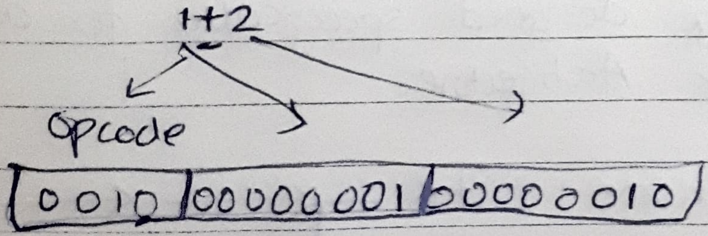
② Low level vs High Level languages

- Low level
  - → Machine language → very close to hardware
    So low level
    Machine oriented
    010001
    Adv: Execution is fast
    Disadv: • Not understandable by humans
    • Programs written in ML are system dependent → • So cannot run in two different machines i.e not portable
  - → Assembly language



M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	FEB
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	•	•	'16

Assembly language: In this language, we can use symbols / numbers i.e mnemonics

ADD - - - $\boxed{1+2}$

SUB

MOV → to move data from one register to another register

∴ Readable by Human. But still it is Low Level lang. ∵ of Very Strong Binding with ML

$$\begin{tikzcd}
\text{Program: ADD, SUB, MOV} \arrow[d] & \boxed{\text{High Level}} \\
\boxed{\text{Assembler}} \arrow[d] & \boxed{\text{Assembly Lang}} \arrow[u] \\
\boxed{\text{M/c Caode}} & \boxed{\text{Machine lang}} \arrow[u] \\
 & \boxed{\text{Hardware}} \arrow[u]
\end{tikzcd}$$

Disadv: Cannot be directly executed on Machine

∴ It is lower than ML language

Also dependent on machine / system
(Not portable)

Every Assembly language designed specifically for a specific Computer Architecture.

To Overcome This

↓

High Level Languages

↓

C, C#, Java, Python, PHP, Perl

Appointment | Notes | Work to do

High Level Languages:- [PORTABLE]

↳ not machine dependent
But Sometimes OS dependent
Ex:- Windows

↳ very close to humans
far from machine code.

⟹ Memory Addresses, Registers, Machine Code → Low Level
Variables, functions, loops, Mathematical expr → High Level
Aphabets, Mathematical notations.
↳ More like English Language
↳ Higher Abstractions
↳ Need not to go much deeper → like → CPU Architecture Specification.

↳ Easy to remember

- Program written in high level → source code.
- Compiler → take src code & convert it into object file ↓

obj file ← Machine code

↓

But only after Complete/ WHOLE Program → it converts.
↓
Machine will execute that program

- Interpreter
  ↳ reads line by line
  ↳ Not Complete
  ↳ Conversion to Machine Code & execution runs parallely

Read ref High Level → Convert it into M/c code → slower

So, Basically we focus on usability of Program not optimal program efficiency