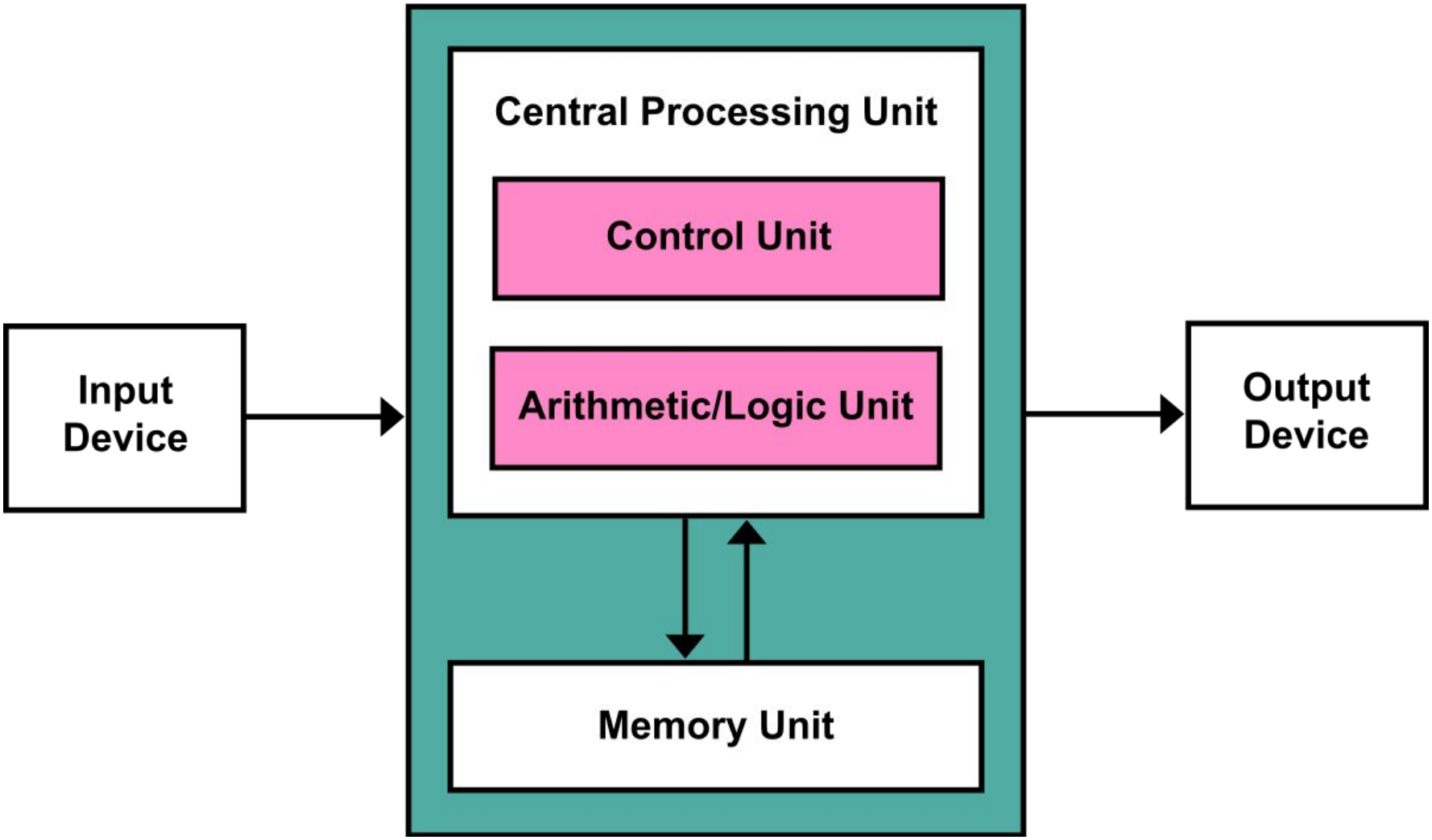
BASIC COMPUTER ORGANIZATION AND DESIGN

Von Neumann Architecture/ Stored Program Organization



Bus

- Special unit use to exchange data between the one part of the computer to the other at a time.
- The direction of signal can be unidirectional or bidirectional depending on the type of bus and type of computer.
- The bus size is determined by amount of information that can be transferred at a time known **as its width**.

