In this project, you are required to extend the MIPS single-cycle implementation by implementing additional instructions. You will use ModelSim simulator [1] to develop and test your code. You are required to implement your assigned **6 instructions** (selected from following 25 instructions). *Note that your set of* ***6 instructions*** *will be emailed to you or your group member.*

R-format (11): brn, brz, balrn, balrz, sll, srl, sllv, srlv, jmadd, jmor, jmsub

I-format (10):bmz, bmn, bmv, balmz, balmn, balmv, jm, jalm, jrs, jrsal

J-format (4): bn, bz, baln, balz,

You must design the revised single-cycle datapath and revised control units which make a processor that executes your instructions as well as the instructions implemented already in the design. After designing the new enhanced processor, you will implement it in Verilog HDL.

Specifications of New Instructions

	Instr	Type	Code	Syntax	Meaning
1.	bmz	I-type	opcode=20	bmz imm16($rs)	if Status [Z] = 1, branches to address found in memory
2.	bmn	I-type	opcode=21	bmn imm16($rs)	if Status [N] = 1, branches to address found in memory
3.	bmv	I-type	opcode=22	bmv imm16($rs)	if Status [V] = 1, branches to address found in memory
4.	balmz	I-type	opcode=23	balmz $rt, imm16($rs)	if Status [Z] = 1, branches to address found in memory, link address is stored in $rt
5.	balmn	I-type	opcode=24	balmn $rt, imm16($rs)	if Status [N] = 1, branches to address found in memory, link address is stored in $rt
6.	balmv	I-type	opcode=25	balmv $rt, imm16($rs)	if Status [V] = 1, branches to address found in memory, link address is stored in $rt
7.	brz	R-type	funct=20	brz $rs	if Status [Z] = 1, branches to address found in register $rs.
8.	brn	R-type	funct=21	brn $rs	if Status [N] = 1, branches to address found in register $rs.
9.	balrz	R-type	funct=22	balrz $rs, $rd	if Status [Z] = 1, branches to address found in register $rs link address is stored in $rd (which defaults to 31)
10.	balrn	R-type	funct=23	balrn $rs, $rd	if Status [N] = 1, branches to address found in register $rs link address is stored in $rd (which defaults to 31)

11.	jmadd	R-type	funct=32	jmadd $rs,$rt	jumps to address found in memory [$rs+$rt], link address is stored in $31
12.	jmor	R-type	funct=37	jmor $rs,$rt	jumps to address found in memory [$rs\|$rt], link address is stored in $31
13.	jmsub	R-type	funct=34	jmsub $rs,$rt	jumps to address found in memory [$rs-$rt], link address is stored in $31
14.	bz	J-type	opcode=24	bz Target	if Status [Z] = 1, branches to pseudo-direct address (formed as j does)
15.	bn	J-type	opcode=25	bn Target	if Status [N] = 1, branches to pseudo-direct address (formed as j does)
16.	balz	J-type	opcode=26	balz Target	if Status [Z] = 1, branches to pseudo-direct address (formed as jal does), link address is stored in register 31
17.	baln	J-type	opcode=27	baln Target	if Status [N] = 1, branches to pseudo-direct address (formed as jal does), link address is stored in register 31
18.	jm	I-type	opcode=18	jm imm16($rs)	jumps to address found in memory (indirect jump)
19.	jalm	I-type	opcode=19	jalm $rt, imm16($rs)	jumps to address found in memory (indirect jump), link address is stored in $rt (which defaults to 31)
20.	jrs	I-type	opcode=18	jrs $rs	jumps to address found in memory where the memory address is written in register $rs.
21.	jrsal	I-type	opcode=19	jrsal $rs	jumps to address found in memory where the memory address is written in register $rs and link address is stored in memory (DataMemory[$rs]).
22.	sll	R-type	func=0	sll $rd, $rt, shamt	shift register $rt to left by shift amount (shamt) and store the result in register $rd.
23.	srl	R-type	func=2	srl $rd, $rt, shamt	shift register $rt to right by shift amount (shamt) and store the result in register $rd.
24.	sllv	R-type	func=4	sllv $rd, $rt, $rs	shift register $rt to left by the value in register $rs, and store the result in register $rd.
25.	srlv	R-type	func=6	srlv $rd, $rt, $rs	shift register $rt to right by the value in register $rs, and store the result in register $rd.

Status Register

Some of the conditional branches test the Z and N bits in the Status register. So the MIPS datapath will need to have a Status register, with the following 3 bits: Z (if the ALU result is zero) , N (if the ALU result is negative) or V (if the ALU result causes overflow). The Status register will be loaded with the ALU results each clock cycle.