Embedded firmware development languages:

Firmware can be in two ways.

① Target Processor/Controller specific language (low level or) assembly level language) Ex:- ALP

② Target Processor/Controller independent language (High level language) Ex:- C, C++, Java, Python etc.

① Assembly level language programming (low level):

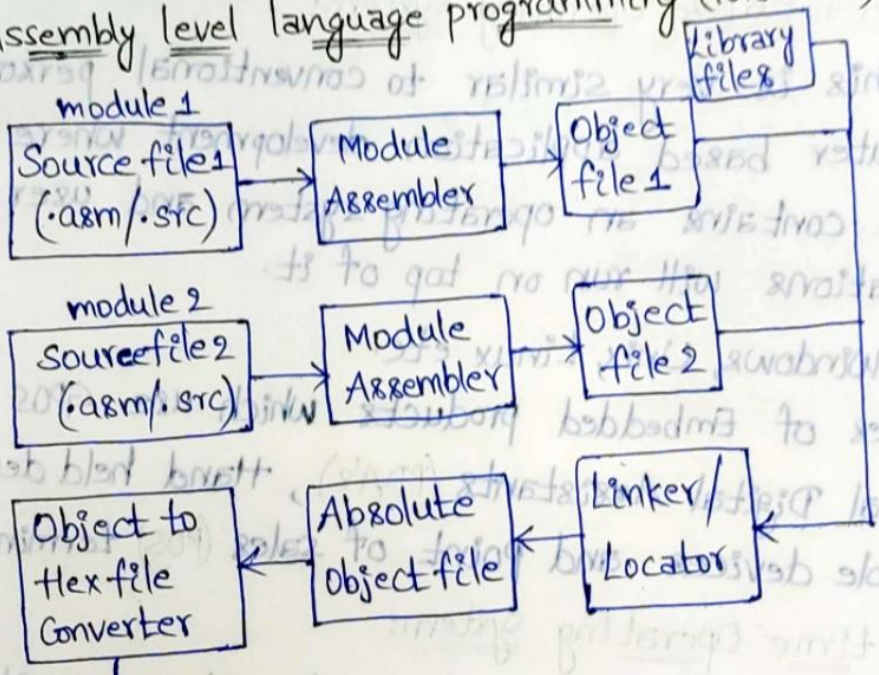

fig: Assembly language to Machine language Conversion Process.

* Assembly language is the human readable notation of "Machine language". They use specific symbols called "Mnemonics".
* A machine level language is processor understandable language. Processor deals with only 1's and 0's.
* Machine language and Assembly language are

processor/controller dependent: A program written for one processor/controller family will not work with others.

* General format for Assembly level language is Opcode followed by operands. Opcode specifies what to do by processor/controller and operands provide the data and information required to perform action specified by opcode.

Some of the opcode is implicitly contains the operand and in such case no operand is required.

Ex:- Consider 8051 ASM instruction `MOV A, #30`
(opcode: MOV A, operand: #30)

This moves the value 30 into Accumulator 'A'. (single)
The same instruction written in Machine language is expressed as 01110100 00011110
(opcode: 01110100, operand: 00011110)

Ex-2: Inc A :: This increments accumulator by value '1'.
(opcode: Inc, no operand: A)

Each line of assembly level language is split into 4 fields.

Label (1) Opcode (2) Operand (3) Comments (4)

Label: (optional) It represents memory location, address of a program, subroutine, code portion etc. They can contain numbers from 0 to 9 and '_' (under-score). They are always suffixed by 'colon' and begin with valid character.

Ex:- SUBROUTINE FOR GENERATING DELAY

DELAY PARAMETER PASSED THROUGH REGISTER R1

RETURN VALUE NONE

REGISTERS USED : R0, R1

```
DELAY: MOV  R0, #255  ; load R0 with 255
       DJNZ R1, Delay ; Decrement R1 & loop till R1=0
       RET            ; Return to calling program
```