**CAO**

* **A simple understanding of Computer**

**Computer** is a device that makes our work easy. Computer is a device that helps us to complete our task easily and speedily.

**But how a transistor gets its value?**  
When a very little amount of electric current passes through transistor it maintains the state of 1 and when there is no electric current then the transistor has the state of 0.

**Then how it’s all connected to computer?**  
This 0’s and 1’s forms the building block of computer. With the combinations of 0 and 1 we create a whole new language

**What is software ?**

Software is a set of instructions that tells the computer what to do, when to do, and how to do.

**So, who converts user code to machine language?**

A software called interpreter that interprets our language code into binary code. Interpreter converts our code into machine language that can be understood by computer.

**Now the question is how we give our input ?**

We give our input with the use of hardware for example like scanner, keyboard, mouse.

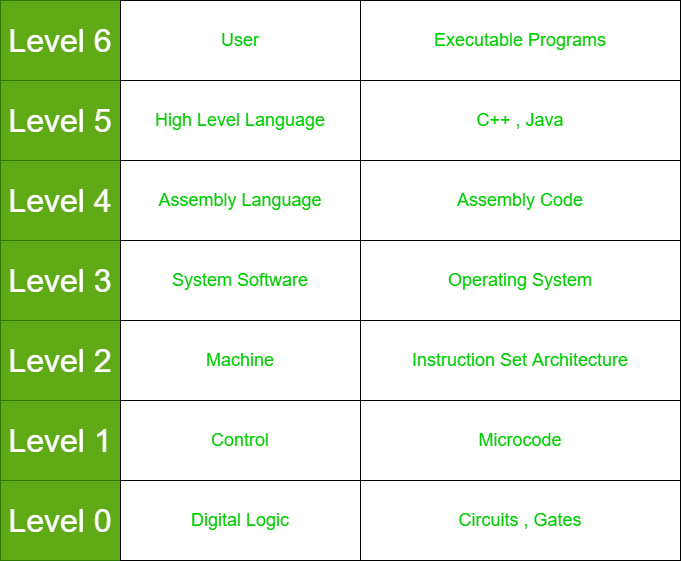
* **Issues in Computer Design**

**Computer Design** is the structure in which components relate to each other.

Following are the issues in computer design:

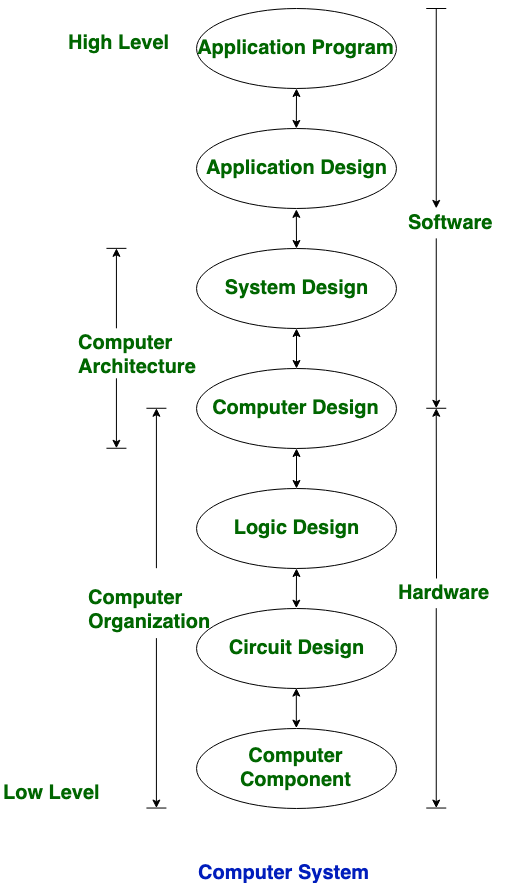
1. **Assumption of infinite speed:**   
   It can’t be assumed the infinite speed of the computer as it is not practical to assume the infinite speed. It creates problems in designer’s thinking as well.
2. **Assumption of infinite Memory:**   
   Like the speed of the computer, memory also can’t be assumed infinite. Storage is always finite and this is an issue in computer design.
3. **Speed mismatch between memory and processor:**   
   Sometimes it is possible that the speed of memory and processor does not match. It may be memory speed is faster or processor speed is faster. A mismatch between memory and processor leads to create problems in designing.
4. **Handling of bugs and errors:**   
   Handling bugs and errors are huge responsibility of any computer designer. Bugs and errors lead to the failure of the computer system. Sometimes these errors may be more dangerous.
5. **Multiple processors:**   
   Designing a computer system with multiple processors leads to the huge task of management and programming. It is a big issue in computer designing.
6. **Multiple threads:**   
   A computer system with multiple threads is always a threat to the designer. A computer with several threads should be able to multi-tasking and multi-processing.
7. **Shared memory:**   
   If there are several processes to be executed at a time then all the processes share the same memory space. It should be managed in a specific way so that collision does not happen.
8. **Disk access:**   
   Disk management is the key to computer design. There are several issues with disk access. It may be possible that the system does not support multiple disks accessing.
9. **Better performance:**   
   It is always an issue. A designer always tries to simplify the system for better performance in reduces power and less cost.

* **Computer System Level Hierarchy**



# 

# Differences between Computer Architecture and Computer Organization



Computer Architecture is a functional description of requirements and design implementation for the various parts of computer. It deals with functional behavior of computer system

**Computer Organization:**

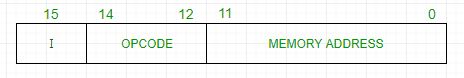
Computer Organization comes after the decide of Computer Architecture first. Computer Organization is how operational attribute are linked together and contribute to realize the architectural specification.

