Computer Architecture

Angshuman Paul

Assistant Professor

Department of Computer Science & Engineering

Outline of the Course

- ***Introduction:***
  - Basic ideas about the subsystems
  - Information representation
- ***Central Processing Unit:***
  - Instruction sets
  - CPU design and its various aspects
  - Performance issues
- ***Memory Hierarchy:***
  - Memory organization,
  - Working principles of various levels of memory architecture
  - Performance issues
- ***Interfacing:***
  - I/O transfer techniques
  - Computer buses, and peripherals
  - Current trends in architecture.

Syllabus

- ***Introduction:*** Basic computer organization, Components of computer systems, information representation.
- ***Central Processing Unit:*** Arithmetic and Logic Unit; Instruction sets; RISC, CISC, and ASIC/ASIP paradigms; Various addressing modes; Assembly language programming; Instruction interpretation: micro-operations and their RTL specification; CPU design, Hardwired and microprogrammed, Performance issues: Parallel processing, Pipelining, Hazards, Advanced parallelization techniques. Cache Coherence protocols, Multicore Architecture.
- ***Memory Hierarchy:*** Memory organization, Various levels of memory architecture and their working principles, Cache memory, Writing strategy, Coherence, Performance issues and enhancement techniques for memory design.
- ***Interfacing:*** I/O transfer techniques: Program controlled, Interrupt controlled and DMA; Introduction to computer buses, Peripherals and current trends in architecture.

Course Logistics

- Instructor: Angshuman Paul
- Contact: apaul@iitj.ac.in
- Class hours: 11 AM-11.50 AM
  - Tuesday
  - Thursday
  - Friday

Course Logistics

- Assignments: At least 2
- Quiz: At least 6 (+ surprise quizzes)
- Viva
- All are compulsory

Books and Study Materials

- Book: D.A. Patterson, J.L. Hennessy (2008), Computer Organization and Design, Morgan Kaufmann, 4th Edition.

- Other useful books:
  - Computer Architecture: A Quantitative Approach, by D.A. Patterson, J.L. Hennessy
  - Essentials of Computer Architecture, by Douglas Comer
  - The Essentials of Computer Organization and Architecture, by Linda Null and Julia Lobur

- NPTEL: https://nptel.ac.in/courses/106/105/106105163/

- Other books & Online materials: Will be informed from time to time

Prerequisites

- None*

*Knowledge about the following will be helpful

- Digital design
- Basic knowledge of programming

In Today's Lecture

- Computer and computer architecture
- Motivation of studying computer architecture
- History of computer
- Basic organization of computer

What is a Computer?

What is a Computer?

- Computer is programmable system that can process information and output meaningful results
  - Laptop, desktop, smartphone, PlayStation, autopilot, etc.

- A digital thermometer for room temperature measurement
  - Can record the temperature
  - Do some processing to display the reading
  - Is it a computer?

What is Computer Architecture?

- Computer architecture: the answer to the following question
  - What does a computer do?
- Computer organization: the answer to the following question
  - How does a computer do what is does?

What is Computer Architecture?
An Analogy

Architect

Building Architecture

1. Initial draft of design
2. Planning
3. Specifying visual appearance
4. Designs that improve
   - Ventilation
   - lighting etc.

Civil Engineer

Structural Design

Implementation of the design specifications

What is Computer Architecture?
An Analogy

Architect

Building Architecture

Computer Architecture

Civil Engineer

Structural Design

Computer Organization

What is Computer Architecture?

Computer Architecture

Deals with the functional aspects of a computer

Computer Organization

Deals with the low level design issues such as devices and circuits

What is Computer Architecture?

Computer Architecture

Deals with the functional aspects of a computer

The view of a computer as presented to software designers

Computer Organization

Deals with the low level design issues such as devices and circuits

The actual implementation of a computer in hardware

Why Should We Study Computer Architecture?

- Write better programs
  - Faster
  - Smaller
  - Less power consuming (less computations involved)

- To make suitable design choices for changing needs and evolving technologies
  - GPU
  - Wearable
  - Datacenter
  - Mobile phones
  - Quantum computing etc.

The Expected Outcomes

- Understanding of the basic components and the design of a computer
- Understanding of the functional aspects of different components
- Identification of the issues involved in the instruction execution
- Identification and analysis the issues related to performance improvement
- Analysis of system requirements

History of Computer

- Charles Babbage: Conceptualization and implementation of first mechanical computer (1833)
  - 8000 parts, weight: 5 ton, Length: 11 feet

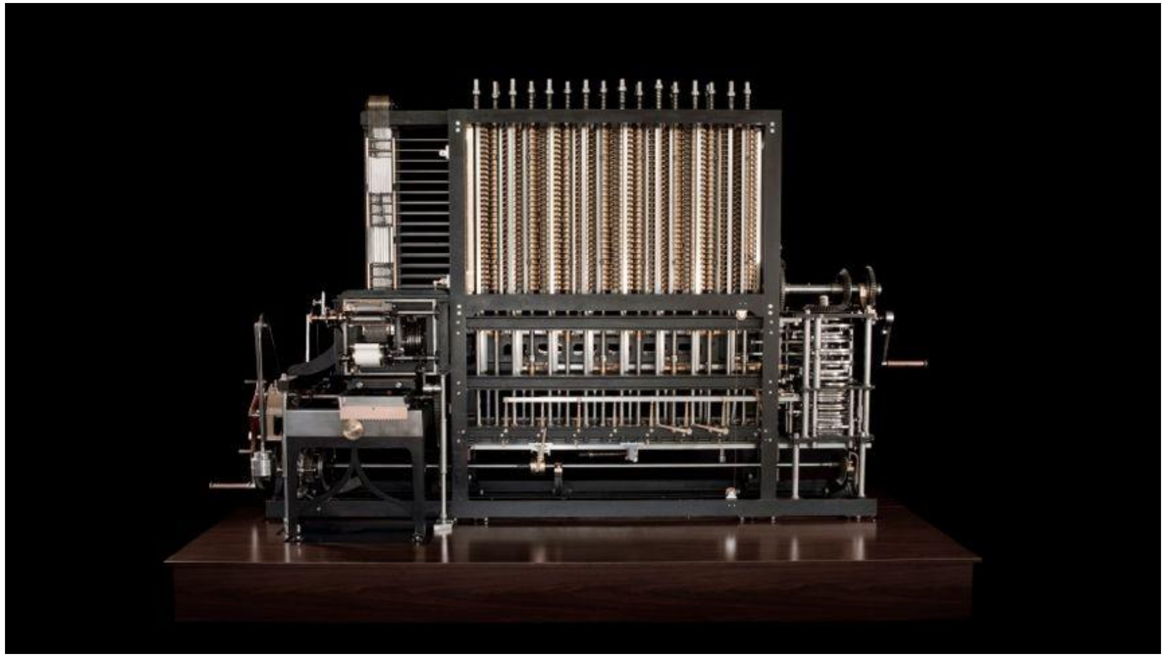
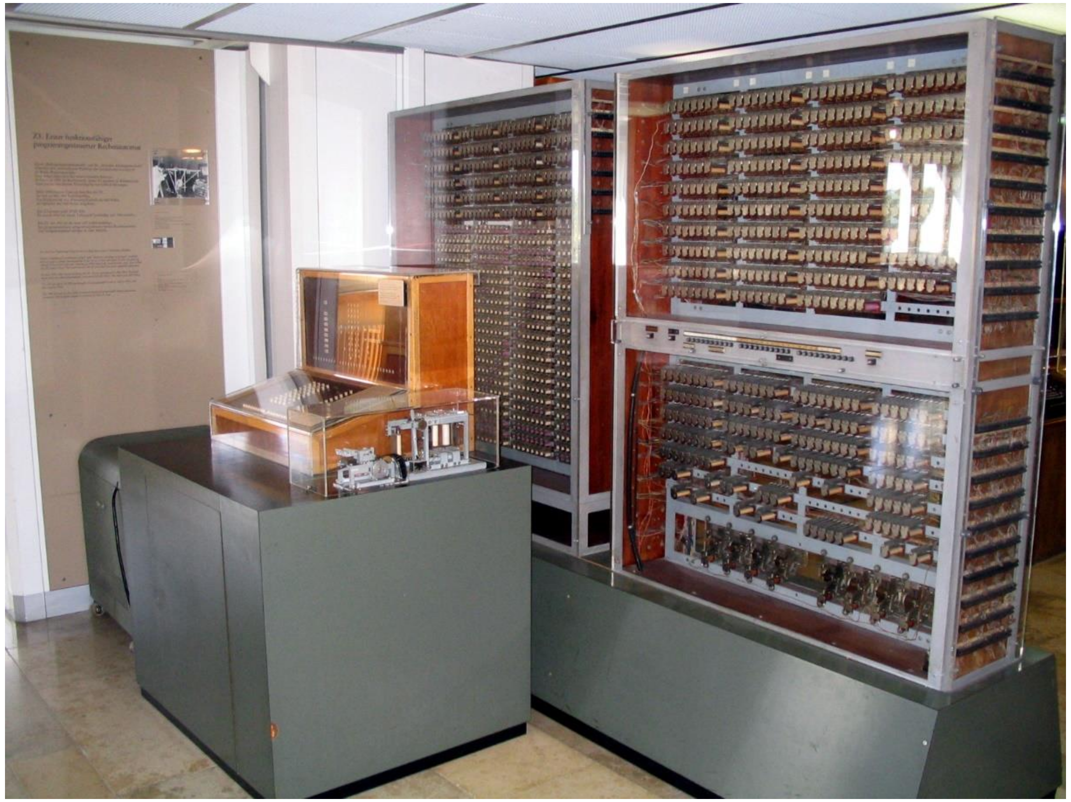


Image source: www.computerhistory.org

History of Computer

- Analog computer: early 20th century (used mechanical and electrical components)
- Digital computer (1938): electric switches and mechanical relays



History of Computer

- First electronic digital programmable computer *Colossus* (1943)
- Used vacuum tubes (2400 in Mark II)

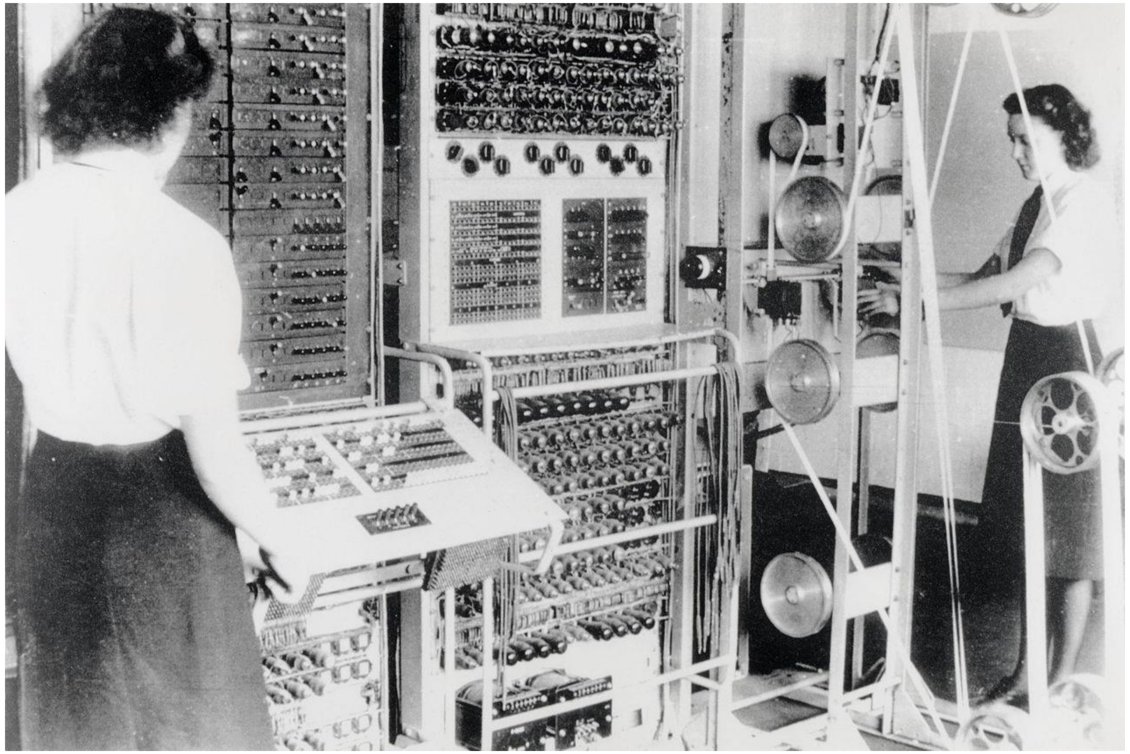


Image Source: https://en.wikipedia.org

History of Computer

- ENIAC: First programmable electronic computer that is Turing-complete (1945)
- 30 ton, 18000 vacuum tubes