BUS

- This amount, expresses in bits, corresponds to the number of physical lines over which data is sent simultaneously.
- The bus speed is defined by its frequency and measured in Mhz.
- The number of data packets sent or received per second.Each time that data is sent or received is called a Cycle.

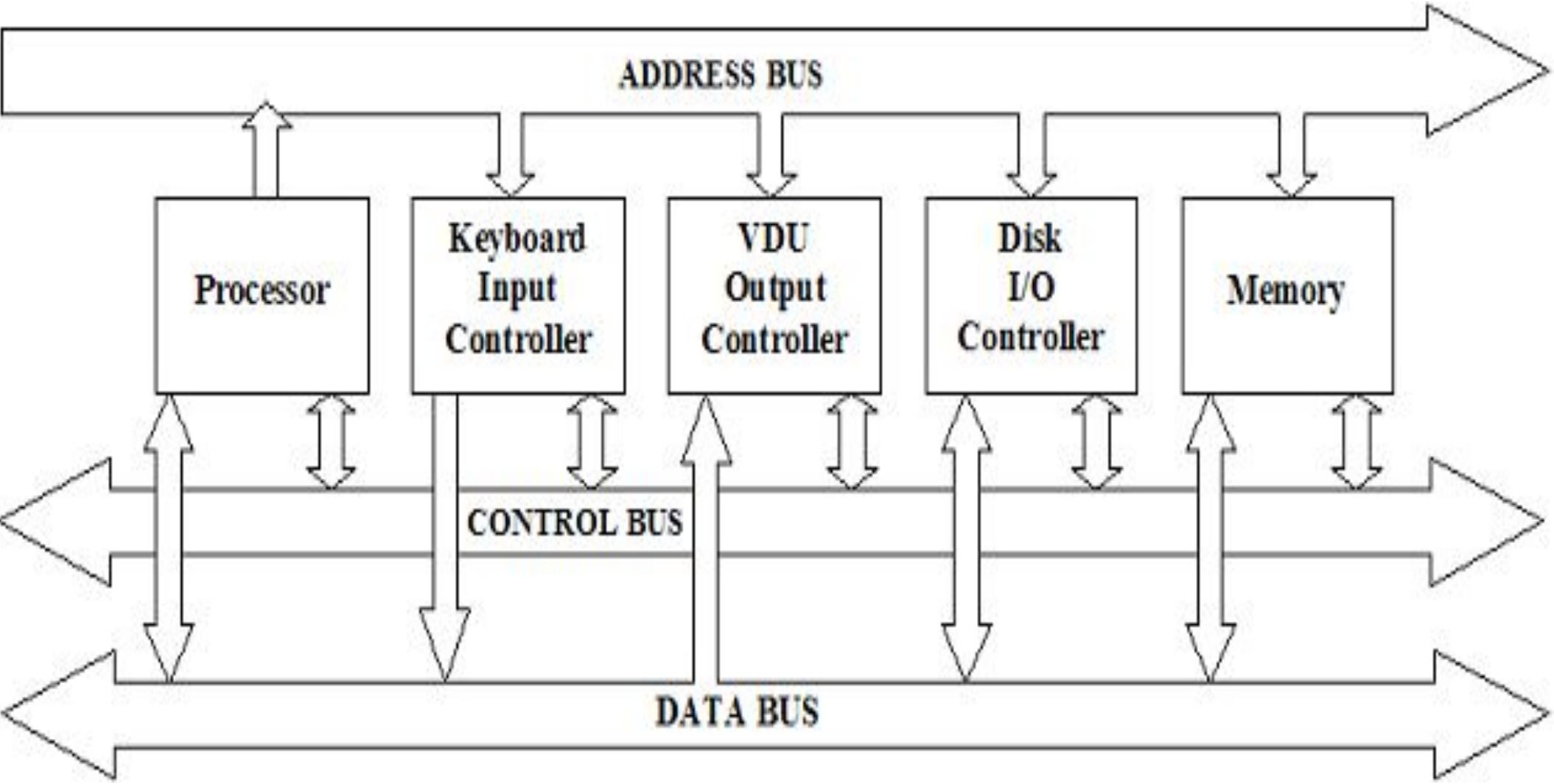
