1. li a5,56

The li (load immediate) pseudo-instruction in RISC-V is used to load an immediate value into a register.

For the instruction li a5, 56, the immediate value 56 is loaded into the register a5.

Binary Representation:

- Opcode: 0010011
- rd (a5/x15): 01111
- funct3: 000
- rs1 (x0): 00000
- Immediate (56): 0000000000111000

Combining these:

The binary representation becomes:

000000001110 00000 000 01111 0010011

2.andi a4,a4,255

Binary Representation:

- Opcode: 0010011
- rd (a4/x14): 01110
- funct3: 111 (for andi)
- rs1 (a4/x14): 01110
- Immediate (255): 0000000011111111

Combining these:

The binary representation becomes:

000000001111 01110 111 01110 0010011

3. addi a5,a5,-8

Full Binary Representation:

- Immediate (12-bit): 1111 1111 1000
- rs1 (a5/x15): 01111
- funct3: 000
- rd (a5/x15): 01111
- opcode: 0010011

The binary representation becomes:

1111 1111 1000 01111 000 01111 0010011

4. mv a0,a1

The mv pseudo-instruction is translated to an addi (add immediate) instruction with an immediate value of 0:

Binary Representation:

- Immediate (12-bit): 000000000000
- rs1 (a1/x11): 01011
- funct3: 000
- rd (a0/x10): 01010

- opcode: 0010011

Combining these:
000000000000 01011 000 01010 0010011

5. srli a3,a3,0x1

Binary Representation:

- Immediate (5-bit shamt): 00001
- funct7: 0000000
- rs1 (a3/x13): 01101
- funct3: 101
- rd (a3/x13): 01101
- opcode: 0010011

Combining these:
0000000 00001 01101 101 01101 0010011

6. or a0,a0,a3

Binary Representation:

- funct7: 0000000
- rs2 (a3/x13): 01101
- rs1 (a0/x10): 01010
- funct3: 110
- rd (a0/x10): 01010
- opcode: 0110011

Combining these:
0000000 01101 01010 110 01010 0110011

7. neg a0,a1

The neg instruction in RISC-V is a pseudo-instruction that performs a negation operation (i.e., it computes the two's complement of a register value).
The instruction neg a0, a1 negates the value in register a1 and stores the result in register a0.

- Binary Representation:
- funct7: 0100000
- rs2 (a1/x11): 01011
- rs1 (x0): 00000
- funct3: 000
- rd (a0/x10): 01010
- opcode: 0110011

Combining these:
0100000 01011 00000 000 01010 0110011

8. sub a4,a4,a5

Binary Representation:

- funct7: 0100000
- rs2 (a5/x15): 01111
- rs1 (a4/x14): 01110
- funct3: 000
- rd (a4/x14): 01110
- opcode: 0110011

Combining these:

0100000 01111 01110 000 01110 0110011

9. srl a5,a0,a5

Binary Representation:

- funct7: 0000000
- rs2 (a5/x15): 01111
- rs1 (a0/x10): 01010
- funct3: 101
- rd (a5/x15): 01111
- opcode: 0110011

Combining these:

0000000 01111 01010 101 01111 0110011

10. subw a0,a4,a0

Binary Representation:

- funct7: 0100000
- rs2 (a0/x10): 01010
- rs1 (a4/x14): 01110
- funct3: 000
- rd (a0/x10): 01010
- opcode: 0111011

Combining these:

0100000 01010 01110 000 01010 0111011

11. slli t0,t0,0x1f

Binary Representation:

- funct6: 000000
- Immediate (shamt): 11111
- rs1 (t0/x5): 00101
- funct3: 001
- rd (t0/x5): 00101
- opcode: 0010011

Combining these:
000000 11111 00101 001 00101 0010011

12. benz a3,21240<__udividi3+0x2c>

13. jr t0

The jr instruction in RISC-V is a pseudo-instruction for "jump register," which is used to jump to an address contained in a register.
The RISC-V equivalent of jr t0 is typically implemented using the jalr (jump and link register) instruction with the destination register set to x0 (the zero register) to discard the return address.
Binary Representation:

- Immediate (12-bit): 000000000000
- rs1 (t0/x5): 00101
- funct3: 000
- rd (x0): 00000
- opcode: 1100111

Combining these:
000000000000 00101 000 00000 1100111

14. jal ra,21214<__udivdi3>

15. bltu a2,a1,2124c <__udivdi3+0x2c>