# UESTC1008: Microelectronic Systems

## Lec 8 Numbers

Dr. Guodong Zhao
School of Engineering
University of Glasgow

# Binary Numeral System

- Used in computers as a series of "off" and "on" switches
- A way to write numbers using only two digits (‘bits’): 0 and 1
- Each digit's place value is twice as much as that of the next digit to the right and the place value increases by a power of two (1's, 2's, 4's place, etc.)
- In decimal, each digit holds ten values, and the place value increases by a power of ten (1's, 10's, 100's place, etc.)

| *Decimal* | *Binary* | *Explanation* |
|---|---|---|
| 1 | 0001 | 0+0+0+1 |
| 2 | 0010 | 0+0+2+0 |
| 3 | 0011 | 0+0+2+1 |
| 4 | 0100 | 0+4+0+0 |
| 5 | 0101 | 0+4+0+1 |
| 6 | 0110 | 0+4+2+0 |
| 7 | 0111 | 0+4+2+1 |
| 8 | 1000 | 8+0+0+0 |
| 9 | 1001 | 8+0+0+1 |
| 10 | 1010 | 8+0+2+0 |
| 11 | 1011 | 8+0+2+1 |
| 12 | 1100 | 8+4+0+0 |
| 13 | 1101 | 8+4+0+1 |
| 14 | 1110 | 8+4+2+0 |
| 15 | 1111 | 8+4+2+1 |
| 16 | 10000 | 16+0+0+0+0 |

http://simple.wikipedia.org/wiki/Binary_numeral_system

# Hexadecimal System

- In mathematics and computing, hexadecimal (also base16, or hex) is a positional numeral system with a radix, or base, of 16.
- It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a–f) to represent values ten to fifteen.
- For example, the hexadecimal number 2AF3 is equal, in decimal, to $(2 \times 16^3) + (10 \times 16^2) + (15 \times 16^1) + (3 \times 16^0)$, or 10995.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $0_{hex}$ | = | $0_{dec}$ | = | $0_{oct}$ | 0 | 0 | 0 | 0 |
| $1_{hex}$ | = | $1_{dec}$ | = | $1_{oct}$ | 0 | 0 | 0 | 1 |
| $2_{hex}$ | = | $2_{dec}$ | = | $2_{oct}$ | 0 | 0 | 1 | 0 |
| $3_{hex}$ | = | $3_{dec}$ | = | $3_{oct}$ | 0 | 0 | 1 | 1 |
| $4_{hex}$ | = | $4_{dec}$ | = | $4_{oct}$ | 0 | 1 | 0 | 0 |
| $5_{hex}$ | = | $5_{dec}$ | = | $5_{oct}$ | 0 | 1 | 0 | 1 |
| $6_{hex}$ | = | $6_{dec}$ | = | $6_{oct}$ | 0 | 1 | 1 | 0 |
| $7_{hex}$ | = | $7_{dec}$ | = | $7_{oct}$ | 0 | 1 | 1 | 1 |
| $8_{hex}$ | = | $8_{dec}$ | = | $10_{oct}$ | 1 | 0 | 0 | 0 |
| $9_{hex}$ | = | $9_{dec}$ | = | $11_{oct}$ | 1 | 0 | 0 | 1 |
| $A_{hex}$ | = | $10_{dec}$ | = | $12_{oct}$ | 1 | 0 | 1 | 0 |
| $B_{hex}$ | = | $11_{dec}$ | = | $13_{oct}$ | 1 | 0 | 1 | 1 |
| $C_{hex}$ | = | $12_{dec}$ | = | $14_{oct}$ | 1 | 1 | 0 | 0 |
| $D_{hex}$ | = | $13_{dec}$ | = | $15_{oct}$ | 1 | 1 | 0 | 1 |
| $E_{hex}$ | = | $14_{dec}$ | = | $16_{oct}$ | 1 | 1 | 1 | 0 |
| $F_{hex}$ | = | $15_{dec}$ | = | $17_{oct}$ | 1 | 1 | 1 | 1 |

# Decimal –to– Binary Conversion

The Process : *Successive Division*

a) Divide the *Decimal Number* by 2; the remainder is the LSB of *Binary Number*.

b) If the quotation is zero, the conversion is complete; else repeat step (a) using the quotation as the Decimal Number. The new remainder is the next most significant bit of the *Binary Number.*

Example:

Convert the decimal number $6_{10}$ into its binary equivalent.

$$2\overline{)6}^{\,3} \quad r = 0 \leftarrow \text{Least Significant Bit}$$

$$2\overline{)3}^{\,1} \quad r = 1$$

$$2\overline{)1}^{\,0} \quad r = 1 \leftarrow \text{Most Significant Bit}$$

$$\therefore \; 6_{10} = 110_2$$

# Dec → Binary : Example #1

*Example*:

Convert the decimal number $26_{10}$ into its binary equivalent.

*Solution*:

$$2\,\overline{)\,26}^{\;13} \quad r = 0 \leftarrow \text{LSB}$$

$$2\,\overline{)\,13}^{\;6} \quad r = 1$$

$$2\,\overline{)\,6}^{\;3} \quad r = 0$$

$$2\,\overline{)\,3}^{\;1} \quad r = 1$$

$$2\,\overline{)\,1}^{\;0} \quad r = 1 \leftarrow \text{MSB}$$

$$\therefore \; 26_{10} = 11010_2$$

# Dec → Binary : Example #2

*Example*:

Convert the decimal number $41_{10}$ into its binary equivalent.

