SISTEMAS OPERATIVOS 2021-1



SANTIAGO ARISTIZABAL CORREA

MANUAL DE USUARIO

CH-MAQUINA

**REQUISITOS TECNICOS**

## **Requerimientos mínimos de Hardware**

* Dado que la aplicación se desarrolló como plataforma web, el único requisito mínimo en cuanto a Hardware es una conexión.

# **DEFINICIÓN DE HERRAMIENTAS USADAS**

La implementación del proyecto CH-MAQUINA fue realizada en el lenguaje de programación JavaScript, además de usar elementos de HTML y CSS.

A continuación, se especifican los elementos usados en el desarrollo de la aplicación.

## **Editores de código**

**Visual Studio Code:** es un editor de código fuente. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado.

1. **Lenguajes de programación:**

**HTML:** Conjunto de etiquetas que permiten generar la interfaz para que el usuario pueda usar la aplicación.

**JavaScript:** Lenguaje de programación usado de lado del cliente para la validación de valores de entrada por parte del usuario por medio de HTML.

**CSS:** Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML y derivados.

1. **Aspectos del programa**

El programa debe simular un procesador muy elemental y una memoria principal a

través de un vector de hasta 9999 posiciones, las cuales pueden ser variadas al

momento de iniciar el programa, se asume por defecto que el chcomputador

empieza con 100 posiciones de memoria para facilitar el proceso de pruebas. El

programa debe estar en capacidad de leer un conjunto de programas en un seudo

lenguaje de máquina que llamaremos CHMAQUINA y los cargará en las

posiciones disponibles de la citada memoria, leerá una instrucción por cada

línea de entrada.

Las primeras posiciones de la memoria estarán reservadas para el núcleo del

sistema operativo (kernel), el tamaño de este deberá poderse ingresar al iniciar la

corrida del simulador, su valor por defecto es 10\*z + 9 posiciones (donde z será el

último digito de la cedula del estudiante).

El programa deberá realizar un chequeo de Sintaxis, produciendo un listado de

errores si los hay, de lo contrario procederá a la carga definitiva del programa en

memoria y quedará listo para ejecución del mismo bajo las reglas de corrida de

múltiples programas

1. **Estructura del proyecto**



El proyecto en su estructura tiene diferentes carpetas las cuales almacenas lo diferentes tipos de archivos y códigos realizados en el desarrollo del proyecto.

1. **Paquete CHmaquina app.js**

En este punto se dará una explicación concisa de algunas partes del código encargadas de ciertas funcionalidades.

let memoria = [];

let btnMemoria = document.getElementById('memoria-btn');

let kernel = document.getElementById('kernel');

let memoriaInput = document.getElementById('memoria-input');

let btnEncender = document.getElementById('encender');

let btnApagar = document.getElementById('apagar');

let modo = document.getElementById('modo');

var file = document.getElementById('files');

var ejecutar = document.getElementById('correrPrograma')//nuevo

let divMemoria = document.getElementById('memoria');

let cerrar = document.getElementById('cerrar');

let inputAcumulador= document.getElementById('acumulador');

var lFinal = [];

let monitor= document.getElementById('monitor-result');//diferente

let impresora= document.getElementById('impresora-result');//diferente

let infoFooter= document.getElementById('footer-home');

let contadorPasoPaso = 0;

var arrayEtiquetas = [];

var arrayVariables = [];

var instrucciones = [];

// variables del contenedor footer

let idColumn= document.getElementById('idColumn');

let programa= document.getElementById('programa');

let ins= document.getElementById('ins');

let rb= document.getElementById('rb');

let rlc= document.getElementById('rlc');

let rlp= document.getElementById('rlp');

//  array de Id

let listId= [];

let numId= 0;

//  array de programa

let listPrograma=[];

let listIns=[];

let listRb=[];

let listRlc=[];

let listRlp=[];

En un inicio se puede encontrar la definicon de las variables que seran utilizadas en la ejecución del sistema con el comando LET, encargado de asignar los valores a estas. Tambien se encuentran declarados los vectores utilizados en el mismo.

**FUNCION ENCENDER()**

function encender() {

    if (Number(memoriaInput.value) > 9999) {

        alert('EL ESPACIO DE MEMORIA HA EXCEDIDO');

        apagar();

    }

    kernel.disabled=true;

    memoriaInput.disabled=true;

    btnEncender.style.display = 'none';

    btnApagar.style.display = 'inline-block';

    modo.innerHTML = 'Modo usuario';

    cerrar.style.display='block';

    files.disabled = false;

    let memoriaMostrar = [];

    let lAcumulador = [0,' Acumulador']

    memoriaMostrar.push(lAcumulador.toString().replaceAll(',',' '));

    // console.log(memoriaMostrar);

    for(let i = 1; i<Number(memoriaInput.value)+1; i++) {

        if(i <= Number(kernel.value)) {

            let so = [i];

            so.push('CHSO\_V2021');

            memoriaMostrar.push(so.toString().replaceAll(',',' '));

        } else{

            memoriaMostrar.push(`${i} - - - - `)

        }

    }

    // console.log(memoriaMostrar);

    document.getElementById('memoria').style.display = 'block';

    document.getElementById('memoria').innerHTML = memoriaMostrar.join('<br></br>');

}

Esta función se encarga de hacer el arranque de la interfaz, aplica la inicialización del mapa de memoria y desactivación de botones de carga de archivo, encendido de máquina y los medidores de memoria y kernel. Permite garantizar que antes de ejecutar los programas, se pueda demostrar que el valor que hay en memoria, no exceda de 9999 posiciones. Si se cumple con esta condición se crea la lista, con el acumulador, el espacio de memoria (kernel) y las posiciones de memoria restantes. Permitiendo dar entrada a la selección de archivos.

