EE2020

Digital Fundamentals

(L1 - Number Systems)

Massimo ALIOTO

Dept of Electrical and Computer Engineering

Email: ***massimo.alioto@nus.edu.sg***

Modified from original slides by Prof. XU Yong Ping

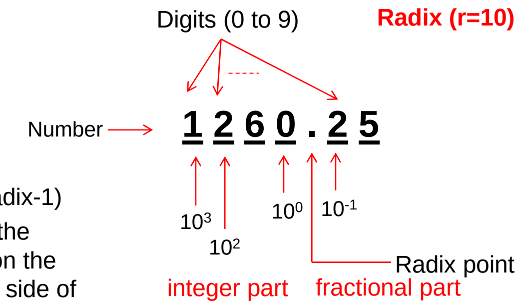
Outline

- Positional number system
- Radix conversion
- Binary arithmetic
- Binary signed representation
- Binary-coded decimal (BCD)

Positional Number System

Decimal number:

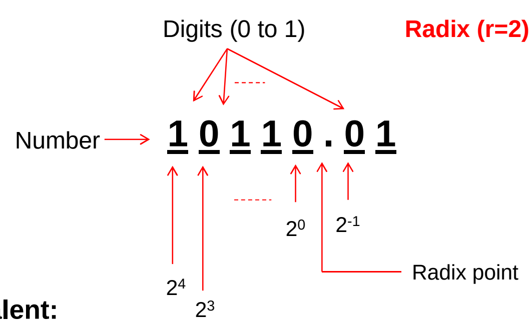
- **Terminologies**
  - Radix (or base)
  - Radix point
  - Digits and a numeral (0 → radix-1)
  - Place value (or weight) is in the power of the base (positive on the left and negative on the right side of the radix point



$$N = 1\times10^3 + 2\times10^2 + 6\times10^1 + 0\times10^0 + 2\times10^{-1} + 5\times10^{-2} = 1260.25$$

*Weighted sum of each digit (each digit is weighted by its place value)

Binary number



Decimal Equivalent:

$$N_{10} = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}$$

$$= 16 + 0 + 4 + 2 + 0 + 0 + \frac{1}{4}$$

$$= 22.25$$

$(10110.01)_2$ = $(22.75)_{10}$

Hexadecimal number

Radix (r=16)

Digits (0 to 15)

Number → **1 8 F 4 . 2 A**

16^3 16^2 ------ 16^0 16^{-1} 16^{-2}

Radix point

Decimal Equivalent:

$$N_{10} = 1\times16^3 + 8\times16^2 + F\times16^1 + 4\times16^0 + 2\times16^{-1} + 10\times16^{-2}$$

$$= 4096 + 2048 + 240 + 4 + \frac{2}{16} + \frac{10}{256}$$

$$= 6388 + \frac{21}{128}$$

$$\approx 6388.16$$

$(18F4.2A)_{16} \cong (6388.16)_{10}$

Hex	Dec
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Octal number

Radix (r=8)

Digits (0 to 7)

Number → **7 5 4 . 2**

8^2 ------ 8^0 8^{-1}

Radix point

$(754.2)_8 = (520.25)_{10}$

Decimal Equivalent:

$$N_{10} = 7\times8^2 + 5\times8^1 + 4\times8^0 + 2\times8^{-1}$$

$$= 448 + 40 + 4 + \frac{2}{8}$$

$$= 492.25$$

Oct	Dec
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
?	8
?	9
?	10

General form of A Number of radix r and its Decimal Equivalent

General form of Number of radix r:

Radix point

$$A_r = (a_n a_{n-1} \dots a_o . a_{-1} \dots a_{-m})_r$$

$$where\ a_n, a_{n-1}, \dots, a_0, ..a_{-m} \in \{0, \dots (r-1)\}$$ (Integer only)

Decimal equivalent:

Radix point is here

$$\begin{aligned} A_r &= (a_n a_{n-1} \dots a_o . a_{-1} \dots a_{-m})_r \\ &= a_n \times r^n + a_{n-1} \times r^{n-1} + \dots a_o \times r^0 + a_{-1} \times r^{-1} + \dots a_{-m} \times r^{-m} \\ &= \sum_{i=-m}^{n} a_i r^i \end{aligned}$$

