

C programming

MODULE 1

CSE Dept.

ASIET



APJ Abdul Kalam Technological University
Adi Shankara Institute of
Engineering and Technology, Kalady
Department of Computer Science & Engineering &
Information Technology



TEXT BOOKS

Text Books

1. Schaum Series, Gottfried B.S., Tata McGraw Hill, Programming with C
2. E. Balagurusamy, McGraw Hill, Programming in ANSI C
3. Asok N Kamthane, Pearson, Programming in C
4. Anita Goel, Pearson, Computer Fundamentals

Reference Books

1. Anita Goel and Ajay Mittal, Pearson, Computer fundamentals and Programming in C
2. Brian W. Kernighan and Dennis M. Ritchie, Pearson, C Programming Language
3. Rajaraman V, PHI, Computer Basics and Programming in C
4. Yashavant P, Kanetkar, BPB Publications, Let us C

Course Outcome

- Analyze a computational problem and develop an algorithm/flowchart to find its solution

Module 1

Basics of Computer Hardware and Software



syllabus

Basics of Computer Architecture: processor, Memory, Input& Output devices Application Software & System software: Compilers, interpreters, High level and low level languages Introduction to structured approach to programming, Flow chart Algorithms, Pseudo code (bubble sort, linear search - algorithms and pseudocode)

Previous University questions

9. Differentiate Machine Language, Assembly Language and High level Language.

(10 marks)

1. a. Compare between compiler and assembler. (3)

b. Mention any four keywords and their meaning. (2)

9. a. Draw the flowchart and develop the algorithm for finding the area of a triangle by reading three sides (5)

b. Explain the different datatypes in C. (5)

Answer any five questions, each carries 5 marks.

- 9 a) Explain any five kinds of operators in C. (5)
 b) Draw a flowchart to find the sum of digits of an integer. (5)

9. Differentiate Machine Language, Assembly Language and High level Language.

(10 marks)

Answer all questions, each carries 5 marks.

- 1 Differentiate between machine language, assembly language and high level languages? What is the difference between compiler and assembler? (5)

1 What is a compiler? How does it differ from an interpreter?

(5)

9 Write an algorithm and draw flow chart for finding the Average mark for one subject in a class of 50 students. (10)

1 What are the advantages of High Level languages? How a high-level language is converted to Machine language? (5)

What is a Computer

- *Computer* is an electronic device that accepts data as input, processes the input data by performing mathematical and logical operations on it, and gives the desired output.

Characteristics of Computer

- Speed
- Accuracy
- Diligence
- Storage capability
- Versatility

Generations of Computer (Not in Syllabus)

- **Five generations**
- Categorized on the basis of
 - Technology used by them (hardware and software)
 - Computing characteristics (speed - number of instructions executed per second)
 - Physical appearance
 - Their applications

Generations of Computer



First

- 1940-56
- Vacuum Tubes



Second

- 1956-63
- Transistors



Third

- 1964-71
- Integrated Circuits



Fourth

- 1971-Present
- Microprocessors



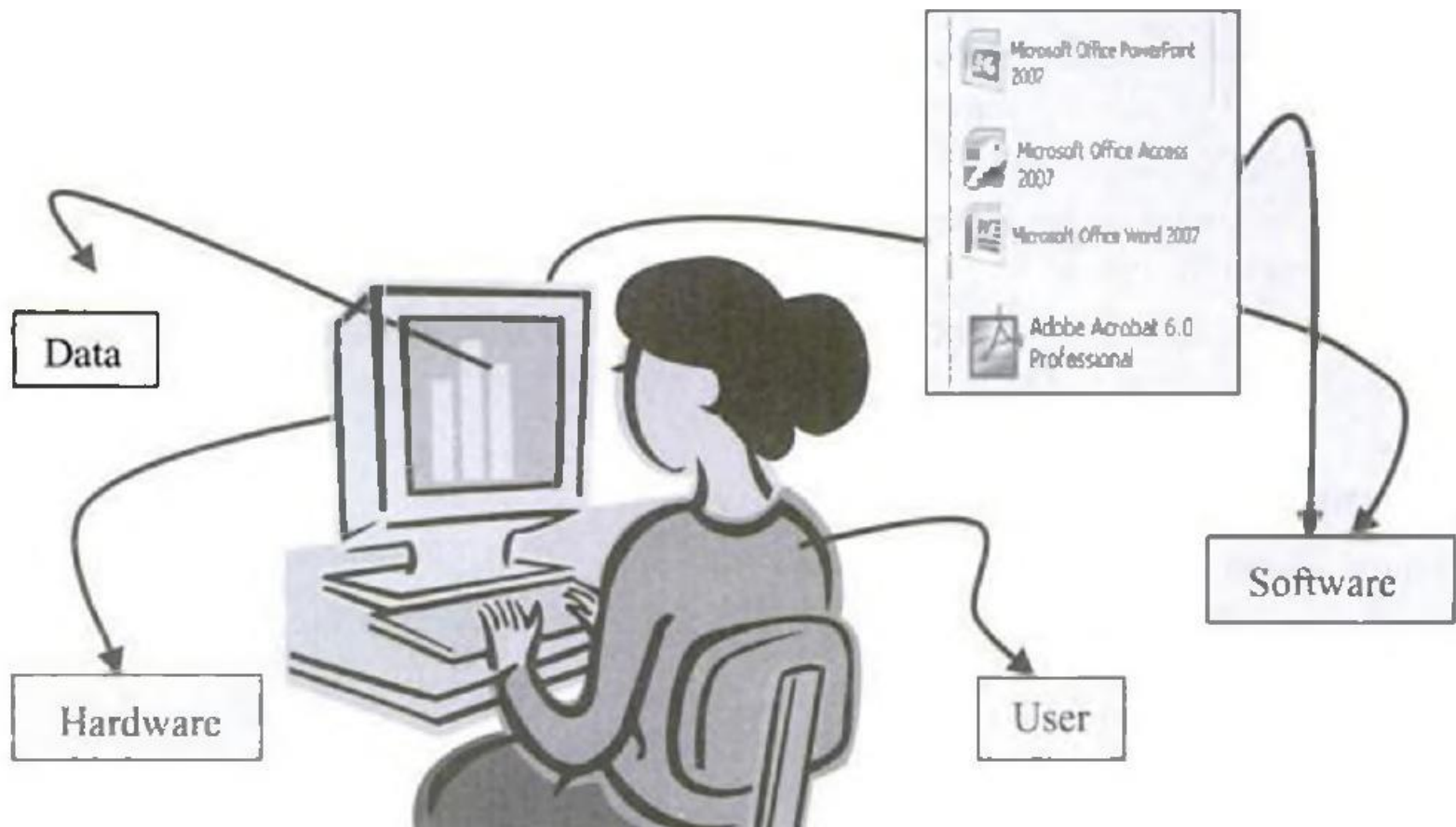
Fifth

- Present and Next
- Artificial Intelligence

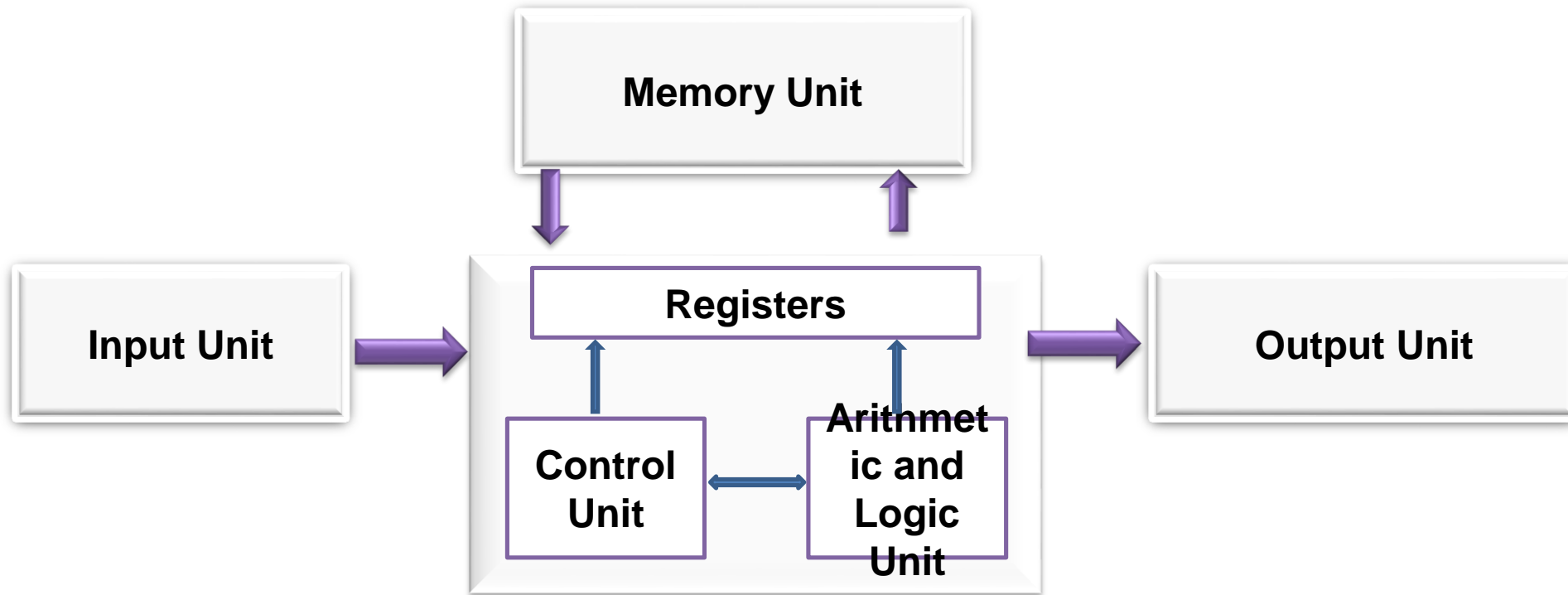


Computer System

- Four parts
 - **Hardware:** Mechanical parts of computer, e.g. Keyboard, monitor, hard disk drive etc.
 - **Software:** Set of instructions (Program) and documentation. Instructs the computer about the task to be performed.
 - **Data** - Isolated values or raw facts.
 - Provided as input to computer for processing
 - **Users** - People who write computer programs or interact with computer.



Components -Computer Hardware



Components -Computer Hardware

- 3 main components

1. **I/O Unit** - user interacts with computer via I/O unit
 - Input unit accepts data from the user (Devices - keyboard, trackball, mouse)
 - Output unit provides processed data to user (Devices - monitor and printer)
2. **CPU** - Controls, coordinates , supervises operations of computer.
 - Arithmetic Logic Unit (ALU) - performs arithmetic and logic operations on input data.
 - Control Unit (CU) - controls overall operations of computer
 - Registers - temporary storage of data, instructions, addresses, intermediate results
3. **Memory Unit** – Stores data, instructions etc.
 - *Main memory or Primary memory* - Stores data, instructions, intermediate results ,output, *temporarily*
 - Secondary memory – Stores data, programs, output *permanently*
 - Magnetic disks, optical disks and magnetic tapes

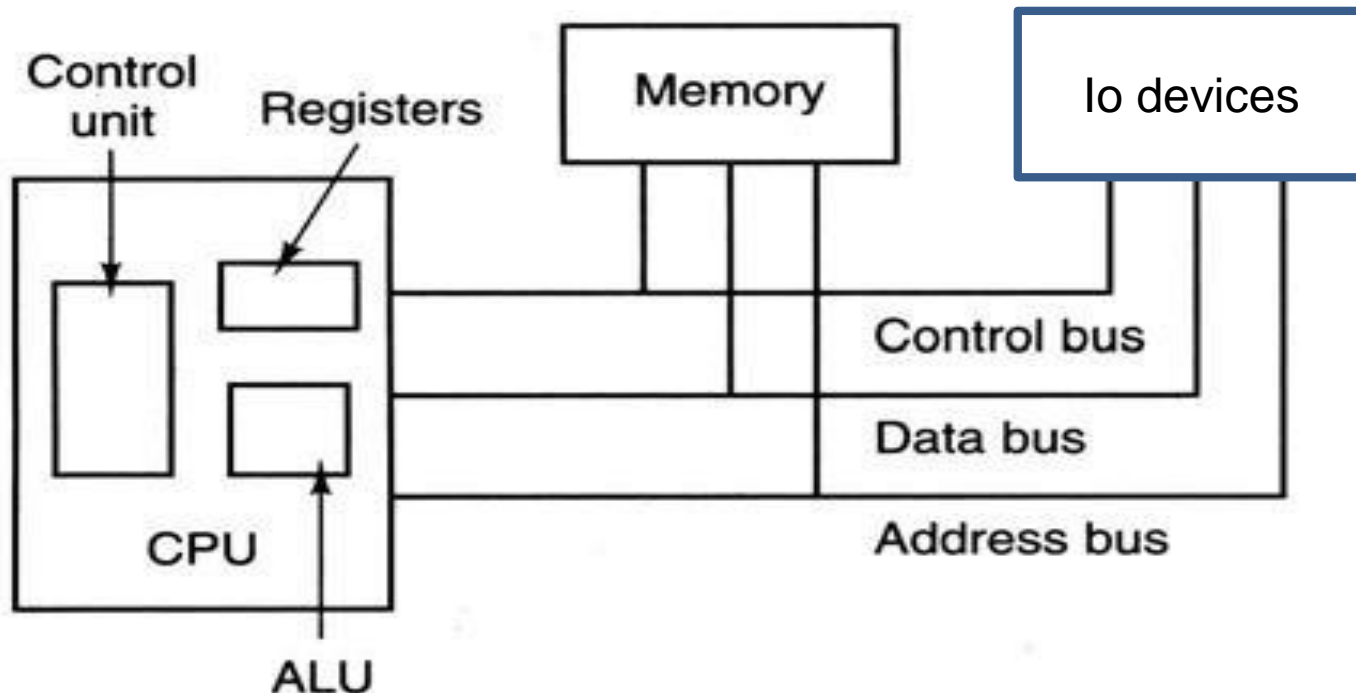
VON NEUMANN ARCHITECTURE

- Computer architecture has undergone incredible changes in the past 20 years but One element has remained constant throughout the years, however, and that is the **von Neumann concept of computer design**.
- The basic concept behind the von Neumann architecture is the ability to **store program instructions in memory along with the data**
- Until von Neumann proposed this possibility, each computing machine was designed and built for a single program.

