UNIVERSIDAD DE GUADALAJARA
Red Universitaria de Jalisco

CUCEI

SEMIANARIO DE SOLUCION DE PROBLEMAS DE ARQUITECTURA DE COMPUTADORAS

REPORTE DE PROYECTO FINAL 2020A

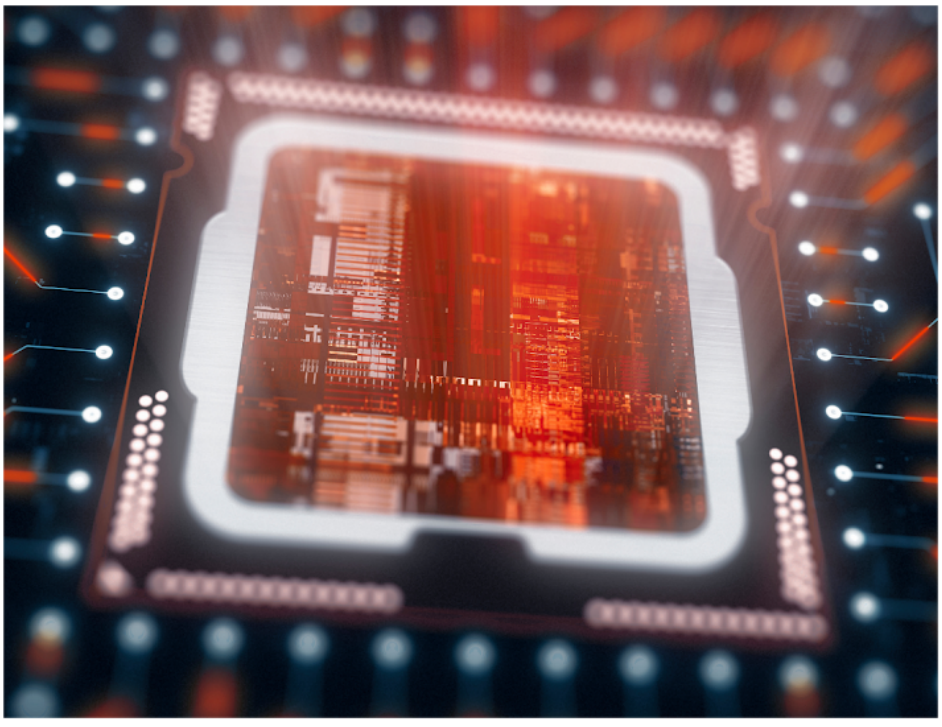
UNIVERSIDAD DE GUADALAJARA
EQUIPO 2

ALUMNOS:

- MIGUEL EDUARDO ESPIRITU TORRES
- CALEB KARIM GARCÍA AGUILAR
- LUIS ANGEL ROSADO ESTRADA

PROFESOR:

- J. ERNESTO LOPEZ ARCE



Introducción

Los procesadores MIPS (Microprocessor without Interlocked Pipelines Stages por sus siglas en inglés, que en español significa Microprocesador sin bloqueo en etapas de segmentado) fueron desarrollados por MIPS Technologies a base de el tipo de arquitectura RISC con registros de propósito general, en este tipo de procesadores las instrucciones en la mayoría de los casos no acceden a la memoria salvo en los casos de carga y descarga, además cuentan con clasificación de registros del tipo registro-registro.

Los diseños arquitectónicos MIPS se utilizan mayormente en productos de línea SGI (siglas de Silicon Graphics International), mayormente sistemas embebidos (Integrados dedicados a funciones específicas en tiempo real) como dispositivos para Windows CE, routers de Cisco y consolas como la N64 y la PSP entre otras y en primeros momentos bastantes fabricantes usaron estos microprocesadores en el diseño de Workstations, compañías tales como ACER, NEC entre otras, sin embargo el uso de estas decayó de forma que su empleo se enfocó a sistemas embebidos.

