

## Introduction

### Computer Architecture

Computer Architecture is a blueprint for design and implementation of a computer system. It provides the functional details and behavior of a computer system and comes before computer organization. Computer architecture deals with 'What to do?'

### Computer Organization

Computer Organization is how operational parts of a computer system are linked together. It implements the provided computer architecture. Computer organization deals with 'How to do?'

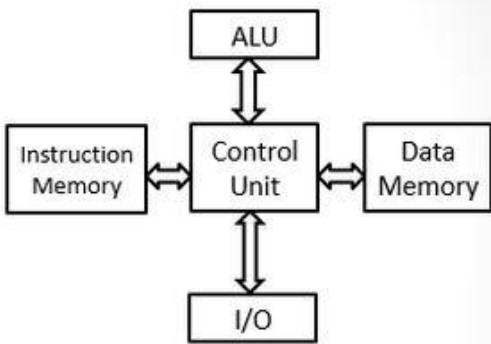
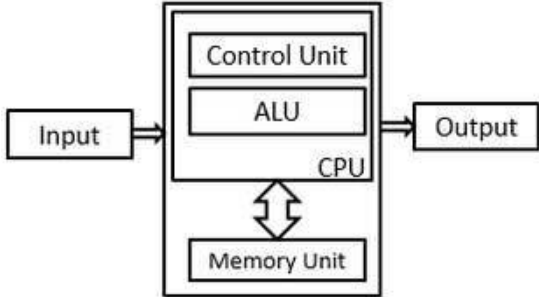
Following are some of the important differences between Computer Architecture and Computer Organization.

Key	Computer Architecture	Computer Organization
Purpose	Computer architecture explains what a computer should do.	Computer organization explains how a computer works.
Target	Computer architecture provides functional behavior of computer system.	Computer organization provides structural relationships between parts of computer system.
Design	Computer architecture deals with high level design.	Computer organization deals with low level design.
Actors	Actors in Computer architecture are hardware parts.	Actor in computer organization is performance.
Order	Computer architecture is designed first.	Computer organization is started after finalizing computer architecture.

### Review of Evolution of Computer System

Generation of computers

## Difference between Von Neumann and Harvard Architecture

Point of Comparison	Harvard Architecture	Von Neumann Architecture
Arrangement	<p>In Harvard architecture, the CPU is connected with both the data memory (RAM) and program memory (ROM), separately.</p>  <p style="text-align: center;"><b>Harvard Model</b></p>	<p>In Von-Neumann architecture, there is no separate data and program memory. Instead, a single memory connection is given to the CPU.</p>  <p style="text-align: center;"><b>Von Neumann Model</b></p>
Hardware requirements	It requires more hardware since it will be requiring separate data and address bus for each memory.	In contrast to the Harvard architecture, this requires less hardware since only a common memory needs to be reached.
Space requirements	This requires more space.	Von-Neumann Architecture requires less space.
Speed of execution	Speed of execution is faster because the processor fetches data and instructions simultaneously.	Speed of execution is slower since it cannot fetch the data and instructions at the same time.
Space usage	It results in wastage of space since if the space is left in the data memory then the instructions memory cannot use the space of the data memory and vice-versa.	Space is not wasted because the space of the data memory can be utilized by the instructions memory and vice-versa.
Controlling	Controlling becomes complex since data and instructions are to be fetched simultaneously.	Controlling becomes simpler since either data or instructions are to be fetched at a time.

