A Custom 19-bit CPU :

- ## Instruction Set – 19 :

- OP CODE – 3 bit

- 5 -Staged Pipeline [F – D – E – M – WB]

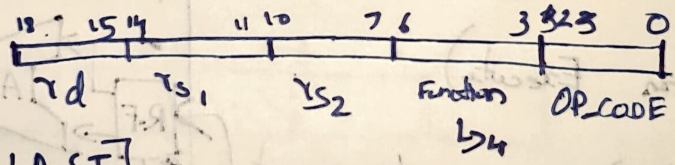
- Instruction memory module will be added

- **Fetch** : • PC
  - Instr_memory

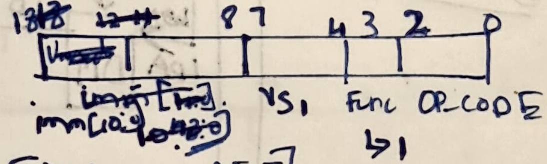
- **Decode** :

- PC_mem = 11 bit
- ~~Data_mem~~ Register file = 4 bit
- Instr = 19 bit
- Data = 8-bit
- Data_mem = 11bit

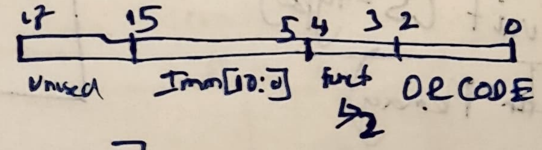
- **R-type** : [Arithmetic, Logical]



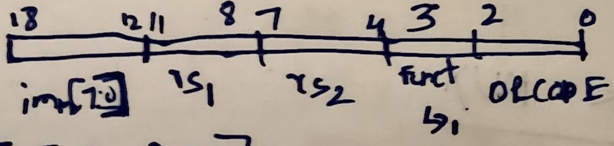
- **M-type** : [LD, ST]



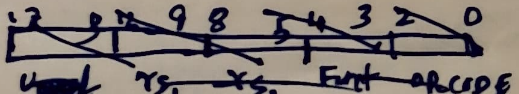
- **J-type** : [JMP, CALL, RET]



- **B-type** : [BEQ, BNE]



- ~~S-type [FFT, ENC, DEC]~~



Operations : (20)

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

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

- JMP addr
- BEQ r1, r2, addr
- BNE r1, r2, addr
- CALL addr
- RET

- LD r1, addr
- ST addr, r1

- FFT r1, r2
- ENC r1, r2
- DEC r1, r2

Note :

Active Low Reset

- **Stage 1:** (Instruction Fetch)
  - Program Counter (synchronous)

    ⇓

    Instruction memory.

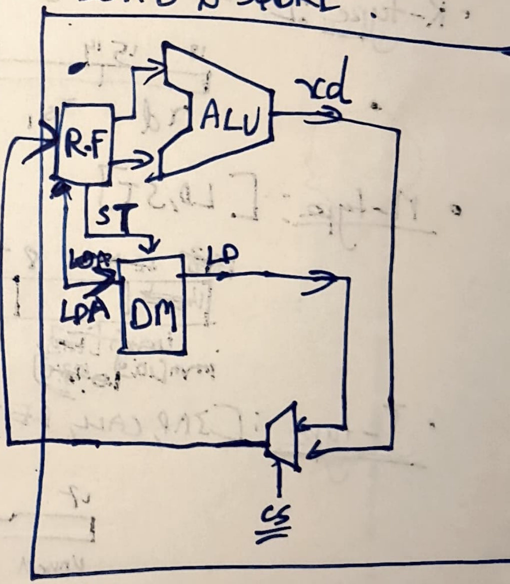
- **Blocks removed:**
  - imm_adder
  - imm_generator.

- **Stage 2:** (Instruction Decode)
  - Decoder
  - Register Files
  - Control Unit ⇒ (~~Covers~~ Covers J-type)
    - ↳ pc_src_out
    - ↳ alu_en_out
    - → ld_str_en_out

- **Stage 3:** (Instruction Execute)
  - ALU Unit (R-type)
  - Branch Unit (B-type)
  - Load Unit (M-type)
  - Store Unit (M-type)
  - Special Application unit (S-type)

- DM_rd_mux unit

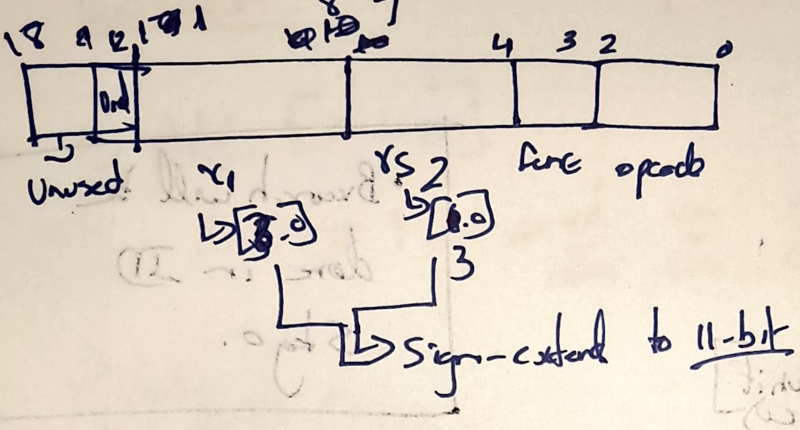
LOAD & STORE :

