TASK 2

Add description for each image. Identify instruction type and exact 32-bit instruction code in the instruction type format.

RISC-V ISA:

The RISC-V ISA, which stands for Reduced Instruction Set Computing is an open standard instruction set architecture (ISA) based on established reduced instruction set computing principles. It's designed to be modular, scalable, and customizable, making it suitable for a wide range of applications, from embedded systems to supercomputers. RISC-V is gaining popularity due to its open nature, which fosters innovation and collaboration across various industries and academic institutions. It offers different standard extensions to accommodate diverse computing requirements, such as integer, floating-point, vector, cryptographic, and more.

INSTRUCTION FORMATS:

1.R-Type: This format is used for instructions that operate on registers only, without any memory access. R-Type instructions typically include operations like arithmetic (addition, subtraction, etc.), logical (AND, OR, XOR), comparison, and shifting.

func7	rs2	rs1	func3	rd	opcode
31:25	24:20	19:15	14:12	11:7	6:0

2. I-Type (Immediate): Instructions in this format involve immediate values (constants) in addition to register operands. I-Type instructions are used for operations where one operand is a register and the other is an immediate value stored within the instruction itself. These instructions have a 12bit constant (or immediate) as one of the two source operands defined within the 32bit instruction word. This constant is considered a 12-bit signed 2's complement integer, and it is always sign extended to provide a 32-bit operand.

imm[11:0]	rs1	func3	rd	opcode
31:20	19:15	14:12	11:7	6:0

3. S-Type (Store-Type): This format is used for memory store instructions. S-Type instructions encode an immediate offset, a source register containing the data to be stored, and a base register specifying the memory address.

imm[11:5]	rs2	rs1	func3	imm[4:0]	opcode
31:25	24:20	19:15	14:12	11:7	6:0

4. B-Type: Instructions in this format are used for branching operations, which involve conditional jumps based on certain conditions being met. B-Type instructions encode a branch offset, two source registers for comparison, and the branch condition.

imm[12|10:5]	rs2	rs1	func3	imm[4:1]|11	opcode
31:25	24:20	19:15	14:12	11:7	6:0

5. U-Type (Upper Immediate-Type) Format: These instructions are used for setting the upper bits of a register with an immediate value. They are primarily used for large immediate values that don't fit into the 12-bit immediate field of I-Type instructions.

imm[31:12]	rd	opcode
31:12	11:7	6:0

6. J-Type (Jump-Type) Format: J-Type instructions are used for unconditional jumps or jumps to an address encoded within the instruction itself. They are used for implementing function calls, returns, and other forms of control flow.

imm[20|10:1]	imm[11|19:12]	rd	opcode
31:20	19:12	11:7	6:0

INSTRUCTIONS:

1.add r6, r2, r1: This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00001	00010	000	00110	0110011

For R type instructions opcode=0110011, func7=0000000. For ADD func3=000.In this opcode will specify the operation to be performed. r1+r2 result will be stored in destination register r6 and r1, r2 are source registers.

r1=00001, r2=00010, r6=00110 (Decoding them in binary format)

2.sub r7, r1, r2: This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00010	00001	000	00111	0110011

For R type instructions opcode=0110011, func7=0000000. For SUB func3=000.In this opcode will specify the operation to be performed. r1- r2 result will be stored in destination register r7 and r1, r2 are source registers.

r1=00001, r2=00010, r7=00111 (Decoding them in binary format)

3.and r8, r1, r3: This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00011	00001	111	01000	0110011

For R type instructions opcode=0110011, func7=0000000. For AND func3=000.In this opcode will specify the operation to be performed. r1& r3 result will be stored in destination register r8 and r1, r3 are source registers.

r1=00001, r3=00011, r8=01000 (Decoding them in binary format).

4.or r9, r2, r5: This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00101	00010	110	01001	0110011

For R type instructions opcode=0110011, func7=0000000. For OR func3=110.In this opcode will specify the operation to be performed. r5 | r2 result will be stored in destination register r9 and r5, r2 are source registers.

r5=00101, r2=00010, r9=01001 (Decoding them in binary format)

5.xor r10, r1, r4: This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00100	00001	100	01010	0110011

For R type instructions opcode=0110011, func7=0000000. For XOR func3=100.In this opcode will specify the operation to be performed. r1 | r2 result will be stored in destination register r9 and r1, r4 are source registers.

r10=01010, r1=00001, r4=00100 (Decoding them in binary format)

6.slt r11, r2, r4: This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00100	00010	110	01011	0110011

For R type instructions opcode=0110011, func7=0000000. For SLT func3=010.In this opcode will specify the operation to be performed. r2 < r4 result will be stored in destination register r11 and r4, r2 are source registers.

r11=01011, r2=00010, r4=00100 (Decoding them in binary format)

7.addi r12, r5, 5: This is a I type instruction because operations are being performed on registers and immediate value is also used.

000000000101	00101	000	01100	0010011

For I type instructions opcode=0010011. For ADDI func3=000.In this opcode will specify the operation to be performed. r5+5 result will be stored in destination register r12 and r5 is source register.

r12=01100, r5=00101, 5=12’b000000000101(Decoding them in binary format)

8.sw r3, r1, 2: This is a S type instruction.

0000000	00011	00001	010	00010	0100011

For S type instructions opcode=0100011, For sw func3=010.In this opcode will specify the operation to be performed

r3=00011, r2=00010, 2=7’b0000000|5’b00010 (Decoding them in binary format).

9.lw r13, r1, 2: This is a I type instruction because operations are being performed on registers and immediate value is also used.

000000000010	00101	010	01101	0000011

For I type instructions opcode=0000011. For ADDI func3=010.In this opcode will specify the operation to be performed.

r13=01101, r1=00001, 2=12’b000000000010

10.beq r0, r1, 15: This is a B type instruction which involve conditional jumps based on certain conditions being met. B-Type instructions encode a branch offset, two source registers for comparison, and the branch condition.

0000000	00001	00000	000	01111	1100011

11.beq r0, r1, 20: This is a B type instruction which involve conditional jumps based on certain conditions being met. B-Type instructions encode a branch offset, two source registers for comparison, and the branch condition.

0000000	00001	00000	001	11000	1100011

12.sll r15, r1, r2(2): This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	00001	01000	001	01111	0110011

For R type instructions opcode=0110011, func7=0000000. For sll func3=001.In this opcode will specify the operation to be performed.

r1=00001, r2(2)=01000(left shift of binary 2),r15=01111

13.srl r16, r14, r2(2): This is a R type instruction because operations are being performed on registers. There is no usage of memory.

0000000	01110	00000	101	10000	0110011

For R type instructions opcode=0110011, func7=0000000. For srl func3=1011.In this opcode will specify the operation to be performed.

r14=01110, r2(2)=00000(right shift of binary 2),r16=10000.