Proyecto de

Electrónica Digital I

Tema: Procesador Turing Completo con Arquitectura de 4 Bits

Integrantes:

- · ***Steven Yosette Mendoza Zapata H.***
- · ***Jeremy Benjamín Chávez H.***

Objetivo General

- Realizar el diseño de un Procesador personalizado a partir de los conocimientos Adquiridos

Objetivos Específicos

- Realizar la construcción de una RAM, un Caché y una ALU
- Crear un lenguaje para el procesador y construir una tabla de funciones sobre sus posibles combinaciones.
- Diseñar un procesador de 4 bits Turing Completo.

Temas

1. Conocimientos iniciales

a. Creación de Compuertas lógicas a partir de transistores
b. Buffer de 3 estados
c. Sistema de Complemento de 2.
d. 1B Multiplexer
e. 4B Multiplexer

2. Unidad Aritmética Lógica

a. Unidad Aritmética
    i. Full Adder
    ii. 4B Adder
    iii. 4B Adder / Substracter
    iv. Construcción de la Unidad Aritmética

b. Unidad Lógica
    i. AND Bitwise (Bit a bit)
    ii. OR Bitwise (Bit a bit)
    iii. 4B NOT
    iv. Construcción de la Unidad Lógica

c. Comparador
    i. Construcción del Comparador

d. Construcción del ALU

3. RAM

a. Flip Flop D
b. 1B Registro
c. 4B Registro
d. RAM 4x4B
e. RAM 16x4B
f. RAM 16x12B

4. Cache

a. Cache 4X4B con 3 Quick Access
b. Cache 8X4B con 3 Quick Access

5. Procesador

a. Modos de Trabajo
b. Construcción del Código Assembly de 3 Segmentos
c. Tablas de Funciones del Procesador
d. Construcción del Procesador

Introducción

El objetivo del proyecto es lograr desarrollar el diseño de un procesador capaz de realizar operaciones aritméticas, lógicas de manipulación de memoria y en el proceso si es posible poder desarrollar un procesador que sea Turing completo.

Para que un Procesador / Computadora pueda realizar cualquier tipo de trabajo que sea computable, mientras tenga el suficiente tiempo y espacio de memoria, tiene que ser un procesador Turing completo, para ello tiene que cumplir las siguiente condiciones:

- **Conjunto completo de instrucciones**: Un procesador debe tener un conjunto completo de instrucciones que permite realizar cualquier cálculo. Esto implica operaciones aritméticas básicas, manipulación de memoria y operaciones lógicas.

- **Capacidad para bucles y condicionales:** Debe ser capaz de realizar bucles (iteraciones) y ejecutar instrucciones condicionales. Esto permite la repetición de operaciones y la toma de decisiones basadas en condiciones.

- **Manipulación de datos**: El procesador debe poder almacenar y manipular datos de manera eficiente.

- **Memoria universal:** El procesador debe tener acceso a una memoria que pueda almacenar tanto datos como instrucciones de programa. Esto incluye la capacidad de leer y escribir en la memoria.

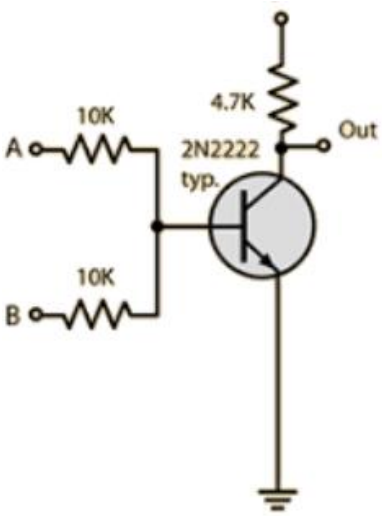
- **Entrada y salida:** Debe poder recibir datos de entrada y enviar datos de salida. Esto le permite interactuar con su entorno y realizar cálculos basados en datos externos.

Para poder llevar a cabo este propósito se tiene que desarrollar 3 componentes importantes de un procesador, la Unidad Aritmética Lógica, una Memoria Principal, y un Caché, así como diseñar la capacidad de poder programar y ejecutar código con la capacidad de realizar operaciones aritmética, lógicas, manipular datos de memoria, poder correr programas cargados en la memoria y poder recibir instrucciones (ser programable), donde se pueda realizar lo antes dicho, verificación de condiciones y bucles.

Conocimientos Previos Necesarios

Creación de compuertas lógicas

Para la creación de las compuertas a partir de transistores BJT 2N2222A, se van a crear a partir de la compuerta NOR, debido a que las compuertas más estables al usar transistores son la NOT y la NOR, Al ser NOR una compuerta universal, nos ayudará a construir el resto de compuertas lógicas.



Al tener la configuración mostrada en la imagen, al A o B estar en alta, activa el transistor, por lo que la salida Out no se alimentara, ya que el camino con menor resistencia es hacia tierra. En caso contrario donde A y B están en baja, no se activa el BJT, por lo que el único camino en el que pide fluir la corriente es la salida Out, por lo que su salida seria 1,

Buffer de 3 estados:

Un buffer de tres estados, también conocido como "buffer tristate" o "buffer de alta impedancia", es un dispositivo de circuito digital que tiene tres estados de operación: alto (1), bajo (0) y alta impedancia (Z). La función principal de un buffer de tres estados es permitir que múltiples dispositivos comparten un bus de datos sin interferir entre sí.

Cuando el buffer está en estado alto, la salida es 1. Cuando está en estado bajo, la salida es 0. Sin embargo, lo que hace único a un buffer de tres estados es su capacidad para entrar en un tercer estado de alta impedancia (Z), donde la salida se desconecta eléctricamente del bus al que está conectado. En este estado, la salida actúa como si estuviera desconectada, lo que significa que no afecta la línea de bus y permite que otros dispositivos conectados al mismo bus tomen el control.

La capacidad de un buffer de tres estados para entrar en el estado de alta impedancia es especialmente útil en situaciones en las que se necesita compartir un bus de datos entre varios dispositivos sin causar conflictos.

Sistema Complemento de 2

El sistema de complemento a dos es una representación numérica utilizada para representar números enteros, tanto positivos como negativos, de manera binaria.

Pasos para encontrar el complemento a dos de un número binario:

- Encuentra el complemento a uno: Cambia todos los bits 0 a 1 y viceversa en la representación binaria del número.

- Suma 1: Suma 1 al resultado obtenido en el paso anterior.

Estos dos pasos juntos te darán el complemento a dos del número original.

1B Multiplexer

Un multiplexor de 1 bit (también conocido como MUX de 1 bit) es un dispositivo de circuito digital que se utiliza para seleccionar uno de dos posibles valores de entrada. Su función principal es dirigir uno de los dos bits de entrada a la salida según una señal de control.

4B Multiplexer

Un multiplexor (MUX) de 4 bits y 2 canales selecciona una de las dos entradas de 4 bits y dirige esa entrada a la salida, dependiendo de las señales de control.

Unidad Aritmética Lógica (ALU)

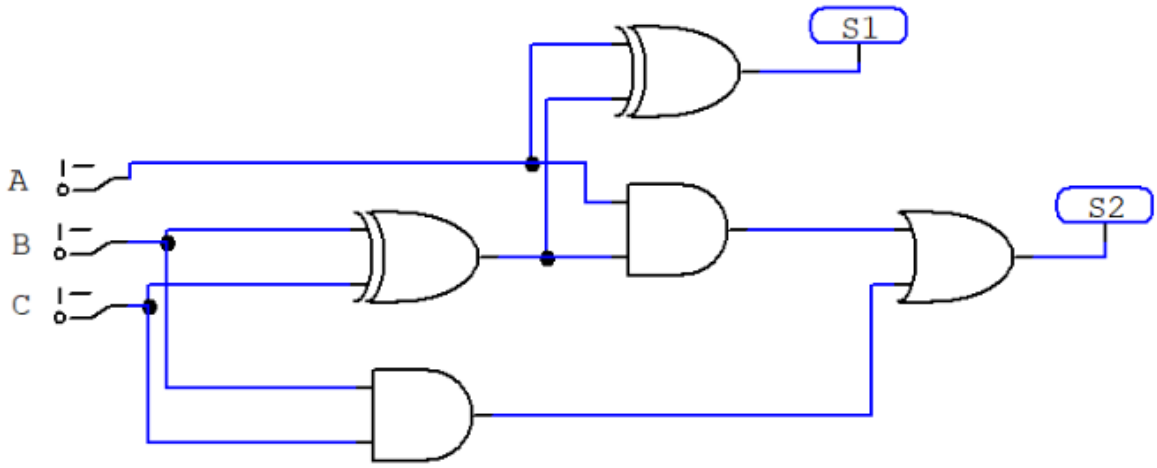
ALU: Unidad Aritmética

- **Full Adder**

Un sumador completo (Full Adder) es un circuito lógico que realiza la suma binaria de tres bits de entrada: A, B y un bit de acarreo de entrada (Cin). Su salida consta de dos bits: la suma (S) y el bit de acarreo de salida (Cout). Este componente es fundamental en la construcción de sumadores más grandes y se utiliza comúnmente en la aritmética binaria.

La función de un sumador completo se puede expresar mediante las siguientes ecuaciones booleanas:

$S = A \oplus B \oplus Cin$
$Cout = (AB) + ((A \oplus B) Cin)$



La salida de la XOR de A, B y Cin representa la suma binaria de los tres bits de entrada.
Acarreo de Salida (Cout):

Este sumador completo se utiliza en cascada para construir sumadores más grandes, como el sumador de 4 bits,, etc. Al encadenar varios sumadores completos, se pueden realizar sumas binarias de números de longitud arbitraria. Es un componente esencial en el diseño de la aritmética binaria en circuitos digitales, por lo que fue el punto de partida para realizar la unidad aritmética.

Entradas			Salidas	
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- **4B Adder**

Un sumador de 4 bits (4-bit adder) es un circuito lógico que realiza la suma binaria de dos números de 4 bits. Se construyó combinando cuatro sumadores completos de 1 bit (Full Adders) en cascada.

