Proyecto Final CI-0120

Estudiantes:

- Marco Ferraro **B82957**
- Gabriel Revillat **B86524**

Contenido

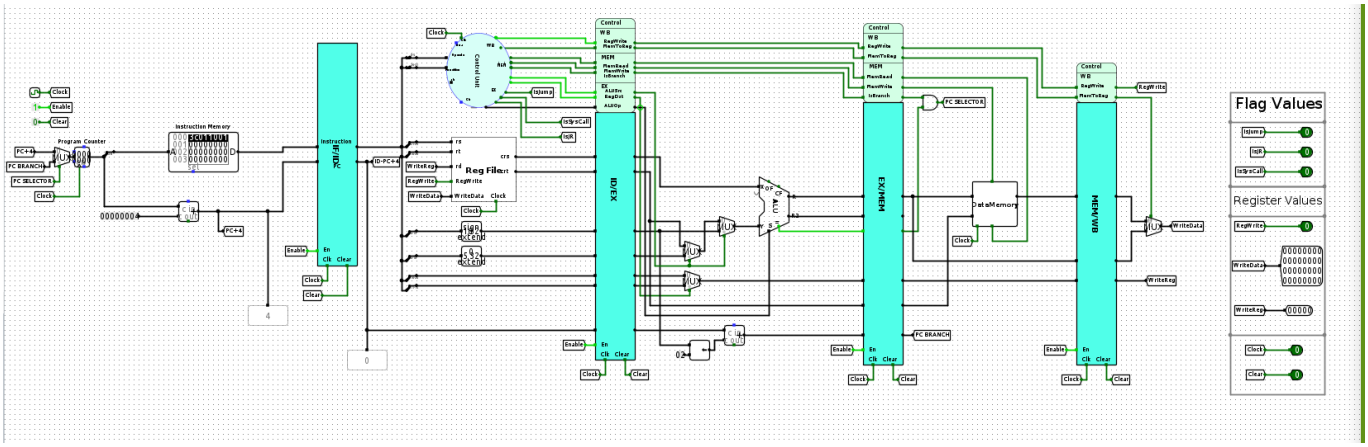
[TOC]

Circuito General

El objetivo de este proyecto es contruir un circuito en **logisim** que simule el funcionamiento de un procesador de arquitectura RISC, además de entender el funcionamiento de *pipelining*.

Logisim es una aplicación desarrollada en Java que nos permite simular el funcionamiento de circuitos lógicos.

Para seguir un buen flujo de trabajo se trabajo con circuitos ya definidos por **logisim**, además de crear un subcircuito para cada componente principal de un CPU RISC.



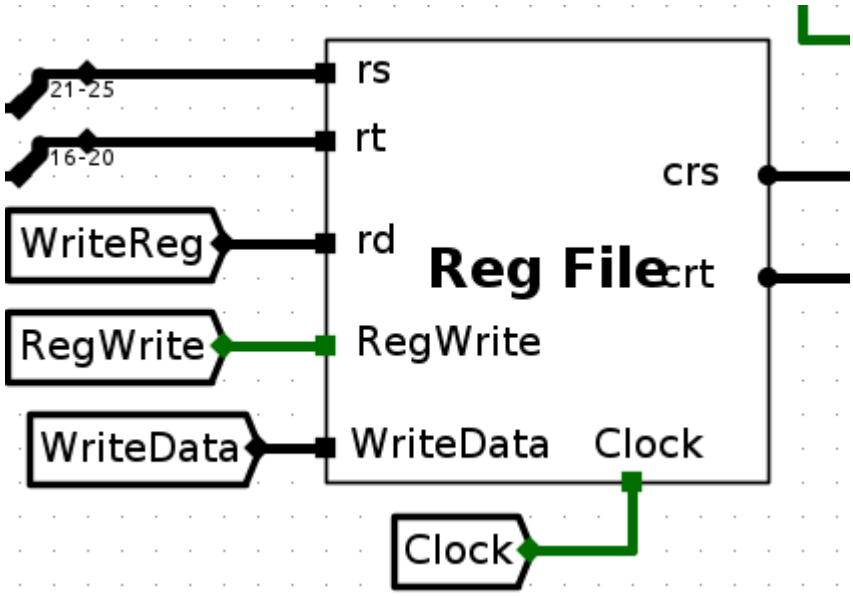
Vista General del Circuito

El circuito maneja un set de instrucciones de **32 bits**, definido en el enunciado del proyecto.

A continuación se mencionarán los principales componentes de del circuito, incluyendo los *stages* de *pipeline* y sus entradas y salidas respectivas.

Register File

El *Register File* es un componente que maneja una matriz de registros dentro del procesador. Además de esto, se puede escribir sobre la memoria de un registro.

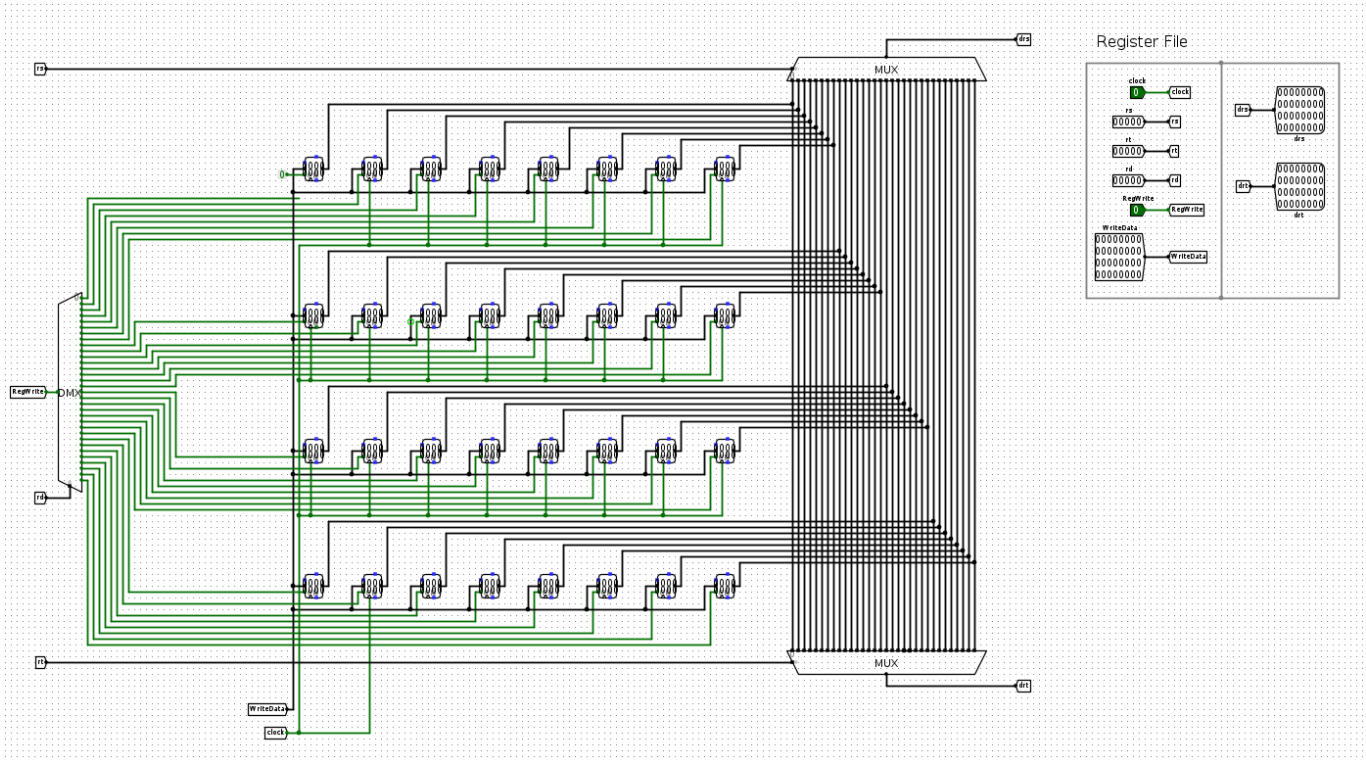


Subcircuito de Register FIle

Cada registro posee **32 bits** de almacenamiento, por lo tanto al *Register File* le ingresan direcciones de 4 bits.