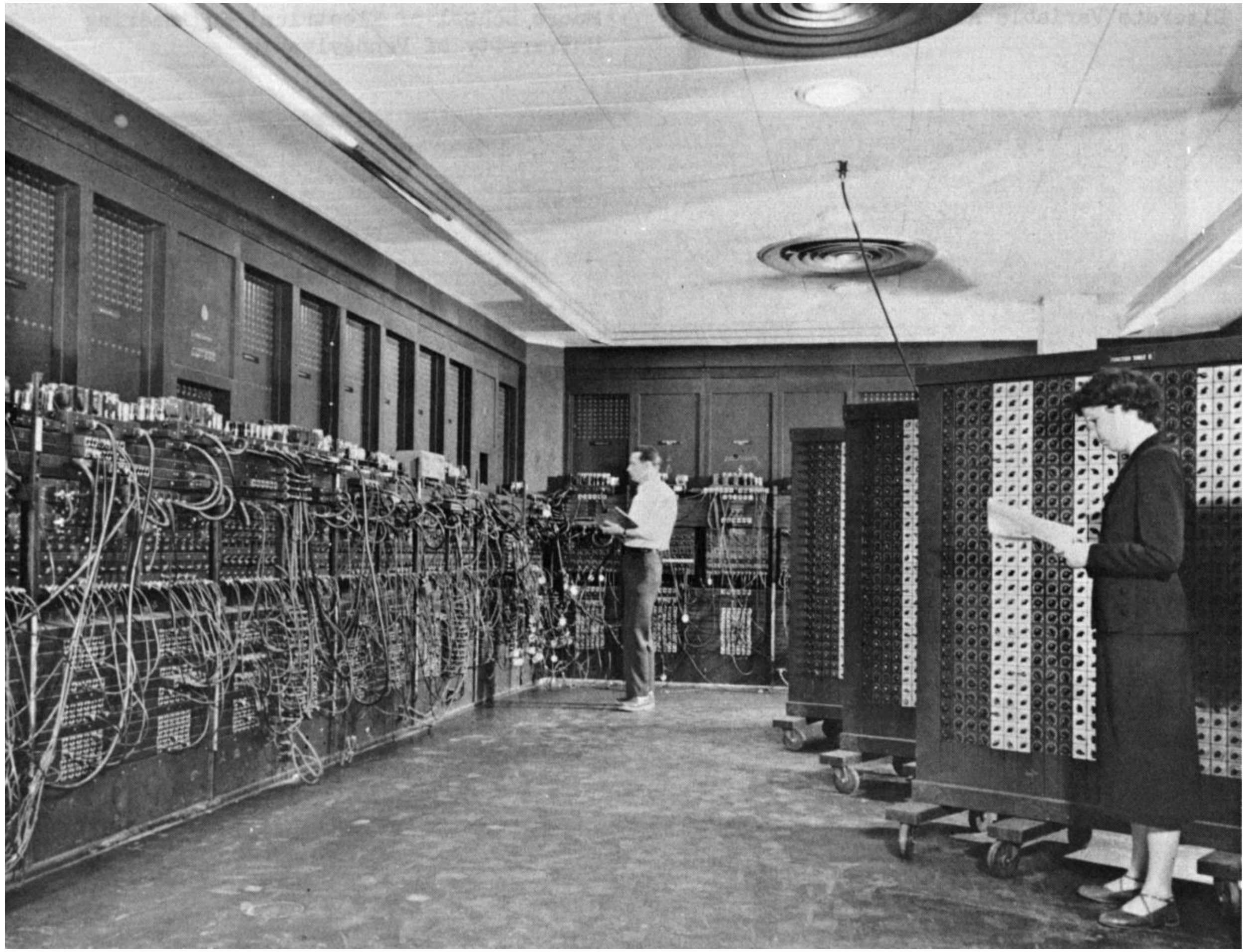


Image Source: https://en.wikipedia.org

History of Computer
Modern Computers

- Idea introduced by Alan Turing in 1936
  - Universal Turing Machine
- Stored program concept
  - Program to be executed is stored in memory
  - Instruction set
  - No re-wiring of the hardware required

Stored-program Computer

- Manchester baby: First stored program computer (1948)

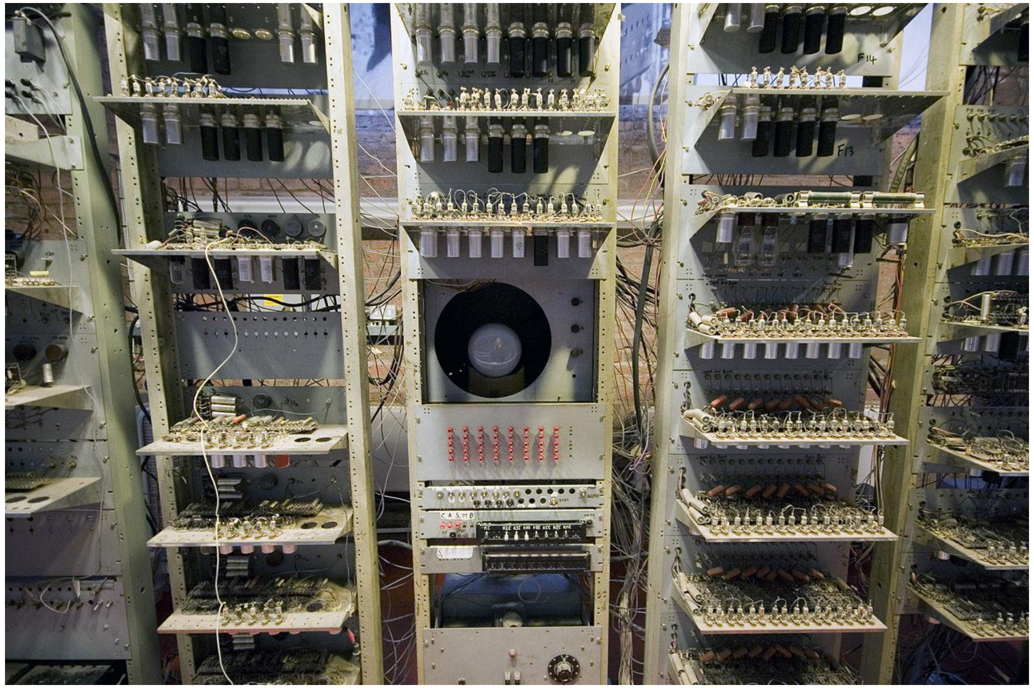


Image Source: https://en.wikipedia.org

Subsequent Developments

- Fully transistorized computer (1955): Harwell CADET
- Single-chip microprocessor (1971): Intel 4004 (4 bit CPU)
- First general purpose microcomputer (mid 70s)
- Home computers (early 80s)

Image Source: https://en.wikipedia.org

Generation of Computing Hardware

Five Generations of Computers

Generations of computers	Generations timeline	Evolving hardware
First generation	1940s-1950s	Vacuum tube based
Second generation	1950s-1960s	Transistor based
Third generation	1960s-1970s	Integrated circuit based
Fourth generation	1970s-present	Microprocessor based
Fifth generation	The present and the future	Artificial intelligence based

First Generation Computers (1940s-50s)

- Main electronic component – vacuum tube
- Main memory – magnetic drums and magnetic tapes
- Machine language programming
- Slow and large in size
- Input/output devices – punched cards and paper tape.
- Examples – ENIAC, UNIVAC1, IBM 650, IBM 701, etc.

Image Source: https://en.wikipedia.org

Second Generation Computers (1950s-60s)

- Main electronic component – transistor
- Memory – magnetic core and magnetic tape / disk
- Use of assembly language
- Improved speed
- Input/output devices – punched cards and magnetic tape.
- Examples – IBM 1401, IBM 7090 and 7094, UNIVAC 1107, etc.

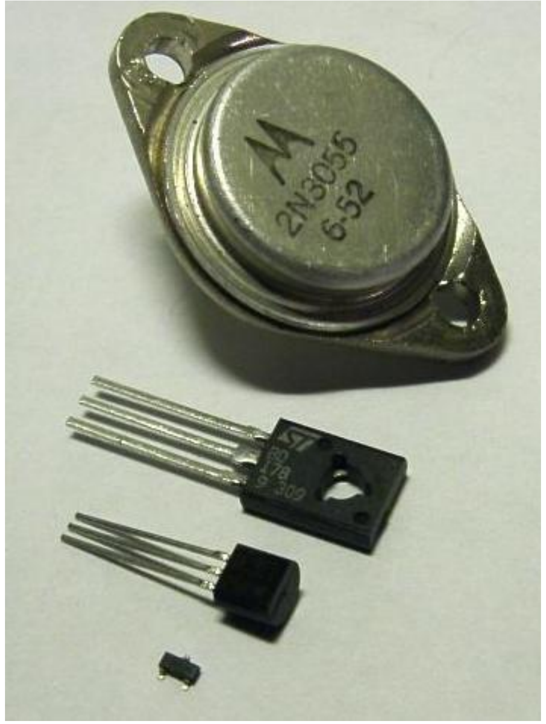


Image Source: https://en.wikipedia.org

Third Generation Computers (1960s-70s)

- Main electronic component – integrated circuits (ICs)
- Memory – large magnetic core, magnetic tape / disk
- Programming language – high level language (FORTRAN, BASIC, Pascal, COBOL, C, etc.)
- Smaller size, improved speed
- Input / output devices – magnetic tape, keyboard, monitor, printer, etc.
- Example – IBM 360, IBM 370, PDP-11, UNIVAC 1108, etc.

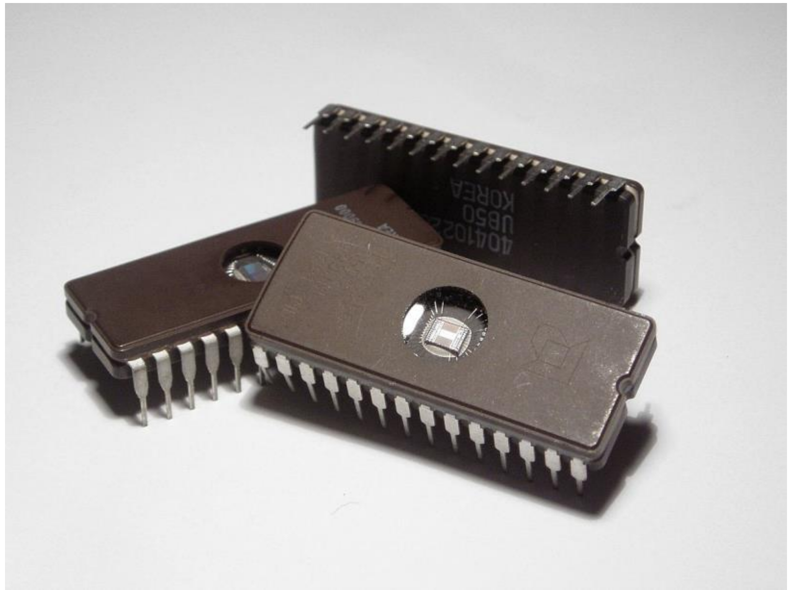


Image Source: https://en.wikipedia.org

Fourth Generation Computers (1970s - Present)

- Use of VLSI (millions of transistors in a chip) and microprocessor.
- Memory – semiconductor memory (such as RAM, ROM, etc.)
- High-level programming language (Java, Python, etc.)
- Smaller size and improved speed
- Network – a group of two or more computer systems linked together.
- Examples – IBM PC, STAR 1000, APPLE II, Apple Macintosh, etc.

Fifth Generation Computers (Present & Future)

- Parallel processing
- Natural language understanding
- Very high speed and very low power consumption
- Input / output device – keyboard, monitor, mouse, trackpad (or touchpad), touchscreen, pen, speech input (recognise voice / speech), light scanner, printer, etc.
- Example – desktops, laptops, tablets, smartphones, etc.

Theoretical / Experimental Computers

- **Quantum computer**
- Chemical computer
- DNA computing
- Optical computer
- Wetware/Organic computer

History of Computer in India

- TIFRAC
  - First digital computer built in India
  - In 1956 by the Tata Institute of Fundamental Research
- ISIJU-1
  - First solid-state digital computer
  - In 1965 by the Indian Statistical Institute and Jadavpur University

Image source: https://artsandculture.google.com/asset/the-indian-statistical-institute-%E2%80%93-jadavpur-university-isi-ju-computer-was-commissioned-by-shri-m-c-chagla-then-union-minister-of-education-at-jadavpur-on-2nd-april-1966/RAHKafu98GJMxQ

Technology Trends: Moore's Law

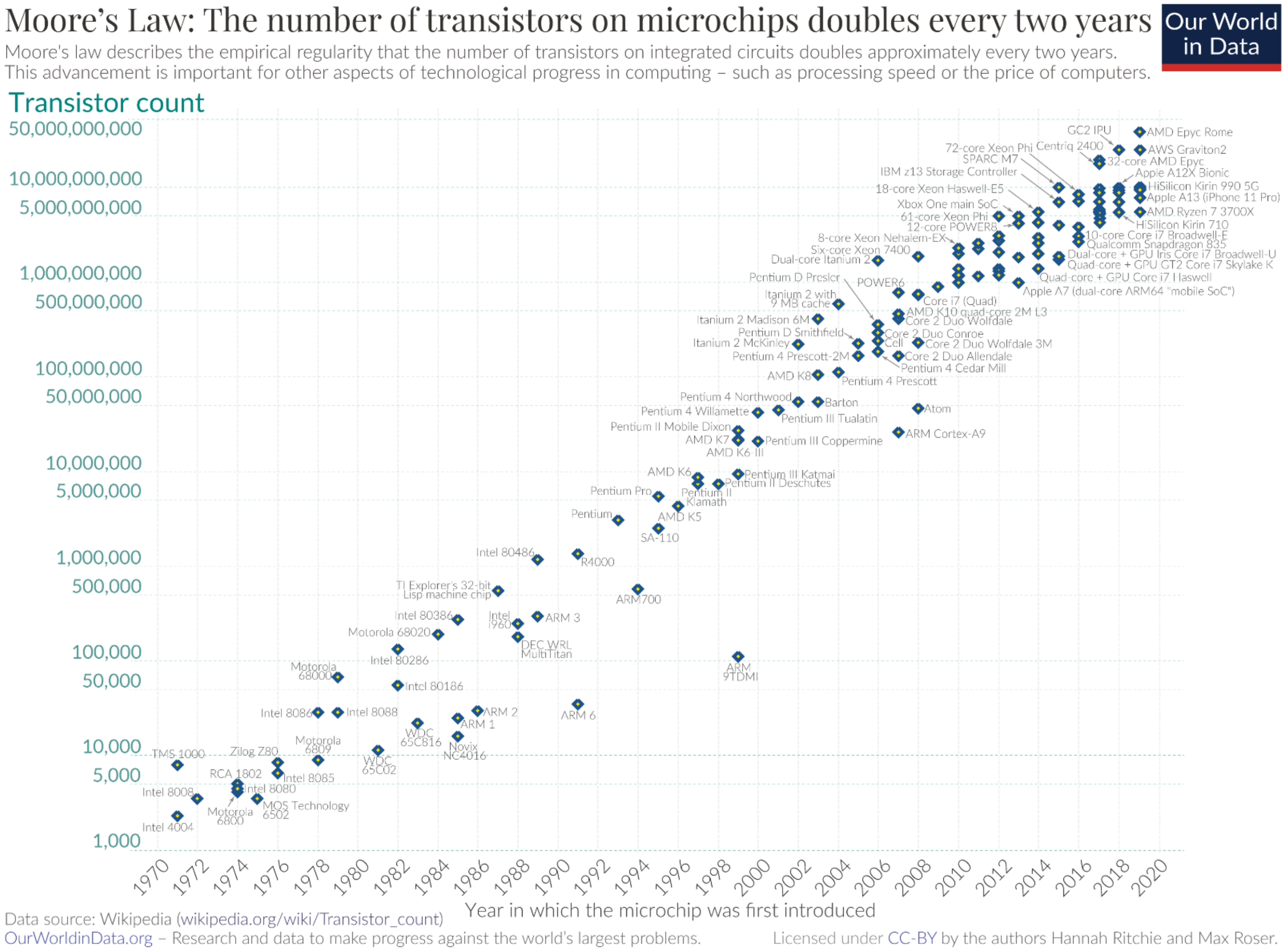


Image Source: https://en.wikipedia.org

Moore's Law
Is it Ending?

