Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI-0119 Proyecto Integrador de Lenguaje Ensamblador y Arquitectura
Integrantes: Kevin Barrantes Salazar, Iván Chavarría Vega, Javier Herrera Mora.
I ciclo de 2019

Plan de trabajo

Introducción

Este proyecto busca construir una máquina de Turing mediante el programa de circuitos Logisim. Esta máquina de Turing está compuesta por una cinta en la que se escriben y leen símbolos, y estados, que están almacenados en varios componentes físicos.

El funcionamiento de esta máquina de Turing es leer un símbolo de alguna posición de la cinta, escribir un símbolo en la cinta según el estado en el que el computador se encuentre, cambiar de estado y moverse a la izquierda o derecha en la cinta.

La cinta de una máquina de Turing en teoría es infinita, pero este computador contará con una cinta con capacidad de almacenar 1 kb de memoria, por tanto, la cantidad de espacios en los que se puede leer y escribir y la capacidad de la cabeza de la máquina para moverse por la cinta será limitada.

Por otra parte, este computador cuenta con una arquitectura RISC, debido a que se van a implementar soluciones utilizando los registros como recurso de memoria.

Para albergar las instrucciones de lenguaje máquina, se utilizará una memoria ROM, pues las instrucciones que controlan el programa no deben ser modificadas durante la ejecución del mismo.

Por otra parte, para representar la cinta se utilizará una memoria RAM, en la que se puede leer y escribir los símbolos de las cintas. La información de esta memoria RAM será cargada a registros, que serán usados durante la ejecución de los diferentes procesos.

Para realizar las operaciones necesarias de escribir y leer los símbolos de las cintas y otras funciones extras del computador, se usarán compuertas lógicas y diversos componentes que pertenecen al ALU.

Además, el circuito se ejecuta secuencialmente mediante etapas en las que se bloquea el funcionamiento de otras partes del circuito que no incidan con el proceso actual.

A partir de la base anterior, se buscará implementar variaciones a las acciones que la máquina puede implementar, utilizando dispositivos de entrada del computador para ejecutar otras acciones además de las de lectura y escritura de símbolos o cambio de estados.

Marco Teórico

En 1937, el matemático inglés Alan Mathison Turing, publicó un artículo sobre el teorema de Gödel, en el cual se introdujo por primera vez el concepto de Máquina de Turing, la cual era una entidad matemática que permitió en ese entonces formalizar el concepto de algoritmo, y a su vez, resultó ser la precursora de los equipos computacionales actuales. Turing con su máquina demostró que existen problemas que no tiene solución, los cuales su máquina no podrá resolver, y por ende, ningún equipo computacional actual será capaz de resolver ese mismo problema. La máquina de Turing puede considerarse como (Alfonseca, S.f):

> Una cinta infinita divida en casillas, cada una de las cuales contiene un símbolo. Sobre dicha cinta actúa un dispositivo que puede adoptar diversos estados y que, en cada instante, lee un símbolo de la casilla sobre la que está situado. En función del símbolo que ha leído y del estado en que se encuentra, realiza las tres acciones siguientes: pasa a un nuevo estado, imprime un símbolo en el lugar del que acaba de leer, y se desplaza una posición hacia la izquierda, o hacia la derecha, o bien la máquina se para.

El concepto de máquina de Turing es tan general y potente, que es posible construir una máquina que sea capaz de simular el comportamiento de otra máquina de Turing cualquiera. Esto es lo que se llama *Universal Turing Machine* (Katz, 2011). Gracias a su existencia, podemos disponer de ordenadores electrónicos, que no son más que máquinas generalizadas capaces de realizar cualquier cálculo computable.

Una de las primeras decisiones a tomar cuando se realiza una Máquina de Turing, es el juego de instrucciones con el que la máquina va a actuar. Para esto se nos presentan dos filosofías de diseño: arquitecturas de computadoras en versión RISC (*Reduced Instruction Set Computer*) y CISC (*Complex Instruction Set Computer*). De manera superficial, la forma en la que se diferencian estas dos arquitecturas es la siguiente: RISC presenta una arquitectura en la cual las instrucciones son de tamaño fijo y presentadas en un reducido número de formatos (de aquí el nombre), mientras que las instrucciones en la arquitectura CISC, se caracterizan por ser muy amplias y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos.

