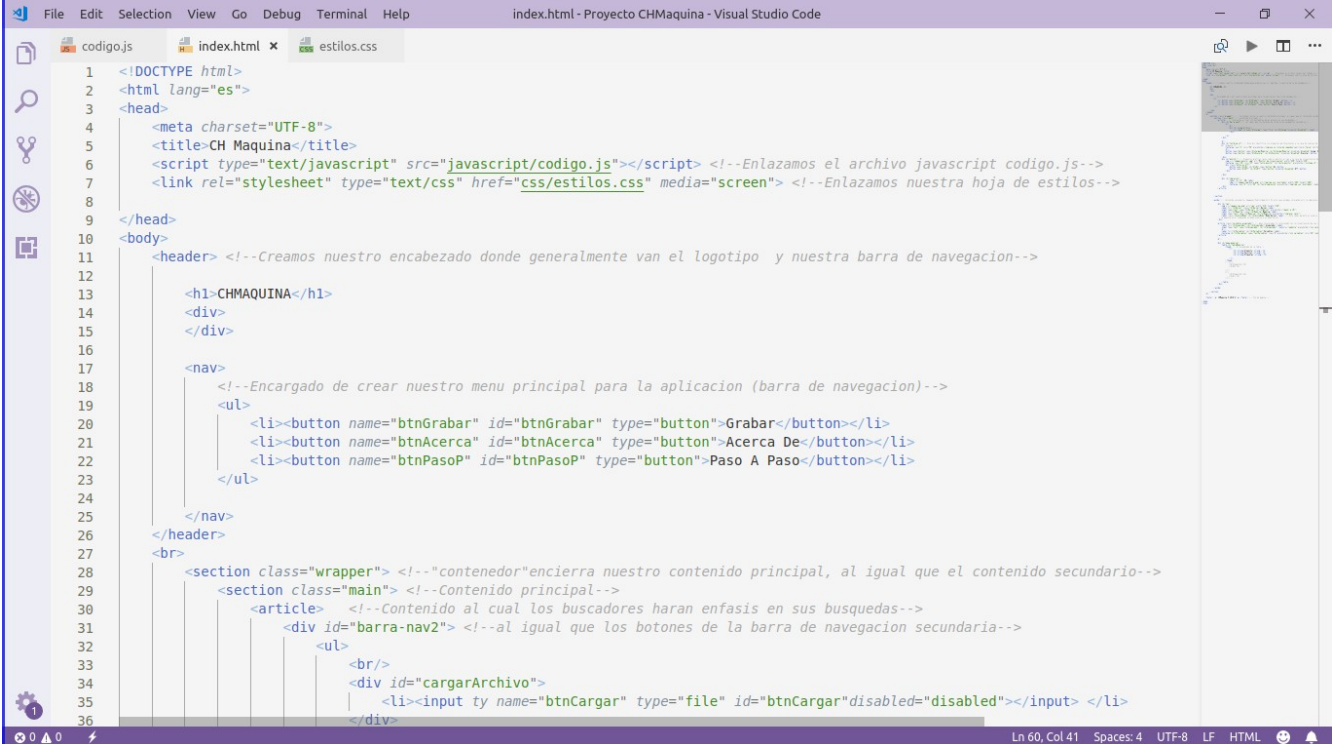


Manual Técnico CH Máquina.

Estructura página web:

Realizando un análisis previo a los requerimientos para el proyecto se plantea como prioridad la necesidad de construir un esquema básico (estructura) para lo cual se usa HTML.



```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>CH Máquina</title>
6   <script type="text/javascript" src="javascript/codigo.js"></script> <!--Enlazamos el archivo javascript codigo.js-->
7   <link rel="stylesheet" type="text/css" href="css/estilos.css" media="screen"> <!--Enlazamos nuestra hoja de estilos-->
8 </head>
9 <body>
10  <header> <!--Creamos nuestro encabezado donde generalmente van el logotipo y nuestra barra de navegacion-->
11    <h1>CHMAQUINA</h1>
12    <div>
13    </div>
14  <nav>
15    <!--Encargado de crear nuestro menu principal para la aplicacion (barra de navegacion)-->
16    <ul>
17      <li><button name="btnGrabar" id="btnGrabar" type="button">Grabar</button></li>
18      <li><button name="btnAcerca" id="btnAcerca" type="button">Acerca De</button></li>
19      <li><button name="btnPasoP" id="btnPasoP" type="button">Paso A Paso</button></li>
20    </ul>
21  </nav>
22 </header>
23 <br>
24 <section class="wrapper"> <!--"contenedor"encierra nuestro contenido principal, al igual que el contenido secundario-->
25   <section class="main"> <!--Contenido principal-->
26     <article> <!--Contenido al cual los buscadores haran enfasis en sus busquedas-->
27       <div id="barra-nav2"> <!--al igual que los botones de la barra de navegacion secundaria-->
28         <ul>
29           <br/>
30           <div id="cargarArchivo">
31             <li><input type="file" id="btnCargar" disabled="disabled"></li>
32           </div>
33         </ul>
34       </div>
35     </article>
36   </section>
37 </section>
```