Los registros para visualizar se llaman **rs** y **rt**. Al registro que se le puede modificar la memoria se conoce como **rd**.

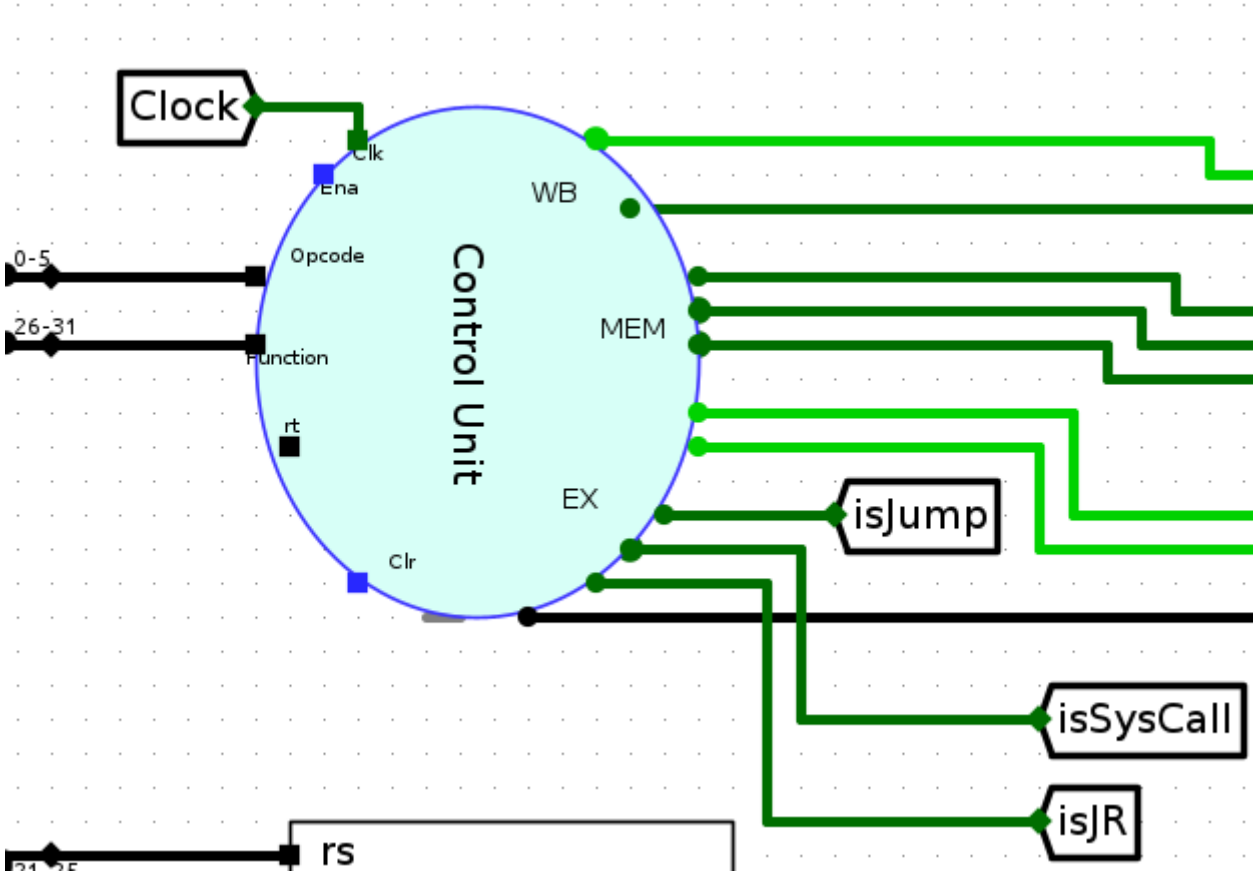
Sus salidas son los valores de los registros **rs** y **rt**



Lógica Interna del Register File

Control Unit

La Unidad de Control en un componente que nos ayuda con la ejecución de las instrucciones. Si manejamos un flujo **Multi Cycle**, la unidad de control deberá manejar varios **Datapaths** y **flags**.



Subcircuito Unidad de Control

Para determinar los valores que cada stage debería de recibir, nos basamos en el formato **R** de intrucciones para procesadores RISC.

	EX				MEM			WB	
Instr.	**RegDst**	**ALUOp 1**	**ALUOp2**	**ALUSrc**	**Branch**	**MemRead**	**MemWrite**	**RegWrite**	**MemToReg**
R	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Splitter de Instrucciones