VON NEUMANN ARCHITECTURE

composed of 3 components :

- central processing unit (CPU),
- memory,
- I/O devices .



- 3 components are connected to each other through a collection of signal lines known as a bus.
 - control bus
 - data bus
 - address bus.
- Each bus contains several wires that allow for the parallel transmission of information between various hardware components.

- The **execution of a program** in a von Neumann machine requires the use of 3 components
- Initially, a program has to be loaded from secondary storage and load it into the memory.

- Once the program is in memory CPU to begin executing the program instructions.
- Each instruction to be executed must first be retrieved from memory sequentially.
- After an instruction fetch it is put into a special register in the CPU, called the **instruction register (IR)**.
- While in the IR, the instruction is decoded to determine what type of operation should be performed.

- If the instruction requires operands, these are fetched from memory
- The instruction is then performed, and the results are stored back into memory
- This process is repeated for each instruction of the program until the program's end is reached. .

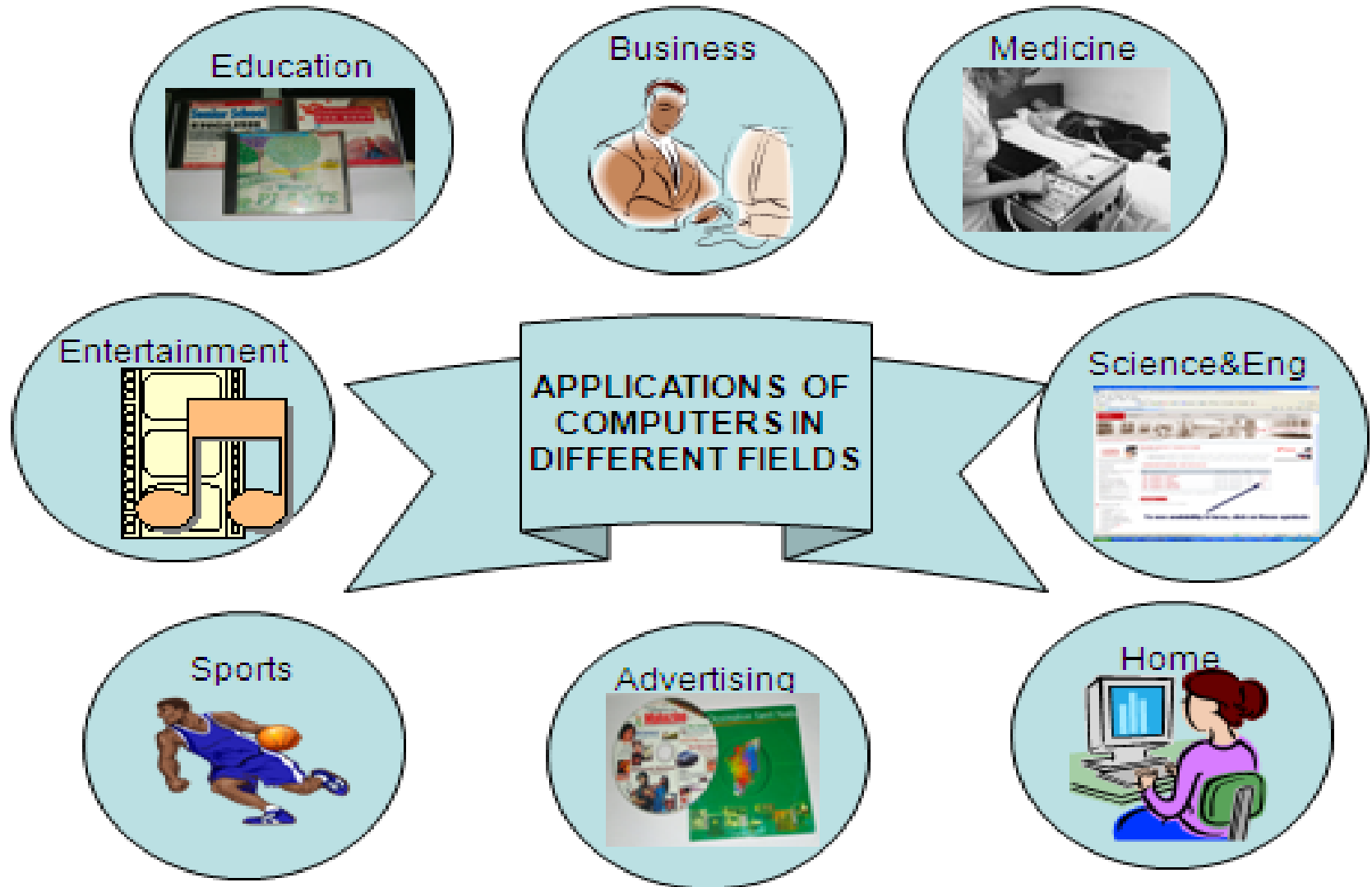
disadvantage

- Here instruction fetch and a data operation cannot occur at the same time because they share a common bus. This is referred to as the **von Neumann bottleneck**

Advantage

- simpler to implement in real hardware.
- Less circuitry

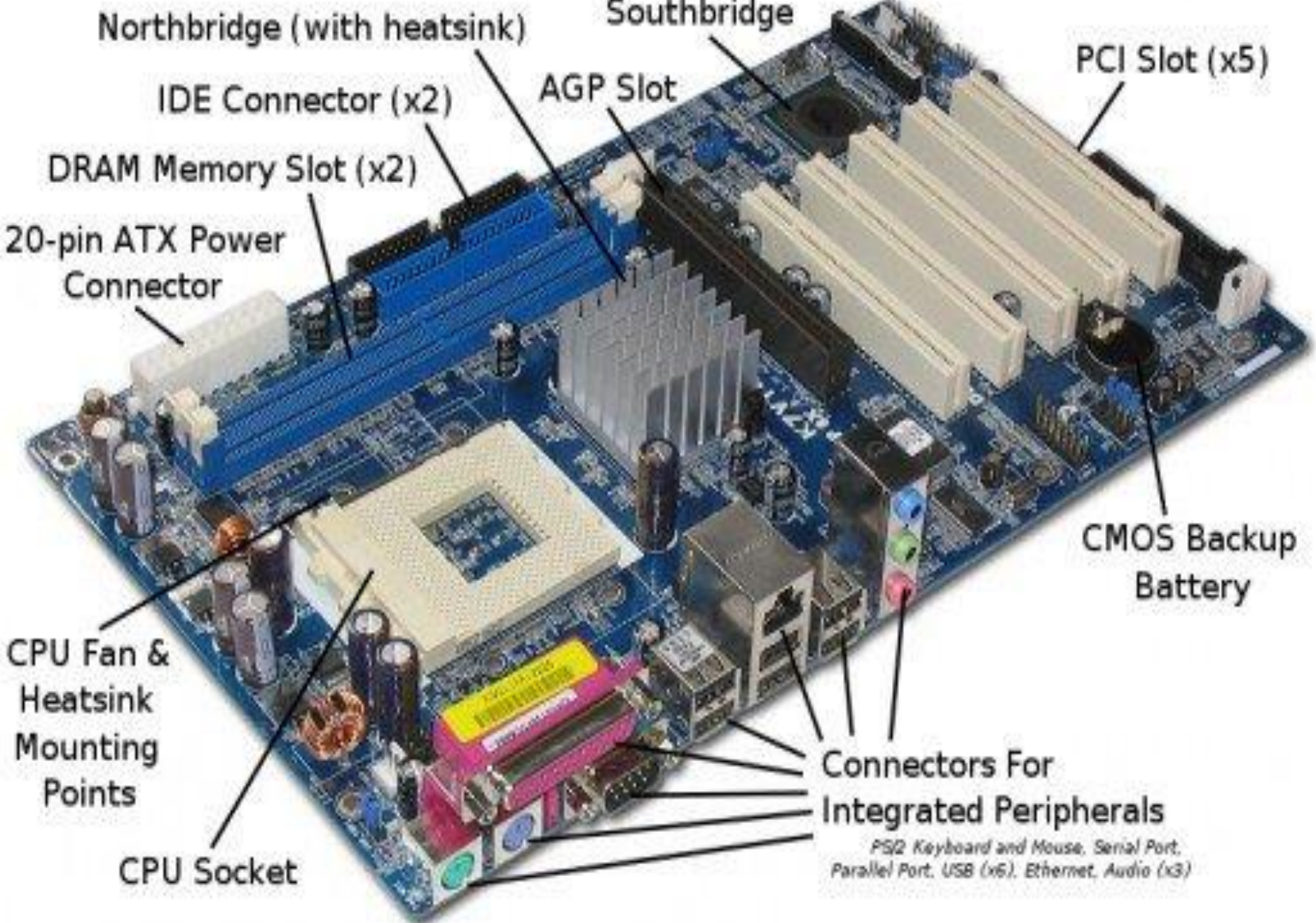
Application of Computers



Computer System Hardware

Central Processing Unit

- CPU or processor - *brain of computer*
- Parts - ALU, CU, Registers
- Executes *stored program instructions*
- Processing
 - Gets data and instructions from memory
 - Interprets instructions and perform operation
 - Sends processed data or result to memory
- Fabricated as IC chip - *microprocessor*
 - Microprocessor plugged into *Motherboard*
 - *Motherboard*: A circuit board with electronic circuit and connects microprocessor with other hardware components





ALU

- Two units – arithmetic unit and logic unit.
- Arithmetic unit
 - Performs arithmetic operations on the data
 - E.g. addition, subtraction, multiplication, division
- Logic unit
 - Performs logic operations.
 - E.g. greater than, less than or equal to condition
- Uses *registers* to hold data being processed.

