Manual técnico ChMaquina

Jhon Alejandro Franco Gómez

jhon.1702011356@ucaldas.edu.co

0000013167

Universidad de Caldas

Manizales, Caldas

Tabla de contenido:

Lenguaje utilizado:

Para el desarrollo del ChMaquina se ha utilizado JavaScript como lenguaje principal, HTML5, CSS3 y Bootstrap 5 para la interfaz de usuario. Siendo estas tecnologías web, se establece la ejecución del ChMaquina en entorno web.

Programación de la interfaz gráfica:

Parte superior:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./assets/css/bootstrap.min.css">
    <link rel="stylesheet" href="./assets/css/styles.css">
    <title>CHMAQUINA</title>
</head>

<body>
    <header class="text-center">
        <h1 class="my-3">
            ChMáquina
        </h1>
        <img src="./assets/img/iconos-linux.png" alt="">
    </header>
    <div class="row mx-5">
        <div class="col mt-3">
            <button class="btn btn-primary my-2 my-sm-0">Cargar .CH<br><input class="" type="file" id="file-input"
                    value="Abrir Archivo"></button>
        </div>

        <div class="col mt-3">
            <button class="btn btn-primary my-2 my-sm-0" type="submit" id="start">Iniciar</button>
            <button class="btn btn-primary my-2 my-sm-0" type="submit" id="stepByStep">Ejecución paso a paso</button>
            <a href="./assets/views/ide.html" class="btn btn-primary">Crear programa .ch</a>
        </div>
    </div>
    <div class="row mx-5">
        <div class="col mt-3">
```

Salidas centrales:

```
    <div class="row mx-5">
        <div class="col my-3">

            <label for="memoria">Memoria:</label>
            <input type="number" name="memoria" id="memoryInput" max="7100" min="120">
            <label for="kernel">Kernel:</label>
            <input type="number" name="kernel" id="kernelInput" max="7100" min="79">
            <a class=""><button class="btn btn-primary my-2 my-sm-0" id="setValues">Establecer valores</button></a>
            <a class=""><button class="btn btn-primary my-2 my-sm-0" id="btnMapa"
                    onclick="MapaMemoria.id = 'MapaMemoriaMostrar'; return false;">Mostrar Mapa de
                    memoria</button></a>
            <a class=""><button class="btn btn-primary my-2 my-sm-0" id="btnMapa"
                    onclick="MapaMemoriaMostrar.id = 'MapaMemoria'; return false;">Quitar Mapa de
                    memoria</button></a>

            <div class="col-sm w-25 text-center border border-primary mt-5">
                <p>Acumulador: </p>
                <p id="acumulador"></p>
            </div>
            <div class="row">
                <div class="col">
                    <h5>Pantalla</h5>
                    <textarea name="pantalla" id="pantalla" rows="1000"
                        style="width:350px;height:250px; resize: none;"></textarea>
                </div>
                <div class="col">
                    <h5>Impresora</h5>
                    <textarea disabled name="impresora" id="impresora" rows="1000"
                        style="width:350px;height:250px; resize: none;"></textarea>
                </div>
            </div>

            <br>
```

Sección inferior:

```

    <div class="row mx-5">
        <div class="col-sm-4">
            <h5>Compilador</h5>
            <textarea disabled name="compilador" id="compilador"
                style="width:300px;height:400px; resize: none;"></textarea>
        </div>


        <div class="col-sm-4">
            <h5>Mapa de variables</h5>
            <textarea disabled name="variables" id="variables" rows="1000"
                style="width:350px;height:250px; resize: none;"></textarea>
            <hó>Etiquetas</hó>
            <br><textarea disabled name="Etiquetas" id="Etiquetas"
                style="width:250px;height:150px; resize: none;"></textarea>
            <br>
        </div>

        <div class="col-sm-4" id="MapaMemoria">
            <h5>Mapa de memoria: </h5>
            <textarea disabled name="Mapa" id="MapaM" style="width:400px;height:500px; resize: none;"></textarea>
        </div>
    </div>

