① FETCHING an instruction and house-keeping
② FETCHING OPERANDS and store them somewhere - maybe temporary registers
③ Execution / actual operation
④ Write back to the register

Assumptions:

* Our datapath makes use of:

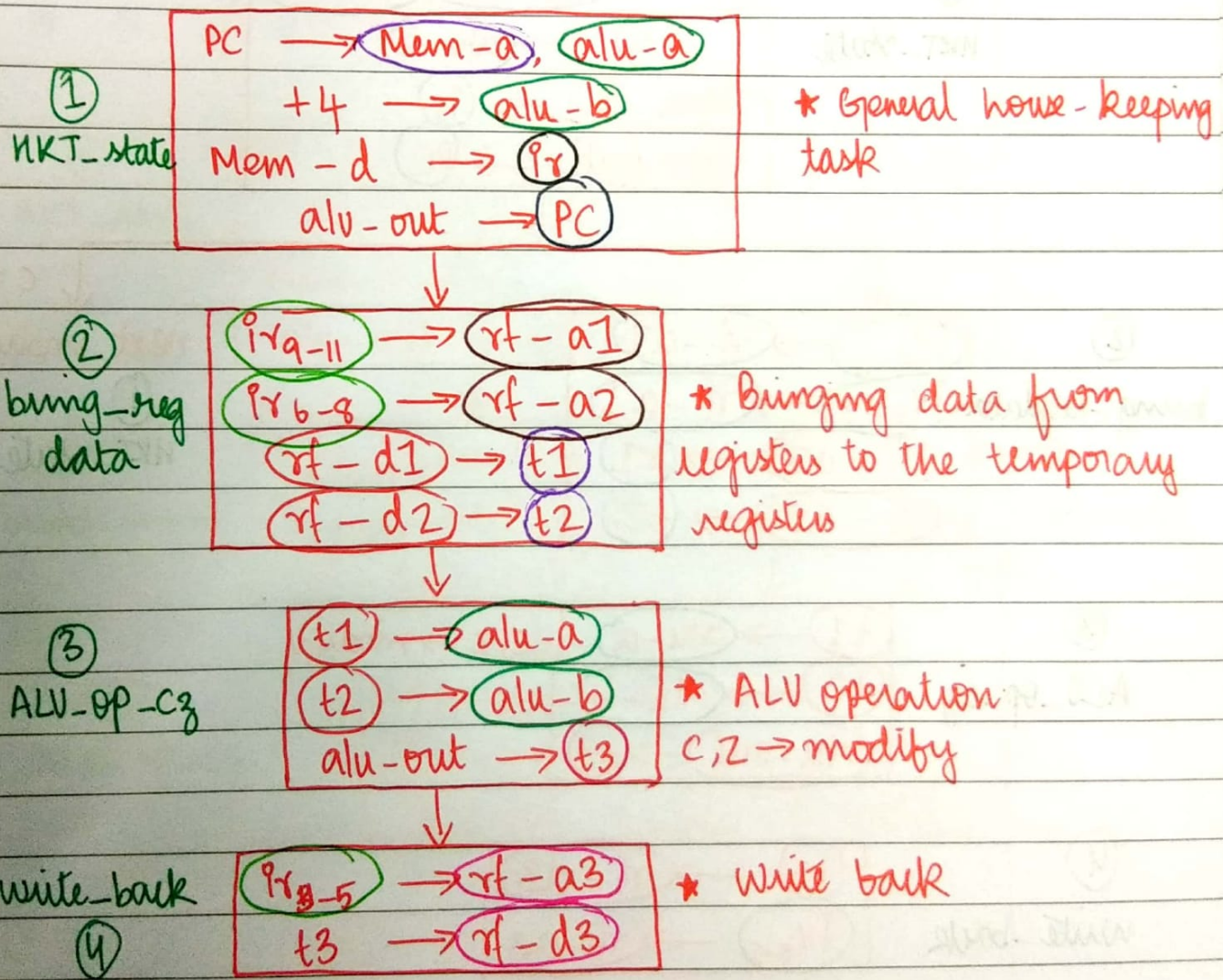
① an ALU for performing arithmetic operations

② Temporary registers - T1 and T2 for ALU, T3 (result of ALU), T4.
③ Register file for storing registers R0 → R7.

HARDWARE FLOWCHARTS:

① ADD

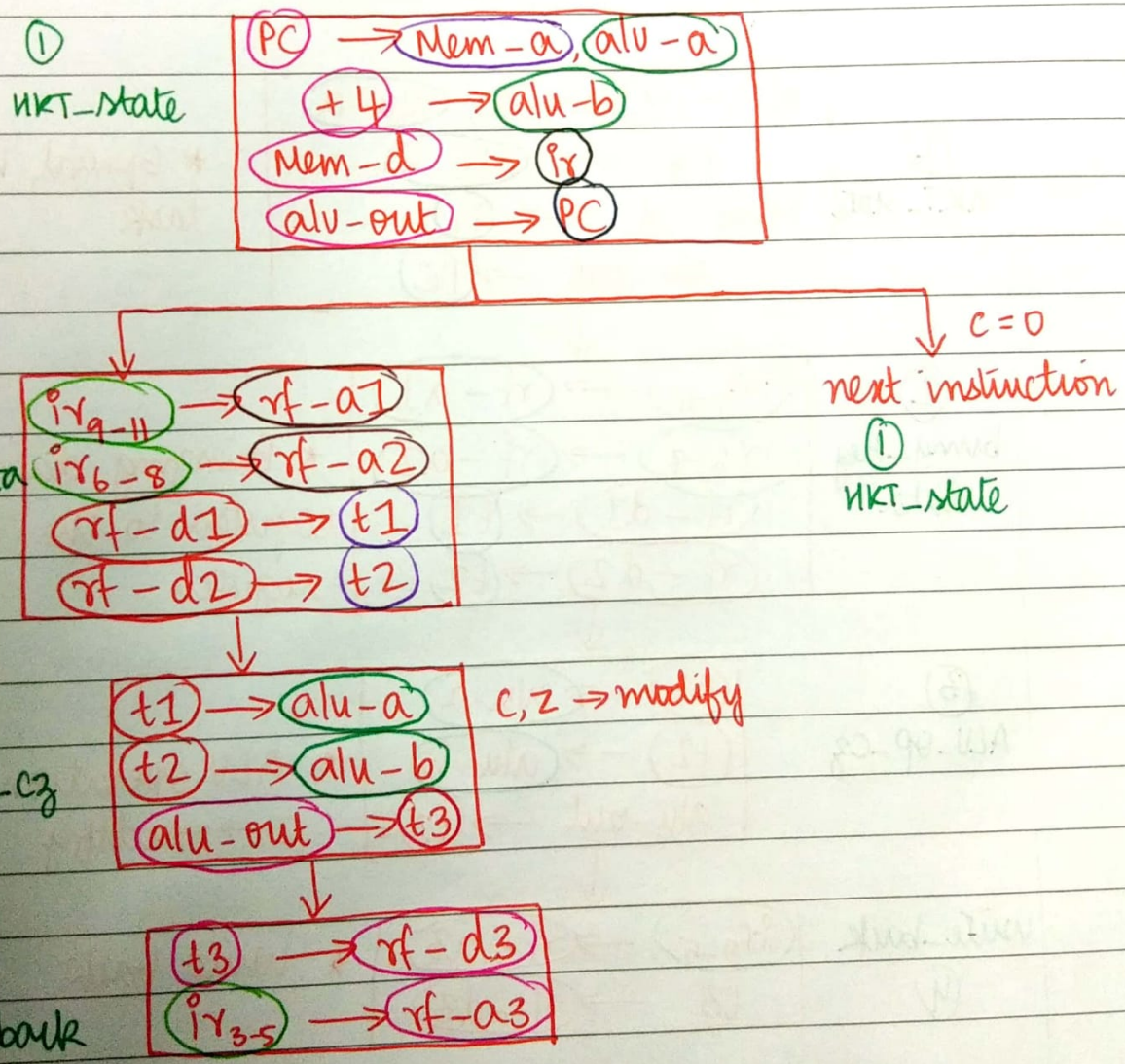
OP	RA	RB	RC	0	00
15 13	12 10	9 8	6 5	3 2	2 1 0



② ADC

OP	RA	RB	RC	0	10
15 12	11 9	8 6	5 3	2 2	1 0

Add (RB+RA) and put it in RC if carry flag is set.