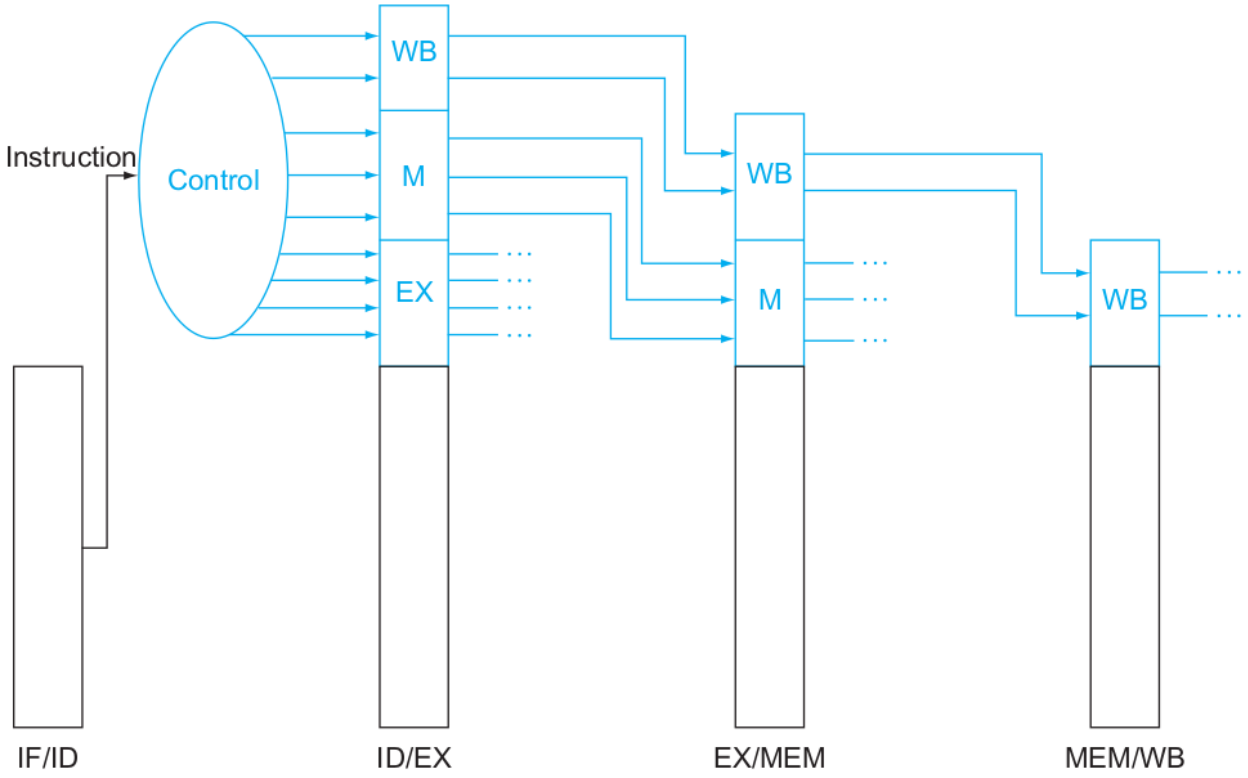
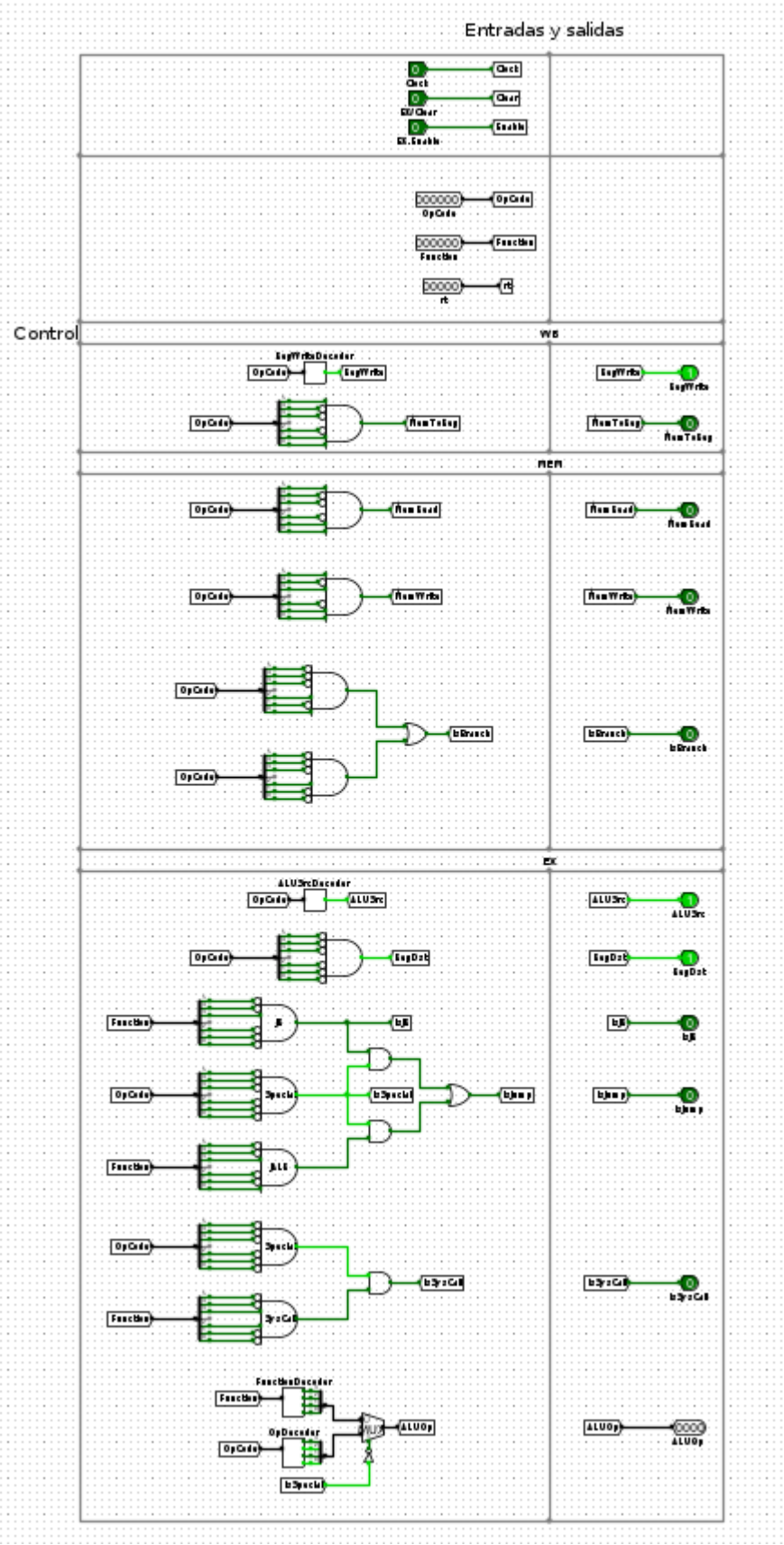


FIGURE 4.50 The control lines for the final three stages. Note that four of the nine control lines are used in the EX phase, with the remaining five control lines passed on to the EX/MEM pipeline register extended to hold the control lines; three are used during the MEM stage, and the last two are passed to MEM/WB for use in the WB stage.

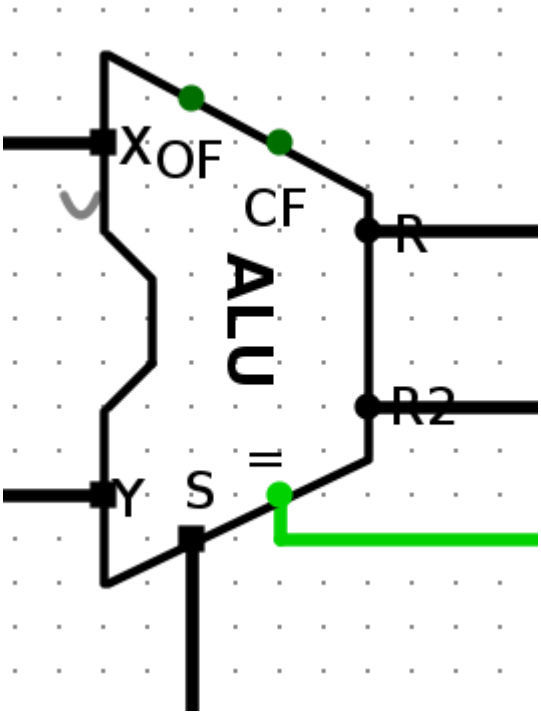
Flujo de los valores de salida para cada Stage