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Computer Architecture** | **Computer Organization** |
| 1. | Architecture describes what the computer does. | Organization describes how it does it. |
| 2. | Computer Architecture deals with functional behavior of computer system. | Computer Organization deals with structural relationship. |
| 3. | In above figure, its clear that it deals with high-level design issue. | In above figure, its also clear that it deals with low-level design issue. |
| 4. | Architecture indicates its hardware. | Where, Organization indicates its performance. |
| 5. | For designing a computer, its architecture is fixed first. | For designing a computer, organization is decided after its architecture. |
| 6. | Computer Architecture is also called as instruction set architecture. | Computer Organization is frequently called as micro architecture. |
| 7. | Computer Architecture comprises logical functions such as instruction sets, registers, data types and addressing modes. | Computer Organization consists of physical units like circuit designs, peripherals and adders. |
| 8. | Architecture coordinates between the hardware and software of the system. | Computer Organization handles the segments of the network in a system. |

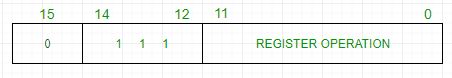
# Computer Organization | Basic Computer Instructions

The basic computer has 16-bit instruction register (IR) which can denote either memory reference or register reference or input-output instruction.

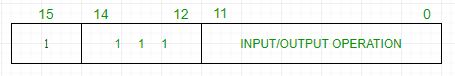
**Memory Reference –** These instructions refer to memory address as an operand. The other operand is always accumulator. Specifies 12-bit address, 3-bit opcode (other than 111) and 1-bit addressing mode for direct and indirect addressing. Hence, DR ← M[AR]AC ← AC + DR, SC ← 0



**Register Reference –** These instructions perform operations on registers rather than memory addresses. The IR (14 – 12) is 111 (differentiates it from memory reference) and IR (15) is 0 (differentiates it from input/output instructions). The rest 12 bits specify register operation.



**Input/Output –** These instructions are for communication between computer and outside environment. The IR (14 – 12) is 111 (differentiates it from memory reference) and IR (15) is 1 (differentiates it from register reference instructions). The rest 12 bits specify I/O operation.



The set of instructions incorporated in16 bit IR register are:

1. Arithmetic, logical and shift instructions (and, add, complement, circulate left, right, etc)
2. To move information to and from memory (store the accumulator, load the accumulator)
3. Program control instructions with status conditions (branch, skip)
4. Input output instructions (input character, output character)

**Difference between assembly language and high-level language :**

|  |  |
| --- | --- |
| **ASSEMBLY LEVEL LANGUAGE** | **HIGH-LEVE**L **LANGUAGE** |
| * It needs an assembler for conversion | * It needs a compiler/interpreter for conversion |
| * In this, we convert an Assembly level language to machine level language | * In this, we convert a high-level language to Assembly level language to machine level language |
| * It is machine dependent | * It is machine-independent |
| * In this mnemonic and codes are used | * In this English statement is used |
| * It supports low-level operation | * It does not support low-level language |
| * In this, it is easy to access hardware component | * In this, it is difficult to access hardware component |
| * In this more compact code | * No compactness |

# Addressing Modes

**Addressing Modes** refers to the way in which the operand of an instruction is specified.

**Addressing modes for 8086 instructions are divided into two categories:**

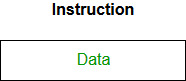
1) Addressing modes for data

2) Addressing modes for branch

An assembly language program instruction consists of two parts

am1

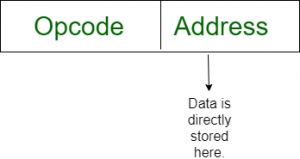
**Implied mode:** In implied addressing the operand is specified in the instruction itself. In this mode the data is 8 bits or 16 bits long and data is the part of instruction. **Zero address instruction** are designed with implied addressing mode



Example: Suppose the instruction is CRC then it means clear carry flag so it is itself and instruction

And operand.

**Immediate addressing mode (symbol #) ->** In this mode data is present in address field of instruction .Designed like one address instruction format

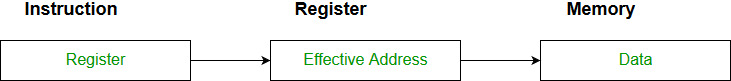


**Register mode:**In register addressing the operand is placed in one of 8 bit or 16bit general purpose registers. The data is in the register that is specified by the instruction.



Here one register reference is required to access the data

**Register Indirect mode**: In this addressing the operand’s offset is placed in any one of the registers BX, BP, SI, DI as specified in the instruction. The effective address of the data is in the base register or an index register that is specified by the instruction.



