IITB-RISC Pipelined-CPU

Final Project Report

EE309: Microprocessors

TEAM ID: 6

Anshu Arora
(22B1207)

Sachi Deshmukh
(22B1213)

Abhineet Agarwal
(22B1219)

Garima Goplani
(22B3958)

Under the guidance of

Prof. Virendra Singh

Department of Electrical Engineering,

Indian Institute of Technology, Bombay

Contributions by Individual Team Members

Contribution	Member
Flowchart for instructions	Sachi, Anshu
Designing a pipeline structure	Sachi, Garima, Anshu
Creating datapath	Sachi, Garima, Anshu
Verification of pipeline hardware	Abhineet
Identifying where to add MUXs and de-MUXs	Garima
MUX tables and control signals	Sachi, Garima
Tabulating the control signals for each instruction	Sachi, Garima, Anshu
Ideation of implementing hazard control	Abhineet, Sachi, Garima, Anshu
Circuit for hazard implementation	Abhineet, Anshu
Writing of code	Abhineet
Debugging of code	Abhineet, Garima
Report formation	Sachi, Anshu

ADA

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T$_1$—>ALU2-A	ALU2_CONTROL ADD

T_2—> ALU2-B	T_3-W
ALU2-C —> T_3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

ADC

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID_{9-11}—>RF-A_1	
IR-ID_{6-8} —> RF-A_2	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D_1 —--> T1	T_1-W
RF-D_2 —--> T2	T_2-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T_1—>ALU2-A	ALU2_CONTROL ADD

T_2—> ALU2-B	T_3-W
ALU2-C —> T_3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back
(If carry == 1)

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

(If carry == 0)
Then we do not write into any register.
Basically we can make RF-W signal 0, and the let the process execute as is.

ADZ

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T$_1$—>ALU2-A	ALU2_CONTROL ADD
T$_2$—> ALU2-B	T$_3$-W

ALU2-C —> T$_3$	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T$_3$—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

(If zero == 1)

Fetch	Controls
T$_{20}$—>RF-D$_3$	RF-W
IR-WB$_{3-5}$—> RF-A$_3$	

(If zero == 0)
Then we do not write into any register.
Basically we can make RF-W signal 0, and the let the process execute as is.

AWC

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T$_1$—>ALU3-A	ALU2_CONTROL ADD
CARRY—> ALU3-B	ALU3_CONTROL ADD
ALU3-C —> T$_4$	T$_3$-W

T4 —> ALU2-A	T_4-W
T2 —> ALU2-B	
ALU2-C —> T3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

ACA

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL ADD
ALU3-C —> T$_4$	T$_3$-W

T4—> ALU2-A	T_4-W
T1 —> ALU2-B	
ALU2-C —> T3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
IR-MR -----> IR-WB	
T_3—>T20	
IP-MR -----> IP-WB	

6. Write Back

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-MR_{3-5}—> RF-A_3	

ACC

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL ADD
ALU3-C —> T$_4$	T$_3$-W

T4 —> ALU2-A	T_4-W
T1 —> ALU2-B	
ALU2-C —> T3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

(If carry == 1)

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

(If carry == 0)
Then we do not write into any register.
Basically we can make RF-W signal 0, and the let the process execute as is.

ACZ

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL ADD
ALU3-C —> T$_4$	T$_3$-W

T4 —> ALU2-A	T_4-W
T1 —> ALU2-B	
ALU2-C —> T3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch Controls

T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

(If zero == 1)

Fetch Controls

T_{20}—>RF-D_3	RF-W
IR-$WB_{3\text{-}5}$—> RF-A_3	

(If zero == 0)
Then we do not write into any register.
Basically we can make RF-W signal 0, and the let the process execute as is.

ACW

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL ADD
ALU3-C —> T$_4$	ALU4_CONTROL ADD

T4 —> ALU4-A	T_3-W
CARRY —> ALU4-B	T4-W
ALU4-C —> T5	T5-W
T1—> ALU2-A	
T5—> ALU2-B	
ALU2-C —> T3	
ALU2-CARRY —--> CARRY	CARRY-W
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

ADI

1. Fetch instruction and IP update.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand and operand fetch

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4.. Execute

Fetch	Controls
T$_1$ —> ALU2-A	ALU2_CONTROL_ADD
IR-EX$_{0-5}$ —> SE[6] —> ALU2-B	T$_3$-W
ALU2-C —> T$_3$	
ALU2-CARRY —--> CARRY	CARRY-W

ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write back

Fetch	Controls
T_{20} —> RF-D_3	RF-W
IR-WB_{6-8} —> RF-A_3	

NDU

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-$ID_{9\text{-}11}$—>RF-A_1	
IR-$ID_{6\text{-}8}$ —> RF-A_2	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D_1 —--> T1	T_1-W
RF-D_2 —--> T2	T_2-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T_1—>ALU2-A	ALU2_CONTROL NAND
T_2—> ALU2-B	T_3-W
ALU2-C —> T_3	

ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-$WB_{3\text{-}5}$—> RF-A_3	

NDC

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T$_1$—>ALU2-A	ALU2_CONTROL NAND
T$_2$—> ALU2-B	T$_3$-W
ALU2-C —> T$_3$	
ALU2-Z —--> Z	ZERO-W

IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
IR-MR -----> IR-WB	
T_3—>T20	
IP-MR -----> IP-WB	

6. Write Back

(If carry == 1)

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-$WB_{3\text{-}5}$—> RF-A_3	

(If carry == 0)

Then we do not write into any register.

Basically we can make RF-W signal 0, and the let the process execute as is.

NDZ

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
T$_1$—>ALU2-A	ALU2_CONTROL NAND
T$_2$—> ALU2-B	T$_3$-W
ALU2-C —> T$_3$	
ALU2-Z —--> Z	ZERO-W

IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

(If zero == 1)

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-$WB_{3\text{-}5}$—> RF-A_3	

(If zero == 0)

Then we do not write into any register.

Basically we can make RF-W signal 0, and the let the process execute as is.

