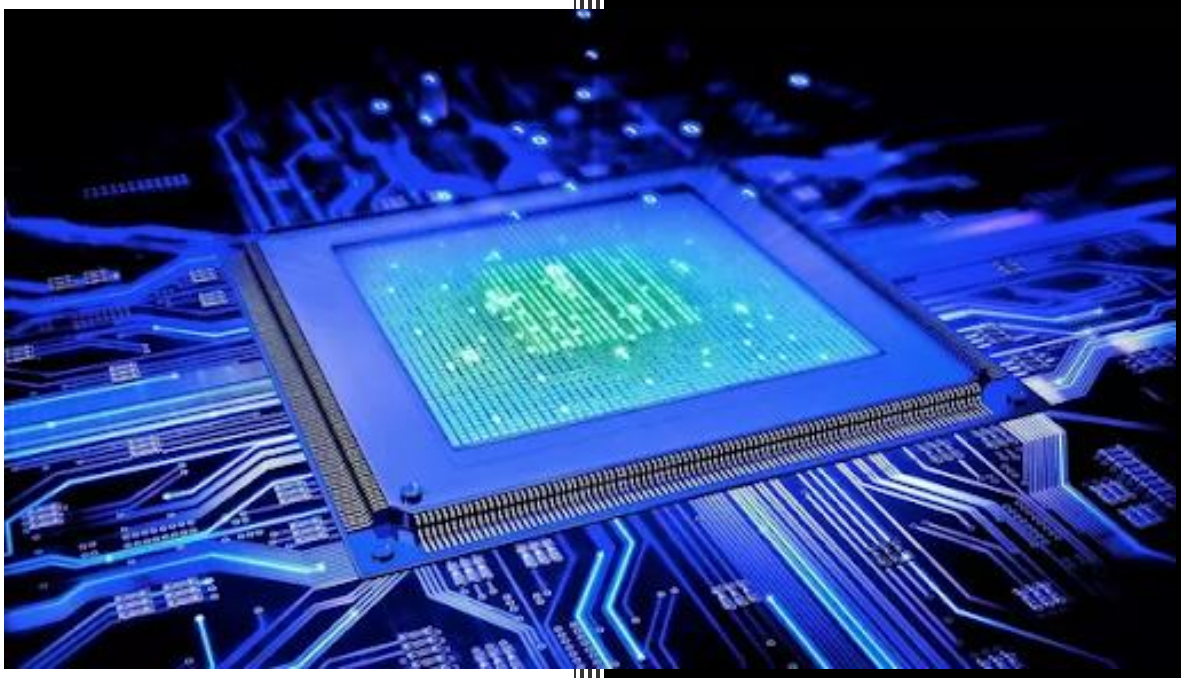
2020

Microprocesador



Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa de Ingeniería Robótica y Mecatrónica

Materia: Sistemas Digitales lll

Docente: Dr. Remberto Sandoval Aréchiga

Alumnos:

José Alfredo Hernández Dueñas

Jesús Francisco Villaseñor Correa

Equipo “5”

“7-B”

30 de Noviembre del 2020

Contenido

Resumen

En el presente documento presentamos la realización de nuestro proyecto que tiene como objetivo diseñar un microprocesador, del tipo RISC utilizando una arquitectura Harvard, para posteriormente realizar operaciones aritméticas básicas como lo es la suma, la resta, la multiplicación.

Para estructurar el microcontrolador se divide el proyecto en varias secciones, comenzando por el análisis del diagrama de caja negra presentado y del set de instrucciones, para posteriormente determinar los elementos que se necesitan para diseñar los bloques internos que conformaran la caja blanca o estructura interna del microprocesador, estos bloques internos tendrán la capacidad de seleccionar los datos recibidos y operar de forma que guarde esos datos y entregue un resultado.

Introducción

El microprocesador surgió de la evolución de distintas tecnologías predecesoras, básicamente de la computación y de la tecnología de semiconductores. El inicio de esta última data de mitad de la década de 1950; estas tecnologías se fusionaron a principios de los años 1970, produciendo el primer microprocesador. Dichas tecnologías iniciaron su desarrollo a partir de la segunda guerra mundial; en este tiempo los científicos desarrollaron computadoras específicas para aplicaciones militares. En la posguerra, a mediados de la década de 1940, la computación digital emprendió un fuerte crecimiento también para propósitos científicos y civiles. La tecnología electrónica avanzó y los científicos hicieron grandes progresos en el diseño de componentes de estado sólido (semiconductores). En 1948 en los laboratorios Bell crearon el transistor.

Microprocesador

El microprocesador (o simplemente procesador) es el circuito integrado central más complejo de un sistema informático; a modo de ilustración, se le suele llamar por analogía el «cerebro» de un ordenador.1

Es el encargado de ejecutar los programas, desde el sistema operativo hasta las aplicaciones de usuario; solo ejecuta instrucciones programadas en lenguaje de bajo nivel, realizando operaciones aritméticas y lógicas simples, tales como sumar, restar, multiplicar, dividir, las lógicas binarias y accesos a memoria.

Puede contener una o más unidades centrales de procesamiento (CPU) constituidas, esencialmente, por registros, una unidad de control, una unidad aritmético lógica (ALU) y una unidad de cálculo en coma flotante (conocida antiguamente como «coprocesador matemático»).

Imagen 1. Intel 4004 (i4004)

RISC

En arquitectura computacional, RISC (del inglés Reduced Instruction Set Computer, en español Computador con conjunto de instrucciones reducido) es un tipo de diseño de CPU generalmente utilizado en microprocesadores o microcontroladores con las siguientes características fundamentales:

- Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.
- Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

Además, estos procesadores suelen disponer de muchos registros de propósito general.

RISC es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse. El tipo de procesador más comúnmente utilizado en equipos de escritorio, el x86, está basado en CISC en lugar de RISC, aunque las versiones más nuevas traducen instrucciones basadas en CISC x86 a instrucciones más simples basadas en RISC para uso interno antes de su ejecución.

La idea fue inspirada por el hecho de que muchas de las características que eran incluidas en los diseños tradicionales de CPU para aumentar la velocidad estaban siendo ignoradas por los programas que eran ejecutados en ellas. Además, la velocidad del procesador en relación con la memoria de la computadora que accedía era cada vez más alta. Esto conllevó la aparición de numerosas técnicas para reducir el procesamiento dentro del CPU, así como de reducir el número total de accesos a memoria.

Arquitectura tipo Harvard

La arquitectura de Harvard es una arquitectura de computadora con pistas de almacenamiento y de señal físicamente separadas para las instrucciones y para los datos. El término proviene de la computadora Harvard Mark I basada en relés, que almacenaba las instrucciones sobre cintas perforadas (de 24 bits de ancho) y los datos en interruptores electromecánicos. Estas primeras máquinas tenían almacenamiento de datos totalmente contenido dentro la unidad central de proceso, y no proporcionaban acceso al almacenamiento de instrucciones como datos. Los programas necesitaban ser cargados por un operador; el procesador no podría arrancar por sí mismo.

En la actualidad la mayoría de los procesadores implementan dichas vías de señales separadas por motivos de rendimiento, pero en realidad implementan una arquitectura Harvard modificada, para que puedan soportar tareas tales como la carga de un programa desde una unidad de disco como datos para su posterior ejecución.

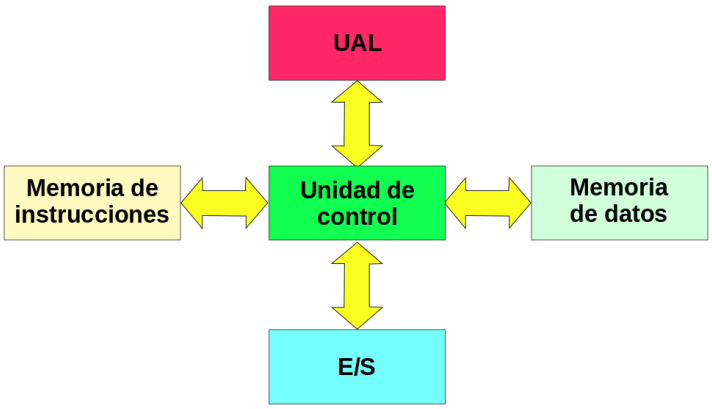


Imagen 2. Diagrama de bloques de una arquitectura Harvard

Requerimientos

Diagrama de caja negra

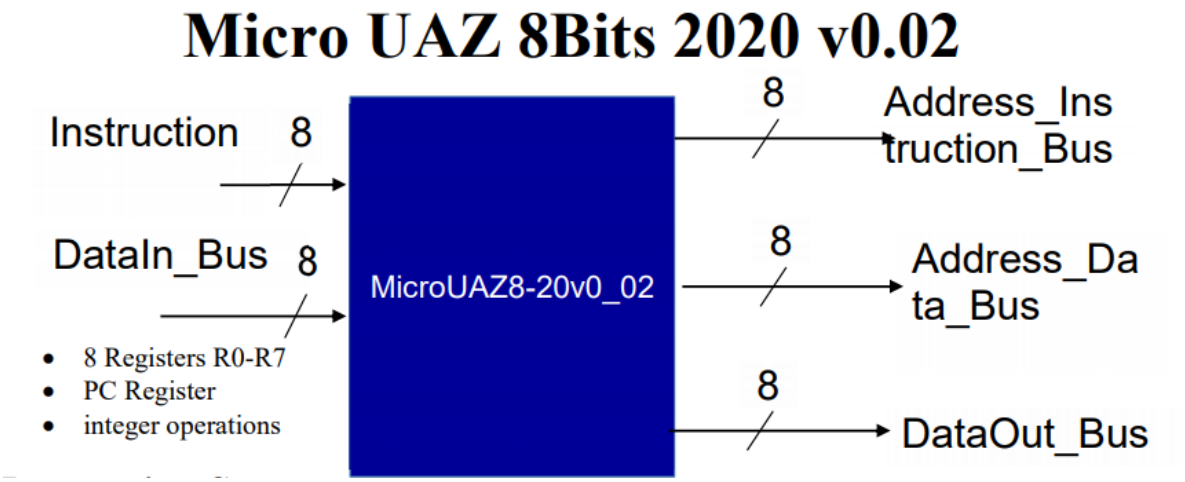


Imagen 3. Diagrama de caja negra del microprocesador

Nombre de la señal	Direccion	Tamaño	Descripcion
Instruction	Entrada	8	Señal de entrada que contiene la instrucción
DataIn_Bus	Entrada	8	Entrada de datos que viene directamente desde afuera del microprocesador.
Address_Instruction_Bus	Salida	8	Es la señal que tiene que realizar la dirección que se va a ejecutar
Address_Data_Bus	Salida	8	Dato de salida que indica la dirección de memoria
DataOut_Bus	Salida	8	Es el dato del registro Rx que contiene una dirección para modificar el PC

Tabla 1. Entradas y salidas del microprocesador

Set de instrucciones

Instruction	Arguments	Description	Comments
LOAD	RX,#NUM	Load #Num to register X	#Num is 3 bits [0,7]
LOAD	RX,[RY]	Load data at address [RY] from memory	RY and RX are 3 bits[0,7]
STORE	#NUM	Store #Num to [RX] address memory	#Num is 3 bits [0,7]
STORE	[RX],RY	Stores data at Register RY in [RX] memory address	RY and RX are 3 bits [0,7]
MOVE	RX,RY	Move data form register RY to RX	RY and RX are 3 bits [0,7]
MATH	RX,OP	DO MATH OPERATION WITH RX, AND STORES RESULT IN R0	OP: 0: R0=R0+RX 1: R0=R0-RX 2: R0= R0<<RX 3: R0= R0>>RY 4: R0=~RX 5: R0=R0&RX 6: R0 = R0|RX 7: R0=R0^RX
JUMP	[RX],COND	JUMP PC TO [RX] ADDRESS IF COND IS TRUE	COND: 0:NO CONDITION 1: NO CONDITION SAVE PC IN R7 2:Z FLAG IS TRUE 3:Z FLAG IS FALSE 4: C FLAG IS TRUE 5: C FLAG IS FALSE 6: N FLAG IS TRUE 7: N FLAG IS FALSE
NOP		NO OPERATION	

Tabla 2. Set de instrucciones con las que trabaja el microprocesador

Arquitectura

Microprocesador Equipo N

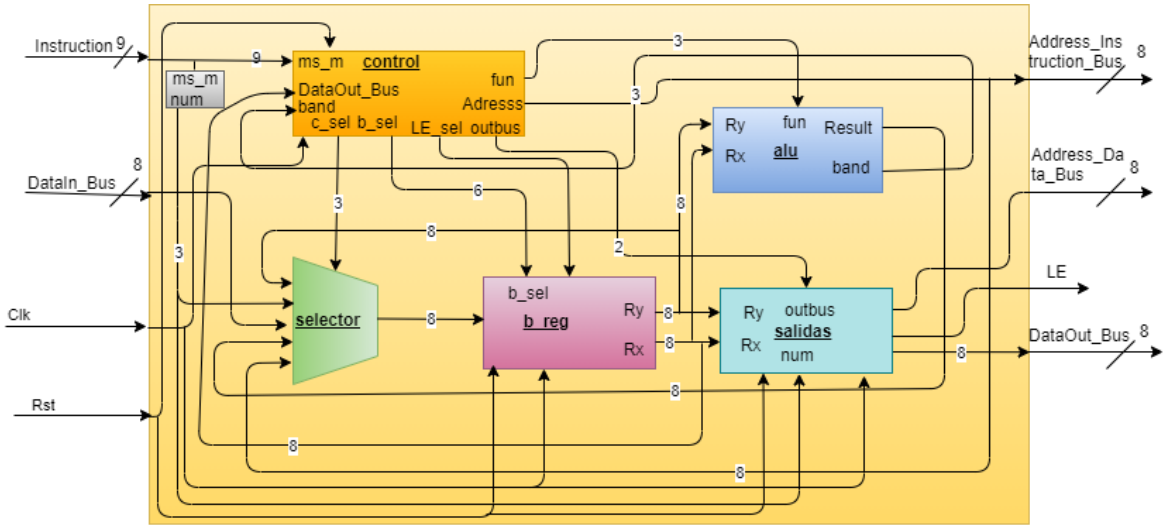


Imagen 3. Diagrama de caja blanca del microprocesador

Interiormente el procesador está compuesto por 6 elementos:

- ms_m
- control
- selector
- b_reg
- alu
- salidas

ms_m

Es el módulo el cual toma los 3 bits menos significativos de la entrada “instruction” para obtener un número dado por el valor del dato directo.

