Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI-0118 Proyecto Integrador de Arquitectura y Ensamblador, Grupo 01
Fecha: 02/05/2019, I ciclo lectivo 2019

Proyecto # 1

Integrantes

B72097 Iván Chavarría Vega

B53474 Javier Herrera Mora

B70998 Kevin Barrantes Salazar

Máquina de Turing

1. Uso de la máquina

El circuito final se ve de la siguiente manera desde Logisim:

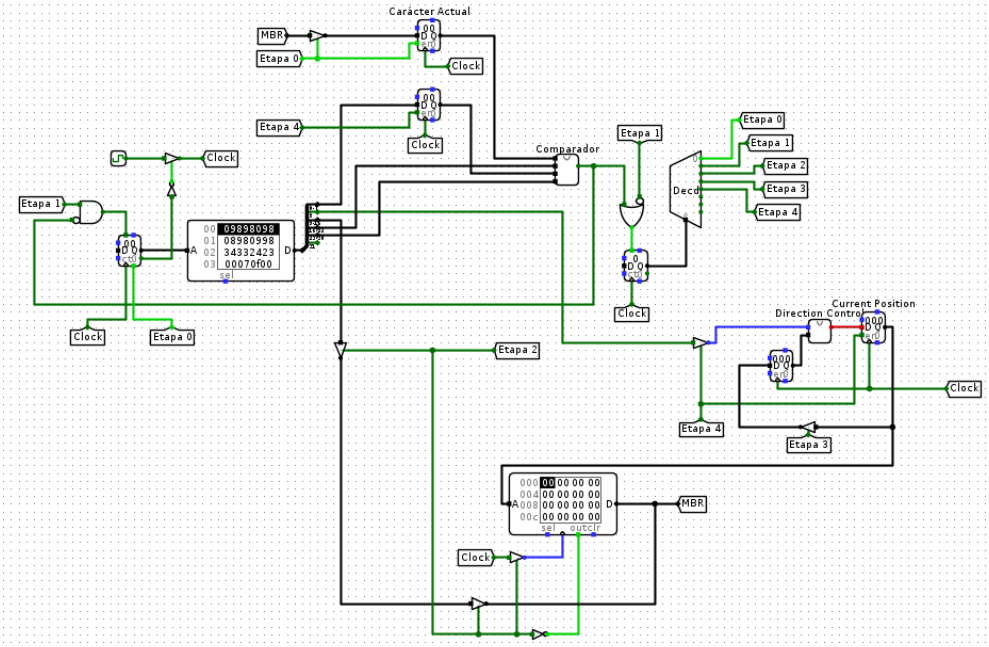

Figura 1: Circuito final desde Logisim

Para usar la Máquina de Turing, es necesario cargar los datos del programa en distintas partes del circuito.

