Explicar el funcionamiento de Collapse OS. La arquitectura del microprocesador z80 y simular ejemplos para z80. (mayo de 2020)

Jhonatan Tituaña, Bryan Azuero, Javier Arteaga

Resumen -

En el presente informe se muestran los resultados de la práctica de la asignatura Arquitectura de computadoras, en la cual se busca conocer el sistema operativo Collapse OS. A lo largo del desarrollo se conocerá todos los aspectos de este sistema y el microprocesador z80, utilizando el simulador del procesador Z80. Durante el transcurso del informe se muestra y se explica el algoritmo que da solución de un problema.

Índice de Términos -
Lenguaje ensamblador microprocesador, simulador, memoria de datos

I. INTRODUCCION

En el presente artículo analizaremos en profundidad el sistema operativo Collapse Os y la arquitectura del microprocesador z80. El microprocesador es un circuito integrado encargado de procesar y ejecutar instrucciones codificadas en lenguaje binario. El microprocesador Z80 realiza las funciones de una unidad central de proceso (CPU), interpretando las instrucciones que le son dadas mediante el lenguaje de ensamblador. En este caso, el simulador del z80 será puesto a prueba en diferentes casos con la finalidad, de entender de mejor manera su arquitectura y funcionalidad.

Los microcontroladores se han constituido en un elemento primordial para el avance de las nuevas tecnologías, de ahí parte la necesidad de no solo conocerlos en sus usos también como emplearlos, la programación se convierte en algo apremiante de conocer, las herramientas con las cuales se pueda adquirir conocimiento son muy importantes.

A. Justificación

A. Conocer el sistema operativo Collapse OS junto con la estructura del microprocesador Z80 y encontrar la relación entre ellos posteriormente realizar un ejemplo del microprocesador Z80.

B. Objetivos generales y específicos

A. INVESTIGAR EL FUNCIONAMIENTO DEL SISTEMA OPERATIVO COLLAPSE O.S Y COMO ES QUE FUNCIONARÁ CON MICROPROCESADORES Z80 DE 8 BITS

B. INVESTIGAR LA ARQUITECTURA DEL MICROPROCESADOR ZILOG Z80 PARA REALIZAR ALGORITMOS QUE NOS PERMITAN HACER SIMULACIONES Y COMPRENDER MEJOR SU FUNCIONAMIENTO.

C. COMPRENDER Y EXPLICAR EL FUNCIONAMIENTO DE COLLAPSE OS

D. CONOCER LA ARQUITECTURA DEL MICROPROCESADOR Z80

SIMULAR EJEMPLOS DEL Z80

II. ESTADO DEL ARTE

El investigador, desarrollador Virgil Dupras es creador de 'Collapse OS', un sistema operativo de código abierto que, afirma, está diseñado para funcionar en aquellos componentes electrónicos fáciles de reciclar. Es decir, es una plataforma que serviría para aprovechar la basura electrónica cuando ya no existan nuevos dispositivos electrónicos. Está pensado para ser usado en los momentos de un mundo postapocalíptico donde la tecnología ya no existe.

A. Microprocesadores

El Intel 8080, lanzado en 1974, fue el primer microprocesador de 8 bits que tuvo éxito entre la comunidad de programadores, teniendo como principal exponente al MITS Altair 8800. Dos años después y con la intención de hacerse con el mercado, Zilog lanzó el Z80, que atrajo a los programadores de la época al ser totalmente compatible con la arquitectura del Intel 8080 y añadir nuevas e interesantes funcionalidades, como un set de instrucciones extendido. Además, no se sentía calificado para desarrollar una nueva arquitectura de microprocesador de 16 bits a partir de todo el tejido, por lo que buscó un arquitecto de procesadores experto. Encontró al Dr. Peuto, que había estado trabajando en la arquitectura mainframe 470 / V6 compatible con IBM de Amdahl desde 1973. Peuto se convirtió rápidamente en el duodécimo empleado de Zilog. (leibson, 2019) [1]

