Computer Fundamentals

Lecture ٩

Numbering Systems

Number Systems - Why Binary?

- Early computer design was decimal
- John von Neumann proposed binary data processing (1945)
  - Simplified computer design
  - Used for both instructions and data
- Natural relationship between on/off switches and calculation using Boolean logic

IBM 650 - 1950s

	
On	Off
True	False
Yes	No
1	**0**

Numbering Systems

- Computers only deal with binary data (0s and 1s).
- All data manipulated by computers must be represented in binary format.
- Machine instructions manipulate many different forms of data:
  - **Numbers**:
    - Integers: 33, +128, -2827
    - Real numbers: 1.33, +9.55609, -6.76E12, +4.33E-03
  - **Alphanumeric characters** (letters, numbers, signs, control characters): examples: A, a, c, 1 ,3, ", +, Ctrl, Shift, etc.
  - **Images** (still or moving): Usually represented by numbers representing the Red, Green and Blue (RGB) colors of each pixel in an image,
  - **Sounds**: Numbers representing sound amplitudes sampled at a certain rate (usually 20kHz).
- In general we have two major data types that need to be represented in computers; numbers and characters.

Common Numbering Systems

- The most widely used numbering systems are listed in below:
  - **Binary** number system
  - **Octal** number system
  - **Decimal** number system
  - **Hexadecimal** (hex) number system

Common Numbering Systems

Name	Base	Symbols
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 , A, B, C, D, E, F

Decimal Number System

- Decimal number system is a **base 10** number system having 10 digits from 0 to 9.
- This means that any numerical quantity can be represented using these 10 digits.
- Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position.
- Let us take an example to understand this.
- Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−
  - In 734, value of 7 is 7 hundreds or 700 or 7×100 or 7×10^2
  - In 971, value of 7 is 7 tens or 70 or 7×10 or 7×10^1
  - In 207, value 0f 7 is 7 units or 7 or 7×1 or 7×10^0

Decimal Number System

- The weightage of each position can be represented as follows −

10^5	10^4	10^3	10^2	10^1	10^0

Examples of positional notation:

$1996_{10} = 1 \times 10^3 + 9 \times 10^2 + 9 \times 10^1 + 6 \times 10^0$
$2000_{10} = 2 \times 10^3$

Binary Number System

- Binary number system has two symbols: **0** and 1, called **bits**.
- This system is thus a **base 2** number system.
- As mentioned earlier, in the decimal system, each column represents a higher power of ten, starting at the right end with 10^0 , e.g.:
- The highest decimal number that can be represented by n bits binary number is $2^n - 1$.
- Thus with an **8 bit** number the maximum decimal number that be represented is $2^8 - 1$ is 255.

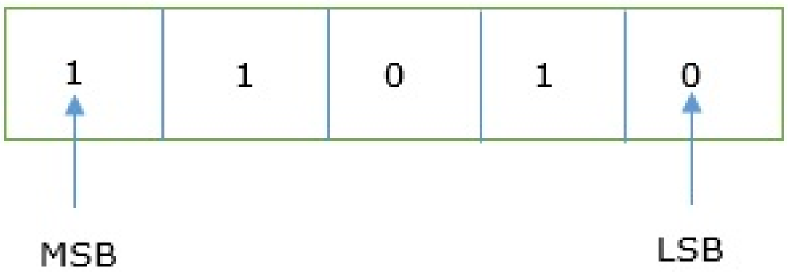
Binary Number System

- Binary number system has two symbols: **0** and 1, called **bits**.
- This system is thus a **base 2** number system.
- As mentioned earlier, in the decimal system, each column represents a higher power of ten, starting at the right end with 10^0 , e.g.:
- Likewise, in the binary number system, which is also positional, each position represents a larger power of two, starting with 2^0 on the right end of the whole number, as displayed here.

2^5	2^4	2^3	2^2	2^1	2^0

Binary Number System (cont'd)

- In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.



And decimal equivalent of this number is sum of product of each digit with its positional value.

