Proyecto Final 2024

Asignatura: Arquitectura y Organización de Computadoras

Institución: Facultad de Ciencias Exactas y Tecnologías

Carreras: Ingeniería Informática y Programador Universitario

Integrantes:

- Brito, Juan Benjamín César
- Díaz Romero, Guillermo Miguel
- Ibarra, Martín Ezequiel
- Muñoz, Ignacio Tadeo
- Rivera Octtaviano, Ramiro Tomás

Índice

Introducción....2
¿Qué es la MIC-1?.... 3
Importancia del Aprendizaje.... 3
Herramienta de Simulación (Logisim)....3
Estructura del documento....4
Camino de Datos....4
Componentes del Camino de Datos....4
1. Unidad aritmética lógica(ALU)....4
2. Registros de Propósito General....7
3. Bus de Datos.... 9
4. Bus de Direcciones....10
5. Program Counter(PC).... 11
6. Registro de Instrucción (IR-Instruction Register)....13
7. Memoria RAM....14

Unidad de Control.... 16
Componentes de la Unidad de Control.... 16
MIR (Microinstruction Register - Registro de Microinstrucción)....16
MPC (Microprogram Counter - Contador de Microprograma).... 17
Decoder (Decodificador)....17
Microprograma....17
Conclusión....18
Bibliografía....19

Introducción

El proyecto propuesto por la asignatura Arquitectura y Organización de Computadoras I tiene el propósito de abarcar todos los conocimientos dados durante el primer cuatrimestre del ciclo lectivo 2024 con el fin de poder simular el funcionamiento y comportamiento real de la arquitectura de una MIC-1. Por lo tanto, en el transcurso de este documento se verá a profundidad la composición, estructura y funcionamiento de cada parte/componente de la MIC-1 mediante el uso del programa “Logisim”.

¿Qué es la MIC-1?

La **MIC-1** (**M**icroprogrammed **I**mplementation of a **C**omputer) es una arquitectura de computadora simple, desarrollada por Andrew Stuart “Andy”, para entender los principios de la arquitectura de computadoras y de la programación en ensamblador y microprogramación.

La idea se basa en usar microprogramas para implementar las instrucciones de nivel superior. En lugar de tener circuitos de controles fijos y dedicados para cada instrucción, la MIC-1 usa una memoria de control programable, en donde se encuentran las secuencias de microinstrucciones que definen el comportamiento de cada instrucción.

Los componentes principales son los siguientes (solo una breve descripción, pues, más adelante en el documento se detallará el uso de cada uno):

- **Control Unit (Unidad de Control):** Responsable de leer las microinstrucciones de la Control Store y generar las señales de control adecuadas para ejecutar las instrucciones de la máquina.
- **Datapath (Camino de Datos):** Incluye registros, ALU (Unidad Aritmético-Lógico) y otros componentes que realizan operaciones aritméticas y lógicas.
- **Memoria:** Memoria principal donde se almacenan las instrucciones del programa y los datos.

Importancia del Aprendizaje

La MIC-1 es una herramienta valiosa para aprender sobre la organización y arquitectura de computadores porque:

- Permite ver cómo se implementan las instrucciones a nivel de hardware.
- Introduce conceptos clave como la microprogramación y el pipeline de una manera accesible.
- Facilita la comprensión de cómo interactúan el hardware y el software en una computadora.

Herramienta de Simulación (Logisim)

Logisim es un simulador lógico que permite diseñar y simular circuitos electrónicos digitales mediante una interfaz gráfica de usuario. Logisim, que se encuentra bajo licencia pública GNU, es software libre diseñado para ejecutarse en Microsoft Windows, Mac OS X y Linux entre otras plataformas. Su código está totalmente en Java y usa la biblioteca de interfaz gráfica de usuario Swing. El principal desarrollador es Carl Burch y ha trabajado en Logisim desde su creación en 2001.

Logism permite el uso de cables verticales u horizontales en dos dimensiones. Para ello se usa la herramienta de cableado, la cual con un simple arrastre del ratón puede crear muchos segmentos de cable. A diferencia de otros simuladores de sofisticación del Logisim, Logisim permite al usuario modificar el circuito durante la simulación.

En este caso, utilizaremos la herramienta Logisim en su versión 2.7.1 (última versión hasta el día de la fecha) para poder diseñar cada parte de la MIC-1 y comprobar su funcionamiento correcto mediante la simulación.

Estructura del documento

Este documento estará dividido en varias secciones donde se explicarán paso a paso cómo se irá desarrollando el proceso de diseño y simulación de la MIC-1. A continuación se encuentra el índice del documento.

Camino de Datos

Es una parte fundamental de la arquitectura de cualquier procesador, es el conjunto de componentes y conexiones que permiten la manipulación y transferencia de datos en el sistema. En términos más simples, es el trayecto que sigue la información desde que es tomada de la memoria, pasando por las unidades de procesamiento hasta que finalmente es almacenada o utilizada para alguna operación.

