Proyecto Final AyOC

MIC-1, Grupo 12

Diseño, implementación y documentación del la MIC-1, una microarquitectura creada por Andrew S. Tanenbaum.

2025

Universidad Nacional de Tucuman - FACET

07/08/2025

****

# Integrantes:

* Moyano Mario
* Morales Soria Santiago
* Yusef Alvaro Alejandro
* Gonzalez Alejo Baltazar
* Godoy Cesar Jose

**Índice**

[Integrantes: 2](#_Toc205506049)

[Introducción 3](#_Toc205506050)

[Camino de Datos 4](#_Toc205506051)

[Registros: 5](#_Toc205506052)

[Unidad Aritmética Lógica (ALU) 7](#_Toc205506053)

[Desplazador: 10](#_Toc205506054)

[Bus de Datos: 11](#_Toc205506055)

[Unidad de Control 12](#_Toc205506056)

[AlmacéndeControl: 13](#_Toc205506057)

[MPC: 14](#_Toc205506058)

[MIR: 15](#_Toc205506059)

[Microprograma: 17](#_Toc205506060)

[Decodificador: 17](#_Toc205506061)

[Conclusión 18](#_Toc205506062)

**La Microarquitectura MIC-1**

# Introducción

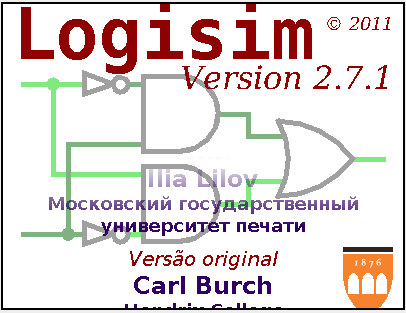
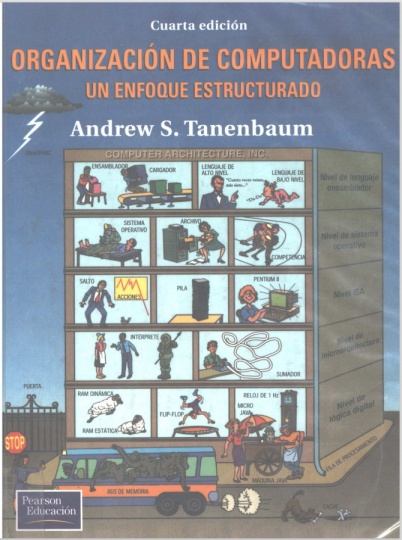
Este proyecto final para la materia Arquitectura y Organización de la Computadora 1 se centra en la implementación y simulación de la MIC-1, una microarquitectura diseñada por Andrew S. Tanenbaum con propósitos educativos, enseñando los conceptos fundamentales de arquitectura de computadoras y diseño de procesadores, presentada en su libro “Organización de Computadoras: Un Enfoque Estructurado”.

La MIC-1 posee una ISA que consiste en una versión simplificada de la *Java Virtual*

*Machine* (JVM), enfocada específicamente en números enteros, conocida como *Integer Java Virtual Machine* (IJVM).

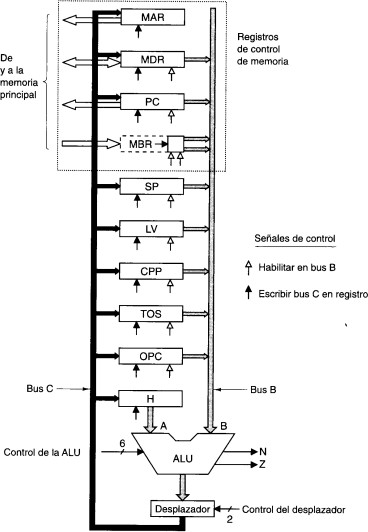
Esta microarquitectura, como se detalla más adelante en este documento, puede separarse en dos grandes partes conceptuales: el Camino de Datos, donde profundizaremos en sus 4 componentes, sus funcionamientos y cómo están conectados; y la Unidad de Control, donde veremos qué rol cumple dentro de la MIC-1.

En este caso para realizar los circuitos de la MIC-1 se utilizó Logisim, un programa desarrollado por Carl Burch, el cual nos permite diseñar y simular circuitos lógicos digitales con una interfaz gráfica sencilla, pudiendo construir y probar el modelo de microarquitectura de una manera interactiva.



# Camino de Datos

El Camino de Datos (Data Path) es el núcleo de la MIC-1 y está diseñado para permitir la ejecución de operaciones aritméticas y lógicas necesarias, controlar que operaciones se van a ejecutar, transportar datos, a fines de ejecutar programas, y como se puede observar en la Fig.1 se compone de:

* Varios registros, donde algunos de ellos tienen un uso general y otros más específicos.
* Una Unidad Aritmética Lógica (ALU).
* Un Desplazador (Shifter).
* Buses de datos.