NCU

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL NAND
ALU3-C —> T$_4$	T$_3$-W

T4 —> ALU2-A	T_4-W
T1 —> ALU2-B	
ALU2-C —> T3	
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

NCC

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL NAND
ALU3-C —> T$_4$	T$_3$-W

T4 —> ALU2-A	T_4-W
T1 —> ALU2-B	
ALU2-C —> T3	
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back

(If carry == 1)

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-$WB_{3\text{-}5}$—> RF-A_3	

(If carry == 0)
Then we do not write into any register.
Basically we can make RF-W signal 0, and the let the process execute as is.

NCZ

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Execute

Fetch	Controls
‘1111111111111111’ —>ALU3-A	ALU3_CONTROL SUB
T2—> ALU3-B	ALU2_CONTROL ADD
ALU3-C —> T$_4$	T$_3$-W

T4 —> ALU2-A	T_4-W
T1 —> ALU2-B	
ALU2-C —> T3	
ALU2-Z —--> Z	ZERO-W
IR-EX -----> IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
T_3—>T20	
IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. Write Back
(If zero == 1)

Fetch	Controls
T_{20}—>RF-D_3	RF-W
IR-WB_{3-5}—> RF-A_3	

(If zero == 0)
Then we do not write into any register.
Basically we can make RF-W signal 0, and the let the process execute as is.

LLI

1. Fetch instruction and IP update IF

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR—>IR-ID	
IP-----> IP-ID	

2. ID—

Fetch	Controls
IR-ID—>IR-RR	
IP-ID ----> IP-RR	

3. Register read—

Fetch	Controls
IR-RR—>IR-EX	
IP-RR ----> IP-EX	

4. Execute—

Fetch	Controls
IR-EX—>IR-MR	
IP-EX -----> IP-MR	

5. Memory read

Fetch	Controls
IR-MR—>IR-WB	
IP-MR -----> IP-WB	

6. Update result onto Register.

Fetch Controls

IR-WB$_{0-8}$ —> SE[9] —> RF-D3	RF-W
IR-WB$_{9-11}$ —> RF-A$_3$	

LW

1. Fetch instruction and IP update.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR—>IR-ID	
IP-----> IP-ID	

2. Understand INSTRUCTION

Fetch	Controls
IR-ID$_{6\text{-}8}$—>RF-A$_1$	
IR-ID—>IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
IR-RR—>IR-EX	
IP-RR ----> IP-EX	

3. Compute Address

Fetch	Controls
T$_1$ —> ALU2-A	ALU2_CONTROL ADD
IR-EX$_{0\text{-}5}$ —> SE[6] —> ALU2-B	T$_3$-W
ALU2-C —> T$_3$	

ALU2-Z —--> Z	ZERO-W
IR-EX—>IR-MR	
IP-EX -----> IP-MR	

4. Read Memory)

Fetch	Controls
T_3—> Memory Address	Mem-Read
Memory Data —> T_{25}	T_{25}-W
T3—>T20	
IR-MR—>IR-WB	
IP-MR -----> IP-WB	

5. Update Register

T_{25}—> RF-D_3	RF-W
IR-$WB_{9\text{-}11}$—> RF-A_3	

SW

1. Fetch instruction and IP update

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID ----> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Compute Address

Fetch	Controls
T$_2$ —> ALU2-A	ALU2_CONTROL ADD
IR-EX$_{0-5}$ —> SE[6] —> ALU2-B	T$_3$-W
ALU2-C —> T$_3$	

IR-EX -----> IR-MR	
IP-EX -----> IP-MR	
T1—>T21	

5. Memory read

IR-MR -----> IR-WB	
T3—>T20	
IP-MR -----> IP-WB	
T21—>T22	

6. . Write back

Fetch	Controls
T_{20} —> Memory Address	Mem-Write
T_{22} —> Memory Data	

BEQ

1. Fetch instruction.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	IR-W
IR—>IR-ID	
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IP—--> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID—>IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR—>IR-EX	
IP-RR ----> IP-EX	

4.Compute if (R$_1$-R$_2$) = 0

Fetch	Controls
T$_1$—>ALU3-A	ALU3_CONTROL SUB
T$_2$—> ALU3-B	
ALU3-Z—> Z	ZERO-W

IR-EX —>IR-MR	
IP-EX -----> IP-MR	
T1—>T21	

IP-EX—>ALU2-A	ALU2_CONTROL ADD
IR-EX$_{0-5}$ —> SE6 (MULTIPLY BY 2)—> ALU2-B	
ALU2-C —> T3	
T3 —-> ALU4-A	ALU4_CONTROL SUB
+2 ——-> ALU4-B	
ALU4-C—–>T5	

5. Memory read

IR-MR -----> IR-WB	
T3—>T20	T20_W
T5 ——-> T23	
IP-MR -----> IP-WB	

6. Write back

If (Z==1) T23 —> IP Else IP-WB —> IP	IP_W

BLT

1. Fetch instruction.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	IR-W
IR—>IR-ID	
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IP—--> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9\text{-}11}$—>RF-A$_1$	
IR-ID$_{6\text{-}8}$ —> RF-A$_2$	
IR-ID—>IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR—>IR-EX	
IP-RR ----> IP-EX	

4.Compute if (R$_1$-R$_2$) = 0

Fetch	Controls
T$_1$—>ALU3-A	ALU3_CONTROL SUB
T$_2$—> ALU3-B	
ALU3-Z—> Z	ZERO-W

ALU3-CARRY —--> CARRY	CARRY-W
IR-EX —>IR-MR	
IP-EX -----> IP-MR	
T1—>T21	

IP-EX—>ALU2-A	ALU2_CONTROL ADD
IR-EX$_{0-5}$ —> SE6 (MULTIPLY BY 2)—> ALU2-B	
ALU2-C —> T3	
T3 —-> ALU4-A	ALU4_CONTROL SUB
+2 ——-> ALU4-B	
ALU4-C—-->T5	

5. Memory read

IR-MR -----> IR-WB	
T3—>T20	T20_W
T5 ——-> T23	
IP-MR -----> IP-WB	

6. Write back

If (Z==0) AND (C == 0) T23 —> IP Else IP-WB —> IP	IP_W

BLE

1. Fetch instruction.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	IR-W
IR—>IR-ID	
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IP—--> IP-ID	

