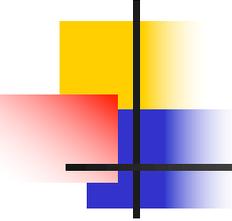


Chapter 1 : Numbering Systems

CPIT 210



Types Of Numbers

- **Natural Numbers**

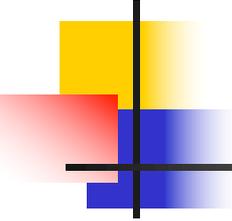
- The number 0 and any number obtained by repeatedly adding a count of 1 to 0

- **Negative Numbers**

- A value less than 0

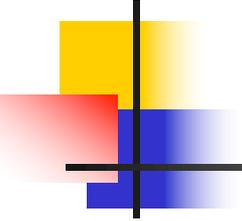
- **Integer**

- A natural number, the negative of a natural number, and 0.
- So an **integer number system** is a system for 'counting' things in a simple systematic way



Exponent Review

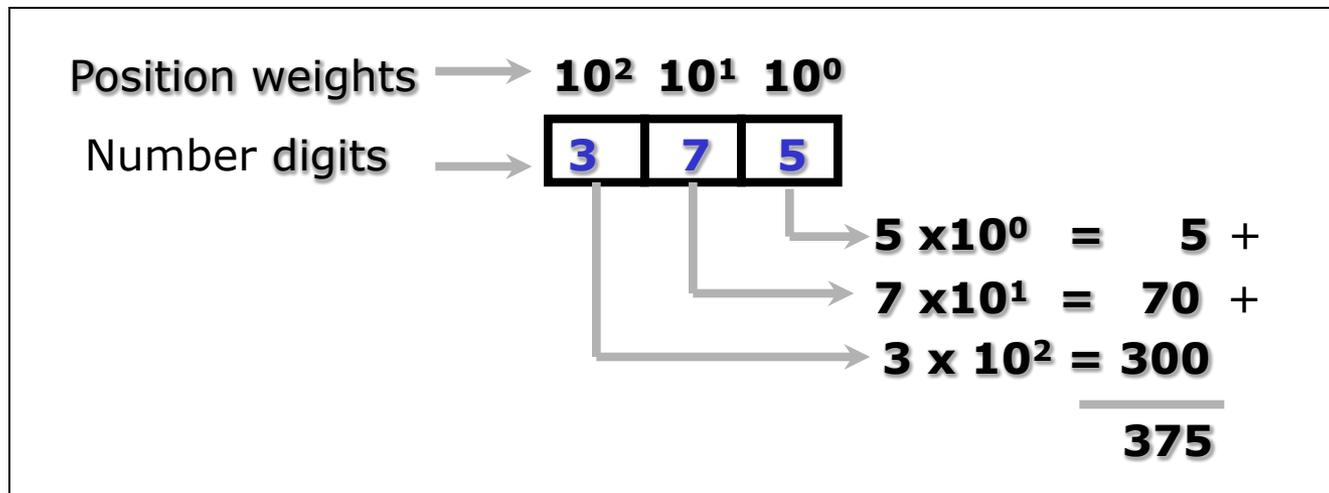
- An exponent (power) tells you how many times to multiply the base by itself:
 - $2^1 = 2$
 - $2^2 = 2 \times 2 = 4$
 - $2^3 = 2 \times 2 \times 2 = 8$
- $2^0 = 1$ (ANY number raised to power 0 is 1)
- $1 / x^2 = x^{-2}$

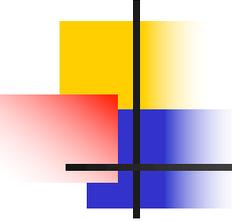


Decimal Numbering System

- How is a **positive integer** represented in decimal?
- Let's analyze the decimal number **375**:

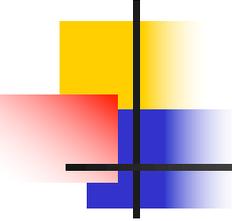
$$\begin{aligned} 375 &= (3 \times 100) + (7 \times 10) + (5 \times 1) \\ &= (3 \times 10^2) + (7 \times 10^1) + (5 \times 10^0) \end{aligned}$$





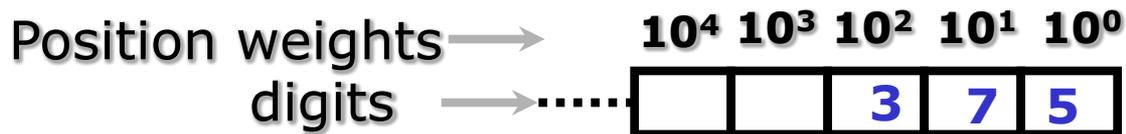
Decimal System Principles

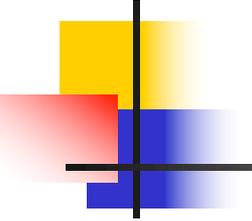
- A decimal number is a sequence of **digits**
- Decimal **digits** must be in the set:
 $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (Base 10)
- Each digit contributes to the **value** the number represents
- The **value** contributed by a **digit** equals the product of the **digit times the weight of the position** of the digit in the number



