Simulación de una máquina de Turing mediante circuitos digitales

Daniel Artavia Cordero, Gloriana Mora Villalta y Luis Carlos Quesada Rodríguez

I. Introducción

EL objetivo de este proyecto es el diseño e implementación de una máquina de Turing a través de circuitos digitales, lenguaje máquina y microprogramación.

De forma general, una máquina de Turing es una entidad computacional abstracta que consta de un dispositivo de control y una secuencia infinita de casillas. Dicho dispositivo puede adoptar un conjunto finito de estados, y realiza operaciones de lectura, escritura y desplazamiento en la cinta. Su comportamiento estará determinado por un grupo de instrucciones en lenguaje máquina que definen la solución a un problema específico.

Este proyecto es desarrollado mediante el *software* de simulación de circuitos digitales Logisim, siguiendo la arquitectura de von Neumann, y con instrucciones de 32 *bits* en formato *RISC*.

II. Marco Teórico

La teoría de autómatas gira en torno al estudio de los modelos computaciones abstractos, así como aquellos problemas que sean solubles a través de estos [1]. Una de las invenciones más destacables en este área fue desarrollada por Alan Mathison Turing, quien introdujo una entidad matemática abstracta a través de la cual demostró la existencia de problemas irresolubles para una máquina, y que resultó ser un indicio de los computadores digitales modernos.

Una máquina de Turing puede ser visualizada como una cinta infinita de celdas, cada una de las cuales contiene un símbolo proveniente de un alfabeto finito. Sobre dicha secuencia actúa un dispositivo de control llamado cabezal, capaz de adoptar un número finito de estados y encargado de operaciones de lectura y escritura [1]. Una representación en alto nivel podría observarse como la figura 1.

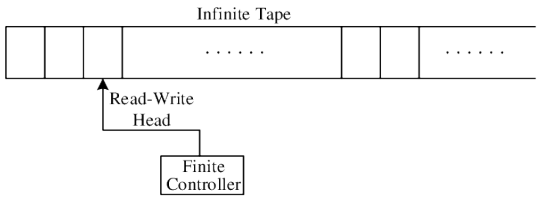

Figura 1. Representación de una máquina de Turing

En primera instancia, la máquina recibe una hilera de símbolos que son situados en la cinta y, dado que esta se extiende infinitamente en ambas direcciones, las celdas restantes almacenan un símbolo especial que representa vacío. El cabezal se posiciona en la celda más izquierda de la entrada, lee su contenido y de acuerdo con su estado actual efectúa tres pasos: cambia de estado, escribe un nuevo símbolo en la celda recién leída, y se desplaza una posición hacia la derecha o izquierda.

Formalmente, se dice que una máquina de Turing consiste en una 7-tupla MT (Q, Σ, Γ, δ, *q0, qaccept, qreject*), tal que:

- Q: Es el conjunto finito de estados del cabezal.
- Σ: Es el conjunto finito de símbolos válidos de entrada.
- Γ: Es el conjunto total de símbolos de la cinta, el cual incluye los símbolos válidos de entrada y el de vacío.
- δ: Es la función de transición δ(q,x_i), donde q $\in$ Q y x_i $\in$ Γ.
- *q0, qaccept, qreject*: Es el estado inicial, de aceptación y de rechazo del cabezal, respectivamente.

Por otro lado, cabe destacar que el surgimiento de esta abstracción computacional fue contemporáneo con el desarrollo de la arquitectura de von Neumann, cuya innovación residía en su capacidad para almacenar las instrucciones de un programa y sus respectivos operandos en memoria ya que, anteriormente, cada computador era diseñado y construido con propósitos específicos. De acuerdo con [2] esta arquitectura consta de tres componentes principales que se ilustran en la figura 2: la unidad central de procesamiento, los módulos de memoria y los dispositivos de entrada y salida.