Como características generales tenemos que:

- Cuentan con un tamaño de instrucción de 32 bits con 1, 2 o 3 operandos dependiendo el tipo de instrucción, este pudiendo ser R, I o J.
- Puede mover bytes, medias palabras y palabras, desde registró hacia la memoria y viceversa.
- Puede procesar números enteros binarios de 32 bits, con y sin signo.
- El procesador MIPS no tiene operaciones al bit.
- Posee 32 registros de 32 bits cada uno por esto se requieren 5 bits para especificar un registro.
- Debido a que MIPS posee palabras de 32 bits las direcciones de memoria corresponden a datos de 8 bits (byte).

Acerca del set de instrucciones que manejaremos en el proyecto

Categoría	Nombre	Sintaxis de la instrucción	Significado	COD|OP	COD|FUNC
Aritméticas	SUMA	add $1,$2,$3	$1 = $2 + $3	000000	100000
	RESTA	sub $1,$2,$3	$1 = $2 - $3	000000	100010
LÓGICAS	AND	and $1,$2,$3	$1 = $2 & $3	000000	100100
	OR	or $1,$2,$3	$1 = $2 | $3	000000	100101
	Inicializar si menor que	slt $1,$2,$3	$1 = ($2 < $3)	000000	101010
	NOP	-----	-----	000000	000000

Objetivos

El objetivo del proyecto es desarrollar un módulo de Datapath Completo segmentado en HDL de Verilog el cual debe manejar instrucciones del tipo R, I y J. Este deberá ser desarrollado en equipos de 3 integrantes cada uno con el rol de Admin, Codificador y encargado de reporte, todo esto debe estar alojado en un repositorio en la plataforma de github.

