Lecture Outline

- Components of the Computer
- Different usages of the computers
- Von Neumann Architecture
- Data and information
- How to store the data in a computer
- Basic Data Types
- Number Representation
- Floating Point Representation
- Data Representation in Computers
- Binary Code Transmission
Lecture Objectives

- Differentiate between data and Information
- Identify the basic functions of the computer
- Data representation inside computers
COMPONENTS OF THE COMPUTER

1. Monitor
2. Motherboard
3. CPU (Microprocessor)
4. Primary storage (RAM)
5. Expansion cards
6. Power supply
7. Optical disc drive
8. Secondary storage (Hard disk)
9. Keyboard
10. Mouse
# A Modern Computer System

<table>
<thead>
<tr>
<th>Banking system</th>
<th>Airline reservation</th>
<th>Web browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilers</td>
<td>Editors</td>
<td>Command interpreter</td>
</tr>
<tr>
<td>Operating system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microarchitecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical devices</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Application programs**
- **System programs**
- **Hardware**

Image Credit: Modern Operating System by Andrew S Tanenbaum
A MODERN COMPUTER SYSTEM

Figure 1.2  A modern computer system.

Image Credit: Operating System Concepts by Silberschatz et. al.
All of these computers are based on the same architecture proposed by Von Neumann in 1945.
Usage of the Computers
Examples of computer applications in society

- Education
- Finance
- Government
- Healthcare
- Science
- Publishing
- Travel
- Industry
VON NEUMANN ARCHITECTURE

- All computers share the same basic architecture, whether it be a multi-million dollar mainframe or a Palm Pilot.

- All have memory, an I/O system, and arithmetic/logic unit, and a control unit.
VON NEUMANN ARCHITECTURE
Main components of the Von Neumann architecture

(1) **CPU** – Central Processing Unit
   (1) **arithmetic-logic unit (ALU)**
       performs the computer's computational and logical functions

(2) **control unit**
    directs other components of the computer to perform certain actions

(3) **input and output** devices
    main-machine interfaces; i.e.,

(2) **memory**
    the computer's main, or fast, memory, such as random access memory (RAM)
The Computer: Top-Level Structure

The Central Processing Unit (CPU)

Image Credit: Computer Organization & Architecture by William Stallings
**DATA AND INFORMATION**

What is Data?

- Data is a collection of facts
  - Numbers
  - Words
  - Measurements
  - Observations
  - Description of things
What is Information?
- processed outcome of data. (it is derived from data)
**Qualitative vs. Quantitative Data**

- Quantitative Data: Numerical Information
- Qualitative Data: Descriptive data

"It was great fun"

Discrete: 5
Continuous: 3.265...
HOW DO WE STORE DATA IN A COMPUTER?
HOW DO WE STORE DATA IN A COMPUTER?

- Computer is a digital device which is capable to handle discrete data/information
- Therefore, we need to convert these analog signals to digital signals after capturing them by the input devices
- This is done with the converter – Analogue to Digital Convertor (ADC)
- Digital signals are represented by numbers (Binary Digits) and need to be stored in the main memory
How do we store data in a computer?
How do we store data in a computer?

- 8 Bits = 1 Byte
- 1024 Bytes = 1 KB
- 1024 KB = 1 MB
- 1024 MB = 1 GB
- 1024 GB = 1 TB
How do we store data in a computer?

- The memory is made up of BYTES
- Each BYTE can be addressed uniquely
- When the address is expressed in Binary, the number of maximum BITs used to write the address specifies the total number of locations available
- If n number of BITs are available then the total number of locations available is $2^n$
- If we have 32 BITs then we can have 4GB of Memory ($2^{32} = 4$ GB)
HOW DO WE STORE DATA IN A COMPUTER?

