

Memoria microcontrolador

Realizado por: Germán Bravo López Iñaki Ormaetxea Orobengoa



6 de enero de 2019

LCSE - Máster universitario en ingeniería de sistemas electrónicos

Tabla de contenido

[1. Introducción 2](#_Toc534552192)

[2. Objetivos 2](#_Toc534552193)

[3. Descripción de la arquitectura 2](#_Toc534552194)

[4. Bloques del sistema 3](#_Toc534552195)

[4.1. Interfaz RS232 3](#_Toc534552196)

[4.2. Memoria RAM 3](#_Toc534552197)

[4.3. Controlador DMA 3](#_Toc534552198)

[4.4. Memoria ROM 3](#_Toc534552199)

[4.5. ALU 3](#_Toc534552200)

[4.6. CPU 4](#_Toc534552201)

[4.7. Fichero ‘top’ 5](#_Toc534552202)

[5. Mejoras 6](#_Toc534552203)

# Introducción

Tras la realización de la interfaz de comunicación RS232 en la práctica anterior, se ha procedido en esta práctica a la creación de un microcontrolador de propósito específico. Este dispositivo, que se va a implementar en VHDL, consta de una serie de módulos principales para su funcionamiento:

* Interfaz RS232: Para recibir órdenes y poder responder a ellas.
* Memoria RAM: Para almacenar los datos recibidos.
* Controlador DMA: Que se encarga de repartir los buses según la tarea a realizar.
* Memoria ROM: Contiene la memoria del programa (instrucciones a realizar).
* ALU: Para la realización de las operaciones pertinentes.
* CPU: Microprocesador que se encarga de ejecutar las instrucciones de programa.

De esta forma se procede a explicar los objetivos y cada módulo creado.

# Objetivos

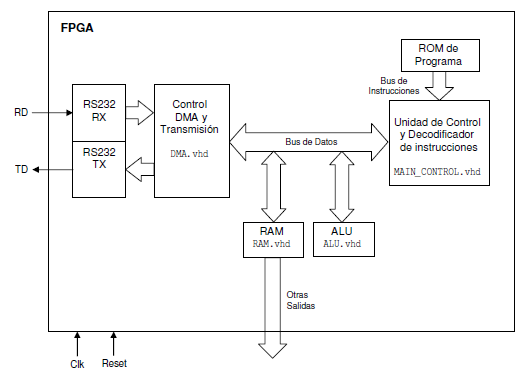
El propósito del microcontrolador es decodificar comandos que se le entregan por puerto serie y actuar según lo que se le solicite.

Para ello, se podrán realizar las siguientes funciones:

* Debe controlar 8 interruptores que encienden o apagan 8 LEDs en la FPGA.
* Cargar un valor de temperatura en un registro dedicado para el control de un termostato, el cual se mostrará por dos displays de 7 segmentos.
* Enviar información sobre el estado de alguno de los periféricos.

# Descripción de la arquitectura

Este sistema tiene una arquitectura tipo Harvard, donde los buses e instrucciones están separados. Para comprender mejor el funcionamiento explicado, en la siguiente figura se observa un diagrama de bloques a alto nivel que facilita el entendimiento.



# Bloques del sistema

## Interfaz RS232

Este módulo (*RS232top.vhd*) se realizó en la práctica anterior y se utiliza en esta de la misma forma como interfaz de comunicación. El único cambio realizado en este es la extracción del convertidor de reloj, pues este convertidor será el que proporcione la señal de reloj a todos los componentes del sistema.

## Memoria RAM

(*RAM.vhd*)

## Controlador DMA

(*DMA.vhd*)

## Memoria ROM

El contenido de este bloque (*ROM.vhd*) ha sido proporcionado por el profesor, contiene la memoria de programa en forma de constantes. Tiene dos puertos de comunicación, una entrada y una salida:

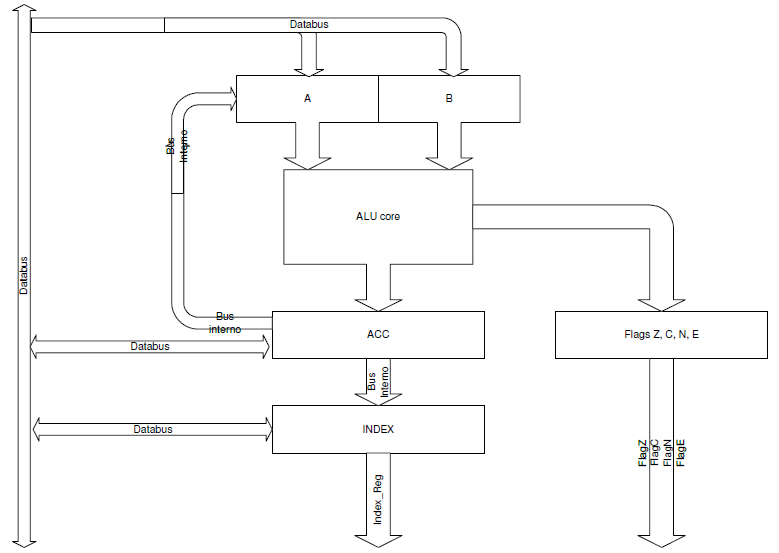
* Entrada (INS\_Addr), con la que se solicita por parte de la CPU la dirección de la instrucción.
* Salida (INS\_Bus), por el cual se manda el contenido de la instrucción solicitada

Es un bloque totalmente combinacional, proporciona el contenido de la instrucción en el mismo instante solicitado.