B. El Z80 tenía ocho mejoras fundamentales

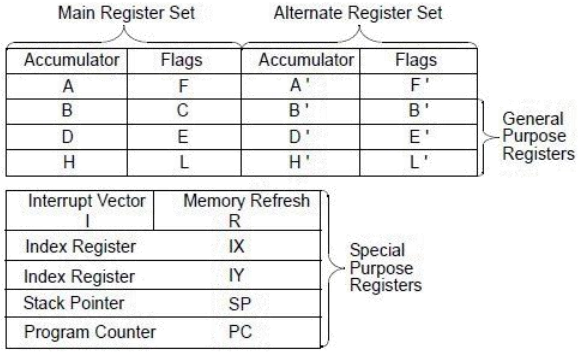
- *Un conjunto de instrucciones mejorado, incluyendo los nuevos registros índice IX e IY y las instrucciones necesarias para manejarlos.*
- *Dos bancos de registros que podían ser cambiados de forma rápida para acelerar la respuesta a interrupciones.*
- *Instrucciones de movimiento de bloques, E/S de bloques y búsqueda de bytes.*
- *Instrucciones de manipulación de bits.*
- *Un contador de direcciones para el refresco de la DRAM integrado, que en el 8080 tenía que ser proporcionado por el conjunto de circuitos de soporte.*
- *Alimentación única de 5 voltios.*
- *Necesidad de menos circuitos auxiliares, tanto para la generación de la señal de reloj como para el enlace con la memoria y la E/S.*
- *Más barato que el Intel 8080.*
- *Un tipo especial de reset que sólo reinicia el contador de programa de modo que el Z80 se puede usar en un sistema de desarrollo ICE [1] [3]*

C. Estructura del Zilog Z80

La informática, como muchos otros procesos, tiene tres partes principales. En un sistema informático, los datos (números y palabras) son ingresados por un dispositivo de entrada, el procesamiento es realizado por una unidad central de procesamiento y los datos son salida por un dispositivo de salida.

Entonces, para un sistema de microprocesador, se usa un teclado como dispositivo de entrada y una pantalla como dispositivo de salida.

Los componentes del Z80 que son más importantes para un programador son los registros que se muestran en la Figura:



El acumulador es un registro de 8 bits utilizado para aritmética y operaciones lógicas.

El registro de bandera se utiliza para contener información sobre los resultados de algunas operaciones. Por ejemplo, el registro de la bandera indica si el resultado de agregar un número al acumulador es positivo, cero o negativo.

Registros de propósito general:

GRUPO 1: los registros principales son

A, B, C, D, E, H, y L

GRUPO 2: los registros alternos

A', B', C', D', E', H' y L

Estos registros son de 8 bits, pero pueden agruparse por pares (sin contar "A Y A'") en registros de 16 bits de la forma BC, DE, HL y B'C', D'E', H'L'. Tienen la particularidad de que solo son accesibles a la vez 6 de ellos, pues están divididos en 2 bancos intercambiables por medio de unas instrucciones.

Una vez hemos visto toda la información correspondiente a los registros, podemos pasar a las instrucciones.

Mencionaremos algunas de las instrucciones que utilizamos entre otras, en los grupos siguientes:

- Transferencias de información: LD, PUSH, POP...
- Operaciones aritméticas: ADD, INC, DEC, CP...
- Operaciones lógicas: AND, OR, XOR...
- Cambios y rotaciones: RLC, RRC, SRA...
- Manipulación de bits: RES, SET, BIT...
- Instrucciones de Control: HALT, NOP, DI, EI...
- Saltos: JP, CALL, JR... [1] [2]

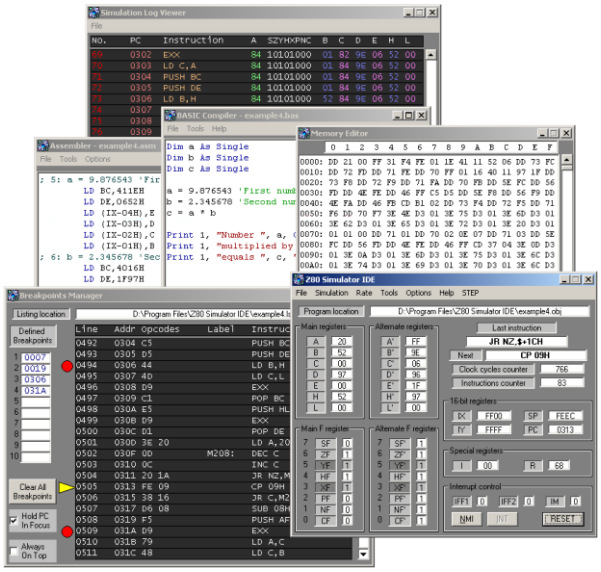
III. DESARROLLO

Z80 Simulator IDE

Z80 Simulator IDE es una potente aplicación que proporciona a los usuarios y educadores del microprocesador Z80 un entorno de desarrollo gráfico fácil de usar para Windows con simulador integrado (emulador), compilador básico Z80, ensamblador, desensamblador y depurador para el microprocesador Zilog Z80 de 8 bits.