***Weighted sum of all digits**

Radix Conversion

Three types of conversions:

- Radix r (r≠10) → Decimal
- Decimal → Radix r (r≠10)
- Conversion among Binary, Octal and Hex numbers

Radix r (r $\neq$ 10) → Decimal

Binary → Decimal **$(10110.01)_2 = (??)_{10}$**

$$(10110.01)_2 \rightarrow 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} = (22.25)_{10}$$

Hex → Decimal **$(18F4.2A)_{16} = (??)_{10}$**

$$\begin{aligned}(18F4.2A)_{16} &= 1\times 16^3 + 8\times 16^2 + F\times 16^1 + 4\times 16^0 + 2\times 16^{-1} + 10\times 16^{-2}\\ &\approx (6388.16)_{10}\end{aligned}$$

***Compute the weighted sum of all digits**

$$\begin{aligned}A_r &= (a_n a_{n-1} \ldots a_o . a_{-1} \ldots a_{-m})_r\\ &= a_n \times r^n + a_{n-1}\times r^{n-1} + \ldots a_o \times r^0 + a_{-1}\times r^{-1} + \ldots a_{-m}\times r^{-m}\\ &= \sum_{i=-m}^{n} a_i r^i\end{aligned}$$

Decimal → Radix r (r $\neq$ 10)

Decimal → Binary **$(102)_{10} = (??)_2$**

$$\begin{aligned}(102)_{10} = A_2 &= (a_n a_{n-1} \ldots a_o . a_{-1} \ldots a_{-m})_r\\ &= a_n \times 2^n + a_{n-1}\times 2^{n-1} + \ldots + a_1 \times 2^1 + a_o\\ &= \underbrace{(a_n \times 2^n + a_{n-1}\times 2^{n-1} + \ldots + a_1\times 2^1)}_{\text{Integer multiple of 2}} + a_o\end{aligned}$$

(Assume integer)

$$\frac{(102)_{10}}{2} \rightarrow$$

Continue dividing quotient by 2

quotient $a_n \times 2^{n-1} + a_{n-1}\times 2^{n-2} + \ldots + a_1$

$$2 \,\big)\, a_n \times 2^n + a_{n-1}\times 2^{n-1} + \ldots + a_1 \times 2 + a_o$$

$$a_n \times 2^n + a_{n-1}\times 2^{n-1} + \ldots + a_1 \times 2$$

a_o

Remainder is a_0

$a_n \times 2^{n-2} + a_{n-1}\times 2^{n-3} + \ldots + a_1$

$$2 \,\big)\, a_n \times 2^{n-1} + a_{n-1}\times 2^{n-2} + \ldots + a_1$$

$$a_n \times 2^{n-1} + a_{n-1}\times 2^{n-2} + \ldots$$

a_1

Remainder is a_1

Decimal → Radix r (r ≠ 10) – cont.

Decimal → Binary **$(102)_{10} = (??)_2$**

Division	Quotient	Remainder
102/2	51	0 → a_0
51/2	25	1 → a_1
25/2	12	1 → a_2
12/2	6	0 → a_3
6/2	3	0 → a_4
3/2	1	1 → a_5
1/2	0	1 → a_6

Stop when the quotient = 0

$(102)_{10} = (1100110)_2$

Check:

$$
\begin{aligned}
N_{10} &= a_6 \times 2^6 + a_5 \times 2^5 + a_4 \times 2^4 + a_3 \times 2^3 \\
&\quad + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0 \\
&= 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 \\
&\quad + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\
&= 64 + 32 + 0 + 0 + 4 + 2 + 0 \\
&= 102
\end{aligned}
$$

How about fraction?

