Arquitectura de Computadores
Proyecto de Aplicación MIPS

Presentado a:
Jose Oliden Sánchez

Realizado por:

Edwin Andrés Calvo
Juan José Restrepo Rosero
Luis Alejandro Balcázar

Facultad de Ingeniería y Ciencias
Ingeniería Electrónica
Santiago de Cali, Valle del Cauca
27 de noviembre del 2021

Tabla de contenido

I. Introducción

La unión de subsistemas en la arquitectura de computadores contempla ya toda la utilidad que podemos sacar de un computador. En el presente trabajo se unirán los elementos de la arquitectura construidos anteriormente, con el fin de implementar una aplicación en donde se evidencie la utilidad de un computador y además la efectividad del conocimiento adquirido a lo largo del curso de Arquitectura de Computadores.

II. Objetivos:

- Probar el funcionamiento del MIPS Computer diseñado a lo largo del semestre.
- Desarrollar un programa en lenguaje ensamblador que permita medir el nivel promedio de gas, la presión mínima y máxima en un cilindro o tanque.
- Acoplar nuevas funcionalidades a nuestro procesador y memoria conectada para resolver un problema específico.

III. Desarrollo:

A. Problema práctico escogido:

El sistema tiene como función principal notificar al usuario acerca del nivel promedio de gas en un cilindro o tanque, además de los valores de presión máximos y mínimos al interior de dicho contenedor con el objetivo de que el usuario pueda detectar a tiempo posibles fugas o anomalías que puedan estar afectando la integridad del mismo o la productividad de una empresa. Es importante mencionar, que los valores de las variables determinadas previamente son hallados a partir de un sistema de sensores integrados y un diseño específico para este proceso, los cuales son los considerados en la memoria de la arquitectura MIPS. De esta manera, se tienen mediciones que

son el resultado de haber hecho el proceso de sensado diariamente durante los 7 días de la semana, esto con el fin de que estos datos sean almacenados, procesados y posteriormente sean presentados en unos informes semanales en los que se puedan analizar dichos datos de nivel promedio del gas y los correspondientes valores máximos y mínimos de presión de forma semanal.

B. Características:

En el diagrama de entradas y salidas se presentan unas de las características principales de la aplicación diseñada.

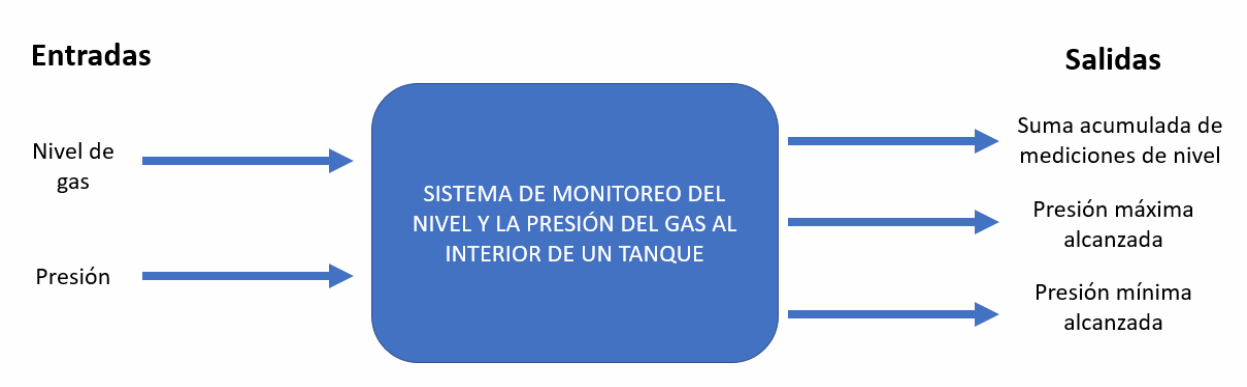

Figura 1. Diagrama de entradas y salidas.

De la imagen anterior, es posible inferir que el promedio de las mediciones nivel de gas no fue calculado directamente, sino que se calculó la suma acumulada de la medición de cada día con el objetivo de que dicho resultado sea dividido entre 7 y se obtenga el promedio. La anterior tarea se completa de esa forma semi automática debido a que la arquitectura implementada no soportaba una operación tal como la de dividir entre 7, por lo que para efectos prácticos se optó por la alternativa expuesta. Además, es importante aclarar que se realizó la suposición de que la mayoría de los datos de nivel de gas y presión, ya se encontraban almacenados en memoria, listos para ser procesados y esa es la parte más importante de la que se trata el código: .