Here two register reference is required to access the data

**Auto Indexed (increment mode)**: Effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next consecutive memory location.**(R1)+**.

|  |
| --- |
| REGISTER |

|  |
| --- |
| MEMO. ADD1 |

|  |
| --- |
| MEMO. ADD2 |

After fetching the instructions ,

|  |
| --- |
| REGISTER |

|  |
| --- |
| MEMO. ADD1 |

|  |
| --- |
| MEMO. ADD2 |

Here one register reference,one memory reference and one ALU operation is required to access the data

**Auto indexed ( decrement mode)**: Effective address of the operand is the contents of a register specified in the instruction. Before accessing the operand, the contents of this register are automatically decremented to point to the previous consecutive memory location. –**(R1)**

|  |
| --- |
| REGISTER |

|  |
| --- |
| MEMO. ADD1 |

|  |
| --- |
| MEMO. ADD2 |

After fetching the instruction it gets decremented,

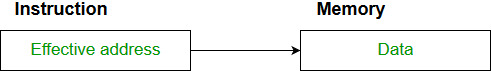
|  |
| --- |
| REGISTER |

|  |
| --- |
| MEMO. ADD1 |

|  |
| --- |
| MEMO. ADD2 |

Here one register reference,one memory reference and one ALU operation is required to access the data

**Direct addressing/ Absolute addressing Mode (symbol [ ]):** The operand’s offset is given in the instruction as an 8 bit or 16 bit displacement element. In this addressing mode the 16 bit effective address of the data is the part of the instruction.



Here only one memory reference operation is required to access the data

**Indirect addressing Mode (symbol @ or () )**:In this mode address field of instruction contains the address of effective address.Here two references are required.  
1st reference to get effective address.  
2nd reference to access the data.

Register

|  |
| --- |
| Data |



Effective address

Register address

Register Indirect- Here one register reference,one memory reference is required to access the data.

Memory Indirect- Here two memory reference is required to access the data.

**Indexed addressing mode**: The operand’s offset is the sum of the content of an index register SI or DI and an 8 bit or 16 bit displacement.

|  |
| --- |
| 100 |
| 101 |
| 102 |
| 103 |
| 104 |
| 105 |
| 106 |
| 107 |

|  |
| --- |
| 4 |

Instruction Index register

|  |
| --- |
| 100  (base address) |

**Based on Transfer of control, addressing modes are:**

**1.Base register addressing mode:** The content of base register is added to the address part of the instruction to obtain the effective address. A base register is assumed to hold a base address and the address field of the instruction gives displacement relative to the base address.

|  |
| --- |
| BASE REGISTER ADDRESS |

|  |
| --- |
| 100 |
| 101 |
| 102 |
| 103 |
| 104 |
| 105 |
| 106 |
| 107 |

|  |
| --- |
| OFFSET VALUE |

**PC relative addressing mode:** PC relative addressing mode is used to implement intra segment transfer of control, In this mode effective address is obtained by adding displacement to PC.

**Advantages of Addressing Modes**

* To give programmers to facilities such as Pointers, counters for loop controls, indexing of data and program relocation.
* To reduce the number bits in the addressing field of the Instruction.

# Difference between Memory based and Register based Addressing Modes

Addressing mode very much depend on the type of CPU organisation. There are three types of CPU organisation:

1. Single Accumulator organisation
2. General register organisation
3. Stack organisation

| **Memory Based Addressing Modes** |  |  | **Register Based Addressing Modes** |
| --- | --- | --- | --- |
| The operand is present in memory and its address is given in the instruction itself. This addressing mode is taking proper advantage of memory address, e.g., Direct addressing mode |  |  | An operand will be given in one of the register and register number will be provided in the instruction.With the register number present in instruction, operand is fetched, e.g., Register mode |
| The memory address specified in instruction may give the address where the effective address is stored in the memory. In this case effective memory address is present in the memory address which is specified in the instruction, e.g., Indirect Addressing Mode |  |  | The register contains the address of the operand. The effective address can be derived from the content of the register specified in the instruction. The content of the register might not be the effective address. This mode takes full advantage of registers, e.g., Register indirect mode |
| The content of base register is added to the address part of the instruction to obtain the effective address. A base register is assumed to hold a base address and the address field of the instruction gives displacement relative to the base address, e.g., Base Register Addressing Mode |  |  | If we are having a table of data and our program needs to access all the values one by one we need something which decrements the program counter/or any register which has base address. Though in this case register is basically decreased, it is register based addressing mode, e.g., In Auto decrements mode |
| The content of the index register is added to the address part that is given in the instruction to obtain the effective address. Index Mode is used to access an array whose elements are in successive memory locations, e.g., Indexed Addressing Mode |  |  | If we are having a table of data and our program needs to access all the values one by one we need something which increment the program counter/or any register which has base address, e.g., Auto increment mode |
| The content of program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction in this case is usually a signed number which can be either positive or negative, e.g., Relative addressing mode |  |  | Instructions generally used for initializing registers to a constant value is register based addressing mode,and this technique is very useful approach, e.g., Immediate mode. |