Registers

- High speed storage areas, least storage capacity, directly accessed & manipulated by CPU during instruction execution
- CPU's *working memory*
- Registers: ACC, IR, PC, MAR, MBR, DR
- Number of registers: Ten to hundreds
 - Depends on type & complexity of processor
- Size of register (word size)
 - 8, 16, 32, 64 bits
 - Nowadays, PCs have 64-bit registers

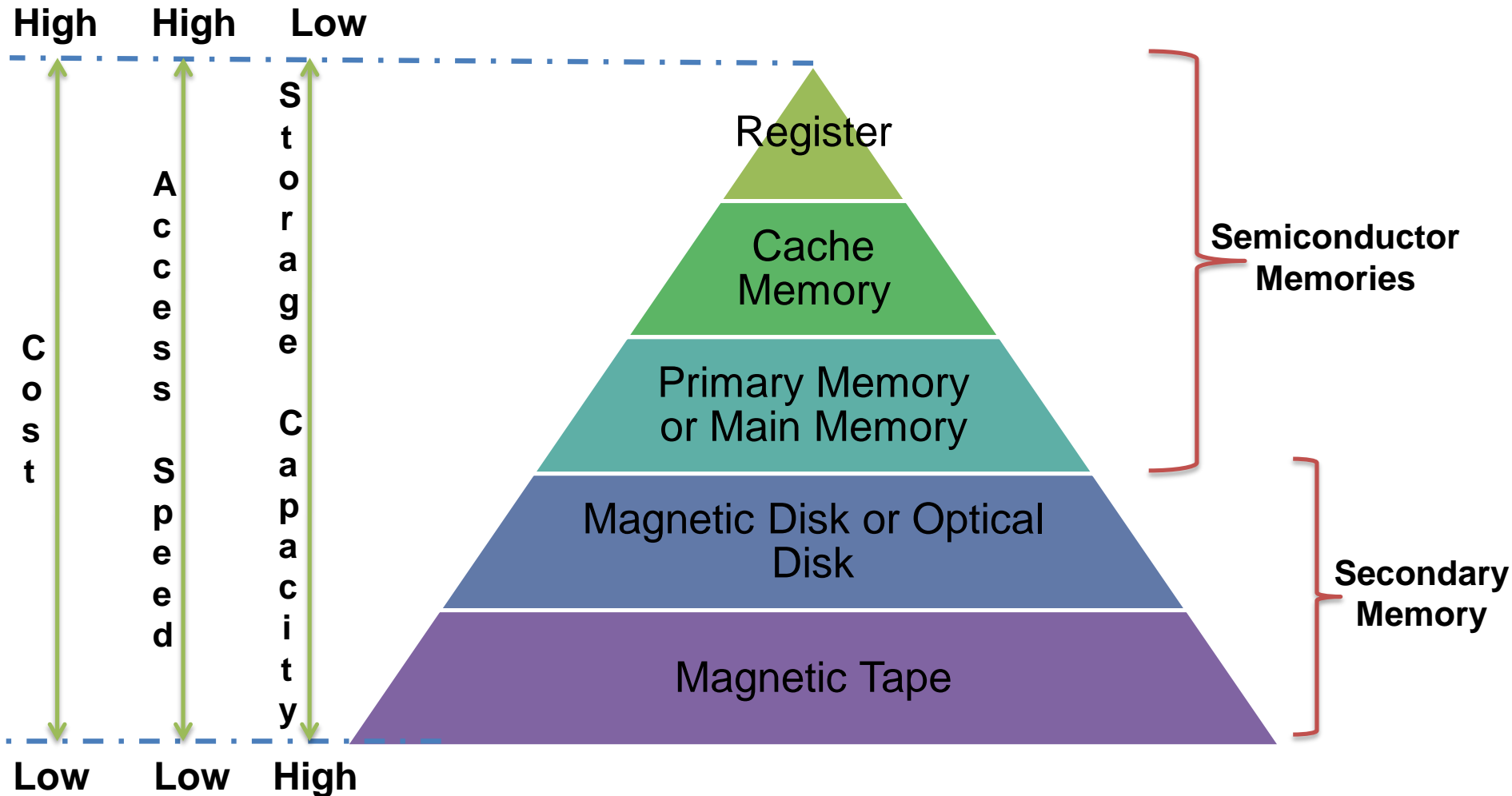
Control Unit

- Organizes processing of data & instructions
 - Tells when to fetch data & instructions, what to do, where to store results, sequencing of events during processing etc.
- Acts as a supervisor
- Controls and coordinates activity of other units
- Does not do actual processing of data
- Coordinates I/O devices
- Holds CPU's Instruction Set (list of all operations that CPU can perform)

Memory Unit

- Cache Memory
- Primary or Main memory
- Secondary memory

Memory Hierarchy



Cache Memory

- Very high speed memory in between RAM & CPU
- Increases speed of processing
- CPU first checks cache for data. If data is not found in cache, then looks for it in RAM
- Cache
 - Built-in Level 1 (L1) cache & Level 2 (L2) cache.
 - Separate chip on motherboard for Level 3 (L3) cache
 - CPUs have 256KB L1 cache & 2MB of L2 cache
- Very expensive, so smaller in size.

Primary Memory

- Main memory of computer. Semiconductor memory.
- Stores data & instructions during processing of data
- Two kinds - RAM and ROM
- RAM
 - Volatile - *temporary storage* for data & instructions
 - Stores input data, before processing; output data before sending to output device; intermediate results
 - *Limited storage* capacity, due to its *high cost*
- ROM
 - Non-volatile, read only memory. *Permanent storage*
 - Stores standard processing programs that permanently reside in computer
 - Comes programmed by manufacturer

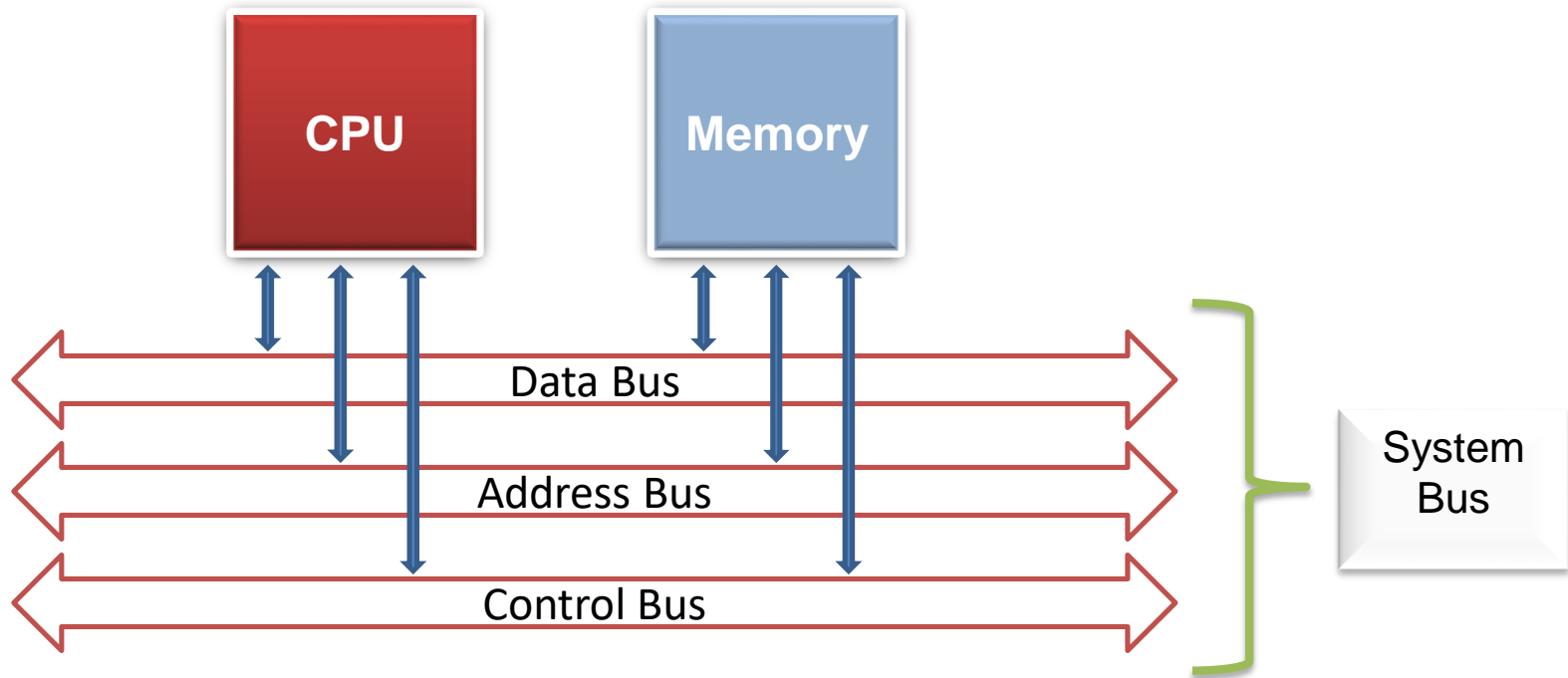
Secondary Memory

- Stores data & instructions *permanently*
- *Non-volatile* memory
- Provides *back-up storage* for data & instructions
- *High storage capacity.*
- *Cheaper* than primary memory
- Takes *longer time to access* data & instructions, than the primary memory
 - E.g. Magnetic tape drive, floppy drive, disk drive, optical disk drive

Bus

- Set of electronic signal pathways that allows information and signals to travel between components
- CPU, I/O unit, memory unit interconnected by bus
- Bus width - number of wires in the bus
- Two types - Internal Bus & External Bus
 - Internal Bus (System Bus) connects components inside motherboard like, CPU and system memory.
 - External Bus (Expansion Bus) connects external devices, I/O ports, drives to rest of computer
- System bus or expansion bus comprise of
 - Control bus, Address bus, Data bus

Interaction between CPU and Memory

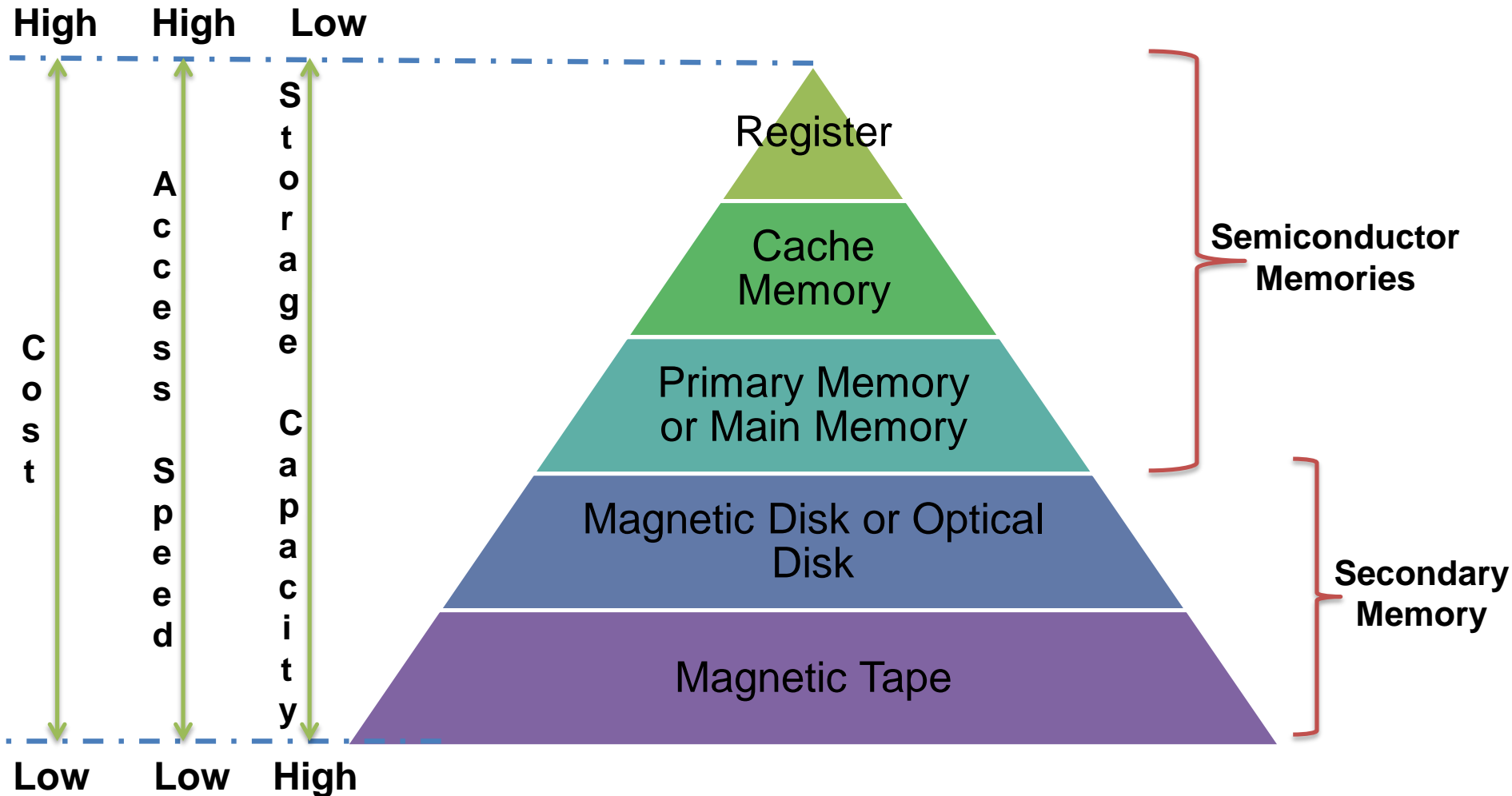


Memory and Storage devices

Memory Representation

- **Binary digit or bit** - 0 or 1
 - Smallest unit of representation of data
- **Byte** - Group of 8 bits
 - Smallest unit of data handled by computer
 - 1 byte stores 2^8 , i.e. 256 different combinations of bits
- **Word**: group of 2, 4 or 8 bytes
- 1 Kilobyte (KB) = 2^{10} = 1024 bytes
- 1 Megabyte (MB) = 2^{20} = 1024KB
- 1 Gigabyte (GB) = 2^{30} = 1024 MB = 1024 * 1024 KB
- Memory logically organized as linear array of locations

Memory Hierarchy



Input and Output Devices

.

