Trabajo práctico integrador
Organización de las computadoras

Simulación de un procesador RISC-V 32 I

Agustin Gregoret, Walter Voegeli.

Introducción

En el siguiente informe, se detalla el proceso de construcción en Verilog, lenguaje utilizado para simular circuitos electrónicos, un procesador RISC-V monociclo con la capacidad de ejecutar diversas instrucciones, incluyendo sumas, restas, salto incondicionales, saltos condicionales e instrucciones de carga y almacenamiento de datos en memoria.

Consigna y desarrollo

Se deberá simular al RISC-V visto en clase con el programa de prueba de la guía práctica 3.

Para desarrollar el procesador, se construyeron los módulos que componen el mismo, en formato digital utilizando el lenguaje verilog, para que una vez codificados y testeado que cada uno cumpla su función, se conecten entre sí a través de la simulación de un datapath que especifique las rutas de conexiones entre ellos. Una vez realizado el procesador, se le cargó en el módulo de memoria de programa, una serie de comandos en formato binario que, a través de la decodificación, uso de memoria de registros, operaciones aritmético-lógicas y de salto, se ejecute el programa cargado.

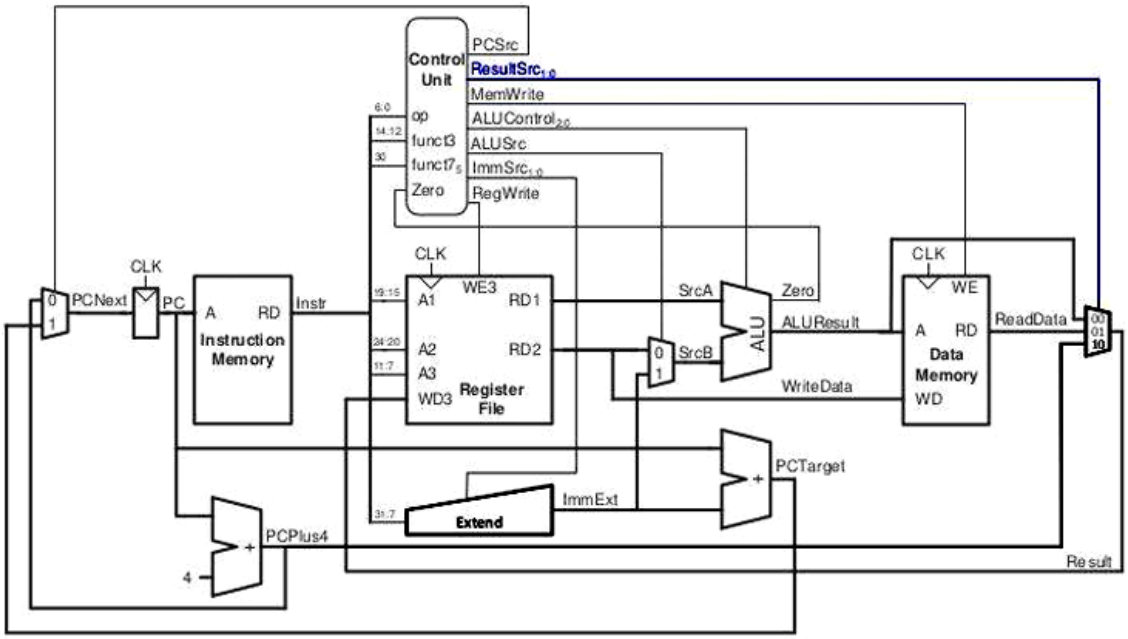

FIG 1: Esquema de los módulos utilizado con sus conexiones para la construcción.

Módulos construidos

ALU: La unidad aritmética lógica, que realiza las operaciones aritméticas suma y resta, y las operaciones lógicas AND y OR. La misma recibe los operandos y una señal de control (ALUControl) que selecciona la operación a realizarse. Devuelve el resultado en la señal res. En caso de obtener un resultado con valor cero, será notificado como una señal alta a la unidad de control.

ALU sum: Unidad aritmética lógica que solamente suma dos valores de 32 bits y devuelve su resultado.

ALU sum 4 bits: Unidad aritmética lógica que solamente suma 4 bits a una señal procedente del contador del programa.

Banco de registros (BR): Es el encargado de controlar los registros del procesador donde se almacenan y leen datos de acceso rápido. En el mismo ingresa una señal de reloj, los operandos a1, a2 (los cuales son las direcciones a leer), el operando a3, la señal wd3 y la bandera we, que indica si hay que escribir en un registro, donde si dicha bandera está activa, se escribirá en el registro con la dirección a3 el dato ingresado en la señal wd3. Este módulo devuelve las señales rd1 y rd2, que son los datos guardados en las direcciones de entrada a1 y a2.

