Chapter 1
Digital Systems and Binary Numbers

gürtaç yemischioğlu
OUTLINE OF CHAPTER 1

Digital Systems
Binary Numbers
Binary Arithmetic
Number-base Conversions
Octal & Hexadecimal Numbers

Complements
Signed Binary Numbers
Binary Codes
Binary Storage & Registers
Binary Logic
1.1 DIGITAL SYSTEMS
DIGITAL SYSTEMS

- Digital age and information age
- Digital computers
  - General purposes
  - Many scientific, industrial and commercial applications
- Digital systems
  - Telephone switching exchanges
  - Digital camera
  - Electronic calculators, PDA's
  - Digital TV
- Discrete information-processing systems
  - Manipulate discrete elements of information
  - For example, \{1, 2, 3, \ldots\} and \{A, B, C, \ldots\}...
DIGITAL SIGNAL

- The physical quantities or signals can assume only discrete values.
- Greater accuracy

ANALOG SIGNAL

- The physical quantities or signals may vary continuously over a specified range.
• Binary digital signal
  – An information variable represented by physical quantity.
  – For digital systems, the variable takes on discrete values.
    • Two level, or binary values are the most prevalent values.
  – Binary values are represented abstractly by:
    • Digits 0 and 1
    • Words (symbols) False (F) and True (T)
    • Words (symbols) Low (L) and High (H)
    • And words On and Off
• Binary values are represented by values or ranges of values of physical quantities.
1.2 BINARY NUMBERS
Decimal Number System

- **Base (also called radix) = 10**
  - 10 digits
  - \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}

- **Digit Position**
  - Integer & fraction

- **Digit Weight**
  - Weight = \((\text{Base})^\text{Position}\)

- **Magnitude**
  - Sum of “Digit x Weight”

- **Formal Notation**

\[
\begin{align*}
&d_2 \times B^2 + d_1 \times B^1 + d_0 \times B^0 + d_{-1} \times B^{-1} + d_{-2} \times B^{-2} \\
&= (512.74)_{10}
\end{align*}
\]
Octal Number System

- Base = 8
  - 8 digits
  - \{0, 1, 2, 3, 4, 5, 6, 7\}

- Weights
  - Weight = \( (Base)^{Position} \)

- Magnitude
  - Sum of “Digit x Weight”

- Formal Notation

\[
\begin{align*}
0_2 \times B^2 + 0_1 \times B^1 + 0_0 \times B^0 + 0_{-1} \times B^{-1} + 0_{-2} \times B^{-2} = 0.875 + 0.0625 \\
(330.9375)_{10} = (330.9375)_{8} = (512.74)_{8}
\end{align*}
\]
Hexadecimal Number System

- **Base = 16**
  - 16 digits \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}

- **Weights**
  - Weight = \((Base)^{Position}\)

- **Magnitude**
  - Sum of “bit x Weight”

- **Formal Notation**

<table>
<thead>
<tr>
<th>Base</th>
<th>16(^2)</th>
<th>16(^1)</th>
<th>16(^0)</th>
<th>16(^{-1})</th>
<th>16(^{-2})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>256</td>
<td>16</td>
<td>1</td>
<td>1/16</td>
<td>1/256</td>
</tr>
<tr>
<td>Bit</td>
<td>(1 \times 256)</td>
<td>(14 \times 16)</td>
<td>(5 \times 1)</td>
<td>(7 \times 1/16)</td>
<td>(10 \times 1/256)</td>
</tr>
</tbody>
</table>

\[ \text{Hexadecimal Value} = \text{Binary Value} \]

\[ \text{Binary Value} = H_2 \times B^2 + H_1 \times B^1 + H_0 \times B^0 + H_{-1} \times B^{-1} + H_{-2} \times B^{-2} \]

\[ (458.4765625)_{10} \]

\[ (1E5.7A)_{16} \]
**Binary Number System**

- **Base** = 2
  - 2 digits {0, 1}, called *binary digits* or "*bits*"

- **Weights**
  - Weight = \((\text{Base})^\text{Position}\)

- **Magnitude**
  - Sum of "*bit* x *Weight*"

- **Formal Notation**

- **Groups of bits**
  - 4 bits = *Nibble*, 8 bits = *Byte*

---

### Binary Number System Example

**Binary Representation:**

- \(1011_2\)

**Binary to Decimal Conversion:**

\[
1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11_{10}
\]

**Decimal to Binary Conversion:**

\[
5.25_{10} = (101.01)_2
\]

---

**Weights Table:**

<table>
<thead>
<tr>
<th>Weight Position</th>
<th>Base</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>-2</td>
<td>2</td>
<td>0.25</td>
</tr>
</tbody>
</table>