Contents

- I/O Unit
- Input Devices
 - Human data devices: Keyboard, mouse, trackball, joystick, digitizing tablet, light pen, touch screen
 - Source data entry devices: Audio, video, optical devices (scanner, MICR, OCR, OMR, bar code reader)
- Output devices
 - Hard copy output: Printer, plotter, microfiche
 - Soft copy devices: Monitor, VDT, video, audio
- I/O ports
- Working of I/O system



Track Ball

I/O Unit

- Composed of two parts
 - Input unit
 - Output unit
- Input unit → Provides input to computer
- Output unit → Receives output from computer

Input Unit

- Accepts input data from user via input device
- Input devices: Keyboard, mouse, trackball, joystick. Also by scanning images, voice recording, video recording
- Input devices use its input interface and translates input data into machine readable form
- Provides transformed input data for processing

Output Unit

- Accepts output data from computer via output device
- Output devices: Display screen, printer, plotter, speaker
- Output device uses its output interface, transforms processed data in machine understandable form to human readable form
- Provides transformed output to user
- Input-Output devices: Provide input to computer and get output from computer. Used by both input unit and output unit.
 - E.g. Hard disk drive, optical disk drive

I/O Devices

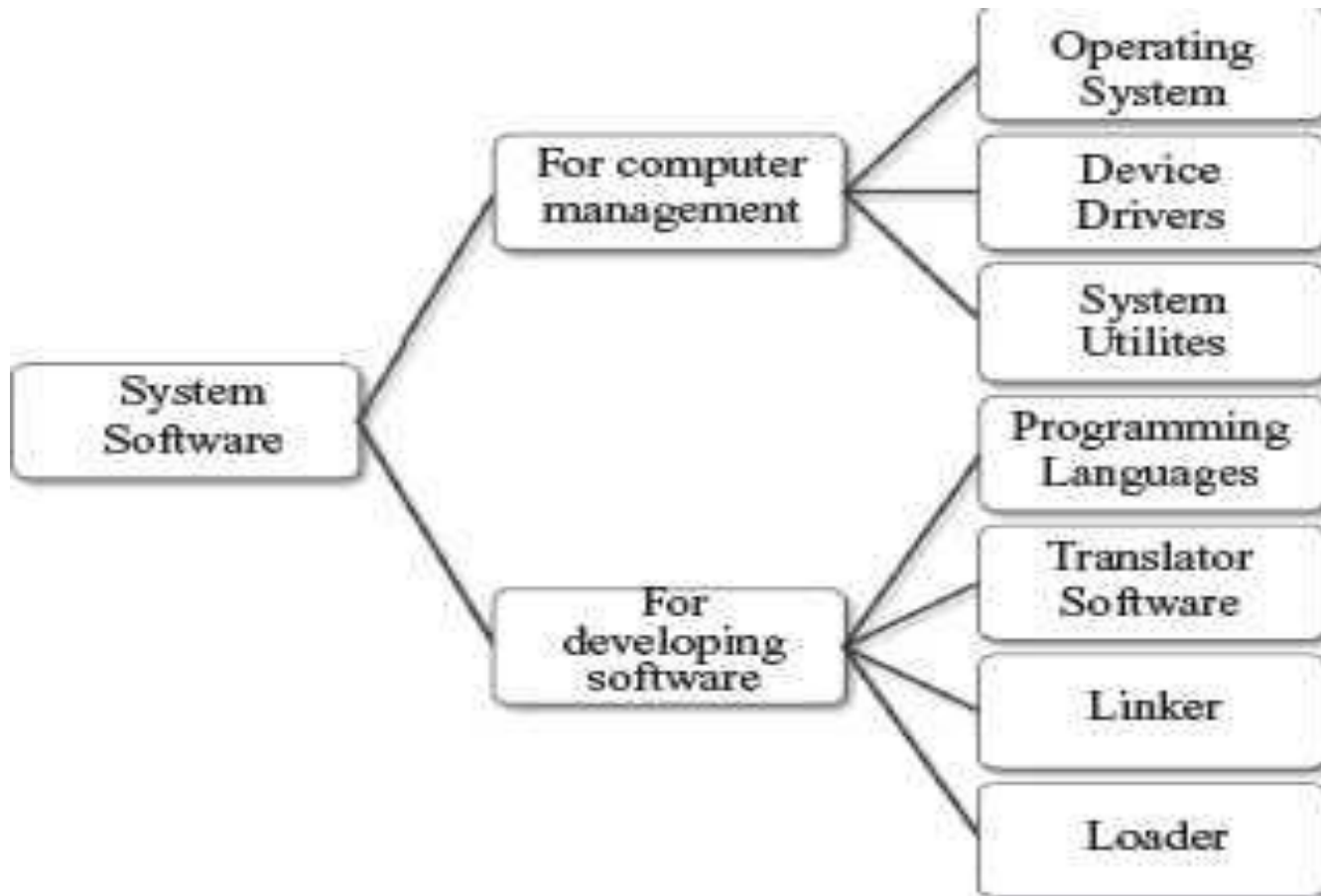
| | |
|-----------------------------|--|
| Input Devices | Keyboard, Mouse, Digitizing Tablet, Track Ball, Joystick, Touch Screen, Light Pen, Speech Recognition System, Digital camera, Scanner, Magnetic Ink Character Recognition (MICR), Optical Character Recognition (OCR), Optical Mark Recognition (OMR), Bar code Reader |
| Output Devices | Monitor, Visual Display Terminal, Printer, Plotter, Computer Output on Microfilm (COM), Video Output System, Audio Response System |
| Input-Output Devices | Hard disk drive, Floppy disk drive, USB drive, CD drive, DVD drive |

Interaction of User and Computer

Types of Software

- Two categories
 - System Software
 - Application Software
- System software :
 - Basic functions that are performed by the computer
- Application software:
 - Used by users to perform specific tasks
 - The user can choose the appropriate application software, for performing a specific task, which provides the desired functionality

Types of Software



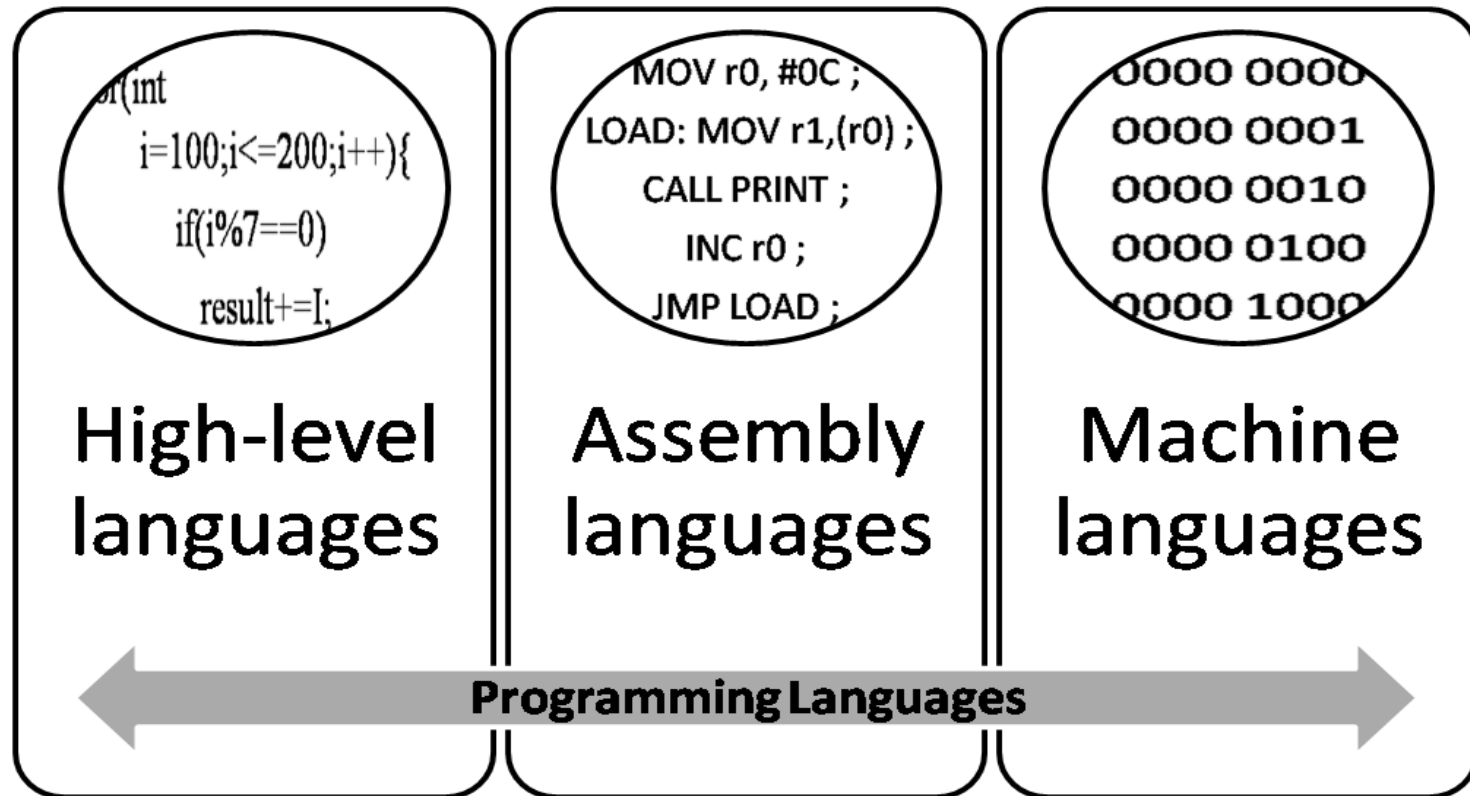
System Software

- Required for working of computer itself
- Controls computer hardware
- Acts as interface between user, application software, computer hardware
- Two categories
 - *Management and functionality of computer*
 - Functionality of components, like processor, I/O devices etc.
 - Provides support for services, like OS, device driver, system utility
 - *Development of application software*
 - Provides software tools like programming language software, translator software, loader, linker

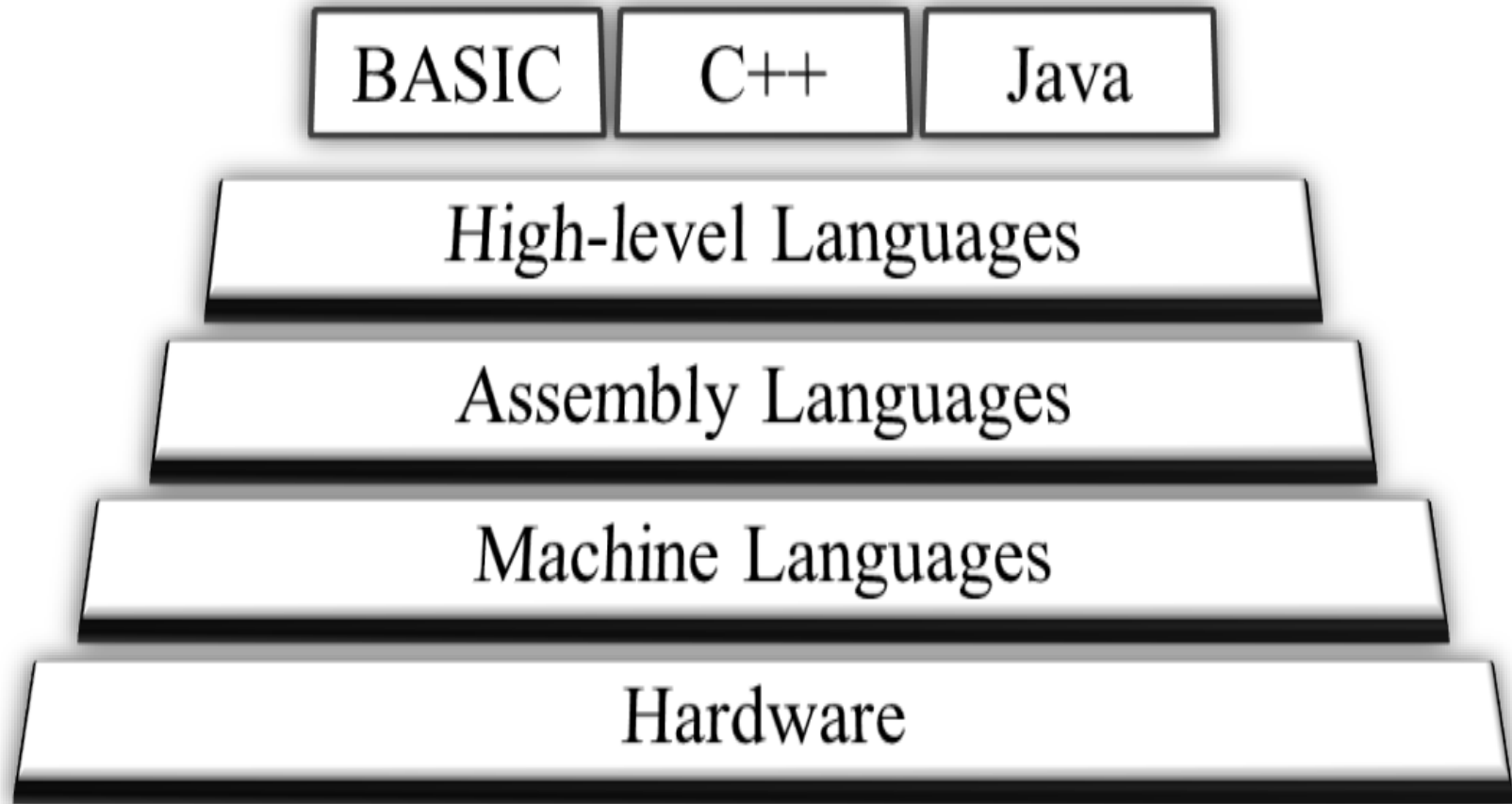
Programming Languages

- A Programming Language consists of a set of vocabulary and grammatical rules, to express the computations and tasks that the computer has to perform.
- codify the algorithms precisely
- Each language has a unique set of keywords and a special syntax for organizing program instructions

Programming Languages



Programming language hierarchy



Generations of Programming languages

First Generation

- Machine Language

Second Generation

- Assembly Language

Third Generation

- C, Cobol, Fortran, Pascal, C++, Java, ActiveX etc.

