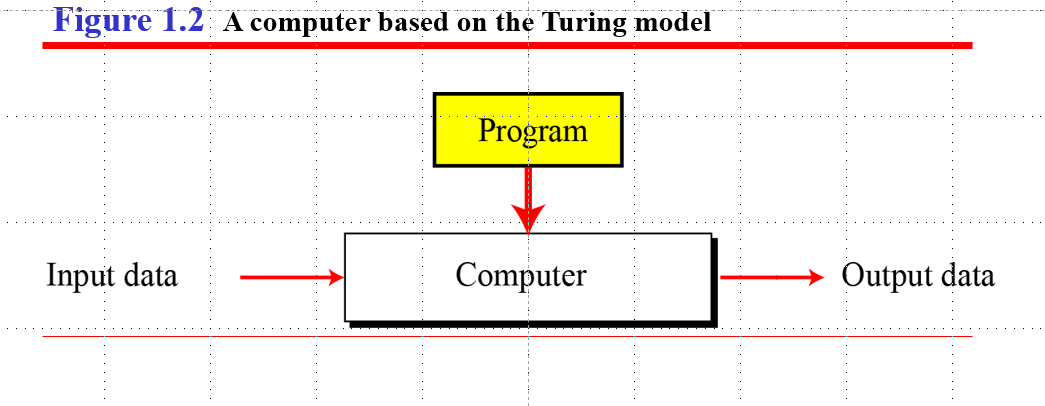
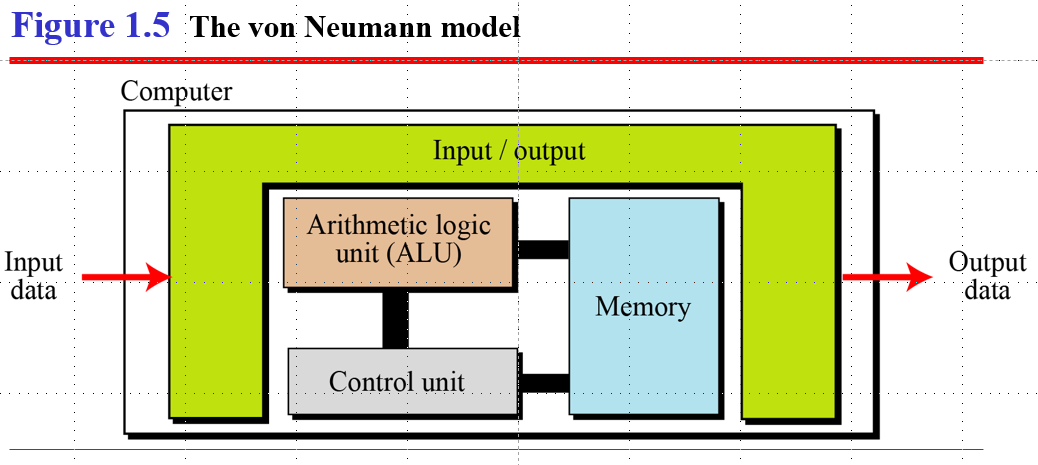
Summary chapters 1-5

# Chapter 1

Input -> Program -> Output



von Neumann is associated with using a single path to access a main memory that holds both instruction and data. Harvard is associated with having separate memories and access paths for instruction and data, allowing transfers to be performed at the same time.

The CPU consists of three parts:

1. ALU (Arithmetic Logic Unit) – Carries out Arithmetical and Logical Operations

2. Control Unit – Controls the activities of the CPU

3. Registers – Memory within the CPU

It is the electronic circuitry that carries out the instructions of a computer program, by performing basic arithmetic, logic, controlling and I/O operations specified by the instructions. It fetches input from the RAM, processes it, sends back computed results.

# Chapter 2

**Basex to Base10**=> 10102 => 1 \* x^position + 0 \* x^position + 1 \* x^position + 0 \* x^position=

8 + 0 + 2 + 0 = 1010

**Hexadecimal to Binary** => A9

1. Convert to Binary by splitting the numbers and converting them alone

A916 = 1010 10012

**Hexadecimal to Octal** => A9

1. Convert to Binary

A916 = 1010 10012

1. Split the Binary into 3 numbers each

1010 10012 = 10 101 0012

1. Convert those numbers to octal

10 101 0012 = 2518

**Decimal to Basex** => 112

1. Number mod x
2. Write the remainder
3. Use the amount of times you could divide it
4. Repeat step 1

11

112 mod 2 = 0

56 mod 2 = 0

28 mod 2 = 0

14 mod 2 = 0

7 mod 2 = 1

3 mod 2 = 1

1 mod 2 = 1

End => 11210 = 11100002 = 64 + 32 + 16 + 0 + 0 + 0 + 0

**Converting fractions** => 0.62510

1. Multiply by the base number you want
2. If it exceeds 1, note the number down and proceed with the fractional part  
   Else, note a 0
3. Repeat step 1