- Difficult to build smaller transistor in 2 year timeframe that are affordable
- Possible Solutions?
  - Quantum computing
  - Neuromorphic Computing

Abstraction

Abstraction: An Example

- Ordering pizza online
  - I specify crust but don’t care about the origin of the flour
  - I specify toppings, but don’t care about the details of fertilizers for veg toppings etc.
- Omits unexpected detail, reduce the complexity of the process of food ordering

Abstraction in Computer

- Programming lab
  - You wrote codes in C, Python etc. for problem solving

- Do you know the answer to the following questions.
  - What material has been used for designing the clock of your computer?
  - How many registers are there in your CPU?
  - How the multiplier circuit has been implemented?
  - What scheduling protocol does your OS use?

Abstraction

00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000

Abstraction

Binary machine language program (for MIPS)

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

Abstraction

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

What we write

Binary machine language program (for MIPS)

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

What computer understands

Abstraction

- Lower level details are hidden to offer a simpler model at higher level
- Hardware and software abstractions

Levels of Abstraction

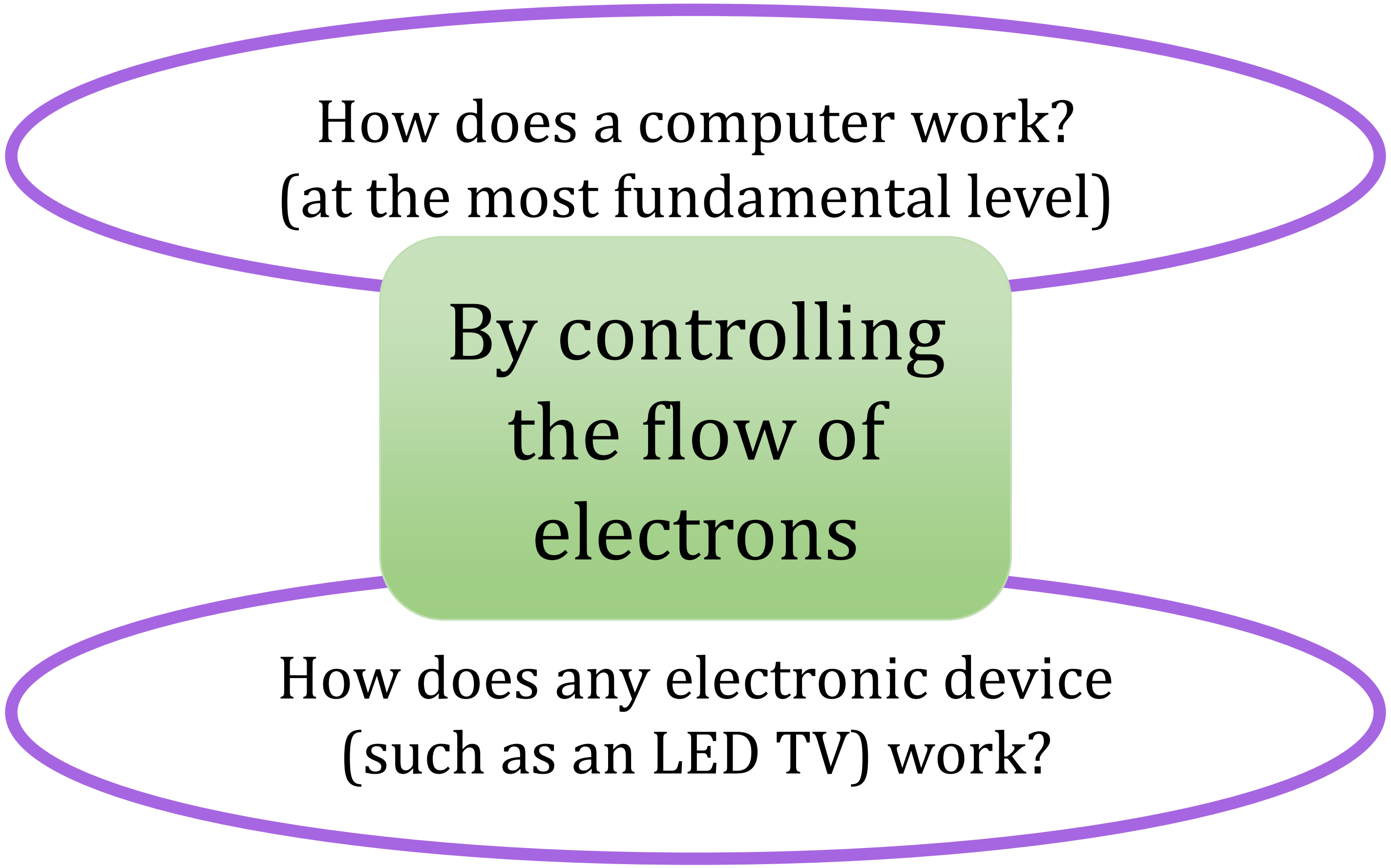
How does a computer work?
(at the most fundamental level)

Levels of Abstraction

How does a computer work?
(at the most fundamental level)

How does any electronic device
(such as an LED TV) work?

Levels of Abstraction



Levels of Abstraction

Problem

Algorithm

Program

Electrons

Levels of Abstraction

Problem → 

→ Algorithm ↓ Program ⋮ ↓ Electrons

Can we control the electrons being a programmer?

Levels of Abstraction

Problem

Algorithm

Program

System Software

Software Hardware Interface

Microarchitecture

Logic

Device

Electrons

Levels of Abstraction

Problem

Algorithm

Program

System Software

Software Hardware Interface

Microarchitecture

Logic

Device

Electrons

Our main focus

Levels of Abstraction

- Problem
- Algorithm
- Application Program
- System Software
- Instruction Set Architecture
- Microarchitecture
- Logic
- Device
- Electrons

Our main focus

Abstraction

- Protects us from change
  - Example: We need not change our C program
    - for Intel and AMD computers
    - for Windows and Linux
    - When the fabrication technology changes etc.
- Writing program becomes easier in high-level languages

Looking Inside a Computer


FSP GROUP INC.
MODEL NO: FSP400-60GLN
WARNING: HAZARDOUS AREA
BALL BEARING FAN
RoHS
Active PFC
W/NOISE KILLER
intel

The Motherboard

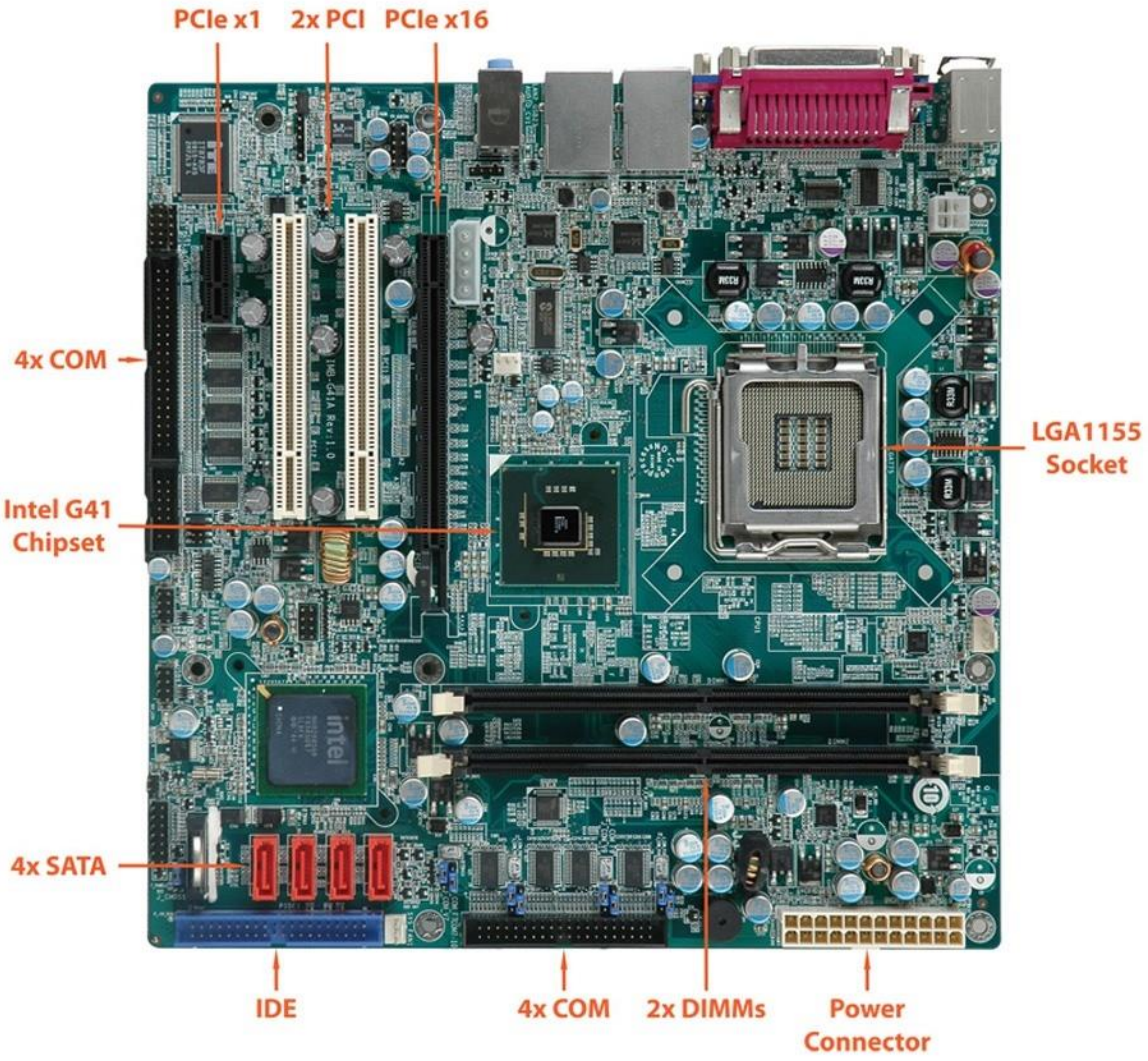
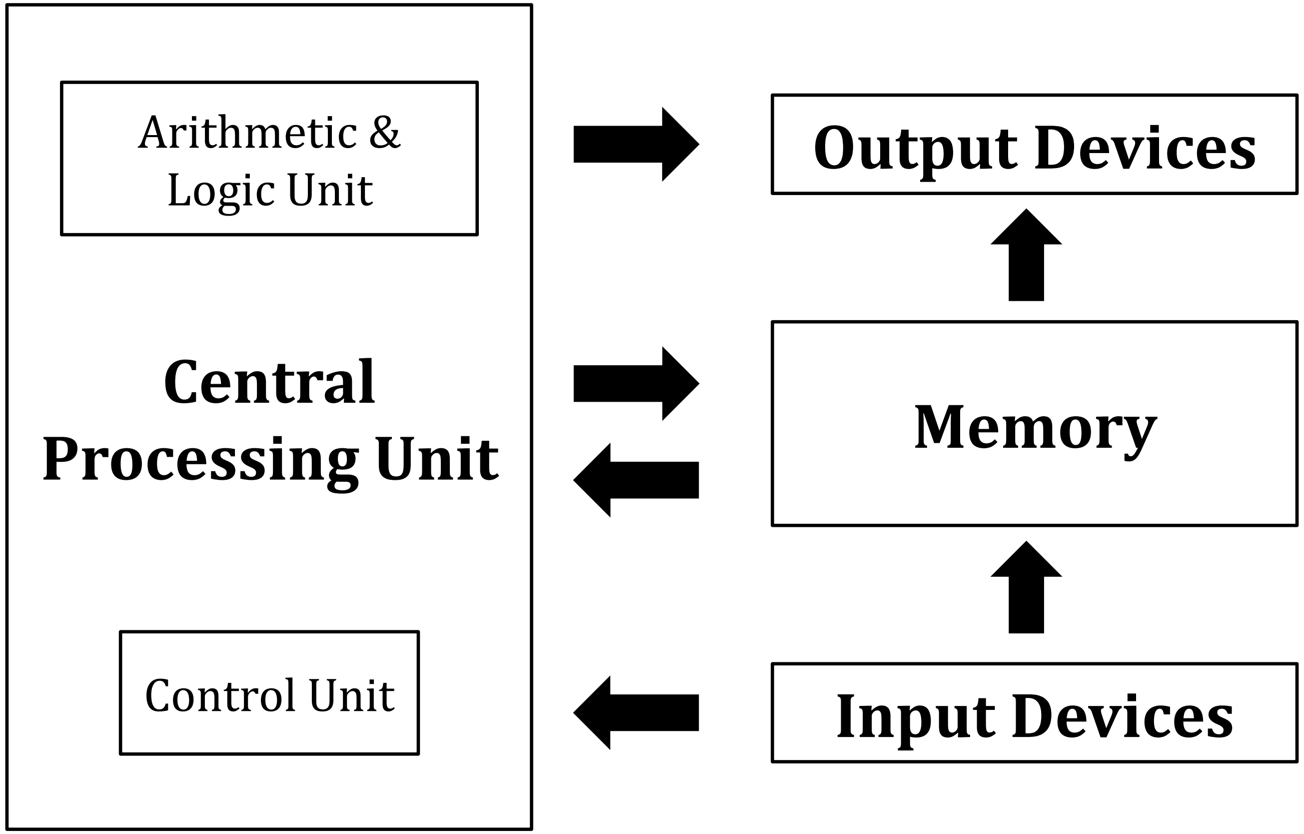
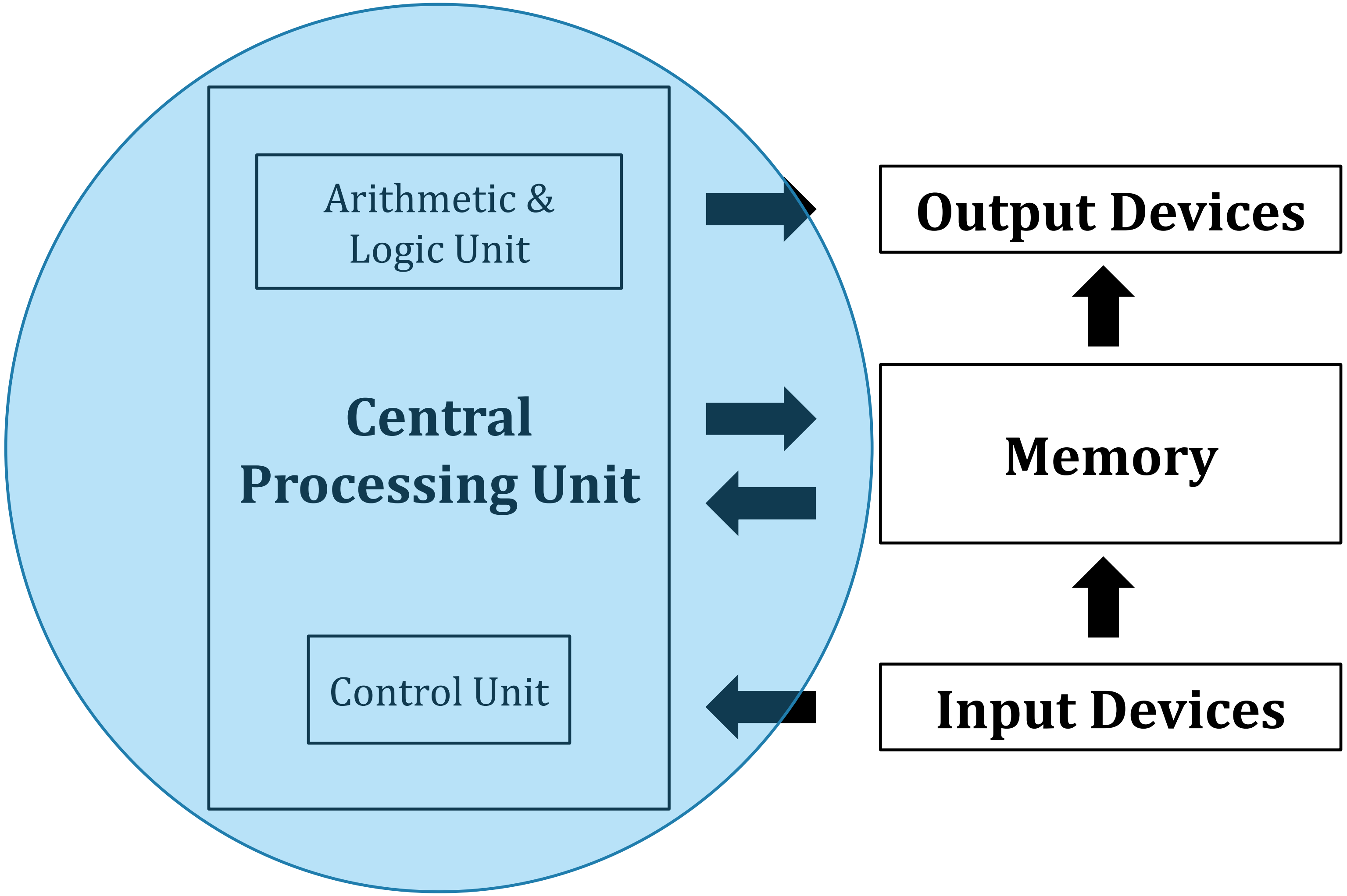