**FUNCIÓN APAGAR ()**

function apagar() {

    location.reload()

    kernel.disabled = false;

    memoriaInput.disabled = false;

    btnApagar.style.display = 'none';

    btnEncender.style.display = 'inline-block';

    modo.innerHTML = 'Modo kernel';

    files.disabled = true;

    btnMemoria.disabled = false;

}

Esta funcion por el contrario de la función encender, activa nuevamente los medidores Kernel y de Memoria permitiendo su modificación, y a su vez cambia el modo del sistema a modo kernel.**FUNCION LEERARCHIVO()**

function leerArchivo(evento) {

    //console.log(evento.target.files);

    for(let i=0; i<evento.target.files.length; i++) {

    document.getElementById('instrucciones').innerHTML = "";

    document.getElementById('variables').innerHTML = "";

    document.getElementById('etiquetas').innerHTML = "";

    let archivo = evento.target.files[i];

    let name= evento.target.files[i].name;

    procesarArchivo(archivo, function(result) {

    let lArchivo = [];

    lArchivo = result.split('\n');

    for(let i=0; i<lArchivo.length; i++) {

        if(lArchivo[i] == "") {

            lArchivo.splice(i, 1);

            i--;

        }

    }

    for(let i=0; i<lArchivo.length; i++) {

        if(lArchivo[i].length == 1) {

            lArchivo.splice(i, 1);

            i--;

        }

        if(lArchivo[i].includes('//')) {

            lArchivo.splice(i,1);

            i--;

        }

    }

    let listaPrueba = [];

    for(instruccion of lArchivo) {

        instruccion=instruccion.trim()

        listaPrueba.push(instruccion.split(' '))

    }

    for(l of lArchivo) {

        for(let j = 0; j< l.length; j++) {

            if(l[j] == '') {

                l.splice(j,1);

                j--

            }else{

                l[j] = l[j].trim();

            }

        }

    }

    for(l of listaPrueba) {

      for(let j = 0; j< l.length; j++) {

          if(l[j] == '') {

              l.splice(j,1);

              j--;

          } else {

              l[j] = l[j].trim();

          }

      }

  }

    let bool = verificarSintaxis(listaPrueba);

    let sum = +kernel.value + +listaPrueba.length;

    // console.log(sum);

    console.log(sum <= Number(memoriaInput.value));

    if(bool.length === 0 && sum <= Number(memoriaInput.value)) {

        // console.log(listaPrueba);

          console.log(listaPrueba);

          let cf = 0;

          for(l of listaPrueba) {

            if(l[0] == 'nueva') {

              let valor = [];

              if(l[2].toUpperCase()== 'C') {

                  for(let i = 4; i<l.length; i++) {

                      valor.push(l[i]);

                  }

                  valor = valor.toString();

                  valor = valor.replaceAll(',', " ");

                  let total = Number(listaPrueba.length)+Number(kernel.value)+1;

                  arrayVariables.push(new Nueva(total, l[1], l[2], valor));

                  listaPrueba.push([total, arrayVariables[cf].nombre, arrayVariables[cf].valor]);

                  cf++;

              } else {

                let total = Number(listaPrueba.length)+Number(kernel.value)+1;

                arrayVariables.push(new Nueva(total, l[1], l[2], l[3]));

                listaPrueba.push([total, arrayVariables[cf].nombre, arrayVariables[cf].valor]);

                cf++;

              }

            }

          }

          console.log(listaPrueba);

        for(instruccion of listaPrueba) {

            lFinal.push(instruccion)

        }

        let cont = Number(kernel.value) + 1;

        for(recorrer of lFinal) {

            if(recorrer[0] == cont) {

                recorrer.shift();

            }

            cont++

        }

        let w = Number(kernel.value)+1;

        for(k of lFinal) {

            k.unshift(w);

            w++;

        }

        let contador = 0;

        let lAcumulador= [0, 'Acumulador']

        let arrayMemoria= [];

        let mostrarOperaciones = [];

        let contenemos;

        let suma = +kernel.value + +lFinal.length;

        console.log(suma);

        arrayMemoria.push(lAcumulador);

        for(let s=1; s<=Number(memoriaInput.value); s++) {

            if(s<=kernel.value) {

                arrayMemoria.push(`${s} CHSO\_V2021`);

            } else if(s>kernel.value && s<=suma){

                arrayMemoria.push(lFinal[contador]);

                contenemos = lFinal[contador].toString().replaceAll(',',' ')

                mostrarOperaciones.push(contenemos);

                contador++;

            } else {

                arrayMemoria.push(`${s} - - - - - - `)

            }

        }

        console.log(mostrarOperaciones);

        // console.log(arrayMemoria);

        // document.getElementById('memoria').innerHTML = arrayMemoria.join('<br></br>');

        instrucciones = arrayMemoria.slice(+kernel.value+1, +suma+1);