Decimal System Principles

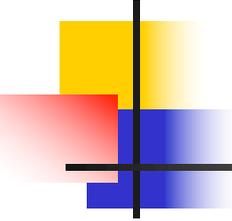
- Position weights are powers of 10
- The weight of the rightmost (**least significant** digit) is 10^0 (i.e.1)
- The **weight** of any position is 10^x , where x is the number of positions to the right of the least significant digit





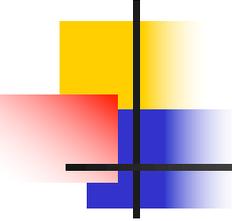
Bits

- In a computer, information is stored using digital signals that translate to **binary numbers**
- A single binary digit (0 or 1) is called a **bit**
 - A single bit can represent two possible states, on (1) or off (0)
- Combinations of bits are used to store values



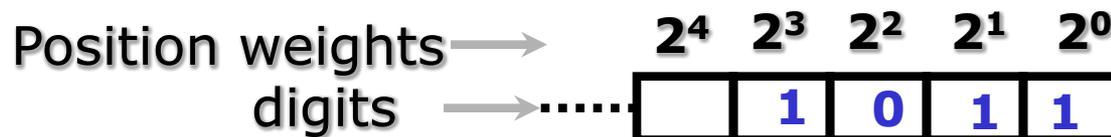
Data Representation

- Data **representation** means encoding **data** into **bits**
 - Typically, **multiple bits** are used to represent the ‘**code**’ of each **value** being represented
- Values being represented may be characters, numbers, images, audio signals, and video signals.
- Although a different scheme is used to encode each type of data, in the end the code is always a string of **zeros** and **ones**



Decimal to Binary

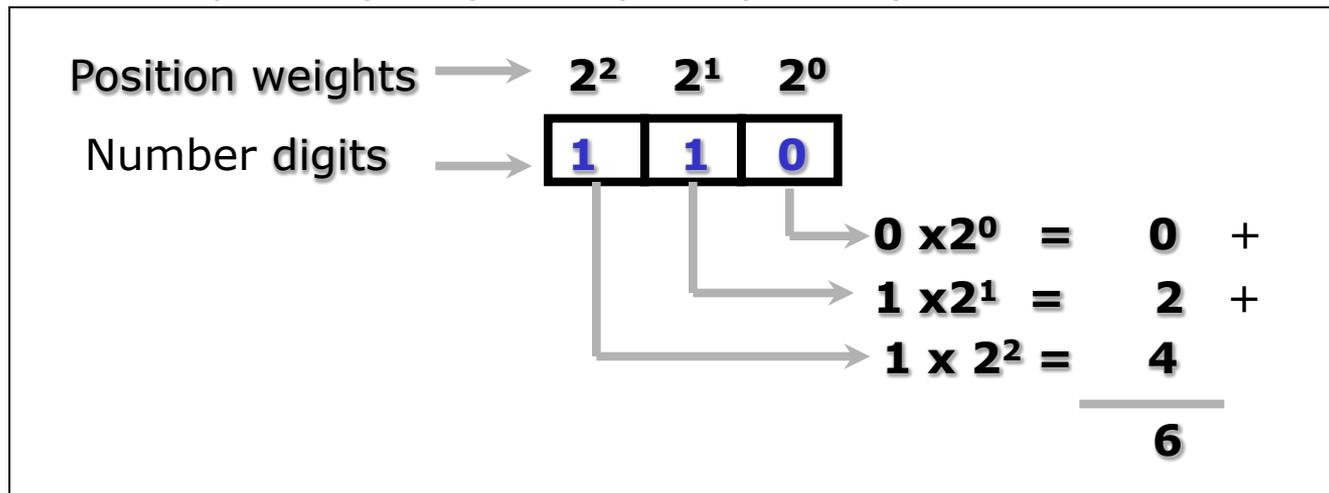
- So in a computer, the only possible digits we can use to encode data are **{0,1}**
 - The numbering system that uses this set of digits is the **base 2 system** (also called the **Binary** Numbering System)
- We can apply all the principles of the base 10 system to the base 2 system



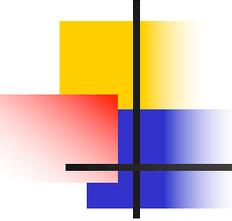
Binary Numbering System

- How is a **positive integer** represented in **binary**?
- Let's analyze the binary number **110**:

$$\begin{aligned} 110 &= (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) \\ &= (1 \times 4) + (1 \times 2) + (0 \times 1) \end{aligned}$$