Binary to Decimal Conversion

- Multiply each binary bit by its column value
  - In binary, our columns are (from right to left)
    - **$2^0 = 1$**
    - **$2^1 = 2$**
    - **$2^2 = 4$**
    - **$2^3 = 8$**
    - **$2^4 = 16$**
    - **$2^5 = 32$**
    - **Etc**

Example

For the Binary Number: 10101_2, Calculating Decimal Equivalent −

Step	Binary Number	Decimal Number
Step 1	**10101_2**	**$((1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$**
Step 2	**10101_2**	**$(16 + 0 + 4 + 0 + 1)_{10}$**
Step 3	**10101_2**	**21_{10}**

Note − 10101_2 is normally written as 10101.

Simplifying Conversion in Binary

- Our digits will either be 0 or 1
  - 0 * anything is 0
  - 1 * anything is that thing
- Just add together the powers of 2 whose corresponding digits are 1 and ignore any digits of 0

- **10110** $= 2^4 + 2^2 + 2^1 = 16 + 4 + 2 =$ **22**
- **1100001** $= 2^6 + 2^5 + 2^0 = 64 + 32 + 1 =$ **97**

Powers of Two

Power of Two	Decimal Value
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1,024

Decimal to Binary Conversion

- Follow the steps:
  - Divide the decimal number by 2
  - Keep the integer quotient for the coming iteration
  - Keep the remainder for the binary digit
  - Repeat the steps till you get 0 as your quotient
  - The binary number is the group of remainder bits written in opposite order

- Example: Convert **19** to **binary**

  19 / 2 = 9 remainder **1**

  9 / 2 = 4 remainder **1**

  4 / 2 = 2 remainder **0**

  2 / 2 = 1 remainder **0**

  1 / 2 = 0 remainder **1**

  - 19 = $\mathbf{10011_2}$

Record the remainders and then write them in opposite order

Decimal to Binary Conversion

Example: Convert **112** to **binary**

Division	Remainder (R)
112 / 2 = 56	0
56 / 2 = 28	0
28 / 2 = 14	0
14 / 2 = 7	0
7 / 2 = 3	1
3 / 2 = 1	1
1 / 2 = 0	1

- 112 = $\mathbf{1110000_2}$

Decimal to Binary Conversion - **Examples**

Convert 200 to binary

200 / 2 = 100 remainder 0
100 / 2 = 50 r 0
50 / 2 = 25 r 0
25 / 2 = 12 r 1
12 / 2 = 6 r 0
6 / 2 = 3 r 0
3 / 2 = 1 r 1
1 / 2 = 0 r 1

200 = 11001000

Convert 16 to binary

16 / 2 = 8 r 0
8 / 2 = 4 r 0
4 / 2 = 2 r 0
2 / 2 = 1 r 0
1 /2 = 0 r 1

16 = 10000

Convert 122 to binary

122 / 2 = 61 r 0
61 / 2 = 30 r 1
30 / 2 = 15 r 0
15 / 2 = 7 r 1
7 / 2 = 3 r 1
3 / 2 = 1 r 1
1 / 2 = 0 r 1

122 = 1111010

Convert 21 to binary

21 / 2 = 10 r 1
10 / 2 = 5 r 0
5 / 2 = 2 r 1
2 / 2 = 1 r 0
1 / 2 = 0 r 1

21 = 10101

Conversion From one radix to another

- From decimal to base-*r*
  - Divide the number and all successive quotients by *r*
  - Example : convert $(165)_{10}$ to base-7

165 / 7 = 23 remainder **4**
23 / 7 = 3 remainder **2**
3 / 7 = 0 remainder **3**

$$(165)_{10} = (324)_7$$

Decimal to Binary Conversion - Another Technique

- Recall to convert from binary to decimal, we add the powers of 2 for each digit that is a 1
- To convert from decimal to binary, we can subtract all of the powers of 2 that make up the number and record 1s in corresponding columns
- Example – Convert 19 to binary.
  - 19 = **16** + **2** + **1**
  - So there is a 16 (2^4), a 2 (2^1) and 0 (2^0)
  - Put 1s in the 4th, 1st, and 0th columns:
  - 19 = 10011_2

Power of Two	Decimal Value
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1,024

