PROBLEM STATEMENT

VICHARAK TASK

- Choose an 19-bit architecture (Instruction size will be 19-bit)
- Define a specialized instruction set tailored to a specific application or a set of applications (eg. signal processing, cryptography),
- Create a detailed architectural specification, including pipeline stage, register file, ALU, and memory interfaces.

Min Ist to be added:

| ADD r1,r2,r3 | SUB r1,r2,r3 | MUL r1,r2,r3 | DIV r1,r2,r3 | INC r1 | DEC r1 |

- Logical Instructions:

| AND r1,r2,r3 | OR r1,r2,r3 | XOR r1,r2,r3 | NOT r1,r2 |

- Control Flow Instruction:
  1) JMP addr : Jump to the address given | PC = addr
  2) BEQ r1,r2,addr : Branch to the address if values in r1 and r2 are equal. | if(r1==r2) PC=addr
  3) BNE r1,r2,addr : Branch if not equal. | if (r1!=r2) PC = addr
  4) CALL addr : Call a subroutine at that address. | stack[SP] = PC + 1; SP = SP-1; PC = addr
  5) RET : return from a subroutine. | SP = SP + 1; PC = stack[SP]

- Memory Access Instructions
  1) LD r1, addr : Load from address to r1 | r1 = memory[addr]
  2) ST addr, r1 : Store r1 at the address | memory[addr] = r1

- Custom Instructions
  1) FFT r1,r2: Do fft at addr r2 & store at address r1.
  2) ENC r1,r2: Encrypt data at addr r2 & store at address r1.
  3) DEC r1,r2: Decrypt data at addr r2 & store at address r1.

APPROACH:

Current knowledge:
- Similarity with 8085 Architecture.
- Theoritically idea to implement custom instructions.
  - Catch: no special hardware req for this.
    So, first step complete the rest in coading.

Current Doubts:
- Implementing the Architecture in Software
- How and where to give the instruction.
- Memory size?
- How to store data to memory directly.
- Flag register implementation.
- Implementation of i/p & o/p w.r.t clock.

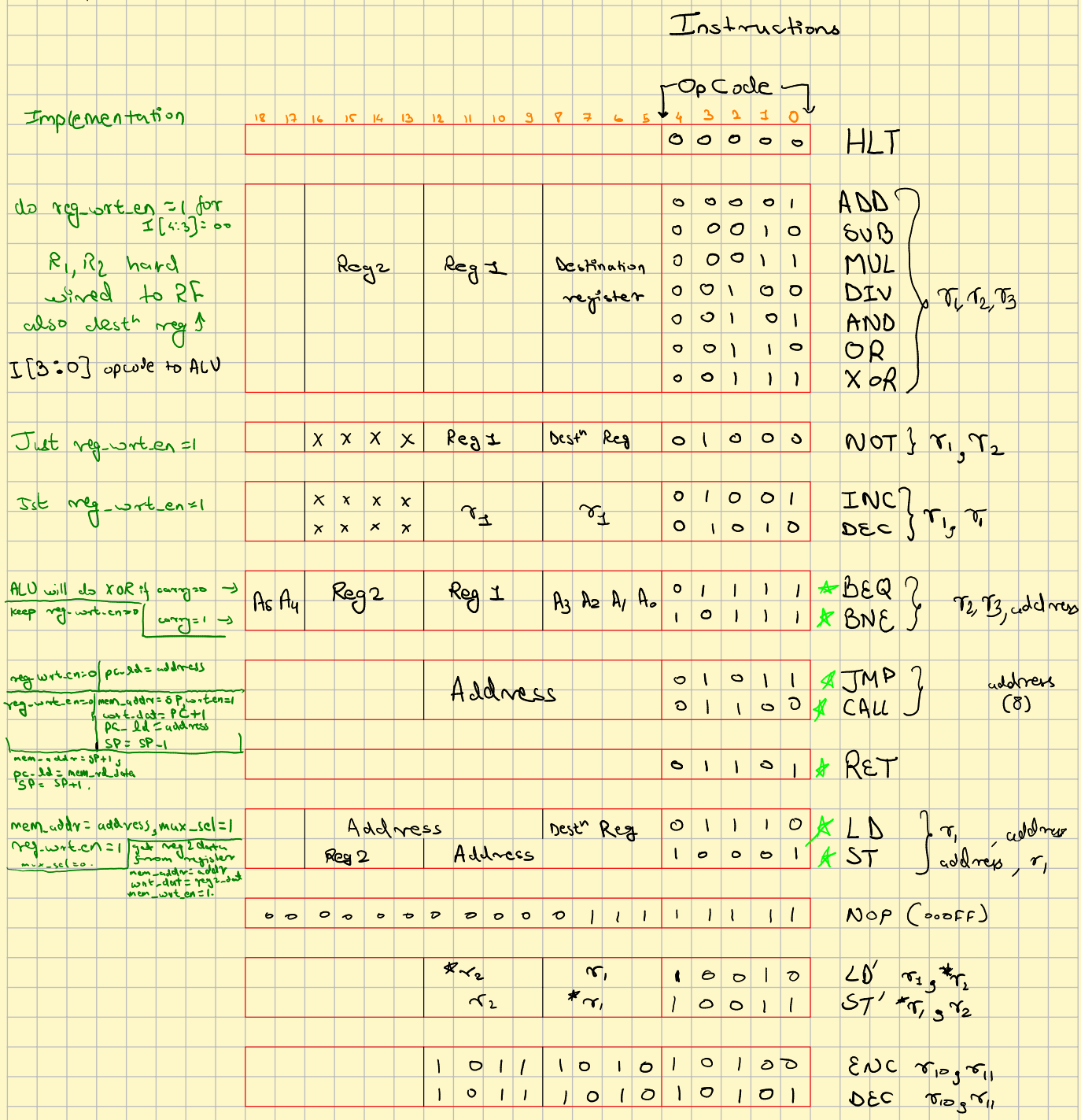
Plan:
Tuesday: Research on 8085 & RISC implementation.
Wednesday: Recall complete verilog knowledge.
Thursday: Draft first architecture clearing all doubts.
Friday: Start coading the first draft.
Saturday: Just frame w.r.t deadline i.e next thursday.