Control

El bloque de control es el encargado de direccionar y decodificar las señales recibidas por parte del módulo ms_m que previamente canalizo los bits de la señal de entrada “Instruction”, para posteriormente enviar el dato correcto seleccionado a cada módulo para que ejecute respectivamente su tarea asignada.

Selector

Modulo encargado de seleccionar el dato que pasara al banco de registros

b_reg

El banco de registros almacena datos de 8 bits para utilizarlos después permitiendo escribir y leer en ellos.

Alu

Es el módulo el cual realiza las operaciones lógicas con las entradas de los registros teniendo como salida el resultado de las operaciones y la activación de las banderas si el resultado es 0, hay un acarreo o es negativo

salidas

Este módulo ayuda a controlar las salidas de escritura, lectura dirección del bus y dirección de memoria del bus de salida

Bloque control

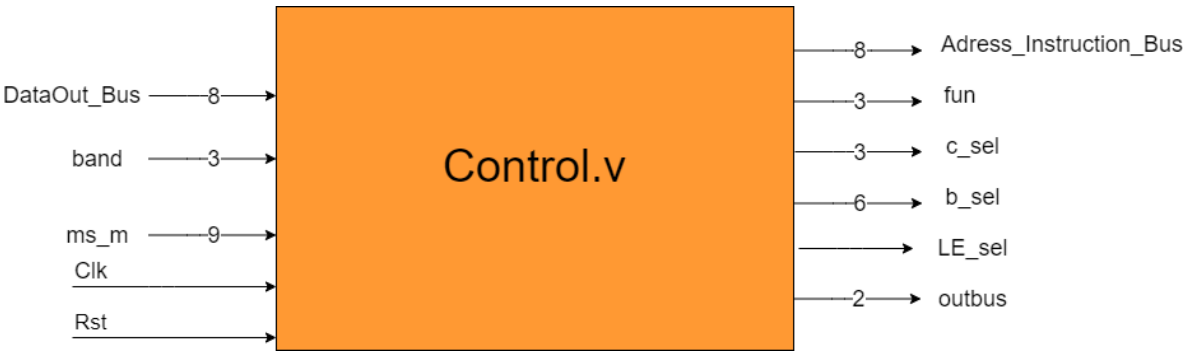


Imagen 4. Diagrama de caja negra de bloque control

Tabla del bloque control

Nombre de la señal	Dirección	Tamaño	Descripción
ms_m	Entrada	9	Es la señal de entrada que contiene la instrucción codificada
DataOut_Bus	Entrada	8	Es el dato del registro Rx que contiene una dirección para modificar el PC
band	Entrada	3	Es la bandera que va a generar la señal a la ALU dependiendo del resultado
c_sel	Salida	3	Es el dato binario que dependiendo de su valor seleccionara el dato que pasara al banco de registro
b_sel	Salida	6	Es el dato binario que selecciona los registros con los que se van a trabajar

LE_sel	Salida	1	Es el dato binario que determina si se leerá o se escribirá en el registro
fun	Salida	3	representa la selección de las operaciones a realizar las cuales son OP: 0=Suma, 1=Resta, 2=Corrimiento a la izquierda, 3=Corrimiento a la derecha, 4=Negación bit a bit, 5=AND bit a bit, 6=OR bit a bit, 7=XOR bit a bit
Adress	Salida	8	Es la señal que tiene que realizar la dirección que se va a ejecutar
outbus	Salida	2	Es la señal para seleccionar que operación hace el banco de registros para el bus de salida

Tabla 3. Entradas y salidas del bloque de control

	ms_m	b_sel	c_sel	LE_sel	fun	outbus	Addres_instruction_Bus
LOAD	001, Rx, NUM	000, RX	010	1	000	00	PC+1
LOAD	010,Rx,NUM	Ry,Ry	001	1	000	01	PC+1
STORE	011,Rx,NUM	000,Rx	000	0	000	10	PC+1
STORE	100,Rx,Ry	Ry,Rx	000	0	000	11	PC+1
MOVE	101,Rx,Ry	Ry,Rx	100	1	000	00	PC+1
MATH	110,Rx,OP	Result,Rx	000	1	OP	00	PC+1
JUMP	1111,Rx,COND	if COND =1 b_sel<=000,R7 else if COND =0 b_sel<= 000,Rx else b_sel<=000,Rx	if COND =1 c_sel<= 011 else c_sel<= 000	if COND=1 LE_sel<= 1 else LE_sel<= 0	000	00	COND = 0 -> i_Rx COND = 1 -> a) PC+1 b) i_Rx COND = 2 -> if Z=1 -> i_Rx else -> PC+1 COND = 3 -> if Z=0 -> i_Rx else -> PC+1 COND = 4 -> if C=1 -> i_Rx else -> PC+1 COND = 5 -> if C=0 -> i_Rx else -> PC+1 COND = 6 -> if N=1 -> i_Rx else -> PC+1 COND = 7 -> if N=0 -> i_Rx else -> PC+1
0	0	0	0	0	000	00	PC+1

Tabla 4.

El bloque de control es el encargado de direccionar y decodificar las señales recibidas por parte del módulo ms_m que previamente canalizo los bits de la señal de entrada “Instruction”, para posteriormente enviar el dato correcto seleccionado a cada módulo para que ejecute respectivamente su tarea asignada.

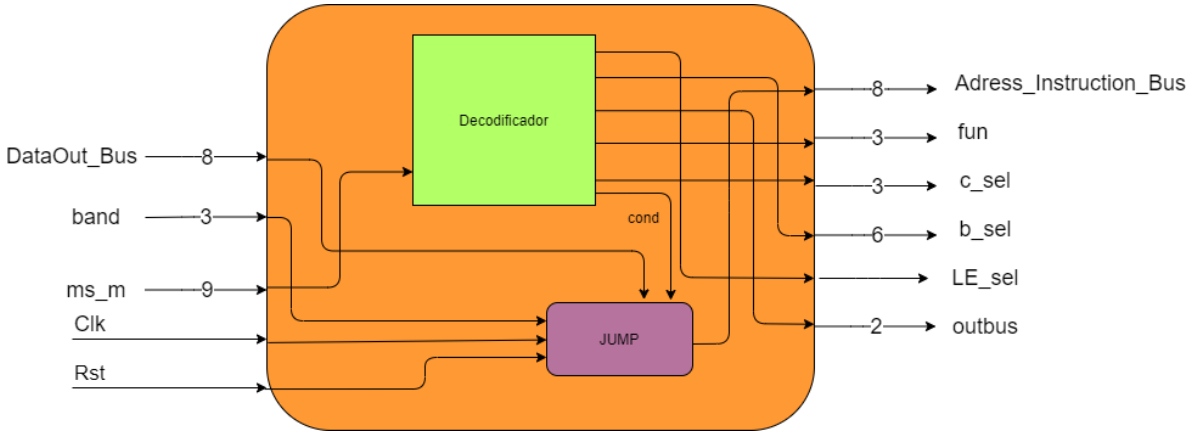


Imagen 5. Diagrama de caja blanca de bloque control

	ms_m	b_sel	c_sel	LE_sel	fun	outbus	cond
LOAD	000, Rx, NUM	0	000, Rx	1	00	010	0001
LOAD	001, Rx, Ry	0	Ry, Rx	1	01	001	0001
STORE	010, Rx, NUM	0	000, Rx	0	10	000	0001
STORE	011, Rx, Ry	0	Ry, Rx	0	11	000	0001
MOVE	100, Rx, Ry	0	Ry, Rx	1	00	100	0001
MATH	101, Rx, OP		R0, Rx	1	00	000	0001
JUMP	110, Rx, COND	0	If COND = 1 -> Rx, Ry else ->Rx, 000	If COND = 1 -> 1 else -> 0	00	If COND = 1 -> 001 else -> 000	{1’b1, COND}
NOP	111	0	0	0	00	000	0001

Tabla 5.

COND	cond	Address_instruction_Bus
000	1000	DataOut_Bus
001	1001	DataOut_Bus
010	1010	If Z=1 -> DataOut_Bus else -> PC+1
011	1011	If Z=0 -> DataOut_Bus else -> PC+1
100	1100	If C=1 -> DataOut_Bus else -> PC+1
101	1101	If C=0 -> DataOut_Bus

		else -> PC+1
110	1110	If N=1 -> DataOut_Bus else -> PC+1
111	1111	If N=0 -> DataOut_Bus else -> PC+1
No jump instruction	0001	PC+1

Tabla 6.

Entrada	Salida
ms_m [8:6] = 101	Rband = band
others	Rband= Rband

Tabla 7.

Bloque alu

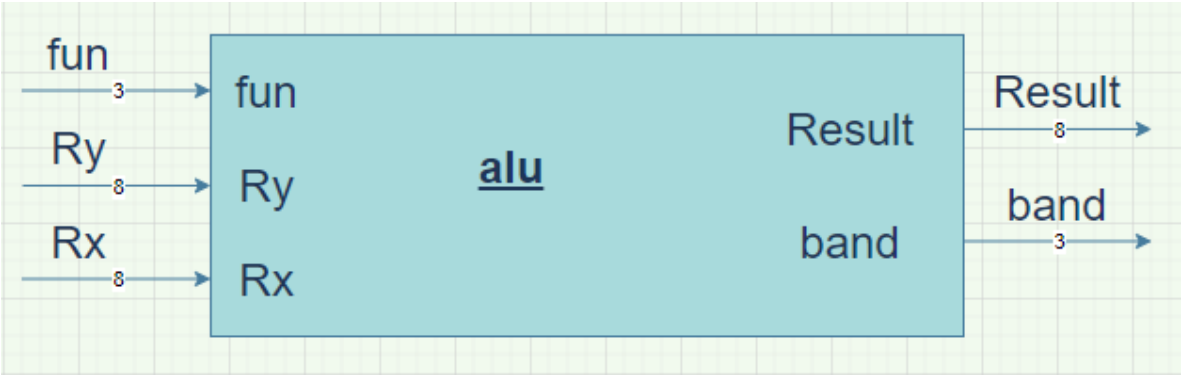


Imagen 6. Diagrama de caja negra de bloque alu

Tabla del bloque alu

Nombre de la señal	Dirección	Tamaño	Descripción
Ry	Entrada	8	Uno de los datos de 8 bits con el cual se llevarán las operaciones seleccionadas
Rx	Entrada	8	Uno de los dos datos de 8 bits con el cual se llevarán las operaciones seleccionadas
fun	Entrada	3	Representa la selección de las operaciones a realizar las cuales son OP: 0=Suma, 1=Resta, 2=Corrimiento a la izquierda,

			3=Corrimiento a la derecha, 4=Negación bit a bit, 5=AND bit a bit, 6=OR bit a bit, 7=XOR bit a bit
Result	Salida	8	Dato de salida el cual representa el resultado de la operación llevada a cabo
band	Salida	3	Dato de salida que representa las banderas activadas por la salida Result

Tabla 8. Entradas y salidas del bloque alu

Es el módulo el cual realiza las operaciones lógicas con las entradas de los registros teniendo como salida el resultado de las operaciones y la activación de las banderas si el resultado es 0, hay un acarreo o es negativo

Bloque b_reg

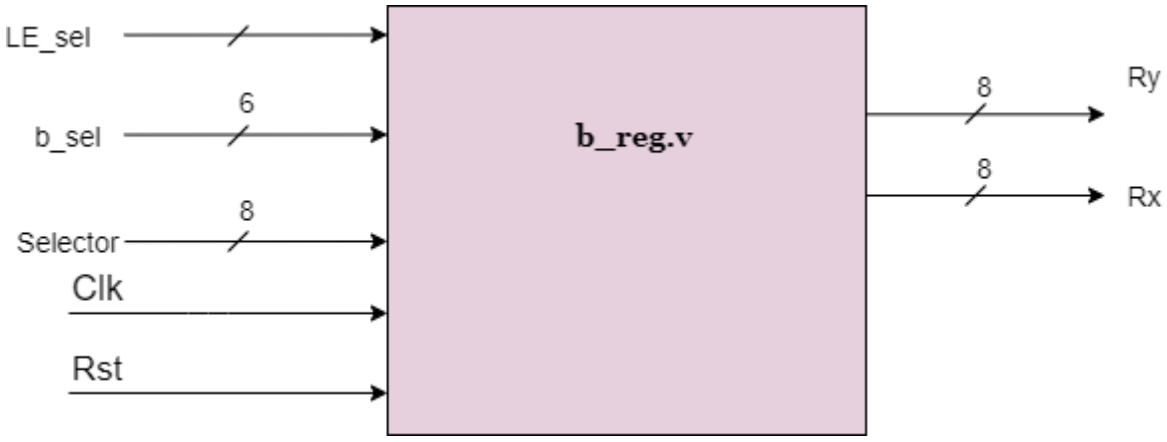


Imagen 7. Diagrama de caja negra de bloque b_reg

Tabla del bloque b_reg

Nombre de la señal	Dirección	Tamaño	Descripción
LE_sel	Entrada	1	Indica si va a leer o escribir en b_reg
b_sel	Entrada	6	Decide en el registro que se usara
selector	Entrada	8	Entrada del dato seleccionado
Ry	Salida	8	Registro seleccionado para Address_Data_Bus
Rx	Salida	8	Registro seleccionado para los datos salida del outbús

Tabla 9. Entradas y salidas del bloque b_reg

LE_sel	Rx	Ry
0	b_sel[2,0]	b_sel[5,3]
1	b_sel[2,0]	b_sel[5,3]

Tabla 10.

