well, knowing RV32E ISA opcodes and their instruction fields encoding, how you recommend the grouping of new ISA more suitable for software simulation?

For an efficient software simulation, grouping instructions based on the complexity of decoding and the similarity of their operations can help improve performance. Here is a suggested grouping for a new ISA more suitable for software simulation based on RV32E ISA:

1. Arithmetic and Logical Instructions (ALU):
   Add, Subtract, Shifts, Bitwise operations (AND, OR, XOR, NOT)
2. Load and Store Instructions (LS):
   Load Word (LW), Store Word (SW)
3. Branch Instructions (BR):
   Branch if Equal (BEQ), Branch if Not Equal (BNE), Branch if Less Than (BLT), Branch if Greater Than or Equal (BGE), Branch if Less Than Unsigned (BLTU), Branch if Greater Than or Equal Unsigned (BGEU)
4. Control Flow and Miscellaneous Instructions (CF):
   Jump and Link (JAL), Jump and Link Register (JALR), Environment Call (ECALL), Environment Break (EBREAK)

To map these groups into new ISA suitable for software simulation, you can use a scheme like this:

1. Opcode (first byte):
   Divide the opcode byte into two parts: group bits and sub-opcode bits.
   Use the most significant bits (e.g., 2 bits) to represent the four primary instruction groups (ALU, LS, BR, CF).
   Use the remaining bits (e.g., 6 bits) to represent the sub-opcodes within each group.
2. Register encoding (second byte):
   Use 4 bits each for the destination register (rd) and the first source register (rs1).

For compressed instructions, you can analyze the RVC (compressed) extension and adapt the scheme above. Since the compressed instructions use 16-bit encoding, you'll have fewer

bits available for opcode and register fields. You might need to limit the available operations within each group or use more creative encoding schemes to accommodate the compressed format.

After defining your new ISA grouping and encoding, you'll need to modify your software simulator accordingly. The simulator should be able to decode the new opcode and register fields and execute the corresponding operations efficiently.