INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

CSN-101 (Introduction to Computer Science and Engineering)

Lecture 14: Binary Number System

Dr. Sudip Roy

Assistant Professor

Department of Computer Science and Engineering

Piazza Class Room: https://piazza.com/iitr.ac.in/fall2019/csn101

[Access Code: csn101@2019]

Moodle Submission Site: https://moodle.iitr.ac.in/course/view.php?id=45

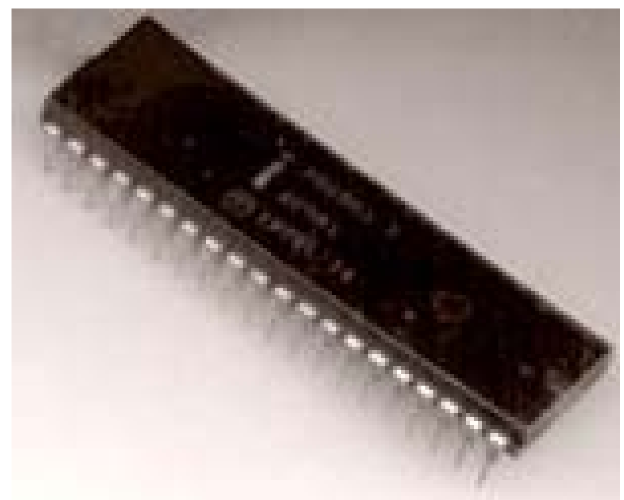
[Enrollment Key: csn101@2019]

Plan for Lecture Classes in CSN-101 (Autumn, 2019-2020)

Week	Lecture 1 (Monday 4-5 PM)	Lecture 2 (Friday 5-6 PM)	
1	Evolution of Computer Hardware and Moore's Law, Software and Hardware in a Computer	Computer Structure and Components, Operating Systems	MTE
2	Computer Hardware: Block Diagrams, List of Components	Computer Hardware: List of Components, Working Principles in Brief, Organization of a Computer System	
3	Linux OS	Linux OS	
4	Writing Pseudo-codes for Algorithms to Solve Computational Problems	Writing Pseudo-codes for Algorithms to Solve Computational Problems	ETE
5	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	
6	C Programming	C Programming	
7	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	
8	Number Systems: Negative number representation, Fractional (Real) number representation	Boolean Logic: Boolean Logic Basics, De Morgan's Theorem, Logic Gates: AND, OR, NOT, NOR, NAND, XOR, XNOR, Truth-tables	
9	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	MTE
10	Different layers of networking, Network components, Type of networks	Network topologies, MAC, IP Addresses, DNS, URL	
11	Different fields of CSE: Computer Architecture and Chip Design	Different fields of CSE: Data Structures, Algorithms and Programming Languages	ETE
12	Different fields of CSE: Database management	Different fields of CSE: Operating systems and System softwares	
13	Different fields of CSE: Computer Networking, HPCs, Web technologies	Different Applications of CSE: Image Processing, CV, ML, DL	Term Project
14	Different Applications of CSE: Data mining, Computaional Geometry, Cryptography, Information Security	Different Applications of CSE: Cyber-physical systems and IoTs	

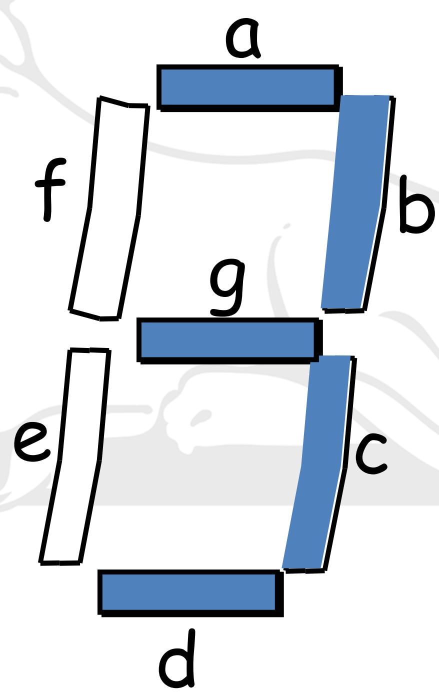
CPU processes binary number

- The first microprocessor to make it into a home computer was the Intel 8080, a complete 8-bit computer on one chip, introduced in 1974.

