

**Generación de código**

**Integrantes**

Farfán de León José Osvaldo

Sección: D06

Salcedo Arellano Alexa

Sección: D06

Víctor Manuel Pacheco Moreno

Sección: D03

**Materia**: Traductores de Lenguajes ll

**Profesor**: Ramos Barajas Armando

Índice

[Introducción 3](#_Toc103191989)

[Características 3](#_Toc103191990)

[Función 3](#_Toc103191991)

[**Objetivos:** 3](#_Toc103191992)

[General 4](#_Toc103191993)

[Particular 4](#_Toc103191994)

[Desarrollo: 4](#_Toc103191995)

[Apéndices 7](#_Toc103191996)

[Acrónimo 19](#_Toc103191997)

[Requisitos funcionales 19](#_Toc103191998)

[Requisitos no funcionales 19](#_Toc103191999)

[Complejidad ciclomática 20](#_Toc103192000)

[COCOMO 20](#_Toc103192001)

[Caja negra 20](#_Toc103192002)

[Caja blanca 21](#_Toc103192003)

[Grafos 22](#_Toc103192004)

[Tabla de transiciones 24](#_Toc103192005)

[Conclusión: 25](#_Toc103192006)

[Bibliografía 25](#_Toc103192007)

# Introducción

En esta actividad mejoraremos la practica previamente realizada en la cual ahora será posible de detectar operaciones directamente en ensamblador específicamente para el lenguaje c++ que es el actual con el cual estamos trabajando, así como también se implementaran algunas operaciones trigonométricas raíz entre otras, respetando la sintaxis así como también el análisis semántico, sintáctico.

El lenguaje ensamblador que se utiliza para traducir programas al código de máquina. Cada microprocesador o micro controlador tiene su propio lenguaje ensamblador con su particular conjunto de instrucciones y sus varios modos de direccionamiento que dependen de la arquitectura del hardware, cada nuevo procesador de INTEL que sale al mercado, el fabricante agrega algunas instrucciones a su conjunto de instrucciones, pero siempre conservan las instrucciones y registros de los modelos anteriores por razones de compatibilidad.

## Características

El analizador léxico tiene que dividir la secuencia de caracteres en palabras con significado propio y después convertirlo a una secuencia de terminales desde el punto de vista del analizador sintáctico, ya que es su entrada. El analizador léxico reconoce las palabras en función de una gramática regular de manera que sus no terminales se convierten en los elementos de entrada de fases posteriores. En Lex, por ejemplo, esta gramática se expresa mediante expresiones regulares.

## Función

La generación de código nos permite ejecutar operaciones a nivel de lenguaje ensamblador, es decir, manejar directamente las operaciones con la computadora, a comparación de los lenguajes de alto nivel tales como c++, java entre otros, estos manejan directamente los registros de la computadora, crenado o mejorando el tiempo de retorno de las operaciones realizadas en el mismo.

# **Objetivos:**

## General

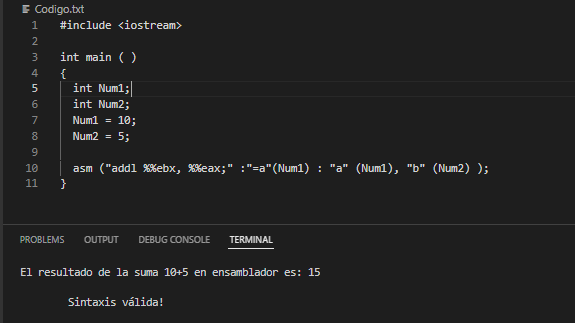
Implementar funciones las cuales acepte el programa y sean ejecutadas directamente en el lenguaje ensamblador para la generación de código, estas serán embebidas con el lenguaje de programación c++, conociendo las diferentes instrucciones, validando los posibles casos de error con el retorno de un mensaje de sintaxis no valida en caso contrario el retorno de los resultados de dichas operaciones.

## Particular

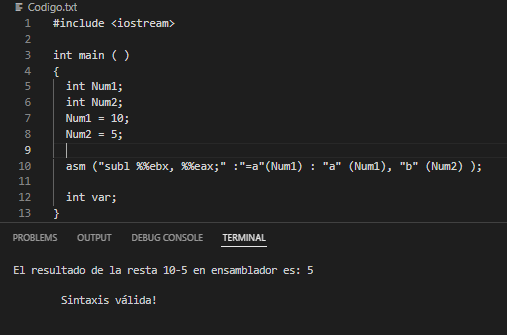
Realizar segmentos de códigos enfocados al funcionamiento correcto de todas las operaciones necesarias para una calculadora científica, a través del ASM embebidos en C++, siendo interpretadas y ejecutadas por visual studio code.