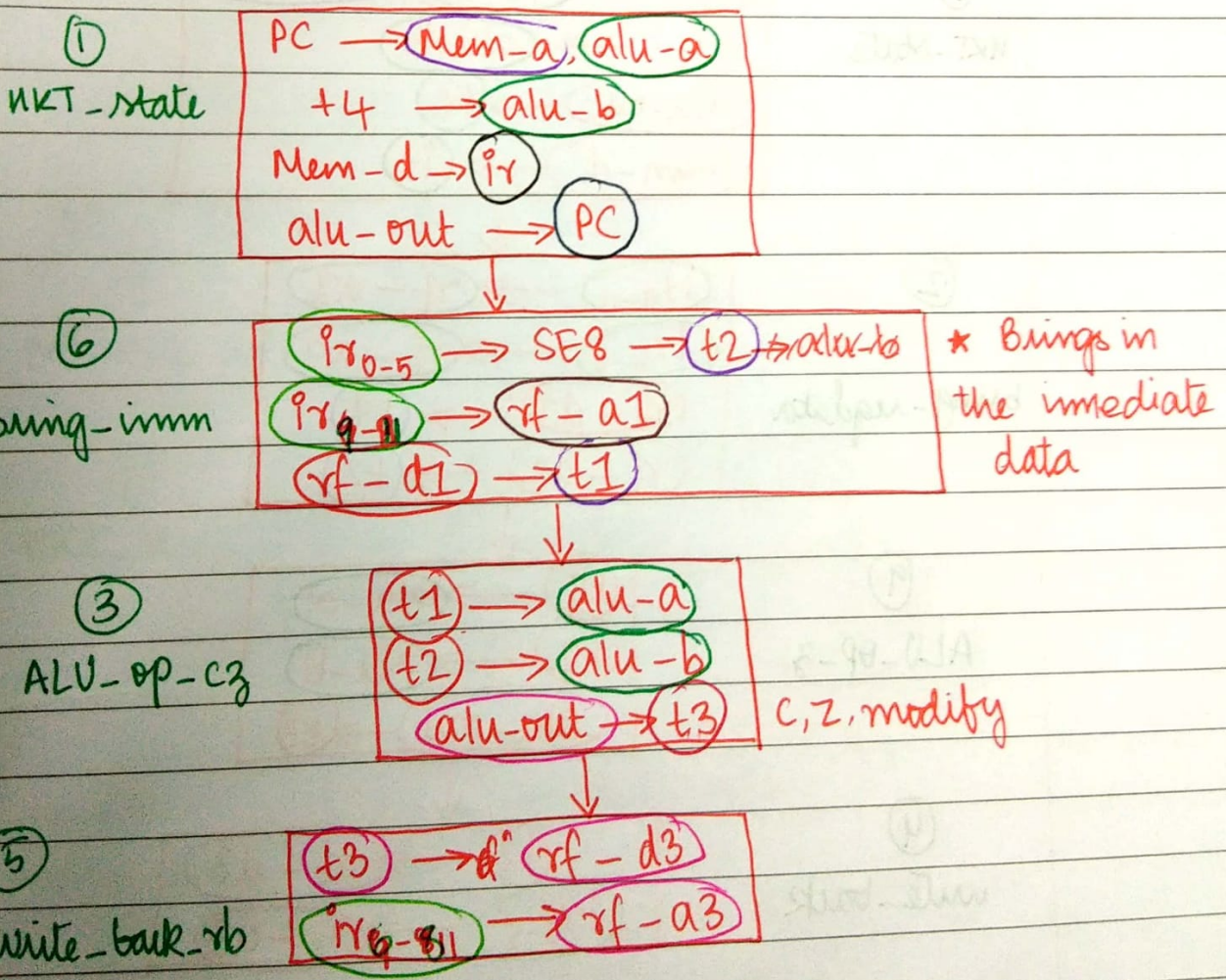
__/__/____

③ ADZ Add R_b to R_a to put the contents in R_c, if zero flag is set.

→ Same as that of ADC.

