Assembly Language

- Its a low-level language (Not machine code) (Human understandable)
- Its a type of programming language which we can use to communicate with the hardware directly

Q- What are addressing modes?

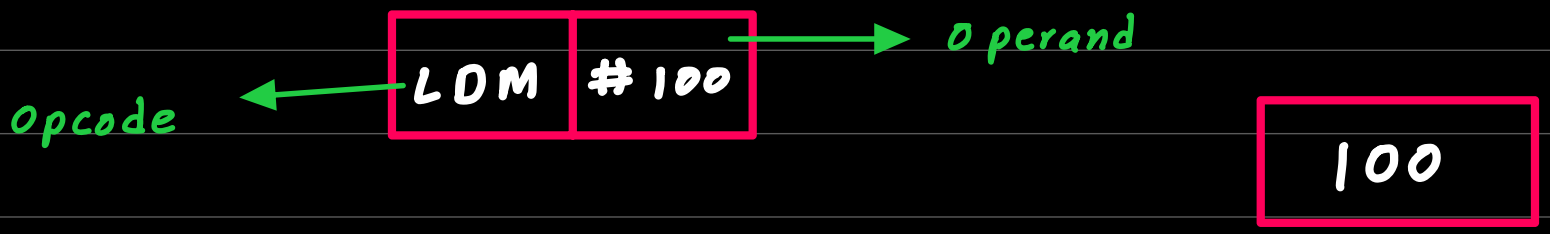
- Methods of accessing memory are known as addressing mode.

Addressing Modes

- Immediate addressing
- Direct addressing
- Indirect addressing
- Indexed addressing
- Symbolic addressing
- Relative addressing

Concept of accumulator: It is a place where the results are stored.

Immediate Addressing



LDM = Immediate addressing

- Load the value next to LDM instruction into accumulator

Direct Addressing

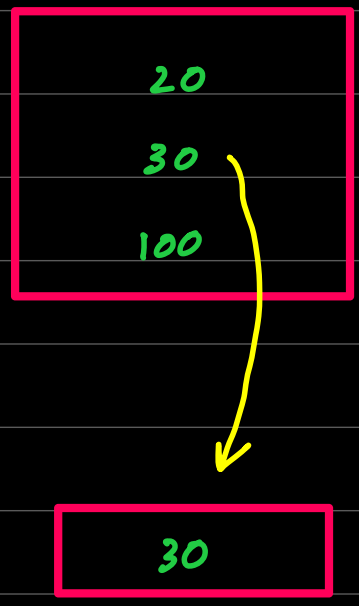
- Memory will be involved

- LDD <Address>

e.g: LDD 301

Location	Memory
300	20
301	30
302	100

- Step 1: Go to the given address
- Step 2: load the value in that address into ACC