RISC es un tipo de arquitectura del microprocesador que utiliza un pequeño, pero sumamente optimizado, conjunto de instrucciones, antes que un conjunto de instrucciones más especializado, estos últimos encontrados comúnmente en otros tipos de arquitecturas (Masood, S.f). Tiene como objetivo dar un soporte más eficiente a los casos que se repiten con más frecuencia, por esto las operaciones que realiza se podrían considerar pocas y básicas, lo cual, debido al tipo de operación, le permite disminuir el tiempo del ciclo, haciéndola bastante eficiente. Debido a esto, se le caracteriza por ser limitado y sencillo. (Dpto. de Arquitectura de Computadores y Automática, S.f)

A su vez, esta arquitectura usa un gran número de registros y/o el uso del compilador para optimizar el uso de los registros. La finalidad es optimizar las referencias a operandos. De esta forma, si el operando ocupa un registro se reducen las transferencias a memoria, y por tanto el tiempo de acceso y la eficiencia del programa. Debido a la alta proporción de instrucciones de bifurcaciones condicionales y de llamada a procedimientos, hay que prestar especial atención al diseño de los cauces de instrucciones, para intentar evitar, en la medida de lo posible, penalizaciones por vaciado del cauce, esto optimizará la segmentación de instrucciones.

El motivo por el cual esta arquitectura usa gran cantidad de registros para su almacenamiento, es porque este es el más rápido que existe, tanto así, que es aún más rápido que la memoria caché y la memoria principal del ordenador (RAM). El motivo de su rapidez se debe a que este tipo de almacenamiento

emplea direcciones mucho más cortas que los otros dos tipos dos tipos mencionados anteriormente, ya que se encuentra integrado en el mismo chip que la Unidad de Control y la ALU (*Arithmetic Logic Unit*), lo cual, lógicamente, minimizará las distancias entre direcciones y de esta manera, su duración. Por esto mismo, es que el almacenamiento en registro suele ser bastante más reducido que el de la memoria caché o el de memoria principal. (Escuela Técnica Superior de Ingenieria, 2010).

Por otra parte, se encuentra la arquitectura CISC, la cual se caracteriza por el tamaño de código, el cual es menor respecto a otros tipos de arquitecturas (RISC, por ejemplo), ya que se dan menos instrucciones por programa, pero, cada una de estas instrucciones, tienen múltiples operaciones por debajo. Gracias a esto, se puede observar/establecer las razones por las cuales el tamaño resultante es diferente entre las dos arquitecturas. Una es el hecho de que las instrucciones complejas de los CISC raramente encuentran ocasión para ser usadas, y otra es que las instrucciones RISC son más cortas que las CISC, y por ello necesitan menos bits y así generar un tamaño idéntico que las CISC utilizando más instrucciones. Por esto mismo, como ya se puede intuir, la ejecución de los programas CISC es más lenta. Una razón es la misma que se esgrimió anteriormente, la tendencia de los compiladores CISC a utilizar instrucciones sencillas. Por otro lado, la unidad de control CISC utiliza microprogramación para implementar instrucciones complejas, lo cual complica la unidad de control. Frente a esto, la sencillez de las unidades de control RISC permiten utilizar mayores frecuencias de reloj y, por tanto, incrementar la velocidad de ejecución (Escuela Técnica Superior de Ingenieria, 2010).

Los conjuntos de instrucciones mencionados anteriormente, tiene que pasar por un ciclo conocido como Ciclo de Instrucción, el cual, según (Torres, 2010):

> Es el período que tarda la unidad central de proceso (CPU) en ejecutar una instrucción de lenguaje máquina. Comprende una secuencia de acciones determinada que debe llevar a cabo la CPU para ejecutar cada instrucción en un programa. Cada instrucción del juego de instrucciones de una CPU puede requerir diferente número de ciclos de instrucción para su ejecución. Un ciclo de instrucción está formado por uno o más ciclos máquina.