Similarance
- 8085 Arch
- RISC ?

Instruction Format

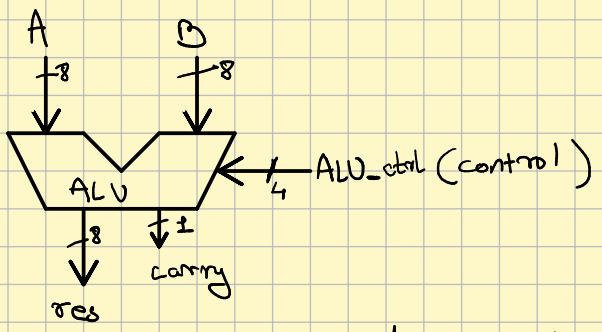
- Instruction Size : 19 bits
- Register Size in it : 4 bit each. ⇒ [16:5] (modular)
- Opcode Size : 5 bits ⇒ [4:0]
- ★ → Beside operation are 2 cycle instruction. So use a NOP after using this just to be sure.

Instructions



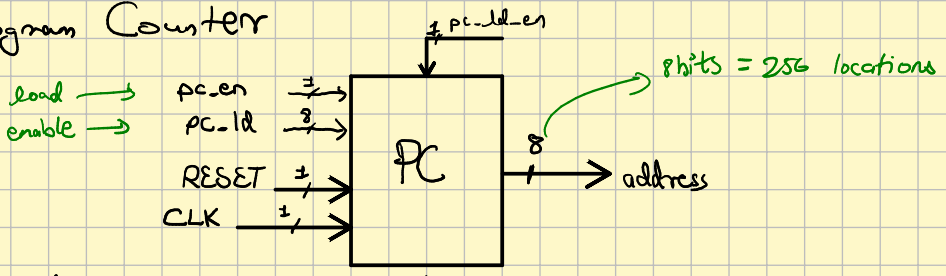
1) ALU Design:

takes 0000 to 1001



if connections to A & B are given through signed wires it can handle -ves

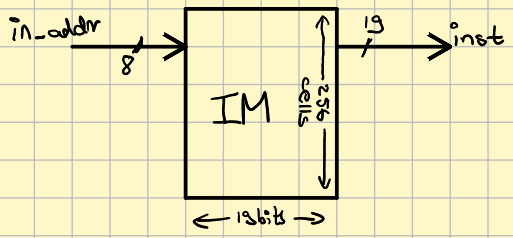
2) Program Counter



the load instructions only loads when it changes its value from a static value So if you jump to a location dont change the pc_ld after the jump is completed.