Examples

- Convert 122 to binary
  - Largest power of 2 <= 122 = 64 leaving 122 – **64** = 58
  - Largest power of 2 <= 58 = 32 leaving 58 – **32** = 26
  - Largest power of 2 <= 26 = 16 leaving 26 – **16** = 10
  - Largest power of 2 <= 10 = 8 leaving 10 – **8** = 2
  - Largest power of 2 <= 2 = 2 leaving 0
  - Done
    - **122 = 64 + 32 + 16 + 8 + 2 = 1111010**

More examples:

- 555 = **512** + **32** + **8** + **2** + **1** = 1000101011
- 200 = 128 + 64 + 8 = 11001000
- 199 = 128 + 64 + 4 + 2 + 1 = 11000111
- 31 = 16 + 8 + 4 + 2 + 1 = 11111
- 60 = 32 + 16 + 8 + 4 = 111100
- 1000 = 512 + 256 + 128 + 64 + 32 + 8 = 1111101000
- 20 = 16 + 4 = 10100

Power of Two	Decimal Value
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1,024

Exercise

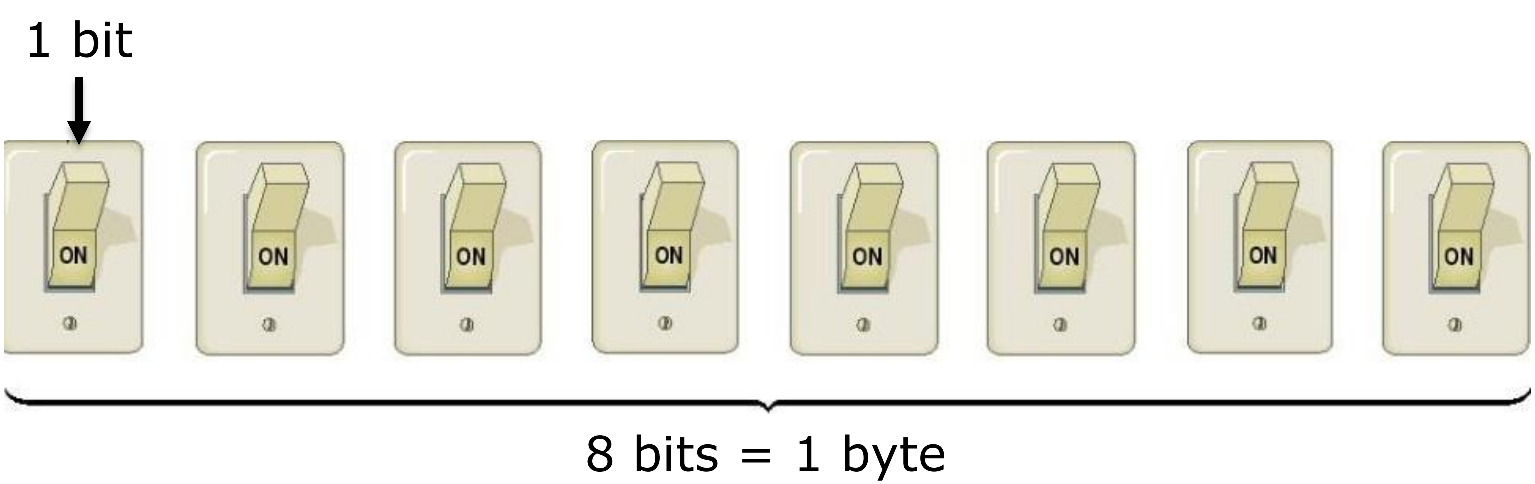
- Convert the binary numbers to decimal:
  - 1001001
  - 00110
  - 10111
  - 11111

Number of Bits

- Notice in our previous examples that for 555 we needed 10 bits and for 25 we only needed 5 bits
- The number of bits available tells us the range of values we can store
- In **8 bits** (1 byte), we can store between 0 and 255
  - 00000000 = 0
  - 11111111 = 255 (128 + 64 + 32 + 16 + 8 + 4 + 2 + 1)
- In **n bits**, you can store a number from **0** to **2^n-1**
  - For 8 bits, 2^8 = 256, the largest value that can be stored in 8 bits is 255
  - What about 5 bits?
  - What about 3 bits?

