UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco

# Proyecto Final DataPath

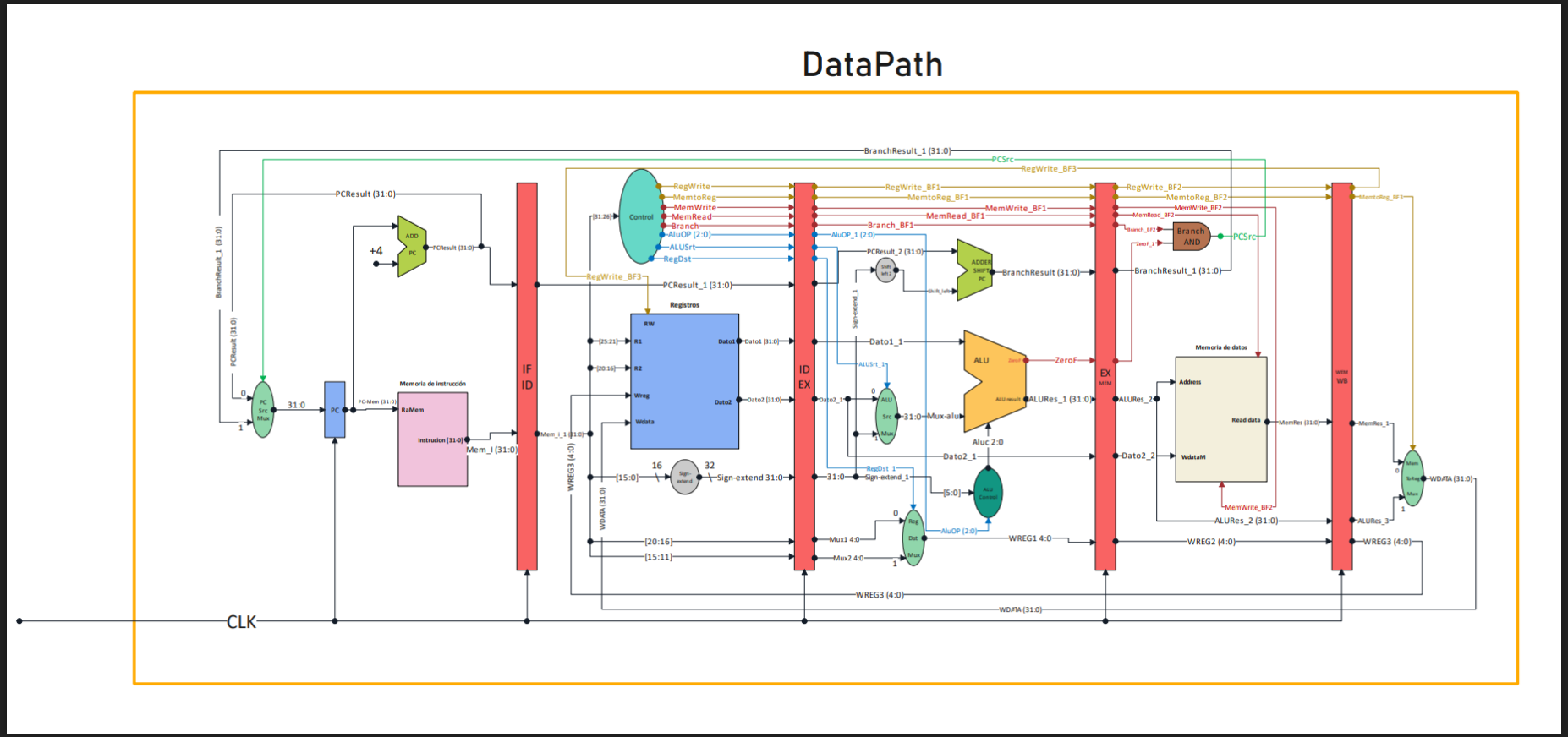

Equipo 2:

- Caleb Karim García Aguilar
- Luis Ángel Rosado Estrada
- Miguel Eduardo Espíritu Torres

