A complete, **pure-simulation** mid-sized pipelined CPU in **Verilog** with a simple L1 cache (direct-mapped, write-through, configurable latency), a toy 32-bit ISA, and a testbench that runs a short program. I made tradeoffs to keep it realistic but understandable:

- 5-stage pipeline (IF → ID → EX → MEM → WB) with pipeline registers.
- Register file with two read ports + one write port.
- Simple forwarding (EX → ID/EX and MEM → EX) and a load-use stall.
- L1 cache: direct-mapped, write-through, configurable hit latency and miss latency (misses stall pipeline for simplicity).
- Instruction memory (IMEM) and main data memory (DMEM) are behavioral arrays; IMEM preloaded in the testbench.
- Toy ISA: R-type, I-type (ALU immediate), LD, ST, BEQ, JAL, and NOP.

Copy-paste the files into a single `.v` file (or split them) and run with Icarus Verilog / ModelSim / VCS.

Toy ISA (32-bit instruction)

All fields are big-endian bit positions (`[31:0]`):

- R-type: `opcode[31:26]=6'b000000`, `rd[25:21]`, `rs1[20:16]`, `rs2[15:11]`, `funct[10:0]`
  - funct defines add/sub/and/or/xor/shl, etc.
- I-type ALU immediate: `opcode = 6'b001000`, `rd[25:21]`, `rs1[20:16]`, `imm[15:0]` (sign-extended)
- LD (load word): `opcode = 6'b010000`, `rd[25:21]`, `rs1[20:16]`, `imm[15:0]`
- ST (store word): `opcode = 6'b010001`, `rs2[25:21]`, `rs1[20:16]`, `imm[15:0]` (store rs2 -> [rs1+imm])
- BEQ: `opcode = 6'b011000`, `rs1[25:21]`, `rs2[20:16]`, `imm[15:0]` (PC-relative words)
- JAL: `opcode = 6'b011001`, `rd[25:21]`, `imm[20:0]` (jump and link; imm sign-extended * 4)
- NOP: `32'h00000000`

Registers: 32 regs x32-bit. Register 0 is wired to zero.

ALU operations for R-type determined by `funct[3:0]`:

- `0000` = ADD
- `0001` = SUB

- `0010` = AND
- `0011` = OR
- `0100` = XOR
- `0101` = SLL
- `0110` = SRL
- `0111` = SRA

  (You can add more.)

Files / Modules

`regfile.v`

`alu.v`

`imem.v`

`dmem.v`

`l1cache.v`

`cpu.v` (top CPU)

`tb.v` (testbench).