Este ciclo se ejecuta una y otra vez desde el momento en que encendemos el ordenador, a su vez, este ciclo nos permite comprender el funcionamiento de la ejecución de las instrucciones del procesador. Internamente, este ciclo es el procesamiento requerido para una instrucción, el cual se encuentra dividido en dos ciclos principales: *fetch* y *execute*, respectivamente.

El *fetch cycle* (Ciclo de Captura) se divide en dos fases: búsqueda y decodificación. La fase de búsqueda se encarga de buscar en la memoria principal la instrucción que se quiere ejecutar, luego de haberla encontrado, la almacena en el registro del CPU para instrucciones. Seguidamente, en la fase de decodificación, el procesador se encarga de averiguar qué es lo que quiere hacer esta instrucción (se identifica el modo de direccionamiento de la instrucción y la ubicación de los datos a tratar).

Una vez terminado el ciclo anterior, la instrucción continua al *execute cycle* (Ciclo de Ejecución), el cual, también se divide en dos fases: ejecución y finalización. Después de que la fase de decodificación haya averiguado qué es lo que se quiere hacer con esta instrucción y se la haya enviado al ciclo siguiente, la fase de ejecución realiza las operaciones que se quieren hacer en dicha instrucción, y finalmente, la fase de finalización se encarga de terminar la instrucción, posiblemente guardando en memoria un resultado o enviando a un dispositivo de salida, dependiendo de la instrucción. Una vez terminado el ciclo, este se vuelve a repetir una y otra vez en un bucle que no se detiene (Snyder, 2015).

Concluyendo, la arquitectura de computadoras en versión RISC tiene una menor interacción con el uso de la memoria principal y la no volátil que la arquitectura CISC. Esta estructura delega muchas de las operaciones o instrucciones a la CPU y sus registros. La estructura CISC en cambio utiliza con mucha mayor frecuencia las memorias de todo el sistema, por ejemplo, una operación sencilla de suma puede

ser realizada entre elementos presente en la memoria volátil y los registros de la CPU.

En el caso de la máquina de Turing esta propiedad no estará tan estrictamente delimitada sin embargo se tendrá una clara inclinación por la arquitectura RISC al implementar registros para muchos de los procesos de decisión y almacenamiento. Otro punto importante a considerar es que las instrucciones RISC son de tamaño fijo y las instrucciones CISC son de tamaño variable (Mano, 1993). Esta propiedad si representa un hecho bastante contundente en términos de la elección de estructura ya que las instrucciones que se planean manejar son hileras de binarios con una estructura fija que detallan toda la información que se necesita para la implementación de cada etapa del proceso de la máquina de estados.

Diseño Preliminar

Lista de componentes que se ocuparan en primera instancia:

1. Registros
2. Memoria RAM
3. Memoria ROM
4. Compuertas lógicas (and, or, not, etc)
5. Buffers controlados
6. Sub-circuitos (cajas negras)
7. Comparadores
8. Operadores aritméticos
9. Contadores
10. Reloj
11. Decodificador

Las siguientes figuras corresponden a las diferentes formas de representaciones en el diagrama de cajas del circuito.

Figura 1: Simbología del diagrama de bloques

Memorias de mayor capacidad: Representan la memoria RAM y ROM en el circuito.

Registros: Son los encargados de guardar la información en una posición en memoria.

Cajas Negras: Representan un sector del circuito del que aún no se ha decidido los componentes que lo forman, pero se tiene la idea general de la función de ese sector.

Funciones de Logisim: Simbolizan todos los componentes que se encuentran en el programa Logisim, es decir, compuertas AND, OR, XOR, buffers, etc.

En general el programa será controlado por un sistema compuesto de un solo reloj, varios contadores y decodificadores. Esto se debe a que el programa no puede trabajar al mismo tiempo todas las etapas, debe completar algunas antes de seguir con la siguiente. Los componentes anteriores permiten que se logren desprender partes del circuito mientras se ejecutan otras, por ejemplo desprender el sistema de escritura en la memoria RAM mientras se hace la búsqueda de la siguiente instrucción.

