CS & IT ENGINEERING

COMPUTER ORGANIZATION AND ARCHITECTURE

Instruction & Addressing Modes

Lecture No.- 01

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Micro Operation

Topic Memory Access

Mov ⇒ Register to Register transfer

Load ⇒ memory to Register "

store ⇒ Register to memory "

Topics to be Covered

Topic Instruction

Topic ISA

Topic Types of Instruction

[MCQ]

#Q. Consider the following program segment. Here R1, R2 and R3 are the general-purpose registers.

	Instruction	Operation	Instruction Size (no. of words)
	MOV R1, (3000)	R1 ← M[3000]	2
LOOP:	MOV R2, (R3)	R2 ← M[R3]	1
	ADD R2,R1	R2 ← R1 + R2	1
	MOV (R3),R2	M [R3] ← R2	1
	INC R3	R3 ← R3 + 1	1
	DEC R1	R1 ← R1 – 1	1
	BNZ LOOP	Branch on not zero	2
	HALT	Stop	1

Assume that the content of memory location 3000 is 10 and the content of the register R3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

Assume that the memory is byte addressable and the word size is 32 bits. If an interrupt occurs during the execution of the instruction "INC R3", what return address will be pushed on the stack?

A 1005

B 1020

C ✓ 1024

D 1040

Solution

addresses for byte addressable mem.

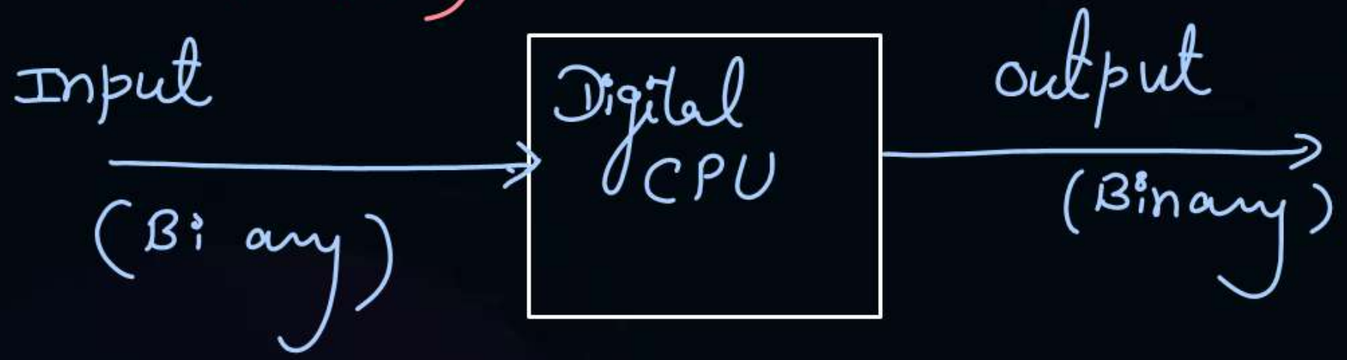
addresses for word addressable mem.

Inst^n size

			words	bytes
1000	1000	R1 ← M[3000]	2	8 bytes
1008	LOOP: 1002	R2 ← M[R3]	1	4 bytes
1012	1003	R2 ← R1 + R2	1	4 bytes
1016	1004	M [R3] ← R2	1	4
1020	1005	R3 ← R3 + 1	1	4
1024	1006	R1 ← R1 – 1	1	4
1028	1007	Branch on not zero	2	8
1036	1009	Stop	1	4

Topic : Digital Computer

(Instruction + data)



Topic : Instruction

```
#include<stdio.h>
void main()
{
int a, b, c;
printf("Enter 2 values: ");
scanf("%d %d", &a, &b);
c = a + b;
printf("Sum = %d", c);
}
```

Topic : Instruction

High-level lang. program

```
#include<stdio.h>
void main()
{
int a, b, c;
printf("Enter 2 values: ");
scanf("%d %d", &a, &b);
c = a + b;
printf("Sum = %d", c);
}
```

