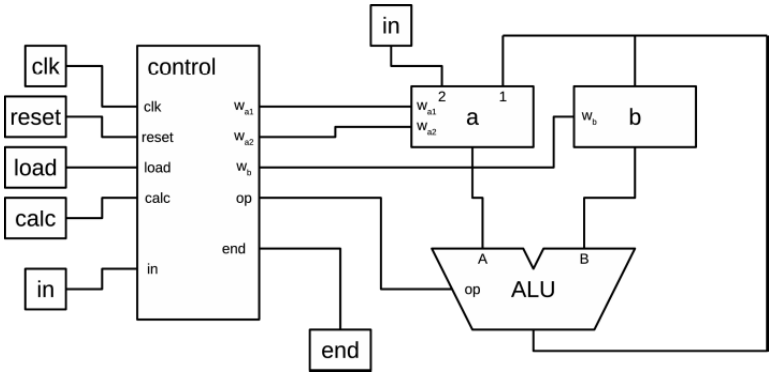
Apellidos: _____Nombre: _____

Ejercicio 1. Considere el sistema digital de la figura que describe una calculadora simple con las siguientes características:

- El registro “a” puede cargarse tanto desde la salida de la ALU (wa1) como desde la entrada externa “in” (wa2).
- La entrada “in” proporciona tanto el dato de entrada de “a” como el código de operación para la unidad de control.
- Cuando se activa la entrada “load”, “a” carga un dato desde “in” y el anterior dato de “a” es transferido a “b”.
- Cuando se activa la señal “calc”, el sistema realiza la operación indicada por los cuatro bits menos significativos de “in” según se indica en la tabla de operación.

load	calc	in	Operación
1	0	- - - -	a ← in; b ← a
0	1	0 0 0 1	a ← a + b
0	1	0 0 1 0	a ← a - b
0	1	0 1 0 0	a ← b
0	1	1 0 0 0	b ← a

- La entrada “reset” produce una puesta a cero asíncrona.
- El sistema parte de un estado inicial READY que espera que se activen “load” o “calc”. Al finalizar cualquier operación, el sistema permanece en un estado END hasta que se desactivan ambas señales “load” y “calc”.



ALU

op	salida
0 0	A + B
0 1	A
1 0	A - B
1 1	B

Describa la unidad de control del sistema digital propuesto empleando alguna de las siguientes técnicas:

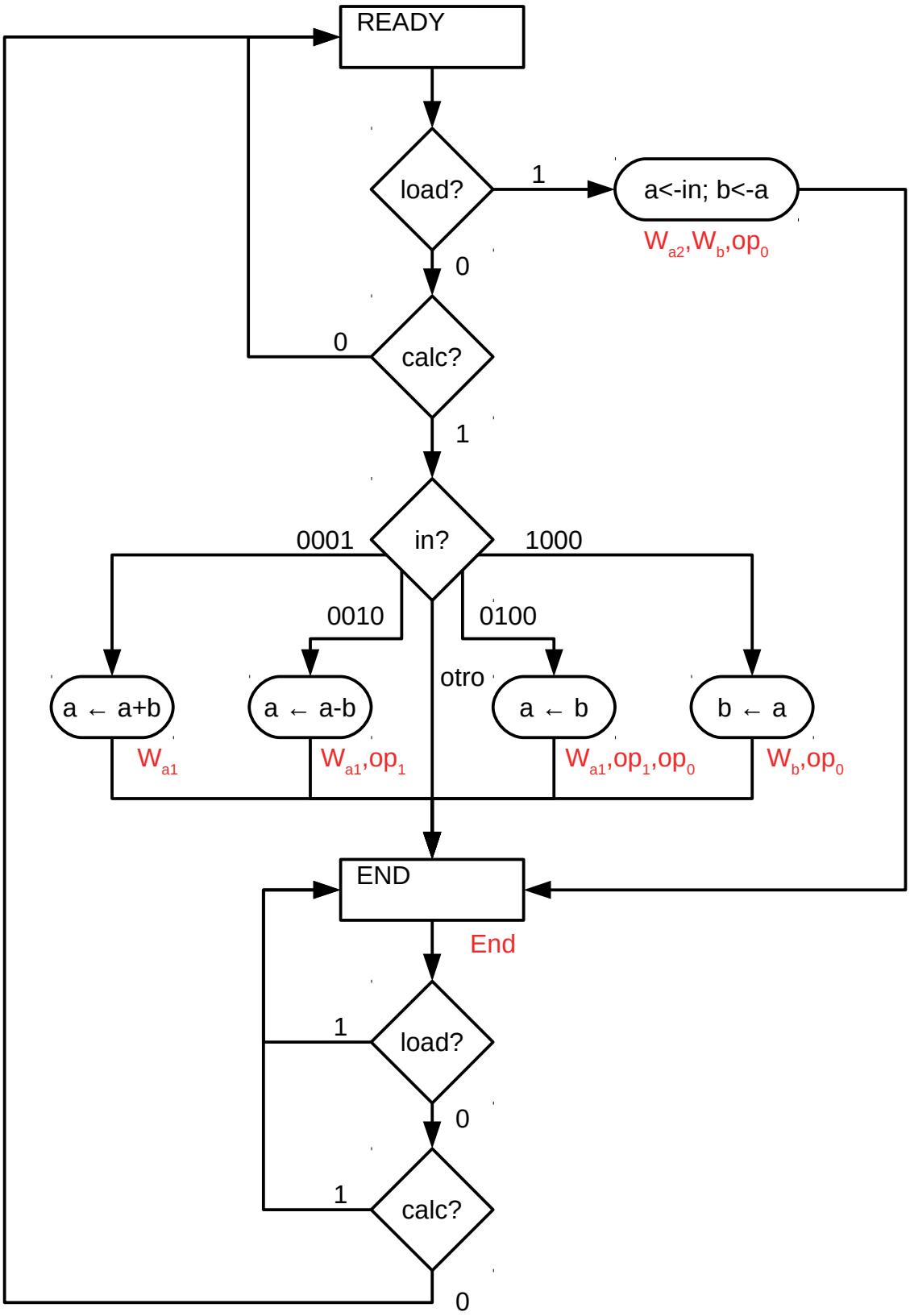
a) Carta ASM de transferencia de registros y de control.

b) Lenguaje Verilog empleando la plantilla suministrada (los espacios a completar están señalados con “<<<<”.

NOTA: NO es necesario hacer los dos tipos de descripciones. Es posible obtener la máxima calificación realizando sólo una de ellas.

Estructura de Computadores. TI. Grupo 1.

Prueba tema 2. 2017/18



```
////////////////////////////////////////////////////////////////////////
// Estructura de Computadores - TI. Grupo 1. Prueba Tema 2. 2017/18 //
////////////////////////////////////////////////////////////////////////

module control #(
    parameter W = 8
    )(
    input wire clk, reset, calc, load,
    input wire [W-1:0] in,
    output reg End, wa1, wa2, wb,
    output reg [1:0] op);

    reg [2:0] state, next_state;   // estado y próximo estado

    parameter READY = 0,          // codificación de estados

              END = 7;

    always @(posedge clk, posedge reset)   // procedimiento de cambio de estado
        if (reset)
            state <= READY;
        else
            state <= next_state;

    always @(*) begin   // procedimiento de próximo estado y salidas
        // valor por defecto de las señales
        End = 1'b0; wa1 = 1'b0; wa2 = 1'b0; wb = 1'b0; op = 2'b00;

        case(state)
        READY:
            if(load == 1'b1) begin  // operación de carga de dato
                wa2=1'b1; wb=1'b1; op=2'b01;
                next_state = END;
            end
            else if(calc == 1'b0)   // no se activa ninguna operación
                next_state = READY;
            else begin              // operaciones de cálculo
                case(in[3:0])
                4'b0001:            // a <- a+b
                    wa1 = 1'b1;
                4'b0010: begin      // a <- a-b
                    wa1 = 1'b1; op = 2'b10;
                end
                4'b0100: begin      // a <- b
                    wa1 = 1'b1; op = 2'b11;
                end
                4'b1000: begin      // b <- a
                    wb = 1'b1; op = 2'b01;
                end
                endcase
                // todas las operaciones de cáculo acaban en END
                next_state = END;
            end
        END: begin
            End = 1'b1;             // señal que indica hemos terminado
            if(load || calc)
                next_state = END;   // esperamos a desactivar entradas
            else
                next_state = READY;
        end
        // no debemos llegar aquí
        default:
            next_state = 'bx;
        endcase
    end
endmodule
```