**Figura1.** Camino de datos de la MIC-1.

## Registros:

Los registros almacenan los datos e instrucciones a utilizar durante la ejecución, cada uno tiene un tamaño de 32 bits y mediante dos líneas de control (una que habilita el bus B y otra el C) y de circuitos *D flip-flops*(*Delay*) logran almacenar datos binarios y direcciones de memoria. Los mismos se encuentran conectados a través de buses de datos para permitir transferencias entre registros y la ALUA.

Entre los registros mas destacables encontramos el PC (Program Counter), el cual contiene la dirección de la próxima instrucción a ser ejecutada; el *IR* o *MBR*, que se encarga de retener la instrucción que está siendo ejecutada, y unos registros de entrada/salida, el *MAR* y el *MDR*, los cuales generalmente se usan para guardar temporalmente datos que fueron leídos de memoria o que tienen que ser escritos en memoria junto a sus direcciones.

Casi todos los registros pueden colocar su contenido en el bus B.

Además los registros implementados en Logisim, tienen la particularidad de contar con dos funciones las cuales se acceden a través de *pin* como se observa en la Fig. 2-a y 2-b. Estas

Son capaces de mantener la información usando su método *Hold*o de cargarla misma usando

*Load*.

Los registros de 32 bits (previstos de antemano) están compuestos por 32 registros de 1 bit, aunque también es posible utilizar 8 registros de 4 bit.  
Los registros de 8 bits (elaboración del grupo) están compuestos por 2 registros de 4 bits, y por último los de 4 bits se componen por 4 registros de 1 bit.

|  |  |
| --- | --- |
| Circuito | Representacion (Logisim) |
| Registro (32 bits) | C:\Users\Zonby\AppData\Local\Microsoft\Windows\INetCache\Content.Word\r32.png |
| Registro (8 bits) | C:\Users\Zonby\AppData\Local\Microsoft\Windows\INetCache\Content.Word\r8b.png |
| Registro (4 bits) |  |
| Flip Flip (Flanco Desc)  D-Latch |  |

## Unidad Aritmética Lógica (ALU)

La Unidad Aritmética Lógica o ALU (**Arithmetic Logic Unit**) es un componente fundamental en los procesadores, y como su nombre lo indica, su principal función dentro del Camino de Datos es la de realizar operaciones aritméticas y lógicas sobre los datos que le van llegando.

Para realizar dicha función, la ALU se divide en varios circuitos combinacionales para realizar las operaciones de **suma, disyunción (OR), conjunción (AND) y negación (NOT)**, pero para seleccionar la operación a realizar utiliza un decodificador.   
En nuestro proyecto incorporamos un circuito llamado **ZN** cuya función es expresar a través de sus salidas las señales de bandera **negativo (N)** y **cero (Z)** según el resultado.

|  |  |  |
| --- | --- | --- |
| Operación | Circuito | Representacion (Logisim) |
| Suma | Sumador completo (Full Adder) |  |
| AND  OR  NOT | Unidad Logica (Logic Unit) |  |
| Cero  (Z)  Negativo (N) | ZN |  |

.Como la MIC-1 es de **32 bits**, tuvimos que construir la ALU de 32 bits juntando **4 ALUs de 8 bits**, compuestas cada una con **8 ALUs de 1 bit**, donde la salida de la siguiente es la entrada de la anterior (acarreo incluido) como se puede apreciar en la siguiente tabla.

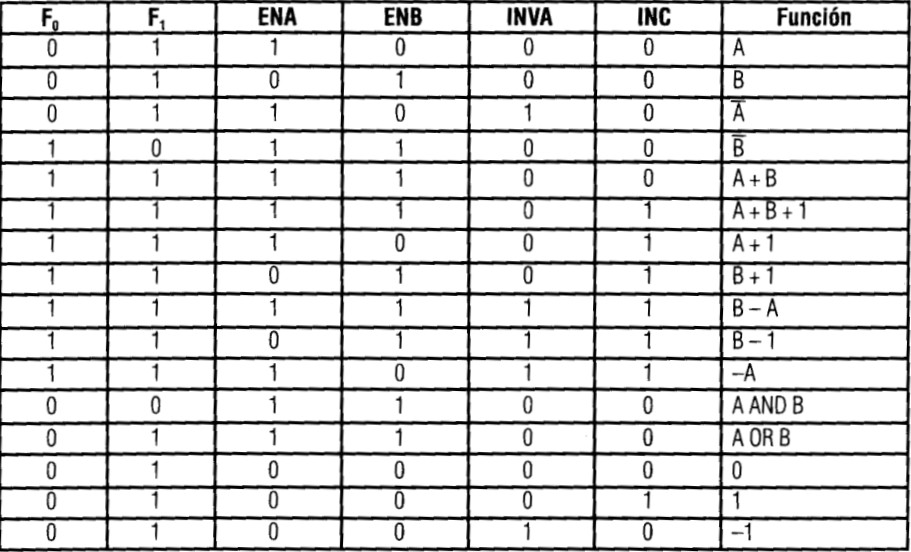
|  |  |
| --- | --- |
| Circuito | Representacion (Logisim) |
| ALU (1 bit) | D:\BACKUP\Uni\UNT\2do_anio\Arquitectura de computadoras\Proyecto\alu 1b.png |
| ALU (8 bits) | D:\BACKUP\Uni\UNT\2do_anio\Arquitectura de computadoras\Proyecto\alu 8b.png |
| ALU (32 bits) | D:\BACKUP\Uni\UNT\2do_anio\Arquitectura de computadoras\Proyecto\alu 32b.png |