### Extended Structure of IAS (Von Neumann) Computer

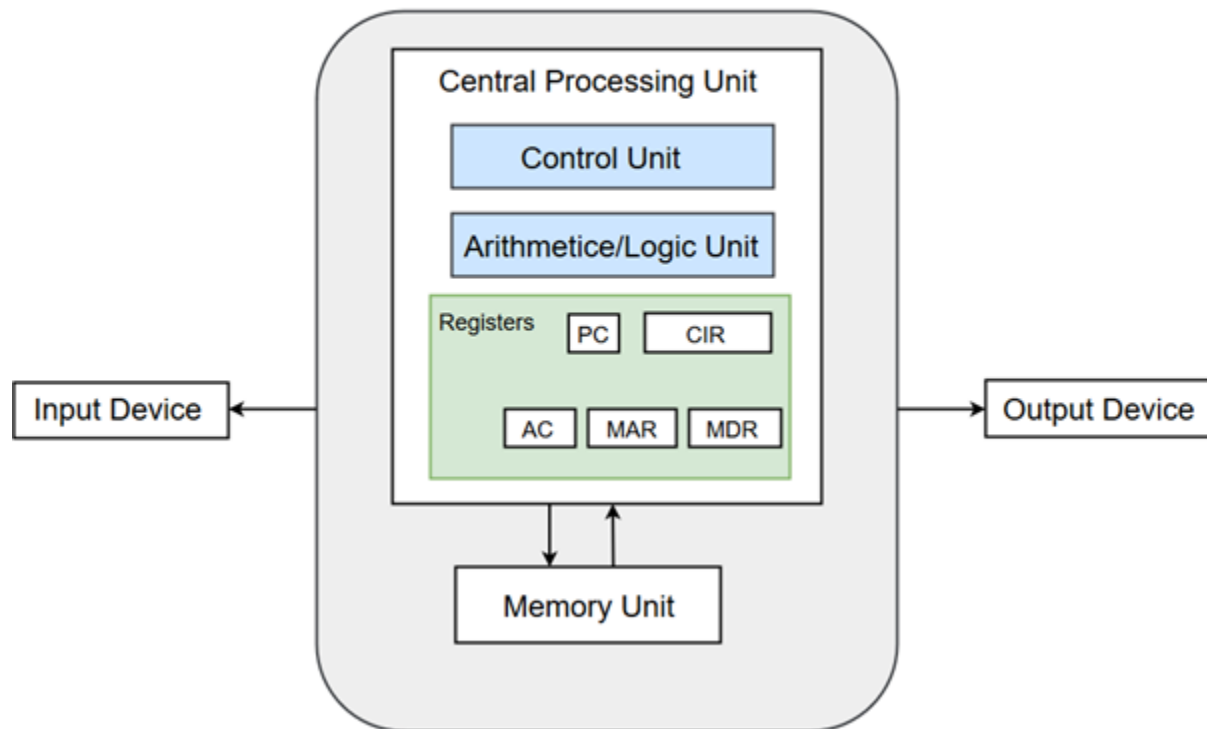
Von-Neumann proposed his computer architecture design in 1945 which was later known as Von-Neumann Architecture. It consisted of a Control Unit, Arithmetic, and Logical Memory Unit (ALU), Registers and Inputs/Outputs.

Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory. This design is still used in most computers produced today.

#### A Von Neumann-based computer:

- Uses a single processor
- Uses one memory for both instructions and data.
- Executes programs following the fetch-decode-execute cycle

#### Von-Neumann Basic Structure:



Components of Von-Neumann Model:

- Central Processing Unit
- Buses
- Memory Unit

### Central Processing Unit

The part of the Computer that performs the bulk of data processing operations is called the Central Processing Unit and is referred to as the CPU.

The Central Processing Unit can also be defined as an electric circuit responsible for executing the instructions of a computer program.

The CPU performs a variety of functions dictated by the type of instructions that are incorporated in the computer.

The major components of CPU are Arithmetic and Logic Unit (ALU), Control Unit (CU) and a variety of registers.

### **Arithmetic and Logic Unit (ALU)**

The Arithmetic and Logic Unit (ALU) performs the required micro-operations for executing the instructions. In simple words, ALU allows arithmetic (add, subtract, etc.) and logic (AND, OR, NOT, etc.) operations to be carried out.

### **Control Unit**

The Control Unit of a computer system controls the operations of components like ALU, memory and input/output devices.

The Control Unit consists of a program counter that contains the address of the instructions to be fetched and an instruction register into which instructions are fetched from memory for execution.

### **Registers**

Registers refer to high-speed storage areas in the CPU. The data processed by the CPU are fetched from the registers.

Following is the list of registers that plays a crucial role in data processing.

Registers	Description
MAR (Memory Address Register)	This register holds the memory location of the data that needs to be accessed.
MDR (Memory Data Register)	This register holds the data that is being transferred to or from memory.
AC (Accumulator)	This register holds the intermediate arithmetic and logic results.
PC (Program Counter)	This register contains the address of the next instruction to be executed.
CIR (Current Instruction Register)	This register contains the current instruction during processing.