El administrador tendrá como cargo organizar las actividades y reuniones del equipo, además de dar seguimiento a las actividades de los demás miembros y brindar el material necesario al codificador para que haga los respectivos módulos para el datapath además de dar seguimiento a el encargado de reporte, una de sus actividades también es desarrollar una presentación en conjunto con cada miembro para mostrar los avances del proyecto.

En cuanto al codificador su tarea principal es codificar el datapath en el correspondiente lenguaje HDL y transmitir información al encargado de reporte a cerca de errores y complicaciones al igual que avances en el proceso de codificación.

El encargado de realizar el reporte tendrá que hacer la correspondiente investigación además de consultar el set de instrucciones necesario a los requerimientos del profesor, tendrá que requerir al codificador las capturas del desarrollo de la práctica y agregar todo lo destacable al reporte.

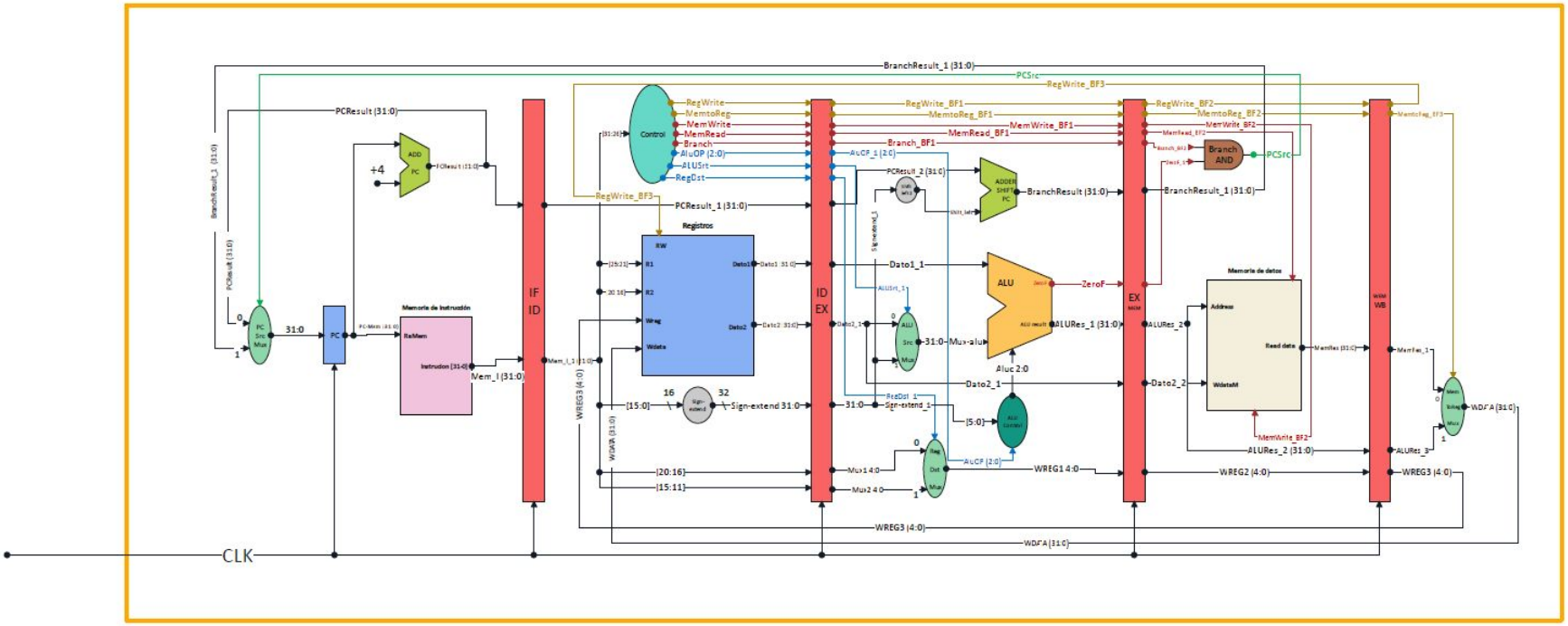
Debido al tiempo reducido para llevar el proyecto acabó se propuso desarrollaremos la fase 1 y 2 para entregarlas por adelantado a la fecha de la fase 1, el objetivo principal para comenzar fue organizar las fechas y horario de reunión, además de ponernos de acuerdo en lo que entendimos que debemos de hacer en el proyecto, el primer paso lo toma el administrador al diseñar el diagrama que seguirá el codificador.