La **Unidad de Control (Control Unit)** le dice a la ALU que operación debe realizar y con que operandos. Si detallamos su estructura y funcionamiento, podemos decir que cuenta con dos entradas **A** y **B**, la primera conectada al registro de retención **H**, y la segunda directamente al **bus B**. Por último, la salida de la ALU se encarga de alimentar la entrada del **Desplazador (Shifter)**.

Además, cuenta con:

* Dos salidas para las banderas:
  + **N** (negatia)
  + **Z** (cero)
* Y seis líneas de control:
  + **F0 y F1**: señales de control que se encuentran conectadas al decodificador y de cuyos valores depende la selección de la operación a realizar.
  + **ENA y ENB**: las cuales sirven para habilitar individualmente las salidas.
  + **INVA**: para invertir lógicamente la entrada A.
  + **INC**: para forzar un acarreo de entrada en el bit menos significativo.

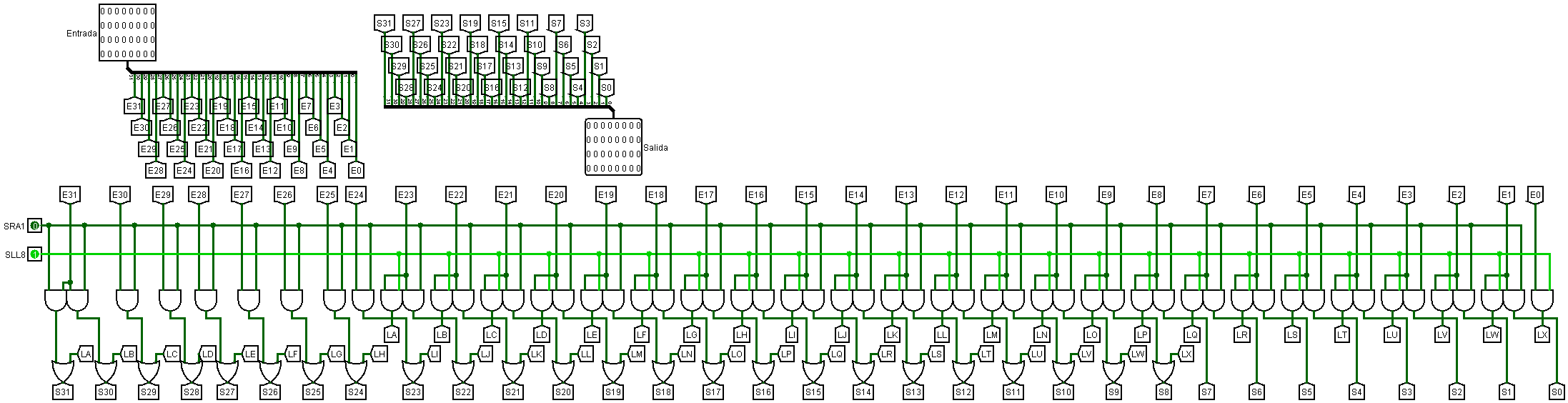
Entonces, una vez que tenemos el registro de retención H como entrada A, a continuación se selecciona un registro como entrada B, su valor se coloca en el Bus B al principio del ciclo y se mantiene ahí durante todo el ciclo. Luego, la UAL efectúa su trabajo y produce un resultado que se pasa al Bus C luego de pasar a través del desplazador. Cerca del final del ciclo, cuando las salidas de la ALU y el desplazador se encuentran estables, una señal de reloj activa el almacenamiento del Bus C en uno más de los registros.

