Instruction Memory Module

Purpose:
This module acts as a **read-only instruction memory (ROM)**. It holds your RISC-V program—a set of instructions—encoded in a `.hex` file. The CPU fetches instructions from this memory during the **Instruction Fetch (IF)** stage, using the program counter (`pc`) value as the address.

ROM Structure

- `mem [0:DEPTH*4-1]` is an array of **1024 bytes** (since DEPTH=256, 256 words × 4 bytes/word).
- Each memory cell holds 8 bits (1 byte).

Initialization

- `initial $readmemh("program.hex", mem);`
  On simulation start, this loads the ROM content from a hex file (`program.hex`). Each line in the hex file represents a byte (or group of bytes) to store in memory.

  Why:
  This lets you change your test program **without changing Verilog code**—just edit your `.hex` file.

Fetching Instructions

- The CPU requests an instruction using a **byte address** (`addr`), usually equal to the current value of the **Program Counter**.
- Instructions are 32 bits (4 bytes), stored in consecutive bytes.
- This line:
  verilog

```verilog
assign instr = { mem[addr+3], mem[addr+2], mem[addr+1], mem[addr] };
```

- Concatenates 4 consecutive bytes from memory into a 32-bit word.
- **Little-endian order:**
    - `mem[addr]` is the **least significant byte**,

   - `mem[addr+3]` is the **most significant byte**.

How It Fits in the Pipeline

- **IF stage:**
  The PC outputs the current instruction address.
  This address is given to the `memory` module so that it outputs `instr`—the 32-bit instruction at that address.
- **Pipeline registers:**
  The fetched instruction (from `instr`) is passed forward to later pipeline stages via buffer registers.

Parameterization

- `parameter DEPTH = 256`
  Lets you easily change memory size in future (e.g., for longer programs).

Summary in Pipeline Context

- The **program counter (PC)** provides the address of the next instruction.
- The **`memory` module** uses this address to output the correct instruction word (in little-endian format).
- The instruction is then latched into the next pipeline stage for decode, execute, etc.
- The module is ROM—for simulation: contents do not change at runtime; only read during execution.