# Desarrollo:

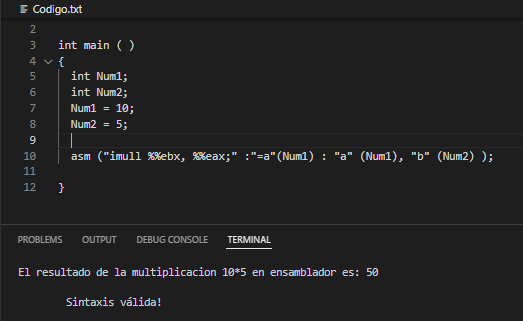
A continuación, se muestra la ejecución de una suma en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



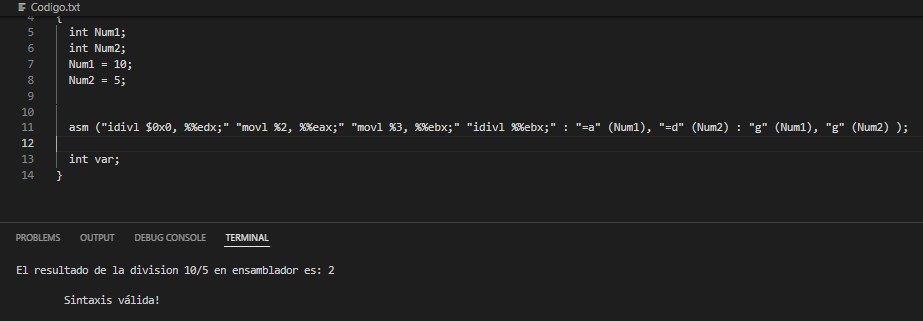
A continuación, se muestra la ejecución de una resta en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



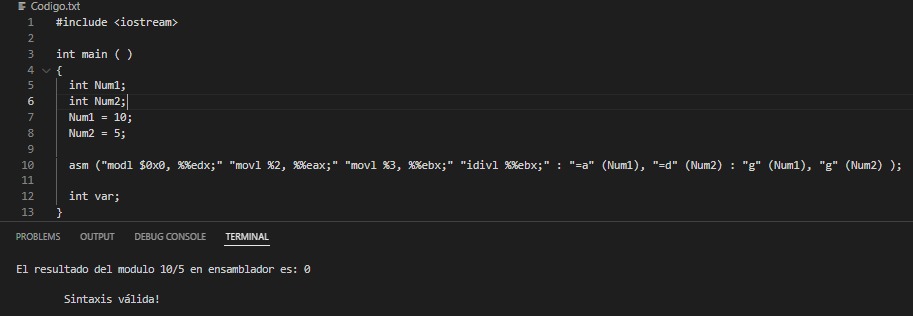
A continuación, se muestra la ejecución de una multiplicación en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



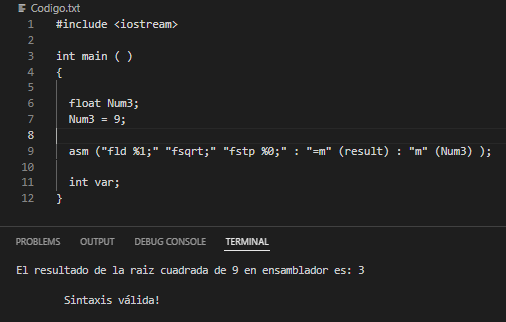
A continuación se muestra la ejecución de una División en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



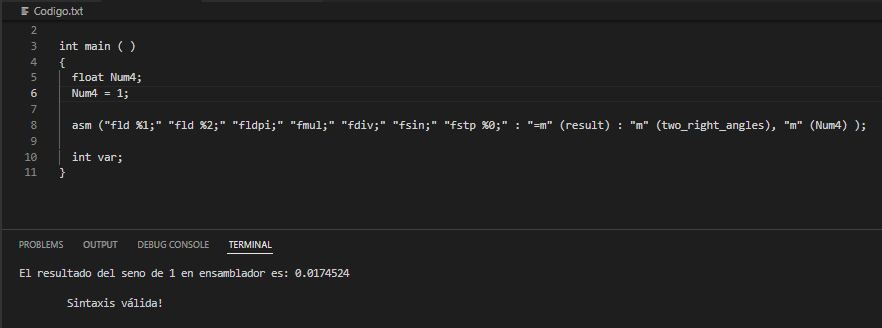
A continuación, se muestra la ejecución de un módulo en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