El banco de registros almacena datos de 8 bits para utilizarlos después permitiendo escribir y leer en ellos.

Bloque selector

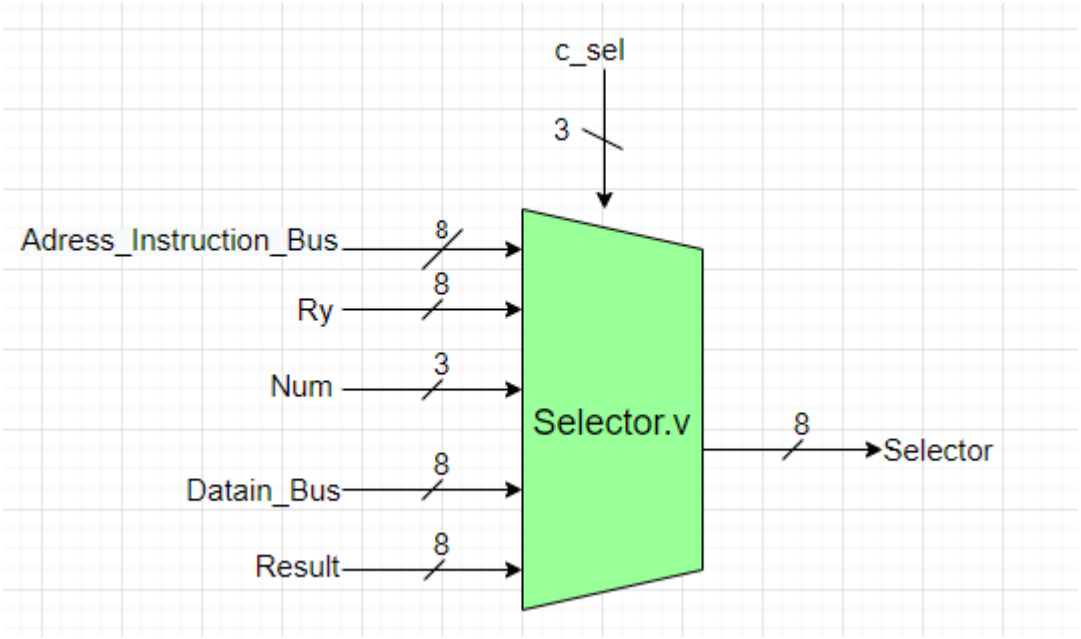


Imagen 8. Diagrama de caja negra de bloque Selector

Tabla del bloque Selector

Nombre de la señal	Dirección	Tamaño	Descripción
c_sel	Entrada	3	Entrada del selector que determinara que dato entrara al registro.
Ry	Entrada	8	Entrada en la cual tendremos el registro Ry.
Num	Entrada	3	Entrada la cual consiste en el valor del dato directo correspondiente a un número.
Datain_Bus	Entrada	8	Entrada de datos que viene directamente desde afuera del microprocesador.
Result	Entrada	8	Es el resultado de la operación desarrollada por el módulo ALU, el cual se guardará en el registro 0.
Address_Instruction_Bus	Entrada	8	Entrada del selector de la cual será guardada en el registro para poder tener un control.
Selector	Salida	8	Salida del selector la cual se dirige hacia el banco de registros para ser guardada.

Tabla 11. Entradas y salidas del bloque Selector

c_sel	Selector
000	Result
001	Datain_Bus
010	num
011	Address_Instruction_Bus
100	Ry
others	0

Tabla 12.

Modulo encargado de seleccionar el dato que pasara al banco de registros

Bloque ms_m

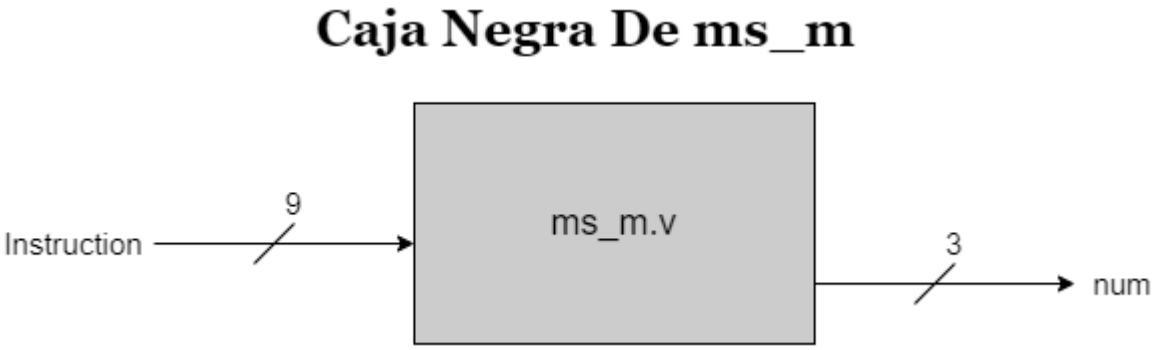


Imagen 9. Diagrama de caja negra de bloque ms_m

Nombre de la señal	Dirección	Tamaño	Descripción
intruction	Entrada	9	Señal de entrada que contiene la instrucción
num	Salida	3	Señal de salida que envía los 3 bits mas significativos de la instrucción inicial

Tabla 11. Entradas y salidas del bloque ms_m

Modulo el cual toma los 3 bits menos significativos de la entrada “instruction” para obtener un número dado por el valor del dato directo.

Bloque salidas

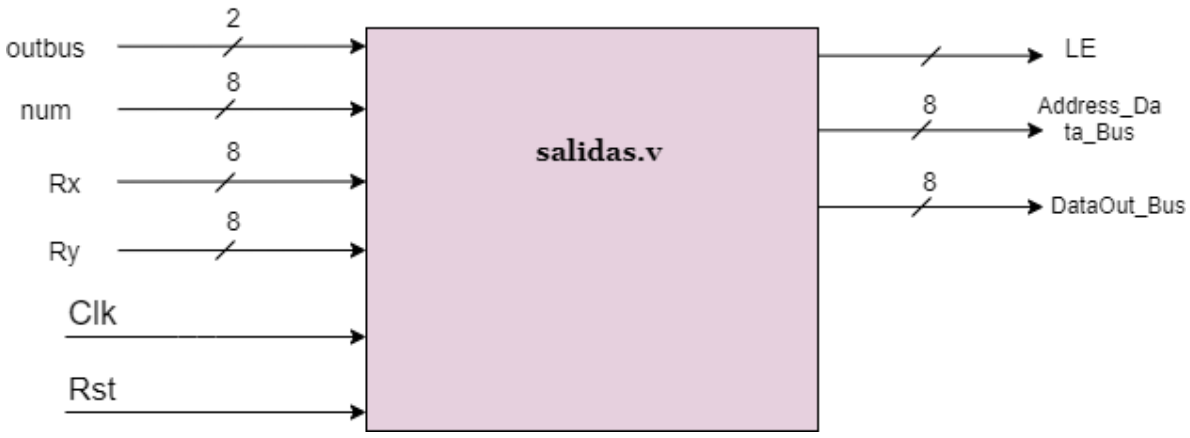


Imagen 10. Diagrama de caja negra de bloque salidas

Tabla del bloque salidas

Nombre de la señal	Dirección	Tamaño	Descripción
Rx	Entrada	8	Indica si va a leer o escribir en b_reg
Ry	Entrada	8	Decide en cual dato entra el registro
num	Entrada	8	Entrada del dato seleccionado para el dato inmediato
outbus	Entrada	2	Entrada del dato seleccionado para el bus de salida
Address_Data_Bus	Salida	8	Dato de salida que indica la dirección de memoria
Data_Out_Bus	Salida	8	Dato de salida para cambiar de dirección
LE_sel	Salida	1	muestra lo que se lee o escribió

Tabla 12. Entradas y salidas del bloque salidas

Instruction	outbus	DataOut_Bus	Address_Data_Bus	LE_sel
NOP	00	0	0	0
LOAD	01	0	Ry	0
STORE #NUM en la dirección [Rx]	10	NUM	Rx	1
STORE dato Ry en la dirección [Rx]	11	Ry	Rx	1

Tabla 13.

Este módulo ayuda a controlar las salidas de escritura, lectura dirección del bus y dirección de memoria del bus de salida

Pruebas

ms_m

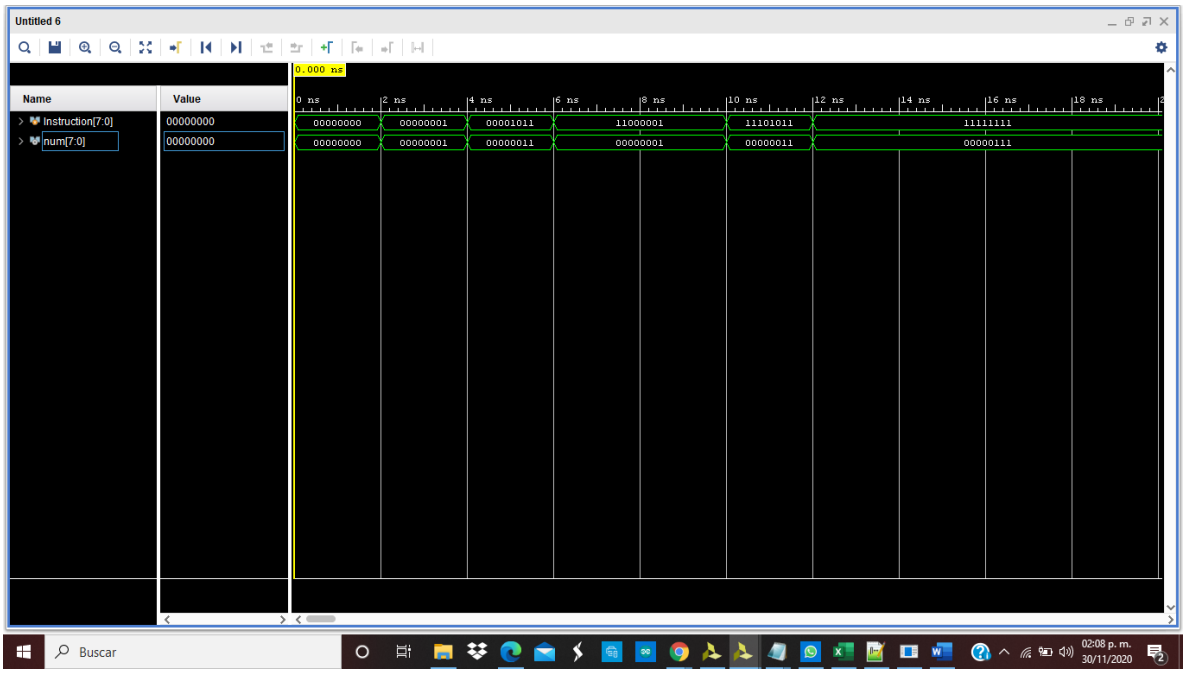


Imagen 11. Simulación del bloque ms_m

En la imagen 11 se puede ver como la salida son los tres bits menos significativos de la señal de entrada.

