Proyecto Final

Construcción de un computador

Profesor: Ricardo Finger C.
Profesor Auxiliar: Sergio Saavedra Torres
Ayudantes de Laboratorio: Alonso Caviedes, Diego González, Daniel Vásquez, Néstor Henríquez.

Importante

Este laboratorio tiene un plazo de 4 semanas. Debe entregar un reporte de avance en la segunda semana. Su nota es "coeficiente 2". Su realización es en parejas.

Objetivos

Esta experiencia tiene como objetivo la construcción de un computador de 8-bits, utilizando los bloques COUNTER, ROM, ALU y WREG, construídos en laboratorios pasados.

Su máquina debe ser capaz de calcular la sumatoria:

$$S = a\sum_{i=1}^{n}\left(\frac{1}{2}\right)^{i} = \frac{a}{2} + \frac{a}{4} + \frac{a}{8} + ...$$

Dónde $a = 1 + Mod(x + y, 7)$, siendo x e y los RUTs de los integrantes del grupo (sin dígito verificador) y $Mod(a, b)$ la operación módulo (resto de la división entera a/b).

Trabajo a realizar

Para realizar esta sesión de laboratorio se requiere:

- Software *SimulIDE*.
- Bloques COUNTER, ROM, ALU y WREG, implementados en *SimulIDE*.
- Multiplexores de 8 entradas. Disponibles en la librería de *SimulIDE* [1].
- Flip Flops D. Disponibles en la librería de *SimulIDE* [2].
- Compuertas lógicas básicas. Disponibles en la librería de *SimulIDE*.
- Displays de 7-segmentos disponibles en u-cursos.

A continuación se listan las actividades de la sesión. Realice el esquemático de los circuitos de forma ordenada, de este modo, facilitará el análisis posterior. Ante cualquier duda o problema que surja con el programa o las simulaciones, consulte a los ayudantes.

[1] Alternativamente, puede utilizar el componente "Función" para implementar un multiplexor 4 a 1 ó 2 a 1.
[2] Alternativamente, puede usar el componente "Latch D"de 8-bits con "Trigger- Clock

