Instruction Format

Instruction Codes

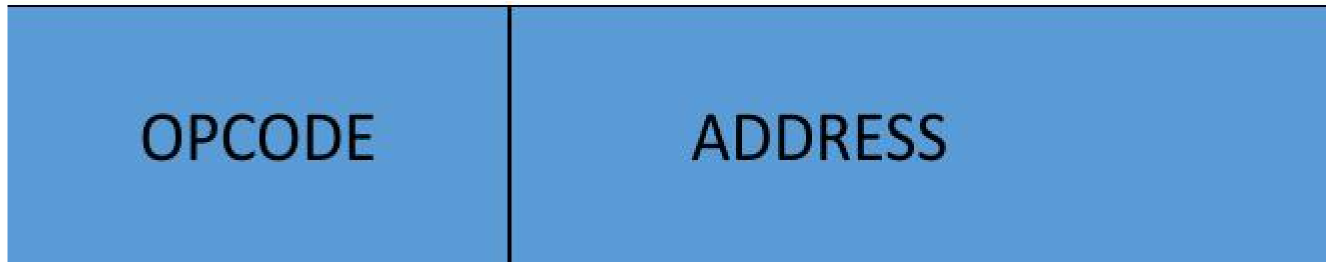
- The user of a computer can control the process by means of a program.
- A program is a set of instructions that specify the operations, operands, and the sequence by which processing has to occur.
- The data processing task may be altered by specifying a new program with different instructions or specifying the same instructions with different data.
- A computer instruction is a binary code that specifies a sequence of micro operations for the computer. Instructions codes together with data are stored in memory.

- The computer reads each instruction from memory and places it in a control register. The control then interprets the binary code of the instruction and proceeds to execute it by issuing a sequence of micro operations .
- Every computer has its own instruction set. The ability to store and execute , the stored program concept, is the most important property of a general purpose computer.

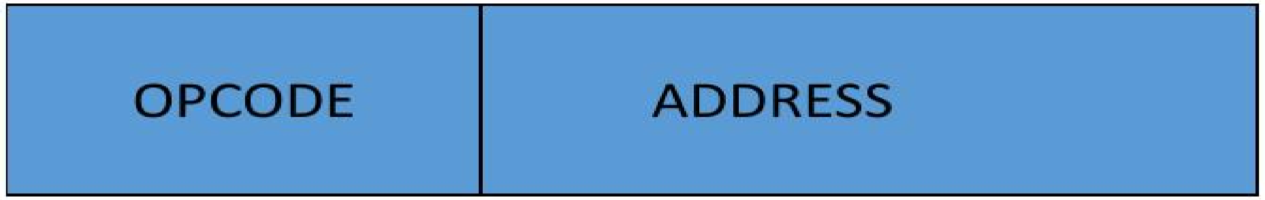
- An instruction code is a group of bits that instruct the computer to perform a specific operation.
- The most basic part of an instruction code is its operation part. The operation code of an instruction is a group of bits that define the operations like
- **Data Movement:** Memory/Register $\leftrightarrow$ Memory/Register.
- **Arithmetic:** Add/Subtract/Multiply/Divide
- **Logic:** Shift/Circulate/AND-OR-NOT
- **Control:** Conditional/Uncondotional Branch, Subroutine etc.
- **Other:** No Operation/Stack Pocessing

As an illustration, consider a computer with 64 distinct operations. One of them being an ADD operation. When this operation code is decoded in the control unit, the computer issues control signals to read an operand from memory and add the operand to a processor register.