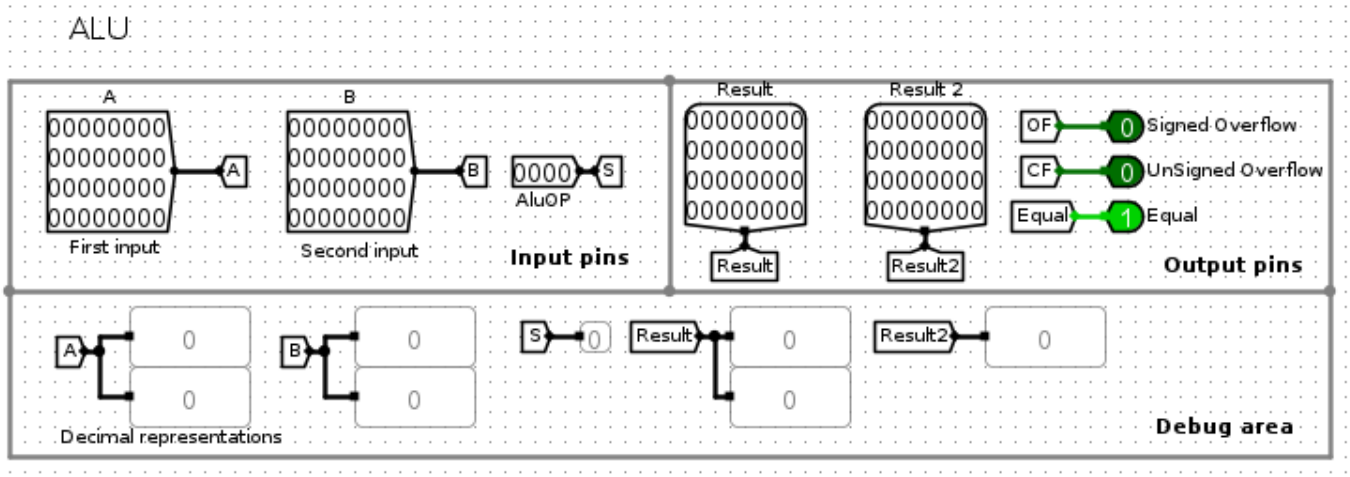
Circuito interno de la Unidad de Control

ALU

La unidad de control es la encargada de expresar operaciones aritméticas y lógicas sobre nuestro CPU. Nuestra UAL o ALU recibe 2 instrucciones de **32 bits**.



Subcircuito ALU



Salidas ALU

Nuestro ALU es de instrucciones reducidas. Notese que el espacio de **AluOP** tiene 4 bits. Las instrucciones básicas que hace es

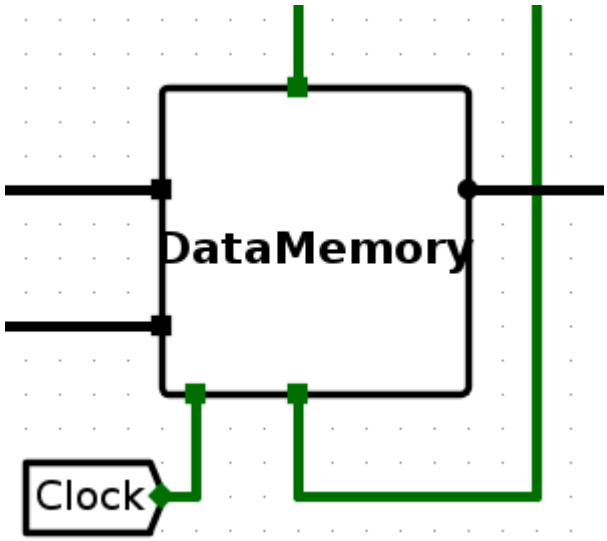
- Suma
- Resta

en la parte aritmética.

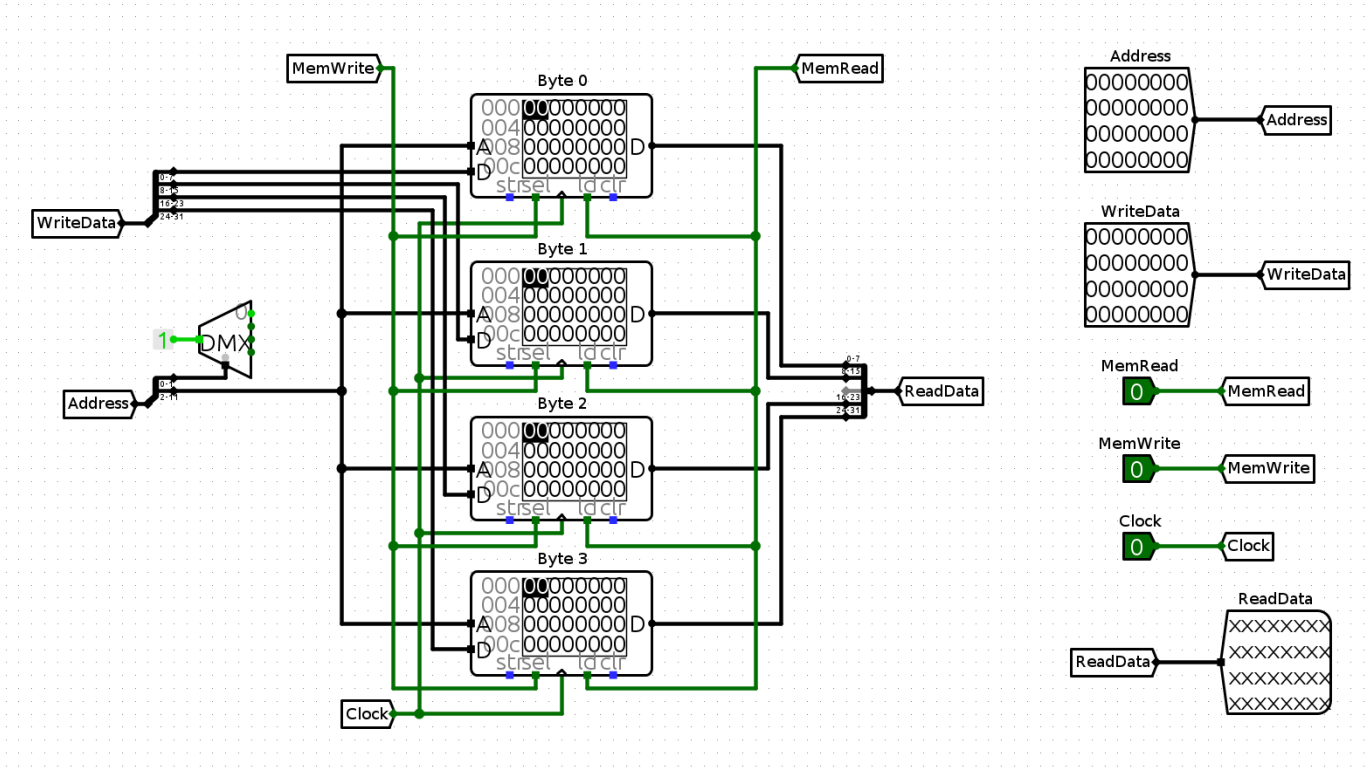
En la parte lógica tenemos una serie de **flags** las cuáles nos sirven para identificar si 2 valores son iguales, si hay un overflow de negativos o positivos.

Data Memory

El componente de data memory es escencial para el fucionamiento del ALU. Si hay una instruccion **load**, existe un multiplexor afuera que va a girar la salida de la intruccion al **DataMemory.** Recordemos que por el **Pipelining**, la memoria a leer viene definida desde la salida del stage anterior.



