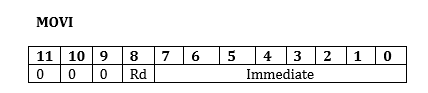
The assembler works as follow

For every assembly code input it generate a valid hexadecimal code which can be copy and pasted in the instruction memory

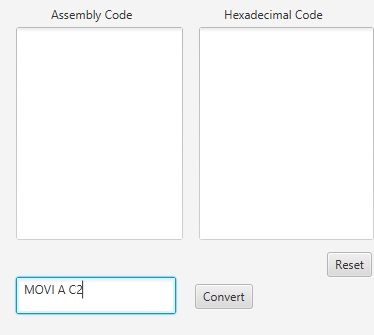
The logic behind the assembler



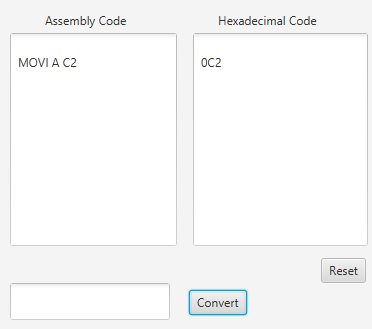
For this instruction depending on the value of Rd the assembler generate equivalent hexadecimal.

For instance

Step-1 placing assembly code in the text filed and press the convert button



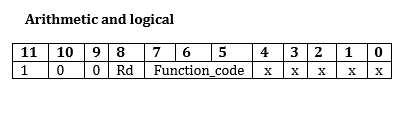
Step-2the hexadecimal and its equivalent Assembly code is placed in the Text Areas



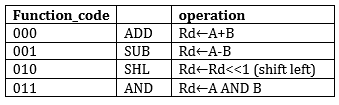
Note:

Here if the value of Rd is A then we have the first 4 bit of the instruction as 0000 which is equivalent to zero in hexadecimal and the immediate value is placed next to it then this hexadecimal code /instruction can be fetched and executed in our CPU in its own Datapath. And if Rd is B the we have the first 4 bit as 0001 which is equivalent to 1 in hexadecimal placing it with the immediate value we have our instruction that can be loaded to the instruction memory .

For Arithmetic instructions



Depending on the value of Rd the first 4 bit will be either 1000 or 1001 which is equivalent to 8 and 9 respectively in hexadecimal so the next bit is determined from the functional code



**And taking the last bit as zero or one which have no effect since our CPU design will handle that**

**So assuming the last bit to be zero**

**Functional code hexadecimal**

**ADD 0000 0**

**SUB 0010 2**

**Shift 0100 4**

**And 0110 6**

**and the last bit in our hexadecimal code can be determined set either zero or one with no problems**

**for our case setting it to be zero and the instruction in hexadecimal have the following format**

**ADD A--------800 ADD B---------900**

**SUB A---------820 SUB B---------920**

**SHL A--------840 SHL B --------940**

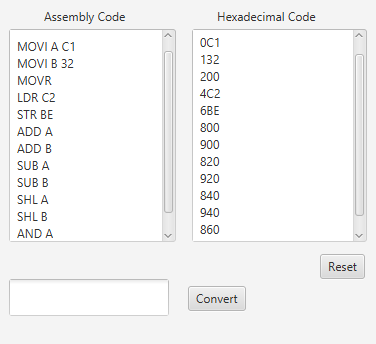
**AND A -------860 AND B-------960**

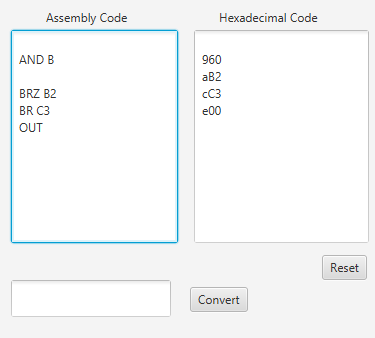
**With the same analogy we have**

**LDR Address--------- 4address ,STR Address - -------6address ,BR Address----------Caddress ,**

**BRZ Address--Aaddress**

**EXAMPLE OF ALL instruction sets and its Hexadecimal Code using Assembler**





**Note :**

**The instruction can be written either in upper or lower**

**And the instruction format is as written above space between each opcode and operand result Location and address or the immediate values in move instruction.**