Decimal → Binary **$(0.58)_{10} = (??)_2$**

$$
\begin{aligned}
(0.58)_{10} &= A_2 = (0.a_{-1}a_{-2}\ldots a_{-m+1}a_{-m})_r \\
&= a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} + \ldots + a_{-m+1} \times 2^{-m+1} + a_{-m} \times 2^{-m}
\end{aligned}
$$

Multiply by 2:

$$
(0.58)_{10} \times 2 = \underset{\uparrow}{a_{-1}} + \underbrace{a_{-2} \times 2^{-1} + \ldots + a_{-m+1} \times 2^{-m+2} + a_{-m} \times 2^{-m+1}}_{\text{fractional part}}
$$

Integer part is a_{-1}

How about fraction? – cont.

Decimal → Binary $(0.58)_{10} = (??)_2$

Multiply by 2	Product	Integer Part
0.58x2	1.16	1 → a_{-1}
0.16x2	0.32	0 → a_{-2}
0.32x2	0.64	0 → a_{-3}
0.64x2	1.28	1 → a_{-4}
0.28x2	0.56	0 → a_{-5}
0.56x2	1.12	1 → a_{-6}
0.12x2	0.24	0 → a_{-7}
0.24x2	0.48	0 → a_{-8}

$(0.58)_{10} = (0.100101)_2$

Check:

$$N_{10} = 1\times 2^{-1} + 1\times 2^{-4} + 1\times 2^{-6}$$
$$= \frac{1}{2} + \frac{1}{16} + \frac{1}{64}$$
$$= 0.578125$$
$$\approx 0.58$$

- The conversion process may never end.
- Where to stop depends on the required precision
- The process only ends when fractional part = 0

Numbers with Different Radixes

Numbers with Different Radixes

Decimal (radix 10)	Binary (radix 2)	Octal (radix 8)	Hexadecimal (radix 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversion among Hex, Octal and Binary

- Hex ←→ Binary
  - Each Hex digit → 4 Binary bits (digits)
  - Or each 4 Binary bits → 1 Hex digit (starting from radix point)
- Octal ←→ Binary
  - Each octal digit → 3 Binary bits
  - Or each 3 Binary bits → 1 Octal digit (starting from radix point)
- Hex ←→ Octal
  - Use Binary as an intermediate step
  - Hex → Binary → Octal
  - Octal → Binary → Hex

Examples (Hex, Octal, Binary)

Hex → Bin:

$(A45F)_{16}$

$(\underline{1010}\ \underline{0100}\ \underline{0101}\ \underline{1111})_2$

Bin → Hex:

$(\underline{11}\ \underline{1010}\ \underline{1101}\ \underline{0111})_2$

$(3\ A\ D\ 7)_{16}$

Oct → Bin:

$(475)_8$

$(\underline{100}\ \underline{111}\ \underline{101})_2$

Bin → Oct:

$(\underline{10}\ \underline{111}\ \underline{101}\ \underline{110})_2$

$(2\ 7\ 5\ 6)_8$

More Examples

Hex → Oct:

$(A45F)_{16}$

⇩

$(1\ 010\ 010\ 001\ 011\ 111)_2$

$(122137)_8$

Oct → Hex:

$(653)_8$

⇩

$(1\ 1010\ 1011)_2$

$(1\ A\ B)_{16}$

For the fractional part: very similar, just group digits by starting from the position after the radix point

Hex → Bin:

$(0\ .\ A45F)_{16}$

⇩

$(0\ .\ 1010\ 0100\ 0101\ 1111)_2$

Radix Conversion (recap)

- Radix r (r≠10) → Decimal
  - Compute weighted sum of all digits
- Decimal → Radix r (r≠10)
  - Integer → Divided by r and take the remainder
  - Fraction → Multiply by r and take the integer
  - Add integer and fraction parts
- Conversion among Binary, Octal and Hex numbers
  - 1 Hex digit = 4 Binary and 1 Oct = 3 Binary, vice versa
  - Hex→Oct: Hex→Binary→Octal, and vice versa. Binary is used as an intermediate step

Binary Arithmetic

- Addition
- Multiplication
- Subtraction
- Division
- MSB and LSB
- Arithmetic using computer

Addition

Addition table:

0 + 0 = 0
0 + 1 = 1
1 + 1 = 10

"1" is the carry to the next higher bit

Example:

