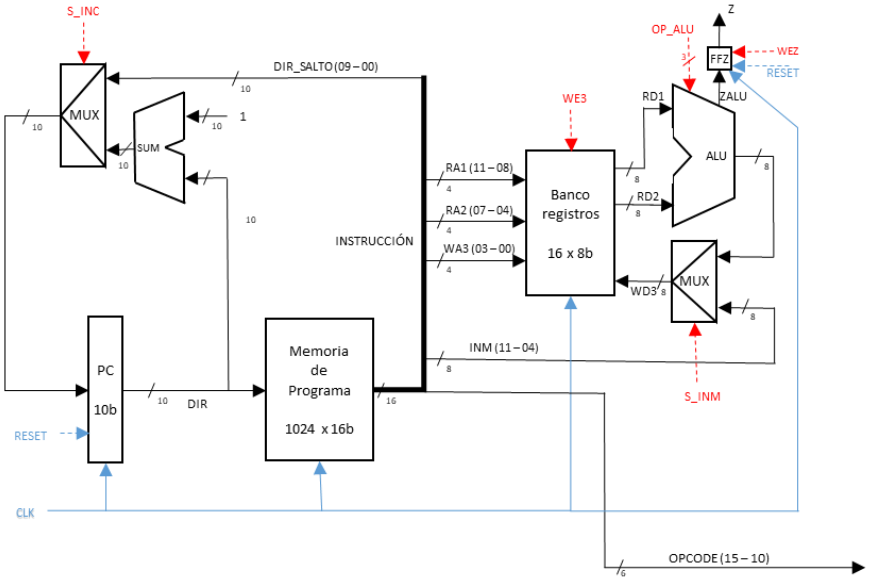
DISEÑO DE LA UNIDAD DE CONTROL DE UNA CPU SIMPLE DE UN SOLO CICLO

El objetivo de este proyecto es diseñar la unidad de control de un procesador lo más sencillo posible. Para ello, nos centraremos en un procesador 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 inicialmente 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, realizando el control del mismo.

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. La siguiente figura representa el camino de datos del procesador. Se han marcado en rojo y con línea discontinua las señales que provendrían de la Unidad de Control



Se aprecia el registro PC de 10 bits que sirve de dirección a la memoria de programa. El dato obtenido de ésta 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_alu)
- 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.
La unidad de control es básicamente un circuito combinacional en este caso, y debe tener la siguiente definición de módulo:

```
module uc(input wire [5:0] opcode, input wire z, output reg s_inc, s_inm, we3, wez,
          output-reg-[2:0] op_alu);
```

Es decir, a partir de los 6 bits menos significativos de la instrucción (opcode mayor posible) y el flag de cero para posibles saltos condicionales, generará las señales de control de ambos multiplexores, la habilitación de escritura del banco de registros y del biestable de cero y las señales de selección de operación de la ALU.

OBJETIVO: DISEÑO DE LA UNIDAD DE CONTROL

a) Estudiar y familiarizarse con el funcionamiento de los módulos de partida: ALU, Banco de registros, multiplexores, registro PC

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

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

c) Implementar la unidad de control que permita ejecutar las siguientes instrucciones

- Salto absoluto incondicional
- Salto condicional (a las condiciones cero y no cero)
- Carga de un valor inmediato en un registro
- Instrucción(es) Aritmética(s)/Lógica(s)

Integrar la unidad de control y el camino de datos en un módulo cpu.v. Visualizar su correcto funcionamiento con el gtkwave. Realizar un programa simple demostrando el uso de las instrucciones implementadas (ej: con un bucle similar al bucle for)