| Step name | Action for R-type instructions | Action for memory reference instructions | Action for branches | Action for jumps |
|---|---|---|---|---|
| Instruction fetch | | IR <= Memory[PC]<br>PC <= PC + 4 | | |
| Instruction decode/register fetch | | A <= Reg [IR[25:21]]<br>B <= Reg [IR[20:16]]<br>ALUOut <= PC + (sign-extend (IR[15:0]) << 2) | | |
| Execution, address computation, branch/jump completion | ALUOut <= A op B | ALUOut <= A + sign-extend (IR[15:0]) | if (A == B)<br>PC <= ALUOut | PC <= {PC [31:28], (IR[25:0]],2'b00)} |
| Memory access or R-type completion | Reg [IR[15:11]] <= ALUOut | Load: MDR <= Memory[ALUOut]<br>or<br>Store: Memory [ALUOut] <= B | | |
| Memory read completion | | Load: Reg[IR[20:16]] <= MDR | | |

**FIGURE e4.5.6** Summary of the steps taken to execute any instruction class. Instructions take from three to five execution steps. The first two steps are independent of the instruction class. After these steps, an instruction takes from one to three more cycles to complete, depending on the instruction class. The empty entries for the Memory access step or the Memory read completion step indicate that the particular instruction class takes fewer cycles. In a multicycle implementation, a new instruction will be started as soon as the current instruction completes, so these cycles are not idle or wasted. As mentioned earlier, the register file actually reads every cycle, but as long as the IR does not change, the values read from the register file are identical. In particular, the value read into register B during the Instruction decode stage, for a branch or R-type instruction, is the same as the value stored into B during the Execution stage and then used in the Memory access stage for a store word instruction.