## ALU

Este bloque (*ALU.vhd*) es el encargado de realizar las operaciones de suma, resta, comparación, etc., que se puedan solicitar mediante las instrucciones de programa.

Tiene dos registros para carga de operandos, un acumulador, un registro específico para carga de índices destinado al acceso indexado a memoria y un registro con 4 bits de estado (Z, C, N y E).



En esta imagen se observa la funcionalidad y el rutado del interior de la ALU. En ella todas las instrucciones que se le ordenan se ejecutan en un solo ciclo de reloj.

Para su realización nos apoyamos en el paquete de constantes proporcionado por el profesor donde se recogían todas las instrucciones que se podían solicitar.

TYPE alu\_op IS (

nop, -- no operation

op\_lda, op\_ldb, op\_ldacc, op\_ldid, -- external value load

op\_mvacc2id, op\_mvacc2a, op\_mvacc2b, -- internal load

op\_add, op\_sub, op\_shiftl, op\_shiftr, -- arithmetic operations

op\_and, op\_or, op\_xor, -- logic operations

op\_cmpe, op\_cmpl, op\_cmpg, -- compare operations

op\_ascii2bin, op\_bin2ascii, -- conversion operations

op\_oeacc); -- output enable

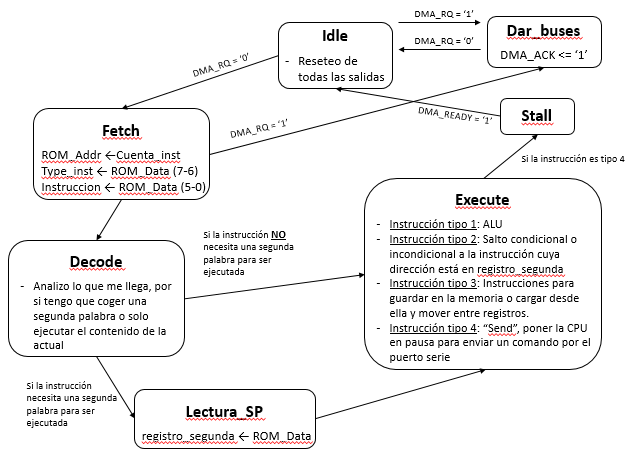
Gracias a esto y al funcionamiento de los flags según la instrucción, se realizó el diseño.

## CPU

La CPU (*MAIN\_CONTROL.vhd*) es el núcleo del sistema, ya que está máquina es la que se encarga de coger una instrucción, entender lo que significa y ejecutar su orden. Principalmente este bloque es capaz de realizar lo siguiente:

* Cuando la DMA solicite los buses, y en ese momento no se esté realizando alguna tarea crítica o que deba ser terminada, se le conceden hasta que termine con ellos.
* Recoger una instrucción de la dirección de memoria solicitada.
* Decodificar esta instrucción para saber qué hay que realizar.
* Ejecutarla, es decir general las microinstrucciones necesarias para resolver lo que se solicite en la instrucción recogida.
* Volver al estado inicial.

Para la construcción de este componente se ha realizado una máquina de estados que se muestra en la figura siguiente.



Acompañando a esta máquina de estados se ha incluido un contador de instrucciones para gestionar la lectura de las instrucciones y además poder realizar la lectura de la segunda palabra sin tener que volver a pasar por *Idle*.

Hay que tener en cuenta que el bus de datos se debe dejar en alta impedancia cuando no se utilice ya que esta señal se comparte con otros módulos que pueden cargar datos en él.

## Fichero ‘top’

Finalmente en este fichero (*PICtop.vhd*) se realizan las instanciaciones y los enrutamientos de los distintos módulos con la finalidad de conectar todas las partes del sistema completo. Para ello se definen señales para cada entrada y salida de los módulos independientes, de forma que queden conectados.

También se ha incluido en este fichero un proceso para poder pasar por los displays los valores de la temperatura que salen cableados directamente de la RAM. Como existen 8 displays en la FPGA, se deben apagar todos excepto el que se desee utilizar para mostrar un dígito u otro.

Mediante estas instrucciones se manda el dato del dígito de las unidades:

temp <= '1' & temp\_l;

disp <= "11111110";

Mediante estas otras se manda el dato del dígito de las decenas:

temp <= '1' & temp\_h;

disp <= "11111101";

Cada conjunto de instrucciones se ejecuta cada 50 microsegundos ya que se comparten salidas en la entidad.

# Mejoras

¿?