|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instr class** | **Branch** | **Jump** | **MemRead** | **Mem**  **Write** | **MemToReg** | **ALUSrc** | **RegWrite** | **FP\_En** | **ALU**  **Op** |
| R-type integer ALU | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 10 |
| I-type integer ALU immediate | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 10 |
| Load word | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 00 |
| Load byte | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 00 |
| Store word | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 00 |
| Store byte | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 00 |
| Conditional branch | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 |
| Unconditional jump / call / ret | 0 | 1 | 0 | 0 | (CALL: maybe 0) | 0 | (CALL: 1 to write link) | 0 | 10/- |
| Floating-point arithmetic | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 11 |
| FP convert | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 11 |
| I/O read | 0 | 0 | 0\* | 0 | 0 | 0 | 1 | 0 | 10\* |
| I/O write / system | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

**Control Unit**

**Branch**

**Jump**

**MemRead**

**MemWrite**

**MemToReg**

**ALUSrc**

**RegWrite**

**FP\_En**

**ALUOp**

* **Data hazards**: RAW (read after write) dependencies between closely scheduled instructions. Common examples: ALU-result used by next instruction, load → use, FP result used by integer or FP instructions.
* **Control hazards**: branches and jumps; penalty depends on where branch is resolved (ID or EX) and whether we have prediction.
* **Structural hazards**: mostly avoided because we have a separate FPU and integer ALU; still watch for shared ports (register file read/write port limits) and memory port contention.
* **Pipeline optimizations used**: short forwarding network (EX/MEM & MEM/WB → EX), small hazard detection to insert single-cycle stalls for load-use, early branch resolution or simple branch prediction, separate FPU pipeline + scoreboard.