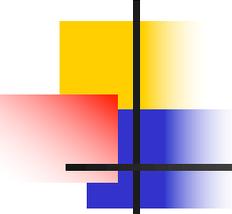


- So a count of **SIX** is represented in binary as **110**



Binary to Decimal Conversion

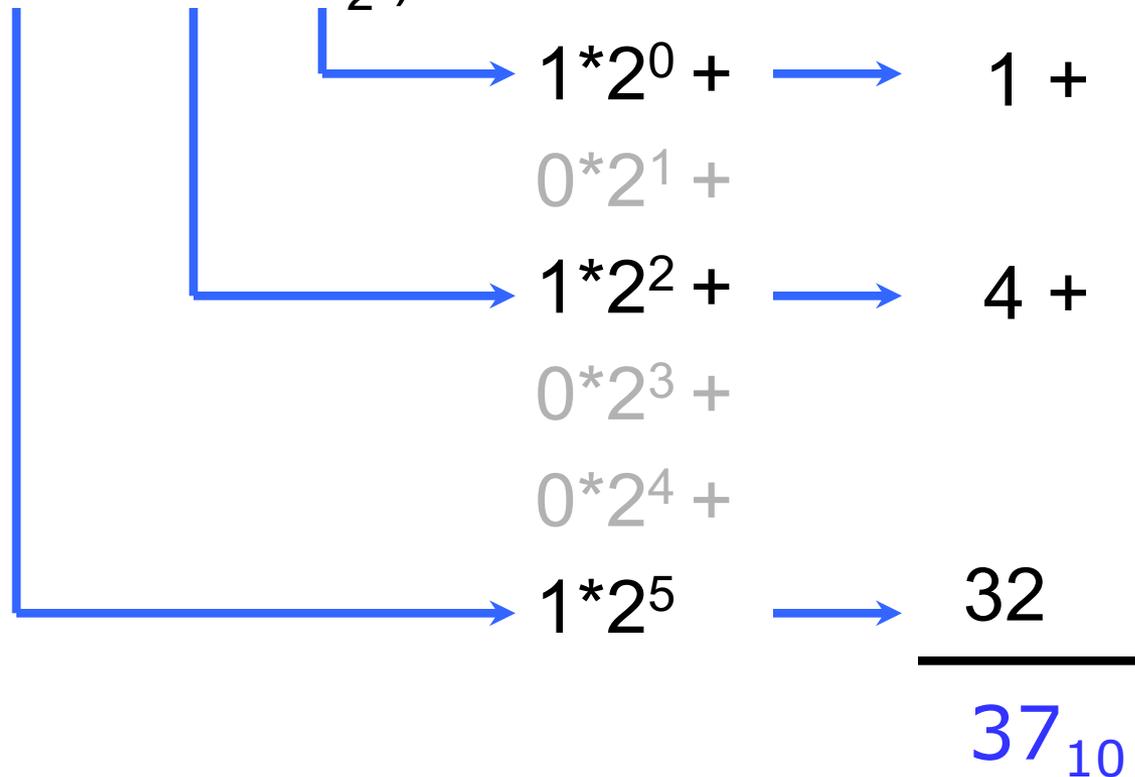
- To convert a base 2 (binary) number to base 10 (decimal):
 - Add all the values (positional weights) where a **one** digit occurs
 - Positions where a **zero** digit occurs do NOT add to the value, and can be ignored

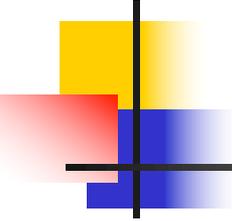


Binary to Decimal Conversion

Example: **Convert binary 100101 to decimal**

(written $1\ 0\ 0\ 1\ 0\ 1_2$) =


$$\begin{array}{r} 1 \cdot 2^0 + \longrightarrow 1 + \\ 0 \cdot 2^1 + \\ 1 \cdot 2^2 + \longrightarrow 4 + \\ 0 \cdot 2^3 + \\ 0 \cdot 2^4 + \\ 1 \cdot 2^5 \longrightarrow \underline{32} \\ \hline 37_{10} \end{array}$$