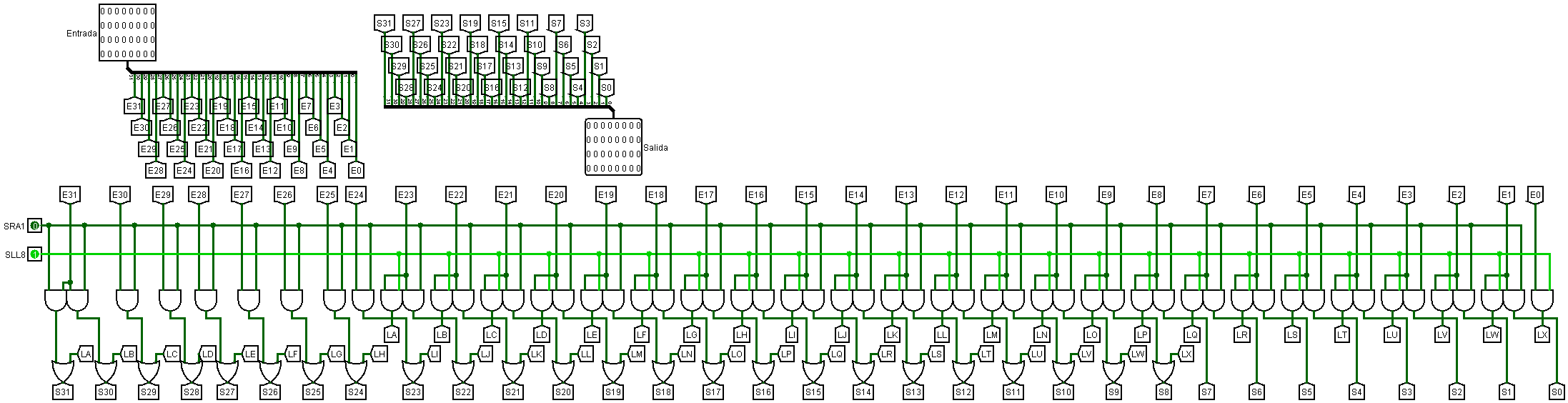
Tabla de verdad de las Funciones de la ALU.

## Desplazador:

El desplazador (**Shifter**) es un componente especializado que contiene tanto una entrada como una salida de 32 *bits*. La entrada llega desde la ALU y la salida se conecta directamente al Bus C.  
Su función es realizar el desplazamiento (lógico o aritmético) de bits de los datos almacenados en los registros y para eso tiene dos señales de control:

* **SLL8 (Desplazamiento lógico a la izquierda):** Desplaza el contenido un byte a la izquierda, convierte además los 8 bits más significativos en ceros.
* **SR1 (Desplazamiento aritmético a la derecha):** Desplaza el contenido un bit a la derecha, convirtiendo el bit menos significativo en 0 y el más significativo según el signo.