        //Mostrar operaciones en el contenedor sin comas

        let listaVariables = [];

        let listaEtiquetas = []; //Lista para variables y etiquetas para Mostrar en el div

        for(linea of instrucciones) {

            if(linea[1] == 'nueva') {

                listaVariables.push(linea)

            } else if(linea[1] == 'etiqueta') {

            let bandera;

            if(linea[3] > instrucciones.length) {

                bandera = true;

            } else {

                bandera = false;

            }

            arrayEtiquetas.push(new Etiqueta(linea[0], linea[2], linea[3], bandera));

            listaEtiquetas.push(linea);

            }

        }

        // Seguimiento de variables en contenedor MEMORIA

          //ARRAY DE INTERFAZ

          // Agrega en el footer las lineas

            // id

          listId.push(zeroFill(numId,3))

          numId++;

            //PRGRAMA

          listPrograma.push(name)

            // INs

          listIns.push(listaPrueba.length)

            //RB

          listRb.push(listaPrueba[0][0])

            //RLC

          listRlc.push(listaPrueba[listaPrueba.length-1][0])

            //RLP

          let rlpValor= listaPrueba[listaPrueba.length-1][0];

          listRlp.push(Number(rlpValor) + Number(arrayVariables.length))

          // agrega en el interfaz

          idColumn.innerHTML= listId.join('</br>');

          programa.innerHTML= listPrograma.join('</br>');

          ins.innerHTML= listIns.join('</br>');

          rb.innerHTML= listRb.join('</br>');

          rlc.innerHTML= listRlc.join('</br>');

          rlp.innerHTML= listRlp.join('</br>');

        let sinEspacios= [];

        let contar;

        for(m of arrayMemoria) {

          contar = m.toString().replaceAll(',', ' ');

          sinEspacios.push(contar);

        }

        console.log(sinEspacios);

        document.getElementById('memoria').innerHTML = sinEspacios.join('<br></br>');

        document.getElementById('instrucciones').innerHTML= mostrarOperaciones.join('<br></br>');

        for(l of listaVariables) {

          let va = document.createElement('span');

          va.append(`${l[0]} ${l[2]}`);

          document.getElementById('variables').appendChild(va);

        }

        for(l of listaEtiquetas) {

          let eti = document.createElement('span');

          eti.append(`${l[0]} ${l[2]}`);

          document.getElementById('etiquetas').appendChild(eti);

        }

        ejecutar.style.display = 'inline-block';

      }else {

          if(sum > Number(memoriaInput.value)){

            alert(`Error: se excede el espacio de memoria`)

            document.getElementById('instrucciones').innerHTML = 'Error de sintaxis';

          }

          else{

              for (let err = 0; err < bool.length; err++) {

                alert(bool[err]);

              }

              location.reload()

          }

        }

      })

    }

  }

Esta clase es la que se encarga de abrir los programas .ch, para luego leer línea tras línea de estos agregándolos así en un arreglo de tipo string, es decir por cada posición del arreglo estarán una a una las líneas del programa correspondiente.

**FUNCIÓN PROCESARARCHIVO()**

function procesarArchivo(ch, callback) {

    var reader = new FileReader();

    reader.readAsText(ch);

    reader.onload = function () {

        callback(reader.result);

    }

}

Esta función continua con el debido proceso de cada uno de los archivos con sintaxis del ch-código a la memoria, los cuales se cargarán a la memoria dejando el espacio libre de los que se reservaron para el sistema operativo, garantizando que no se exceda el espacio de memoria.

**FUNCIÓN VERIFICARSINTAXIS()**

function verificarSintaxis(lista) {

        let errores = [];

        //Listas para almacenar los nombres y valores de las variables

        let valoresVariables = [0];

        let nombreVariables = ['acumulador'];

        //Etiquetas

        direccionEtiqueta = [];

        nombreEtiqueta = [];

        for(let instruccion=0; instruccion < lista.length; instruccion++) {

          let linea = "";

          for (let index = 0; index < lista[instruccion].length; index++) {

            linea += " " + lista[instruccion][index];

          }

          if(lista[instruccion][0].toLowerCase() == 'nueva') {

            if (lista[instruccion].length > 4) {

              errores.push("Error de sintaxis, más de 4 operadores especificados: " + linea);

            }else if (lista[instruccion].length < 3) {

              errores.push("Error de sintaxis, menos de 3 operadores especificados: " + linea);

            }

            switch (lista[instruccion][2]) {

              case "C":

                if (lista[instruccion].length == 3) {

                  valoresVariables.push("");

                }

                break;

              case "I":

                if (lista[instruccion].length > 3) {

                  let num = lista[instruccion][3];

                  let verificList = [1,2,3,4,5,6,7,8,9,0]

                  for (let i = 0; i < lista[instruccion].length; i++) {

                    if (!(verificList.includes(Number(num)))){

                        errores.push("Error de sintaxis, el tipo de dato no es un entero: " + linea);

                        break;

                    }

                  }

                }else{

                  valoresVariables.push("0");

                }

                break;

              case "R":

                if (lista[instruccion].length > 3) {

                  let num = lista[instruccion][3];

                  let verificList = ["1","2","3","4","5","6","7","8","9","0","."]