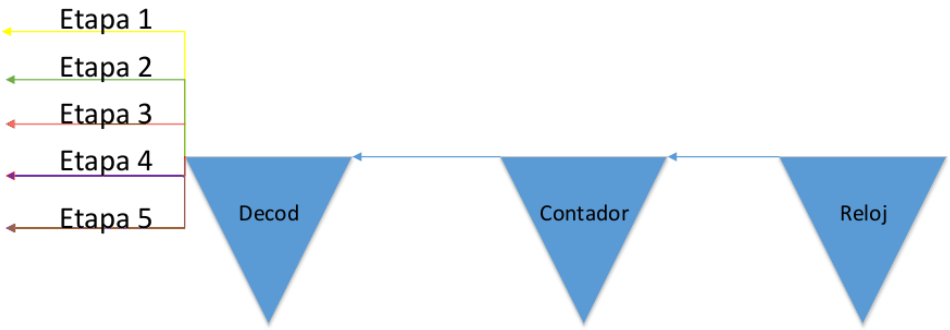

Figura 2: Controlador de etapas

El diseño constará de diferentes etapas y no necesariamente las establecidas por la teoría de la máquina de Turing. Para mayor facilidad en la representación y lectura del esquema del circuito se separa esta sección del circuito cuya función básica es habilitar o deshabilitar el paso de cada una de las etapas del circuito general(ver figura 3).

Reloj: envía pulsos a través del circuito, lo que provoca que las entradas de los diversos componentes se actualicen.

Contador: este componente del circuito aumenta en uno con cada pulso del reloj. El número guardado en el contador permite acceder a una dirección de la memoria ROM o a una de las salidas del decodificador (ver Figura 3).

Decodificador: Es el componente encargado de hacer la transición de etapas. Por medio de buffers de control habilita o deshabilita el paso del reloj a secciones del circuito.

Colores: En las figuras 2 y 3 se utiliza cableado con distintos colores para identificar cuáles componentes están implicados en cada una de las etapas.

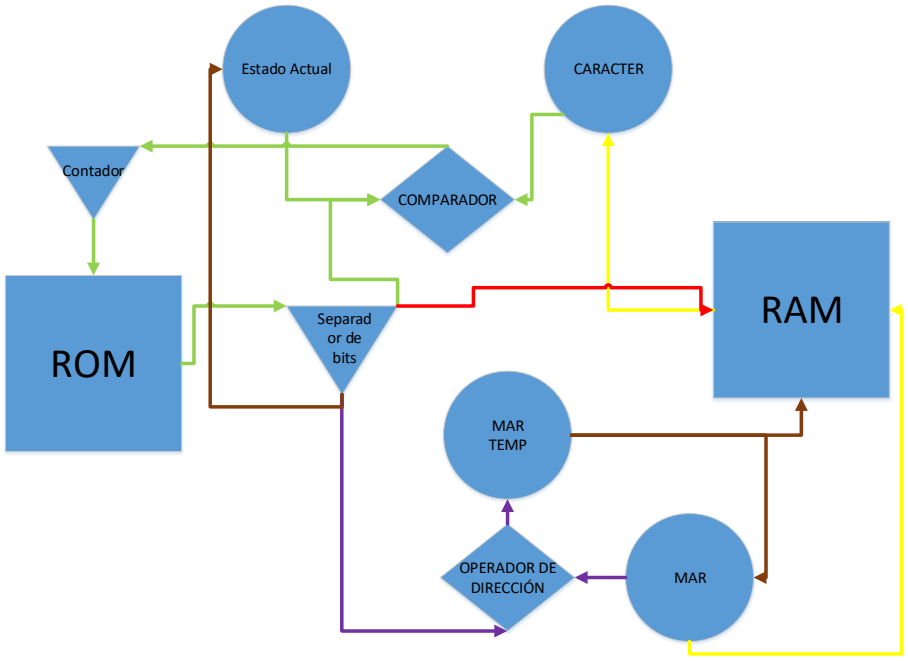

Figura 3: Esquema general

Primera etapa: ya que la dirección de memoria ingresada al inicio como parámetro representa la dirección del carácter inicial se carga se carga en primera instancia a sus respectivos registros desde la memoria RAM para poder buscar posteriormente la instrucción que se debe ejecutar.