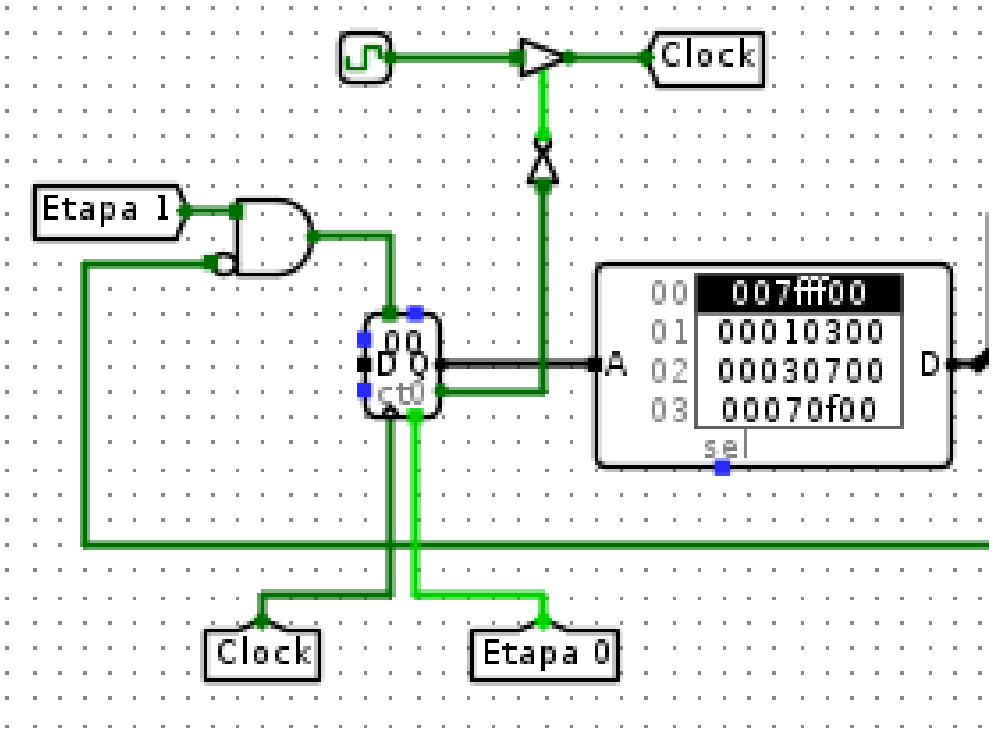

Figura 2: Memoria ROM de la Máquina de Turing

Las instrucciones del programa son cargadas en la memoria ROM, la cuál se muestra en la figura 2. Para modificar los datos de la memoria ROM, es posible modificar un dato haciendo click izquierdo sobre la celda. Una vez seleccionada la celda, el usuario puede agregar las instrucciones del programa (todas ellas son números en hexadecimal). Otra opción para cargar datos a la memoria ROM es pulsar click derecho sobre la memoria ROM y seleccionar la opción *cargar imagen*. Posteriormente se selecciona el archivo que contiene las instrucciones.

```
v2.0 raw
7FFF00 10300 30700 70F00 55AA01 81C110 83C120 87C140
1010202 1030602 1070E02 155AA02 1600201 2010204 2030604 2070E04
255AA04 2600601 4010208 4030608 4070E08 455AA08 4600E01 8010310
8030710 8070F10 855AB10 87F0202 10010320 10030720 10070F20 1055AB20
107F0604 20010340 20030740 20070F40 2055AB40 207F0E08
```

Figura 3: Formato de datos en Logisim

En la figura 3 se puede observar el formato que debe contener los datos del archivo para que la memoria ROM pueda leer esas instrucciones. La primera línea del archivo es obligatoria y las demás lineas que sean escritas, deben contener como máximo ocho números, además debe existir un espacio entre cada valor.

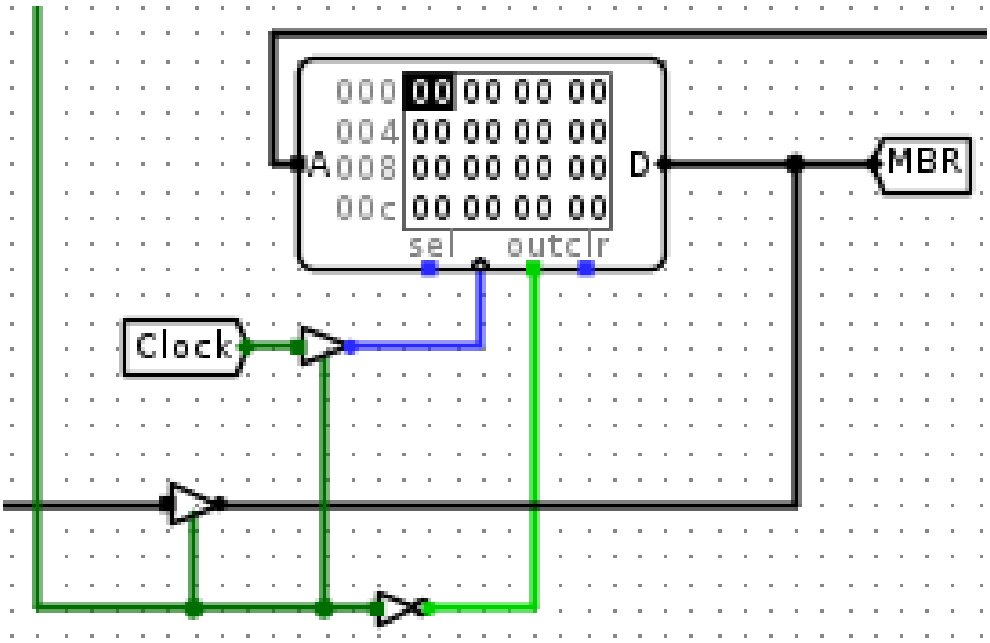

Figura 4: Memoria RAM de la Máquina de Turing

Por otra parte, la información escrita en la cinta es guardada en la memoria RAM. Para cargar los datos en el componente mostrado en Figura 4, es posible seguir alguno de los dos procedimientos descritos para la carga de datos en la memoria ROM. Los resultados de la ejecución del programa pueden ser consultados en la memoria RAM.