Profesor:Jorge Ernesto López Arce

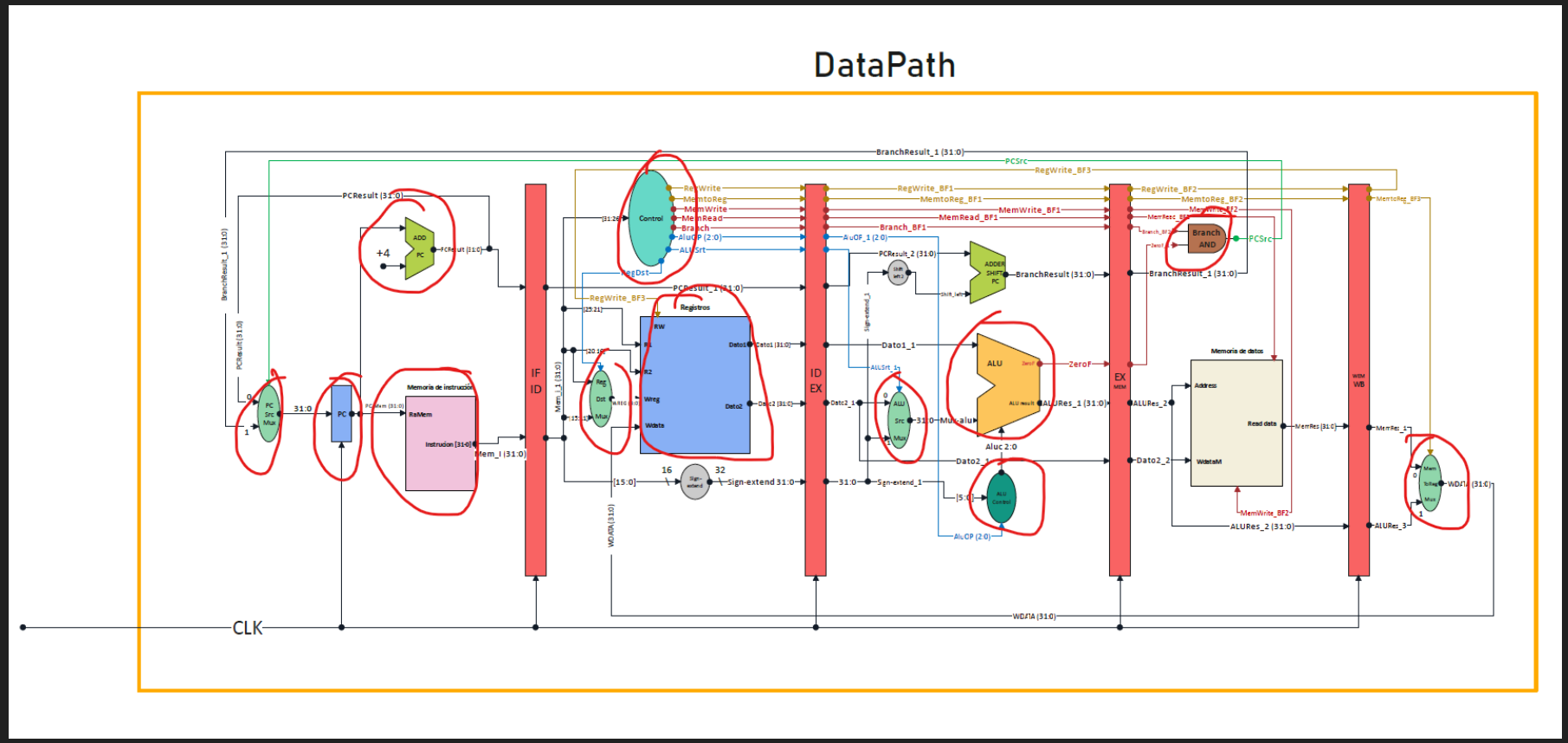

Seminario de Solución de Problemas de Arquitectura de Computadoras

# Diagrama final



# Diagrama Fase 1

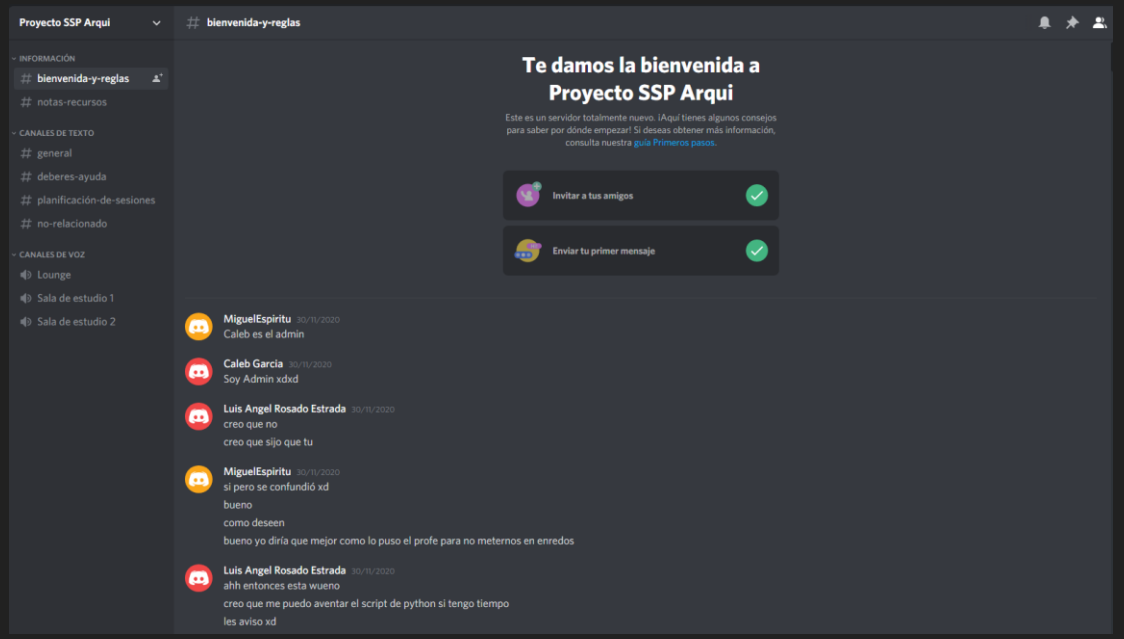
Para el diagrama utilicé un programa llamado “Wondershare EdrawMax”, decidí hacer el diagrama final desde el principio para que nos resultara más fácil lo que teníamos que hacer al final del proyecto, los módulos de la fase 1, son los que están en rojo. Para instrucciones tipo R.



# Reuniones de trabajo

Durante la realización del proyecto final tuvimos varias reuniones para ponernos de acuerdo, ayudarnos mutuamente,  resolver dudas, etc.