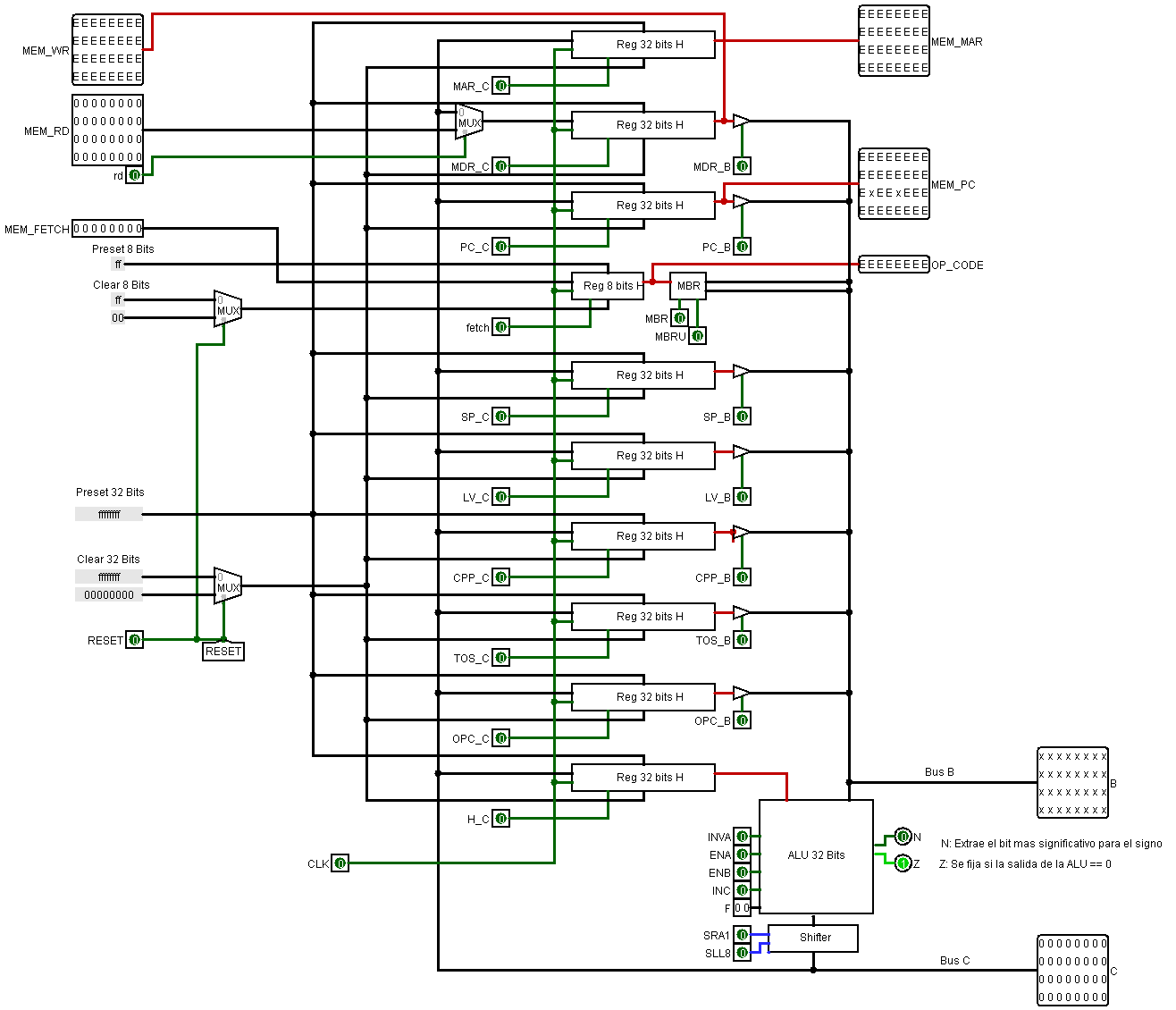




## Bus de Datos:

Es un canal compartido que permite la transferencia de datos tales como operandos e instrucciones entre todos los componentes del Camino de Datos (Data Path).

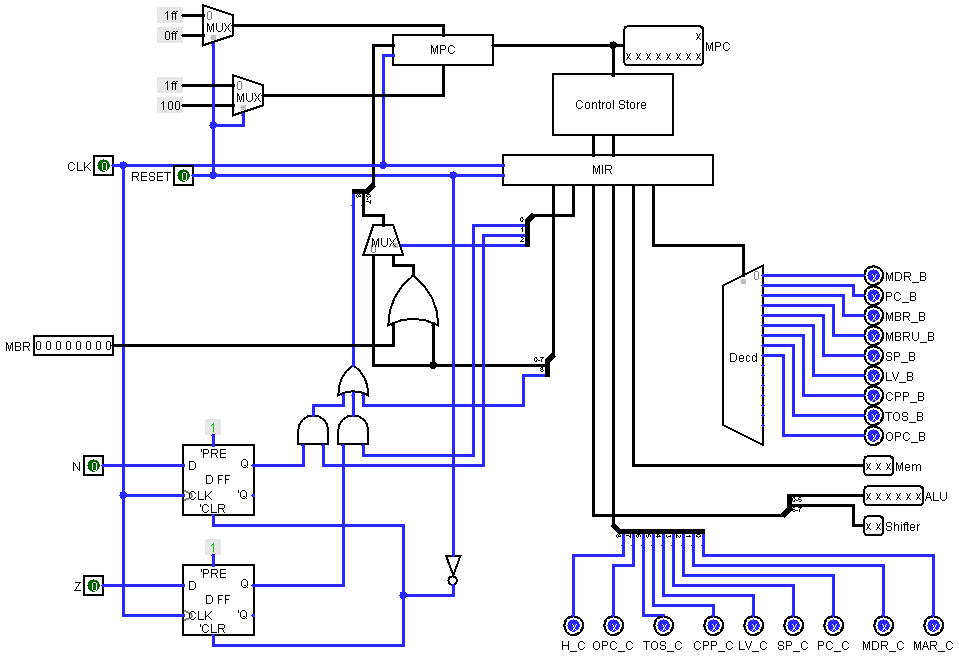
Los buses A, B y C (de 32 bits cada uno) conectan la ALU con varios registros dentro del Data Path como PC, MAR, H, con la ALU. Para realizar su funcionamiento, se utilizan señales de multiplexado, decodificadores y buferes para seleccionar tanto el origen como el destino de los datos e instrucciones, dichas señales son enviadas desde la unidad de control.

  
Data Path

# Unidad de Control

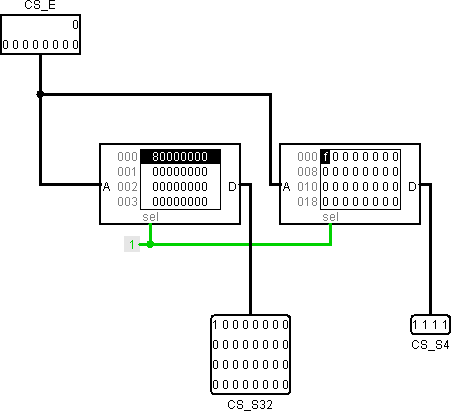
La unidad de control (Control Unit) gestiona y coordina las operaciones que se van a realizar. Para realizar cada instrucción debe generar las señales de control pertinentes y además determinar hacia donde deben llegar. Primero decodifica de la memoria las instrucciones, luego emite las señales de control para manejar a la ALU, registros, memoria, o donde fuese necesario.