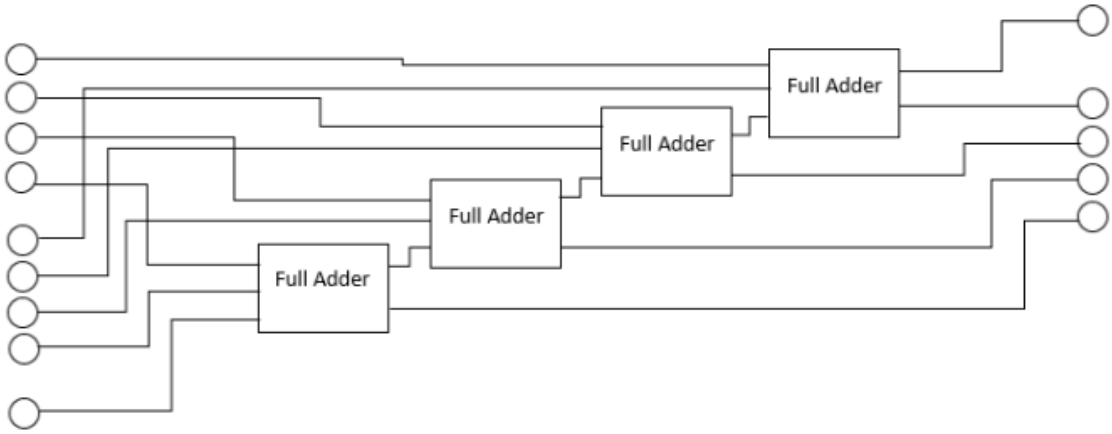
La operación consiste en cuatro etapas, una para cada par de bits correspondientes en A y B, para la etapa menos significativa, el bit de acarreo de entrada (Cin) es 0, para las etapas siguientes, el bit de acarreo de entrada proviene del bit de acarreo de la etapa anterior. La salida del sumador de 4 bits consta de un resultado de 4 bits y un bit de acarreo de salida, que se tomará como overflow (sobrecarga).

Entradas:

Dos números de 4 bits que llamaremos A (A3, A2, A1, A0) y B (B3, B2, B1, B0).

Salidas:

Un resultado de 4 bits S (S3, S2, S1, S0) que representa la suma de A y B.
Un bit de acarreo de salida (Cout) que indica si hubo un acarreo fuera del bit más significativo, en este caso es tratado como overflow.



- **4B Adder / Substracter**

Un Sumador/Restador de 4 bits es un circuito lógico que realiza la suma o resta binaria de dos números de 4 bits, por medio de un selector de operaciones. Se construyó a partir del Adder de 4B previamente explicado.

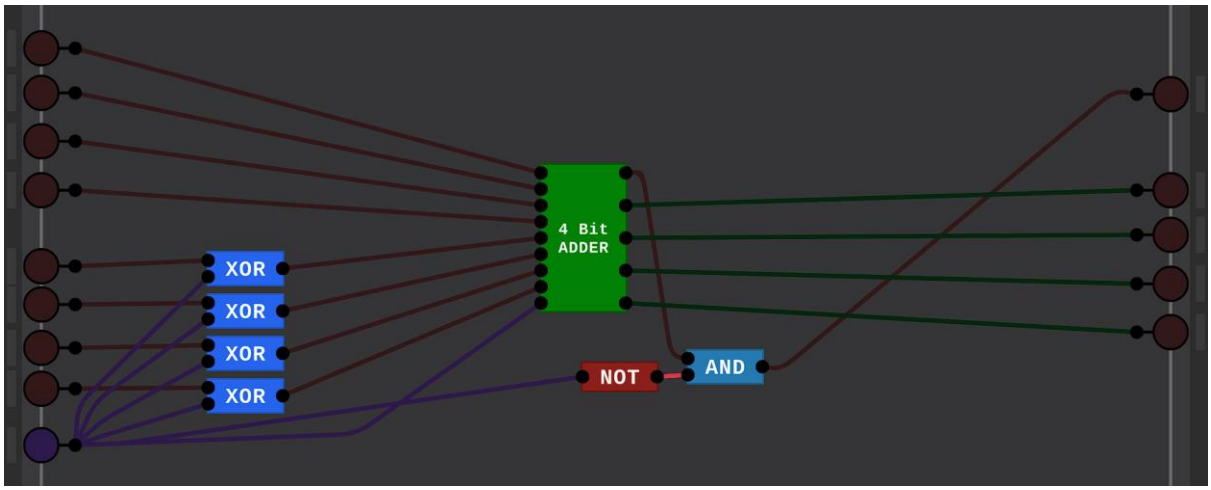
Cuando el selector de operaciones (Op), esté en 0, las entradas A y B, del circuito van directamente a las entradas correspondientes del 4B Adder y la entrada del carry queda en baja, pero cuando Op esté en 1, se niegan las entradas B y se añade una señal en alta en la entrada del carry del 4B Adder, con el propósito de convertir la operación de suma, en una resta usando, el sistema de complemento de 2.

Entradas:

Dos números de 4 bits que llamaremos A (A3, A2, A1, A0) y B (B3, B2, B1, B0) y un Selector de Operaciones llamado Op

Salidas:

Un resultado de 4 bits S (S3, S2, S1, S0) que representa la suma de A y B.
Un bit de acarreo de salida (Cout), que luego servirá para definir el overflow positivo y negativo.



- **Construcción de la Unidad Aritmética**

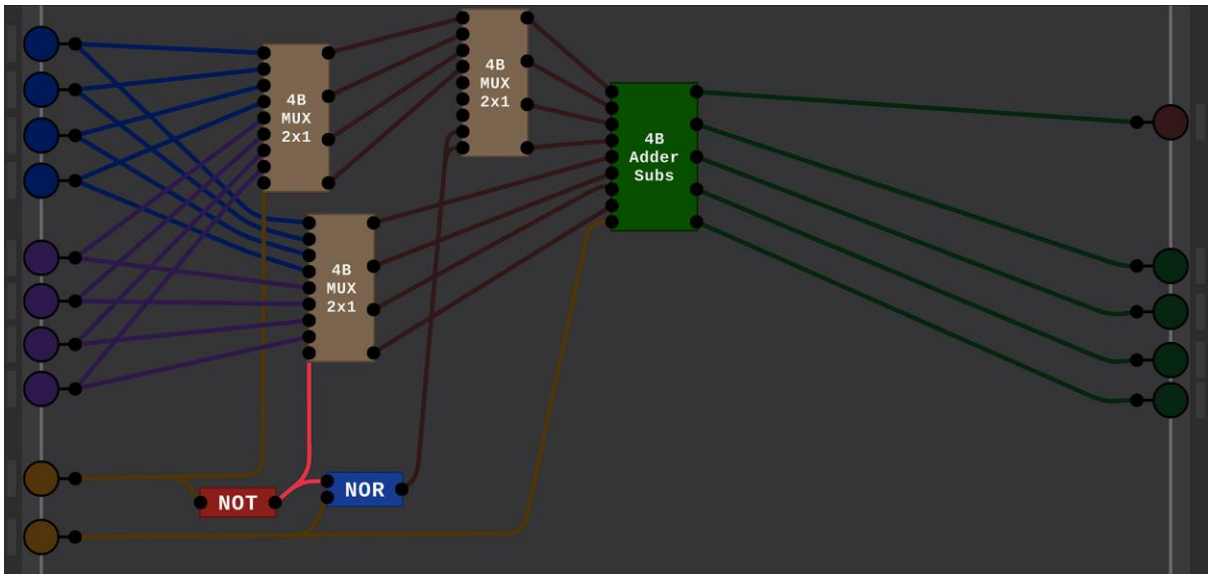
La unidad aritmética es un circuito que controla un conjunto de funciones aritméticas mediante selectores, en este caso, la Unidad Aritmética, contará con la posibilidad de elegir entre la función de (A + B), (A - B), (A + 1) y (B - A), donde A y B son números de 4 bits, a las entradas.

Entradas:

Dos números de 4 bits que llamaremos A (A3, A2, A1, A0) y B (B3, B2, B1, B0) y un Selector de Operaciones de 2 bits, llamado Op (Op1, Op0).

Salidas:

Un resultado de 4 bits S (S3, S2, S1, S0) que representa la salida de la función seleccionada. Un bit de acarreo de salida (Cout), que luego servirá para definir el overflow positivo y negativo.



Arithmetic Unit		
Op1	Op0	Function
0	0	A + B
0	1	A - B
1	0	A + 1
1	1	B - A

ALU: Unidad Lógica

- **4B AND Bitwise (Bit a bit)**

La operación AND bitwise es una operación lógica que se realiza bit a bit entre dos números binarios. El resultado de la operación AND en un bit, es 1, si los 2 bits correspondientes son 1.

Entradas:

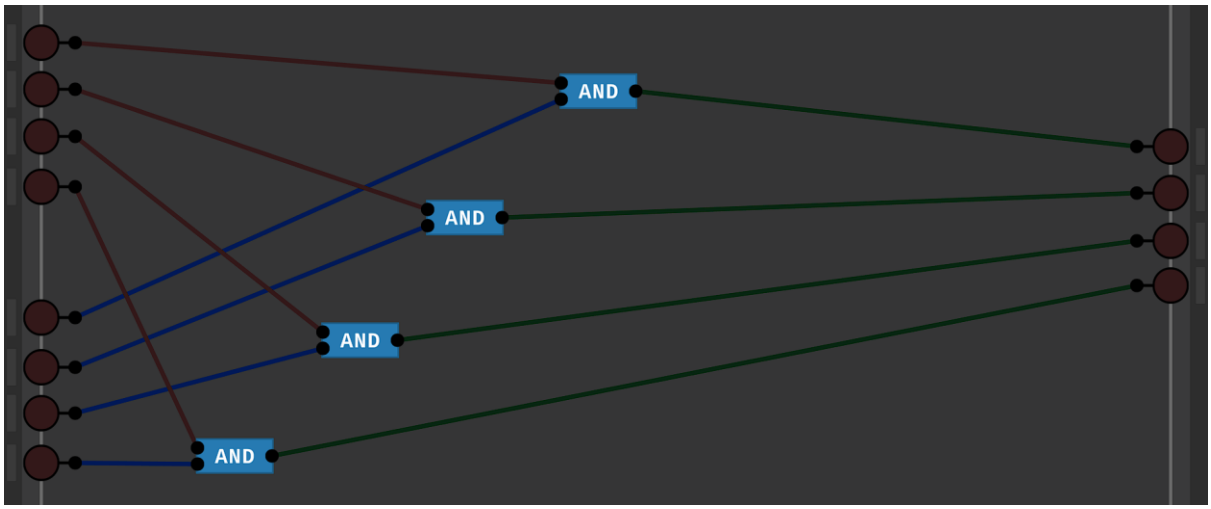
Número de 4 bits A (A3, A2, A1, A0) y B (B3, B2, B1, B0)

La operación AND bitwise se realiza de la siguiente manera para obtener el resultado (R) de 4 bits:

R3 = A3 AND B3
R2 = A2 AND B2
R1 = A1 AND B1
R0 = A0 AND B0

Salidas:

Número de 4 Bits R (R3, R2, R1, R0)



- **4B OR Bitwise (Bit a bit)**

La operación OR bitwise es una operación lógica que se realiza bit a bit entre dos números binarios. El resultado de la operación OR en un bit, es 1, si al menos uno de los bits correspondientes es 1.