Programming statements

Compiler

Language Translation

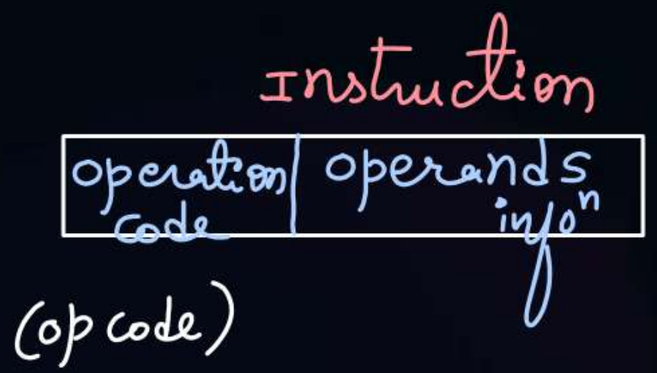
Low level lang. prog. or machine code or binary code or byte code

Instruction

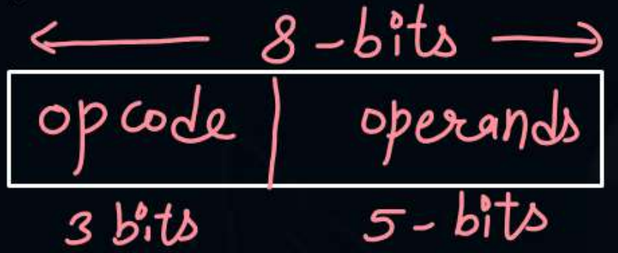
```
10111000
10000001
11110010
01010101
11110110
01010101
10001111
10100011
00111101
```

Topic : Instruction

A group of bits which instructs computer to perform some operation



example:- A CPU supports 8-bit instructions



Topic : Instruction

while CPU architecture design, scientist decides that the CPU can perform 8 various operations.

$2^3 \Rightarrow$ opcode = 3 bits

operations	opcode
Addition	000
subtraction	001
1's complement	010
.	.
.	.
.	.
.	.
.	.
.	111

Inst^n

3 bits

Topic : Instruction

ex:- Inst^{n}

4 bits

⇓

CPU can perform max. $2^4 = 16$ distinct type of operations

or

CPU can support max $2^4 = 16$ distinct instructions types.

Topic : ISA (Instⁿ set architecture)

Collection of all instⁿˢ supported by a CPU

---

size of instⁿ set
or
size of ISA

} ⇒ no. of instⁿˢ supported by CPU

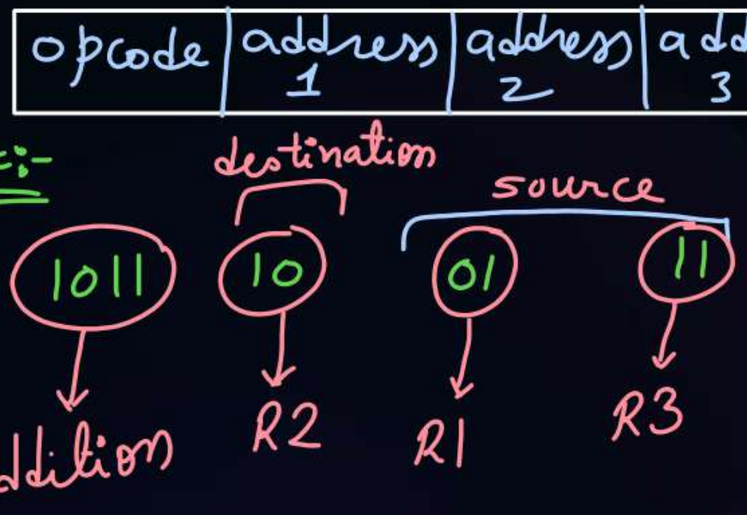
Topic : Types of Instruction

Based on operands info^n in inst^n

- 3-Address Instruction:
- 2-Address Instruction:
- 1-Address Instruction:
- 0-Address Instruction:

Topic : 3-Address Instruction

Max 3 addresses (→ for operands) can be specified within an instruction



$R2 \leftarrow R1 + R3$

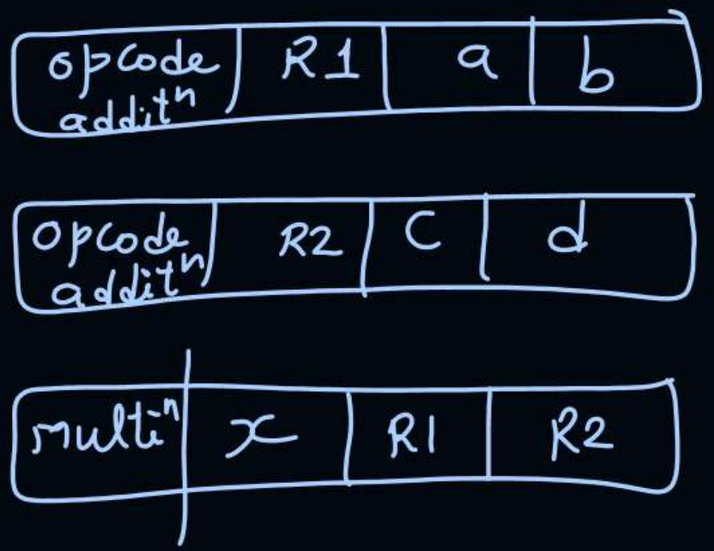
among 3 operands

- 2 operands ⇒ Source
- 1 operand ⇒ destination

$x = (a+b) * (c+d)$

a, b, c, d, x are memory operands

Using 3-address instns :-



$R1 \leftarrow a+b$

$R2 \leftarrow c+d$

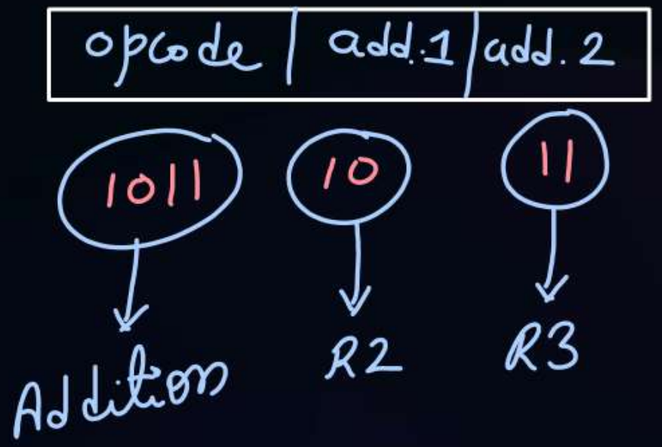
$x \leftarrow R1 * R2$

Topic : 2-Address Instruction

Max 2 addresses can be specified within an instruction

1 operand ⇒ only source

1 operand ⇒ source & destination both



$R2 \leftarrow R2 + R3$ [default]