- Our pc’s are having 512MB of main memory (RAM).
  - How many bits are used to address the memory locations of the PC?
**PROCESSING OF THE DATA**

Instructions + Data → Eg. Sort the name in Alphabetical order → Binary → Memory

- Convert

- Example: Sort the name in alphabetical order
Programmers Point of View

- Programmers need to use data in their programs.
- The architecture says, it is required to store them in the main memory before use.
- Therefore, it is required to find a way to put them in memory.
- As there may have differences in the data, he sets his requirement through what we call a DATA TYPE.
- Through a data type, architecture tells the computer that data must be stored in a particular way in the main memory.
- For that, the data must have a representation.
Basic Data Types: Character Data

- Numeric
  - 0 1 2 ... 9

- Alphabetic
  - a b c ...... z

- Special
  - # @ % ( $ &
Basic Data Types: Numeric Data

- **Integer**
  - + & - whole numbers
  - 4251, -582
  - Most significant bit
  - Least significant bit

- **Real**
  - All numbers including everything between integers
  - 0.23, 0, 5½, -2.3,
NUMBER REPRESENTATION

- Fixed Point Representation
  - 12.548

- Floating Point Representation
  - Scientific Notation
    - 12.054 → 1.2054 * 10\(^1\)
  - Computer Notation
    - 12.65 → 0.1265*10\(^2\)
FLOATING POINT REPRESENTATION

Mantissa/argument

15.23 * 10^4

Radix/base

Exponent
DATA REPRESENTATION IN COMPUTERS

- How do computers represent data?
- Most computers are digital
  - Recognize only **two discrete states**: on or off
  - Computers are electronic devices powered by electricity, which has only two states, on or off

```
1 1 0 0 1 1 1 0
```
Data Representation in Computers

- **Binary representation**
  - A number system that has just two unique digits, 0 and 1
  - The two digits represent the two off and on states

<table>
<thead>
<tr>
<th>Binary Digit (bit)</th>
<th>Electronic Charge</th>
<th>Electronic State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>
DATA REPRESENTATION IN COMPUTERS

- BCD (Binary Coded Decimal)
  - 4 bit code for numeric values only
  - 9 → 1001
DATA REPRESENTATION IN COMPUTERS

- ASCII (American Standard Code for Information Interchange)
  - 7 bit code for all 128 characters
  - A = 1000001

- EBCDIC (Extended BCD Interchange Code)
  - 8 bit ASCII
# Data Representation in Computers