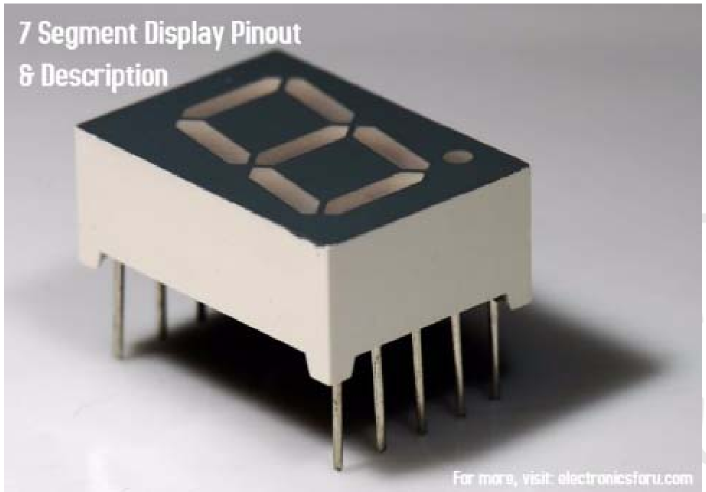


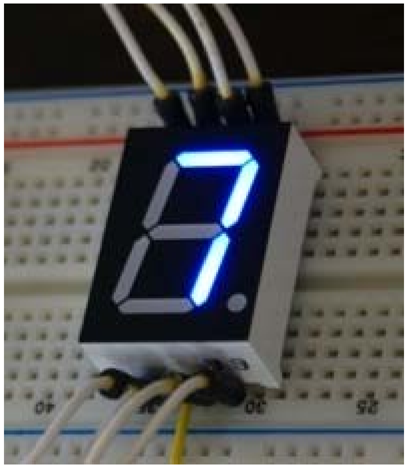
Seven-segment display:

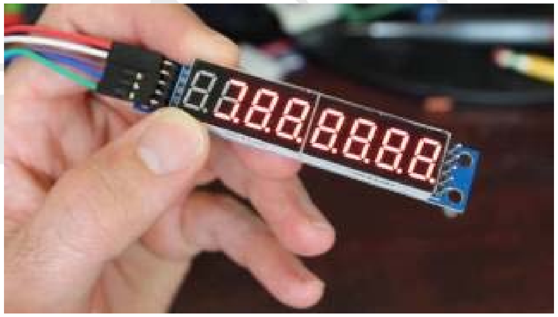
- **Seven-segment display:**
  - 7 LEDs (light emitting diodes), each one controlled by an input
  - 1 means "on", 0 means "off"
  - Display digit "3"?
    - Set a, b, c, d, g to 1
    - Set e, f to 0



7-Segment Display:







7-Segment Display:

What's A 7-Segment Display?

A 7-segment display is a package with 7 bar-shaped LEDs arranged to allow the display of many useful digits and some letters.

Each segment (labeled A-G) contains an LED which may be individually controlled. DP is an eighth LED, the decimal point.

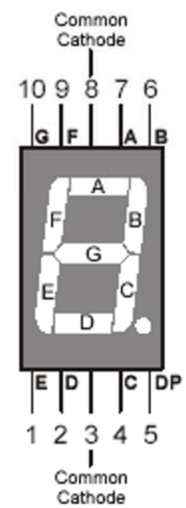


Figure 6-2
7-Segment Display Part Drawing and Pin Map

7-Segment Display:

Common cathode means that each segment's cathode is connected to common pins – 3 & 8, allowing the anode of each to be connected to the controller.