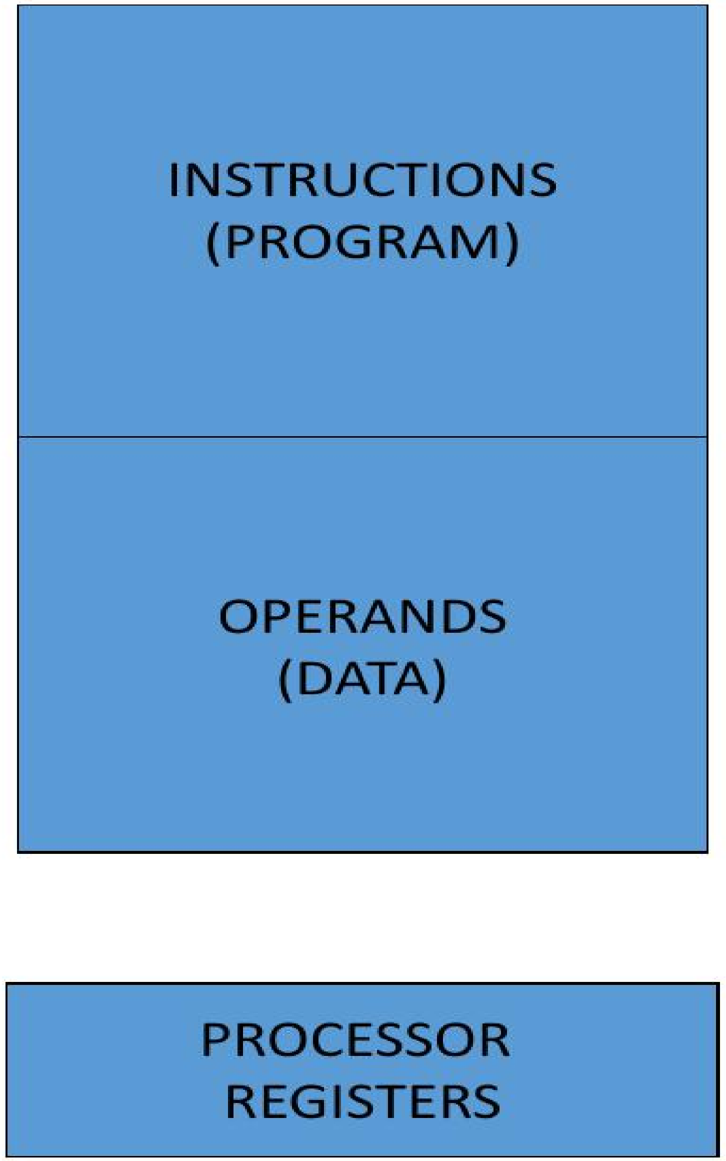
- Instruction Code specifies operation and registers where the operands are to be found.
- Instruction Code format with two parts



- Opcode specifies the operation to be performed
- Address tells the control where to find an operand in memory.



Instruction Format



STORED PROGRAM ORGANIZATION

Instruction Format

- On the basis of number of address instruction are classified as:
  - Zero Address Instructions
  - One Address Instructions
  - Two Address Instructions
  - Three Address Instructions
- Let’s use **X = (A+B)*(C+D)** expression to showcase the procedure.

Zero Address Instructions

- A stack based computer do not use address field in instruction.To evaluate a expression first it is converted to revere Polish Notation i.e. Postfix Notation.
  - Expression: X = (A+B)*(C+D)
  - Postfixed : X = AB+CD+*
  - TOP means top of stack
  - M[X] is any memory location

Zero Address Instructions

- The instruction format in which there is no address field is called zero address instruction
- In zero address instruction format, stacks are used
- In zero order instruction format, there is no operand

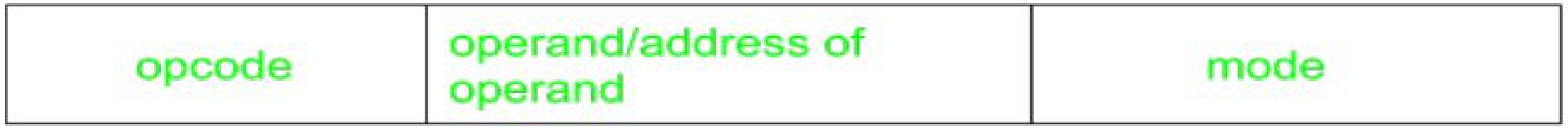
Zero Address Instructions

- **Expression: 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)
MUL	TOS ← (C + D) (A + B)
POP X	M [X] ← TOS