Selector

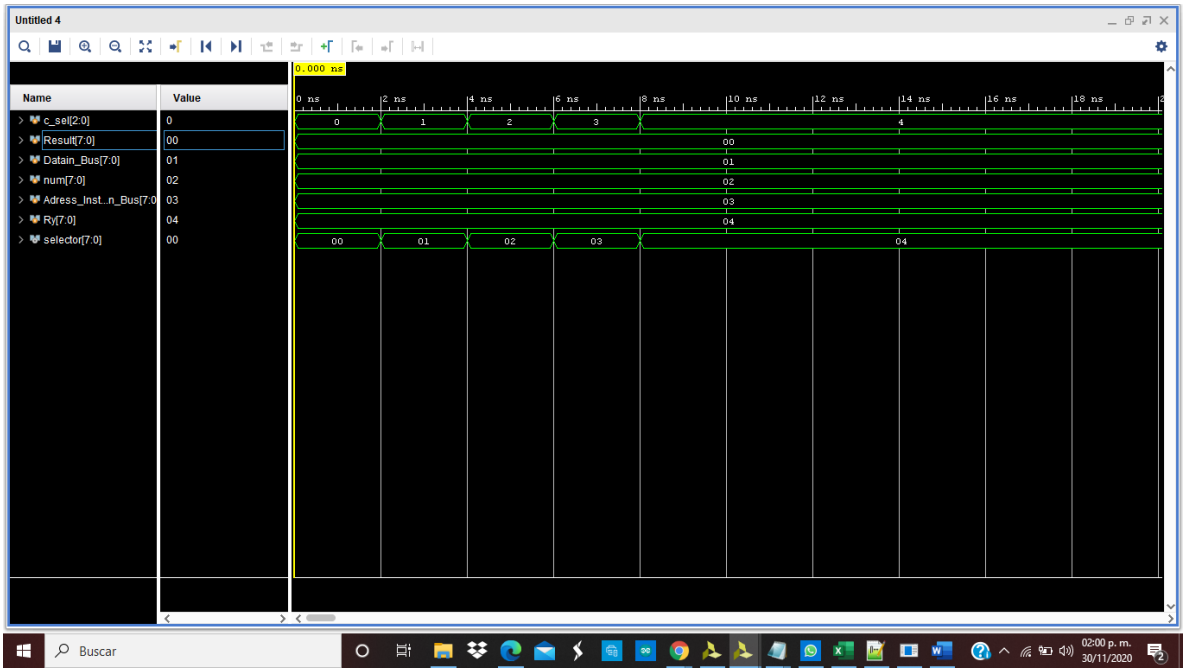


Imagen 12. Simulación del bloque selector

En esta simulación se puede ver que dependiendo del valor de la entrada selector la salida van tomando el valor de las diferentes entradas justo como se describen en la tabla 11 correspondiente a este modulo

Alu

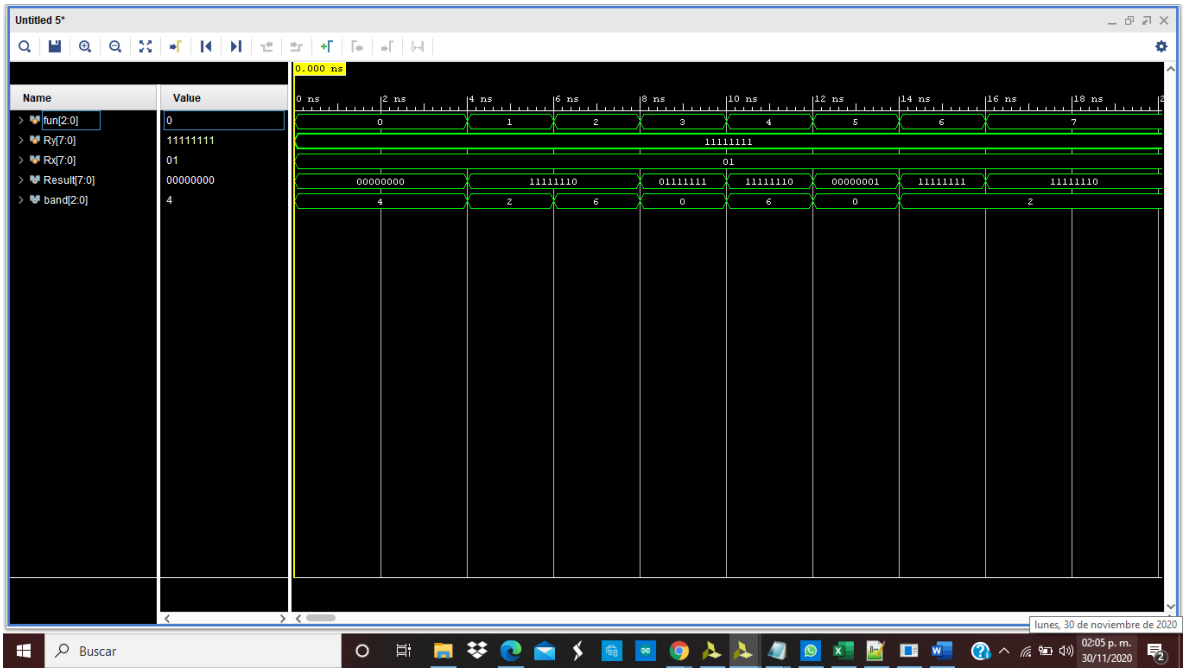


Imagen 13. Simulación del bloque al

En la imagen trece se puede ver como al darle un valor a la entrada fun determinara una operación entre Rx y Ry dando el resultado en la salida result y activando la bandera correspondiente según sea el caso como se explica en la tabla 8 correspondiente a esta simulación

b_reg

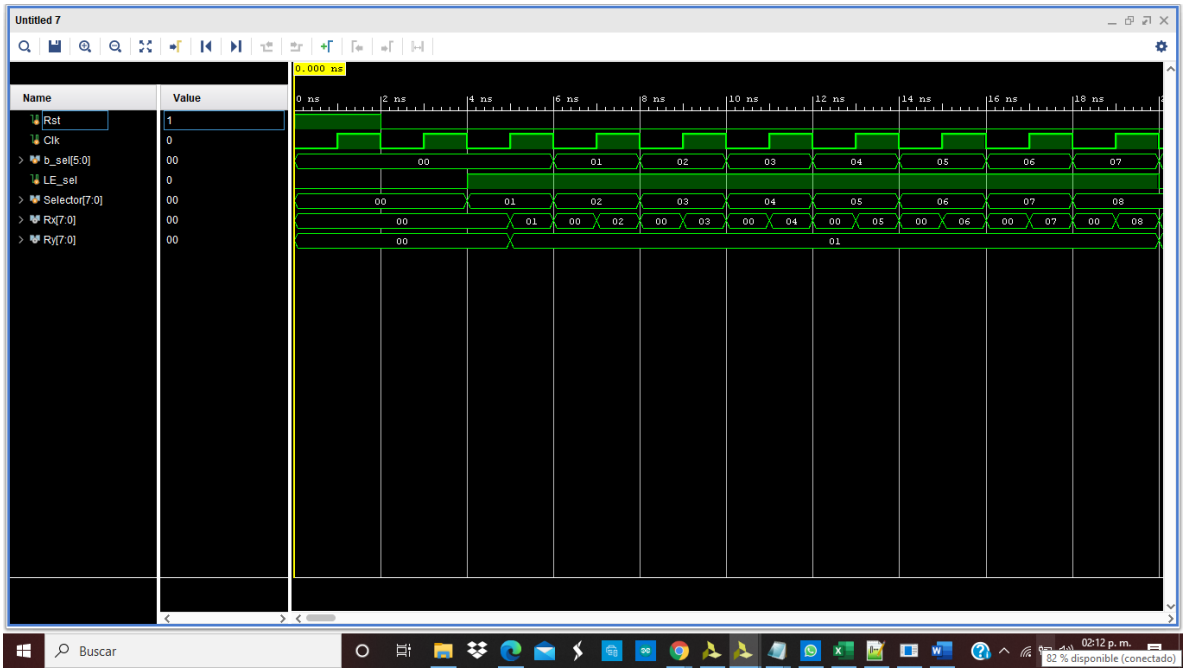


Imagen 14. Simulación del bloque b_reg

Para la simulación de la pagina once se puede ver el comportamiento de las salidas correspondientes a lo explicado en la tabla 9 la cual describe lo mostrado en esta simulación de la imagen 14.

Salidas

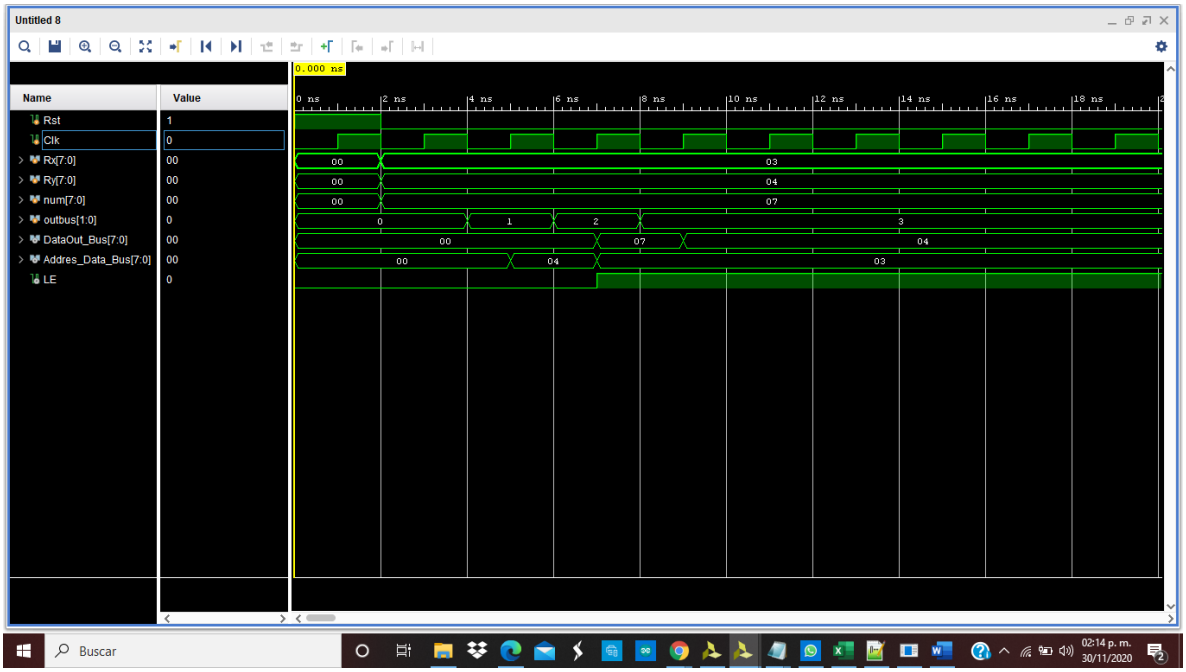


Imagen 15. Simulación del bloque salidas

Para la simulación de la imagen 15 se muestra el comportamiento descrito en la tabla 11 correspondiente a esta simulación.

Jump

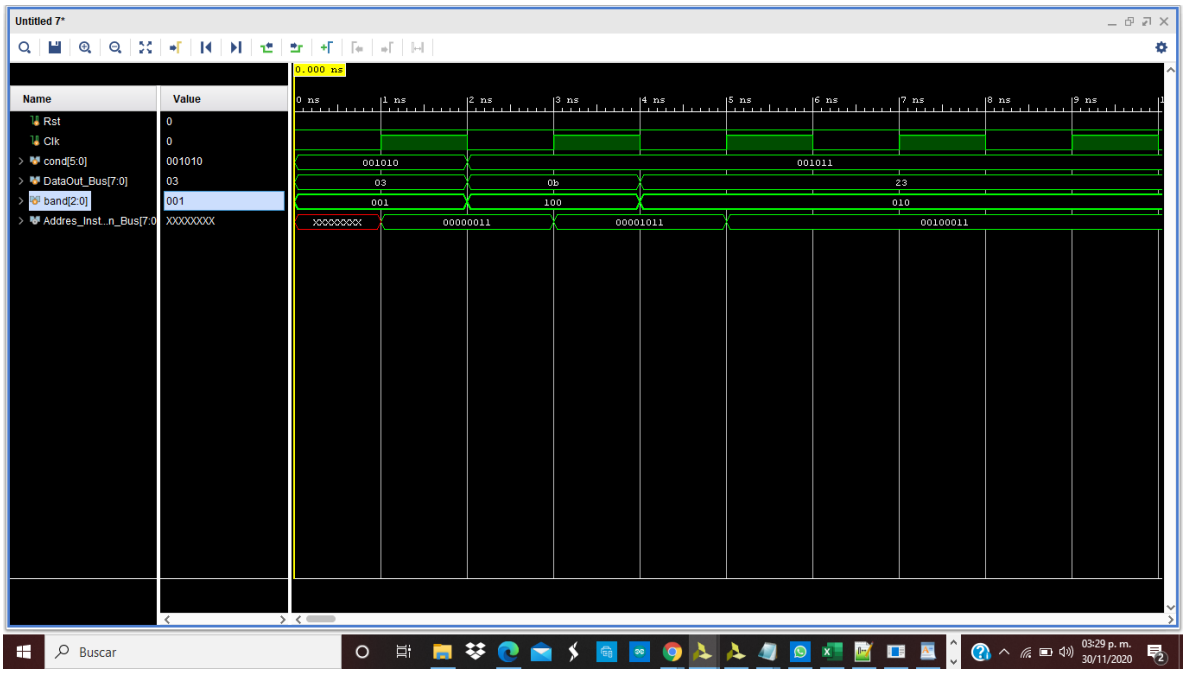


Imagen 16. Simulación del JUMP

Para la simulación de la imagen 16 muestra la función jump de la tabla 4 perteneciente al módulo de control

