Types of Instruction

Essential registers for instruction execution

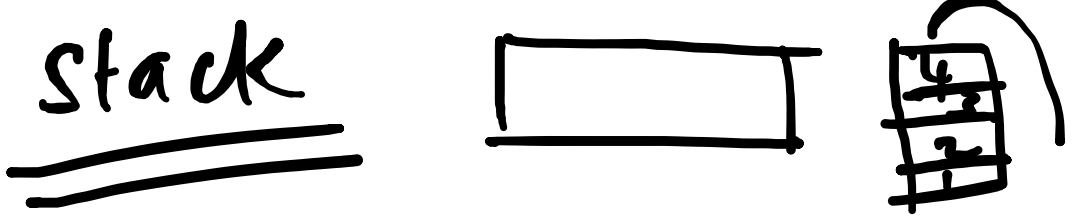
- **Program Counter (PC) :**
  It contains the address of an instruction to be executed next. The PC is updated by the CPU after each instruction is executed so that it always points to the next instruction to be executed. A branch or skip instruction will also modify the content of the PC.
- **Instruction 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 is analyzed.
- **Memory Buffer (or Data) Register (MBR or MDR) :**
  It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory. Contents of MBR are directly connected to the data bus.
- **Memory Address Register (MAR) :**
  It stores the memory locations of instructions that need to be fetched from memory or stored into memory. Contents of MAR are directly connected to the address bus.

Instruction types

- Based on the operation (opcode)
- Based on number of operands
- Based on type of operand
- Based on addressing modes

Based on the operation (OPERANDS)

- Data transfer instructions
- Arithmetic instructions
- Logic instructions
- Control transfer instructions
- Processor control instructions



- **Data Transfer instructions**

- Data transfer instructions are the instructions which transfers data in the microprocessor. They are also called copy instructions.

- MOVE, LOAD, EXCHANGE, INPUT, OUTPUT, PUSH, POP

- **Arithmetic instructions**

- Arithmetic Instructions are the instructions which perform basic arithmetic operations such as addition, subtraction and a few more.

- In 8085 microprocessor, the destination operand is generally the accumulator.

- ADD, SUBTRACT, INCREMENT, DECREMENT, CONVERT BYTE/WORD AND COMPARE

- **Logical instructions:**

  - AND, OR, exclusive OR, shift/rotate and test

- **Conditional instructions:**

  - conditional, unconditional, call subroutine and return from subroutine.

  - JUMP

- **Processor Control Instructions:**

  - flag manipulations

  - set, clear, complement carry flag

Based on the number of Operands

- Zero address instruction
- One address instruction
- Two address instruction
- Three address instruction

- Three Address instruction

  - It has one opcode and three address fields(OPERANDS). One address field is used for destination and two address fields for source.

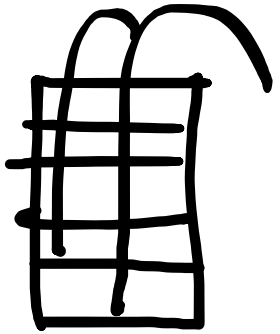
Opcode	Destination	Source 1	Source 2

Example:

```
X = (A + B) x (C + D)
```

Solution:

```
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 x R2
```



- Two Address instruction

  - It has one opcode and two address fields(OPERANDS). One address field is common and can be used for either destination or source and other address field for source.

Opcode	Destination / Source 1	Source 2

Example:

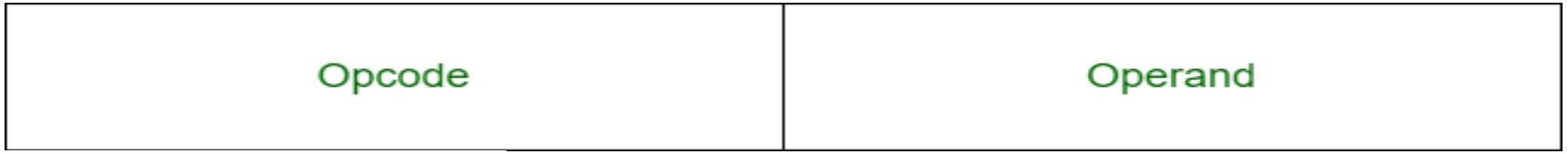
```
X = (A + B) x (C + D)
```

Solution:

```
MOV R1, A       R1 <- M[A]
ADD R1, B       R1 <- R1 + M[B]
MOV R2, C       R2 <- M[C]
ADD R2, D       R2 <- R2 + D
MUL R1, R2      R1 <- R1 x R2
MOV X, R1       M[X] <- R1
```

- One address instruction

    - It has only two fields. One for opcode and other for operand.



