




MANUAL DE USUARIO



PRIMER PROYECTO DE LABORATORIO
ORGANIZACIÓN DE COMPILADORES Y LENGUAJES 2
SERGIO FERNANDO OTZOY GONZALEZ

CONTENIDO

Objetivos	2
Generales	2
Específicos	2
Alcance	2
Panorama general de la aplicación	2
Sintaxis.....	3

OBJETIVOS

GENERALES

1. Dar al usuario una guía breve sobre el uso del lenguaje débil tipado augus
2. Explicar la utilidad y limitaciones del lenguaje

ESPECÍFICOS

1. Explicar los comandos aceptados por el lenguaje August
2. Explicar los reportes generados y su ubicación
3. Explicar el flujo de operaciones para operaciones varias

ALCANCE

Este manual de usuario está dirigido a cualquier persona que, con un propósito didáctico, desea observar entender el funcionamiento del intérprete del lenguaje August. Es deseable conocimientos sobre compiladores, aunque no es necesario.

Este manual no ahonda en la forma en la que el intérprete funciona, si no en como interactuar con la aplicación que contiene el lenguaje de forma embebida. Se explica los comandos aceptados y la estructura convencional del lenguaje.

PANORAMA GENERAL DE LA APLICACIÓN

La aplicación consiste en una IDE para el lenguaje August. Este lenguaje está embebido a la aplicación. A través de la aplicación se puede abrir, guardar y crear archivos de código augus (*.aug), se puede también realizar las operaciones usuales de un IDE, tal como la ejecución del código, el reporte de errores y debugger.

La aplicación genera reportes automáticamente luego de finalizar una ejecución de código.

SINTAXIS

4. Sintaxis de Augus

La sintaxis de Augus es similar al lenguaje PHP, pero con la salvedad de solo utilizar instrucciones simples, además de tener solamente las operaciones básicas que determina un lenguaje ensamblador similar a MIPS. El manejo de apuntadores también se basa en el comportamiento de PHP y no de C.

4.1. Definición de registros (variables)

\$t0..\$tn	Temporales
\$a0..\$an	Parámetros
\$v0..\$vn	Valores devueltos por funciones
\$ra	Simulador de dirección de retorno por nivel
\$s0..\$sn	Pilas
\$sp	Puntero de la pila

En los espacios de registros de las siguientes instrucciones pueden ser ocupados por constantes de cualquier tipo.

4.2. Instrucciones simples y unarias:

main:	Inicio del programa. (obligatorio)
label:	Definición del inicio de una etiqueta.
goto label;	Salto incondicional hacia una etiqueta.
\$t1 = 10;	Asignación numérica.
\$t1 = 'hola'; \$t1 = "hola";	Asignación de una cadena de caracteres.
\$t1 = \$t2;	Copia simple.
\$t1 = - \$t2;	Negativo.
\$t1 = &\$t2;	\$t1 es un puntero a la dirección de \$t2.
unset(\$t1);	Destruye la variable \$t1.
print(\$t1);	Imprime en pantalla el contenido de \$t1. Solo se pueden imprimir registros no operaciones.
\$t1 = read();	Lee la entrada del teclado queda en \$t1.
#comment	Comentario de una sola línea.
exit;	Finaliza la ejecución. (opcional)

4.1. Conversiones:

\$t1 = (int) \$t2;	Si \$t2 tiene decimales son eliminados. Si \$t2 es un carácter se toma su código ASCII. Si \$t2 es una cadena se toma el ASCII del primer carácter. Si \$t2 es un arreglo se aplica al primer elemento las reglas anteriores.
\$t1 = (float) \$t2;	Si \$t2 es entero se agrega ".0". Si \$t2 es un carácter se toma su código ASCII con decimal. Si \$t2 es una cadena se toma el ASCII del primer carácter más el decimal. Si \$t2 es un arreglo se aplica al primer elemento las reglas anteriores.
\$t1 = (char) \$t2;	Si \$t2 es un número de 0 a 255 se convierte en el carácter basado en ASCII. Si \$t2 es un número mayor a 255 entonces se aplica el módulo 256 para extraer el ASCII. Si \$t2 es un decimal, se quitan los decimales y se aplican las reglas anteriores. Si \$t2 es una cadena, se almacena solo el primer carácter. Si \$t2 es un arreglo, se almacena solo el primer valor aplicando las reglas anteriores.