    <div class="row mx-auto">

    </div>

```

Programación del ChMáquina:

Inicialización del procesamiento lógico:

```
/**
 * ------------------------------------------------------------------------
 * Apartado donde se crean las primeras instancias al ingresar al ChMáquina
 * ------------------------------------------------------------------------
 */

var todosLosProgramas = new Array();

var reader;
var lineaRetorne;
//Se obtienen los valores de la memoria y el kernel
let kernelValue;
let memoryValue;
//Se crea el vector de memoria
var memoria = new Array();
document.getElementById('kernelInput').value = 79;
document.getElementById('memoryInput').value = 120;
const setValues = document.getElementById('setValues').addEventListener('click', () => {
    kernelValue = parseInt(document.getElementById('kernelInput').value);
    memoryValue = parseInt(document.getElementById('memoryInput').value);

    //El espacio del Kernel debe ser inferior al total de la memoria
    if (kernelValue < memoryValue) {
        for (let i = 0; i <= kernelValue; i++) {
            memoria[i] = "Linux Kernel - Jhon Franco";
        }
    } else {
        alert("La memoria debe ser mayor que el Kernel");
    }

});
```

```
//Se instancia el valor inicial del acumulador
var acumulador = 0;
document.getElementById('acumulador').innerText = acumulador;

//Se crea el almacenamiento para las variables que cree el usuario
var almacenamientoVariables = new Array();
var almacenamientoEtiquetas = new Array();
```

Configuración inicial de la interfaz:

```
/**...
//Permite cargar el archivo al ChMáquina
const inputFile = document.getElementById('file-input').addEventListener('change', abrirArchivo);

//Botón que da inicio a la ejecución del ChMáquina con los programas cargados en la memoria
const startBtn = document.getElementById('start');
startBtn.addEventListener('click', start);

const stepBtn = document.getElementById('stepByStep');
stepBtn.addEventListener('click', stepByStep);

const btnSJFNoExp = document.getElementById('SJFNoExp');
btnSJFNoExp.addEventListener('click', SJFNoExp);

const btnFCFS = document.getElementById('FCFS');
btnFCFS.addEventListener('click', start);
```

¿Cómo se abre un archivo?

```
function abrirArchivo(evt) {

    let file = evt.target.files[0];
    reader = new FileReader();

    reader.onload = function (e) {
        let content = e.target.result;
        if (!verificarErrores(content)) {
            let instruccion = content.split('\n');
            if (memoria.length + instruccion.length < memoryValue) {
                let programa = new Array();
                for (let i = 0; i < instruccion.length; i++) {
                    if (!instruccion[i].includes("//") || !instruccion[i].includes("/")) {
                        memoria.push(instruccion[i].trim());
                        programa.push(instruccion[i].trim());
                    }
                    if (instruccion[i].includes("retorne")) {
                        lineaRetorne = i;
                    }
                }
                todosLosProgramas.push(programa);
            } else {
                //Valida que la memoria no se desborde
                alert("La memoria está llena, no se pueden agregar más instrucciones");
            }
            document.getElementById('compilador').value += instruccion.join('') + '\n';
            document.getElementById('compilador').scrollTop = document.getElementById('compilador').scrollHeight
        }
    }
    reader.readAsText(file);
}
```

Explicación:
Se utiliza el API File para el manejo de archivos, mediante un input file se carga el archivo .ch, y se desglosa el contenido del archivo para validarlo como se ve a continuación.

Validación de errores:

```
function verificarErrores(contenido) {
    var fragmentoTexto = contenido.split("\n");
    var error = false;
    for (var i = 0; i < fragmentoTexto.length; i++) {

        var palabra = fragmentoTexto[i].split(" ");

        if (palabra[0] == "cargue" || palabra[0] == "pare" || palabra[0] == "itere" ||
            palabra[0] == "almacene" || palabra[0] == "nueva" || palabra[0] == "lea" ||
            palabra[0] == "sume" || palabra[0] == "reste" || palabra[0] == "multiplique" ||
            palabra[0] == "divida" || palabra[0] == "potencia" || palabra[0] == "modulo" ||
            palabra[0] == "concatene" || palabra[0] == "elimine" || palabra[0] == "extraiga" ||
            palabra[0] == "y" || palabra[0] == "o" || palabra[0] == "no" || palabra[0] == "muestre" ||
            palabra[0] == "imprima" || palabra[0] == "vaya" || palabra[0] == "vayasi" ||
            palabra[0] == "etiqueta" || palabra[0] == "xxx" || palabra[0] == "retorne") {
        }
        else {
            if (palabra[0] == null || palabra[0] == "" || palabra[0] == "//") {

