Assembler

Compiled Instruction:

LNG_BIT	OPCODE	DONT CARE	RDST	RSRC1	RSRC2
1-bit	5-bits	1-bit	3-bits	3-bits	3-bits

Compiler Steps:

A header is added to the top of the compiled file to match the *.mem file generated with modelsim

1. The assembler first matches all lines that start with '.' of word (non-empty / non-commented) lines.
   `/^(\w|\.).*/` asm
   ```
   # this is a full line comment                       (not-matched)
   EMPTY-LINE                                          (not-matched)
   .org 0 #this is another command                     (matched)
   ADD R5, R1, R7   # this is a commented section      (matched)
   ```
2. After that the assembler splits matched lines on '#' and takes anything before the '#' (commented section) and strip to remove extra white spaces

   assembly `ADD R5, R1, R7   # this is a commented section`
   would be transformed to

   'ADD R5, R1, R7'

   `op_code = ''`
3. Then the assembler splits on ',' and strips extra spaces to extract the tokens from the instruction lines

   python `['ADD R5', 'R1', 'R7'] op_code = ''`
4. Then the first token is split into 2 tockens

   python `['ADD', 'R5', 'R1', 'R7'] op_code = ''`
5. Then the instruction is replaced with its OP-CODE and if 'Rdst' is not found the assembler adds 3 Xs to account for the missing Rdst token

   `op_code = '{INSTRUCTION-OPCODE}X{XXX if not Rdst}'`
6. Finally the assembler adds the memory location number and first bit 'LNG_BIT' and adds the register numbers and Xs to make the full instruction 16 characters as required

Instruction Set:

```
{
    INC  = "00000",
    ADD  = "00001",
    IADD = "00001",
    SUB  = "00010",
    DEC  = "00011",
    AND  = "00101",
    OR   = "00110",
    NOT  = "00111",
    MOV  = "01000",
    LDM  = "01001",
    LDD  = "01010",
    POP  = "01011",
    IN   = "01100",
    STD  = "10000",
    PUSH = "10010",
    CALL = "10011",
    RET  = "10101",
    RTI  = "10111",
    NOP  = "11000",
    JZ   = "11001",
    JC   = "11010",
    SETC = "11100",
    CLRC = "11101",
    OUT  = "11110",
    JMP  = "11111"
}
```

Registers:

```
{
    "R0": "000",
    "R1": "001",
    "R2": "010",
    "R3": "011",
    "R4": "100",
    "R5": "101",
    "R6": "110",
    "R7": "111",
}
```