---

**Magnitude Example:**

- \(1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}\)
## Binary Numbers

The power of 2

<table>
<thead>
<tr>
<th>n</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$2^0 = 1$</td>
</tr>
<tr>
<td>1</td>
<td>$2^1 = 2$</td>
</tr>
<tr>
<td>2</td>
<td>$2^2 = 4$</td>
</tr>
<tr>
<td>3</td>
<td>$2^3 = 8$</td>
</tr>
<tr>
<td>4</td>
<td>$2^4 = 16$</td>
</tr>
<tr>
<td>5</td>
<td>$2^5 = 32$</td>
</tr>
<tr>
<td>6</td>
<td>$2^6 = 64$</td>
</tr>
<tr>
<td>7</td>
<td>$2^7 = 128$</td>
</tr>
<tr>
<td>8</td>
<td>$2^8 = 256$</td>
</tr>
<tr>
<td>9</td>
<td>$2^9 = 512$</td>
</tr>
<tr>
<td>10</td>
<td>$2^{10} = 1024$</td>
</tr>
<tr>
<td>11</td>
<td>$2^{11} = 2048$</td>
</tr>
<tr>
<td>12</td>
<td>$2^{12} = 4096$</td>
</tr>
<tr>
<td>20</td>
<td>$2^{20} = 1M$</td>
</tr>
<tr>
<td>30</td>
<td>$2^{30} = 1G$</td>
</tr>
<tr>
<td>40</td>
<td>$2^{40} = 1T$</td>
</tr>
</tbody>
</table>
1.3 BINARY ARITHMETIC
**Binary Arithmetic**

**Addition**

Decimal Addition

```
  1  1
+  5  5
  1  1  0
```

- **Carry**: 1
- **Result**: 110
- **Equation**: 11 (Base 10) ≥ Base
- **Operation**: Subtract a Base

---

*6 October, 2016*
BINARY ARITHMETIC

ADDITION
Binary Addition - Column Addition

1 1 1 1 1 1
1 1 1 1 1 0 1
+ 1 0 1 1 1 1
---
1 0 1 0 1 0 0

≥ (2)_{10}
SUBTRACTION
Binary Subtraction - Borrow a “Base” when needed

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 0 & 1 \\
0 & 2 & 0 & 0 & 2 & 2 \\
\hline \\
1 & 0 & 1 & 1 & 1 & 1 \\
\hline \\
0 & 1 & 1 & 0 & 1 & 1 & 0 \\
= 54 \\
\end{array}
\]

\[
\begin{array}{cccccc}
= 23 \\
= 77 \\
= (10)_2
\end{array}
\]
MULTIPLICATION
Binary Multiplication – Bit by bit

\[
\begin{array}{c}
\times \\
10111 \\
\hline
00000 0 \\
10111 1 \\
00000 0 \\
10111 1 \\
\hline
111001110
\end{array}
\]
1.4 NUMBER-BASE CONVERSION
NUMBER BASE CONVERSION

Evaluate
Magnitude

Decimal
(Base 10)

Octal
(Base 8)

Binary
(Base 2)

Hexadecimal
(Base 16)

Evaluate
Magnitude

Evaluate
Magnitude

Evaluate
Magnitude
Decimal to binary conversion

- Divide the number by the ‘Base’ (=2)
- Take the remainder (either 0 or 1) as a coefficient
- Take the quotient and repeat the division
Decimal (integer) to binary conversion

Example: \((13)_{10}\)

<table>
<thead>
<tr>
<th>Quotient</th>
<th>Remainder</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 / 2 = 6</td>
<td>1</td>
<td>a_0 = 1</td>
</tr>
<tr>
<td>6 / 2 = 3</td>
<td>0</td>
<td>a_1 = 0</td>
</tr>
<tr>
<td>3 / 2 = 1</td>
<td>1</td>
<td>a_2 = 1</td>
</tr>
<tr>
<td>1 / 2 = 0</td>
<td>1</td>
<td>a_3 = 1</td>
</tr>
</tbody>
</table>

Answer: \((13)_{10} = (a_3 a_2 a_1 a_0)_{2} = (1101)_{2}\)
NUMBER BASE CONVERSION

Decimal (fraction) to binary conversion

• Multiply the number by the ‘Base’ (=2)
• Take the integer (either 0 or 1) as a coefficient
• Take the resultant fraction and repeat the division
## Decimal (fraction) to binary conversion

**Example:** \((0.625)_{10}\)

<table>
<thead>
<tr>
<th>Integer</th>
<th>Fraction</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.625</td>
<td>* 2 = 1</td>
<td>(a_{-1} = 1)</td>
</tr>
<tr>
<td>0.25</td>
<td>* 2 = 0</td>
<td>(a_{-2} = 0)</td>
</tr>
<tr>
<td>0.5</td>
<td>* 2 = 1</td>
<td>(a_{-3} = 1)</td>
</tr>
</tbody>
</table>

Answer: \((0.625)_{10} = (0.a_{-1}a_{-2}a_{-3})_2 = (0.101)_2\)
Decimal (integer) to octal conversion

Example: \((175)_{10}\)

<table>
<thead>
<tr>
<th>Quotient</th>
<th>Remainder</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>175 \div 8 = 21</td>
<td>7</td>
<td>(a_0 = 7)</td>
</tr>
<tr>
<td>21 \div 8 = 2</td>
<td>5</td>
<td>(a_1 = 5)</td>
</tr>
<tr>
<td>2 \div 8 = 0</td>
<td>2</td>
<td>(a_2 = 2)</td>
</tr>
</tbody>
</table>

Answer: \((175)_{10} = (a_3 \, a_2 \, a_1 \, a_0)_{2} = (257)_{8}\)
Decimal (fraction) to octal conversion

Example: \((0.3125)_{10}\)

<table>
<thead>
<tr>
<th>Integer</th>
<th>Fraction</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3125  * 8 = 2 . 5</td>
<td>a₁ = 2</td>
<td></td>
</tr>
<tr>
<td>0.5 * 8 = 4 . 0</td>
<td>a₂ = 4</td>
<td></td>
</tr>
</tbody>
</table>