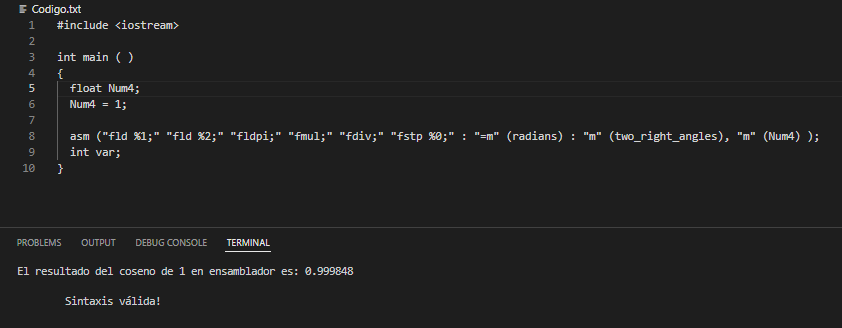
A continuación, se muestra la ejecución de una raíz cuadrada en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



A continuación, se muestra la ejecución de un seno en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



A continuación, se muestra la ejecución de un coseno en lenguaje ensamblador directamente desde nuestro programa la cual es ejecutada en asm y devuelto el resultado a el usuario.



# Apéndices

Ensamblador.h

#include "Funciones.h"

#include <iostream>

using namespace std;

//asm ("addl %%ebx, %%eax;" :"=a"(Num1) : "a" (Num1), "b" (Num2) );

void Suma(){

    int var1 = Num1;

    int var2 = Num2;

    \_\_asm ("addl %%ebx, %%eax;" :"=a"(Num1) : "a" (Num1), "b" (Num2) );

    cout <<"El resultado de la suma "<< var1 << "+" << var2 << " en ensamblador es: " << Num1 << endl;

}

//asm ("subl %%ebx, %%eax;" :"=a"(Num1) : "a" (Num1), "b" (Num2) );

void Resta (){

    int var1 = Num1;

    int var2 = Num2;

    \_\_asm ("subl %%ebx, %%eax;" :"=a"(Num1) : "a" (Num1), "b" (Num2));

    cout <<"El resultado de la resta " << var1 << "-" << var2 << " en ensamblador es: " << Num1 << endl;

}

//asm ("imull %%ebx, %%eax;" :"=a"(Num1) : "a" (Num1), "b" (Num2) );

void Multiplicacion (){

    int var1 = Num1;

    int var2 = Num2;

    \_\_asm ("imull %%ebx, %%eax;" :"=a"(Num1) : "a" (Num1), "b" (Num2) );

    cout <<"El resultado de la multiplicacion " << var1 << "\*" << var2 << " en ensamblador es: " << Num1 << endl;

}

//asm ("idivl $0x0, %%edx;" "movl %2, %%eax;" "movl %3, %%ebx;" "idivl %%ebx;" : "=a" (Num1), "=d" (Num2) : "g" (Num1), "g" (Num2) );

void Division (int va, int va2){

    int var1 = va;

    int var2 = va2;

    \_\_asm("movl $0x0, %%edx;"

        "movl %2, %%eax;"

        "movl %3, %%ebx;"

        "idivl %%ebx;"

        : "=a" (va), "=d" (va2)

        : "g" (va), "g" (va2)

    );

    cout <<"El resultado de la division " << var1 << "/" << var2 << " en ensamblador es: " << va << endl;

}

//asm ("modl $0x0, %%edx;" "movl %2, %%eax;" "movl %3, %%ebx;" "idivl %%ebx;" : "=a" (Num1), "=d" (Num2) : "g" (Num1), "g" (Num2) );

void Modulo (int va, int va2){

    int var1 = va;

    int var2 = va2;

    \_\_asm("movl $0x0, %%edx;"

            "movl %2, %%eax;"

            "movl %3, %%ebx;"

            "idivl %%ebx;"

            : "=a" (va), "=d" (va2)

            : "g" (va), "g" (va2)

    );

    cout <<"El resultado del modulo " << var1 << "/" << var2 << " en ensamblador es: " << va2 << endl;

}

//asm ("fld %1;" "fsqrt;" "fstp %0;" : "=m" (result) : "m" (Num3) );

void SQRT(float n1){

    float result = 0;

