Escuela de Ingeniería Electrónica
Licenciatura en Ingeniería Electrónica
EL3313 Taller de Diseño Digital
II Semestre 2025

TEC | Tecnológico de Costa Rica

Instructivo de laboratorio

Laboratorio 4: Microcontrolador

Profesor

Luis C. Rosales

1. Introducción

El laboratorio 4 constituye el proyecto final del curso y por ello es una actividad integradora que requiere de trabajo en equipo para diseñar e implementar un sistema digital complejo. El proyecto tiene como parte principal el diseño de un microcontrolador, donde la implementación del mismo debe basarse en la arquitectura del set de instrucciones `rv32i`. Un programa, que se ejecutará en dicho microcontrolador, deberá orquestar diferentes módulos que controlan entradas de un ADC, `botones` y `switches`, salidas hacia `LEDs` y display de `7 segmentos`.

La complejidad del sistema a implementar solo es manejable con disciplina de implementación: cada módulo debe tener las pruebas (*testbenches*) necesarias para asegurar su funcionamiento correcto a nivel de pre-síntesis, post-síntesis y post-implementación en FPGA. Esto es relevante, en particular, para el microcontrolador a implementar pues de otra forma no se podrá asegurar que el programa implementado esté ejecutándose de manera correcta.

Para el desarrollo de este laboratorio se deberán acatar las siguientes indicaciones:

1. Lea y trate de comprender todo el trabajo solicitado antes de iniciarlo.
2. Durante el desarrollo del diseño y la validación, asegúrese de documentar todas las tablas de verdad, circuitos, figuras, diagramas (flujo, estado, bloques, tiempos), entre otros, que sean necesarios.
3. Para la presentación funcional, se le pedirá que muestre algunos de los circuitos propuestos, ya sea en implementación o simulación. Dicha selección se hará el día de la presentación.
4. Asegúrese de que el avance del proyecto y los aportes de cada estudiante queden debidamente protocolados con suficientes "commit" en el repositorio a lo largo de todo el periodo

disponible para trabajar en el laboratorio. Además, asegúrese de registrar todo el trabajo identificado y que continuamente revisan el trabajo asignado y actualizan su estado.

5. Todos los bloques de diseño deben ser **sintetizables** en SystemVerilog. El sistema debe funcionar con un único reloj de 10 *MHz*.

2. Objetivos

1. Utilizar las herramientas del diseño digital para construir un sistema basado en un microcontrolador, lo que incluye un componente de software (programa que corre el microcontrolador) que deberá ser desarrollado en lenguaje ensamblador.
2. Diseñar la implementación de hardware en forma modular, separando tareas de control del procesamiento de los datos.
3. Diseñar e interconectar periféricos mapeados en memoria en un sistema de procesamiento básico.
4. Comprender cómo utilizar *IP-Cores* como parte de un diseño digital complejo.
5. Profundizar sobre el enlace e interacción entre software y hardware digital.
6. Plantear una estrategia de trabajo en equipo adecuada para diseñar un sistema complejo e implementarlo con restricciones de tiempo existentes.

3. Investigación previa

Para el desarrollo de este proyecto se debe indagar sobre los siguiente aspectos:

1. Investigue sobre la arquitectura RISC-V. Preste especial atención a las instrucciones que forman parte del conjunto básico de instrucciones para números enteros de 32 bits, `rv32i`.
2. Investigue sobre las diferencias entre un lenguaje de programación como C y ensamblador. Explique qué es *bare-metal programming*.
3. Investigue sobre cómo se almacenan los datos en una memoria. ¿Qué es little-endian y big-endian?
4. Explique el concepto de *periféricos mapeados en memoria*. ¿Cuál es el método utilizado para leer o escribir datos/instrucciones a un periférico?
5. Investigue sobre el uso de los *IP-Cores* en Vivado para memorias RAM y ROM, así como el ADC para entradas analógicas.

4. Procedimiento

A continuación se describen aspectos fundamentales para el desarrollo de las diferentes etapas de este proyecto.

4.1. Resumen

En este laboratorio se desarrollará un sistema basado en un microcontrolador el cual será capaz de monitorizar la temperatura ambiente de una oficina. El dato de temperatura se adquirirá de un sensor analógico de temperatura y será mostrado mediante el display de 7 segmentos.