10111 + 110 = 11101

```
  1 1        ← Carry
   1 0 1 1 1
+      1 1 0
---------------
   1 1 1 0 1
```

Multiplication

Multiplication table:

0 x 0 = 0
0 x 1 = 0
1 x 1 = 1

Example:

10111 x 110 = 10001010

```
      1 0 1 1 1   <------   Multiplicand
   x      1 1 0   <------   Multiplier
   ---------------
      0 0 0 0 0  }
    1 0 1 1 1    }  <------ Partial products
+ 1 0 1 1 1      }
----------------------
  1 0 0 0 1 0 1 0  <------  Product
```

Multiplication:
→ Shift then Add
→ Only need "add" operation

Subtraction

Subtraction table:

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 – 1 = 1 ← with a borrow from the next (higher) bit

Example:

10111 - 110 = 10101

```
  1 1 0 1 1
-     1 1 0
---------------
  1 0 1 0 1
```

Division

100101/101 = ?

```
         0111   <——  quotient
101 ) 100101
 0
 1     101
       1000
 1      101
         111
 1       101
          10    <——  remainder
```

Check in decimal

- **Set** quotient to 0
- Align leftmost digits in dividend and divisor
- **Repeat**
  - **If** that portion of the dividend above the divisor is greater than or equal to the divisor
    - **Then** subtract divisor from that portion of the dividend and
    - Concatenate 1 to the right hand end of the quotient
    - **Else** concatenate 0 to the right hand end of the quotient
  - Shift the divisor one place right
- **Until** dividend is less than the divisor
- quotient is correct, dividend is remainder
- **STOP**

Division (shift and subtract)
→ Shift then subtraction
→ Only need "subtract" operation

Arithmetic using computer

- Only addition and subtraction are needed for 4 binary arithmetic operations
- Subtraction needs more elements than addition in hardware
- Subtraction can be performed by adding a negative number
- Thus, a computer may only use **adders** to perform all binary arithmetic operations
- This requires an appropriate representation of the negative binary numbers

Signed Binary numbers

- Three ways to represent the signed binary numbers
  - Signed binary (Sign + magnitude)
  - 1's complement
  - 2's complement

MSB and LSB of a Binary Number

- MSB
  - Most significant bit
- LSB
  - Least significant bit

1 0 1 1 0 0 1

(Left-most bit) MSB — LSB (Right-most bit)

***For integer binary number only**

Unsigned Binary number

Unsigned binary number (*n* bits)

1	1	0	1	…	…	1	0

Magnitude

(No sign, always positive)

Range of unsigned binary number:

Max value of a 4-bit number:

$$\overset{2^4\,2^3\,2^2\,2^1\,2^0}{1111 = 1\,0\,0\,0\,0 - 1} \rightarrow (2^4)_{10} - 1$$

Example:

Decimal	Signed binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Max value of *n*-bit unsigned number in decimal → $2^n - 1$. Range: $0 \sim (2^n-1)$

Signed Binary – Signed Magnitude (S-M)

Signed binary number (*n* bits)

	1	0	1	…	…	1	0

Magnitude

Sign bit { **0 – represents a positive number**
1 – represents a negative number

MSB is a sign bit

Example:

Decimal	S-M
3	011
2	010
1	001
+0	000
-0	**100**
-1	**101**
-2	**110**
-3	**111**

Note: Two zeros

Negative numbers

"1" in MSB position for all negative numbers

Signed Magnitude – cont.

More examples:

00111010 = +0111010 = $(58)_{10}$

11100101 = - 1100101 = $(-101)_{10}$

10000001 = - 0000001 = $(-1)_{10}$

01111111 = +1111111 = $(+127)_{10}$

Range of binary number represented by S-M:

For a *n*-bit Signed binary (S-M), its magnitude is 2^{n-1} bits.

