¿Cómo puedo modificar el código para añadir una nueva instrucción?

La clave es entender bien el Mips que hemos estudiado en clase y ser ordenado. Vamos a verlo con un ejemplo paso a paso. Queremos añadir una nueva instrucción:

- lw_di rt, inm.

Esta instrucción es como un lw normal, pero la dirección es el campo inmediato de la propia instrucción. Es decir, si en un lw normal la dirección es `"rs+inm", aquí es "inm" y el campo rs no se utiliza.

Hay muchas formas de incluir esta instrucción. Vamos a implementar una sencilla que consiste en añadir un multiplexor en la entrada A de la ALU. En la entrada 0 de este mux tendremos rs (que en la etapa EX se llama busA_EX), y en la 1 tendremos un 0. A la señal de control del mux la llamaremos ALUsrc_A y a la salida Mux_out_A. El funcionamiento es sencillo. En la etapa de ejecución la nueva instrucción elegirá la entrada 1 de este nuevo mux, y el resto de instrucciones elegirán el 0. Para cualquier otra cosa, la nueva instrucción hará lo mismo que un lw normal.

Vamos a ver qué cambios tenemos que hacer en la ruta de datos y la unidad de control:

- En primer lugar tenemos que definir el formato de instrucciones. Lo más fácil es que sea todo igual que el lw, pero con un código de operación distinto. Elegimos el 000101 que es el primero que está sin usar.
- A continuación vamos a actualizar la unidad de control en UC.vhd. Como hemos añadido un multiplexor, la unidad de control deberá incluir la nueva señal: ALUsrc_A, y además deberá incluir la nueva instrucción. Los cambios serían:

  1. añadir ALUSrc_A : out STD_LOGIC; en la definición de la entidad UC.

  2. añadir ALUSrc_A en cada una de las opciones del case asignándole el valor ‘0’.

  3. añadir una nueva línea al case asignando los valores correctos:

     WHEN "000101" => Branch <= '0'; ALUSrc_A <= '1'; ...

Ahora veamos los cambios en el archivo MIPs_segmentado.

1. hay que actualizar la definición del componente UC incluyendo la nueva señal. Quedaría así:

   a. ALUSrc_A : out STD_LOGIC;

2. hay que definir todas las señales nuevas que vamos a usar:

   a. ALUSrc_A_ID: la señal ALUSrc_A generada en la etapa ID (es deci r la salida de la unidad de control). Como es una señal de un bit será un std_logic.

   b. ALUSrc_A_EX: la señal ALUSrc_A en el ciclo siguiente, que es la que se va a utilizar.

c. Mux_out_A: salida del nuevo multiplexor que hemos colocado en la etapa EX en la entrada A de la ALU. Como es una señal de 32 bits será un std_logic_vector (31 downto 0)

3. Actualizamos la instanciación del componente UC incluyendo la nueva salida ALUSrc_A_ID:

a) UC_seg: UC port map (IR_op_code => IR_ID(31 downto 26), Branch => Branch, RegDst => RegDst_ID, **ALUSrc_A => ALUSrc_A_ID**, ALUSrc => ALUSrc_ID, MemWrite => MemWrite_ID, MemRead => MemRead_ID, MemtoReg => MemtoReg_ID, RegWrite => RegWrite_ID);

4. Pasar la señal **ALUSrc_A_ID** de la etapa ID a la etapa EX a través del banco de registros Banco_EX. Es un módulo con muchas entradas. Habría que añadir como entrada **ALUSrc_A_ID** y como salida **ALUSrc_A_EX.** Y dentro del código tratar estas dos señales como se hace con todas las demás: la salida se actualiza en los flancos ascendentes de reloj. Se pone a cero si reset vale '1' y si no se hace la asignación **ALUSrc_A_EX <= ALUSrc_A_ID;**

5. incluir el nuevo multiplexor:

a) muxALU_src_A: mux2_1 port map (Din0 => busA_EX, DIn1 => "00000000000000000000000000000000", ctrl => ALUSrc_A_EX, Dout => Mux_out_A);

6. cambiar la entrada de la ALU que antes recibía busA_EX y ahora recibe Mux_out_A:

a) ALU_MIPs: ALU PORT MAP (DA => Mux_out_A, DB => Mux_out, ALUctrl => ALUctrl_EX, Dout => ALU_out_EX);

Ya sólo queda poner una de estas nuevas instrucciones en la memoria de instrucciones y comprobar que funciona. Es decir que accede a la dirección que indica el inmediato y guarda el valor en el registro correspondiente.