Fourth Generation

- .Net (VB.Net, C#.NET etc.)
- Scripting Languages (Javascript, Microsoft Frontpage etc.)

Fifth Generation

- LISP, Prolog

Classification of Computer Languages



HIGH LEVEL LANGUAGE

ASSEMBLY LANGUAGE

MACHINE LANGUAGE

HARDWARE

Machine Language

- Language directly understood by a computer.
- Normally written as strings of binary 0s and 1s.
 - For eg: 101101010 has a specific meaning for a computer
- Specific to a hardware
- called machine code or object code.
- An instruction consists of two parts:
 - Operation Code (OPCODE)
 - Address of one or more memory locations (OPERANDS)

Machine Language

- Instruction Format:



- OPCODE tells the computer which operation to perform from the instruction set of the computer.
- OPERAND tells the address of the data on which the operation is to be performed.

Machine Language

- A sample machine language program:

```
0010000000000001100111001
0011000000000010000100001
0110000000000011100101110
101000111111011100101110
0000000000000000000000000
```

In Binary
(Difficult to read and understand)

Machine Language

- Advantages:
 - Can be executed very fast
 - Suited for small computers having limited memory.
- Disadvantages:
 - Machine dependent
 - Difficult to program
 - Error prone
 - Difficult to modify.

Assembly Language

- Programming language that overcomes the limitations of machine level language.
- Uses Mnemonic Operation Codes and Symbolic Addresses.
 - For eg: ADD, SUB, etc.

Assembly Language

- An instruction consists of three parts:
 - A label - symbolic address of the instruction
 - An operation code - operation to be performed
 - An operand - symbolic address of memory
- Format:
 - **LABEL: OPERATION CODE OPERAND**
- Eg:- **ADD A, B** ; Adds the values at the operands addressed by **A and B**, result at **A**

Assembly Language

■ Advantages:

- Easy to understand
- Easy to locate and correct errors
- Easy to modify
- Efficiency almost same as machine language

■ Disadvantages:

- Machine dependant
- An intermediate translating program is required
- Cannot be executed in small sized computers
- Knowledge of hardware required.

High Level Languages

- Machine independent
- Knowledge about the internal structure of the computer is not necessary
- Uses English words and mathematical symbols.

High Level Languages

- Advantages:
 - Machine independent
 - Easy to learn and use
 - Less errors during program development
 - Easy to modify and maintain

- Disadvantages:
 - Takes more time to run and require more storage

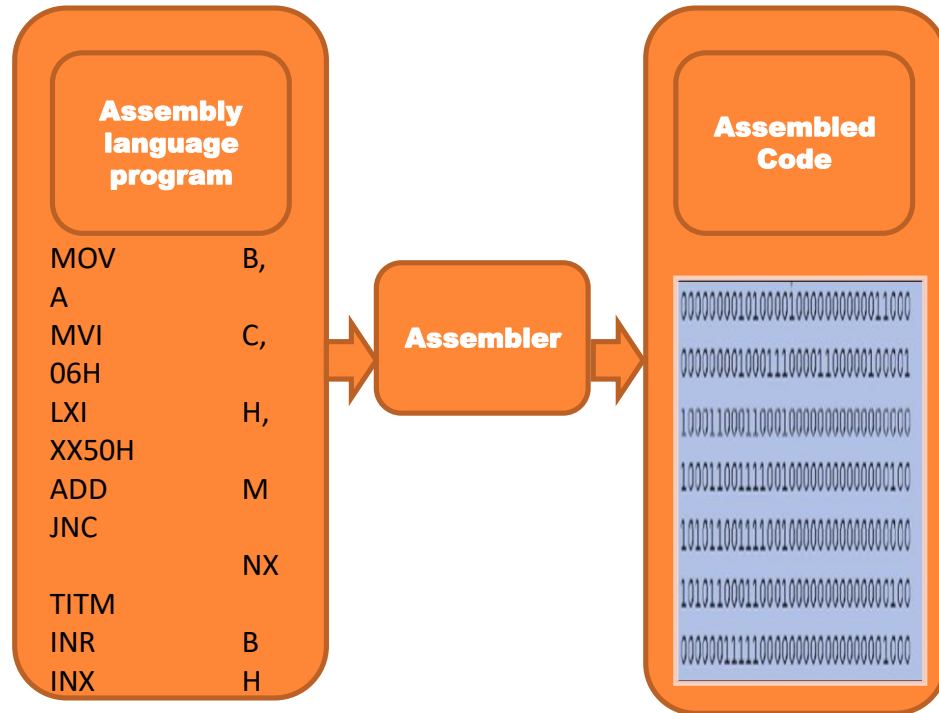
Translator Software (eg for system software)

- Converts program in High-level language & Assembly language to Machine-level language
- Creates *object code*
- Three kinds:
 - Assembler
 - Compiler
 - Interpreter
- Assembler converts Assembly language program to Machine language
- Compiler & Interpreter converts High-level language program to Machine language



Assembler (Translator Software 1)

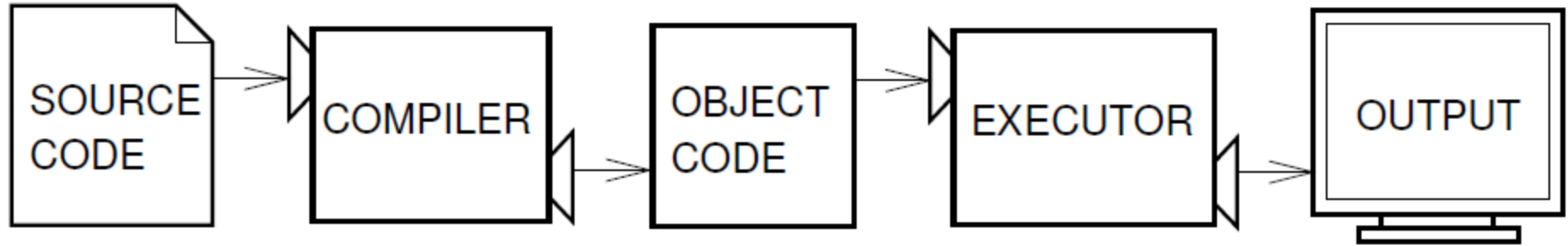
- Symbolic representation of Machine code
- Converts program written in Assembly language into Machine code
- Has one-to-one correspondence between simple Assembly statements and Machine language instructions



Compiler(Translator Software 2)

- Translator program (software) that translates high level language program into object code or executable.
- A program written in high level language is known as **source code**.

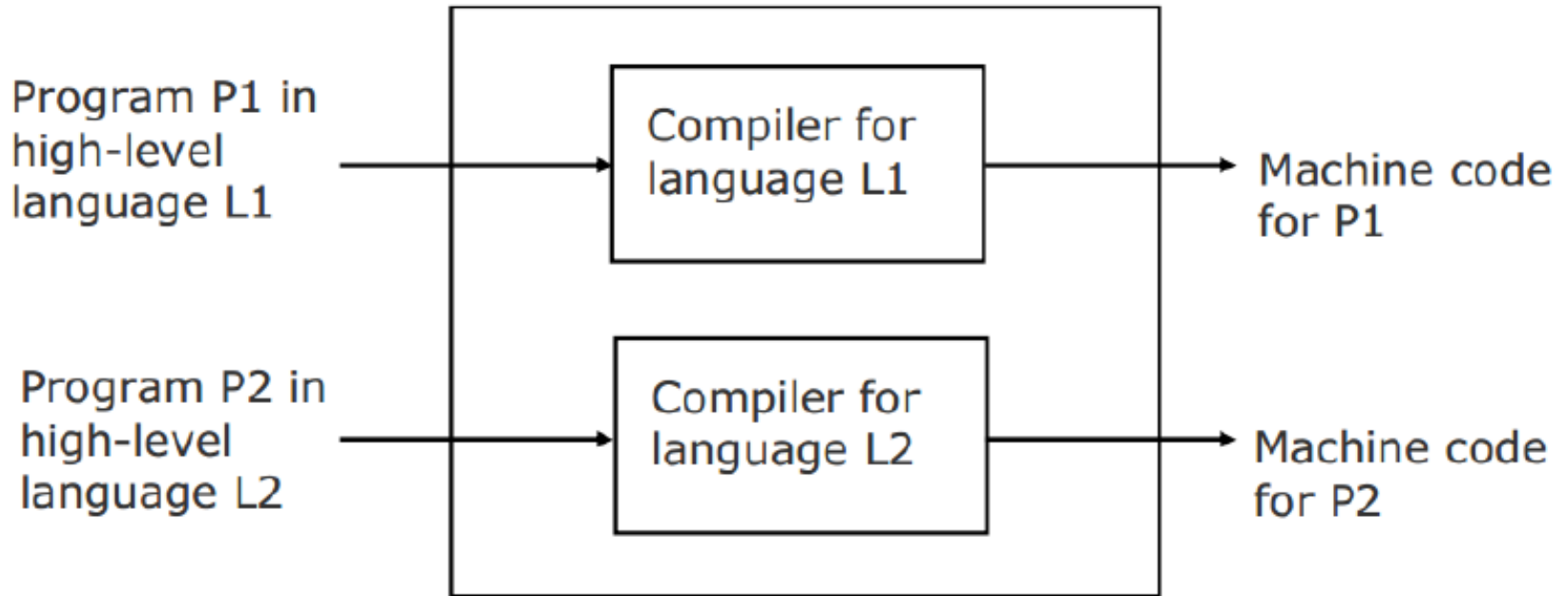
Compiler



Compiler

- Every language (eg: Python, C, C++, JAVA, etc) has its own compiler
- During compilation, it brings out any errors related to the syntax of the language.
- Logical errors cannot be detected by the compiler.

Compiler



A computer supporting languages L1 and L2

Illustrating the requirement of a separate compiler for each high-level language supported by a computer

Interpreter (Translator Software 3)

- Purpose similar to Compiler.
- Functions in a different manner than Compiler
- Performs line-by-line execution of source code during program execution
- Example – BASIC, Python

Interpreter

- It is a high level language translator
- Takes one statement of a program, translates it into machine language and executes it.
- **Difference**
 - Compiler translates the entire source program in to machine language and is not involved in execution.

Interpreter



Comparison: Compiler vs. Interpreter

| Compiler | Interpreter |
|---|---|
| Looks at entire source code | Looks at a source code line-by-line |
| Entire source code converted into object code | Line-by-line conversion |
| Object code executed multiple times | For each execution, source code interpreted and then executed |
| During execution, Source Code and Compiler not needed | Source code and Interpreter needed during execution |
| Compiled programs execute faster | Programs execute slower |

Algorithms & Flow charts

Algorithm

- Algorithm is a finite sequence of steps to solve a problem.
- Algorithm is an English-like representation of the logic which is used to solve the problem.
- It is a step- by-step procedure for solving a task or a problem.
- **Characteristics of good algorithm:**
 1. An algorithm should be finite.
 2. Steps in an algorithm should be unambiguous.
 3. Algorithm should be precise.
 4. Algorithm must have 0 or more inputs and 1 or more output.
- Best Algorithm can be selected depending on the **time and space complexity**.

Control Structures

- Control structures specify the statements to be executed and the order of execution of statements.
- Three types
 - **Sequential**— instructions are executed in linear order
 - **Selection (branch or conditional)**—it asks a true/false question and then selects the next instruction based on the answer.
 - **Iterative (loop)**—it repeats the execution of a block of instructions.