Una vez arranca el proceso de codificación es cuando el administrador coordinará el arranque de las funciones del codificador y el encargado de reporte, de aquí en adelante crea la presentación y de vez en cuando pregunta por los avances de cada uno de los miembros hasta finalizar la fase e incluir todo el material en la presentación de avances.

Posteriormente intercambiar los roles de manera estratégica para la fase final con el mismo procedimiento de intermediarios que en la fase 1 y 2.

El diagrama después de varias fases de solución y corrección de errores resultó será el que adjuntamos en una hoja completa a continuación, esto para su mejor apreciación:

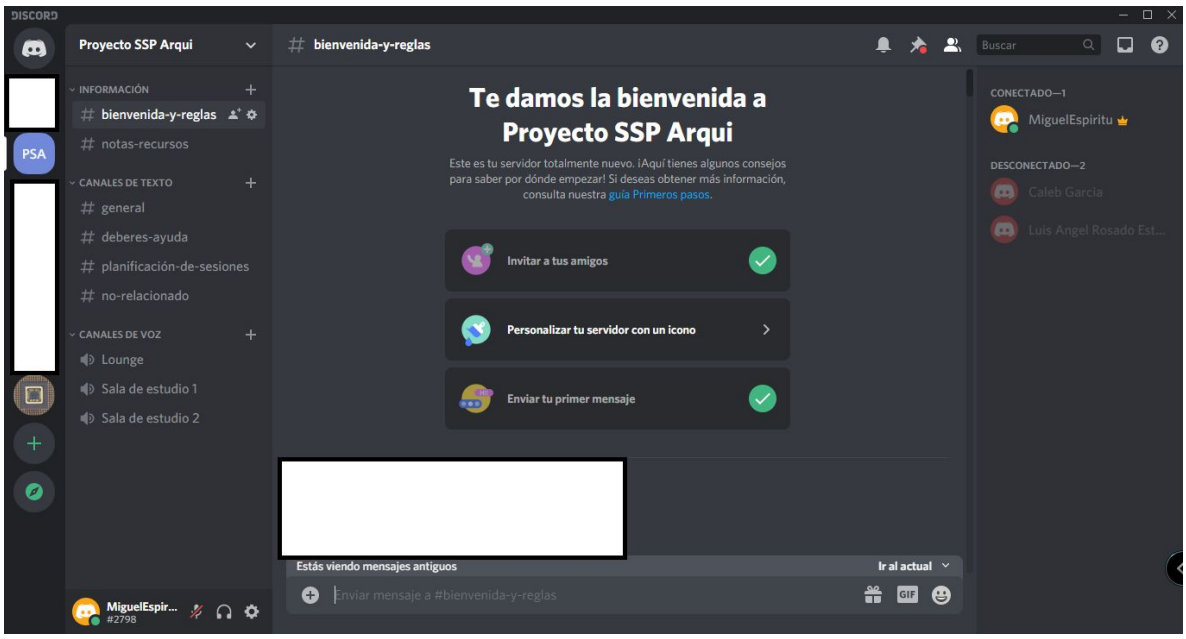
DataPath

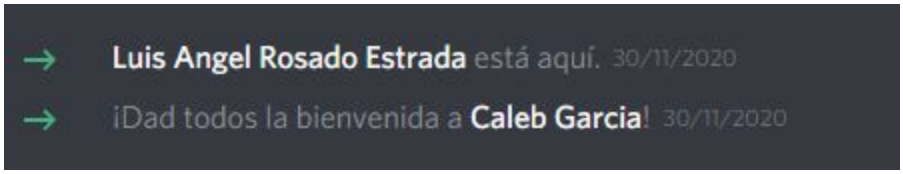

BranchResult_1 (31:0)
PCSrc
RegWrite_BF3
PCResult (31:0)
+4
ADD
PC
IF
ID
Control
RegWrite
MemtoReg
MemWrite
MemRead
Branch
AluOP (2:0)
ALUSrt
RegDst
RegWrite_BF1
MemtoReg_BF1
MemWrite_BF1
MemRead_BF1
Branch_BF1
AluCF_1 (2:0)
RegWrite_BF2
MemtoReg_BF2
MemWrite_BF2
MemRead_BF2
Branch_BF2
Branch AND
PCResult_1 (31:0)
PCResult_2 (31:0)
BranchResult (31:0)
BranchResult_1 (31:0)
Registros
RW
R1
R2
Wreg
Wdata
Dato1
Dato2
Dato1_1
Dato2_1
ID
EX
ALU
ZeroF
ALURes_1 (31:0)
Mux-alu
Aluc 2:0
EX
MEM
ALURes_2
Dato2_2
Memoria de datos
Address
Read data
WideMI
MemRes (31:0)
MemWrite_BF2
ALURes_2 (31:0)
WREG2 (4:0)
MEM
WB
MemRes_1
ALURes_3
WREG3 (4:0)
WDFA (31:0)
Memoria de instrucción
RaMem
Instrucion (31:0)
Mem_I (31:0)
PC
31:0
WREG3 (4:0)
WDFA (31:0)
16
32
Sign-extend
Sign-extend 31:0
Sign-extend_1
[15:0]
[5:0]
[20:16]
[15:11]
Mux1 4:0
Mux2 4:0
WREG1 4:0
CLK