    \_\_asm("fld %1;" "fsqrt;" "fstp %0;" : "=m" (result) : "m" (n1) );

    cout <<"El resultado de la raiz cuadrada de " << n1 << " en ensamblador es: " << result << endl;

}

//asm ("fld %1;" "fld %2;" "fldpi;" "fmul;" "fdiv;" "fsin;" "fstp %0;" : "=m" (result) : "m" (two\_right\_angles), "m" (degree) );

void SINX(float degree){

    float result, two\_right\_angles = 180.0f;

    \_\_asm("fld %1;"

            "fld %2;"

            "fldpi;"

            "fmul;"

            "fdiv;"

            "fsin;"

            "fstp %0;"

            : "=m" (result)

            : "m" (two\_right\_angles), "m" (degree)

    );

    cout <<"El resultado del seno de " << degree << " en ensamblador es: " << result << endl;

}

//asm ("fld %1;" "fld %2;" "fldpi;" "fmul;" "fdiv;" "fstp %0;" : "=m" (radians) : "m" (two\_right\_angles), "m" (degree) );

void COSX(float degree){

    float result, two\_right\_angles = 180.0f, radians;

    \_\_asm("fld %1;"

            "fld %2;"

            "fldpi;"

            "fmul;"

            "fdiv;"

            "fstp %0;"////

            : "=m" (radians)

            : "m" (two\_right\_angles), "m" (degree)

    );

    \_\_asm ( "fld %1;"

            "fcos;"

            "fstp %0;" : "=m" (result) : "m" (radians)

    );

    cout <<"El resultado del coseno de " << degree << " en ensamblador es: " << result << endl;

}

Analizador\_Sintactico.h

bool Ensamblador(){

int Num1, Num2;

    SepararCadena();

    if(cadenaEvaluar == "(" ){

        GenerarCadenaNueva ( cadenaEvaluar.length() );

        SepararCadena();

        GenerarCadenaNueva ( cadenaEvaluar.length() );

        SepararCadena();

        if( cadenaEvaluar == "addl" ){

            if(extraer\_variables()){

                Suma();

                return true;

            }

        }

        else if( cadenaEvaluar == "subl" ){

            if(extraer\_variables()){

                Resta();

                return true;

            }

        }

        else if( cadenaEvaluar == "imull" ){

            if(extraer\_variables()){

                Multiplicacion();

                return true;

            }

        }

        else if( cadenaEvaluar == "idivl" ){

            return division();

        }

        else if( cadenaEvaluar == "modl" ){

            return modulo();

        }

        else if( cadenaEvaluar == "fld" ){

            for(int i = 0 ; i <= 5 ; i++){

                funcion\_mamalona();

            }

            if(cadenaEvaluar == "fsqrt"){

                return sqrt();

            }

            else{

                for(int i = 0 ; i <= 17 ; i++){

                    funcion\_mamalona();

                }

                if( cadenaEvaluar == "fsin" ){

                    return sinx();

                }

                else if (cadenaEvaluar == "fstp"){

                    return cosx();

                }

            }

        }

        return false;

    }

}

bool sqrt(){

    string comilla = "\"";

    string sobras = "";

    float va,va2;

    bool fin = false;

    do{

        funcion\_mamalona();

        if(cadenaEvaluar == comilla){

            sobras += "-";

            funcion\_mamalona();

            sobras += "-";

        }

        else if(Encontrado2(cadenaEvaluar) != -1){

            sobras += "x";

            int variable\_ensamblador1 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

            va =  stof(datos[variable\_ensamblador1].valor);

            for (int i =0 ; i <= 2 ; i++){

                funcion\_mamalona();

                sobras += cadenaEvaluar;

            }

            fin = true;

            break;

        }else{

            sobras += cadenaEvaluar;

        }

    }while(fin != true);

    if((sobras == ";--fstp%0;----m--result):----x));")  ){

        funcion\_mamalona();

        SQRT(va);

        return true;

    }

    return false;

}

bool sinx(){

    string comilla = "\"";

    string sobras = "";

    float va,va2;

    bool fin = false;

    do{

        funcion\_mamalona();

        if(cadenaEvaluar == comilla){

            sobras += "-";

            funcion\_mamalona();

            sobras += "-";

        }

        else if(Encontrado2(cadenaEvaluar) != -1){

            sobras += "x";

            int variable\_ensamblador1 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

            va =  stof(datos[variable\_ensamblador1].valor);

            for (int i =0 ; i <= 2 ; i++){

                funcion\_mamalona();

                sobras += cadenaEvaluar;