Image Source: https://www.industrialpcpro.com/imb-g41a-industrial-micro-atx-motherboard

Subsystems of a Digital Computer (A Simplified View)

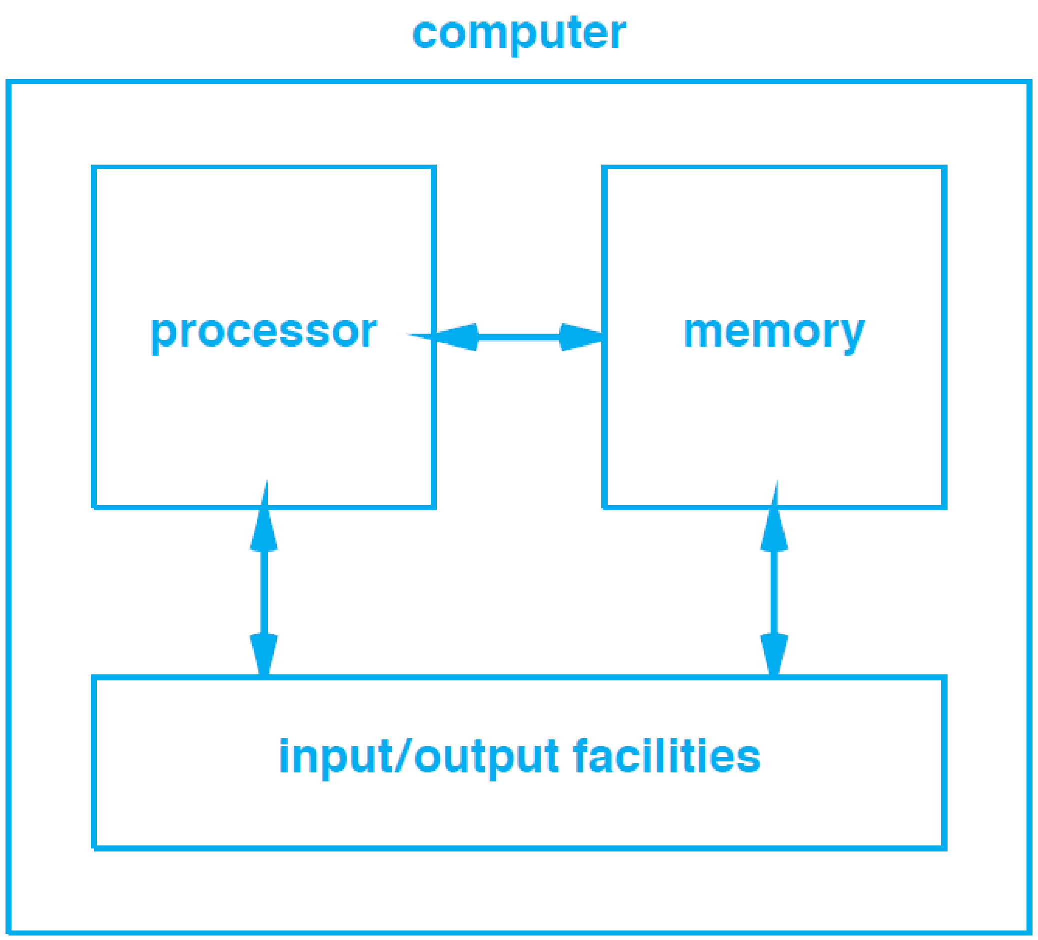


Subsystems of a Digital Computer (A Simplified View)



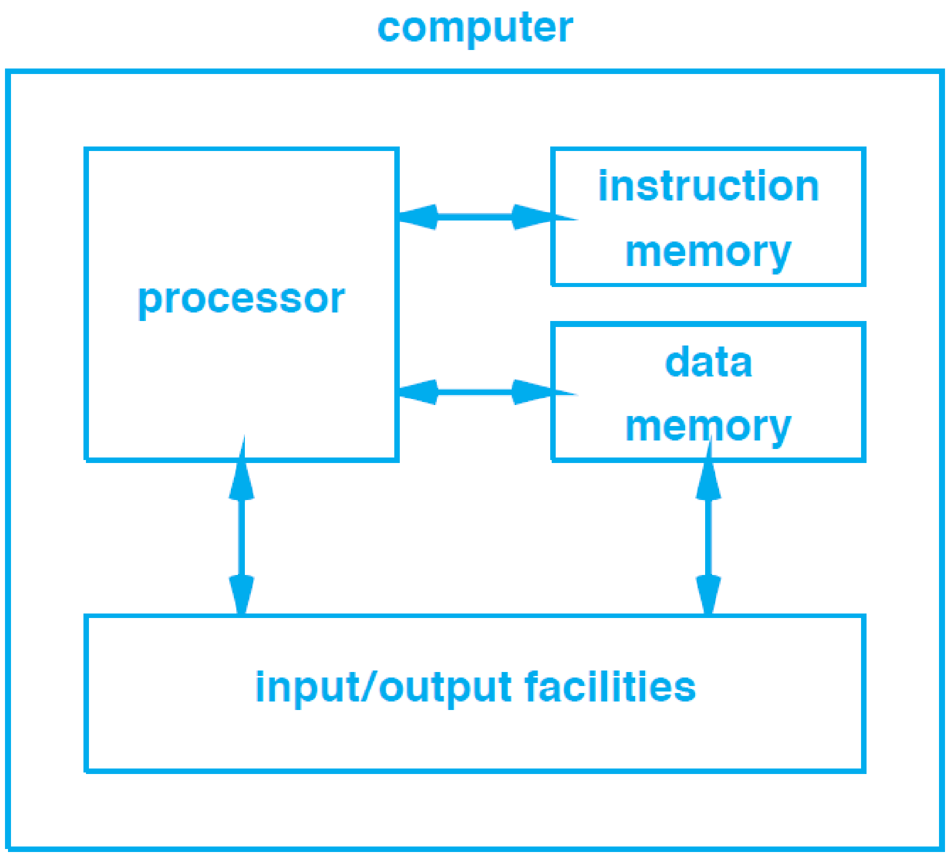
Von Neuman Architecture

Same memory for program and data



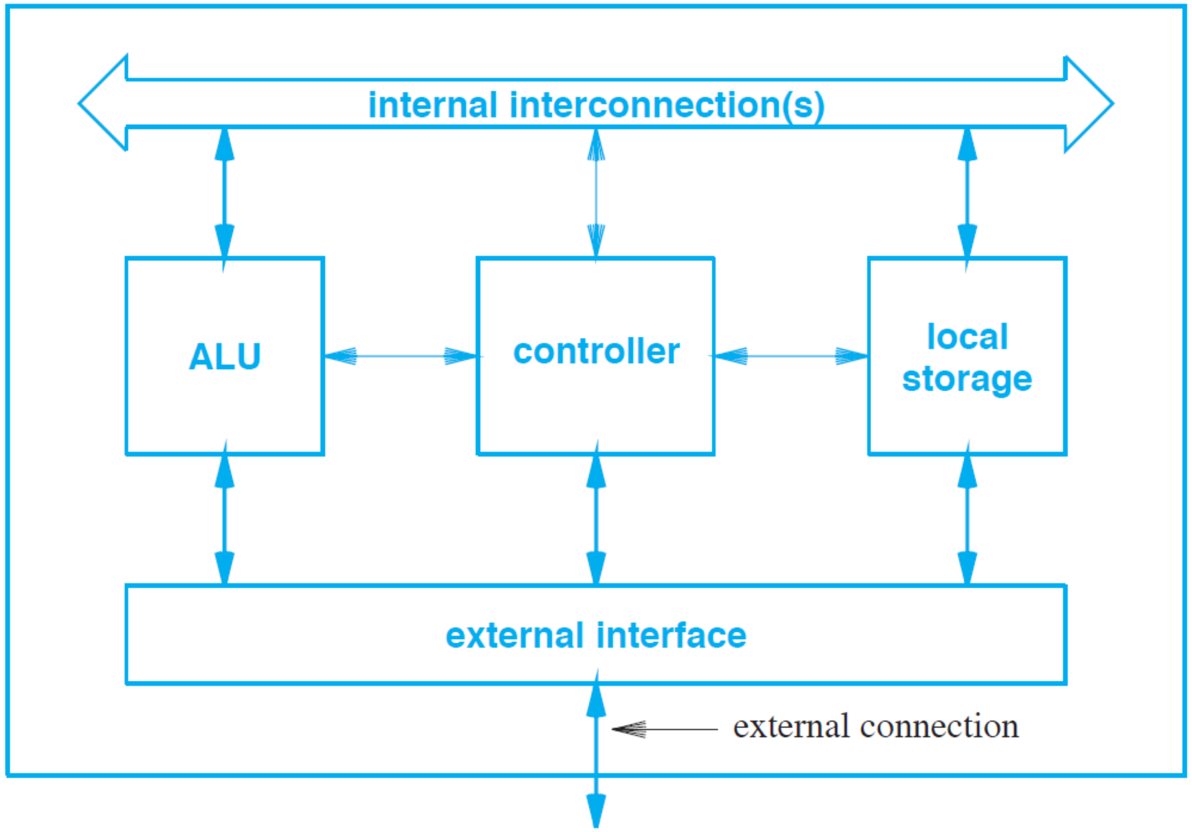
Harvard Architecture

Separate memory for program and data



The Processor

- The brain of the computer system
- Major component:
  - Arithmetic Logic Unit (ALU)
  - Control Unit (CU)
  - Registers (Local Storage)
    - General Purpose
    - Special Purpose



The Arithmetic Logic Unit

- Responsible for all the arithmetic and logic operations on data processed by a computer
- The circuitry must be able to perform all the arithmetic and logic operations included in the instruction set
- Arithmetic operations
  - Addition, subtraction, multiplication, division
- Logic operations
  - AND, OR, NOT, Shift, Compare etc.