Segunda etapa: consiste en comparar el estado inicial y carácter actual de la posición de memoria con cada uno de las instrucciones que se encuentra en la memoria ROM. Este proceso se repite hasta encontrar una instrucción que coincida.

La memoria ROM tiene una entrada que es la posición de memoria a leer. Para recorrer todas las posiciones de memoria de la ROM es necesario un contador que actualice en cada pulso del reloj la posición a acceder. La memoria ROM envía los datos del nuevo estado y nuevo carácter al separador de bits.

De esta manera el sistema será capaz por si solo de identificar cuál es la instrucción que debe ejecutar basado en las condiciones que tiene actualmente el sistema. En este proceso se usaran la memoria ROM, un registro en el que guardará el estado y carácter actual, el comparador, el separador de bits que proporciona el carácter y el estado leídos desde la ROM y una compuerta AND que compara la información proporcionada por ROM y el registro.

Tercera etapa: Se debe sobre-escribir sobre el espacio de memoria en el que esta posicionado el cursor de la memoria RAM el carácter siguiente procedente de la memoria ROM. Para esto se utiliza un separador de bits que toma solamente la información que contiene el nuevo estado y el carácter a escribir.

Cuarta etapa: Se mueve el puntero de la RAM según lo que indique la instrucción de la memoria ROM. Por medio del operador de dirección y la dirección guardada en el registro MAR se suma o resta una posición. Esta nueva posición se guarda en el registro MAR temporal, ya que si se guardara en el registro MAR directamente esto crearía un error al alimentar la entrada con la salida en un mismo ciclo de reloj.

Quinta etapa: Finalmente el MAR TEMP envía la posición del puntero de dirección a la memoria RAM, y al mismo tiempo, guarda esa dirección en el registro MAR. Además, el separador de bits envía los datos al registro de Estado para actualizarlo causando que el ciclo se reinicie ya que el estado actual no coincide con el de la instrucción.

Cronograma de actividades

Simbología:

- KBS: Kevin Barrantes Salazar
- ICV: Iván Chavarría Vega
- JHM: Javier Herrera Mora
- **21-28 de marzo**: Se realiza el plan de trabajo para el proyecto. Este documento contiene una descripción general de la máquina de Turing a realizar, un diseño preliminar básico, el cronograma de actividades, y la organización de tareas.

  La entrega de este documento se realiza el 28 de marzo.

  Por otra parte, se realiza el primer informe de avance, que explica cómo se construyó el plan de trabajo, cuáles tareas hizo cada integrante del equipo y las decisiones más importantes que se tomaron con respecto al diseño de la computadora.

  - Plan de trabajo: KBS, ICV, JHM.
  - Informe de avance semanal: KBS, ICV, JHM.
  - Reunión de revisión: KBS, ICV, JHM.
- **28 de marzo - 11 de abril**: Se realiza el diseño definitivo de la computadora, explicando en detalle las partes por las que el diseño de la máquina de Turing estará compuesta.

  Se explican los cambios hechos al diseño definitivo con respecto al diseño preliminar, cuáles fueron las motivaciones y limitaciones que obligaron a hacer variantes al proyecto.

  La entrega de este documento se realiza el 11 de abril.

  - Diseño primera etapa: KBS.
  - Diseño segunda etapa: ICV.
  - Diseño tercera etapa: JHM.
  - Diseño cuarta etapa: JHM.
  - Corrección de errores: KBS, ICV, JHM.
  - Documento de diseño final: KBS, ICV, JHM.
  - Reunión de revisión: KBS, ICV, JHM.

  Además, se realiza el segundo y tercer informe de avance semanal en el que se reportará las tareas realizadas por cada integrante de equipo en el diseño definitivo del proyecto.

  La entrega del segundo y tercer informe semanal se realiza el 4 y 11 de abril respectivamente.

- **11 de abril - 2 de mayo**: Durante tres semanas, se realizará la implementación del computador en el programa Logisim, la documentación final del computador y se ejecutarán ejemplos de prueba para validar el correcto funcionamiento de la máquina de Turing.