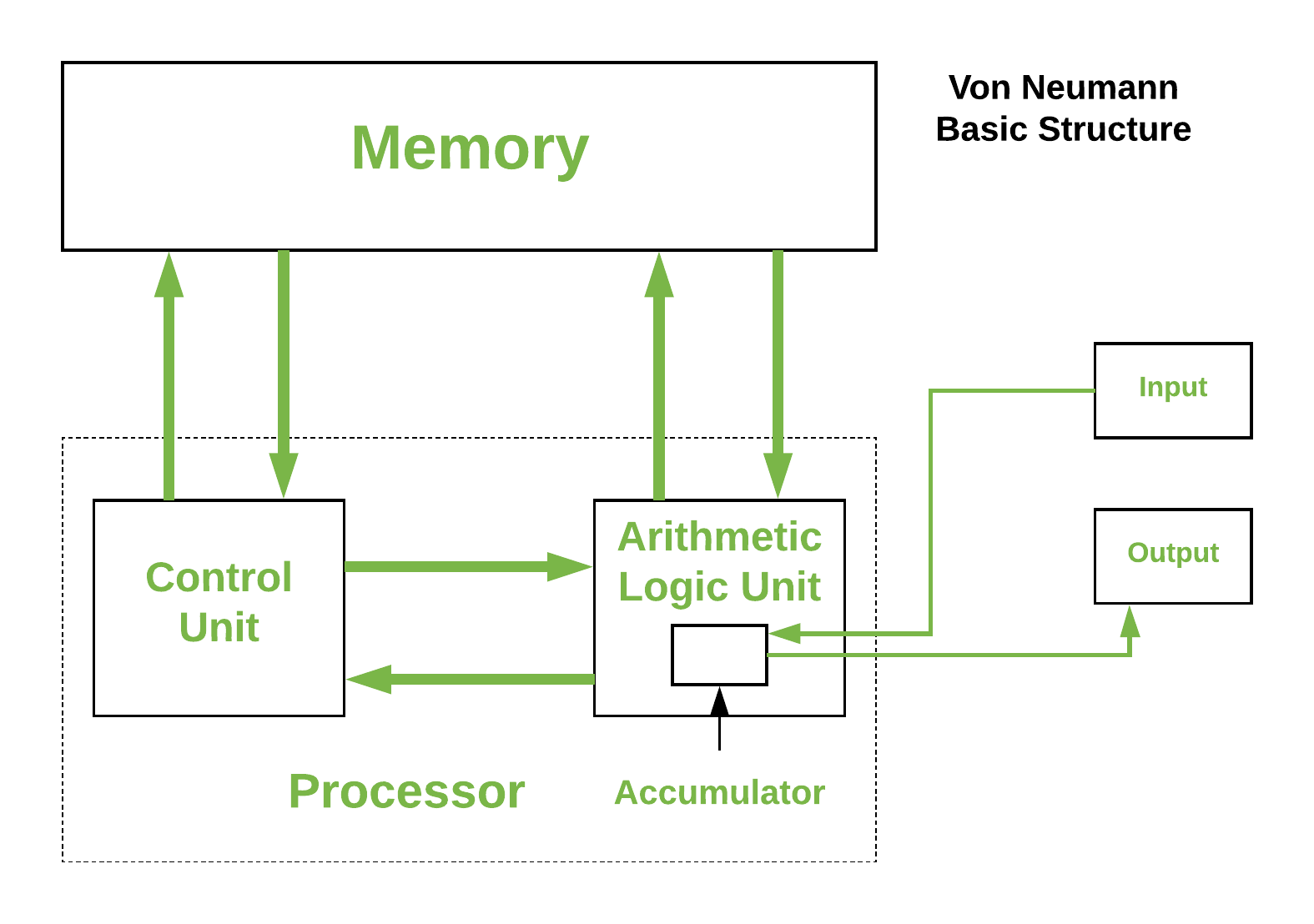
# Computer Organization | Von Neumann architecture

Historically there have been 2 types of Computers:

1. **Fixed Program Computers –** Their function is very specific and they couldn’t be programmed, e.g. Calculators.
2. **Stored Program Computers –** These can be programmed to carry out many different tasks, applications are stored on them, hence the name

In this stored-program concept, programs and data are stored in a separate storage unit called memories and are treated the same. This novel idea meant that a computer built with this architecture would be much easier to reprogram.

The basic structure is like,



It is also known as **IAS** computer and is having three basic units

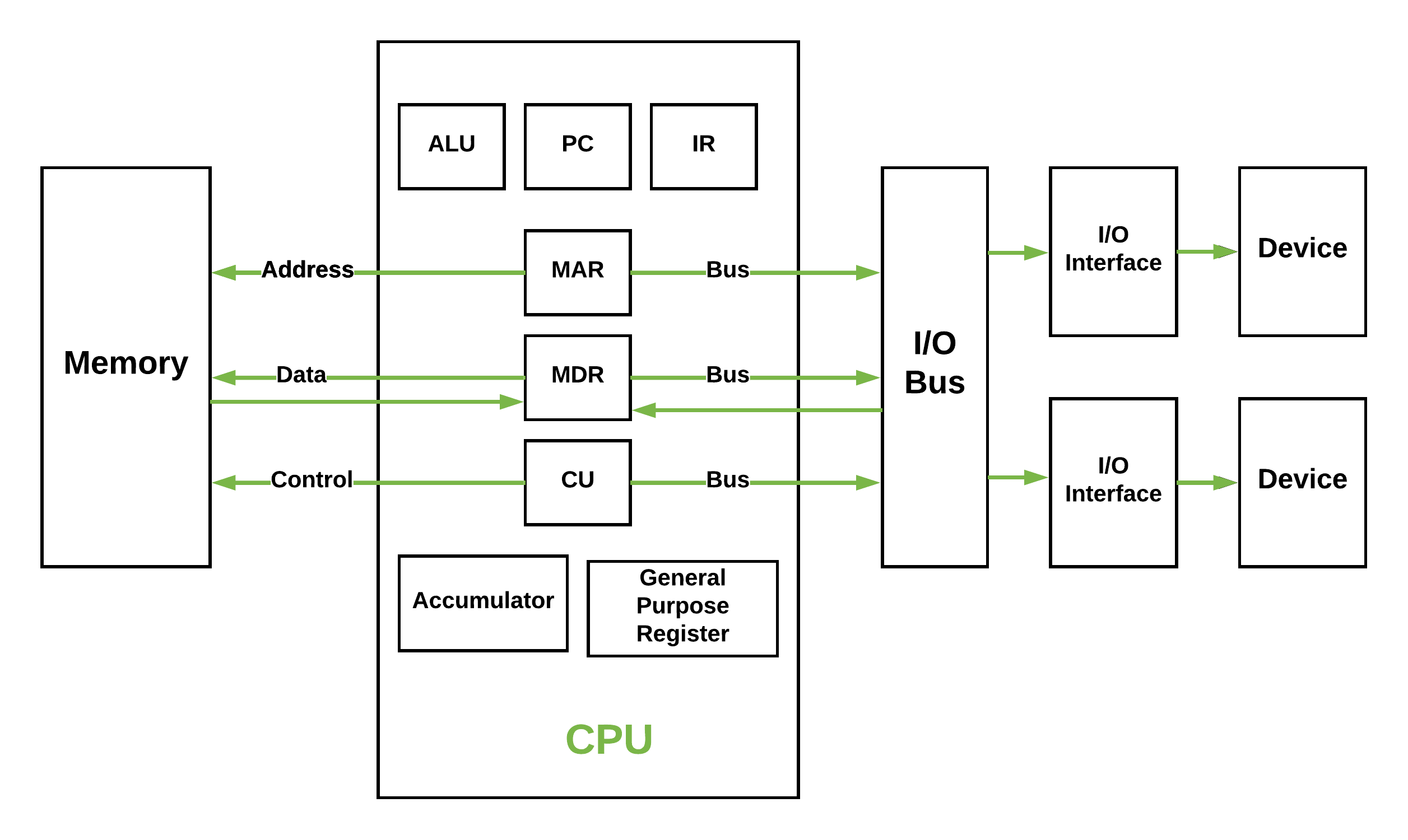
1. The Central Processing Unit (CPU)
2. The Main Memory Unit
3. The Input/Output Device