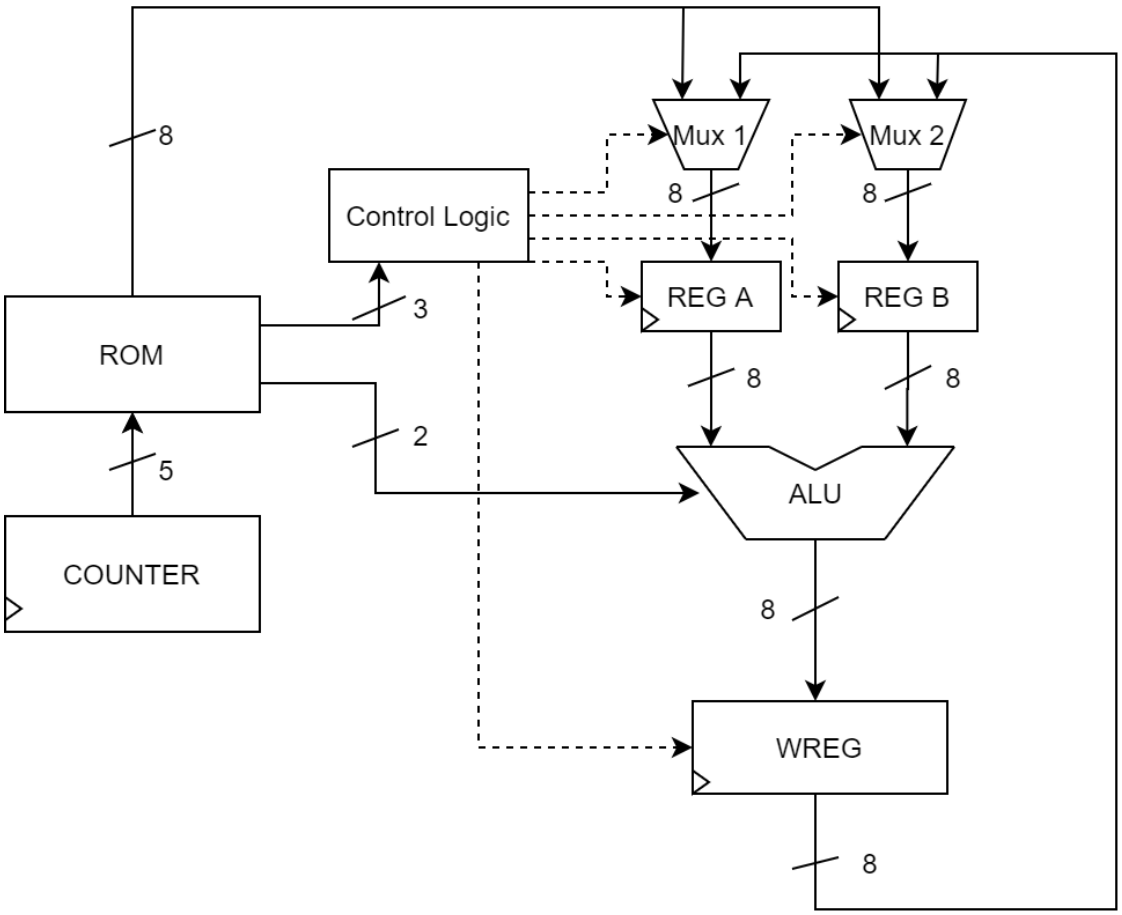

Figura 1: Diagrama de Bloques Laboratorio 4

1. Lenguaje ensamblador

El conjunto de instrucciones que debe ejecutar su computador es el siguiente:

Cuadro 1: LENGUAJE ENSAMBLADOR

Instrucción	Descripción
ADD	Suma los valores almacenados en REG A y REG B. El resultado se guarda en WREG.
OR	Realiza la operación OR bit-a-bit entre REG A y REG B. El resultado se guarda en WREG.
MOV A,W	Copia el valor de REG A en WREG.
LOAD A	Carga un valor desde la ROM a REG A.
LOAD B	Carga un valor desde la ROM a REG B.
SHIFT W	Desplaza los bits de WREG a la derecha. El resultado se guarda en WREG.
MOV W,A	Copia el valor de WREG en REG A.
MOV W,B	Copia el valor de WREG en REG B.

Para codificar cada instrucción, usted dispone de 13 bits (ROM Output) repartidos de la siguiente forma:

- 8 bits de datos. Para cargar un valor de la memoria de programa en algún registro.
- 5 bits de instrucciones. Repartidos en 2 bits de control para la ALU y 3 bits para controlar elementos adicionales.

Se pide que codifique cada instrucción del lenguaje ensamblador (es decir, usando palabras binarias de 13-bits), para ello:

a Escoja los bits que usará para el control de la ALU (2-bits) y para el control de registros y muxes (3-bits).

b Defina el valor que deben tener los 8 bits. Use "bbbbbbbb" si la instrucción requiere un operando, o "xxxxxxxx" si el operando no es utilizado.

Luego, su conjunto de instrucciones debe quedar de la forma:

Cuadro 2: LENGUAJE DE MÁQUINA

	Bits de Instrucción	**Bits de Datos**
Instruction 0	$i_4 i_3 i_2 i_1 i_0$	$d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0$
.	.	.
.	.	.
.	.	.
Instruction 7	$i_4 i_3 i_2 i_1 i_0$	$d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0$

2. Lógica de control

Los elementos adicionales a controlar con los bits de instrucción (no usados en la ALU) y su función se describen a continuación:

Cuadro 3: Elementos adicionales (Control Logic)

Elemento	**Bits de Control**	**Descripción**
Mux 1	1-bit	Selecciona el origen de los datos dirigidos a REG A, entre WREG o ROM
Mux 2	1-bit	Selecciona el origen de los datos dirigidos a REG B, entre WREG o ROM
REG A	1-bit	Habilita o deshabilita la escritura de datos en REG A.
REG B	1-bit	Habilita o deshabilita la escritura de datos en REG B.
WREG	1-bit	Habilita o deshabilita la función de desplazamiento de WREG.