Entradas:

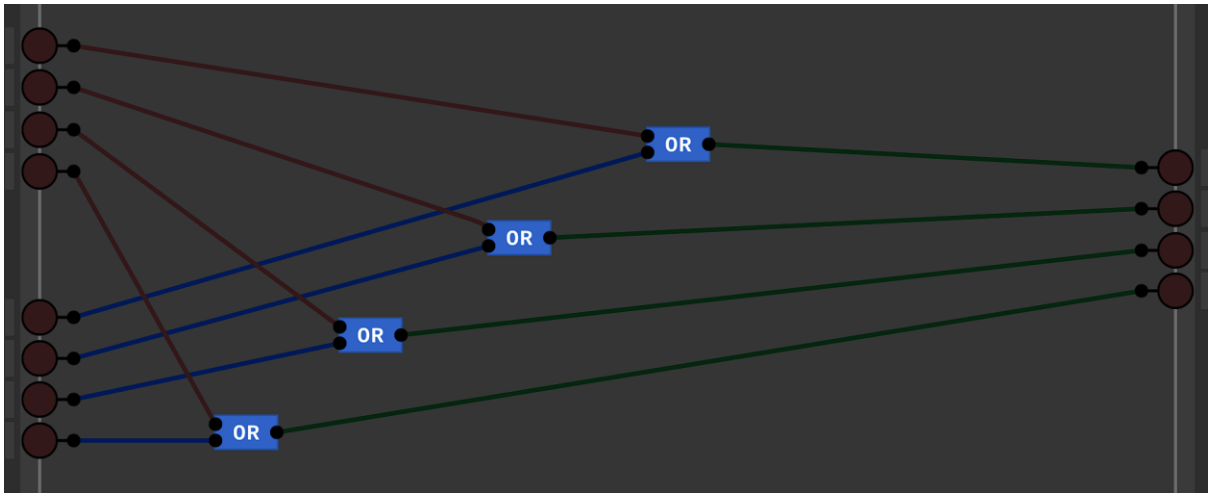
Número de 4 bits A (A3, A2, A1, A0) y B (B3, B2, B1, B0)

La operación OR bitwise se realiza de la siguiente manera para obtener el resultado (R) de 4 bits:

R3 = A3 OR B3
R2 = A2 OR B2
R1 = A1 OR B1
R0 = A0 OR B0

Salidas:

Número de 4 Bits R (R3, R2, R1, R0)



- **4B NOT**

La operación NOT bitwise es una operación lógica que se realiza bit a bit a un número binario de 4 bits. El resultado de la operación NOT en un bit, es la inversión de la entrada.

Entradas:

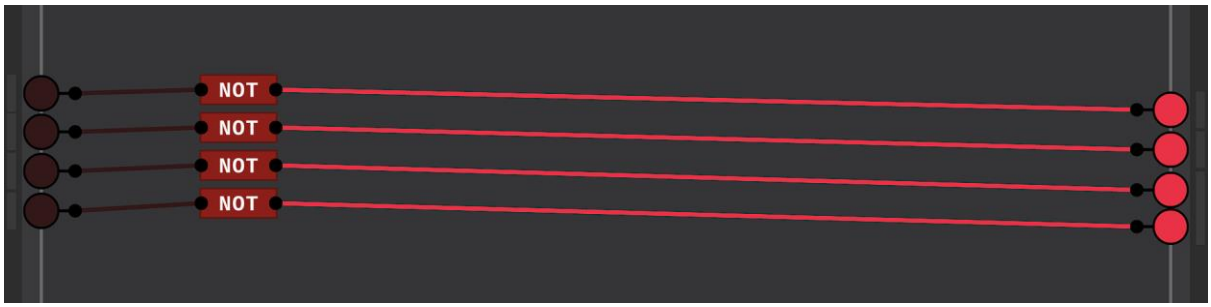
Número de 4 bits A (A3, A2, A1, A0)

La operación NOT bitwise se realiza de la siguiente manera para obtener el resultado (R) de 4 bits:

R3 = NOT (A3)
R2 = NOT (A2)
R1 = NOT (A1)
R0 = NOT (A0)

Salidas:

Número de 4 Bits R (R3, R2, R1, R0)



- **Construcción de la Unidad Lógica**

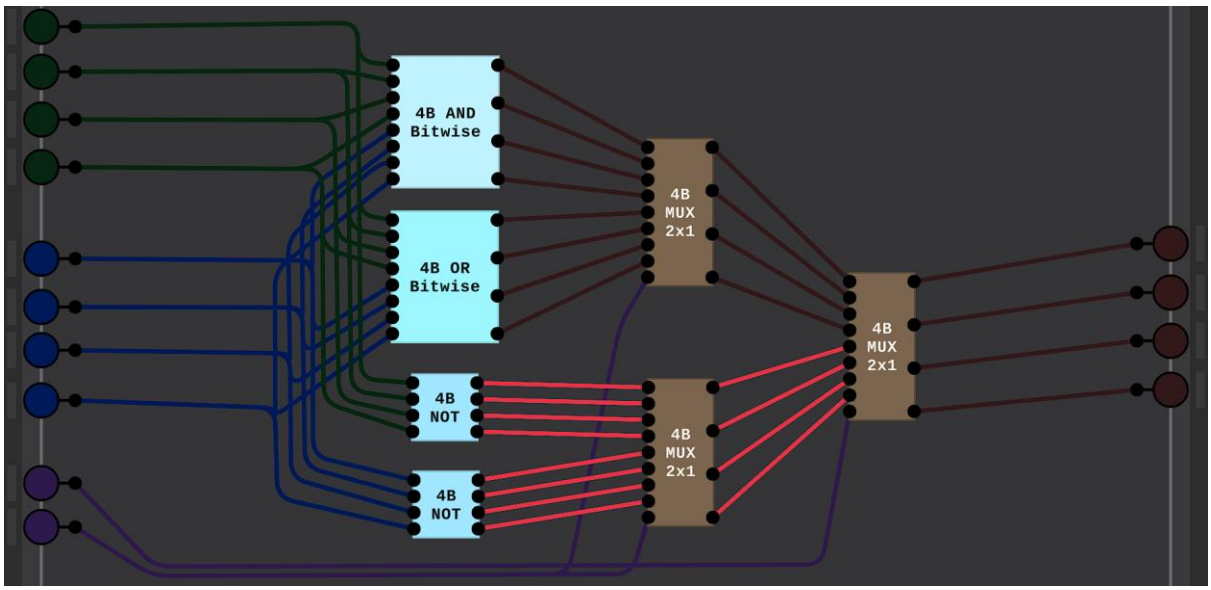
La unidad lógicas es un circuito que controla un conjunto de funciones lógicas, mediante selectores, en este caso, la Unidad Lógica, contará con la posibilidad de elegir entre la funciones bitwise de (A AND B), (A OR B), (NOT A) y (NOT B), donde A y B son números de 4 bits, a las entradas.

Entradas:

Dos números de 4 bits que llamaremos A (A3, A2, A1, A0) y B (B3, B2, B1, B0) y un Selector de Operaciones de 2 bits, llamado Op (Op1, Op0).

Salidas:

Un resultado de 4 bits S (S3, S2, S1, S0) que representa la salida de la función seleccionada. Un bit de acarreo de salida (Cout), que luego servirá para definir el overflow positivo y negativo.



Logic Unit		
Op1	Op0	Function
0	0	A AND B
0	1	A OR B
1	0	NOT(A)
1	1	NOT(B)

ALU: Comparador

- **Construcción del 4B Comparador**

Un comparador de 4 bits es un circuito lógico que compara dos números de 4 bits y determina su relación (mayor que, menor que o igual). El resultado de la comparación se expresa a través de señales de salida que indican si uno es mayor, menor o igual al otro.

Para realizar la comparación bit a bit de A y B, compara el bit más significativo primero y luego los bits menos significativos.

Para cada par de bits correspondientes en A y B, de manera decreciente, del bit más significativo al menos:
Si A[i] > B[i], establece la señal "A > B" en 1 y las demás en 0.
Si A[i] < B[i], establece la señal "A < B" en 1 y las demás en 0.
Si A[i] = B[i], pasa al siguiente bit y repite el proceso.:

Si todas las comparaciones de bits son iguales, establece la señal "A = B" en 1 y las demás en 0.

Entradas:

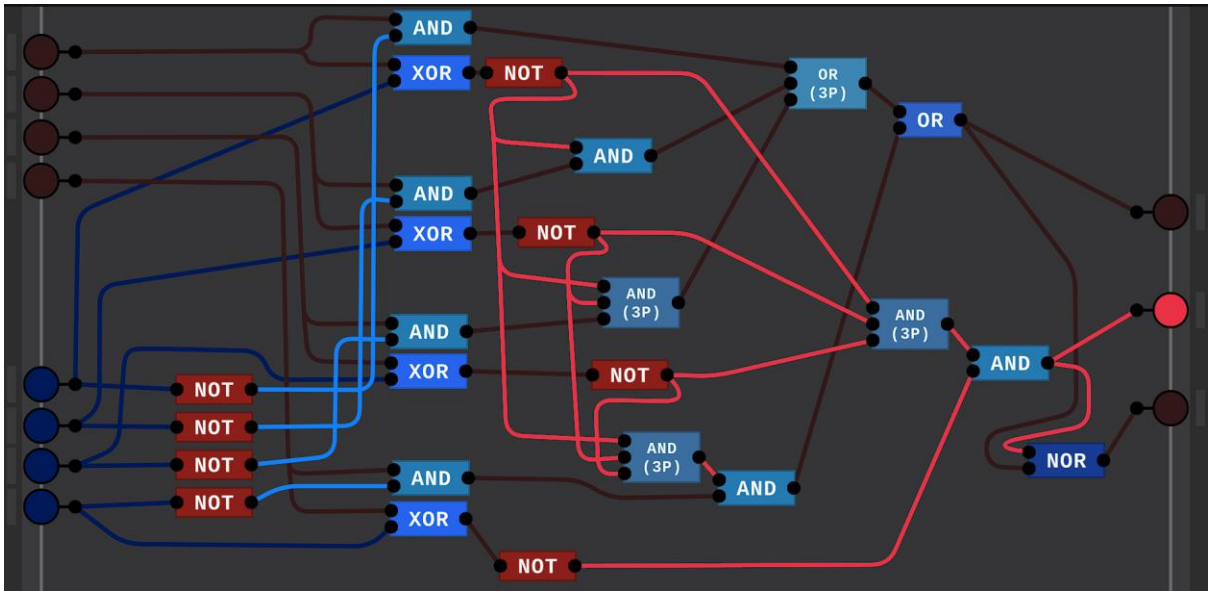
Dos números de 4 bits, A (A3, A2, A1, A0) y B (B3, B2, B1 B0).