Types of Bus

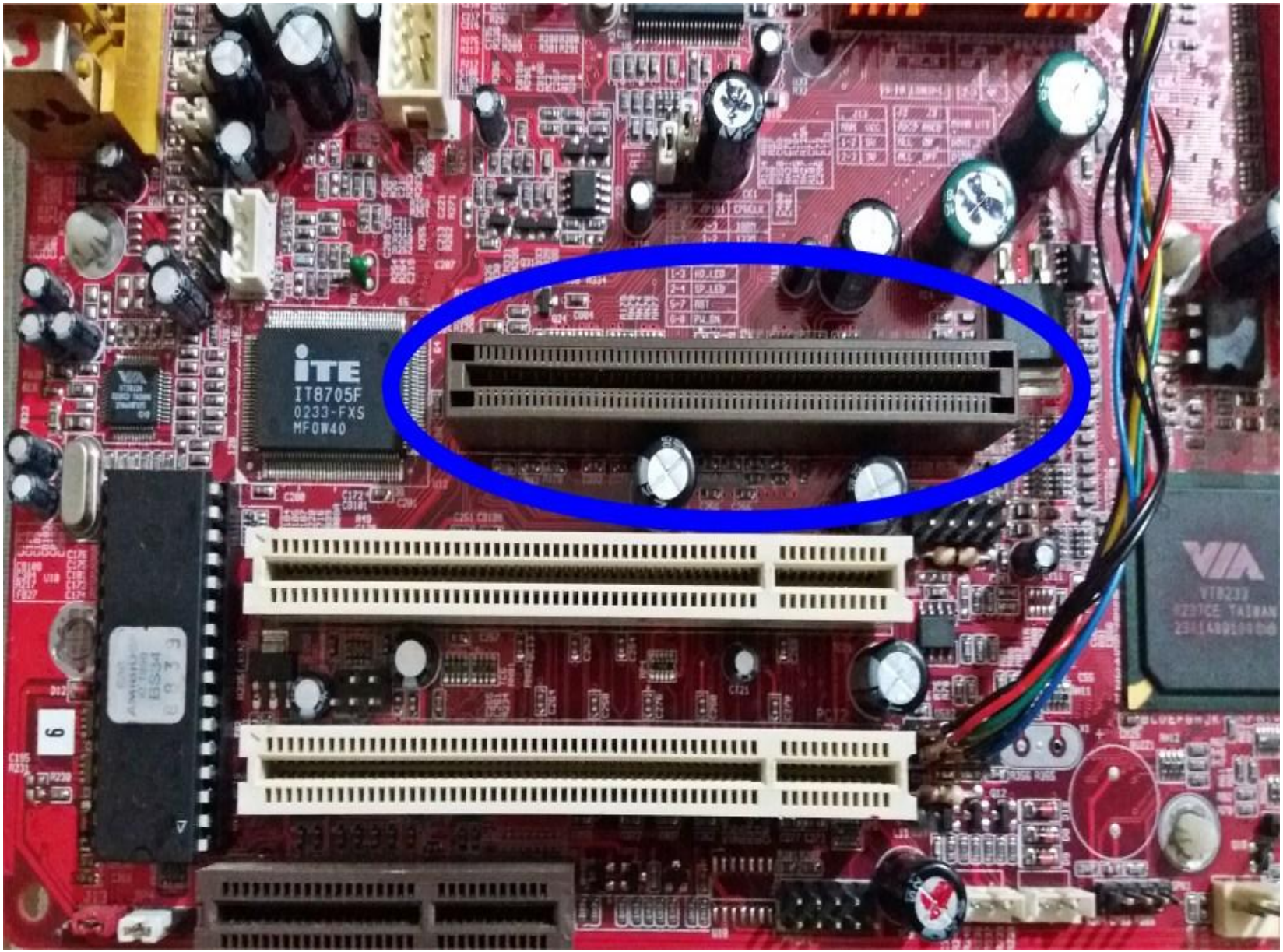
- **Control Bus** – Used by the CPU to direct and monitor the actions of the other functional areas of the computer, transmit variety of individual signals necessary to control and manage the operations of the computer.
- It is used to transmit variety of signals like read , write ,acknowledge etc.