4.2. Microcontrolador

Usted deberá diseñar e implementar un microcontrolador de 32 bits. Para esto, puede reutilizar la unidad aritmético-lógica (ALU) y el banco de registros desarrollados como parte del laboratorio 1 y 2. Su microcontrolador debe implementar las instrucciones necesarias del subconjunto `rv32i` para la ejecución de su programa. Considere la siguiente lista como una base de las instrucciones que su procesador debe soportar. Asegúrese de que su microcontrolador sea **sintetizable**, de otra manera no lo podrá ejecutar en la tarjeta con FPGA.

```
lw, sw,
sll, slli, srl, srli, sra, srai,
add, and, xor, or, sub,
addi, andi, xori, ori,
beq, bne, blt, bge,
slt, slti, sltu, sltui,
jal
```

La Figura 1 muestra los componentes principales del sistema computacional a desarrollar. El microcontrolador utilizará una memoria ROM para almacenar el programa y una memoria RAM para almacenar los datos. Dichas memorias se podrán acceder mediante buses **independientes**. Como se observa en la Figura 2, estos buses corresponden a `ProgAddress_o[31:0]` y `ProgIn_i[31:0]` para la memoria de instrucciones, y `DataAddress_o[31:0]`, `DataOut_o[31:0]` y `DataIn_i[31:0]` para la memoria de datos. Utilice esta nomenclatura para estos puertos. Note que los datos a los que el microcontrolador tiene acceso (`read`) pueden provenir tanto de la memoria RAM de datos como de los módulos periféricos descritos en este documento, como se representa en la Figura 1.

El microcontrolador deberá operar a una frecuencia de 10 *MHz*. Dicha señal de reloj deberá entrar al diseño mediante la entrada `clk_i` (Figura 2). Esta señal de reloj también es común con los otros bloques del sistema. Genérela utilizando un PLL.

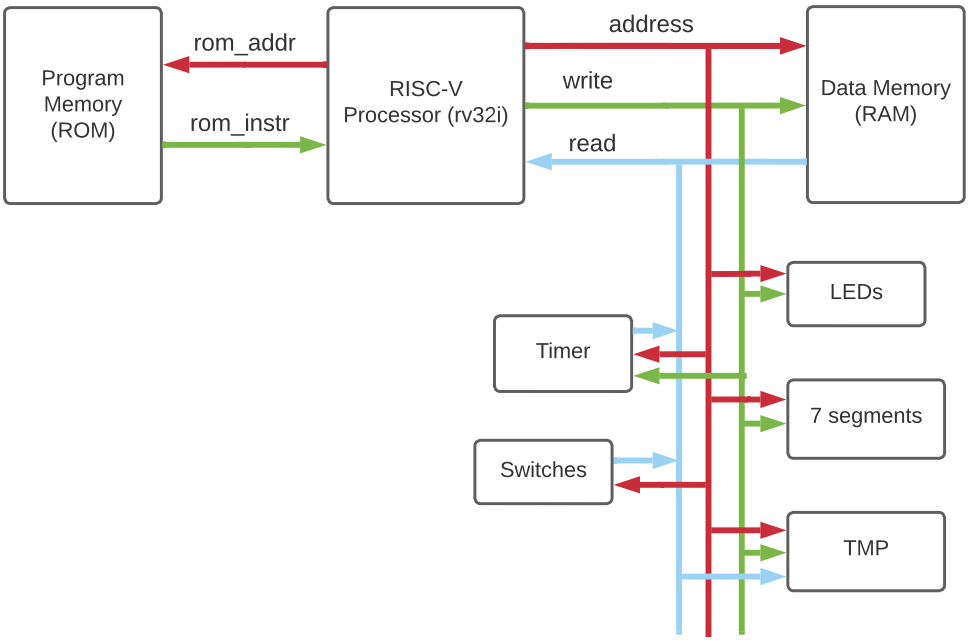

Figura 1: Diagrama de bloques del microcontrolador a desarrollar.

