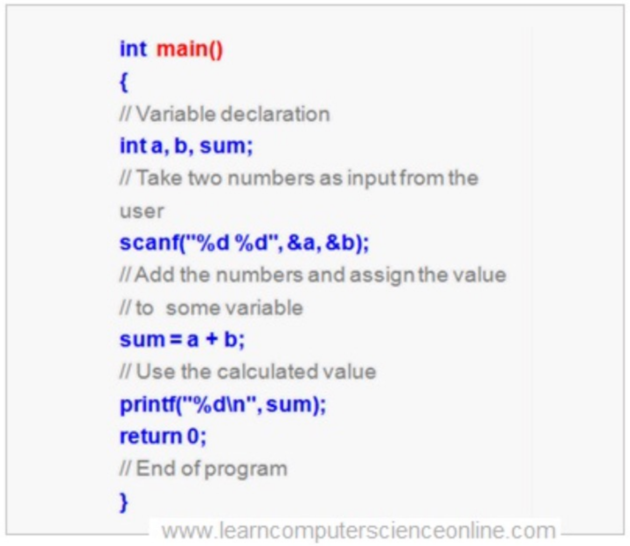
Instructions and instruction format

Instructions

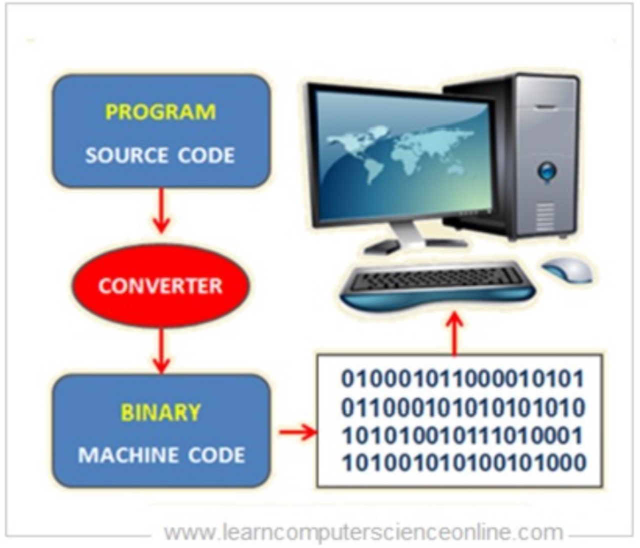
- To command a computer’s hardware, you must speak its language. The words of a computer’s language are called instructions, and its vocabulary is called an instruction set.
- Every task performed by the computer needs a program. A program is a set of statements , in other words instructions that directs the computer to perform a task.
- Instructions are kept in the computer as a series of high and low electronic signals and may be represented as numbers. In fact, each piece of an instruction can be considered as an individual number, and placing these numbers side by side forms the instruction

Program Source Code



```
int main()
{
// Variable declaration
int a, b, sum;
// Take two numbers as input from the
user
scanf("%d %d", &a, &b);
// Add the numbers and assign the value
// to  some variable
sum = a + b;
// Use the calculated value
printf("%d\n", sum);
return 0;
// End of program
}
```