Types of Bus

- **Data Bus** – Use to handle the transfer of all data and instruction between different parts of computer. It is bidirectional and sometimes called memory bus.
- **Address Bus** – Used to transport memory addresses which the CPU want to access in order to read and write data.It is unidirectional bus. Before data or instructions can be written into or read from memory bye the CPU or I/O sections, an address must be transmitted to memory over the address bus.

Types of Bus


ADDRESS BUS
Processor
Keyboard Input Controller
VDU Output Controller
Disk I/O Controller
Memory
CONTROL BUS
DATA BUS


iTE
IT8705F
0233-FXS
MF0W40
VIA
VT8233

Register

- Special memory unit used for storing data on temporary basis.
- Its not part of the CPU.
- Used for quick and direct accessibility of small amount of data for processing.
- Registers can contain the address of memory location where data is stored rather than the actual data itself.

Register Organization

- All the instructions are interpreted and executed by the CPU and **data transfer between the various units of computer system done through special memory called Registers. It is used to stored data on temporary basis.**
- Register is a **temporary storage unit for quick, direct accessibility of a small amount of data for processing.**
- Registers are identified by the number of bits they can store. For example 8 bit register or 32 bit register.

Register Organization

Accumulator (A):

- Used to accumulate the results of a large number of instructions.
- **8 bit register**
- Most versatile register.

R registers:

- Set of **eights registers** that are named R0,R1,R2,R3,R4,R5,R6.R7
- Used as auxiliary registers.
- Ex: ADD A,R4

Register Organization

- **Program Counter (PC):**
- A program counter is a special register in a computer processor that contains **the address (location) of the next instruction to be executed.**
- As each instruction gets fetched, the program counter increases its stored value by 1.
- Since some instructions require 2 or 3 bytes the PC will be incremented by 2 or 3 in these cases.
- there is no way to directly modify its value.

Register Organization

Stack Pointer (SP):

- It is a special purpose register which holds the address of the top of the stack.
- When a value push (insert) onto the stack, the processor firsts increments the value of SP and then stores the value at the resulting memory location.
- When a value is popped off the stack, the processor returns the value from the memory location indicated by SP and then decrements the value of SP.

Register Organization

Memory Address Register (MAR):

- It holds the address of active memory location.
- MAR holds the memory location of data that needs to be accessed.

Memory Buffer Register (MBR):

- It stores the contents of memory word read from or written in memory.
- An instruction word placed on this register is transferred to the instruction register.
- A word to be stored in a memory location must first transfer to the MBR from where it is written in memory.

Register transfer language

- The symbolic notation used to describe the micro operation transfers among register is called a register transfer language.

- We call it "register transfer" language is because statements involve transferring and operating on data and loading it in some register

Symbols Used in RTL

- The following notation is often used when specifying RTL.

BASIC SYMBOLS:

- **R :** An R followed by a number refers to a register.