**Control Unit –**

A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions and controlling how data moves around the system.

**Arithmetic and Logic Unit (ALU) –**

The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need, e.g. Addition, Subtraction, Comparisons. It performs Logical Operations, Bit Shifting Operations, and Arithmetic Operation.

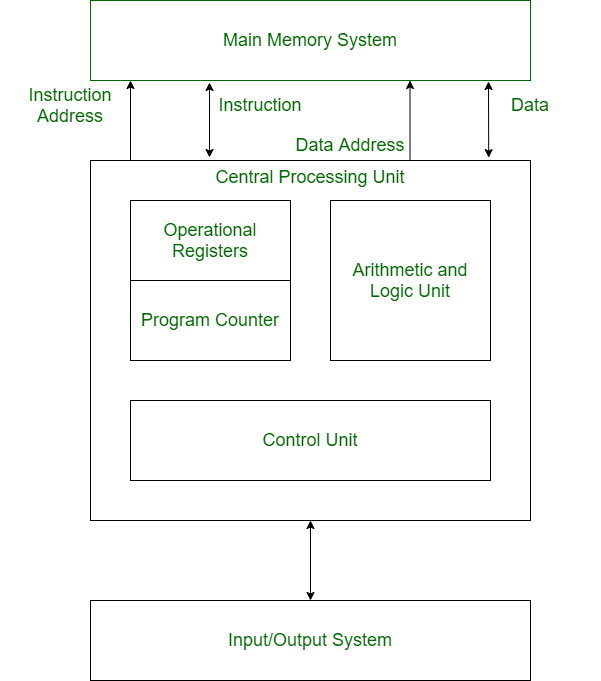


* **Main Memory Unit (Registers) –**
  1. **Accumulator:** Stores the results of calculations made by ALU.
  2. **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The PC then passes this next address to Memory Address Register (MAR).
  3. **Memory Address Register (MAR):** It stores the memory locations of instructions that need to be fetched from memory or stored into memory.
  4. **Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
  5. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
  6. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.

# **Harvard Architecture**

**Harvard Architecture** is the computer architecture that contains separate storage and separate buses (signal path) for instruction and data.

*It was basically developed to overcome the bottleneck of Von Neumann Architecture*



* **Buses:**  
  Buses are used as signal pathways. In Harvard architecture there are separate buses for both instruction and data. Types of Buses:  
  **Data Bus:** It carries data among the main memory system, processor and I/O devices.  
  **Data Address Bus:** It carries the address of data from processor to main memory system.  
  **Instruction Bus:** It carries instructions among the main memory system, processor and I/O devices.  
  **Instruction Address Bus:** It carries the address of instructions from processor to main memory system.
* **Operational Registers:**  
  There are different types of registers involved in it which are used for storing address of different types of instructions.  
  For example, Memory Address Register and Memory Data Register are operational registers.
* **Program Counter:**  
  It has the location of the next instruction to be executed. Program counter then passes this next address to memory address register.
* **Arithmetic and Logic Unit:**  
  Arithmetic logic unit is that part of the CPU that operates all the calculations needed. It performs addition, subtraction, comparison, logical Operations, bit Shifting Operations and various arithmetic operations.
* **Control Unit:**  
  Control unit the part of CPU that operates all processor control signals. It controls the input and output devices and also control the movement of instructions and data within the system.
* **Input/Output System:**  
  Input devices are used to read data into main memory with the help of CPU input instruction. The information from a computer as output are given through Output devices. Computer gives the results of computation with the help of output devices.