Subcircuito DataMemory



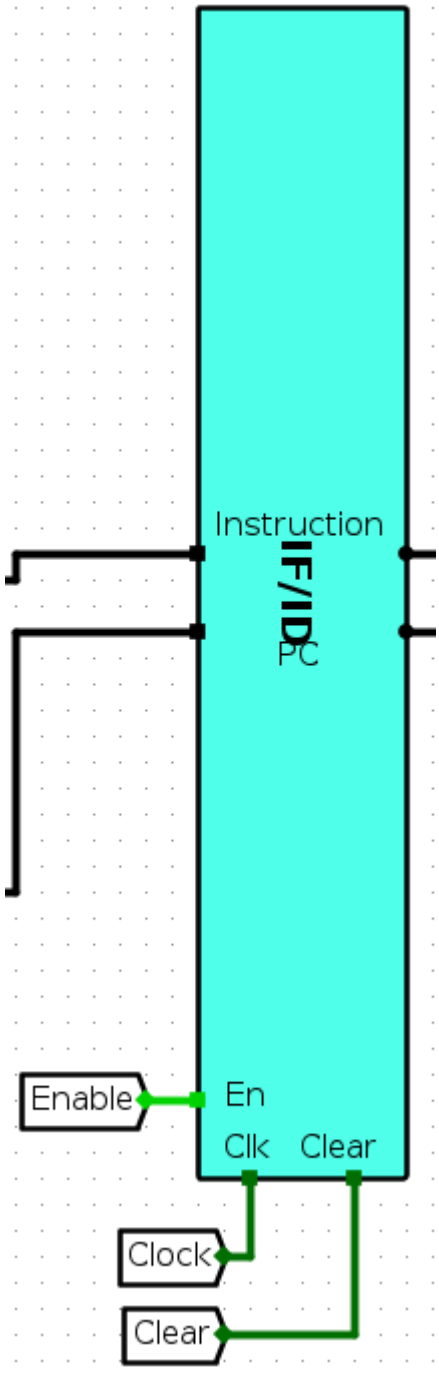
Circuito DataMemory

Nótese que el tunnel llamado **WriteData** es el mismo que se utiliza para el **Register File**

IF/ID

El stage **IF/ID** se encarga del

- Instruction Fetch
- Instruction Decode



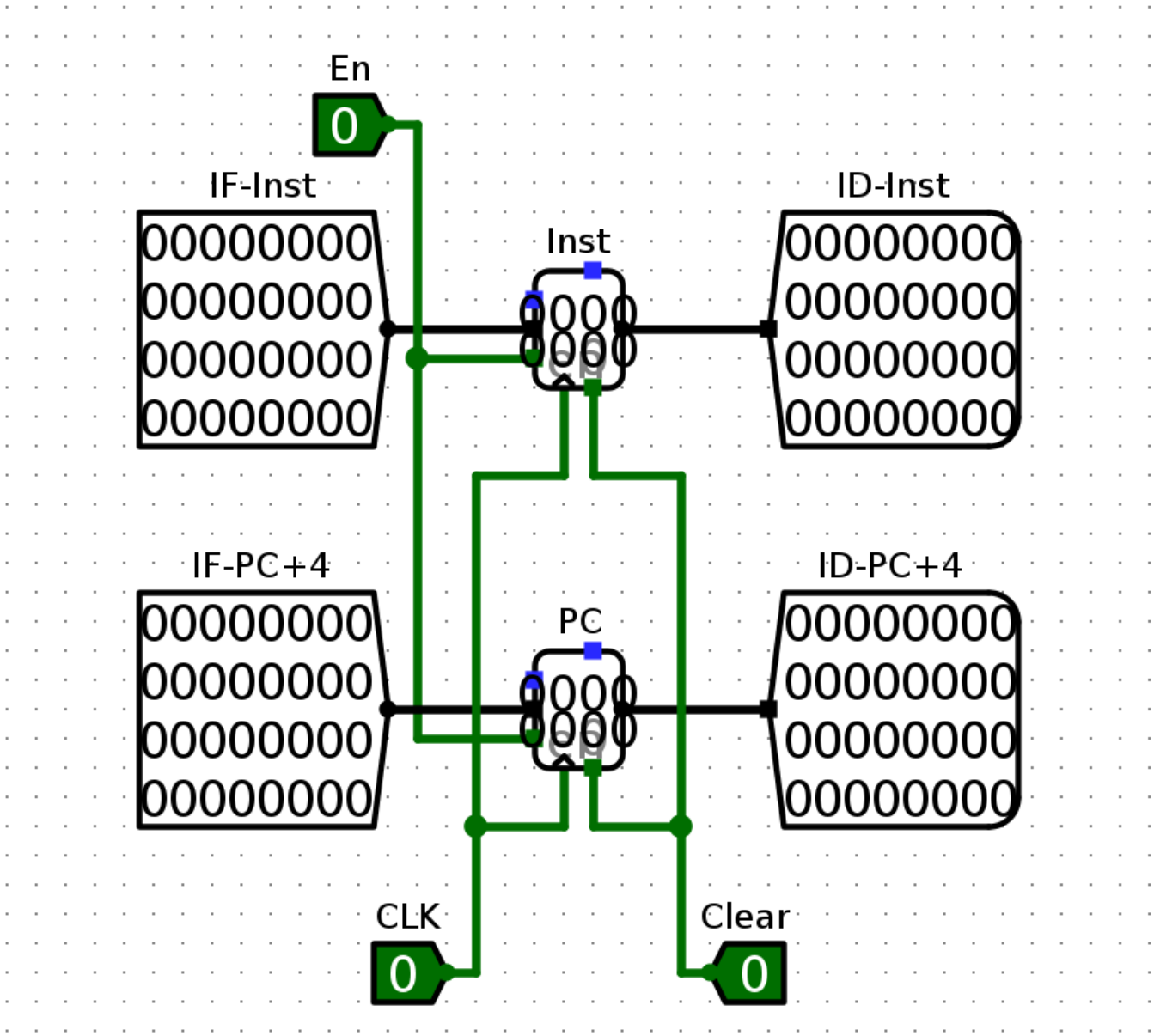
Subcircuito IF/ID

El proceso de **Instruction Fetch** se encarga de:

- Obtener la instruccion desde **Instruction Memory**
- Aumentar el valor de **Program Counter**

Mientras que el proceso de **Intruction Decode**:

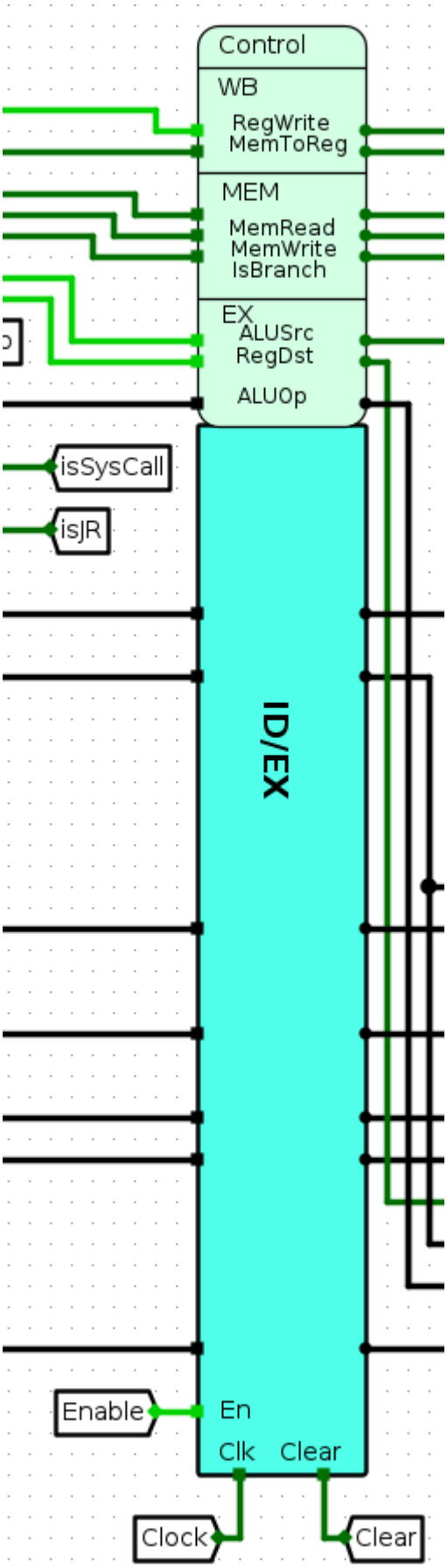
- Obtiene los valores de los registros de la instrucción
-



