Design made by the team comprising of:

- Mayank Gupta, 210101002
- Ayush Joshi, 210100036
- Soham Rahul Inamdar, 210100149
- Harsh Amit Shah, 210100063

ADD:

S1: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable
if ((!Z and $T1_0$) or (!C and $T1_1$) = 1) then Back to S1	

S3: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
$T1_{8-6}$ → RF-A2	T3-Enable
RF-D1 → $T2$	
RF-D2 → $T3$	

S6: Execution

$T2$ → ALU-A	
$T3$ → ALU-B	
ALU-C → $T2$	ADD
if (TL_{15} = 0) then ALU-zero → Z	Set zero flag
if ($T1_{15-12}$ = 0000) then ALU-carry → C	Set carry flag

S13: Store

$T2$ → RF-D3	RFW-Enable
$T1_{5-3}$ → RF-A3	

ADC:

S1: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable
if ((!Z and $T1_0$) or (!C and $T1_1$) = 1) then Back to S1	

S3: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
$T1_{8-6}$ → RF-A2	T3-Enable
RF-D1 → $T2$	
RF-D2 → $T3$	

S6: Execution

$T2$ → ALU-A	
$T3$ → ALU-B	
ALU-C → $T2$	ADD
if (TL_{15} = 0) then ALU-zero → Z	Set zero flag
if ($T1_{15\text{-}12}$ = 0000) then ALU-carry → C	Set carry flag

S13: Store

$T2$ → RF-D3	RFW-Enable
$T1_{5-3}$ → RF-A3	

ADZ:

S1: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable

PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable
if ((!Z and $T1_0$) or (!C and $T1_1$) = 1) then Back to S1	

S3: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
$T1_{8-6}$ → RF-A2	T3-Enable
RF-D1 → $T2$	
RF-D2 → $T3$	

S6: Execution

$T2$ → ALU-A	
$T3$ → ALU-B	
ALU-C → $T2$	ADD
if (TL_{15} = 0) then ALU-zero → Z	Set zero flag
if ($T1_{15-12}$ = 0000) then ALU-carry → C	Set carry flag

S13: Store

$T2$ → RF-D3	RFW-Enable
$T1_{5-3}$ → RF-A3	

ADI:

S2: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable

S4: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
RF-D1 → $T2$	

S7: Execution

$T2$ → ALU-A	
$T1_{5-0}$ → SE 10 → ALU-B	ADD
ALU-C → $T2$	
if (TL_{14} xor TL_{12} = 1) then ALU-zero → Z	Set zero flag
if (!TL_{14} · $T1_{12}$ = 1) then ALU-carry → C	Set carry flag

S14: Store

$T2$ → RF-D3	RFW-Enable
$T1_{8-6}$ → RF-A3	

NDU:

S1: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable
if ((!Z and $T1_0$) or (!C and $T1_1$) = 1) then Back to S1	

S3: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
$T1_{8-6}$ → RF-A2	T3-Enable
RF-D1 → $T2$	
RF-D2 → $T3$	

S6: Execution

$T2$ → ALU-A	

$T3$ → ALU-B	
ALU-C → $T2$	NAND
if (TL_{15} = 0) then ALU-zero → Z	Set zero flag
if ($T1_{15\text{-}12}$ = 0000) then ALU-carry → C	Set carry flag

S13: Store

$T2$ → RF-D3	RFW-Enable
$T1_{5-3}$ → RF-A3	

NDC:

S1: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable
if ((!Z and $T1_0$) or (!C and $T1_1$) = 1) then Back to S1	

S3: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
$T1_{8-6}$ → RF-A2	T3-Enable
RF-D1 → $T2$	
RF-D2 → $T3$	

S6: Execution

$T2$ → ALU-A	
$T3$ → ALU-B	
ALU-C → $T2$	NAND
if (TL_{15} = 0) then ALU-zero → Z	Set zero flag
if ($T1_{15\text{-}12}$ = 0000) then ALU-carry → C	Set carry flag

S13: Store

$T2$ → RF-D3	RFW-Enable
$T1_{5-3}$ → RF-A3	

NDZ:

S1: Fetching instruction

PC → Mem_Address	Memory Read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC enable
if ((!Z and $T1_0$) or (!C and $T1_1$) = 1) then Back to S1	

S3: Understand and read operands

$T1_{11-9}$ → RF-A1	T2-Enable
$T1_{8-6}$ → RF-A2	T3-Enable
RF-D1 → $T2$	
RF-D2 → $T3$	

S6: Execution

$T2$ → ALU-A	
$T3$ → ALU-B	
ALU-C → $T2$	NAND
if (TL_{15} = 0) then ALU-zero → Z	Set zero flag
if ($T1_{15\text{-}12}$ = 0000) then ALU-carry → C	Set carry flag