Max magnitude: $(2^{n-1}-1)_{10}$
Range: $-(2^{n-1}-1)_{10} \sim +(2^{n-1}-1)_{10}$

Arithmetic using Binary Numbers (S-M)

- Computer performs binary arithmetic operations using only
  - Adders
  - Multipliers
- Subtraction is performed by adding a negative number

Examples of subtraction using S-M binary representation:

```
  3        011           3        011           2        010
- 2  ⇨   + 110         - 1  ⇨   + 101         - 1  ⇨   + 101
-------  -------     -------    -------     -------    -------
  1       1001 (1)10     1        1000 (0)10     1        111 (-3)10
          ↑     ✓                 ↑     ✘                     ✘
      Discarded               Discarded
```

***S-M representation cannot be used for addition of two number with opposite signs or subtraction when using a simple adder (dedicated hardware is needed for all possible sign combinations)**

Complement Representation

- **Complement representations of a number**
  - Radix complements
  - Diminished complements
- **Definitions:**

  \- ***Radix Complement***
  of a n-digit integer number A with radix (***r***):

$$A^* = r^n - A$$

  \- ***Diminished radix complement***
  of a n-digit integer number A with radix (***r***):

$$A^* = r^n - A - 1$$

Diminished Radix Complement

$$A^* = r^n - A - 1 \quad \text{or} \quad A^* = (r^n - 1) - A$$

Examples:

Decimal number:

A = 2375 → $A^* = (10000_{10} - 1) - 2375_{10} = 9999_{10} - 2375_{10} = 7624_{10}$

A = 0919 → $A^* = (10000_{10} - 1) - 0919_{10} = 9080_{10}$

Octal number:

A = 406 → $A^* = (1000_8 - 1) - 406_8 = 777_8 - 406_8 = 371_8$

A = 0671 → $A^* = (10000_8 - 1) - 0671_8 = 7777_8 - 0671_8 = 7106_8$

Hex number:

A = 4A09 → $A^* = (10000_{16} - 1) - 4A09_{16} = FFFF_{16} - 4A09_{16} = B5F6_{16}$

A = 0A7F → $A^* = (10000_{16} - 1) - 0A7F_{16} = FFFF_{16} - 0A7F_{16} = F580_{16}$

Binary number:

A = 1001 → $A^* = (10000_2 - 1) - 1001_2 = 1111_2 - 1001_2 = 0110_2$

A = 1100 → $A^* = (10000_2 - 1) - 1100_2 = 1111_2 - 1100_2 = 0011_2$

Diminished **radix 2** complement
is found by reversing the bits

1’s Complement

- “1’s Complement” is the *diminished radix complement* of binary numbers
- 1’s complement of a *n-bit* number is **$A^* = (2^n – 1) - A$**
- 1’s complement of a binary number can be obtained by reversing the bits, i.e. “1” → “0” and “0” → “1”, since

$$(2^n - 1)_{10} = \underbrace{1000\ldots000}_{n+1 \text{ bits}} - 1 = \underbrace{111\ldots111}_{n \text{ bits}}$$

Binary number (n=8): 01011100

1’s Complement: 11111111 - 01011100 = 10100011

Reversing the bits

1’s Complement representation of signed binary number

No change for positive numbers and use 1’s complement for negative numbers

Decimal	1’s Complement
3	011
2	010
1	001
+0	000
-0	**111**
-1	**110**
-2	**101**
-3	**100**

Still two zeros

Magnitude range: $-(2^{n-1}-1) \sim (2^{n-1}-1)$

3 – 2 = 3+(-2)=1

```
   011
+  101
--------
(1)000
 +    1
--------
   001 ✓
```

3 – 1 = 3+(-1) = 2

```
   011
+  110
--------
(1)001
 +    1
--------
   010 ✓
```

***It has no problem to perform subtraction, but needs to shift and add the carry**

2's Complement of a Binary Number

- "2's Complement" is the *radix complement* of binary numbers
- 2's complement of a *n-bit* number can be obtained by adding "1" to its **1's complement**, i.e.,

$$\begin{aligned} \mathbf{A^*} &= \mathbf{2^n - A} \\ &= \mathbf{(2^n - A - 1) + 1} \\ &= \mathbf{1\text{'s complement} + 1} \end{aligned}$$