2. Understand instruction

Fetch	Controls
IR-ID$_{9-11}$—>RF-A$_1$	
IR-ID$_{6-8}$ —> RF-A$_2$	
IR-ID—>IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W
RF-D$_2$ —--> T2	T$_2$-W
IR-RR—>IR-EX	
IP-RR ----> IP-EX	

4.Compute if (R$_1$-R$_2$) = 0

Fetch	Controls
T$_1$—>ALU3-A	ALU3_CONTROL SUB
T$_2$—> ALU3-B	
ALU3-Z—> Z	ZERO-W

ALU3-CARRY —--> CARRY	CARRY-W
IR-EX —>IR-MR	
IP-EX -----> IP-MR	
T1—>T21	

IP-EX—>ALU2-A	ALU2_CONTROL ADD
IR-EX$_{0-5}$ —> SE6 (MULTIPLY BY 2)—> ALU2-B	
ALU2-C —> T3	
T3 —-> ALU4-A	ALU4_CONTROL SUB
+2 ——-> ALU4-B	
ALU4-C—->T5	

5. Memory read

IR-MR -----> IR-WB	
T3—>T20	T20_W
T5 ——-> T23	
IP-MR -----> IP-WB	

6. Write back

If (Z==1) OR (C==0) T23 —> IP Else IP-WB —> IP	IP_W

JAL

1. Fetch instruction

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Instruction decode

IR-ID —-> IR-RR	
IP-ID ----> IP-RR	

3. Register Read

IR-RR ----> IR-EX	
IP-RR ----> IP-EX	

4. Compute Instruction pointer

Fetch	Controls
IP-EX —> ALU2-A	ALU2_CONTROL ADD
IR-EX$_{0-8}$ —> SE[9] (MULTIPLY BY 2)——-> ALU2-B	IP-W
ALU2-C —> T3	ALU3_CONTROL SUB
T3——-> ALU3-A	
+2——-> ALU3-B	
ALU3-C ——-> IP	
IP-EX -----> IP-MR	

IR-EX -----> IR-MR	

5. MEMORY READ

IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. WRITE BACK

IR-WB 9-11 —> RF-A_3	RF-W
IP-WB—--> RF-D3	

JLR

1. Fetch instruction

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	
IR-----> IR-ID	
IP-----> IP-ID	

2. Instruction decode

IR-ID —-> IR-RR	
IP-ID ----> IP-RR	
IR-ID6-8 —--> RF-A2	

3. Register Read

IR-RR ----> IR-EX	
IP-RR ----> IP-EX	
RF-D2 —-> T2	

4. Compute Instruction pointer

Fetch	Controls
IR-EX —-> IR-MR	
IP-EX -----> IP-MR	
T2 —> IP	

5. MEMORY READ

IR-MR -----> IR-WB	
IP-MR -----> IP-WB	

6. WRITE BACK

IR-WB 9-11 —> RF-A_3	RF-W
IP-WB—---> RF-D3	

JRI

1. Fetch instruction.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	
IR—-> IR-ID	
IP-----> IP-ID	

2. Understand instruction

Fetch	Controls
IR_{9-11}—>$RF-A_1$	
IR-ID —-> IR-RR	
IP-ID ----> IP-RR	

3. Register read

Fetch	Controls
$RF-D_1$ —--> T1	T_1-W
IR-RR—> IR-EX	
IP-RR ----> IP-EX	

4.. Compute Address

Fetch	Controls
T_1 —> ALU2-A	ALU2_CONTROL ADD
$IR-EX_{0-8}$ —> SE[9] (MULTIPLY BY 2) —> ALU2-B	T_3-W
ALU2-C —> T_3	
IR-EX —> IR-MR	
IP-EX -----> IP-MR	

5. Read Memory

Fetch	Controls
T_3—> Memory Address	Mem-Read
Memory Data —> T_{25}	T_3-W
IR-MR—-> IR-WB	
IP-MR -----> IP-WB	
T3 —--> T20	

6. Writing back into IP

T_{25}—>IP	IP-W

LM

Load multiple registers whose address is given in the immediate field (one bit per register, R0 to R7 from left to right) in reverse order from right to left, i.e, registers from R7 to R0 if the corresponding bit is set. Memory address is given in reg A. Registers which are expected to be loaded from consecutive memory addresses

1. Fetch instruction and IP update.

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	

2. Understand INSTRUCTION

Fetch	Controls
IR$_{9-11}$—>RF-A$_1$	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W

4. Compute Addresses, for each register

Fetch	Controls
T1 —-> ALU2-A	ALU2_CONTROL SUB
+2 —-> ALU2-B	T$_1$-W
ALU2C ——-> T1	

5. Read Memory —-- WILL HAVE TO ADD 8 STALLS (MULTIPLE STALLS — DEFINE CONTROL SIGNALS FOR A STALL— GIVE CONTROL BASED ON IMMEDIATE BITS)

Put a mux before memory add with 8 inputs t1, t6 to t12
The input in every cycle will be decided by the control signal
Control signal will be the count.
One more demux for mem data
And another count_reg variable to count the bit number. This count_reg acts as the control for demux.

Fetch	Controls
T_1—> Memory Address	Mem-Read
Memory Data —> REG0	REG0-W
T_6—> Memory Address	Mem-Read
Memory Data —> REG1	REG1-W
T_7—> Memory Address	Mem-Read
Memory Data —> REG2	REG2-W
T_8—> Memory Address	Mem-Read
Memory Data —> REG3	REG3-W
T_9—> Memory Address	Mem-Read
Memory Data —> REG4	REG4-W
T_{10}—> Memory Address	Mem-Read
Memory Data —> REG5	REG5-W
T_{11}—> Memory Address	Mem-Read
Memory Data —> REG6	REG6-W
T_{12}—> Memory Address	Mem-Read
Memory Data —> REG7	REG7-W

SM

Store multiple registers whose address is given in the immediate field (one bit per register, R0 to R7 from left to right) in reverse order from right to left, i.e, registers from R7 to R0 if corresponding bit is set. Memory address is given in reg A. Registers which are expected to store must be stored to consecutive addresses.