Se pide que implemente una lógica combinacional de control que decodifique los 3-bits dedicados al control de registros y muxes en las señales de los 1 bit que controlan cada componente.

3. Construcción del Circuito

Implemente en *SimulIDE* el circuito de la Figura 1. Para ello:

a Utilice los bloques ya construidos en laboratorios anteriores. Sea cuidadoso y ordenado en sus conexiones y en la distribución del espacio de trabajo.

b Implemente la lógica de control con compuertas lógicas básicas o bloques funcionales. Alternativamente, puede utilizar el componente "Función"de la librería.

c Para REG A y REG B, considere agregar una lógica adicional para las señales de reloj y de control (habilitación), para su correcto funcionamiento.

d Compruebe el funcionamiento de su circuito al realizar conexiones, para esto, puede ser útil usar "Led (Barra)" y habilitar la animación ($Propiedades > Animate = True$).

e Conecte displays LED disponibles en u-cursos en los registros: A, B y W. Pruebe operaciones básicas como cargar y sumar registros.

4. Implementación del Programa

Con su computadora lista, es hora de implementar el programa que calcule:

$$S = a\sum_{i=1}^{n}\left(\frac{1}{2}\right)^{i} = \frac{a}{2} + \frac{a}{4} + \frac{a}{8} + ...$$

Use un tipo de dato *binary point = 5*, esto es, 3-bits para la parte entera y 5-bits para la parte decimal. Cambie los displays para observar el valor en los registros: A, B y C, con el tipo de dato recién definido. Fuera de esto, ¡Note que esta definición no cambia su hardware!

Escriba un programa que calcule, la sumatoria pedida y guarde el resultado en REG B. Para ello:

a Escriba una tabla con sus instrucciones en "assembler" que muestre, instrucción por instrucción ejecutada, el valor (decimal) actual en cada registro: A,B y W. Note que cada instrucción se ejecuta en un único ciclo de reloj.

b Note que necesitará instrucciones de inicialización "I" y de bucle "J". Éstas últimas se repetirán indefinidamente en su computador.

c Compruebe que el conjunto de instrucciones realice la suma pedida, identificando un patrón de repetición.

d Traduzca sus instrucciones de "assembler" a lenguaje de máquina a cargar en su memoria ROM.

e Configure su contador de tal forma que:

- En el primer ciclo ejecute las (I+J) instrucciones de inicialización y primera iteración del bucle.
- En los ciclos posteriores sólo ejecute las (J) instrucciones del bucle, indefinidamente.

f Encienda la simulación y grabe los resultados capturando su pantalla en vídeo. Relate brevemente el funcionamiento del computador.

Responda:

- ¿Hasta qué valor puede aproximar su computador la suma? Explique.
- ¿Qué sucede con la ejecución después de un largo tiempo? ¿Es aún válido el resultado? Comente.

Entregas

Entrega Parcial

Se debe entregar un informe desarrollando las secciones:

- Introducción
- Lenguaje Ensamblador (1)
- Lógica de Control (2)
- Implementación del Programa (4), apartados (a) hasta (d).
- Conclusiones (parciales)

Note que esta entrega es puramente teórica y se puede realizar sin haber implementado el circuito. Si usted ya implementó el circuito puede entregar mas partes, o la entrega final completa.
Fecha de entrega: Viernes 26 de Junio 23:59 hrs.

Entrega Final

Complemente el informe de su entrega parcial, desarrollando completamente todas las secciones:

- Introducción (puede extender)
- Lenguaje Ensamblador (igual que en entrega parcial)
- Lógica de Control (igual que en entrega parcial)
- Construcción del circuito (nueva sección, incluir esquemáticos)
- Implementación del Programa (incluir apartados e y f)
- Análisis (nueva sección)
- Conclusiones (extender)

Esta entrega incluye el informe en formato *PDF*, los archivos de simulación (.simu y .data) y el link del vídeo explicativo.
Fecha de entrega: Viernes 17 de Julio 23:59 hrs.