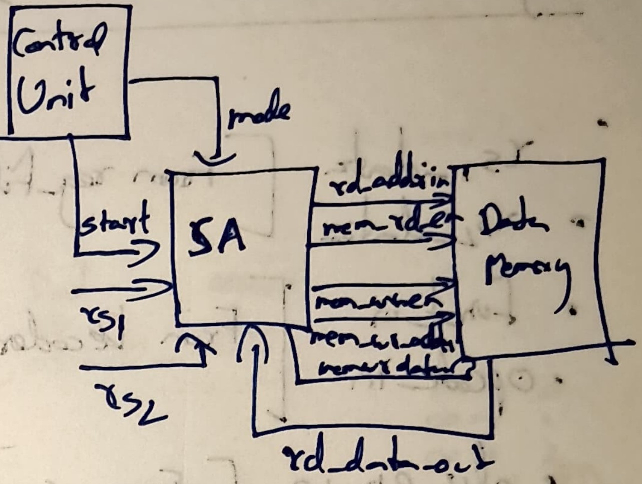


- **Stage 4:** (Instruction Memory)
  - ↳ Data Memory

- **Stage 5:**
  - ↳ Write-Back MUX

- Special type Instruction: [Encryption A Decryption]

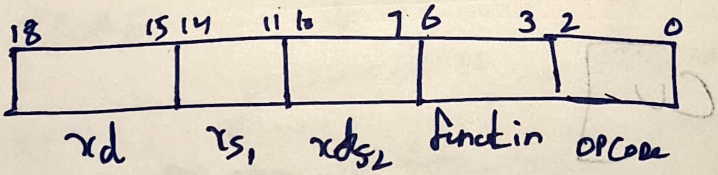




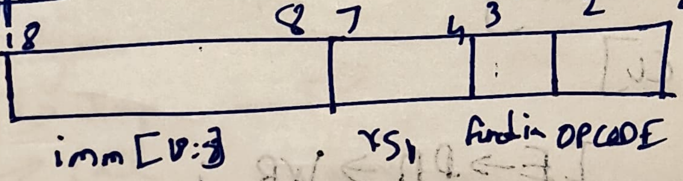
- ENC r1, r2
- DEC r1, r2

Instruction Set Format (FINAL)

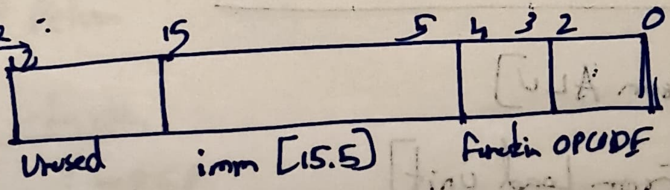
- R type:



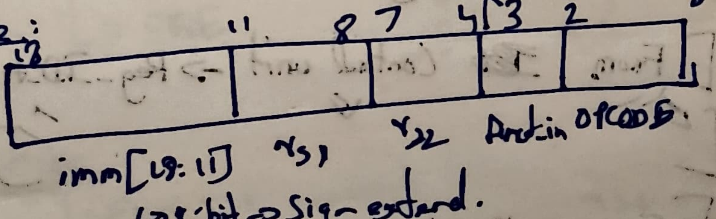
- M-type:



- J-type:



- B-Type:



- Signals to be moved from ID -> IF :

- rs1_data [From reg_file]
- rs2_data
- funct_in [From Decoder]
- opcode_in
- alu_en_in [From Control unit (CU)]
- reg_wr_in [From CU]
- imm_addr_in [From decoder]
- mem_wr_in [From CU]
- ~~branch_en_in~~ mem_rd_in
- Start [From CU]
- mode_enc_dec
- wb-addr_out [From decoder]
- wr_back_sel_out [From CU]

- Branch will be done in ID Stage.

- Signals to be moved from IE -> DM -> WB :

- alu_result_in [From ALU]
- ld_result_in [From Load unit]
- wrbacksel_in [From ~~ID~~ Control unit -> Reg_ID_to_IE]
- reg_wr_in [ ]
- wb_addr_in [From Decoder -> Reg_ID_to_IE].

~~Test~~

Testing:

Testbench:

1.) Add: [R-type]

Instr => 19bit => rd rs1 rs2 funct OPCODE

0001 _ 0010 _ 0011 _ 0000 _ 000

2.) Sub: ~~Instr~~ =>

R-Type: => 19bit

↳ 10 Instruction

r2 + r3

rs_1 rs_2 => 2

↳ 1 → location

3.) M-type:

↳ LOAD => 19bit

↳ STORE

imm[10:0] rs1 FUNCT OPCODE

00000000011 _ 0010 _ 0 _ 001

LD => 1

ST => 0

DM store the data

4.) J-type:

↳ Jump

↳ Call

↳ Return

imm[10:0] funct OPCODE

Unused 000 _ 00 _ 010

01

10

5.) B-type

↳ BEQ

↳ BNE

6 imm[?:0] rs1 rs2 funct OPCODE

_ 0010 _ 0011 _ 0 _ 011

1

6.) S-type

↳ ENC - 1

↳ DEC - 0

Unused rs1 rs2 funct OPCODE

0010 _ 0011 _ 0 _ 100

1

0000
0001
0010
001
010
010
01
10
100
10
10
11
11
11
111