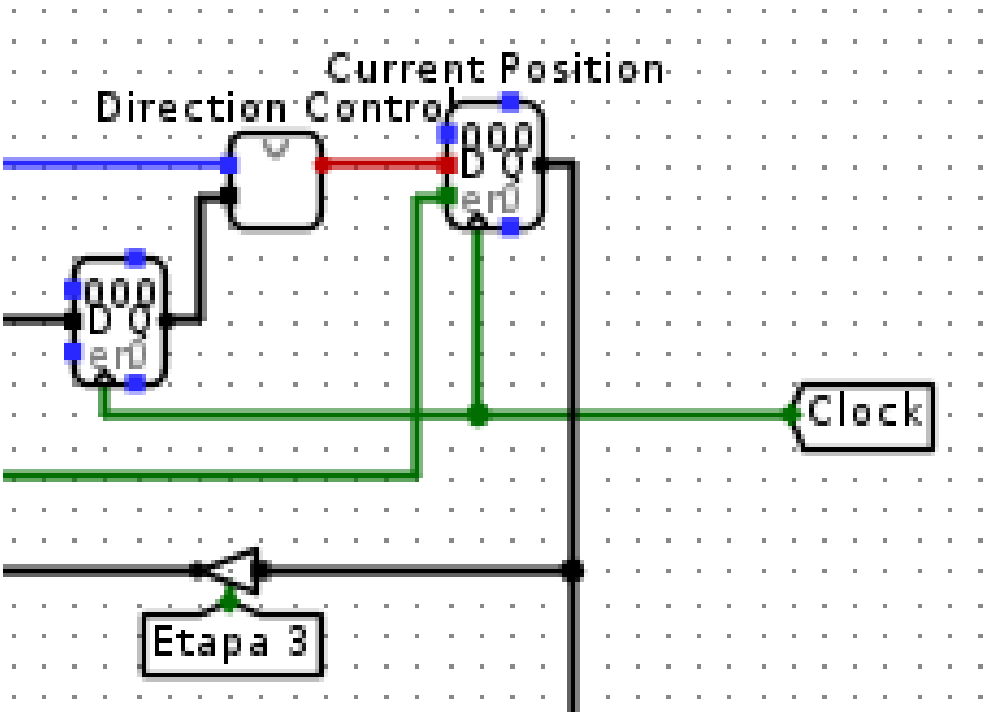

Figura 5: Registro Current Position

Para configurar la posición inicial de la cabeza en la cinta, es necesario modificar la celda de la memoria RAM que se debe leer. Para esto, se debe modificar la información del registro *Current Position*.

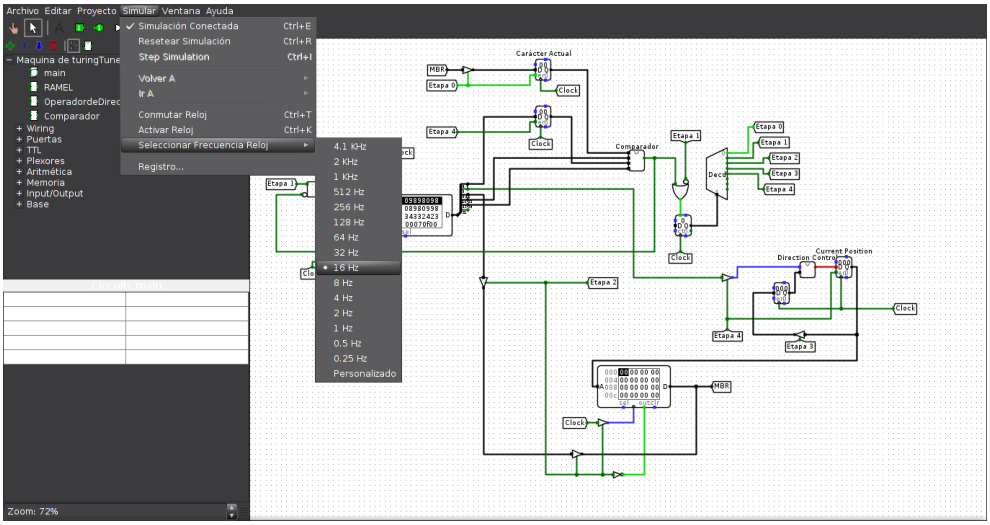

Figura 6: Configuración para ejecución automática del programa

Finalmente, para ejecutar el programa, es recomendable configurar Logisim para que ejecute cada pulso del programa se ejecute automáticamente. Como se muestra en la figura 6, se pulsa click izquierdo sobre la opción *simular*, y accedemos a las distintas frecuencias de reloj que se despliegan al colocar el cursor sobre la opción *Seleccionar Frecuencia Reloj*. Una vez que se selecciona la frecuencia, se escoge la opción *Activar Reloj*. Para detener la ejecución automática, el usuario puede presionar los botones *Ctrl* y *K*.