Se crea una barra de navegación y la botonera temporal con los botones 'Grabar', 'Acerca De' y 'Paso a Paso'.

```
File Edit Selection View Go Debug Terminal Help index.html - Proyecto CHMaquina - Visual Studio Code
codigo.js index.html x estilos.css
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
</ul>
</div>

<br/>
<div id="textArea-ch"> <!--Este div identifica los elementos pertenecientes a la caja de ingreso de comandos
del lenguaje ch -->
<textarea rows="4" cols="50" placeholder="ingrese su lista de comandos" spellcheck="false" id="txtComandos" r
<br/>
<button type="button" name="btnCargarMemoria" id="btnCargarMemoria" disabled="disabled">Cargar En Memoria</b
<button type="button" name="btnEjecutar" id="btnEjecutar" disabled="disabled">Ejecutar</button>

</div>
<div id="monitor"> <!--Elementos graficos tales como el monitor, el cpu y la caja de visualizacion de texto corre

<textarea rows="10" cols="35" name="texto_monitor" id="texto_monitor" placeholder="sistema ch" readonly="rea
<div class="boton-on-off">
<button name="btnOn" id="btnOn" type="button">On</button>
<button name="btnOff" id="btnOff" type="button" disabled="disabled">Off</button>
</div>

</div>

<div id="impresion">
<label id="ch" >CH</label>

<textarea rows="6" cols="20" id="texto_impresion" placeholder="imprima sus resultados" readonly="readonl
</div>
</article>

</section>

<aside> <!--Contenido secundario (Imagenes-Publicidad-etc) En este caso estamos colocando alli la impresion-->
```

Un contenedor para la caja de texto donde se escribirán los comando CH así como los botones 'ejecutar' y 'cargar en memoria'.

Un contenedor para almacenar el cuadro que mostrará información cuando sea requerido por los ch-programas y sus respectivos botones de encendido y apagado de la maquina.

Un contenedor para el área de impresión en el cual se mostrara la información que se imprima en nuestros programas ch

```
File Edit Selection View Go Debug Terminal Help index.html - Proyecto CHMaquina - Visual Studio Code
codigo.js index.html x estilos.css
73 <div id="cpu">
74 
75 <label for="txtKernel" class="lblKernel">Kernel</label>
76 <input type="text" name="txtKernel" id="txtKernel" placeholder="(mayor a 10)">
77 <label for="txtMemoria" class="lblMemoria">Memoria</label>
78 <input type="text" name="txtMemoria" id="txtMemoria" placeholder="ingresar valor">
79 <label class="lblAvisoModo" id="lblAvisoModo">Modo Kernel</label> <!--Este label permite al usuario informarle s
80 la maquina esta trabajando en modo usuario o en modo kernel-->
81 </div>
82
83 <article class="variables-acumulador"> <!--Aquí colocaremos lo relacionado con la visualizacion de las variables del
84 <label for="txtAcumulador" id="lblAcumulador">Acumulador</label>
85 <input type="text" name="txtAcumulador" id="txtAcumulador" readonly="readonly" placeholder="sin valores">
86 <br/>
87 <label for="txtVariables" id="lblVariables">Variables</label>
88 <textarea id="txtVariables" name="textVariables" rows="6" placeholder="sin variables" cols="20" readonly="readon
89 </article>
90
91 <br/>
92
93 <div id="mapa memoria">
94 <table id="tablaMemoria">
95 <thead> <!--El encabezado de la tabla-->
96 <tr>
97 <th><strong>Elemento</strong></th>
98 <th><strong>Posicion</strong></th>
99 <th><strong>Etiquetas</strong></th>
100 </tr>
101 </thead>
102 <!--<tr>
103 <td>Elemento1</td>
104 <td>00</td>
105 </tr>
106 </table>
107
108
```

Los contenedores que almacenarán información tal como las entradas de 'kernel' o SO, el tamaño total de la memoria operativa, una caja de texto con información relevante sobre el uso de las variables y la variable especial 'acumulador'.

Por ultimo un contenedor llamado 'mapa_memoria' el cual almacena en una tabla toda la información en tiempo real relacionada con el estado de la memoria y las distintas instrucciones de ejecución.

Memoria Operativa.

El método recibe tres variables, el tamaño de la memoria especificado por el usuario, el kernel o tamaño de SO también asignado por el usuario y un booleano 'error' en caso de generarse excepciones en el encendido del sistema (Método que veremos mas adelante).