1. Fetch instruction and IP update. (S$_{41}$)

Fetch	Controls
IP —> Memory Address	Mem-Read
Memory Data —> IR	ALU1_CONTROL ADD
IP —> ALU1-A	IR-W
+2 —> ALU1-B	IP-W
ALU1-C —> IP	

2. Understand INSTRUCTION

Fetch	Controls
IR$_{9-11}$—>RF-A$_1$	

3. Register read

Fetch	Controls
RF-D$_1$ —--> T1	T$_1$-W

4. Compute Addresses, for each register

Fetch	Controls
T1 —-> ALU3-A	ALU3_CONTROL ADD
+2 —-> ALU3-B	T$_6$-W
ALU3C ——--> T6	
T6 ——--> ALU4-A	ALU4_CONTROL ADD
+2 —-> ALU4-B	T$_7$-W
ALU4-C —--> T7	

T7 —---> ALU5-A	ALU5_CONTROL ADD
+2 —---> ALU5-B	T_8-W
ALU5-C —---> T8	
T8 —--> ALU6-A	ALU6_CONTROL ADD
+2 —---> ALU6-B	T_9-W
ALU6-C —--> T9	
T9 —--> ALU7-A	ALU7_CONTROL ADD
+2 —---> ALU7-B	T_{10}-W
ALU7-C —--> T10	
T10 —--> ALU8-A	ALU8_CONTROL ADD
+2 —---> ALU8-B	T_{11}-W
ALU8-C —--> T11	
T11 —--> ALU9-A	ALU9_CONTROL ADD
+2 —---> ALU9-B	T_{12}-W
ALU9-C —--> T12	

5. Read Memory —-- WILL HAVE TO ADD 8 STALLS (MULTIPLE STALLS — DEFINE CONTROL SIGNALS FOR A STALL— GIVE CONTROL BASED ON IMMEDIATE BITS)

Fetch	Controls
T_1—> Memory Address	Mem-Read
REG0 —> Memory Data	mem-W
T_6—> Memory Address	Mem-Read
REG1 —> Memory Data	mem-W
T_7—> Memory Address	Mem-Read
REG2 —> Memory Data	mem-W
T_8—> Memory Address	Mem-Read
REG3 —> Memory Data	mem-W
T_9—> Memory Address	Mem-Read
REG4 —> Memory Data	mem-W

T_{10}—> Memory Address	Mem-Read
REG5 —> Memory Data	mem-W
T_{11}—> Memory Address	Mem-Read
REG6 —> Memory Data	mem-W
T_{12}—> Memory Address	Mem-Read
REG7 —> Memory Data	mem-W

Implementation of data forwarding

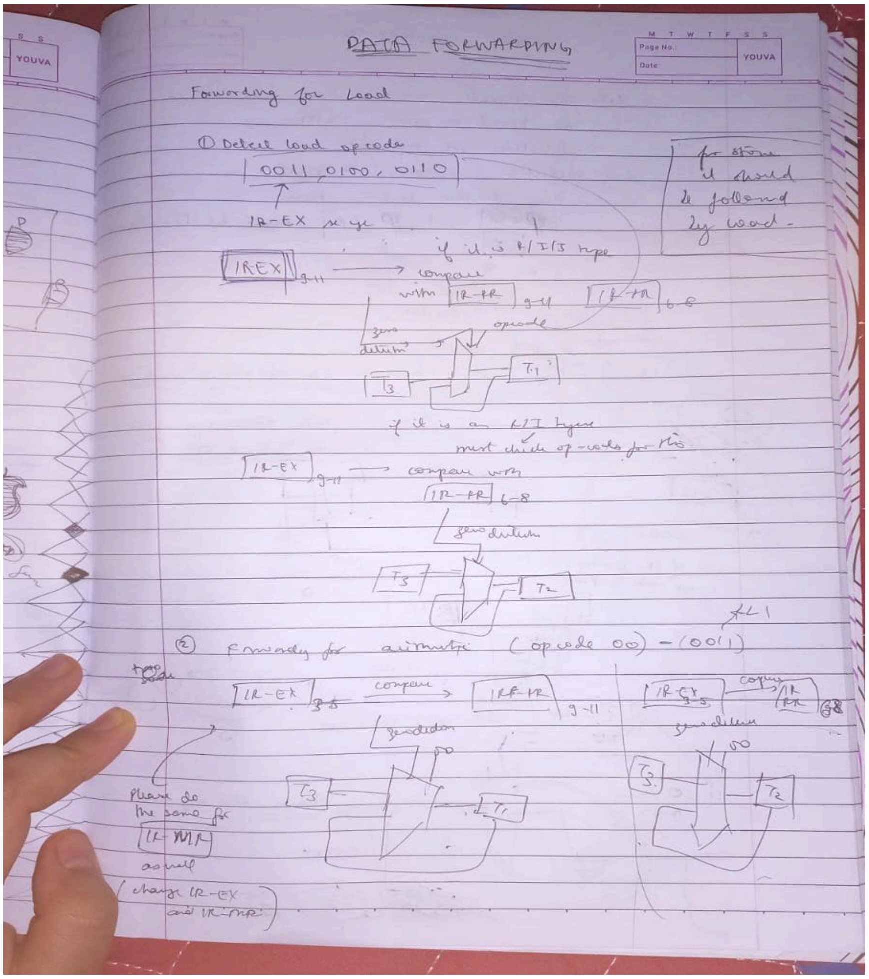
Say an instruction like ADA comes consecutively. Then if there is data dependency, the correct result wlll not be produced as the pipelined instructions' registers wont be updated faster than the speed at which the instructions are coming.
So for this we need to detect in the instructions where data dependency would be possible and then detect if actually there is a scope for data dependency

If first four bits of op-code are 00_01, 00_10 compareIR- EX3-5 with IR-RR8-8 and IR-RR9-11
If first four bits are 00_00, compare IR-EX6_8 with IR-RR8-8 and IR-RR9-11
If first four bits are 11_01, 11_00, 01_00, 01_01, 00_11 compare IR-EX9_11 with IR-RR8-8 and IR-RR9-11

We are using a comparator to check if the values are equal, and if they are equal then we are shorting T3 and T1 or T3 and T2 accordingly

Similarly we are also implementing data forwarding for when the data dependent instructions are not consecutive but two steps apart.