**Advantage of Harvard Architecture:**  
Harvard architecture has two separate buses for instruction and data. Hence, CPU can access instructions and read/write data at the same time. This is the major advantage of Harvard architecture.

| VON NEUMANN ARCHITECTURE | HARVARD ARCHITECTURE |
| --- | --- |
| Based on stored program computer concept. | Based on Harvard Mark I relay based model. |
| Same physical memory address is used for instructions and data. | Separate physical memory address is used for instructions and data. |
| There is common bus for data and instruction transfer. | Separate buses are used for transferring data and instruction. |
| Two clock cycles are required to execute single instruction. | An instruction is executed in a single cycle. |
| It is cheaper in cost. | It is costly than Von Neumann Architecture. |
| CPU can not access instructions and read/write at the same time. | CPU can access instructions and read/write at the same time. |
| It is used in personal computers and small computers. | It is used in micro controllers and signal processing. |

# **Interaction of a Program with Hardware**

Lightbox

**• Instruction Set Architecture (ISA)**

- Serves as an interface between software and hardware.

- Typically consists of information regarding the programmer's view of the

architecture (i.e. the registers, address and data buses, etc.).

Also consists of the instruction set.

Types of ISA :

1.Accumulator based (acc=acc+r1 =>1 address instruction)

2.Stack based (tos=tos +next =>0 address instruction)

3.Memory-memory based( mem[a]=mem[a]+mem[b]=>2,3 address instruction)

4.Register-memory based(r1=mem[a]=>2 address instruction)

Also called **load store architecture** .

5. Register register based(R1=r2+r3 =>3 address instruction)

|  |  |
| --- | --- |
| RISC | CISC |
| Reduced inst.set archit. | Complex inst.set architec. |
| Hardware friendly (min. no. of instruction  And all inst.are of same length,interpretation is easy) | Hardware’s hell (hundred’s of instruction  Not of same length and interpretation is difficult) |
| Difficult for compilers as it is not fancy. | Easy for compilers |

**Reduced Instruction Set Architecture (RISC) –**   
The main idea behind is to make hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating, and storing operations just like a load command will load data, store command will store the data.

**Complex Instruction Set Architecture (CISC) –**   
The main idea is that a single instruction will do all loading, evaluating, and storing operations just like a multiplication command will do stuff like loading data, evaluating, and storing it, hence it’s complex.

**RISC:** Reduce the cycles per instruction at the cost of the number of instructions per program.

**CISC:** The CISC approach attempts to minimize the number of instructions per program but at the cost of increase in number of cycles per instruction.

**Characteristic of RISC –** 

1. Simpler instruction, hence simple instruction decoding.
2. Instruction comes undersize of one word.
3. Instruction takes a single clock cycle to get executed.
4. More number of general-purpose registers.
5. Simple Addressing Modes.
6. Less Data types.
7. Pipeline can be achieved.

**Characteristic of CISC –** 

1. Complex instruction, hence complex instruction decoding.
2. Instructions are larger than one-word size.
3. Instruction may take more than a single clock cycle to get executed.
4. Less number of general-purpose registers as operation get performed in memory itself.
5. Complex Addressing Modes.
6. More Data types.

| **RISC** | **CISC** |
| --- | --- |
| Focus on software | Focus on hardware |
| Uses only Hardwired control unit | Uses both hardwired and micro programmed control unit |
| Transistors are used for more registers | Transistors are used for storing complex  Instructions |
| Fixed sized instructions | Variable sized instructions |
| Can perform only Register to Register Arithmetic operations | Can perform REG to REG or REG to MEM or MEM to MEM |
| Requires more number of registers | Requires less number of registers |
| Code size is large | Code size is small |
| An instruction execute in a single clock cycle | Instruction takes more than one clock cycle |
| An instruction fit in one word | Instructions are larger than the size of one word |

# **Difference between RISC and CISC processor**

A **microprocessor** is a processing unit on a single chip. It is an integrated circuit which performs the core functions of a computer CPU

**Characteristics of a micro processor:** 

* **Instruction Set –**   
  Set of complete instructions that the microprocessor executes is termed as the instruction set.
* **Word Length –**   
  The number of bits processed in a single instruction is called word length or word size. Greater the word size, larger the processing power of the CPU.
* **System Clock Speed –**   
  Clock speed determines how fast a single instruction can be executed in a processor. The microprocessor’s pace is controlled by the System Clock. Clock speeds are generally measured in million of cycles per second (MHz) and thousand million of cycles per second (GHz).

**Classification of Microprocessors:**

**RISC:**  
It stands for Reduced Instruction Set Computer. It is a type of microprocessor architecture that uses a small set of instructions of uniform length. These are simple instructions which are generally executed in one clock cycle. RISC chips are relatively simple to design and inexpensive.The setback of this design is that the computer has to repeatedly perform simple operations to execute a larger program having a large number of processing operations.

-hardware friendly (a)min no. of instruction

(b)all instruction of same length

(c)interpretation is easy

-compiler’s hell

**CISC:**   
It stands for Complex Instruction Set Computer. These processors offer the users, hundreds of instructions of variable sizes. CISC architecture includes a complete set of special purpose circuits that carry out these instructions at a very high speed. These instructions interact with memory by using complex addressing modes. CISC processors reduce the program size and hence lesser number of memory cycles are required to execute the programs. This increases the overall speed of execution.

