Universidad Nacional Autónoma De México

Facultad De Ingeniería

Materia: Estructura y Programación de Computadoras

Profesor: Alberto Templos Carbajal

Proyecto Final De Estructura y Programación de Computadoras (Simulador Z80)

Grupo: 6

2024-2

Integrantes:

- Carreón Gonzalez Emmanuel
- Feria Ambriz Héctor Eduardo
- Marín Montaño Josué
- Rufino López María Elena

Índice

Introducción

Presentación del Sistema desarrollado

Simulador Z80 en página web desarrollado con JavaScript, HTML y CSS como tecnologías principales.

Objetivo

El proyecto final de la materia realizado fue un simulador en línea del microprocesador Z80. El objetivo principal de este proyecto fue poder implementar un simulador más accesible y eficiente que el simulador con el que se trabajó a lo largo del semestre. A la interfaz gráfica del simulador se le añadieron botones especiales para poder acceder al manual, con el objetivo de facilitar su uso y se enteren de las restricciones que tiene para poder codificar en el mismo. También se añadió un botón para acceder al código fuente del simulador.

Características principales

- La sintaxis de este simulador es similar al simulador usado en clase.
- Se puede compilar
- Se puede limpiar la memoria
- Se puede ejecutar paso a paso el programa

Información adicional de importancia

La principal razón por la que se decidió desarrollar este proyecto fue por la ineficiencia del simulador que se usó en el semestre. Esto debido a que, para poder usarlo se necesitaba tener un sistema operativo de 32 bits, y para poder usarlo en un sistema de 64 bits era necesario instalar aparte un simulador que pueda emular el sistema solicitado. Sumado a este inconveniente también la manipulación del simulador SZ80 no es fácil y no tiene un manual accesible para manipularlo.

Es importante agregar que, en la parte de la codificación del simulador, no se logró añadir el uso de una pila, por lo tanto todas las instrucciones que incluyan el uso de una pila no podrán ser utilizadas sino hasta una versión del simulador que lo permita. En la codificación es importante el uso adecuado del espaciado en las instrucciones para que pueda leer satisfactoriamente el programa codificado. Y como se menciona en el manual técnico, este simulador no carga ni descarga los archivos generados.

Análisis

Especificación de Requerimientos de Software (ERS) - Proyecto Simulador Z80

Introducción

El simulador del microprocesador Z80 es un programa que te permite codificar en el lenguaje ensamblador Z80, compilarlo y generar sus respectivas líneas con dominio .HEX y .LST.

El proyecto se desarrolló siguiendo el paradigma orientado a objetos, haciendo uso de la herencia de clases, interfaz gráfica, carga de datos, etc.

Requerimientos Funcionales

- Uso de variables y datos
    - Uso de números en decimal y hexadecimal
    - Uso de datos y etiquetas
    - Almacenamiento en memoria
    - Limpiar memoria
- Generación de archivos .HEX y .LST
- Visualización de la ejecución paso a paso
- Interfaz Gráfica
    - Interfaz intuitiva de las diferentes funciones del programa
    - Campos de texto para añadir el código y memoria
    - Botones Para la selección de las operaciones del simulador
    - Resultados visuales de los archivos generados.

Requerimientos No Funcionales

- Desempeño:
    - Respuesta eficiente al compilar y generar los archivos.
- Portabilidad:
    - Compatibilidad con sistemas operativos Windows, Linux y macOS
    - Compatibilidad con dispositivos móviles
- Mantenibilidad:
    - Código modular y con una estructura de fácil entendimiento para su modificación.

Diseño.

Introducción.

El diseño de sistema implementado, se basó en un estilo arquitectónico orientado a los sistemas interactivos, puesto que se ha empleado el MVC (modelo-vista-controlador) para que el usuario pudiera interactuar con la interfaz del sistema y poderle proporcionar sus respectivos archivos solicitados.

El lenguaje propuesto para el programa se realizó en JavaScript, orientado a objetos; por lo que la metodología destinada fue el Diseño OO donde definimos los objetos principales del sistema . HTML y CSS para la estructura de la página web y sus respectivas interfaces de los objetos.

Arquitectura del sistema: El sistema está compuesto de varios archivos esenciales.

- **Z80.js:** El código define la estructura básica de un emulador de la CPU Z80 con memoria y registros, así como funciones para compilar y cargar programas en memoria, manipular banderas de estado, verificar condiciones y ejecutar instrucciones. En esencia, permite simular el comportamiento de la CPU Z80 ejecutando programas escritos en ensamblador.
- **Compiler.js:** El código es un compilador para ensamblador Z80, que convierte código fuente en lenguaje ensamblador en bytes de máquina que pueden ser ejecutados por una CPU Z80.
- **hyl.html**: Incluye scripts de JavaScript que contiene las funciones del traductor de ASM a LST y HEX, además de las definiciones para leer las instrucciones del simulador y las alertas de error.
- **index.html**: Es el archivo HTML principal del programa, contiene las etiquetas de los apartados principales de lo mostrado al usuario.
- **styles.css**: Este archivo contiene los estilos de la GUI con el fin de ser más amigable con el usuario y de fácil entendimiento para su utilización.

