Introducción

El siguiente proyecto consiste en diseñar un procesador en VHDL, el procesador será capaz de realizar una serie de instrucciones que las ejecutará después de la lectura de éstas en la memoria.

El procesador se vale de varios componentes para lograr el objetivo de ejecutar instrucciones/operaciones.

Aquí una lista de componentes:

“Control Unit”: La unidad principal de control, ésta se encarga de la interconexión de todos los demás componentes, éste es quien se encarga de hacer lo necesario ya sea usar, controlar otros componentes para “ejecutar” la operación.

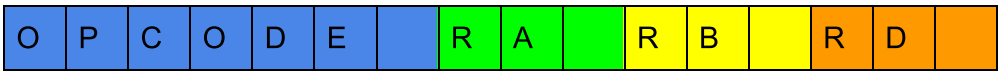
“ALU”: La unidad lógica aritmética, ésta es la que realiza las operaciones más básicas dentro del procesador, como su nombre dice generalmente las operaciones que realiza se dividen en 2 de tipo aritmético: (suma, resta, suma con acarreo, …) y tipo lógico: (and, or, xor, …) este se puede diseñar de diferentes formas y generalmente tiene banderas para complementar la información del resultado.

“Registros”: Como vimos en clase hay 2 tipos de registros, de propósito general y específico los de uso general sirven para poder realizar la instrucción que le llega al procesador, por ejemplo si se requiere sumar usamos 2 registros para los sumandos y en un tercero podemos guardar el resultado. Los de tipo específico son reservados para un tipo de información ejemplo el *“Program counter”* es quien tiene la ubicación de memoria actual de la instrucción y va en aumento para secuencialmente realizar el programa que se encuentra en la RAM. *“Instruction register”* es quien tiene la instrucción actual, entre otros…

EL procesador generalmente lo usamos para ejecutar programas, y tiene diferentes fases para conseguir esto generalmente estas son fetch, decode y execute. En mi implementación del procesador cada uno hacía lo siguiente:

- **Fetch:** Busca la instrucción que toca ejecutar en la RAM para eso establece el bus de direcciones a la dirección de la instrucción (generalmente PC) y la data_OUT de la RAM al “IR” (instruction register).

- **Decode:** Una vez estando la instrucción en “IR” se descompone según mi formato la instrucción de 16 bits:



Y pone el procesador en el estado correspondiente a la instrucción

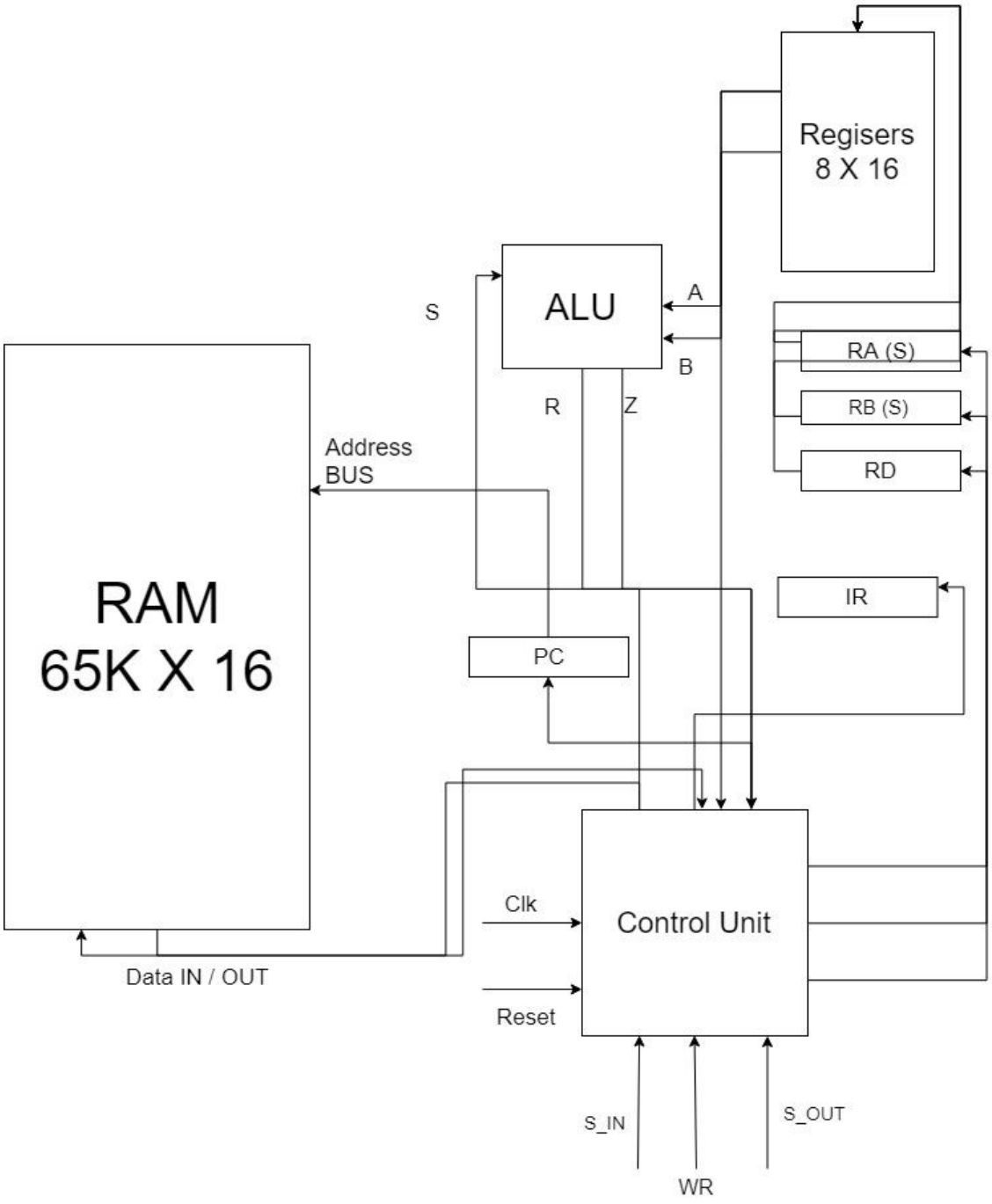
- **Execute:** Uno por cada instrucción requerida, algunas necesitaban más clocks que otras por lo que se número en orden de ejecución con el mismo nombre ejemplo: “execute_addi”, “execute_addi1”, “execute_addi2”