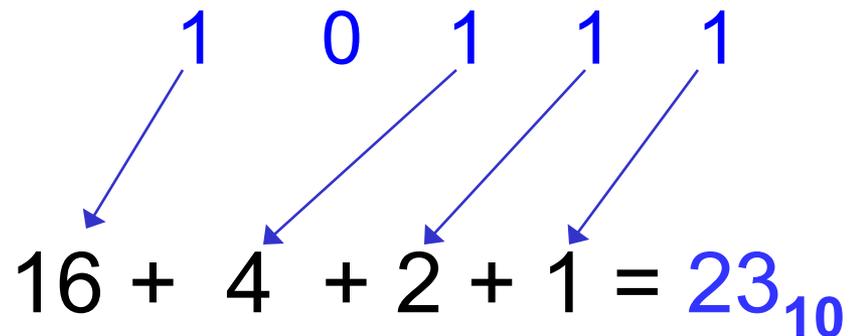
Binary to Decimal Conversion

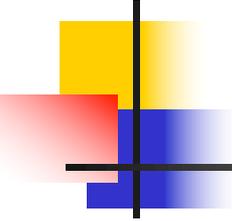
- Example #2: 10111_2

positional powers of 2: 2^4 2^3 2^2 2^1 2^0

decimal positional value: **16** **8** **4** **2** **1**

binary number:


$$16 + 4 + 2 + 1 = 23_{10}$$



Binary to Decimal Conversion

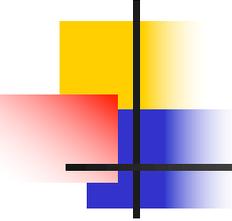
- Example #3: 110010_2

positional powers of 2: 2^5 2^4 2^3 2^2 2^1 2^0

decimal positional value: 32 16 8 4 2 1

binary number:

$$\begin{array}{cccccc} & 1 & 1 & 0 & 0 & 1 & 0 \\ & \downarrow & \searrow & & & \swarrow & \\ 32 & + & 16 & + & 2 & = & 50_{10} \end{array}$$



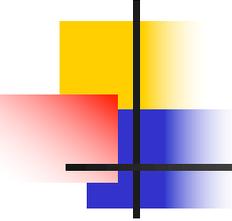
Decimal to Binary Conversion

The **Division** Method:

- 1) Start with your number (call it N) in base 10
- 2) Divide N by 2 and record the remainder
- 3) If (quotient = 0) then stop
else make the quotient your new N , and go back to step 2

The **remainders** comprise your answer, starting with the last remainder as your first (leftmost) digit.

In other words, divide the decimal number by 2 until you reach zero, and then collect the remainders in reverse.



Decimal to Binary Conversion

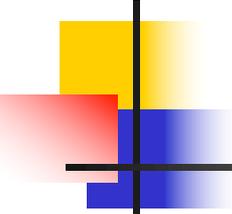
Using the **Division Method**:

Divide decimal number by 2 until you reach zero, and then collect the **remainders** in reverse.

Example 1: 22_{10} = 10110_2

	<u>Rem:</u>
2) <u>22</u>	0
2) <u>11</u>	1
2) <u>5</u>	1
2) <u>2</u>	0
2) <u>1</u>	1
0	





Decimal to Binary Conversion

Using the **Division Method**

Example 2:

$$56_{10} = 111000_2$$

$$2 \overline{) 56}$$

$$2 \overline{) 28}$$

$$2 \overline{) 14}$$

$$2 \overline{) 7}$$

$$2 \overline{) 3}$$

$$2 \overline{) 1}$$

0

Rem:

0

0

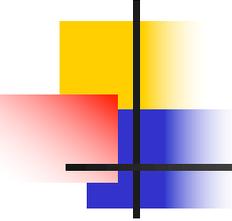
0

1

1

1

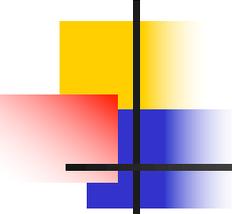




Decimal to Binary Conversion

*The **Subtraction** Method:*

- Subtract out largest power of 2 possible (without going below zero), repeating until you reach 0.
 - Place a **1** in each position where you COULD subtract the value
 - Place a **0** in each position that you could NOT subtract out the value without going below zero.



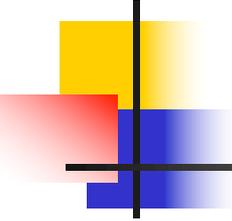
Decimal to Binary Conversion

Example 1:

21_{10}

21	2^6	2^5	2^4	2^3	2^2	2^1	2^0
<u>- 16</u>	64	32	16	8	4	2	1
5			1	0	1	0	1
<u>- 4</u>							
1							
<u>- 1</u>							
0							

Answer: $21_{10} = 10101_2$



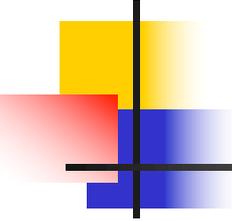
Decimal to Binary Conversion

Example 2:

56_{10}

56	2^6		2^5	2^4	2^3	2^2	2^1	2^0
- <u>32</u>	64		32	16	8	4	2	1
24			1	1	1	0	0	0
- <u>16</u>								
8								
- <u>8</u>								
0								

Answer: $56_{10} = 111000_2$



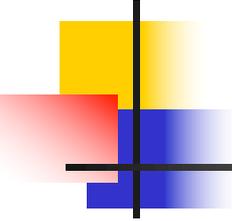
Octal Numbering System

- Base: 8
- Digits: 0, 1, 2, 3, 4, 5, 6, 7

■ Octal number: 357_8

$$= (3 \times 8^2) + (5 \times 8^1) + (7 \times 8^0)$$

- To convert to base 10, beginning with the **rightmost** digit, multiply each **n**th digit by $8^{(n-1)}$, and add all of the results together.