Deco

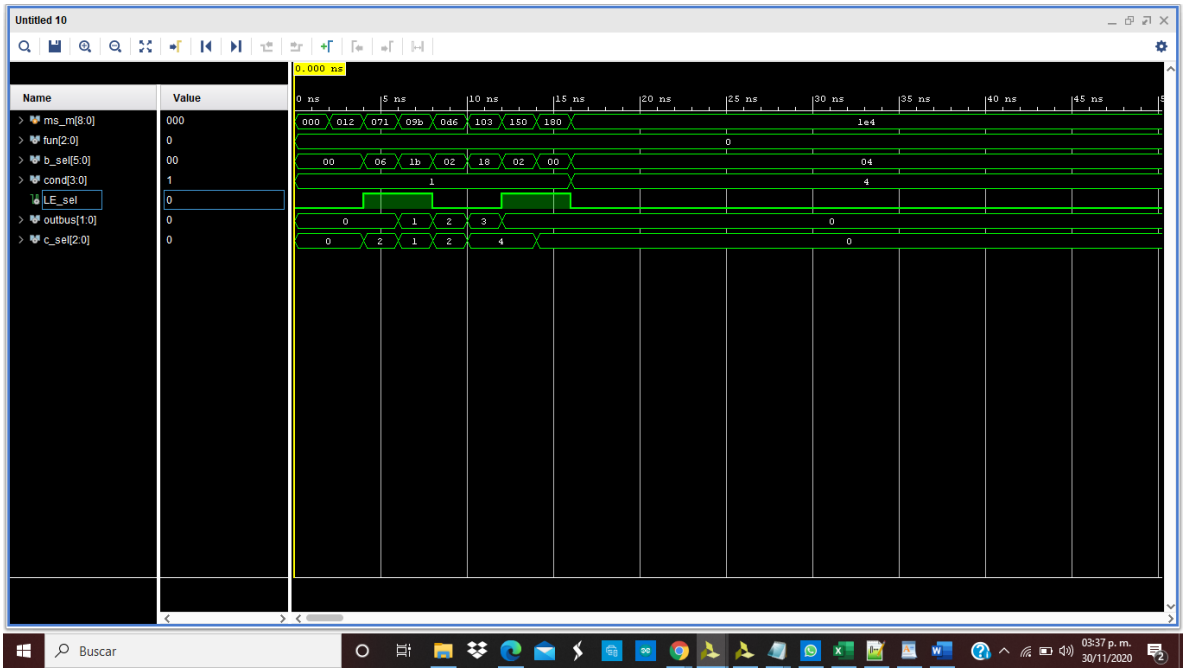


Imagen 17. Simulación del decodificador

En la simulación de la imagen 17 se muestra la simulación de las funciones restantes de la tabla 4 perteneciente al módulo de control

Control

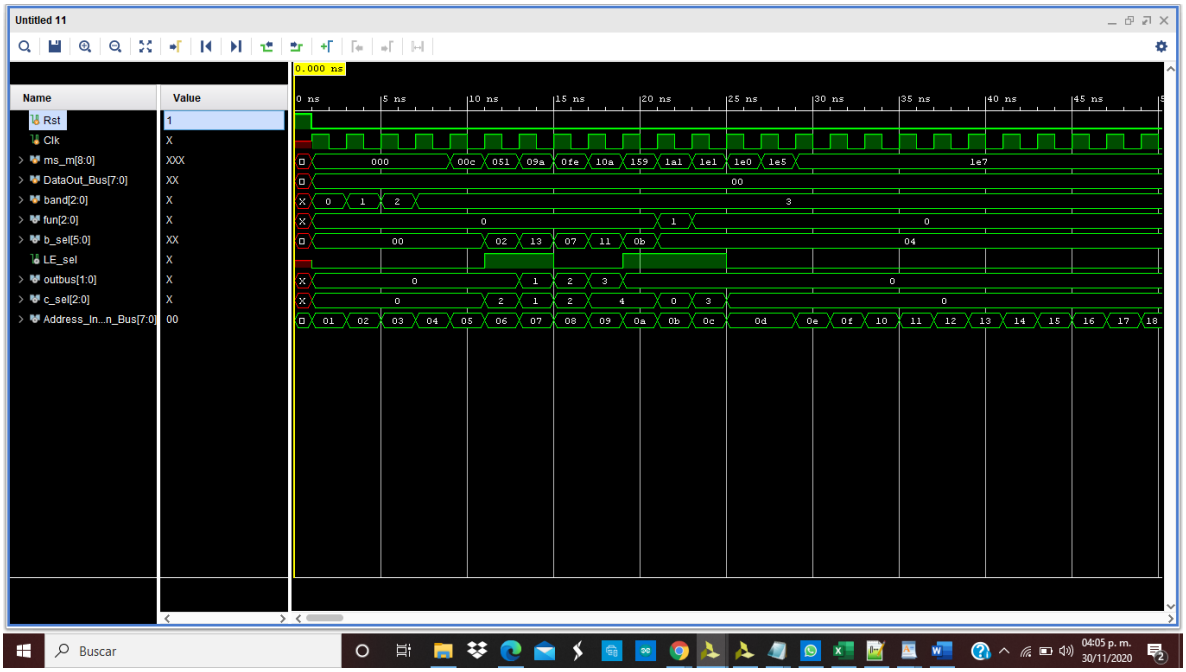


Imagen 18. Simulación del bloque control

En la simulación de la tabla de la imagen 18 se muestra el comportamiento completo de las funciones explicadas en la tabla 4

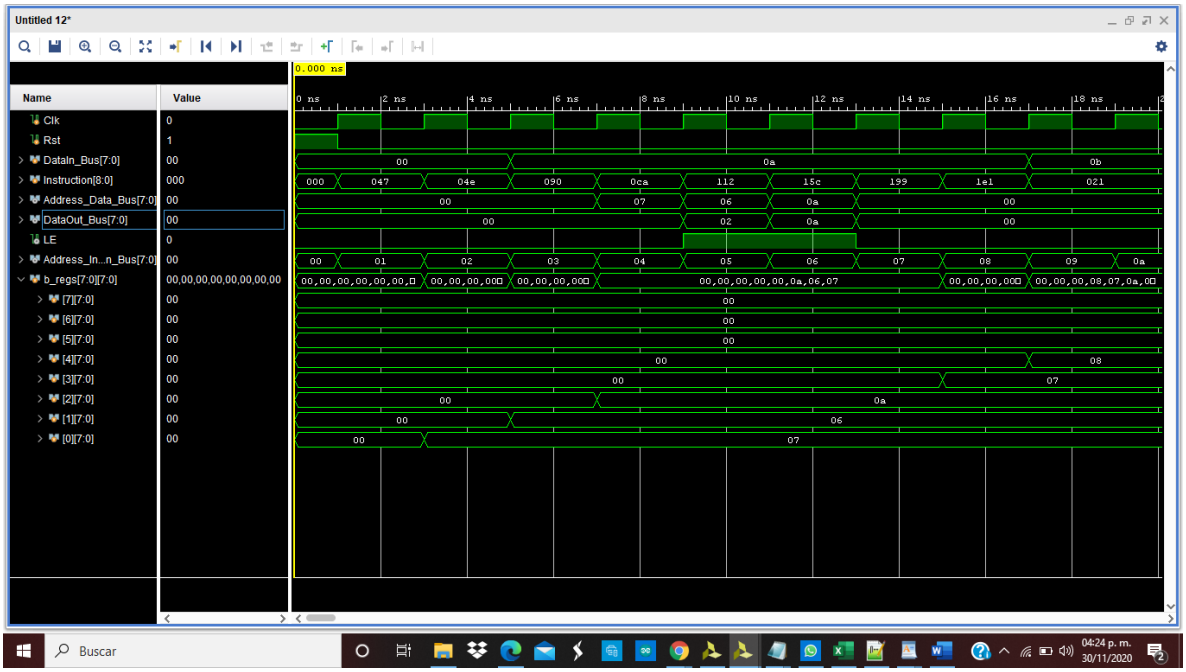


Imagen 19. Simulación del microprocesador

En la simulación de la imagen 19 se muestra el funcionamiento completo del microprocesador las cuales las describe la tabla 1 y 2

Micromemorias

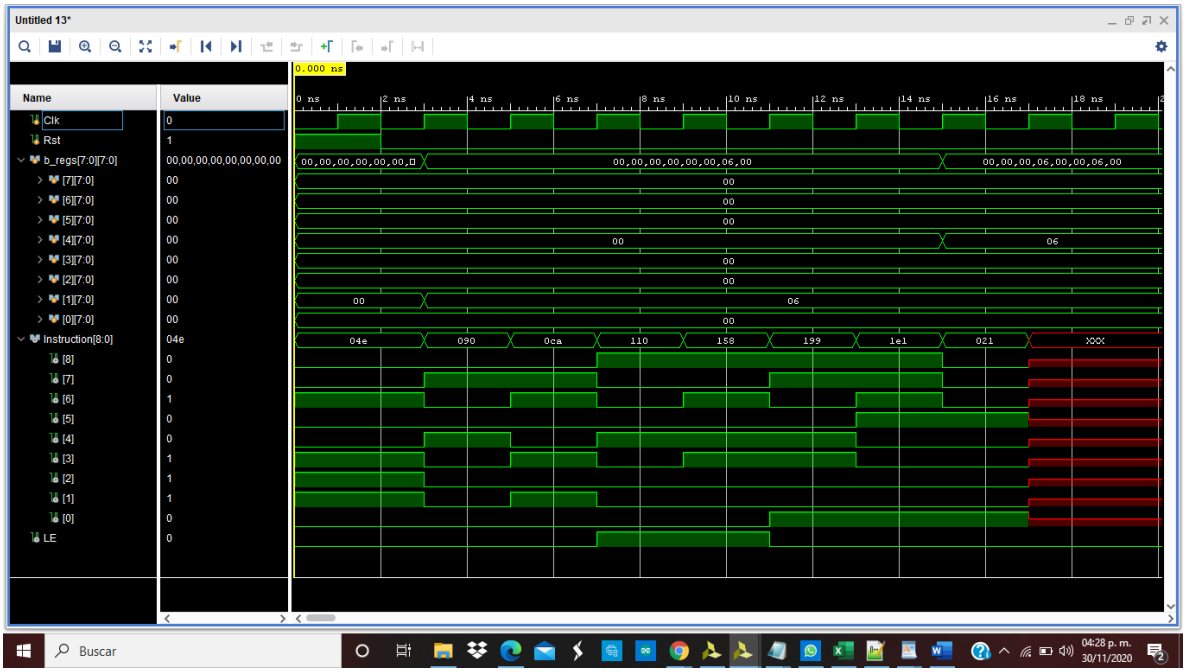
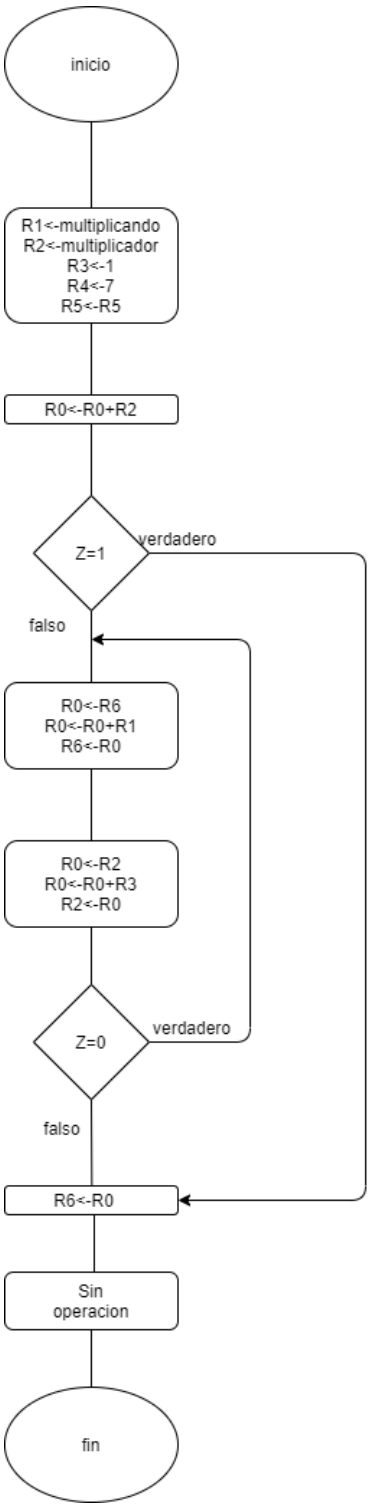


Imagen 20. Simulación de las memorias

Para la simulación de la imagen 11 se muestra el funcionamiento del micro procesador con los datos proporcionado por las memorias implementadas

Programas para la multiplicación y división

Multiplicación



1	Load R1,multiplicando	000001011
2	Load R2,multiplicador	000010001
3	Load R3,1	000011001
4	Load R4,7	000100111
5	Load R5,R5	001101101
6	Math R2,0	101010000
7	jump R5,2	110101010
8	Move R0,R6	100000110
9	Math R1,0	101001000
10	Move R6,R0	100110000
11	Move R0,R2	100000010
12	Math R3,1	101011001
13	Move R2,R0	100010000
14	Jump R4,3	110100010
15	Move R0,R6	100000110
16	No operation	111111111

Tabla 14.