The Arithmetic Logic Unit: Typical Steps of Operation

1. Bring operand from memory/ input device
2. Performs operation
3. Writes the result to a register/ memory/ output device

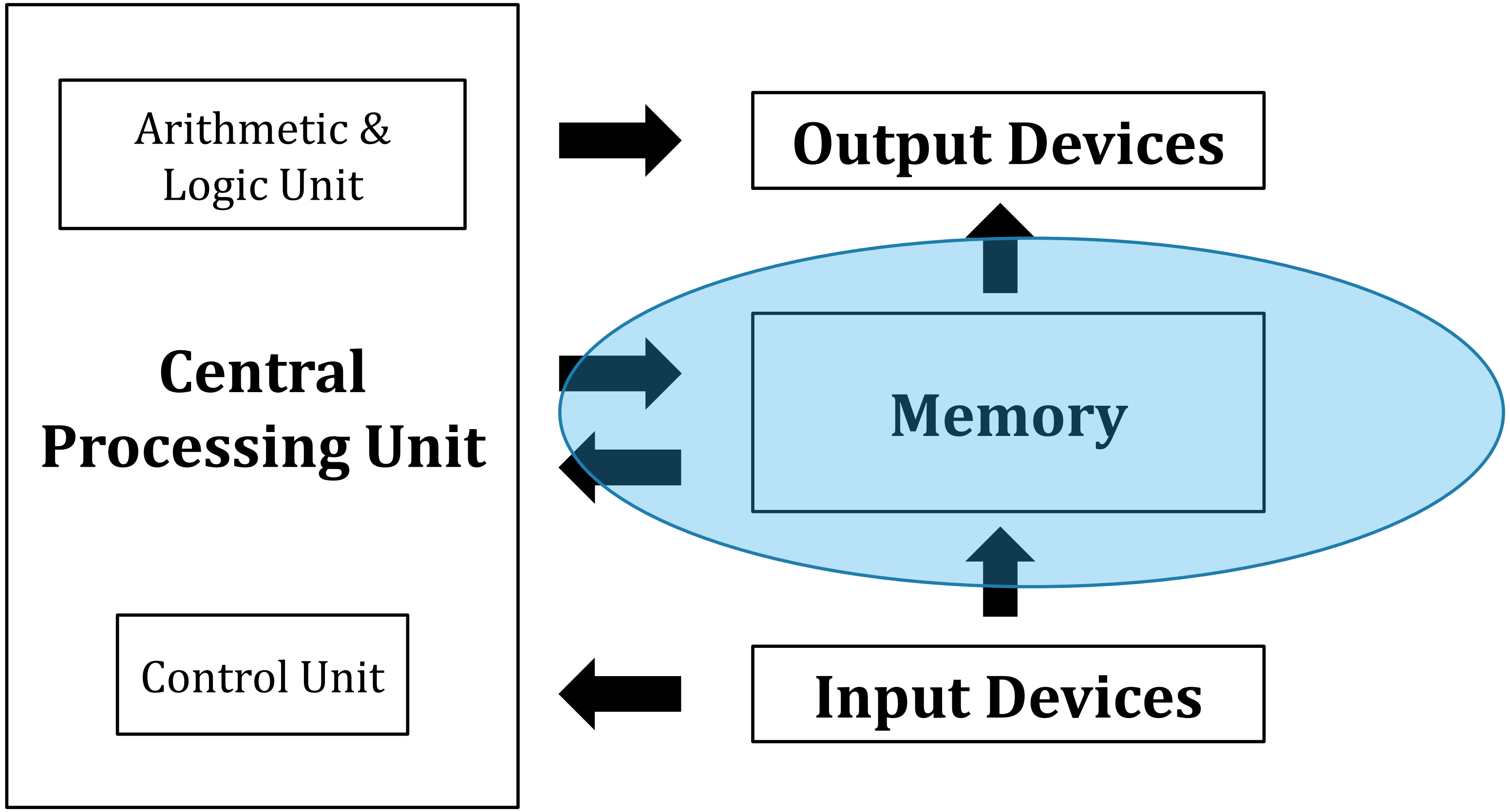
The Control Unit

1. Responsible for controlling the operations of CPU, memory and I/O devices
2. Senses the states of different devices
3. Decodes the next instruction to be executed
4. Depending on the next instruction to be executed, sends control signal

The Registers

1. Fast storage location inside the CPU
2. Temporarily holds data / address
3. Volatile (can’t hold data when turned off)
4. Example: MAR, IR, $R_{0,}$ $R_{1,}$ Accumulator, etc.

Subsystems of a Digital Computer (A Simplified View)



The Memory Unit

- Primary memory
  - Fast and smaller size
  - Volatile or Non-volatile
  - CPU has direct access
  - Example: RAM, ROM
- Secondary Memory
  - Slow and larger size
  - Non-volatile
  - Not directly accessible by CPU
  - Example: Hard disk

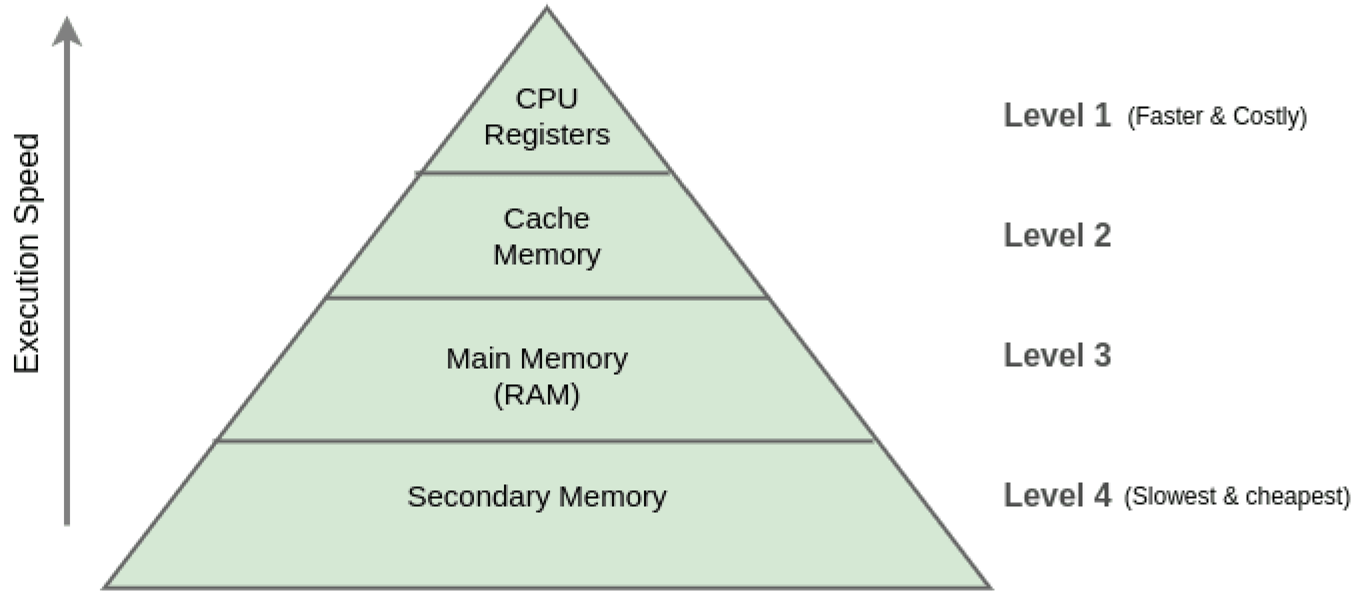
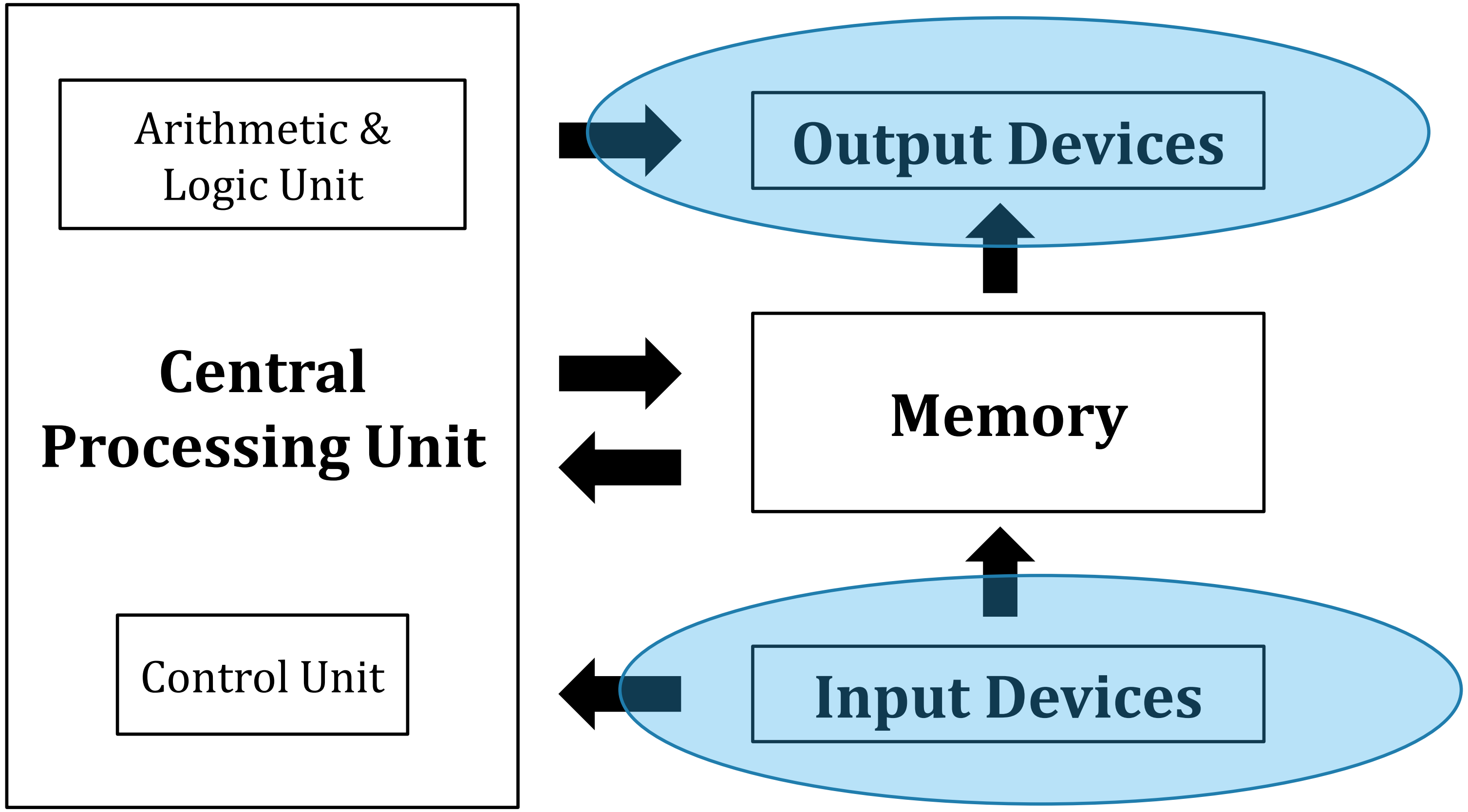


Image Credit: https://nlogn.in/what-is-cpu-cache-memory-in-computer-architecture/

Subsystems of a Digital Computer (A Simplified View)



The Input-Output Devices

- Input Devices

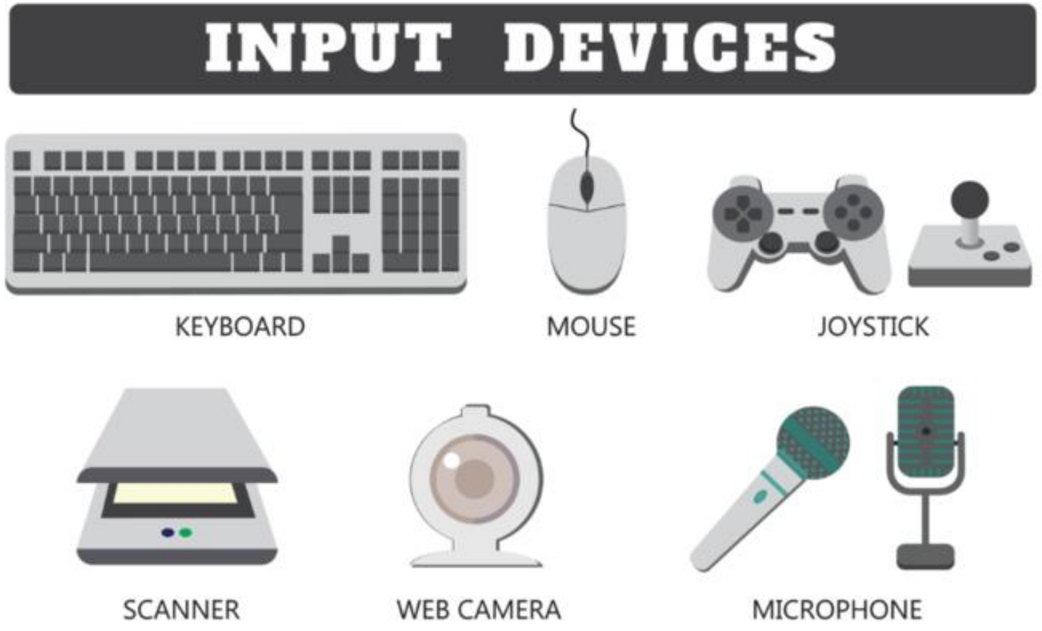


Image Credit: https:https://gradeup.co/notes-on-input-output-devices-i-a61f6521-eda5-11e5-9ba0-09a38122a82f