Estas reuniones se llevaron a cabo a través de un grupo de discord creado específicamente para el proyecto final.



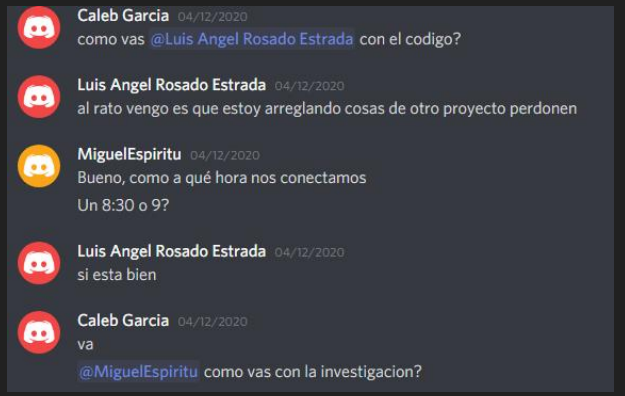
# Reuniones de trabajo

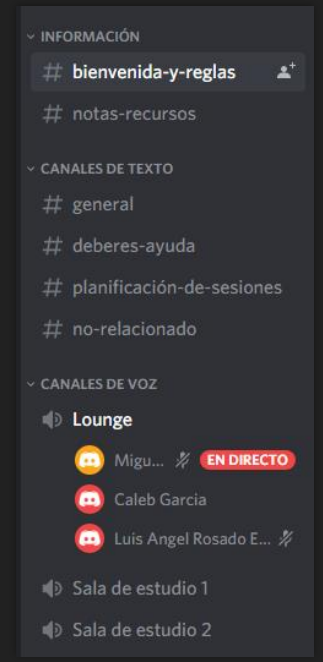
Tuvimos varias reuniones durante la semana, nos reunimos estos días:

En estas reuniones resolvimos duda y nos pusimos de acuerdo en ciertas cosas.

- Lunes 30 de Noviembre después de la clase con una duración aproximada de 10 min (9:00am - 9:10am)
- Miércoles 2 de diciembre, a las 7:00 pm con una duración de aproximadamente 40 minutos.
- Viernes 4 de diciembre, a las 8:30pm con una duración aproximada de 20 minutos
- Sábado 5 de diciembre a las 8:30pm con una duración aproximada de 1 hora.

