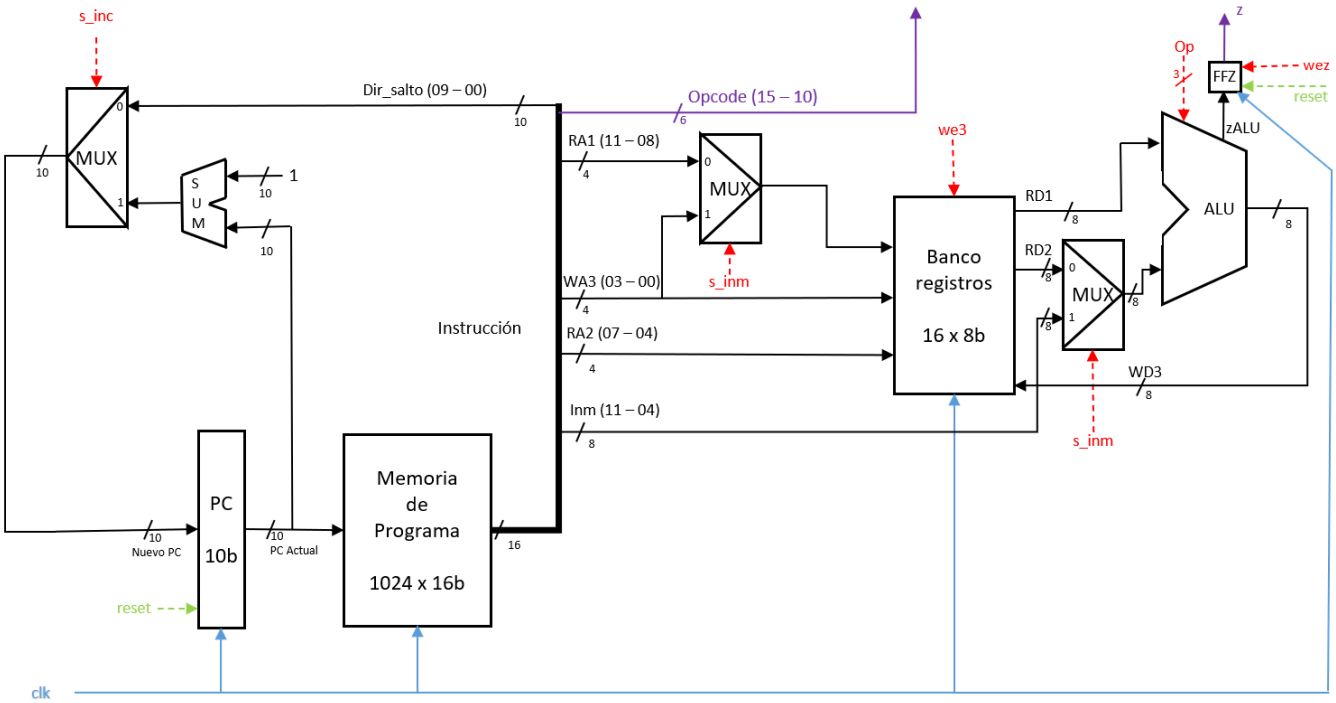
PRÁCTICA 2: SIMULACIÓN DE LA UNIDAD DE CONTROL DE UNA CPU SIMPLE

DESCRIPCIÓN

El objetivo de la práctica es lograr una mejor comprensión sobre cómo funcionan el procesador y su unidad de control. Para ello, nos centraremos en un procesador muy simple de un sólo ciclo. Para que un procesador pueda ejecutar instrucciones en un solo ciclo sin recurrir al paralelismo en su implementación debemos separar las memorias de instrucciones y de datos de forma que se pueda realizar el acceso a ambas dentro del mismo ciclo (al estilo de la arquitectura Harvard). En este ejemplo el procesador no va a tener una memoria de datos propiamente dicha, sino que operará con su banco de registros como memoria de datos. Esta estructura es típica de algunos microcontroladores, procesadores muy sencillos con una memoria de programa no volátil, diseñados para funcionar integrados en otro artefacto como una lavadora o un coche, por ejemplo.

Para analizar el funcionamiento del procesador, estudiaremos separadamente el camino de datos de la unidad de control que lo gobierna y los modelaremos por separado también. En la figura se han marcado en rojo y con línea discontinua las señales que provendrían de la Unidad de Control.



La figura representa el camino de datos del procesador. La parte superior izquierda muestra la lógica (sumador SUM y un multiplexor) que se emplea para actualizar el contador de programa (PC). El PC, de 10 bits, sirve de dirección a la memoria de programa. El dato obtenido de esta memoria es la instrucción, de 16 bits. Esos 16 bits codifican varios tipos de instrucciones diferentes:

Instrucción de operación aritmética o lógica: *Opcode* de 4 bits (15-12), campo de primer registro operando de 4 bits (RA1,11-8) cuyo contenido será el operando RD1 hacia la ALU si la señal s_inm está a 0, campo de segundo registro operando de 4 bits (RA2, 7-4) cuyo contenido en RD2 será el segundo operando en la ALU si la señal s_inm está a 0 y campo de registro de destino de 4 bits (WA3, 3-0) donde se almacenará el resultado. El nuevo PC será el PC previo incrementado en 1, para lo que será necesario que la señal s_inc valga 1. Estas instrucciones deben afectar al flag de zero y escribir en el banco de registros con las señales de control correspondientes.