La función valida que se los datos ingresados por el usuario (kernel y tamaño memoria) cumplan con los lineamientos necesarios y en caso de no ser especificados por el usuario calcula un valor por defecto, Crea un array unidimensional, estructura que alojará las instrucciones operativas y de ejecución.

```

/*Esta funcion recibe los parametros del tamano inicial del vector de memoria
principal, valida si cumplen con las condiciones y asigna los parametros kernel a la memoria ppal.*/
let crear_Memoria_Principal = (tamanoMemoria, kernel, error) => {
  if(encendido==true && apagado==false) {
    if (tamanoMemoria < 40) {
      document.getElementById("texto_monitor").innerHTML= "el valor ingresado es demasiado bajo, por favor ingrese un
      "desbordamientos de memoria";
    } else if (kernel == undefined) {
      memoria_Principal = new Array(tamanoMemoria);
      kernelSistema = Math.pow(10, 1) + 9; //Donde Math.pow() es el metodo que nos permite elevar a una potencia*/
      cargar_SistemaOperativo(kernelSistema);
      memoria_Principal[0]=acumulador;
      console.log(memoria_Principal, "holal ", kernelSistema);
      if(error!="error") {
        document.getElementById("texto_monitor").innerText = "Hola, el sistema se ha iniciado correctamente";
      }
    } else if (kernel < 10) {
      console.log("El tamano inicial del sistema Operativo debe ser mayor a 10");
    } else {
      memoria_Principal= new Array(tamanoMemoria);
      kernelSistema = kernel;
      cargar_SistemaOperativo(kernelSistema);
      memoria_Principal[0]=acumulador;
      if(error!="error") {
        console.log(memoria_Principal, "Hola, el sistema ha sido iniciado de manera correcta", kernelSistema);
        document.getElementById("texto_monitor").value = "Hola, el sistema se ha iniciado correctamente";
      }
    }
  }
};

```

Cargar el sistema operativo:

```

/*Esta funcion carga el sistema operativo en el vector memoria principal*/

let cargar_SistemaOperativo = function(kernel){
  for (var i=1; i<=kernel; i++){
    memoria_Principal[i]="SO";
  }
};

```

La función crear_Memoria_Principal() a su vez hace un llamado a esta pequeña función la cual simula la carga en memoria del sistema operativo y recibe como parámetro el argumento 'kernel' para realizar la asignación de sistema operativo.

Variables Globales:

A continuación se detalla la declaración de las variables principales, el uso de estas variables sera requerido constantemente por las distintas funciones creadas para el funcionamiento de la maquina.


```

document.addEventListener("DOMContentLoaded", evt =>{
  let diccionarioDirecciones={}; /*Este diccionario me permitira obtener la ubicacion de una variable en el arreglo de memoria
  let acumulador=0; /*La variable acumulador de vital importancia para la realizacion de las distintas operaciones
  en el chmaquina*/
  let memoria_Principal=[]; /*Se crea un vector cuya funcion sera la de almacenar las instrucciones de nuestro lenguaje ch*/
  let kernelSistema; /*Variable que contendra el sistema operativo, por defecto si no se ingresa un valor en kernel este sera
  (10*z+9) el valor minimo del kernel es de 10 posiciones*/
  let diccionarioEtiquetas={};
  //Esta variable me permite llevar una cuenta y conocer que tanto se ha usado de la memoria
  let contadorMemoria=0;
  //Este vector es muy importante pues dentro almacena cada uno de los vectores que a su vez contienen cada una de las lineas
  let vectorInstrucciones=[];

  /*Esta variable determina si el sistema se encuentra encendido o apagado--Por defecto se encuentra en OFF*/
  let encendido=false;
  let apagado=true;

  //Esta variable permite alternar entre ejecutar las instrucciones y cargar las instrucciones en memoria:
  let instruccion=false;

  //Variable imprescindible, contiene en su interior la lista de los programas en memoria(posicion donde empieza, posicion o
  let listaProgramas=[];

  //Contiene la informacion del programa que actualmente se esta ejecutando:
  let infoPrograma=[];

  //Obteniendo referencia del boton 'On':
  let btnOn= document.querySelector("#btnOn");

```

Explicación detallada a cada una de las variables:

diccionarioDirecciones: Este diccionario me permitirá obtener la ubicación de una variable en el arreglo de memoria principal.

acumulador: De vital importancia para la realización de las distintas operaciones

en el ch-maquina, es la variable intermediaria o temporal que nos permite realizar sumas, restas, multiplicaciones entre otras.

memoria_Principal: Vector cuya función sera la de almacenar las instrucciones operativas y de ejecución de nuestro ch-maquina.

kernelSistema: Variable que contendrá el sistema operativo.

diccionarioEtiquetas: Este diccionario permite asociar una etiqueta creada por el usuario y una posición en memoria operativa.

vectorInstrucciones: Este vector es muy importante pues dentro almacena cada uno de los vectores que a su vez contienen cada una de las líneas de código.

encendido, apagado: Esta variable determina si el sistema se encuentra encendido o apagado--Por defecto se encuentra en OFF.

instrucción: Esta variable permite alternar entre ejecutar las instrucciones y cargar las instrucciones en memoria.

listaProgramas: Variable imprescindible tipo Lista, contiene en su interior la lista de los programas en memoria(posición donde empieza, posición donde termina,vector de instrucciones)

infoPrograma: Contiene información útil del programa que actualmente se esta ejecutando.