Pruebas

Introducción

Una incidencia, falla o error es un resultado o comportamiento no esperado del sistema o módulo que es sometido a las pruebas. En consecuencia, el objetivo de las pruebas es encontrar errores en el análisis , diseño y/o desarrollo de software a la medida, para corregirlos y mejorar la calidad del producto antes de ser liberado. Es necesario ahondar más en las pruebas funcionales, como lo puede ser la navegación de GUI, usabilidad, integrales. Entonces, las pruebas funcionales que se van a realizar serán manuales y de caja blanca, ya que se conoce todo el código fuente.

Ahora bien, con relación al plan de pruebas, decidimos verificar el correcto funcionamiento de la interfaz, las respuestas y la fiabilidad en los resultados.

Casos de prueba

OP01	
Verificar operaciones LD, ADD en registros principales (Ejecución Total).	
Pasos por seguir para realizar la prueba	**Resultado Esperado**

*Precondición de abrir el simulador Z80 en el navegador del sistema
 	
1. Escribir en el apartado de “Código” las líneas del programa de prueba01
LD A, 5
LD B, 10
ADD A, B
LD C, A
HALT
 	
2. Dar clic en el botón “Compilar”
 	
3. Dar clic al botón “Ejecución total”
	· Debemos poder escribir las líneas de código sin inconvenientes en el recuadro marcado
	
	· Observaremos que en la memoria se escribe el código de nuestro programa
	
	· Observamos que el programa comienza a ejecutar las líneas de código puestas en memoria una tras otra de manera automática pasando el PC sobre cada espacio de la memoria modificando los registros correspondientes

OP02	
Verificar operaciones de incremento y decremento sobre los registros principales (Ejecución Paso a Paso)	
Pasos por seguir para realizar la prueba	**Resultado Esperado**

*Precondición de abrir el simulador Z80 en el navegador del sistema
 	
1. Escribir en el apartado de “Código” las líneas del programa de prueba02
LD A, 7
INC A
LD B, A
DEC B
HALT
 	
2. Dar clic en el botón “Compilar”
 	
3. Dar clic al botón “Ejecución paso a paso” repetidas veces hasta terminar de ejecutar las líneas del programa
	· Debemos poder escribir las líneas de código sin inconvenientes en el recuadro marcado
	
	· Observaremos que en la memoria se escribe el código de nuestro programa
	
	· Observaremos que cada vez que presionamos una vez el botón de “Ejecución paso a paso” el PC avanza una dirección, de modo que se ejecuta de a poco el programa. De esta manera podemos ver cómo se modifican y actualizan los registros con cada instrucción

OP03	
Verificar operaciones de salto JP y uso de etiquetas (Ejecución Paso a Paso)	
Pasos por seguir para realizar la prueba	**Resultado Esperado**

*Precondición de abrir el simulador Z80 en el navegador del sistema
 	
1. Escribir en el apartado de “Código” las líneas del programa de prueba03
inicio:
LD A, 10
LD B, 5
JP suma
LD D, 20
suma:
ADD A, B
LD C, A
	· Debemos poder escribir las líneas de código sin inconvenientes en el recuadro marcado

2. Dar clic en el botón “Compilar”	· Observaremos que en la memoria se escribe el código de nuestro programa
3. Dar clic al botón “Ejecución paso a paso” repetidas veces hasta terminar de ejecutar las líneas del programa	· Observamos la ejecución paso a paso, en la que saltamos correctamente a la etiqueta “suma” y realizando las operaciones correspondientes en esta. Observamos tambien que al ser un salto directo, ignora la línea “LD D, 20” y no se modifica ese registro

OP04	
Verificar operación de salto JR en un ciclo de incremento (Ejecución Paso a Paso)	
Pasos por seguir para realizar la prueba	**Resultado Esperado**

*Precondición de abrir el simulador Z80 en el navegador del sistema
 	
1. Escribir en el apartado de “Código” las líneas del programa de prueba04
inicio:
LD A, 10
LD B, 1
LD C, 5
ciclo:
DEC C
JR Z, fin
ADD A, B
JP ciclo
fin:
HALT
	· Debemos poder escribir las líneas de código sin inconvenientes en el recuadro marcado