## Buses

Buses are the means by which information is shared between the registers in a multiple-register configuration system.

A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer.

Von-Neumann Architecture comprised of three major bus systems for data transfer.

Bus	Description
Address Bus	Address Bus carries the address of data (but not the data) between the processor and the memory.
Data Bus	Data Bus carries data between the processor, the memory unit and the input/output devices.
Control Bus	Control Bus carries signals/commands from the CPU.

## Memory Unit

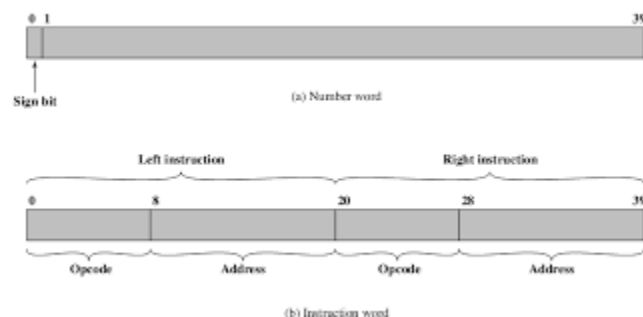
A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of the storage. The memory stores binary information in groups of bits called words. The internal structure of a memory unit is specified by the number of words it contains and the number of bits in each word.

### Memory and Instruction Format of IAS

The memory of the IAS consists of 1000 storage locations, called words, of 40 bits each. Both data and instructions are stored there. Numbers are represented in binary form, and each instruction is a binary code.

Each number is represented by a sign bit and a 39-bit value.

A word may also contain two 20-bit instructions, with each instruction consisting of an 8-bit operation code specifying the operation to be performed and a 12-bit address designated one of the words in memory (numbered from 0 to 999).



## MIPS

MIPS (Microprocessor without Interlocked Pipelined Stages) is a reduced instruction set computer (RISC) instruction set architecture (ISA) developed by MIPS Computer Systems, now MIPS Technologies, based in the United States.

### Instruction formats in MIPS

Instructions are divided into three types: R, I and J. Every instruction starts with a 6-bit opcode. In addition to the opcode, R-type instructions specify three registers, a shift amount field, and a function field; I-type instructions specify two registers and a 16-bit immediate value; J-type instructions follow the opcode with a 26-bit jump target.

The following are the three formats used for the core instruction set:

Type	format (bits)						Examples
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)	Used by add, sub, sll(shift left logical), srl(shift right logical etc.
I	opcode (6)	rs (5)	rt (5)	Immediate (16)			Used by lw (load word), sw (store word) etc
J	opcode (6)	address (26)					Used by j(jump), jal (Jump and Link)

Rs= First Source Operand

Rt= Second Source Operand (Register Target)

Rd= Result Operand (Register Destination)

Shamt= Shift Amount

Funct= Function code

\*For instructions that share an opcode, the funct parameter contains the necessary control codes to differentiate the different instructions.

Immediate= offset value for loading and storing values

Address= target Address

### Addressing Modes

The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows:

1. To give the programming versatility to the user.
2. To reduce the number of bits in addressing field of instruction.

### Types of Addressing Modes

Below we have discussed different types of addressing modes one by one:

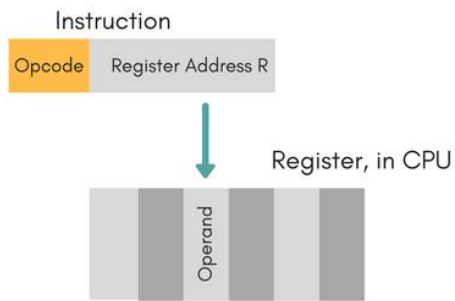
#### Immediate Mode

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

#### Register Mode

In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.