                  for (let i = 0; i < lista[instruccion].length; i++) {

                    if (!((verificList.includes(num)))){

                        errores.push("Error de sintaxis, el tipo de dato no es un Real/Decimal: " + linea);

                        break;

                    }

                  }

                }else{

                  valoresVariables.push("0");

                }

                break;

              case "L":

                if (lista[instruccion].length > 3) {

                  if (!(lista[instruccion][3]==("0")) && !(lista[instruccion][3]==("1"))) {

                    errores.push("Error de sintaxis, el tipo de dato no es un Boolean: " + linea);

                    break;

                  }

                } else {

                  valoresVariables.push("0");

                }

                break;

              default:

                errores.push("Error de sintaxis, no se reconoce el tipo de variable: " + linea);

            }

            nombreVariables.push(lista[instruccion][1]);

            if (lista[instruccion].length == 4) {

              valoresVariables.push(lista[instruccion][3]);

            }

          }else if(lista[instruccion][0].toLowerCase() =='lea') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='cargue') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada: " + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='almacene') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='vaya') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='vayasi') {

            if (lista[instruccion].length > 3) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 3) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='etiqueta') {

            if (lista[instruccion].length > 3) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 3) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            let num = lista[instruccion][2];

            let verificList = "1234567890";

            for (let i = 0; i < num.length; i++) {

              if (!(verificList.includes(num[i]))){

                errores.push("Error de sintaxis, el tipo de dato no es un número: " + linea);

              }

            }

            direccionEtiqueta.push(instruccion[2]);

            nombreEtiqueta.push(lista[instruccion][1]);

          }else if(lista[instruccion][0].toLowerCase() =='sume') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada  :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='reste') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() == 'multiplique') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='divida') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada  :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() == 'potencia') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() == 'modulo') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='concatene') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada :" + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='elimine') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

                errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

                errores.push("La variable " + lista[instruccion][1] + " no ha sido creada: " + linea);

            }

          }else if(lista[instruccion][0] =='Y') {

            if (lista[instruccion].length > 4) {

              errores.push("Error de sintaxis, más de 4 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 4) {

              errores.push("Error de sintaxis, menos de 4 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada: " + linea);

            }else{

              if (!(nombreVariables.includes(lista[instruccion][2]))) {

                errores.push("La variable " + lista[instruccion][2] + " no ha sido creada: " + linea);

              }else{

                if (!(nombreVariables.includes(lista[instruccion][3]))) {

                  errores.push("La variable " + lista[instruccion][3] + " no ha sido creada: " + linea);

                }

              }

            }

          }else if(lista[instruccion][0] == 'O') {

            if (lista[instruccion].length > 4) {

              errores.push("Error de sintaxis, más de 4 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 4) {

              errores.push("Error de sintaxis, menos de 4 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada:" + linea);

            }else{

              if (!(nombreVariables.includes(lista[instruccion][2]))) {

                errores.push("La variable " + lista[instruccion][2] + " no ha sido creada: " + linea);

              }else{

                if (!(nombreVariables.includes(lista[instruccion][3]))) {

                  errores.push("La variable " + lista[instruccion][3] + " no ha sido creada: " + linea);

                }

              }

            }

          }else if(lista[instruccion][0].toLowerCase() =='NO') {

            if (lista[instruccion].length > 3) {

              errores.push("Error de sintaxis, más de 3 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 3) {

              errores.push("Error de sintaxis, menos de 4 operadores especificados: " + linea);

            }

            if (!( lista[instruccion].includes(lista[instruccion][1]))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada: " + linea);

            }else{

              if (!( lista[instruccion].includes(lista[instruccion][2]))) {

                errores.push("La variable " + lista[instruccion][2] + " no ha sido creada: " + linea);

              }

            }

          }else if(lista[instruccion][0].toLowerCase() == 'muestre') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1])) && !(lista[instruccion][1]==("acumulador")) ) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada: " + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() == 'imprima') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 2) {

              errores.push("Error de sintaxis, menos de 2 operadores especificados: " + linea);

            }

            if (!(nombreVariables.includes(lista[instruccion][1])) && !(lista[instruccion][1]==("acumulador"))) {

              errores.push("La variable " + lista[instruccion][1] + " no ha sido creada: " + linea);

            }

          }else if(lista[instruccion][0].toLowerCase() =='retorne') {

            if (lista[instruccion].length > 2) {

              errores.push("Error de sintaxis, más de 2 operadores especificados: " + linea);

            }

            if (lista[instruccion].length < 1) {

              errores.push("Error de sintaxis, menos de 1 operadores especificados: " + linea);

            }

          } else {

            errores.push("No se reconoce la intrucción: " + linea);

          }

        }

        return errores;

      }

Esta función se encarga de cargar el archivo .ch y comenzar a analizar línea por línea de este y verificar su sintaxis y si es correcta lo carga en la tabla de memoria clasifica las variables y las etiquetas y las carga a las tablas correspondientes, si todo sale bien carga el dato del programa a la tabla de procesos.

**FUNCIÓN MOMEMORIA()**

function moMemoria() {

    divMemoria.style.display = 'block';

    cerrar.style.display = 'inline-block';

}

Esta función se encarga de administrar los Sectores de memoria que se ejecutaran durante el funcionamiento del programa, a su vez, permite Mostrar la posición del sector y su contenido al momento de ejecutar el archivo Ingresado.

