Unit - 03
Control Unit

Instruction set :- The operation of the processor is determined by the instruction that it executes known as machine instruction or computer instruction.

- The collection of different instructions that processor can execute is refers to as a processor instruction set.
- A computer instruction is a binary code that instruct the computer to perform a specific operation.

Types of instructions:-

The computer instruction can be classified into three categories -
1) Data Transfer Instruction
2) Data Manipulation Instruction.
3) Program Control Instruction

Data transfer Instruction :- possess transfer of data from one location to another without changing the binary information content.

Data manipulation Instructions are those that perform arithmetic, logic and shift operation.

Program Control Instructions provide decision making capabilities and change the path taken by the program when executed

DATA TRANSFER INSTRUCTION

Data transfer instruction move the data from one location to another without changing the source content.

- These instructions are used to bring the data to and from memory to register.

Example:-

① MOVE:- It denotes transfer of data from one register to another
② STORE:- Transfer from a processor register into memory
③ LOAD:- A transfer from memory to register.
④ Exchange:- It swaps information between the two registers or a register and memory.

⑤ <u>Input</u>:- Transfer from input terminal to processor register.

⑥ <u>Output</u>:- Tranfer from processor register to output terminal.

⑦ <u>PUSH</u>:- Transfer from processor register to top of the stack.

⑧ <u>POP</u>:- Transfer from top of the stack to the processor register.

⑨ <u>clear</u>:- Transfer of word zeroes to the destination.

⑩ <u>SET</u>:- Transfer of words once to the destination.

① MOV R_1 [MEM] ⇒ Memory to register transfer

② MOV [MEM], R_2 ⇒ Register to memory transfer

③ MOV R, R_2 ⇒ Register to register transfer

④ IN R, PORT ⇒ Read from an I/O PORT

⑤ OUT PORT, R_2 ⇒ Write from an I/O PORT

Data Manipulation Instruction :-

- Data manipulation instructions perform operation on the data.
- These can be divided into three types-

i) Arithmetic operation or instruction.

ii) Logical and bit manipulation instruction

iii) shift operation

1) Arithmatic instruction:- These instructions are used to perform arithmatic operation.

for Ex:-

a) Add:- It perform the addition of two operands.

b) subtract:- It perform difference of two operand.

c) Multiply:- It performs product of two operands.

d) Division:- It performs the division of two operands and calculate quotient and remainder.

e) Add with carry:- It compute the sum of two operands with carry.

f) Subtract with borrow:- It compute the difference with borrow.

g) Increment:- Add 1 to the operand.

h) Decrement:- subtract 1 from operand.

i) Negate (2's complement):- It changes the sign of the operand.

Ex-
```
Add  R1, R2
SUB  R1, R2
MUL  R1, R2
INC  R1
DEC  R1
```

2) Logical instructions:-

- These perform logical and bit manipulation instruction on the bits of the data.
- They perform binary operations on the string of bits stored in the register.

For Ex:- AND, OR, NAND, NOR, NOT, EX-OR, EX-NOR

Ex:- AND R_1, R_2 → It performs AND operation bitwise on R_1, R_2.

3) Shift micro operation:-

The operation in which the bits of the ward are moved to the left or right.

ex:- Logical shift left; Logical shift right,
Arithmetic shift left; Rotate left,
Arithmetic shift right; rotate right

Program Control Instruction

- A program control is a type of instruction when executed may change the address value in the program counter and causes the flow of control to be altered.
- These instructions specify conditions for altering the content of the program counter.

Ex. a) Jump(Branch):- Unconditional transfer, load PC with the specified address

b) Jump (Conditional):- Test specified conditions, either load PC with specifiedaddress or do nothing based on the conditions.

c) Return:- Replace content of the PC and other registers from the known locations.

d) Skip:- Implement to PC to skip the next instruction.

e) skip(conditional):- Test specified conditions either skip or do nothing based on the condition.

f) Halt:- Stop program execution.

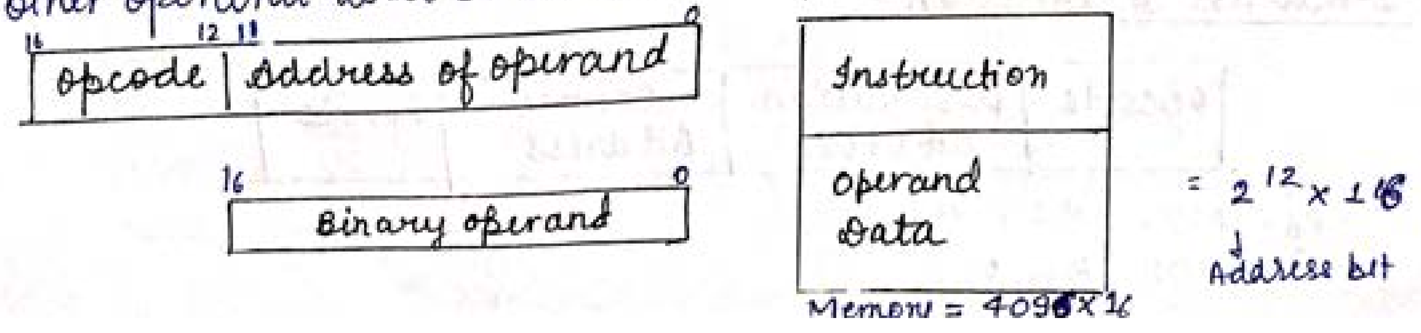
g) No operation:- No operation is perform but program execution is continue.