Example:

```
X = (A + B) x (C + D)
```

Solution:

```
LOAD A      AC <- M[A]
ADD B       AC <- AC + M[B]
STORE T     M[T] <- AC
LOAD C      AC <- M[C]
ADD D       AC <- AC + M[D]
MUL T       AC <- AC x M[T]
STORE X     M[X] <- AC
```

- **Zero address instructions:**

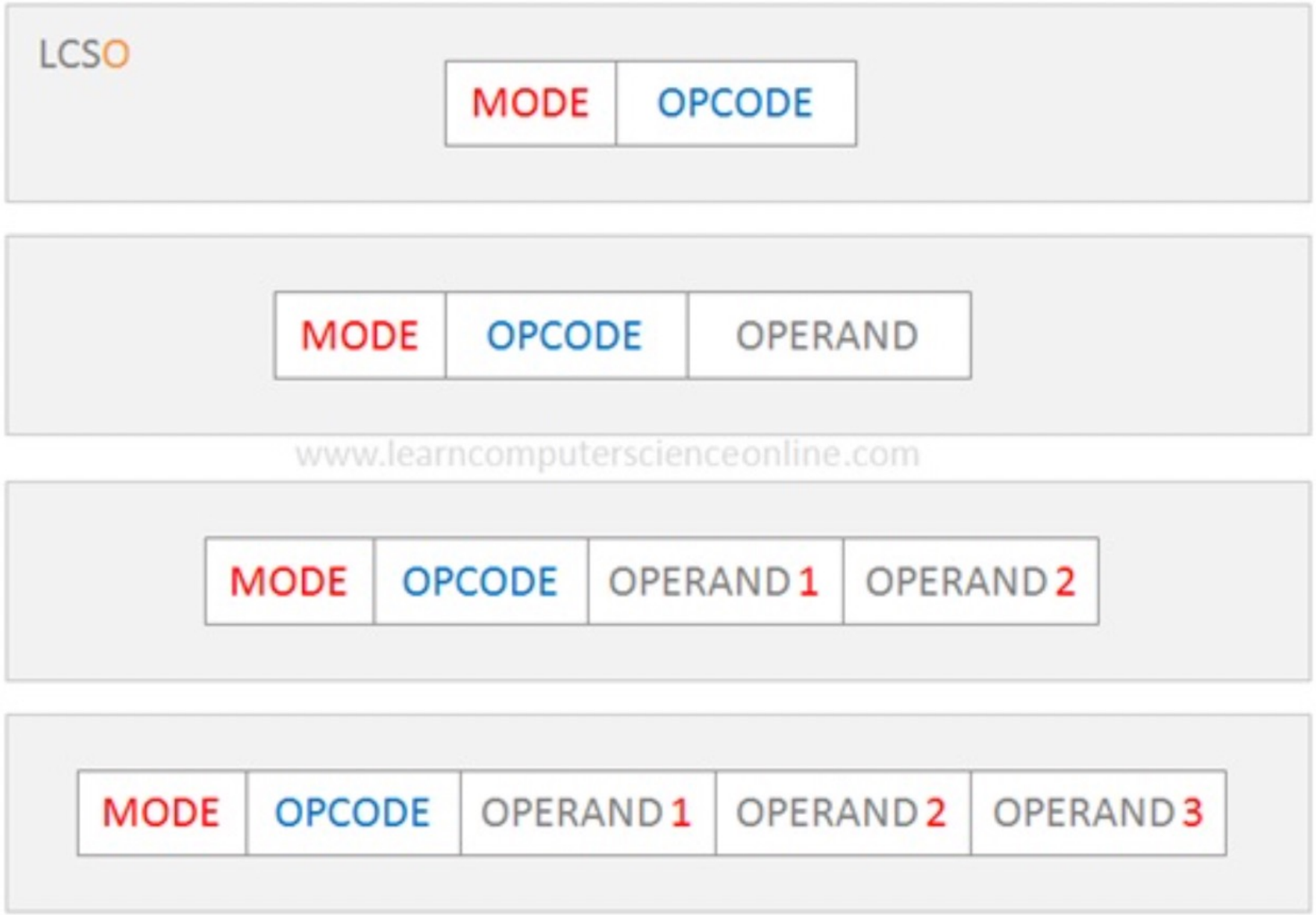
  - It has one opcode and no address fields.