-hardware hell-(a)hundred’s of instruction

(b)not of same length

(c)interpretation is difficult

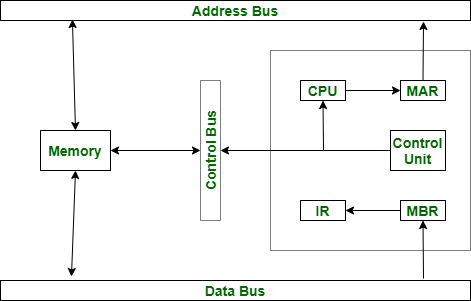
-compiler’s heaven

-decoding becomes diffiicult as don’t now where the instruction is ending as we have to look for them byte by byte due to variable length.

| **CISC** | **RISC** |
| --- | --- |
| Complex instruction set architecture | Reduced instruction set architecture |
| A large set of instruction | A smaller set of instruction. |
| Variable-length encodings of the instructions. | Fixed-length encodings of the instructions are used. |
| Aim to reduce the program instructions | Aim to improve the speed of the computer |
| CISC supports array. | RISC does not supports array. |
| Complex and simple instruction | Simple instruction |
| Implementation programs are hidden from machine level programs. The ISA provides a clean abstraction between programs and how they get executed. | Implementation programs exposed to machine level programs. Few RISC machines do not allow specific instruction sequences. |
| Various addressing modes | Fewer addressing modes |
| The stack is being used for procedure arguments and return addresses. | Registers are being used for procedure arguments and return addresses. Memory references can be avoided by some procedures. |

# **Essential Registers for Instruction Execution**

These are various registers required for execution of instruction : Program Counter (PC), Instruction Register (IR), Memory Buffer (or Data) Register (MBR or MDR), and Memory Address Register (MAR).



* **Program Counter (PC) :**  
  It contains the address of an instruction to be executed next. The PC is updated by the CPU after each instruction executed so that it always points to the next instruction to be executed. A branch or skip instruction will also modify the content of the PC.
* **Instruction Register (IR) :**  
  it contains the instruction most recently fetched or executed. The fetched instruction is loaded into an IR, where the opcode and operand specifier are analysed.
* **Memory Buffer (or Data) Register (MBR or MDR) :**  
  it contains a word of data to be written to memory . Contents of MBR are directly connected to the data bus.
* **Memory Address Register (MAR) :**  
  It contains the address of a location of main memory from where information has to be fetched . Contents of MAR is directly connected to the address bus.

# **Single Accumulator based CPU organization**

Advantages –

* One of the operands is always held by the accumulator register. This results in short instructions and less memory space.
* Instruction cycle takes less time because it saves time in instruction fetching from memory.
* Disadvantages –
* When complex expressions are computed, program size increases due to the usage of many short instructions to execute it. Thus memory size increases.
* As the number of instructions increases for a program, the execution time increases.

# Stack based CPU Organization

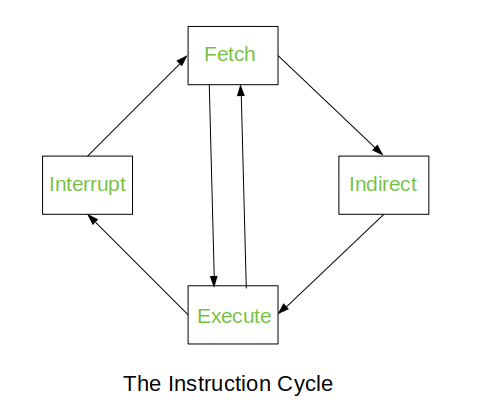
The advantages of Stack based CPU organization –

* Efficient computation of complex arithmetic expressions.
* Execution of instructions is fast because operand data are stored in consecutive memory locations.
* Length of instruction is short as they do not have address field.

The disadvantages of Stack based CPU organization –

* The size of the program increases.

# Instruction Cycles



 new 2-bit register called *Instruction Cycle Code*(ICC). The ICC designates the state of processor in terms of which portion of the cycle it is in:-

00 : Fetch Cycle   
01 : Indirect Cycle   
10 : Execute Cycle   
11 : Interrupt Cycle

At the end of the each cycles, the ICC is set appropriately

The Fetch Cycle –  
At the beginning of the fetch cycle, the address of the next instruction to be executed is in the *Program Counter*(PC).

Step 1: The address in the program counter is moved to the memory address register(MAR), as this is the only register which is connected to address lines of the system bus.

Step 2: The address in MAR is placed on the address bus, now the control unit issues a READ command on the control bus, and the result appears on the data bus and is then copied into the memory buffer register(MBR). Program counter is incremented by one, to get ready for the next instruction.(These two action can be performed simultaneously to save time)

Step 3: The content of the MBR is moved to the instruction register(IR).

The Indirect Cycles –

Once an instruction is fetched, the next step is to fetch source operands. *Source Operand* is being fetched by indirect addressing( it can be fetched by any [**addressing mode**](https://www.geeksforgeeks.org/addressing-modes/), here its done by indirect addressing). Register-based operands need not be fetched. Once the opcode is executed, a similar process may be needed to store the result in main memory

Step 1: The address field of the instruction is transferred to the MAR. This is used to fetch the address of the operand.   
Step 2: The address field of the IR is updated from the MBR.(So that it now contains a direct addressing rather than indirect addressing)   
Step 3: The IR is now in the state, as if indirect addressing has not been occurred.

The Execute Cycle

The other three cycles(*Fetch, Indirect and Interrupt*) are simple and predictable. Each of them requires simple, small and fixed sequence of micro-operation.