Desarrollo

Primeros pasos

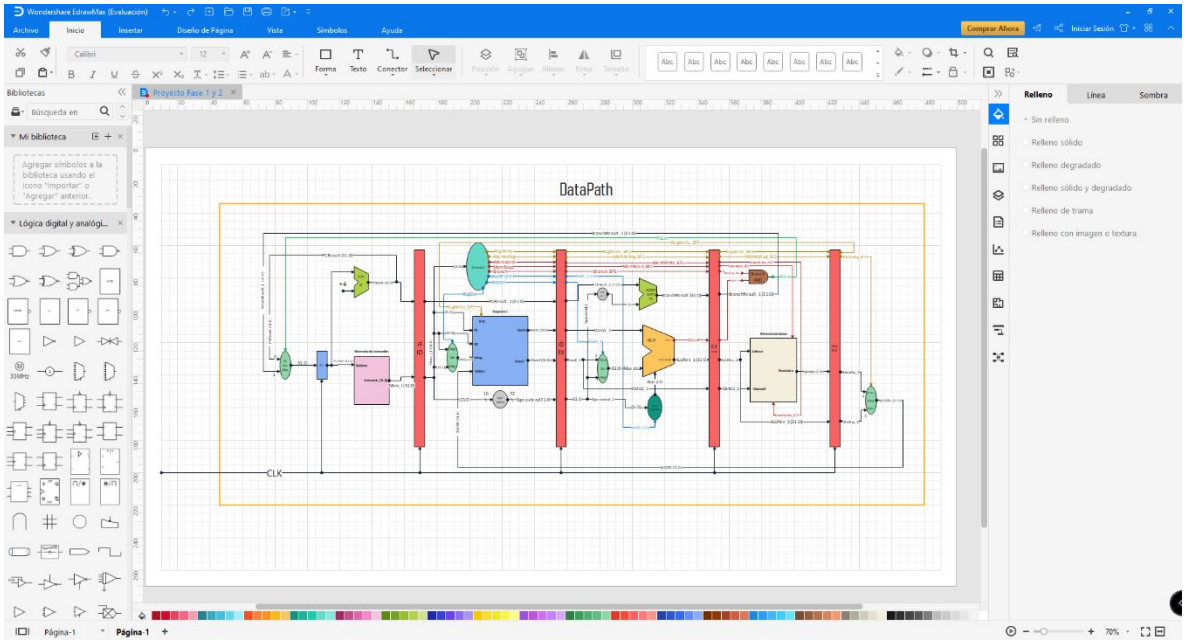
Para comenzar identificamos a los miembros del equipo para crear nuestro medio de comunicación, en este caso un servidor en discord para comunicación constante además de transmisión de pantalla para apoyarnos en cualquier inconveniente.





Administrador

Una vez establecido el canal de comunicación y tomados los roles del equipo lo primordial fue que el administrador nos proporcionará el diagrama que se manejaría en la codificación, hecho esto cada uno de los miembros comenzó con sus labores.



El diagrama fue desarrollado en Wondershare EdrawMax, las dificultades presentadas durante esta fase para el administrador fueron problemas en el nombrado de los cables y las conexiones, además de la decisión de ejecutar la fase 1 y 2 en un solo paso, de esta forma acelerar el tiempo de conclusión, estos errores fueron notados por el codificador y notificados al administrador para la posterior resolución de ellos y revisión por cada uno de los miembros del equipo.

Otra de las tareas del administrador fue crear el archivo powerpoint en el que trabajaríamos para presentar avances al profesor está siendo resuelta por SharePoint de la plataforma de office 365 de Microsoft.

Codificador

El trabajo del codificador lleva un gran peso en este proyecto, a continuación se mostrarán los módulos más destacables del proyecto y las principales dificultades que se nos presentaron en el proceso de codificación:

- Código de la unidad de control

```
module UnidadControl(
input [5:0]op,
output reg MemToReg,
output reg MemToWrite,
output reg [2:0]ALUOP,
output reg regwrite
);


always@*
begin
case(op)
        0:begin
        MemToReg <= 1;
        MemToWrite <= 0;
        ALUOP <= 1;
        regwrite <= 1;
        end
        43: ALUOP <= 2;
        35: ALUOP <= 2;
        4: ALUOP <= 3;
endcase
end


endmodule

```

- Codigo ALU_Control

```
module ALU_CONTROL(
input [2:0]ALUOp,
input [5:0]FunCode,
output reg [2:0]ALUCt
);


always @*
begin
case(ALUOp)
        1:      begin
                        case(FunCode)
                                36: ALUCt <= 0; //and
                                37: ALUCt <= 1; //or
                                32: ALUCt <= 2; //suma
                                34: ALUCt <= 3; //resta
                                42: ALUCt <= 4; //slt
                                default: ALUCt<=0;
                        endcase
                end
        2: ALUCt <= 0;
        3: ALUCt <= 0;
endcase
end
endmodule
```

- Módulo PC

```
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
22
```

- Módulo Testbetch

```
...lo-/OneDrive/Escritorio/Proyecto/TB_DataPath.v
Ln#
1   `timescale 1ns / 1ns
2
3   module TB_DataPath;
4
5           reg clk = 0;
6
7           DataPath uut (
8                   .clk(clk)
9           );
10
11  always #50 clk = !clk;
12
13  initial
14  begin
15      @(posedge clk);
16  end
17
18
19  endmodule
20
21
```