4.2. Instrucciones aritméticas:

\$t1 = \$t2 + \$t3;	Suma. Solamente si \$t2 y \$t3 son cadenas entonces se concatena.
\$t1 = \$t2 - \$t3;	Resta.
\$t1 = \$t2 * \$t3;	Multiplicación.
\$t1 = \$t2 / \$t3;	División.
\$t1 = \$t2 % \$t3	Residuo.
\$t1 = abs(\$t2);	Valor absoluto.

4.3. Instrucciones lógicas:

\$t1 = !\$t2;	Not, si \$t2 es 0 \$t1 es 1, si \$t2 es 1 \$t1 es 0.
\$t1 = \$t2 && \$t3;	And, 1 para verdadero, 0 para falso.
\$t1 = \$t2 \$t3;	Or, 1 para verdadero, 0 para falso.
\$t1 = \$t2 xor \$t3;	Xor, 1 para verdadero, 0 para falso.

4.4. Instrucciones bit a bit:

\$t1 = ~\$t2;	Not.
\$t1 = \$t2 & \$t3;	And.
\$t1 = \$t2 \$t3;	Or.
\$t1 = \$t2 ^ \$t3;	Xor.
\$t1 = \$t2 << \$t3;	Shift de \$t2, \$t3 pasos a la izquierda.
\$t1 = \$t2 >> \$t3;	Shift de \$t2, \$t3 pasos a la derecha.

4.5. Instrucciones relacionales:

\$t1 = \$t2 == \$t3;	\$t1 = 1 si \$t2 es igual a \$t3, sino 0.
\$t1 = \$t2 != \$t3;	\$t1 = 1 si \$t2 no es igual a \$t3, sino 0.
\$t1 = \$t2 >= \$t3;	\$t1 = 1 si \$t2 es mayor o igual a \$t3, sino 0.
\$t1 = \$t2 <= \$t3;	\$t1 = 1 si \$t2 es menor o igual a \$t3, sino 0.
\$t1 = \$t2 > \$t3;	\$t1 = 1 si \$t2 es mayor a \$t3, sino 0.
\$t1 = \$t2 < \$t3;	\$t1 = 1 si \$t2 es menor a \$t3, sino 0.

4.6. Arreglos, cadenas y structs

Hay dos tipos de arreglos: numérico y asociativo.

El numérico funciona como los arreglos convencionales, por ejemplo, asignar en el índice 4 del arreglo el valor uno: \$t1[4] = 1;. El asociativo funciona como un struct de C, o como una clase en C++ (solo datos), por ejemplo, \$t1 es el registro con dos componentes uno llamado nombre de tipo cadena y otro llamado edad de tipo numérico: \$t2['nombre'] = 'carlos'; \$t2['edad'] = 20;.

Las cadenas de caracteres también son consideradas arreglos.

\$t1 = array();	Define \$t1 como un arreglo o un struct, para diferenciarlos se utiliza ya sea el valor numérico o el nombre asociativo.
\$t1[4] = 1;	Asignación de un valor numérico (1) a un índice del arreglo (4).
\$t1['nombre'] = 'carlos'; \$t1["nombre"] = "carlos";	Asignación de un valor cadena (carlos) a un componente del struct (nombre).
\$t1 = \$t1[4];	Acceso a un índice del arreglo.
\$t1 = \$t2['nombre'];	Acceso a un componente del struct.
\$t1 = 'hola'; print(\$t1[0]); #imprime h	Acceder a un carácter de una cadena.

4.7. Instrucciones de control:

if (\$t1) goto label;	Salto condicional, si \$t1 es 1 salto, sino sigue la siguiente instrucción, \$t1 puede ser cualquier tipo de instrucción simple . Las estructuras de control se implementan utilizando este salto condicional.
-----------------------	---