MANUAL TECNICO OPERATIVO

MICROPROCESADOR Z80

Brian Julian Moreno Niampira

Lenguajes de Programación

UNIVERSIDAD NACIONAL

2018-2

Contenido

1. INTRODUCCION

Los microprocesadores son un componente muy importante, que al día de hoy muchos dispositivos cuentan con uno de ellos, para que realice el procesamiento del sistema y se puedan generar muchos resultados como sistemas partiendo de uno de ellos, y que ha surgido con la evolución de tecnologías, iniciando desde los años 1950, pero que a inicios de 1970 de creo el primer microprocesador a partir de la segunda guerra mundial. En este proyecto realizaremos la emulación de microprocesador Z80 el cual fue lanzado al mercado en julio de 1976, por la compañía Zilog, que fue uno de los mas importante para la década de los 80 debido a que fue ensamblado en maquinas como la Sinclair ZX Spectrum, Amstrad CPC o los ordenadores de sistema MSX. ES uno de los procesadores mas exitosos y que al día de hoy sigue siendo utilizado en sistemas embebidos. La finalidad de este proyecto es realizar la emulación de este microprocesador iniciando un un ensamblado de código enlenguaje ensamblador a un archivo objeto de código maquina, pasando luego por un Carcagor-Enlazador para finalmente llegar al microprocesador.

2. OBJETIVOS GENERALES DEL SISTEMA

EL objetivo general es realizar un emulador para el microprocesador Z80, el cual pueda ensamblar código e lenguaje ensamblador a código maquina, realizar el proceso de cargado y enlazado de estos códigos maquina y finalmente realice el procesamiento de esta información guardada en memoria, teniendo en cuanta que

este puede realizar el procesamiento de 256 instrucciones, en las cuales se pueden realizar operaciones aritméticas, lógicas, incrementos y decrementos de un bit, lectura y escritura de datos, desplazamiento de bits, transporte de señales el el bus de datos, bus de direcciones y bus de control. Entre otras, y buscamos emular la gran mayoría de instrucciones.

3. OBJETIVOS ESPECIFICOS

- Programar una sección del ensamblador que pueda pasar los códigos en lenguaje
- ensamblador a código maquina en tiempo real.
- Realizar una sección para cargar y enlazar estos códigos a a memoria, para que
- puedan ser procesados, esto también que se pueda ver en tiempo real.
- Programa el apartado para el funcionamiento del microprocesador, que podamos ver
- como se realiza la esta parte.
- Realizar una interfaz bien interactiva para cada uno de los apartados anteriores.
- Enlazar las secciones para llevar a cabo el completo funcionamiento.

4. CONTENIDO TEORICO

MICROPROCESADOR Z80

La arquitectura del procesador Z80, que desde 1980 cogió fama y que hasta el día de hoy se utilizan en sistemas embebidos, cuenta con un set de 256 instrucciones las cuales las podemos ver en enlace adjunto en la bibliografía[3], y una extensión de 80 instrucciones con respecto a la clásica CPU 8080, realiza un transporte de señales mediante 3 buses como lo son el bus de direcciones, el bus de datos y el de control. Cuenta con un alto grado de programabilidad, Régimen de interrupción uniforme, con la posibilidad de encadenar las prioridades de los circuitos periféricos, reloj único y se alimenta de 5 Voltios para poder se ejecutado. Este microprocesador funciona leyendo iterativamente cada una de las posiciones de la memoria ROM y las va ejecutando, ayudándose de los datos ubicados en la CPU, los periféricos o las memorias, la transferencia interna de datos es a través del CPU.

Dentro de su arquitectura cuenta con 18 registros de 8 bits, y 4 de 16 bits, donde estos están en memoria RAM estática. Los registros incluyen 2 bancos de 6 registros de propósito general cada uno de 8 bits, o en pares como registros de 16 bits. Dos bancos de registros, el acumulador y las banderas, cuenta con 6 registros de propósito especial, 4 registros de 16 bits PC, SP, IX e IY, y 2 registros de 8 bits, el registro del refresh "R" y el registro de interrupciones "I", la aritmética de 8 bits de las funciones, y las instrucciones lógicas de la CPU se ejecutan en la ALU. Puede realizar operaciones aritméticas como sumar y restar, operaciones lógicas como and, or y xor,

comparaciones, desplazamientos a derecha e izquierda, incremento y decremento de bytes, poner bits en 0 y 1 lógico, y comprobar el estado de los bits.

UNIDADES FUNCIONALES

Dentro de estas se encuentran

1. Unidad Aritmética y Lógica (ALU)

   Esta unidad de capaz de realizar sumas binarias, operaciones lógicas, complemento a dos, corrimiento de un bit a la derecha o a la izquierda, registro de resultados importantes como el acarreo, signo, acarreo auxiliar, paridad o si el resultado es zero,comparaciones, poner, limpiar o probar un bit.