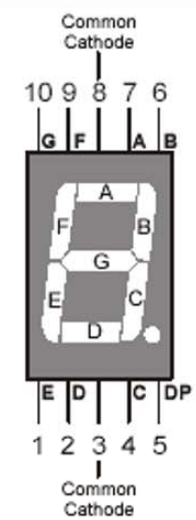


Figure 6-2
7-Segment Display Part Drawing and Pin Map

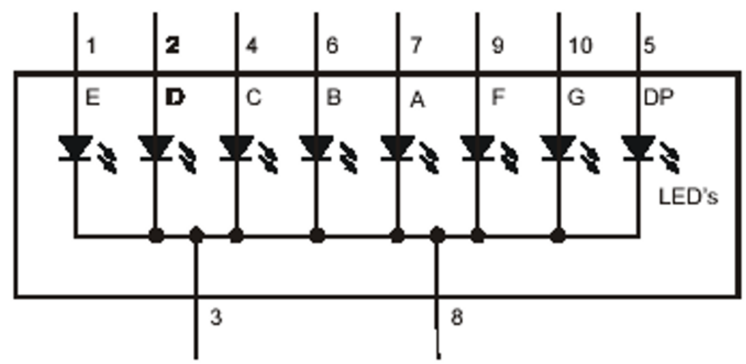
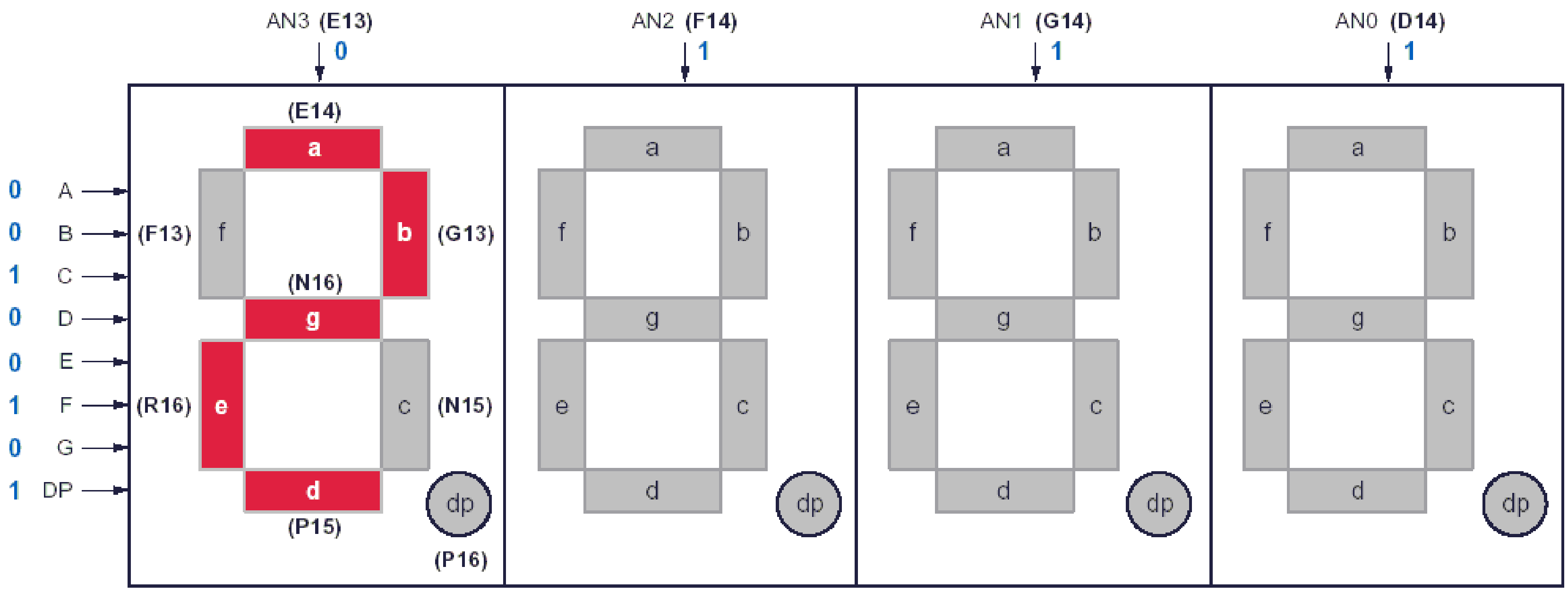
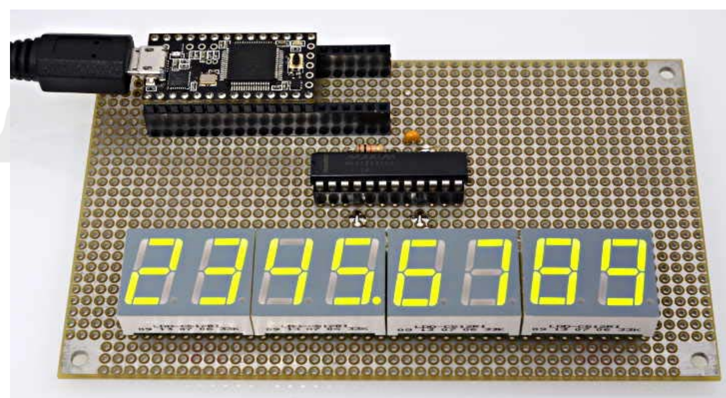


Figure 6-3
7-Segment Schematic

Array of 7-Segment Displays:

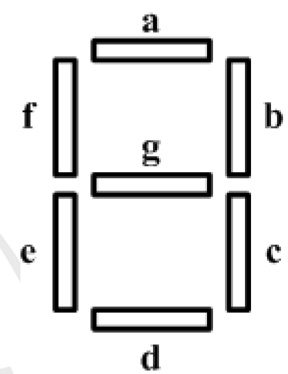


Seven-Segment LED Digit Control



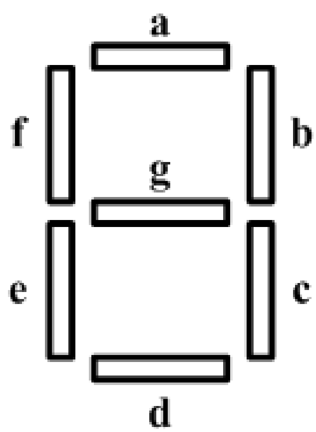
Specification:

- Input: A 4-bit binary value that is a BCD coded input.
- Outputs: 7 bits, a through g for each of the segments of the display.
- Operation: Decode the input to activate the correct segments.