S13: Store

$T2$ → RF-D3	RFW-Enable
$T1_{5-3}$ → RF-A3	

LHI:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-Enable

S4: Understand and read operands

$T1_{11-9}$ → RF_A1	
RF_D1 → T2	T2-Enable

S8: Execution

$T1_{8-0}$ → T2	T2-Enable
T2 → 7LShifter	Left_Shift_by_7bits
7LShifter → T2	T2-Enable

S19: Store

T2 → RF_D3	RF-WE
$T1_{11-9}$ → RF_A3	

LW:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-E

S5: Understand and read operands

$T1_{8-6}$ → RF_A1	
RF_D1 → T2	T2-Enable

S7: Execution

$T2$ → ALU-A	
$T1_{5-0}$ → SE 10 → ALU-B	ADD
ALU-C → $T2$	
if (TL_{14} xor TL_{12} = 1) then ALU-zero → Z	Set zero flag
if (!TL_{14} · $T1_{12}$ = 1) then ALU-carry → C	Set carry flag

S16: Load memory into register

T2 → Mem_Address	MRD
Mem_Data → T2	T2-E

S19: Store

T2 → RF_D3	
$T1_{11-9}$ → RF_A3	RF-WE

SW:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-E

S3: Understand and read operands

$T1_{8-6}$ → RF_A1	
RF_D1 → T2	T2-Enable
$T1_{11-9}$ → RF_A2	
RF_D2 → T3	

S7: Execution

$T2$ → ALU-A	

$T1_{5-0}$ → SE 10 → ALU-B	ADD
ALU-C → $T2$	
if (TL_{14} xor TL_{12} = 1) then ALU-zero → Z	Set zero flag
if (!TL_{14} · $T1_{12}$ = 1) then ALU-carry → C	Set carry flag

S17: Store value to memory

T2 → Mem_Address	Memory write
T3 → Mem_Data	

S19: Store

$T1_{11-9}$ → RF_A3	RF-WE
T2 → RF_D3	

LM:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-E

S4: Understand and read operands

$T1_{11-9}$ → RF_A1	
RF_D1 → T2	T2-Enable

S9: Execution

For loop i (0 to 7){
If ($T1_i$ = 1) then
T2→ Memory_address
Memory_data→ T3
R(i)→ RF_A3
	Used for loop to check which bit is 1.
	If the bit is 1, we need to load the memory of address stored in T2.

T3 → RF_D3 T2→ ALU-A +2 → ALU-B ALU-C→ T2 }	If for loop isn't possible to fabricate on the FPGA board, then 32 states will have to be defined.

SM:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-E

S4: Understand and read operands

$T1_{11-9}$ → RF_A1	
RF_D1 → T2	T2-Enable

S10: Execution

For loop i (0 to 7){
If ($T1_i$ = 1) then
R(i)→ RF_A1
RF_D1 → T3
T2→ Memory_address
T3→ Memory_data
T2→ ALU-A
+2 → ALU-B
ALU-C→ T2
}
	Used for loop to check which bit is 1.
	If the bit is 1, we need to store the value of R(i) in the memory of address stored in T2.
	If for loop isn't possible to fabricate on the FPGA board, then 32 states will have to be defined.

BEQ:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-E

S3: Understand and read operands

$T1_{11\text{-}9} \rightarrow$ RF_A1	
RF_D1 → T2	T2-Enable
$T1_{8-6} \rightarrow$ RF_A2	
RF_D2 → T3	

S6: Execution

$T2 \rightarrow$ ALU-A	
$T3 \rightarrow$ ALU-B	
ALU-C → $T2$	SUBTRACT
if ($TL_{15} = 0$) then ALU-zero → Z	Set zero flag
if ($T1_{15\text{-}12} = 0000$) then ALU-carry → C	Set carry flag

S18: Update PC

PC+4 → ALU_A	ADD
$T1_{5-0} \rightarrow$ SE → ALU_B	
If (z) then ALU_C→ PC	

JAL:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-Enable

S11: Execution

PC→ ALU-A	
$T1_{8\text{-}0} \rightarrow$ SE 7 → ALU-B	
ALU-C → PC	PC-Enable

$T1_{8-0}$ → T2	T2-Enable

S15: Store

PC → RF_D3	RF-Write Enable
$T1_{11-9}$ → RF_A3	

JLR:

S2: Fetching instruction

PC → Mem_Address	Memory read
Mem_Data → T1	T1-Enable
PC → ALU-A	ADD
+2 → ALU-B	
ALU-C → PC	PC-Enable

S5: Understand and read operands

$T1_{8-6}$ → RF_A1	
RF_D1 → T2	T2-Enable

S12: Execution

PC→ ALU-A	
T2→ ALU-B	
ALU-C → PC	PC-Enable

S15: Store

PC → RF_D3	RF-Write Enable
$T1_{11-9}$ → RF_A3	