2. El contador del programa.

   Contiene la dirección a memoria que va a ser accedida, este registro es de 16 bits, obtiene el código de la instrucción a ejecutarse, en la CPU y se incrementa secuencialmente en 1, de tal manera podrá recorrer toda la ROM.

3. El apuntador del stack

   Este registro de 16 bits contiene la direccio de memoria RAM en forma descendente, se almacenan los contenidos de un par de registros.

4. Registros de propósito general

   Son registros de 8 bits separados en dos grupos, en el grupo 1 están A, B, C, D, E, H, y L y en el grupo 2 A', B', C', D', E', H' y L. Con estos registros podemos recibir datos desde la memoria, enviar datos hacia la memoria, incrementar o decrementar en uno su contenido, formar una dirección con el contenido de un par de registros, transferir datos entre los registros y obtener un operando durante las funciones de la ALU.

5. Registros de índice

   Son registros de 16 bits con nombres IX e IY, cada uno de 16 bits y conservan direcciones base que se usan para modo de direccionamiento indexado, en este modo un registro de índice se usa como base para apuntar a una región de la memoria[1].

6. Registros de interrupciones

   Con nombre I, es un registro de 8 bits, el cual va almacenando los 8 bits más significativos de la dirección indirecta mientras que el dispositivo que interrumpe

proporciona los 8 bits menos significativos de la dirección índice, y con esto podemos hacer que se localicen en cualquier parte de la memoria y que se puedan tener acceso en un tiempo muy corto.

7. Registro de banderas

   El microprocesador contiene un registro de 8 bits que monitorea las operaciones de la ALU, la información que almacenan estos flips-flops se conoce como banderas de estado, las banderas se actualizan después de cada operación con alguno de los registros.

8. Registro para refrescar memorias dinámicas

Esta arquitectura se puede visualizar en la Figura 1. Donde podemos ver como se encuentran ubicados cada uno de los registros, buses donde se realiza el transporte de señales, la ALU y entre otros ya mencionados.