Métodos Encendido-Apagado.

el primer método 'encenderSistema' utiliza dos booleanos creados previamente como variables globales 'encendido' y 'apagado', por supuesto la variable encendido cambiara a True, adicional se ejecutan acciones activando y desactivando botones para evitar un uso incorrecto por ejemplo y que una vez cargada la maquina no se pueda modificar valores como el 'Kernel' del sistema y a su vez invocando funciones como la que vimos previamente de 'crear_Memoria_Principal'.

```
//Obteniendo referencia del boton 'On':
let btnOn= document.querySelector("#btnOn");

//la funcion flecha enciende el sistema
let encenderSistema = error => {
  encendido=true;
  apagado=false;
  //Habilitar los elementos de la interfaz para su posterior uso
  document.getElementById("btnOn").disabled=true;
  document.getElementById("btnOff").disabled=false;
  document.getElementById("txtMemoria").readOnly=true;
  document.getElementById("txtKernel").readOnly=true;
  document.getElementById("txtComandos").readOnly=false;
  document.getElementById("btnCargar").disabled=false;
  document.getElementById("btnCargarMemoria").disabled=false;
  //Valida si en los campos de memoria y kernel se han escrito parametros, de lo contrario asigna valores por defecto
  if(document.getElementById("txtMemoria").value==""){
    crear_Memoria_Principal(100,undefined,error);
    document.getElementById("txtMemoria").value=memoria_Principal.length;
    document.getElementById("txtKernel").value= kernelSistema;
  }else{
    var memoria=parseInt(document.getElementById("txtMemoria").value);
    var kernel= parseInt(document.getElementById("txtKernel").value);
    crear_Memoria_Principal(memoria,kernel,error);
  }
  generarTabla(true);

  return encendido;
};

//Agregando un evento tipo click cada vez que se teclee el boton 'on':
btnOn.addEventListener("click",encenderSistema);
```

Por su parte la función 'apagarSistema' modifica además de los booleanos 'encendido' y 'apagado' el estado de las cajas de texto para ingreso de comandos entre otros y habilita entradas como el 'kernel' o el 'tamaño de memoria'. aparte se invoca métodos como 'limpiarInterfaz', 'eliminarTabla' y 'limpiarSistema' los cuales veremos más adelante y cuyo propósito es limpiar las variables del sistema y la interfaz gráfica.

```
//Obteniendo referencia al boton 'Off'
let btnOff=document.querySelector("#btnOff");

let apagarSistema = () => {
  apagado=true;
  encendido=false;
  document.getElementById("btnOn").disabled=false;
  document.getElementById("btnOff").disabled=true;
  document.getElementById("txtMemoria").readOnly=false;
  document.getElementById("txtKernel").readOnly=false;
  document.getElementById("txtComandos").readOnly=true;
  document.getElementById("btnCargar").disabled=true;
  document.getElementById("lblAvisoModo").textContent="Modo Kernel";
  document.getElementById("btnEjecutar").disabled=true;
  document.getElementById("btnCargarMemoria").disabled=true;

  //Borra de la interfaz las modificaciones hechas
  limpiarInterfaz();
  eliminarTabla();
  //limpia las variables principales del sistema, entre ellas el vector de memoria principal
  limpiarSistema();
  return apagado;
};

btnOff.addEventListener("click",apagarSistema);
```

Verificación Errores al crear Variables.

Teniendo en cuenta esto se ha diseñado una estructura que usa 'switch' validando qué tipo de dato se ha ingresado.

Como podemos visualizar en la siguiente captura las validaciones en caso de que el carácter ingresado sea 'I', se valida si la línea de código contiene la estructura correcta y de ser así, se valida si se ha ingresado un número entero válido.

```

switch (vectorFrase[2]) {
    case "I":
        if (vectorFrase.length == 4) {
            if (Number.isInteger(parseInt(vectorFrase[3]))) { //valida si es un entero.de lo co
                var entero = parseInt(vectorFrase[3]);
                var posEnMemoria = almacenarVariableMemoriaPrincipal(entero);
                /*Almacena el dato en memoria principal y retorna
                la posicion donde fue almacenado*/
                infoPrograma[4][vectorFrase[1]] = posEnMemoria;
                /*Agrega al diccionario el nombre de la variable y la posicion
                donde se almaceno el dato en memoria principal*/
                document.getElementById("txtVariables").value += vectorFrase[1] + ":" + entero
            } else {
                document.getElementById("texto_monitor").value = "";
                document.getElementById("texto_monitor").value += "error encontrada, la variabl
                return "";
            }
        } else {
            var entero = 0;
            var posEnMemoria = almacenarVariableMemoriaPrincipal(entero);
            /*Almacena el dato en memoria principal y retorna
            la posicion donde fue almacenado*/
            infoPrograma[4][vectorFrase[1]] = posEnMemoria;
            /*Agrega al diccionario el nombre de la variable y la posicion
            donde se almaceno el dato en memoria principal*/
            document.getElementById("txtVariables").value += vectorFrase[1] + ":" + entero + "\
        }

        break;

    case "R":
        if (vectorFrase.length == 4) {
            var flotante = parseFloat(vectorFrase[3]); //convierte la variable del vector en fl

