ROM (INST_MEM.vhd)

Task:

There's no real need to modify this file structurally, but I'd like to expand the instruction set from **0–255 to 0–1023**. Ideally, we can generate the instructions randomly while applying constraints to ensure they are valid **RISC-V instructions**.

Note: Check the link below for a reference on how I generated valid RISC-V instructions using random values with constraints:

You can see this in my SUPERSCALAR_CPU repository (tb_decoder.vhd)

In VHDL: use IEEE.math_real.ALL; uniform randomize generator located here

https://github.com/NoridelHerron/SUPERSCALAR_CPU/blob/main/test_benches/tb_decoder.vhd

In System Verilog:

real rand_real; logic [6:0] op; logic [4:0] rd, rs1, rs2; logic [2:0] funct3; logic [6:0] funct7; logic [31:0] instr;

```
  // Generate uniform random number between 0 and 1
   rand_real = $urandom_range(0, 10000) / 10000.0;

   // Select opcode based on probability ranges
   if (rand_real < 0.10)
       op = LOAD;
   else if (rand_real < 0.20)
       op = S_TYPE;
   else if (rand_real < 0.30)
       op = JAL;
   else if (rand_real < 0.60)
       op = B_TYPE;
   else if (rand_real < 0.80)
       op = I_IMME;
   else
       op = R_TYPE;

   // Generate random registers and function fields
```

```
rd     = $urandom_range(1,31);
rs1    = $urandom_range(0,31);
rs2    = $urandom_range(0,31);
funct3 = $urandom_range(0,7);
funct7 = $urandom_range(0,127);

if (op == R_TYPE) begin
   if (funct3 == 3'd0 || funct3 == 3'd5) begin
      // restrict funct7 to either 0 or 32
      if ($urandom_range(0,1) == 0)
         funct7 = 7'd0;
      else
         funct7 = 7'd32;
      end
   end
else if (op == I_IMME) begin
   if (funct3 == 3'd5) begin
      // restrict funct7 to 0 or 32 as well
      if ($urandom_range(0,1) == 0)
         funct7 = 7'd0;
      else
         funct7 = 7'd32;
   end
end

// Build instruction: default R-type format
instr = {funct7, rs2, rs1, funct3, rd, op};
```

This example shows how to keep the instructions within valid opcode and field ranges while still introducing randomness.