            }

            fin = true;

            break;

        }else{

            sobras += cadenaEvaluar;

        }

    }while(fin != true);

    if((sobras == ";--fstp%0;----m--result):----two\_right\_angles),----x));")  ){

        funcion\_mamalona();

        SINX(va);

        return true;

    }

    return false;

}

bool cosx(){

    string comilla = "\"";

    string sobras = "";

    float va,va2;

    bool fin = false;

    do{

        funcion\_mamalona();

        if(cadenaEvaluar == comilla){

            sobras += "-";

            funcion\_mamalona();

            sobras += "-";

        }

        else if(Encontrado2(cadenaEvaluar) != -1){

            sobras += "x";

            int variable\_ensamblador1 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

            va =  stof(datos[variable\_ensamblador1].valor);

            for (int i =0 ; i <= 2 ; i++){

                funcion\_mamalona();

                sobras += cadenaEvaluar;

            }

            fin = true;

            break;

        }else{

            sobras += cadenaEvaluar;

        }

    }while(fin != true);

    if((sobras == "%0;----m--radians):----two\_right\_angles),----x));")  ){

        funcion\_mamalona();

        COSX(va);

        return true;

    }

    return false;

}

void funcion\_mamalona(){

    GenerarCadenaNueva ( cadenaEvaluar.length() );

    SepararCadena();

}

bool extraer\_variables(){

    string va1;

    funcion\_mamalona();

        if(cadenaEvaluar == "%" ){

            funcion\_mamalona();

            if(cadenaEvaluar == "%" ){

                funcion\_mamalona();

                if(cadenaEvaluar == "ebx" ){

                    funcion\_mamalona();

                    if(cadenaEvaluar == "," ){

                        funcion\_mamalona();

                        if(cadenaEvaluar == "%" ){

                            funcion\_mamalona();

                            if(cadenaEvaluar == "%" ){

                                funcion\_mamalona();

                                if(cadenaEvaluar == "eax" ){

                                    funcion\_mamalona();

                                    if(cadenaEvaluar == ";" ){

                                        funcion\_mamalona();

                                        funcion\_mamalona();

                                        if(cadenaEvaluar == ":" ){

                                            funcion\_mamalona();

                                            funcion\_mamalona();

                                            if(cadenaEvaluar == "=" ){

                                                funcion\_mamalona();

                                                if(cadenaEvaluar == "a" ){

                                                    funcion\_mamalona();

                                                    funcion\_mamalona();

                                                    if(cadenaEvaluar == "(" ){

                                                        funcion\_mamalona();

                                                        int variable\_ensamblador1 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

                                                        Num1 =  stoi(datos[variable\_ensamblador1].valor);

                                                        va1 = cadenaEvaluar;

                                                        if(variable\_ensamblador1 != -1){

                                                            funcion\_mamalona();

                                                            if(cadenaEvaluar == ")" ){

                                                                funcion\_mamalona();

                                                                if(cadenaEvaluar == ":" ){

                                                                    funcion\_mamalona();

                                                                    funcion\_mamalona();

                                                                    if(cadenaEvaluar == "a" ){

                                                                        funcion\_mamalona();

                                                                        funcion\_mamalona();

                                                                        if(cadenaEvaluar == "(" ){

                                                                            funcion\_mamalona();

                                                                            if(cadenaEvaluar == va1 ){

                                                                                funcion\_mamalona();

                                                                                if(cadenaEvaluar == ")" ){

                                                                                    funcion\_mamalona();

                                                                                    if(cadenaEvaluar == "," ){

                                                                                        funcion\_mamalona();

                                                                                        funcion\_mamalona();

                                                                                        if(cadenaEvaluar == "b" ){

                                                                                            funcion\_mamalona();

                                                                                            funcion\_mamalona();

                                                                                            if(cadenaEvaluar == "(" ){

                                                                                                funcion\_mamalona();

                                                                                                int variable\_ensamblador2 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

                                                                                                Num2 =  stoi(datos[variable\_ensamblador2].valor);

                                                                                                if(variable\_ensamblador2 != -1){

                                                                                                    funcion\_mamalona();

                                                                                                    if(cadenaEvaluar == ")" ){

                                                                                                        funcion\_mamalona();

                                                                                                        if(cadenaEvaluar == ")" ){

                                                                                                            funcion\_mamalona();