La unidad central de procesamiento (*CPU*) está constituida por la unidad de control (*CU*), la unidad aritmético-lógica (*ALU*) y registros de uso general o determinado. La unidad de control orquesta el flujo de ejecución de instrucciones mediante la generación de señales de control. Toda operación generada bajo una señal de control se denomina microoperación. La unidad aritmético-lógica se encarga de las operaciones como sumas, restas, multiplicaciones, divisiones y comparaciones. Los registros son utilizados para almacenar y transferir datos e instrucciones de forma temporal.

En lo relacionado con los módulos de memoria se destaca la *ROM (Read Only Memory)*, el cual es un dispositivo de almacenamiento secundario no volátil en el que se guardan los datos, programas de propósito general, e instrucciones de arranque de la máquina que permanecerán inalterables. Por el contrario, la *RAM (Random Access Memory)* es de almacenamiento primario volátil y permite la lectura y escritura de su contenido.

Por último, los dispositivos de entrada y salida le permiten a la máquina establecer comunicación con el usuario para recibir o enviar información.

Ahora bien, como se mencionó anteriormente, la *CPU* posee

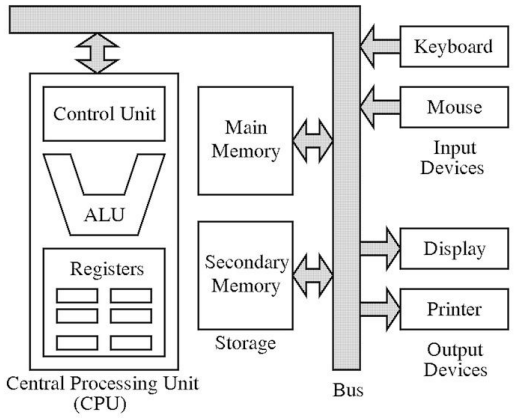

Figura 2. Componentes de la arquitectura von Neumann

registros con utilidades específicas. Según [3] algunos de ellos serían descritos como sigue:

- *PC (Program Counter)*: Guarda la dirección de la siguiente instrucción a ser traída desde memoria (*fetch*).
- *IBR (Instruction Buffer Register)*: Contiene la instrucción traída desde memoria.
- *IR (Instruction Register)*: Contiene el código de la operación mientras es decodificado por la unidad de control.
- *MBR (Memory Buffer Register)*: Se utiliza para recibir una palabra (*word*) proveniente de la memoria o a ser almacenada en la misma. También los dispositivos de entrada y salida lo utilizan para recibir o enviar información.
- *MAR (Memory Address Register)*: Especifica la dirección de memoria en la que se debe leer o escribir la palabra del *MBR*.

Además, las computadoras basadas en dicha arquitectura siguen un ciclo de ejecución comúnmente denominado *fetch, decode and excecute*. Durante el ciclo de *fetch* el código de operación de la siguiente instrucción se carga en el *IR*, y la porción de direccionamiento en el *MAR*. Esta instrucción podría ser tomada directamente del *IBR*, o bien ser traída desde memoria y cargada sucesivamente en el *MBR*, *IBR*, *IR* y *MAR*. Una vez concretada la etapa anterior, la unidad de control interpreta el código de operación y envía las señales de control pertinentes para la transferencia de datos y la ejecución de operaciones.

Para finalizar, es importante tomar en consideración que cada microprocesador puede implementar un formato de instrucciones distinto. En este caso, se hará énfasis en la arquitectura de instrucciones *RISC (Reduced Instruction Set Computing*), que comparte las siguientes características [4]:

- Ejecución de una instrucción por ciclo de máquina: Un ciclo consiste en traer dos datos desde sus registros, operar sobre ellos a través de la unidad aritmético-lógica, y guardar el resultado en un registro.
- Operaciones de registro a registro: A excepción de las instrucciones *LOAD* y *STORE* que requieren acceso a memoria, esta arquitectura optimiza su rendimiento a partir del uso de registros, y enfatiza el principio de localidad espacial y temporal.
- Formato de instrucción simple: La longitud de una instrucción es fija e inscrita en los límites del tamaño de palabra, lo cual optimiza el proceso de *fetch* y evita los fallos de paginación. Asimismo, cada porción de bits representa un campo fijo, por ejemplo, el código de operación y los operandos, de forma que la decodificación y acceso a operandos ocurre simultáneamente.
- Modo de direccionamiento simple: En la mayoría de casos, los operandos de una instrucción son recuperados mediante direccionamiento de registros, lo que es más eficiente que los accesos directos a memoria principal.

