Yes, all members make significant contributions to this homework.

B. Please explain how many types of instructions are supported in your processor, and explain the format of each type of instructions (e.g., which bits are used as the operation or function code, which bits are used to index the 1st, 2nd or 3rd operand, and which bits are used to store the immediate number). You can draw figures to better explain your answer.

3 types of instructions are supported in our processor – R-type, I-type, and J-type. Their formats, in general, are as shown:

Type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R-Type	Opcode (5 bits that will be read, 5th bit will always be 0 as opcode only needs 4 bits)					Rd			Rs			Rt			Opcode (Unused)	
I -Type	Opcode (5 bits that will be read)					Rd			Rs			Immediate				
J-Type	Opcode (5 bits that will be read)					Target Address										

Specific Instructions that require modifications of the format

I-Type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
li	Opcode (5 bits that will be read)					Rd			Immediate value							
load, store	Opcode (5 bits that will be read)					Rd			RT			Offset (Optional)				
move	Opcode (5 bits that will be read)					Rd			Rs			Unused				

J-Type	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
rtn, halt	Opcode (5 bits that will be read)					Unused										

For branch types, the specific instructions are given in the table below.

C. Please explain the format of each instruction (including the format of this instruction and its operation codes, and other information if needed).

li	Format of instruction: ● Bit 15 - bit 8: Immediate value ● Bit 7 - bit 5: Rd ● Bit 4 - bit 0: opcode Opcode: 00000 00
add	Format of instruction ● Bit 15- bit 14: Opcode ● Bit 13 - bit 11: Rt ● Bit 10- Bit 8: Rs ● Bit 7 - Bit 5: Rd ● Bit 4 - Bit 0: Opcode Opcode: 00010 00
and	Format of instruction ● Bit 15- bit 14: Opcode (Unused) ● Bit 13 - bit 11: Rt ● Bit 10- Bit 8: Rs ● Bit 7 - Bit 5: Rd ● Bit 4 - Bit 0: Opcode Opcode: 00100 00
or	Format of instruction ● Bit 15- bit 14: Opcode ● Bit 13 - bit 11: Rt ● Bit 10- Bit 8: Rs ● Bit 7 - Bit 5: Rd ● Bit 4 - Bit 0: Opcode Opcode: 00110 00
load	Format of instruction ● Bit 15 - bit 11: Offset ● Bit 10- Bit 8: R1 ● Bit 7 - Bit 5: Rd ● Bit 4 - Bit 0 : Opcode

	Opcode: 01000 00
store	Format of instruction ● Bit 15 - bit 14: Opcode (Unused bits) ● Bit 13 - bit 11: R1 ● Bit 10- Bit 8: Rd ● Bit 7 - Bit 5: Offset ● Bit 4 - Bit 0 : Opcode Opcode: 01010 00
move	Format of instruction ● Bit 15 - bit 11: Unused ● Bit 10- Bit 8: Rs ● Bit 7 - Bit 5: Rd ● Bit 4 - Bit 0 : Opcode Opcode: 01100 00
addi	Format of Instruction: ● Bit 15 - bit 8: Immediate value ● Bit 7 - bit 5: Rd ● Bit 4 - bit 0: opcode Opcode: 01110 00
andi	Format of instruction: ● Bit 15 - bit 11: Immediate value ● Bit 10 - bit 8: Rs ● Bit 7 - bit 5: Rd ● Bit 4 - bit 0: opcode Opcode: 10000 00
ori	Format of instruction: ● Bit 15 - bit 11: Immediate value ● Bit 10 - bit 8: Rs ● Bit 7 - bit 5: Rd ● Bit 4 - bit 0: opcode Opcode: 10010 00
ble	Format of instruction: ● Bit 14 - bit 15: Unused bits ● Bit 13 - bit 11: Rt ● Bit 10 - bit 8 : Rs ● Bit 7 - bit 5: Immediate value ● Bit 4 - bit 0: opcode Opcode: 10100 00
bne	Format of instruction: ● Bit 15 - bit 8: Immediate value

	● Bit 7 - bit 5: Rs ● Bit 4 - bit 0: opcode Opcode: 10110 00
jump	Format of instruction: ● Bit 15 - bit 5: Target address ● Bit 4 - bit 0: opcode Opcode: 11000 00
call	Format of instruction: ● Bit 15 - bit 5: Target address bit ● Bit 4 - bit 0: opcode Opcode: 11010 00
rtn	Format of instruction: ● Bit 15 - bit 5:Unused/reserved bit ● Bit 4 - bit 0: opcode Opcode: 11100 00
halt	Format of instruction: ● Bit 15 - bit 5: Unused/reserved bit ● Bit 4 - bit 0: opcode Opcode: 11110 00

Opcode bits are subject to change in each instruction format, but since in R-type, it was decided as 7 bits, then all formats of instructions' opcodes are in 7 bit format. For R-type, the 7 bits of opcode are separated, having 5 bits in the front and 2 extra (unused bits) at the back. Mainly, the first 5 bits of the binary code are opcodes being considered (Actually only using 4 bits where the least significant bit is the 4th of the 5 bits and the most significant bit is the first bit of the 5 bits).

D. Fill the following tables with the machine codes of each instruction of the testing programs:

Test program 1:

instruction	machine code (binary)	machine code (hex)
li $r1, 1	00000 000 00000001	0001
li $r2, 2	00000 001 00000010	0102
li $r3, 10	00000 010 00001010	020A
add $r2, $r1, $r2	00010 001 000 001 00	1104
ble $r2, $r3, -1	10100 111 001 010 00	A15C
halt	11110 000000000 00	F000

Test program 2:

instruction	machine code (binary)	machine code (hex)
li $r1, 6	00000 000 00000110	0006
li $r2, 5	00000 001 00000101	0105
andi $r3, $r1, 3	10000 010 000 00011	820C

ori $r4, $r3, 8	10010 011 010 01000	9348
halt	11110 000000000 00	F000

Test program 3:

instruction	machine code (binary)	machine code (hex)
li $r1, 6	00000 000 00000110	0006
li $r2, 5	00000 001 00000101	0105
and $r3, $r1, $r2	00100 010 000 001 00	2204
li $r8, 0	00000 111 00000000	0700
store $r3, $r8	01010 000 111 010 00	50E8
or $r4, $r1, $r2	00110 011 000 001 00	3304
li $r8, 1	00000 111 00000001	0701
store $r4, $r8	01010 000 111 011 00	50EC
li $r8, 1	00000 111 00000001	0701
load $r7, $r8	01000 110 111 000 00	46E0
halt	11110 0000000000	F000

Test program 4:

instruction	machine code (binary)	machine code (hex)
li $r1, 6	00000 000 00000110	0006
li $r2, 4	00000 001 00000100	0104
call 7	11010 00000000111	D007
move $r4, $r3	01100 011 010 00000	6340
li $r1, 7	00000 000 00000111	0007
li $r2, 8	00000 001 00001000	0108
call 3	11010 00000000011	D003
move $r5, $r3	01100 100 010 00000	6440
jump 3	11000 00000000011	C003
add $r3, $r1, $r2	00010 010 000 001 00	1204
rtn	11100 00000000000	E000
halt	11110 00000000000	F000