0.62510 to base2

1.250 = 1

.250

.500 = 0

1.00 = 1

.00

End => .1012

0.63410 to base8

5.072 = 5

.072

0.567 = 0

4.608 = 4

.608

4.864 = 4

0.864

6.912 = 6

.912

7.296 = 7

.296

2.368

…

…

End => .5044…

# Chapter 3

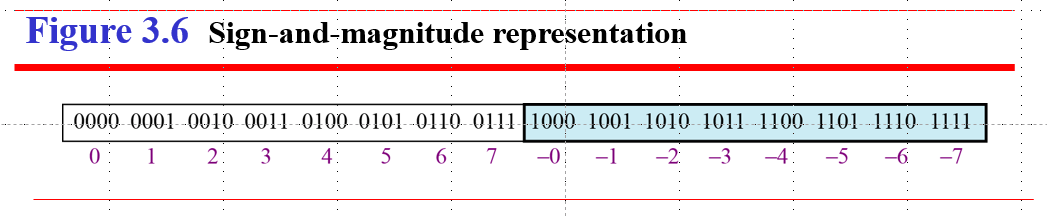
8-bit -> 00000000 (8 zero’s or one’s)

7 in 8-bit -> 00000111

258 in 16-bit -> 00000001 00000010

There is an overflow if you try to add a number that’s too big

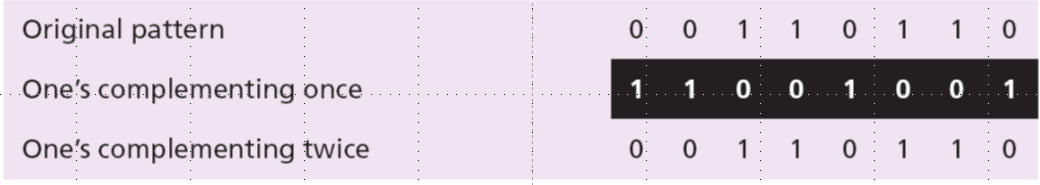
=> 8-bit can only store 255 at max (28 - 1)



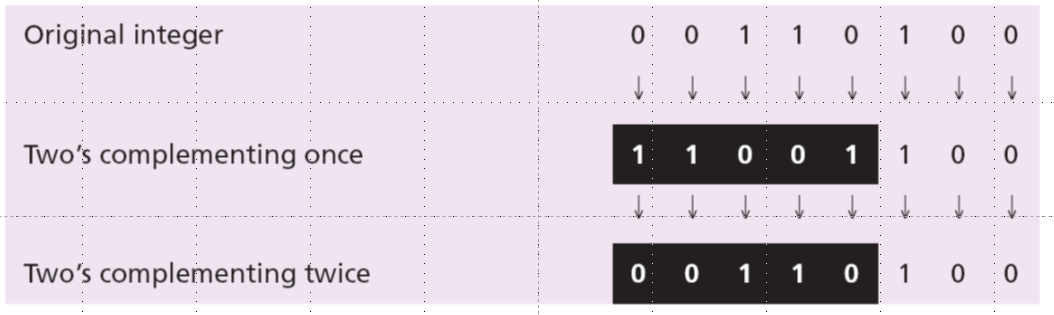
**Negative and positive numbers in 8 bits**

1. Change 8-bit to 7-bit
2. Put number in 7-bit
3. If positive, add 0 to the beginning  
   If negative, add 1 to the beginning

