Universidad de Puerto Rico
Recinto de Mayagüez
CIIC 4082 – Arquitectura de Computadoras II

Tarea #1:
Diseño de CPU

Prof. José Navarro-Figueroa
16 de septiembre de 2022
Capella Muñiz, Eithan M. 802-19-9206

Introducción:

A través de logisim, se desarrolló un CPU de 8 bits con una memoria RAM de 16x8 para almacenar las instrucciones y la data. Esta fue diseñada como una herramienta educativa, por el cual varias partes tienen etiquetas describiendo la funcionalidad o los datos dentro de los registros y las líneas. También tiene inputs manuales en los datapaths por si el estudiante quiere verificar la funcionalidad de los componentes.

El CPU puede realizar 4 operaciones, un Load, Store, Add y Negate. Por ende una calculadora de suma y resta. Para estas operaciones se usan números de 8 bits. Por el cuál hay que estar pendiente de no tener un overflow for sumar números más gigantes que el número máximo permitido: “256”

El Ram del circuito está codificado en hexadecimal, el formato de las instrucciones en lenguaje de máquina deben ser de la siguiente manera:

Tabla 1: Set de instrucciones en lenguaje de máquina

Instr	Formato	Operación	Comentarios
Load	00ddaaaa	R[dd]<-Mem[aaaa]	Cargar registro dd con contenido de memoria aaaa
Store	01ddaaaa	Mem[aaaa]<-R[dd]	Guardar contenido de registro dd en memoria aaaa
Add	10ddssxx	R[dd]<-R[dd]+R[ss]	Suma de registros
Neg	11ddxxxx	R[dd]<- 2’s comp R[dd]	Complemento de dos del registro

Proceso de solución del problema:

Para poder crear el cpu, primero se tienen que identificar los componentes básicos necesitados

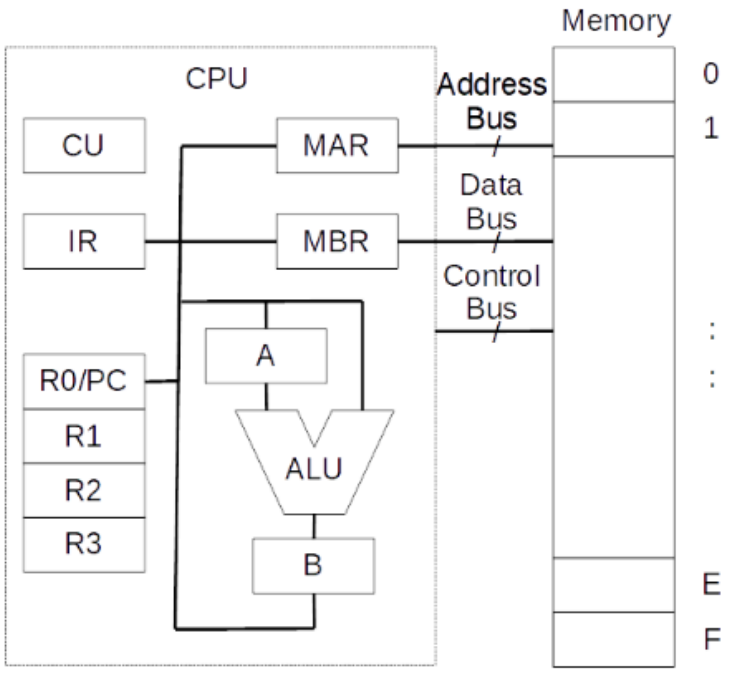

Figura 1: Arquitectura del CPU

.

Por el cual se necesita crear 4 registros separados de 8 bits R0-R1, Los registros IR, MBR son de 8 Bits. El Registro de MAR es de 4 Bits. A y B son registros de 8 Bits. El datapath para la mayoría de operaciones del ALU son de 8 Bits también, pero el datapath para acceder las posiciones en la memoria son de 4 bits. Como se puede observar en las siguientes figuras.