```

En caso de ingresarse un numero Real 'R', validando igualmente que la cadena ingresada contenga la estructura correcta


```

case "R":
    if (vectorFrase.length == 4) {
        var flotante = parseFloat(vectorFrase[3]); //convierte la variable del vector en flotante
        if (!isNaN(flotante)) { //valida si es un numero
            flotante = vectorFrase[3];
            var posEnMemoria = almacenarVariableMemoriaPrincipal(flotante);
            infoPrograma[4][vectorFrase[1]] = posEnMemoria;
            /*Agrega al diccionario el nombre de la variable y la posicion
            donde se almaceno el dato en memoria principal*/
            document.getElementById("txtVariables").value += vectorFrase[1] + ":" + flotante + "\n";
        } else {
            document.getElementById("texto_monitor").value = "Error encontrada, la variable no es un numero";
            return "";
        }
    } else {
        var flotante = 0; //convierte la variable del vector en flotante
        var posEnMemoria = almacenarVariableMemoriaPrincipal(flotante);
        infoPrograma[4][vectorFrase[1]] = posEnMemoria;
        /*Agrega al diccionario el nombre de la variable y la posicion
        donde se almaceno el dato en memoria principal*/
        document.getElementById("txtVariables").value += vectorFrase[1] + ":" + flotante + "\n";
    }
    break;
case "C":
    if (vectorFrase.length == 4) {
        var cadena = vectorFrase[3];
        var posEnMemoria = almacenarVariableMemoriaPrincipal(cadena);
        infoPrograma[4][vectorFrase[1]] = posEnMemoria;
        /*Agrega al diccionario el nombre de la variable y la posicion
        donde se almaceno el dato en memoria principal*/
        document.getElementById("txtVariables").value += vectorFrase[1] + ":" + cadena + "\n";
    } else {
        var cadena = "Error";
        var posEnMemoria = almacenarVariableMemoriaPrincipal(cadena);
    }
}

```

Por ultimo realizando la validación para datos booleanos o Lógicos, al igual que las validaciones se invoca el método 'almacenarVariableMemoriaPrincipal' se encarga de posicionar la variable en memoria operativa y luego 'infoPrograma' guarda en un diccionario en qué posición fue almacenada dicha variable.

```

case "L":
    if (vectorFrase.length == 4) {
        if (vectorFrase[3] == "0" || vectorFrase[3] == "1") {
            var logica = vectorFrase[3];
            var posEnMemoria = almacenarVariableMemoriaPrincipal(logica);
            infoPrograma[4][vectorFrase[1]] = posEnMemoria;
            /*Agrega al diccionario el nombre de la variable y la posicion
            donde se almaceno el dato en memoria principal*/
            document.getElementById("txtVariables").value += vectorFrase[1] + ":" + logica
        }
        else {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "error encontrada, la variabl";
            return "";
        }
    } else {
        var logica = "0";
        var posEnMemoria = almacenarVariableMemoriaPrincipal(logica);
        infoPrograma[4][vectorFrase[1]] = posEnMemoria;
        /*Agrega al diccionario el nombre de la variable y la posicion
        donde se almaceno el dato en memoria principal*/
        document.getElementById("txtVariables").value += vectorFrase[1] + ":" + logica + "\
    }
    break;

default:
    document.getElementById("texto_monitor").value = "";
    document.getElementById("texto_monitor").value += "se ha reconocido un error:no se reco
    return "";
}
} else {
    document.getElementById("texto_monitor").value = "";
    document.getElementById("texto_monitor").value += "se ha reconocido un error:el nombre de una v
    " la cantidad maxima de caracteres ha sido excedida (maximo 255) \n";
    return "";
}

```

Estructura utilizada para instrucción Almacene:

Primero se pregunta si la linea de código ingresada contiene las dos instrucciones obligatorias 'almacene' y 'nombre de variable destino', lo segundo es preguntar si el nombre de la variable solicitada para su guardado ha sido previamente declarada o existe en memoria operativa.

Por ultimo y si todo ha ido bien se almacena el dato que contiene el acumulador en la variable solicitada por la instrucción 'almacene'.

```

case "almacene":
    if (k != 0) {
        /*Este condicional valida si el vector contiene la estructura correcta*/
        if ((vectorFrase.length) != 2) {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "Excepcion encontrada, estructura incorrecta \n";
            return "";
        } else {
            if (infoPrograma[4][vectorFrase[1]] != undefined) { /*Esta condicion valida si dentro del diccionario existe
            /*Dentro de memoria principal almacene el dato que se encuentra en el acumulador*/

                //Preguntando si instruccion es positivo, de lo contrario no se ejecuta
                if(instruccion){
                    if (Number.isInteger(memoria_Principal[infoPrograma[4][vectorFrase[1]]))) {

                        /*Si el valor a guardar es un entero, convierte el numero a entero y lo almacena*/
                        memoria_Principal[infoPrograma[4][vectorFrase[1]]] = parseInt(acumulador);
                        document.getElementById("txtVariables").value += vectorFrase[1] + ": " + memoria_Principal[infoPrograma[4][vectorFrase[1]]] + "\n";
                    } else {
                        memoria_Principal[infoPrograma[4][vectorFrase[1]]] = acumulador;
                        document.getElementById("texto_monitor").value = "";
                        document.getElementById("txtVariables").value += vectorFrase[1] + ": " + memoria_Principal[infoPrograma[4][vectorFrase[1]]] + "\n";
                    }
                }
            } else {
                document.getElementById("texto_monitor").value = "";
                document.getElementById("texto_monitor").value += "excepcion encontrada: variable " + vectorFrase[1] + " no existe\n";
                return "";
            }
        }
    }
}
break;

```

Funcionalidades instrucción Muestra.