If first four bits of op-code are 00_01, 00_10 compare IR- MR3-5 with IR-RR8-8 and IR-RR9-11
If first four bits are 00_00, compare IR-MR6_8 with IR-RR8-8 and IR-RR9-11
If first four bits are 11_01, 11_00, 01_00, 01_01, 00_11 compare IR-MR9_11 with IR-RR8-8 and IR-RR9-11



Implementation of branch prediction

Lets say while encountering a branch instruction we make a prediction in the ID stage itself (based on some FSM- the most easy to implement is the history bit)

Now we must cross-check and calculate whether our prediction was true or false.

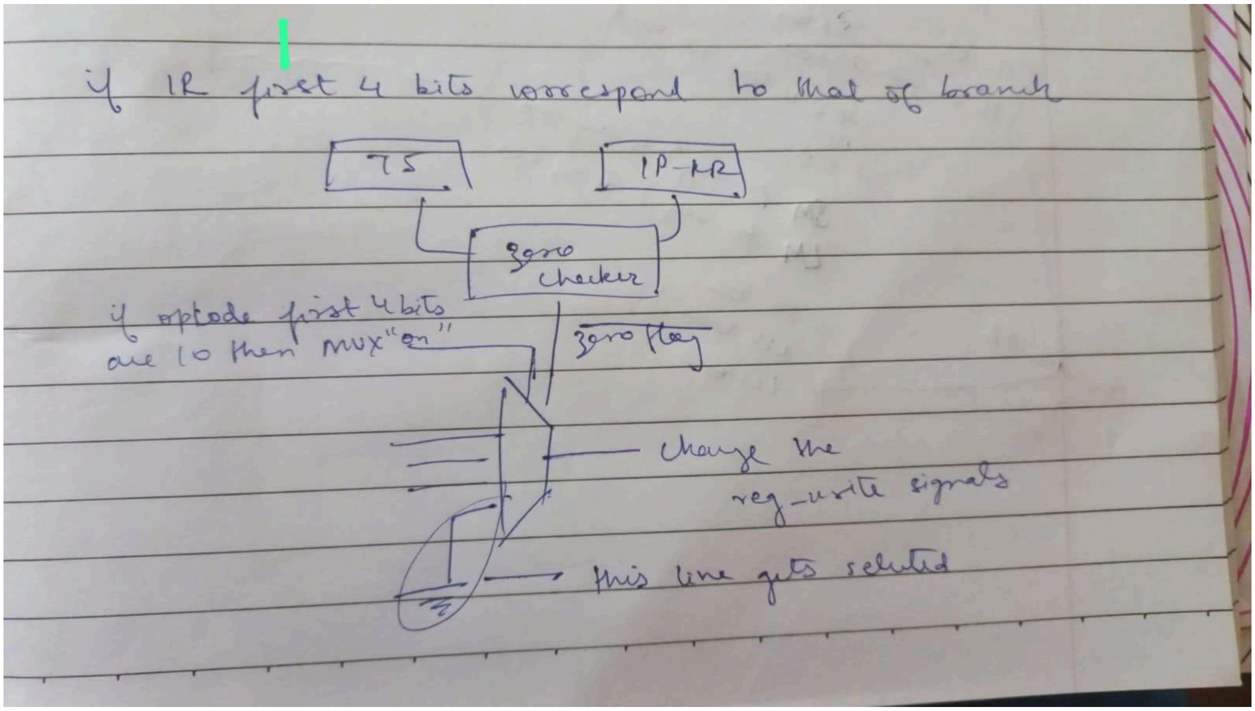
T5 and IP-RR must be the same if the prediction we have done is correct.

This will be done by using a comparator (subtractor) and checking the zero flag

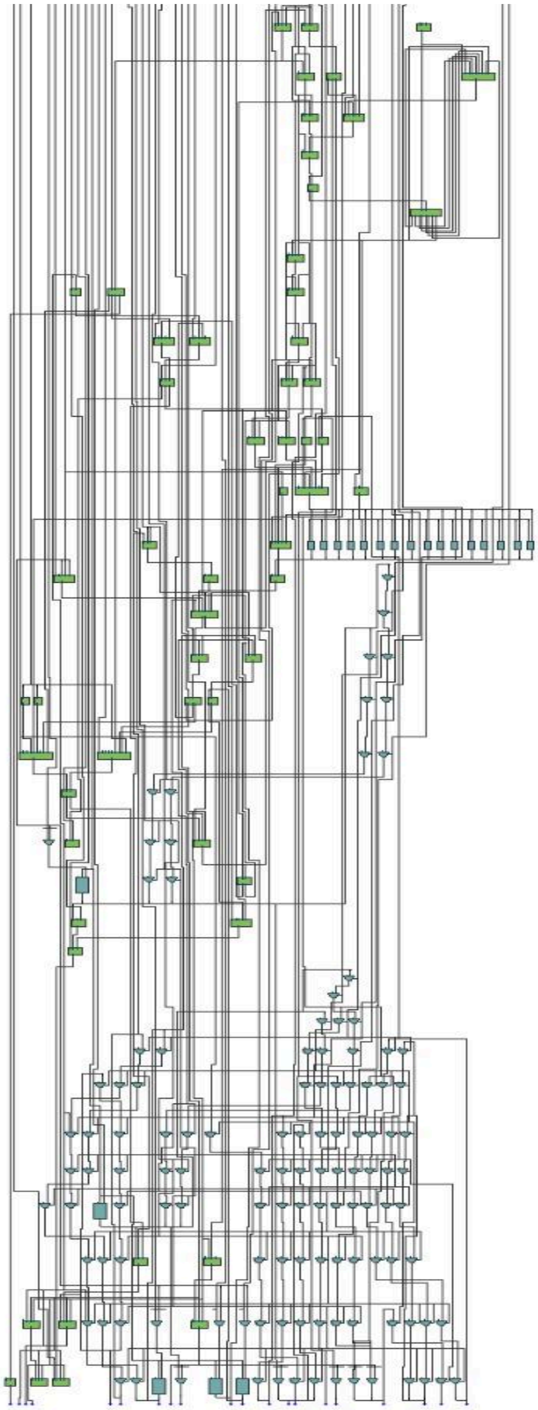
Say the values are different, then we must make the three instructions that entered the pipeline null and void. We do this by setting the write signals of all temporary registers as 0.