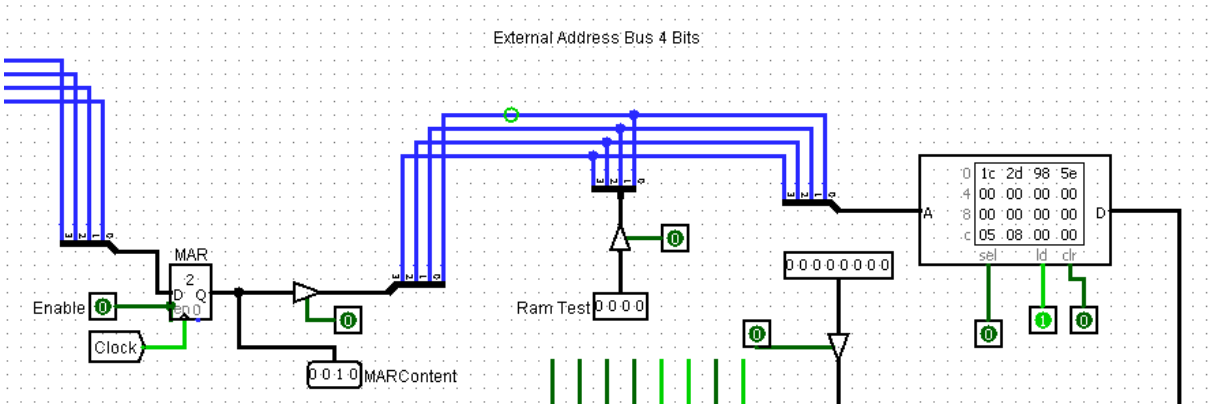

Figura 2: Internal and External Address Bus 4 Bits

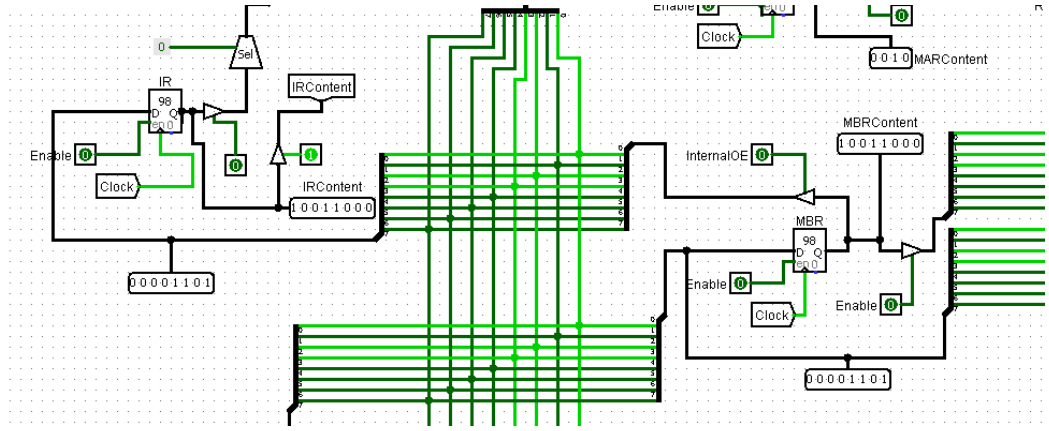

Figura 3: Internal and External Address Bus 8 Bits.

El ALU no tuvo mucha dificultad ya que Logisim cuenta con una serie de adders y negators. Los cuales fueron juntados para tener las dos operaciones del ALU.

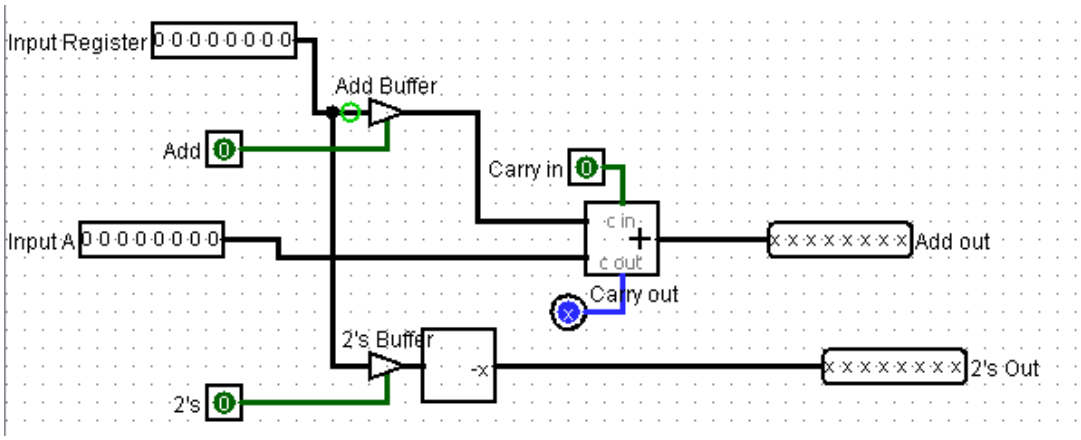

Figura 4: ALU con su adder y negator.