**FUNCIÓN CERRARDIV()**

function cerrarDiv() {

    divMemoria.style.display = 'none';

    cerrar.style.display = 'none'

}

Esta función utiliza el evento onclick(), es decir, que cuando se ejecuta un click sobre este, procede esconder el contenedor de memoria, y se esconde así mismo.

**FUNCIÓN CORRERARCHIVO()**

function correrArchivo(lista,inicio,fin, arrayEtiquetas, arrayVariables, acumulador){

    console.log('entra al correr archivo');

    console.log(lista);

    for(let instruccion = inicio; instruccion<fin; instruccion++){

        if(lista[instruccion][1].toLowerCase() =='lea') {//Preguntar al Profe

            for(nombreVar of arrayVariables){

                if(nombreVar.nombre == lista[instruccion][2].trim()){

                    let valorNew =prompt(`Ingrese el valor de la variable ${lista[instruccion][2]}`);

                    nombreVar.valor= valorNew;

                }

            }

            console.log('lea');

        }else if(lista[instruccion][1].toLowerCase() =='cargue') {

          console.log('entra al cargue');

            for(cargar of arrayVariables){

                if(lista[instruccion][2] == cargar.nombre){

                    acumulador=cargar.valor;

                    console.log(acumulador);

                }

            }

            inputAcumulador.value = acumulador;

            console.log(acumulador);

        }else if(lista[instruccion][1].toLowerCase() =='almacene') {

            for(almacene of arrayVariables){

                if(lista[instruccion][2] == almacene.nombre){

                    almacene.valor = acumulador;

                }

            }

            inputAcumulador.value=acumulador;

        }else if(lista[instruccion][1].toLowerCase() =='vaya') { //FALTA POR HACER-- PREGUNTAR AL PROFE

            for(etiquetas of arrayEtiquetas) {

                if(etiquetas.nombre== lista[instruccion][2]) {

                    if(etiquetas.sobrepasa == false) {

                        correrArchivo(lista, etiquetas.valor, lista[instruccion-1], arrayEtiquetas, arrayVariables, acumulador);

                    }

                }

            }

            console.log('Me salí');

        }else if(lista[instruccion][1].toLowerCase() =='vayasi') {

          if(acumulador < 0) {

            for(e of arrayEtiquetas) {

              if(lista[instruccion][2].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa == false) {

                correrArchivo(lista, e.valor, lista.length, arrayEtiquetas, arrayVariables, acumulador);

              } else {

              }

            }

          } else if(acumulador > 0) {

            for(e of arrayEtiquetas) {

              if(lista[instruccion][3].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa === false) {

                correrArchivo(lista, e.valor, lista.length, arrayEtiquetas, arrayVariables, acumulador)

              } else {

              }

            }

          }

          console.log('Me salí x2');

          console.log(inicio);

          console.log(fin);

          console.log(instruccion);

        }else if(lista[instruccion][1].toLowerCase() =='sume') {

            for(sume of arrayVariables){

                if(lista[instruccion][2] == sume.nombre){

                    acumulador+= sume.valor;

                }

                inputAcumulador.value =acumulador;

            }

        }else if(lista[instruccion][1].toLowerCase() =='reste') {

            for(reste of arrayVariables){

                if(lista[instruccion][2] == reste.nombre){

                    acumulador-= reste.valor;

                }else if(lista[instruccion][2] == 'acumulador'){

                  acumulador = 0

                }

                inputAcumulador.value = acumulador;

            }

        }else if(lista[instruccion][1].toLowerCase() =='multiplique') {

            for(multi of arrayVariables){

                if(lista[instruccion][2] == multi.nombre){

                    acumulador= acumulador \* multi.valor;

                }

            }

        }else if(lista[instruccion][1].toLowerCase() =='divida') {

            for(div of arrayVariables){

                if(lista[instruccion][2] == div.nombre && div.valor != 0){

                    acumulador= acumulador / div.valor;

                }

            }

        }else if(lista[instruccion][1].toLowerCase() =='potencia') {

            for(potencia of arrayVariables){

                if(lista[instruccion][2] == potencia.nombre && potencia.valor.isInteger()){

                    acumulador= acumulador \*\* potencia.valor;

                }

            }

        }else if(lista[instruccion][1].toLowerCase() =='modulo') {

            for(mod of arrayVariables){

                if(lista[instruccion][2] == mod.nombre){

                    let modulo= acumulador % mod.valor;

                    alert(`El modulo de ${acumulador} % ${mod.valor} = ${modulo}(linea ${mod.id})`);

                }

            }

        }else if(lista[instruccion][1].toLowerCase() =='concatene') {

            for(concatene of arrayVariables){

                if(lista[instruccion][2].trim() == concatene.nombre){

                    let cad= acumulador +' '+ concatene.valor;

                    inputAcumulador.type= 'text';

                    acumulador = cad;

                    inputAcumulador.value = acumulador;

                }

            }

        }else if(lista[instruccion][1].toLowerCase() =='elimine') {

            console.log(acumulador);

            let lol = lista[instruccion][2].trim()

            acumulador=acumulador.replaceAll(lol,'')

            inputAcumulador.value=acumulador;