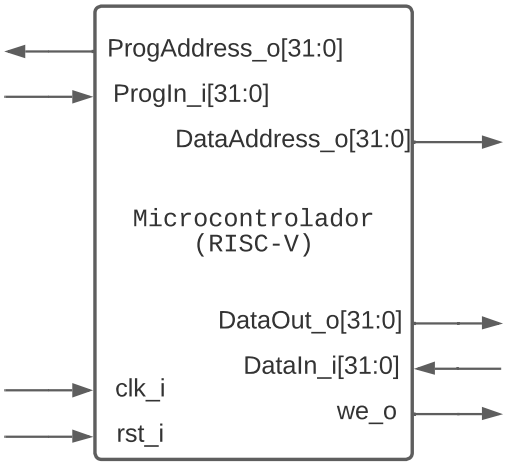

Figura 2: Diagrama de alto nivel del *core* para el microcontrolador.

La Figura 3 muestra el mapa de memoria considerado para el diseño de este sistema computacional. La memoria de programa tiene un tamaño recomendado de 512 palabras de instrucción, i.e., 2 KiB. Esta memoria de programa debe colocarse en el espacio entre `0x0000` y `0x0fff` en el mapa de memoria, siempre iniciando en la posición `0x0000`.

La memoria de datos posee un tamaño de 256 palabras, i.e., 1 KiB, y se debe colocar en el espacio de memoria entre `0x1000` y hasta `0x1fff`, siempre iniciando en la dirección `0x1000`. Las siguientes direcciones en memoria corresponden a los registros internos de los periféricos:

- `Switches`: registro de datos en `0x2000`.
- `LEDs`: registro de datos: `0x2004`.
- `7-segmentos`: registro de datos: `0x2008`.
- `Timer`: registro de control en `0x2018`, registro de datos en `0x201C`.
- `Sensor de temperatura`: registro de control en `0x2030`, registro de datos en `0x2034`.

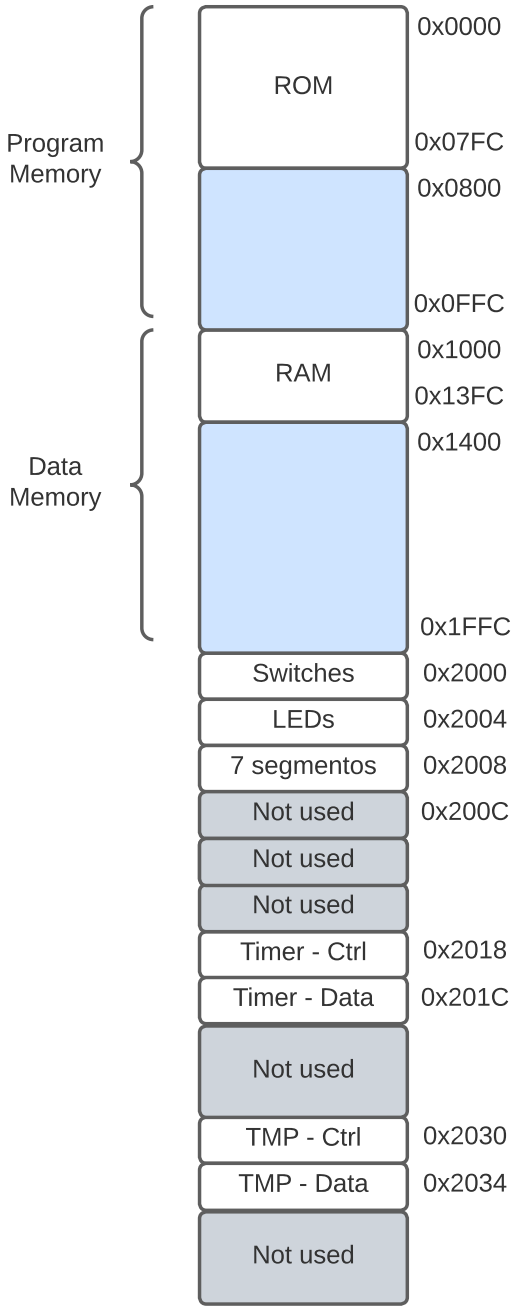

Figura 3: Mapa de memoria.

4.3. Periféricos

El microcontrolador a desarrollar debe ser capaz de comunicarse con los siguientes periféricos: `temporizador`, `7 segmentos`, `LEDs`, `Switches/botones` y un ADC para obtener mediciones de temperatura.
Para el desarrollo de estos periféricos y sus interfaces se recomienda hacer referencia a los diseños planteados en laboratorios anteriores, particularmente para la interfaz serie, además de tomar previsiones con respecto a circuitos de acondicionamiento necesarios (e.g., sincronizador y anti-rebotes).

