[ 1011.101_{10} = 1 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 1 \times 10^0 + 1 \times 10^{-1} + 0 \times 10^{-2} + 1 \times 10^{-3} \]

**For base 2**

\[ 1011.101_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \]

**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.
- $1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
Binary Numbers

**Bit**: 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).

**Byte**: 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

Example: interpret the following bits
0100100001000101 0101100001000001

▲ 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**

Unsigned Integers

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

<table>
<thead>
<tr>
<th>Number</th>
<th>MSB</th>
<th>Least Significant Bit (LSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>329</td>
<td>101</td>
<td>01</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>00</td>
</tr>
</tbody>
</table>

- What do these unsigned binary numbers represent?

<table>
<thead>
<tr>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
</tr>
<tr>
<td>1111</td>
</tr>
<tr>
<td>0001</td>
</tr>
<tr>
<td>0111</td>
</tr>
<tr>
<td>1011</td>
</tr>
<tr>
<td>0110</td>
</tr>
<tr>
<td>1010</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1100</td>
</tr>
<tr>
<td>1001</td>
</tr>
</tbody>
</table>
Example 2: What numbers can be represented with 3 bits?
## Unsigned Integers

**Example 2:** What numbers can be represented with 3 bits?

<table>
<thead>
<tr>
<th>$2^2$</th>
<th>$2^1$</th>
<th>$2^0$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Binary Arithmetic

◆ Base-2 addition – just like base-10!

Add from right to left, propagating carry

\[
\begin{align*}
10010 + 1001 &= 11011 \\
10010 + 1011 &= 11101 \\
1111 + 1 &= 10000
\end{align*}
\]

Subtraction, multiplication, division,…
Signed Binary Integers

Determine the range of values for n bits

n bits $\rightarrow 2^n$ distinct values

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

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

$-2^{n-1} \rightarrow 2^{n-1}$

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: $-2^{n-1} + 1 \rightarrow 2^{n-1} - 1$

Representations
- $01111_{binary} \Rightarrow 15_{decimal}$
- $11111 \Rightarrow -15$
- $00000 \Rightarrow 0$
- $10000 \Rightarrow -0$

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: \(-2^{n-1} - 1 \rightarrow 2^{n-1} - 1\)

Representations
- \(00110_{\text{binary}} \Rightarrow 6_{\text{decimal}}\)
- \(11001 \Rightarrow -6\)
- \(00000 \Rightarrow 0\)
- \(11111 \Rightarrow -0\)

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:

\[
\begin{align*}
0 &= + \\
1 &= -
\end{align*}
\]
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: $-2^{n-1} \ldots 2^{n-1} - 1$

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

\[
\begin{align*}
\text{00101} & \quad (5) \\
\text{11010} & \quad (1’s \text{ comp}) \\
+ \quad \text{1} & \quad + \quad \text{1} \\
\text{11011} & \quad (-5) \\
\end{align*}
\]

\[
\begin{align*}
\text{01001} & \quad (9) \\
\text{10110} & \quad (1’s \text{ comp}) \\
+ \quad \text{1} & \quad + \quad \text{1} \\
\text{10111} & \quad (-9)
\end{align*}
\]

MSB still encodes the sign:
0 = +
1 = −
Two’s Complement

Example 3: What is $0110101_2$ in decimal? What is its 2’s complement?
Two’s Complement

**Example 3**: What is $0110101_2$ in decimal? What is its 2’s complement?

$$0110101_2 = 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 53_{10}$$
Two’s Complement

**Example 3:** What is $0110101_2$ in decimal?
What is its 2’s complement?

\[
0110101_2 = 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0
\]
\[
= 53_{10}
\]

\[
\begin{array}{c}
0110101 \\
1001010 \quad (1's \ comp)
\end{array}
\]
\[
+ 1
\]
\[
1001011 \quad (-53)
\]
2’s Complement

Positional number representation with a twist

- **MSB** has a *negative* weight

\[
\begin{align*}
0110 &= 2^2 + 2^1 = 6 \\
1110 &= -2^3 + 2^2 + 2^1 = -2
\end{align*}
\]
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’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’s Complement

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

\[
\begin{align*}
0110 &= 2^2 + 2^1 = 6 \\
1110 &= -2^3 + 2^2 + 2^1 = -2
\end{align*}
\]

11111111
2’s Complement

Positional number representation with a twist

- **MSB** has a *negative* weight

\[
\begin{align*}
0110 &= 2^2 + 2^1 = 6 \\
1110 &= -2^3 + 2^2 + 2^1 = -2
\end{align*}
\]
2’s Complement

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

\[
\begin{align*}
0110 &= 2^2 + 2^1 = 6 \\
1110 &= -2^3 + 2^2 + 2^1 = -2
\end{align*}
\]

\[
\begin{align*}
11111111 &= -2^7 + \sum_{i=0}^{6} 2^i = -1
\end{align*}
\]
Two’s Complement

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

\[ \begin{align*}
0110 & = 2^2 + 2^1 = 6 \\
1110 & = -2^3 + 2^2 + 2^1 = -2
\end{align*} \]
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

\[
\begin{align*}
011010000 & \quad \text{(1’s comp)} \\
100101111 & \quad \text{(flip)} \\
+ & \quad 1 \\
100110000 & \quad \text{(copy)}
\end{align*}
\]
Two’s Complement Negation

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

<table>
<thead>
<tr>
<th>Number</th>
<th>Decimal Value</th>
<th>Negated Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110</td>
<td>6</td>
<td>1010</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>1001</td>
</tr>
<tr>
<td>0000</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1111</td>
<td>−1</td>
<td>0001</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>1100</td>
</tr>
<tr>
<td>1000</td>
<td>−8</td>
<td>1000 (??)</td>
</tr>
</tbody>
</table>
Decimal to Binary Conversion

Positive numbers

1. start with empty result
2. if decimal number is odd, prepend ‘1’ to result else prepend ‘0’
3. divide number by 2, throw away fractional part (INTEGER divide)
4. if number is non-zero, go back to 2 else you are done

Negative numbers

do above for positive version of number and negate result.
## Decimal to Binary Conversion

<table>
<thead>
<tr>
<th>Number</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>123</td>
<td>01111011</td>
</tr>
<tr>
<td>35</td>
<td>00100011</td>
</tr>
<tr>
<td>-35</td>
<td>11011101</td>
</tr>
<tr>
<td>1007</td>
<td>01111101111</td>
</tr>
</tbody>
</table>
Hexadecimal Notation

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

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

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

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hex</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
<td>13</td>
</tr>
<tr>
<td>1110</td>
<td>E</td>
<td>14</td>
</tr>
<tr>
<td>1111</td>
<td>F</td>
<td>15</td>
</tr>
</tbody>
</table>
Binary to Hex Conversion

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

```
0111 0101 0001 1110 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.