④ ADI I-type

OP	R_a	R_b	Imm6
15 12	11 9	8 6	5 0

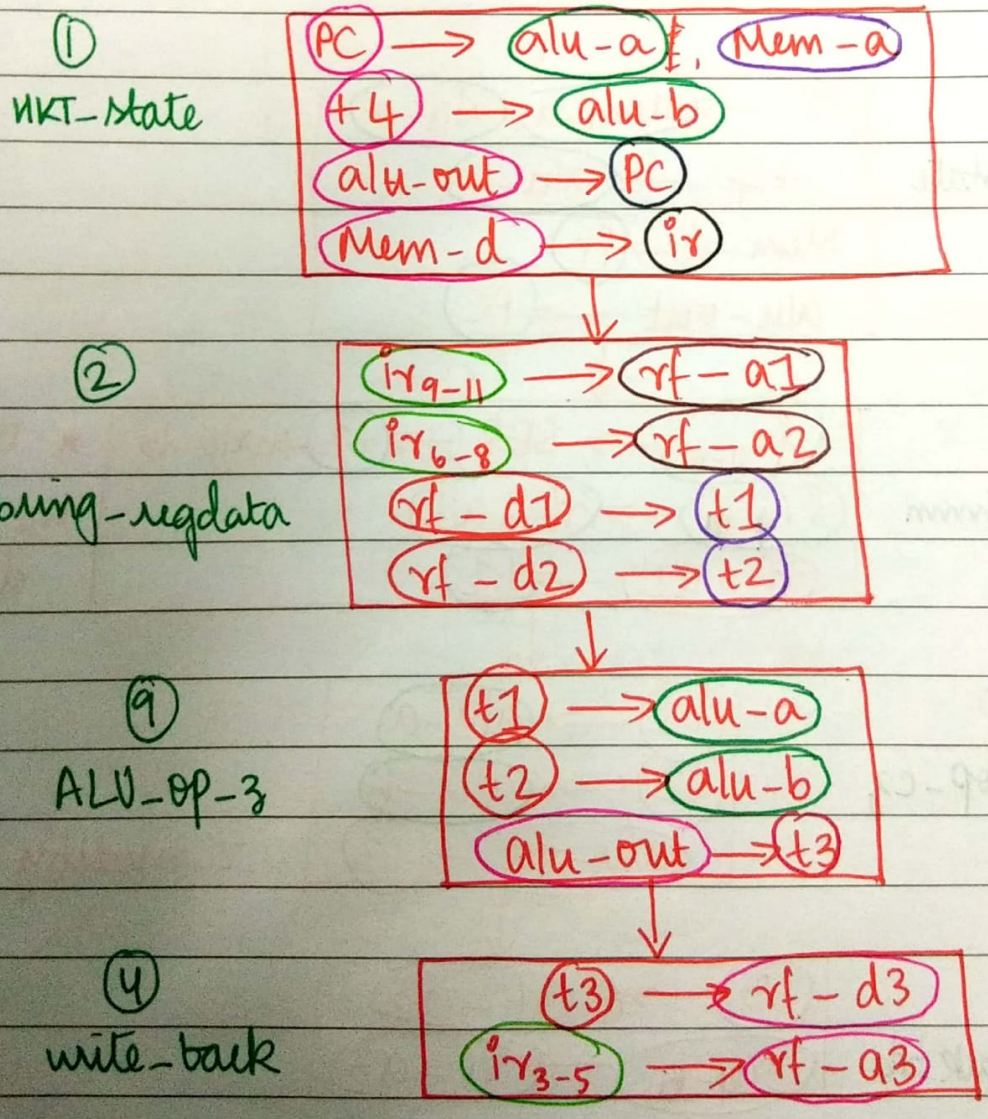


(5) NDU

NAND the contents of Registerb to RegA to store them in RegC.

Modifies only z flag ⇒ Diff state

OP	RA	RB	RC	0	00
15 12	11 9	8 6	5 3	2 2	1 0



⑥ NAND – Modifies only Z flag (NDC)