La instrucción muestra contiene dos partes: primero la palabra reservada 'lea', seguido del nombre de variable que desea que sea leída. Es por ello que debe validarse si el comando cuenta con dos palabras.

Corroborada la estructura del comando se procede a preguntar si se desea mostrar la información del acumulador y de no ser así, si existe actualmente en memoria el nombre de la variable que se desea mostrar.

Si al final del proceso se cumplen todas las condiciones se mostrará la información en el área 'texto_monitor'.

```

case "muestre":
    if(k!=0) {
        if (vectorFrase.length != 2) {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "Excepcion encontrada, estructura incorrecta \n";
            return "";
        } else {
            if (vectorFrase[1] == "acumulador") {
                if(instruccion){
                    if (acumulador == undefined) {
                        document.getElementById("texto_monitor").value += "actualmente el acumulador se encuentra vacio\n";
                        return "";
                    } else {
                        document.getElementById("texto_monitor").value += acumulador;
                    }
                }
            } else {
                if ((infoPrograma[4][vectorFrase[1]]) != undefined) {
                    if(instruccion){
                        document.getElementById("texto_monitor").value += memoria_Principal[infoPrograma[4][vectorFrase[1]]];
                    }
                } else {
                    document.getElementById("texto_monitor").value = "";
                    document.getElementById("texto_monitor").value += "excepcion encontrada: variable " + vectorFrase[1] + " no encontrada\n";
                    return "";
                }
            }
        }
    }
}

```

Funcionalidad instrucción Imprima:

La instrucción imprima posee una estructura muy similar a la instrucción anterior, se realizan las mismas validaciones pero se cambia el destino en el cual serán visualizados los datos en pantalla, esta vez el elemento encargado de mostrar los datos es el text-area 'texto-impresion'.

```

case "imprima":
    if(k!=0) {
        /*Limpie la caja de texto de impresion*/
        document.getElementById("texto_impresion").value = "";
        if (vectorFrase.length != 2) {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "Excepcion encontrada, estructura incorrecta \n";
            return "";
        } else {
            if (vectorFrase[1] == "acumulador") {
                //Preguntando si instruccion es positivo, de lo contrario no se ejecuta
                if(instruccion){
                    if (acumulador == undefined) {
                        document.getElementById("texto_impresion").value += "actualmente el acumulador se encuentra v
                        return "";
                    } else {
                        document.getElementById("texto_impresion").value += acumulador+"\n";
                    }
                }
            } else {
                if ((infoPrograma[4][vectorFrase[1]]) != undefined) {
                    //Preguntando si instruccion es positivo, de lo contrario no se ejecuta
                    if(instruccion){
                        document.getElementById("texto_impresion").value += memoria_Principal[infoPrograma[4][vectorF
                    ]}
                } else {
                    document.getElementById("texto_monitor").value = "";
                    document.getElementById("texto_monitor").value += "excepcion encontrada: variable " + vectorFrase
                    return "";
                }
            }
        }
    }
}
break;

```

Comandos Lógicos 'Y','O' y 'NO'.

La palabra reservada 'NO', el primer operando que es una variable lógica y el resultado que se almacena en el segundo operando. Por ello se valida si la instrucción contiene estos tres componentes.

Seguido se debe validar si las dos variables ingresadas existen en memoria ya que si se ingresa variables inexistentes se debe generar una excepción, por ultimo se debe validar si las variables ingresadas corresponde a tipo de dato 'L' (lógico).

Si todo ha ido bien se ejecuta el método 'operacionNot' detallado mas adelante.


```

y el resultado se almacena en el segundo operando.*/
case "NO":
    if (k != 0) {
        /*Este condicional valida si el vector contiene la estructura correcta*/
        if ((vectorFrase.length) != 3){
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "Excepcion encontrada, estructura incorrecta \n";
            return "";
        } else {
            if ((infoPrograma[4][vectorFrase[1]] != undefined)&&(infoPrograma[4][vectorFrase[2]] != undefined)){/*Est
                //Pregunta si las variables invocadas son booleanas, de lo contrario arroje una excepcion
                if ((memoria_Principal[infoPrograma[4][vectorFrase[1]]]=="0" || memoria_Principal[infoPrograma[4][vec
                    &&(memoria_Principal[infoPrograma[4][vectorFrase[2]]=="0" || memoria_Principal[infoPrograma[4][vecto
                    //Preguntando si instruccion es positivo, de lo contrario no se ejecuta
                    if(instruccion){
                        //Asigne a la tercer variable el resultado de la operacion entre las variables 1 y dos
                        memoria_Principal[infoPrograma[4][vectorFrase[2]]]=operacionNot(memoria_Principal[infoProgram
                    }
                } else {
                    document.getElementById("texto_monitor").value = "";
                    document.getElementById("texto_monitor").value += "excepcion encontrada: solo se permite realizar
                    return "";
                }
            } else {
                document.getElementById("texto_monitor").value = "";
                document.getElementById("texto_monitor").value += "excepcion encontrada: Revise que las variables hal
                return "";
            }
        }
    }
}
break;