### **Advantages**

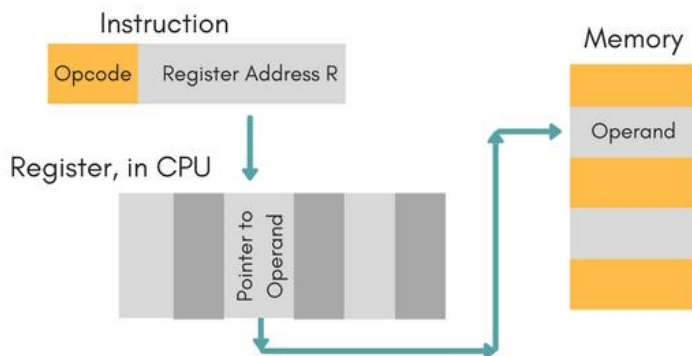
- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

### **Disadvantages**

- Very limited address space
- Using multiple registers helps performance but it complicates the instructions.

### **Register Indirect Mode**

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.



### **Auto Increment/Decrement Mode**

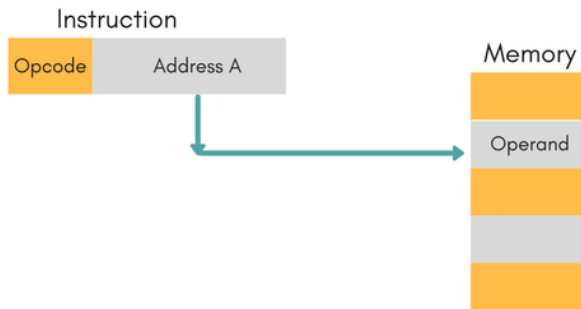
In this the register is incremented or decremented after or before its value is used.



## Direct Addressing Mode

In this mode, effective address of operand is present in instruction itself.

- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.

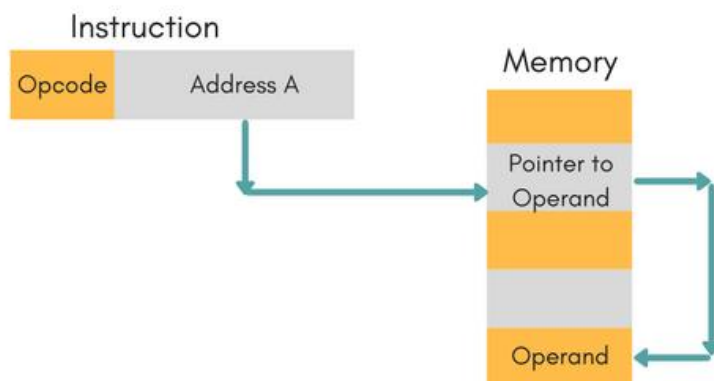


**For Example:** ADD R1, 4000 - In this the 4000 is effective address of operand.

**NOTE:** Effective Address is the location where operand is present.

## Indirect Addressing Mode

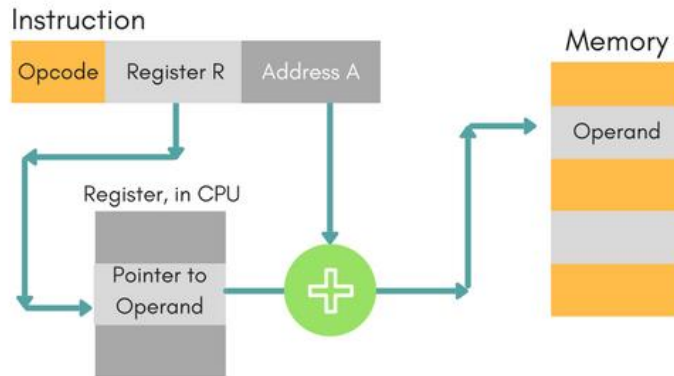
In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.



## Displacement Addressing Mode

In this the contents of the indexed register is added to the Address part of the instruction, to obtain the effective address of operand.

$EA = A + (R)$ , In this the address field holds two values, A(which is the base value) and R(that holds the displacement), or vice versa.



## Relative Addressing Mode

It is a version of Displacement addressing mode.

In this the contents of PC(Program Counter) is added to address part of instruction to obtain the effective address.

$EA = A + (PC)$ , where EA is effective address and PC is program counter.

The operand is A cells away from the current cell(the one pointed to by PC)

## Base Register Addressing Mode

It is again a version of Displacement addressing mode. This can be defined as  $EA = A + (R)$ , where A is displacement and R holds pointer to base address.

## Stack Addressing Mode

In this mode, operand is at the top of the stack. For example: ADD, this instruction will *POP* top two items from the stack, add them, and will then *PUSH* the result to the top of the stack.