Instrucción de operación aritmética o lógica con constante inmediata: *Opcode* de 4 bits (15-12), operando constante inmediata de 8 bits (11-4) y campo de registro de 4 bits que indica qué registro va a ser a la vez primer operando y destino de la operación (Reg ● Cte_Inm → Reg). Para ello, s_inm debe estar a 1 con lo que se logra que el contenido del registro sea el primer operando y destino, y también que el segundo operando sea la constante contenida en la instrucción. Como en el caso anterior, debemos incrementar el PC y obviamente permitir escrituras en el banco de registros y en el flag de cero. Las operaciones permitidas son sólo cuatro:

- LI: que carga la constante en el registro destino.
- ADI: suma el registro especificado con la constante y el resultado queda en el registro.

- SBI: resta el registro especificado y la constante y el resultado queda en el registro.
- NAI: realiza la operación NAND entre el registro y constante y resultado queda en el registro.

Instrucciones de saltos (absolutos): incondicional y condicionales al flag de cero activado o desactivado (J, JZ y JNZ respectivamente): Opcode de 6 bits (15-10) y los 10 bits restantes (9-0) serán el destino del salto (nuevo PC). Es necesario controlar la señal s_inc para hacer llegar el destino del salto a la entrada del PC. En el caso del salto incondicional se hace siempre y en los condicionales se evalúa la condición (zero o no zero) y desactiva s_inc para dar el salto o se activa para incrementar el PC si la condición que sea ha sido falsa.

Codificación	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Saltos (J, JZ, JNZ)	Op	Op	Op	Op	Op	Op	D	D	D	D	D	D	D	D	D	D
Oper. ALU (ADD, SUB…)	Op	Op	Op	Op	R1	R1	R1	R1	R2	R2	R2	R2	Rd	Rd	Rd	Rd
Oper. Inm.(LI,ADI,SBI,NAI)	Op	Op	Op	Op	C	C	C	C	C	C	C	C	Rod	Rod	Rod	Rod

La unidad de control para este camino de datos tendría como entradas:

- las señales comunes de reloj y *reset*
- los 6 bits más significativos de la instrucción (*Opcode* mayor posible) proveniente del camino de datos.
- el *flag* de cero para la ejecución de los saltos condicionales, también proveniente del camino de datos.

y generaría las salidas:

- señales de control de los multiplexores (s_inc que controla el multiplexor que afecta al PC y s_inm que controla simultáneamente los dos restantes)
- la habilitación de escritura del banco de registros (we) y del flag de cero (wez)
- señales de selección de operación de la ALU (ALUOp)

La misión de la unidad de control será activar correctamente las señales de salida a lo largo del ciclo que dura la instrucción de forma que se ejecute correctamente la instrucción determinada por los 6 bits (o menos) de *Opcode*.

OBJETIVO: SIMULACIÓN DE LA UNIDAD DE CONTROL A MODO DE TESTBENCH

a) Estudiar y familiarizarse con el funcionamiento de los módulos suministrados: ALU, Banco de registros, multiplexores, registro PC, memoria de programa. En particular, poner atención al fichero progfile.dat que sirve para inicializar la memoria de programa y el fichero regfile.dat que pone valores iniciales a los registros del banco de registros.

b) Realizar un módulo que represente el camino de datos, con la siguiente definición

```
module microc(output wire [5:0] Opcode, output wire zero, input wire clk, reset, s_inc, s_inm, we,
wez, input wire [2:0] ALUOp);
```

c) Realizar modificaciones al fichero progfile.dat y crear un fichero *testbench* microc_tb.v, que contenga código que permita simular la ejecución de las instrucciones que el profesor indicará (de entre las anteriores). La idea es escribir un programa simple en binario en el fichero progfile.dat formado por las instrucciones pedidas. De acuerdo a dicho programa, en el *testbench* pedido generaremos las señales de control como si provinieran de una unidad de control (no es necesario crear un módulo específicamente para la unidad de control). Para hacer realista la simulación, supondremos que en la primera mitad del ciclo de reloj la unidad de control estaría ocupada decodificando la instrucción y que las señales de control se emitirían a partir de la mitad del ciclo hasta su fin en que recomienza el ciclo de la siguiente instrucción. Todo ello se conseguirá mediante la introducción de los retardos adecuados. Visualizar su correcto funcionamiento con el *Gtkwave*.

ENTREGA

Al final de la sesión, después de haber mostrado los resultados a los profesores para su evaluación, se deberá entregar en la tarea del Campus Virtual creada para ello los ficheros microc.v, y microc_tb.v, así como cualquier fichero necesario para su funcionamiento y comprobación. Indicar en cada uno de los ficheros Verilog los nombres de los autores (dos personas).