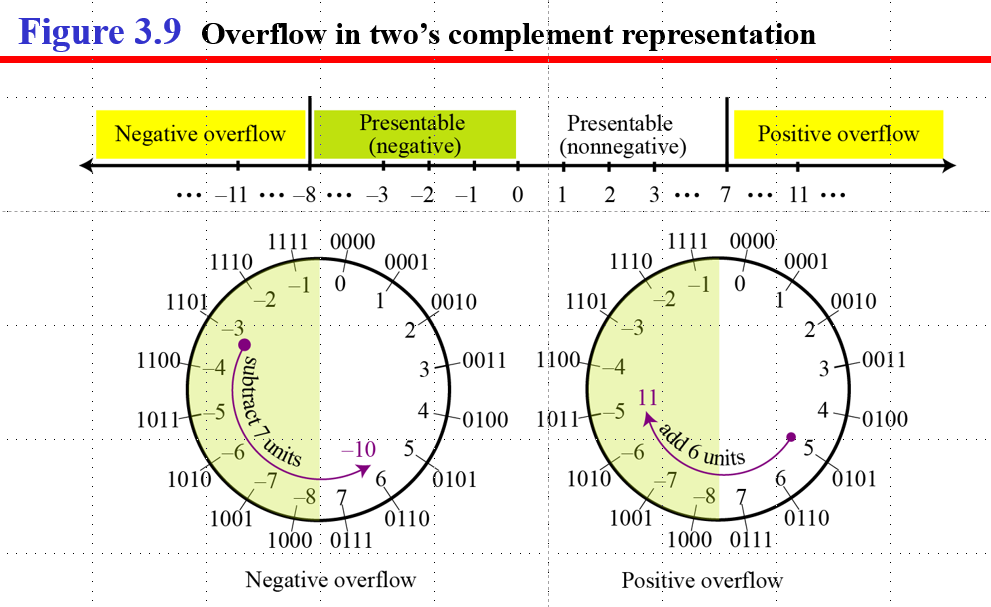
**One’s complement**

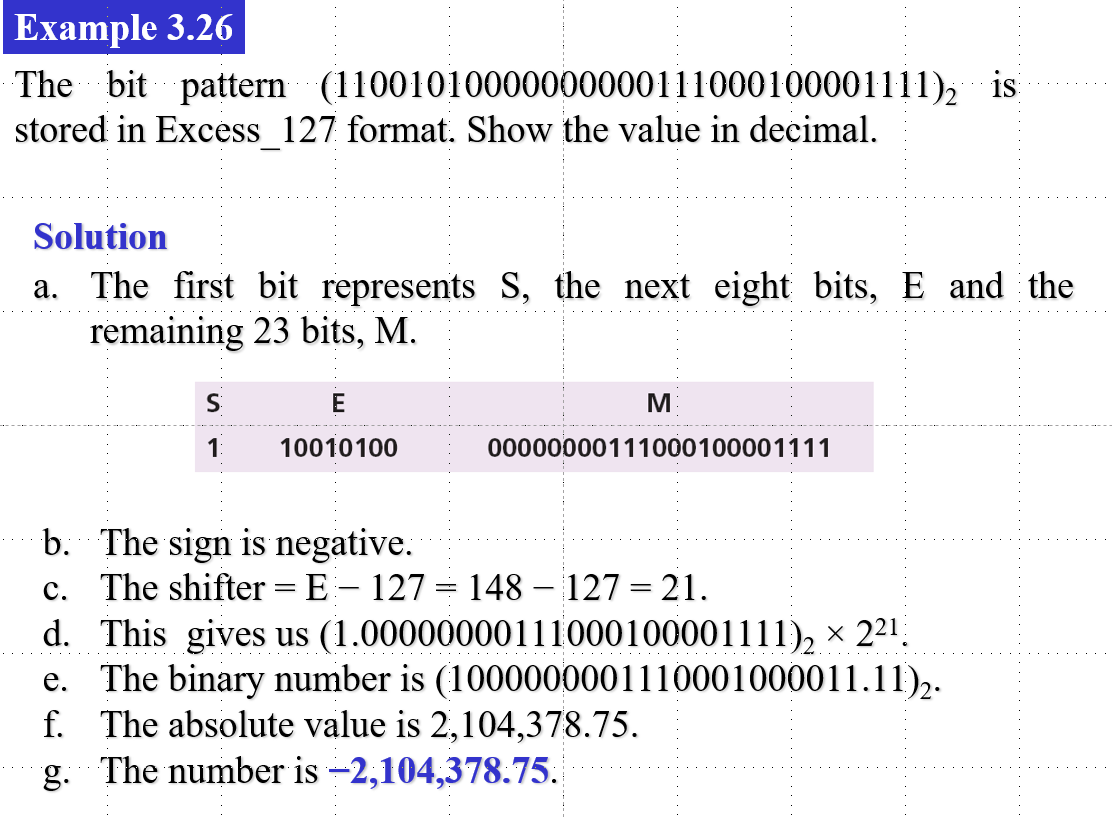


**Two’s complement**

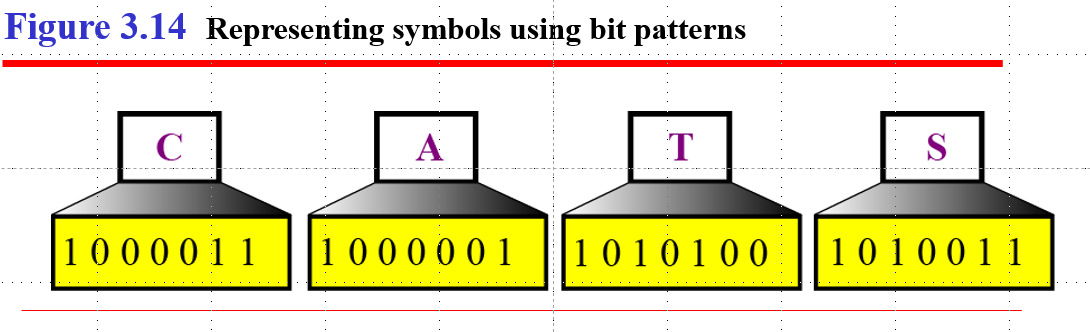


Two’s complement = one’s complement + 1





**Storing text**



**Storing audio**

Audio is different from text, it has infinite values. Thus, we record some of them. (This is called sampling; measuring a finite amount and record them).