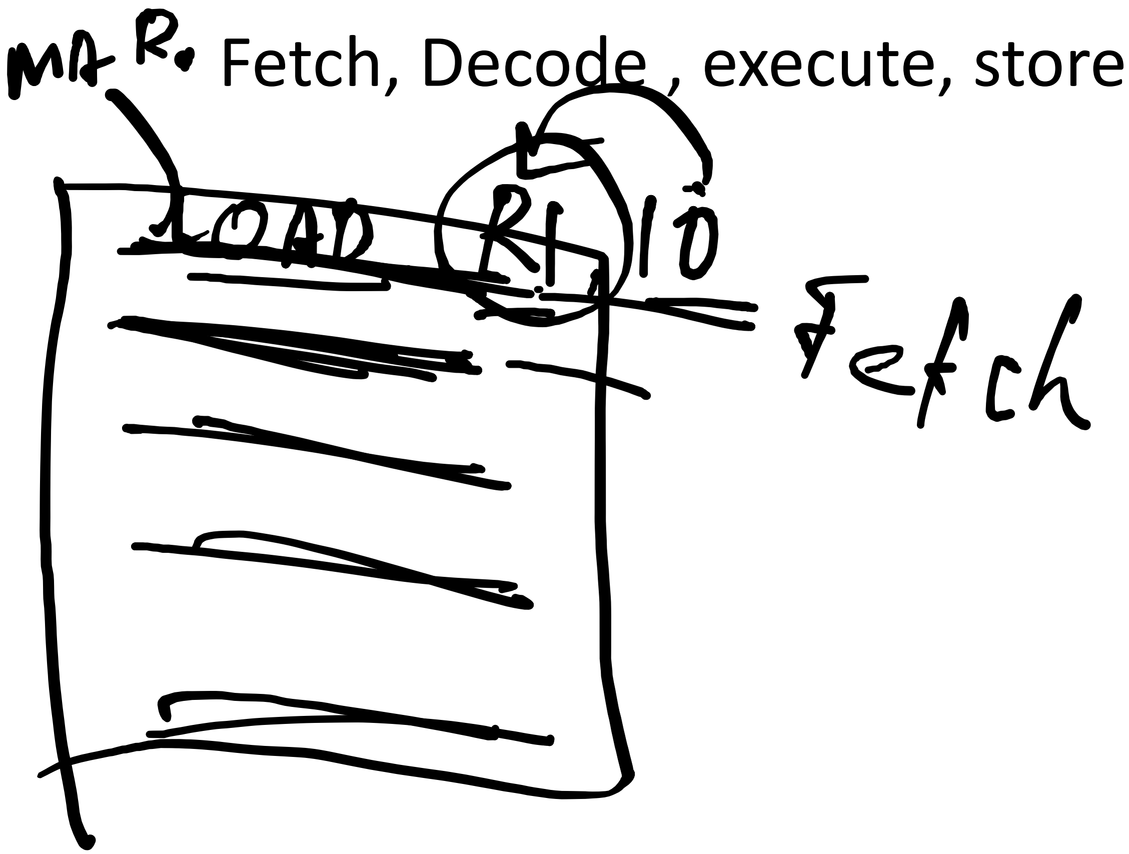
Program Compilation

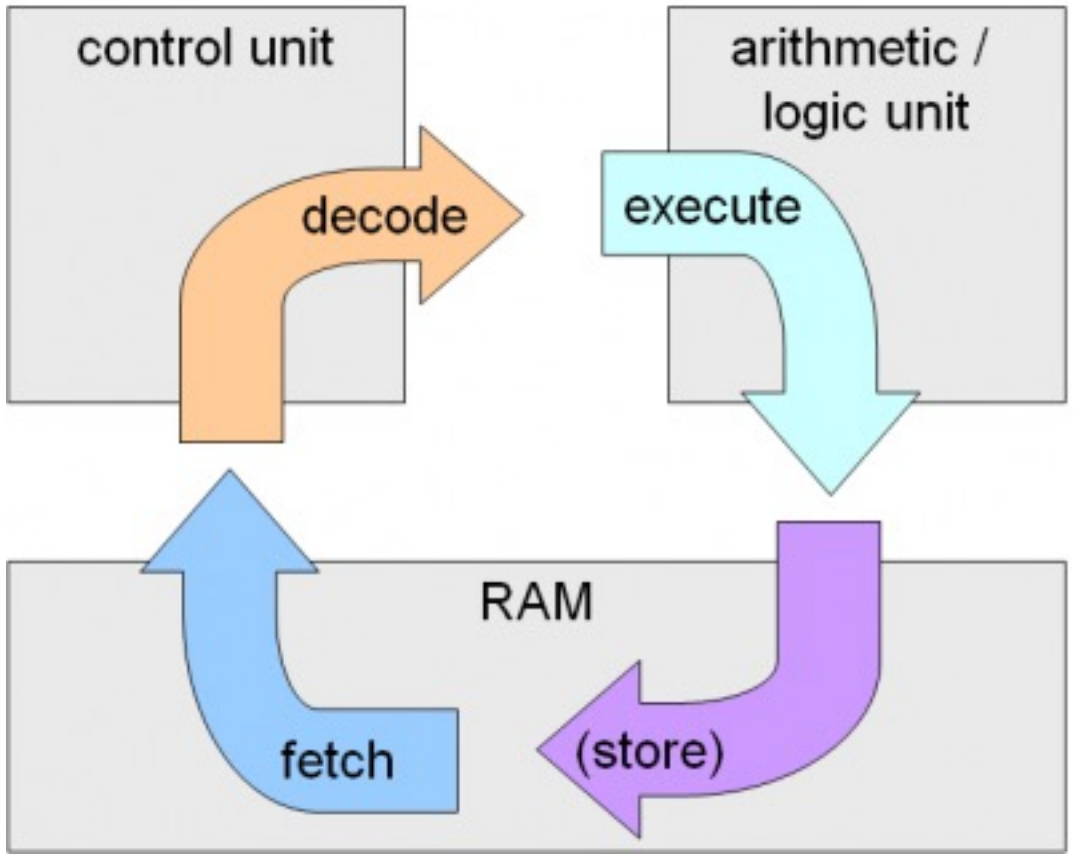


Machine → Processor

Instruction cycle

- Each phase of Instruction Cycle can be decomposed into a sequence of elementary micro-operations.
- Fetch, Decode , execute, store





Instruction format

- The instruction format describes the layout of instruction in terms of group of bits called fields of the instruction.
- Each field in the instruction format, provides a specific information to the processor. The processor needs to be informed regarding the operation to be performed and the location of data.
- The type of instruction format supported by the CPU depends on the instruction set architecture (ISA) implemented by the CPU

- Instruction set architecture

An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.