La comunicación entre el procesador, los periféricos y la memoria de datos se muestra de manera simplificada en la Figura 1. Esta se realiza por medio de un bloque que administre los buses de datos (entrada y salida, `write` y `read`), la señal de `write_enable` y `read_enable`, y el bus de direcciones, de tal manera que multiplexe las señales de acuerdo a lo establecido en el mapa de memoria (Figura 3).

- **Switches/botones**: Este bloque permite mapear los interruptores y 4 botones de la tarjeta a un registro accesible por el procesador. Este módulo debe considerar **sincronización y anti-rebote**. Se recomienda colocar los 16 interruptores como los 16 bits menos significativos del registro e inmediatamente después los botones. Note que se mencionan solo cuatro botones ya que requiere reservar un botón para el *reset* del sistema.
- **LEDs**: Este bloque permite mapear los LEDs de la tarjeta a un registro accesible por el procesador. Colóquelos como los 16 bits menos significativos del dato en el registro correspondiente.
- **7 segmentos**: Este bloque permite mapear los *displays* de 7 segmentos y su lógica de control a un registro accesible por el procesador.
- **Timer**: Este bloque permite contar con un temporizador programable (en su cuenta). Dicho módulo cuenta con un registro de control y uno de datos para indicar el valor al que deberá contar. Dicho valor volverá a cero en el momento en que dicha cuenta termine y se mantendrá en ese valor hasta que se reinicie una cuenta.
- **TMP**: Este módulo presenta como interfaz un registro de control y uno de datos. Mediante el registro de control usted le indicará al módulo la captura de un nuevo dato de temperatura y para cuál de los dos sensores se desea realizar dicha lectura. Puede considerar sensores como el TMP36 o el LM35. A lo interno, este módulo deberá utilizar el IP XADC para las conversión analógica-digital. Una vez que la conversión se haya dado, el valor digital obtenido se deberá colocar en el registro de datos y una bandera en el registro de control indicará que se tiene una nueva lectura de temperatura.

4.4. Aplicación

Deberá desarrollar un programa, empleando lenguaje ensamblador para `rv32i`, que se ejecutará en su microcontrolador. Dicho programa deberá leer el sensor de temperatura cada 1, 2, 5 o 10 segundos, según se indique mediante uno de cuatro switches. Una vez realizada la lectura y

convertido el valor de binario a decimal, dicho valor será mostrado en el display de 7 segmentos. Utilice LEDs como parte de su diseño para indicar que es lo que su aplicación está realizando (e.g., en espera de finalización de temporizador, o tomando lectura del sensor de temperatura). Se recomienda diseñar un mapa de indicadores para la presentación final.

5. Evaluación

Las fechas oficiales para este laboratorio son las siguientes. Recuerde que para las revisiones, tanto intermedia como final, debe contar con simulaciones post-síntesis con temporización e implementación en FPGA.

- **Inicio:** 07/Nov/2025
- **Planteamiento de la solución:** 18/Nov/2025
- **Revisión intermedia:** 27/Nov/2025 (*"Hello World"* en RISC-V *core*)
- **Revisión final:** 05/Dic/2025.

El presente laboratorio tiene un valor del **24 %** de la nota final del curso y se evaluará de la siguiente manera:

1. Funcionalidad intermedia: RISC-V *core* 15 %
2. Funcionalidad final: Aplicación 40 %
3. Planteamiento de la solución 15 %
   (diagramas de bloques, flujo, estado, etc)
4. Documentación técnica del sistema desarrollado 10 %
   Se distribuirá el peso de la documentación en:
   a) El código sigue la guía de codificación, además de incluir documentación en el código apropiada (al menos documenta: puertos, constantes, parámetros, tipos de variable y descripción de bloques). (5 %)
   b) Descripción funcional de unidades desarrolladas (15 %)
   c) Datos, resultados (tablas, gráficos, simulaciones, reportes de herramientas), y análisis de datos y resultados, comparaciones. (5 %)
   Detalle los errores y complicaciones del diseño, si se resolvieron y cómo se resolvieron. Si no se resolvieron, ¿cómo se resolverían?

   Recuerde citar adecuadamente en su documentación cualquier información proveniente de fuentes bibliográficas.
5. Uso de GitHub 10 %
6. Planeamiento 10 %