C. Diseño del sistema

El diseño del sistema de cómputo se compuso principalmente de una arquitectura MIPS multiciclo modelada en el software Vivado. En esta sección se incluyeron diferentes multiplexores, una unidad de control con sus respectivas señales de control, la memoria y diferentes registros, esto con el fin de establecer una conexión correcta entre los diferentes bloques del sistema y así ejecutar las instrucciones deseadas. Cabe mencionar la gran importancia de esta sección dado que es gracias a ella que se realizan los ciclos de manera adecuada según el tipo de instrucción que se esté realizando. Además, sobre esta parte, desde el diseño se tuvieron que realizar múltiples procesos de depuración esto con el fin de corroborar que para cada uno de los componentes del esquema se estuviera entregando la entrada/salida necesaria según sea el caso.

A continuación, una imagen del esquemático final de la arquitectura.

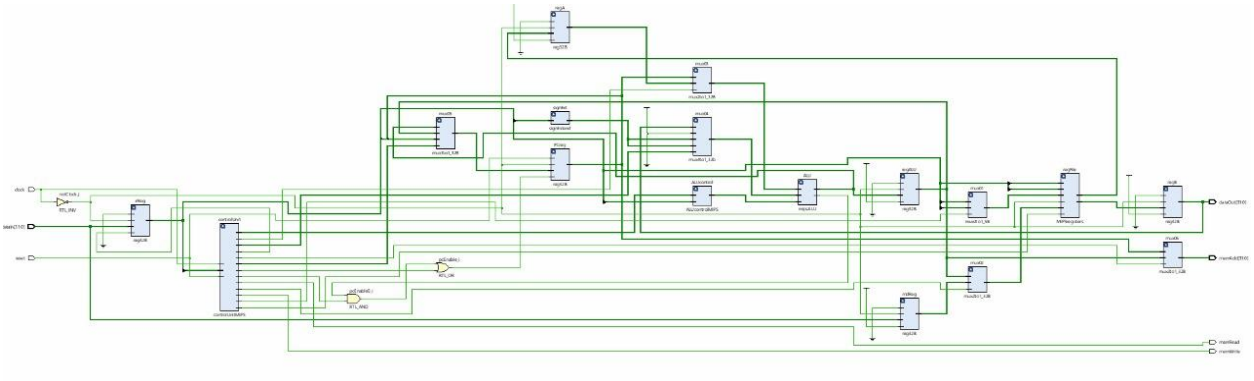

Figura 2. Arquitectura MIPS multiciclo

Por otro lado, también fue necesaria una conexión de la arquitectura creada con un procesador, este permitía que se hicieran las respectivas operaciones de lectura o escritura apoyándose en la lógica proveída por la arquitectura. Para este proceso, se necesitó además instanciar cada uno de los componentes, esto para que se constituyera un sistema completo que interconectara los diferentes registros, la memoria, entre otros bloques vitales del mismo.