Binary number (n=8): 01011100

2's Complement: $\underbrace{10100011}_{\text{1's complement}} + 1 = \underbrace{10100100}_{\text{2's complement}}$

1's complement **2's complement**

2's Complement representation of signed binary number

No change for positive numbers and use 2's complement for negative numbers

Decimal	2's Complement
3	011
2	010
1	001
0	000
-1	**111**
-2	**110**
-3	**101**
-4	**100**

Only one zero

3 – 2 = 3+(-2)=1

```
   011
+  110
--------
(1)001
```

Carry ignored

3 – 1 = 3+(-1) = 2

```
   011
+  111
--------
(1)010
```

Carry ignored

- **No problem in performing subtraction**
- **Carry is discarded (there is NO NEED to shift and add the carry, thus more hardware efficient**

Magnitude range: $-(2^{n-1}) \sim (2^{n-1}-1)$

Signed Binary Number (Recap)

- **Sign+Magnitude**
  - Two zero representations (+/- zeros)
  - It cannot correctly perform subtraction
  - Magnitude range: $-(2^{n-1}-1) \sim (2^{n-1}-1)$
- **1's Complement (Diminished radix complement)**
  - Defined as: $\mathbf{A^* = (2^n - 1) - A}$
  - 1's complement can be obtained by reversing the bits
  - Two zero representations (+/- zeros)
  - It can correctly perform subtraction, but needs to shift and add the carry
  - Magnitude range: $-(2^{n-1}-1) \sim (2^{n-1}-1)$
- **2's Complement (Radix complement)**
  - Defined as: $\mathbf{A^* = 2^n - A}$
  - One zero representation
  - It can correctly perform subtraction by just ignoring the carry
  - 2's complement can be obtained by adding "1" to its 1's complement
  - Magnitude range: $-(2^{n-1}) \sim (2^{n-1}-1)$

Positive numbers are same in all 3 signed binary number representations

4-bit Signed Binary Numbers Table

Signed Representations of Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Binary-Coded Decimal (BCD)

- BCD is a code to represent ten decimal digits (0 – 9)
- Each decimal digit is represented by a 4-bit binary number

Decimal → BCD:

Decimal → $(5\ 9\ 8)_{10}$

BCD → 0101 1001 1000

Note:
Six numbers, from **1010 to 1111**, are not used in BCD.

Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Decimal number addition with BCD

(Assume that BCD is used)

```
  4          0100
+ 3    ⇨   + 0011
-----      ----------
  7          0111 ✓
```

```
  5          0101
+ 7    ⇨   + 0111
-----      ----------
 12          1100 ✗
```

$(12)_{10}$ in BCD representation is 0001 0010

Not a legitimate BCD code

```
  8          1000
+ 9    ⇨   + 1001
-----      ----------
 17        1 0001 ✗
```

The result is in legitimate BCD code, but the sum is wrong

$(17)_{10}$ in BCD representation is 0001 0111

What is the problem?

- Decimal addition is a modulo-10 scheme and a carry is generated when the sum > 9
- 4-bit binary addition is a modulo-16 scheme and the carry is only generated when the sum > 15
- **Need to generate carry when sum>9, so: what about adding 6 to the result?**

```
    5          0101
  + 7    ⇨   + 0111
  -----      ----------
   12          1100 ✘
             + 0110
             ----------
            1  0010
            1    2  ✓
```

```
    8          1000
  + 9    ⇨   + 1001
  -----      ----------
   17       1  0001 ✘
          +    0110
          -------------
            1  0111
            1    7  ✓
```

The results are correct after adding 6!

What is the Rule?

- For decimal addition: S = A + B using BCD code

If $S \leq 9$ → Sum = S and carry = 0
(No correction is needed)

If $S > 9$ → Sum = S + 6 and carry = 1
(Need to be corrected by adding 6)

Summary of the Lecture

- We have covered
  - Position number system (radix 10, 2, 8 and 16)
  - Conversion among decimal, binary, octal and hex)
  - Binary arithmetic
  - Signed binary number representations (S-M, 1's complement and 2's complement
  - Arithmetic using signed binary numbers
  - Binary-coded decimals
  - Decimal addition using BCD