Para probar el procesador se nos asignó crear un programa que multiplique 2 números que van a ser suministrados por el sistema. El procesador como vimos entiende instrucciones en binario en mi caso de 16 bits, al conjunto de instrucciones se les llama lenguaje de máquina sin embargo es bien difícil escribir un programa en lenguaje de máquina por eso se creó el lenguaje ensamblador el cual consiste en traducir el código de operación en mnemónicos. Por lo que a continuación está el programa de multiplicar 2 números en ensamblador:

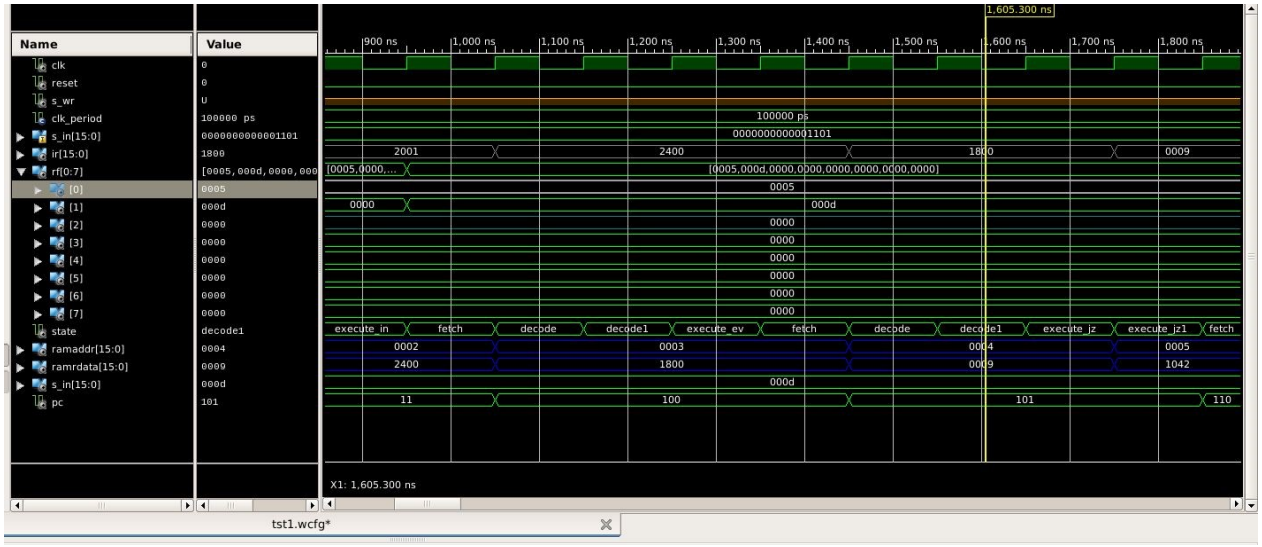
```
IN R0
IN R1
EV R0
JZ OUTPUT
PROC: ADD R2, R1
DEC R0
JNZ PROC
OUTPUT: OUT R2
HALT
```