Unidad de Control (Control Unit)



## Almacén de Control:

El almacén de control (**Control Store**) es una memoria que contiene microinstrucciones a fines de ejecutar las instrucciones de la **ISA**. De cierta forma es parecida a la memoria principal donde sus instrucciones se ejecutan en orden según su dirección.  
  
Como el almacén de control es prácticamente una memoria, necesita sus propios registros para almacenar tanto la microinstrucción ejecutándose actualmente como la próxima por ser ejecutada, y dichas funciones la realizan el **MPC** (Microprogram Counter para la proxima) y el **MIR** (Microinstruction Register para la actual).



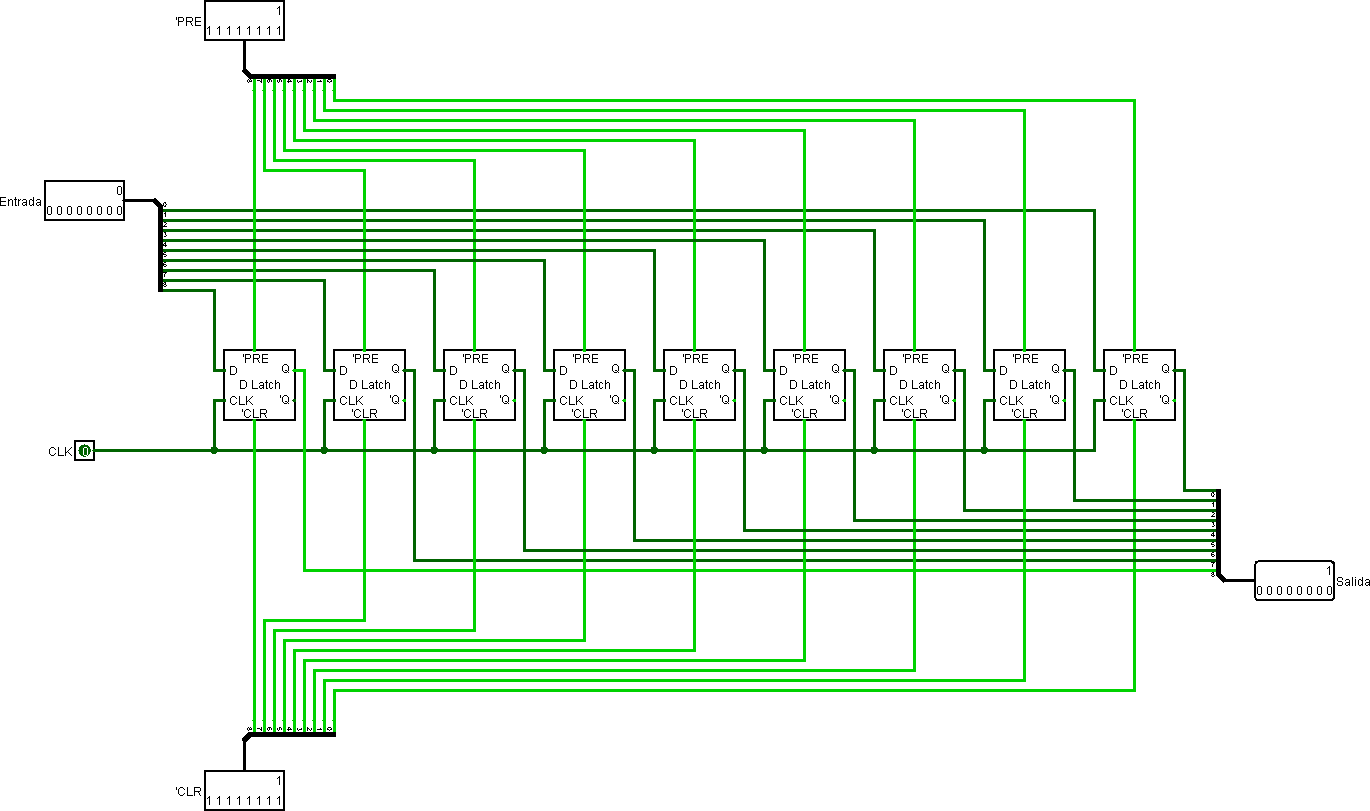
Almacén de Control en Logisim.

## MPC:

Su función principal es mantener la dirección de la próxima microinstrucción a ejecutar en el almacén de control. Cada microinstrucción en el almacén de control tiene una dirección única, y el MPC se encarga de rastrear y seleccionar la siguiente microinstrucción que debe ser ejecutada.

Este puede ser actualizado de diferentes maneras:

* Incrementarse secuencialmente para ejecutar la siguiente microinstrucción en orden.
* Ser cargado con una nueva dirección basada en ciertos campos de la microinstrucción actual, permitiendo saltos dentro del programa de microinstrucción.

MPC en Logisim.

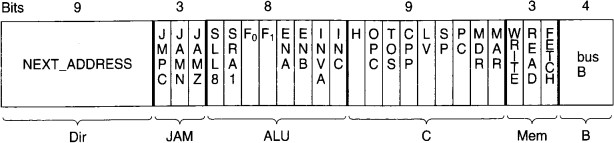
## MIR:

El **Microinstruction Register (MIR)** tiene la función de almacenar la microinstrucción que se encuentra actualmente en ejecución.