It can be viewed as a programmer's manual because it's the portion of the machine that's visible to the assembly language programmer, the compiler writer, and the application programmer.

The ISA defines the supported data types, the registers, how the hardware manages main memory, key features (such as virtual memory), which instructions a microprocessor can execute, and the input/output model of multiple ISA implementations.

- The general format for machine instruction is

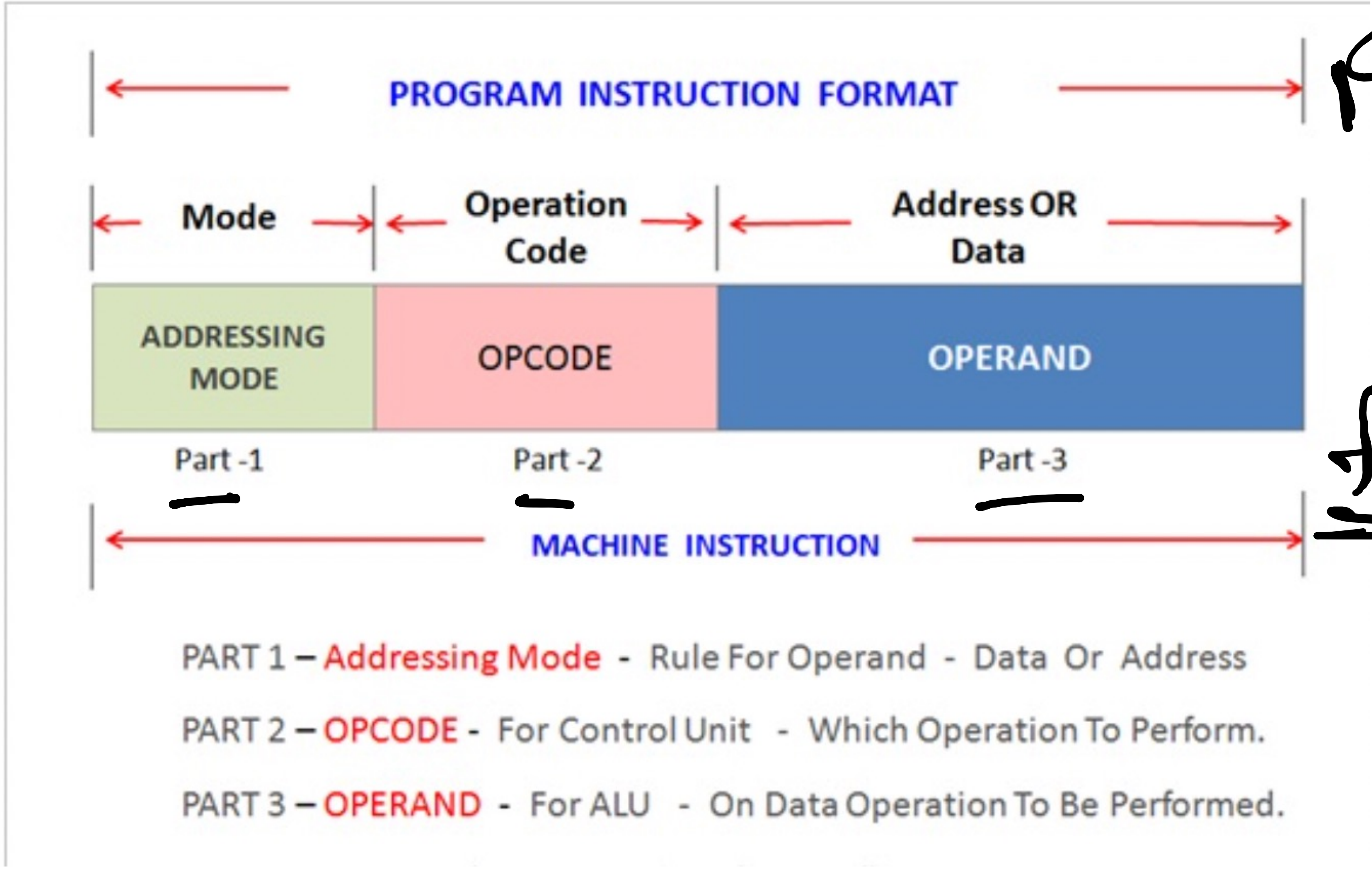
[mode:] Mnemonic [Operand, Operand] [; Comments]

ADD R1, R2

R1, 100

930002

field



PART 1 – Addressing Mode - Rule For Operand - Data Or Address

PART 2 – OPCODE - For Control Unit - Which Operation To Perform.