El siguiente programa permite simular y ejecutar los programas que se desee hacer sin la necesidad de tener el microprocesador Z80 físicamente.

Se programa en ensamblador el cual es un lenguaje de programación muy fácil de entender ya que está calificado de bajo nivel, y constituye la representación más directa del Código máquina específico para cada arquitectura de computadoras legible por un programador.



III. EJEMPLOS EN Z80 SIMULATOR IDE

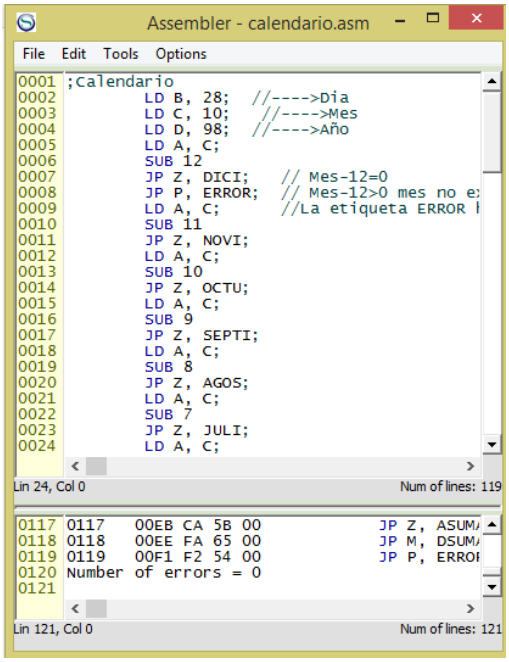
A. Calendario

Los ejemplos van a ser realizados en el Z80 Simulator IDE programados en lenguaje ensamblador.
NOTA: el programa almacena los números en código hexadecimal, para nosotros ingresaremos números en decimal.

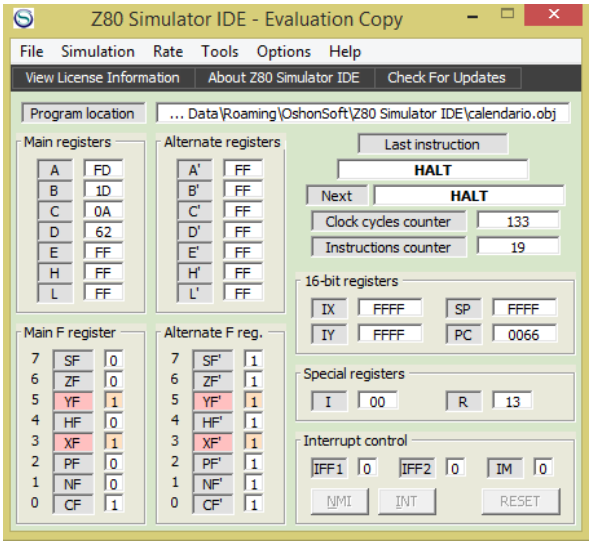
1. Instrucciones
   - LD: cargar datos a/desde los registros
   - SUB: sustracción
   - JP: salto
   - INC: incremento
   - HALT: esperar por interrupción o reset
   - END: fin

2. Ejecución del programa

El primer programa lo llamaremos “calendario” consiste de tres entradas (día, mes, año), el cual nos va devolver el día siguiente.
Nota: si el día ingresado es 31 de diciembre del 2019 el resultado ser 01 de enero del 2020, solo en ese caso cambiaras todas las salidas.
Ejemplo:
Utilizaremos la siguiente fecha:
28 de octubre de 1998
R: 29 de octubre de 1998



El codigo fuente sera escrito en la parte de Assembler el resultado lo observaremos en la interfaz principal.



Los resultados los observaremos de la siguiente manera:
B= el dia
C= el mes
D= el año
Todo esto en codigo hexadecimal.