Circuito IF/ID

ID/EX

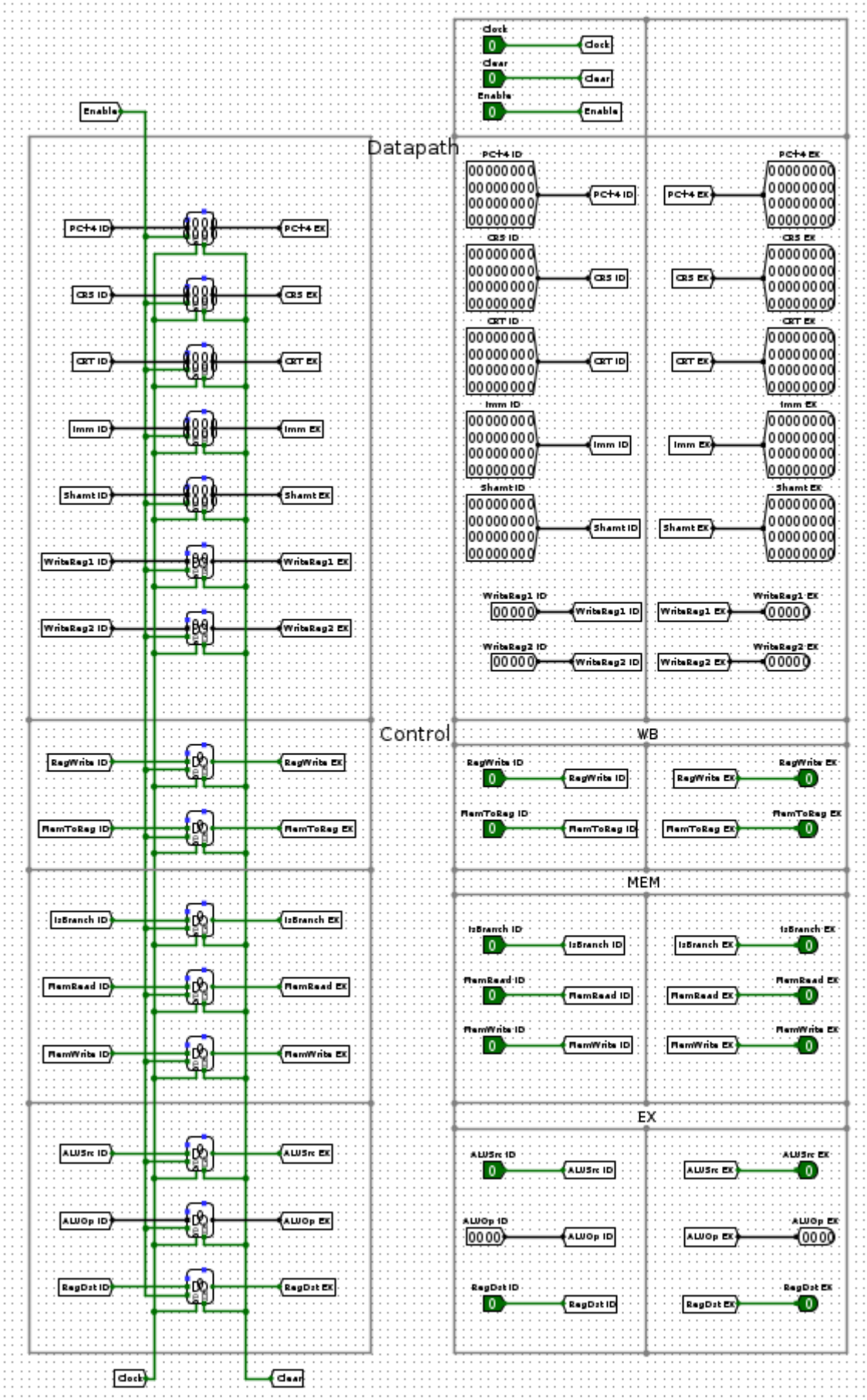
Este stage es más complejo. Ya se explico que hace un proceso **Instruction Decode** anteriormente, por lo tanto solo se va a mencionar el **Execute**.



Subcircuito ID/EX

Execute es un proceso que:

- En caso de ser una referencia de memoria, setea el OFFSET y la Base
- Es caso de recibir una referencia de un operacion aritemetica, hace la mate, por decirlo así.