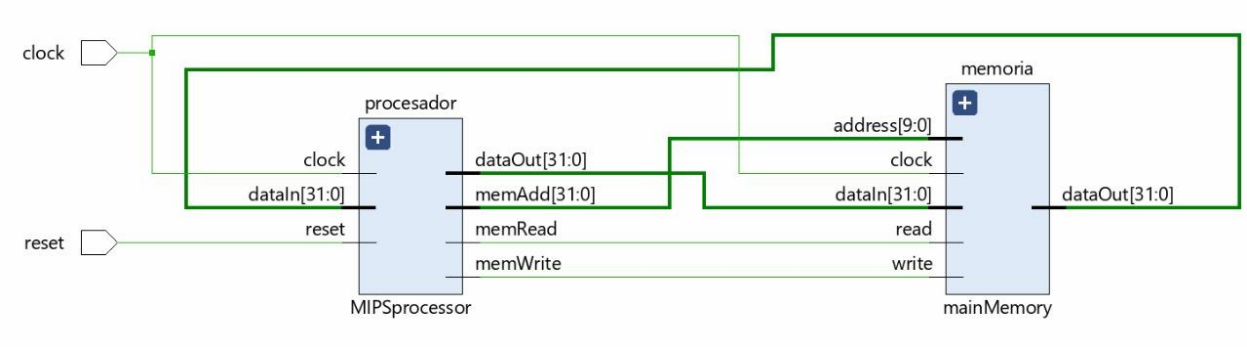

Figura 3. Conexión del procesador con la memoria principal

Finalmente, es posible ver, cómo en memoria se tienen las diferentes instrucciones y datos que fueron almacenados para que estos fueran ejecutados y así llevar a cabo la aplicación planteada.

```
architecture Behavioral of mainMemory is
    type memType is array (0 to 1023) of std_logic_vector (31 downto 0);
    signal memory: memType :=   (X"22100032", --0
                                 X"20080001", --1
                                 X"20090008", --2
                                 X"20120000", --3
                                 X"11090005", -- 4. loop
                                 X"8E110001", --5
                                 X"02519020", --6
                                 X"22100001", --7
                                 X"21080001", --8
                                 X"08000004", --9. Loop
                                 X"AE120000", --10. FIN
                                 X"03E00008",-- 11. Suma acumulada de valores de presión
                                 X"2010003C", --12
                                 X"8E080000", --13
                                 X"8E090001", --14
                                 X"200A0001", --15
                                 X"200B0007", --16
                                 X"210D0000", --17
                                 X"200E0001", --18
                                 X"114B0008", --19. Loop
                                 X"012D602A", --20
                                 X"118E0004", --
                                 X"22100001", --22. Comeback
                                 X"8E090001", --
                                 X"214A0001", --24
                                 X"08000013", --
                                 X"212D0000", --26
                                 X"08000016", --
                                 X"AE150002", --28 Salir. (calculo menor)
                                 X"2010003C", --
```

Figura 4. Memoria de la arquitectura MIPS con datos e instrucciones

D. Simulaciones

A continuación, se enseña una imagen de la ejecución de una instrucción, en este caso es un LW cuyo código es 8C010080 y es de tipo I, esta es una instrucción que requiere de 5 ciclos, es decir que necesita hasta la etapa de WriteBack. En la imagen presentada se puede ver cómo el cursor amarcillo se encuentra justo en la etapa de Fetch, razón por la cual es posible visualizar el estado de algunas de las señales de control correspondientes a este ciclo si nos fijamos en la columna de los valores de fetch, donde se puede observar que IorD está en 0, MemRead en 1, PcWrite en 1 e IRWrite en 1 que efectivamente son algunos de los valores de las señales de control que deben estar en dicho estado para el correcto funcionamiento de la instrucción, de esta manera se puede concluir que las señales de control para los demás ciclos se activan y desactivan cuando es necesario también