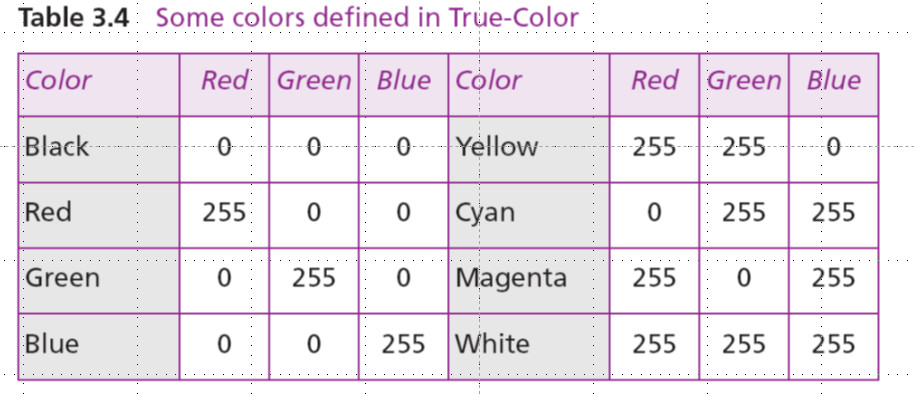
Quantization refers to storing values by rounding the number to the nearest integer.

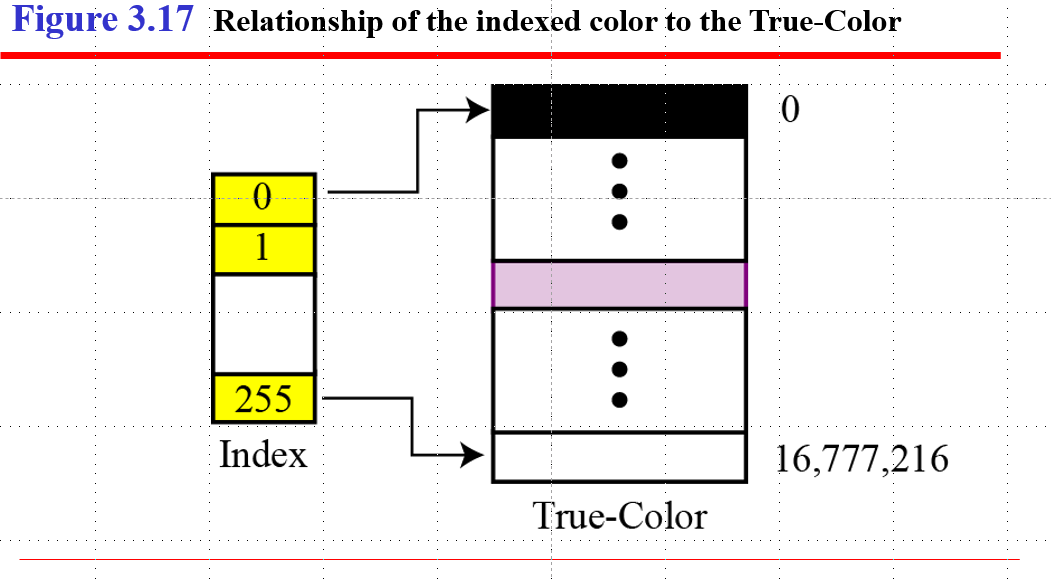
Bit rate = amount of samples per second \* bits per sample.

e.g. R = 40,000 \* 16 = 640,000 bits per second

Mp3 is the dominant standard for storing auio.