- Módulo ALU

```
module ALU(
input [2:0] AluCT,
input [31:0] A,B,
output reg [31:0]z,
output reg zf
);

always @(AluCT,A,B)
begin
case(AluCT)
        0:z <= A&B;
        1:z <= A|B;
        2:z <= A+B;
        3:z <= A-B;
        4:z <= A<B?1:0;
        default: z <= 0;
endcase
end
always @*
begin
 //escribir
 if (z == 0)
   zf = 1;

 //leer
 else
    zf = 0;

end
```

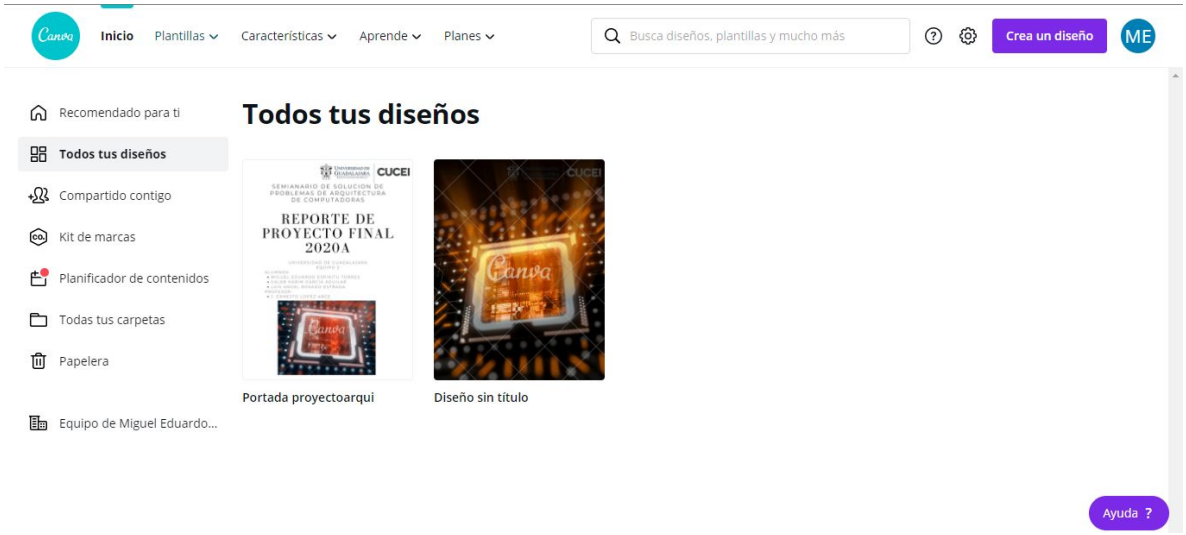
Los módulos más destacables del proyecto principalmente los que controlaban las funciones de otros módulos, estos siendo el módulo de ALU_Control el cual se encarga de mandar señales al módulo ALU para ejecutar las respectivas operaciones, unidad de control que es la que principalmente dirige el datapath dando instrucciones al ALU_Control sobre una operación en particular o una operación brindada por “FUNC”, como módulos no mucho menos importantes consideramos a ALU quien es quien ejecuta las operaciones, PC quien es el primer paso para el pipelining y el testbench del módulo general del datapath.