Bits and Bytes

- The most basic unit of storage in a device is represented by a **bit**, having a value of 1 or O.
- Computers work with collections of bits, grouping them to represent larger pieces of data, such as letters of the alphabet.
- **Eight bits** make up one **byte**. A byte is the amount of memory needed to store one alphanumeric character.
- With one byte, the computer can represent one of 256 different symbols or characters.



Data Representation

- How is a letter converted to binary form and back?

Step 1.
The user presses the capital letter **D** (shift+D key) on the keyboard.

Step 2.
An electronic signal for the capital letter **D** is sent to the system unit.

Step 3.
The signal for the capital letter **D** is converted to its ASCII binary code (01000100) and is stored in memory for processing.

Step 4.
After processing, the binary code for the capital letter **D** is converted to an image, and displayed on the output device.

Binary Operations

- We learn the binary operations using truth tables

X	Y	AND
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

X	NOT
0	1
1	0

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- Given two bits, apply the operator
  - 1 AND 0 = 0
  - 1 OR 0 = 1
  - 1 XOR 0 = 1
- Apply the binary (Boolean) operators *bitwise* (in columns) to binary numbers as in
  - 10010011 AND 00001111 = 00000011

Examples

- **AND – if both bits are 1 the result is 1, otherwise 0**
  - 11111101 AND 00001111 = 00001101
  - 01010101 AND 10101010 = 00000000
  - 00001111 AND 00110011 = 00000011
- **OR – if either bit is 1 the result is 1, otherwise 0**
  - 10101010 OR 11100011 = 11101011
  - 01010101 OR 10101010 = 11111111
  - 00001111 OR 00110011 = 00111111
- **NOT – flip (negate) each bit**
  - NOT 10101011 = 01010100
  - NOT 00001111 = 11110000
- **XOR – if the bits differ the result is 1, otherwise 0**
  - 10111100 XOR 11110101 = 01001001
  - 11110000 XOR 00010001 = 11100001
  - 01010101 XOR 01011110 = 00001011

Octal Number System

- The Octal Number System is another type of computer and digital numbering system which uses the Base-8 system.
- There are only 8 symbols or possible digit values, there are 0, 1, 2, 3, 4, 5, 6, 7.
- Each Octal number can be represented using only **3 bits**, with each group of bits having a distich values between **000** (for 0) and **111** (for 7).

Octal Numbers System Table

- We use only **3 bits** to represent Octal Numbers. Each group will have a distinct value between 000 and 111.

Decimal Number	3-bit Binary Number	Octal Number
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
8	001 000	10 (1+0)
9	001 001	11 (1+1)
Continuing upwards in groups of three		

Convert **Decimal** to **Octal**

- Follow the steps:
  - Divide the decimal number by 8
  - Keep the integer quotient for the coming iteration
  - Keep the remainder for the octal number
  - Repeat the steps till you get 0 as your quotient

165 / 8 = 20 remainder 5
20 / 8 = 2 remainder 4
2 / 8 = 0 remainder 2

$$(165)_{10} = (245)_8$$

Convert **Binary** to **Octal**

- The first step is to group the binary digits in the set of 3.
- Write an octal symbol for each group underneath.
- This will give you an octal number that arrived from a binary number.

Example – convert the binary number 1010111100 to octal

$= (1010111100)_2$

$= (001\ 010\ 111\ 100)_2$

$= (\ 1 \quad 2 \quad 7 \quad 4\ )_8$

$= (1274)_8$

Convert **Octal** to **Binary**

- Because each octal digit can be represented by a 3-bit binary number, it is very easy to convert from octal to binary..
- Octal Digit 0 1 2 3 4 5 6 7
- Binary 000 001 010 011 100 101 110 111

Octal Digit	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Example: Let's convert the octal numbers $\mathbf{25_8}$ and $\mathbf{140_8}$ to binary

$$\begin{array}{cc} 2 & 5 \\ \downarrow & \downarrow \\ 010 & 101 \end{array} \qquad \begin{array}{ccc} 1 & 4 & 0 \\ \downarrow & \downarrow & \downarrow \\ 001 & 100 & 000 \end{array}$$

10101_2 1100000_2

Hexadecimal Number System

- The hexadecimal number system uses sixteen digits/alphabets: **0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F** with the base number as 16.
- Here, A-F of the hexadecimal system means the numbers 10-15 of the decimal number system respectively.
- This system is used in computers to reduce the large-sized strings of the binary system.
- For example: $7B3_{16}$, $6F_{16}$, $4B2A_{16}$ are some examples of numbers in the hexadecimal number system.