Salidas:

Las Tres señales de salida que indican la relación de magnitud:

- (A > B) A mayor que B
- (A = B) A igual a B
- (A < B).A menor que B

.



ALU: Construcción de la Unidad Aritmética Lógica

La Unidad Lógica Aritmética es un circuito que controla un conjunto de funciones lógicas y aritmética, mediante selectores, en este caso, la Unidad Lógica Aritmética, contará con la posibilidad de elegir entre las funciones lógicas bitwise de (A AND B), (A OR B), (NOT A) y (NOT B) y las aritméticas (A + B), (A - B), (A + 1) y (B - A) donde A y B son números de 4 bits, a las entradas.

Además de los antes mencionado, cumple la función de comparar los valores A y B, mediante el comparador, quedando en alta 1 de las 3 relacionales (A > B), (A = B) o (A < B) en dependencia en la relación de tamaños de ambos números.

A partir de la comparación de los datos, se pudo crear una salida que indique que el resultado es negativo, si se aplica (A - B) y (A < B) o si se aplica (B - A) y (A > B). Se creó otra salida para indicar que hay overflow. Y una más para indicar si el dato es resultado de una operación lógica. En conjunto esas 3 salidas representan el tipo de dato de la salida del ALU.

Result Type			
T			Data Type
Logical / Arithmetic	Overflow	Negative	
0	0	0	4B bool
0	0	1	Undefined
0	1	0	Undefined
0	1	1	Undefined
1	0	0	4B unsigned int
1	0	1	4B int
1	1	0	Overflow
1	1	1	Overflow

Entradas:

Dos números de 4 bits que llamaremos A (A3, A2, A1, A0) y B (B3, B2, B1, B0) y un Selector de Operaciones de 3 bits, llamado Op (Op2, Op1, Op0).

Salidas:

Un resultado de 4 bits S (S3, S2, S1, S0) que representa la salida de la función seleccionada.
C (C2, C1, C0): Tres señala que representen la relación de tamaño entre A y B, (A > B), (A = B) y (A < B).
T (T2, T1, T0):Tres salidas que representan el tipo de dato de la salida del procesador

ALU				
	Op2	Op1	Op0	Function
Logic	0	0	0	A AND B
	0	0	1	A OR B
	0	1	0	NOT(A)
	0	1	1	NOT(B)
Arithmetic	1	0	0	A + B
	1	0	1	A - B
	1	1	0	A + 1
	1	1	1	B - A

RAM

Flip Flop D

Un flip-flop D, también conocido flip-flop de datos, es un tipo de elemento de almacenamiento digital que puede almacenar un solo bit de información. Es un tipo de flip-flop muy utilizado en circuitos digitales para la implementación de registros y memorias.

La operación básica del flip-flop D se puede describir de la siguiente manera:

Entradas:

D (Data): Es la entrada de datos que se desea almacenar en el flip-flop.
CLK (Clock): Es la señal de reloj que sincroniza la operación del flip-flop. El estado del flip-flop D se actualiza en dependencia de la señal de reloj (por ejemplo, en el flanco de subida o bajada, o por nivel, dependiendo del diseño).
Reset: Resetea el valor guardado a 0 en el próximo tick de reloj,

Salidas:

Q (Output): Representa el estado actual almacenado en el flip-flop D. Es el bit que se lee desde el flip-flop.
Q (Complement Output): Es la salida complementaria de Q. Si Q es 1, será 0, y viceversa.

Funcionamiento:

Cuando la señal de reloj CLK hace una transición (por ejemplo, de bajo a alto), el estado en la entrada D se copia en la salida Q.
Si D es 1, entonces en el próximo flanco de reloj, Q será 1.
Si D es 0, entonces en el próximo flanco de reloj, Q será 0.

1B Registro

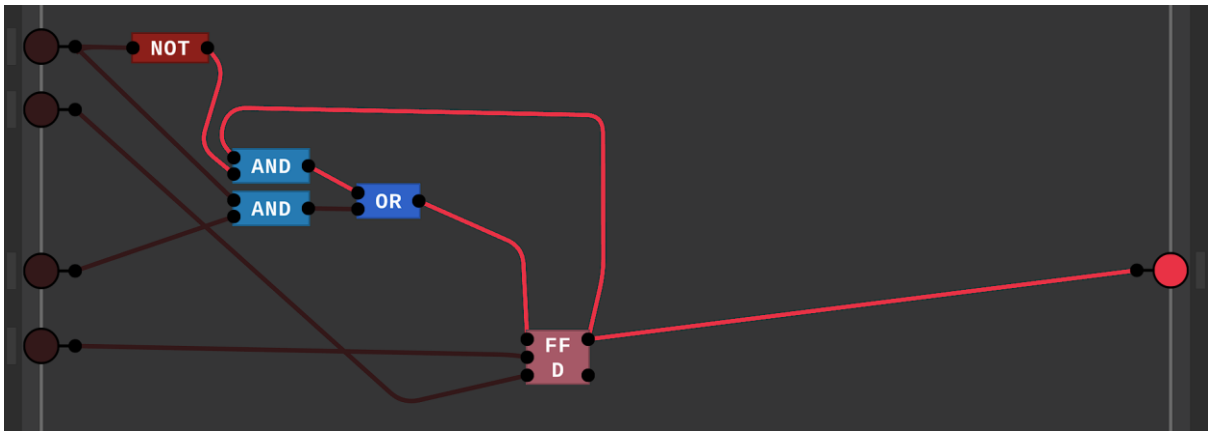
El registro de 1 bit, es un circuito secuencial con la capacidad de almacenar un dato de un bit, hecho a partir de un Flip Flop D, añadiendo una pequeña funcionalidad para facilitar su uso en estructuras de guardado de memoria, siendo la entrada Load, la cual controla cuando se guarda el dato de la entrada D (Data).

Entradas:

D (Data): Es la entrada de datos que se desea almacenar en el registro.
CLK (Clock): Es la señal de reloj que sincroniza la operación del registro.
Reset: Reinicia el valor guardado a 0 en el próximo tick de reloj,
Load: Permite guardar el valor en D, cuando este esté en alta y en baja retiene el valor guardado.

Salidas:

Q (Output): Número almacenado actualmente en el registro.
.



4B Registro

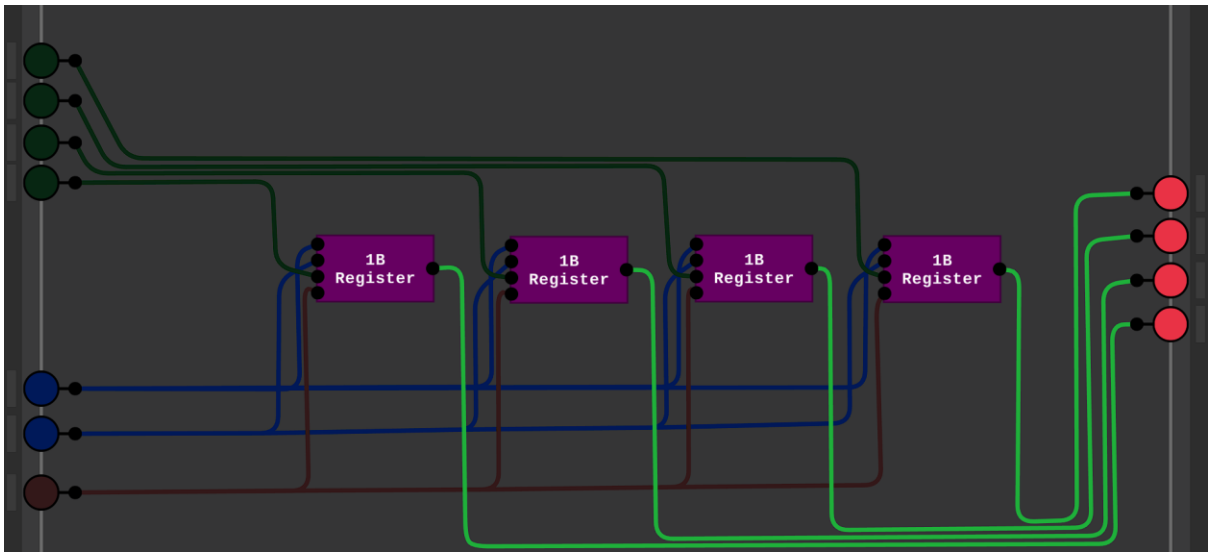
El registro de 4 bit, es un circuito secuencial con la capacidad de almacenar un dato de 4 bit, construido a partir de 4 registros de 1 bit, en paralelo, conectando el clock, reset y load, de cada registro, para mantener su sincronización, aumentando el tamaño del bit a almacenar, conservando el resto de características del registro de 1 bit.

Entradas:

D (D3, D2, D1, D0): Es el conjunto de datos que se desea almacenar en los registros.
CLK (Clock): Es la señal de reloj que sincroniza la operación de los registros.
Reset: Resetea el valor guardado a 0 en todos los registros, en el próximo tick de reloj,
Load: Permite guardar el valor de D, cuando este esté en alta y en baja retiene el valor guardado.

Salidas:

Q (Q3, Q2, Q1, Q0): Número de 4 bits almacenado actualmente en el registro.



RAM 4x4B

RAM (Random Access Memory) es la memoria temporal de un dispositivo electrónico que se utiliza para almacenar datos y ejecutar programas.

La RAM de 4 Lineas (Registros) de 4 Bits, está hecha mediante Registros de 4 bits, haciendo uso de un selector de bits 2 para elegir la dirección del registro, para almacenar el dato, siendo ahora las condiciones de cada registro de 4B, para que su entrada load esté en alta, que el selector y su posición sean iguales, así como que la entrada load de la RAM esté en alta.

El selector además de servir como dirección del registro para su escritura, también selecciona los registro para su lectura, colocando las salidas de los registros en multiplexers, para elegir la señal mediante la dirección, siendo la salida del último multiplexer la señal guardada en el registro de la dirección seleccionada.