Example:

```
X = (A + B) x (C + D)
```

Solution:

```
LOAD A      AC <- M[A]
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) x (A + B)
POP X       M[X] <- TOS
```



Zero address

One address

Two address

Three address

Based on addressing modes

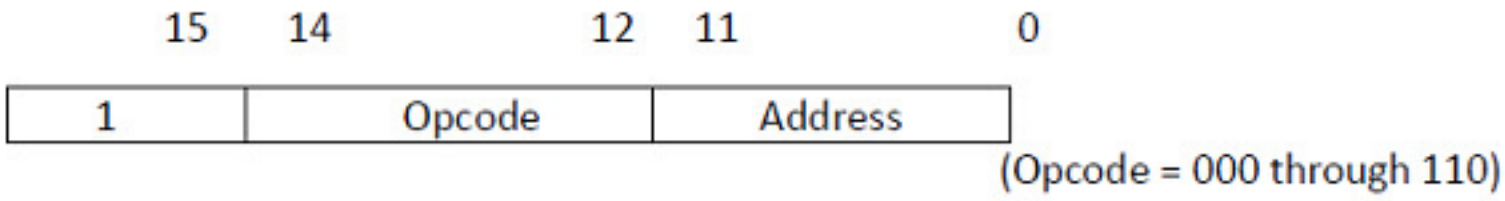
- Direct addressing instruction
- Indirect addressing instruction
- Immediate addressing instruction

Based on type of Operand

- Memory reference instruction
- Register reference instruction
- Input-output instruction

- Memory reference instruction

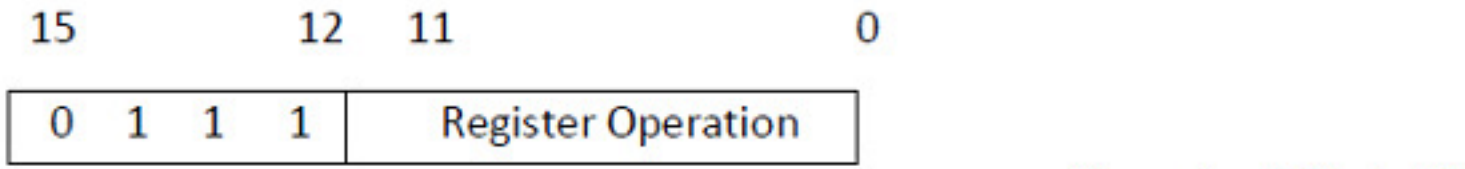
- A memory-reference instruction uses 12 bits to specify an address and one bit to determine the addressing mode I. I is the same as 0 for direct address and to 1 for indirect address.



(a) Memory Reference Instruction

- Register reference instruction

- The register reference instructions are identified by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction. It determines an operation on or a test of the AC register. An operand from memory is not required because the additional 12 bits are used to determine the operation or test to be implemented.

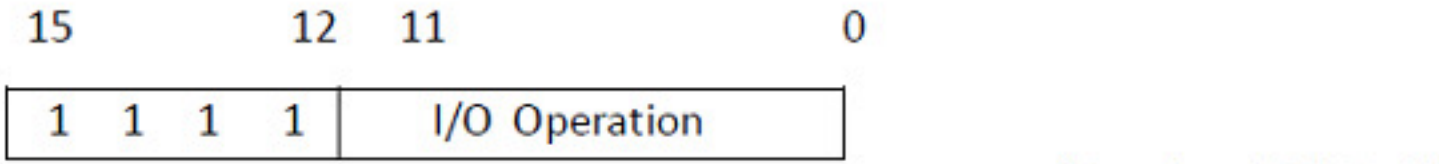


(Opcode = 111, I = 0)

(b) Register Reference Instruction

- Input/Output reference instruction

-An input-output instruction does not require a reference to memory and is identified by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits can determine the type of input-output operation or test implemented.



(Opcode = 111, I = 1)

(c) Input-Output Instruction

- The type of instruction is identified by the computer control from the four bits in positions 12 through 15 of the instruction. If the three opcode bits in positions 12 through 14 are not similar to 111, the instruction is a memory-reference type and the bit in position 15 is taken as the addressing mode I.
- If the 3-bit opcode is similar to 111, the control then examines the bit in position 15. If this bit is 0, the instruction is a register-reference type. If the bit is 1, the instruction is an input-output type.

Problem :

- 1) Example: translate the following C statement to assembly language

  x=u*(y+z);

  ADD R1,Y,Z

  MUL R2,R1,U

2)A processor has 40 distinct instructions and 24 general purpose registers. A 32-bit instruction word has an opcode, two registers operands and an immediate operand. The number of bits available for the immediate operand field is_____. (GATE CS 2016)

Soln:

40 distinct instructions → $\log_2 40 = 6$ bits are needed for opcode.

24 general purpose registers → $\log_2 24 = 5$ bits for register operand.

As, two register operands are required so, total 10 bits are needed for register operand field.

Suppose for immediate operand we require x bits.

Opcode	Register operand	Immediate operand
6 bits	10 bits	x bits

Opcode + Register operand + Immediate operand = 32

$6 + 10 + x = 32$

$\therefore x = 16$

Number of bits available for the immediate operand field (x) is 16 bits.