2. Casos de prueba

Para estar totalmente seguros de que la máquina estaba funcionando correctamente, esta se sometió a diferentes casos de prueba, los cuales sucedieron con normalidad y se obtuvo el resultado esperado. Para cada uno de estos casos, la máquina recibe un *set* de instrucciones los cuales se introducen en la memoria ROM, así como la hilera a la que se le quiere determinar cierta condición, esta es introducida en la memoria RAM. Para delimitar el final de la hilera, se usó *7f*, esto es para que la máquina sepa en qué momento detenerse en la RAM y hacer la comparación. Tres de estas pruebas fueron los siguientes:

- **Es palíndromo:** recorre la hilera de izquierda a derecha y de derecha a izquierda para comprobar si los datos introducidos en la memoria RAM es palíndromo.

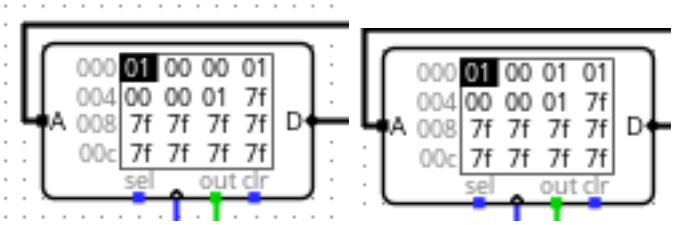

Figura 7: Datos introducidos en dos ejecuciones de Es Palíndromo

En Figura 7 se observa el estado inicial de la RAM (en dos ejecuciones diferentes). Para este caso, primero se probó con un dato que sea palíndromo (la imagen de la izquierda), y luego con un dato que no sea palíndromo (imagen de la derecha).

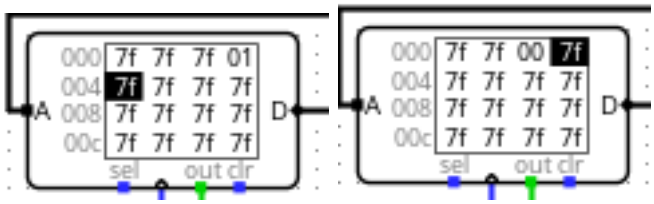

Figura 8: Resultado de ejecutar Es Palíndromo

Figura 8 muestra el resultado de ambas ejecuciones. Donde *01* representa que la hilera introducida sea palíndromo, y *00* que no lo sea. Sabiendo esto, si se observa la primera ejecución (izquierda), el resultado fue *01* indicando que esta hilera es palíndromo, lo cual si observamos Figura 7 es correcto. La segunda ejecución (derecha) fue con un dato no palíndromo, ver Figura 7, y el resultado fue *00* indicando correctamente que esta hilera no es palíndromo.

- **Duplicar línea:** lee una hilera de datos hasta encontrarse con el símbolo "#", que es el carácter de parada. Una vez encontrado el símbolo, se escribe a la derecha del mismo todas las letras antes del carácter de parada, pero en orden inverso (primero la letra más cercana a "#", luego la segunda letra más cercana) hasta llegar al carácter "␣"que se encuentra antes del inicio de la hilera. Los valores usados en este programa son "␣", valor 7F; "#", valor 55; .ª", valor 01; "b", valor 03, ç", valor 07; y ".", valor 60.

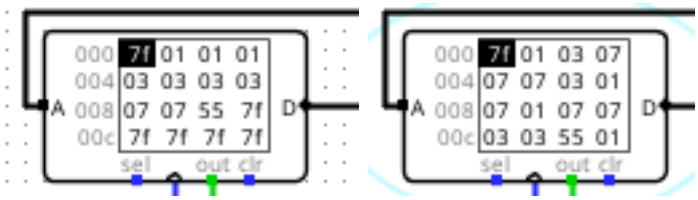

Figura 9: Datos introducidos en dos corridas de Duplicar Línea

En la figura 9, en la memoria RAM izquierda se observa el estado inicial de la cinta que representa la hilera "␣ aaabbbbcc #". En otra ejecución del programa se adjuntaron los datos de la hilera "␣ abcccbacaccbb#aa ␣ aa ␣aaaba", que se pueden observar en la memoria RAM de la derecha.