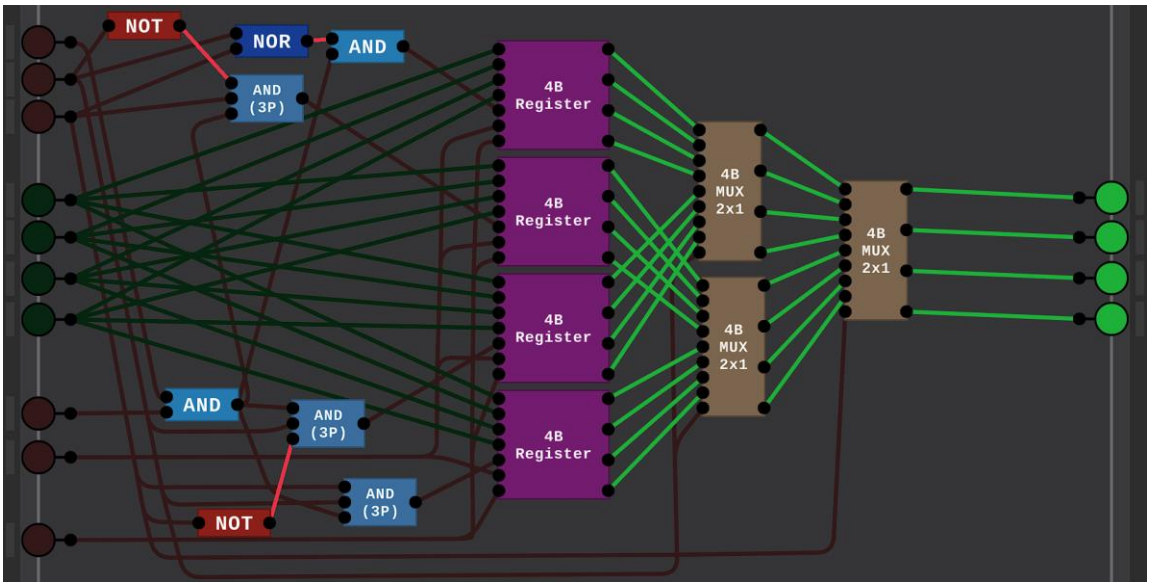
Mientras que la entrada del Clock, Reset y Data de la RAM están conectadas a cada una de las entradas correspondientes de los registros, para garantizar la sincronización de los registros, pero permitiendo la lectura y escritura de datos de maneras aisladas.

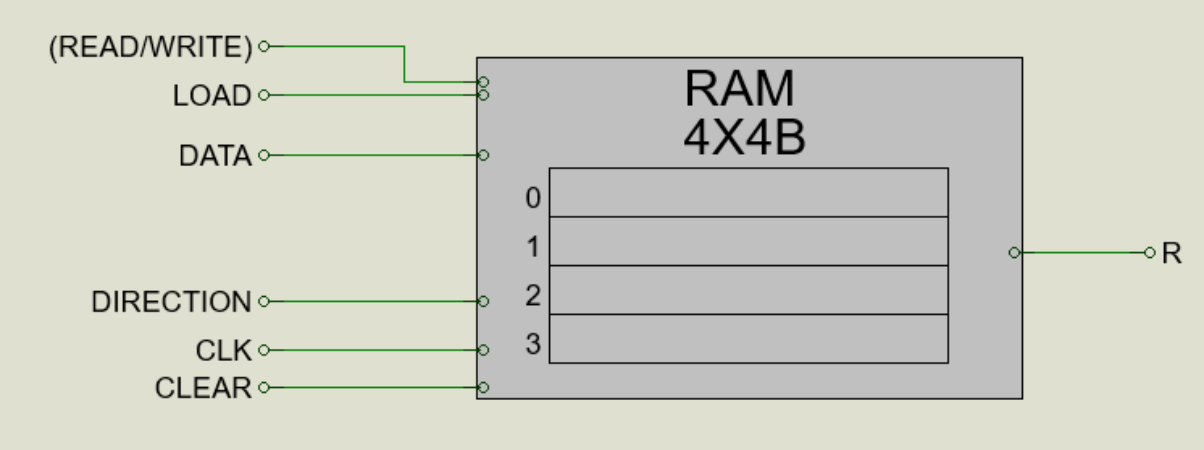
Entradas:

D (D3, D2, D1, D0): Es el conjunto de datos que se desea almacenar en un registro de 4 bits específico.
CLK (Clock): Es la señal de reloj que sincroniza la operación de los registros.
Reset: Resetea el valor guardado a 0 en todos los registros, en el próximo tick de reloj.
Load: Permite guardar el valor D, en un registro específico.
S (S1, S0) (Selector) Dirección del registro a Leer/Escribir.

Salidas:

Q (Q3, Q2, Q1, Q0): Número de 4 bits almacenado actualmente en el registro seleccionado.




(READ/WRITE)
LOAD
DATA
DIRECTION
CLK
CLEAR
RAM
4X4B
0
1
2
3
R

RAM 16x4B

La RAM de 16 Lineas (Registros) de 4 Bits, está hecha mediante 4 RAMs de 4x4B, conectando el Clock, la Data, el Reset y los 2 bits menos significativos del selector de la dirección, en las entradas correspondiente de cada RAM para su sincronización.

Y condicionando las entradas de Load, para que esté en alta, si la entrada Load y los 2 bits más significativos del Selector, son iguales a la dirección asignada de la RAM.

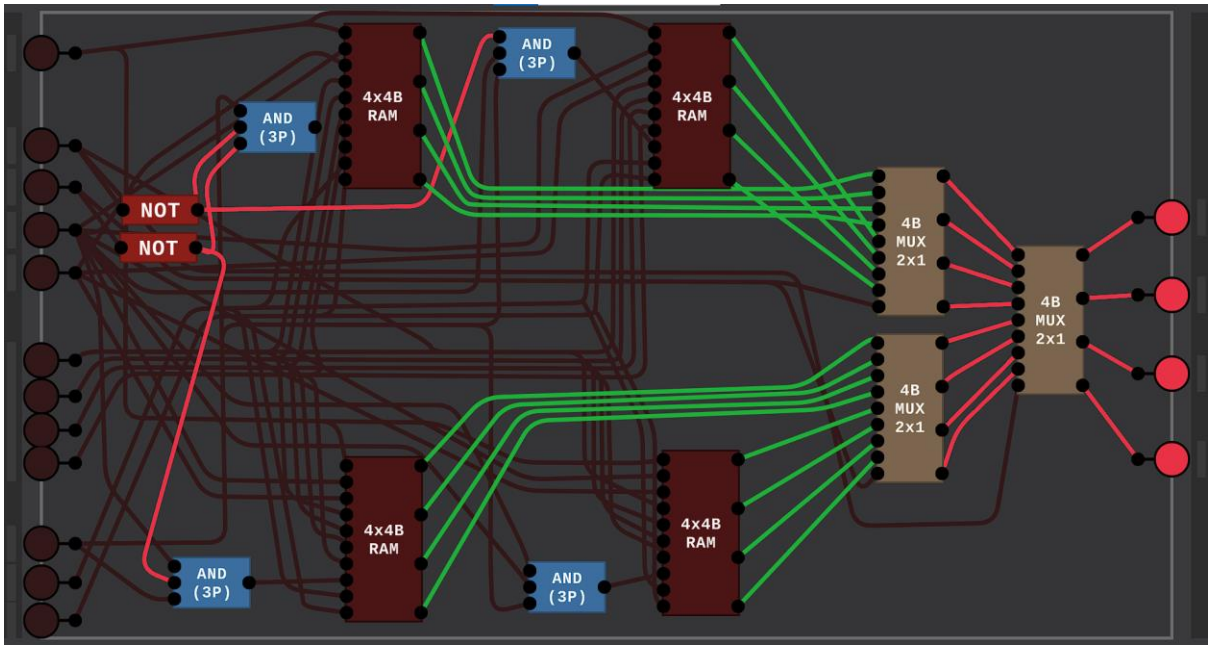
El selector además de servir como dirección del registro para su escritura, también selecciona los registro para su lectura,al igual que la RAM 4x4B, solo que ahora seleccionando la lectura o escritura de 1 entre 16 registros de 4 bits.

Entradas:

D (D3, D2, D1, D0): Es el conjunto de datos que se desea almacenar en un registro de 4 bits específico.
CLK (Clock): Es la señal de reloj que sincroniza la operación de los registros.
Reset: Resetea el valor guardado a 0 en todos los registros, en el próximo tick de reloj.
Load: Permite guardar el valor D, en un registro específico.
S (S3, S2, S1, S0) (Selector) Dirección del registro a Leer/Escribir.

Salidas:

Q (Q3, Q2, Q1, Q0): Número de 4 bits almacenado actualmente en el registro seleccionado.



RAM 16x12B

La RAM de 16 Lineas (Registros) de 12 Bits, está hecha mediante 3 RAMs de 12x4B en paralelo, para aumentar la el tamaño de los números a almacenar envés de la cantidad,conectando el Clock, la Data, el Reset y el selector de la dirección, en las entradas correspondiente de cada RAM para su sincronización.

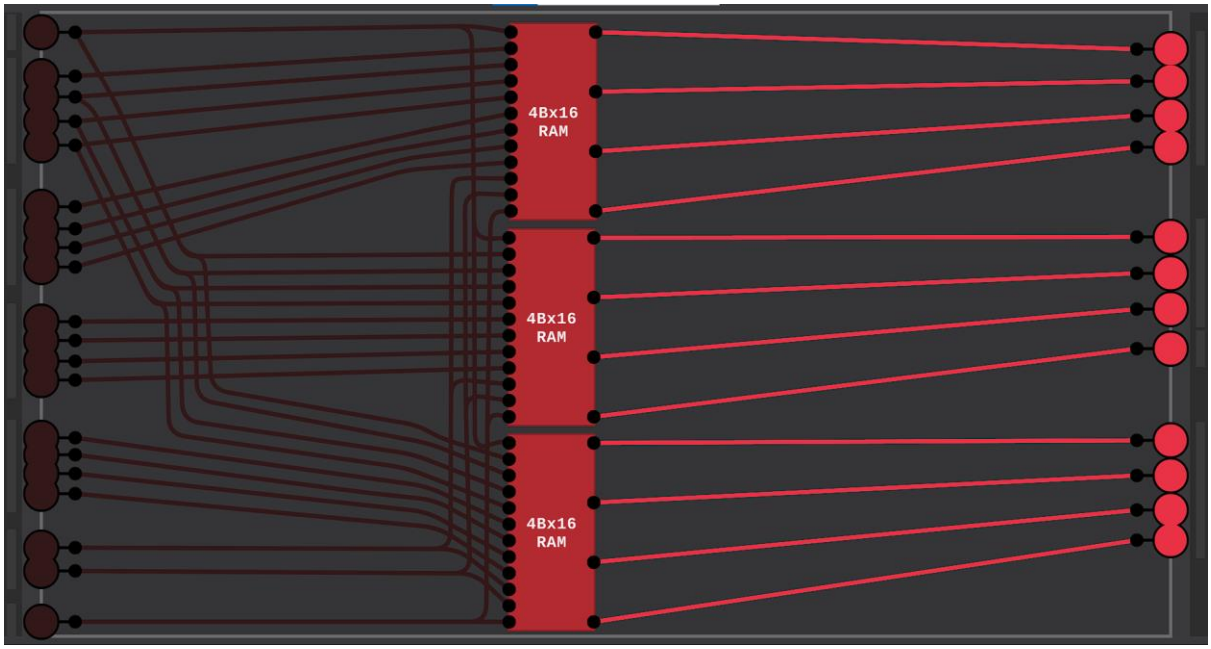
Se desarrolló la RAM hasta un tamaño de 16x12B, debido a que, el código assembly diseñado para el procesador es de 12B cada línea de código, y se dejó de 16 líneas debido a que es suficiente para programar programas muy sencillos, pero el tamaño de la RAM todavía conserva un tamaño considerable pequeño, con el objetivo de no volver el circuito complejo.