            }
            else {
                alert("Error en la linea numero: " + i);
                error = true;
            }
        }
    }
    return error;
}
```

Explicación: Recibe el contenido del archivo y lo desfragmenta en renglones y luego en palabras, para validar que cada instrucción escrita coincida con un repertorio de palabras reservadas.

Explicación de los algoritmos más complejos y/o relevantes:

Nota: Todos los algoritmos reciben un parámetro que mostrará una alerta al usuario cuando elija el modo paso a paso.

Nueva:

```
function nueva(ins, sbs) {
    if (sbs == 1) {
        alert("Se crea una nueva variable");
    }
    let nuevaVariable = new Object();
    if (ins.length == 4) {
        nuevaVariable.nombre = ins[1];
        nuevaVariable.tipo = ins[2];
        nuevaVariable.valor = parseInt(ins[3]);
    } else if (ins.length == 3) {
        nuevaVariable.nombre = ins[1];
        nuevaVariable.tipo = ins[2];

        switch (ins[2]) {
            case "C":
                nuevaVariable.valor = ' ';
                break;
            case "I" || "R":
                nuevaVariable.valor = 0;
                break;
            case "L":
                nuevaVariable.valor = 0;
                break;
            default:
                break;
        }
    }
    almacenamientoVariables.push(nuevaVariable);
    document.getElementById('variables').value += "Variable: " + nuevaVariable.nombre + "\n";
    document.getElementById('variables').scrollTop = document.getElementById('variables').scrollHeight
}
```

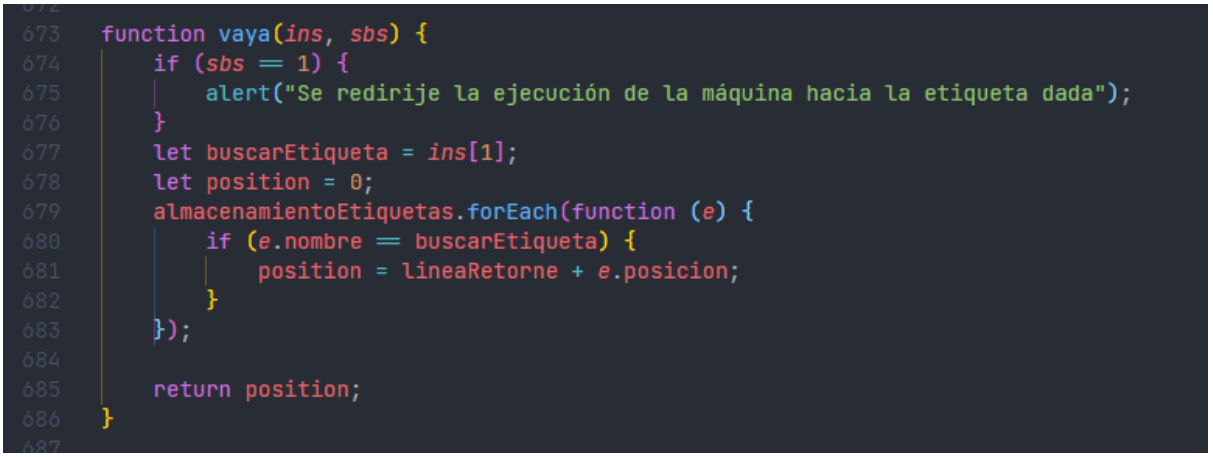
Las variables del ChMaquina se crean haciendo uso de los objetos que proporciona JavaScript, los valores de este objeto son dados por los operandos de la instrucción.
En caso de que no haya un valor establecido para la variable, esta determinará el tipo de variable que se está creando y se le asignará el valor por defecto de acuerdo a su tipo.
Posteriormente se agregará al almacenamiento de variable y se mostrará al usuario en la sección Variables.

Etiqueta:

```
function etiqueta(ins, sbs) {
    if (sbs == 1) {
        alert("Se crea una nueva etiqueta");
    }
    let index = parseInt(ins[2]);
    let nuevaEtiqueta = new Object();
    nuevaEtiqueta.nombre = ins[1];
    nuevaEtiqueta.posicion = index;
    almacenamientoEtiquetas.push(nuevaEtiqueta);
    document.getElementById('Etiquetas').value += "Etiqueta: " + nuevaEtiqueta.nombre + "\n";
    document.getElementById('Etiquetas').scrollTop = document.getElementById('Etiquetas').scrollHeight
}
```

Crea una nueva etiqueta, similar a las variables, solo que esta almacena un valor numérico que representa la posición de la instrucción que se quiere almacenar. Posteriormente la agrega al almacenamiento de etiquetas.

Nueva:



```
function vaya(ins, sbs) {
    if (sbs == 1) {
        alert("Se redirije la ejecución de la máquina hacia la etiqueta dada");
    }
    let buscarEtiqueta = ins[1];
    let position = 0;
    almacenamientoEtiquetas.forEach(function (e) {
        if (e.nombre == buscarEtiqueta) {
            position = lineaRetorne + e.posicion;
        }
    });

    return position;
}
```

El algoritmo recibe una etiqueta, la cual buscará en el almacenamiento de etiquetas, en caso de encontrarla tomará la línea de retorne, y la sumará con la posición almacenada en la etiqueta para devolver la posición a la que se debe ir.

vayasi:

```
function vayasi(sbs, ins) {
    if (sbs == 1) {
        alert("Valida las condiciones, y determina a cual etiqueta debe ir");
    }
    let ac = parseInt(acumulador);
    let position = 0;
    let et1 = ins[1];
    let et2 = ins[2];
    if (ac > 0) {
        almacenamientoEtiquetas.forEach(function (e) {
            if (e.nombre == et1) {
                position = lineaRetorne + e.posicion;
            }
        });

        return position;
    } else if (ac < 0) {
        almacenamientoEtiquetas.forEach(function (e) {
            if (e.nombre == et2) {
                position = lineaRetorne + e.posicion;
            }
        });

        return position;
    } else if (ac === 0) {
        console.log("No debería hacer nada...");
    }
}
```

Exactamente lo mismo que `vaya` pero con las condiciones siguientes:

- Si el valor del acumulador es mayor a cero, salte a la instrucción que corresponde a la etiqueta indicada por el primer operando.
- Si el valor del acumulador es menor a cero, salte a la instrucción que corresponde a la etiqueta indicada por el segundo operando.
- Si el valor del acumulador es igual a cero, salte a la siguiente instrucción.

`retorne`:

```
function retorne(ins, sbs) {
    if (sbs == 1) {
        alert("Se finaliza la ejecución del programa actual \n y se da paso a el siguiente programa cargado en memoria. Además se
        reestablece el acumulador, se muestra por pantalla y se imprime en la impresora el valor final del acumulador.");
    }

    document.getElementById('impresora').value += "\n" + "Acumulador final del programa: " + acumulador + "\n";
    document.getElementById('pantalla').value += "Acumulador final del programa: " + acumulador + "\n";
    lineaRetorne = memoria.indexOf("retorne");
    almacenamientoEtiquetas = [];
    almacenamientoVariables = [];

}
```

El algoritmo es bastante sencillo, simplemente imprime y muestra cuando el programa ha terminado, con el valor del acumulador final y obtiene la línea para dar inicio al siguiente programa.
Adicionalmente limpia el almacenamiento de variables y etiquetas para evitar confusiones entre programas.

El resto de algoritmos son sumamente sencillos, y además están comentados en líneas de código dentro del proyecto, por lo que no han sido explicados en el manual técnico.

Ahora se muestra la programación del entorno de desarrollo para el lenguaje CH.