III. Diseño preliminar

Esta implementación seguirá la arquitectura de von Neumann, cuyos componentes principales son la unidad central de procesamiento (*CPU*), los módulos de memoria, y los dispositivos de entrada y salida. De forma preliminar será adaptado al desarrollo de una máquina de Turing, como sigue:

1. Módulos de memoria
   La mayor limitante en el diseño y desarrollo de una máquina de Turing es la definición del tamaño de la cinta de datos ya que, físicamente, no es posible reservar memoria infinita. Por lo tanto, se decide hacer una representación finita de la misma utilizando una memoria *RAM*, que permitirá la lectura y escritura de su contenido. Sus celdas contendrán un símbolo representado en siete *bits* y tendrá una capacidad total de 1KB.
   Asimismo, el dispositivo de control de la máquina requiere de un conjunto de instrucciones que le indiquen cómo debe comportarse en función de su estado actual y el símbolo recién leído en la cinta, lo cual involucra 3 factores: el estado al que debe transicionar, el nuevo símbolo que debe escribir, y en qué dirección debe desplazarse. Lo anterior será definido por el usuario y específico al problema que desee solucionar, además, se mantendrá inalterado durante la ejecución del programa. Por los motivos anteriores, estas instrucciones serán almacenadas en memoria una *ROM* con capacidad total de 1KB, su formato es descrito en detalle en la siguiente sección.

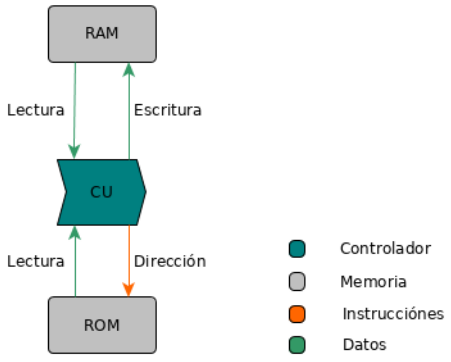

Figura 3. Interacciones entre la unidad de control y los módulos de memoria.

Por último, cabe destacar que durante el flujo de ejecución del programa se requerirá accesar la memoria *ROM* secuencialmente hasta localizar la instrucción adecuada, lo cual es costoso en términos de eficiencia temporal. Por ello se plantea utilizar una segunda memoria *ROM* de menor tamaño, en la que se almacenarán las direcciones de memoria de las instrucciones que contengan un estado en específico dentro de la memoria *ROM* principal. De esta manera, cuando se esté localizando una instrucción en la memoria principal se tendrá garantía de que va estar en un rango de direcciones determinado, y no tendrá que recorrerse por completo en el peor de los casos.

2. Formato de instrucciones
   En esta implementación se decide utilizar una arquitectura de 32 *bits* para la representación de instrucciones, cuya estructura puede observarse en la figura 4 y consiste en lo siguiente:
   - El estado interno del dispositivo de control se representa con ocho *bits*.
   - Los símbolos de la cinta se representan con siete *bits*.
   - La dirección en que se debe desplazar el dispositivo de control (izquierda o derecha) se representa con un *bit*.
   - Existe un *bit* sobrante, que será el más izquierdo de la instrucción, sujeto a cambios futuros de diseño.