Hence effectively, even though the three instructions are being executed, they are making no changes, hence it is like they dont have any impact

However, on making wrong predictions the effective CPI will be hit, so it is important to have a good branch prediction.


if IR first 4 bits correspond to that of branch
T5
IP-RR
zero checker
if opcode first 4 bits are 10 then MUX "on"
zero flag
change the reg_write signals
this line gets selected

RTL Netlist (a portion of it)

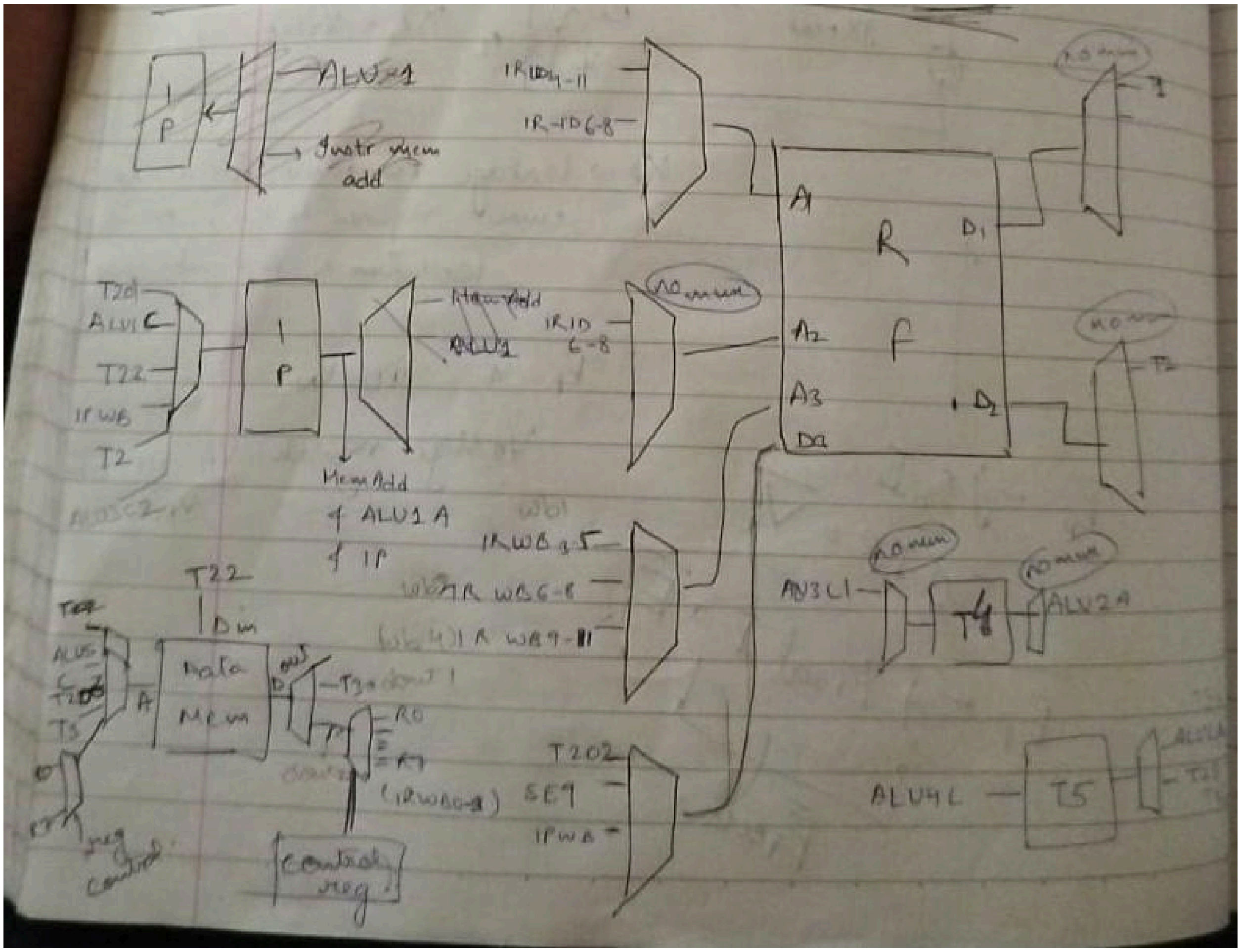


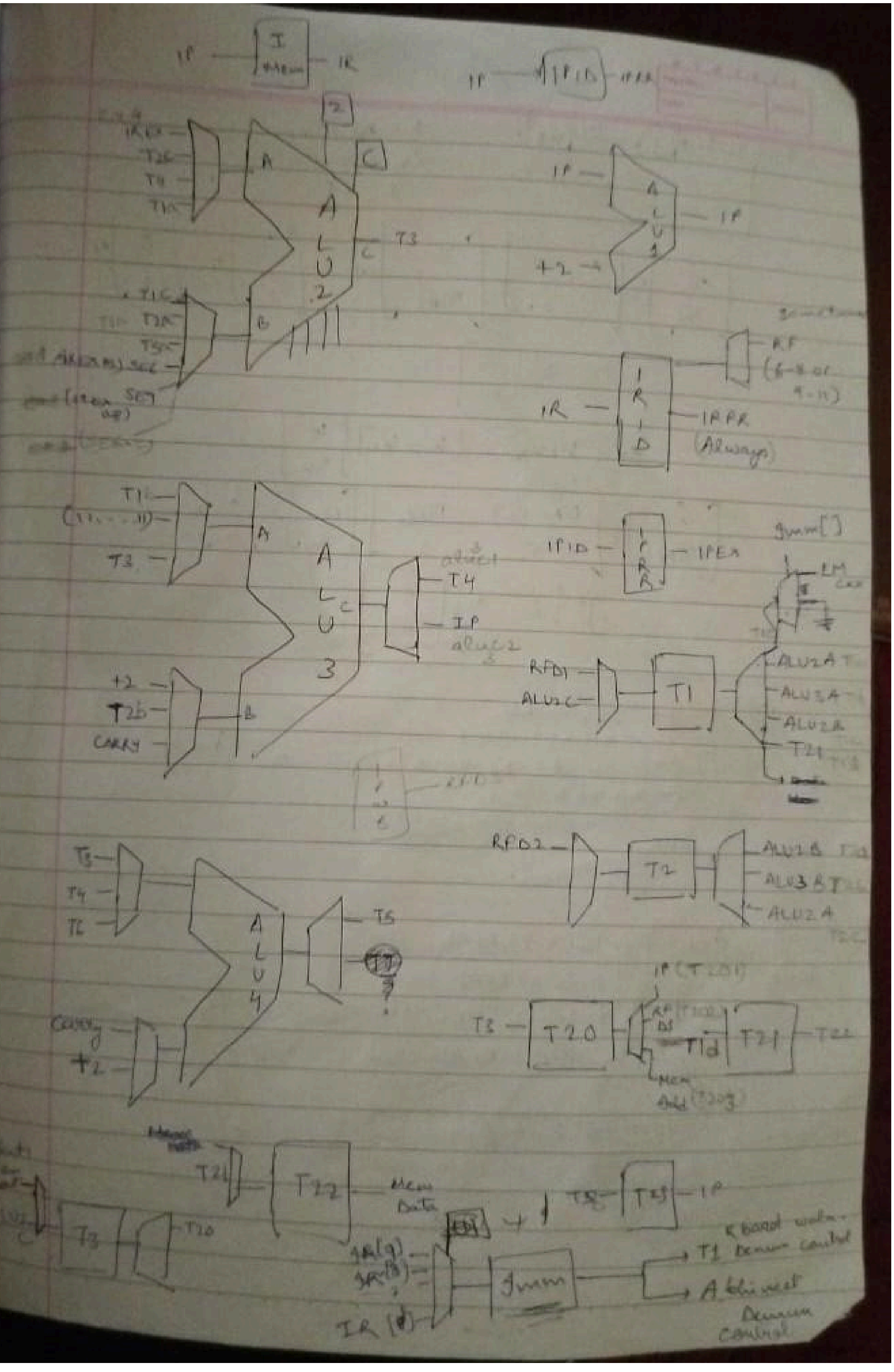
ALU Control Signals-

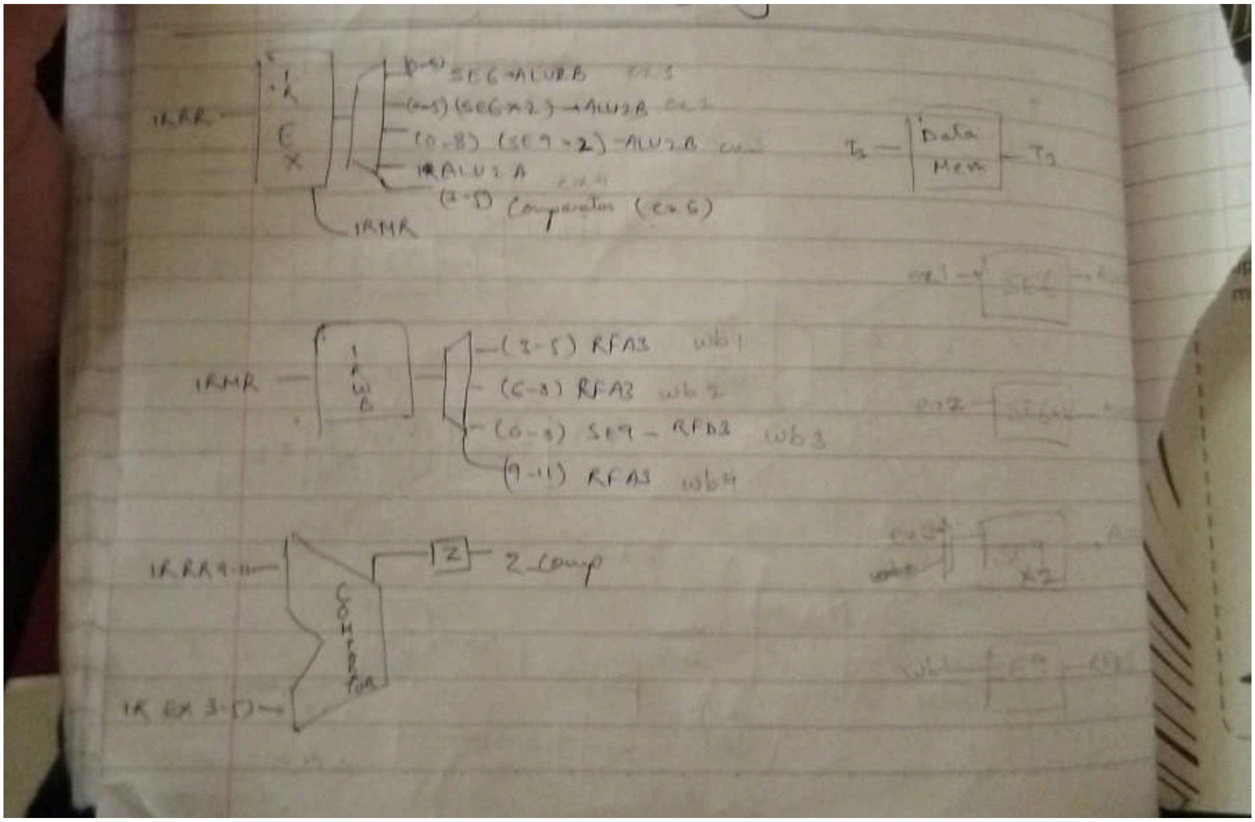
[ALU Controls]

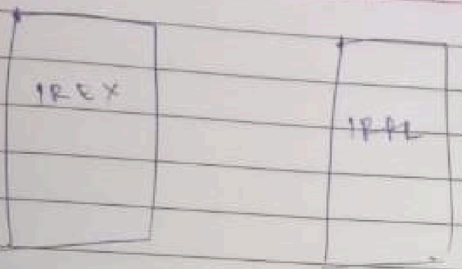
MUX Control Signals-

[MUX Controls]



