****

**Instituto Tecnológico de Costa Rica**

**Curso: IC-3101 Arquitectura de Computadores**

**I Semestre 2015**

**Tarea Programada #2:**

**Evaluador de expresiones matemáticas**

**Profesor:**

**Erick Hernández Bonilla**

**Alumnos:**

**Isaac Campos Mesén 2014004626**

**Melissa Molina Corrales 2013006074**

**Edwin Rees Sandí**

**Fecha de entrega:**

**18-05-2015**

Índice

Propósito y Descripción………………………………………………………………………………………..3

Buffers a utilizar para el almacenamiento de los datos………………………………………….4

Procesamiento de la entrada………………………………………………………………………………..5

Validación de la expresión matemática ingresada…………………………………………………6

Sustitución de variables……………………………………………………………………………………….9

Operaciones……………………………………………………………………………………………………….12

Manejo de strings e integers……………………………………………………………………………….14

Resolución de la expresión matemática……………………………………………………………….16

Propósito y Descripción

El objetivo de este proyecto es crear un software que permita evaluar expresiones matemáticas, con el fin de facilitar la manera en que se resuelven estas funciones al usuario, este software deberá leer una expresión matemática expresada con incógnitas, junto con los valores de esas incógnitas, para así poder sustituir los valores en la expresión matemática leída y mostrar paso a paso su resolución.

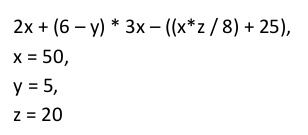


Figura 1. Expresión matemática junto con las incógnitas.

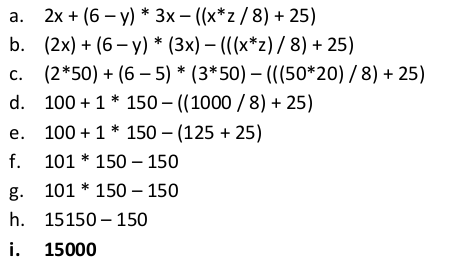
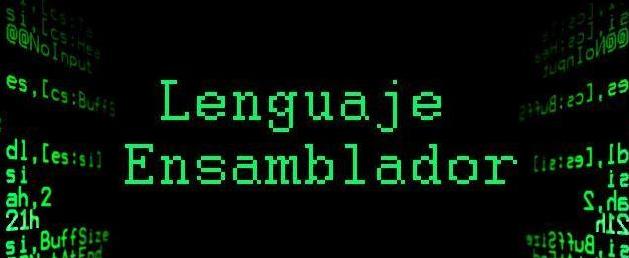


Figura 2. Sustitución y resolución paso a paso de la expresión matemática.

Este software será programado en YASM (Ensamblador) en la sintaxis de Intel x64.

El lenguaje ensamblador es el lenguaje de [programación](http://www.ecured.cu/index.php/Programaci%C3%B3n) utilizado para escribir [programas informáticos](http://www.ecured.cu/index.php/Programas_inform%C3%A1ticos) de bajo nivel, y constituye la representación más directa del [Código máquina](http://www.ecured.cu/index.php/C%C3%B3digo_m%C3%A1quina)  para cada arquitectura de computadoras legible por un programador.

Buffers a utilizar para el almacenamiento de los datos

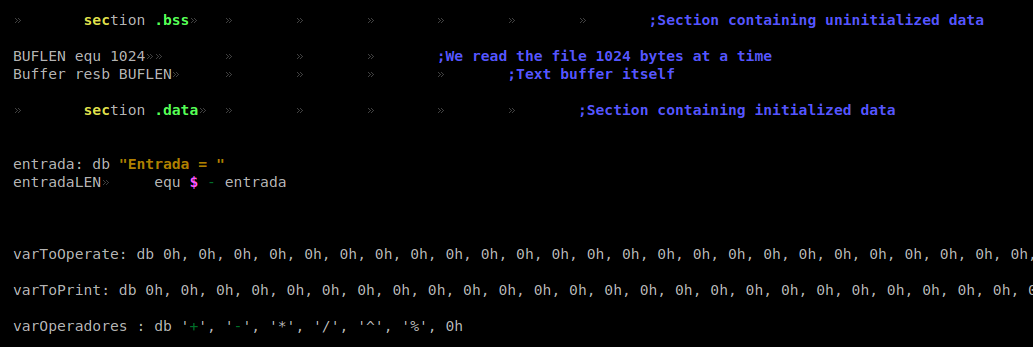
Para el almacenamiento de los datos, se define un buffer en donde leemos el archivo de 1024 bytes a la vez.