   Asimismo, el conjunto de instrucciones para resolver un problema determinado se espera recibir en la memoria *ROM* por parte del usuario, por tal razón, se considera utilizar una arquitectura basada en microinstrucciones que puedan ser ejecutadas, de preferencia, simultáneamente. Por la forma en que se manejan las instrucciones lo más similar sería una arquitectura del tipo *RISC*, ya que se busca minimizar el acceso a memoria para obtención y uso de datos, utilizando como memoria temporal el conjunto mínimo necesario de registros para llevar a cabo las funciones de la máquina de Turing.

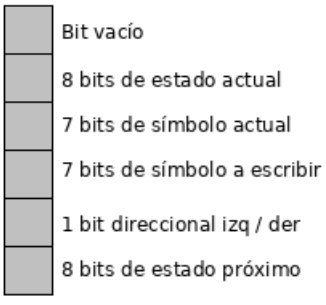

Figura 4. Estructura de una instrucción

3. Unidad central de procesamiento (*CPU*)
   La unidad central de procesamiento consta de un contador de programa *(PC)* que se encargará de los desplazamientos a través de la memoria *RAM*, un registro con el símbolo leído en la celda actual, y un registro con el estado actual del cabezal. Estos dos últimos serán empleados por una unidad de búsqueda como criterios de localización de instrucciones en memoria *ROM*, y que finalmente serán colocadas en un registro de instrucción (*IBR*). Seguidamente, la unidad de control (*CU*) deberá decodificar dicha instrucción y transmitir una serie de señales que permitan actualizar los valores del registro de estado, de símbolo actual con su escritura en memoria *RAM*, y el desplazamiento del cabezal hacia la izquierda o derecha en memoria mediante el incremento o decremento del contador de programa (*PC*).

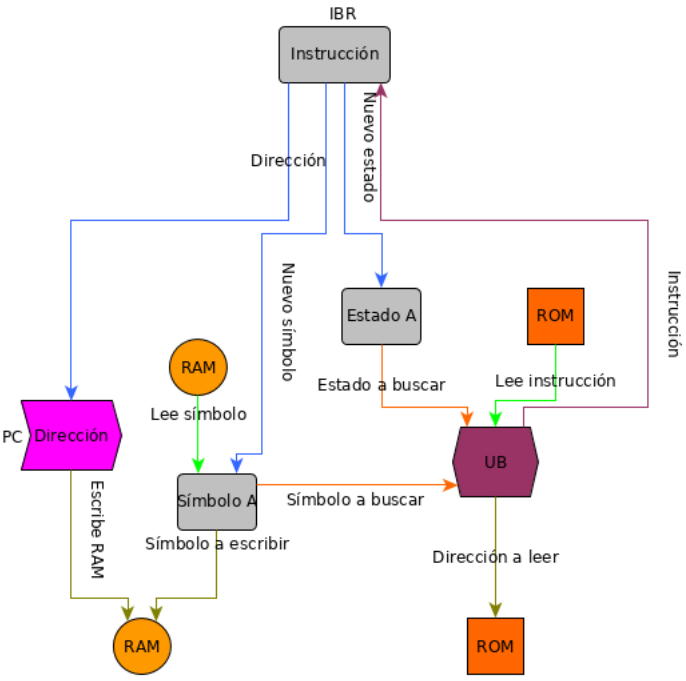

Figura 5. Diagrama de bloques de la unidad central de procesamiento

4. Dispositivos de entrada y salida
   Se espera que el usuario introduzca el conjunto de instrucciones para dar resolución a un problema específico así como los datos sobre los que desea operar mediante archivos de texto.
   Por otro lado, se implementará una visualización que le permita al usuario seguir la ejecución del programa, incluyendo transiciones de estado del dispositivo de control, lecturas y escrituras, desplazamientos y resultado final. Los estados podrían representarse con *LEDS* y los símbolos leídos o escritos con una pantalla.

III-A. Decisiones y consideraciones

Los siguientes planteamientos están sujetos a cronograma:

- La responsabilidad del usuario es cargar un archivo de texto con el conjunto de instrucciones específico a su problema, sin embargo, la elaboración manual de las mismas puede resultar tediosa y propensa a errores. Por tanto, se considera la posibilidad de implementar un

programa en lenguaje ensamblador que se encargue de esta tarea.