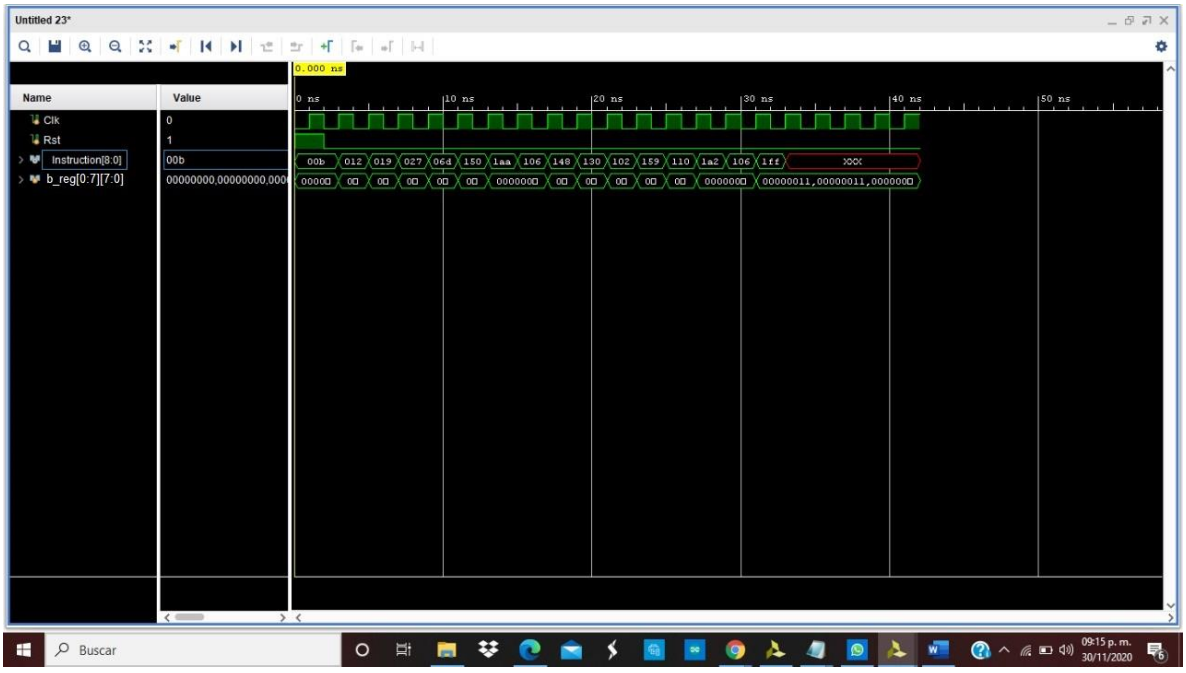
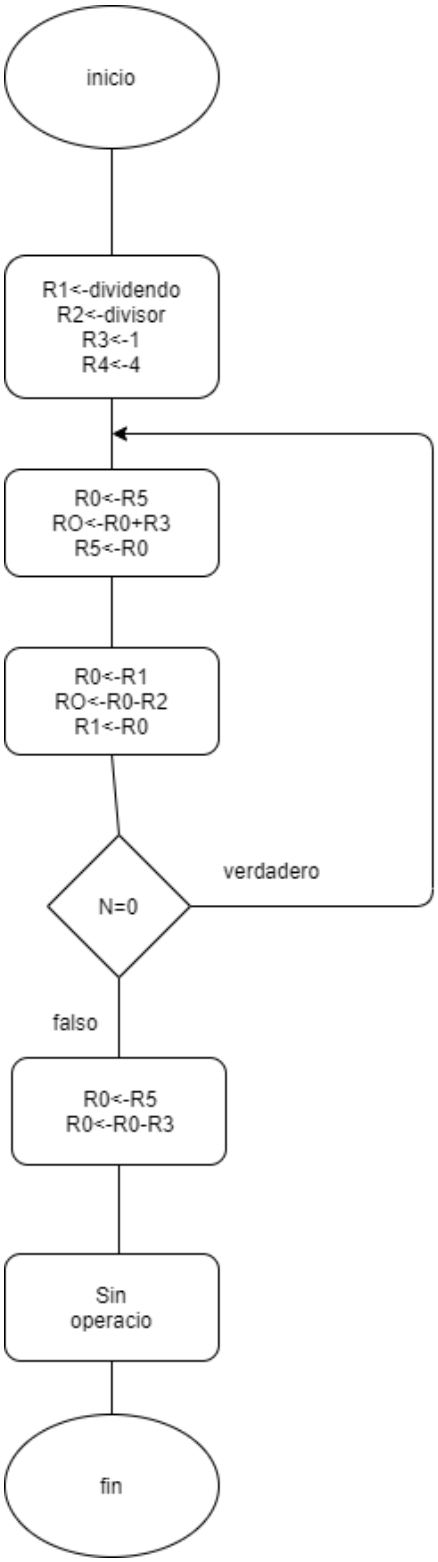


Imagen 21 simulación del programa de multiplicación

En la imagen 21 se puede ver como al multiplicar 3*1 el resultado es 3 como se puede notar en el registro R0 con un valor binario de 3.

División


inicio
R1<-dividendo
R2<-divisor
R3<-1
R4<-4
R0<-R5
RO<-R0+R3
R5<-R0
R0<-R1
RO<-R0-R2
R1<-R0
N=0
verdadero
falso
R0<-R5
R0<-R0-R3
Sin
operacio
fin

1	Load R1,dividendo	1110
2	Load R2,divisor	10010
3	Load R3,1	11001
4	Load R4,4	100100
5	Move R0, R5	100000101
6	Math R3, 0	101011000
7	Math R5,R0	100101000
8	Move R0, R1	100000001
9	Math R2, 1	101010001
10	Move R1,R0	100001000
11	jump R4 ,7	110100111
12	Move R0,R5	100000101
13	Math R3, 1	101011001
14	sin operación	111111111

Tabla 15.

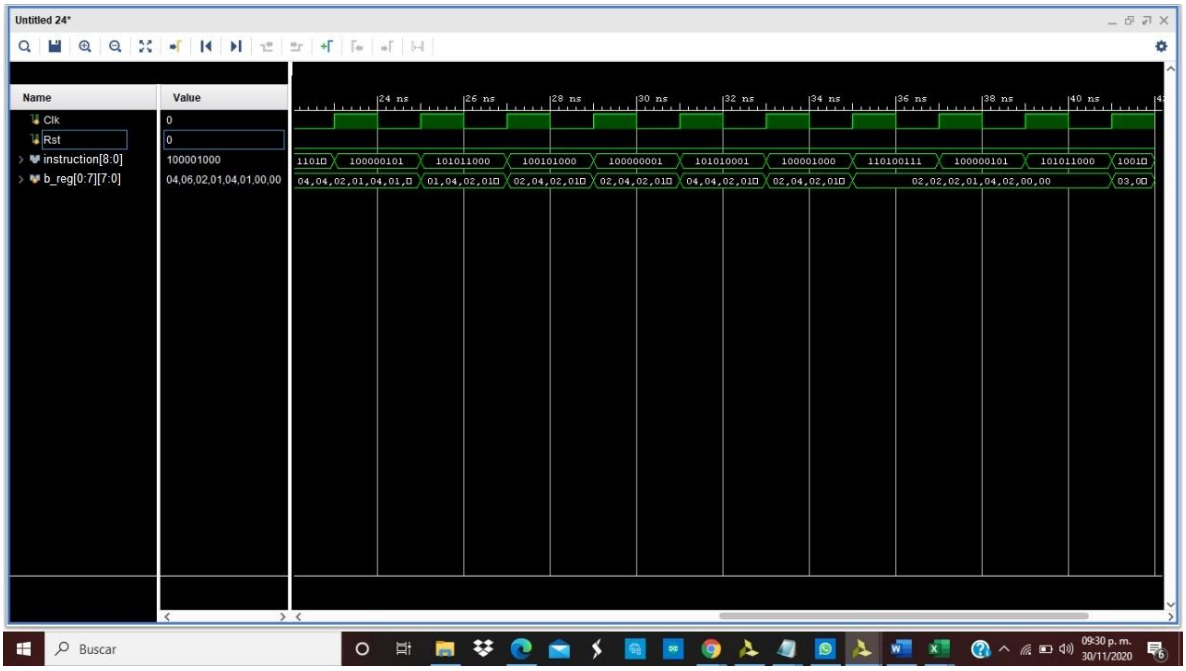


Imagen 22. simulación del algoritmo de división

En la imagen 22 se pude ver como al dividir 6 entre 2 se muestra en el registro R0 el resultado el cual es 3.

Análisis de resultados

El sistema funcionó satisfactoriamente respecto a lo especificado y trabajado como se pude ver en las simulaciones proporcionadas y al analizando las tablas de verdad para cada modulo con las cuales se puede comprobar el funcionamiento para cada módulo del proyecto tanto como el proyecto en general.

Conclusión

Este proyecto conllevo un gran reto para las involucrados siendo la primera vez por lo menos de manera personal en la que se lleva un proyecto de este tipo el cual trajo grandes retos tantos de aprendizaje, pero sin embargo a cambio se adquirió una gran experiencia en el campo y sobre todo el conocimiento necesario para sobrellevar retos similares si más que decir ante este proyecto y la materia se agradece la ayuda y el tiempo proporcionado a lo largo del curso.

Referencias

[1] M. Morris Mano, *Diseño Digital 3ra Edición*. California State University, Los Ángeles. Pearson Educación, 2003.

[2] Thomas L. Floyd, Fundamentos de Sistemas Digitales 9na Edición. Madrid. Pearson Educación, 2006

[3] Parra. (2019). Microprocesadores, RED TERCER MILENIO S.C.,

http://www.aliat.org.mx/BibliotecasDigitales/sistemas/Microprocesadores.pdf

[4] Francisco Pineda. (2015). Reduced instruction set computing, https://es.wikipedia.org/wiki/Reduced_instruction_set_computing

[5] Techopedia. (2019). Harvard Architecture, https://www.techopedia.com/

[6] List Differences (2019). Difference Between Von Neumann and Harvard Architecture?, https://www.listdifferences.com/

Apéndice

Para este proyecto hubo un reto importante el cual requirió de un gran esfuerzo el cual si bien tubo su complejidad lo más complejo fue el hecho de trabajar en conjunto con varios equipos a la vez y a distancia con esto trayendo más de una vez algunas pequeñas discrepancias, de cómo llevar a cabo el proyecto, generalmente más a errores gramaticales como que algunos usaran mayúsculas y otros minúsculas algunos usaron guiones y otros no que aunque parezcan errores simples si no se corrigen a tiempo al momento de integrar los trabajos el trabajo de todos se puede volver algo tedioso el tener que corregirlos sobre todo al hacer las conexiones de este o al hacer la synthesis que generalmente es cuando tedas cuenta de ello, pero en fin como experiencia y aprendizaje este tipo de trabajos son excelentes justamente para tener la experiencia de trabajar con múltiples grupos de trabajos ,en lo personal realizamos nuestro mayor esfuerzo para este proyecto, aunque faltan cosas que mejorar y aprender más a la comunicación entre equipos que desde nuestro punto de vista fue lo que consideramos un estilo de trabajar nuevo y finalmente consideramos mejorar para futuros proyectos.

Códigos

ms_m

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 25.10.2020 21:05:07
8     // Design Name:
9     // Module Name: ms_m
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module ms_m
25    #(parameter n = 8 )
26    (
27        input [n-1:0] Instruction,
28        output reg [n-1:0] num
29        );
30        reg [n-1:0]NC;
31     always@*
32            begin
33            NC = 8'b0;
34            num <= {NC[7:3],Instruction[2:0]};
35            end
36    endmodule
```

Selector

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 25.10.2020 19:55:02
8     // Design Name:
9     // Module Name: selector
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module selector(
25     input [2:0] c_sel,
26        input [7:0] Result,
27         input [7:0] Datain_Bus,
28         input [7:0] num,
29         input [7:0] Adress_Instruction_Bus,
30          input [7:0] Ry,
31        output reg [7:0] selector
32        );
33
34
35
36        always@*
37           begin
38           case (c_sel)
39           3'b000 :begin selector<=Result;end
40           3'b001 :begin  selector<=Datain_Bus;end
41           3'b010 :begin  selector<=num;end
42           3'b011 :begin  selector<=Adress_Instruction_Bus;end
43           3'b100 :begin  selector<=Ry;end
44           default: begin  selector<=0;end
45           endcase
46
47     end
48    endmodule
```

Alu

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 08.11.2020 16:31:36
8     // Design Name:
9     // Module Name: alu
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module alu(
25        input [2:0] fun,
26        input [7:0] Ry,
27        input [7:0] Rx,
28        output [7:0] Result,
29        output [2:0] band
30        );
31        reg [8:0] resultado;
32        always@*
33        begin
34        case(fun)
35        3'b000: begin resultado<=Ry+Rx; end
36        3'b001: begin resultado<=Ry-Rx; end
37        3'b010: begin resultado<=Ry<<Rx; end
38        3'b011: begin resultado<=Ry>>Rx; end
39        3'b100: begin resultado<=~Rx; end
40        3'b101: begin resultado<=Ry&Rx; end
41        3'b110: begin resultado<=Ry|Rx; end
42        3'b111: begin resultado<=Ry^Rx; end
43
44        endcase
45        end
46        assign Result=resultado[7:0];
47
48        assign band[0]=&(~resultado);
49        assign band[1]=resultado[7];
50        assign band[2]=resultado[8];
51    endmodule
```

b_reg

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 27.10.2020 20:53:37
8     // Design Name:
9     // Module Name: b_reg
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module b_reg(
25     input Rst,
26        input Clk,
27        input [5:0]b_sel,
28        input  LE_sel,
29        input [7:0] Selector,
30        output  [7:0] Rx,
31        output  [7:0] Ry
32        );
33
34        reg [7:0] b_regs [7:0];
35
36        always @(posedge Clk, posedge Rst) begin
37            if(Rst)
38                begin
39                    b_regs[0]<=0;
40                    b_regs[1]<=0;
41                    b_regs[2]<=0;
42                    b_regs[3]<=0;
43                    b_regs[4]<=0;
44                    b_regs[5]<=0;
45                    b_regs[6]<=0;
46                    b_regs[7]<=0;
47
48                end
49            else
50                if (LE_sel)
51
52                        b_regs[b_sel[2:0]]<=Selector;
```

```
53                      end
54
55
56                    assign Rx=b_regs[b_sel[2:0]];
57                    assign Ry=b_regs[b_sel[5:3]];
58
59    endmodule
```