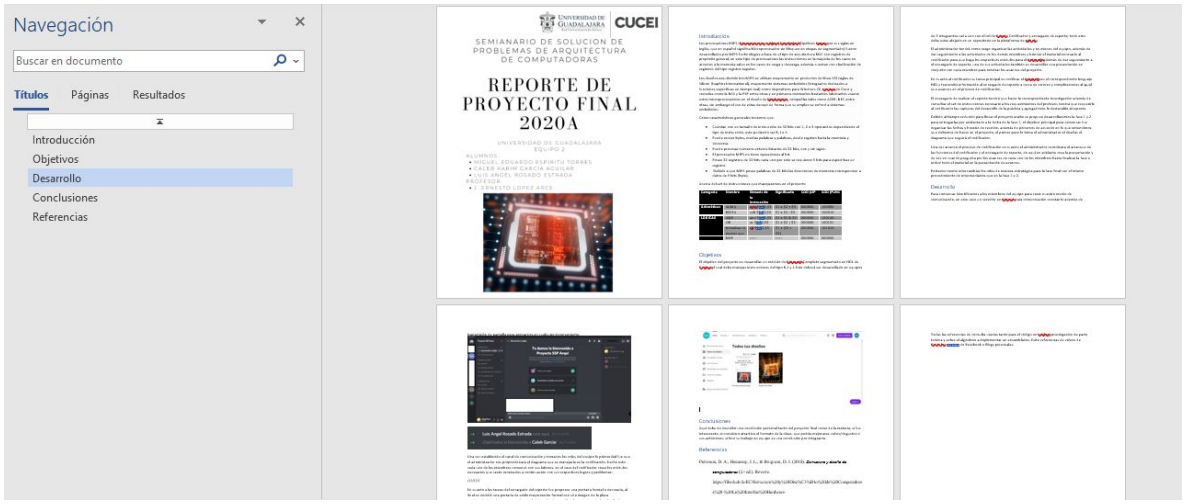
Las dificultades a destacar fueron precisamente en estos módulos, los cuales son los más importantes, era primordial la correcta codificación de la unidad de control y el controlador de la alu, puesto que estos dependen de la instrucción que actualmente está en ejecución, principalmente de los 6 bits más significativos y siendo instrucción tipo R de los 6 bits menos significativos.

Reporte

En cuanto a las tareas del encargado del reporte fue proponer una portada formal o decorada, al final se decidió una portada de estilo mayormente formal con una imagen de la placa semiconductora de un procesador con la intención de ser afín a la materia, esta fue diseñada en una plataforma de diseño llamada Canva, posteriormente se estructuró el formato del reporte y se hizo la investigación, cabe destacar que costó un tanto la investigación del set de instrucciones ya

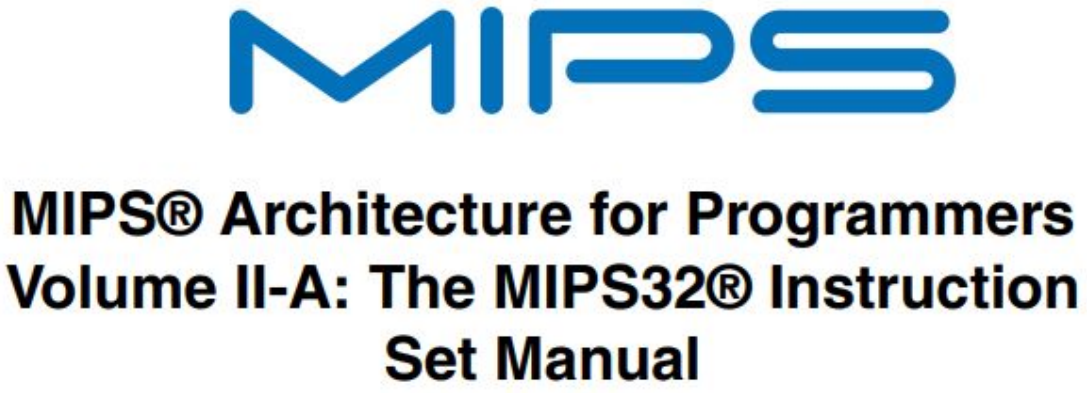
que existía la confusión a cerca de a que set se apegaban los testeos proporcionados por el profesor, al final decidimos utilizar un manual que alguna vez el profesor nos proporcionó, con este se soluciono un problema que era como se manejaría la instrucción en lenguaje máquina, más hasta el momento sigue la duda si será necesario que el datapath sea capaz de usar todo el set de instrucciones propuesto en el manual.





Categoría	Nombre	Sintaxis de la instrucción	Significado	COD|OP	COD|FUNC
Aritméticas	SUMA	add $1,$2,$3	$1 = $2 + $3	000000	100000
	RESTA	sub $1,$2,$3	$1 = $2 - $3	000000	100010
LOGICAS	AND	and $1,$2,$3	$1 = $2 & $3	000000	100100
	OR	or $1,$2,$3	$1 = $2 | $3	000000	100101
	Inicializar si menor que	slt $1,$2,$3	$1 = ($2 < $3)	000000	101010
	NOP	-----	-----	000000	000000

Estas instrucciones son las que hasta el momento el profesor ha requerido en nuestro datapath.