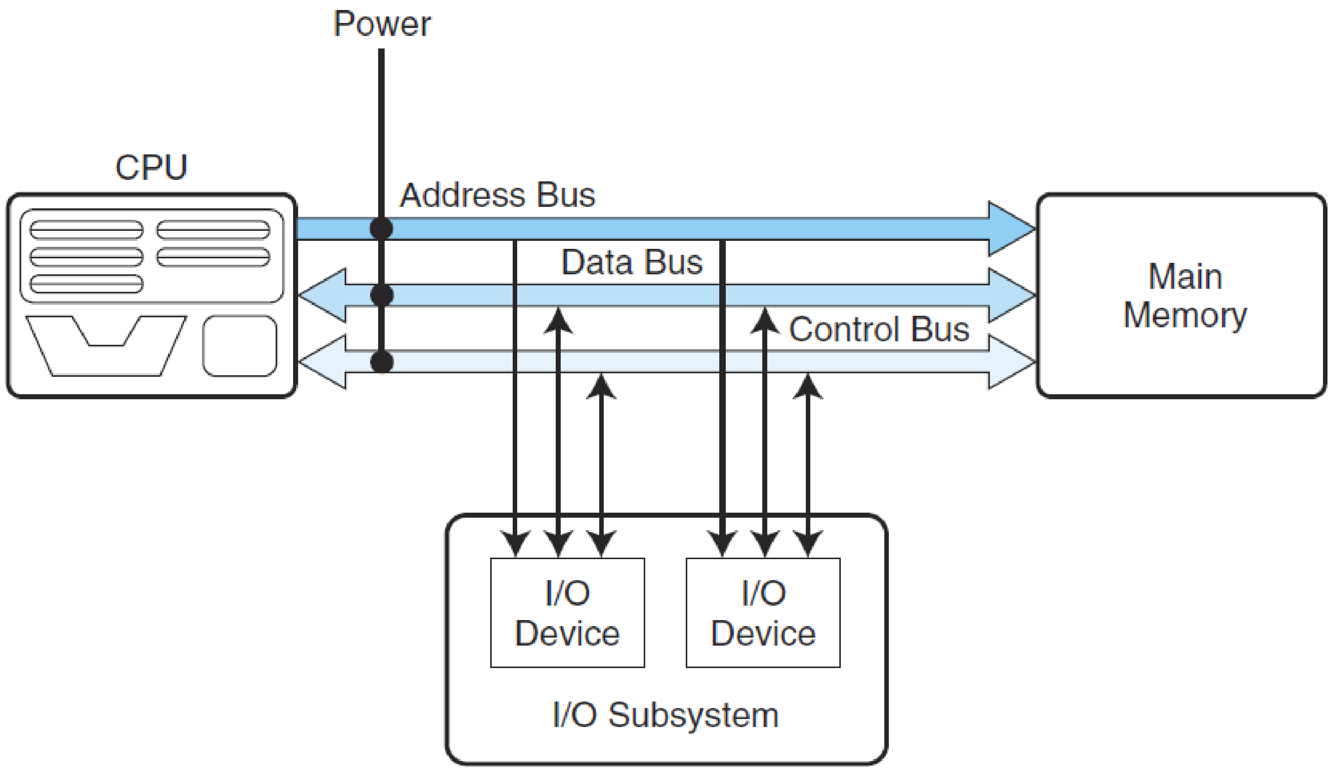
The Input-Output Devices

- Output Devices

Image Credit: https://computenepal.blogspot.com/2021/05/computer-system-input-output-devices.html

Computer Buses

- Communication pathway between different components of a computer



Computer Clock

- Used for synchronizing the different components inside a computer
- Dictates the occurrence of events in executing an instruction
  - An event in a sequence of events can occur only when the clock triggers the logic circuits
- A major factor in determining the speed of computer
- Designed using quartz crystal

Performance Metrics

Performance of a Computer

- How to define the performance of a computer?
- Speed
  - Execution time: Time for the completion of a task (from start to end)
    - Important in most situations including personal computers
    - Also called response time
  - Throughput/ bandwidth: Total number of work done in a given time
    - Important in places like datacenter

Factors Affecting the Performance of a Computer

- Processor
- I/O Controllers and peripherals
- Memory
- System software
- Instruction set

Performance of a Computer

- How to define the performance of a computer?
- Speed
  - Execution time: Time for the completion of a task
    - Important in most situations including personal computers
  - Throughput/ bandwidth: Total number of work done in a given time
    - Important in places like datacenter

Performance of a Processor

- Why do we need to focus on the performance of a processor
  - Wait for I/O

- Performance metric: CPU execution time
  - Time taken by the CPU for computing
  - Includes OS routines executed for the program
  - Does not include the wait time for I/O

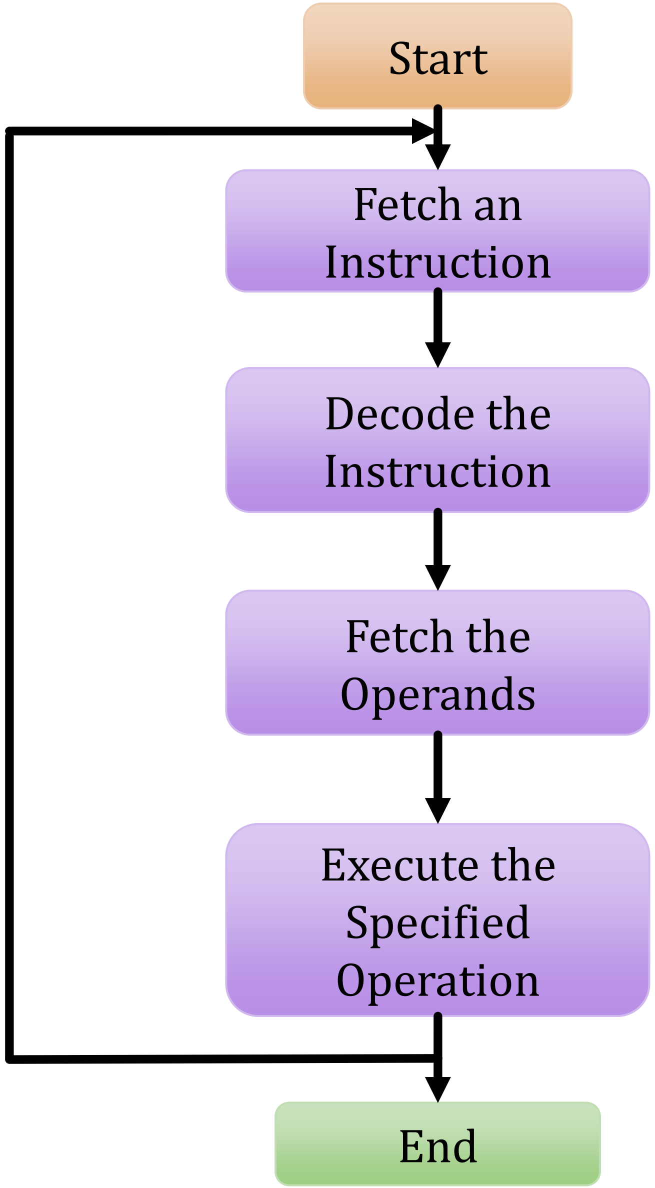
Performance in Terms of CPU Execution Time

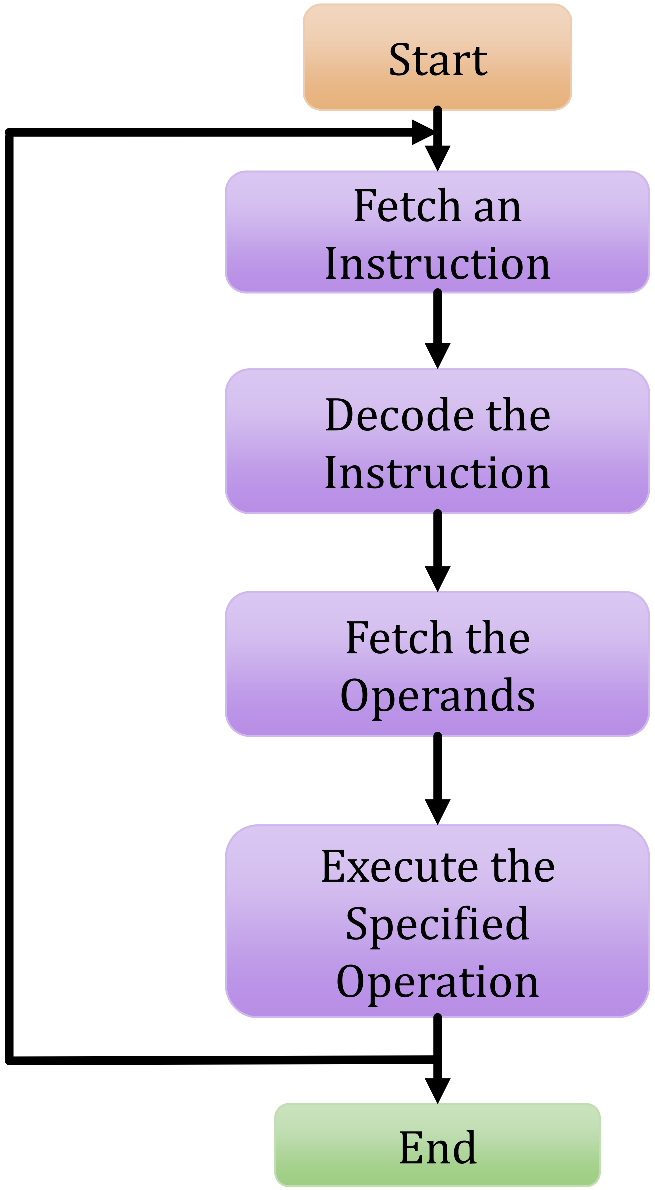
- Performance $= \dfrac{1}{CPU\ Execution\ Time}$

- For a given program in two computers (actually CPUs of) A and B

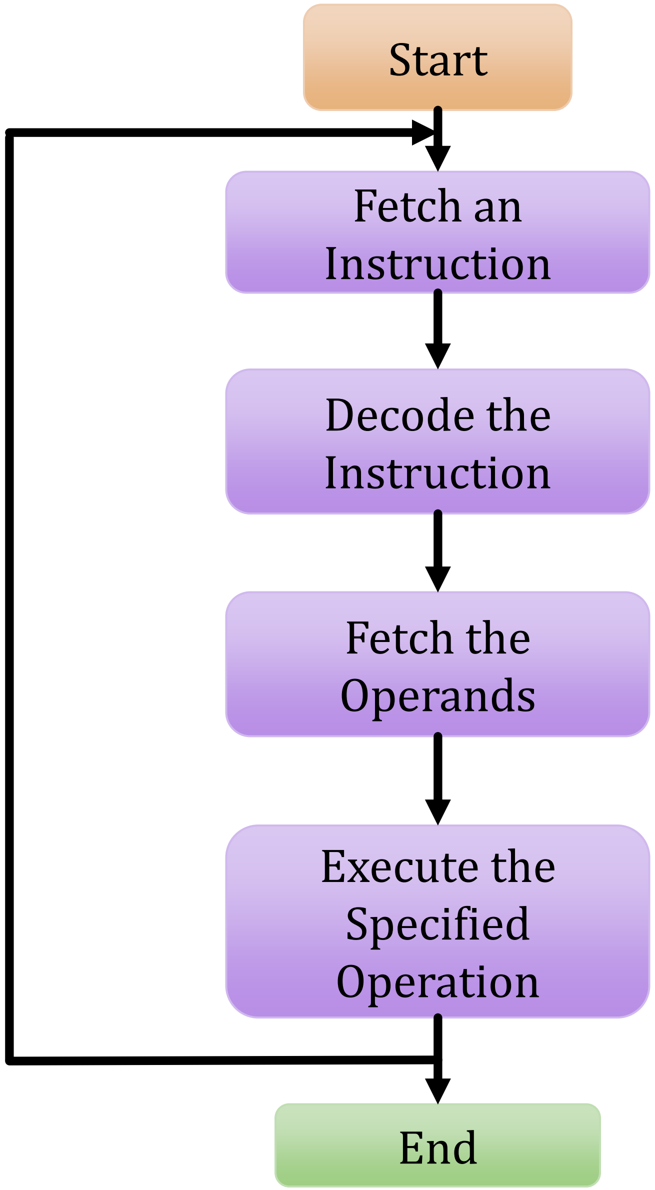
$$\frac{Performance_A}{Performance_B} = \frac{CPU\ Execution\ Time_B}{CPU\ Execution\ Time_A} = n = speedup$$

Steps Involved in Processing of an Instruction


Start
Fetch an Instruction
Decode the Instruction
Fetch the Operands
Execute the Specified Operation
End

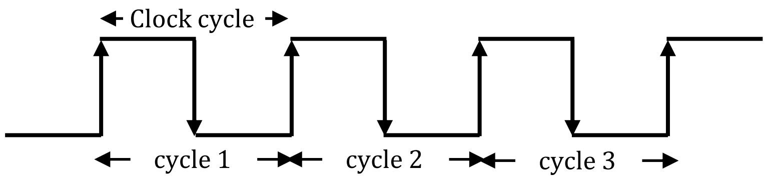


- Each step of instruction cycle may contain a sequence of micro-operations



- Each step of instruction cycle may contain a sequence of micro-operations

- Each micro-operation: triggered by the computer clock and executed in one clock cycle



CPU Execution Time and the Computer Clock

- A machine instruction
  - Several micro-operations
  - Several clock cycles

- Average Clock Cycles per Instruction (CPI)

- CPU Execution Time = Total Number of Clock Cycles × CPU Clock Cycle Time
  = Total Number of Clock Cycles × (1/Clock Rate)
  = (Number of Instructions × CPI) × (1 / Clock Rate)

CPU Performance Equation

- CPU Execution Time = Total Number of Clock Cycles × CPU Clock Cycle Time

  = Total Number of Clock Cycles × (1/Clock Rate)

  = (Number of Instructions * CPI) × (1 / Clock Rate)

  = Number of Instructions × CPI × CPU Clock Cycle Time

  = $I \times CPI \times C$

  = $I \times CPI \times (1/f)$

Factors Affecting CPU Execution Time

- A machine instruction
  - Several micro-operations
  - Several clock cycles

- Average Clock Cycles per Instruction (CPI)

- CPU Execution Time = Total Number of Clock Cycles × CPU Clock Cycle Time

  = Total Number of Clock Cycles × (1/Clock Rate)

  = (Number of Instructions × CPI) ×(1 / Clock Rate)

Factors Affecting CPU Execution Time

- A machine instruction
  - Several micro-operations
  - Several clock cycles

- Average Clock Cycles Per Instruction (CPI)

- CPU Execution Time = Total Number of Clock Cycles × CPU Clock Cycle Time

  = Total Number of Clock Cycles × (1/Clock Rate)

  = (Number of Instructions× CPI) × (1 / Clock Rate)

Factors Affecting CPU Execution Time

- A machine instruction
  - Several micro-operations
  - Several clock cycles

- Average Clock Cycles per Instruction (CPI)

- CPU Execution Time = Total Number of Clock Cycles × CPU Clock Cycle Time

  = Total Number of Clock Cycles × (1/Clock Rate)

  = (Number of Instructions × CPI) × (1 / Clock Rate)

Calculation of CPI

Instruction Type	Frequency of Occurrence	Clock Cycle
ALU Instructions	55%	5
Load Instructions	25%	6
Store Instructions	10%	4
Branch Instructions	10 %	2