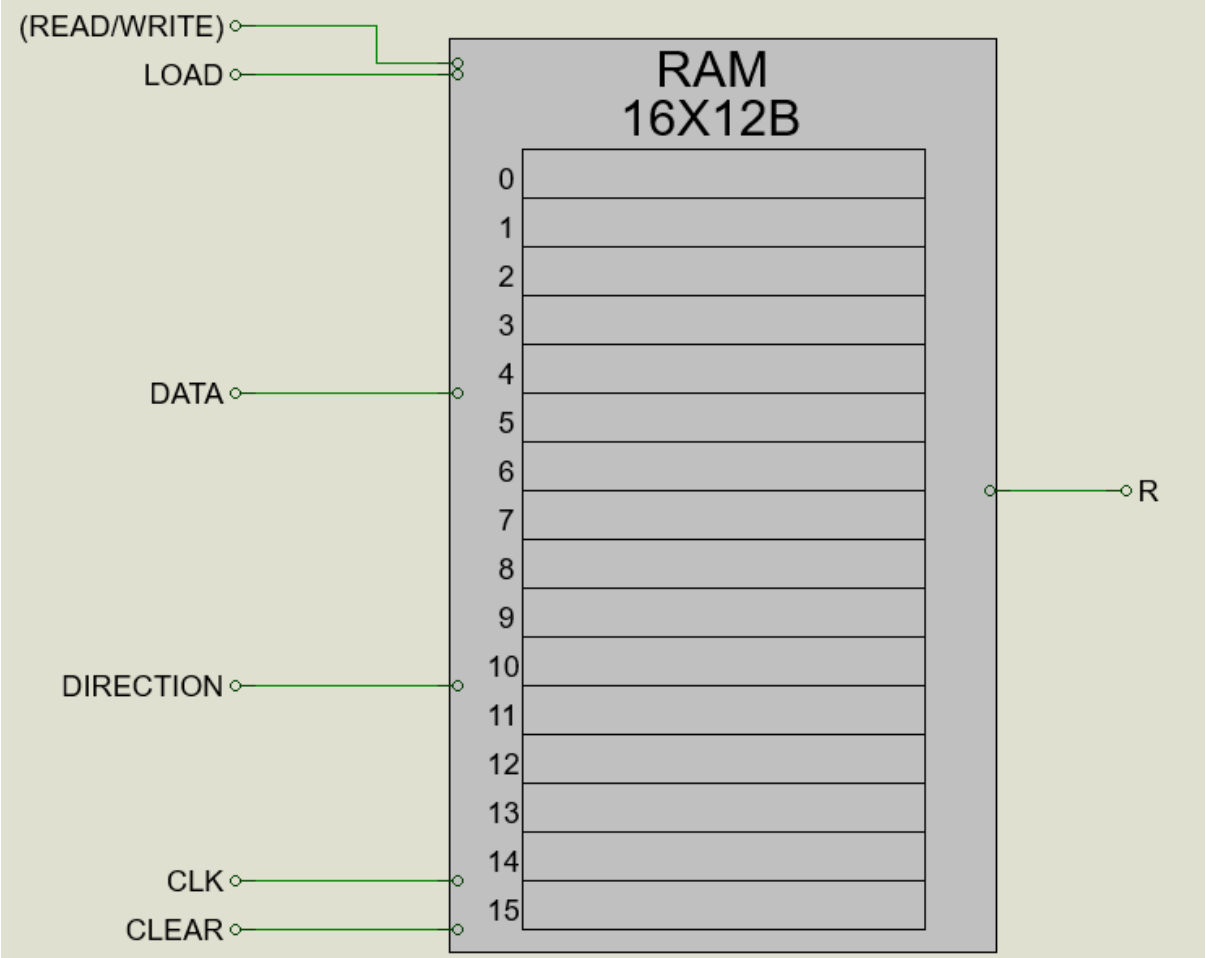
Entradas:

Code (Op3, Op2, Op1, Op0, A3, A2, A1, A0, B3, B2, B1, B0): Es un conjunto de datos, de 12B que la RAM puede almacenar, pensado para poder guardar una línea del código diseñado para el procesador.
CLK (Clock): Es la señal de reloj que sincroniza la operación de los registros.
Reset: Resetea el valor guardado a 0 en todos los registros, en el próximo tick de reloj.
Load: Permite guardar el valor D, en un registro específico.
S (S3, S2, S1, S0) (Selector) Dirección del registro a Leer/Escribir.

Salidas:

Q (Op3, Op2, Op1, Op0, Q3, Q2, Q1, Q0, B3, B2, B1, B0): Número de 12 bits almacenado actualmente en el registro seleccionado.




(READ/WRITE)
LOAD
RAM
16X12B
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
DATA
DIRECTION
CLK
CLEAR
R

Cache

Cache 4X4B con 3 Quick Access

La función principal de la caché es almacenar temporalmente datos e instrucciones que la CPU necesita con frecuencia, de modo que pueda acceder a ellos más rápidamente que si tuviera que recuperarlos de la memoria principal, además de poder proporcionar más de un dato al mismo tiempo, y junto con la RAM poder siempre que se quiera trabajar con 2 datos al mismo tiempo.

El Caché de 4 Lineas (Registros) de 4 Bits con 3 Registros de Acceso Rápido, es similar a la RAM 4x4B, solamente que el caché está diseñado para solo almacenar variables, no código, y con la finalidad de poner asignar y obtener varias datos en un mismo pulso de reloj. Por lo que tiene unas entradas directas a r0, r1 y r2, con una entrada extra cada uno para activar o desactivar la escritura del registro, y una salida directa a cada uno de los registros de acceso rápido.

Entradas:

D (D3, D2, D1, D0): Es el conjunto de datos que se desea almacenar en un registro de 4 bits específico.
CLK (Clock): Es la señal de reloj que sincroniza la operación de los registros.
Reset: Resetea el valor guardado a 0 en todos los registros, en el próximo tick de reloj.
Load: Permite guardar el valor D, en un registro específico.
S (S1, S0) (Selector) Dirección del registro a Leer/Escribir.

Dr0: Entrada de 4 bits, para guardar un dato en r0.
Wr0: Permite escribir en el registro r0.

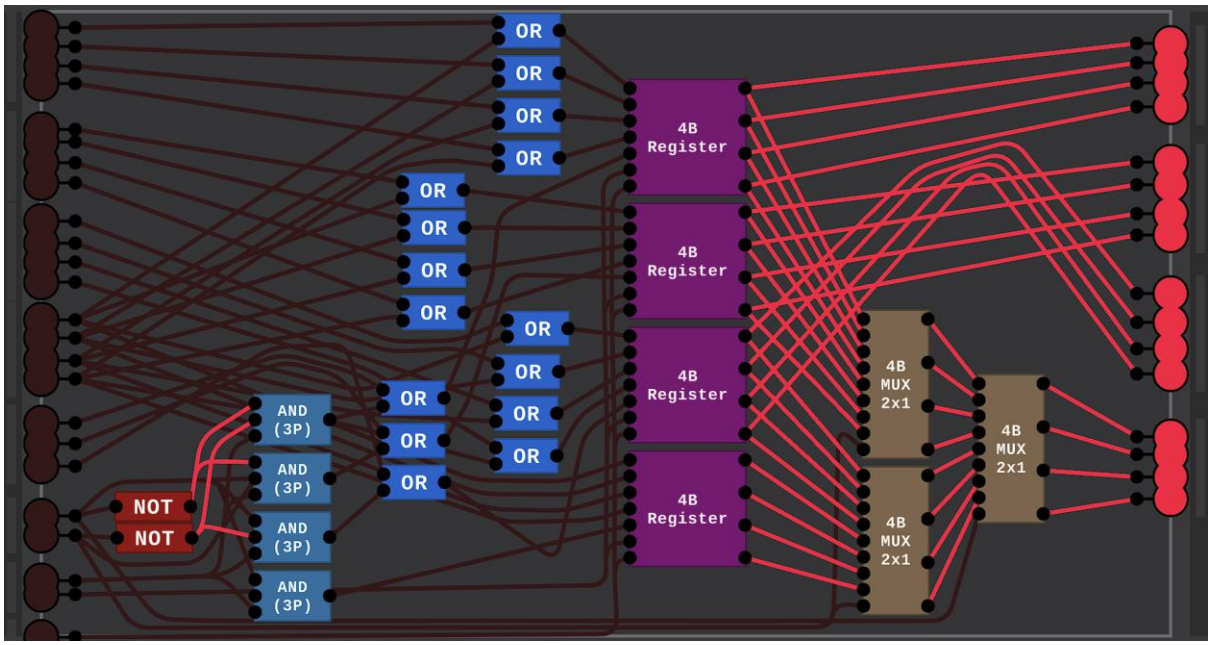
Dr1: Entrada de 4 bits, para guardar un dato en r1.
Wr1: Permite escribir en el registro r1.

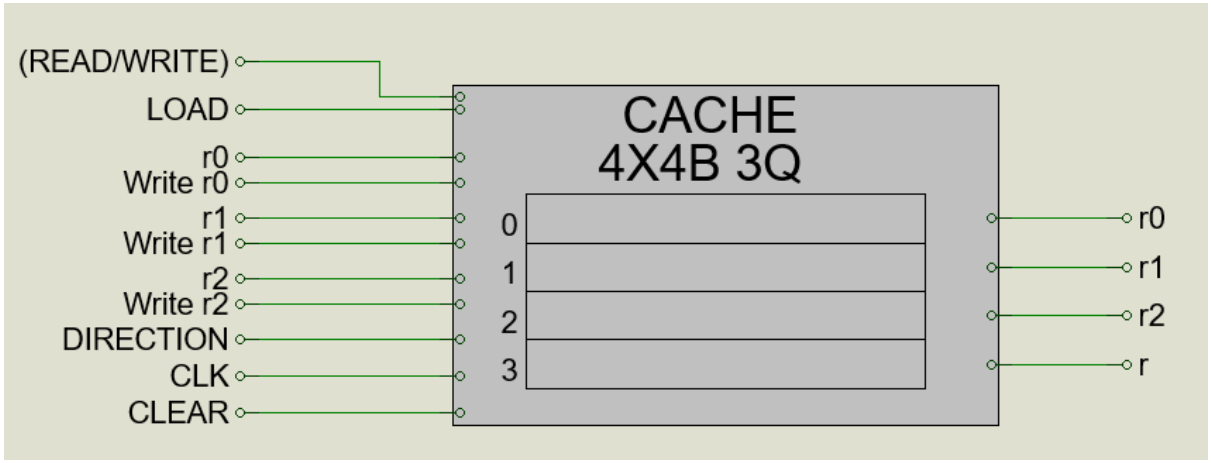
Dr2 Entrada de 4 bits, para guardar un dato en r2.
Wr2: Permite escribir en el registro r2.