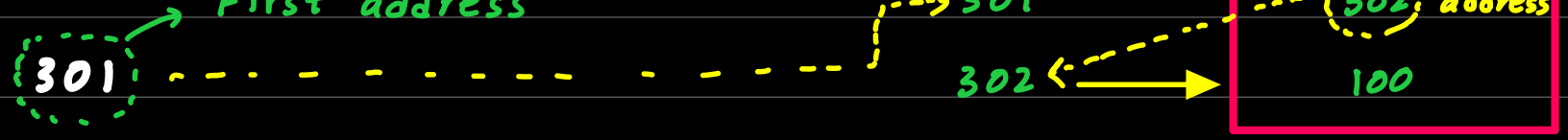


Indirect Addressing

Indirect Addressing = Lambi

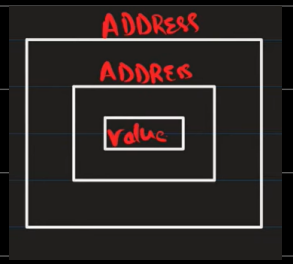
- 2 addresses involved

- LDI 301



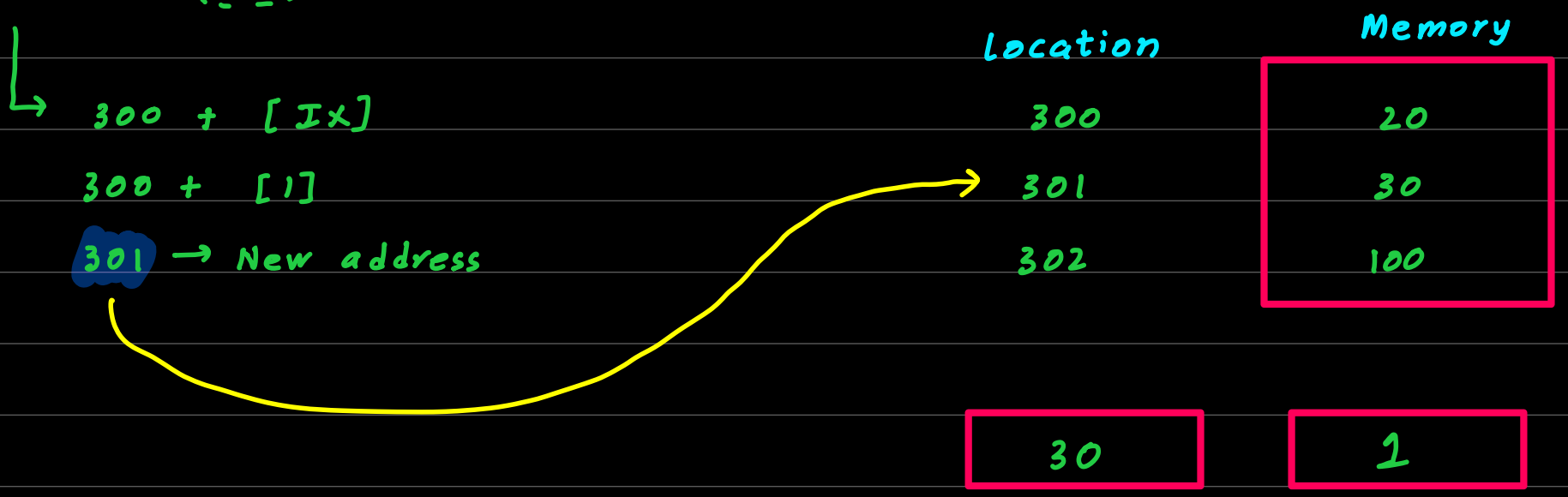
* If question comes about what is an addressing mode, tou explain the procedure of that addressing mode.

* No double jump



Index Addressing

- Calculation is required
- There will be a register known as index register (temporary memory location)
- LDX 300 → initial address



Instructions

STO	<address>	Store the content of accumulator at the given address.
ADD	<address> (can be a value as well with #)	Add the content of the given address to ACC. (Added value stored in Acc)
INC	<register>	Add 1 to the content of Acc or Ix
DEC	<register>	Subtract 1 to the content of Acc or Ix.
CMP	<address>	Compare content of ACC with content of address
JPE	<address>	Following a compare instruction, jump to address if comparison was true.
JPN	<address>	Following a compare instruction, jump to address if comparison was False
JMP	<address>	Jump to the given address
OUT		Output the character represented by the ASCII code in ACC.
END		End of assembly code

Symbolic Addressing

Location Memory

AB (301 30)

↘ Label

Symbol Table

Symbolic Address	Absolute Address
AB	301

Q- Explain how Assembler makes entries to the symbol table?

- The assembler scans the assembly language instructions in sequence.
- When it meets a symbolic address, it checks if that symbol is in the symbol table
- If not, it adds it to the symbol table in the symbolic address column.
- If it is already in the symbol table, it checks if absolute address is known
- If the absolute address is known, it is entered in the appropriate cell.
- If the absolute address is unknown, leave it as unknown

One - Pass Assembler

- It puts the machine code instructions into the computer memory

* One - pass assembler converts source-code into machine code and loads it into the memory.

Two - Pass Assembler

- Needs to scan the source program twice, so they can replace labels in the assembly program with memory addresses in the machine code program.

* Machine code is produced in the 3rd Pass.

Absolute Addressing: The operand is a numeric address

Symbolic Addressing: The operand is a symbol/label and that represents memory location

Relative Addressing: Form the address by adding the given number to the base address and then load the value on that address.