Para recibir la expresión matemática se define un buffer, llamado **entrada**, en esta se almacenará la expresión matemática junto con las variables a sustituir en la expresión.

Para almacenar la expresión matemática leída y realizar sobre esta las operaciones correspondientes, se utilizará el buffer **varToOperate**.

Para ir almacenando el resultado de las operaciones que se vayan realizando, se utilizará el buffer **varToPrint**, la cual también nos permitirá ir imprimiendo el resultado en pantalla. Asimismo para almacenar los operadores y poderlos utilizar en la resolución de la expresión matemática se usará el buffer **varOperadores**.

Importante mencionar que los buffers **varToOperate** y  **varToPrint**, comenzarán con caracteres nulos.



**Buffer**

**Operadores**

**Donde se realizarán las operaciones correspondientes para la resolución de la expresión.**

**Donde se imprimirán los resultados**

Procesamiento de la entrada

Al decir procesamiento de la entrada, se refiere a la manera en que se almacenará la expresión que el software recibirá, una vez haya hecho las validaciones correspondientes. Para esto se utilizará un procedimiento llamado **procesarEntrada**, el cual recorre byte por byte la entrada hasta la primera coma, esto porque después de la coma siguen las variables a sustituir en la expresión y lo que necesitamos almacenar es solo la expresión para trabajar en ella, así este procedimiento almacena la expresión en el buffer **varToOperate.**

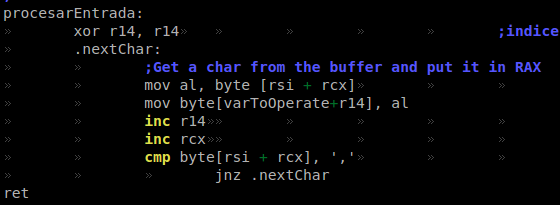


Figura 3. Procedimiento para procesar la entrada



Figura 4. Forma en que la expresión es almacenada en varToOperate.

Validación de la expresión matemática ingresada

Manejo de errores

En el caso de la operación multiplicación hay que tomar en cuenta, que esta puede estar tácita por ejemplo:

Y en el caso de las variables que estén unidas **xy** la multiplicación se hace explicita **x \* y**, sino esta se tomaría como una sola variable, para poder solucionar este problema se plantea un procedimiento el cual recorrerá el buffer  **varToOperate**, donde estará almacenada la expresión y pondrá un asterisco en el lugar correspondiente.



Figura 5. Procedimiento para poner asteriscos en el caso de la operación multiplicación.

Como complemento a este procedimiento se utilizarán otros procedimientos: **isOperador**, el cual identifica si el carácter que está en **varToOperate**  es un operador.



Figura 6. Procedimiento para poner identificar si un carácter es un operador.

**validarPonerAsteriscos**, el cual se encarga de validar los casos en donde no es necesario poner asteriscos en la expresión y en caso de que se necesiten, este los agregará a la expresión matemática en el lugar correspondiente.

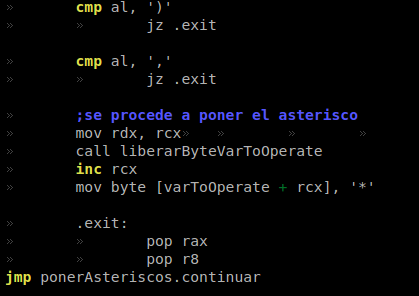
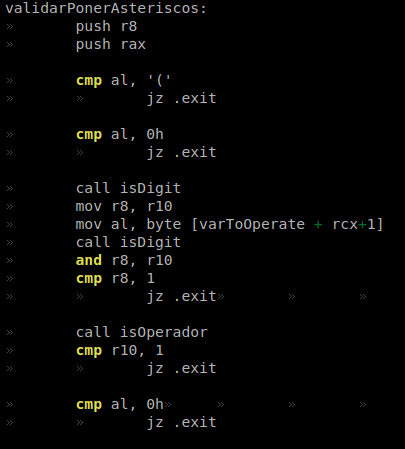


Figura 7. Procedimiento validarPonerAsteriscos

Sustitución de variables

Para la sustitución de variables en la expresión matemática, se planteó de la siguiente manera:

Se utilizará un índice **(RCX)**, para recorrer el buffer de entrada después de la coma, y así se realizará una comparación con el buffer **varToOperate**, de esta manera se irá recorriendo y comparando las variables del buffer de entrada con las variables del buffer **varToOperate**, así en donde estas coincidan se sustituirá por su valor en la expresión.