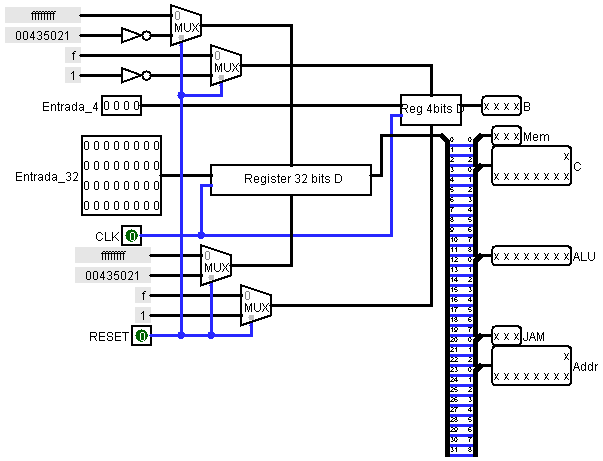
Cuando una microinstrucción es leída desde el almacén de control, se carga en el MIR, y posteriormente su contenido será utilizado por la unidad de control para generar las señales de control necesarias para la ejecución de la microinstrucción.

Cada microinstrucción en el MIR contiene campos para:

* Especificar la **operacion** a realizar en la ALU.
* Determinan **qué** **registros** deben ser leídos o escritos.
* Controlan las señales de **entrada y salida**.
* Indican las **condiciones** para los saltos y la actualización del MPC.

 Formato de las microinstrucciones.

MIR



## Microprograma:

El Microprograma es una secuencia de microinstrucciones almacenadas en el almacén de control, donde cada una de ellas especifica una secuencia de operaciones necesarias para ejecutar una instrucción de máquina. Desempeña un papel vital en la arquitectura pues al traducir las instrucciones de alto nivel del programa a una serie de pasos en bajo nivel permite simplificar la elaboración de programas.

Su funcionamiento consiste en descomponer las instrucciones de máquina en microinstrucciones individuales que controlan las operaciones internas del procesador. Cada instrucción de máquina tiene una secuencia correspondiente en el microprograma. Además, el

microprograma incluye instrucciones para manejar el control de flujo, como saltos

condicionales e incondicionales, utilizando el MPC para seguir la secuencia correcta.

Cadamicroinstruccióncontienecamposquesondecodificadosenseñalesdecontrolporel decodificador. Estas señales dirigen las operaciones de los diferentes componentes del procesador, asegurando que las tareas se realicen en el orden correcto y que los datos se muevan adecuadamente entre los distintos elementos del sistema.

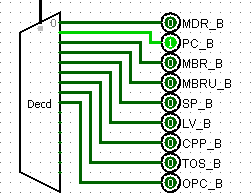
## Decodificador:

El decodificador es un componente crucial dentro de la MIC-1.Las microinstrucciones en el MIR contienen varios campos que representan diferentes operaciones necesarias para la ejecución de una instrucción a nivel de microarquitectura. El decodificador traduce estos

Campos en señales de control específicas que pueden ser entendidas y ejecutadas por los componentes de la MIC-1. Estas señales de control se utilizan para activar y desactivar componentes como la UAL, los registros, la memoria y los buses.

Por ejemplo, las señales de control pueden indicar qué operación debe realizar la ALU, qué registros deben ser leídos o escritos, o cómo deben ser direccionadas las operaciones de memoria.

El decodificador también ayuda a controlar el flujo de datos dentro de la microarquitectura, asegurando que las operaciones se realicen en el orden correcto y que los datos se muevan adecuadamente entre los diferentes componentes.



**Figura17.**Decodificador dentro de la Unidad de Control, en Logisim.

# Conclusión

Porúltimoyparadarcierre,consideramosqueesnecesarioresumirtodoloquehemos aprendido en este proyecto, donde exploramos a detalle todos los componentes y sus

funciones dentro de la microarquitectura MIC-1.

Como dijimos al comienzo, la MIC-1 se divide conceptualmente en dos grandes partes: El Camino de Datos, el núcleo, donde se producen las operaciones aritméticas-lógicas a través delaUAL,yseguardanlosdatosmediantelosdiversosRegistrosconfuncionescomo*Load* y*Hold*,pasandotodoestoatravésdelDesplazadorpudiendohacerusodesucapacidadpara realizar un desplazamiento lógico o aritmético, llevando así la información a su salida para uno de los dos Buses, los cuales se encargan de transportar datos binarios y direcciones de memoria.Ylasegundagranparteconceptual,laUnidaddeControl,seencargademarcarun ritmo para coordinar y gestionar el cuándo y en qué orden se van a realizar las distintas

operacionesalmacenadascomosecuenciasdemicroinstruccionesenelMicroprograma,quea su vez se encuentra en el Almacén de Control, el cual es funcionalmente una memoria y por lo mismo hace uso de dos registros especializados: el MPC, encargado de contener la dirección de memoria de la próxima microinstrucción a ejecutar; y el MIR, el cual almacena

lamicroinstruccionesqueseencuentraenejecución,endondeelcontenidofuncionacomola alimentación del Decodificador, el cual traduce el contenido en señales de control.