```

Las instrucciones 'Y' y 'O' contienen una estructura similar, se valida si la instrucción contiene 4 comando:

La palabra reservada puede ser 'Y' y 'O', seguido de las dos variables a operar y un cuarto parámetro donde se almacena la variable. Se debe verifica si las tres variables involucradas en la operación existen en memoria operativa, ademas de ello se verifica también si estas variables son de tipo Lógico y por ultimo se invoca el método 'operacionOr' u 'operacionAnd' dependiendo del caso.

```

case "0":
    if (k != 0) {
        /*Este condicional valida si el vector contiene la estructura correcta*/
        if ((vectorFrase.length) !=4){
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "Excepcion encontrada, estructura incorrecta \n";
            return "";
        } else {
            if ((infoPrograma[4][vectorFrase[1]] != undefined)&&(infoPrograma[4][vectorFrase[2]] != undefined)
                &&(infoPrograma[4][vectorFrase[3]] != undefined)) {/*Esta condicion valida si dentro del diccionario

                //Pregunta si las variables invocadas son booleanas, de lo contrario arroje una excepcion
                if ((memoria_Principal[infoPrograma[4][vectorFrase[1]]]=="0" || memoria_Principal[infoPrograma[4][vec
                (memoria_Principal[infoPrograma[4][vectorFrase[2]]]=="0" || memoria_Principal[infoPrograma[4][vectorF
                (memoria_Principal[infoPrograma[4][vectorFrase[3]]]=="0" || memoria_Principal[infoPrograma[4][vectorF

                //Preguntando si instruccion es positivo, de lo contrario no se ejecuta
                if(instruccion){
                    //Asigne a la tercer variable el resultado de la operacion entre las variables 1 y dos
                    memoria_Principal[infoPrograma[4][vectorFrase[3]]]=operacionOr(memoria_Principal[infoPrograma
                    | memoria_Principal[infoPrograma[4][vectorFrase[2]]]);
                }

            } else {
                document.getElementById("texto_monitor").value = "";
                document.getElementById("texto_monitor").value += "excepcion encontrada: solo se permite realizar
                return "";
            }

        } else {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "excepcion encontrada: Revise que las variables hal
            return "";
        }
    }
}

```

```

case "Y":
    if (k != 0) {
        /*Este condicional valida si el vector contiene la estructura correcta*/
        if ((vectorFrase.length) !=4) {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "Excepcion encontrada, estructura incorrecta \n";
            return "";
        } else {
            if ((infoPrograma[4][vectorFrase[1]] != undefined)&&(infoPrograma[4][vectorFrase[2]] != undefined)
                &&(infoPrograma[4][vectorFrase[3]] != undefined)) {/*Esta condicion valida si dentro del diccionario exis

                //Pregunta si las variables invocadas son booleanas, de lo contrario arroje una excepcion
                if ((memoria_Principal[infoPrograma[4][vectorFrase[1]]]=="0" || memoria_Principal[infoPrograma[4][vec
                (memoria_Principal[infoPrograma[4][vectorFrase[2]]]=="0" || memoria_Principal[infoPrograma[4][vectorF
                (memoria_Principal[infoPrograma[4][vectorFrase[3]]]=="0" || memoria_Principal[infoPrograma[4][vectorF

                if(instruccion){
                    //Asigne a la tercer variable el resultado de la operacion entre las variables 1 y dos
                    memoria_Principal[infoPrograma[4][vectorFrase[3]]]=operacionAnd(memoria_Principal[infoProgram
                    | memoria_Principal[infoPrograma[4][vectorFrase[1]]]);
                    //Escriba en la caja de texto de variables el valor de la variable
                }

            } else {
                document.getElementById("texto_monitor").value = "";
                document.getElementById("texto_monitor").value += "excepcion encontrada: solo se permite realizar
                return "";
            }

        } else {
            document.getElementById("texto_monitor").value = "";
            document.getElementById("texto_monitor").value += "excepcion encontrada: Revise que las variables hal
            return "";
        }
    }
}

```

Por ultimo los métodos encargados de realizar las operaciones correspondientes.

```
}  
//La funcion recibe dos parametros y realiza una operacion And entre ambos  
function operacionOr(variable1,variable2){  
    if(variable1=="0" && variable2=="0"){  
        return "0";  
    }else{  
        return "1";  
    }  
}  
  
//La funcion recibe dos parametros y realiza una operacion And entre ambos  
function operacionAnd(variable1,variable2){  
    if(variable1=="1" && variable2=="1"){  
        return "1";  
    }else{  
        return "0";  
    }  
}  
  
//La funcion recibe un parametro y produce su respectiva negacion  
function operacionNot(variable1) {  
    if (variable1 == "0") {  
        return "1";  
    } else {  
        return "0";  
    }  
}
```

Comando 'vaya'.