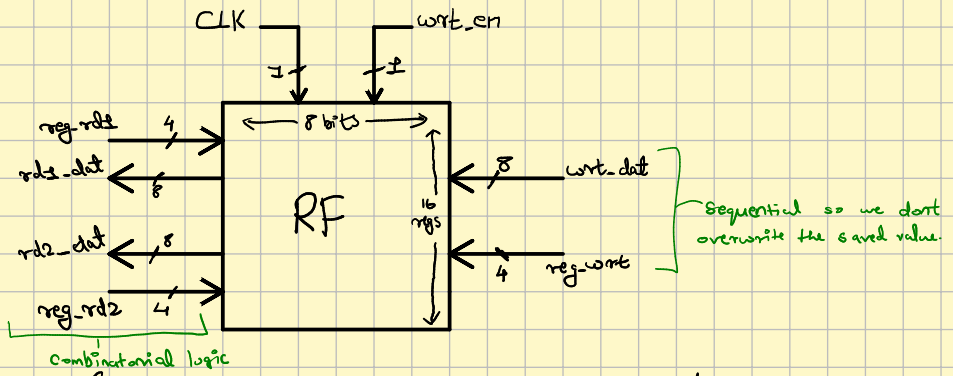
3) Instruction memory

we will store the instructions in it.



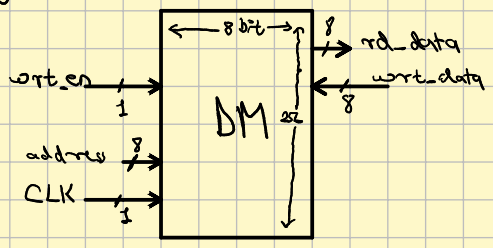
- The instructions can be either stored manually or using a file.
  - ↓ wont synthesize this feature in hardware?

4) Register file:

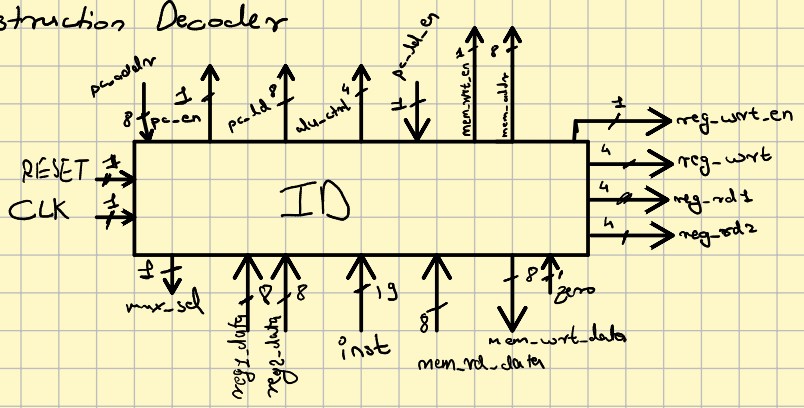


- Total 16 registers from r1 to r16 can be selected by 0000 to 1111.
- 2 data read lines and one data write line.

5] Data Memory:

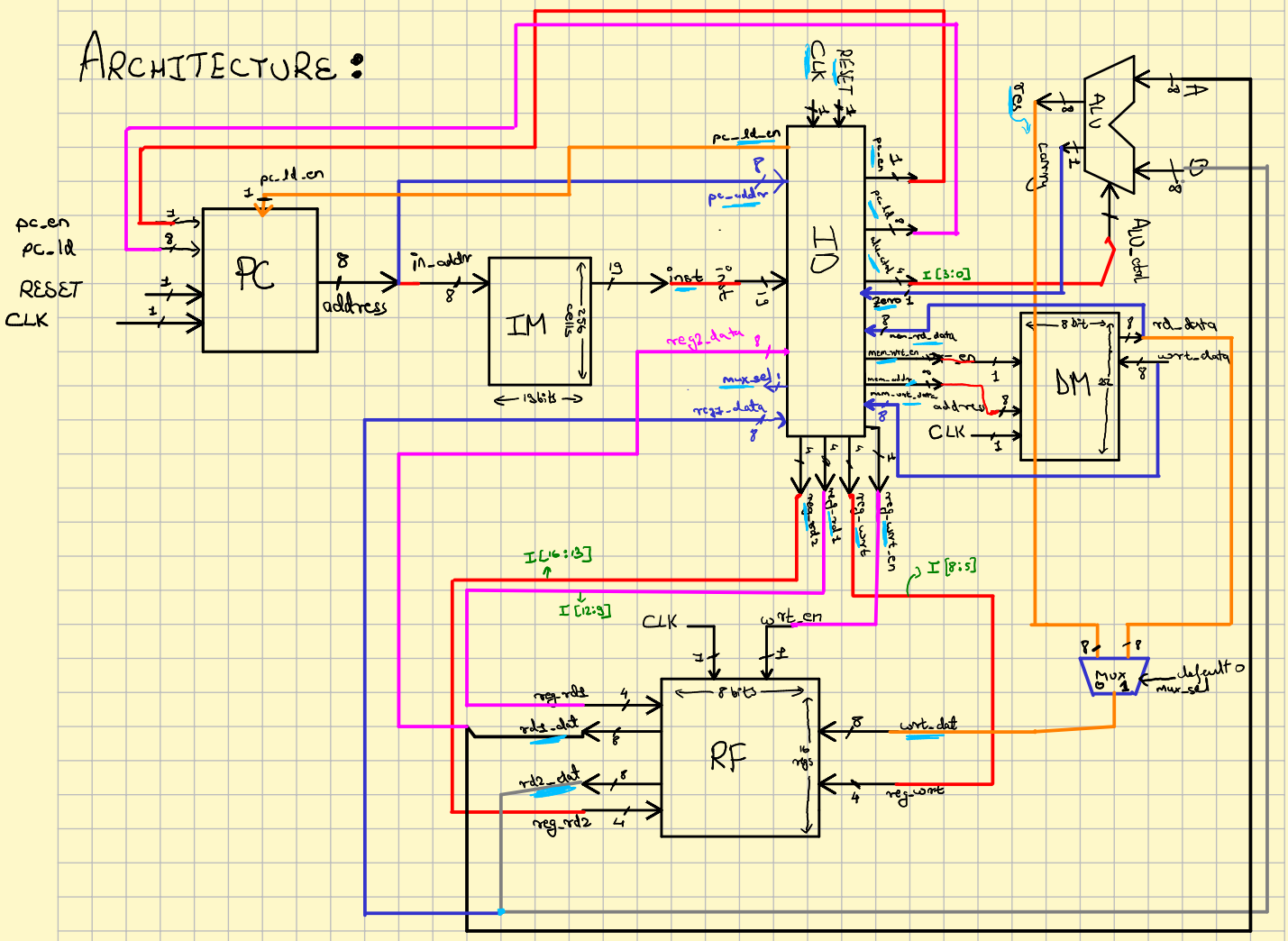


6] Control / Instruction Decoder



★Name of wire in CPU_top

ARCHITECTURE :



$R_0 = 12$, $R_1 = 4$

ADDR			
0	ADD r_2, r_0, r_1	$r_2 = 16$	02041
1	SUB r_3, r_0, r_1	$r_3 = 8$	02062
2	ST 00, r_1	00 = 4	02011
3	NOP		000FF
4	MUL r_4, r_2, r_3	$r_4 = 128$	06483
5	BNE r_1, r_2, 20	Jump to 20	24297
6	NOP		000FF
7	AND r_6, r_1, r_0	$r_6 = 4$	002C5
8	JMP 14	Jump to 14	001CB
9	NOP		000FF
10	LD r_9, 00	$r_9 = 4$	0012E
11	NOP		000FF
12	JMP 24	Jump to 24	00309
13	NOP		000FF
14	OR r_7, r_3, r_6	$r_7 = 12$	0C6E6
15	JMP 09	Jump to 09	0012B
16	NOP		000FF
17	NOP		000 FF
18	0000		0000
19	0000		0000
20	DIV r_5, r_4, r_2	$r_5 = 8$	048A4
21	BEQ r_1, r_1, 07	Jump to 07	022EF
22	NOP		000FF
23	NOP		000FF
24	NOT r_{10}, r_8	$r_{10} = 111$	01148
25	INC r_6, r_6	$r_6 = 5$	00CC9
26	CALL 30		0083C
27	NOP		000FF
28	0000	halt	00000
29	0000		00000
30	INC r_1, r_1	$r_1 = 5$	00229
31	RET	back to 27	0000D
32	NOP		000FF

	NOP	$r_8 = 49$	000FF
	ENC r_{10}, r_{11}	$r_9 =$	01754
	NOP	$r_{10} = 80$	00FF
	NOP	$r_{11} = 00$	00FF
	SUB r_{11}, r_{10}, r_8	$r_{12} =$	1B562
	INC r_{10}, r_{10}	$r_{13} = BC$	01549
	ADD r_{14}, r_8, r_{15}	$r_{14} = 49$	1F1C1
	NOP	$r_{15} = 0003$	000FF
	DEC r_{10}, r_{11}		01755
	NOP		000FF

63	JMP 78	009CB
64	NOP	000FF
65	LD' r_9, r_{11}	01732
66	NOP	000FF
67	XOR r_{12}, r_9, r_{13}	1B387
68	NOP	000FF
69	ST' r_{10}, r_{12}	01953
70	NOP	000FF
71	INC r_{10}, r_{10}	01549
72	INC r_{11}, r_{11}	01769
73	DEC r_{14}, r_{14}	01DCA
74	NOP	000FF
75	BNE r_{14}, r_{15}, 11111	7FDF7
76	NOP	000FF
77	RET	0000
78	NOP	000FF