Salidas

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 27.10.2020 23:07:22
8     // Design Name:
9     // Module Name: salidas
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module salidas(
25     input Rst,
26        input Clk,
27        input [7:0] Rx,
28        input [7:0] Ry,
29        input [7:0] num,
30        input [1:0] outbus,
31        output reg [7:0] DataOut_Bus,
32        output reg [7:0] Addres_Data_Bus,
33        output reg LE
34        );
35
36        always @(posedge Clk, posedge Rst) begin
37            if(Rst)
38                begin
39                    DataOut_Bus<=0;
40                    Addres_Data_Bus<=0;
41                    LE<=0;
42                end
43            else
44                case (outbus)
```

```
45                      2'b00:
46                          begin
47                              DataOut_Bus<=0;
48                              Addres_Data_Bus<=0;
49                              LE<=0;
50                          end
51                      2'b01:
52                          begin
53                              DataOut_Bus<=0;
54                              Addres_Data_Bus<=Ry;
55                              LE<=0;
56                          end
57                      2'b10:
58                          begin
59                              DataOut_Bus<=num;
60                              Addres_Data_Bus<=Rx;
61                              LE<=1;
62                          end
63                      2'b11:
64                          begin
65                              DataOut_Bus<=Ry;
66                              Addres_Data_Bus<=Rx;
67                              LE<=1;
68                          end
69                  endcase
70                  end
71
72      endmodule
```

Jump

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 10.11.2020 23:26:41
8       // Design Name:
9       // Module Name: jump
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
22
23
```

```
24    module jump(
25      input Rst,
26      input Clk,
27      input [3:0] cond,
28      input [7:0] DataOut_Bus,
29      input [2:0]band,
30      output [7:0] Address_Instruction_Bus
31        );
32
33        reg [7:0] PC;
34
35        always @(posedge Clk, posedge Rst) begin
36            if(Rst)
37                PC<=0;
38            else
39                case (cond)
40                    4'b0000:
41                        PC<=PC;
42                    4'b0001: // etapa de decodificacion de intruccion
43                        PC<=PC+1'b1;
44                    4'b1000:
45                        PC<=DataOut_Bus;
46                    4'b1001:
47                        PC<=DataOut_Bus;
48                    4'b1010:
49                        if(band[0])
50                            PC<=DataOut_Bus;
51                        else
52                            PC<=PC+1'b1;
53                    4'b1011:
54                        if(~band[0])
55                            PC<=DataOut_Bus;
56                        else
57                                PC<=PC+1'b1;
58                    4'b1100:
59                        if(band[2])
60                            PC<=DataOut_Bus;
61                        else
62                                PC<=PC+1'b1;
63                    4'b1101:
64                        if(~band[2])
65                            PC<=DataOut_Bus;
66                        else
67                                PC<=PC+1'b1;
68                    4'b1110:
69                        if(band[1])
70                            PC<=DataOut_Bus;
71                        else
72                                PC<=PC+1'b1;
73                    4'b1111:
74                        if(~band[1])
75                            PC<=DataOut_Bus;
76                        else
77                                PC<=PC+1'b1;
78                    default: PC<=PC+1'b1;
79                endcase
80        end
```

```
81          assign Address_Instruction_Bus=PC;
82
83      endmodule
```

Deco

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 10.11.2020 21:07:10
8       // Design Name:
9       // Module Name: deco
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
22
23
24      module deco(
25
26          input [8:0] ms_m,
27          output reg[2:0] fun,
28          output reg[5:0] b_sel,
29          output reg[3:0] cond,
30          output reg LE_sel,
31          output reg[1:0] outbus,
32          output reg[2:0] c_sel
33          );
34          always @(ms_m)
35          begin
36          case (ms_m [8:6])
37          3'b001: begin
38          fun<=0; b_sel<={3'b000,ms_m[5:3]}; LE_sel<=1; outbus<=2'b00;
c_sel<=3'b010; cond<=4'b0001; end
39          3'b010: begin
40          fun<=0; b_sel<={ms_m[2:0],ms_m[5:3]}; LE_sel<=1; outbus<=2'b01;
c_sel<=3'b001; cond<=4'b0001; end
41          3'b011: begin
42          fun<=0; b_sel<={3'b000,ms_m[5:3]}; LE_sel<=0; outbus<=2'b10;
c_sel<=3'b010; cond<=4'b0001; end
43          3'b100: begin
```

```
44          fun<=0; b_sel<={ms_m[2:0],ms_m[5:3]}; LE_sel<=0; outbus<=2'b11;
c_sel<=3'b100; cond<=4'b0001; end
45          3'b101: begin
46          fun<=0; b_sel<={ms_m[2:0],ms_m[5:3]}; LE_sel<=1; outbus<=2'b00;
c_sel<=3'b100; cond<=4'b0001; end
47          3'b110: begin
48          fun<=ms_m[2:0]; b_sel<={3'b000,ms_m[5:3]}; LE_sel<=1;
outbus<=2'b00; c_sel<=3'b000; cond<=4'b0001; end
49          3'b111: begin
50          if(ms_m[2:0]==3'b001)
51          begin
52          c_sel<=3'b011;
53          LE_sel<=1;fun<=0; b_sel<={3'b000,ms_m[5:3]};  outbus<=2'b00;
cond<={1'b0,ms_m[2:0]};
54          end
55          else
56          begin
57          c_sel<=3'd0;
58          LE_sel<=0;fun<=0; b_sel<={3'b000,ms_m[5:3]};  outbus<=2'b00;
cond<={1'b0,ms_m[2:0]};
59          end
60          end
61          3'b000: begin
62          fun<=0; b_sel<=0; LE_sel<=0; outbus<=2'b00; c_sel<=3'b000;
cond<=4'b0001; end
63          endcase
64          end
65      endmodule
```

Control

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 30.11.2020 15:42:38
8       // Design Name:
9       // Module Name: control
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
22
23
```

```
24  module control(
25
26
27
28      input [7:0] DataOut_Bus,
29      input [8:0] ms_m,
30      input [2:0] band,
31      output LE_sel,
32      output [5:0] b_sel,
33      output [1:0] outbus,
34      output [2:0] fun,
35      output [7:0] Address_Instruction_Bus,
36      output [2:0] c_sel,
37      input Clk,
38      input Rst
39      );
40
41      wire [3:0]cond; //Cable interno que une al Jump
42      reg [2:0] band2; //Regidtro para  las banderas
43
44      deco deco(
45          .ms_m(ms_m),
46          .cond(cond),
47          .LE_sel(LE_sel),
48          .b_sel(b_sel),
49          .outbus(outbus),
50          .fun(fun),
51          .c_sel(c_sel)
52      );
53
54      jump jump(
55          .band(band2),
56          .DataOut_Bus(DataOut_Bus),
57          .cond(cond),
58          .Clk(Clk),
59          .Rst(Rst),
60          .Address_Instruction_Bus(Address_Instruction_Bus)
61          );
62
63
64          always @(posedge Clk, posedge Rst) begin
65                  if(Rst)
66                      begin
67                          band2<=0;
68                      end
69                  else
70                      if (ms_m[8:6]==3'b101)
71                          band2<=band;
72              end
73  endmodule
```

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company: Universidad Autónoma de Zacatecas
4     // Engineer: Equipo N
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date:
8     // Design Name:
9     // Module Name:
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module Microprocesador_Equipo_N(
25        input [8:0] Instruction,
26        input [7:0] Datain_Bus,
27        output [7:0] Address_Instruction_Bus,
28        output [7:0] DataOut_Bus,
29        output [7:0] Addres_Data_Bus,
30        output LE,
31       input Clk,
32        input Rst
33        );
34
      //////////////////////////////////////////////////////////////////////////////////
35    wire [7:0] W_num;
36    wire [2:0] W_c_sel;
37    wire [5:0] W_b_sel;
38    wire [1:0] W_LE_sel;
39    wire [7:0] W_selector;
40    wire [7:0] W_RX;
41    wire [7:0] W_RY;
42    wire [2:0] W_fun;
43    wire [7:0] W_Result;
44    wire [2:0] W_band;
45    wire [2:0] W_outbus;
46
47
      //////////////////////////////////////////////////////////////////////////////////
48    control
```

```
49      controlador(
50          .ms_m(Instruction),
51          .DataOut_Bus(W_RX),
52          .band(W_band),
53          .Address_Instruction_Bus(Address_Instruction_Bus),
54          .outbus(W_outbus),
55          .b_sel(W_b_sel),
56          .LE_sel(W_LE_sel),
57          .c_sel(W_c_sel),
58          .Clk(Clk),
59          .Rst(Rst),
60          .fun( W_fun)
61          );
62
63
64
        ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
65      alu
66          operaciones(
67          .Rx(W_RX),
68          .Ry  (W_RY),
69          .fun(W_fun),
70          .Result(W_Result),
71          .band(W_band)
72          );
73
        ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
74      ms_m
75      numero(
76          .Instruction(Instruction),
77
78          .num(W_num)
79      );
80
        ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
81      b_reg
82      registros(
83          .b_sel(W_b_sel),
84          .LE_sel(W_LE_sel),
85          .Selector(W_selector),
86          .Rx(W_RX),
87          .Ry(W_RY),
88          .Clk(Clk),
89          .Rst(Rst)
90      );
91
        ////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
92      selector
93      sel(
94          .c_sel(W_c_sel),
95          .Result(W_Result),
96          .num(W_num),
97          .Datain_Bus(Datain_Bus),
```

```
98          .Ry(W_RY),
99          .Adress_Instruction_Bus(Address_Instruction_Bus),
100         .selector(W_selector)
101     );
102
        //////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
103     salidas
104     controlsalidas(
105         .outbus(W_outbus),
106         .Rx(W_RX),
107         .Ry(W_RY),
108         .num(W_num),
109         .DataOut_Bus(DataOut_Bus),
110         .Addres_Data_Bus(Addres_Data_Bus),
111         .LE(LE),
112         .Clk(Clk),
113         .Rst(Rst)
```

Memoria Ram

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////
////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 18.11.2020 09:50:29
8       // Design Name:
9       // Module Name: memoriaRam
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////
///////////////
22
23
24      module memoriaRam
25      #(parameter Ld=256, m=8)(
26          input LE,
27          input [m-1:0] address_data,
28          input [m-1:0] i_data,
29          output reg [m-1:0] o_data
30          );
31
32            reg [(m-1):0]mem[0:(Ld-1)];
33
```

```
34          initial
35              begin
36                  $readmemh("RAM.mem",mem);
37              end
38
39          always @(address_data,i_data,LE) begin
40              if(LE)
41                  begin
42                      mem[address_data]<=i_data;
43                      o_data<=mem[address_data];
44                  end
45              else
46                  o_data<=mem[address_data];
47          end
48      endmodule
```

Memoria Rom

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 18.11.2020 09:52:11
8       // Design Name:
9       // Module Name: memoriaROM
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
22
23
24      module memoriaROM
25      #(parameter Ld=256, m=8, n=9)(
26          input [m-1:0] address,
27          output reg [n-1:0] d_out
28          );
29           reg [(n-1):0]mem[0:(Ld-1)];
30
31            initial
32                begin
33                    $readmemb("ROM.mem",mem);
34                end
35
36            always @(address) begin
```

```
37              d_out<=mem[address];
38          end
39    endmodule
```

Micromemorias

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 18.11.2020 19:04:11
8     // Design Name:
9     // Module Name: Micropromemorias_Equipo_N
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module Micropromemorias_Equipo_N(
25     input Clk,
26        input Rst
27        );
28
29          wire [8:0] Instruction;
30          wire [7:0] DataIn_Bus;
31          wire [7:0] Address_Instruction_Bus;
32          wire [7:0] DataOut_Bus;
33          wire [7:0] Address_Data_Bus;
34          wire LE;
35
36         Microprocesador_Equipo_N  Micro(
37          .Clk(Clk),
38          .Rst(Rst),
39          .Instruction(Instruction),
40          .DataIn_Bus(DataIn_Bus),
41          .Address_Instruction_Bus(Address_Instruction_Bus),
42          .DataOut_Bus(DataOut_Bus),
43          .Address_Data_Bus(Address_Data_Bus),
44          .LE(LE)
45          );
46
```

```
47          memoriaROM #(256,8,9) InstructionMem(
48          .address(Address_Instruction_Bus),
49          .d_out(Instruction)
50          );
51
52          memoriaRam #(256,8,9) DataMem(
53          .LE(LE),
54          .i_data(DataOut_Bus),
55          .address_data(Address_Data_Bus),
56          .o_data(DataIn_Bus)
57          );
58
59      endmodule
```