                                                                                                            if(cadenaEvaluar == ";" ){

                                                                                                                funcion\_mamalona();

                                                                                                                return true;

                                                                                                            }

                                                                                                        }

                                                                                                    }

                                                                                                }else{

                                                                                                    cout << "Error operacion asm invalida" << endl;

                                                                                                    return false;

                                                                                                }

                                                                                            }

                                                                                        }

                                                                                    }

                                                                                }

                                                                            }

                                                                            else{

                                                                                cout << "Error operacion asm invalida" << endl;

                                                                                return false;

                                                                            }

                                                                        }

                                                                    }

                                                                }

                                                            }

                                                        }else{

                                                            cout << "Error, no se ha definido la variable: " << cadenaEvaluar << endl;

                                                            return false;

                                                        }

                                                    }

                                                }

                                            }

                                        }

                                    }

                                }

                            }

                        }

                    }

                }

            }

        }

}

bool division(){

    string comilla = "\"";

    string sobras = "";

    string sobras2 = "";

    string var1="",var2="";

    string aux1="",aux2="";

    int va,va2;

    bool fin = false;

    int cont = 0;

    do{

        funcion\_mamalona();

        if(cadenaEvaluar == comilla){

            sobras += "-";

            funcion\_mamalona();

            sobras += "-";

        }

        else if(Encontrado2(cadenaEvaluar) != -1){

            sobras += "x";

            if(cont == 0){

                int variable\_ensamblador1 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

                va =  stoi(datos[variable\_ensamblador1].valor);

                var1 = datos[variable\_ensamblador1].id;

                cont++;

            }

            else if(cont == 1){

                int variable\_ensamblador2 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

                va2 =  stoi(datos[variable\_ensamblador2].valor);

                var2 = datos[variable\_ensamblador2].id;

                for(int i = 0 ; i <= 17 ; i++){

                    if(i==7 ){

                        aux1 = cadenaEvaluar;

                    }

                    if(i == 14){

                            aux2 = cadenaEvaluar;

                    }

                    sobras2 += cadenaEvaluar;

                    funcion\_mamalona();

                }

                fin = true;

                break;

            }

        }else{

            sobras += cadenaEvaluar;

        }

    }while(fin != true);

    if((var1 == aux1) &&  (var2 == aux2)  && (sobras == "$0x0,%%edx;--movl%2,%%eax;--movl%3,%%ebx;--idivl%%ebx;----a--x),--d--x")  ){

        Division(va,va2);

        return true;

    }

    return false;

}

bool modulo(){

    string comilla = "\"";

    string sobras = "";

    string sobras2 = "";

    string var1="",var2="";

    string aux1="",aux2="";

    int va,va2;

    bool fin = false;

    int cont = 0;

    do{

        funcion\_mamalona();

        if(cadenaEvaluar == comilla){

            sobras += "-";

            funcion\_mamalona();

            sobras += "-";

        }

        else if(Encontrado2(cadenaEvaluar) != -1){

            sobras += "x";

            if(cont == 0){

                int variable\_ensamblador1 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

                va =  stoi(datos[variable\_ensamblador1].valor);

                var1 = datos[variable\_ensamblador1].id;

                cont++;

            }

            else if(cont == 1){

                int variable\_ensamblador2 = Encontrado2(cadenaEvaluar);//guardamos el id de la primera variable

                va2 =  stoi(datos[variable\_ensamblador2].valor);

                var2 = datos[variable\_ensamblador2].id;

                for(int i = 0 ; i <= 17 ; i++){

                    if(i==7 ){

                        aux1 = cadenaEvaluar;

                    }

                    if(i == 14){

                            aux2 = cadenaEvaluar;

                    }

                    sobras2 += cadenaEvaluar;

                    funcion\_mamalona();

                }

                fin = true;

                break;

            }

        }else{

            sobras += cadenaEvaluar;

        }

    }while(fin != true);

    if((var1 == aux1) &&  (var2 == aux2)  && (sobras == "$0x0,%%edx;--movl%2,%%eax;--movl%3,%%ebx;--idivl%%ebx;----a--x),--d--x")  ){

        Modulo(va,va2);

        return true;

    }

    return false;

}

# Acrónimo

Cocomo -> COnstructive COst MOdel

# Requisitos funcionales

* El diagrama debe leer la entrada de los datos de usuario incluyendo las operaciones de asm que interactúan directo con ensamblador
* El programa validará lo sintáctico y lo semántico.
* En caso de que, si se encuentre errores, arrojara un mensaje de sintaxis invalido, haciendo saber al usuario que debe verificar su código.

