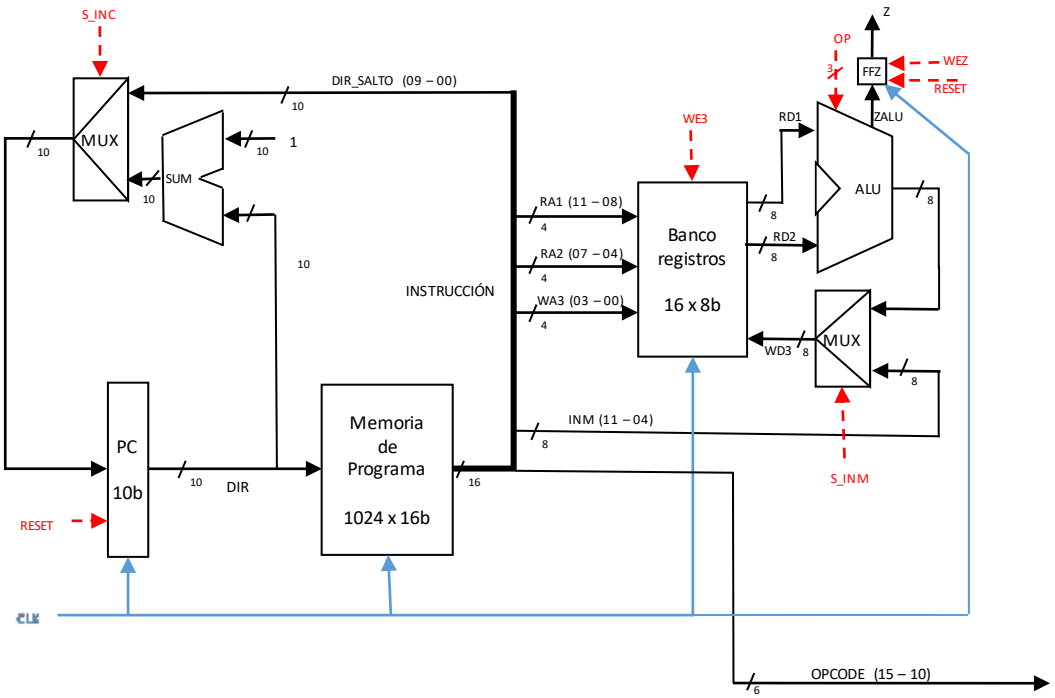
PRÁCTICA 2: SIMULACIÓN DE LA UNIDAD DE CONTROL DE UNA CPU SIMPLE

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. Se aprecia el registro PC de 10 bits que 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 salto: Opcode de 6 bits (15-10) y los 10 bits restantes (9-0) serán el nuevo PC si el multiplexor que controla la entrada al PC tiene su entrada de selección s_inc a cero. En las instrucciones que no sean saltos, s_inc se pondrá a 1 provocando que el nuevo PC sea el PC previo incrementado en 1.

Instrucción de salto condicional si Z: Opcode de 6 bits (15-10) y los 10 bits restantes (9-0) serán el nuevo PC si el flag de cero vale 1 (Z=1). En el caso contrario (Z=0), el nuevo PC será el PC previo incrementado en 1.

Instrucción de salto condicional si no Z: Opcode de 6 bits (15-10) y los 10 bits restantes (9-0) serán el nuevo PC si el flag de cero vale 0 (Z=0). En el caso contrario (Z=1), el nuevo PC será el PC previo incrementado en 1.

Instrucción de carga de una constante inmediata: Opcode de 4 bits (15-12), constante inmediata de 8 bits (11-4) y campo de registro de destino de 4 bits (3-0) indicando el registro destino donde se escribirá la constante (WA3), siempre que el multiplexor que provee el dato a escribir tenga la entrada s_inm a 1.

Instrucción de operación aritmética o lógica: Opcode de 4 bits (15-12), campo de primer registro operando de 4 bits (11-8, ra1), campo de segundo registro operando de 4 bits (7-4, ra2) y campo de registro de destino de 4 bits (3-0) donde se almacenará el resultado (siempre que el multiplexor tenga s_inm a cero). Estas instrucciones son las únicas que deben afectar al flag de cero Z.

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
Carga Inm. (LI)	Op	Op	Op	Op	C	C	C	C	C	C	C	C	Rd	Rd	Rd	Rd
Oper. ALU (ADD, SUB…)	Op	Op	Op	Op	R1	R1	R1	R1	R2	R2	R2	R2	Rd	Rd	Rd	Rd

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

- las señales comunes de reloj y reset
- los 6 bits menos significativos de la instrucción (Opcode mayor posible)
- el valor del flag de cero para la ejecución de los saltos condicionales

y generaría las salidas:

- señales de control de ambos multiplexores (s_inc y s_inm)
- la habilitación de escritura del banco de registros (we3)
- señales de selección de operación de la ALU (op)
- la habilitación de escritura del flag de cero (wez)

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(input wire clk, reset, s_inc, s_inm, we3, wez input wire [2:0] op, output wire z, output wire [5:0] opcode);
```

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 (máximo dos personas).