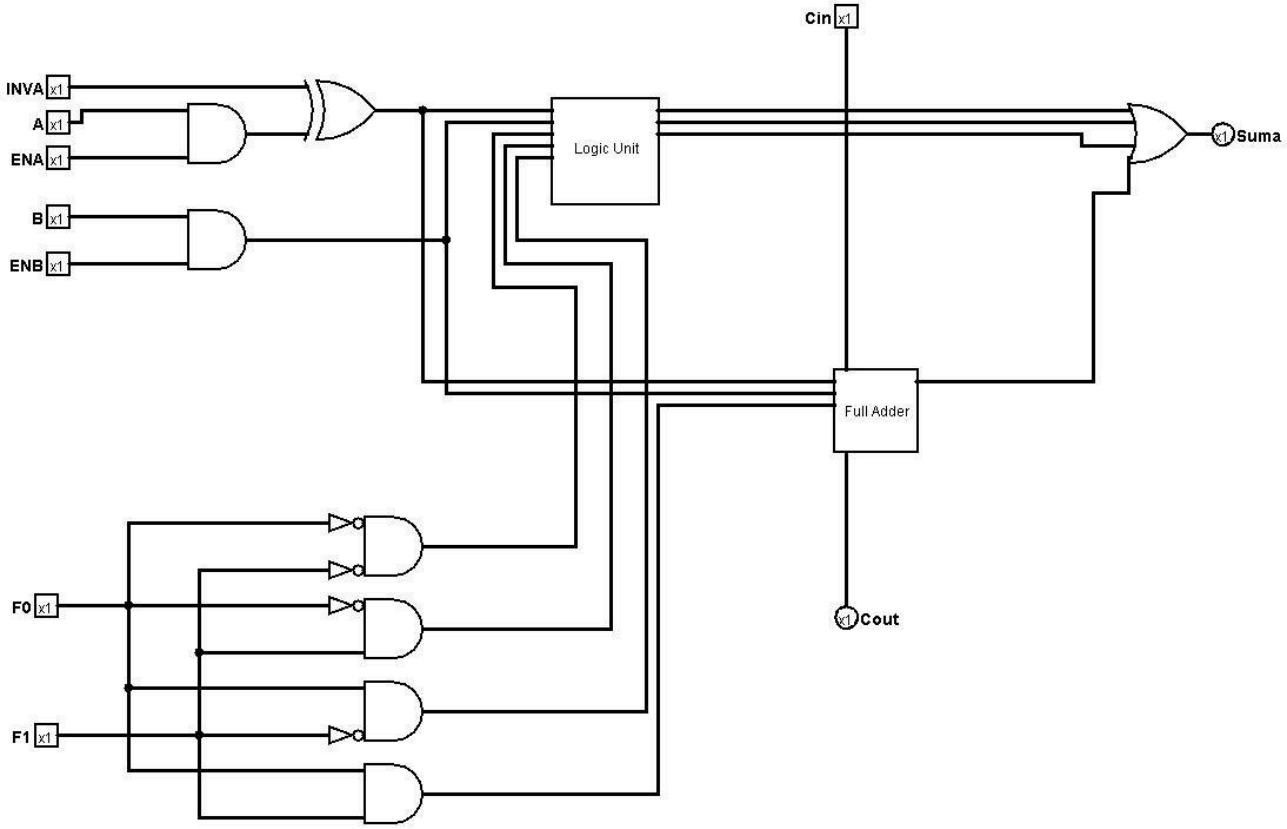
El camino de datos en un procesador, y específicamente en la microarquitectura IJVM, incluye una serie de componentes clave que trabajan juntos para ejecutar las instrucciones del programa. Cada uno de estos componentes tiene una función específica y están interconectados para garantizar que los datos fluyan correctamente y se procesan según las necesidades de cada instrucción.

Componentes del Camino de Datos

1. Unidad aritmética lógica(ALU)

Es uno de los componentes fundamentales dentro del camino de datos en la arquitectura de cualquier procesador. Su principal función es realizar todas las operaciones aritméticas y lógicas necesarias para la ejecución de las instrucciones del programa. En la microarquitectura IJVM, la ALU es central para la ejecución de las instrucciones más básicas y frecuentes, como ADD, SUB, AND, y OR. Sin una ALU, el procesador no podría realizar ninguna operación sobre los datos, haciendo imposible la ejecución de cualquier programa. La eficiencia y diseño de la ALU impactan directamente en el rendimiento del procesador.

Ejemplo de una ALU de 1 bit:



Funciones de la ALU

La ALU es responsable de ejecutar las siguientes operaciones principales:

- **Operaciones aritméticas: Suma y resta:** Son las operaciones más comunes de la ALU, calculan direcciones de memorias, manejan contadores y procesan datos numéricos.
- **Operaciones Lógicas: AND, OR, XOR, NOT:** Estas operaciones manipulan y analizan bits individuales dentro de una palabra de datos. Son fundamentales para operaciones como comparaciones y configuraciones de banderas (flags) en el procesador.
- **Operaciones de Desplazamiento (Shifts): Desplazamientos a la izquierda y a la derecha (Shift Left, Shift Right):** Estas operaciones desplazan los bits de una palabra a la izquierda o a la derecha, lo cual es equivalente a multiplicar o dividir por potencias de dos, respectivamente.