Stored Program Organization OR Processor Organization with accumulator OR Single Accumulator Organization

To organize a computer in this method processor has a register called accumulator and an instruction code format with two parts

Opcode	Address of operand

- The first part specify the operation to be performed and the second one specify an address of operand in memory.
- The other operand will be store in the processor register.



In this method the processor performs the operation specified by the opcode on the data specified/stored on the memory location and the content of the accumulator register.

Instruction format:

The most common fields on the instruction are

Opcode	Address of operand	Mode

- Operation field (opcode) specify the operation to be performed.
- Address field which contains the location of the operand (Register or memory location).
- Mode Field which specifies how operand is to be found.

Note:- Generally CPU organizations are -

1) General register organization
2) Stack Organization
3) Single Accumulator Organization

- An instruction is of various length depending upon the number of addresses It contains.
- On the basis of no. of addresses instruction can be classified as:-

1) 3- Address Instructions
2) 2- Address Instructions
3) 1- Address Instruction
4) Zero Address Instruction

1) 3- Address Instruction:- This format of the instruction has three address to specify a register or memory location.

Opcode	Destination Address	Source Address	Source Address	Mode

Ex- ADD R_1, R_2, R_3

2) 2-Address Instruction:-

Opcode	Destination Address	Source Address	Mode

Ex:- MOV R1, A
ADD R1, B

3) 1 - Address Instruction :-

Opcode	Address of operand	Mode

- It has only one address to specify a register or a memory location
- This uses an accumulator register to store one operand or intermediate result.

Ex:- Load A
ADD B

4) Zero Address Instruction:-

- This format does not contain any address field.
- The stack based computer do not use addressfield in the instruction to perform operation because the operation is performed on the two top items of the stack.

Ex:- $\boxed{\text{Opcode}}$

- ADD

Q) write the program by using -

a) 3- Address Instruction
b) 2- Address Instruction
c) 1- Address Instruction
d) 0- Address Instruction

$$X = (A+B)^* (C+D)$$

Ans) 3 Address Instruction

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

2) Address Instruction

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

3) One Address Instruction.

LOAD A	AC ← M[A]
ADD B	AC ← AC + M[B]
~~LO~~ 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

4) Zero Address Instruction

$$X = (A+B)*(C+D)$$

Prefix Exp ⇒ X = A B + C D + *

PUSH A	Top = A
PUSH B	Top = B
ADD	Top = A+B
PUSH C	Top = C
PUSH D	Top = D
ADD	Top = C+D
MUL	Top = (A+B)*(C+D)
POP X	M[X] = Top

Q2) $X = (A-B) + C * (D * E - F) / (G + H * K)$

or

$$\frac{(A-B) + C * (D * E - F)}{G + H * K}$$

Ans) 3 Address Instruction:-

```
SUB  R1, A, B
MUL  R2, D, E
SUB  R2, R2 F
MUL  R2, R2, C
ADD  R2, R1, R2
MUL  R3, H, K
ADD  R3, R3 G
DIV  X, R3 R2
```

2 Address Instruction

```
MOV  R1, A
SUB  R1, B
MOV  R2, D
MUL  R2, E
SUB  R2, F
MUL  R2, C
ADD  R1, R2
MOV  R3, H        ~~STORE X~~
MUL  R3, K
ADD  R3, G
DIV  R1 R3
                  STORE  X, R3
```

1 Address

LOAD A
SUB B
STORE T
LOAD D
MUL E
SUB F
MUL C
STORE T
ADD T
LOAD H
MUL K
ADD G
STORE T_1
LOAD T
DIV T
STORE X

Zero Address

AB − CDE × F − * + GHK * + /

PUSH A
PUSH B
SUB
PUSH C
PUSH D
PUSH E
MUL
PUSH F
SUB
MUL
ADD
PUSH G
PUSH H
PUSH K
MUL
ADD
DIV
POP

2) X = A*B + C*D + E*F

3 Address

MUL R_1, A, B
MUL R_2, C, D
MUL R_3, E, F
ADD R_2, R_2, R_3
ADD X, R_1, R_2

2 Address

MOV R_1, A
MUL R_1, B.
MOV R_2, C
MUL R_2, D
MOV R_3, E
MUL R_3, F
ADD R_2, R_3.
ADD R_1, R_2
MOV X, R_1

1 Address

LOAD A
MUL B
STORE T
LOAD C
~~LOA~~
MUL D
STORE T_1
LOAD E
LOAD F
STORE T_2
LOAD T
ADD T_1
ADD T_2
STORE X

Postfix.

AB*CD*+EF*+

Push A
Push B
MUL
Push C
Push D
MUL
ADD
Push E
Push F
MUL
ADD
POP