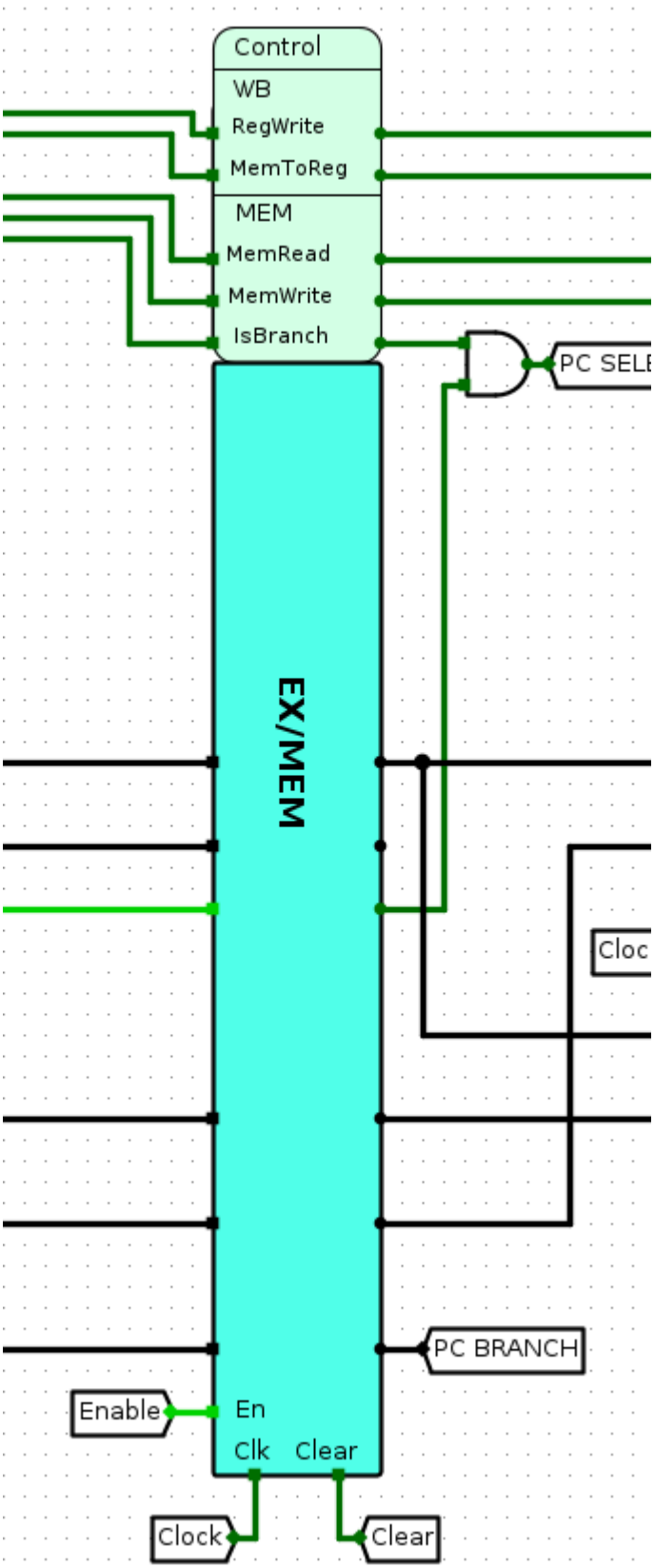


Circuito

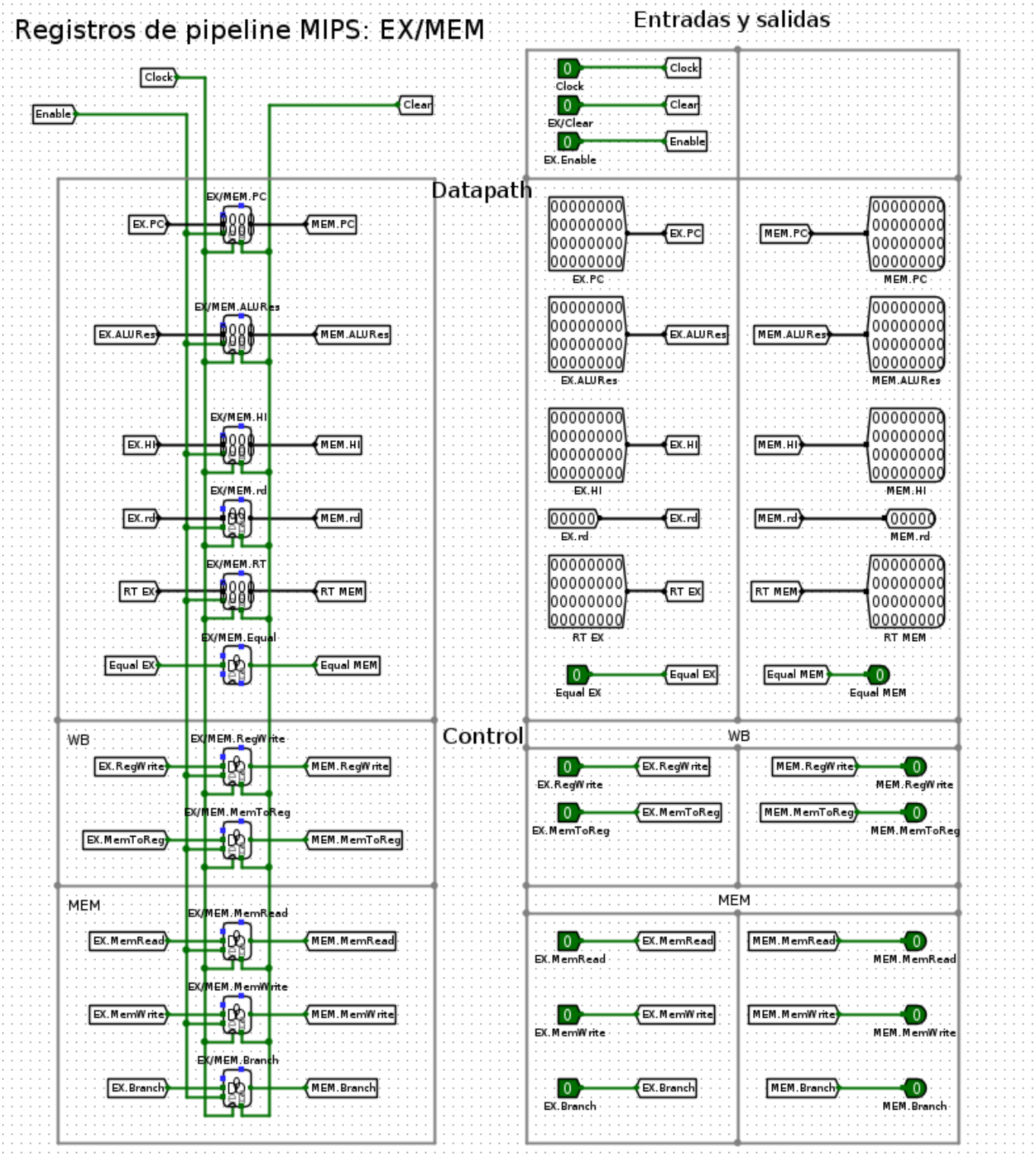
EX/MEM

El proceso **Memory** se encarga de:

- Acceder a memoria
- Si recibe un Branch, modificar el Program Counter



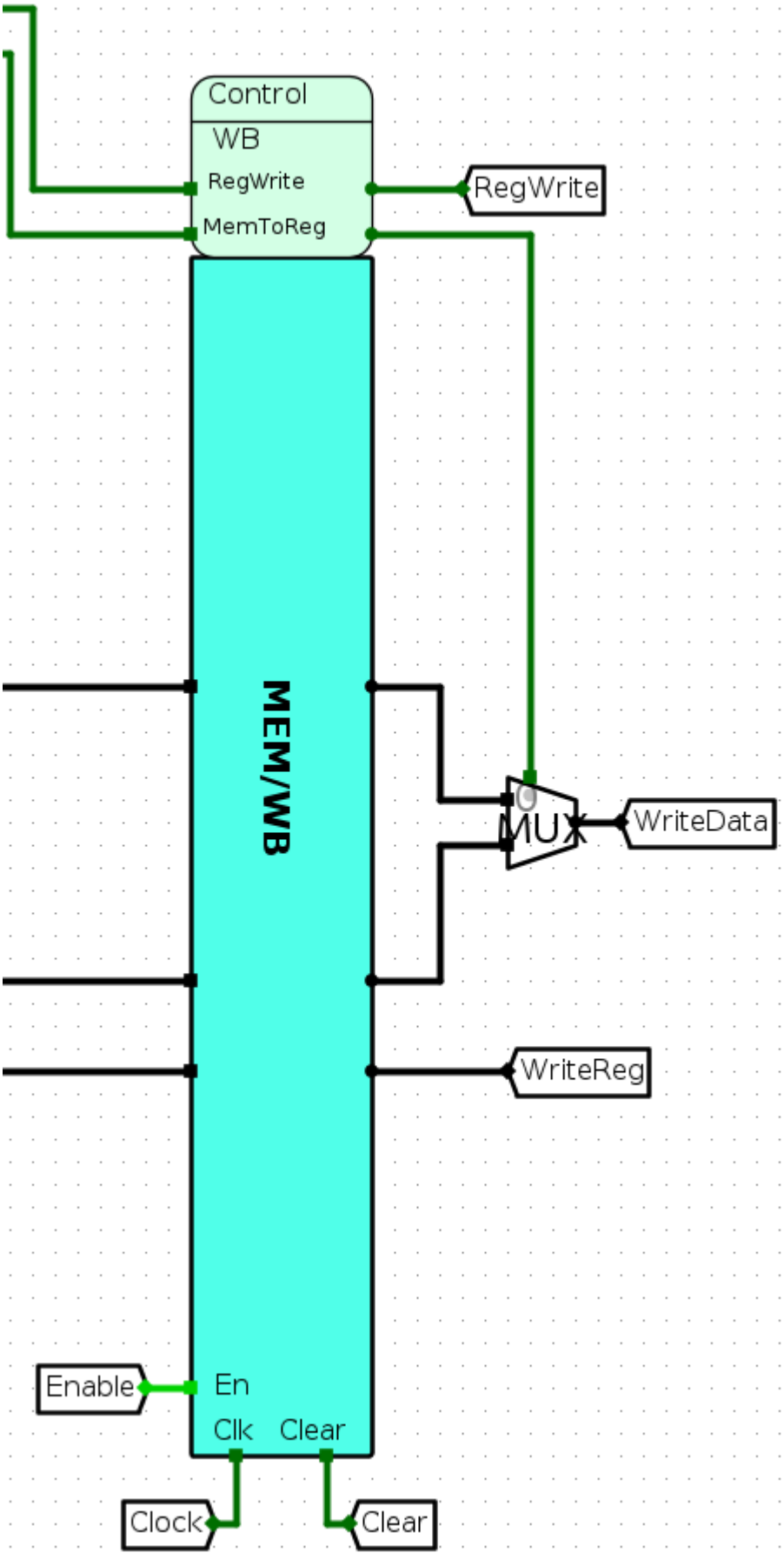
Surcircuito EX/MEM



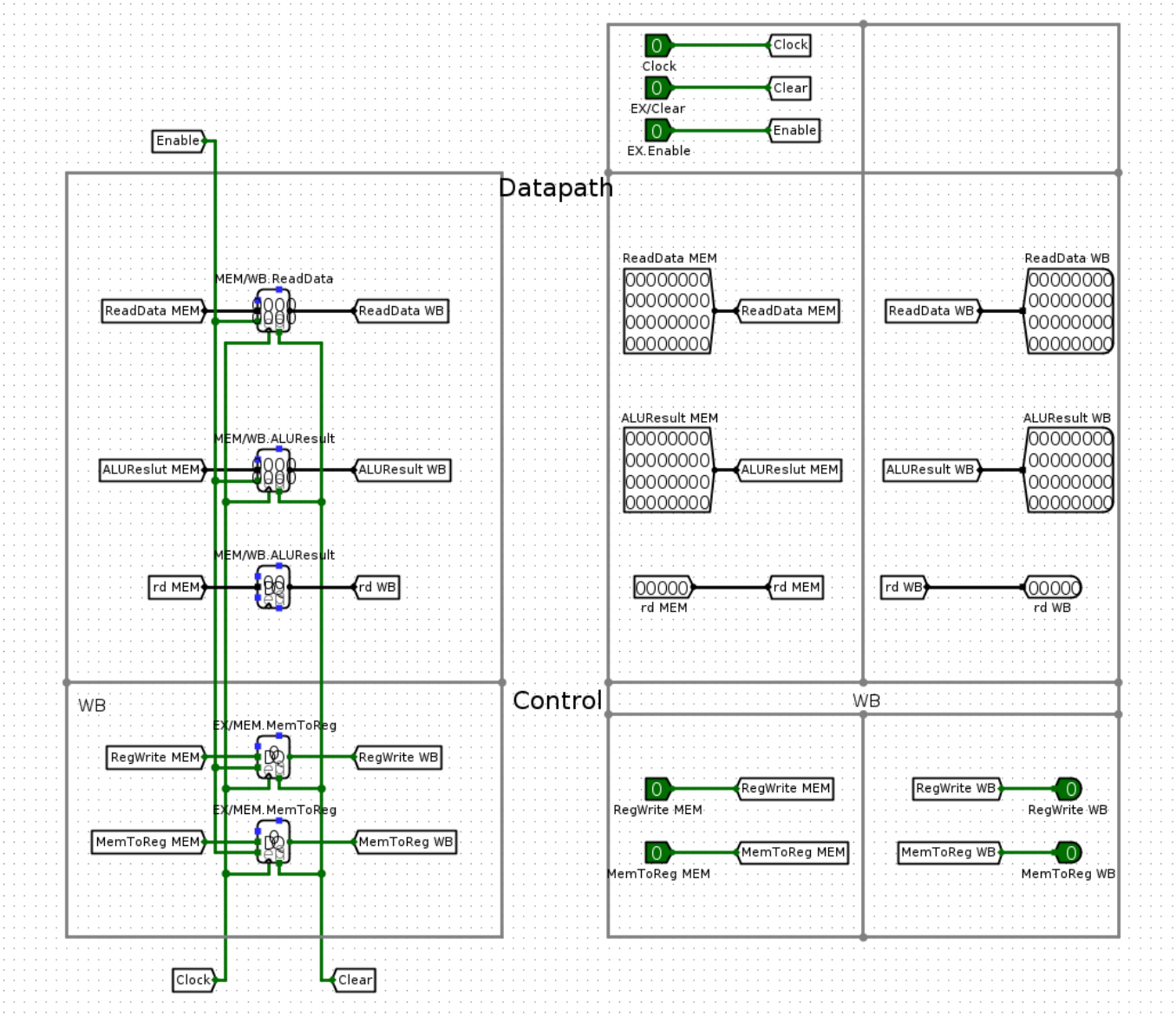
Circuito

MEM/WB

Finalmente, tenemos **WriteBack.** WB se encarga de poner un valor en un registro respectivo, por lo tanto está en comunicación directa con el **Register File.**

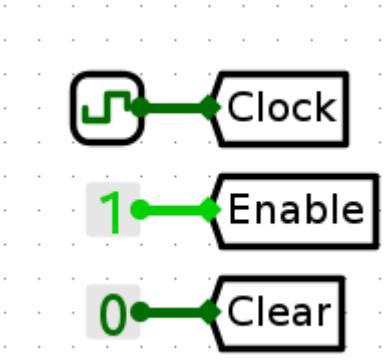


SubCircuito MEM/WB

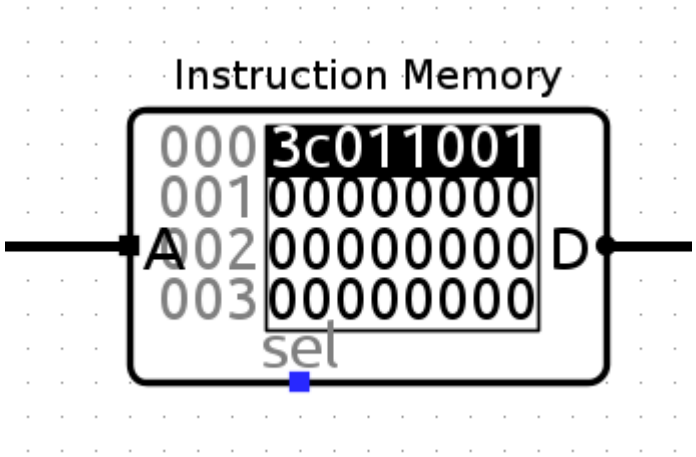


Circuito

Componentes Externos



Reloj global para todo el Circuito



Almacenamiento de las Intrucciones