Answer: \((0.3125)_{10} = (0.a₁ a₂ a₃)₂ = (0.24)₈\)
1.5 OCTAL & HEXADECIMAL NUMBERS
OCTAL & HEXADECIMAL NUMBERS

- Binary to octal conversion
- $8 = 2^3$
- Each group of 3 bits represents an octal digit

**Example:**

Assume Zeros

```
(01 01 10 . 01 0)₂
```

(2 6 . 2)₈

Works *both* ways (*Binary to Octal & Octal to Binary*)

<table>
<thead>
<tr>
<th>Octal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>
OCTAL & HEXADECIMAL NUMBERS

- Binary to hexadecimal conversion
- $16 = 2^4$
- Each group of 4 bits represents a hexadecimal digit

**Example:**

Assume Zeros

(0001 0110 . 0100)$_2$

(1 6 . 4)$_{16}$

Works both ways (Binary to Hex & Hex to Binary)
OCTAL & HEXADECIMAL NUMBERS

- Octal to hexadecimal conversion
- Convert to binary as an intermediate step

Example: (2 6 . 2)\textsubscript{8}

Assume Zeros

(0 0 1 0 1 1 0 . 0 1 0 0)\textsubscript{2}

(1 6 . 4)\textsubscript{16}
## OCTAL & HEXADECIMAL NUMBERS

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Octal</th>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>07</td>
<td>07</td>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>08</td>
<td>10</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>09</td>
<td>11</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>A</td>
<td>1010</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>B</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>C</td>
<td>1100</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>D</td>
<td>1101</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>E</td>
<td>1110</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>F</td>
<td>1111</td>
</tr>
</tbody>
</table>
1.6 COMPLEMENTS
COMPLEMENTS

• Complements are used in digital computers to simplify the subtraction operation and for logical manipulation. Simplifying operations leads to simpler, less expensive circuits to implement the operations.

• There are two types of complements for each base-\( r \) system:
  – diminished radix complement.
  – the radix complement
• **Diminished Radix Complement (r-1)’s Complement**
  - Given a Number = \( N \), base = \( r \), digits = \( n \),
  - The \((r-1)’\)'s complement of \( N \) is defined as:
    \[(r^n - 1) - N\]

• **Example for 6-digit decimal numbers:**
  - 9’s complement is \((r^n - 1)-N = (10^6-1)-N = 999999-N\)
  - 9’s complement of 546700 is 999999-546700 = 453299
• For decimal numbers $r = 10$ and
  – $(r-1) = 9$, this is called 9’s complement of $N$.
  – $(10^n-1)-N$.
  – $10^n$ represents a number that consist of a single 1 followed by $n$ 0’s.
  – $10^n-1$ is a number represented by $n$ 9’s.

• For binary numbers $r = 2$ and
  – $r-1 = 1$, this is called 1’s complement of $N$.
  – $(2^n-1)-N$.
  – $2^n$ represents a binary number that consist of a 1 followed by $n$ 0’s.
COMPLEMENTS

• For Example: 0011

\[
\begin{align*}
(2^4 - 1) &= 0011 \\
2^4 &= 10000 \\
-0011 &= 11000
\end{align*}
\]

• When subtracting binary digit from 1 we can have either 1-0 =1 or 1-1 =0 which causes bit to change from 0 to 1 or from 1 to 0.

• 1’s complement of a binary is formed by changing 1’s to 0’s and 0’s to 1’s.

• Example: 0011011 \rightarrow 1100100
• **Example for 7-digit binary numbers:**
  
  – 1’s complement is \((r^n - 1) - N = (2^7 - 1) - N = 1111111 - N\)
  
  – 1’s complement of 1011000 is 1111111 - 1011000 = 0100111

• **Observation:**
  
  – Subtraction from \((r^n - 1)\) will never require a borrow
  
  – Diminished radix complement can be computed digit-by-digit
COMPLEMENTS

1's Complement (\textit{Diminished Radix} Complement)

- All ‘0’s become ‘1’s
- All ‘1’s become ‘0’s
- If you add a number and its 1’s complement ...

\[
\begin{array}{c}
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
+ & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\]
COMPLEMENTS

• **Radix Complement, the r's Complement**
  
  Given a Number = \(N\), base = \(r\), digits = \(n\),
  
  – The \(r\)'s complement of \(N\) is defined as:
    
    \[ r^n - N \text{, for } N \neq 0 \text{ and } 0 \text{ for } N = 0. \]

• The \(r\)'s complement is obtained by adding 1 to the \((r - 1)\)'s complement, since
  
  \[
  \begin{align*}
  \text{– } &= [(r^n - 1) - N] + 1. \\
  \text{– } &= r^n - N
  \end{align*}
  \]
COMPLEMENTS

• 10’s complement of $N$ can be formed:
  – By leaving all least significant 0’s unchanged.
  – By subtracting first non-zero least significant digit by 10
  – By subtracting all higher significant digits from 9.