Hexadecimal Numbers System Table

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Convert from Decimal to Hexadecimal

- Convert the decimal number **650** to hexadecimal by repeated division by **16**.

$$650 / 16 = 40 \text{ remainder } 10$$
$$40 / 16 = 2 \text{ remainder } 8$$
$$2 / 16 = 0 \text{ remainder } 2$$

$$650_2 = 28A_{16}$$

Convert Binary to Hexadecimal

- Simply break the binary number into 4-bit groups, starting at the right-most bit and replace each 4-bit group with the equivalent hexadecimal symbol as in the following example.
- Convert the binary number (1100101001010111) to hexadecimal:

Solution:

$$\begin{array}{cccc} 1100 & 1010 & 0101 & 0111 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ C & A & 5 & 7 \end{array} = CA57_{16}$$

Convert Hexadecimal to Decimal

- One way to find the decimal equivalent of a hexadecimal number is to first convert the hexadecimal number to binary and then convert from binary to decimal.
- Convert the hexadecimal number (1C) to decimal:

1 C

0001 1100 $= 2^4 + 2^3 + 2^2 = 16+8+4 = 28_{10}$

Convert Hexadecimal to Decimal

- Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here:

…	16^5	16^4	16^3	16^2	16^1	16^0

- Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

$\mathbf{27FB_{16}} = 2 \times 16^3 + 7 \times 16^2 + 15 \times 16^1 + 11 \times 16^0$

$= 8192 + 1792 + 240 + 11$

$= 10235_{10}$

Numbers in Different Bases

- Good idea to memorize!

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10

Binary Operations

- We learn the binary operations using truth tables

X	Y	AND
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

X	NOT
0	1
1	0

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- Given two bits, apply the operator
  - 1 AND 0 = 0
  - 1 OR 0 = 1
  - 1 XOR 0 = 1
- Apply the binary (Boolean) operators *bitwise* (in columns) to binary numbers as in
  - 10010011 AND 00001111 = 00000011

Examples

- **AND** – if both bits are 1 the result is 1, otherwise 0
  - 11111101 **AND** 00001111 = 00001101
  - 01010101 **AND** 10101010 = 00000000
  - 00001111 **AND** 00110011 = 00000011
- **OR** – if either bit is 1 the result is 1, otherwise 0
  - 10101010 **OR** 11100011 = 11101011
  - 01010101 **OR** 10101010 = 11111111
  - 00001111 **OR** 00110011 = 00111111
- **NOT** – flip (negate) each bit
  - **NOT** 10101011 = 01010100
  - **NOT** 00001111 = 11110000
- **XOR** – if the bits differ the result is 1, otherwise 0
  - 10111100 **XOR** 11110101 = 01001001
  - 11110000 **XOR** 00010001 = 11100001
  - 01010101 **XOR** 01011110 = 00001011

Binary Addition

- To add 2 bits, there are four possibilities
  - 0 + 0 = 0
  - 1 + 0 = 1
  - 0 + 1 = 1
  - 1 + 1 = 2 – we can't write 2 in binary, but 2 is 10 in binary, so write a 0 and carry a 1
- To compute anything useful (more than 2 single bits), we need to add binary numbers
- This requires that we chain together carrys
  - The carry out of one column becomes a carry in in the column to its left

Binary Addition (cont'd)

- With 3 bits (the two bits plus the carry), we have 4 possibilities:
  - 0 + 0 + 0 = 0
  - 2 zeroes and 1 one = 1
  - 2 ones and 1 zero = 2 (carry of 1, sum of 0
  - 3 ones = 3 (carry of 1 and sum of 1)
- Example:

Carry:		1	1	0	0	Initial carry in is 0
X:		0	1	1	1	
Y:	+	0	1	1	0	
Sum:		1	1	0	1	
			Carry	Carry	No Carry	

Check your work, convert to decimal!