Sequential Algorithm

Find the sum of two numbers

Step 1 : start

Step 2 : input first number as a

Step 3 : input second number as b

Step 4 : add a and b and assign to a variable sum

Step 5 : print sum

Step 6 : stop

Algorithm Involve Decision Making

The larger of two numbers

Step 1 : Start

Step 2 : input first number as a

Step 3 : input second number as b

Start 4 : if $a > b$ then

Start 4.1 : print a Goto step 5

Start 4 .2: else print b

Start 5: Stop

Algorithm Involve **Repetition**

To print “hello” 5 times

Step 1 : Start

Step 2 : input string

Step 3 : Initialize $i=0$.

Step 4: **repeat following until $i \geq 5$**

Step4.1: print hello

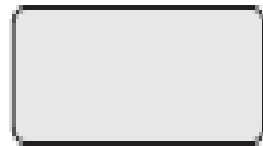
step4.2: set $i=i+1$

[End of loop]

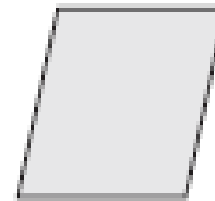
Step5:stop

Flowchart

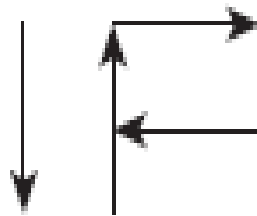
- Flow chart is the pictorial representation of an algorithm .
- Processes are represented in **boxes** and flow of control is represented by **arrows**.
- Flowcharts are usually drawn using some **standard symbols**.
- **Advantages:**
 1. Better way of communication.
 2. Proper documentation.
 3. Efficient coding.
 4. Proper Debugging.
- **Disadvantages:**
 1. Complex logic makes flowchart become complex and clumsy.
 2. Alteration or modification require redrawing of flowchart



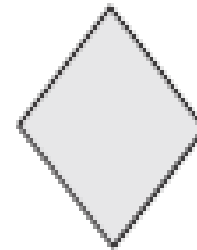
Start or end
symbol



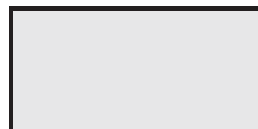
Input/Output
symbol



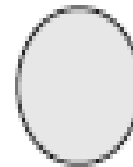
Arrows



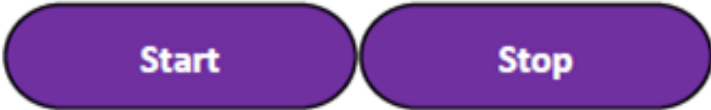
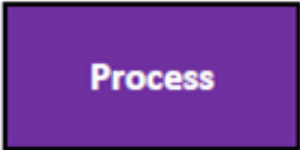
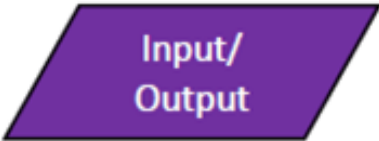

Decision symbol



Processing step



Connector

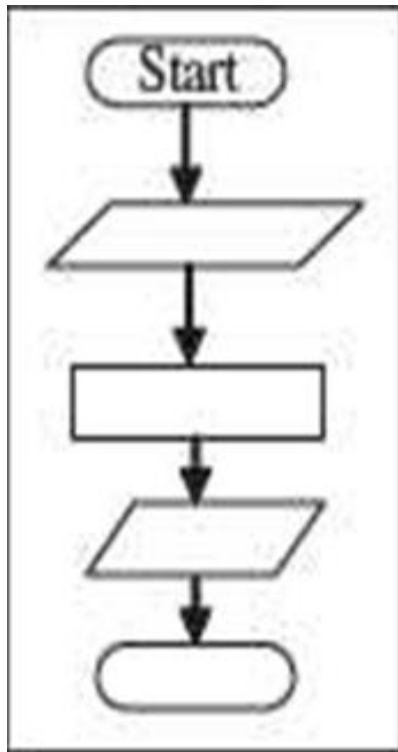
| Symbol | When it is used: |
|--|--|
|  | <p>This shapes indicates the start or end of a flowchart</p> |
|  | <p>A rectangular box represents a process, this is doing something. E.g. $\text{total} = \text{num1} + \text{num2}$</p> |
|  | <p>A parallelogram represents input or output. E.g. Input num1 Output total</p> |
|  | <p>A diamonde shape represents a decision, YES or NO e.g. is it a weekday?</p> |

Connector

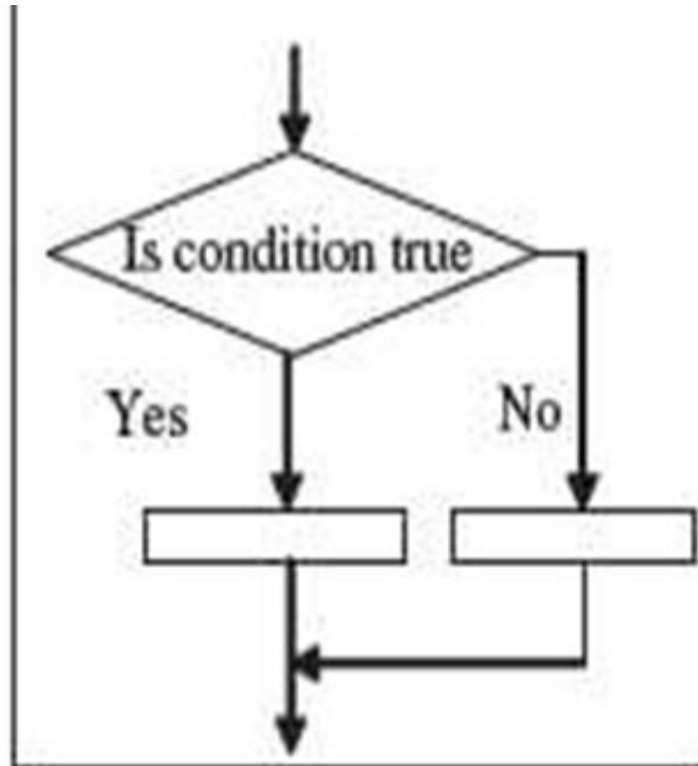
- A circle is used to join the different parts of a flow chart.
- The use of connectors gives a neat appearance to a flow chart.
- They are necessary if a flow chart extends over more than one page and the different parts are to be joined together.



Flow chart for control structures

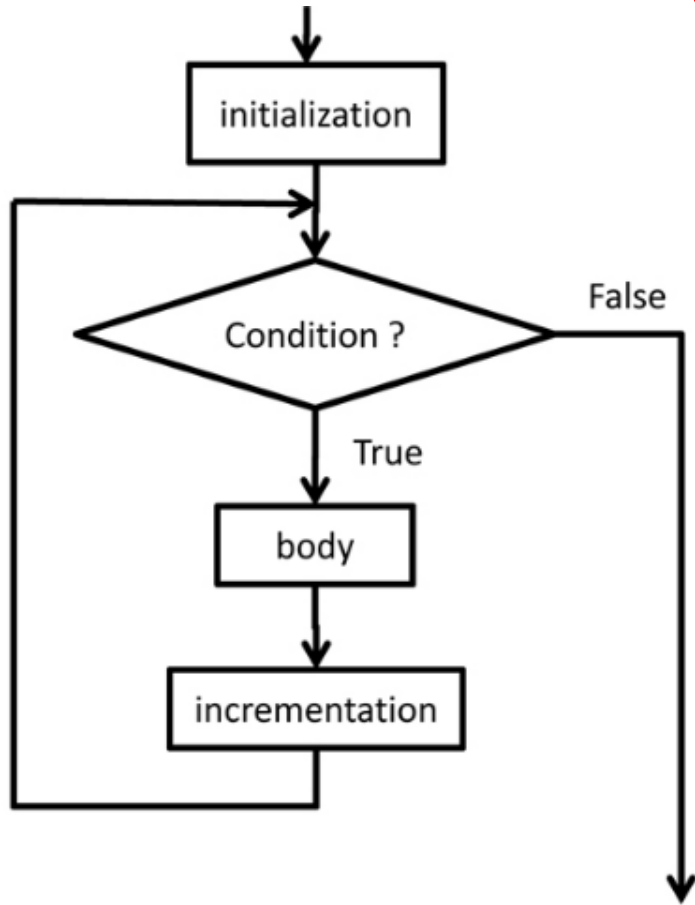


Sequence

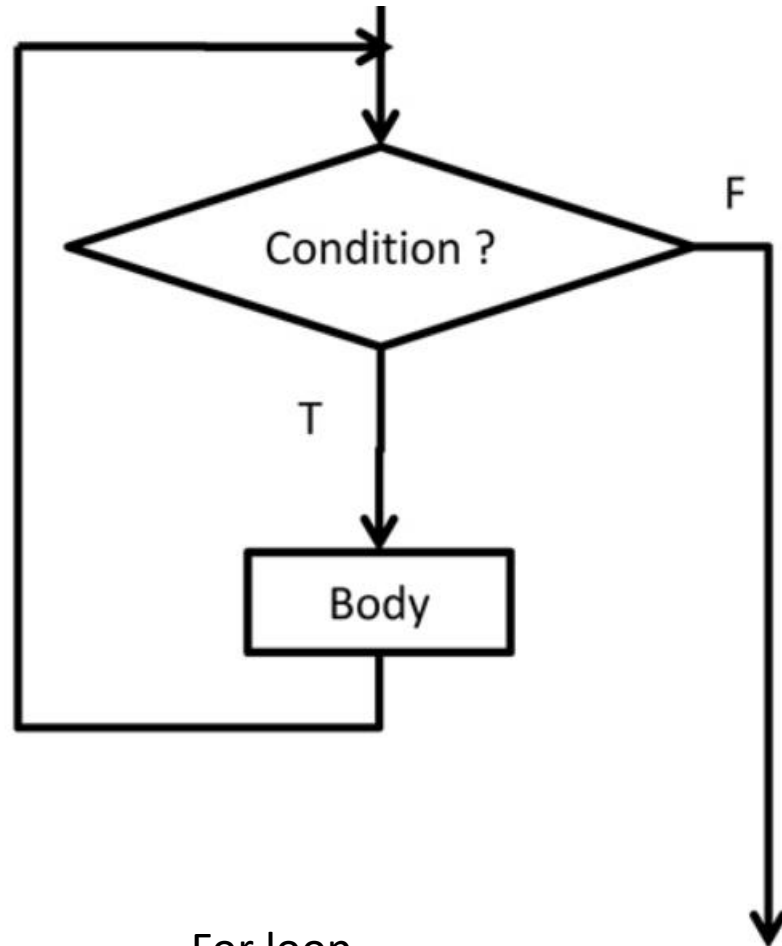


Selection

Flow chart for control structures (iteration)



While loop



For loop

Sequential flowchart Example: Draw a flow chart to add two numbers.

Step 1 : start

Step 2 : input first number as A

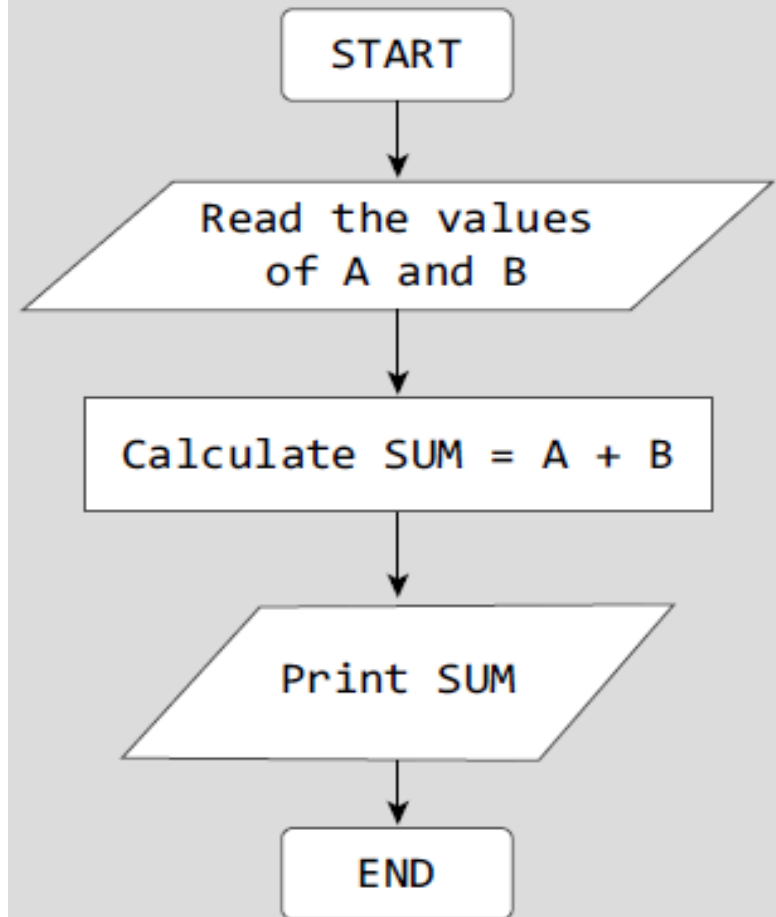
Step 3 : input second number as B

Step 4 : add A and B and assign to a variable sum

Step 5 : print sum

Step 6 : stop

Solution



Selection (branch or conditional)

- find the largest of two numbers

Step 1: Start

Step 2: Read two numbers a & b.