# Requisitos no funcionales

* Limpieza de pantalla automáticamente
* Respuesta de resultado de las operaciones en pantalla

# Complejidad ciclomática

𝑀=𝐸−𝑁+2𝑃

M = 85 – 89 + 2

M = 6

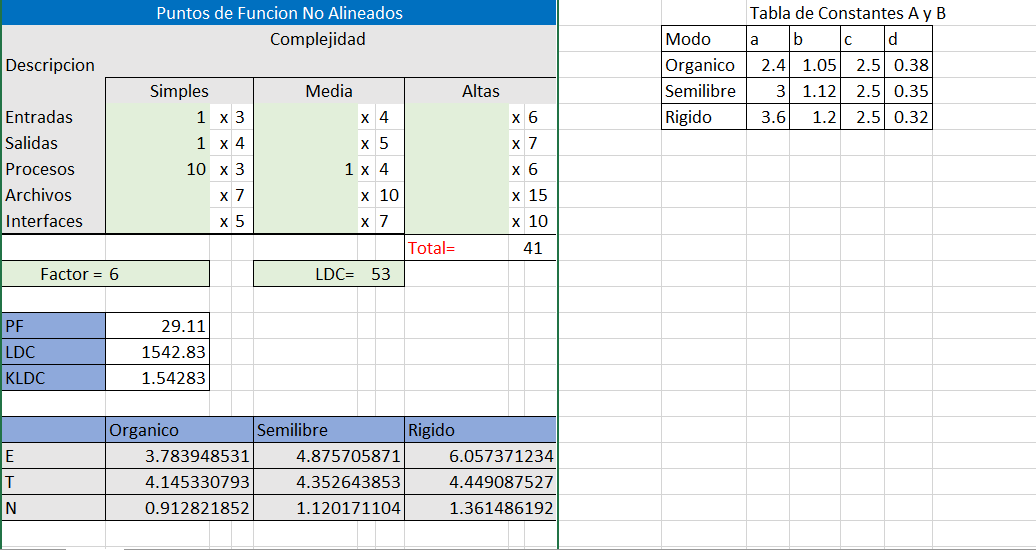
M = Complejidad Ciclomatica

E = Numero de Aristas del grafo

N = Numero de Nodos

P = Numero de Nodos conexos, Nodos de salida

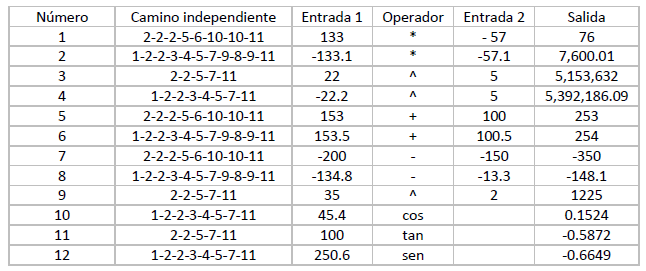
# COCOMO



# Caja negra



# Caja blanca

****

# Grafos









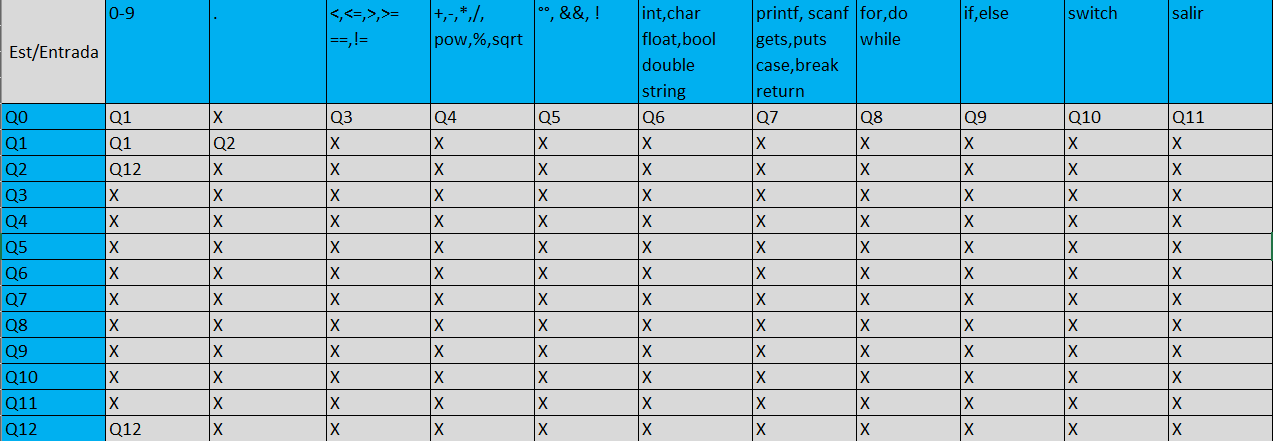








# Tabla de transiciones



Diccionario de Datos

**int:** Palabra reservada para indicar que es un tipo de dato entero.

**float:** Palabra reservada para indicar que es un tipo de dato flotante real.

**bool:** Palabra reservada para indicar que es un tipo de dato booleano.

**string**: Palabra reservada para indicar que es un tipo de dato cadena.

**char:** Palabra reservada para indicar que es un tipo de dato caracter.

**switch:** Palabra reservada para indicar que es una estructura selectiva.

**case:** Palabra reservada para indicar que es un caso de la estructura switch.

**default:** Palabra reservada para indicar que es un caso por defecto de la estructura switch.

**break:** Palabra reservada para indicar cuando se debe romper un ciclo o iteración.

**if:** Palabra reservada para indicar que es una estructura selectiva.

**else:** Palabra reservada para indicar que es una estructura selectiva.

**do:** Palabra reservada para indicar que es una estructura iterativa.

**while:** Palabra reservada para indicar que es una estructura iterativa.

**for:** Palabra reservada para indicar que es una estructura iterativa.

**void:** Palabra reservada para indicar que es una función que no retorna valor.

**class:** Palabra reservada para indicar que es una estructura del tipo clase.

**cout:** Palabra reservada para mostrar datos en pantalla.

**cin:** Palabra reservada para recibir datos ingresados del usuario desde pantalla.

**include:** Palabra reservada para incluir cabeceras o librerías.

**return:** Palabra reservada para retornar un parámetro.

**{ Caracter del sistema apertura de llave para determinar bloque de código.**

**} Caracter del sistema cerradura de llave para determinar bloque de código.**

**[ Caracter del sistema apertura de corchetes para delimitar un arreglo.