PART 3 – OPERAND - For ALU - On Data Operation To Be Performed.

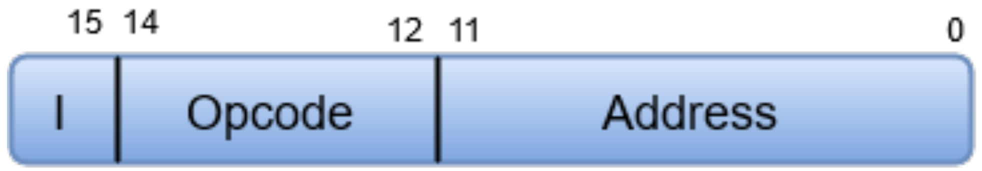
- In instruction format, the bits are grouped into 3 parts.
  - the 1st part specifies the addressing mode (MODE)
  - the 2nd part specifies the operation code (OPCODE)
  - the 3rd part is the address of the data (OPERAND)

- **ADDRESSING MODE :**

  - defines the manner in which the data is defined in instruction format. Represented by single bit.

  - For example, in the 16 bit instruction format the address mode is 15th bit.

  - directs the CPU to locate the data (OPERAND) as specified in the instruction format



ADD R1, R2

ADD R1, (1001)

ADD R1, 4

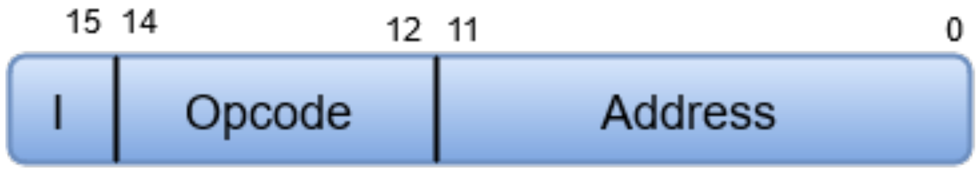
- **OPCODE**

  - during program execution, instruction is fetched into instruction register (IR). IR can be 8 bit, 16 bit, 32 bit register and a part of IR is used to store the operation code

  - the second part of instruction format is called operation code. Eg, in a 16 bit instruction format, group of three bits (bit number 12,13,14) indicates operation code (OPCODE)

  - OPCODe is decoded by the decoder of control unit.

  - control unit decodes the OPCODE as per instruction set architecture of CPU. Then then control unit generates control signals for ALU to execute the desired operation on OPERAND



- **OPERAND**

  - the operand part of instruction specifies either data or address that needs to be operated by the processor

  - after decoding, the CPU needs to know the data on which operation is to be performed as the OPCODE

  - depending upon the type of format supported by processor, instruction might have zero to three operands.

  - operand may contain either data or address of data. The addressing mode value directs the CPU accordingly to fetch the data

  - if value of addressing mode is 0, then it is direct referencing. For direct referencing, operand field contains the register where operand is stored

  - if addressing mode is 1, then it is indirecting referencing. The operand here is the memory address that points to data

- **Direct addressing mode :**

  - Add the content of R1 and 1001 and store back to R1:

    Add R1, (1001)

  Here 1001 is the address where operand is stored.

- **Indirect addressing mode:**

  - LOAD R1, @500

Above instruction is used to load the content of memory location stored at memory location 500 to register R1. In other words, we can say, effective address is stored at memory location 500.