Estructura interna de la ALU

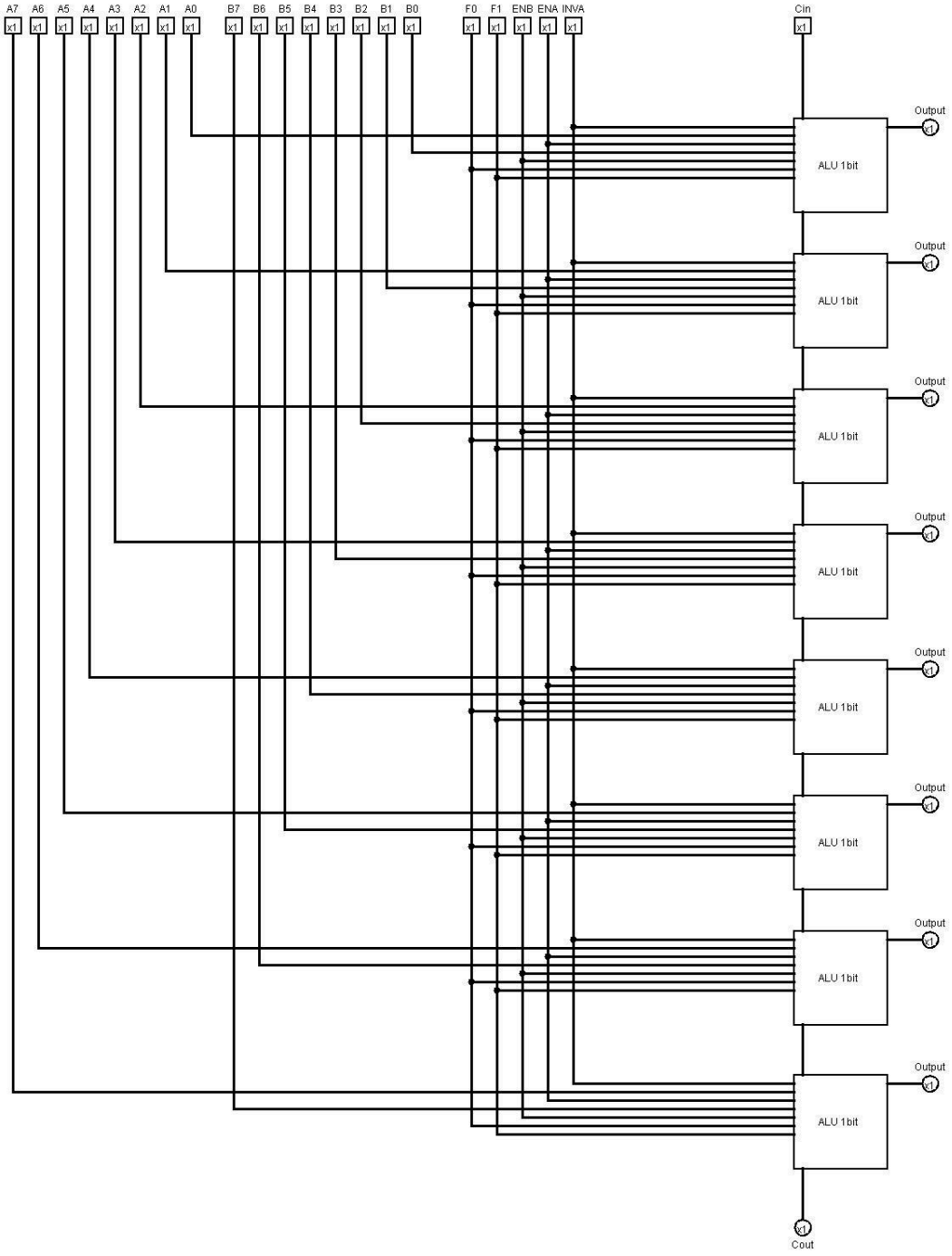
La ALU está compuesta por una serie de circuitos lógicos y aritméticos para llevar a cabo las operaciones ya mencionadas. Estos circuitos incluyen:

- **Sumadores/Restadores: Sumador de Ripple Carry:** Un diseño común en ALUS básicas es el sumador de ripple carry, que permite

sumar dos números binarios bit por bit, propagando un acarreo (carry) de un bit al siguiente.

- **<u>Circuitos Lógicos</u>: Puertas AND, OR, XOR, y NOT**; Estas puertas lógicas permiten que la ALU ejecute operaciones lógicas sobre los datos binarios. Cada puerta realiza una función específica sobre los bits de entrada para generar la salida deseada.
- **<u>Circuitos de Desplazamiento</u>: Desplazadores (Shifters):** Los desplazadores son circuitos especializados dentro de la ALU que permiten desplazar los bits de una palabra a la izquierda o derecha, moviendo bits y rellenando con ceros donde sea necesario.

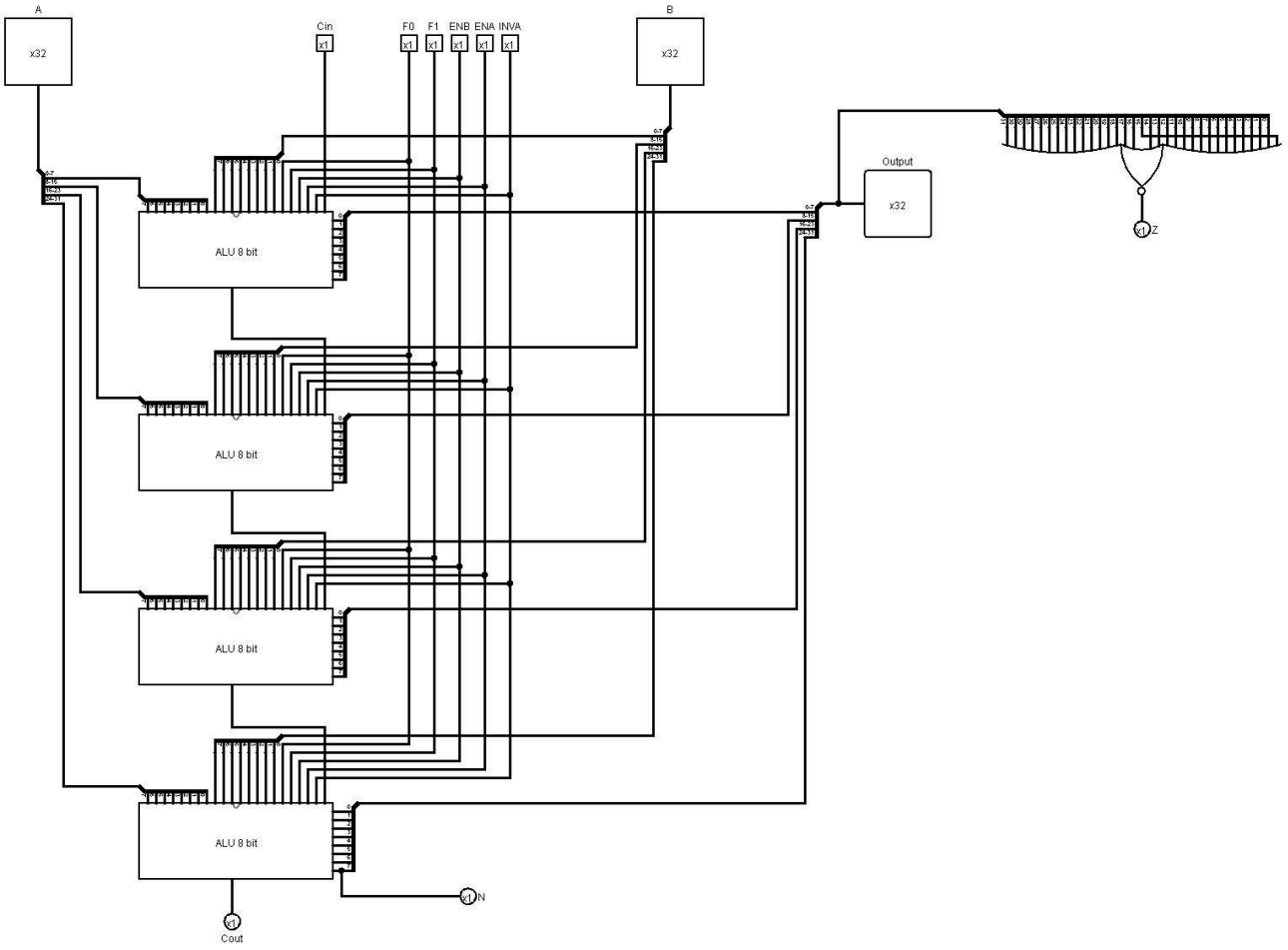
Ejemplo con una ALU de 8 bits:



Interacción con Otros Componentes:

- **Entrada y Salida de la ALU:** La ALU recibe sus operandos de los registros a través del bus de datos. Después de realizar la operación, la ALU envía el resultado de vuelta a un registro o directamente a la memoria, dependiendo de la instrucción ejecutada.
- **Control de Operaciones:** La Unidad de Control del procesador es responsable de enviar las señales adecuadas a la ALU para indicar qué operación debe realizarse. Esto es crucial, ya que la misma ALU puede realizar múltiples operaciones, y la Unidad de Control debe indicar específicamente cuál se requiere en cada momento.

Ejemplo con una ALU de 32 bits (combina 4 ALUs de 8 bits):



2. Registros de Propósito General

Estos registros actúan como almacenamiento temporal para los datos y resultados intermedios durante la ejecución de las instrucciones, permitiendo un acceso rápido y eficiente a los datos que se manipulan en la ALU y otros componentes. Los registros de propósito general cumplen varias funciones críticas en el proceso de ejecución de instrucciones:

Funciones:

- **Almacenamiento Temporal:** Los registros de propósito general se utilizan para almacenar datos temporales que se necesitan para realizar operaciones aritméticas, lógicas, y otras funciones dentro del procesador.
- **Acceso Rápido a Datos:** Los datos almacenados en estos registros pueden ser accedidos y manipulados mucho más rápidamente que los datos almacenados en la memoria principal. Esto es crucial para el rendimiento del procesador, ya que evita los retardos asociados con la lectura y escritura en la memoria RAM.
- **Intermediarios entre la Memoria y la ALU**: Los registros de propósito general sirven como intermediarios entre la memoria y la ALU. Cuando se necesita realizar una operación, los datos se cargan desde la memoria a los registros, la ALU realiza la operación, y luego el resultado puede ser almacenado nuevamente en un registro o en la memoria.

Estructura:

- **Flip-Flops:** Cada bit de un registro está implementado típicamente usando un flip-flop, que es un circuito biestable capaz de almacenar un bit de información. Un registro de 32 bits, por ejemplo, estará compuesto por 32 flip-flops.
- **Tamaño del Registro:** El tamaño de los registros de propósito general (es decir, el número de bits que pueden almacenar) varía según la arquitectura. En la microarquitectura IJVM, los registros típicamente tienen un tamaño de 32 bits, lo que permite almacenar enteros de 32 bits y otras formas de datos.
- **Multiplexores:** Los multiplexores se utilizan para seleccionar cuál de los registros se leerá o escribirá en un momento dado. Esto es controlado por la unidad de control, que genera las señales necesarias para seleccionar el registro adecuado en cada ciclo de instrucción.

Interacción con Otros Componentes:

- **ALU:** Los registros de propósito general alimentan la ALU con los operandos necesarios para las operaciones aritméticas y lógicas. Después de que la ALU realiza una operación, el resultado puede ser almacenado de nuevo en uno de estos registros.
- **Bus de Datos:** Los registros están conectados al bus de datos, lo que permite que los datos sean transferidos entre la memoria, la ALU, y otros registros según sea necesario.
- **Memoria**: Los registros también actúan como intermediarios entre la memoria y la ALU. Por ejemplo, una instrucción podría cargar un valor desde una dirección de memoria a un registro, realizar una operación en la ALU, y luego almacenar el resultado en otra dirección de memoria.

Control de los Registros:

- **Señales de Control:** La unidad de control del procesador genera señales para controlar qué registro será leído o escrito en un momento dado.
- **Banderas (Flags):** Los registros pueden estar asociados con banderas que indican el estado del procesador (por ejemplo, si un resultado es cero o negativo). Estas banderas pueden ser actualizadas automáticamente como resultado de operaciones realizadas en la ALU y almacenadas en registros específicos.

3. Bus de Datos

Permite la transferencia de datos entre los distintos componentes del procesador, como la ALU, los registros de propósito general, y la memoria. Sin el bus de datos, estos componentes no podrían comunicarse eficazmente entre sí, lo que haría imposible la ejecución de cualquier instrucción.

