```verilog
// decoder: The Instruction Decoder module. It decodes the instructions and generates the control signals that are going to be used throughout the pipeline.

// Parameter List:
// clk: the clock (input)
// instruction: the instruction (input)
// wrA: the address for the write port (output)
// immediate: the sign-extended immediate (output)
// aluOp: the ALU control signals (the LS bit is used as control signal for the multiplexer too) (output)

// Author: Nestoras Tzartzanis
// Date: 1/25/96
// EE577b Verilog Example

// Include definition of the op-codes
#include "opcodes.h"

// Include definition of the control signals
#include "control.h"

module decoder (clk, instruction, wrA, immediate, aluOp);

input clk;
input [7:0] instruction;
output [2:0] wrA;
output [7:0] immediate;
output [1:0] aluOp;

// The outputs are defined as registers too
reg [2:0] wrA;
reg [7:0] immediate;
reg [1:0] aluOp;

// Definition of the decoder latency time
parameter decDelay = 70;

always @(posedge clk) begin
    #decDelay // A case-statement is used for the decoder implementation. For this simple instruction set, the statement aluOp = instruction[7:6] would have been enough, but the casez-statement is used to indicate the general way to implement the decoder
    casez (instruction[7:6])
        'MV: // The instruction is a move
            aluOp = 'aluMv;
        'ADD: // The instruction is an add
            aluOp = 'aluAdd;
        'XOR: // The instruction is a xor
            aluOp = 'aluXor;
        default:
            $strobe("unknown OPCODE %h ", instruction[7:6]);
    endcase

    // The register addresses are directly assigned from the instruction
    wrA = instruction[2:0];

    // The immediate is sign-extended. The immediate is in instruction[6:3] and instruction[6] is the sign bit.
    immediate = {{4{instruction[6]}}, {instruction[6:3]});

end //always
endmodule
```