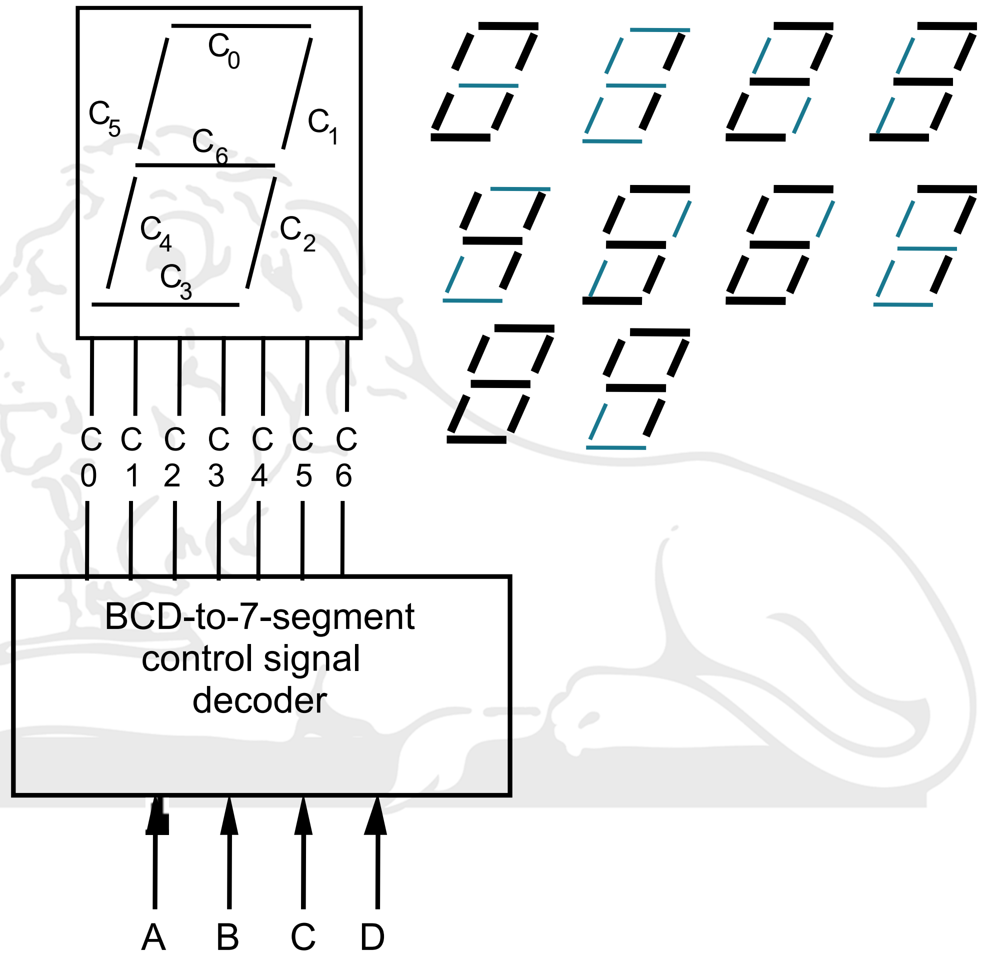
Formulation:

- Construct a truth table



Decimal Digit	Input BCD	Seven-Segment Decoder Outputs a b c d e f g
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	1 0 1 1 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	1 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 1 0 1 1
All other inputs		0 0 0 0 0 0 0

7-Segment Decoder:



The computer memory and the binary number system

Memory devices

- A memory device is a gadget that helps you record information and recall the information at some later time.

Example:

Memory devices (cont.)

- Requirement of a memory device:

• A memory device must have more than 1 states

(Otherwise, we can't tell the difference)

Example:

Memory device in state 0

Memory device in state 1

The switch is a *memory device*

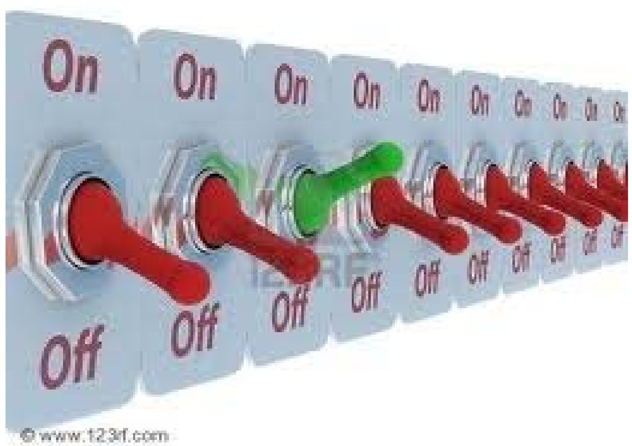
- The electrical switch is a memory device:

- The electrical switch can be in one of these 2 states:

  - off (we will call this state 0)
  - on (we will call this state 1)