• 2’s complement of $N$ can be formed:
  – By leaving all least significant 0’s and the first 1 unchanged and replacing 1’s with 0’s and 0’s with 1’s in all other higher significant digits.
COMPLEMENTS

• Example: Base-10
  – The 10's complement of 012398:
    
    \[
    \begin{array}{cccccc}
    9 & 9 & 9 & 9 & 9 & 10 \\
    - & 0 & 1 & 2 & 3 & 9 & 8 \\
    \hline
    9 & 8 & 7 & 6 & 0 & 2
    \end{array}
    \]
  – The 10's complement of 246700:
    
    \[
    \begin{array}{cccccc}
    9 & 9 & 9 & 10 & 0 & 0 \\
    - & 2 & 4 & 6 & 7 & 0 & 0 \\
    \hline
    7 & 5 & 3 & 3 & 0 & 0
    \end{array}
    \]
COMPLEMENTS

• 2's Complement (Radix Complement)
  – Take 1’s complement then add 1
  – Toggle all bits to the left of the first ‘1’ from the right

*Example:*
Number: 1 0 1 1 0 0 0 0
1’s Comp.: 0 1 0 0 1 1 1 1

\[ \begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & & & & \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
\end{array} \]

or

\[ \begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & & & & \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
\end{array} \]
COMPLEMENTS

- Example: Base-2
  - The 2's complement of 1101100 is 0010100

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 \\
+ & 1 & & & & & \\
\hline
0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
\]

or

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
\]
COMPLEMENTS

• Example: Base-2
  – The 2's complement of 0110111 is 1001001
COMPLEMENTS

• Complement of the complement restores the number to its original value
  – r’s complement of N is \( r^n - N \)
  – The complement of the complement is
  \[ r^n - (r^n - N) = N \rightarrow \text{original number} \]
The subtraction of two $n$-digit unsigned numbers $M - N$ in base $r$ can be done as follows:

1. Add the minuend $M$ to the r’s complement of the subtrahend $N$.
   
   - N r’s complement = $r^n - N$
   
   - $= M + r^n - N$
   
   - $= M - N + r^n$

2. If $M \geq N$, the sum will produce an end carry $r^n$, which can be discarded; what is left is the result $M - N$.

3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r’s complement of $(N - M)$. To obtain the answer in a familiar form, take the r’s complement of the sum and place negative sign in front.
• Example 1.5
  – Using 10's complement, subtract 72532 – 3250.
  \[ M = 72532 \quad N = 3250 \]
  Step1: Take 10’s complement of N

\[
\begin{array}{cccccc}
9 & 9 & 9 & 1 & 0 & 0 \\
- & 3 & 2 & 5 & 0 \\
\hline
9 & 6 & 7 & 5 & 0
\end{array}
\]
Step 2: To find the SUM, Add 10’s complement of N to M

```
72532
+ 96750
---
169282
```

Sum = 169282

Step 3: To find the ANSWER discard end carry $10^5$, Subtract $r^n$ from SUM

```
169282
- 100000
---
69282
```

Answer = 69282
COMPLEMENT

- Example 1.6
  - Using 10's complement, subtract 3250 - 72532.
  
  \[ M = 3250 \quad N = 72532 \]
  
  Step1: Take 10’s complement of N

\[
\begin{array}{cccccc}
9 & 9 & 9 & 9 & 1 & 0 \\
- & 7 & 2 & 5 & 3 & 2 \\
\hline
2 & 7 & 4 & 6 & 8
\end{array}
\]
Step 2: To find the SUM, Add 10’s complement of N to M

\[
\begin{array}{c}
03250 \\
+ 27468 \\
\hline
30718
\end{array}
\]

SUM = 30718

There is no end carry!

Step 3: To find the ANSWER take the \(-(10’s \text{ complement of SUM})\)

\[
\begin{array}{c}
9 9 9 9 1 0 \\
- 3 0 7 1 8 \\
\hline
6 9 2 8 2
\end{array}
\]

ANSWER = -6 9 2 8 2
Example 1.7: Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction (a) $X - Y$ and (b) $Y - X$ using 2’s complements.

(a) Step 1: Take the 2’s complement of $Y$

```
 1 0 0 0 0 1 1
```

Step 2: Add 2’s complement of $Y$ to $X$

```
 1 0 1 0 1 0 0
+ 0 1 1 1 1 1 0 1
```

```
1 0 0 1 0 0 0 1
```

There is an end carry!
Step 3: Discard the end carry.

\[
\begin{align*}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
+ & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
& 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{align*}
\]

ANSWER = 0 0 0 1 0 0 0 1

(b) \( Y - X \): Step 1: Take the 2’s complement of \( X \)

\[
\begin{align*}
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0
\end{align*}
\]
Step 2: Add 2’s complement of X to Y

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 1 & 1 & \\
+ & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1
\end{array}
\]

There is no end carry!

SUM =

Step 3: To find the answer Y - X = - (2’s complement of SUM)

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 1 & 1 & \\
\circulo & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}
\]
COMPLEMENTS

- Subtraction of unsigned numbers can also be done by means of the \((r - 1)\)'s complement. Remember that the \((r - 1)\)'s complement is one less than the \(r\)'s complement.

Example 1.8: Repeat Example 1.7, but this time using 1's complement.

(a) \(X - Y = 1010100 - 1000011 (84 - 67 = 17)\)

Step 1: Take the 1’s complement of \(Y\)

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
\end{array}
\]
COMPLEMENTS

Step 2: Add 1’s complement of Y to the X

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
+ & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
\hline
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

SUM = 16

Step 3: Remove the end carry and add 1 (End-around carry)

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
+ & 1 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 1 & \end{array}
\]

ANSWER = 17
COMPLEMENTS

Example 1.8: Repeat Example 1.7, but this time using 1's complement.
(a) $Y - X = 1000011 - 1010100$ ($67 - 84 = -17$)
Step 1: Take the 1’s complement of $X$

\[
\begin{align*}
1 & \ 0 & \ 1 & \ 0 & \ 1 & \ 0 & \ 0 \\
0 & \ 1 & \ 0 & \ 1 & \ 0 & \ 1 & \ 1
\end{align*}
\]
COMPLEMENTS

Step 2: Add 1’s complement of X to the Y

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
+ & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
\hline
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{array}
\]

SUM = 11011100

There is NO end carry!