        }else if(lista[instruccion][1].trim() =='Y') {

            let primerOperando = 0;

            let segundoOperando = 0;

            for(v of arrayVariables) {

                if(lista[instruccion][2].trim() == v.nombre) {

                    console.log(v.valor);

                    primerOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`)

                }

                if (lista[instruccion][3].trim() == v.nombre) {

                    console.log(v.valor);

                    segundoOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`)

                }

                if(lista[instruccion][4].trim() == v.nombre) {

                    console.log(v.nombre);

                    if(primerOperando && segundoOperando == 1) {

                        v.valor = 1;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    } else if (primerOperando && segundoOperando == 0) {

                        v.valor = 0;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }

                }

            }

        }else if(lista[instruccion][1].trim() =='O') {

            let primerOperando = 0;

            let segundoOperando = 0;

            for(v of arrayVariables) {

                if(lista[instruccion][2].trim() == v.nombre) {

                    console.log(v.valor);

                    primerOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`);

                }

                if (lista[instruccion][3].trim() == v.nombre) {

                    console.log(v.valor);

                    segundoOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`)

                }

                if(lista[instruccion][4].trim() == v.nombre) {

                    console.log(v.nombre);

                    if(primerOperando || segundoOperando == 1) {

                        v.valor = 1;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }else if (primerOperando || segundoOperando == 0) {

                        v.valor = 0;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }

                }

            }

        }else if(lista[instruccion][1].trim() =='NO') {

            console.log('Entró a NO');

        }else if(lista[instruccion][1].toLowerCase() =='muestre') {

            if(lista[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                monitor.innerHTML= `El resultado de (${lista[instruccion][2]}) es: ${acumulador}`;

            }

            else{

                for(muestre of arrayVariables ){

                    if(lista[instruccion][2].trim() == muestre.nombre){

                            console.log(muestre.valor);

                            monitor.innerHTML= `El resultado de (${lista[instruccion][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

        }else if(lista[instruccion][1].toLowerCase() =='imprima') {

            if(lista[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                impresora.innerHTML= `El resultado de (${lista[instruccion][2]}) es: ${acumulador}`;

            }

            else{

                for(muestre of arrayVariables ){

                    if(lista[instruccion][2].trim() == muestre.nombre){

                            console.log(muestre.valor);

                            impresora.innerHTML= `El resultado de (${lista[instruccion][2]})es igual a ------> ${muestre.valor}`;

                    }

                }

            }

        }else if(lista[instruccion][1].toLowerCase() == 'extraiga'){

            longitud= acumulador.length;

            console.log(`la longitud del acumulador es ${longitud}`);

            let extraer = [];

            for(i=0; i<Number(lista[instruccion][2]); i++) {

                extraer.push(acumulador[i]);

            }

            acumulador = extraer.join('');

            inputAcumulador.value=acumulador;

            /\* acumulador = acumulador.slice(Number(instruccion[2]));

            console.log(acumulador); \*/

            // inputAcumulador.value = acumulador;

            // acumulador=acumulador.slice(Number(lista[instruccion][2]));

            // console.log(acumulador);

            //

        }else if(lista[instruccion][1].toLowerCase() =='retorne') {

            acumulador= 0;

            alert('PROGRAMA TERMINADO')

        }else if(lista[instruccion][1].toLowerCase() =='raiz') { /\* toma la raiz cuadrada del código \*/

            for(raiz of arrayVariables){

                if(lista[instruccion][2].trim() == raiz.nombre){

                    let ra= raiz.valor;

                    let resultadoRaiz= Math.sqrt(ra);

                }

            }

            console.log(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

            alert(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

        }

    }

}

Esta función es una de las principales, ya que será la encargada de albergar las funciones con las que trabajará el sistema, por ejemplo Multiplicar, Restar, Factorial, etc.

**FUNCIÓN EJECUTARPASOAPASO()**

function ejecutarPasoAPaso(){

  stepByStep(instrucciones,contadorPasoPaso,arrayEtiquetas,arrayVariables);

};

Este método invoca la función STEPBYSTEP creada para realizar el proceso de ejecución paso a paso respectivamente.

**FUNCION STEPBYSTEP()**

function stepByStep(lista, contadorPasoAPaso, arrayEtiquetas, arrayVariables){

    if(lista[contadorPasoAPaso][1].toLowerCase() == 'nueva') {

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

    }else if(lista[contadorPasoAPaso][1].toLowerCase() == 'etiqueta') {

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

    }

    else if(lista[contadorPasoAPaso][1].toLowerCase() =='lea') {//Preguntar al Profe

      for(nombreVar of arrayVariables){

          if(nombreVar.nombre == lista[contadorPasoAPaso][2].trim()){

              let valorNew =prompt(`Ingrese el valor de la variable ${lista[contadorPasoAPaso][2]}`);

              nombreVar.valor= valorNew;

          }

      }

      console.log('lea');

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='cargue') {

    console.log('entra al cargue');

      for(cargar of arrayVariables){

          if(lista[contadorPasoAPaso][2] == cargar.nombre){

              acumulador=cargar.valor;

              console.log(acumulador);

          }

      }

      inputAcumulador.value = acumulador;

      console.log(acumulador);

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='almacene') {

      for(almacene of arrayVariables){

          if(lista[contadorPasoAPaso][2] == almacene.nombre){

              almacene.valor = acumulador;

          }

      }

      inputAcumulador.value=acumulador;

