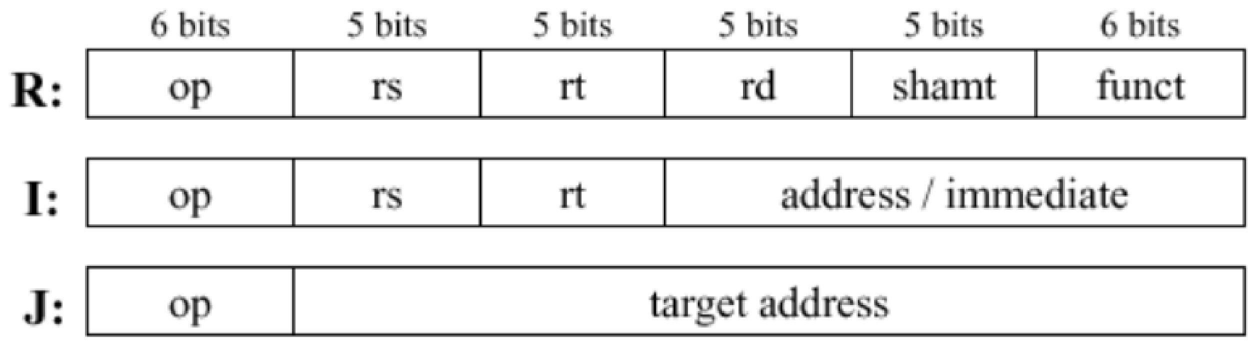
Problem Statement:
Implementing the following MIPS instructions:

Instruction	Opcode/funct(hex)	description	
ADD	0/20	R[rd] = R[rs] + R[rt]	Rs,rt read /rd write
SUB	0/22	R[rd] =R[rs] - R[rt]	Rs,rt read/ rd write
AND	0/24	R[rd] = R[rs] & R[rt]	“
OR	0/25	R[rd] = R[rs] | R[rt]	“
NOR	0/27	R[rd] =~(R[rs] |R[rt])	“
SLL	0/00	R[rd] = R[rt] << shamt	Rt read/rd
SRL	0/02	R[rd] = R[rt] >> shamt	“
ADDI	8	R[rt] = R[rs] + SignExtImm	Rs read/ rt write
ORI	d	R[rt] = R[rs] | ZeroExtImm	Rs read/ rt write
ANDI	c	R[rt] = R[rs] & ZeroExtImm	“
LD	23	R[rt]= M[R[rs]+SignExtImm]	Rs read/ rt write
ST	2b	M[R[rs]+SignExtImm] = R[rt]	Rs, rt read
J	2	PC=JumpAddr	
BEQ	4	if(R[rs]=R[rt]) PC=PC+4+BranchAddr*4	
BNE	5	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr*4	



	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R:	op	rs	rt	rd	shamt	funct
I:	op	rs	rt	address / immediate		
J:	op	target address				

A is always R[rs]
B is always R[rt]
ALUOut always PC+signextimm<<2

Inputs to ALU:

ALUSrcA	ALUSrcB
R[rs]	R[rt]
shamt	SignExtImm
PC	ZeroExtImm
	SignExtImm<<2
	4

STAGES:

IF:

IR = IM[PC]

PC =PC +4

ID:

A= REG[IR[25-21]]

B=REG[IR[20-16]]

ALUOut= PC + (SignExtendIR[15-0]<<2)

EX:

R Type

ALUOut = A op B

Memory Reference:

ALUOut = A+SignExtendimm<<2

Branch

If(A == B) PC = ALUOut

J

PC = PC[31-28] concat IR[26-0]<<2

MEM:

If LD:

MDR = addr[ALUOut]

If ST:

Addr[ALUOut] = B

R type/

Reg[WN] = ALUOut

WB:

Reg[WN] = MDR

Control signals:

Control signal	When asserted	When deasserted
PCWrite	Allows PC to be written into	
MemReadI	Read instr pointed by Addr	
MemRead	“	
MemWrite	Write into Addr WriteData of DM	
MemtoReg	WD<= ALUOut	WD<= MDR
IRWrite	When IR can be written into	
RegDst	WN<= rd	WN<= rt
RegWrite	Reg[WN]<=WD	
ALUSrcA	00:PC 01: A 10: shamt = IR[11:6]	
ALUSrcB	000: 4 001: B 010: SignExtImm 011: SignExtImm<<2 100: ZeroExtImm	
ALUOp	100: Add 110: Sub 011: And 001: Or 101: Nor 010: ShiftRight 000: ShiftLeft	
PCSrc	00: PC[31:28]concatIR[26-0]<<2 01: PC = ALUResult 10: PC = ALUOut	
Branch//using as internal signal in control unit	If its a branch condition	
ALUOutEn	Write into ALUOut	

STATE FSM

State 0	IF(common for all instr)
	
	IR = IM[PC]
	PC =PC +4
		MemReadI= 1
		PCWrite = 1
		IRWrite = 1
		RegWrite = 0
		PCSrc = 01
		ALUSrcA = 00
		ALUSrcB = 000
		ALUOp = 100(add)

State 1	ID(common for all instr)
	A= REG[IR[25-21]]
	B=REG[IR[20-16]]
	ALUOut= PC +
	(SignExtendIR[15-0]<<2)
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don’t care

State 2	EX ADD
	ALUOut = A + B
		ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 100
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don’t care

State 3	MEM R Type
	Reg[Wn] = ALUOut
		ALUOutEn = 0
		MemtoReg = 1
		RegDst = 1
		RegWrite = 1
		PCWrite = 0
		MemReadI = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		All other dont care

State 4	EX SUB
	ALUOut = A- B
		ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 110
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0