Memory cell used by a computer

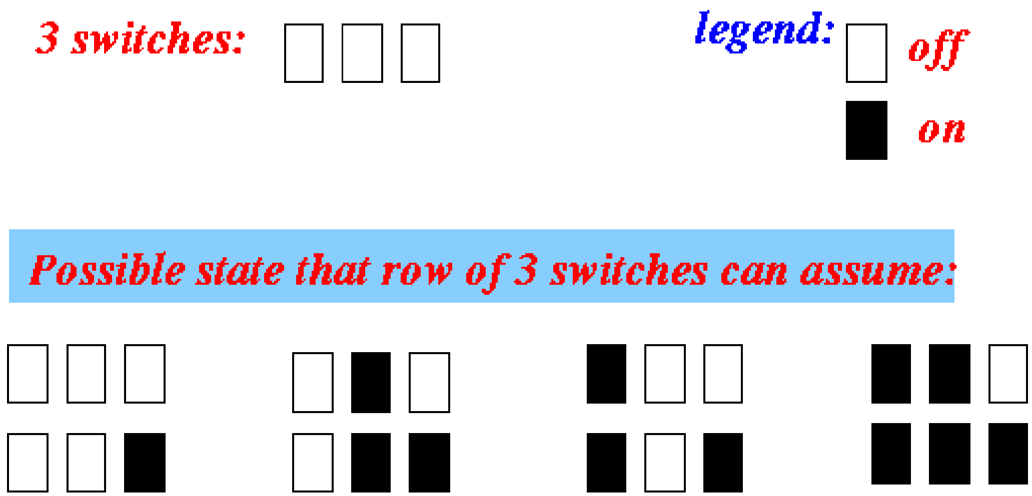
- *One* switch can be in one of 2 states
- A row of *n* switches:



can be in one of 2^n states !

Memory cell used by a computer (cont.)

- Example: row of 3 switches



- A row of 3 switches can be in one of $2^3 = 8$ states.
- The 8 possible states are given in the figure above.

Representing numbers using a row of switches

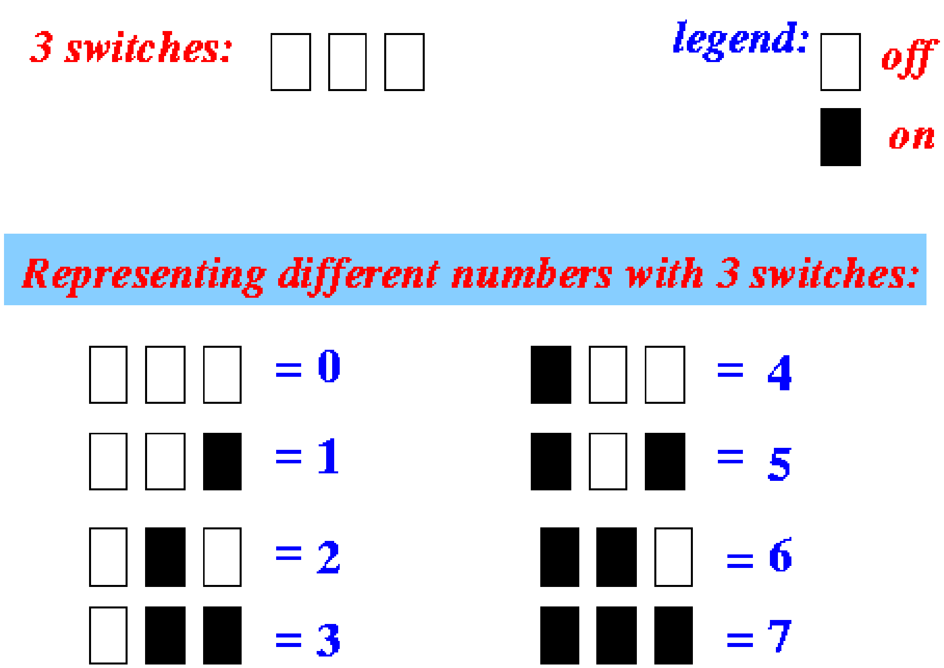
- We saw how information can be represented by *number* by using a code (agreement)
- Recall: we can use numbers to represent marital status information:

> - 0 = single
> - 1 = married
> - 2 = divorced
> - 3 = widowed

Representing numbers using a row of switches (cont.)

- We can represent each number using a different state of the switches.

Example:



Representing numbers using a row of switches (cont.)

- To complete the knowledge on how information is represented inside the computer, we will now study:

  - How to use the different states of the switches to represent different *numbers*

- The representation scheme has a name:

  - the binary number system

The *binary number* system

- The binary number system uses 2 digits to encode a number:

  - 0 = represents no value
  - 1 = represents a unit value

- That means that you can *only* use the digits 0 and 1 to write a *binary number*

  – Example: some binary numbers

  - 0
  - 1
  - 10
  - 11
  - 1010
  - and so on.

The *binary number* system (cont.)