**

**] Caracter del sistema cerradura de corchetes para delimitar un arreglo.**

**( Caracter del sistema apertura de paréntesis, para llamar una función.**

**) Caracter del sistema cerradura de paréntesis, para llamar una función.**

**#** Caracter del sistema que se utiliza con la palabra include.

**, Caracter del sistema que se utiliza para declarar distintas variables.**

**. Caracter del sistema que se utiliza para declarar números flotantes.**

**: Caracter del sistema para utilizarlo después de la palabra reservada case.**

**;** Caracter del sistema para finalizar una línea de código.

**/ Operador aritmético utilizado para la división.**

**\* Operador aritmético utilizado para la multiplicación.**

**- Operador aritmético utilizado para la resta.**

**+ Operador aritmético utilizado para la suma.**

**%** Operador aritmético utilizado para el residuo.

**& Operador lógico utilizado para la operación and.**

**| Operador lógico utilizado para la operación or.**

**< Operador condicional utilizado para identificar al dato menor.**

**>** Operador condicional utilizado para identificar al dato mayor.

**= Operador de asignación utilizado para asignar un valor a alguna variable.**

**! Operador condicional se utiliza para identificar si un dato es diferente de otro.**

# Conclusión:

Durante el curso siempre trabajamos con el lenguaje de programación C++, pero al llegar a esta parte tuvimos que dar un cambio un poco extraño ya que esta vez ocupábamos implementar funciones de programación en lenguaje ASM y eso fue un poco raro porque a como habíamos implementado todo el programa necesitábamos arreglar muchas cosas del código ya que no se esperaba ingresar nada de lenguaje ensamblador, pero una vez implementado de la manera correcta logramos ejecutar las funciones en asm para realizar las diversas operaciones en nuestro lenguaje de programación y con esto obtuvimos exitosamente la actividad solicitada.

# Bibliografía

Gálvez, S.. (2020). Tema 2. Análisis lexicográfico. Traductores,

Compiladores e Intérpretes(pp.1-15). Jalisco, México : UMA.

Aho, A.V., Sethi, R., Ullman, J.D. (1990), Compiladores: principios, técnicas y herramientas, Tema 3, p´aginas: 85-158.

Louden, K.C. (1997), Compiler Construction: Principles and Practice, Tema 2, paginas: 31-93.