Salidas:

Qr0: Dato de 4 bits almacenado en r0.
Qr1: Dato de 4 bits almacenado en r1.
Qr2: Dato de 4 bits almacenado en r2

Q (Q3, Q2, Q1, Q0): Número de 4 bits almacenado actualmente en el registro seleccionado.


OR
OR
OR
OR
OR
OR
OR
OR
OR
OR
OR
OR
OR
OR
OR
4B Register
4B Register
4B Register
4B Register
4B MUX 2x1
4B MUX 2x1
4B MUX 2x1
AND (3P)
AND (3P)
AND (3P)
AND (3P)
NOT
NOT


(READ/WRITE)
LOAD
r0
Write r0
r1
Write r1
r2
Write r2
DIRECTION
CLK
CLEAR
CACHE
4X4B 3Q
0
1
2
3
r0
r1
r2
r

Cache 8X4B con 3 Quick Access

El Caché de 8 Lineas (Registros) de 4 Bits con 3 Registros de Acceso Rápido, es similar a la RAM 8x4B, solamente que el caché está diseñado para solo almacenar variables, no código, y con la finalidad de poner asignar y obtener varias datos en un mismo pulso de reloj. Por lo que tiene unas entradas directas a r0, r1 y r2, con una entrada extra cada uno para activar o desactivar la escritura del registro, y una salida directa a cada uno de los registros de acceso rápido.

Entradas:

D (D3, D2, D1, D0): Es el conjunto de datos que se desea almacenar en un registro de 4 bits específico.
CLK (Clock): Es la señal de reloj que sincroniza la operación de los registros.
Reset: Resetea el valor guardado a 0 en todos los registros, en el próximo tick de reloj.
Load: Permite guardar el valor D, en un registro específico.
S (S2, S1, S0) (Selector) Dirección del registro a Leer/Escribir.

Dr0: Entrada de 4 bits, para guardar un dato en r0.
Wr0: Permite escribir en el registro r0.

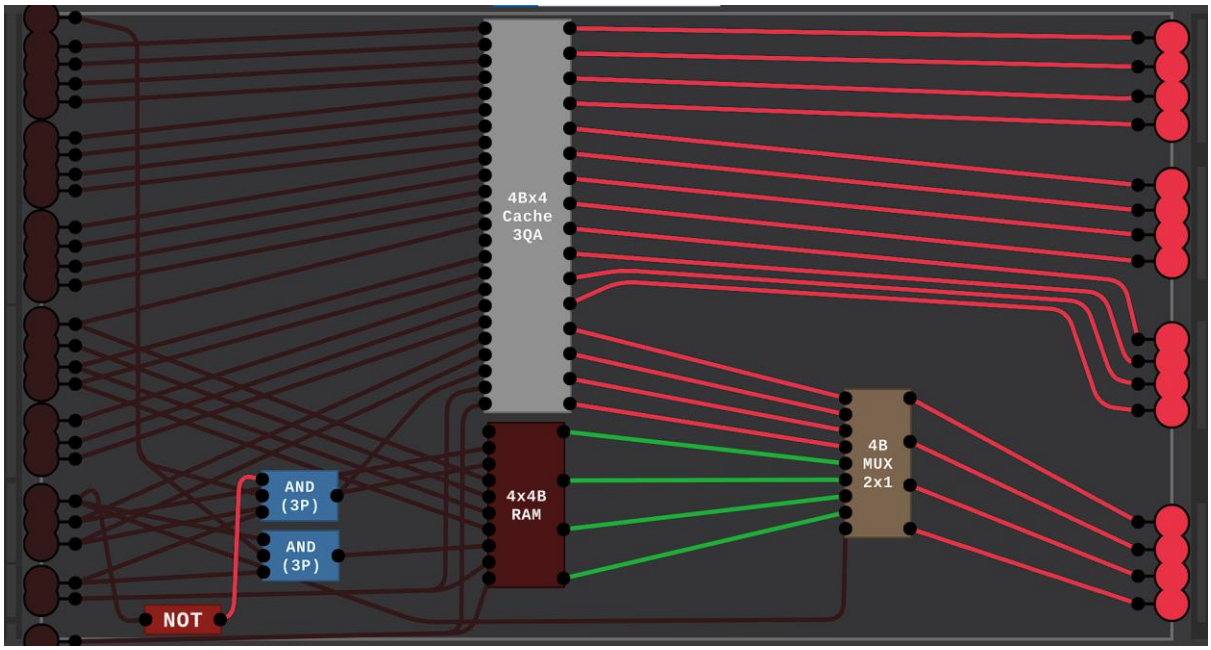
Dr1: Entrada de 4 bits, para guardar un dato en r1.
Wr1: Permite escribir en el registro r1.

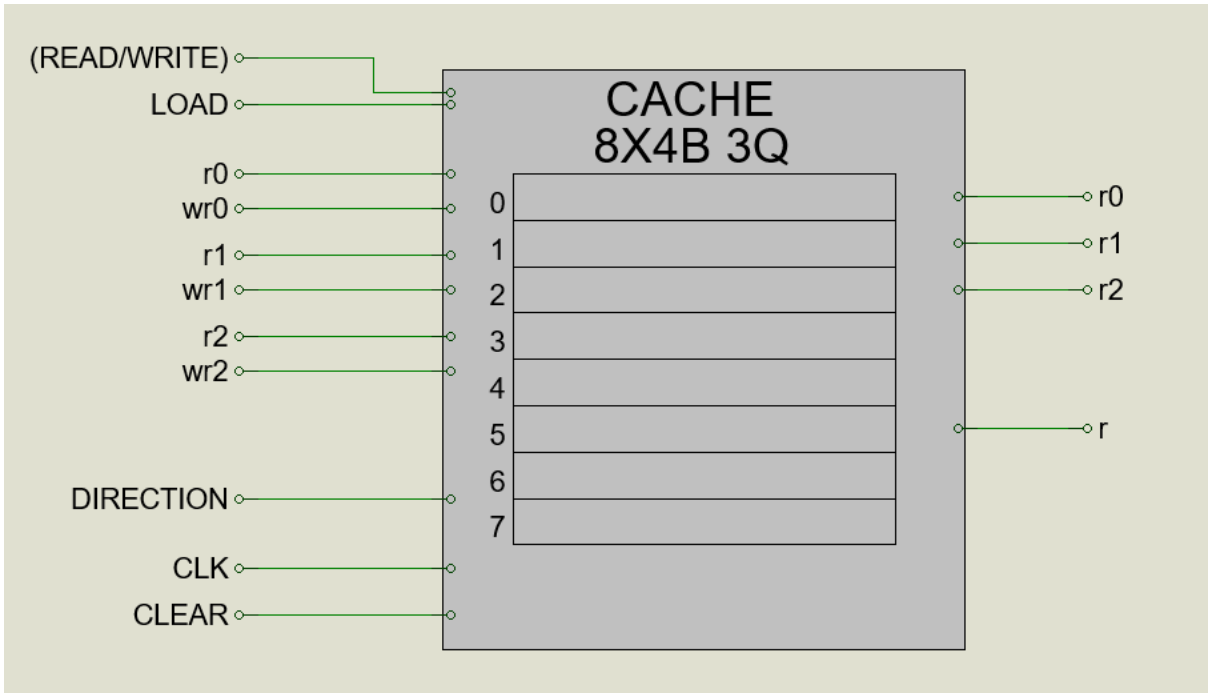
Dr2 Entrada de 4 bits, para guardar un dato en r2.
Wr2: Permite escribir en el registro r2.

Salidas:

Qr0: Dato de 4 bits almacenado en r0.
Qr1: Dato de 4 bits almacenado en r1.
Qr2: Dato de 4 bits almacenado en r2

Q (Q3, Q2, Q1, Q0): Número de 4 bits almacenado actualmente en el registro seleccionado.


4Bx4
Cache
3QA
4x4B
RAM
4B
MUX
2x1
AND
(3P)
AND
(3P)
NOT


(READ/WRITE)
LOAD
r0
wr0
r1
wr1
r2
wr2
DIRECTION
CLK
CLEAR
CACHE
8X4B 3Q
0
1
2
3
4
5
6
7
r0
r1
r2
r

Procesador

Modos de Trabajo

Los modos de trabajo del procesador, se dividen en Procesamiento en tiempo real, Ejecución, Programación y Edición. Estos modos se seleccionan fuera del código del procesador, ya que solo depende de las entradas del usuario.

- **Modo de Procesamiento en tiempo real:** Realiza petición de una línea de código para luego ejecutarla, al terminar y dar su salida, espera por la próxima línea de instrucción, por lo que se procesa línea a línea en tiempo real, lo que el usuario requiere.

- **Modo de Ejecución:** Va leyendo línea a línea el código guardado en la RAM, ejecutando línea a línea las instrucciones programadas, sin hacer uso de alguna entrada del usuario.

- **Modo de Programación**: Ingresa todas las entradas, menos las de selección de modos, en la línea a partir de donde se quedó el contador de línea de la RAM, desde que se paró el conteo la última vez.

- **Modo de Edición**:Cambia el número almacenado en el contador de línea de la RAM.

Construcción del Código Assembly de 3 Segmentos

Para realizar la tabla de funciones primero se debe determinar el tamaño de una línea de código, siendo el tamaño de 4 bits el mínimo para que puedan alcanzar la cantidad deseada de operaciones, luego se le añaden 8 bits a una linea de codigo ya que para realizar las operaciones necesitamos de 0 a 2 datos, por que tenemos que determinar con qué valores se van a realizar las operaciones, así que para cumplir todas las instrucciones, necesitamos un mínimo de 12B para que se puedan comprimir todas las instrucciones.

Por lo antes explicado, el código se divide en 3 segmentos, el comando, la variable A y la variable B, cada uno contando con 4 bits. Lo cual le da el nombre de Lenguaje de 3 Segmentos, o Trees Assembly, debido a que el acrónimo (3s) suena parecido al inglés Trees.