- The value that is *encoded (represented)* by a binary number is computed as follows:

Binary number	Value encoded by the binary number
$d_{n-1}\ d_{n-2}\ \ldots\ d_1\ d_0$	$d_{n-1}\times 2^{n-1} + d_{n-2}\times 2^{n-2} + \ldots + d_1\times 2^1 + d_0\times 2^0$

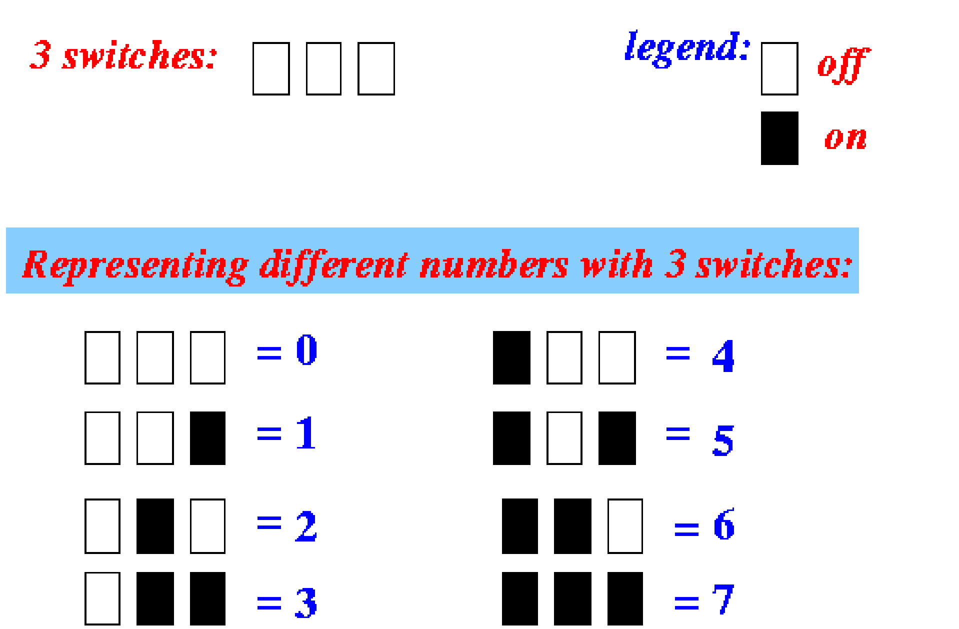
The *binary number* system (cont.)

Example:

Binary number	Value encoded by the binary number
0	$0 \times 2^0 = 0$
1	$1 \times 2^0 = 1$
10	$1 \times 2^1 + 0 \times 2^0 = 2$
11	$1 \times 2^1 + 1 \times 2^0 = 3$
1010	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 2 = 10$

The *binary number* system (cont.)

- Now you should understand how the different states of these 3 switches represent the numbers 0-7 using the binary number system:



A cute *binary number* joke

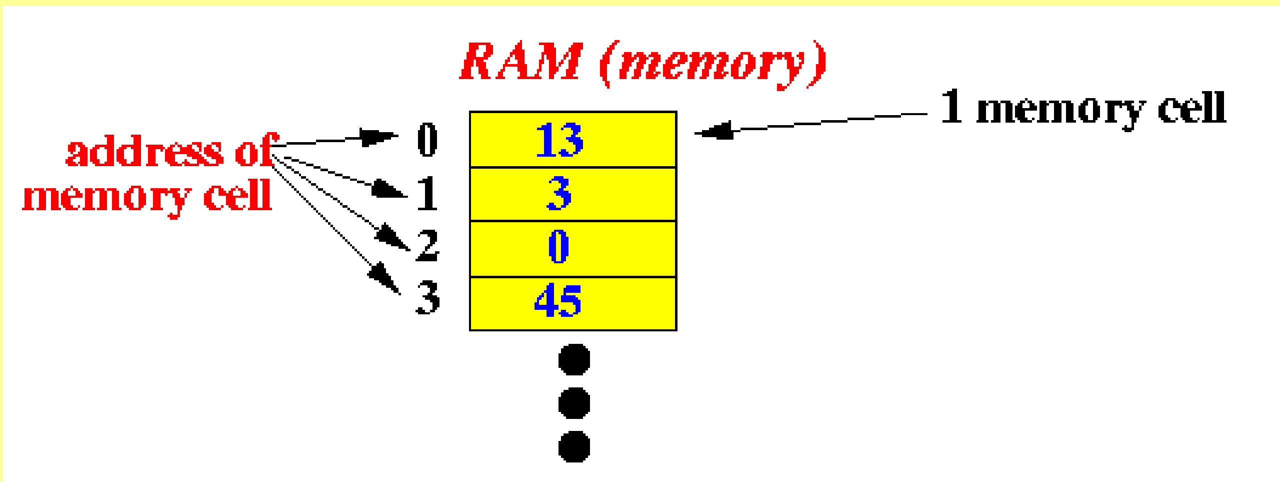
- Try to understand this joke:

(Read: there are binary 10 (= 2) types of people: those who understand binary (numbers) and those who don't)

What does all this have to do with a computer ?