  Durante semana y media se espera realizar la implementación del computador. Posterior a esa semana y media, se ejecutarán los programas de prueba en el computador con el fin de encontrar fallos en el diseño, finalizando así la implementación en la segunda semana.

  En la tercera semana se realizará la documentación del funcionamiento de la máquina de Turing y se espera alcanzar un pequeño avance en la preparación de la exposición del proyecto.

  La entrega de la implementación, la documentación y los programas de prueba se realiza el 2 de mayo.

  Además, el día 18 de abril se realiza la entrega del tercer informe semanal, en el que se explicarán los avances en la implementación del computador. El 25 de abril se entrega el cuarto informe semanal, en el que se explicarán los detalles finales de la implementación del computador.

  Finalmente, el 2 de mayo se realiza la entrega del quinto informe semanal, en el que se detallará el proceso de documentación del computador, así como la solución a posibles fallos corregidos durante esa semana. Además, se explicarán los primeros detalles de la preparación para la exposición del proyecto.

  - Implementación primera etapa en logisim: KBS.
  - Implementación segunda etapa en logisim: ICV.
  - Implementación tercera etapa en logisim: JHM.
  - Implementación cuarta etapa en logisim: ICV.
  - Programas de prueba: KBS.
  - Corrección de errores: KBS, ICV, JHM.
  - Archivo de programa final en logisim: KBS, ICV, JHM.
  - Preparación de material para la defensa final : KBS, ICV, JHM.
  - Reunión de revisión: KBS, ICV, JHM.

- **2-16 de mayo**: Se continúa con la preparación de la defensa del proyecto. Durante estas semanas se preparan los recursos necesarios para explicar el computador implementado.

  La defensa del proyecto se realiza el 16 de mayo.

  Además, el 9 de mayo se realiza la entrega del sexto informe semanal y el 16 de mayo se realiza la entrega del séptimo informe semanal.

  Ambos informes contendrán detalles sobre la preparación de la exposición sobre la máquina de Turing implementada.

  - Terminar revisión de material para la defensa final: KBS, ICV, JHM.
  - Exposición y defensa final: KBS, ICV, JHM.

Nota: Las reuniones semanales se harán todos los martes después de la clase de lenguaje ensamblador.

Referencias

Alfonseca, M. (S.f). *La máquina de turing.* Descargado de http://www.sinewton.org/numeros/numeros/43-44/Articulo33.pdf

Dpto. de Arquitectura de Computadores y Automática. (S.f). *Arquitectura del procesador.* Universidad Complutense de Madrid, Facultad de Informática. Descargado de http://www.fdi.ucm.es/profesor/jjruz/ec-is/temas/Tema%202-Arquitectura%20del%20procesador.pdf

Escuela Técnica Superior de Ingenieria. (2010). *Sistemas electrónicos para el tratamiento de la información.* Universidad de Valencia. Descargado de http://ocw.uv.es/ingenieria-y-arquitectura/sistemas-electronicos-para-el-tratamiento-de-la-informacion/seti_materiales/seti6_ocw.pdf

Katz, J. (2011). *Notes on complexity theory.* UMD Department of Computer Science. Descargado de http://www.cs.umd.edu/~jkatz/complexity/f11/all.pdf

Mano, M. M. (1993). *Arquitectura de computadoras.* Descargado de https://books.google.co.cr/books?hl=en&lr=&id=2wWZyKu60cAC&oi=fnd&pg=PR17&dq=arquitectura+risc+y+cisc+morris&ots=DSIIl38uwr&sig=ZanC8_lcVMdJLNZ4SUc5xMZJk48&redir_esc=y#v=onepage&q=arquitectura%20risc%20y%20cisc%20morris&f=false

Masood, F. (S.f). *Risc and cisc, computer architecture.* National University of Sciences and Technology (NUST). Descargado de https://arxiv.org/ftp/arxiv/papers/1101/1101.5364.pdf

Snyder, L. (2015). *Fluency with information technology, 6th edition.* University of Washington.

Torres, U. I. R. (2010). *Infraestructura tecnológica: Ciclo fetch.* Descargado de https://www.scribd.com/doc/30420235/Ciclo-Fetch#fullscreen&from_embed