Portada del manual que contiene las instrucciones.

Conclusiones

Administrador Fase 1: Caleb Karim García Aguilar

En este proyecto, fue muy bueno para comprender de una manera mucho más amplia la arquitectura de un procesador MIPS. En mi rol de administrador en este proyecto, no fue tan difícil debido a que, organizar a mi equipo de trabajo fue muy sencillo, ya que todos nos comportamos a la altura, con responsabilidad y buena actitud para trabajar, además que logramos ponernos de acuerdo fácilmente para hacer las reuniones. Para la realización del diagrama no tuve muchas complicaciones a excepción de que, en un inicio no encontraba el programa adecuado donde hacerlo, y tambíen tuve un poco de confusión al decidir qué diagrama íbamos a utilizar para

esta fase, una vez consultado con mis compañeros, llegamos al acuerdo que íbamos a trabajar sobre el diagrama completo del datapath.

En general creo que trabajamos muy bien como equipo, ya que cada uno cumplió en tiempo y forma con sus responsabilidades.

Sobre el semestre y la materia creo que, aunque fue un poco corto el tiempo, aprendí muchísimo sobre la arquitectura de un procesador ya que yo venía en 0 acerca de estos temas.

Codificador Fase 1:Luis Angel Rosado Estrada

El proyecto sí fue difícil ya que esta actividad requería mucho tiempo, y en esta semana con los cambios de fechas de salida se juntó con todos los otros proyectos, a pesar de que los tiempos no estuvieron a nuestro favor pienso que mi equipo se comportó a la altura , con la seriedad que correspondía en todo momento, ahora sobre la parte que me tocó en concreto la verdad tuve que realizar horas y horas de investigación desde cómo debería ser el diseño ya que un multiplexor cambiaba por completo todo el proyecto, también fue investigar todo acerca de pipeline, y como se debería inicializar el reloj(clk).

ahora este semestre fue muy agotador, yo pienso que por la modalidad, pero la materia me despertó cierto interés ya que saber cómo funcionan los ordenadores a detalle e incluso armando componentes aunque sean muy básicos, me gusta.

Encargado de Reporte Fase 1: Miguel Eduardo Espiritu Torres

Hablando del proyecto considero que la organización de mi equipo fue buena, el desarrollo de la práctica en algunas secciones fue algo confusa debido a la ambigüedad de los requerimientos y la información dada en los archivos PDF, puesto que en el archivo de la fase 1 que debería manejar instrucciones R se requieren instrucciones de otro tipo y considero que requeria mas tiempo para esclarecer dudas en el proceso de la práctica, pero dando por entendido la cuestión del tiempo limitado para esta, considero que fue bien desarrollada por el equipo en medida de nuestro entendimiento.

En cuanto al semestre considero que fue bien desarrollado aunque los alumnos avanzan a ritmos diferentes, realmente fue algo complicado apoyar a compañeros y apoyate de compañeros, considero que parte del problema fue entender el HDL como un lenguaje de programación más que como un código que describe algo físico, algo complicado del semestre fue encontrar fuentes de información apegadas a los requerimientos del profesor.

Referencias

Patterson, D. A., Hennessy, J. L., & Bruguera, D. J. (2018). *Estructura y diseño de computadores* (2.ª ed.). Reverte.

https://fiberhub.tk/EC/Estructura%20y%20Dise%C3%B1o%20de%20Computadore

s%20-%20La%20Interfaz%20Hardware-Software%20%284a%20Edici%C3%B3n%29.pdf

Universidad de Valladolid, Hernández Cerezo, A., Tejedor García, C., & Alonso Iglesias, R. (2010, octubre). *Arquitectura MIPS* (N.º 1). Universidad de Valladolid. https://www.infor.uva.es/~bastida/OC/TRABAJO1_MIPS.pdf

2.2.2 Arquitectura CISC RISC y X86 - fhwjesus2. (s. f.). Arquitectura CISC RISC. Recuperado 8 de octubre de 2020, de https://sites.google.com/site/fhwjesus2/home/procesador-cpu/componentes-cpu/2-2arquitectura-cisc-risc