- Recall what we have learned about the Computer RAM memory:

- The RAM consists of multiple memory cells:



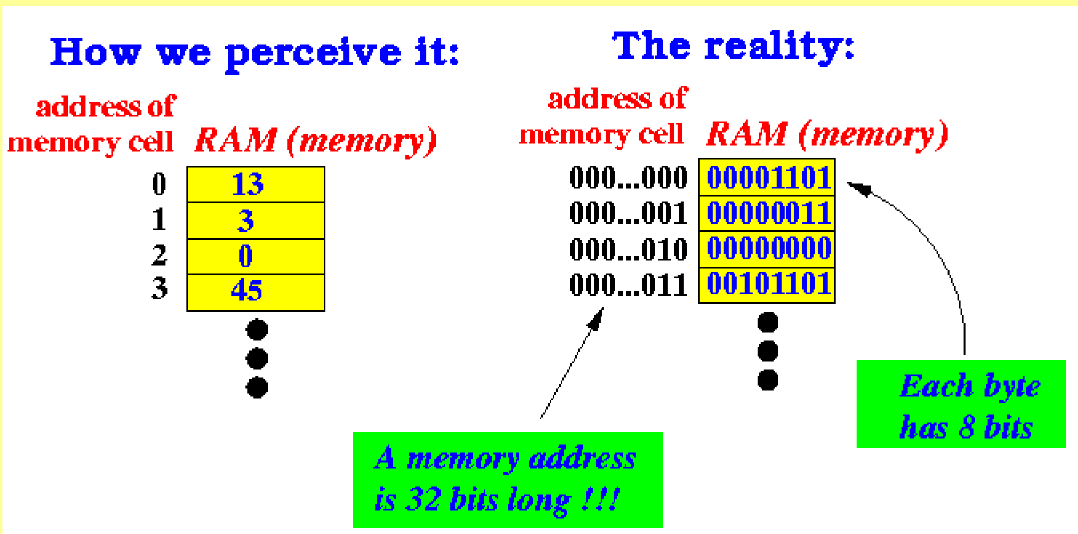
Each

What does all this have to do with a computer ? (cont.)

- The connection between the computer memory and the binary number system is:

• The computer system uses the binary number encoding to store the number

Example:



What does all this have to do with a computer ? (cont.)

- *Note*: the address is also expressed as a binary number

A computer can have over 4,000,000,000 bytes (4 Gigabytes) of memory.

So we need a 32 bites to express the address

Computer memory

- A computer is an electronic device
- Structure of a RAM memory:

> • The RAM memory used by a computer consists of a large number of electronic switches
>
> • The switches are organized in rows
>
> • For historical reason, the number of switches in one row is 8

Computer memory (cont.)

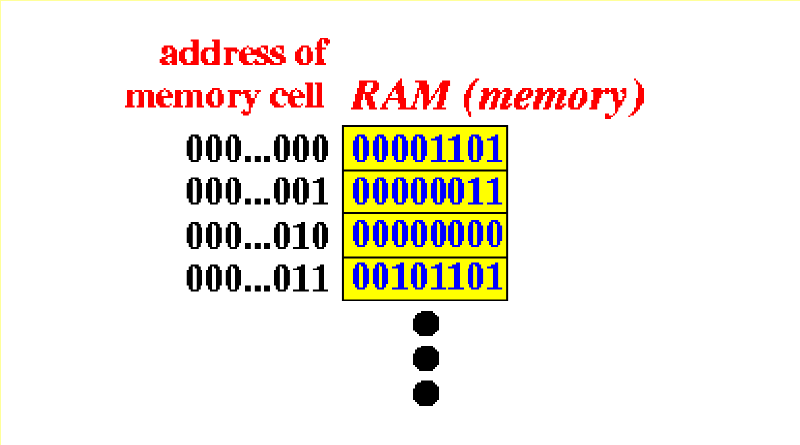
Details

- In order to store text information in a computer, we need to encode:
    - 26 upper case letters ('A', 'B', and so on)
    - 26 lower case letters ('a', 'b', and so on)
    - 10 digits ('0', '1', and so on)
    - 20 or so special characters ('&', '%', '$', and so on)

  for a total of about 100 different symbols
- The nearest *even* power 2^n that is larger than 100 is:
    - $2^7 = 128 \geq 100$
- For a reason beyond the scope of this course, an 8^{th} switches is added

Computer memory (cont.)

- This is was a portion of the RAM memory looks like:



- What information is stored in the RAM memory depends on:

• The type of data (this is the context information)

Example of types: marital status, gender, age, salary, and so on.

• This determines the encoding scheme used to interpret the number

Computer memory *jargon*:

- bit = (binary digit) a *smallest* memory device

  A bit is in fact a switch that can remember 0 or 1
- (The digits 0 and 1 are digits used in the binary number system)

- Byte = 8 bits

  A byte is in fact one row of the RAM memory

- KByte = kilo byte = 1024 (= 2^{10}) bytes (approximately 1,000 bytes)
- MByte = mega byte = 1048576 (= 2^{20}) bytes (approximately 1,000,000 bytes)
- GByte = giga byte = 1073741824 (= 2^{30}) bytes (approximately 1,000,000,000 bytes)
- TByte = tera byte

Combining adjacent memory cells

- A byte has 8 bits and therefore, it can store:

> • $2^8 = 256$ different patterns
>
> (These 256 patterns are: 00000000, 00000001, 00000010, 00000011, ... 11111111)

Combining adjacent memory cells (cont.)