Calculation of CPI

Instruction Type	Frequency of Occurrence	Clock Cycle
ALU Instructions	55%	5
Load Instructions	25%	6
Store Instructions	10%	4
Branch Instructions	10 %	2

$$CPI = (55/100) \times 5 + (25/100) \times 6 + (10/100) \times 4 + (10/100) \times 2$$

$$= 4.85$$

Calculation of CPI

Instruction Type	Frequency of Occurrence	Clock Cycle
ALU Instructions	55%	5 (CPI_{ALU})
Load Instructions	25%	6 (CPI_{Load})
Store Instructions	10%	4 (CPI_{Store})
Branch Instructions	10 %	2 (CPI_{Brach})

$$CPI = (55/100) \times 5 + (25/100) \times 6 + (10/100) \times 4 + (10/100) \times 2$$

$$= 4.85$$

Calculation of CPI

Instruction Type	Frequency of Occurrence	Clock Cycle
ALU Instructions	55% (F_{ALU})	5 (CPI_{ALU})
Load Instructions	25% (F_{Load})	6 (CPI_{Load})
Store Instructions	10% (F_{Store})	4 (CPI_{Store})
Branch Instructions	10 % (F_{Brach})	2 (CPI_{Brach})

$$CPI = (55/100) \times 5 + (25/100) \times 6 + (10/100) \times 4 + (10/100) \times 2$$

$$= 4.85$$

Calculation of CPI

Instruction Type	Frequency of Occurrence	Clock Cycle
ALU Instructions	55% (F_{ALU})	5 (CPI_{ALU})
Load Instructions	25% (F_{Load})	6 (CPI_{Load})
Store Instructions	10% (F_{Store})	4 (CPI_{Store})
Branch Instructions	10 % (F_{Brach})	2 (CPI_{Brach})

$$CPI = (55/100) \times 5 + (25/100) \times 6 + (10/100) \times 4 + (10/100) \times 2$$

$$= (F_{ALU} * CPI_{ALU}) + (F_{Load} * CPI_{Load}) + (F_{Store} * CPI_{Store}) + (F_{Branch} * CPI_{Branch})$$

$$= 4.85$$

Calculation of CPI

Instruction Type	Frequency of Occurrence	Clock Cycle
ALU Instructions	55% (F_{ALU})	5 (CPI_{ALU})
Load Instructions	25% (F_{Load})	6 (CPI_{Load})
Store Instructions	10% (F_{Store})	4 (CPI_{Store})
Branch Instructions	10 % (F_{Brach})	2 (CPI_{Brach})

CPI = (55/100) × 5+ (25/100) × 6+ (10/100) × 4+ (10/100) × 2

$= \sum_{i=1}^{n} F_i \times CPI_i$

= 4.85

Calculation of CPU Execution Time

- A program with 5,00,0000 instructions
- CPU clock frequency 2 GHz
- CPI = 2.4

What is the CPU execution time?

Calculation of CPU Execution Time

$$\textbf{CPU Execution Time} = (5 \times 10^6) \times (\mathbf{2.4}) \times \left(\frac{\mathbf{1}}{\mathbf{2 \times 10^9}}\right)$$

- A program with 5,00,0000 instructions; i.e. $I = 5 \times 10^6$
- CPU clock frequency 2 GHz; i.e $f = 2 \times 10^9$
- CPI = 2.4

Calculation of CPU Execution Time

CPU Execution Time = $(5 \times 10^6) \times (\mathbf{2.4}) \times \left(\frac{\mathbf{1}}{\mathbf{2 \times 10^9}}\right) = \mathbf{6 \times 10^{-3} s}$

- A program with 5,00,0000 instructions; i.e. $I = 5 \times 10^6$
- CPU clock frequency 2 GHz; i.e $f = 2 \times 10^9$
- CPI = 2.4

Calculation of CPU Execution Time

CPU Execution Time = $(5 \times 10^6) \times (\mathbf{2.4}) \times \left(\frac{\mathbf{1}}{\mathbf{2 \times 10^9}}\right) = \mathbf{6\ ms}$

- A program with 5,00,0000 instructions; i.e. $I = 5 \times 10^6$
- CPU clock frequency 2 GHz; i.e $f = 2 \times 10^9$
- CPI = 2.4

Instructions Per Second

- $IPS = \frac{Number\ of\ Instructions}{CPU\ Execution\ time}$

- $MIPS = \frac{Number\ of\ Instructions}{CPU\ Execution\ time \times 10^6}$ (***m***illion ***i***nstructions ***p***er ***s***econd)

Instructions Per Second

- $MIPS = \frac{Number\ of\ Instructions}{CPU\ Execution\ time \times 10^6}$ (***m***illion ***i***nstructions ***p***er ***s***econd)

- CPU Execution Time =(Number of Instructions × CPI) × (1 / Clock Rate)

- $MIPS = \frac{Clock\ Rate}{CPI \times 10^6}$

Instructions Per Second: Example

- A computer with 200 MHz clock rate
- Details of instruction execution

Instruction category	Percentage of occurrence	No. of cycles per instruction
ALU	35	1
Load & store	30	2
Branch	15	3
Others	20	5

Find MIPS.

Instruction category	Percentage of occurrence	No. of cycles per instruction
ALU	35	1
Load & store	30	2
Branch	15	3
Others	20	5

Amdahl's Law

- Speedup of a computer after enhancement
- Speedup in some operations, but not all
  - Overall speedup (S_o)
  - Speed up of the part that has been benefitted due to enhancement (S)
  - proportion of execution time that the part benefiting from improved resources originally occupied (p)
- $S_o = \frac{1}{(1-p)+\frac{p}{S}}$

Amdahl's Law

- Example:
  - Suppose in a program, multiplication is required 30% of time.
  - A speedup of 30 is possible for multiplication.
  - What is the overall speedup?

Amdahl's Law

- Example:
  - Suppose in a program, multiplication is required 30% of time.
  - A speedup of 30 is possible for multiplication.
  - What is the overall speedup?
  - $S_o = \frac{1}{(1-p)+\frac{p}{S}}$

Amdahl's Law

- Example:
  - Suppose in a program, multiplication is required 30% of time.
  - A speedup of 30 is possible for multiplication.
  - What is the overall speedup?
  - $S_o = \frac{1}{(1-p)+\frac{p}{S}}$
  - $S = 30, p = 0.3$
  - $S_o = \frac{1}{(1-0.3)+\frac{0.3}{30}} = \frac{1}{0.7+0.01} = 1.4$

Amdahl's Law

- Example:
  - Suppose in a program, multiplication is required 60% of time.
  - A speedup of 30 is possible for multiplication.
  - What is the overall speedup?
  - $S_o = \frac{1}{(1-p)+\frac{p}{S}}$
  - $S = 30, p = 0.6$
  - $S_o = \frac{1}{(1-0.6)+\frac{0.6}{30}} = \frac{1}{0.4+0.02} = 2.38$

Amdahl's Law

- Example:
  - Suppose in a program, multiplication is required 90% of time.
  - A speedup of 30 is possible for multiplication.
  - What is the overall speedup?
  - $S_o = \frac{1}{(1-p)+\frac{p}{S}}$
  - $S = 30, p = 0.9$
  - $S_o = \frac{1}{(1-0.9)+\frac{0.9}{30}} = \frac{1}{0.1+0.03} = 7.69$

Amdahl's Law

- Multiplication is required 30% of time; $S_o = 1.4$
- Multiplication is required 60% of time; $S_o = 2.38$
- Multiplication is required 90% of time; $S_o = 7.69$

- What can we conclude from this?

Amdahl’s Law

- Multiplication is required 30% of time; $S_o = 1.4$
- Multiplication is required 60% of time; $S_o = 2.38$
- Multiplication is required 90% of time; $S_o = 7.69$

- **Design Principle: Make the common case fast**

Representation of Information in a Computer

Data and Information

- Data and information
  - Are they same?
- Data (in computing): a sequence of symbols that may represent a fact, observation, event, etc.
- Digital data: a sequence of bits
- Bit: 0 and 1
  - Why 0 and 1?

Byte

- The smallest collection of bits that the components of a computer can access and operate on

- How big is byte?

- Symbol of bit: b   kilobit: kb

- Symbol of byte: B   kilobyte: kB

Larger Units of Data: Decimal

- 1 kilobyte (kB)= 1000 byte = 10^3 byte
- 1 megabyte (MB)= 10^3 kB = 10^6 byte
- 1 gigabyte (GB)= 10^3 MB = 10^9 byte

- These definitions are used in data transfer, hard drives, DVDs, internal bus, etc.

Larger Units of Data: Binary

- 1 kibibyte (kiB)= 1024 byte = 2^{10} byte
- 1 mebibyte (MiB)= 2^{10} kiB =2^{20} byte
- 1 gibibyte (GiB)= 2^{10}MiB =2^{30} byte

Why Digital Data/ Digital Computer?

Why Digital Data/ Digital Computer?

- Storage
- Processing
- Transmission

The Bit Pattern

- No intrinsic meaning
  - Depends on the interpretation
- A given pattern of bit may represent
  - Number, character etc.
  - Control signals
  - Status of peripheral devices etc.

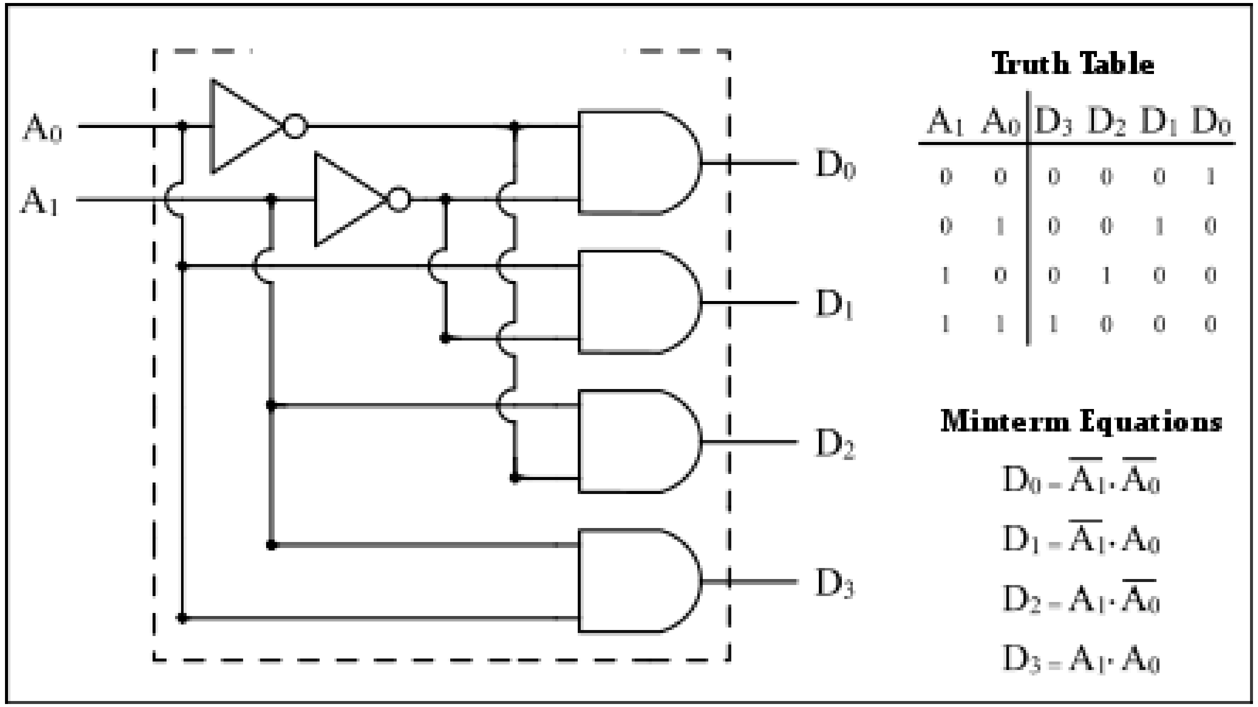


Image source: https://www.wikiwand.com/en/Binary_decoder

The Bit Pattern

- No intrinsic meaning
  - Depends on the interpretation
- A given pattern of bit may represent
  - Number, character etc.
  - Control signals
  - Status of peripheral devices etc.

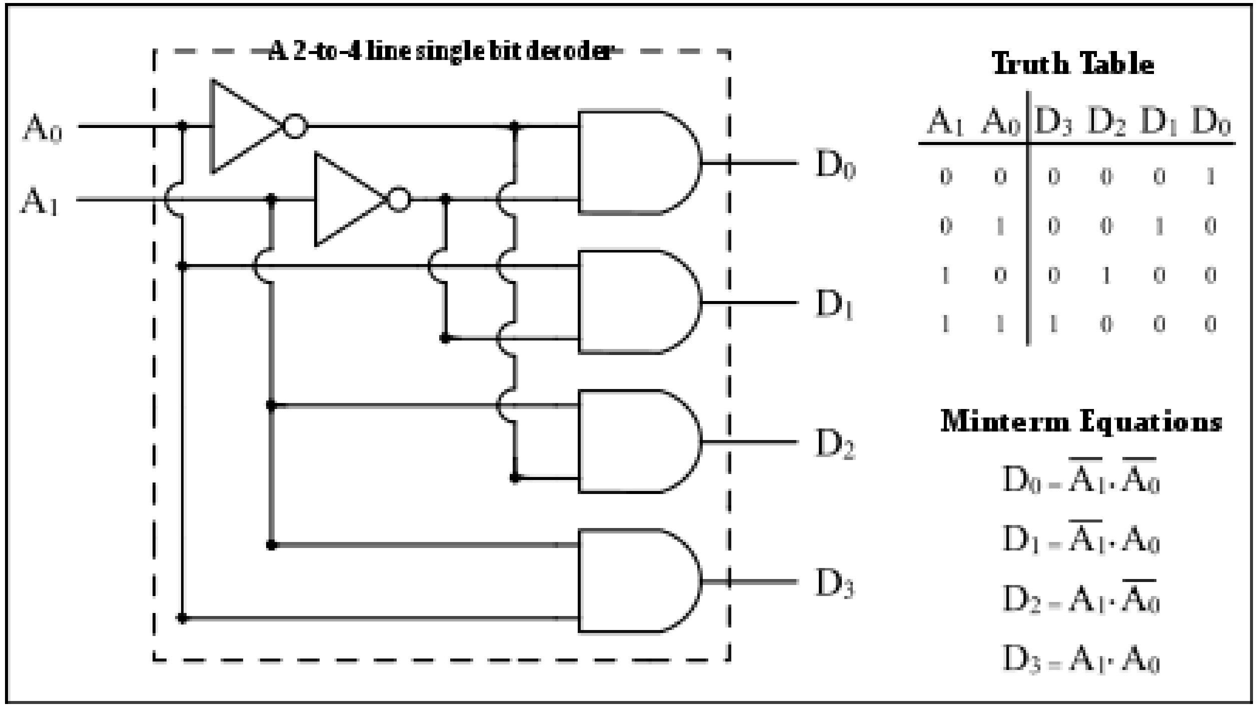
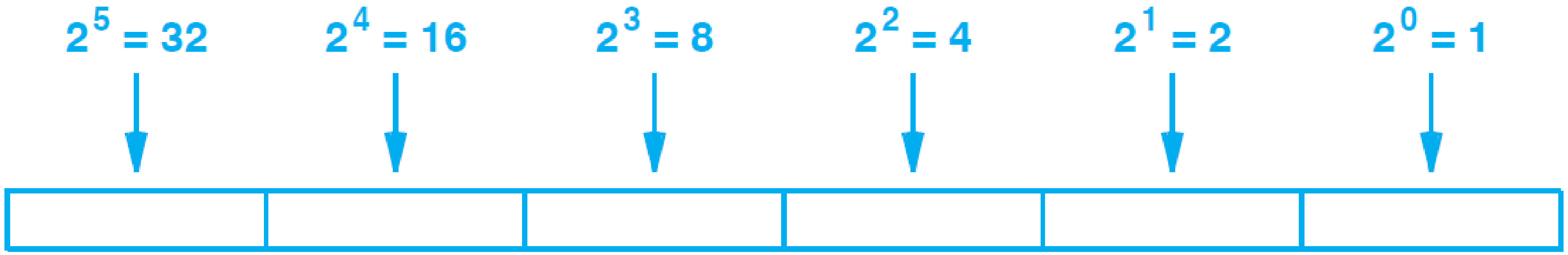