Para su implementación se plantearon diversos procedimientos:

El procedimiento **cambiarVariables**, el cual irá recorriendo el buffer de entrada donde están las variables, hasta que haya llegado al fin de línea y ya no hayan variables por recorrer.

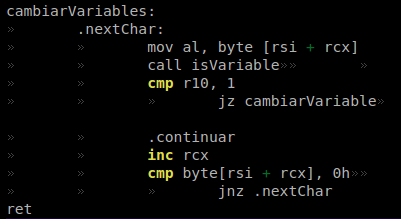


Figura 8. Procedimiento cambiarVariables

Como complemento a este procedimiento se utilizará el procedimiento **isVariable**, el cual se encargará de identificar si un carácter en el buffer es una variable.

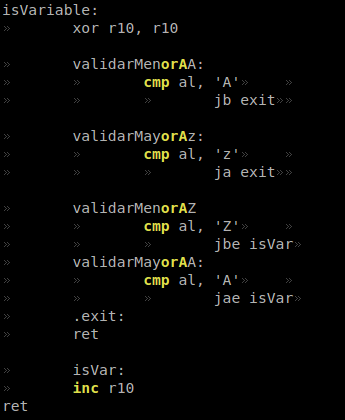


Figura 9. Procedimiento isVariable

El procedimiento **cambiarVariable**, recorrerá el buffer **varToOperate** y sustituirá las variables con el valor correspondiente.

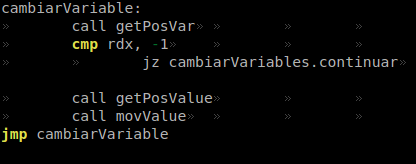


Figura 10. Procedimiento cambiarVariable

Para obtener la posición en el buffer de entrada de la variable a sustituir en el buffer **varToOperate** se implementa el procedimiento **getPosVar**, este buscará la variable y en el caso de que esta no este retornará -1.

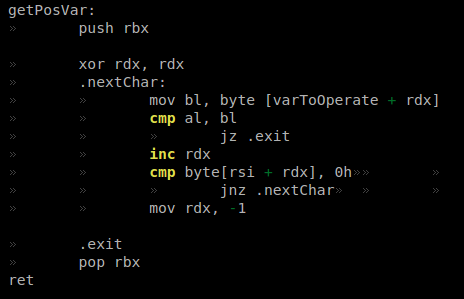


Figura 11. Procedimiento getPosVar

Asimismo para obtener el valor de las variables a sustituir se usará el procedimiento **getPosValue**.

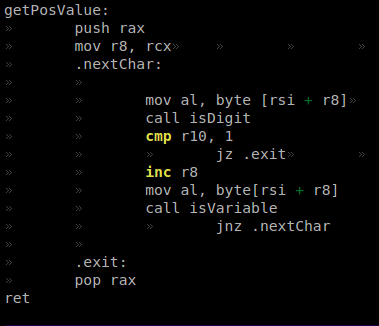


Figura 12. Procedimiento getPosValue

Operaciones

Para la implementación de las operaciones se manejará de la siguiente forma:

Se crean procedimientos para realizar las operaciones básicas suma, resta, multiplicación y división, además de las operaciones de módulo y exponente.

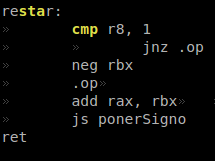


Figura 14. Procedimiento para restar.

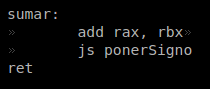


Figura 13. Procedimiento para sumar.

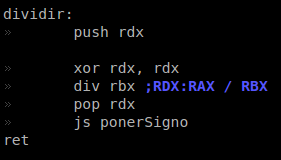


Figura 16. Procedimiento para dividir.