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].trim().toLowerCase() === 'vaya') {

    for(etiquetas of arrayEtiquetas) {

        if(etiquetas.nombre== lista[contadorPasoAPaso][2].trim()) {

            if(etiquetas.sobrepasa == false) {

              console.log(etiquetas.valor);

              let newPaso = etiquetas.valor;

              stepByStep(lista, newPaso,arrayEtiquetas,arrayVariables);

            }

        }

    }

    console.log('Me salí');

    let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].trim().toLowerCase() =='vayasi') {

      if(acumulador < 0) {

        for(e of arrayEtiquetas) {

          if(lista[contadorPasoAPaso][2].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa == false) {

            if(e.valor < lista.length && e.valor >= 0) {

              console.log(e.valor);

              let newPaso = e.valor;

              stepByStep(lista, newPaso, arrayEtiquetas, arrayVariables);

            } else {

              alert(`El valor de ${e.nombre} es mayor a la cantidad de instrucciones del programa`);

            }

          } else {

          }

        }

      } else if(acumulador > 0) {

        for(e of arrayEtiquetas) {

          if(lista[contadorPasoAPaso][3].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa === false) {

            if(e.valor < lista.length && e.valor >= 0) {

              console.log(e.valor);

              let newPaso = e.valor;

              stepByStep(lista, newPaso, arrayEtiquetas, arrayVariables);

            } else {

              alert(`El valor de ${e.nombre} es mayor a la cantidad de instrucciones del programa`);

            }

          } else {

          }

        }

      }

      console.log('Me salí x2');

      console.log(contadorPasoAPaso);

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='sume') {

      for(sume of arrayVariables){

          if(lista[contadorPasoAPaso][2] == sume.nombre){

              acumulador+= sume.valor;

          }

          inputAcumulador.value =acumulador;

      }

      confirm(`el contador va en: ${contadorPasoAPaso}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='reste') {

      for(reste of arrayVariables){

          if(lista[contadorPasoAPaso][2] == reste.nombre){

              acumulador-= reste.valor;

          }else if(lista[contadorPasoAPaso][2] == 'acumulador'){

            acumulador = 0

          }

          inputAcumulador.value = acumulador;

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='multiplique') {

      for(multi of arrayVariables){

          if(lista[contadorPasoAPaso][2] == multi.nombre){

              acumulador= acumulador \* multi.valor;

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='divida') {

      for(div of arrayVariables){

          if(lista[contadorPasoAPaso][2] == div.nombre && div.valor != 0){

              acumulador= acumulador / div.valor;

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='potencia') {

      for(potencia of arrayVariables){

          if(lista[contadorPasoAPaso][2] == potencia.nombre && potencia.valor.isInteger()){

              acumulador= acumulador \*\* potencia.valor;

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='modulo') {

      for(mod of arrayVariables){

          if(lista[contadorPasoAPaso][2] == mod.nombre){

              let modulo= acumulador % mod.valor;

              alert(`El modulo de ${acumulador} % ${mod.valor} = ${modulo}(linea ${mod.id})`);

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='concatene') {

      for(concatene of arrayVariables){

          if(lista[contadorPasoAPaso][2].trim() == concatene.nombre){

              let cad= acumulador +' '+ concatene.valor;

              inputAcumulador.type= 'text';

              acumulador = cad;

              inputAcumulador.value = acumulador;

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='elimine') {

      console.log(acumulador);

      let lol = lista[contadorPasoAPaso][2].trim()

      acumulador=acumulador.replaceAll(lol,'')

      inputAcumulador.value=acumulador;

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].trim() =='Y') {

      let primerOperando = 0;

      let segundoOperando = 0;

      for(v of arrayVariables) {

          if(lista[contadorPasoAPaso][2].trim() == v.nombre) {

              console.log(v.valor);

              primerOperando = v.valor;

              console.log(`El valor de ${v.nombre} = ${v.valor}`)

          }

          if (lista[contadorPasoAPaso][3].trim() == v.nombre) {

              console.log(v.valor);

              segundoOperando = v.valor;

              console.log(`El valor de ${v.nombre} = ${v.valor}`)

          }

          if(lista[contadorPasoAPaso][4].trim() == v.nombre) {

              console.log(v.nombre);

              if(primerOperando && segundoOperando == 1) {

                  v.valor = 1;

                  console.log(`El valor de ${v.nombre} = ${v.valor}`)

              } else if (primerOperando && segundoOperando == 0) {

                  v.valor = 0;

                  console.log(`El valor de ${v.nombre} = ${v.valor}`)

              }

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].trim() =='O') {

      let primerOperando = 0;

      let segundoOperando = 0;

      for(v of arrayVariables) {

          if(lista[contadorPasoAPaso][2].trim() == v.nombre) {

              console.log(v.valor);

              primerOperando = v.valor;

              console.log(`El valor de ${v.nombre} = ${v.valor}`);

          }

          if (lista[contadorPasoAPaso][3].trim() == v.nombre) {

              console.log(v.valor);

              segundoOperando = v.valor;

              console.log(`El valor de ${v.nombre} = ${v.valor}`)

          }

          if(lista[contadorPasoAPaso][4].trim() == v.nombre) {

              console.log(v.nombre);

              if(primerOperando || segundoOperando == 1) {

                  v.valor = 1;

                  console.log(`El valor de ${v.nombre} = ${v.valor}`)