Además de estas reuniones, cada día de la semana estuvimos en constante comunicación por chat en discord, ofreciendo ayuda mutua, para resolver lo antes posible cualquier duda que tuviera cada uno.



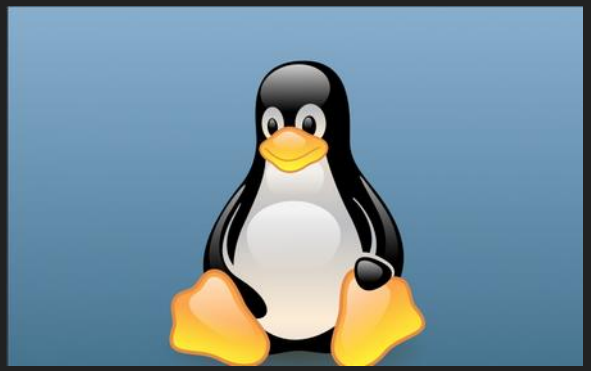



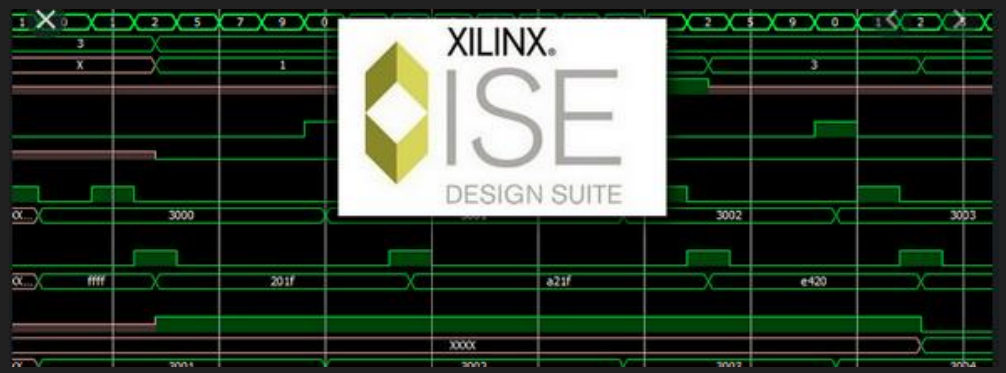
# Código

Al ser este un trabajo de suma importancia opte por buscar nuevas herramientas para programar este proyecto.

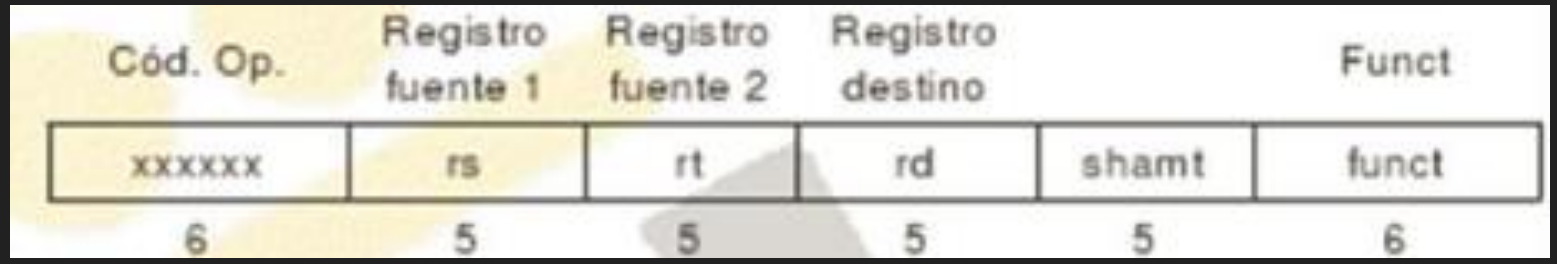
Utilicé una maquina virtual y  linux e utilice ISE de Xilinx ya que con las investigaciones sabia que Xilinx eran los amos del mercado de FPGAs por lo tanto deduje que su herramienta para utilizar el lenguaje de descripcion de hardware deberia ser buena.