  For eg:

  R2 means the second register.

- **M :** refers to memory, with memory address written inside square braces.

  For eg:

  M[R2] refers to the contents of the memory address in R2.

  or

  M[42] refers to the contents of memory address 42.

Symbols Used in RTL

- **An arrow :** An arrow pointing to the left is used for a transfer of data.

  For eg:

  R6 <--- R1 stores the value of R1 into R6.

- **A comma :** represents simultaneous transfers.

  For eg:

  R1 <--- R3 , R2 <--- R4 means

  " store R3 into R1 and at the same time , store R4 into R2."

Mathematical and Logical Symbols

- A subscripted letter followed by a colon is a conditional. i.e.
  - **K_1 : R2 <--- R1** means

    " if K_1 is true , then store R1 into R2."

    (K_1 can refer to any Boolean function such as output of circuit.)

- Addition is indicated by the + sign.
  - **R1 <--- R2 + R3** means

    " Add R2 and R3 and store them into R1."

Mathematical and Logical Symbols

- Subtraction is handled not with the minus sign but with the compliment.
- A one's complement subtraction means
  - subtracting from 1 or
  - replacing 0 with 1 and 1 with 0.
- A two's complement subtraction is a 1's complement and addition of 1.
- Complement of R4 is R4'
- For eg:
  - R5 <--- R3 + R4' (means R3 is subtracted from R4)
  - R5 <--- R3 + R4' + 1 (two's complement)

Mathematical and Logical Symbols

- Addition of constant is also possible and Subtraction of constant does use minus sign.
- For eg:

  R9 <--- R3 - 1

Complex Logical Symbols

- Condition can have more than one test value.
  - If R5 will be stored in R4 only when both K_1 and K_2 are true , the RTL would be written as

    " $K_1 . K_2$: R4 <--- R5 "

  - If R5 were to be stored when either K_1 or K_2 were , a + sign would be used

    " $K_1 + K_2$: R4 <--- R5 "

Complex Logical Symbols

- ANDing and ORing of registers uses AND and OR symbols .
  - To store R3 AND R2 into R1 , the RTL would be written as

    " R1 <--- R3 ^ R2"
  - To store R3 OR R2 into R1 , the RTL would be written as

    " R1 <--- R3 v R2"

Complex Logical Symbols

- Shift left and shift right are unary operators.
- They only shift one place left or right as per instruction.
  - "R1 sr R1" is typical RTL shift right instruction.
  - "R1 sl R1" is typical RTL shift left instruction.

Complex Logical Symbols

- If-then-else is implemented with commas and multiple colons.
  - if K_1 , then store R4 into R6, else if K_2 , then store R5 into R6, else store R7 into R6 is written as:
    - **K_1 : R6 <--- R4 , K_1' K_2 : R6 <--- R5 ,K_1' K_2' : R6 <--- R5**

