Práctica 4: ANÁLISIS SEMÁNTICO Y GENERACIÓN DE CÓDIGO

Objetivo de la práctica

El objetivo de esta práctica es finalizar la construcción del compilador para el lenguaje de programación **ALFA**. Para ello, se debe tomar como punto de partida el analizador sintáctico desarrollado en la práctica anterior. El compilador final deberá traducir programas escritos en lenguaje **ALFA** a sus equivalentes en ensamblador, es decir la entrada al compilador será texto que contenga un programa **ALFA** y la salida del compilador será texto que contenga instrucciones en lenguaje ensamblador **NASM**.

Descripción de la semántica de ALFA

En este apartado se describe la semántica del lenguaje **ALFA** agrupando temáticamente las restricciones semánticas.

Cualquier duda respecto a la resolución de algún aspecto de la semántica del lenguaje **ALFA**, será resuelta por el profesor de prácticas.

Declaración y uso de identificadores

Las restricciones semánticas que afectan a la declaración y el uso de los identificadores son las siguientes:

- Las variables globales, las locales y las funciones deben ser definidas antes de ser utilizadas.
- Los parámetros de las funciones se definen en la propia cabecera de la función.
- Las variables globales, las locales, las funciones y sus parámetros deben ser únicos dentro de su ámbito de aplicación.

Expresiones lógicas

Las restricciones semánticas relativas a las expresiones lógicas son las siguientes:

- En las expresiones lógicas sólo pueden aparecer datos de tipo lógico. Por lo tanto, todas las subexpresiones, variables y constantes empleadas en una expresión lógica tienen que ser de ese tipo.
- Como puede observarse en la gramática de **ALFA**, el lenguaje dispone de los siguientes operadores:
    - disyunción
    - conjunción
    - negación

Expresiones aritméticas

En relación a las expresiones aritméticas, las restricciones semánticas son las siguientes:

- En las expresiones aritméticas sólo pueden aparecer datos de tipo numérico. Por lo tanto, todas las subexpresiones, variables y constantes empleadas en una expresión aritmética tienen que ser de ese tipo.
- Como puede observarse en la gramática de **ALFA**, los operadores binarios disponibles para operaciones aritméticas son los siguientes:
    - suma
    - resta
    - multiplicación
    - división
- El operador unario disponible para operaciones aritméticas es el cambio de signo.

Expresiones de comparación

Las restricciones semánticas que afectan a las expresiones de comparación son:

- Las comparaciones sólo pueden operar con datos de tipo numérico y el resultado de la comparación es de tipo lógico.
- Los operadores disponibles en **ALFA** son los siguientes:
    - igualdad de operandos
    - desigualdad de operandos
    - el primer operando menor o igual que el segundo
    - el primer operando mayor o igual que el segundo
    - el primer operando menor que el segundo
    - el primer operando mayor que el segundo

Asignaciones

Las asignaciones válidas son aquellas en las que las partes izquierda y derecha son del mismo tipo.

Vectores

Las variables de tipo vector tienen la semántica habitual de los lenguajes de programación con las siguientes peculiaridades:

- Sólo pueden contener datos de tipo básico (no existen vectores de vectores).
- Sólo son de una dimensión.

- El tamaño de los vectores no podrá exceder nunca el valor de 64.
- Para acceder a los elementos de los vectores se utilizará cualquier expresión de tipo entero. Esta expresión debe tener un valor entre 0 y el tamaño definido para el vector menos 1 (ambos incluidos).
- Tras la declaración de una variable de tipo vector, ésta aparecerá siempre indexada, y se utilizará de la misma manera que cualquier otro objeto que pueda ocupar su misma posición.

Estructuras de control de flujo de programa iterativas y condicionales

La semántica de las estructuras de control de flujo de programa iterativas (while) y condicionales (if e if-else) es similar a la de otros lenguajes de programación de alto nivel.

Operaciones de entrada/salida

La operación de entrada lee datos escalares y los almacena en variables. Está contemplada la lectura de datos enteros y lógicos.

La operación de escritura de datos de tipo escalar trabaja con expresiones de tipo lógico o numérico.