Sign extender (SI): es un circuito combinacional que genera el campo de datos inmediato, de 12 ó 20 bits, para las instrucciones aritméticas, lógicas, carga, almacenamiento, y saltos. Reorganiza la información del campo inmediato, de acuerdo con el código de operación, para ser utilizada por el procesador para

ejecutar la instrucción correspondiente. Estas tareas involucran reordenar los bits, completar con 0 y desplazar el dato resultante de acuerdo al tipo de operación.

Memoria de programa (IM): Es la región de la memoria del sistema donde se aloja el programa que se está ejecutando. En este trabajo se almacenará el código de prueba proporcionado por la cátedra.

Memoria de datos (DM): Es la región de la memoria del sistema donde se alojan los datos
utilizados por el programa (datos dinámicos y estáticos, pila).

Multiplexores: Se utiliza para seleccionar, a partir de varias entradas de 32 bits, una salida. Funciona de la forma que por medio de una señal recibida de la unidad de control, especifique cuál de las múltiples conexiones entrantes, será la que continuará por el datapath al siguiente módulo. En este trabajo se utilizarán dos tipos de multiplexores: de 2 entradas a 1 salida y de 3 entradas a una salida.

Program counter (PC): Es un registro de 32 bits que guarda la dirección de la instrucción que se está ejecutando . A él ingresan la señal del reloj (clk) y el siguiente pc, para así devolver como salida el pc actual, que se actualiza con cada pulso de reloj.

Unidad de control (UC): Es la unidad que controla el funcionamiento de los módulos que realizan múltiples operaciones y necesitan de una señal que les indique cómo proceder. Este módulo está compuesto del decodificador principal (main decoder) y del decodificador de la unidad aritmético lógica (ALU decoder)

Main decoder: recibe el código de operación y devuelve los valores de las señales branch, jump, resSrc, memWrite, aluSrc, immSrc, regWrite y aluOp correspondientes según el tipo de operación (load, store, branch, entre otras).

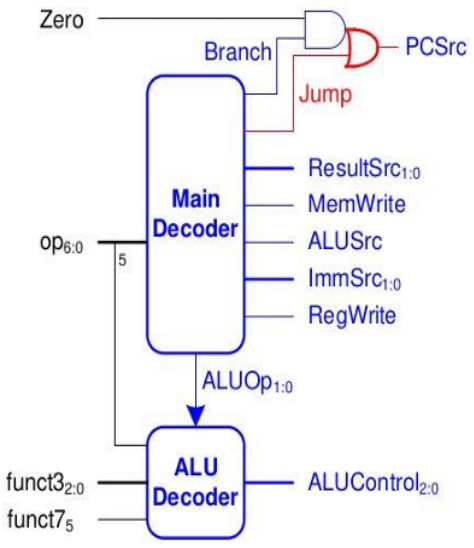

FIG 2: Esquema de la unidad de control.

ALU Deco: recibe el código de operación, f7, f3 y aluop y, si la última está activa (es decir, si se debe utilizar la alu), devuelve en ALUControl la operación que debe realizar la ALU.

Datapath: Es el camino de los datos, el bus de señales que conectan a los módulos entre sí. Dependiendo de la conexión que se necesite, el datapath variará en tamaño de "cables" , es decir, que variará entre 1 y 32 bits de transmisión de datos consecutivos. En este trabajo, el datapath establecerá un clock en común para los módulos que lo necesiten para que haya una sincronización de ejecución entre ellos.

Procesador: Engloba e instancia la unidad de control, el datapath y la memoria de datos para que funcionen en conjunto de manera sincrónica.

Pruebas de ejecución individuales.

Para cada módulo se realizó una prueba de ejecución individual para comprobar que los mismos realicen su trabajo correctamente, para ello se construyó un "test bench" que consiste en asignar valores a las señales de entrada de cada módulo en diferentes tiempos y, por medio de el programa gtkWave, se simula y visualiza el comportamiento del módulo, como así también las devoluciones de señales que realiza. De este modo, se observa si el módulo se comporta como debería.