Funciones:

- **Transferencia de Datos:** El bus de datos se encarga de mover los datos entre la memoria, los registros, la ALU, y otros componentes. Cuando se requiere que un dato pase de un componente a otro, este dato viaja por el bus de datos.
- **Comunicación entre Componentes:** Sirve como el medio principal de comunicación dentro del procesador. Sin un bus de datos, cada componente necesitaría conexiones directas con todos los demás, lo que sería impracticable.

Estructura del Bus de Datos:

- **Ancho del Bus:** El ancho del bus de datos es crucial, ya que determina cuántos bits de información pueden ser transferidos simultáneamente. En la microarquitectura IJVM, el bus de datos suele ser de 32 bits, lo que permite la transferencia de una palabra completa de datos en un solo ciclo de reloj.
- **Unidireccional vs. Bidireccional:** Los buses de datos pueden ser unidireccionales (donde los datos solo fluyen en una dirección) o bidireccionales (donde los datos pueden fluir en ambas direcciones). En la mayoría de las arquitecturas modernas, incluyendo la IJVM, el bus de datos es bidireccional, lo que permite tanto lecturas como escrituras de datos.

Interacción con Otros Componentes:

- **Memoria Principal (RAM):** El bus de datos conecta la memoria principal con otros componentes del procesador. Cuando se necesita

leer o escribir un dato en la memoria, este dato se transfiere a través del bus de datos.

- **Registros de Propósito General:** Los registros de propósito general leen y escriben datos a través del bus de datos. Por ejemplo, cuando la ALU realiza una operación, el resultado se puede escribir de vuelta en un registro a través del bus.
- **ALU:** La ALU obtiene sus operandos desde los registros a través del bus de datos y también envía los resultados de las operaciones a los registros o la memoria por el mismo medio.

Control del Bus de Datos:

- **Arbitraje del Bus:** Dado que varios componentes pueden intentar utilizar el bus de datos al mismo tiempo, se necesita un mecanismo para determinar cuál componente tiene prioridad. Este proceso se conoce como "arbitraje del bus". En muchas arquitecturas, la unidad de control gestiona este arbitraje, asegurando que solo un componente utilice el bus de datos a la vez.
- **Señales de Control:** La unidad de control del procesador emite señales que indican cuándo un componente puede leer o escribir datos en el bus. Estas señales son esenciales para coordinar el flujo de datos y evitar conflictos.

4. Bus de Direcciones

A diferencia del bus de datos, que transporta la información (datos) que se va a procesar, el bus de direcciones transporta las direcciones de memoria donde esos datos se almacenan o desde donde se van a recuperar. Es decir, el bus de direcciones indica "dónde" se deben leer o escribir los datos.

Funciones:

El bus de direcciones tiene la función principal de especificar la ubicación en la memoria o en otros dispositivos donde se realizará una operación de lectura o escritura.

Sus funciones clave incluyen:

- **Especificación de Direcciones de Memoria:** Cuando el procesador necesita acceder a un dato o instrucción en la memoria, utiliza el bus de direcciones para enviar la dirección específica de la ubicación en la memoria RAM donde se encuentra el dato o instrucción.
- **Selección de Dispositivos:** Además de la memoria, otros dispositivos en el sistema pueden estar mapeados a direcciones específicas. El bus de direcciones se utiliza para seleccionar cuál de estos dispositivos está siendo accedido en un momento dado.

Estructura del Bus de Direcciones:

- **Ancho del Bus**: El ancho del bus de direcciones, es decir, el número de líneas o cables que lo componen, determina cuántas ubicaciones diferentes en la memoria se pueden direccionar. En la microarquitectura IJVM, el bus de direcciones suele ser de 16 a 32 bits. Por ejemplo, un bus de 16 bits puede direccionar hasta 2^16 = 65,536 ubicaciones de memoria diferentes.
- **Unidireccionalidad:** A diferencia del bus de datos, que suele ser bidireccional, el bus de direcciones es generalmente unidireccional. Esto significa que las direcciones solo fluyen desde el procesador hacia la memoria o los dispositivos, nunca en la dirección opuesta.

Interacción con Otros Componentes:

- **Memoria Principal (RAM):** Cuando el procesador necesita leer o escribir datos en la memoria, primero coloca la dirección de la ubicación de memoria deseada en el bus de direcciones. La memoria utiliza esta dirección para identificar el lugar específico donde los datos deben ser almacenados o recuperados.
- **Unidad de Control:** La unidad de control del procesador utiliza el bus de direcciones para gestionar el flujo de las operaciones de lectura y escritura, asegurándose de que las direcciones correctas se envíen al bus en el momento adecuado.
- **Bus de Datos:** El bus de direcciones trabaja en conjunto con el bus de datos. Mientras que el bus de direcciones selecciona la ubicación de memoria, el bus de datos transporta los datos que se almacenan o recuperan de esa ubicación.

Control del Bus de Direcciones:

- **Generación de Direcciones:** Las direcciones que se envían a través del bus de direcciones son generadas por el Program Counter (PC) para las instrucciones, o por registros y otros mecanismos de direccionamiento cuando se accede a los datos.
- **Multiplexores:** En algunos diseños, se utilizan multiplexores para seleccionar entre diferentes fuentes de direcciones. Por ejemplo, en ciertos ciclos, la dirección podría venir del contador de programa, mientras que en otros podría provenir de un registro de propósito general.

5. Program Counter(PC)

También conocido como contador de programa es responsable de llevar la cuenta de la posición actual en el flujo de ejecución del programa, apuntando a la siguiente instrucción que debe ser ejecutada.

Funciones:

El Program Counter tiene la función de señalar la dirección de memoria de la próxima instrucción que el procesador debe ejecutar. Esencialmente, controla el orden secuencial en el que se ejecutan las instrucciones, lo cual es fundamental para la operación correcta del programa.