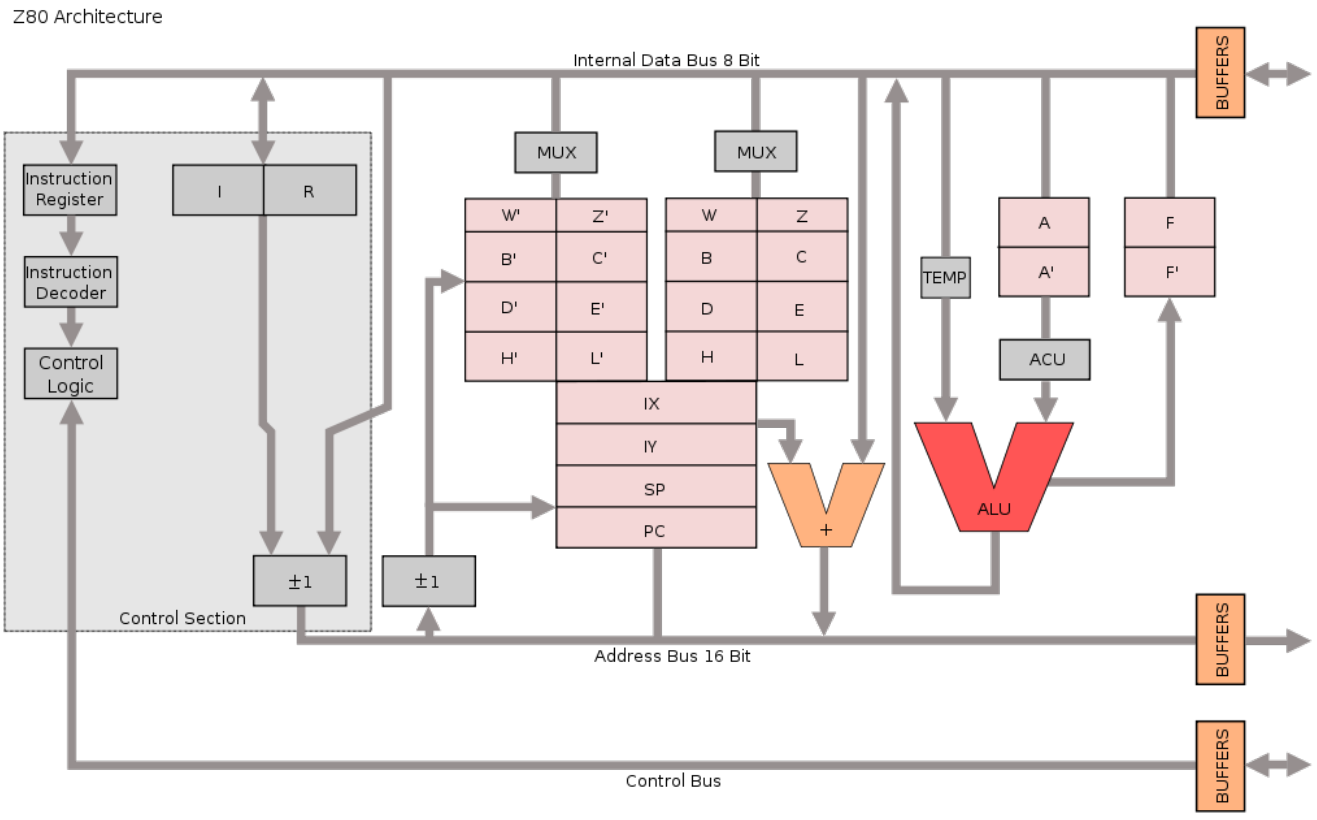

Figura 1

TERMINALES DEL MICROPROCESADOR

- LÍNEAS DE DIRECCIONES (A0 - A15), Pines 30-40, 1-5 respectivamente

  Se forman con 16 líneas de direcciones, tienen la facultad de establecerse en

tercer estado, estas señales proporcionan las direcciones correspondientes a intercambios de datos entre la memoria, la CPU y los puertos de los periféricos, la capacidad de direccionamiento con 16 bits es de 64 Kbytes y 256 puertos de entrada y salida, son activas en estado alto, los 8 bits menos significativos se usan para permitirle al usuario seleccionar los 256 puertos E/S, (A0-A7), en donde A0 es el bit menos significativo[4].

- LÍNEAS DE DATOS (DO - D7), Pines 14, 15, 12, 8, 7, 9, 10 y 13

  Se forman con 8 líneas de datos bidireccionales con capacidad del tercer estado, son activas en nivel alto, se utilizan para el intercambio de datos con la memoria, y periféricos de E/S[4].

- CICLO DE MAQUINA UNO (M1)

  Salida activa en nivel bajo, indica que en este ciclo de máquina uno el microprocesador va a obtener el código operacional de una instrucción, en las instrucciones que tienen un código operacional de 2 bytes esta señal se opera al obtener cada uno de los bytes del código operacional, al igual que para indicar el reconocimiento de un ciclo de interrupción cuando ocurre (IORQ)'[4].

- REQUERIMIENTO DE MEMORIA (MREQ)', Pin 19

  Salida activa en nivel bajo, esta señal indica una petición que interrelaciona a la memoria con la CPU, obtiene una dirección valida de las líneas de direccionamiento, esta terminal tiene capacidad del tercer estado[4].

- REQUERIMIENTO DE E/S (IORQ), Pin 20

  Es salida triestado activa en nivel bajo, esta señal indica que la mitad baja del bus de direcciones mantiene una dirección válida de E/S, para efectuar una operación de lectura o escritura de E/S, se genera esta señal cuando el ciclo de maquina 1 (M1) reconoce una interrupción, indica que el vector de respuesta de la interrupción se coloca en el bus de datos, las operaciones de reconocimiento de interrupción ocurren durante el ciclo de maquina 1, mientras que las operaciones de E/S nunca se producen durante este ciclo[4].

- LECTURA (RD), Pin 21

  Salida triestado activa en nivel bajo, indica que la CPU desea leer datos desde la memoria de un dispositivo externo de E/S, el dispositivo E/S se direcciona a la memoria o al periférico, se usa esta terminal para dirigir los datos al bus de datos de la CPU[4].

- ESCRITURA (WR)', pin 22

  Salida triestado activa en nivel bajo, indica que el bus de datos de la CPU va a obtener datos válidos para ser almacenados en la memoria o en algún dispositivo de E/S[4].

- REFRESCO DE LA MEMORIA DINÁMICA (RFSH)', Pin 28

  Salida activa en nivel bajo, indica que los siete bits inferiores de las líneas de direccionamiento contienen una dirección válida de refresco de memoria, se utiliza para el mantenimiento de datos en memorias dinámicas, con esta se efectúa una lectura de refrescamiento para todas las memorias dinámicas[4].

- PARO (HALT), Pin 18

  Salida que activa en nivel bajo, indica que la CPU realiza una instrucción por software de paro (HALT), y que espera una interrupción (NMI)' o (INT)' antes de que continué la operación, mientras permanezca en este estado la CPU ejecuta operaciones NOP, para mantener activo el refresco de las memorias dinámicas, al aplicarse un reset se continua con la operación[4].