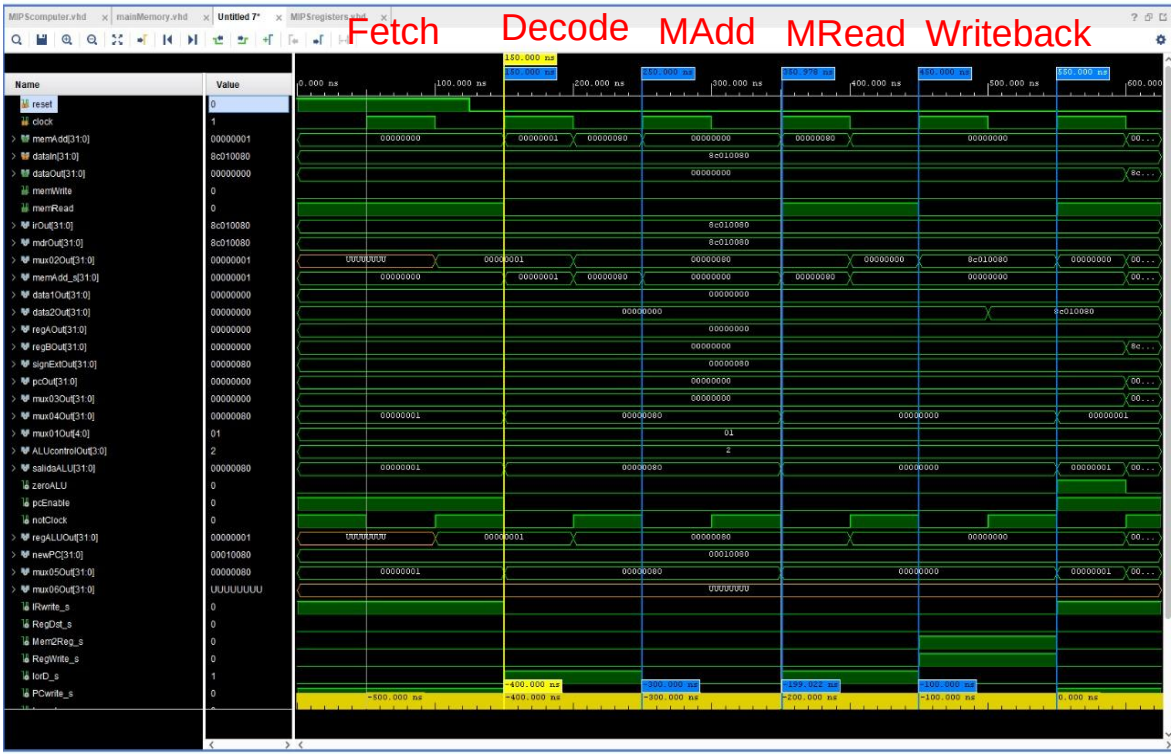

Figura 5. Prueba de simulación de los ciclos con instrucción LW

Adicionalmente, se hicieron las mismas pruebas anteriores, pero esta vez implementando una instrucción tipo R, en este caso un Add, la cual requiere de 4 ciclos, entre ellos la etapa de Execute. Así, se pueden ver los resultados a continuación:

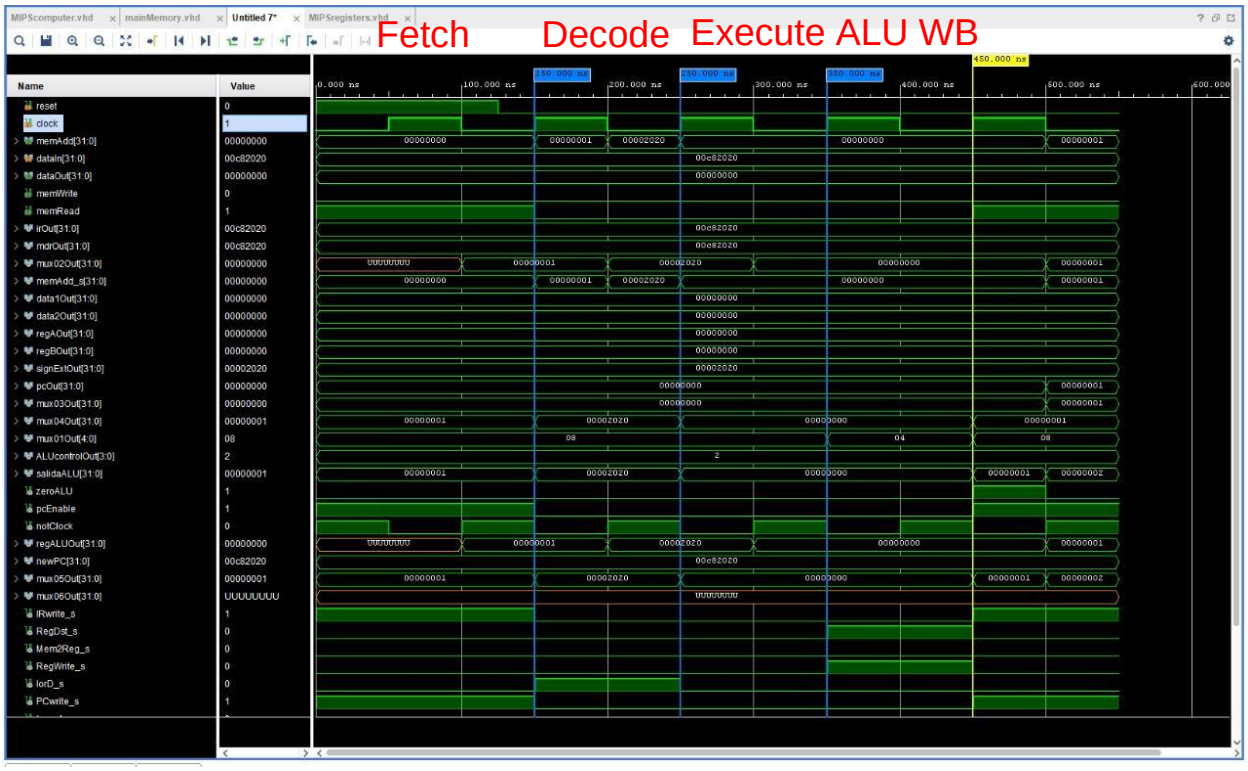

Figura 6. Prueba de simulación de los ciclos con instrucción Add

Por último, sobre este apartado también se realizaron pruebas con un SW, que es una instrucción de tipo I, pero a diferencia del LW requiere solo de 4 ciclos y el último corresponde a MemWrite, los resultados fueron los enseñados a continuación:

Figura 7. Prueba de simulación de los ciclos con instrucción SW

Además, se realizaron varias pruebas para la parte que unía el procesador con la memoria, y en una de estas se obtuvo lo presentado a continuación:

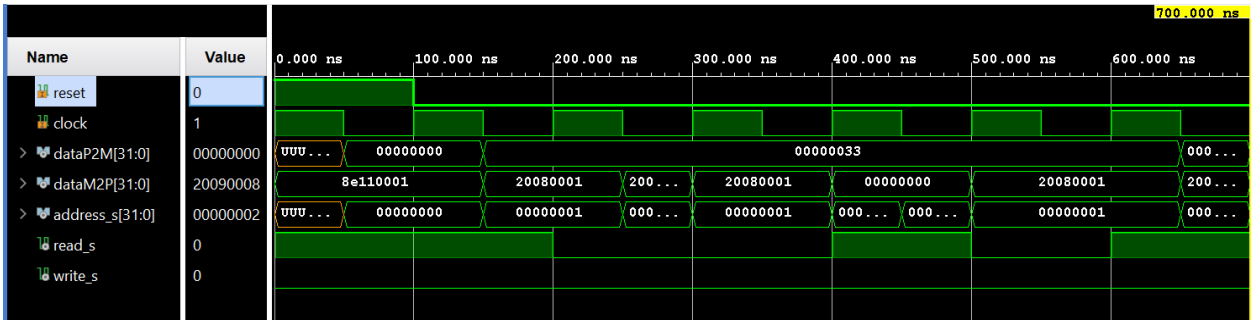

Figura 8. Simulación 1 de la memoria junto al procesador.

De la figura anterior es importante aclarar que la instrucción con la que se está trabajando es un LW, e incluso es posible saber la instrucción completa dado que la señal *dataM2P* la brinda en hexadecimal, lo cual es correcto dado que ese es el primer dato que debe viajar de la memoria al

procesador. Además, se puede observar que luego del primer ciclo de apagar el reset, se calcula la dirección de la instrucción siguiente que en este caso es la que esté en la posición 1 en hexadecimal. No obstante, se presentan algunos comportamientos no esperados en algunas señales como lo es en el "*read*" o en el dato que va del procesador a la memoria. Sin embargo, es posible afirmar que el programa no presenta ningún error de compilación y que muchas de las instrucciones son realizadas de manera relativamente bien.

Finalmente, se hizo también una segunda prueba, esto con una instrucción LW que pertenecía al código de la aplicación, donde se puede ver cómo se realiza la correcta ejecución para este caso, de la instrucción, realizando los ciclos de Fetch, Decode, MemAddress, MemRead y WriteBack.

Figura 9. Simulación 2 de la memoria junto al procesador.

IV. Conclusiones

- Se puede concluir sobre la práctica que el computador con arquitectura MIPS se desempeña de buena manera, esto dado que las señales esperadas que se activaran lo hacían para cada uno de los ciclos, fetch, decode, Execute, MemAddres, entre otras, lo cual permitió la correcta visualización paso por paso de las instrucciones tipo J, R e I.
- Se obtuvo, al final, un esquemático que conectaba la arquitectura MIPS con el procesador, esto con el fin de ejecutar las instrucciones deseadas, sin embargo, aunque este compilaba y no habían errores en la simulación se encontró que el proyecto no daba los resultados esperados, esto pudo deberse a alguna falla en la conexión o un dato indeseado.
- Finalmente, se logró tener un mejor aprendizaje sobre la teoría y asimismo sobre la práctica simulada de una arquitectura MIPS, un tema de vital importancia a lo largo del semestre el cual se pudo trabajar en el software de modelamiento Vivado.

V. Referencias

[1] PATTERSON, David A.; HENNESSY, John L. Computer Organization and Design: The Hardware/Software Interface. 5th Edition, Revised Printing. Morgan Kaufmann, 2014.Chapter 5.