- El *bit* sobrante podría reintegrarse a la instrucción y permitir comportamientos diferentes de lo tradicional. Se considera la posibilidad de aunarlo con el *bit* de dirección, lo que generaría 2 nuevas posibilidades que podrían ser: no desplazarse, y no desplazarse ni enviar la orden de escritura en la memoria *RAM*.

REFERENCIAS

[1] J. E. Hopcroft, R. Motwani y J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006, ISBN: 0321455363.

[2] R. J. Tocci, *Digital Systems: Principles and Applications (5th Ed.)* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991, ISBN: 0-13-213133-1.

[3] W. Stallings, *Computer Organization and Architecture: Designing for Performance*, 8th. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009, ISBN: 9780136073734.

[4] J. L. Hennessy y D. A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011, ISBN: 012383872X, 9780123838728.

IV. Cronograma y actividades

Semana	**Periodo**	**Actividad**
1	18/03 - 22/03	Entrega inicial del enunciado. Lectura y comprensión del problema. Familiarización con el ambiente de desarrollo de circuitos digitales Logisim.
2	25/03 - 29/03	Recopilación de fundamentos teóricos para el sistema a construir. Discusión grupal sobre especificaciones técnicas preliminares de la máquina. Elaboración del plan de trabajo e informe semanal.
3	01/04 - 05/04	Elaboración de un documento que describe en detalle el diseño y funcionamiento de la implementación de la máquina de Turing.
4	08/04 - 12/04	Crear un módulo utilizando la memoria *ROM* que permita la búsqueda de estados siguientes a partir de un conjunto de símbolos y estados predefinidos. Además, dar inicio al desarrollo de la unidad de control que llevará a cabo la coordinación de microinstrucciones para operar la máquina.
5	15/04 - 19/04	Avanzar el desarrollo de la unidad de control a un estado cercano a la finalización. Añadir las funcionalidades que permiten controlar el puntero a la memoria *RAM*, además de la lectura y escritura de símbolos en la misma.
6	22/04 - 26/04	Finalizar el desarrollo de la unidad de control, con las funcionalidades de conexión con el módulo de memoria *ROM* y *RAM*, que permita la ejecución de un programa cualquiera cargado en la memoria de programa y memoria de datos. Desarrollo de un componente de representación de resultados para poder visualizar qué ocurre con el programa en ejecución.
7	29/04 - 03/05	Preparación de programas de prueba y material de exposición para llevar a cabo una presentación acerca de la implementación de la máquina de Turing.

V. DISTRIBUCIÓN DE TAREAS

Semana	**Periodo**	**Distribución**
2	25/03 - 29/03	Introducción y reporte semanal: Daniel Artavia Cordero. Marco Teórico y bibliografía: Gloriana Mora Villalta. Diseño preliminar: Luis Carlos Quesada Rodríguez.
3	01/04 - 05/04	Diseño detallado de la máquina de Turing: 2 sesiones presenciales para toma de decisiones conjunta y 2 sesiones virtuales para la redacción del documento.
4	08/04 - 12/04	Implementación de búsqueda de instrucciones en la memoria *ROM*: Luis Carlos Quesada Rodríguez. Iniciar implementación de la unidad de control: Daniel Artavia Cordero y Gloriana Mora Villalta.
5	15/04 - 19/04	Estado cercano a finalización de la unidad de control: Gloriana Mora Villalta y Luis Carlos Quesada Rodríguez. Implementación de lectura, escritura y desplazamiento en memoria *RAM*: Daniel Artavia Cordero.
6	22/04 - 26/04	Finalización de la unidad de control: Gloriana Mora Villalta y Luis Carlos Quesada. Visualizador de la ejecución del programa: Daniel Artavia Cordero.
7	29/04 - 03/05	Cada integrante va elaborar 2 programas de pruebas. El material de exposición y repartición de secciones se discutirá en sesiones virtuales.