- ESPERA (WAIT), Pin 24

  Es una entrada activa en nivel bajo, le indica al microprocesador que la memoria direccionada o los dispositivos periféricos de E/S no son tan rápidos como para realizar una transferencia de datos a la velocidad de la CPU, o no están listos para una transferencia de información, la CPU continua con el estado de espera durante todo el tiempo que esta terminal es activa, esto les permite a los otros dispositivos sincronizarse con la CPU[4].

- REQUISICIÓN DE INTERRUPCIÓN MASCARABLE (INT), Pin 16

  Entrada activa en nivel bajo, esta terminal se acciona con dispositivos E/S externos, una requisición (INT)' se atiende al final de la instrucción que se ejecuta, si el enable interno del Flip Flop de interrupción IFF1 controlado por software se encuentra habilitado, y si la requisición de bus no esta activa, al aceptar la CPU una interrupción envía una señal de reconocimiento, la petición de E/S se realiza durante el ciclo de máquina 1, al principio del siguiente ciclo de instrucción, esta petición solo es valida bajo control del programa interno, reconociendo la CPU tres modos diferentes de interrupción[4].

- INTERRUPCIÓN NO MASCARABLE (NMI)', Pin 17

  Entrada que se activa con un flanco de bajada mediante un impulso que

identifica una interrupción obligada, posiciona al contador de programa (PC) en la dirección 0066h desde donde continua el proceso, esta tiene una prioridad más alta que la interrupción (INT)' y siempre se reconoce al final de la instrucción que se ejecuta, independientemente del estado del IFF1, el contador de programa PC se almacena automáticamente en el stack pointer externo de forma que el usuario regrese al programa en el mismo punto del que fue interrumpido[4].

- REHABILITACIÓN (RESET), Pin 26

  Entrada que se activa con un flanco de bajada mediante un impulso, obliga a la CPU a reiniciar su actividad, coloca al contador de programa (PC) en la localidad de inicio de memoria 0000h, desde donde empieza el proceso, durante este tiempo el bus de direcciones y el bus de datos adquieren el estado de alta impedancia y todas las terminales de control de salida adquieren el estado inactivo[4].

- REQUERIMIENTO DE LAS TERMINALES DE LA CPU (BUSRQ), Pin 25

  Esta entrada es activa en nivel bajo, le indica a la CPU que coloque todas sus líneas en estado de alta impedancia, (tan pronto el ciclo de maquina 1 actual termine), a petición del periférico externo que desea tomar el control del sistema, regresa el control a la CPU cuando esta señal (BUSRQ)' pasa al nivel alto, se utiliza para pedir que el bus de direcciones, el bus de datos y las terminales de salida triestado del bus de control vayan a un estado de alta impedancia de tal forma que otros dispositivos controlen esos buses[4].

- ENTREGA DE LAS TERMINALES DE LA CPU (BUSAK)', Pin 23

  Salida activa en nivel bajo, es una indicación para el periférico que efectúa una petición (BUSRQ)' de que su petición ha sido concedida por parte del microprocesador, sirve para indicar al dispositivo que solicita este reconocimiento, que el bus de direcciones, el bus de datos y el bus de las terminales de control triestado han sido puestos en su estado de alta impedancia y que el dispositivo externo puede ahora controlar estas terminales[4].

- RELOJ (CK), Pin 6

  Entrada configurada por un tren de impulsos útiles, es la diferencia que permite la secuencia de tiempos de operación, se implanta físicamente con un oscilador de onda cuadrada cuya frecuencia depende del tipo de características de la CPU Z80, requiere oscilación de una fase con niveles TTL, una forma de satisfacer todos los requerimientos de voltaje es por medio de una resistencia de activación "pull up" de 330 ohms conectada entre +Vcc y la terminal de

salida de un oscilador implantado con circuitos TTL que generen oscilaciones[4].

- ALIMENTACIÓN POSITIVA DE +5 VOLTS (Vcc), Pin 11

  Esta es una entrada de alimentación de tensión de +5 volts con un 5 % de tolerancia[4].

- TIERRA (GND), Pin 29

  Terminal de alimentación negativa, requiere un potencial de 0.0 volts que sirven de referencia para la interconexión de los dispositivos[4].