En las 2 primeras líneas se lee la entrada más adelante el mnemónico “EV” fue creado por mí y el objetivo de este es que el registro RS pase por el ALU para que este active la bandera de Z en caso en que el valor en el registro sea 0, y así poder usar el mnemónico JZ que salta a una dirección si es que la bandera de Z está abierta. Por lo que si quiero multiplicar 0 * X el salta al output que es solo imprimir R2 que es el registro que contendrá el resultado. De lo contrario continua con el proceso de multiplicar que se convierte en una suma sucesiva cada vez se le suma al registro R2 el registro R1 que es el 2do factor y se decrementa uno el registro R0 que es el primer factor. Este proceso continúa mientras R0 no sea 0 para eso JNZ.

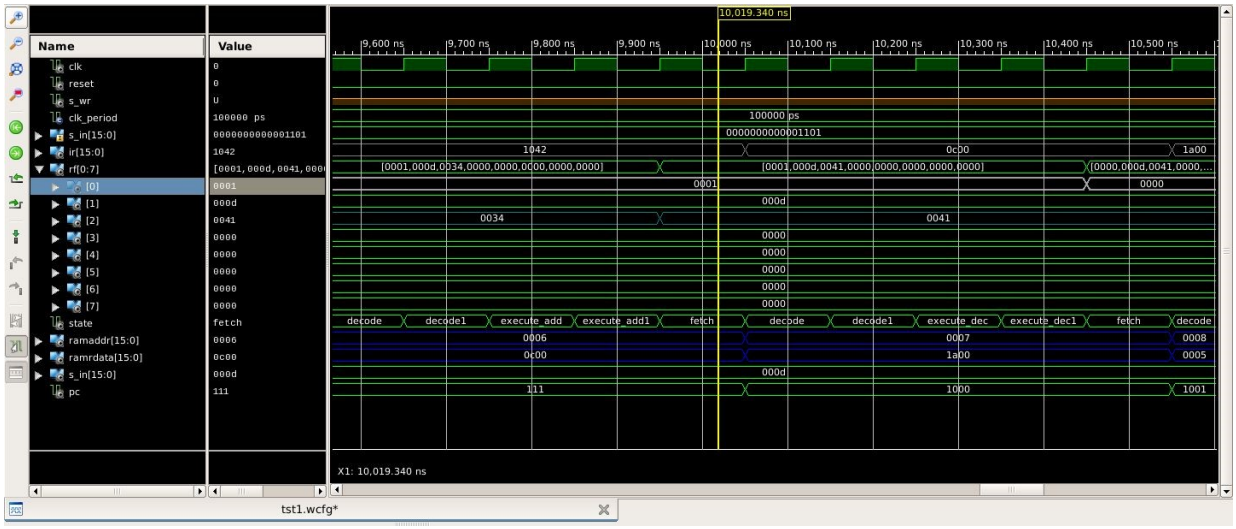
Diagrama en bloque



Test-Bench.



En este caso queremos multiplicar 13 * 5 (R0 con R1), y el resultado estará en R2.
Ro = gris, R1 = verde, R2 = cyan.



En este caso ya es la etapa final del programa después de varios ciclos de reloj, se ve como va la transición de R2 a finalmente x”41” en hexadecimal que es 65 en decimal. (Hacer ZOOM al documento para ver en detalle)

Problemas encontrados

Después de haber visto en clase el funcionamiento y la lógica de las partes más básicas del procesador. Realmente uno de los problemas no fue entender el funcionamiento del procesador sino la implementación de éste. Al no haber tenido experiencia con el lenguaje VHDL anteriormente este fue un problema a la hora de desarrollar este proyecto, sin embargo gracias al internet pude ver algunos ejemplos de implementaciones además mientras desarrollaba podía buscar cualquier consulta que tenía si bien, no aparecían todas hubo muchas que sí pude conseguir respuesta.

Otro de los problemas fue que al principio pensé que algunas operaciones se podían ejecutar en un ciclo de reloj sin embargo después de un tiempo pude darme cuenta que necesitaban más de un ciclo de reloj sobretodo aquellas que interactúan con otros módulos fuera del “control unit”. Este problema lo pude resolver añadiendo más estados para una instrucción.

En conclusión este proyecto me gustó mucho y la clase sistemas digitales en general. Porque realmente desconocía cómo funcionaba la computadora desde las partes más sencillas. Además era interesante la traducción del algoritmo en lenguaje ensamblador y luego un lenguaje de máquina. Que luego podía ejecutar el procesador.