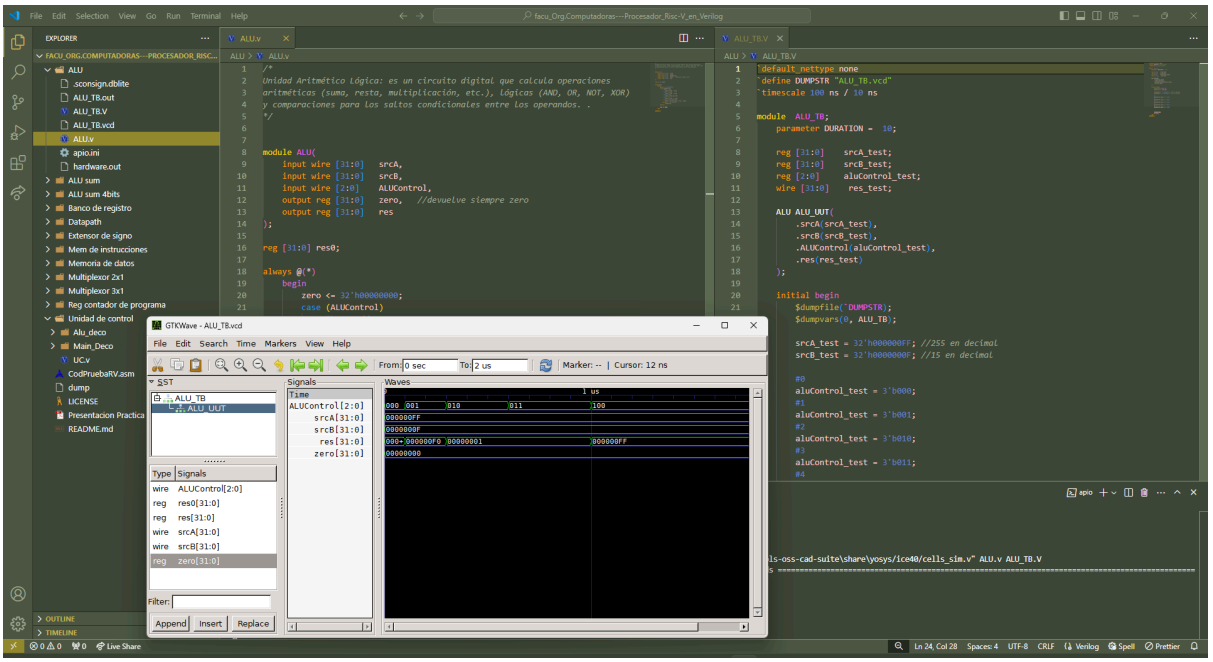

FIG 3: Simulación del funcionamiento del módulo ALU utilizando el programa GTKWave.

En la figura 3, se testeó la unidad de control asignando valores en 32 bits hexadecimal en la entrada A y B del módulo, y luego mediante la variación de los valores de la entrada aluControl, que corresponde a la que indica la unidad de control, se especifica realizar distintas operaciones con esos valores ingresados, como suma, resta, and, or y SLT.

Así como se realizó esta prueba en el módulo ALU, se pueden encontrar todas las demás pruebas en las distintas carpetas correspondientes a cada uno en el archivo comprimido que contiene el procesador.

Prueba de ejecución.

En el momento de ejecutar la prueba final nos encontramos con 2 grandes problemas principales:

un error en las instrucciones de salto (jal) y un problema de desfases en las instrucciones que requieren leer datos desde la memoria de datos y cargarlos en el banco de registros (lw). Para el primer error se hizo un seguimiento de los datos con ayuda del programa GTKwave, hasta ver que los datos del tipo jal estaban siendo mal manejados en el módulo de extensión de signo. Una vez corregido la manera en la que este módulo tomaba y ordenaba los bits correspondientes al inmediato de la instrucción tipo J, los bucles for y while funcionaban correctamente.

El segundo error ocurría al momento de leer las posiciones [0], [1], [2] de la memoria de datos y cargarlas en los registros [5] , [6] y [7] en ese orden, sin embargo lo que ocurría es que no se cargaba el dato [5], el registro [6] contenía el dato almacenado en [0] y el registro [7] cargaba el dato [1]. Esto ocurría ya que la memoria de datos y el banco de registros estaban instanciados dentro del datapath, por lo que el dato que la carga del registro sucedía al mismo tiempo que la búsqueda del dato en la memoria de datos, generando una carga al registro de un dato que aún no estaba listo. Esto se soluciona instanciando la memoria de datos fuera del datapath, forma más acorde a los procesadores reales donde la memoria de datos no es parte interna del procesador. Al instanciar dicha memoria dentro del módulo del procesador, fuera del datapath, la propagación del clk hace que la salida de la memoria de datos contenga el dato correcto al momento de enviarlo a la entrada de escritura del banco de registros.