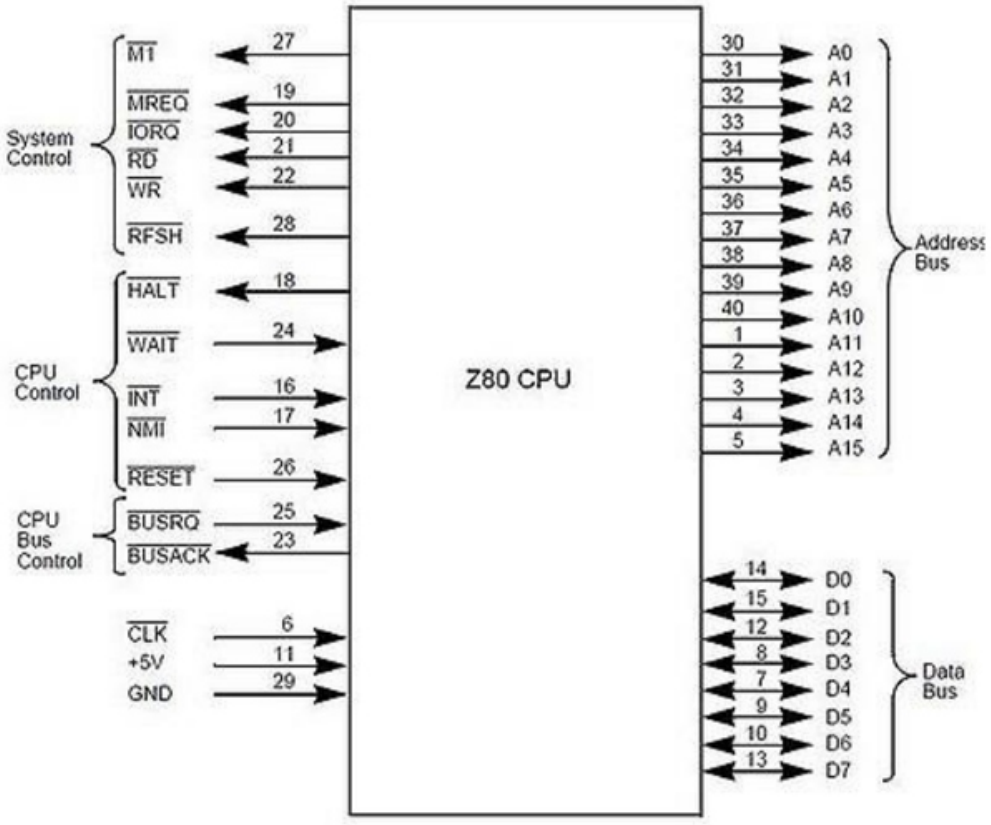

Figura 2

5. VISTAS DEL PROGRAMA

Las vistas que se pueden observar en el programa son la de inicio que se muestra en la Figura 3, la cual se encarga de dar inicio a este emulador, y que nos direcciona a la zona de ensamblado, cargador enlazador y procesador, además de mostrarnos información acerca de el funcionamiento del programa.

Figura 3

Otra de las vistas que se verán en este programa serán las del ensamblador como se muestra en la figura 4, el cual contiene dos paneles donde el de la izquierda permite entrada de texto como lo es el código en lenguaje ensamblador, la cual estará seccionada para ingresar etiquetas, instrucciones y parámetros y el otro panel de la derecha nos muestra el resultado de ensamblar este código, mostrando la instrucción que se compilo, su código binario producto de este trabajo y si tiene dato asociado.