Realmente la parte más complicada del proyecto era el desarrollo de las señales de control, ya que para poder hacer que el proceso sea automático, se necesita evitar conflictos dentro de los data paths. Para realizar esto, utilizamos varios buffers en las salidas y entradas para evitar conflictos, fuera más eficiente crear registros que tengan estos buffers ya que los registro de Logisim solo tienen un **enable** en la entrada.

Una vez estos buffers ya sean puestos podemos hacer las operaciones del CPU de forma manual, por el cual podemos tomar en cuenta cuántos ciclos se necesitan por operación. Por ejemplo, la operación de Load nos toma 7 ciclos, ya que varios de los registros necesitan un ciclo para recibir la operación ya que, estos solo reciben información en el "Rising Edge" del clock. Una vez sabemos cuántos ciclos son necesarios para realizar cada operación. Se puede crear un CU.

Este control unit va a tomar en cuenta que instrucción de lenguaje de máquina tenemos que realizar:

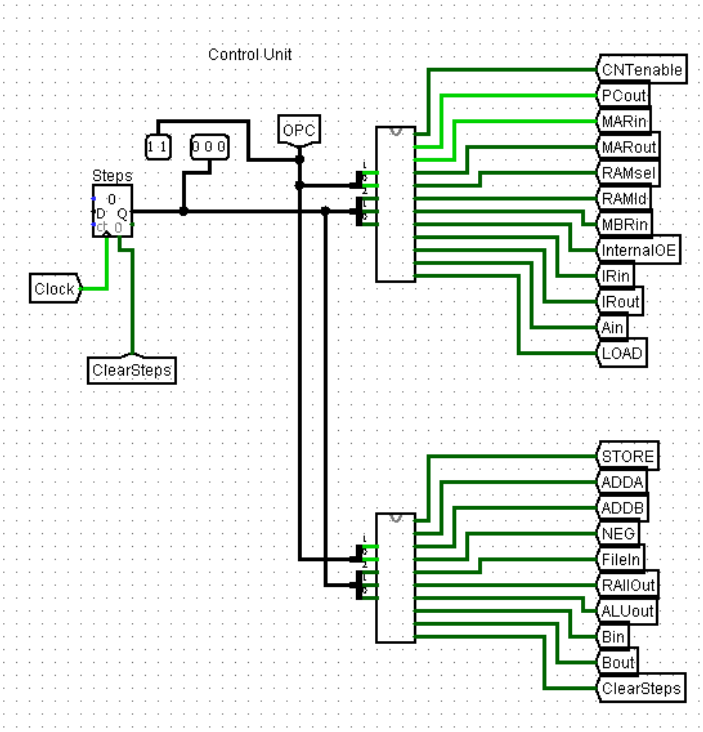

Figura 5: El Control Unit

Esto se puede hacer tomando la instrucción dentro del IR y pasándola por un bit selector y un decoder, donde podemos identificar las 4 operaciones del CPU. De aquí hacemos un branching en el cual manejamos los buffers de cada registro para poder realizar las operaciones necesarias. Por ejemplo la operación ADD. Una vez el PC llegue a la instrucción, vamos a enviar la señal al registro MAR, hacemos 1 ciclo, la enviamos luego al RAM para hacerle read al contenido de la instrucción ADD, supongamos que esta es 10 01 10 xx, osea un Load R1 R2 los dos bits xx pueden ser ignorados. Esto pasa al MBR y hacemos otro ciclo. Esta información se pasa al IR y se hace otro ciclo más. Por el cual hicimos la operación del Fetch ahora falta ejecutar esta. Por el cual se sabe que se necesita guardar la información del R1 en el registro A esto requiere otro ciclo y con este mismo pensamiento podemos identificar cada buffer que se necesita abrir y cerrar para realizar las operaciones. A través de este proceso se genera una tabla de unos y ceros el cual usando el combinational analysis de logisim se puede formar nuestro CU, con los 6 inputs, la operación a realizar y la cantidad de ciclos que toma.