Octal to Decimal Conversion

- Example 1: 357_8

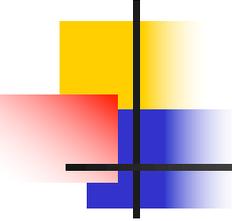
positional powers of 8: 8^2 8^1 8^0

decimal positional value: 64 8 1

Octal number: 3 5 7

$$(3 \times 64) + (5 \times 8) + (7 \times 1)$$

$$= 192 + 40 + 7 = 239_{10}$$



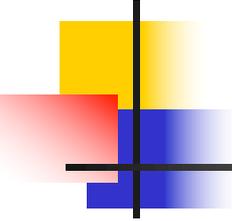
Octal to Decimal Conversion

- Example 2: 1246_8

positional powers of 8:	8^3	8^2	8^1	8^0
decimal positional value:	512	64	8	1
Octal number:	1	2	4	6

$$(1 \times 512) + (2 \times 64) + (4 \times 8) + (6 \times 1)$$

$$= 512 + 128 + 32 + 6 = 678_{10}$$



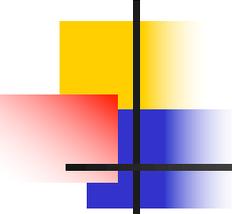
Decimal to Octal Conversion

The **Division** Method:

- 1) Start with your number (call it N) in base 10
- 2) Divide N by 8 and record the remainder
- 3) If (quotient = 0) then stop
else make the quotient your new N , and go back to step 2

The **remainders** comprise your answer, starting with the last remainder as your first (leftmost) digit.

In other words, divide the decimal number by 8 until you reach zero, and then collect the remainders in reverse.



Decimal to Octal Conversion

Using the **Division** Method:

Example 1: $214_{10} = 326_8$

$$8 \overline{) 214}$$

$$8 \overline{) 26}$$

$$8 \overline{) 3}$$

0

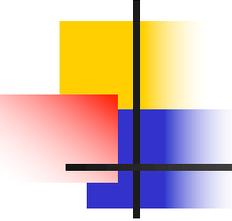
Rem:

6

2

3

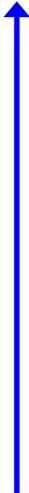


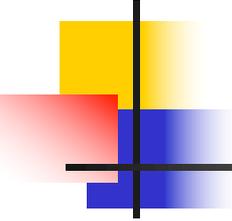


Decimal to Octal Conversion

Example 2: $4330_{10} = 10352_8$

$8 \overline{) 4330}$	<u>Rem:</u>
$8 \overline{) 541}$	2
$8 \overline{) 67}$	5
$8 \overline{) 8}$	3
$8 \overline{) 1}$	0
0	1

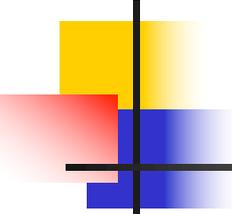




Decimal to Octal Conversion

*The **Subtraction** Method:*

- Subtract out multiples of the largest power of 8 possible (without going below zero) each time until you reach 0.
 - Place the **multiple value** in each position where you COULD subtract the value.
 - Place a **0** in each position that you could NOT subtract out the value without going below zero.

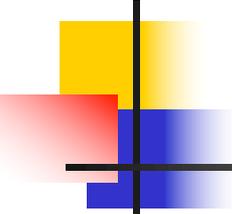


Decimal to Octal Conversion

Example 1: **315_{10}**

315		8^2	8^1	8^0
<u>- 256</u>	(4 x 64)	64	8	1
59		4	7	3
<u>- 56</u>	(7 x 8)			
3				
<u>- 3</u>	(3 x 1)			
0				

Answer: $315_{10} = 473_8$



Decimal to Octal Conversion

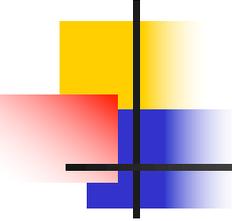
Example 2:

2018_{10}

$$\begin{array}{r} 2018 \\ - \underline{1536} \text{ (3 x 512)} \\ 482 \\ - \underline{448} \text{ (7 x 64)} \\ 34 \\ - \underline{32} \text{ (4 x 8)} \\ 2 \\ - \underline{2} \text{ (2 x 1)} \\ 0 \end{array}$$