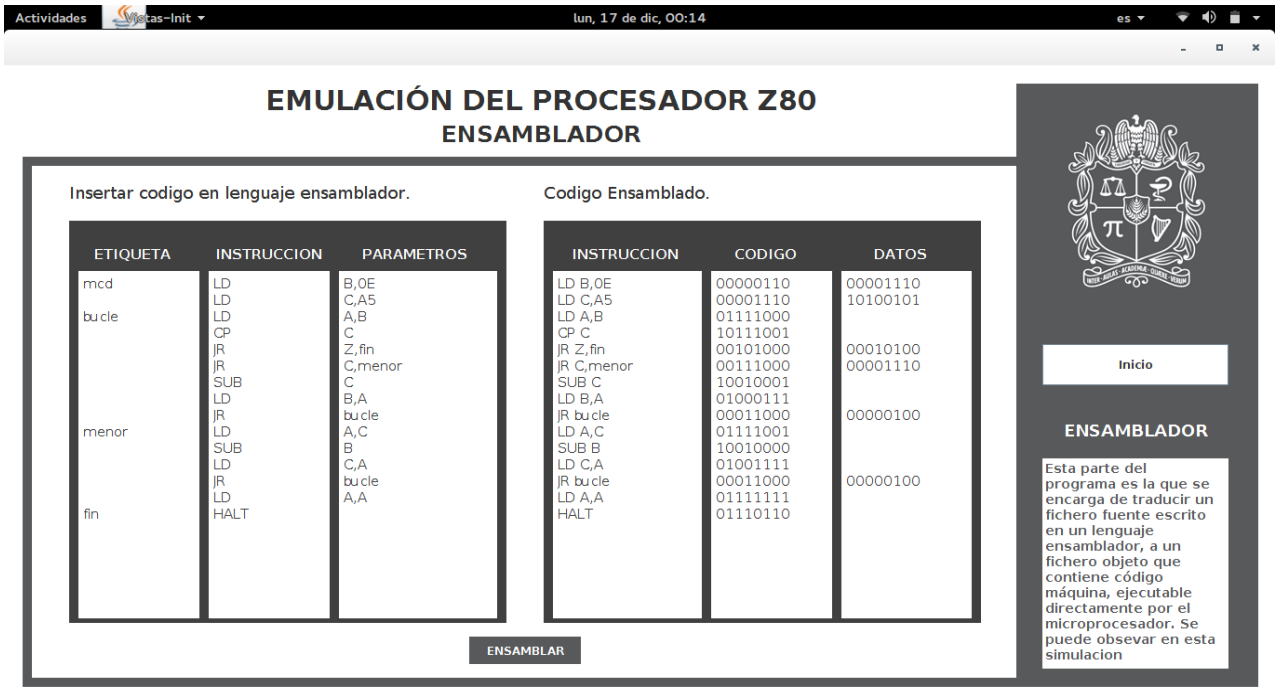

Figura 4

La vista de la parte del Cargador-Enlazador se ve puede observar en la Figura 5. donde podremos ir cargando cada uno de los códigos maquina en memoria con su respectivo dato, la memoria se actualiza y veremos en tiempo real lo que va sucediendo, se podrá indicarle la posición inicial para donde se cargará el programa, y si desea cargar instrucción por instrucción o todo el programa cargándolo todo.