## ASCII Alphabet Characters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>65</td>
<td>01000001</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>01000010</td>
</tr>
<tr>
<td>C</td>
<td>67</td>
<td>01000011</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>01000100</td>
</tr>
<tr>
<td>E</td>
<td>69</td>
<td>01000101</td>
</tr>
<tr>
<td>F</td>
<td>70</td>
<td>01000110</td>
</tr>
<tr>
<td>G</td>
<td>71</td>
<td>01000111</td>
</tr>
<tr>
<td>H</td>
<td>72</td>
<td>01001000</td>
</tr>
<tr>
<td>I</td>
<td>73</td>
<td>01001001</td>
</tr>
<tr>
<td>J</td>
<td>74</td>
<td>01001010</td>
</tr>
<tr>
<td>K</td>
<td>75</td>
<td>01001011</td>
</tr>
<tr>
<td>L</td>
<td>76</td>
<td>01001100</td>
</tr>
<tr>
<td>M</td>
<td>77</td>
<td>01001101</td>
</tr>
<tr>
<td>N</td>
<td>78</td>
<td>01001110</td>
</tr>
<tr>
<td>O</td>
<td>79</td>
<td>01001111</td>
</tr>
<tr>
<td>P</td>
<td>80</td>
<td>01010000</td>
</tr>
<tr>
<td>Q</td>
<td>81</td>
<td>01010001</td>
</tr>
<tr>
<td>R</td>
<td>82</td>
<td>01010010</td>
</tr>
<tr>
<td>S</td>
<td>83</td>
<td>01010011</td>
</tr>
<tr>
<td>T</td>
<td>84</td>
<td>01010100</td>
</tr>
<tr>
<td>U</td>
<td>85</td>
<td>01010101</td>
</tr>
<tr>
<td>V</td>
<td>86</td>
<td>01010110</td>
</tr>
<tr>
<td>W</td>
<td>87</td>
<td>01010111</td>
</tr>
<tr>
<td>X</td>
<td>88</td>
<td>01011000</td>
</tr>
<tr>
<td>Y</td>
<td>89</td>
<td>01011001</td>
</tr>
<tr>
<td>Z</td>
<td>90</td>
<td>01011010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>97</td>
<td>01100001</td>
</tr>
<tr>
<td>b</td>
<td>98</td>
<td>01100010</td>
</tr>
<tr>
<td>c</td>
<td>99</td>
<td>01100011</td>
</tr>
<tr>
<td>d</td>
<td>100</td>
<td>01100100</td>
</tr>
<tr>
<td>e</td>
<td>101</td>
<td>01100101</td>
</tr>
<tr>
<td>f</td>
<td>102</td>
<td>01100110</td>
</tr>
<tr>
<td>g</td>
<td>103</td>
<td>01100111</td>
</tr>
<tr>
<td>h</td>
<td>104</td>
<td>01101000</td>
</tr>
<tr>
<td>i</td>
<td>105</td>
<td>01101001</td>
</tr>
<tr>
<td>j</td>
<td>106</td>
<td>01101010</td>
</tr>
<tr>
<td>k</td>
<td>107</td>
<td>01101011</td>
</tr>
<tr>
<td>l</td>
<td>108</td>
<td>01101100</td>
</tr>
<tr>
<td>m</td>
<td>109</td>
<td>01101101</td>
</tr>
<tr>
<td>n</td>
<td>110</td>
<td>01101110</td>
</tr>
<tr>
<td>o</td>
<td>111</td>
<td>01101111</td>
</tr>
<tr>
<td>p</td>
<td>112</td>
<td>01110000</td>
</tr>
<tr>
<td>q</td>
<td>113</td>
<td>01110001</td>
</tr>
<tr>
<td>r</td>
<td>114</td>
<td>01110010</td>
</tr>
<tr>
<td>s</td>
<td>115</td>
<td>01110011</td>
</tr>
<tr>
<td>t</td>
<td>116</td>
<td>01110100</td>
</tr>
<tr>
<td>u</td>
<td>117</td>
<td>01110101</td>
</tr>
<tr>
<td>v</td>
<td>118</td>
<td>01110110</td>
</tr>
<tr>
<td>w</td>
<td>119</td>
<td>01110111</td>
</tr>
<tr>
<td>x</td>
<td>120</td>
<td>01111000</td>
</tr>
<tr>
<td>y</td>
<td>121</td>
<td>01111001</td>
</tr>
<tr>
<td>z</td>
<td>122</td>
<td>01111010</td>
</tr>
</tbody>
</table>
DATA REPRESENTATION IN COMPUTERS

- ASCII (American Standard Code for Information Interchange)
  - 7 bit code for all 128 characters
  - A = 1000001 → 65
**Data Representation in Computers**

- **Unicode** provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.

- Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one.

- 8, 16 or 32 bits per character
DATA REPRESENTATION IN COMPUTERS

Independent vowels

<table>
<thead>
<tr>
<th>Code</th>
<th>Sinhala Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0D85</td>
<td>அ அயனா (Ayanna)</td>
</tr>
<tr>
<td>0D86</td>
<td>ஆ ஆயனா (Aayanna)</td>
</tr>
<tr>
<td>0D87</td>
<td>இ இயனா (Iyanna)</td>
</tr>
<tr>
<td>0D88</td>
<td>ஈ ஈயனா (Ieyanna)</td>
</tr>
<tr>
<td>0D89</td>
<td>இய இயனா (Iyanna)</td>
</tr>
</tbody>
</table>