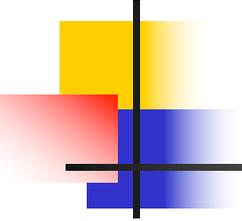
8^4	8^3	8^2	8^1	8^0
4096	512	64	8	1
	3	7	4	2

Answer: $2018_{10} = 3742_8$



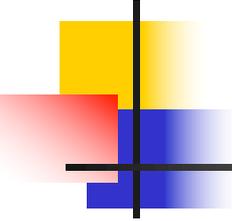
Hexadecimal (Hex) Numbering System

- Base: 16
- Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Hexadecimal number: $1F4_{16}$
 $= (1 \times 16^2) + (F \times 16^1) + (4 \times 16^0)$



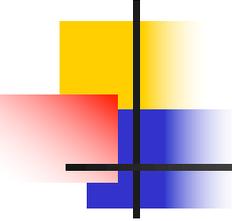
Hexadecimal (Hex) Extra Digits

<u>Decimal Value</u>	<u>Hexadecimal Digit</u>
10	A
11	B
12	C
13	D
14	E
15	F



Hex to Decimal Conversion

- To convert to base 10:
 - Begin with the rightmost digit
 - Multiply each **n**th digit by $16^{(n-1)}$
 - Add all of the results together



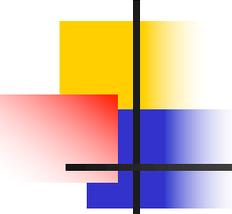
Hex to Decimal Conversion

■ Example 1: $1F4_{16}$

positional powers of 16:	16^3	16^2	16^1	16^0
decimal positional value:	4096	256	16	1
Hexadecimal number:	1	F	4	

$$(1 \times 256) + (F \times 16) + (4 \times 1)$$
$$= (1 \times 256) + (15 \times 16) + (4 \times 1)$$

$$= 256 + 240 + 4 = 500_{10}$$

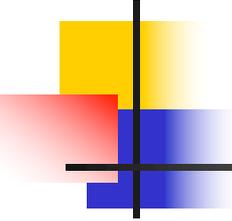


Hex to Decimal Conversion

■ Example 2: $25AC_{16}$

positional powers of 16:	16^3	16^2	16^1	16^0
decimal positional value:	4096	256	16	1
Hexadecimal number:	2	5	A	C

$$\begin{aligned} & (2 \times 4096) + (5 \times 256) + (A \times 16) + (C \times 1) \\ &= (2 \times 4096) + (5 \times 256) + (10 \times 16) + (12 \times 1) \\ &= 8192 + 1280 + 160 + 12 = 9644_{10} \end{aligned}$$



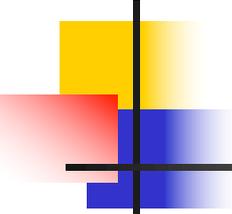
Decimal to Hex Conversion

The **Division** Method:

- 1) Start with your number (call it N) in base 10
- 2) Divide N by 16 and record the remainder
- 3) If (quotient = 0) then stop
else make the quotient your new N , and go back to step 2

The **remainders** comprise your answer, starting with the last remainder as your first (leftmost) digit.

In other words, divide the decimal number by 16 until you reach zero, and then collect the remainders in reverse.



Decimal to Hex Conversion

Using The **Division** Method:

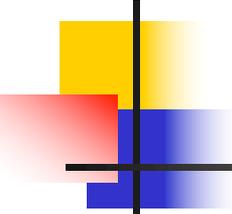
Example 1: $126_{10} = 7E_{16}$

$$16 \overline{) 126}$$

$$\begin{array}{r} 16 \overline{) 7} \\ \underline{0} \\ 0 \end{array}$$

Rem:

$$\begin{array}{r} 14 = E \\ 7 \end{array} \quad \uparrow$$



Decimal to Hex Conversion

Example 2:

$$603_{10} = 25B_{16}$$

$$16 \overline{) 603}$$

$$16 \overline{) 37}$$

$$16 \overline{) 2}$$

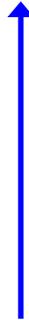
0

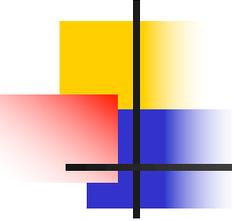
Rem:

11=B

5

2

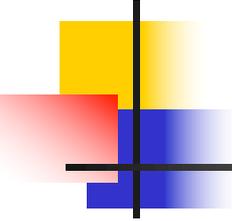




Decimal to Hex Conversion

*The **Subtraction** Method:*

- Subtract out multiples of the largest power of 16 possible (without going below zero) each time until you reach 0.
 - Place the **multiple value** in each position where you COULD to subtract the value.
 - Place a **0** in each position that you could NOT subtract out the value without going below zero.