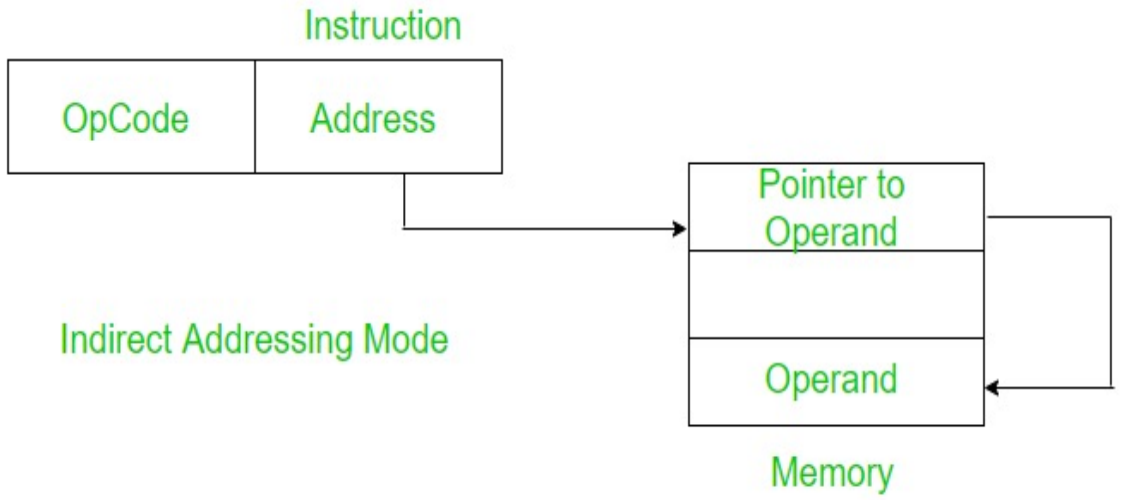
- **Immediate addressing mode**

  - the operand is a part of the instruction.