		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 5	EX AND
	ALUOut = A&B
		ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 011
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 6	EX OR
	ALUOut A|B
		ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 001
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 7	EX NOR
	ALUOut = !(A||B)
		ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 101
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 8	EX SLL	ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 000
		ALUOutEn = 1

		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 9	EX SRL	ALUSrcA= 01
		ALUSrcB =001
		ALUOp = 010
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 10	EX ADDI
	ALUOut = A+SignExtendimm
		ALUSrcA= 01
		ALUSrcB =010
		ALUOp = 100
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 11	MEM I type	ALUOutEn = 0
		MemtoReg = 1
		RegDst = 0
		RegWrite = 1
		PCWrite = 0
		MemReadI = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		All other dont care

State 12	EX ORI	ALUSrcA= 01
		ALUSrcB =010
		ALUOp = 001
		ALUOutEn = 1

		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 13	EX ANDI	ALUSrcA= 01
		ALUSrcB =010
		ALUOp = 011
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 14	EX LD/ST
	ALUOut = A+SignExtendimm<<2
		ALUSrcA= 01
		ALUSrcB =011
		ALUOp = 100
		ALUOutEn = 1
		MemReadI = 0
		PCWrite = 0
		RegWrite = 0
		MemRead = 0
		MemWrite = 0
		IRWrite = 0
		Branch = 0
		All other don't care

State 15	MEM Ld
	MDR = addr[ALUOut]
		MemWrite = 0
		MemRead = 1
		MemReadI = 0
		PCWrite = 0
		IRWrite = 0
		ALUOutEn = 0
		RegWrite = 0
		All other dont care

State 16	WB LD	MemWrite = 0
		MemRead =0
		MemReadI =0
		PCWrite =0
		IRWrite =0
		RegWrite =1

		RegDst = 0
		MemtoReg =0
		All other dont care

State 17	MEM ST
	Addr[ALUOut] = B
		ALUOutEn = 0
		MemWrite =1
		MemRead =0
		MemReadI =0
		IRWrite =0
		RegWrite =0
		PCWrite =0
		All other dont care

State 18	EX J
	PC = PC[31-28] concat
	IR[26-0]<<2
		PCSrc =00
		PCWrite = 1
		MemReadI =0
		All other dont care

State 19	EX BEQ
	If(A == B) PC = ALUOut
		Branch = 1
		ALUSrcA = 01
		ALUSrcB =01
		ALUOp = 110
		PCWrite = Branch & Zero
		PCSrc = 10
		ALUOutEn = 0
		MemReadI = 0
		All other dont care

State 20	EX BNE	Branch = 1
		ALUSrcA = 01
		ALUSrcB =01
		ALUOp = 110
		PCWrite = Branch & !Zero
		PCSrc = 10
		ALUOutEn = 0
		MemReadI = 0
		All other dont care

State flow:
ADD:
State 0 -> 1 ->2->3->0
SUB:
State 0->1->4->3->0
AND:
State 0->1->5->3->0
OR:
State 0->1->6->3->0
NOR:
State 0->1->7->3->0
SLL:
State 0->1->8->3->0
SRL:
State 0->1->9->3->0
ADDI:
State 0->1->10->11->0
ORI:
State 0->1->12->11->0
ANDI:
State 0->1->13->11->0
LD:
State 0->1->14->15->16->0
ST:
State 0->1->14->17->0
J:
State 0->1->18->0
BEQ:
State 0->1->19->0
BNE:
State 0->1->20->0

What modules do i need?
There’s an FSM that takes care of transitioning between IF,ID,EX,MEM,WB
Define the states of the fsm, along with control signals, and next state give present state(control_unit).
Define the ALU.
Define ALUOut
Define the register file, and registers A and B
Define muxes ALU_A, ALU_B, PC_select,Mem_to_Reg
Define the PC, IR, MDR
Lastly define the instruction and data memory.(make files)

Define the top module

ALU
Inputs: ALUOp, ALU_A, ALU_B,clk
Outputs: Zero, ALU_Result

PC_mux
Inputs: PCSrc, PC_Jump, PC_ALUOut,PC_ALUresult
Outputs: PC

ALU_A_mux:
Inputs : PC,Reg_A,shamt, ALUSrcA
Outputs : ALU_A

ALU_B_mux:
Inputs : Reg_B, SignExtImm, SignExtAddr, ZeroExtImm, ALUSrcB
Outputs: ALU_B

ALUOut:
Inputs : ALU_Result, clk, ALUOutEn,reset
Output: ALUOut

Reg_A, Reg_B, MDR

Register_File:
Inputs : Read_reg_1, Read_reg_2, Write_reg, RegWrite, write_data, clk
Outputs: Read_data_1, Read_data_2

Instr_reg:
Inputs: IRWrite, Instr, clk
Outputs: Reg_rs,Reg_rt, Reg_rd, Imm, shamt, funct, opcode

PC_reg:
Inputs : PCWrite, PC,clk
Outputs: PC

MemtoReg_mux:
Inputs: mem_data, ALUOut, MemtoReg
Outputs: Reg_write_data

Instr_mem:
Inputs: Instr_Addr, MemReadI,clk
Outputs: Instr

Data_mem:
Inputs: Data_Addr, MemRead, MemWrite, Mem_write_data,clk
Outputs: Mem_data

Control unit:
Inputs: opcode, funct, Zero, clk, reset
Outputs: PCSrc, ALUSrcA, ALUSrcB, ALUOp, ALUOutEn, RegWrite, IRWrite, RegDst, MemWrite, MemRead, MemReadI, MemtoReg, PCWrite
Branch is an internal signal i suppose?