Step 3: If $a > b$ then

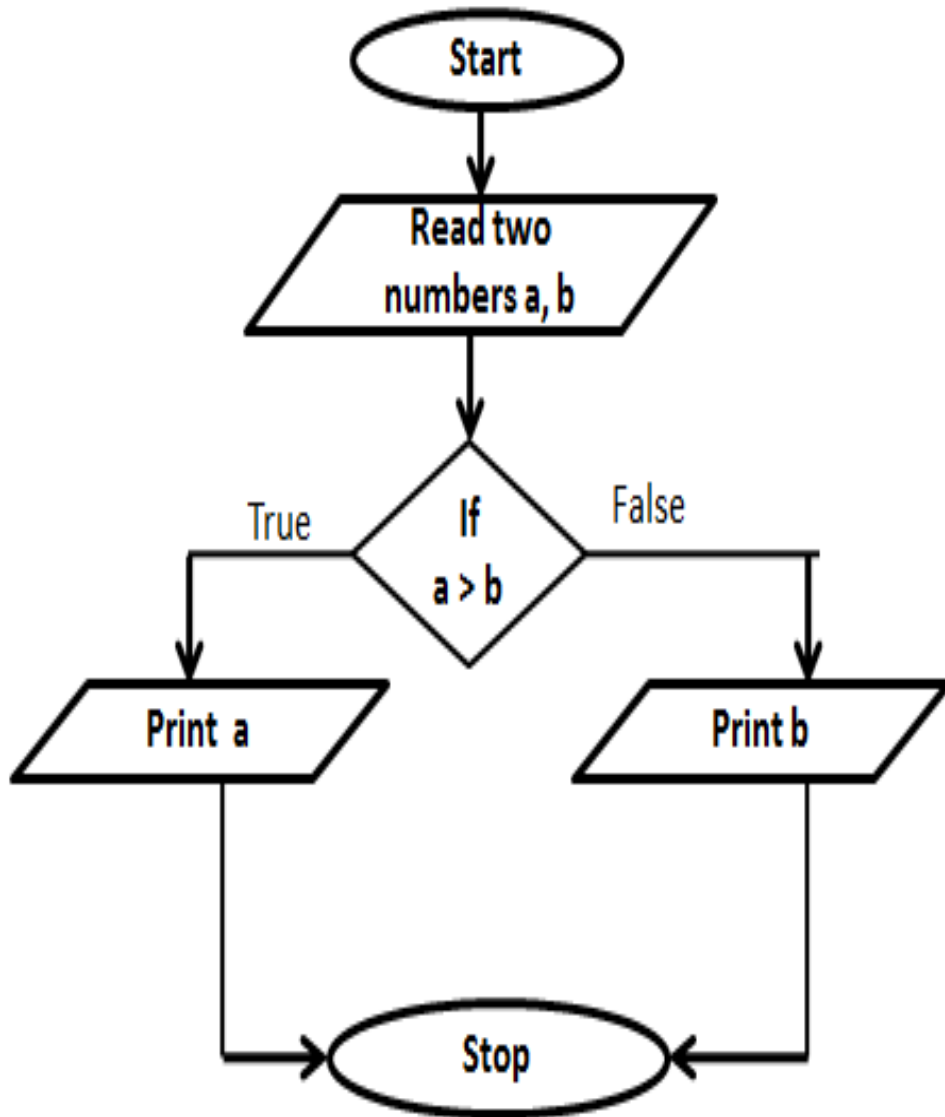
Step 3.1 : Print largest number is a

Step 3.2 : Go to step 5

Step 4: else

Step 4.1 : Print largest number is b

Step 5: Stop



Largest among three numbers

- Let the numbers be 5,4,6

First check $5 > 4$ is true then check $5 > 6$. is true??

No So print 6 is largest

- Let the numbers be 2,3,4

Check $2 > 3$ is true ??? NO

So check $3 > 4$ is true ??? No

So print 4 is largest.

- If a,b,c are the numbers we will check

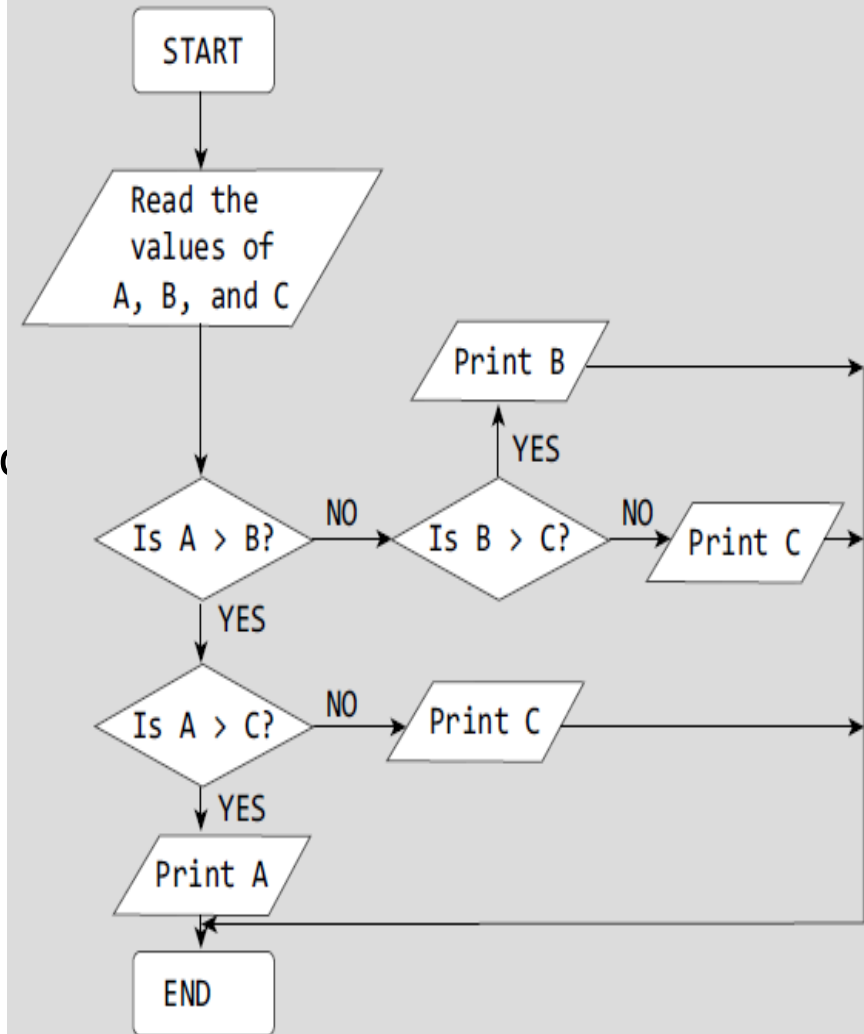
1) Check $a > b$.If it is true check $a > c$

2) If 1 is not true check $b > c$. Is true print b is largest else print c is largest.

Algorithm - Largest of 3 numbers

1. Start.
2. Read three numbers to a,b,c
3. If $a > b$ then
 - 3.1 . If $a > c$ then
 - 3.1.1 Print the largest number is a
 - 3.1.2 Go to step 6
 - 3.2 else
 - 3.2.1. Print the largest number is c
 - 3.2.2. Go to step 6
4. else if $b > c$ then
 - 4.1 Print the largest number is b
 - 4.2 Go to step 6
- 5 .else
Print the largest number is c
6. Stop

Solution



Display the grade of a student .

step 1. Start

step 2: Read the marks of 3 subjects to mark1,mark2,mark3.

step3: Find the total mark by adding the 3marks and assign to a variable **sum**.

step 4: Calculate $\text{average} = \text{sum}/3$.

step 5: If $\text{average} \geq 80$ and $\text{average} \leq 100$

step 5.1 print grade='A' Goto 10.

step 5.2 Else goto step 6

Step6: If $\text{average} \geq 61$ and $\text{average} \leq 80$

step 6.1 print grade='B' . Goto 10.

step 6.2 Else goto step 7

step 7 : If $\text{average} \geq 41$ and $\text{average} \leq 60$

step 7.1 print grade=C . Goto 10.

step 7.2 Else goto step 8

step 8 . If $\text{average} \geq 0$ and $\text{average} \leq 40$

step 8.1 print grade= F . Goto 10.

step 8.2 Else goto step 9

STEP 9: Print strange grade.

STEP10. Stop.

Example for Iterative Algorithm

- Print numbers from 1 - 10

Step 1. Start

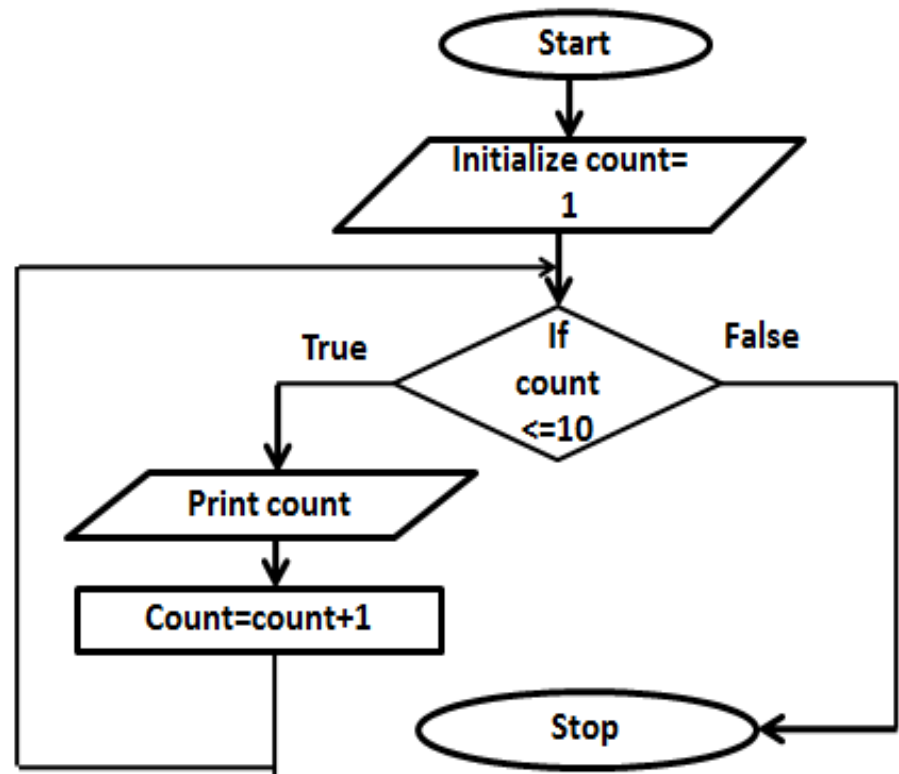
Step 2. Initialize a variable ,count with 1.

Step 3. Repeat the following untill count \leq 10 .