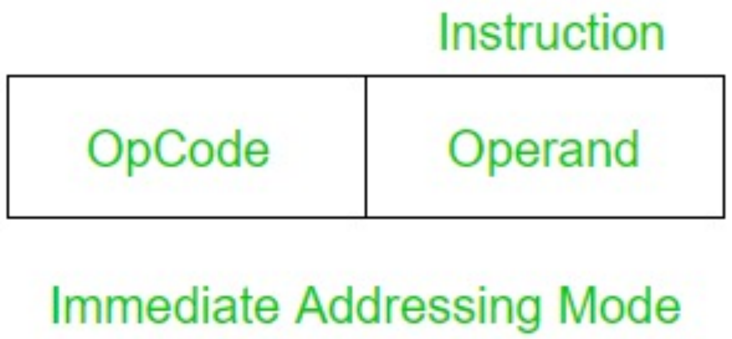
  - ADD 5

  Add 5 to accumulator

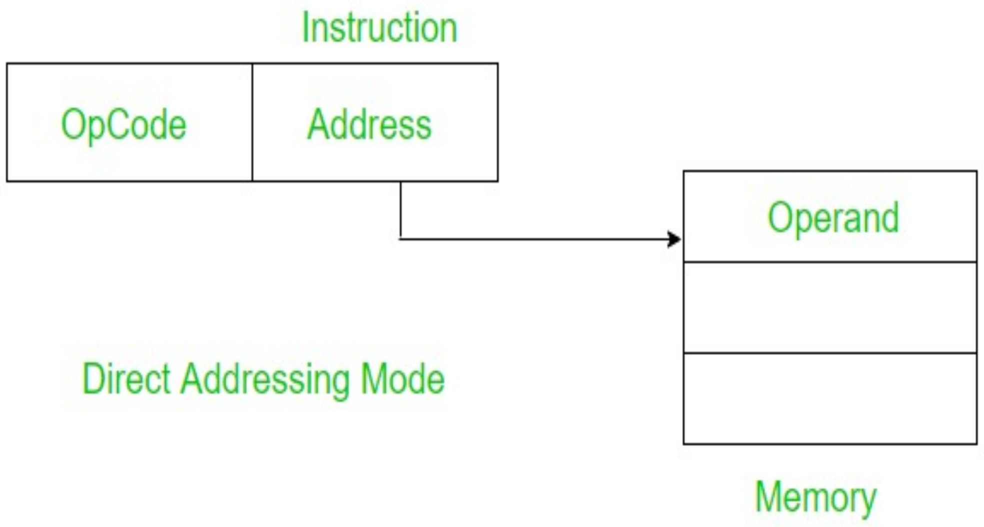
- Indirect

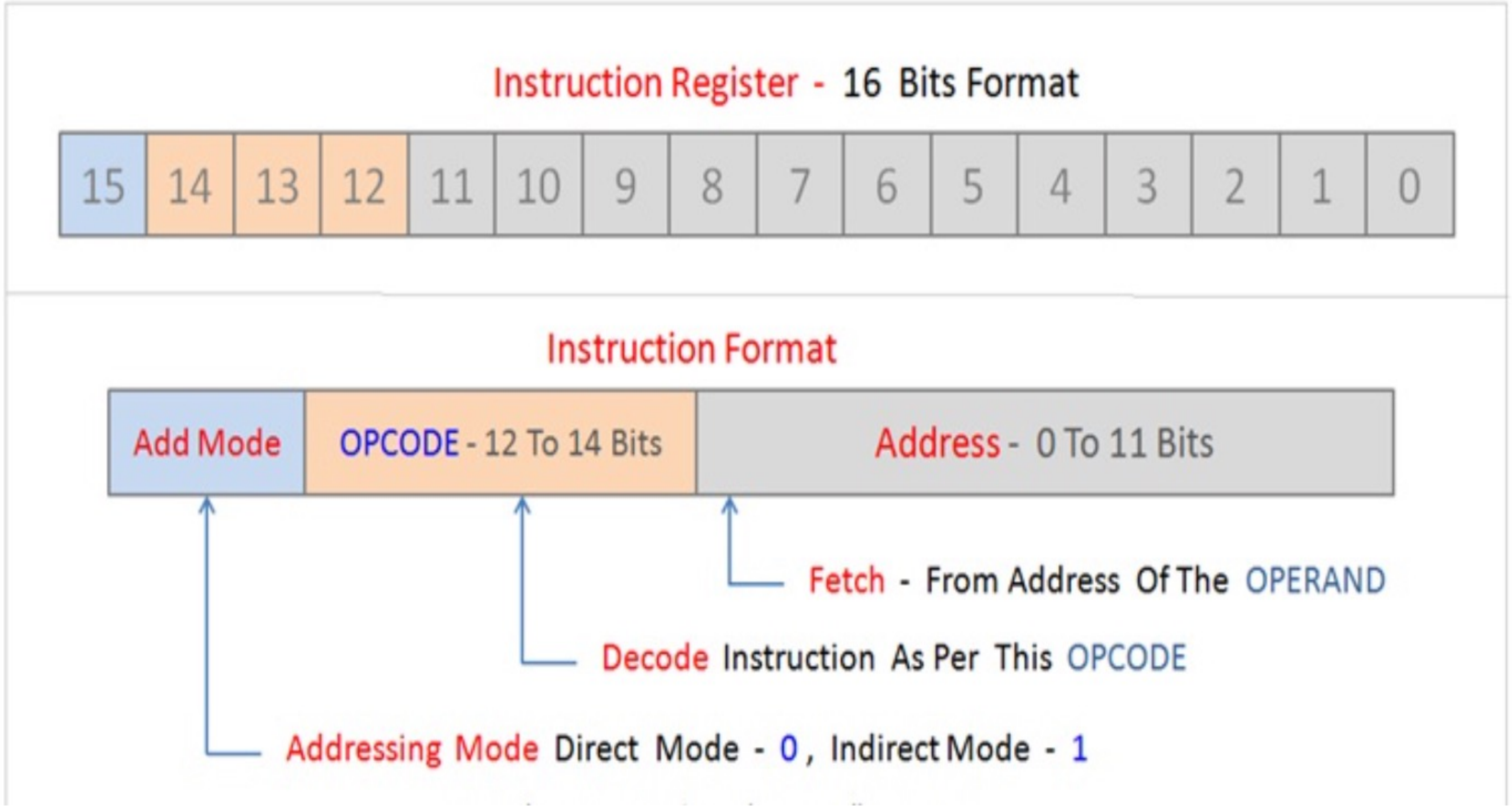


Immediate

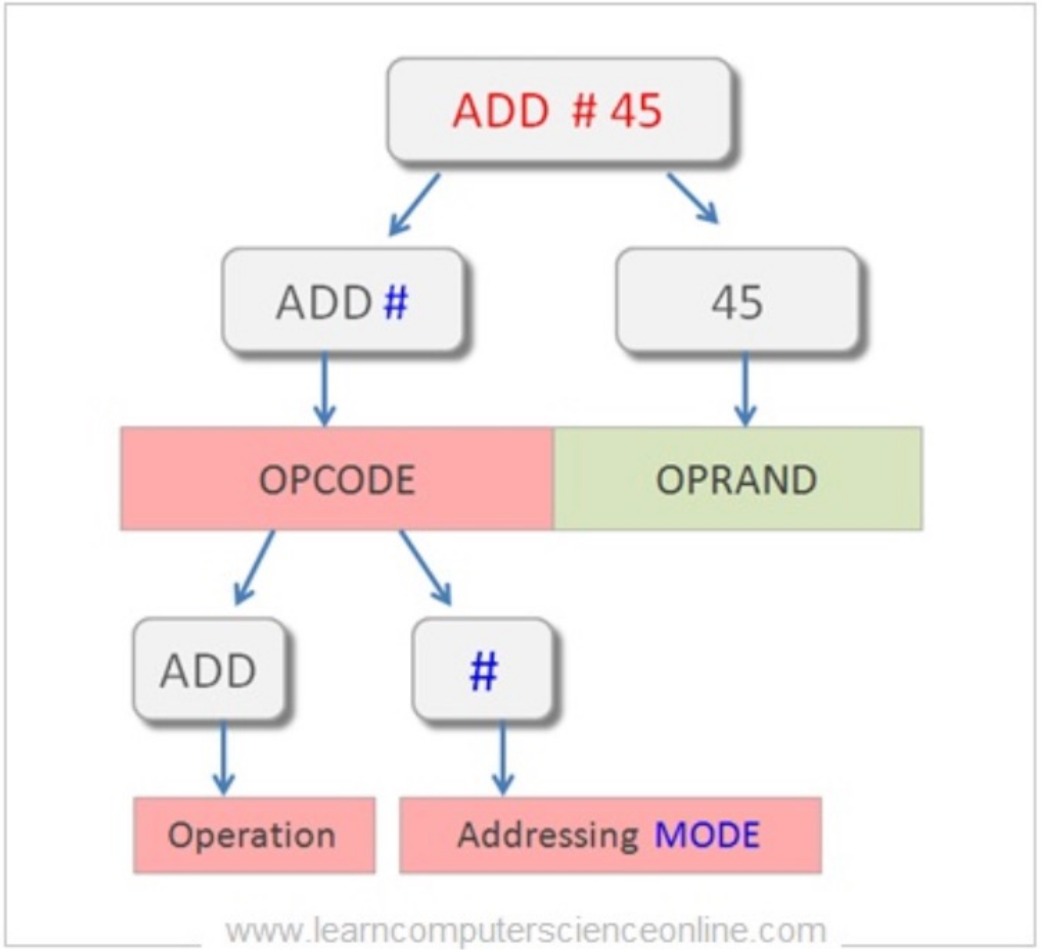


- Direct

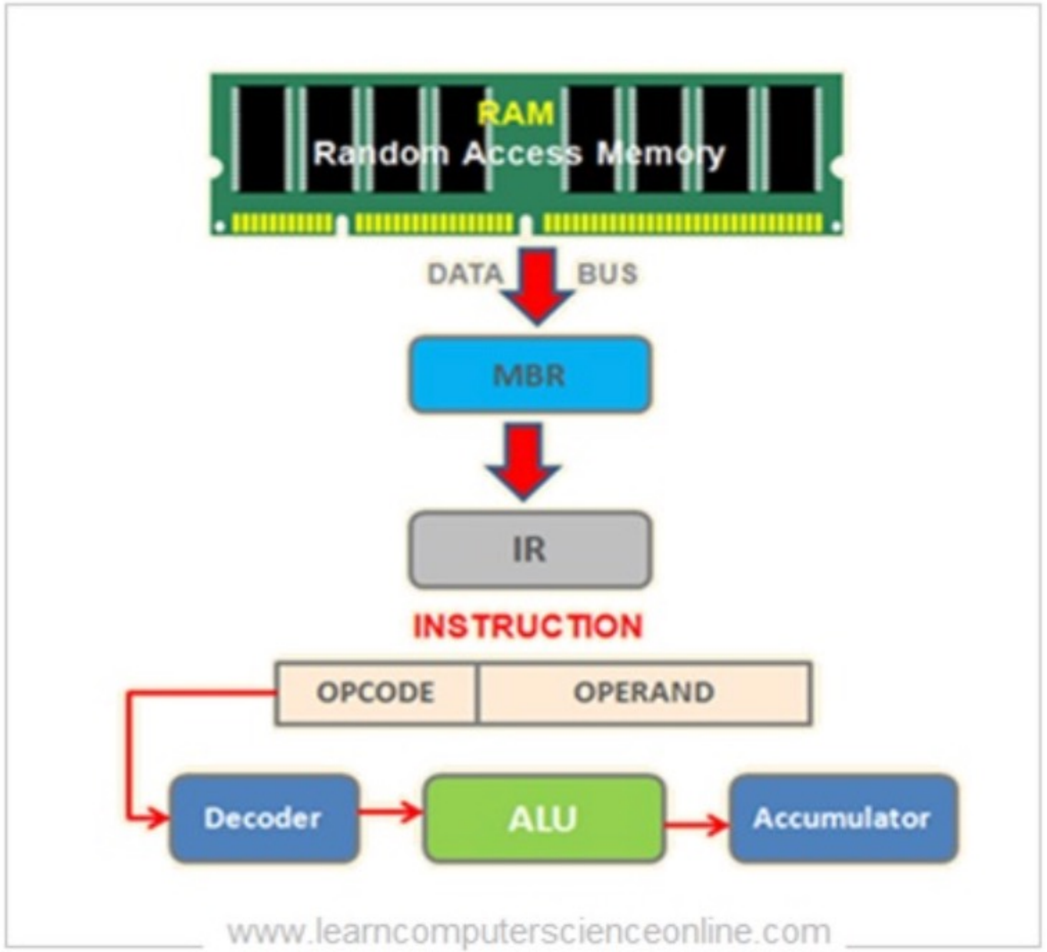



Instruction Register - 16 Bits Format
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Instruction Format
Add Mode
OPCODE - 12 To 14 Bits
Address - 0 To 11 Bits
Fetch - From Address Of The OPERAND
Decode Instruction As Per This OPCODE
Addressing Mode Direct Mode - 0 , Indirect Mode - 1