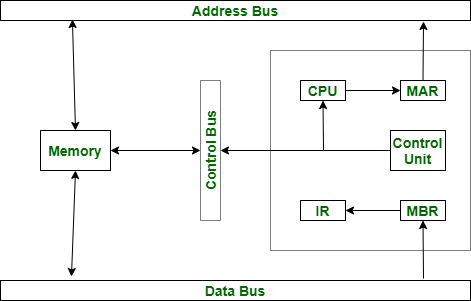
We begin with the IR containing the ADD instruction.   
Step 1: The address portion of IR is loaded into the MAR.   
Step 2: The address field of the IR is updated from the MBR, so the reference memory location is read.   
Step 3: Now, the contents of R and MBR are added by the ALU. The

Interrupt Cycle**:**  
At the completion of the Execute Cycle, a test is made to determine whether any enabled interrupt has occurred or not. If an enabled interrupt has occurred then Interrupt Cycle occurs. The natare of this cycle varies greatly from one machine to another.

Step 1: Contents of the PC is transferred to the MBR, so that they can be saved for return.   
Step 2: MAR is loaded with the address at which the contents of the PC are to be saved.   
PC is loaded with the address of the start of the interrupt-processing routine.   
Step 3: MBR, containing the old value of PC, is stored in memory.

# **Essential Registers for Instruction Execution**

These are various registers required for execution of instruction : Program Counter (PC), Instruction Register (IR), Memory Buffer (or Data) Register (MBR or MDR), and Memory Address Register (MAR).



# **Machine Instructions**

Machine Instructions are commands or programs written in machine code of a machine (computer) that it can recognize and execute.

**Machine instructions**

**1. Data transfer instructions**– move, load exchange, input, output

**2. Arithmetic instructions** – add, subtract, increment, decrement, convert byte/word and compare..

**3. Logic instructions** – AND, OR, exclusive OR, shift/rotate and test

**4**.**String manipulation instruction** – load, store, move, compare and scan for byte/word

**5. Control transfer instructions** – conditional, unconditional, call subroutine and return from subroutine.   
 **6. Loop control instructions-**

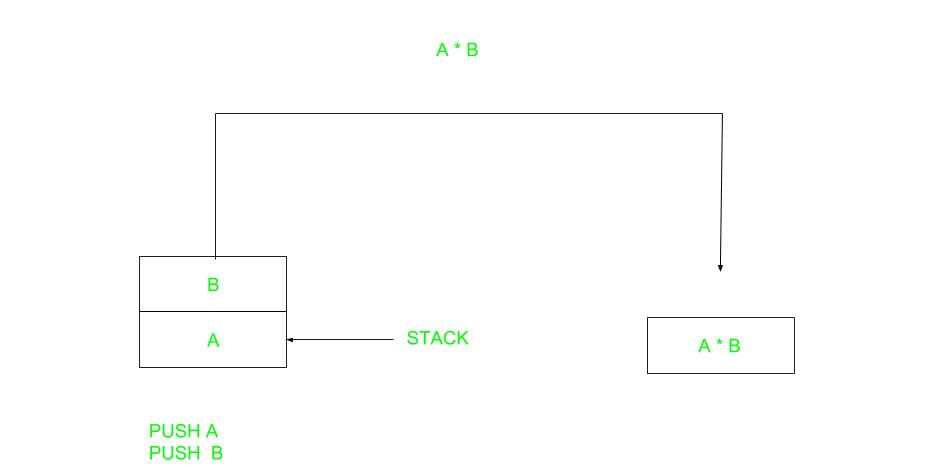
**7. Processor control instructions-** 

Flag manipulation:

PUSHF, POPF: Push flags onto stack, pop flags off stack.

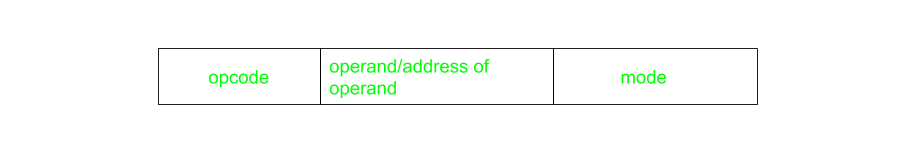
# **Instruction Formats (Zero, One, Two and Three Address Instruction)**

1. **Zero Address Instructions –**

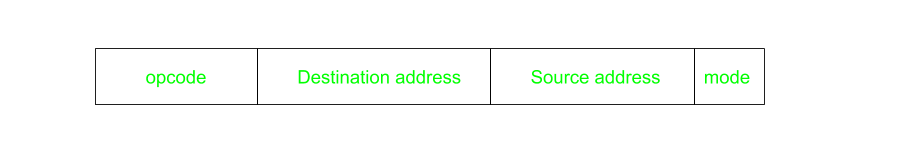


A stack-based computer does not use the address field in the instruction. To evaluate an expression first it is converted to reverse Polish Notation i.e. Postfix Notation.

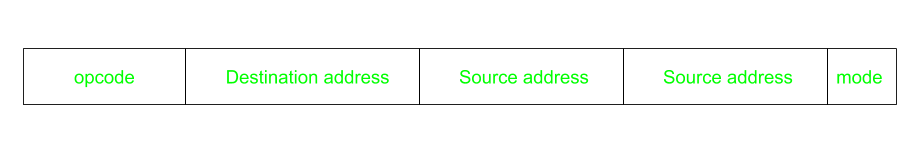
**2 .One Address Instructions –**   
This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.



**3.Two Address Instructions –**   
This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address.



**4.Three Address Instructions –**   
This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only



As the no. of address in the instruction keeps on increasing the program gets slower as more no of times we need to access the memory and accessing is fast in the cpu than memory.