Tablas de Funciones del Procesador

Ya teniendo definida la estructura del código y los modos de trabajo el código se puede adaptar a aplicar distintas funciones en dependencia del modo de trabajo a conveniencia.

Además, hay que tomar en cuenta que las funciones tienen que contribuir para cumplir las condiciones de turing. Por lo que después de un tiempo se llegó al siguiente resultado.

CPU									
Working Mode		Operation Type	M1	M0	Op3	Op2	Op1	Op0	Functio
RUN	In Time Processing	Logic	0	0	0	0	0	0	A AND
			0	0	0	0	0	1	A OR
			0	0	0	0	1	0	NOT(A
			0	0	0	0	1	1	NOT(B
		Arithmetic	0	0	0	1	0	0	A + B
			0	0	0	1	0	1	A - B
			0	0	0	1	1	0	A + 1
			0	0	0	1	1	1	B - A
		Storage	0	0	1	0	0	0	RA = B
			0	0	1	0	0	1	RA = rB
			0	0	1	0	1	0	rA = B
			0	0	1	0	1	1	rA = RB
		Special Op	0	0	1	1	0	0	Read(rA
			0	0	1	1	0	1	Clear Ca
			0	0	1	1	1	0	r0 = A; r1
			0	0	1	1	1	1	Line Count
	Executable	Logic	0	1	0	0	0	0	RA AND
			0	1	0	0	0	1	RA OR
			0	1	0	0	1	0	NOT(RA
			0	1	0	0	1	1	NOT(rB
		Arithmetic	0	1	0	1	0	0	RA + rB
			0	1	0	1	0	1	RA - rB
			0	1	0	1	1	0	RA + 1
			0	1	0	1	1	1	rB - RA
		Storage	0	1	1	0	0	0	RA = B
			0	1	1	0	0	1	RA = rB
			0	1	1	0	1	0	rA = B
			0	1	1	0	1	1	rA = RB
		Special Op	0	1	1	1	0	0	Skip
			0	1	1	1	0	1	Jump to RA if R
			0	1	1	1	1	0	Stop
			0	1	1	1	1	1	Jump to RA if R
Programming	Coding	Code Writing	1	0	X	X	X	X	Write Code (Rcount

	Editing	Change RAM line	1	1	X	X	X	X	Line_Count

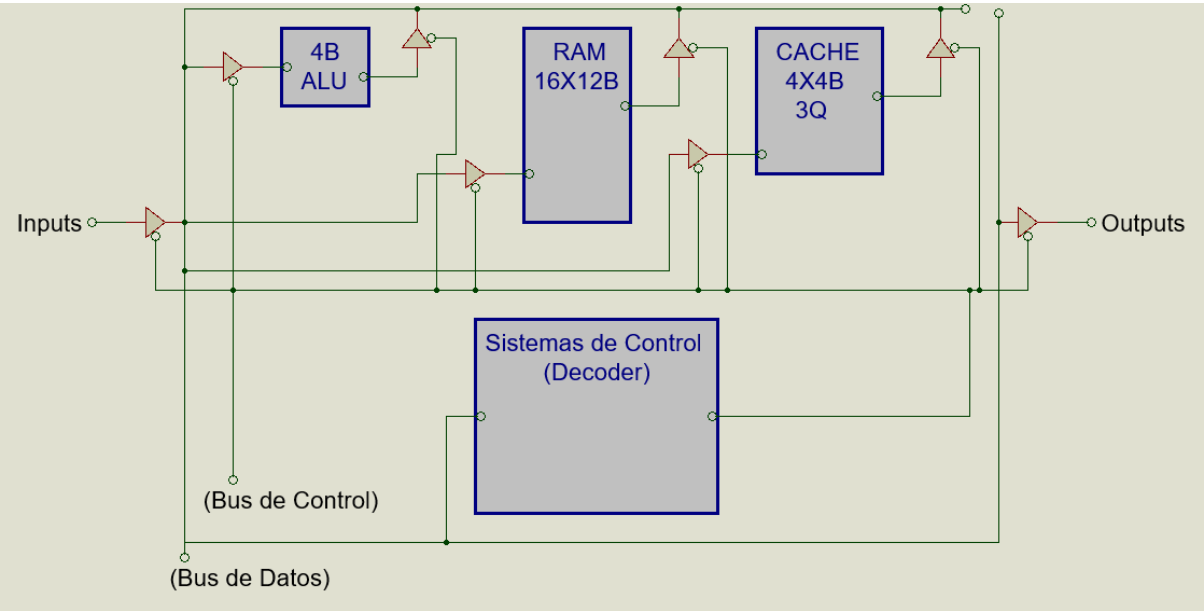
Construcción del Procesado

Para la construcción final del procesador se decidió que todos los componentes, dispusieron de todos los datos del sistema, ya sea las entradas, las salidas del ALU, la RAM y del Cache, por lo que todas las entradas y salidas de los componentes finales irían conectados a un bus de varias líneas, pero para el control del acceso a ciertos datos, el control de los componentes, el ingresos de datos en el bus, y la asignación de datos sacados del bus, se colocaron buffers de 3 estados para que un sistema de control controlado por las entradas o la última línea de código leída de la RAM, dependiendo el código de ejecución, se encargue de dar las instrucciones a cada componente mediante un bus de datos de control, de donde se conectan los selectores del 3 estado de los buffer.

Algunos detalles importantes a mencionar, es que en el caché, registro 2, se guardan las respuestas de las operaciones realizadas, para evitar la pérdida del resultado del procesamiento.

Ejemplo de un Programa, hecho para el procesador, el cual multiplica los datos de del registro 0 y 1, del caché.

```
  # Variable Initialization
[1,0,1,0,  0,0,1,0,  0,0,0,0], # mov  r2,  0          # r2 = 0
[1,0,0,1,  1,1,0,1,  0,0,0,1], # mov  R13, r1         # R13 = r1
[1,0,1,0,  0,0,1,1,  0,0,0,1], # mov  r3,  1          # r3 = 1

  # Calculations
[0,1,0,0,  1,1,1,0,  0,0,0,0], # sum  R14, r0         # r2 = R14 + r0
[1,0,0,1,  1,1,1,0,  0,0,1,0], # mov  R14, r2         # R14 = r2
[0,1,0,1,  1,1,0,1,  0,0,1,1], # sub  R13, r3         # r2 = R13 - r3
[1,0,0,1,  1,1,0,1,  0,0,1,0], # mov  R13, r2         # R13 = r2

  # Do while
[1,1,0,1,  0,0,1,1,  0,0,1,0], # jmp! R3,  r2         # Jump to R3, if(r2 != R15)

  # Result
[1,0,1,1,  0,0,1,0,  1,1,1,0], # mov r2, R14          # r2 = R14
[1,1,1,0,  0,0,0,0,  0,0,0,0], # Stop                 # return void

  # Skip
[1,1,0,0,  0,0,0,0,  0,0,0,0], # int  0
[1,1,0,0,  0,0,0,0,  0,0,0,0], # int  0
[1,1,0,0,  0,0,0,0,  0,0,0,0], # int  0

  # Variables
[1,1,0,0,  0,0,0,0,  0,0,0,0], # int  0               # Down Counter
[1,1,0,0,  0,0,0,0,  0,0,0,0], # int  0               # Mult
[1,1,0,0,  0,0,0,0,  0,0,0,0]  # int  0               # Jump Condition
```


4B
ALU
RAM
16X12B
CACHE
4X4B
3Q
Inputs
Outputs
Sistemas de Control
(Decoder)
(Bus de Control)
(Bus de Datos)

Conclusión

En conclusión, se logró diseñar mediante una serie de pasos el ALU, la RAM y el Caché, en una arquitectura de 4 bits, luego se diseñó un control de todo los componentes para crear el procesador, y en el camino desarrollando un lenguaje de programación, diseñado específicamente para dicho procesador, cumpliendo con la condiciones de turing, ya que el procesador logró realizar operaciones aritmética, lógicas, de manipulación de memoria, logró realizar la carga y programación de programas y el uso de entradas, por lo que es un procesador turing completo, aunque bastante básico.

Entre las mejoras que se le podrían hacer al procesador están:

- Aumentar la listas de instrucciones nativas del procesador.
- Implementar un sistema de manejo de excepciones.
- Añadir una unidad de memoria no volátil.
- Implementar un sistema para trabajar de mejor manera con distintos tipo de datos, y con mayor cantidad de ellos, como adicción del tipo de dato de punto flotante
- Aumento de la capacidad de la memoria principal (RAM) y el Caché, así como agregar un dual channel a la primera, para que pueda trabajar con 2 datos en un mismo pulso de reloj.
- Aumentar los bit de la arquitectura del procesador, para mejorar su rendimiento, tener la capacidad de poder trabajar con números más grandes.
- Implementar la capacidad de poder trabajar con estructuras de datos como matrices.
- Realizar un sistema para poder trabajar con cálculos gráficos
- Y pensando de una manera ambiciosa, la implementación de varios núcleos e hilos, así como la creación de un lenguaje de mayor nivel que el previo.

Gracias al desarrollo del presente proyecto se logró desarrollar una gran cantidad de habilidades en relación a la electrónica digital y a la estructura de computadoras, logrando recopilar información y habilidades, de manera eficiente y poniendo en práctica lo aprendido, por lo que fue de gran ayuda para nuestro desarrollo académico.

Bibliografía

- Electronics Tutorials. (s.f.). Complemento a dos (Two's complement). https://www.electronics-tutorials.ws/logic/logic_9.html
- Wikipedia. (s.f.). Two's complement. https://en.wikipedia.org/wiki/Two%27s_complement#:~:text=A%20two's%2Dcomplement%20number%20system,the%20corresponding%20power%20of%20two.
- GeeksforGeeks. (s.f.). Full Adder in Digital Logic. https://www.geeksforgeeks.org/full-adder-in-digital-logic/
- GeeksforGeeks. (s.f.). 4-bit Binary Adder-Subtractor. https://www.geeksforgeeks.org/4-bit-binary-adder-subtractor/
- Wikipedia. (s.f.). Multiplexor. https://es.wikipedia.org/wiki/Multiplexor#:~:text=En%20el%20campo%20de%20la,pueden%20comunicarse%20al%20mismo%20tiempo.
- WitsCAD. (s.f.). Arithmetic Logic Unit Design. https://witscad.com/course/computer-architecture/chapter/arithmetic-logic-unit-design
- Virtual Labs, IIT Kharagpur. (s.f.). Computer Organization & Architecture Lab: Experiment 9. http://vlabs.iitkgp.ac.in/coa/exp9/index.html
- ExamRadar. (s.f.). Processor Design. https://examradar.com/processor-design/
- Wikipedia. (s.f.). Turing completo. https://es.wikipedia.org/wiki/Turing_completo