Step 3: To find the answer Y – X = - (1’s complement of SUM)

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 1 & 0 \\
\hline
\end{array}
\]

ANSWER = 00100001 = -17
1.7 SIGNED BINARY NUMBERS
SIGNED BINARY NUMBERS

• To represent negative integers, we need a notation for negative values.
• It is customary to represent the sign with a bit placed in the left most position of the number since binary digits.
• The convention is to make the sign bit  – 0 for positive and 1 for negative.
• Example
  – Signed-magnitude representation: 10001001
  – Signed-1’s complement representation: 11110110
  – Signed-2’s complement representation: 11110111
• Table 1.3 lists all possible four-bit signed binary numbers in the three representations.
# Signed Binary Numbers

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Signed-Z's Complement</th>
<th>Signed-1's Complement</th>
<th>Signed Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>+7</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
</tr>
<tr>
<td>+6</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
</tr>
<tr>
<td>+5</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>+4</td>
<td>0100</td>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>+3</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
</tr>
<tr>
<td>+2</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
</tr>
<tr>
<td>+1</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>+0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>-0</td>
<td>-</td>
<td>1111</td>
<td>1000</td>
</tr>
<tr>
<td>-1</td>
<td>1111</td>
<td>1110</td>
<td>1001</td>
</tr>
<tr>
<td>-2</td>
<td>1110</td>
<td>1101</td>
<td>1010</td>
</tr>
<tr>
<td>-3</td>
<td>1101</td>
<td>1100</td>
<td>1011</td>
</tr>
<tr>
<td>-4</td>
<td>1100</td>
<td>1011</td>
<td>1100</td>
</tr>
<tr>
<td>-5</td>
<td>1011</td>
<td>1010</td>
<td>1101</td>
</tr>
<tr>
<td>-6</td>
<td>1010</td>
<td>1001</td>
<td>1110</td>
</tr>
<tr>
<td>-7</td>
<td>1001</td>
<td>1000</td>
<td>1111</td>
</tr>
<tr>
<td>-8</td>
<td>1000</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
SIGNED BINARY NUMBERS

• Arithmetic addition
  – The addition of two numbers in the signed-magnitude system follows the rules of ordinary arithmetic.
    • If the signs are the same;
      – we add the two magnitudes and give the sum the common sign.
    • If the signs are different;
      – we subtract the smaller magnitude from the larger and give the difference the sign of the larger magnitude.
SIGNED BINARY NUMBERS

- The addition of two signed binary numbers with negative numbers represented in signed-2's-complement form is obtained from the addition of the two numbers, including their sign bits.
- A carry out of the sign-bit position is discarded.

• Example:

\[
\begin{array}{cccccccc}
+ & 6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
+ & 13 & + & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
+ & 19 & + & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
- & 6 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
+ & 13 & + & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
+ & 7 & + & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]
SIGNED BINARY NUMBERS

Example:

\[ \begin{array}{c c}
+ & 6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
- & 13 & + & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
- & 7 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array} \]

\[ \begin{array}{c c}
- & 6 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
- & 13 & + & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
- & 19 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
\end{array} \]
SIGNED BINARY NUMBERS

• Arithmetic Subtraction
  – In 2's-complement form:
    1. Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including sign bit).
    2. A carry out of sign-bit position is discarded.

\[(\pm A) - (+B) = (\pm A)\pm (-B)\]
\[(\pm A) - (-B) = (\pm A)\pm (+B)\]
SIGNED BINARY NUMBERS

- Example:
  - \((-6) - (-13) \Rightarrow (11111010 - 11110011)\)

\[
\begin{array}{c}
-6 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
+13 & -0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
+7 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
\]
1.8 BINARY CODES
BINARY CODES

• BCD Code
  – A number with k decimal digits will require 4k bits in BCD.
  – Decimal 396 is represented in BCD with 12 bits as:
    \[ \begin{array}{lllllllllll}
    & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
    \end{array} \]
    – Each group of 4 bits representing one decimal digit.
A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.

The binary combinations 1010 through 1111 are not used and have no meaning in BCD.
**Example:**

- Consider decimal 185 and its corresponding value in BCD and binary:

\[
\begin{align*}
(185)_{10} & = (000110000101)_{\text{BCD}} \\
& = (10111001)_{2}
\end{align*}
\]
BCD Addition

- Consider the addition of two decimal digits in BCD
- Since each digit does not exceed 9
- The sum cannot be greater than $9 + 9 + 1 = 19$
  - the 1 in the sum being a previous carry
- Suppose we add the BCD digits as if they were binary numbers
- The binary sum will produce a result in the range from 0 to 19
- In binary this will be from 0000 to 10011, but in BCD it is from 0000 to 1 1001
  - The first 1 being a carry and the next four bits being the BCD digit sum
• BCD Addition
  – When binary sum is equal to or less than 1001 (without a carry)
    • The corresponding BCD digit is correct
  – When the binary sum is greater than or equal to 1010
    • The result is an invalid BCD digit
  – The addition of \(6 = (0110)_2\) to the binary sum converts it to the correct digit and also produces a carry as required
  – This is because the difference between a carry in the most significant bit position of the binary sum and a decimal carry.
    • \(16 - 10 = 6\)
• BCD Addition