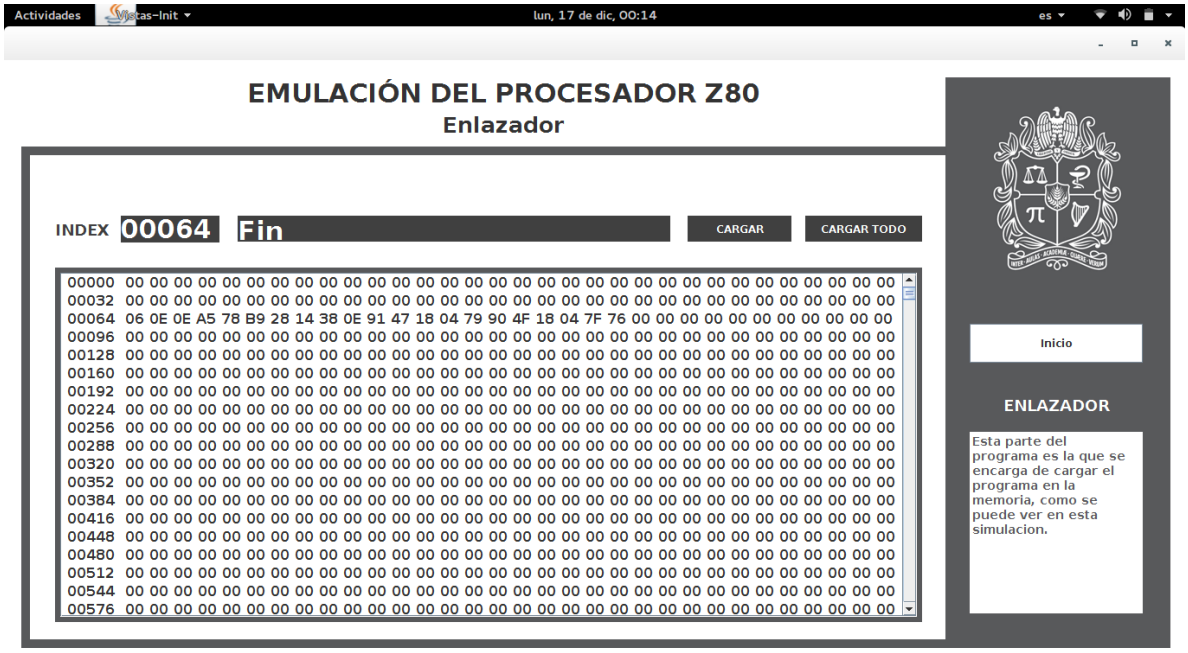

Figura 5

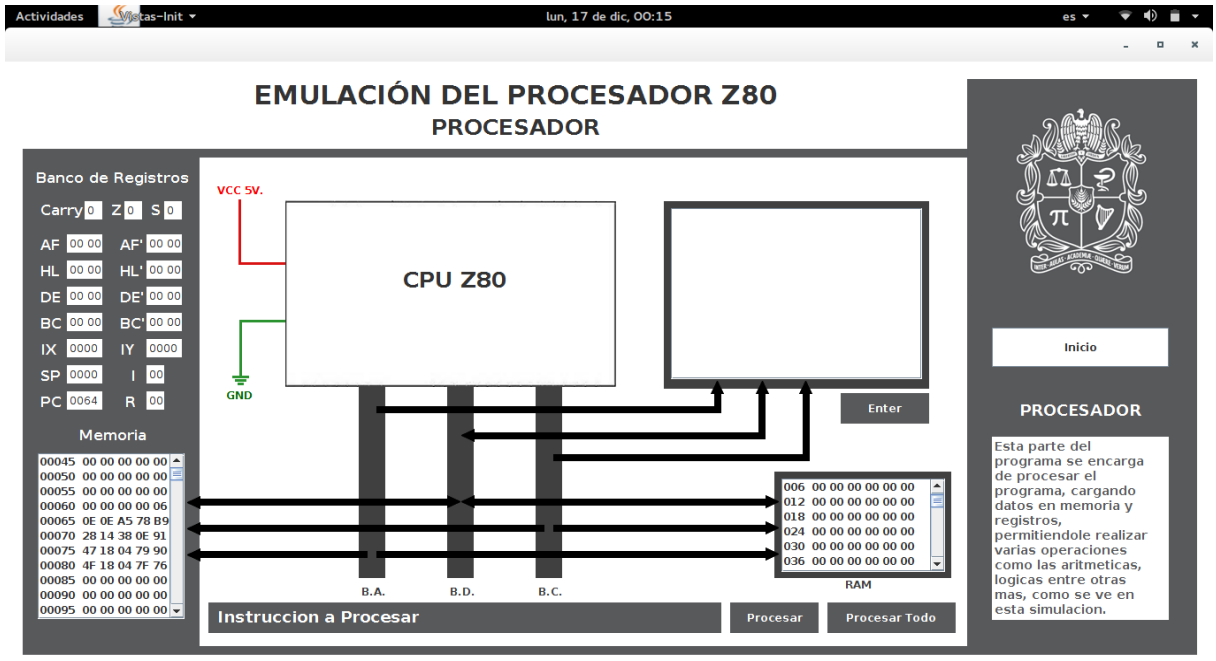

Figura 6.

Finalmente la vista del microprocesador donde podemos ir ejecutando cada una de las acciones y ver que sucede el los registros almacenados en la ROM y como se van cambiando de acuerdo a las instrucciones que se estén ejecutando. Lo podemos ver en la Figura 6. donde podemos ir procesando cada unas de las instrucciones, ejecutar todas las instrucciones y ver resultados, como también podemos ver y hacernos una idea de como funcionan estas arquitecturas, donde incluimos un periférico de entrada y salida de texto en la parte superior derecha.

6. MICROPROCESADOR CONSTRUIDO

El microprocesador, ensamblador y enlazador-Cargador se realizaron con el fin de emular lo mas cercano posible el procesamiento de información mediante esta arquitectura mencionada ya anteriormente, en este trabajo se lograron trabajar con muchas instrucciones que se mencionaran a continuación las cuales pueden ser usadas para procesar información mediante ellas.

- CARGA(86 Instrucciones LD)

  En este podemos guardar información en registros o direcciones de memoria proveniente de otros registros, entrada de datos o otras direcciones de memoria por ejemplo LD (HL),D, donde cargamos lo que tenemos guardado en el registro D a la dirección de memoria dada por los registros HL.

- INCREMENTO(12 Instrucciones INC)

  Donde podemos incrementar en 1 registros de 8 y 16 bits guardados en los registros como por ejemplo inc b, donde incrementamos en 1 el dato que esta alojado en b.

- DECREMENTO(12 Instrucciones DEC)

  Realizamos en decremento en 1 en datos de 8 o 16 bist guardados en registros, como por ejemplo DEC DE.

- SALTO(5 Instrucciones JR)

  Como necesitamos realizar saltos dentro de la dirección de memoria para no realizar iteraciones secuenciales solamente, y que mediante etiquetas podemos llevar a cabo estos saltos, implementamos instrucciones de saltos que verifican información de las banderas como el carry, ZERO y S las cuales evalúan si operaciones como restas o sumas nos dejan acarreo, o como resultado cero, menor, mayor. Como ejemplo de esta podemos dar la JR Z,fin que si la bandera ZERO es verdadera saltamos a la dirección de memoria que tiene la etiqueta fin, donde acaba la ejecución del bloque de instrucciones para cierto programa.

- SUMA(13 Instrucciones ADD)

  Una de las funciones del procesador presentes en la ALU es realizar operaciones aritméticas entre datos, ADD es una en la que se pueden sumar datos entre si y guardarlos en el registro A como resultado, para esto sumamos A con otro registro como por ejemplo ADD A,B.