IP
I
Mem
IR
IP
IPID
IPRR
IRID
T26
T4
T4
A
Z
C
A
L
U
2
C
T3
T1C
T2A
T5C
ARER(B) SEL
SE
IP
A
L
U
1
IP
+2
IR
RF
(6-8 or 9-11)
IRPR
(Always)
T1
(11...11)
T3
A
A
L
U
3
C
T4
IP
alu c2
+2
T2b
CARRY
B
IPID
IPRR
IPEX
Imm[]
EM
RFD1
ALU2C
T1
ALU2A
ALU3A
ALU2B
T21
RFD2
T2
ALU2B
ALU3B
ALU2A
T3
T4
T6
A
L
U
4
T5
T7
?
Carry
+2
IP (T201)
RF D1
T1d
T3
T20
T21
T22
Mem Add
Mem Data
T21
T22
Mem Data
data Mem
R(U)
T3
T20
+
T8
T23
IP
IR(9)
IR(8)
IR(8)
Imm
R board value
T1 Demux control
A Demux Demux Control





→ if op-code 00-01, 00-10} compare $EX_{2-5} \longrightarrow RR_{6-8} / RR_{9-11}$

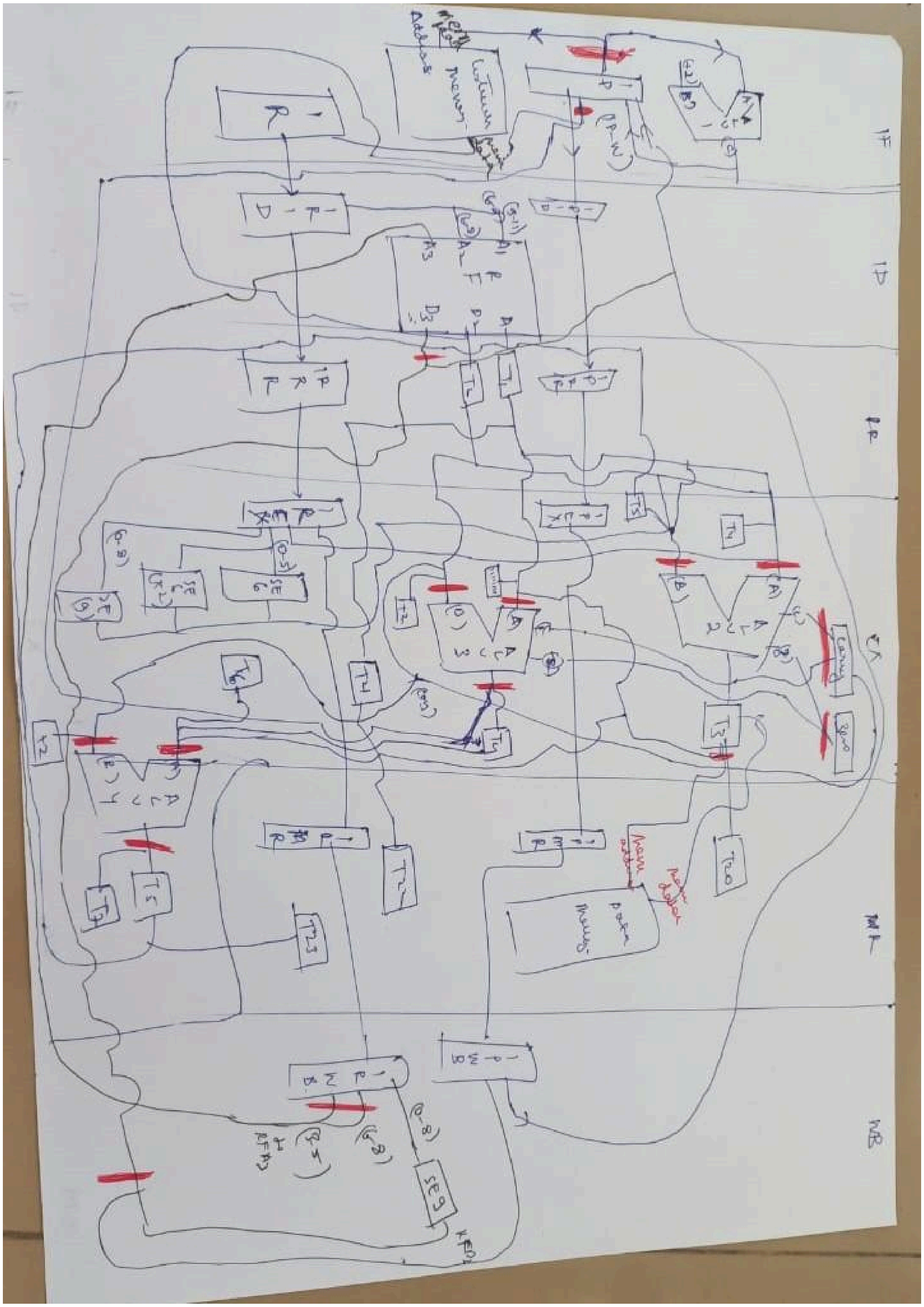
→ if op-code 00-00} compare $EX_{6-8} \longrightarrow RR_{6-8} / RR_{9-11}$

→ if op-code 11-01, 11-00, 01-00, 01-01, 00-11} compare $EX_{9-11} \longrightarrow RR_{6-8} / RR_{9-11}$

we have purposefully left out somethings since we ~~will~~ are not changing register/store type. Hence they should NOT be forwarded

→ in LM and SM we need to check for register 7 usage
01-10 01-11
this is what should be check
if IR_RR 6-8 / IR-RR 9-11 is 111 or not

Rough Datapath of CPU-


IF
ID
RR
EX
MA
WB
Instruction Memory
Data Memory
ALU 1
ALU 2
ALU 3
ALU 4
RF
IR
PC
SE6
SE9
Carry
Zero
T1
T2
T3
T4
T5
T6
T7
T10
T23