  – Consider the following three addition of two decimal digits in BCD

```
  4 0 1 0 0 0
+  5 0 1 0 1
  9 1 0 0 1

  4 0 1 0 0 0
+  8 1 0 0 1
  12 1 1 0 0
  +  0 1 1 0
  1 0 0 1 0

  8 1 0 0 0 0
+  9 1 0 0 1
  17 1 0 0 1
  +  0 1 1 0
  1 0 1 1 1
```

Correct BCD digit sum (2)  Carry
Correct BCD digit sum (7)  Carry
BINARY CODES

• The addition of two n-digit unsigned BCD numbers follows the same procedure.
  – Consider the addition of $184 + 576 = 760$ in BCD

  \[
  \begin{array}{c}
  \text{Binary Sum} \\
  0 1 1 1 1 \\
  + \ 0 1 0 1 1 \\
  \hline
  1 0 0 0 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{Add 6} \\
  1 1 1 0 0 \\
  + \ 0 1 1 1 0 \\
  \hline
  1 1 1 1 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{BCD Sum} \\
  0 1 1 1 1 \\
  + \ 0 1 1 0 1 \\
  \hline
  1 0 0 1 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{BCD Sum} \\
  0 1 1 0 0 \\
  + \ 0 1 1 1 0 \\
  \hline
  1 0 0 1 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{BCD Sum} \\
  0 1 1 1 0 \\
  + \ 0 1 1 1 0 \\
  \hline
  1 0 0 0 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{BCD Sum} \\
  0 1 1 1 1 \\
  + \ 0 1 1 1 1 \\
  \hline
  1 0 0 1 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{BCD Sum} \\
  0 1 1 1 0 \\
  + \ 0 1 1 1 0 \\
  \hline
  1 0 0 0 0 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c}
  \text{BCD Sum} \\
  0 1 1 1 0 \\
  + \ 0 1 1 1 0 \\
  \hline
  1 0 0 0 0 \\
  \end{array}
  \]
The representation of signed decimal numbers in BCD is similar to the representation of signed numbers in binary.

We can use either the familiar sign and magnitude system or the signed-complement system.

The sign of a decimal number is usually represented with four bits to conform to the 4-bit code of the decimal digits.

- “+” with 0 0 0 0 and “-” with 1001 (BCD equivalent of 9)
The signed-complement system can be either the 9’s or the 10’s complement.

- But the 10’s complement is the one most often used.

To obtain the 10’s complement of a BCD number:

- First take the 9’s complement
  - 9’s complement is calculated from the subtraction of each digit from 9.
  - Then add one to the least significant digit
BINARY CODES

• The procedures developed for the signed-2’s complement system in the previous section apply also to the signed-10’s complement system for decimal numbers.

• Addition is done by:
  – adding all digits,
  – including the sign digit and
  – discarding the end carry.

• This assumes that all negative numbers are in 10’s complement form.
BINARY CODES

• Consider the addition (+375) + (-240) = +135

• Step1: Find the 10’s complement of (-240)

\[
\begin{array}{c}
9 \\
1 \\
0 \\
0 \\
\hline
2 \\
4 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
7 \\
6 \\
0 \\
\end{array}
\]

• Step 2: Add 10’s complement of (-240) to 375 and discard the end carry.

\[
\begin{array}{c}
0 \\
3 \\
7 \\
5 \\
\hline
9 \\
7 \\
6 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
1 \\
3 \\
5 \\
\end{array}
\]
# Binary Codes

## Table 1.5

*Four Different Binary Codes for the Decimal Digits*

<table>
<thead>
<tr>
<th>Decimal Digit</th>
<th>BCD 8421</th>
<th>2421</th>
<th>Excess-3</th>
<th>8, 4, −2, −1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0011</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0001</td>
<td>0100</td>
<td>0111</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0010</td>
<td>0101</td>
<td>0110</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0011</td>
<td>0110</td>
<td>0101</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0100</td>
<td>0111</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>1011</td>
<td>1000</td>
<td>1011</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>1100</td>
<td>1001</td>
<td>1010</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>1101</td>
<td>1010</td>
<td>1001</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1110</td>
<td>1011</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1111</td>
<td>1100</td>
<td>1111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unused bit combinations</th>
<th>1010</th>
<th>0101</th>
<th>0000</th>
<th>0001</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1011</td>
<td>0110</td>
<td>0001</td>
<td>0010</td>
</tr>
<tr>
<td></td>
<td>1100</td>
<td>0111</td>
<td>0010</td>
<td>0011</td>
</tr>
<tr>
<td></td>
<td>1101</td>
<td>1000</td>
<td>1101</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>1110</td>
<td>1001</td>
<td>1110</td>
<td>1101</td>
</tr>
<tr>
<td></td>
<td>1111</td>
<td>1010</td>
<td>1111</td>
<td>1110</td>
</tr>
</tbody>
</table>
Gray code:
- The output data of many physical systems produce quantities that are continuous.
- These data must be converted into digital form before they are applied to a digital system.
- Continuous or analog information is converted into digital form by means of an analog-to-digital converter.
• Gray code:
  – It is convenient to use gray code to represent the digital data when it is converted from analog data.
  – The advantage is that only bit in the code group changes in going from one number to the next.
    • Error detection.
    • Representation of analog data.
    • Low power design.