HKT → if ZC

↓ Z=1 → Same states as those for NAND

↓ C=0 → next instruction

⑦ NDZ → same as that of NDC

⑧ LW → I-type instruction

OP	RA	RB	9mm6
15 12	11 9	8 6	5 0

Memory address is computed as → $[R_B] + 9mm$

① HKT_state

PC → alu-a, Mem-a

Mem-d → ir

+4 → alu-b

alu-out → PC

⑱ bring_imm2

ir_{6-8} → rf-a2

rf-d2 → t1

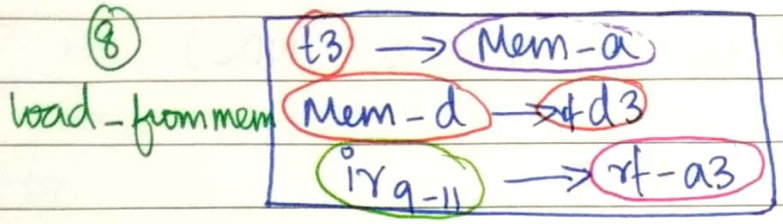
ir_{0-5} → SE8 → t2

⑨ ALU_op_3

t1 → alu-a

t2 → alu-b

alu-out → t3

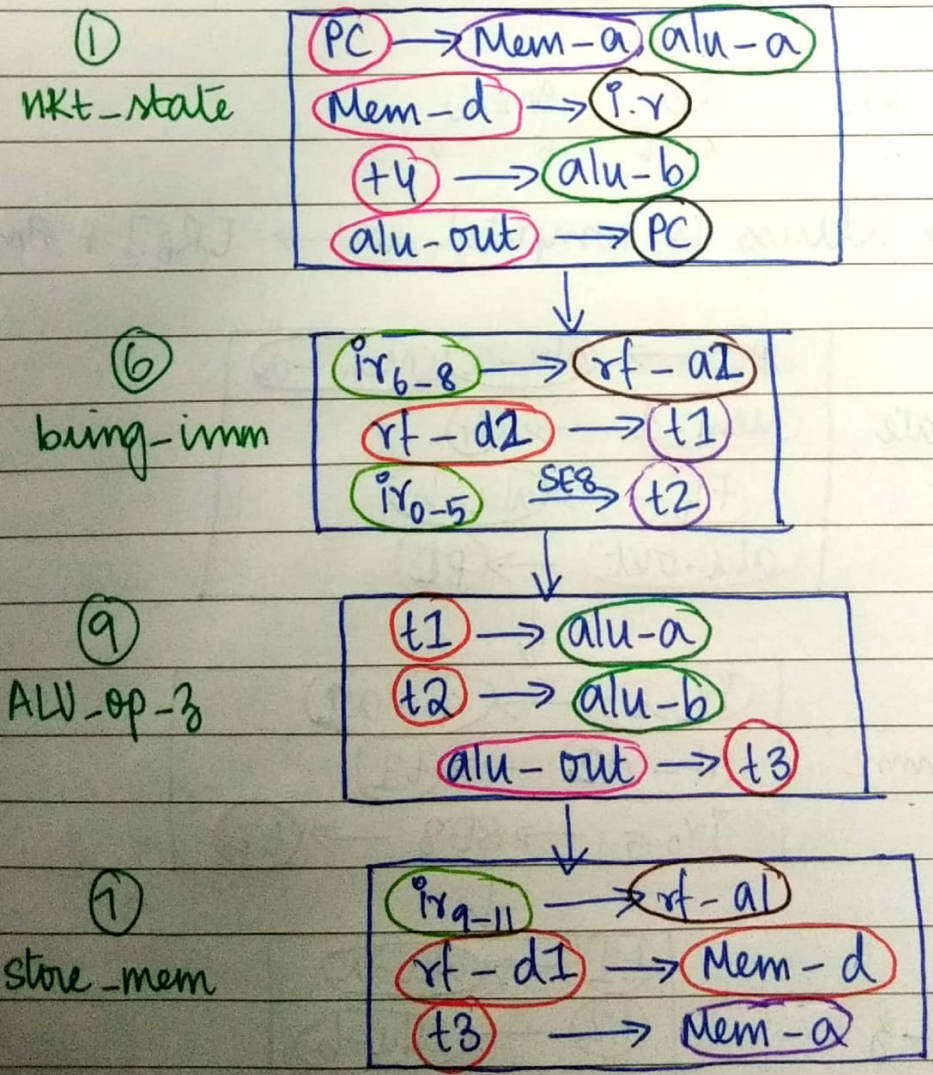


9 SW

Store value from register A to memory.

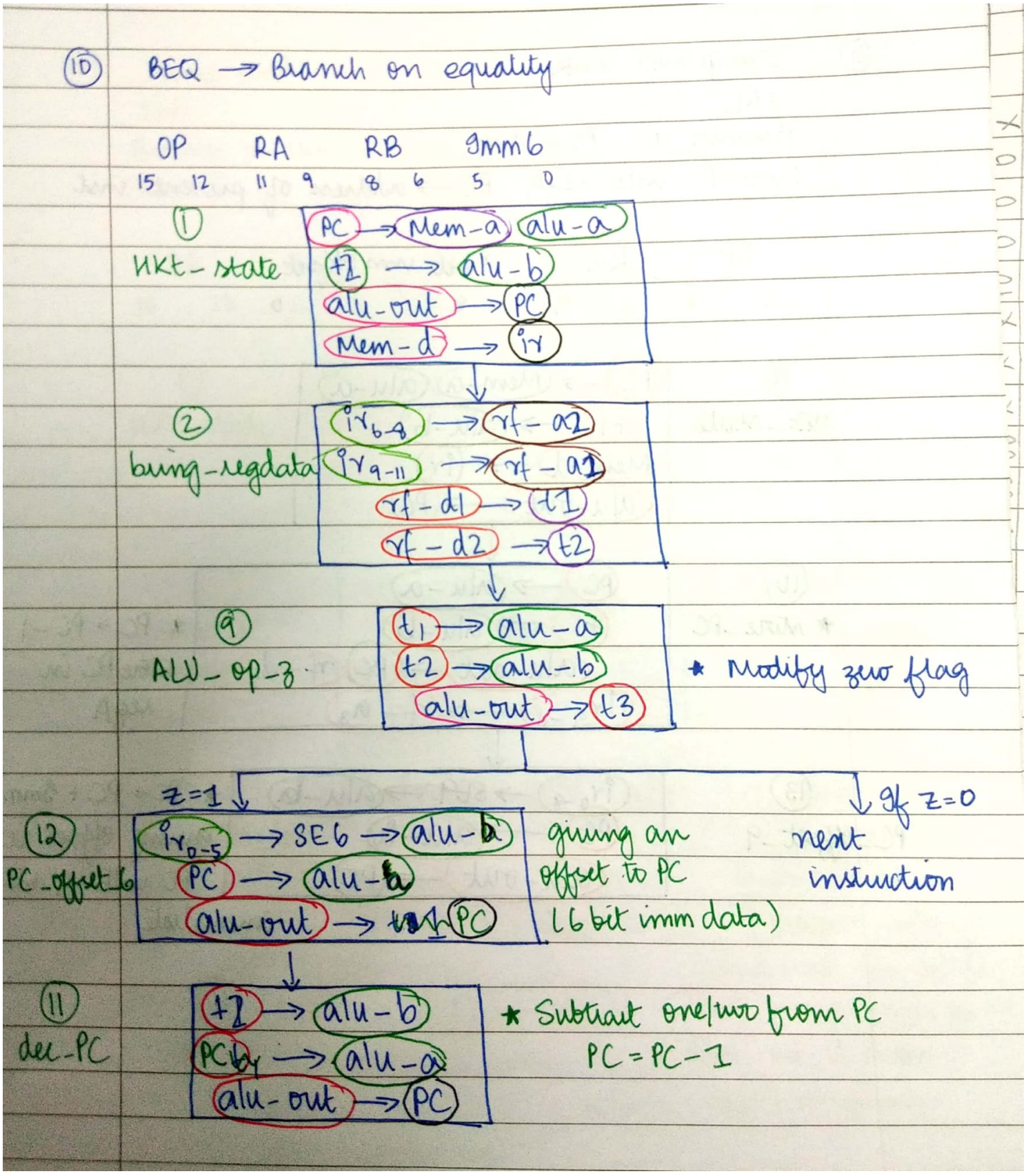
Memory can be computed as $[R_b] + 9mm$.