              }else if (primerOperando || segundoOperando == 0) {

                  v.valor = 0;

                  console.log(`El valor de ${v.nombre} = ${v.valor}`)

              }

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].trim() =='NO') {

      console.log('Entró a NO');

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='muestre') {

      if(lista[contadorPasoAPaso][2].toUpperCase().trim() == 'ACUMULADOR'){

          monitor.innerHTML= `El resultado de (${lista[contadorPasoAPaso][2]}) es: ${acumulador}`;

      }

      else{

          for(muestre of arrayVariables ){

              if(lista[contadorPasoAPaso][2].trim() == muestre.nombre){

                      console.log(muestre.valor);

                      monitor.innerHTML= `El resultado de (${lista[contadorPasoAPaso][2]}) es igual a --> ${muestre.valor}`;

              }

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='imprima') {

      if(lista[contadorPasoAPaso][2].toUpperCase().trim() == 'ACUMULADOR'){

          impresora.innerHTML= `El resultado de (${lista[contadorPasoAPaso][2]}) es: ${acumulador}`;

      }

      else{

          for(muestre of arrayVariables ){

              if(lista[contadorPasoAPaso][2].trim() == muestre.nombre){

                      console.log(muestre.valor);

                      impresora.innerHTML= `El resultado de (${lista[contadorPasoAPaso][2]})es igual a ------> ${muestre.valor}`;

              }

          }

      }

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() == 'extraiga'){

      longitud= acumulador.length;

      console.log(`la longitud del acumulador es ${longitud}`);

      let extraer = [];

      for(i=0; i<Number(lista[contadorPasoAPaso][2]); i++) {

          extraer.push(acumulador[i]);

      }

      acumulador = extraer.join('');

      inputAcumulador.value=acumulador;

      /\* acumulador = acumulador.slice(Number(contadorPasoAPaso[2]));

      console.log(acumulador); \*/

      // inputAcumulador.value = acumulador;

      // acumulador=acumulador.slice(Number(lista[contadorPasoAPaso][2]));

      // console.log(acumulador);

      //

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='retorne') {

      acumulador= 0;

  }else if(lista[contadorPasoAPaso][1].toLowerCase() =='raiz') { /\* toma la raiz cuadrada del código \*/

      for(raiz of arrayVariables){

          if(lista[contadorPasoAPaso][2].trim() == raiz.nombre){

              let ra= raiz.valor;

              let resultadoRaiz= Math.sqrt(ra);

          }

      }

      console.log(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

      alert(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

      let unSpace= lista[contadorPasoAPaso].toString().replaceAll(',',' ')

      confirm(`La instruccion es ${unSpace}`)

  }

  contadorPasoPaso++;

}

Esta función es la encargada de recorrer el mapa de memoria y según la instrucción invoca otra función y le envía parámetros para que los opere y entregue un resultado, por cada instrucción le preguntará al usuario si desea seguir viendo la siguiente, si el usuario indica que no, finaliza el proceso.

**CLASS NUEVA.JS**

Class Nueva {

    constructor(id, nombre, tipo, valor) {

        this.\_id = id;

        this.\_nombre = nombre;

        this.\_tipo = tipo;

        this.\_valor= valor;

    }

    get id() {

        return this.\_id;

    }

    set id (id) {

        this.\_id = id;

    }

    get nombre() {

        return this.\_nombre;

    }

    set nombre (nombre) {

        this.\_nombre = nombre;

    }

    get tipo() {

        return this.\_tipo;

    }

    set tipo (tipo) {

        this.\_tipo = tipo;

    }

    get valor() {

        return this.\_valor;

    }

    set valor (valor) {

        this.\_valor = valor;

    }

}

En esta clase se crean las variables a utilizar en el objeto.

**CLASS ETIQUETA.JS**

class Etiqueta {

    constructor(id, nombre, valor, sobrepasa) {

        this.\_id = id;

        this.\_nombre = nombre;

        this.\_sobrepasa = sobrepasa;

        this.\_valor= valor;

    }

    get id() {

        return this.\_id;

    }

    set id (id) {

        this.\_id = id;

    }

    get nombre() {

        return this.\_nombre;

    }

    set nombre (nombre) {

        this.\_nombre = nombre;

    }

    get sobrepasa() {

        return this.\_sobrepasa;

    }

    set sobrepasa (sobrepasa) {

        this.\_sobrepasa = sobrepasa;

    }

    get valor() {

        return this.\_valor;

    }

    set valor (valor) {

        this.\_valor = valor;

    }

}

En esta clase se crean las varianbles a utilizar en la funcion VAYASI para ejecutarlo satisfactoriamente.

function zeroFill( number, width )

{

  width -= number.toString().length;

  if ( width > 0 )

  {

    return new Array( width + (/./.test( number ) ? 2 : 1) ).join( '0' ) + number;

  }

  return number + ""; // always return a string

}

function ejecutarPrograma(){

  console.log('entra al ejecutar porgrama');

  correrArchivo(instrucciones, 0,instrucciones.length, arrayEtiquetas, arrayVariables,0);

  ejecutar.style.display= 'none'

}

Esta funcion es llamada por la función LEERARCHIVO(), su objetivo es agregar ceros a la izquierda de un numero.