3.1 : print count

3.2 : set count=count +1

Step 4. Stop



Algorithm Involve **Repetition**

Sum of n natural numbers

step1: Start

step 2: Read the limit assign in
to variable **n**

step 3: Assign $\text{sum}=0, i=1$

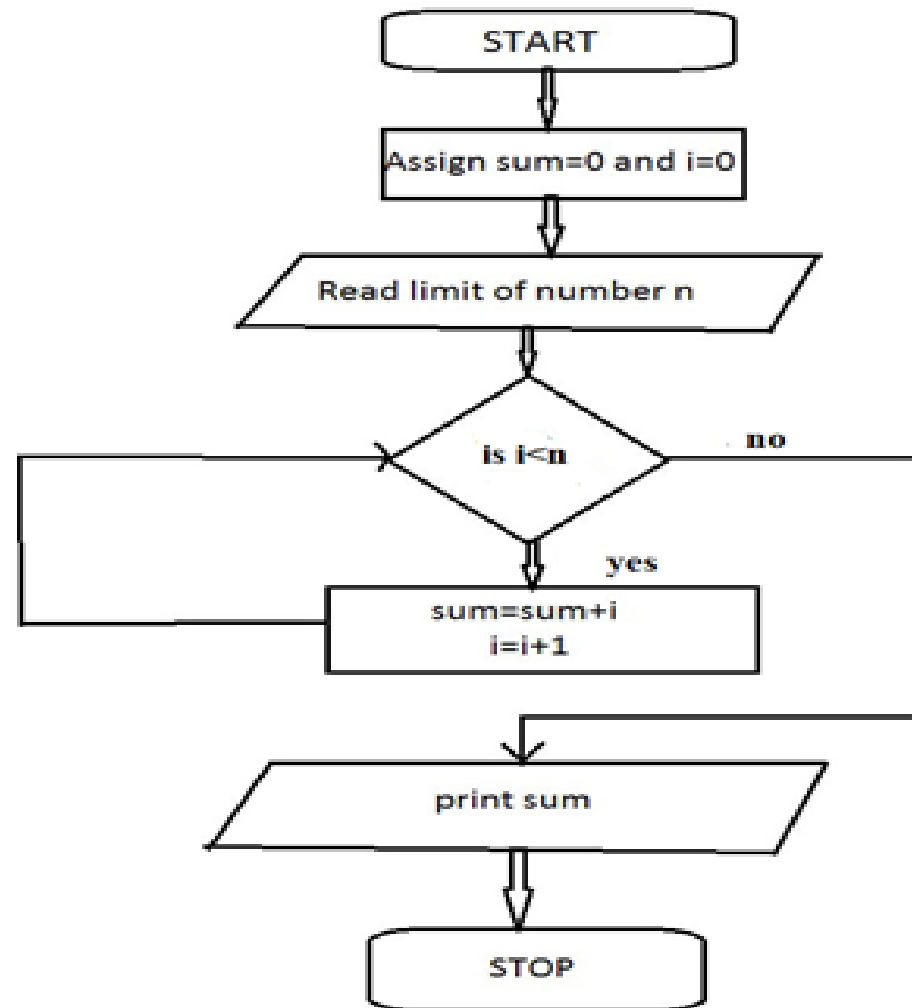
step4: Repeat the following until
 $i > n$

Step 4.1 :Find $\text{Sum}=\text{Sum} + i$

Step 4.2: $i=i+1$

step5. Print sum.

step6. Stop



Selection (branch or conditional)

- Prepare algorithm & Flow chart for the following
 1. Given number is odd or even
 2. Given number is positive or negative.
 3. Largest among two numbers
 4. Largest among three numbers
 5. Smallest among two numbers.
 6. Roots of a quadratic equation (b^2-4ac)
 7. Check whether a number say m is divisible by another number say n

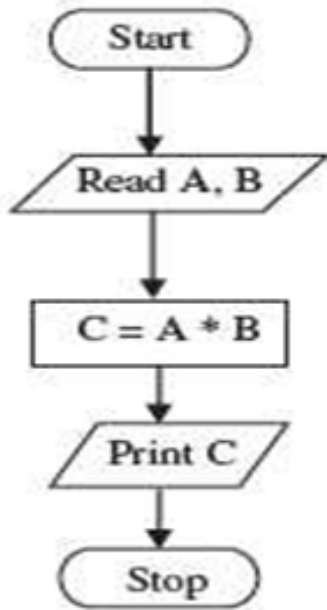
Prepare algorithm &Flowchart for the following

1. Calculate simple interest
2. Calculate the volume and surface area of cube.
3. Calculate the volume of a sphere.
4. Calculate net salary of an employee by reading his name, age, empid, basic pay, DA- 50% of BP

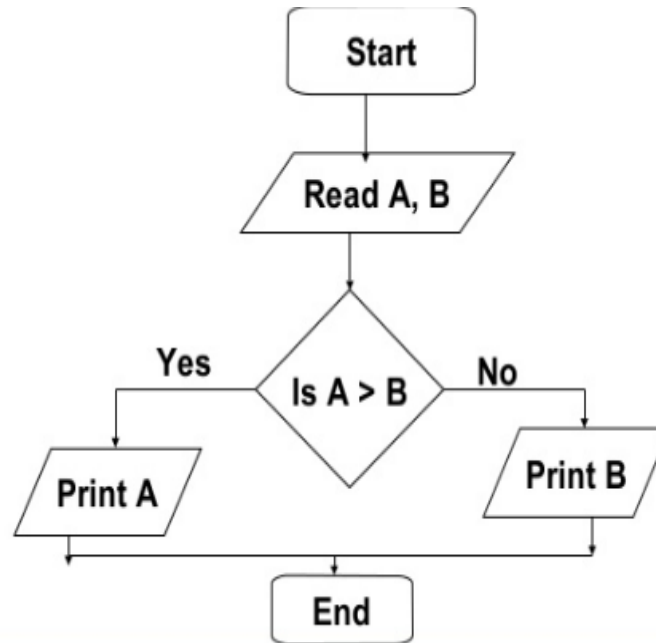
HRA= 30% of BP, Net sal= DA+HRA+BP

Example for sequence, control & iteration problem

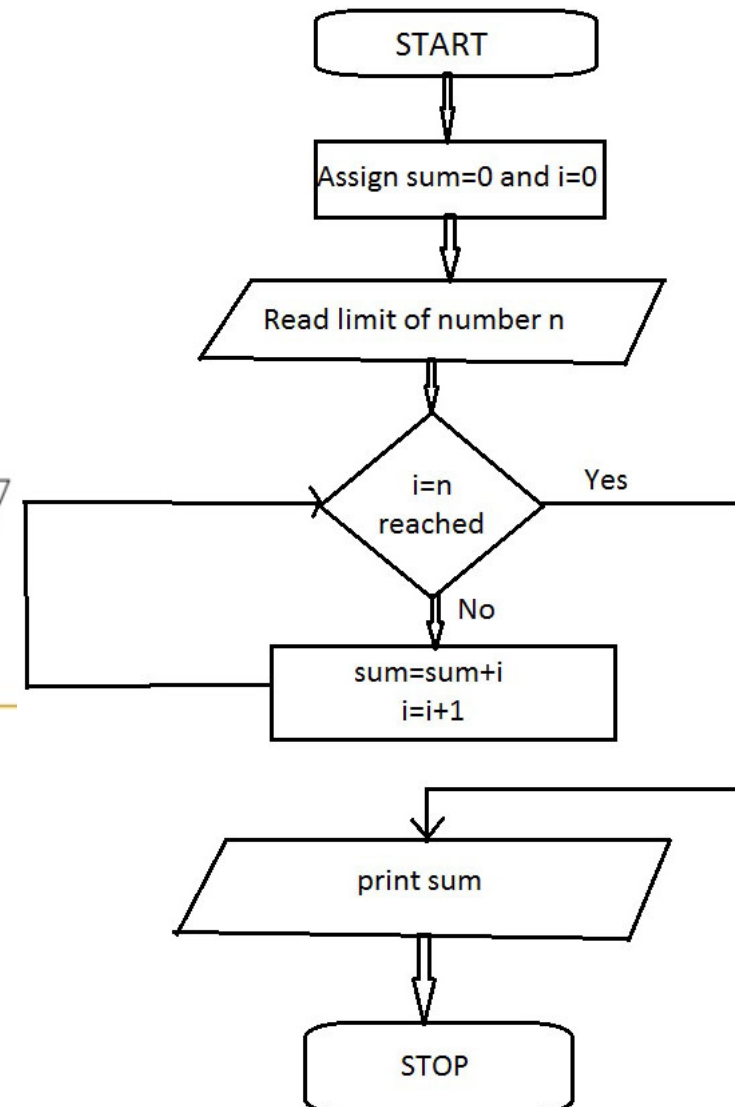
Sum of two numbers



Largest of 2 numbers



Sum of n numbers



Difference between Algorithm, and Flowchart

| Algorithm | <ul style="list-style-type: none">•An algorithm is a sequence of instructions used to solve a particular problem.•An algorithm can be represented using a flowchart or a pseudo code•Structural concept |
|-----------|---|
| Flowchart | <ul style="list-style-type: none">•Flowchart is a graphical representation of the algorithm.•Tool to represent an algorithm |

Display all palindrome numbers in a range

Step1: 1. Start

step 2: Read the limit1

step 3: Read the limit2

step 4.Repeat the following until **i** in specified range limit 1 and limit2

step 4.1 Assign num=i, rev=0

step 4.2 Repeat the following until num>0

Step 4.2.1 $rem = num \% 10$

Step 4.2.1 $rev = rev * 10 + rem$

Step 4.2.1 $num = num // 10$

step 4.3 check whether **i == rev:**

step 4.3.1 Print the rev

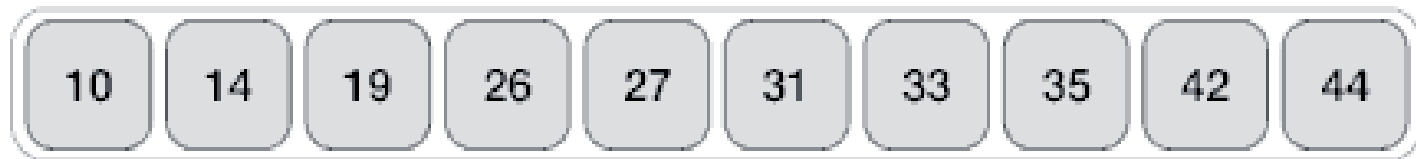
step5.stop

Linear search

- In this type of search, a sequential search is made over all items one by one.
- Every item is checked and if a match is found then that particular item is returned
- otherwise the search continues till the end of the data collection.

Linear search

Linear Search



=
33

algorithm

step1:start

Step 2: initialize i as 0

Step 3: repeat untill $i < n$

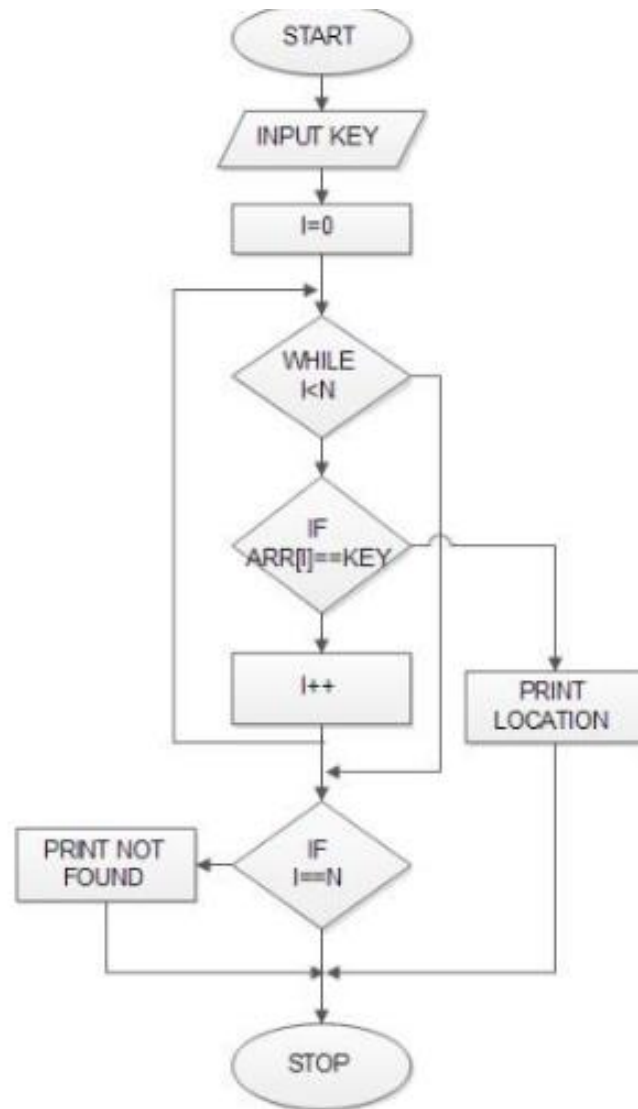
Step 3.1: if current element == x then go to step 6

step3.2 :Print Element x Found at location i and
go to step 5

Step 3.3: Set i to $i + 1$

Step 4: Print element not found

Step 5: stop



Bubble Sort

We take an unsorted array for our example. Bubble sort takes $O(n^2)$ time so we're keeping it short and precise.



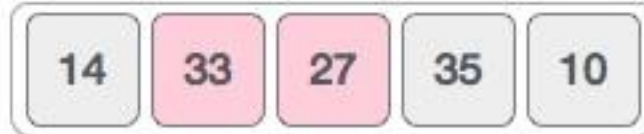
Bubble sort starts with very first two elements, comparing them to check which one is greater.



In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.



We find that 27 is smaller than 33 and these two values must be swapped.



The new array should look like this –



Next we compare 33 and 35. We find that both are in already sorted positions.



Then we move to the next two values, 35 and 10.



We know then that 10 is smaller 35. Hence they are not sorted.



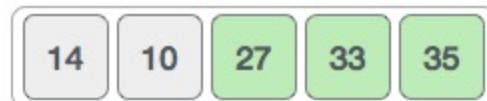
We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this –



To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this –



Notice that after each iteration, at least one value moves at the end.



And when there's no swap required, bubble sorts learns that an array is completely sorted.

10

14

27

33

35

Now we should look into some practical aspects of bubble sort.

Algorithm

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

```
begin BubbleSort(list)

  for all elements of list
    if list[i] > list[i+1]
      swap(list[i], list[i+1])
    end if
  end for

  return list

end BubbleSort
```