Las funciones principales del PC incluyen:

- **Rastreo de la Instrucción Actual:** El PC contiene la dirección de memoria de la instrucción que el procesador debe ejecutar a continuación. Una vez que la instrucción se ha recuperado, el valor del PC se incrementa para apuntar a la siguiente instrucción en secuencia.
- **Control de Flujo del Programa:** Además de simplemente avanzar secuencialmente, el PC también se puede modificar en ciertas operaciones como saltos o llamadas a subrutinas, lo que permite cambiar el flujo de control del programa.

Estructura del Program Counter:

- **Registro de Dirección:** El PC es esencialmente un registro que almacena una dirección de memoria. Dependiendo de la arquitectura, el ancho de este registro puede variar, pero en el IJVM, típicamente manejaría direcciones de 16 a 32 bits.
- **Actualización Automática:** Tras la ejecución de una instrucción, el PC se incrementa automáticamente, generalmente en una cantidad fija que corresponde al tamaño de una instrucción. Este incremento asegura que el procesador continúe ejecutando instrucciones en orden secuencial.
- **Control Directo por Instrucciones:** En ciertas instrucciones como saltos (JUMP) o llamadas a subrutinas (CALL), el valor del PC no se incrementa de manera secuencial, sino que se actualiza directamente con una nueva dirección, permitiendo cambiar el flujo del programa.

Interacción con otros componentes:

- **Unidad de Control**: La unidad de control del procesador utiliza el valor almacenado en el PC para determinar qué instrucción se debe recuperar y ejecutar a continuación. Esta unidad es la que se encarga de incrementar el PC o modificar su valor en caso de que una instrucción lo requiera.
- **Bus de Direcciones:** El valor del PC se coloca en el bus de direcciones para señalar la ubicación de la memoria donde se encuentra la próxima instrucción a ser ejecutada. La memoria, en respuesta, envía la instrucción correspondiente a través del bus de datos.

- **Memoria RAM:** La RAM almacena las instrucciones del programa. El PC interactúa directamente con la memoria, ya que su valor indica la dirección específica en la RAM de la próxima instrucción.

6. Registro de Instrucción (IR-Instruction Register)

El IR es responsable de almacenar la instrucción que está siendo ejecutada actualmente, permitiendo que la unidad de control del procesador la decodifique y ejecute.

Funciones:

El Registro de Instrucción (IR) tiene la función de retener la instrucción que ha sido extraída de la memoria para su ejecución. Esto incluye tanto el código de operación (opcode) como cualquier operando asociado.

Las funciones principales del IR incluyen:

- **Almacenamiento Temporal de la Instrucción Actual:** Una vez que el Program Counter (PC) ha señalado la dirección de la siguiente instrucción y esta ha sido recuperada de la memoria, la instrucción se almacena en el IR. Esto asegura que la instrucción esté disponible para ser decodificada y ejecutada por la unidad de control.
- **Interacción con la Unidad de Control:** El IR entrega la información necesaria a la unidad de control para que esta pueda determinar qué operación debe realizar el procesador. La unidad de control, a su vez, genera las señales necesarias para llevar a cabo la instrucción.

Estructura del IR-Instruction Register:

- **Contenido del IR:** El IR típicamente contiene el código de operación (opcode), que indica qué tipo de operación se debe realizar (por ejemplo, suma, salto, carga), y puede incluir campos adicionales para operandos, que especifican los datos o direcciones sobre los cuales se opera.
- **Tamaño del Registro**: El tamaño del IR depende de la longitud de la instrucción en la arquitectura. En la microarquitectura IJVM, donde las instrucciones pueden variar en tamaño, el IR debe ser lo suficientemente grande como para contener toda la instrucción.

Interacción con otros componentes:

- **Unidad de Control:** La principal interacción del IR es con la unidad de control. Una vez que la instrucción está almacenada en el IR, la unidad de control la decodifica para determinar la serie de micro-operaciones que se deben realizar para ejecutar la instrucción.
- **Bus de Datos:** El IR recibe las instrucciones a través del bus de datos, después de que estas han sido extraídas de la memoria. Una

vez que la instrucción está en el IR, ya no se necesita una conexión directa con el bus de datos hasta que se necesite una nueva instrucción.

- **Memoria RAM:** Las instrucciones se recuperan de la memoria RAM y se cargan en el IR para su ejecución. El contenido del IR proviene directamente de la memoria en respuesta a la dirección indicada por el PC.

7. Memoria RAM

Es el almacenamiento temporal y de alta velocidad que permite al procesador ejecutar programas de manera eficiente. Durante la ejecución de un programa, las instrucciones y los datos se cargan en la RAM y se acceden continuamente a medida que el procesador avanza en la ejecución; no solo almacena el código y los datos del programa en ejecución, sino que también es el lugar donde se maneja la pila (stack) del IJVM.

Funciones:

La memoria RAM es un tipo de memoria volátil que se utiliza para almacenar temporalmente los datos y las instrucciones que la CPU necesita mientras ejecuta programas. Su volatilidad significa que los datos almacenados se pierden cuando se apaga el sistema.

Las funciones principales de la memoria RAM incluyen:

- **Almacenamiento Temporal de Instrucciones:** Las instrucciones de un programa se cargan en la memoria RAM desde un almacenamiento secundario, como un disco duro, y son recuperadas desde la RAM por el procesador para su ejecución.
- **Almacenamiento Temporal de Datos:** Los datos que se manipulan durante la ejecución de un programa también se almacenan en la RAM. Esto incluye variables, estructuras de datos, y cualquier otro tipo de información que el programa necesita para funcionar.
- **Intermediario de Datos:** La RAM actúa como un intermediario entre la CPU y los dispositivos de almacenamiento más lentos. Esto permite que el procesador acceda rápidamente a los datos sin tener que esperar a que sean recuperados desde un disco duro, por ejemplo.