Instruction Format

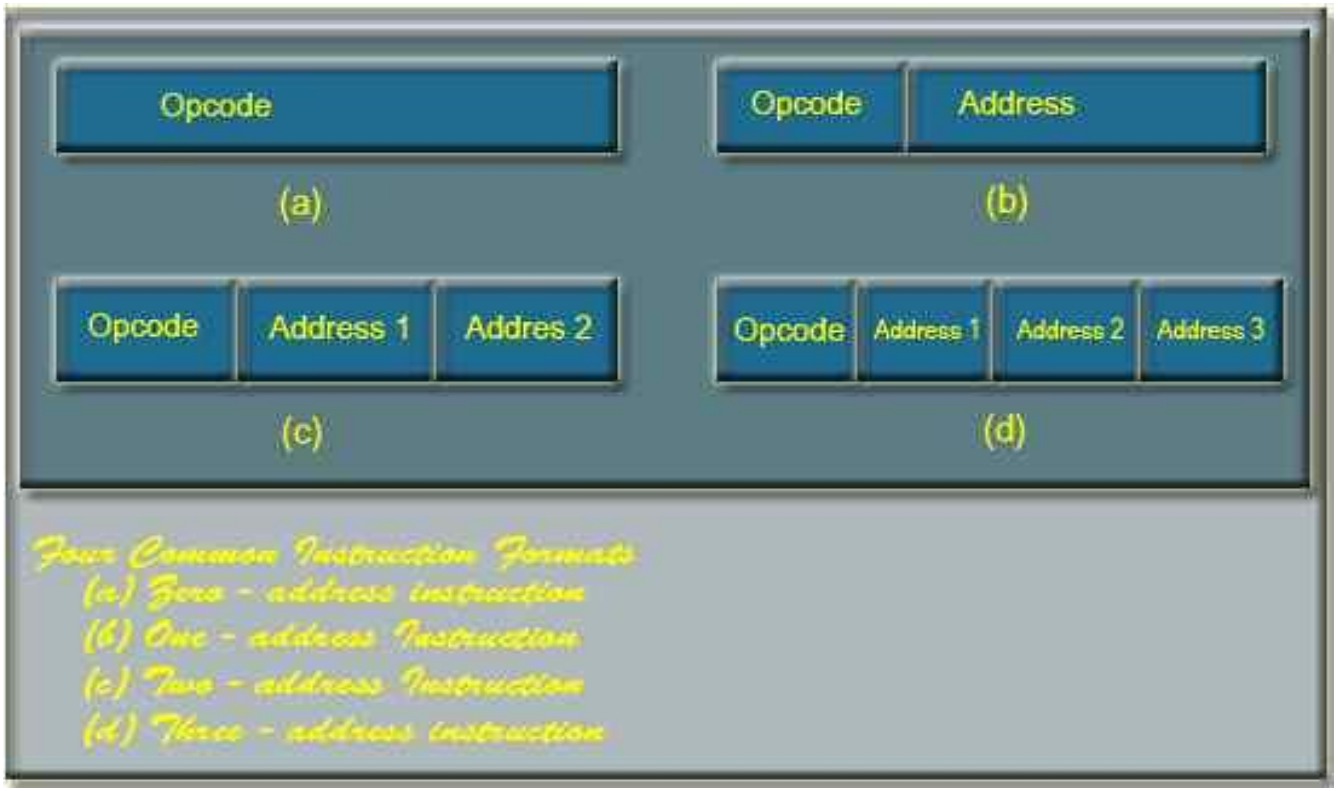
- An instruction is a command to the microprocessor to perform a given task on a specified data.
- Each instruction has two parts.

  A. opcode (Task to be performed,opcode)

  B. operand (Data to be operated on)
- The operand can be 8/16 bit data, an internal register, a memory location or 8/16 bit address.

Instruction Format

- An instruction format defines the layout of the bits of an instruction.



Zero Address / Zero Word Instruction

- Zero Address / Zero Word Instruction:
- A stack based computer do not use address field in instruction.
- Stack is used,
- Arithmetic operation pops two operands from the stack and pushes the result.
- Also called stack organization.

Zero Address / Zero Word Instruction

- Example: X= (A + B) – (C + D)
- PUSH A (tos <— A)
- PUSH B (tos <— B)
- ADD (tos <— A + B)
- PUSH C (tos <— C)
- PUSH D (tos <— D)
- ADD (tos <— C + D)
- SUB (tos <— (A +B) – (C+ D))
- POP X M[X] <— TOS

One Address / One Word Instruction

- One Address / one Word Instruction:
- One address can be a register name or memory address.
- Single accumulator organization.
- It uses AC register for all data manipulation.
- Instruction: ADD X
- Microoperation: AC <— AC + M[X]