OP	RA	RB	9mm6
15 12	11 9	8 6	5 0



(10) BEQ → Branch on equality

OP	RA	RB	9mm 6
15 12	11 9	8 6	5 0



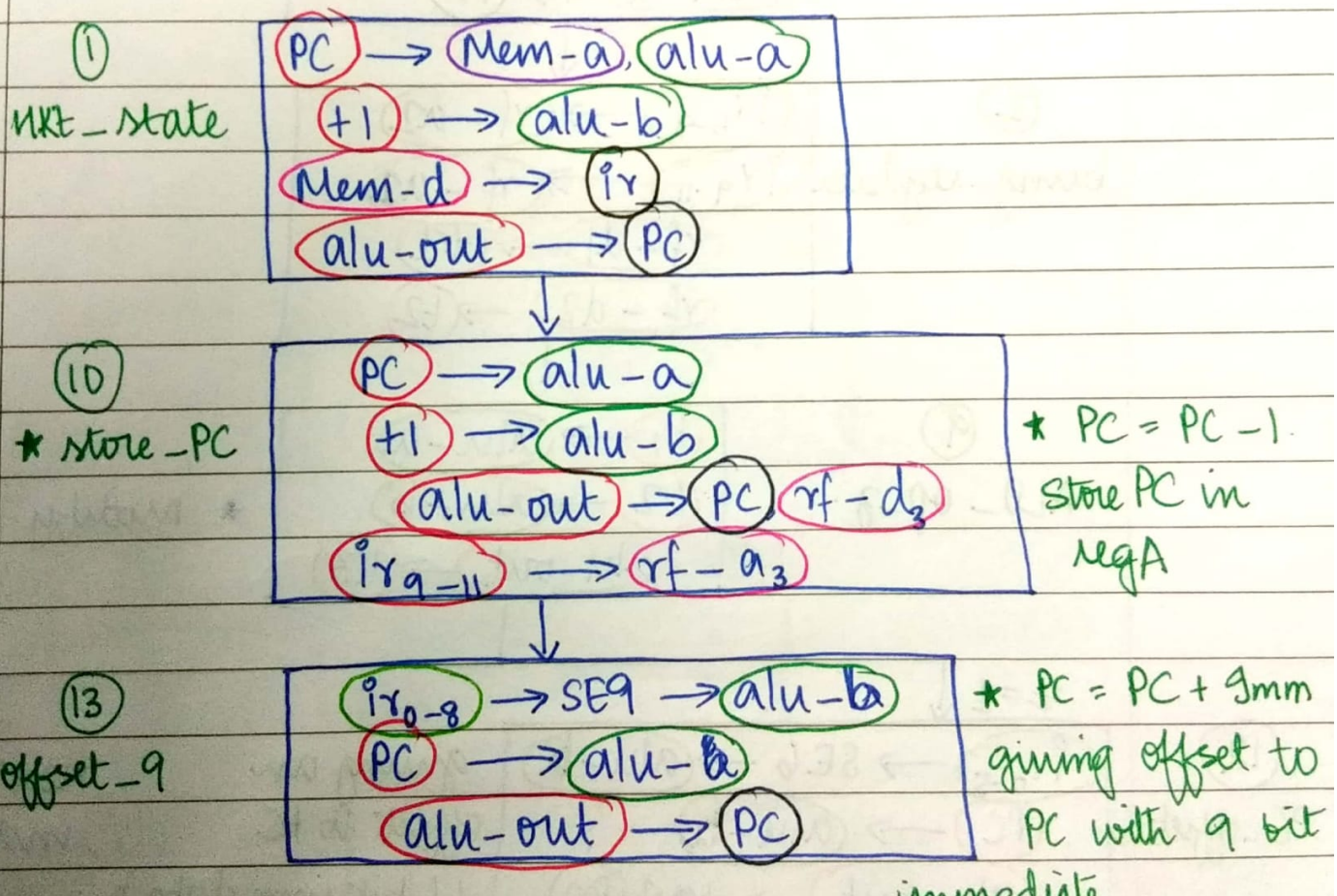
(11) Jump and link

JAL

Branch to PC + 9mm

Store PC into regA PC → address of present inst

OP	Ra	9 bit imm offset
15 12	11 9	8 0



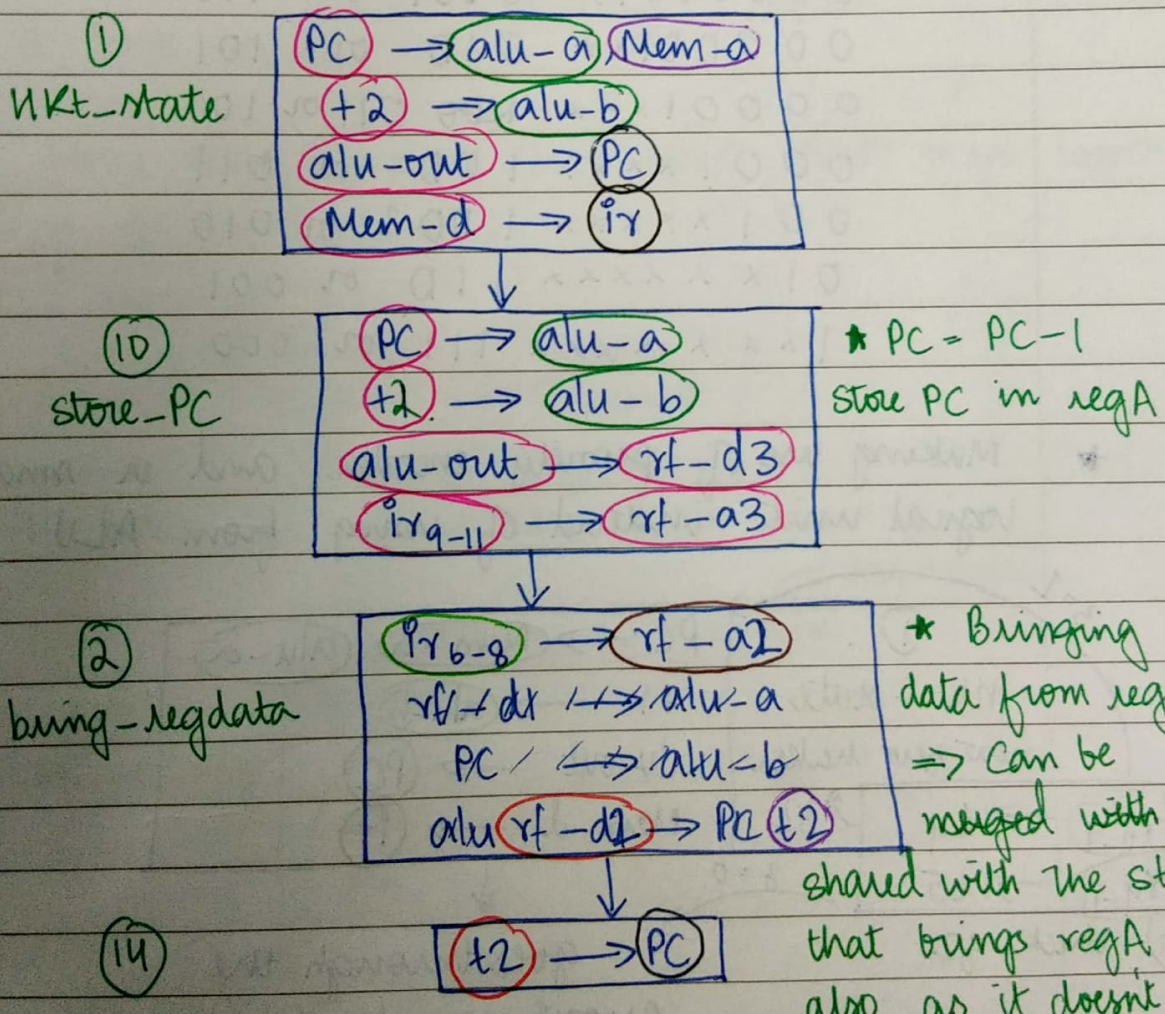
(12) Jump and link-to register

JLR

Branch to the address in RegB