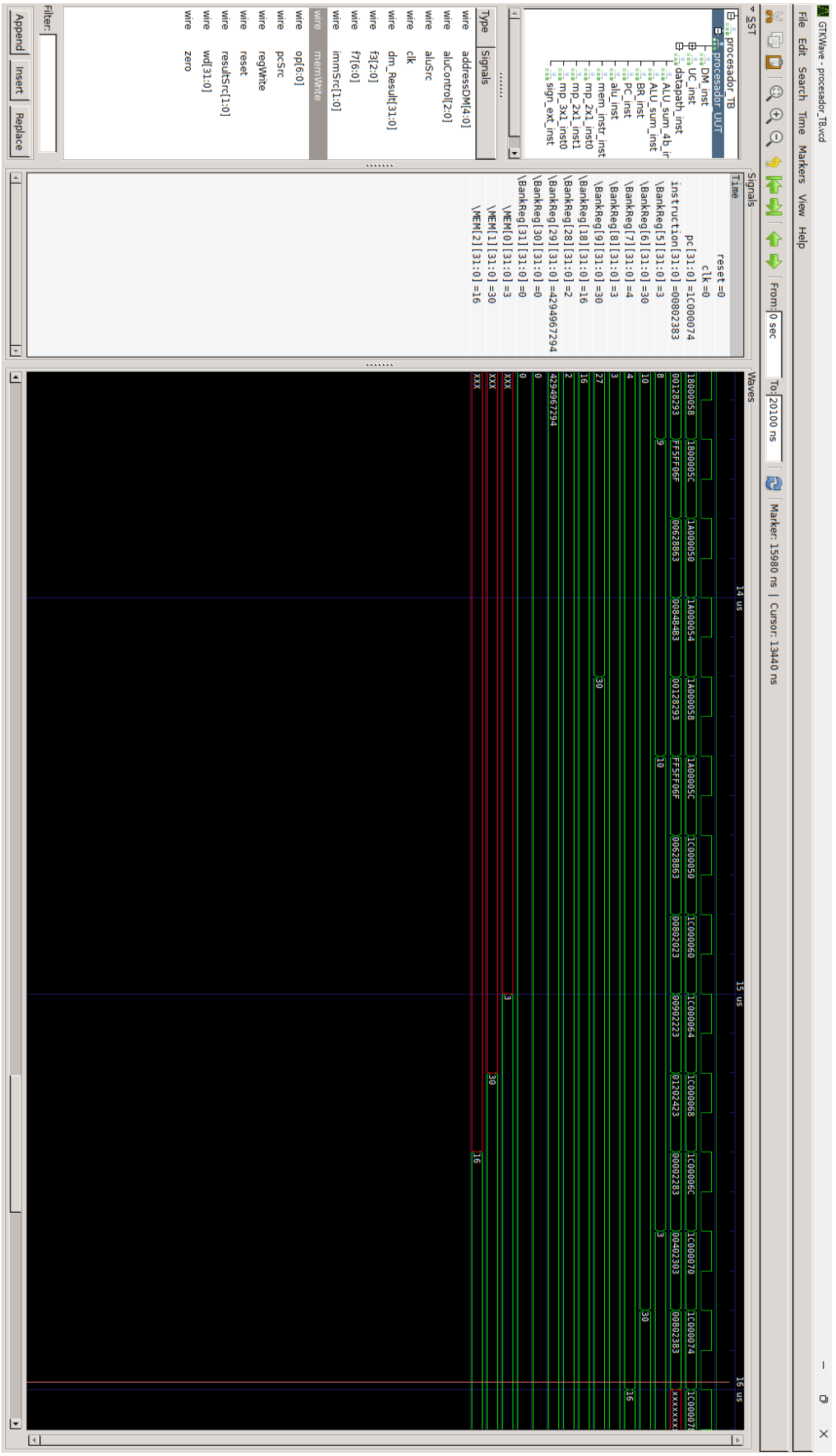

FIG 4: Resultado final de la simulación, donde se observa la instrucción load word cargar los valores almacenados en el data memory a los registros 5, 6 y 7.