Pasaron varias cosas desafortunadas por mi poco conocimiento tanto de linux como de ISE pero al final me termino gustando bastante trabajar en linux y pienso en seguir utilizandolo en proyectos venideros





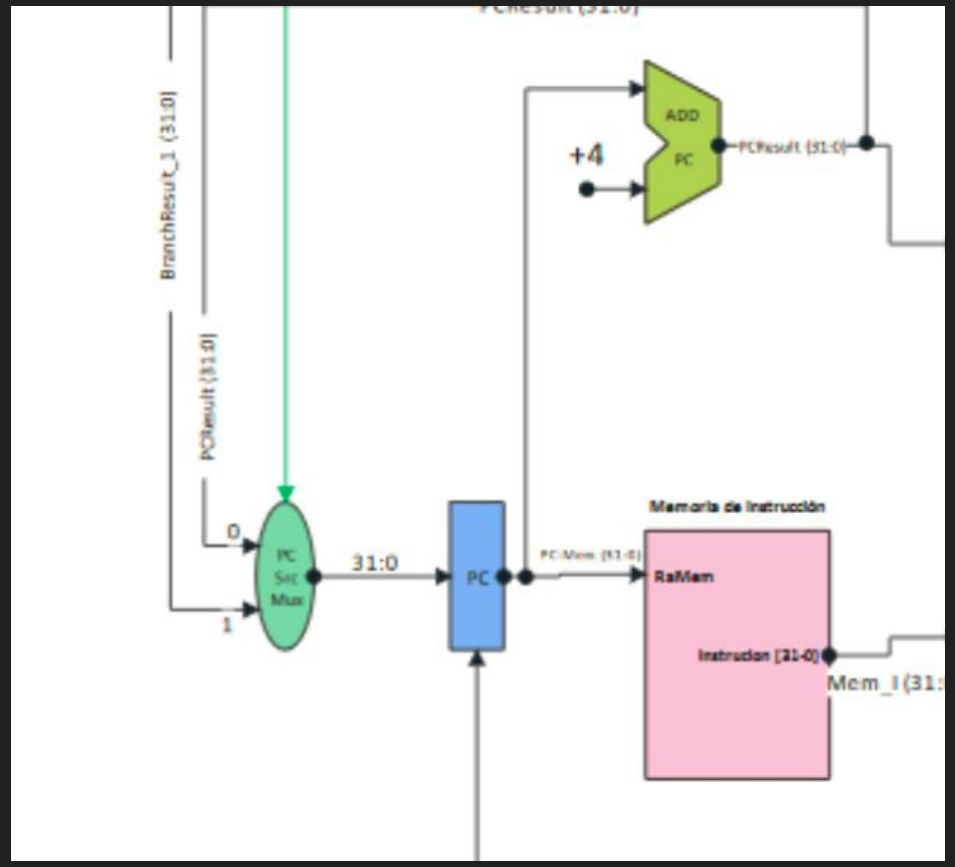
# Instrucciones de Tipo R




El tipo de formato de las instrucciones aritméticas y lógicas es de tipo R y el número de operandos en una operación de este tipo es siempre tres. Estos operandos son siempre registros, el modo de direccionamiento empleado es,por tanto,de registro.

La parte a destacar a diferencia de los otros tipos de instrucciones es la ultima parte llamada Function que es quien ordena cual operacion en especifico se debe ejecutar.

# El Ciclo Fetch




El ciclo fetch consta de 3 modulos:
-Memoria de Instrucciones
-PC(Program Counter)
-ADD

Este ciclo sirve para optimizar la lectura de instrucciones culesquiera sea su tipo separandolas en bytes para luego juntarlar en una instruccion de 32 bits

Ademas simplifica la manera de trabajar las instrucciones de salto ya que solo se modifica el valor de entrada de PC (debe ser multiplo de 4) para que PC le diga a la Instruction Memory que lea cierta instruccion en especifico

# Archivos en Verilog

Tal vez una de las partes mas complicadas de este proyecto fue la implementacion de codigo maquina a partir del uso de instanciar archivos tanto en el BR y en IstructionMem

Surgieron problemas a partir de que el archivo no completaba por comleto las memorias y al intentar implementarlo y crear un esquema RTL surgian errores, pero al final modifique un poco el archivo directamente y asi solucione el problema

```
module BR(
input Regwrite,
input [31:0]Din,
input [4:0]RA1,RA2,WA,
output  [31:0]DR1,DR2

);

reg [31:0]Breg[0:128];

initial
begin

$readmemb("/home/ise/Desktop/Proyecto/TestF1_BReg.mem",Breg);
```