Binary Addition (cont’d)

- Example:

111+101 = ?

$$
\begin{array}{r}
111 \\
111 \\
+101 \\
\hline
1100
\end{array}
$$

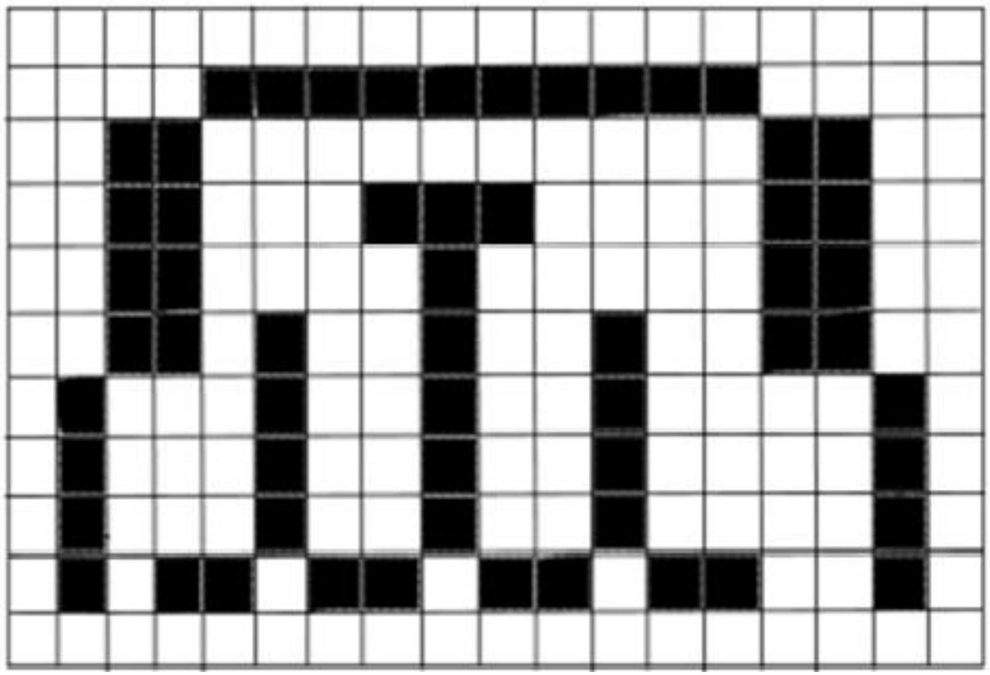
Network Addresses

- Internet Protocol (IP) version 4 uses 32-bit addresses
- Each of the numbers in this example represents 8 bits (or 1 byte) of the address, also known as an octet.
- 1 octet = 8 bits (0..255)
  - Each octet is separated by a period
- The address 10.251.136.253
  - Stored as 00001010.11111011.10001000.11111101 in binary
  - Omit the periods when storing the address in the computer

Image Files

- Images stored as sequences of pixels (picture elements)
  - row by row, each pixel is denoted by a value
- A 1024x1024 pixel image will comprise 1024 individual dots in one row for 1024 rows (1M pixels)
- This file is known as a bitmap
- In a black and white bitmap, we can store whether a pixel is white or black with 1 bit
  - The 1024x1024 image takes 1Mbit (1 megabit)
- A color image is stored using red, green and blue values
  - Each color can be between 0 and 255 (8 bits)
  - So each pixel takes 3 bytes
  - The 1024x1024 image takes 3MBytes

Black and white bitmap and its corresponding binary file



```
000000000000000000
000011111111110000
001100000000001100
001100011100001100
001100001000001100
001101001001001100
010001001001000010
010001001001000010
010001001001000010
010110110110110010
000000000000000000
```

Exercises

- Convert *223_{10} into binary system.*

- Q 1: How would you represent 10111 in the decimal number system?
  - A) 23
  - B) 24
  - C) 25
  - D) 22

Exercises

- Convert the following binary numbers to **decimal** and **hexadecimal** number system.
  - 11010101
  - 1110110
  - 00011
  - 100011