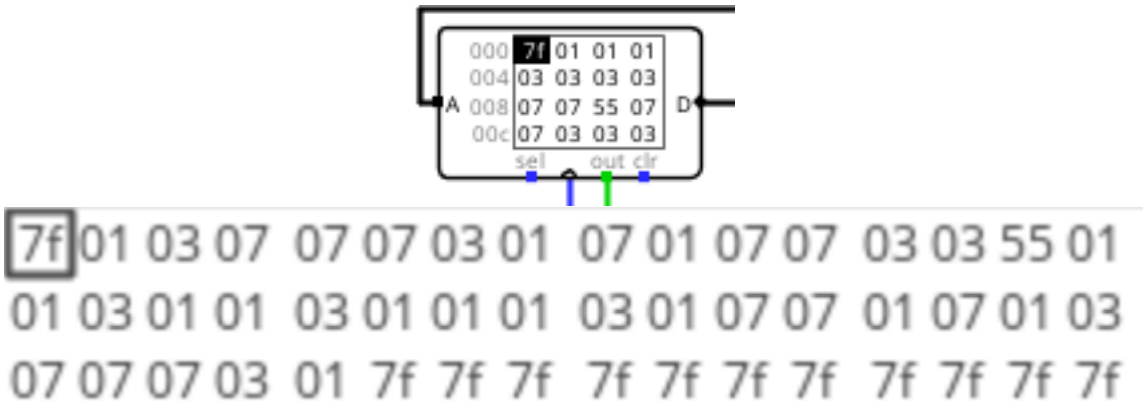

Figura 10: Resultado de las ejecuciones de Duplicar Línea

En la figura 10 se observa el resultado de la ejecución del programa. En la memoria RAM de la superior, cambiando los valores de los datos por caracteres, obtenemos la hilera "␣ aaabbbbcc #

ccbbbbaaa", por lo que la ejecución es exitosa, mientras que en la memoria RAM de abajo, podemos observar que la segunda ejecución del programa también fue exitosa, pues se retorna la hilera "_ abcccbacaccbb # aabaabaaabaccacabcccba", tomando en cuenta que cada "_"fue sustituido por una letra.

- **Es divisible entre tres:** Toma un número en base binaria y verifica si es divisible por tres. Dicho numero se encuentra encerrado por los caracteres delimitadores interpretados como blancos. La cabeza de la máquina de Turing debe estar posicionado en el dígito de menor magnitud y de ahí partir hacia los dígitos de mayor magnitud para luego volver nuevamente hasta la posición del dígito menos significativo e imprimir si el número en cuestión es múltiplo de 3 o no. Si el resultado es no se imprime *03* y si el resultado es sí entonces se imprime un *0f*.

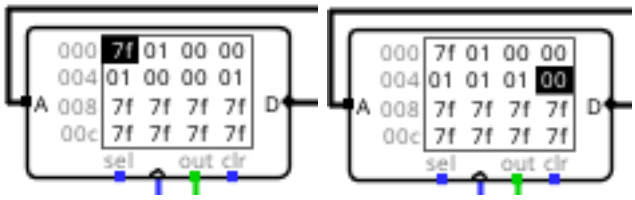

Figura 11: Datos introducidos en dos corridas de Divisible por Tres

Como se puede ver en la Figura 11 se probaron el número 73 y 78. El segundo es divisible entre 3 mas el segundo no.

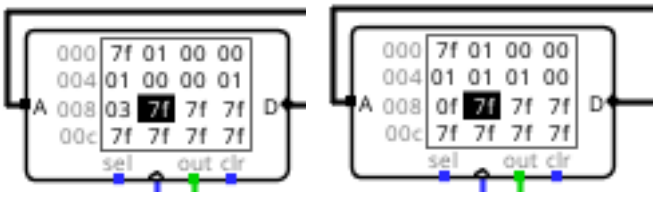

Figura 12: Resultados de las ejecuciones de Divisible por Tres

Como se esperaba luego de la ejecución de la máquina de Turing los dígitos en el extremo derecho no iguales al blanco (7f) de las RAMs en la figura 12 corresponden a los dígitos representativos de no y sí respectivamente.