One Address Instructions

- This use a implied ACCUMULATOR register for data manipulation.
- One operand is in accumulator and other is in register or memory location.
- Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it.



opcode	operand/address of operand	mode

  - Expression: X = (A+B)*(C+D)
  - AC is accumulator
  - M[] is any memory location
  - M[T] is temporary location

One Address Instructions

- The instruction format in which the instruction uses only one address field is called the one address instruction format
- In this type of instruction format, one operand is in the accumulator and the other is in the memory location
- It has only one operand
- It has two special instructions LOAD and STORE

One Address Instructions

- **Expression: X = (A+B)*(C+D)**

LOAD A	AC ← M [A]
ADD B	AC ← A [C] + M [B]
STORE T	M [T] ← AC
LOAD C	AC ← M [C]
ADD D	AC ← AC + M [D]
MUL T	AC ← AC M [T]
STORE X	M [X] ← AC

Two Address Instructions

- The instruction format in which the instruction uses only two address fields is called the two address instruction format.
- This type of instruction format is the most commonly used instruction format.
- As in one address instruction format, the result is stored in the accumulator only, but in two addresses instruction format the result can be stored in different locations.
- This type of instruction format has two operands.
- Assembly language instruction – MOV R1, A; ADD R1, B etc.

Two Address Instructions

- **Expression: X = (A+B)*(C+D)**

MOV R1, A	$R1 \leftarrow M[A]$
ADD R1, B	$R1 \leftarrow R1 + M[B]$
MOV R2, C	$R2 \leftarrow M[C]$
ADD R2, D	$R2 \leftarrow R2 + M[D]$
MUL R1, R2	$R1 \leftarrow R1 \quad R2$
MOV X, R1	$M[X] \leftarrow R1$

Three Address Instructions

- The instruction format in which the instruction uses the three address fields is called the three address instruction format.
- It has three operands.
- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.
- Assembly language instruction – ADD R1, A, B etc.

Expression: X = (A+B)*(C+D)

ADD R1, A, B	R1 ← M [A] + M [B]
ADD R2, C, D	R2 ← M [C] + M [D]
MUL X, R1, R2	M [X] ← R1 R2

The advantage of the three-address format is that it results in **short programs** when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions **require too many bits to specify three addresses**.

Addressing Modes

The control unit of a computer is designed to go through an instruction cycle that is divided into three major phases:

1. Fetch the instruction from memory
2. Decode the instruction.
3. Execute the instruction.

The decoding done in step 2 determines the operation to be performed, the addressing mode of the instruction and the location of the operands.

Addressing Modes (contd...)

Instructions may be defined with a variety of addressing modes, and sometimes, two or more addressing modes are combined in one instruction.

- The mode field is issued to locate the operands needed for the operation.
- There may or may not be an address field in the instruction.
- The instruction may have more than one address field, and each address field may be associated with its own particular addressing mode.

1. **Implied Mode:** The operands are specified implicitly in the definition of the instruction in this mode. e.g. ***complement accumulator.***

   All register reference instructions that issue an accumulator are implied-mode instructions.

   Zero-address instructions in a stack-organized computer are implied mode instructions since the operands are implied to be on top of the stack.

2. **Immediate Mode:** The operand itself is specified in the instruction in this mode.

   The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.

   Immediate-mode instructions are useful for initializing registers to a constant value.

3 Register Mode:

In this mode, the operands are in registers that reside within the CPU.

4 Register Indirect Mode:

In this mode, the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

5 Direct Address Mode:

In this mode, the effective address is equal to the address part of the instruction.

The operand resides in memory and its address is given directly by the address field of the instruction.

6 Indirect Address Mode:

In this mode, the address field of the instruction gives the address where the effective address is stored in memory.

7 Relative Address Mode:

In this mode, the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

8 Indexed Addressing Mode:

In this mode, the content of an index register is added

to the address part of the instruction to obtain the effective address.

(The index register is a special CPU register that contains an index value.)

9 Base Register Addressing Mode:

In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register.