or

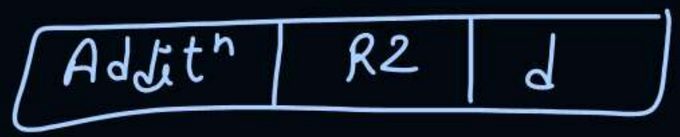
$R3 \leftarrow R2 + R3$

$$x = (a + b) * (c + d)$$

$R1 \leftarrow a$

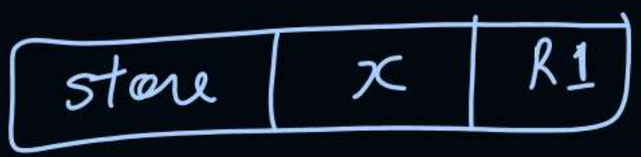
$R1 \leftarrow R1 + b$

$R2 \leftarrow C$



$R2 \leftarrow R2 + d$

$R1 \leftarrow R1 * R2$



$x \leftarrow R1$

Topic : 1-Address Instruction

Max 1 address can be specified within an instruction

2nd operands

1st operand ⇒ AC

ex:-

$AC \leftarrow AC + R2$

$$x = \underbrace{(a+b)}_{AC} * \underbrace{(c+d)}_{R1}$$

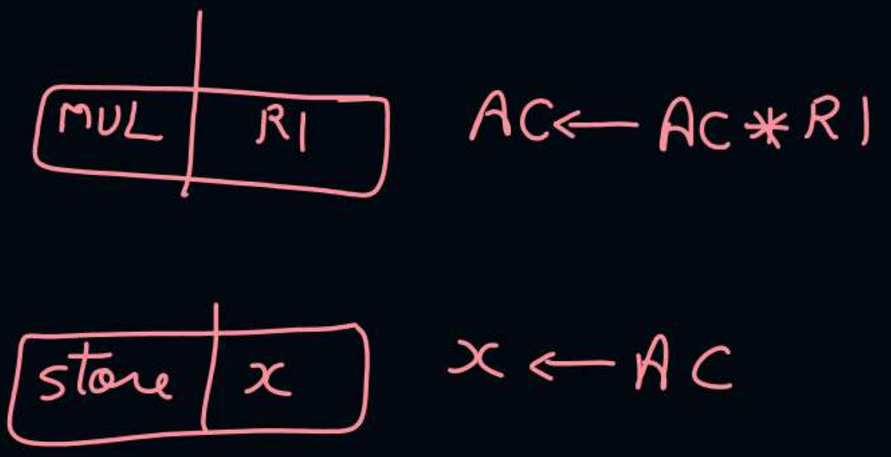
$AC \leftarrow c$

$AC \leftarrow AC + d$

$R1 \leftarrow AC$

$AC \leftarrow a$

$AC \leftarrow AC + b$



$AC \leftarrow AC * R1$

$x \leftarrow AC$

$$x = \underbrace{(a+b)}_{\cancel{R1}\ AC} * \underbrace{(c+d)}_{\cancel{AC}\ R2}$$

$AC \leftarrow a$

$AC \leftarrow AC + b$

$R1 \leftarrow AC$

$AC \leftarrow c$

$AC \leftarrow AC + d$

$R2 \leftarrow AC$

$AC \leftarrow R1$

$AC \leftarrow AC * R2$

$x \leftarrow AC$

Topic : 0-Address Instruction

No any address can be specified within an instruction

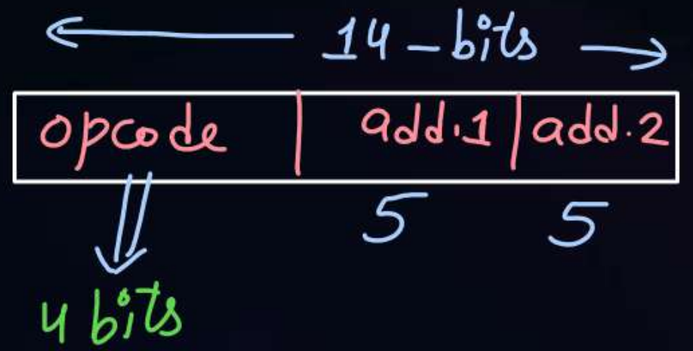
opcode

⇒ Every inst^{n} must have opcode part.

[NAT]

Ans ⇒ 16, 1

#Q. Consider a digital computer which supports only 2-address instructions each with 14-bits. If address length is 5-bits then maximum and minimum how many instructions the system can support?

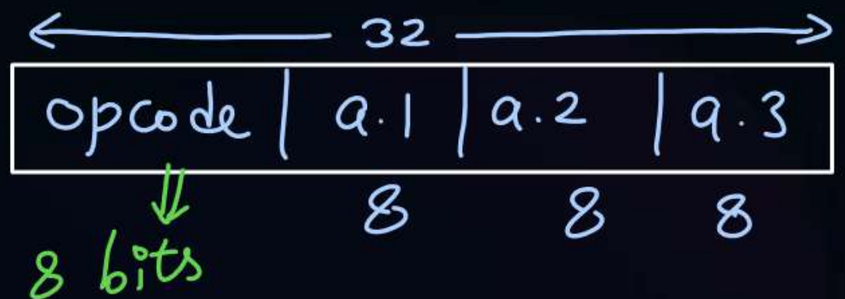


$$Max = 2^4 = 16$$

$$Min = 1$$

[NAT]

#Q. Consider a digital computer which supports only 3-address instructions each with 32-bits. If address length is 8-bits then maximum and minimum how many instructions the system can support?



$max = 2^8 = 256$

$min = 1$

2 mins Summary

- **Topic** Micro-operations
- **Topic** Instructions
- **Topic** Instruction Set Architecture
- **Topic** Types of Instructions
- **Topic** Opcode

Happy Learning

THANK - YOU