Estructura de la Memoria RAM:

- **Celdas de Memoria:** La RAM está compuesta por una gran cantidad de celdas de memoria, cada una de las cuales puede almacenar un bit de información. Estas celdas están organizadas en una matriz de filas y columnas, y se accede a ellas mediante direcciones específicas.
- **Dirección de Memoria:** Cada celda de memoria en la RAM tiene una dirección única, que es utilizada por la CPU para acceder a la

información almacenada. El bus de direcciones del procesador se encarga de especificar estas direcciones.
- **Tiempo de Acceso:** Período necesario para leer o escribir datos en una celda de memoria. La RAM se caracteriza por su capacidad de acceso aleatorio, lo que significa que cualquier celda de memoria puede ser accedida directamente en un tiempo constante, sin importar su ubicación.

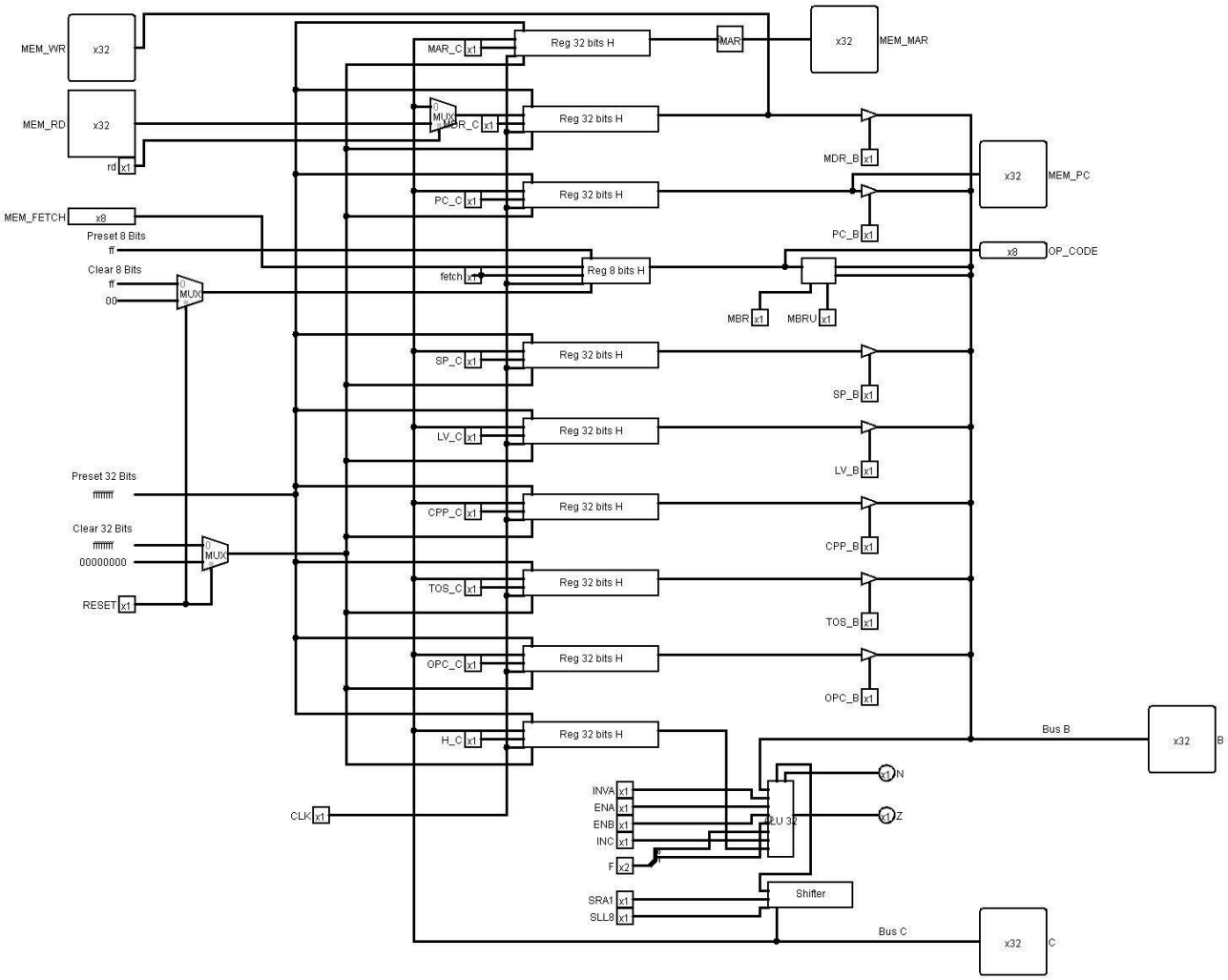
Interacción con otros componentes:

- **Bus de Datos:** Los datos que se almacenan o recuperan de la RAM se transfieren a través del bus de datos. Cuando el procesador necesita leer o escribir datos en la RAM, esos datos fluyen a través de este bus.
- **Bus de Direcciones:** El bus de direcciones se utiliza para especificar la ubicación exacta dentro de la RAM donde se deben almacenar o recuperar los datos. La dirección se coloca en este bus y se envía a la RAM para acceder a la celda de memoria correspondiente.
- **Controlador de Memoria:** El controlador de memoria gestiona las operaciones de lectura y escritura en la RAM. Este componente recibe las señales de control del procesador y coordina el acceso a la memoria para garantizar que los datos se transfieran correctamente.

Operaciones típicas en la Memoria RAM:

- **Lectura:** Cuando la CPU necesita leer un dato o una instrucción, envía la dirección de memoria correspondiente a través del bus de direcciones. La RAM recupera el dato de esa ubicación y lo envía de vuelta a la CPU a través del bus de datos.
- **Escritura:** Para almacenar un dato en la RAM, la CPU coloca el dato en el bus de datos y la dirección correspondiente en el bus de direcciones. Luego, la RAM almacena el dato en la ubicación especificada.