2. Dar clic en el botón “Compilar”	· Observaremos que en la memoria se escribe el código de nuestro programa
3. Dar clic al botón “Ejecución paso a paso” repetidas veces hasta terminar de ejecutar las líneas del programa	· Observamos la ejecución paso a paso, en esta realizamos un ciclo sencillo en que se suma el valor de B en el registro A una cantidad de veces C. Podemos ver en la ejecución dentro del “ciclo” que JR revisa la bandera Z de modo que cuando el registro C es igual a 0, la ejecución salta a la etiqueta “fin” terminando el programa

OP05	
Verificar operación ORG para dejar el programa en un sitio distinto de memoria	
Pasos por seguir para realizar la prueba	**Resultado Esperado**

*Precondición de abrir el simulador Z80 en el navegador del sistema
 	
1. Escribir en el apartado de “Código” las líneas del programa de prueba05
org 500
LD A, 5
LD B, 10
ADD A, B
LD C, A
HALT
 	
2. Dar clic en el botón “Compilar”
 	
3. En el apartado “Memoria” escribimos “500” y damos clic en el botón “Dirigir”
 	
4. Dar clic al botón “Ejecución paso a paso” repetidas veces hasta terminar de ejecutar las líneas del programa
	· Debemos poder escribir las líneas de código sin inconvenientes en el recuadro marcado
	
	· Observaremos que en la memoria se escribe el código de nuestro programa, en este caso la linea “org 500” hace que comience a escribir el programa en la dirección 500 de memoria
	
	· Podemos ver como nos posiciona en la memoria en la dirección indicada(en este caso la dirección 500)
	
	· Observamos la ejecución paso a paso, realizamos una operación sencilla de suma partiendo en la dirección 500 de memoria.

Reporte de resultados de las pruebas de software

Caso de Prueba	Resultado	Observaciones
OP01	Error	El simulador marca un error en la línea que contiene la instrucción HALT
OP02	Error	El simulador marca un error en la línea que contiene la instrucción HALT
OP03	RE	En este programa no pusimos la instrucción HALT y no hubo errores marcados por el simulador
OP04	Error	El simulador marca un error en la línea que contiene la instrucción HALT
OP05	Error	El simulador marca un error en la línea que contiene la instrucción HALT

Mantenimiento

Los mantenimientos que pueden ser aplicables a nuestro sistema son los siguientes:

- Mantenimiento preventivo:
    - Como es bien sabido, el mantenimiento preventivo de una página web implica una serie de acciones proactivas para evitar problemas antes de que ocurran y asegurar que la página web funcione de manera óptima. Un buen mantenimiento preventivo es revisar las actualizaciones de software; Cómo se utilizaron interfaces gráficas y botones para cada operación que realiza el simulador, la actualización de los controladores del mismo es de suma importancia. Actualizar los frameworks y las librerías de JavaScript que importamos.
    - Revisiones de seguridad: Paulatinamente se tienen que realizar revisiones, escaneos de seguridad periódicos, esto con el fin de detectar vulnerabilidades y malware.
    - Optimizaciones de rendimiento: Se tiene que hacer limpieza de la base de datos
    - Copia de seguridad: Es necesario tener una copia de seguridad del código fuente por cualquier adversidad.
    - Monitoreo y análisis: Monitorear el simulador que esté funcionando en su óptima manera, para que en caso de encontrar alguna falla poder corregirla..
    - Pruebas: Para esta parte es necesario hacer pruebas en diferentes navegadores que funcionen de la misma forma, así como en los diferentes sistemas operativos y dispositivos que lo soportan.11
- Mantenimiento correctivo:
    - Identificación y corrección de bugs: Identificar los errores de código que pueda generar, y lograr depurar dicho error para que trabaje con normalidad nuevamente.
    - Restauración de funcionalidad: Aquí se reparan las funcionalidades que no estén operando como se debería esperar (como los botones), y en dado caso que dicha falla persista, actualizar el framework a una versión más reciente.
    - Seguridad: En caso de encontrarse comprometido el proyecto, eliminar el código malicioso a la brevedad.
    - Optimización de rendimiento: En caso de encontrar que el simulador funciona de una manera lenta, encontrar la falla y tratar de actualizar dicha falla para solucionarlo.
    - Actualización de contenidos: Arreglar errores tipográficos, actualizar información desactualizada y corregir cualquier inconsistencia en el contenido. Si se ha perdido contenido debido a un error, restaurarlo desde las copias de seguridad.
    - Compatibilidad con Navegadores y Dispositivos: verificar que el sitio funcione correctamente en todos los navegadores y dispositivos. Solucionar los problemas de compatibilidad específicos.
    - Respuesta a Feedback de Usuarios: Responder y solucionar a los problemas reportados por los usuarios lo antes posible para mejorar su experiencia.
    - Documentación: Documentar cualquier cambio realizado durante el mantenimiento correctivo para futuras referencias. Llevar un registro detallado de los problemas y las soluciones implementadas para identificar patrones y mejorar la prevención.