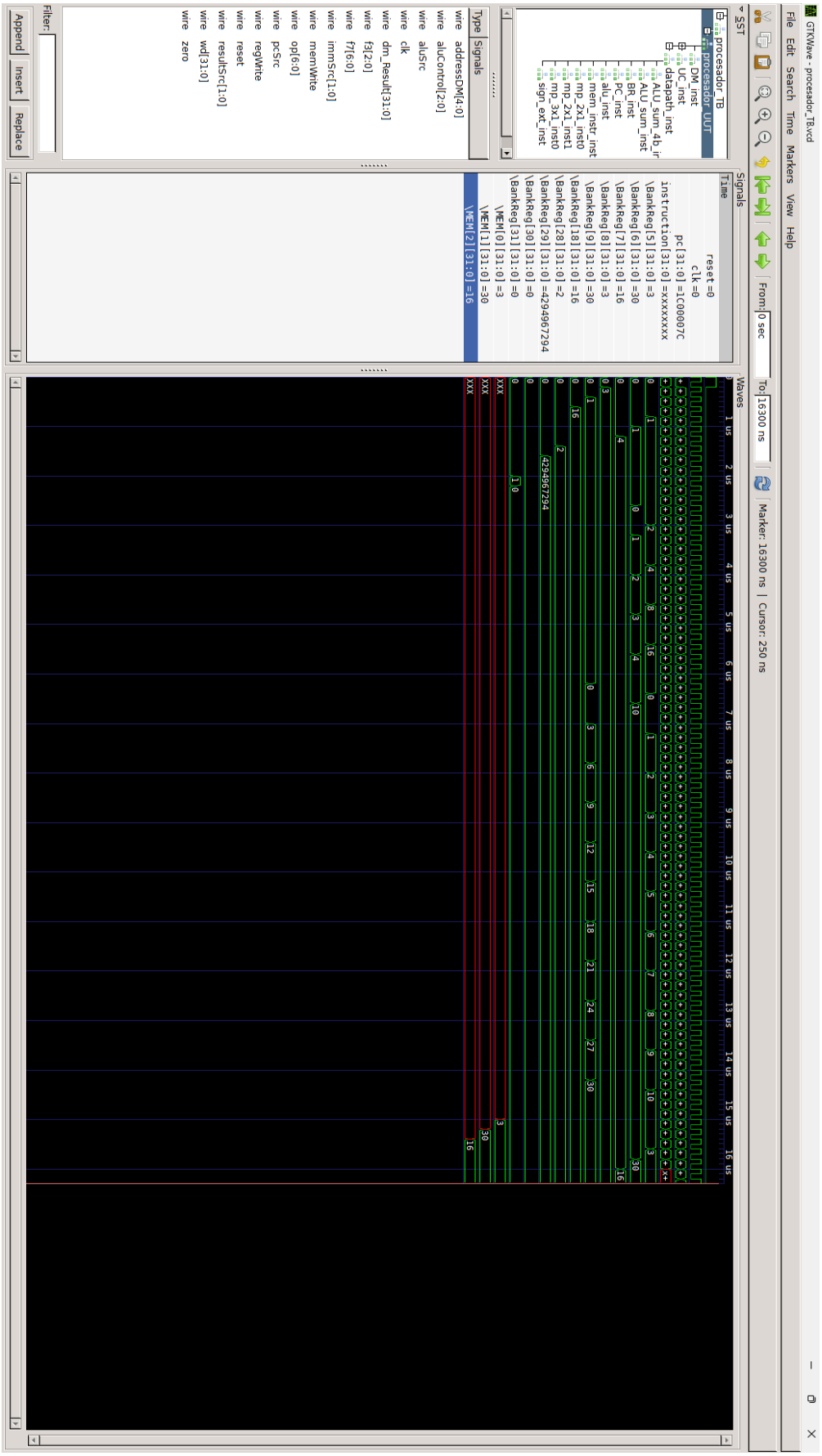

FIG 5: Valores que obtienen las variables principales del procesador a lo largo de la ejecución del programa.

Conclusión

La construcción de los módulos, conexiones y las pruebas de un procesador risc-v es un proceso arduo que conlleva un nivel elevado de entendimiento del funcionamiento individual de cada componente como sus conexiones que, haciendo un todo, es capaz de lograr ejecutar múltiples secuencias de instrucciones que componen trabajo de datos en memoria, operaciones aritmético lógicas y saltos de ejecución entre otros.
A pesar de las dificultades atravesadas, se pudieron superar realizando análisis exhaustivos bit a bit a través de la herramienta gtkwave para encontrar el módulo que fallaba y complementando con el uso de la teoría y los diagramas de la materia.

Como estudiantes, este trabajo nos ayudó a comprender y aprender el funcionamiento básico de un procesador y la lógica detrás de cada componente, como así también el criterio y modo de trabajo al desarrollar cualquier artefacto de procesamiento de datos en bajo nivel.