The three memory addressing modes of the LC-3

First, when you "label" a line of Assembly Language code, that label is just an *alias for the memory address* in which the intruction or data word is stored when the program is running. Consider the following code:

```
          .ORIG x3000    ; Load program into memory starting at address x3000
                         ; each instruction assembles to one 16-bit word,
                         ; i.e. a single address in memory
label1    first line of code
          second line of code
label2    third line of code
```

So label1 just means "address x3000", and label2 means "address x3002", etc.

Note that in the following instructions, the label is actually stored as part of the assembled instruction, as a 9-bit two's complement number. This means that these instructions can work only if the label is less than about 256 lines before or after the instruction itself.

SO:

LEA: Load Effective Address:
`LEA R1, label`
translates "label" into the address that it stands for, and stores that value in R1.
No access is made to memory
(so this is really a "pseudo" data movement instruction - no data is actually moved).

LD: "direct" memory addressing mode:
`LD R1, label`
accesses memory address "label", and copies the value stored there into R1
One memory access.

LDI: "indirect" memory addressing mode:
`LDI R1, label`
accesses memory address "label" & obtains the value stored there;
interprets that value as yet another address (let's call it p1);
accesses p1, and copies the value stored there into R1.
Total of two memory accesses.

LDR: "relative" (or "register") memory addressing mode:

```
LDR R1, R6, offset  ; where "offset" is a 6 bit 2's complement number
                    ; i.e. in range -32 to +31
```

gets value in R6 and adds offset to it;
interprets that value as an address (let's call it p2);
accesses p2, and copies the value stored there into R1.
One Register access, one memory access.

As you can imagine, LEA is often used in conjunction with LDR: first we use LEA to store an address (e.g. the first address of an array) to a register, then we use LDR to get the value stored at that address; then we increment the register (i.e. point to the next address) & repeat.

LDI and LDR are generally used to access locations in memory too far away to be reached by LD *(remember the +/- 256 lines restriction)*.