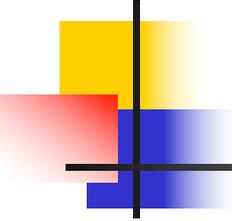


Decimal to Hex Conversion

Example 1: 810_{10}

810		16^2	16^1	16^0
<u>- 768</u>	(3 x 256)	256	16	1
42		3	2	A
<u>- 32</u>	(2 x 16)			
10				
<u>- 10</u>	(10 x 1)			
0				

Answer: $810_{10} = 32A_{16}$

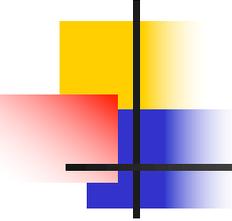


Decimal to Hex Conversion

Example 2: **156**₁₀

156		16^2	16^1	16^0
- 144	(9 x 16)	256	16	1
<u> </u>			9	C
12				
- 12	(12 x 1)			
<u> </u>				
0				

Answer: $156_{10} = 9C_{16}$



Binary to Octal Conversion

The maximum value represented in 3 bit is:

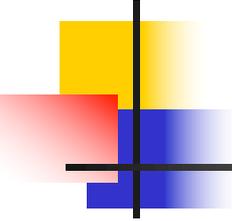
$$2^3 - 1 = 7$$

So using 3 bits we can represent values from

0 to 7

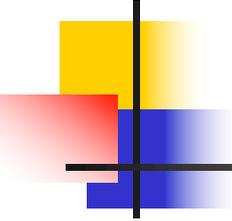
which are the digits of the Octal numbering system.

Thus, three binary digits can be converted to one octal digit.



Binary to Octal Conversion

<u>Three-bit Group</u>	<u>Decimal Digit</u>	<u>Octal Digit</u>
000	0	0
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7



Octal to Binary Conversion

Ex : Convert 742_8 to binary

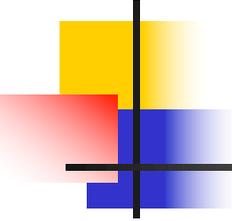
Convert each octal digit to 3 bits:

$$\begin{array}{rcl} 7 & = & 111 \\ 4 & = & 100 \\ 2 & = & 010 \end{array}$$

$\swarrow \quad \swarrow \quad \swarrow$

$$111 \quad 100 \quad 010$$

$$742_8 = 111100010_2$$



Binary to Octal Conversion

Ex : Convert 10100110_2 to octal

Starting at the right end, split into groups of 3:

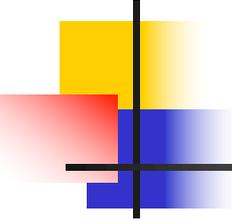
10 100 110 \rightarrow

110 = 6

100 = 4

010 = 2 (pad empty digits with 0)

$10100110_2 = 246_8$



Binary to Hex Conversion

The maximum value represented in 4 bit is:

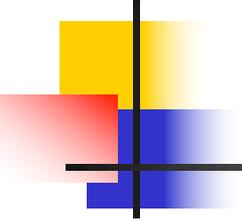
$$2^4 - 1 = 15$$

So using 4 bits we can represent values from

0 to 15

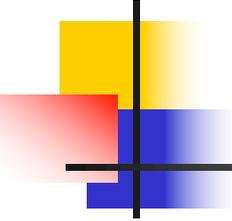
which are the digits of the Hexadecimal numbering system.

Thus, **four binary** digits can be converted to **one hexadecimal** digit.



Binary to Hex Conversion

<u>Four-bit Group</u>	<u>Decimal Digit</u>	<u>Hexadecimal Digit</u>
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	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F



Binary to Hex Conversion

Ex : Convert 110100110_2 to hex

Starting at the right end, split into groups of 4:

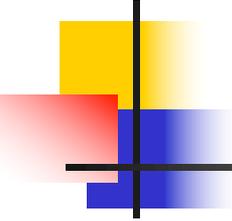
1 1010 0110 →

0110 = 6

1010 = A

0001 = 1 (pad empty digits with 0)

$110100110_2 = 1A6_{16}$



Hex to Binary Conversion

Ex : Convert $3D9_{16}$ to binary

Convert each hex digit to 4 bits:

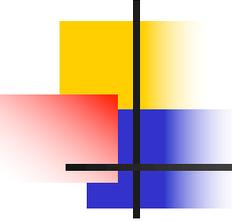
$$3 = 0011$$

$$D = 1101$$

$$9 = 1001$$

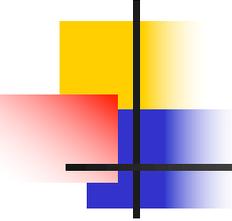
0011 1101 1001 \rightarrow

$3D9_{16} = 1111011001_2$ (can remove leading zeros)