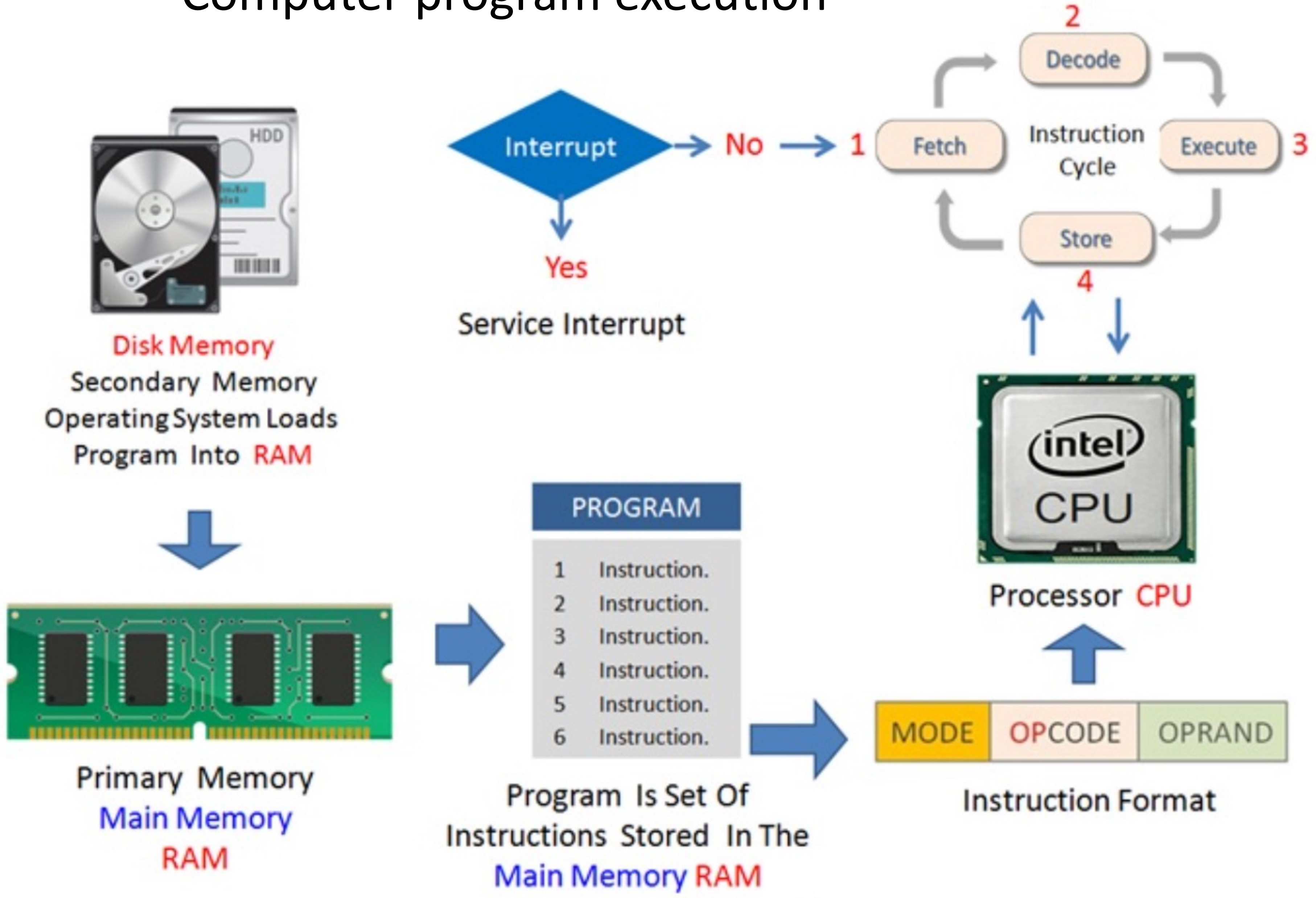
Program Source Code



Program Compilation



Computer program execution


Interrupt
No
Yes
Service Interrupt
1
Fetch
2
Decode
3
Execute
4
Store
Instruction Cycle
HDD
Disk Memory
Secondary Memory
Operating System Loads Program Into RAM
intel
CPU
Processor CPU
PROGRAM
1 Instruction.
2 Instruction.
3 Instruction.
4 Instruction.
5 Instruction.
6 Instruction.
Primary Memory
Main Memory
RAM
Program Is Set Of Instructions Stored In The Main Memory RAM
MODE
OPCODE
OPRAND
Instruction Format