- Each pattern can are encoded exactly one number:

  - 00000000 = 0
  - 00000001 = 1
  - 00000010 = 2
  - 00000011 = 3
  - ...
  - 11111111 = 255

Therefore, one byte can store one of 256 possible values

(You can store the number 34 into a byte, but you cannot store the number 456, the value is out of range)

Combining adjacent memory cells (cont.)

- Exploratory stuff:

- The following computer program illustrates the effect of the out of range phenomenon:

```
public class test
{
public static void main(String args[])
   {
             byte x = (byte) 556;
             System.out.println(x);
    }
}
```

1 0 0 0 1 0 1 1 0 0

Combining adjacent memory cells (cont.)

- Compile and run:

```
>> javac test.java
>> java test
44
```

1000

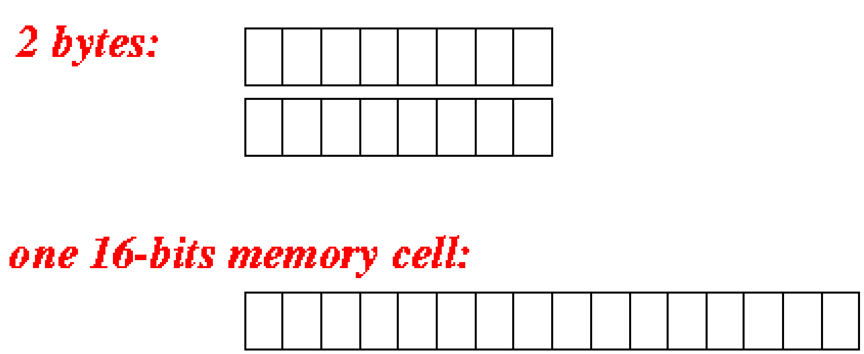
- This phenomenon is called overflow (memory does not have enough space to represent the value)

  This is the *same* phenomenon when you try to compute 1/0 with a calculator; except that the calculator was programmed (by the manufacturer) to reported the error (and the computer is *not*).

Combining adjacent memory cells (cont.)

- The computer can combine adjacent bytes (memory cells) and use it as a larger memory cell

  Schematically:

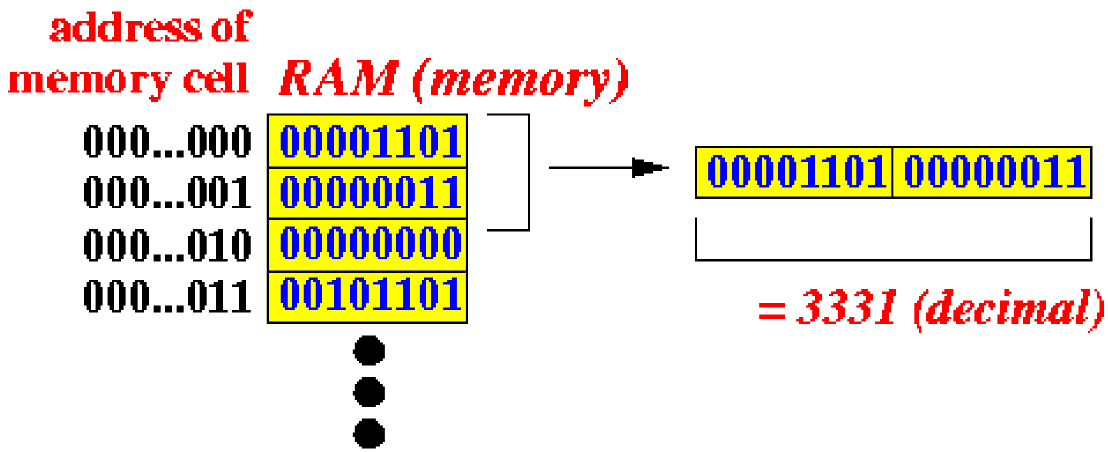


A 16 bits memory cell can store one of $2^{16} = 65536$ different patterns.

Therefore, it can represent (larger) numbers ranging from: 0 – 65535.

Combining adjacent memory cells (cont.)

- Example: how a computer can use 2 *consecutive* bytes as a 16 bits memory cell:



- The bytes at address 0 and address 1 can be interpreted as a 16 bits memory cell (with address 0)

Combining adjacent memory cells (cont.)

- When the computer accesses the RAM memory, it specifies:

  - The memory location (address)
  - The number of bytes it needs

Combining adjacent memory cells (cont.)

- The computer can also:

  - combine 4 *consecutive* bytes and use them as a 32 bits memory cell

    - Such a memory call can represent numbers ranging from: $0 - (2^{32}-1)$ or $0 - 4294967295$

  - combine 8 *consecutive* bytes and use them as a 64 bits memory cell

    - Such a memory call can represent numbers ranging from: $0 - (2^{64}-1)$ or $0 - 18446744073709551615$

Combining adjacent memory cells (cont.)

- There is no need (today) to combine 16 *consecutive* bytes and use them as a 128 bits memory cell
- But this may change in the future...

Bridging the Digital Divide