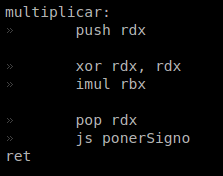


Figura 15. Procedimiento para multiplicar.

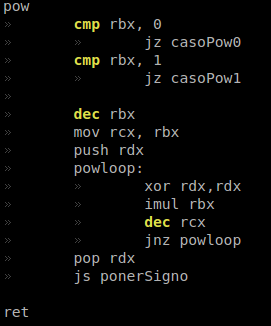
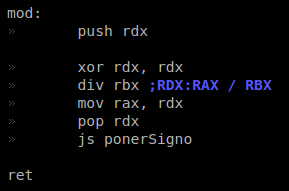


Figura 17. Procedimiento para realizar la operación de exponente.

Figura 16. Procedimiento para realizar la operación de módulo.



Para el manejo de los números negativos en las operaciones se implementó el procedimiento **ponerSigno**, el cual si el resultado de la operación es con signo hace complemento a 2 y pone el signo de menos en la variable, donde **neg** hace el complemento a 2.

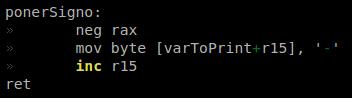


Figura 18. Procedimiento para ponerle signo al resultado de una operación.

Manejo de strings e integers

Una parte importante en la realización de este software es el manejo de los strings y de los integers, ya que al recibir la entrada, cada carácter ingresado la consola lo estará manejando como string y por tanto en el momento de almacenamiento necesitamos que los números ingresados se conviertan a integer para poder realizar operaciones sobre ellos para resolver la expresión matemática ingresada y poder mostrar el resultado en pantalla como string.

Para solucionar este problema se implementó dos algoritmos uno que convierte de string a integer y otro que convertirá el resultado de integer a string

Atoi

Este procedimiento se encarga de convertir todo lo que sea string a integer.

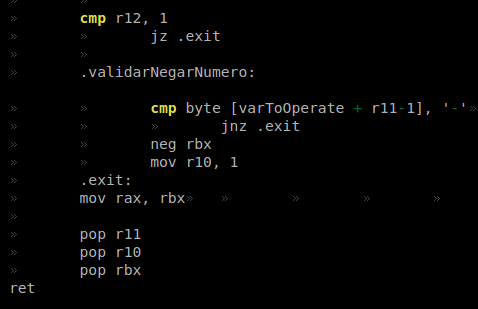
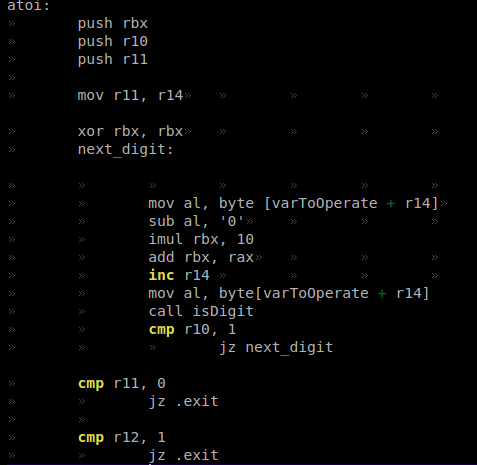


Figura 19. Atoi

Itoa

Este procedimiento se encarga de convertir todo lo que sea integer a string para poder mostrar el resultado en la pantalla.

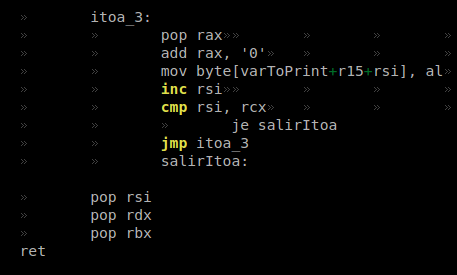
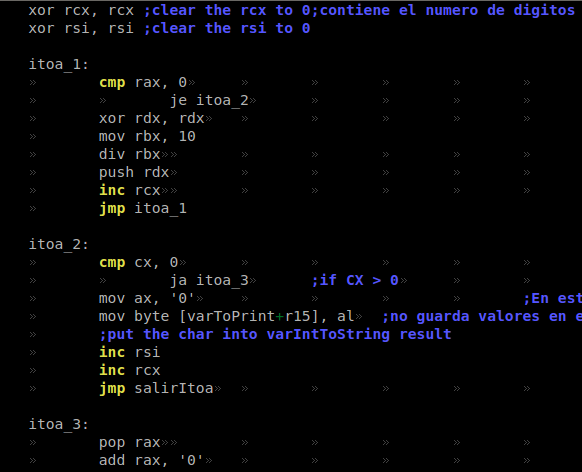