<table>
<thead>
<tr>
<th>Gray Code</th>
<th>Decimal Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0011</td>
<td>2</td>
</tr>
<tr>
<td>0010</td>
<td>3</td>
</tr>
<tr>
<td>0110</td>
<td>4</td>
</tr>
<tr>
<td>0111</td>
<td>5</td>
</tr>
<tr>
<td>0101</td>
<td>6</td>
</tr>
<tr>
<td>0100</td>
<td>7</td>
</tr>
<tr>
<td>1100</td>
<td>8</td>
</tr>
<tr>
<td>1101</td>
<td>9</td>
</tr>
<tr>
<td>1111</td>
<td>10</td>
</tr>
<tr>
<td>1110</td>
<td>11</td>
</tr>
<tr>
<td>1010</td>
<td>12</td>
</tr>
<tr>
<td>1011</td>
<td>13</td>
</tr>
<tr>
<td>1001</td>
<td>14</td>
</tr>
<tr>
<td>1000</td>
<td>15</td>
</tr>
</tbody>
</table>

6 October, 2016
BINARY CODES

• American Standard Code for Information Interchange (ASCII) Character Code (Refer to Table 1.7)
• A popular code used to represent information sent as character-based data.
• It uses 7-bits to represent 128 characters:
  – 94 Graphic printing characters.
  – 34 Non-printing characters.
• Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return).
• Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).
### Table 1.7
American Standard Code for Information Interchange (ASCII)

<table>
<thead>
<tr>
<th>$b_7b_6b_5$</th>
<th>$b_4b_3b_2b_1$</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>`</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>SOH</td>
<td>DC1</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>STX</td>
<td>DC2</td>
<td>`</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>ETX</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>EOT</td>
<td>DC4</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>ENQ</td>
<td>NAK</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>ACK</td>
<td>SYN</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>BEL</td>
<td>ETB</td>
<td>,</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>BS</td>
<td>CAN</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>HT</td>
<td>EM</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>LF</td>
<td>SUB</td>
<td>*</td>
<td>:</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>VT</td>
<td>ESC</td>
<td>+</td>
<td>;</td>
<td>K</td>
<td>[</td>
<td>k</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>FF</td>
<td>FS</td>
<td>.</td>
<td>&lt;</td>
<td>L</td>
<td>\</td>
<td>l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>CR</td>
<td>GS</td>
<td>-</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td>m</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>SO</td>
<td>RS</td>
<td>.</td>
<td>&gt;</td>
<td>N</td>
<td>\</td>
<td>n</td>
<td>~</td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>SI</td>
<td>US</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>-</td>
<td>o</td>
<td>DEL</td>
<td></td>
</tr>
</tbody>
</table>
# Binary Codes

## Control Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>Null</td>
</tr>
<tr>
<td>SOH</td>
<td>Start of heading</td>
</tr>
<tr>
<td>STX</td>
<td>Start of text</td>
</tr>
<tr>
<td>ETX</td>
<td>End of text</td>
</tr>
<tr>
<td>EOT</td>
<td>End of transmission</td>
</tr>
<tr>
<td>ENQ</td>
<td>Enquiry</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>BEL</td>
<td>Bell</td>
</tr>
<tr>
<td>BS</td>
<td>Backspace</td>
</tr>
<tr>
<td>HT</td>
<td>Horizontal tab</td>
</tr>
<tr>
<td>LF</td>
<td>Line feed</td>
</tr>
<tr>
<td>VT</td>
<td>Vertical tab</td>
</tr>
<tr>
<td>FF</td>
<td>Form feed</td>
</tr>
<tr>
<td>CR</td>
<td>Carriage return</td>
</tr>
<tr>
<td>SO</td>
<td>Shift out</td>
</tr>
<tr>
<td>SI</td>
<td>Shift in</td>
</tr>
<tr>
<td>SP</td>
<td>Space</td>
</tr>
<tr>
<td>DLE</td>
<td>Data-link escape</td>
</tr>
<tr>
<td>DC1</td>
<td>Device control 1</td>
</tr>
<tr>
<td>DC2</td>
<td>Device control 2</td>
</tr>
<tr>
<td>DC3</td>
<td>Device control 3</td>
</tr>
<tr>
<td>DC4</td>
<td>Device control 4</td>
</tr>
<tr>
<td>NAK</td>
<td>Negative acknowledge</td>
</tr>
<tr>
<td>SYN</td>
<td>Synchronous idle</td>
</tr>
<tr>
<td>ETB</td>
<td>End-of-transmission block</td>
</tr>
<tr>
<td>CAN</td>
<td>Cancel</td>
</tr>
<tr>
<td>EM</td>
<td>End of medium</td>
</tr>
<tr>
<td>SUB</td>
<td>Substitute</td>
</tr>
<tr>
<td>ESC</td>
<td>Escape</td>
</tr>
<tr>
<td>FS</td>
<td>File separator</td>
</tr>
<tr>
<td>GS</td>
<td>Group separator</td>
</tr>
<tr>
<td>RS</td>
<td>Record separator</td>
</tr>
<tr>
<td>US</td>
<td>Unit separator</td>
</tr>
<tr>
<td>DEL</td>
<td>Delete</td>
</tr>
</tbody>
</table>
BINARY CODES

• ASCII has some interesting properties:
  – Digits 0 to 9 span Hexadecimal values $30_{16}$ to $39_{16}$
  – Upper case A-Z span $41_{16}$ to $5A_{16}$
  – Lower case a-z span $61_{16}$ to $7A_{16}$
    • Lower to upper case translation (and vice versa) occurs by flipping bit 6.
BINARY CODES

• Error-Detecting Code
  – To detect errors in data communication and processing, an **eighth bit** is sometimes added to the ASCII character to indicate its parity.
  – A **parity bit** is an extra bit included with a message to make the total number of 1's either even or odd.