```
document.getElementById('downloadBtn').addEventListener('click', download);

function download() {
    let btnDownload = document.getElementById('downloadBtn');
    content = document.getElementById('ide').value;
    if (verificarErrores(content)) {
        alert('No se puede descargar, porque el archivo contiene errores.');
    } else {
        btnDownload.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(content));
        btnDownload.setAttribute('download', "archivo.ch");
    }
}

function verificarErrores(contenido) {
    var fragmentoTexto = contenido.split("\n");
    var error = false;
    for (var i = 0; i < fragmentoTexto.length; i++) {

        var palabra = fragmentoTexto[i].split(" ");

        if (palabra[0] == "cargue" || palabra[0] == "pare" || palabra[0] == "itere" ||
            palabra[0] == "almacene" || palabra[0] == "nueva" || palabra[0] == "lea" ||
            palabra[0] == "sume" || palabra[0] == "reste" || palabra[0] == "multiplique" ||
            palabra[0] == "divida" || palabra[0] == "potencia" || palabra[0] == "modulo" ||
            palabra[0] == "concatene" || palabra[0] == "elimine" || palabra[0] == "extraiga" ||
            palabra[0] == "y" || palabra[0] == "o" || palabra[0] == "no" || palabra[0] == "muestre" ||
            palabra[0] == "imprima" || palabra[0] == "vaya" || palabra[0] == "vayasi" ||
            palabra[0] == "etiqueta" || palabra[0] == "xxx" || palabra[0] == "retorne") {
        }
        else {
            if (palabra[0] == null || palabra[0] == "" || palabra[0] == "//") {

            }
```

Explicación:
Primeramente se define el botón para la descarga, y posteriormente se definen los métodos a utilizar.
El primero es `download()` el cual se encarga de tomar el botón de descarga, tomar el contenido del área de escritura, y a ese botón se le asignan los atributos para procesar dicho contenido y asignarlo como descarga al botón.

Se puede ver que debajo de esta función, se definió el mismo algoritmo para verificar errores que se usó en el compilador principal. La razón de esto es para evitar que se pueda descargar un archivo que contenga errores de sintaxis.

Algoritmos de planificación de procesos no expropiativos:

SJF:

```
function SJFNoExp() {
    alert("Método: SJF No Expropiativo")
    todosLosProgramas.sort();
    console.log(todosLosProgramas);
    todosLosProgramas.forEach(item => {
        for (let i = 0; i < item.length; i++) {

            let ins = item[i].split(" ");
            switch (ins[0]) {
                case "nueva":
                    nueva(ins);
                    break;
                case "almacene":
                    almacene(ins);
                    break;
                case "cargue":
                    cargue(ins);
                    break;
                case "lea":
                    lea(ins);
                    break;
                case "sume":
                    sume(ins);
                    break;
                case "reste":
                    reste(ins);
                    break;
                case "multiplique":
                    multiplique(ins);
                    break;
                case "divida":
                    divida(ins):
```

La explicación del algoritmo es bastante simple, ordena los programas en base a las ráfagas de CPU por cada programa/proceso, de menor a mayor.

FCFS:

```
function FCFS() {
    for (let i = 0; i ⩽ memoria.length - 1; i++) {
        document.getElementById('MapaM').value += memoria[i] + "\n";
        document.getElementById("MapaM").scrollTop = document.getElementById("MapaM").scrollHeight
        let ins = memoria[i].split(" ");
        switch (ins[0]) {
            case "nueva":
                nueva(ins);
                break;
            case "almacene":
                almacene(ins);
                break;
            case "cargue":
                cargue(ins);
                break;
            case "lea":
                lea(ins);
                break;
            case "sume":
                sume(ins);
                break;
            case "reste":
                reste(ins);
                break;
            case "multiplique":
                multiplique(ins);
                break;
            case "divida":
                divida(ins);
                break;
            case "concatene":
```

Funciona de manera estructurada con los procesos cargados en la memoria, no hay un método de ordenamiento y ejecuta una tarea a la vez.