Figura 20. Itoa

Resolución de la expresión matemática

Para la resolución de la expresión una vez implementados todos los procedimientos anteriores el algoritmo se efectuará de la siguiente forma:

1. En el registro **(R9)** , se pasará la operación que se desee efectuar, así ira iterando **varToOperate** hasta el final de este buffer e irá haciendo las operaciones en este orden:

* División
* Multiplicación
* Suma
* Resta

1. Esto estará funcionando como un ciclo, dentro del cual se estará llamando al procedimiento **quitarParentesis**, el funcionamiento de este ciclo sería de la siguiente forma:

* Limpia el buffer **varToPrint**, que se utilizará para guardar el resultado de las operaciones y la que nos permitirá imprimir en pantalla.
* Busca la operación a realizar.
* Realiza la operación.
* Ajusta la variable **varToPrint**, es decir se le agregan los bytes del buffer **varToOperate** siguientes a la operación.
* Se quitan los paréntesis innecesarios
* Se imprime lo que hay en **varToPrint**
* Se pasan todos los datos que este en **varToPrint** a **varToOperate**

Este ciclo se irá repitiendo por cada operador que haya en la expresión ingresada hasta que ya no quede ninguna operación por resolver.

Obtener Operandos

Para poder realizar las operaciones básicas e ir resolviendo paso a paso la expresión, se necesita obtener los operandos y saber el operador para poder realizar la operación deseada. Para esto se implementan procedimientos que nos permiten obtener los operandos y dependiendo del operador llamar a los procedimientos de las operaciones para efectuar dicha operación.

Lo que se llevará a cabo para obtener los operandos es un procedimiento donde podamos recorrer la variable **varToOperate** y obtener el inicio del primer operando y el inicio del segundo operando, con el fin de guardar estos operandos en los registros **RAX**  y **RBX** para luego pasarle estos operandos a los procedimientos de las operaciones.

**¿Cómo hacemos esto?**

Bien creamos un procedimiento llamado **buscarOperacion**, el cual busca el inicio del primer operando y va incrementándose hasta encontrarse con un operador para luego llamar a otro procedimiento **ObtenerOperandos** y decrementar el índice hasta encontrar el inicio del primer operando.

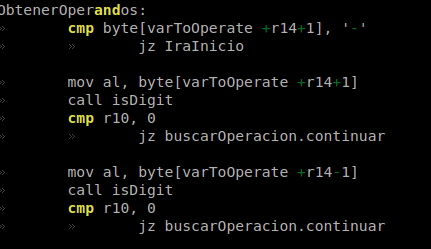


Figura 22. ObtenerOperandos

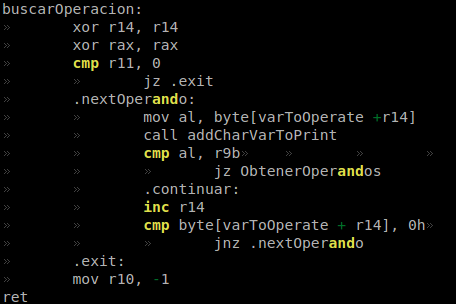


Figura 21. buscarOperacion

Así una vez tengamos los operandos, el procedimiento **realizar Operación** determinará cual operación es la que se desea realizar y llamará al procedimiento correspondiente para resolver la operación.

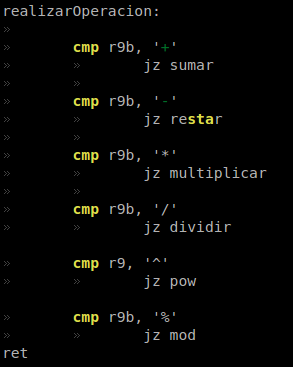


Figura 23. realizarOperacion