# Pipeline

```
1  `timescale 1ns / 1ns
2
3  module PC(
4  input [31:0] PCNext,
5  input CLK,
6  output reg  [31:0]PCResult
7  );
8  initial begin
9
10 PCResult <= 0;
11
12 end
13
14 always @(posedge CLK)
15 begin
16
17 PCResult <= PCNext;
18 end
19
20 endmodule
21
```

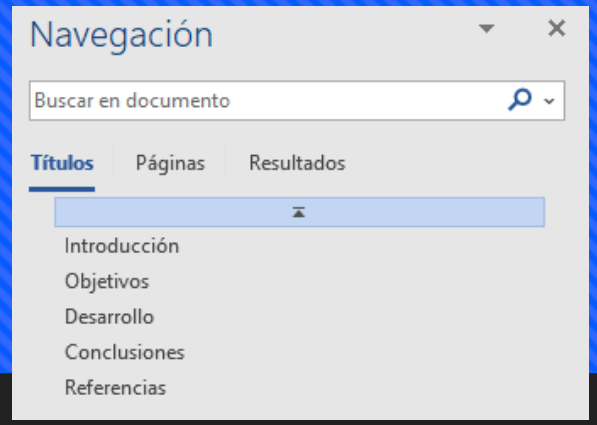
**Buffers(PIPELINE)**

Una de las nuevas cosas en este proyecto fue la implementacion del pipelining lo que simplemente es hacer esperar a ciertos datos para que el flujo de datos sea congruente.

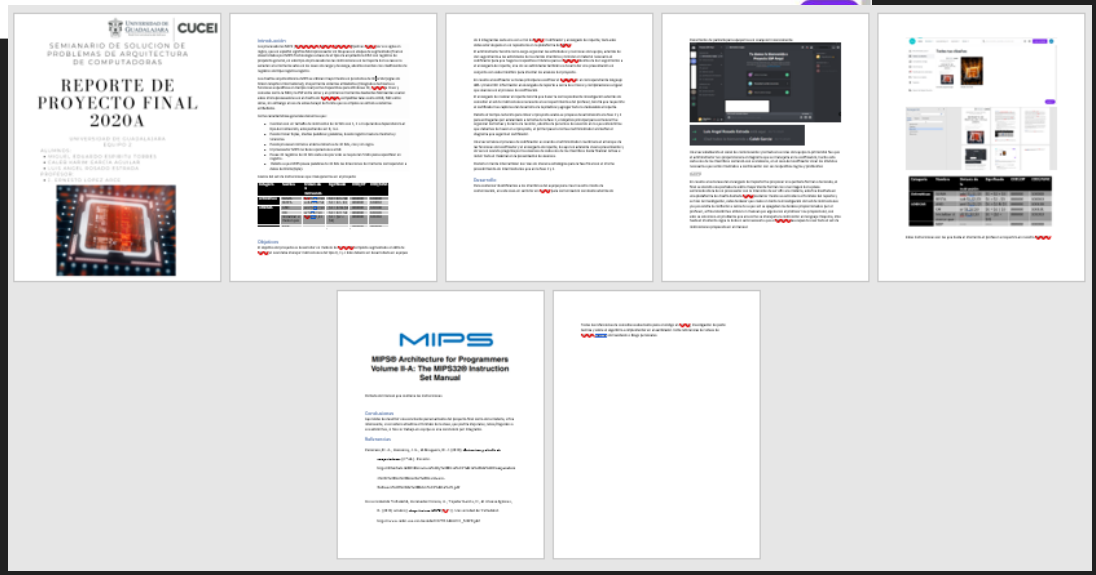
```
1  `timescale 1ns / 1ns
2
3  module IF_ID(
4     input [31:0]PCResult,
5     input [31:0]Mem_I,
6     input clk,
7     output reg [31:0]PCResult1,
8     output reg [31:0]Mem_I1
9      );
10
11 always @(posedge clk)
12 begin
13 PCResult1 = PCResult;
14 Mem_I1 = Mem_I;
15
16 end
17 endmodule
18
```

# Reporte

- Sobre el reporte se decidió manejar una portada mayormente formal en lugar de una decorada, se utilizó la plataforma de diseño "Canva" para su creación, las mayores dificultades fueron encontrar un set de instrucciones que se acoplara a las necesidades del testeo del profesor de forma que todo saliera acorde a lo señalado en el test que nos presentó, además otra dificultad fue encontrar el lenguaje máquina que manejaría el módulo, todo esto se resolvió al encontrar el manual de set de instrucciones de MIPS.





# Agradecemos su atención!