Store PC in regA

OP		RA		RB		000-000	
15	12	11	9	8	6	5	0



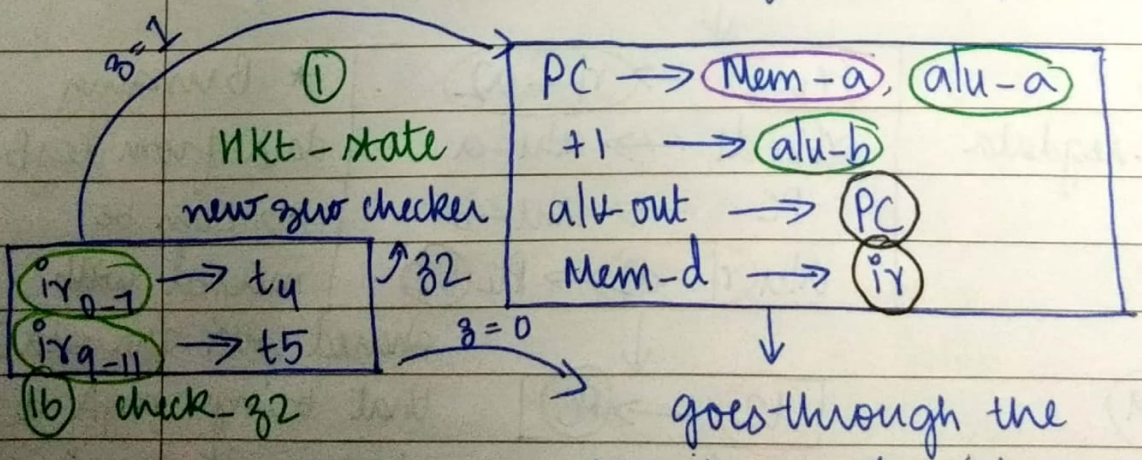
(13) Load multiple:

OP	RA	0 + 8 bits corresponding to Reg ⇒ 0
15 12	11 9	8 0

Priority encoder :

R00 R07	
00000001	000 or 111
00000001x	001 or 110
0000001xx	010 or 101
000001xxx	011 or 100
00001xxxx	100 or 011
0001xxxxx	101 or 010
001xxxxxx	110 or 001
01xxxxxxx	111 or 000
1xxxxxxxx	111 or 000

