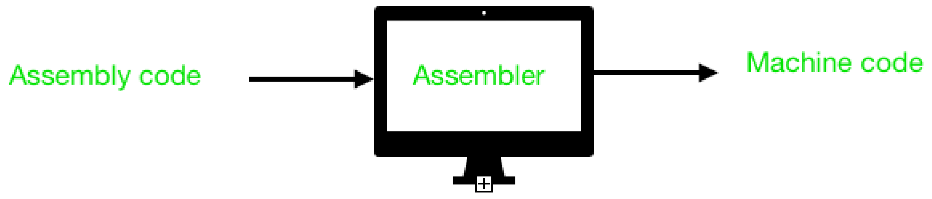
# Introduction of Assembler

**Assembler** is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader.



It generates instructions by evaluating the mnemonics (symbols) in operation field and find the value of symbol and literals to produce machine code. Now, if assembler do all this work in one scan then it is called single pass assembler, otherwise if it does in multiple scans then called multiple pass assembler. Here assembler divide these tasks in two passes:

- **Pass-1:**
  1. Define symbols and literals and remember them in symbol table and literal table respectively.
  2. Keep track of location counter
  3. Process pseudo-operations
- **Pass-2:**
  1. Generate object code by converting symbolic op-code into respective numeric op-code
  2. Generate data for literals and look for values of symbols

## Introduction to working of Assembler

Working of Pass-1:

Define Symbol and literal table with their addresses.
Note: Literal address is specified by LTORG or END.

Working of Pass-2:

Pass-2 of assembler generates machine code by converting symbolic machine-opcodes into their respective bit configuration(machine understandable form). It stores all machine-opcodes in MOT table (op-code table) with symbolic code, their length and their bit configuration. It will also process pseudo-ops and will store them in POT table(pseudo-op table).

Various Data bases required by pass-2:

1. MOT table(machine opcode table)
2. POT table(pseudo opcode table)
3. Base table(storing value of base register)
4. LC ( location counter)