well, I want to design ISA mappable to RV32EMC but with easiest software simulation implemntation, can you help me with it?

Sure, I can help you design an ISA that is mappable to RV32EMC but with an easier software simulation implementation. To achieve this, we will focus on simplifying the instruction set and reducing the complexity of decoding and executing instructions.

Here's an outline of the simplified ISA:

1. Only support a subset of RV32EMC instructions that are essential for most applications. This could include basic arithmetic, logical, and memory operations.
2. Use a fixed instruction length (e.g., 32 bits) to simplify decoding. This will avoid the need to support variable-length instructions like the compressed instructions in the RISC-V C extension.
3. Use a consistent format for all instructions, which will make decoding easier. For example, you can use the same positions for opcode, source, and destination registers across all instructions.
4. Remove support for complex addressing modes, which simplifies memory operations. Instead, only support simple addressing modes like base+offset.
5. Do not support pipelining or out-of-order execution, as these features add significant complexity to the simulator.

Based on these design principles, let's create a simplified ISA:

c

Copy code

```c
typedef enum { NOP, // No operation ADD, // Add SUB, // Subtract AND, // Bitwise
AND OR, // Bitwise OR XOR, // Bitwise XOR SHL, // Shift left SHR, // Shift right
LUI, // Load upper immediate AUIPC, // Add upper immediate to PC JAL, // Jump
and link JALR, // Jump and link register BEQ, // Branch if equal BNE, // Branch
if not equal LW, // Load word SW, // Store word } opcode_t; typedef struct {
opcode_t opcode; uint8_t rd; uint8_t rs1; uint8_t rs2; int32_t imm; }
instruction_t;
```

Now, you can create a simple function to decode and execute these simplified instructions:

c Copy code

```c
void decode_and_execute(instruction_t instr) { switch (instr.opcode) { case NOP:
break; case ADD: registers[instr.rd] = registers[instr.rs1] +
registers[instr.rs2]; break; // Implement the remaining instructions } }
```

In this simplified ISA, you can reduce the complexity of decoding and executing instructions. However, remember that this comes at the cost of reduced flexibility and expressiveness compared to the full RV32EMC ISA. You may need to extend or adapt the simplified ISA based on your specific application requirements.