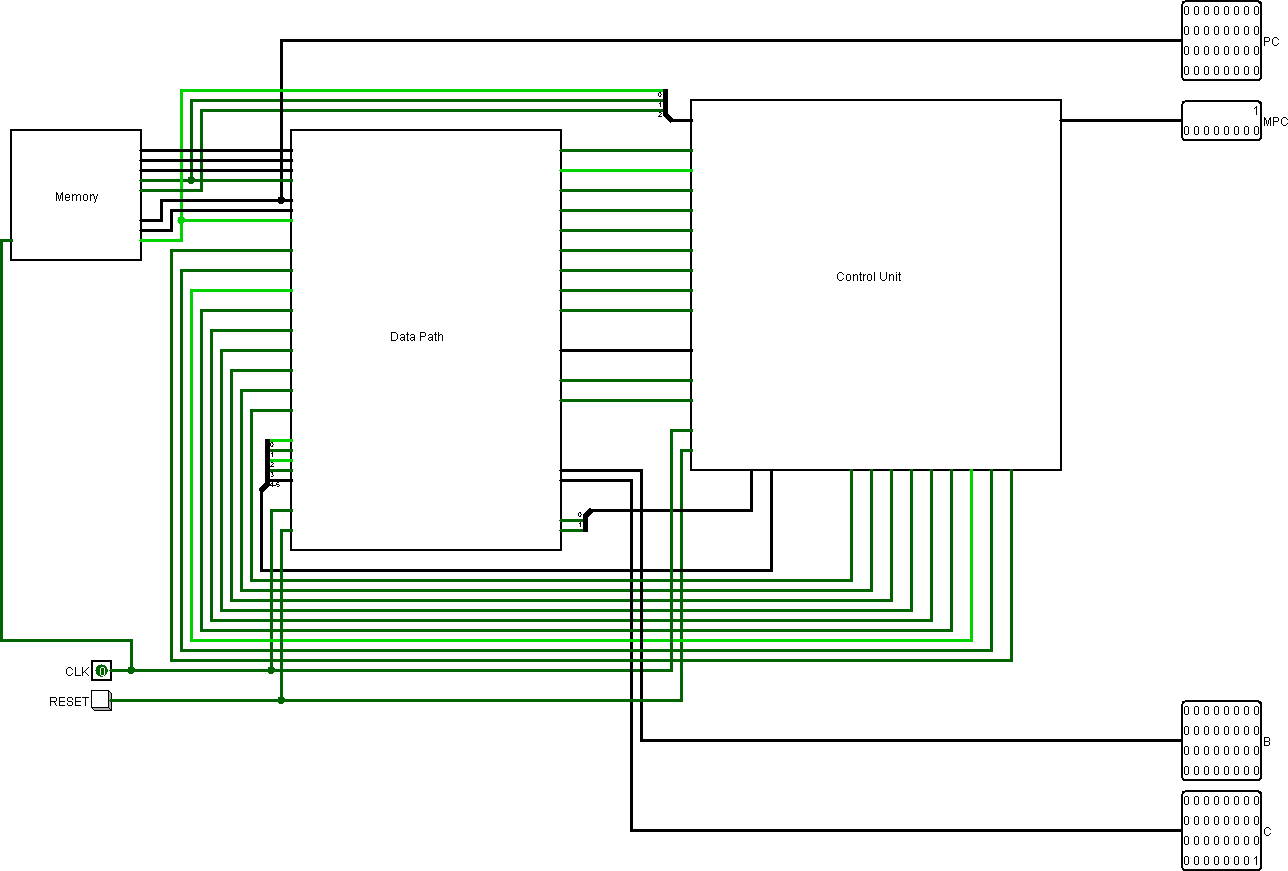
Además, a lo largo del proyecto entendimos cómo las instrucciones de alto nivel son desglosadas en operaciones relativamente más sencillas, lo que permite descomponer

operacionescomplejasenpasosmássimplesqueunaCPUpuedeejecutar,yenconjuntocon el uso del programa Logisim y su interfaz sencilla, fuimos capaces de comprender lo visto

durantelo largode la asignatura,de unamanera completa, didáctica,interactiva y visual.

En resumen, este proyecto nos ha permitido desglosar y comprender los componentes y procesos esenciales de la microarquitectura MIC-1, proporcionando así una valiosa

experienciasobrelaArquitectura yOrganizaciónde Computadoras.



**Figura18.**MIC-1enLogisim.