- SUMA CON ACARREO(9 Instrucciones con acarreo ADC)

  Aveces los datos no son de 8 bits y al sumar los primeros 8 se nos produce un acarreo el cual es muy importante tenerlo en cuenta al realizar la suma del siguiente par de 8 bits, para esto se creo esta instrucción, ADC A,C.

- RESTA(9 Instrucciones de SUB)

  Otra de las operaciones aritméticas son la resta y podemos realizar la resta entre dos datos con esta instrucción, que si generamos acarreo se activa la bandera del carry, el resultado se aloja en el registro A.

- RESTA CON ACARREO(9 Instrucciones SBC)

  Cuando realizamos la resta de los primero 8 bits entre dos datos aveces se genera un acarreo activando la bandera carry, donde si el numero es de 16 bits, al seguir restando los siguientes 8 bits debemos tener encenta este acarreo.

- AND(9 Instrucciones AND)

  Podemos realizar operaciones lógicas entre el registro A y otro registro que indiquemos como por ejemplo AND B donde aplica la operación lógica and.

- OR(9 Instrucciones OR)

  Podemos realizar operaciones lógicas entre el registro A y otro registro que indiquemos como por ejemplo OR B donde aplica la operación lógica or.

- XOR(9 Instrucciones XOR)

  Podemos realizar operaciones lógicas entre el registro A y otro registro que indiquemos como por ejemplo XOR B donde aplica la operación lógica xor.

- CP (9 Instrucciones CP)

  Podemos realizar restas entre el registro A y cualquier otro registro donde el

resultado no afecta en nada, pero si activa las banderas, muy usado para condicionales. CP C.

- IN(1 Instrucción de entrada de datos)

  A la hora de trabajar con periféricos de entrada, queremos ingresar datos a nuestro programa que se esta procesando, mediante esta instrucción le indicamos al procesador que debe esperar una entrada, esta instrucción viene con una dirección asociada, la cual me indica el puerto de entrada y almacenamiento de esta entrada. IN A,(01H).

- OUT

  Luego de ver el procesamiento de este microprocesador queremos ver la efectividad de lo que hace, y enviar respuesta a los periféricos y que lo usuarios puedan evidenciar que esta haciendo los procedimientos de manera adecuada. Para esto se crea esta instrucción la cual viene acompañada de una dirección, que me indica el puerto a donde debe ir el resultado alojado en el registro A, y que sera mostrado al usuario. OUT (01H),A.

7. ESCENARIOS DE PRUEBAS

Iniciamos con un caso de prueba suave donde verificaremos si un números es múltiplo de otro, los dos ingresados desde el lenguaje ensamblador, imprimirá 1 si uno es múltiplo del otro y 0 si no lo es, el código ensamblador es el siguiente.

```
Mult    LD   A,15
        LD   B,09
bucle   SUB  B
        JR C,no
        JR Z,si
        JR bucle
si      LD A,01
        OUT (01H),A
no      LD A,00
        OUT (01H),A
fin     HALT
```

Pero ahora para los casos de prueba mas complejos y que funciona adecuadamente se planteo inicialmente el algoritmo para calcular el máximo común divisor de dos números, donde tomamos como ejemplo los numero 110 y 165 dándonos como resultado el numero 55. A continuación el código en lenguaje ensamblador listo para introducir al programa.

```
Mcd     IN    A,(01H)
```

```
        LD    B,A
        IN    A,(02H)
        LD    C,A
bucle   LD    A,B
        CP    C
        JR    Z,salida
        JR    C,menor
        SUB   C
        LD    B,A
        JR    bucle
menor   LD    A,C
        SUB   B
        LD    C,A
        JR    bucle
salida  OUT   (03H),A
fin     HALT
```

Como se menciono este código debe introducirse en la ventana del ensamblador organizado por columnas y llevarlo a la parte final de procesamiento, en el procesador le pedirá ingresar los datos y ahí mismo retornara el resultado.

8. RESPONSABLE

Este trabajo fue realizado por el Brian Julian Moreno Niampira, estudiante de Ingeniería de Sistemas y Computación de la Universidad Nacional de Colombia sede Bogotá, el cual fue asignado por el docente Jorge Eduardo Ortiz como proyecto de la materia Lenguajes de programación. Julian Moreno realizo en diseño y programación de este programa realizado para emular el trabajo del microprocesador Z80.

9. BIBLIOGRAFIA

- LenguajeEnsamblador_MicroProcesador_Z80.pdf
- https://es.wikipedia.org/wiki/Zilog_Z80
- http://clrhome.org/table/
- http://galia.fc.uaslp.mx/~cantocar/microprocesadores/TUTORIALES/EL_MICRO_Z80/ARQUITECTURA_DEL_MICROPROCE.HTM