• Example:
  – Consider the following two characters and their even and odd parity:

<table>
<thead>
<tr>
<th></th>
<th>With even parity</th>
<th>With odd parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII A = 1000001</td>
<td>01000001</td>
<td>11000001</td>
</tr>
<tr>
<td>ASCII T = 1010100</td>
<td>11010100</td>
<td>01010100</td>
</tr>
</tbody>
</table>
• Error-Detecting Code
  – **Redundancy** (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
  – A simple form of redundancy is **parity**, an extra bit appended onto the code word to make the number of 1’s odd or even. Parity can detect all single-bit errors and some multiple-bit errors.
  – A code word has **even parity** if the number of 1’s in the code word is even.
  – A code word has **odd parity** if the number of 1’s in the code word is odd.
  – Example:
    • Message A: 10001001 1 (even parity)
    • Message B: 10001001 0 (odd parity)
1.9 BINARY STORAGE & REGISTERS
• Registers
  – A **binary cell** is a device that possesses two stable states and is capable of storing one of the two states.

  ![1/0](image)

  – A **register** is a group of binary cells. A register with \( n \) cells can store any discrete quantity of information that contains \( n \) bits.

  ![01101011](image)

  \( n \) cells \( \rightarrow 2^n \) possible states
BINARY STORAGE & REGISTERS

- A binary cell
  - Two stable state
  - Store one bit of information
  - Examples: flip-flop circuits, ferrite cores, capacitor

- A register
  - A group of binary cells
  - AX in x86 CPU

- Register Transfer
  - A transfer of the information stored in one register to another.
  - One of the major operations in digital system.
  - An example in next slides.
Inputs: Keyboard, mouse, modem, microphone

Outputs: CRT, LCD, modem, speakers
Figure 1.1 Transfer of information among register
The other major component of a digital system
- Circuit elements to manipulate individual bits of information
- Load-store machine
  - LD R1;
  - LD R2;
  - ADD R3, R2, R1;
  - SD R3;

Figure 1.2 Example of binary information processing
1.10 BINARY LOGIC
BINARY LOGIC

• Definition of Binary Logic
  – Binary logic consists of binary variables and a set of logical operations.
  – The variables are designated by letters of the alphabet, such as $A$, $B$, $C$, $x$, $y$, $z$, etc., with each variable having two and only two distinct possible values: 1 and 0,
  – Three basic logical operations: AND, OR, and NOT.
1. **AND** operation is represented by a dot (\( \cdot \)) or by the absence of an operator.
   - Example: \( X \cdot Y = Z \) or \( X Y = Z \)
     - “\( X \text{ AND } Y \) is equal to \( Z \)”.
     - \( Z = 1 \) if only \( X = 1 \) and \( Y = 1 \); otherwise \( Z = 0 \).
       - \( X, Y \) and \( Z \) are binary variables and can be equal either to 1 or 0, nothing else.
2. **OR** operation is represented by a plus (+).

   - Example: \( X + Y = Z \)
     - “\( X \) **OR** \( Y \) is equal to \( Z \).”
     - \( Z = 1 \) if \( X = 1 \) **OR** \( Y = 1 \) or if both \( X = 1 \) **OR** \( Y = 1 \).
     - If both \( X = 0 \) **OR** \( Y = 0 \), then \( Z = 0 \).
3. **NOT** operation is represented by a prim (‘) sometimes by an overbar (\(\overline{\text{-}}\))

- Example: \(X' = Z\) or \(\overline{X} = Z\)
  - “**NOT** \(X\) is equal to \(Z\)”.  
  - If \(X = 1\), then \(Z = 0\), but if \(X = 0\), then \(Z = 1\).  
  - The **NOT** operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.
**BINARY LOGIC**

- Truth tables, Boolean expressions and Logic Gates

### AND

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Z = X \cdot Y \]

### OR

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Z = X + Y \]

### NOT

<table>
<thead>
<tr>
<th>X</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \overline{X} = Z \]
BINARY LOGIC

AND

OR
**Example of binary signals**

<table>
<thead>
<tr>
<th>Volts</th>
<th>Signal range for logic 1</th>
<th>Transition occurs between these limits</th>
<th>Signal range for logic 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Logic 1
- Logic 0
- Undefined
BINARY LOGIC

• Graphic Symbols and Input-Output Signals for Logic gates:

(a) two input AND gate

(b) two input OR gate

(c) NOT gate or Inverter
BINARY LOGIC

- Graphic Symbols and Input-Output Signals for many input logic gates:

(a) three input AND gate

\[ F = A \cdot B \cdot C \]

(b) four input OR gate

\[ F = A + B + C + D \]

6 October, 2016

INTRODUCTION TO LOGIC DESIGN