Signals Summary

Signals Needed: Signals needed for this stage
Signals Generated: Signals that are generated at each specific stage
Signals Passed: Signals that move towards the next stage in the pipeline

	IF	IDDR	EX	MEM	WB
Signals Needed	Stall BranchPred	isImm Stall	isImm isJump isBranch isALU ALUfunc isComp	isStore	WrSignal isLoad
Signals Generated	None	isImm	isJump isBranch isALU ALUfunc isComp	isStore	isLoad
Signals Passed	None	isImm	WrSignal	WrSignal	None

This table shows how the signals that are not generated by any stage are generated and how they change the behaviour of the pipeline

Signal	How is Generated?	Use
Stall	This signal is generated by the Stalling unit through the comparison between the registers needed by the instruction and the register currently being written by any other previous instruction	It sends a NOP through the IDDR/EX phase. Also, it avoids the PC to being increased as well as avoid clocking new values in the IF/IDDR Buffer.
BranchPred	It is generated if there is a Branch/Jump instruction in the IDDR or in the EX phase of the pipeline.	It sends a NOP through the IF/IDDR Buffer and avoids the value to be clock into the latches. Also, it restrains the PC to being incremented until the branching is resolved.

Signals Boolean Expression

Signal	Expression/Explanation
isImm	OR of first 6-bits of instruction
isALU	OPCODE[5]
isComp	!OPCODE[5] & !OPCODE[4] & OPCODE[3:0]
isBranch	isComp & isImm
isJump	isComp & OPCODE[2]
WrSignal	isALU OR !isImm OR isJump
isLoad	OPCODE == 6b010010
isStore	OPCODE == 6b011010
ALUfunc	OPCODE[2:0] or 3b011 if OPCODE == RSHF

Part 2: Cycle calculation

Case1:

1) addi s1,s1,0x1
2) addi s1,s1,0x1
3) addi s1,s1,0x1

IF	IDDR	EX	MEM	WB
1				
2	1			
3	2	1		
3	2	NOP	1	
3	2	NOP	NOP	1
	3	2	NOP	NOP

	3	NOP	2	NOP
	3	NOP	NOP	2
		3	NOP	NOP
			3	NOP
				3

Answer: 12 cycles

Case2:

1) addi s2, s1,0x1
2) subi a0,s1,0x1
3) jal t1, 0(t0)

IF	IDDR	EX	MEM	WB
1				
2	1			
3	2	1		
	3	2	1	
		3	2	1
			3	2
				3

Answer: 8 cycles

Part3: Binary Code

Case2

1) addi s1,s1,0x1 = 10000 | 0000000000000001 | 1000 | 1000
2) addi s1,s1,0x1 = 10000 | 0000000000000001 | 1000 | 1000
3) addi s1,s1,0x1 = 10000 | 0000000000000001 | 1000 | 1000

Case2

1) addi s2, s1,0x1 = 10000 | 0000000000000001 | 1001 | 1000
2) subi a0,s1,0x1 = 10000 | 1111111111111111 | 0001 | 1000
3) jal t1, 0(t0) = 001100 | 0000000000000000 | 0101 | 0110