One Address / One Word Instruction

- Example: X= (A + B) - (C + D)
- LOAD A (AC <— M[A])
- ADD B (AC <— AC + M[B])
- STORE T (M[T] <— AC)
- LOAD C (AC <— M[C])
- ADD D (AC <— AC + M[D])
- SUB T (AC <— AC - M[T])
- STORE X (M[X] <— AC)

Two Address / Two Word Instruction

- Two Address / two Word Instruction:
- Two address registers or two memory locations are specified.
- Assume that the destination address is same as that of the first operand
- Instruction: ADD R1, R2
- Micro operation: R1 <— R1 + R2

Two Address / Two Word Instruction

- Example: X= (A + B) – (C + D)
- MOV R1, A (R1 <— M[A])
- ADD R1,B (R1 <— R1+ M[B])
- MOV R2,C (R2 <— M[C])
- ADD R2,D (R2 <— R2 + M[D])
- SUB R1,R2 (R1 <— R1 - R2)
- MOV X,R1 (M[X] <— R1)

Three Address / Three Word Instruction

- Three Address / three Word Instruction:
- memory addresses for the two operands and one destination needs to be specified.
- It is also called as a general register organization.
- Instruction: ADD R1, R2, R3
- Micro operation: R1 <— R2 + R3

Three Address / Three Word Instruction

- Example: X= (A + B) – (C + D)
- ADD R1, A, B (R1 <— M[A]+ M[B])
- ADD R2, C,D (R2 <— M[C] + M[D])
- SUB X, R1,R2 (M[X] <— R1 - R2)

Micro-operations

- Register Transfer Micro-operation
- Arithmetic Micro-operation
- Logic Micro-operation
- Shift Micro-operation

Arithmetic Micro - operation

Arithmetic Microoperations

$R3 \leftarrow R1 + R2$	Addition
$R3 \leftarrow R1 - R2$	Subtraction
$R2 \leftarrow \overline{R2}$	1's Complement
$R2 \leftarrow \overline{R2} + 1$	2's Complement
$R3 \leftarrow R1 + \overline{R2} + 1$	Subtraction!
$R1 \leftarrow R1 + 1$	Increment
$R1 \leftarrow R1 - 1$	Decrement

Logical Micro - operation

operation.

Sr. No.	RTL Notation	Explanation
1.	R3 ← R1 v R2	Symbol v stands for logical OR here logical OR operation is performed between bit value of register R1 and R2 and result store in R3
2.	R3 ← R1 ^ R2	Symbol ^ stands for logical AND here logical AND operation is performed between bit value of register R1 and R2 and result store in R3
3.	R3 ← R1 ⊕ R2	Symbol ⊕ stands for logical XOR(exclusive OR) here logical XOR operation is performed between bit value of register R1 and R2 and result store in R3

Shift Micro-operation

- Logical shift micro operation
- Arithmetic shift micro operation
- Circular shift micro operation

INTERRUPTS

- An **interrupt** is a signal sent to the processor that **interrupts** the current process.
- It may be generated by a hardware device or a software program.
- A hardware **interrupt** is often created by an input device such as a mouse or keyboard.
- An **interrupt** is sent to the processor as an **interrupt** request,

TYPES OF INTERRUPTS

- Although interrupts have highest priority than other signals, there are many type of interrupts but basic type of interrupts are :

1. Hardware interrupts

2. S/W Interrupts

TYPES OF INTERRUPTS

Hardware interrupts can be classified into two types:

Maskable Interrupt: The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor.

Non Maskable Interrupt: The hardware which cannot be delayed and should process by the processor immediately.

TYPES OF INTERRUPTS

Hardware Interrupts: If the signal for the processor is from external device or hardware is called hardware interrupts.

- **Example:** From keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts.

TYPES OF INTERRUPTS

Software Interrupts: Software interrupt can also divided in to two types they are :

- Normal Interrupts: the interrupts which are caused by the software instructions are called software instructions.
- Exception: unplanned interrupts while executing a program is called Exception.

For example: while executing a program if we got a value which should be divided by zero is called a exception.