**Storing images**

Color is encoded in 24 bits

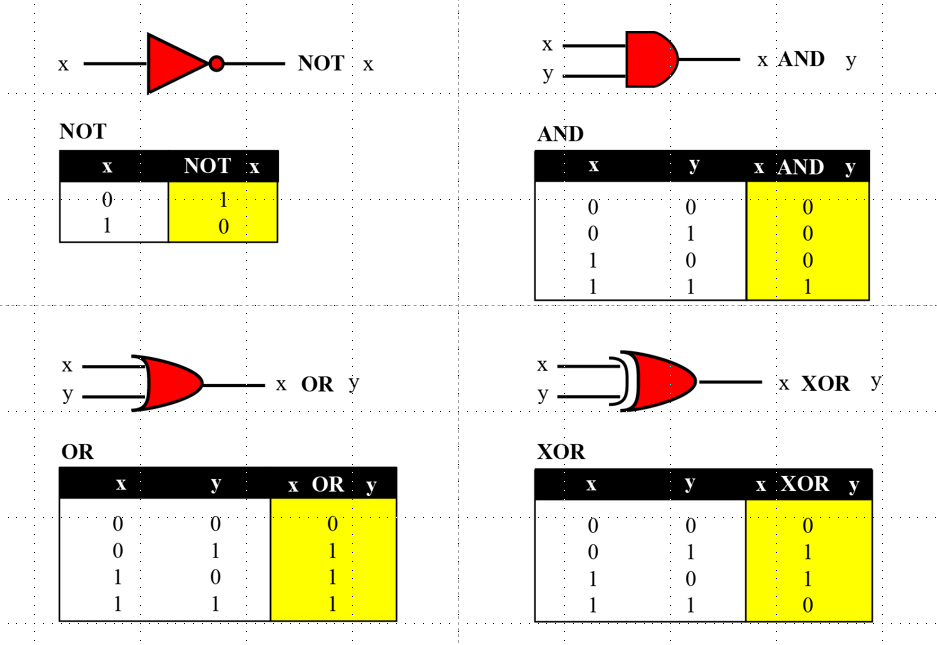


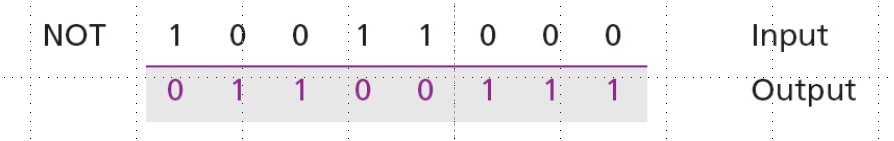
**Storing video**

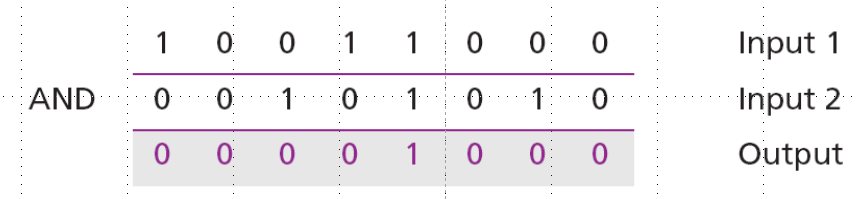
Video is basically a representation of images (frames). So, if we know how to store images, we know how to store a video. Each frame is transformed into a set of bit patterns and stored. The combination is then the video.

# Chapter 4

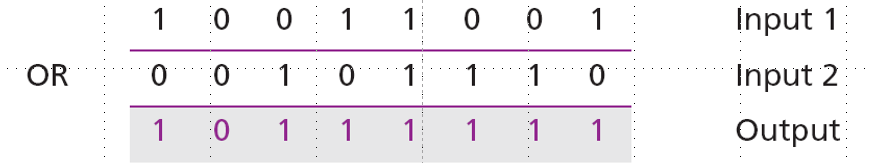
**Logic operations**



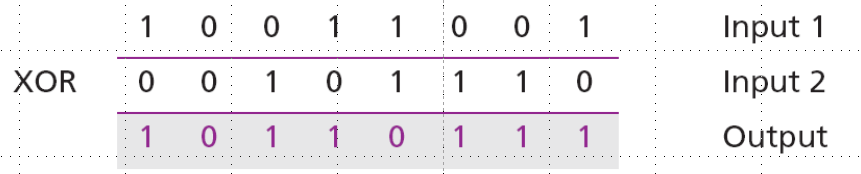
 Note: Flips every 0 to 1 and 1 to 0



Note: If there are two 1’s, return a 1



Note: If there is one 1, return a 1

 Note: If it has a 1 and 0, return 1, else return 0

NOT is called “Complementing”

AND is called “Unsetting”

OR is called “Setting”

XOR is called “Flipping”