B. Serie de Finobacci

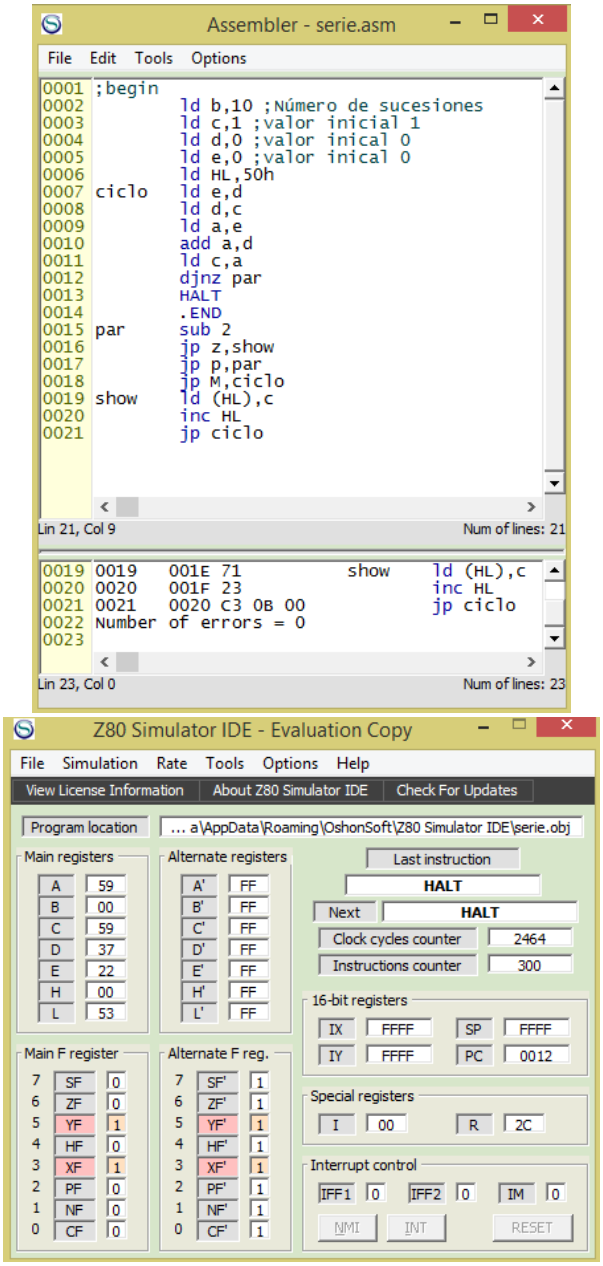
1. Instrucciones
   - LD: cargar datos a/desde los registros
   - JP: salto
   - HALT: esperar por interrupción
   - ADD: sumar
   - DJNZ: detectar cero-saltar
   - INC: incremento
   - END: fin

2. Ejecucion del programa

El segundo programa lo llamaremo solo “Finobacci” el cual va a realizar la siguiente serie que consiste en sumar un numero con el anetior y asi sucesivamente:

0,1,1,2,3,5,8,13…….

Nota: nosotro ingresaremos los primeros numero y la cantidad de numeros de la serie que deseemos.



El resultado sera mostrado en el registro “C” el cual ira cambiando mientras avanza el proceso.

IV CONCLUSIONES

- De presentarse un evento catastrófico, este proyecto significaría una posibilidad de reestablecer el desarrollo tecnológico a base de pocos recursos utilizando un sistema operativo que no exige de muchos requisitos físicos para implementar un equipo que pueda ejecutarlo.
- El sistema basado en un procesador z80 ofrece la posibilidad de ser construido a base de materiales reciclados, sin embargo, su interfaz no es precisamente la más amigable para un usuario promedio, por lo que las personas que tengan mayor dominio sobre los dispositivos electrónicos destacaran.
- La arquitectura basada en el modelo de Von Newman ha prevalecido desde 1945 y demuestra lo eficaz que es en la práctica.

RECOMENDACIONES.

Manejar conocimientos en área de programación y sistemas digitales seria de vital importancia en el escenario con el que se justifica este proyecto. Por lo que se recomienda:

- Diseñar sistemas basados en este tipo de arquitectura, entendiendo su funcionamiento y simulando aplicaciones que sean útiles para el ser humano, considerando que en la actualidad este tipo de sistemas, por su antigüedad, tienden a ser más accesibles.
- Difundir entre la comunidad científica el estado de desarrollo en el que se encuentra el proyecto, para que sea de conocimiento general e incentive a futuros proyectos investigación.

IV. BIBLIOGRAFÍA

[1] S. leibson, «Historia de la computacion,» 2019. [En línea]. Available: https://www.eejournal.com/article/in-memoriam-dr-bernard-peuto-architect-of-zilogs-z8000-and-z8/.

[2] E. G. Sally Adee, «25 Microchips,» IEEE Spectrum, mayo 2009. [En línea]. Available: https://sci-hub.tw/10.1109/MSPEC.2009.4907384. [Último acceso: 03 junio 2020].

[3] A. Davies, «Trade-offs in fixed-point multiplication,» Digital arithmetic, junio 1979. [En línea]. Available: https://sci-hub.tw/10.1049/ij-cdt.1979.0023.