Se debe validar primero, si la estructura ingresada a la instrucción es la correcta, adicional debe validarse si el nombre de etiqueta que se indica efectivamente existe, de lo contrario no se puede continuar con la ejecución.

```

1039     case "vaya":
1040         if(k!=0){
1041             if(vectorFrase.length!=2){
1042                 document.getElementById("texto_monitor").value="";
1043                 document.getElementById("texto_monitor").value+="Excepcion encontrada, estructura in
1044                 return "";
1045             }
1046         }else{
1047             //Si el operando se encuentra en el diccionario de etiquetas
1048             if(vectorFrase[1] in infoPrograma[5] || k==0){
1049                 if(instruccion){
1050                     //Si el operando asignado a la etiqueta es menor que la variable kernel lanc
1051                     var posicion=buscarFrase(vectorFrase[1]);
1052                     if(posicion!=undefined){
1053                         return IraInstruccion(posicion);
1054                     }else if(k==0){
1055                         break;
1056                     }else{
1057                         document.getElementById("texto_monitor").value="";
1058                         document.getElementById("texto_monitor").value+="Excepcion encontrada,et
1059                         return "";
1060                     }
1061                 }
1062             }
1063         }
1064     }
1065 }
1066 }else{
1067     //Si tras haber realizado el segundo chequeo no se encuentra la etiqueta, despli
1068     document.getElementById("texto_monitor").value="";
1069     document.getElementById("texto_monitor").value+="excepcion encontrada:etiqueta "
1070     return "";
1071 }
1072 }
1073 }

```

El método 'buscarFrase' me permite obtener la línea de código indicada por la etiqueta inmediatamente después de la instrucción 'vaya', también me permite consultar si la línea de código solicitada no excede los límites del programa.

```

1230
1231
1232     }
1233
1234     function buscarFrase(etiqueta){
1235         //obtenga la posicion en memoria donde se encuentra la linea de codigo solicitada
1236         var posicion=infoPrograma[5][etiqueta];
1237         if(posicion!=undefined && posicion!=0) {
1238             var frase = vectorInstrucciones[posicion-1];
1239             //elimina los espacios de la linea de codigo
1240             // var lineaCodigo=frase.split(" ");
1241             //elimina el primer elemento de array el cual es un ""
1242             //lineaCodigo.shift();
1243             //obtenga la siguiente linea de codigo en memoria principal necesaria para hacer la busqueda
1244             var siguienteLinea = vectorInstrucciones[posicion];
1245
1246             for (var i = 0; i < vectorInstrucciones.length; i++) {
1247                 //si la linea de codigo solicitada se encuentra en el vector que contiene todas las instrucc
1248                 if ((frase[0] == vectorInstrucciones[i][0]) && frase[1] == vectorInstrucciones[i][1]) {
1249                     if(siguienteLinea==undefined){
1250                         return i;
1251                     }else if ((siguienteLinea[0] == vectorInstrucciones[i + 1][0]) && siguienteLinea[1] == v
1252                         return i;
1253                     }
1254                     break;
1255                 }
1256             }
1257         }else {
1258             return undefined;
1259         }
1260     }
1261     /*Esta funcion genera una tabla con el contenido almacenado en memoria principal*/
1262     function generarTabla(bandera){
1263         //Obtener la referencia del elemento body
1264         let mapaMemoria= document.querySelector("#mapa_memoria");
1265         //Invocamos el elemento table

```

Por último la función que realiza la magia de saltar de una línea hacia otra cambiando el orden de ejecución esta a cargo del método 'IraInstruccion', el método recibe como argumento la posición en memoria de la línea solicitada, ubica la posición del vector en memoria y continua a partir de allí la ejecución del programa.


```

1210 }
1211 //La funcion recibe un parametro y produce su respectiva negacion
1212 function operacionNot(variable1) {
1213     if (variable1 == "0") {
1214         return "1";
1215     } else {
1216         return "0";
1217     }
1218 }
1219
1220 /**
1221  * Cuando se ingresa en una instruccion vayasi, el metodo se encarga de saltar a la linea especificada e
1222  * @param {*} posicion : posicion en memoria principal donde se encuentra la linea solicitada
1223  */
1224 function IraInstruccion(posicion) {
1225     for(var i=posicion; i<vectorInstrucciones.length; i++){
1226         var condicion=ejecutarInstrucciones(vectorInstrucciones[i],"",1);
1227         if(condicion!=undefined){
1228             return condicion;
1229         }
1230     }
1231 }
1232
1233
1234 function buscarFrase(etiqueta){
1235     //obtenga la posicion en memoria donde se encuentra la linea de codigo solicitada
1236     var posicion=infoPrograma[5][etiqueta];
1237     if(posicion!=undefined && posicion!=0) {
1238         var frase = vectorInstrucciones[posicion-1];
1239         //elimina los espacios de la linea de codigo
1240         // var lineaCodigo=frase.split(" ");
1241         //elimina el primer elemento de array el cual es un ""
1242         //lineaCodigo.shift();
1243         //obtenga la siguiente linea de codigo en memoria principal necesaria para hacer la busqueda
1244         var siguienteLinea = vectorInstrucciones[posicion];
1245     }

```