HARDWARE FLOWCHARTS

1. ADD

OP	RA	RB	RC	0	00

① HKT

PC → mem-a, alu-a
+4 → alu-b
Mem-d → IR
ALUOut → PC

House keeping task

↓

② bring-regdata

IR_{9-11} → rf-a1
IR_{6-8} → rf-a2
rf-d1 → t1
rf-d2 → t2

↓

③ ALU_OP_CZ

t1 → alu-a
t2 → alu-b
ALUOut → t3

↓

④ write-back

IR_{3-5} → rf-a3
t3 → rf-d3

2. ADC

OP	RA	RB	RC	0	10

Add RB, RA and put it in RC if carry flag is set

① HICT

PC → mem-a, alu-a
t4 → alu-b
mem-d → IR
alu-out → PC

C = 0
① HICT

② bring_reg data

IR 9-11 → rf-a1
IR 6-8 → rf-a2
rf-d1 → t1
rf-d2 → t2

③ ALU=OP_CZ

t1 → alu-a
t2 → alu-b
ALUOut → t3

④ write_back

t3 → rf-d3
IR 3-5 → rf-a3

③ ADZ

Add RB to RA to put the contents in RC, if zero flag is set

→ Same as that of ADC.

4. ADL

OP	R_A	R_B	R_C
15 12	11 9	8 6	5 3

① HKT - state

$PC \rightarrow$ mem-a, alu-a
$+4 \rightarrow$ alu-b
mem-a $\rightarrow$ ir
alu-out $\rightarrow$ PC

② bring reg. data

$ir_{9-11} \rightarrow rf\text{-}a_1$
$ir_{6-8} \rightarrow rf\text{-}a_2$
$rf\text{-}d_1 \rightarrow t_1$
$rf\text{-}d_2 \rightarrow t_2, t_3$

③ ALU - OP - cy

$t_2 \rightarrow$ alu-a
$t_3 \rightarrow$ alu-b
alu-out $\rightarrow t_2$

$t_2 \rightarrow$ alu-b
$t_1 \rightarrow$ alu-a
alu-out $\rightarrow t_3$

④ write back

$t_3 \rightarrow rf\text{-}d_3$
$ir_{3-5} \rightarrow rf\text{-}a_3$

5. ADI

I-type

OP	RA	RB	Imm6

(1) HKT

PC → mem-a, alu-a
t4 → alu-b
mem-d → IR
ALUOut → PC

↓

(6) bring_immdata

IR_{0-5} → SE8 → t2
IR_{9-11} → rf-a1
rf-d1 → t1

↓

(3) ALU_OP_CZ

t1 → alu-a
t2 → alu-b
ALUOut → t3

↓

(5) write_back_RB

t3 → rf-d3
IR_{6-8} → rf-a3

6. NDU

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

Modifies only z flag ⇒ diff state.

OP	RA	RB	RC	0	00

① HKT

PC → alu-a, mem-a
+4 → alu-b
ALUOut → PC
Mem-d → IR

↓

② bring_regdata

IR_{9-11} → rf-a1
IR_{6-8} → rf-a2
rf-d1 → t1
rf-d2 → t2

↓

⑨ ALU_OP_Z

t1 → alu-a
t2 → alu-b
ALUOut → t3

↓

④ write_back

t3 → rf-d3
IR_{3-5} → rf-a3

7. NDC

NAND → Modifies only z flag

HICT → if zC

↓ C=1 → same states as those for NAND

↓ C=0 → next instruction

8. NDZ

Same as that of NDC.

9. LHI → Load higher immediate

OP	RA	9-bit immediate

(1) HICT

PC → mem-a, alu-a
+2 → alu-b
ALUOut → PC
mem-d → IR

↓

(19) data-extension

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

10. LW

I-type instruction

OP	RA	RB	Imm 6

memory address is computed as → $[R_B] + Imm$

① HKT

PC → alu-a, mem-a
mem-d → IR
+4 → alu-b
ALUOut → PC

⑥ bring_imm2

IR_{6-8} → rf-a2
rf-d2 → t1
IR_{0-5} → SE8 → t2

⑨ ALU_OP_2

t1 → alu-a
t2 → alu-b
ALUOut → t3

⑧ load_from_mem

t3 → mem-a
mem-d → rf-d3
IR_{9-11} → rf-a3

11. SW

Store value from register A to memory

Memory can be computed as [RB] + Imm

OP	RA	RB	Imm6

① HKT

PC → mem-a, alu-a
mem-d → IR
+4 → alu-b
alu-out → PC

⑥ bring-imm

IR_{6-8} → rf-a2
rf-d2 → t1
IR_{0-5} $\xrightarrow{SE8}$ t2

⑨ ALU_OP_Z

t1 → alu-a
t2 → alu-b
ALUOut → t3

⑦ store-mem

IR_{9-11} → rf-a1
rf-d1 → mem-d
t3 → mem-a

12. LM

Load multiple

OP	RA	0+8 bits corresponding to Reg7 → 0

Priority encoder:

R0							R7				
0	0	0	0	0	0	0	1	0	0	0	(or) 1 1 1
0	0	0	0	0	0	1	x	0	0	1	(or) 1 1 0
0	0	0	0	0	1	x	x	0	1	0	(or) 1 0 1
0	0	0	0	1	x	x	x	0	1	1	(or) 1 0 0
0	0	0	1	x	x	x	x	1	0	0	(or) 0 1 1
0	0	1	x	x	x	x	x	1	0	1	(or) 0 1 0
0	1	x	x	x	x	x	x	1	1	0	(or) 0 0 1
1	x	x	x	x	x	x	x	1	1	1	(or) 0 0 0

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

(1) HKT

PC → mem-a, alu-a
+1 → alu-b
ALUOut → PC
mem-d → IR

z=1

IR0-7 → t4
IR9-11 → t5

new zero checker

(16) check-z2

z=0 goes through the priority encoder (if the input is zero stop, else connect feedback), use some control signal here

↓

* t4 is storing the immediate value, updated.

If say, 1xxx xxxx → output of priority encoder → 000

Use a decoder which does the following:

000 → 01111111 → ①

Then go to the state whether the data from the address given in register RA is stored in RD

Increment the value given in register RA to put it back in RA

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

t5 holds the incremented address value

(15).

LM_load reg

t5 → mem-a, alu2-a
mem-d → rf-d3
P0 t1 → rf-d3
t1 → alu2-b
alu2-out → t5

incrementer

loading into register.

register 0-7

13. SM

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 in t5 will store the address of the memory.

(17)

SM_storemem

t5 → mem-a, alu2-a
PO → rf-a1
rf-d1 → mem-d
+1 → alu2-b
alu-out → t5

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

↓

(18)

Priority_out

~~t4 → PE, AND~~
PO → t6, dec
dec-out → AND
AND-out → t4

14. BEQ

Branch on equality

OP	RA	RB	Imm6

(1) HKT

PC → mem-a, alu-a
+2 → alu-b
ALUOut → PC
mem-d → IR

↓

(2) bring-regdata

IR_{6-8} → rf-a2
IR_{9-11} → rf-a1
rf-d1 → t1
rf-d2 → t2

↓

(9) ALU_OP_t

t1 → alu-a
t2 → alu-b
ALUOut → t3

Z=1

(12) PC-offs-6

IR_{0-5} → SE6 → alu-b
PC → alu-a
ALUOut → PC

giving an offset to PC (6 bit imm data)

if Z=0. next instruction

↓

(11) dec-PC

+2 → alu-b
PC → alu-a
ALUOut → PC

Subtract one/two from PC
PC = PC-1

15. JAL

Jump and Link

Branch to PC + imm

Store PC into reg A PC → address of present inst

OP	RA	9-bit imm offset

(1)
HKT

PC → mem-a, alu-a
+1 → alu-b
mem-d → IR
ALUOut → PC

↓

(10)
Store-PC

PC → alu-a
+1 → alu-b
ALUOut → PC, rf-d3
IR_{9-11} → rf-a_3

PC = PC − 1
Store PC in reg

↓

(13)
PC-offset-9

IR_{0-8} → SE9 → alu-b
PC → alu-a
ALUOut → PC

PC = PC + imm
giving offset to PC with 9 bit immediate

18. JLR

Jump and Link to register

Branch to the address in RegB

Store PC in reg A

OP RA RB 000_000

(1) HKT

PC → alu-a, mem-a
+2 → alu-b
ALUOut → PC
mem-d → IR

↓

(10) Store_PC

PC → alu-a
+2 → alu-b
ALUOut → rf-d3
IR_{9-11} → rf-a3

PC = PC - 1
Store PC in reg A

↓

(2) bring_regdata

IR_{6-8} → rf-a2
rf-d2 → t2

→ Bringing data from reg B.
→ can be shared with the state that brings reg A to t1 also as it doesn't affect anything.

↓

(14) t2_PC

t2 → PC

17. JRI

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

① HKT - state

PC → mem-a, alu-a
+1 → alu-b
mem-d → ir
alu-out → PC

② using reg. data

ir_{9-11} → r_f − a_1
r_f − d_1 → t_1

③ PC - offset - 9

ir_{0-8} → SE9 → alu-b
t_1 → alu-a
alu-out → PC