Tamil Vowels

<table>
<thead>
<tr>
<th>Code</th>
<th>Tamil Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0B85</td>
<td>தமிழ் வெறுமை (A)</td>
</tr>
<tr>
<td>0B86</td>
<td>தமிழ் வெறுமை (A)</td>
</tr>
<tr>
<td>0B87</td>
<td>தமிழ் வெறுமை (I)</td>
</tr>
<tr>
<td>0B88</td>
<td>தமிழ் வெறுமை (I)</td>
</tr>
<tr>
<td>0B89</td>
<td>தமிழ் வெறுமை (U)</td>
</tr>
<tr>
<td>0B8A</td>
<td>தமிழ் வெறுமை (U)</td>
</tr>
<tr>
<td>0B8B</td>
<td>தமிழ் வெறுமை (UU)</td>
</tr>
<tr>
<td>0B8C</td>
<td>தமிழ் வெறுமை (UU)</td>
</tr>
<tr>
<td>0B8D</td>
<td>தமிழ் வெறுமை (UU)</td>
</tr>
<tr>
<td>0B8E</td>
<td>தமிழ் வெறுமை (E)</td>
</tr>
<tr>
<td>0B8F</td>
<td>தமிழ் வெறுமை (EE)</td>
</tr>
</tbody>
</table>
Binary Code Transmission

- Pulse Train
- Parity bit
  - Even parity
  - Odd parity

ASCII – ‘A’
Parity: Odd & Even Parity

- Computers can sometimes make errors when they transmit data.

  - **Even/odd parity:**
    - is basic method for detecting if an odd number of bits has been switched by accident.

  - **Odd parity:**
    - The number of 1-bit must add up to an odd number

  - **Even parity:**
    - The number of 1-bit must add up to an even number
The computer knows which parity it is using

If it uses an even parity:
  - If the number of 1-bit add up to an odd number then it knows there was an error:

If it uses an odd:
  - If the number of 1-bit add up to an even number then it knows there was an error:

However, If an even number of 1-bit is flipped the parity will still be the same. But an error occurs
  - The even/parity can’t detect this error:
Parity: Odd & Even Parity (Contd.)

- It is useful when an odd number of 1-bits is flipped.

- Suppose we have an 7-bit binary word (7-digits).
  - If you need to change the parity you need to add 1 (parity bit) to the binary word.
  - You now have 8 digit word.
  - However, the computer knows that the added bit is a parity bit and therefore ignore it.
EXAMPLE (1)

- Suppose you receive a binary bit word “0101” and you know you are using an odd parity.
- Is the binary word errored?
- The answer is yes:
  - There are 2 1-bit, which is an even number
  - We are using an odd parity
  - So there must have an error.
Parity Bit

- A single bit is appended to each data chunk
  - makes the number of 1 bits even/odd
- Example: even parity
  - 1000000 (1)
  - 1111101 (0)
  - 1001001 (1)
- Example: odd parity
  - 1000000 (0)
  - 1111101 (1)
  - 1001001 (0)
Assume we are using **even parity** with 7-bit ASCII.

The letter V in 7-bit ASCII is encoded as 0110101.

How will the letter V be transmitted?
- Because there are four 1s (an even number), parity is set to zero.
- This would be transmitted as: 01101010.

If we are using an **odd parity**:
- The letter V will be transmitted as 01101011.
**Exercise 1**

- Suppose you are using an odd parity. What should the binary word “1010” look like after you add the parity bit?

- **Answer:**
  - There is an even number of 1-bits.
  - So we need to add another 1-bit.
  - Our new word will look like “10101”.
EXERCISE 2

Suppose you are using an even parity. What should the binary word “1010” look like after you add a parity bit?

Answer:
- There is an even number of 1’s.
- So we need to add another 0
- Our new word will look like “10100”.
THANK YOU

Next Week

Lecture 02: Number Systems