Conversion between Binary and Hex - Try It Yourself

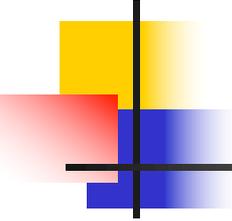
- Convert the following numbers:
 - 1010111101_2 to Hex
 - $82F_{16}$ to Binary
- (Answers on NEXT slide)



Answers

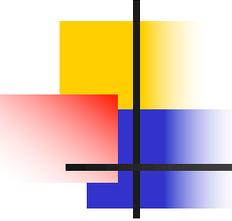
■ $1010111101_2 \rightarrow 10\ 1011\ 1101$
 $= 2BD_{16}$

■ $82F_{16} = 0100\ 0010\ 1111$
 $\rightarrow 10000101111_2$



Octal to Hex Conversion

- To convert between the Octal and Hexadecimal numbering systems
 - Convert from one system to **binary** first
 - Then convert from binary to the new numbering system



Hex to Octal Conversion

Ex : Convert $E8A_{16}$ to octal

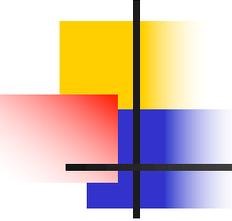
First convert the hex to binary:

$1110\ 1000\ 1010_2$
↓ ↓ ↓ ↓
111 010 001 010 and re-group by 3 bits
(starting on the right)

Then convert the binary to octal:

7 2 1 2

So $E8A_{16} = 7212_8$



Octal to Hex Conversion

Ex : Convert 752_8 to hex

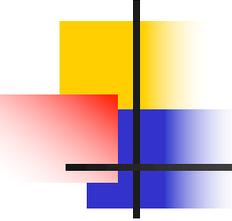
First convert the octal to binary:

$111\ 101\ 010_2$
 ↙ ↘ ↘
 $0001\ 1110\ 1010$ re-group by 4 bits
 (add leading zeros)

Then convert the binary to hex:

1 E A

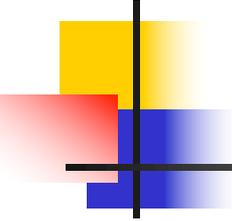
So $752_8 = 1EA_{16}$



Binary Addition

- Add one digit at a time
- Obtain a sum and a carry
- Similar to decimal addition – but pay attention to the base

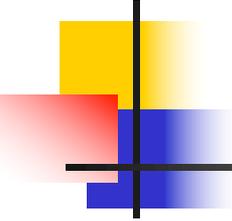
0	10	11	01
$\begin{array}{r} 0110 \\ 0111 \\ \hline 1 \end{array}$	$\begin{array}{r} 0110 \\ 0111 \\ \hline 01 \end{array}$	$\begin{array}{r} 0110 \\ 0111 \\ \hline 101 \end{array}$	$\begin{array}{r} 0110 \\ 0111 \\ \hline 1101 \end{array}$



Binary Addition

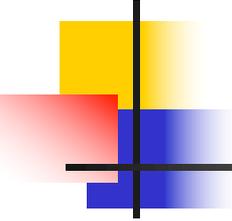
Table 1.5 One-bit adder.

a	b	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



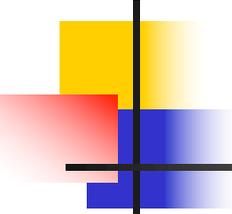
Signed Numbers

- Signed numbers are mostly stored in *two's complements* form
 - Leading bit is 0 for positive numbers and 1 for negative
 - For n bits, the range of numbers that can be stored is:
 - $-2^{n-1}; 2^{n-1}-1$
- **To derive the binary negative (two's complement) of a number:**
 - Determine the magnitude (how many bits)
 - Find the binary equivalent of the magnitude
 - Complement each bit
 - Add 1



Signed Numbers

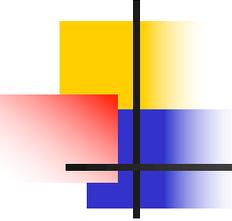
- **Example:**
 - **Derive the 6-bit binary negative (two's complement) of 17**
 - Determine the magnitude (how many bits)
 - **6 bits**
 - Find the binary equivalent of the magnitude
 - **010001**
 - Complement each bit (one's complement)
 - **101110**
 - Add 1
 - **101111**



Signed Numbers

Table 1.6 Signed and unsigned 4-bit numbers.

Binary	Positive	Signed (two's complement)
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	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1



Binary Subtraction

- Take the two complement of the second operand
- Then add
- For signed numbers:
 - Ignore the carry-out of the higher order
 - If two numbers of the same sign are added, and a result of the opposite sign is obtained, there's an overflow
 - Ex: $7 - 5$; $-7 - 5$
- For Unsigned number
 - A carry-out of zero in the higher-order bit indicates overflow
 - Ex: $5 - 7$