Image source: https://www.wikiwand.com/en/Binary_decoder

Binary Weighted Positional Representation



Binary Weighted Positional Representation

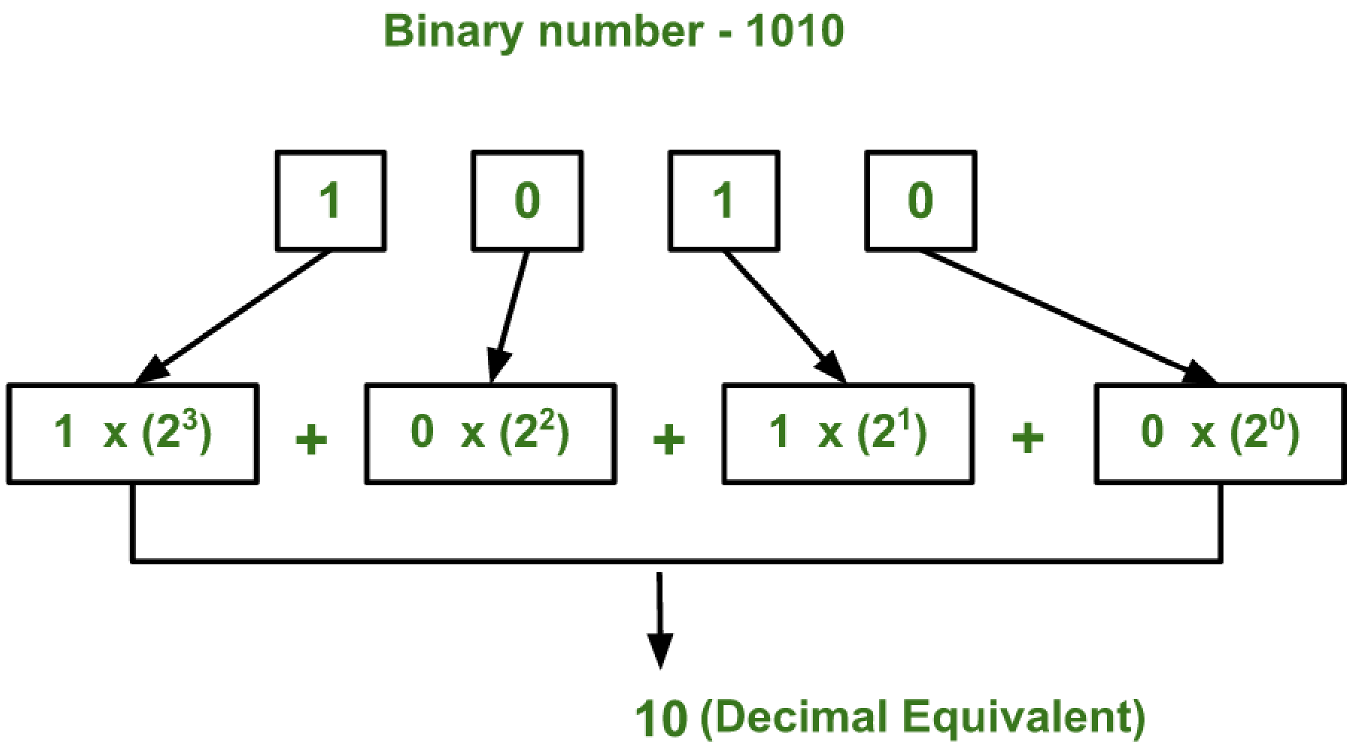


Image source: https://www.geeksforgeeks.org/program-binary-decimal-conversion/

Bit Ordering

- Left to Right ordering
- LSB and MSB
- Issues during transmission
- Specification by designer

Hexadecimal Notation

Hex Digit	Binary Equivalent	Decimal Equivalent
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
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Character Set

- Each computer system defines a character set
- A set of characters the processor and the I/O devices agree to use
  - Uppercase and lowercase letters: A, B, C, …,a, b, c, …
  - Digits: 1, 2, 3, …
  - Punctuation: ,,;,”, .,…, etc.
- Each character should fit into a byte
  - How many characters can we accommodate using 8 bit byte?

Character Set

- Bit pattern of a character: decided by the architect
- Peripheral devices and the computing system must agree on the bit pattern
- ASCII representation
  - 8 bit
  - 128 characters
  - Special symbols

Unsigned Integers

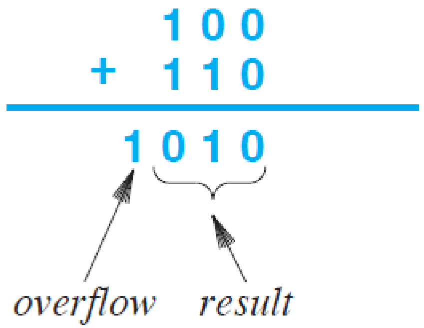
- k bit representation
- Values represented: 0 to $2^k - 1$
- What if we subtract b from a with $b > a$?
- What if we add two numbers c and d such that $c + d \geq 2^k$

Unsigned Integers

- k bit representation
- Values represented: 0 to $2^k - 1$
- What if we subtract b from a with $b > a$? **Underflow**
- What if we add two numbers c and d such that $c + d \geq 2^k$ **Overflow**

Unsigned Integers: Wraparound

- Add/ subtract the numbers
- Represent lower *k* bits of the result as answer
- Hardware sets the overflow or underflow bit to indicate the respective situation



Numbering Bits and Bytes

- Importance during data transmission
  - Which one would be transferred first: MSB or LSB?
- Little endian: stores and transmits bytes from LSB to MSB
  - Bit little endian
- Big endian: stores and transmits bytes from MSB to LSB
  - Bit big endian

Numbering Bits and Bytes

10010101 00110101 11001101 10101011

- Little endian: stores and transmits bytes from least significant to most significant
  - Bit little endian
- Big endian: stores and transmits bytes from most significant to least significant
  - Bit big endian

00011101 10100010 00111011 01100111

(a) Integer 497,171,303 in binary positional representation

	loc. i	*loc. i+1*	*loc. i+2*	*loc. i+3*	
. . .	01100111	00111011	10100010	00011101	. . .

(b) The integer stored in little endian order

	loc. i	*loc. i+1*	*loc. i+2*	*loc. i+3*	
. . .	00011101	10100010	00111011	01100111	. . .

(c) The integer stored in big endian order

Signed Binary Integers

- Sign magnitude
  - A sign bit (0 for positive and 1 for negative)
  - Bits for magnitude (positional representation)

- One's complement
  - k bit representation
  - MSB: 0 for positive, the rest $(k-1)$ bits indicate magnitude
  - Negative number: invert all bits from the corresponding positive number

One's Complement

Bit Pattern	Decimal Value
0000 0000	0
1111 1111	-0
0000 0001	1
1111 1110	-1
0000 0011	3
1111 1100	-3
0001 1111	31
1110 0000	-31

Image source: https://www.cs.uaf.edu/2000/fall/cs301/notes/notes/node33.html

Two’s Complement

- Positive Integer: same as one’s complement

- Negative integer:
  1. Take the corresponding positive integer
  2. Invert each bit
  3. Add 1

Binary String	Unsigned (positional) Interpretation	Sign Magnitude Interpretation	One's Complement Interpretation	Two's Complement Interpretation
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
1010	10	−2	−5	−6
1011	11	−3	−4	−5
1100	12	−4	−3	−4
1101	13	−5	−2	−3
1110	14	−6	−1	−2
1111	15	−7	−0	−1

Sign Extension

- Copying an integer Q of k bit to a storage of more than k bit
- Decimal equivalent 1000 in two's complement
  - -8
- I want to copy this number to an 8-bit storage

?	**?**	**?**	**?**	**1**	**0**	**0**	**0**

Sign Extension

- Copying an integer Q of k bit to a storage of more than k bit
- Decimal equivalent 1000 in two's complement
  - -8
- I want to copy this number to an 8-bit storage

1	**1**	**1**	**1**	**1**	**0**	**0**	**0**

Sign Extension

- Copying an integer Q of k bit to a storage of more than k bit

- Decimal equivalent 0100 in two's complement
  - 4
- I want to copy this number to an 8-bit storage

?	**?**	**?**	**?**	**0**	**1**	**0**	**0**

Sign Extension

- Copying an integer Q of k bit to a storage of more than k bit
- Decimal equivalent 0100 in two's complement
  - 4
- I want to copy this number to an 8-bit storage

0	**0**	**0**	**0**	**0**	**1**	**0**	**0**

Sign Extension

- In two’s complement arithmetic, when an integer Q composed of k bits is copied to an integer of more than k bits, the additional high-order bits are made equal to the top bit of Q.

- Extending the sign bit ensures that the numeric value of the two will be the same if each is interpreted as a two’s complement value.

Floating Point

- Any number can be expresses using a mantissa and an exponent

- Example in decimal:
  - $-98671 = -98.671 \times 10^3 = -986.71 \times 10^2$
  - Normalization: -9.8671×10^4 (by eliminating leading zeros)

- Example in binary:
  - $101.10 = 10.110 \times 2^1 = 0.010110 \times 2^4$
  - Normalization: 1.0110×2^2 (by eliminating leading zeros)
  - Leading bit is always 1 (except for the number 0)

Floating Point

- Example in binary:
  - $101.10 = 10.110 \times 2^1 = 0.010110 \times 2^4$
  - Normalization: 1.0110×2^2 (by eliminating leading zeros)
  - Leading bit is always 1 (except for the number 0)
- Computer need not store leading bit
  - Hardware concatenates a leading 1 in mantissa during computation

Single and Double Precision

- Single precision: 32 bit representation

- Double precision: 64 bit representation

An Example

- Single precision representation for decimal value 17.25
- $17.25_{10} = 10001.01_2$

31 30 23 22 0

S	exponent	mantissa (bits 0 - 22)

- $10001.01 = 1.000101 \times 2^4 = 1.00010100000000000000000 \times 2^4$
- $S = 0$
- $\text{exponent} = 4_{10} = 100_2 = 00000100_2$
- $\text{Mantissa} = 1.00010100000000000000000$

0	00000100	00010100000000000000000

An Example

- Single precision representation for decimal value -17.25

- $-17.25_{10} = -10001.01_2$
- $-10001.01 = -1.000101 \times 2^4 = -1.00010100000000000000000 \times 2^4$
- $S = 1$
- exponent $= 4_{10} = 100_2 = 00000100_2$
- Mantissa $= 1.00010100000000000000000$

1	00000100	00010100000000000000000

An Example

- Single precision representation for decimal value -0.25
- $-0.25_{10} = -0.001_2$

31	30 23	22 0
S	exponent	mantissa (bits 0 - 22)

- $-0.001 = -1 \times 2^{-3} = -1.00000000000000000000000 \times 2^{-3}$
- $S = 1$
- Mantissa $= 1.00000000000000000000000$
- exponent $= -3_{10} = -011_2 = -00000011_2$ **How to store this '−' sign?**

1	?	00000000000000000000000

An Example

- Single precision representation for decimal value -0.25

- $-0.25_{10} = -0.001_2$
- $-0.001 = -1 \times 2^{-3} = -1.00000000000000000000000 \times 2^{-3}$
- $S = 1$
- Mantissa $= 1.00000000000000000000000$
- exponent $= -3_{10} + 127_{10}(\boldsymbol{bias}) = 124_{10} = 01111100_2$

1	01111100	00000000000000000000000

The IEEE Standard 754

- Single precision: 32 bit representation

31	30 … 23	22 … 0
S	Biased Exponent	Mantissa

- Double precision: 64 bit representation

63	62 … 52	51 … 0
S	Biased Exponent	Mantissa

Floating Point Error

0	11	10011

$1.10011 \times 2^3 = 1100.11 = 12.75_{10}$

0	01	00011

$1.00011 \times 2^1 = 10.0011 = 2.1875_{10}$

Result of Addition: $1110.1111 = 14.9375_{10}$

After Normalization: $1.1101111 \times 2^3 = 14.9375_{10}$

The number that can be stored

$$1.11011 \times 2^3 = 14.75_{10}$$

The IEEE Standard 754

- Single precision: 32 bit representation

31	30 ... 23	22 ... 0
S	Biased Exponent	Mantissa

- Double precision: 64 bit representation

63	62 ... 52	51 ... 0
S	Biased Exponent	Mantissa

The IEEE Standard 754

31	30 23	22 0
S	Biased Exponent (b bits)	Mantissa

Type	Biased Exponent	Mantissa	Sign
Positive Zero	0	0	0
Negative Zero	0	0	1
Deformalized Numbers	0	Non zero	0/1
Infinities	255 (single precision) $2^b - 1$	0	0/1
NaN	255 (single precision) $2^b - 1$	Non zero	0/1
Normalized Numbers	1 to 254 1 to $(2^b - 2)$	Non zero	0/1