* Making use of priority encoder and a small logical unit instead of using from ALU.



goes through the priority encoder (if the input is zero stop, else connect feedback) x use some control signal here

↓

* t4 is storing the immediate value, updated

If say, 1xxxxxxx → output of priority encoder → 000

Use a decoder which does the following:

000 → 0111 1111 — ①

Then go to the state where the data from the address given in register RA is stored in R0. Increment the value given in register RA to put it back in RA

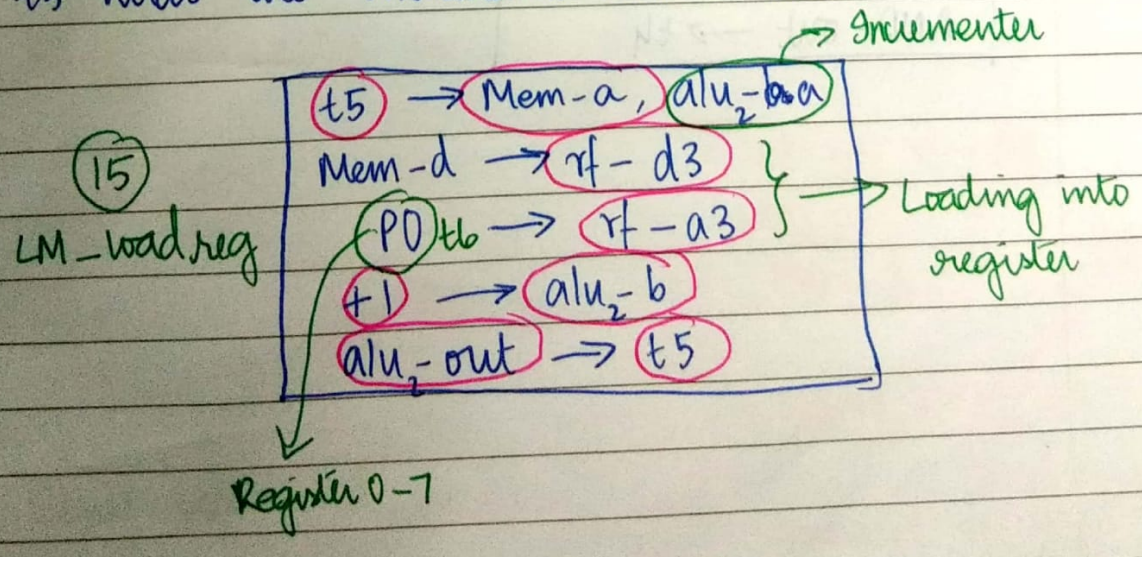
Now take ① and the input, AND them together. feedback.

t5 ← Mem-a

Mem-d → rf-d3

PO → rf-a3

t5 holds the incremented address value.



(14) Store multiple:

Same steps, but instead of loading into registers we write to memory from the mentioned addresses of registers.

The only state that will change is:

* t5 will store the address of the memory.

(16)

SM_storemem

t5 → Mem-a, alu_2-a

t6 ← PO → rf-a1

rf-d1 → Mem-d

+1 → alu_2-b

alu-out → t5

} storing the value in register to the address given by t5.

(17)

Priority_out

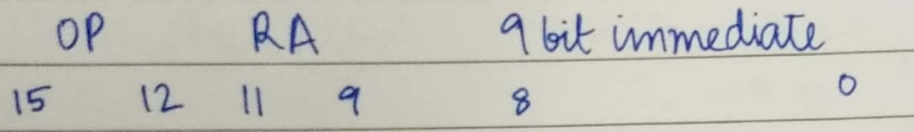
t4 → PE, AND

PO → t6, dec

dec-out → AND

AND-out → t4

(15) LHI – Load higher immediate



(1)
FETCH_state

PC → Mem-a, alu-a
+2 → alu-b
alu-out → PC
Mem-d → ir

(18) (19)
data_extension

ir_{0-8} → Data extension → rf-d3
ir_{9-11} → rf-a3

__/__/____

★ rf-a1 and rf-a2 are connected to:

ir_{9-11} → rf-a1
t6 → rf-a1

ir_{6-8} → rf-a2

★ Stuff connected to alu-a and alu-b:

PC → alu-a
t1 → alu-a
ir_{0-5} → SE6 → alu-a ×

+1 → alu-b
t2 → alu-b
-1 → alu-b
ir_{0-8} → SE9 → alu-b
ir_{0-5} → SE6 → alu-b

★ Stuff connected to t1 and t2

rf-d1 → t1
rf-d2 → t1

rf-d2 → t2
ir_{0-5} → SE6 → t2

★ Stuff connected to rf-a3, rf-d3

ir_{3-5} → rf-a3
ir_{6-8} → rf-a3
ir_{9-11} → rf-a3
t6 → rf-a3

t3 → rf-d3
alu-out → rf-d3
Mem-d → rf-d3

★ Stuff connected to t3

alu-out → t3

★ Stuff connected to PC:

alu-out → PC
t2 → PC