Operación del sistema:

Al momento el circuito carece de un CU ya que este esta incompleto. Asumiendo que este estuviera hecho, el proceso para inicial el sistema, sería: Cargar el RAM con las instrucciones deseadas con un file. Luego usar CTRL + T si uno quiere hacer los ticks manualmente. Si desea que este fuera automático, fuera simplemente utilizar CTRL + K.

Como este no está completado, se necesita ir manualmente sobre cada operación:

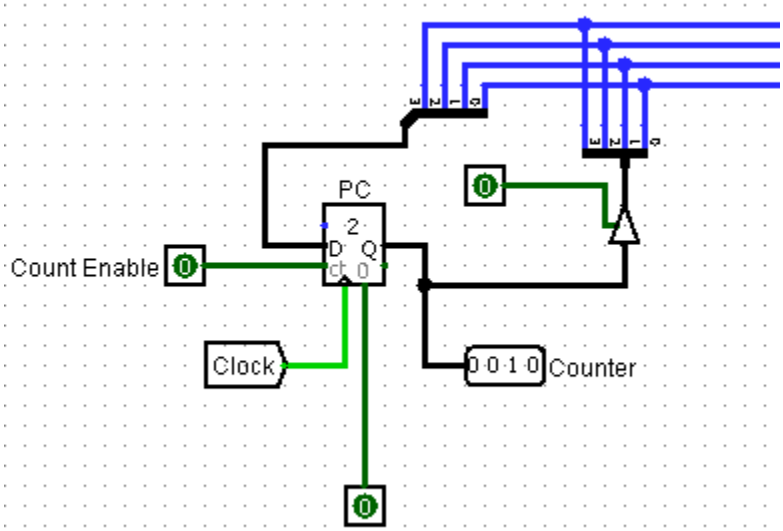

Figura 6: Program Counter

Cada registro va a tener cerca de él dos enables, uno de input y otro de output como puede ser visto aquí. Por ende uno actuaría como si fuese un CU.

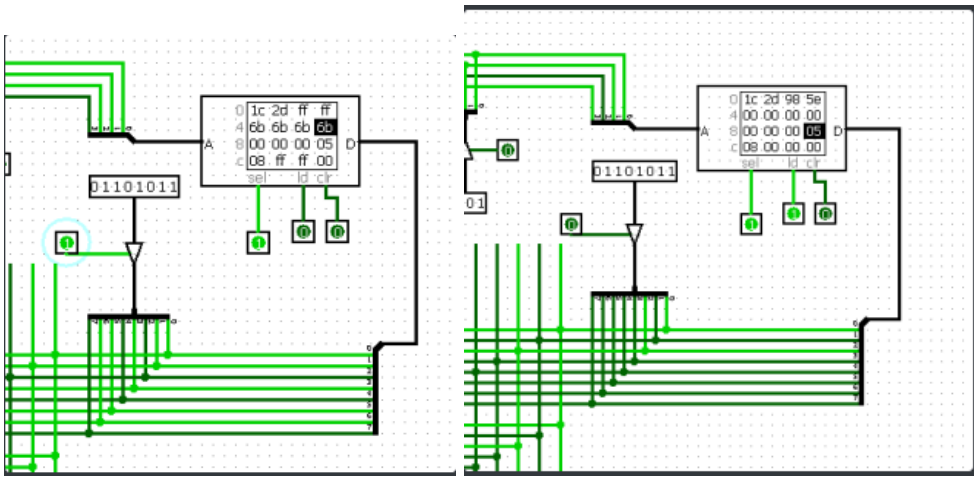

Figura 7: Write y Read del RAM (Por orden)

Distribución de Tareas: No Aplica.
Los otros integrantes van a entregar sus propios reportes basados en mi circuito y explicaciones de cómo funciona. Esto debido a que intenté darles toda la información necesaria para que ellos intentarán terminar el proyecto, pero al fin del día yo hice mis propios documentos y videos ya que ellos no habían realizado nada del trabajo (al momento de yo hacer todos estos documentos)

Comenzaron a trabajar durante la noche del viernes de la entrega. Por el cuál yo me encontré trabajando solo durante las semanas. Le envié un email sobre ello: "[CIIC4082-100] Conflicto con integrantes del grupo" este es el título del email.

Google Drive con el video y todo los documentos:

https://drive.google.com/drive/folders/1dkCHoFnBjz0NgAFu-IhD-y3T3j2hOPVM?usp=sharing