Test benchs

Tb_ms_m

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 25.10.2020 21:18:38
8       // Design Name:
9       // Module Name: tb_ms_m
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
22
23
24      module tb_ms_m();
25      reg [7:0] Instruction;
26          wire [7:0] num;
27          wire [7:0] Instruction2;
28         ms_m uut(
29         .Instruction(Instruction),
30         .Instruction2(Instruction2),
31         .num(num)
32         );
33         initial
```

```
34        begin
35
36         Instruction = 0;
37
38
39         #2Instruction = 8'b00000001;
40
41
42         #2Instruction = 8'b00001011;
43
44
45         #2Instruction = 8'b11000001;
46
47
48         #2Instruction = 8'b11000001;
49
50
51         #2Instruction = 8'b11101011;
52
53
54         #2Instruction = 8'b11111111;
55        end
56
57    endmodule
```

Tb_selector

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 25.10.2020 20:09:40
8     // Design Name:
9     // Module Name: tb_selector
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module tb_selector( );
```

```
25    reg [2:0] c_sel;
26    reg [7:0] Result;
27    reg [7:0] Datain_Bus;
28    reg [7:0] num;
29    reg [7:0] Adress_Instruction_Bus;
30    reg [7:0] Ry;
31    wire [7:0] selector
32    ;
33
34    selector uut(
35    .c_sel(c_sel),
36    .Result(Result),
37    .Datain_Bus(Datain_Bus),
38    .num(num),
39    .Ry(Ry),
40    .Adress_Instruction_Bus(Adress_Instruction_Bus),
41    .selector(selector)
42    );
43    initial
44    begin
45    c_sel = 3'd0;
46    Result = 8'd0;
47    Datain_Bus = 8'd1;
48    num = 8'd2;
49    Ry =8'd4;
50    Adress_Instruction_Bus= 8'd3;
51
52
53    #2
54    c_sel = 3'd1;
55    #2
56    c_sel = 3'd2;
57    #2
58    c_sel = 3'd3;
59    #2
60    c_sel = 3'd4;
61    end
62    endmodule
```

Tb_alu

```
1     timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 08.11.2020 17:02:18
8     // Design Name:
9     // Module Name: tb_alu
10    // Project Name:
11    // Target Devices:
```

```
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
///////////////
22
23
24    module tb_alu;
25    reg [2:0] fun;
26        reg [7:0] Ry;
27        reg [7:0] Rx;
28        wire [7:0] Result;
29        wire [2:0] band;
30
31        alu uut(
32        .fun(fun),
33        .Ry(Ry),
34        .Rx(Rx),
35        .Result(Result),
36        .band(band)
37        );
38
39        initial
40        begin
41        Ry=8'b11111111;
42        Rx=8'b00000001;
43        fun=0;
44
45        #2 fun=3'b000;
46        #2 fun=3'b001;
47        #2 fun=3'b010;
48        #2 fun=3'b011;
49        #2 fun=3'b100;
50        #2 fun=3'b101;
51        #2 fun=3'b110;
52        #2 fun=3'b111;
53        end
54    endmodule
```

Tb_b_reg

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 27.10.2020 21:10:16
8     // Design Name:
9     // Module Name: tb_b_reg
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module tb_b_reg;
25    reg Rst;
26    reg Clk;
27    reg[5:0]b_sel;
28    reg LE_sel;
29    reg[7:0] Selector;
30    wire [7:0] Rx;
31    wire [7:0] Ry;
32
33    b_reg uut(
34    .Rst(Rst),
35    .Clk(Clk),
36    .b_sel(b_sel),
37    .LE_sel(LE_sel),
38    .Selector(Selector),
39    .Rx(Rx),
40    .Ry(Ry)
41    );
42    initial
43    begin
44    Rst=1;
45    Clk=0;
46    b_sel=0;
47    LE_sel=0;
48    Selector=0;
49    #2 Rst=0; b_sel=6'b000_000; LE_sel=0; LE_sel=8'b00000000;
50    #2 b_sel=6'b000000; LE_sel=1; Selector=8'b00000001;
```

```
51      #2 b_sel=6'b000001; LE_sel=1; Selector=8'b00000010;
52      #2 b_sel=6'b000010; LE_sel=1; Selector=8'b00000011;
53      #2 b_sel=6'b000011; LE_sel=1; Selector=8'b00000100;
54      #2 b_sel=6'b000100; LE_sel=1; Selector=8'b00000101;
55      #2 b_sel=6'b000101; LE_sel=1; Selector=8'b00000110;
56      #2 b_sel=6'b000110; LE_sel=1; Selector=8'b00000111;
57      #2 b_sel=6'b000111; LE_sel=1; Selector=8'b00001000;
58
59      #2 b_sel=6'b001000; LE_sel=0;
60      #2 b_sel=6'b011010; LE_sel=0;
61      #2 b_sel=6'b101100; LE_sel=0;
62      #2 b_sel=6'b111110; LE_sel=0;
63
64      end
65       always
66              #1 Clk = !Clk;
67      endmodule
```

Tb_salidas

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 27.10.2020 23:43:56
8       // Design Name:
9       // Module Name: tb_salidas
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
22
23
24      module tb_salidas;
25       reg Rst;
26          reg Clk;
27          reg [7:0] Rx;
28          reg [7:0] Ry;
29          reg [7:0] num;
30          reg [1:0] outbus;
31          wire [7:0] DataOut_Bus;
32          wire [7:0] Addres_Data_Bus;
```

```
33          wire LE;
34
35          salidas uut(
36          .Rst(Rst),
37          .Clk(Clk),
38          .Rx(Rx),
39          .Ry(Ry),
40          .num(num),
41          .outbus(outbus),
42          .DataOut_Bus(DataOut_Bus),
43          .Addres_Data_Bus(Addres_Data_Bus),
44          .LE(LE)
45          );
46
47          initial
48              begin
49              Rst=1;
50
51              Clk=0;
52              Rx=0;
53              Ry=0;
54              num=0;
55              outbus=0;
56
57              #2 Rst=0; Rx=8'd3; Ry=8'd4; num=8'd7;
58              outbus=0;
59              #2 outbus=2'b01;
60              #2 outbus=2'b10;
61              #2 outbus=2'b11;
62              end
63
64          always
65              #1 Clk = !Clk;
66      endmodule
```

Tb_jump

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 16.11.2020 15:35:33
8       // Design Name:
9       // Module Name: tb_jump
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
```

```
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module tb_jump( );
25    reg Rst;
26    reg Clk;
27    reg [5:0] cond;
28    reg [7:0] DataOut_Bus;
29    reg [2:0]band;
30    wire [7:0] Addres_Instruction_Bus;
31
32    jump uut(
33    .Rst(Rst),
34    .Clk(Clk),
35    .cond(cond),
36    .DataOut_Bus(DataOut_Bus),
37    .band(band),
38    .Addres_Instruction_Bus(Addres_Instruction_Bus)
39    );
40
41    initial
42    begin
43    band=3'b001;
44    DataOut_Bus=8'b00000011;
45    cond=4'b1010;
46    Clk=0;
47    Rst=1;
48    Rst=0;
49    #2 band=3'b100;
50    cond=4'b1011;
51    DataOut_Bus=8'b00001011;
52    #2 band=3'b010;
53    cond=4'b1011;
54    DataOut_Bus=8'b00100011;
55     end
56    always
57            #1 Clk = !Clk;
58    endmodule
```

Tb_deco

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
```

```
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 10.11.2020 22:50:51
8     // Design Name:
9     // Module Name: tb_deco
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module tb_deco();
25    reg [8:0] ms_m;
26        wire[2:0] fun;
27        wire[5:0] b_sel;
28        wire[3:0] cond;
29        wire LE_sel;
30        wire[1:0] outbus;
31        wire[2:0] c_sel;
32
33        deco uut(
34        .ms_m(ms_m),
35        .fun(fun),
36        .b_sel(b_sel),
37        .cond(cond),
38        .LE_sel(LE_sel),
39        .outbus(outbus),
40        .c_sel(c_sel)
41        );
42
43        initial
44        begin
45        ms_m=0;
46
47        #2ms_m=9'b000_010_010;
48        #2ms_m=9'b001_110_001;
49        #2ms_m=9'b010_011_011;
50        #2ms_m=9'b011_010_110;
51        #2ms_m=9'b100_000_011;
52        #2ms_m=9'b101_010_000;
53        #2ms_m=9'b110_000_000;
54        #2ms_m=9'b111_100_100;
55        end
56    endmodule
```

Tb_control

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 10.11.2020 20:29:40
8     // Design Name:
9     // Module Name: tb_control
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module tb_control();
25    reg Rst;
26        reg Clk;
27        reg [8:0] ms_m;
28        reg [7:0] DataOut_Bus;
29        reg [2:0] band;
30        wire [2:0] fun;
31        wire [5:0] b_sel;
32        wire LE_sel;
33        wire [1:0] outbus;
34        wire [2:0] c_sel;
35        wire [7:0] Address_Instruction_Bus
36    ;
37    control uut(
38    .Rst(Rst),
39    .Clk(Clk),
40    .ms_m(ms_m),
41    .DataOut_Bus(DataOut_Bus),
42    .band(band),
43    .fun(fun),
44    .b_sel(b_sel),
45    .LE_sel(LE_sel),
46    .outbus(outbus),
47    .c_sel(c_sel),
48    .Address_Instruction_Bus(Address_Instruction_Bus)
49    );
50    initial
```

```
51      begin
52       Rst=1;
53       #1 Rst=0;
54       Clk=0;
55       ms_m=0;
56       DataOut_Bus=0;
57       band=0;
58       #2  band=1;
59       #2  band=2;
60       #2  band=3;
61       #1 Rst=0;
62       #1  ms_m=9'b000_001_100;
63       #2  ms_m=9'b001_010_001;
64       #2  ms_m=9'b010_011_010;
65       #2  ms_m=9'b011_111_110;
66       #2  ms_m=9'b100_001_010;
67       #2  ms_m=9'b101_011_001;
68       #2  ms_m=9'b110_100_001;
69       #2  ms_m=9'b111_100_001;
70       #2  ms_m=9'b111_100_000;
71       #2  ms_m=9'b111_100_101;
72       #2  ms_m=9'b111_100_111;
73      end
74      always
75              #1 Clk = !Clk;
76      endmodule
```

Tb_microprocesador

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:
4       // Engineer:
5       // José Alfredo Hernández Dueñas
6       //Jesús Francisco Villaseñor Correa
7       // Create Date: 16.11.2020 18:23:36
8       // Design Name:
9       // Module Name: tb_microUAZ8-2020v0_02
10      // Project Name:
11      // Target Devices:
12      // Tool Versions:
13      // Description:
14      //
15      // Dependencies:
16      //
17      // Revision:
18      // Revision 0.01 - File Created
19      // Additional Comments:
20      //
21
        //////////////////////////////////////////////////////////////////////////////////
```

```
22
23
24    module tb_microUAZ8_2020v0_02( );
25    reg [8:0] Instruction;
26    reg [7:0] Datain_Bus;
27    wire [7:0] Address_Instruction_Bus;
28    wire [7:0] DataOut_Bus;
29    wire [7:0] Addres_Data_Bus;
30    wire LE;
31    reg Clk;
32    reg Rst;
33
34    Microprocesador_Equipo_N uut(
35    .Instruction(Instruction),
36    .Datain_Bus(Datain_Bus),
37    .Address_Instruction_Bus(Address_Instruction_Bus),
38    .DataOut_Bus(DataOut_Bus),
39    .LE(LE),
40    .Clk(Clk),
41    .Rst(Rst)
42    );
43
44    initial
45        begin
46         Instruction=0;
47        Rst=1;
48    Clk=0;
49
50        Datain_Bus=8'b0010110;
51        #2 Instruction=9'b000_010_010;
52        #2 Instruction=9'b001_110_001;
53        #2 Instruction=9'b010_011_011;
54        #2 Instruction=9'b011_010_110;
55        #2 Instruction=9'b100_000_011;
56        #2 Instruction=9'b101_010_000;
57        #2 Instruction=9'b110_000_000;
58        #2 Instruction=9'b111_100_100;
59        end
60        always
61            #1 Clk = !Clk;
62    endmodule
```

Tb_micromemorias

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     // José Alfredo Hernández Dueñas
6     //Jesús Francisco Villaseñor Correa
7     // Create Date: 18.11.2020 19:43:18
8     // Design Name:
9     // Module Name: tb_Micropromemorias_Equipo_N
10    // Project Name:
11    // Target Devices:
12    // Tool Versions:
13    // Description:
14    //
15    // Dependencies:
16    //
17    // Revision:
18    // Revision 0.01 - File Created
19    // Additional Comments:
20    //
21
      //////////////////////////////////////////////////////////////////////////////////
22
23
24    module tb_Micropromemorias_Equipo_N;
25    reg Clk;
26        reg Rst;
27
28        Micropromemorias_Equipo_N uut(
29            .Clk(Clk),
30            .Rst(Rst)
31            );
32
33        initial
34            begin
35            Rst=1;
36            Clk=0;
37
38            #2 Rst=0;
39
40            end
41
42        always
43            #1 Clk = !Clk;
44    endmodule
```