*Solution*:

$$2\overline{)\,41}^{\;20} \quad r = 1 \leftarrow \text{LSB}$$

$$2\overline{)\,20}^{\;10} \quad r = 0$$

$$2\overline{)\,10}^{\;5} \quad r = 0$$

$$2\overline{)\,5}^{\;2} \quad r = 1$$

$$2\overline{)\,2}^{\;1} \quad r = 0$$

$$2\overline{)\,1}^{\;0} \quad r = 1 \leftarrow \text{MSB}$$

$$\boxed{\therefore\ 41_{10} = 101001_2}$$

# Binary –to– Decimal Process

The Process : *Weighted Multiplication*

a) Multiply each bit of the *Binary Number* by it corresponding bit-weighting factor (i.e. Bit-0→$2^0$=1; Bit-1→$2^1$=2; Bit-2→$2^2$=4; etc).

b) Sum up all the products in step (a) to get the *Decimal Number*.

Example:

Convert the binary number $0110_2$ into its decimal equivalent.

$$\begin{array}{ccccccccc} 0 & & 1 & & 1 & & 0 & & \\ 2^3 & & 2^2 & & 2^1 & & 2^0 & & \\ 8 & & 4 & & 2 & & 1 & & \\ 0 & + & 4 & + & 2 & + & 0 & = & 6_{10} \end{array}$$

Bit-Weighting Factors

$$\boxed{\therefore\ 0110_2 = 6_{10}}$$

# Binary → Dec : Example #1

*Example*:

Convert the binary number $10010_2$ into its decimal equivalent.

*Solution*:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 0 | | 0 | | 1 | | 0 | | |
| $2^4$ | | $2^3$ | | $2^2$ | | $2^1$ | | $2^0$ | | |
| 16 | | 8 | | 4 | | 2 | | 1 | | |
| 16 | + | 0 | + | 0 | + | 2 | + | 0 | = | $18_{10}$ |

$$\therefore 10010_2 = 18_{10}$$

# Binary → Dec : Example #2

*Example*:

Convert the binary number $0110101_2$ into its decimal equivalent.

*Solution*:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 1 | | 1 | | 0 | | 1 | | 0 | | 1 | | |
| $2^6$ | | $2^5$ | | $2^4$ | | $2^3$ | | $2^2$ | | $2^1$ | | $2^0$ | | |
| 64 | | 32 | | 16 | | 8 | | 4 | | 2 | | 1 | | |
| 0 | + | 32 | + | 16 | + | 0 | + | 4 | + | 0 | + | 1 | = | $53_{10}$ |

$$\therefore 0110101_2 = 53_{10}$$

# Negative Binary Numbers

Four different systems for representing negative numbers have been used in digital computers

1. The first one is called **signed magnitude**. The leftmost bit is the sign bit (0 is + and 1 is -) and the remaining bits hold the absolute magnitude of the number.
2. The second system, called **one's complement**, also has a sign bit with 0 for a plus and 1 for minus. To negate a number, replace each 1 by 0 and each 0 by a 1. This holds for the sign bit as well.

# Negative Binary Numbers

3. The third system, called **two's complement**, also has a sign bit that is 0 for plus and 1 for minus.
   - Negating numbers is a two-step process. First, each 1 is replaced by a 0 and each 0 by a 1, just as in one's complement. Second, 1 is added to the result.

   - 00000110 (+6)
   - 10000110 (-6 in signed magnitude)
   - 11111001 (-6 in one's complement)
   - 11111010 (-6 in two's complement)

# Negative Binary Numbers

4. The fourth system, which for *m*-bit numbers is called excess $2^{m-1}$, represents a number by storing it as the sum of itself and $2^{m-1}$.
   - For example, for 8-bit numbers, $m = 8$, the system is called excess 128 and a number is stored as its true value plus 128. Thus, -3 becomes -3 + 128 = 125.
     - In this case, the numbers from -128 to +127 map onto 0 to 255.
     - This system is identical to two's complement with the sign bit reversed.

# Negative Binary Numbers

– Both signed magnitude and one's complement have two representations for zero: a plus zero, and a minus zero. This is undesirable.

– The two's complement system does not have this problem

# Terms

- **Byte**
  - contains 8 bits

- **Halfword or double byte**
  - contains 16 bits

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |

- **Word**
  - on the ARM Cortex M will have 32 bits

# Terms

- **SI-decimal abbreviations**
  - International System of Units
  - Represent powers of 10
  - 2 kilovolts = 2000 volts
- **IEC-binary abbreviations**
  - International Electrotechnical Commission
  - Represent powers of 2
- **kB**
  - Kilo Byte
  - A unit of information or computer storage
  - 1 kB = $2^{10}$ bytes = 1024 bytes
- **MB**
  - Mega Byte
  - 1 MB = $2^{20}$ bytes = 1048576 bytes
- **GB**
  - Giga Byte
  - 1 GB = $2^{30}$ bytes = 1,073,741,824 bytes

Tera Byte (TB) $2^{40}$ Peta Byte (PB) $2^{50}$ byte