Por último, se muestra a continuación el data path completo:



Unidad de Control

La Unidad de Control es el componente responsable de dirigir y coordinar todas las operaciones dentro de la microarquitectura MIC-1. Su función principal es interpretar las instrucciones almacenadas en el Registro de Instrucción (IR) y generar las señales de control necesarias para que los otros componentes del procesador realicen las operaciones requeridas.

Componentes de la Unidad de Control

MIR (Microinstruction Register - Registro de Microinstrucción)

Función: El MIR es un registro que almacena la microinstrucción actual que está siendo ejecutada. En la MIC-1, cada microinstrucción dicta las señales de control que deben ser enviadas a diferentes partes del procesador para realizar una operación específica. Estas microinstrucciones se obtienen del microprograma almacenado en la memoria de control.

Contenido: Cada microinstrucción en el MIR especifica qué operaciones deben realizarse en un ciclo de reloj dado. Esto incluye acciones como mover datos entre registros, realizar operaciones en la ALU, o controlar el flujo de microinstrucciones.

Importancia: El MIR es crucial porque actúa como la "hoja de ruta" para la ejecución de cada instrucción de nivel superior (macroinstrucción). Sin el MIR, la unidad de control no tendría una forma de saber qué señales de control activar en cada momento.

MPC (Microprogram Counter - Contador de Microprograma)

Función: El MPC es similar al Program Counter (PC), pero en lugar de apuntar a la siguiente instrucción del programa principal, el MPC apunta a la siguiente microinstrucción dentro del microprograma. Este registro determina cuál es la próxima microinstrucción que debe cargarse en el MIR.

Actualización: El MPC puede ser actualizado de varias maneras, dependiendo de la lógica de la microinstrucción actual. Puede incrementarse secuencialmente o puede ser cargado con un valor nuevo, dependiendo de si la microinstrucción actual requiere un salto en el flujo de control del microprograma.

Importancia: El MPC asegura que las microinstrucciones se ejecuten en el orden correcto, o que se tomen los saltos necesarios dentro del microprograma para manejar situaciones como saltos condicionales, subrutinas, etc.

Decoder (Decodificador)

Función: El decodificador en la MIC-1 toma las señales del MIR y las traduce en señales de control específicas que se envían a los diferentes componentes del procesador (como la ALU, los buses, los registros, etc.). Estas señales de control activan o desactivan las operaciones que deben realizarse en un ciclo de reloj específico.

Operación: El decodificador esencialmente convierte los bits de la microinstrucción en las señales necesarias para realizar tareas como leer o escribir en un registro, seleccionar una operación de la ALU, controlar los buses, etc.

Importancia: Sin el decodificador, las microinstrucciones almacenadas en el MIR no podrían traducirse en acciones concretas dentro del procesador. El decodificador actúa como el "traductor" que convierte las instrucciones en acciones.

Microprograma

Descripción: El microprograma es una secuencia de microinstrucciones almacenadas en una memoria de control, a menudo ROM (Read-Only Memory). Este conjunto de microinstrucciones define cómo se deben ejecutar las instrucciones de máquina de nivel superior (macroinstrucciones).

Función: El microprograma actúa como un "guion" para la ejecución de cada instrucción de nivel superior. Por ejemplo, cuando una instrucción IJVM es cargada, el microprograma correspondiente se activa, y a través del MPC y el MIR, se ejecutan las microinstrucciones necesarias para realizar la tarea.

Estructura: Cada microinstrucción dentro del microprograma contiene campos específicos que controlan diferentes aspectos del procesador, como la ALU, el manejo de los buses, la manipulación de los registros, y las operaciones de memoria.

Importancia: El microprograma permite que una misma arquitectura física (como la MIC-1) pueda ejecutar diferentes conjuntos de instrucciones simplemente cambiando el microprograma almacenado en la memoria de control. Esto hace que la MIC-1 sea extremadamente flexible y capaz de ejecutar el conjunto de instrucciones IJVM.

La Unidad de Control en la microarquitectura MIC-1 es el núcleo de la operación del procesador, coordinando todas las acciones necesarias para ejecutar las instrucciones IJVM. A través de sus componentes clave, como el MIR, MPC, el decodificador y la microprograma, la unidad de control traduce las instrucciones de alto nivel en las microinstrucciones detalladas necesarias para manipular los datos y gestionar el flujo de control dentro de la CPU. Sin una unidad de control eficiente y bien diseñada, la microarquitectura MIC-1 no podría funcionar correctamente.

Conclusión

A lo largo de este proyecto, hemos tenido la oportunidad de profundizar en la microarquitectura MIC-1, tal como se describe en el libro "Structured Computer Organization" de Andrew S. Tanenbaum. Empezamos analizando los componentes principales del Camino de Datos y de la Unidad de Control, entendiendo cómo interactúan entre sí para ejecutar las instrucciones del procesador.

Una de las cosas más importantes que aprendimos es cómo se controla el flujo de datos dentro de la CPU y cómo cada componente juega un rol específico para que el procesador funcione correctamente.

Creemos que se podría mejorar en algunos aspectos. Por ejemplo, se podría optimizar el microprograma para hacer que algunas operaciones sean más rápidas, o incluso añadir más funciones a la ALU para expandir sus capacidades.

Trabajar en este proyecto nos ha dado una visión clara de cómo se construyen y operan los procesadores desde cero. La MIC-1 es un modelo simple y muy efectivo para enseñarnos los fundamentos de la arquitectura de computadoras.

Bibliografía

- Información sobre MIC-1: https://en.wikipedia.org/wiki/MIC-1
- Información sobre logisim: https://es.wikipedia.org/wiki/Logisim
- Apuntes brindados por la cátedra de Arquitectura y Organización de Computadoras.
- Información sobre el Datapath: https://darkdans.blogspot.com/2017/02/el-camino-de-datos-datapath.html