Funciones

Las funciones se rigen por las siguientes restricciones semánticas:

- Sólo se permiten funciones con retorno de tipos básicos (lógico o numérico)
- Los parámetros de las funciones sólo pueden ser de tipos básicos (lógico o numérico)
- Las variables locales de las funciones sólo pueden ser de tipo básico (lógico o numérico)
- En las sentencias de llamadas a funciones, sólo es necesario comprobar la corrección del número de argumentos. No es necesario realizar ninguna comprobación de la correspondencia de tipos.
- En las llamadas a funciones, los parámetros actuales no pueden ser llamadas a otras funciones.
- Una sentencia de retorno de función solamente debe aparecer en el cuerpo de una función.
- En el cuerpo de una función obligatoriamente tiene que aparecer al menos una sentencia de retorno.
- En una sentencia de retorno el tipo de la expresión debe de coincidir con el tipo de retorno de la función.

Gestión de errores semánticos

Los errores semánticos se informarán en la salida estándar con el siguiente formato:

******Error semantico en lin <n° línea>: <mensaje>**

A continuación se presenta un programa de ejemplo con un error semántico en la línea 4: la variable "y" no ha sido declarada.

```
main
  {
    int x ;
    printf y ;
  }
```

El compilador debe informar de este error semántico con el siguiente mensaje:

`****Error semantico en lin 4:` Acceso a variable no declarada (y).

A continuación se muestra la lista de los errores semánticos que debe detectar el compilador y sus correspondientes mensajes (se puede observar que los mensajes no contienen ni eñes ni acentos):

****Error semantico en lin X: Declaracion duplicada.

****Error semantico en lin X: Acceso a variable no declarada (<nombre_variable>).

****Error semantico en lin X: Operacion aritmetica con operandos boolean.

****Error semantico en lin X: Operacion logica con operandos int.

****Error semantico en lin X: Comparacion con operandos boolean.

****Error semantico en lin X: Condicional con condicion de tipo int.

****Error semantico en lin X: Bucle con condicion de tipo int.

****Error semantico en lin X: Numero incorrecto de parametros en llamada a funcion.

****Error semantico en lin X: Asignacion incompatible.

****Error semantico en lin X: El tamanyo del vector <nombre_vector> excede los limites permitidos (1,64).

****Error semantico en lin X: Intento de indexacion de una variable que no es de tipo vector.

****Error semantico en lin X: El indice en una operacion de indexacion tiene que ser de tipo entero.

****Error semantico en lin X: Funcion <nombre_funcion> sin sentencia de retorno.

****Error semantico en lin X: Sentencia de retorno fuera del cuerpo de una función.

****Error semantico en lin X: No esta permitido el uso de llamadas a funciones como parametros de otras funciones.

****Error semantico en lin X: Variable local de tipo no escalar..

Generación de código

El compilador debe traducir los programas válidos escritos en **ALFA** a los correspondientes programas en ensamblador **NASM** directamente, es decir, no se utilizará en la práctica ninguna representación intermedia.

Esta generación de código se realizará mediante la asociación de acciones a la reducción de las reglas de la gramática, descrita previamente en el analizador sintáctico.

En el laboratorio, se explicarán conceptos básicos de **NASM** para la realización de la práctica.

La gestión de la entrada y salida se realizará mediante la librería auxiliar (**alfalib.o**) disponible en la plataforma moodle. El profesor indicará el procedimiento a seguir para utilizar dichas librerías.

Gestión de errores en tiempo de ejecución

El compilador sólo comprobará dos errores en tiempo de ejecución:

- Índice de un vector fuera de rango
- División por cero

Si durante la ejecución de un programa se produce alguno de los dos errores anteriores, el programa terminará de manera ordenada y mostrará un mensaje de error adecuado atendiendo al siguiente formato (obsérvese la ausencia de acentos):

****Error de ejecucion: Indice fuera de rango.
****Error de ejecucion: Division por cero.

Invocación del compilador

El compilador deberá cumplir los siguientes requisitos:

- Nombre del programa fuente que contenga la rutina principal (main) del compilador: **alfa.c**
- El ejecutable se invocará de la siguiente manera:

alfa <nombre fichero entrada> <nombre fichero salida>

- Descripción del fichero de entrada y de salida:
  - El fichero de entrada contiene un programa escrito en lenguaje **ALFA**.
  - El fichero de salida contiene un programa escrito en lenguaje ensamblador NASM que funcionalmente es equivalente al programa contenido en el fichero de entrada.

Normas de entrega

Se entregará a través de Moodle un único fichero comprimido (*.zip*) que deberá cumplir los siguientes requisitos:

- Deberá contener todos los fuentes (ficheros *.l, .y, .h y .c*) necesarios para resolver el enunciado propuesto. No es necesario incluir los ficheros *lex.yy.c* ni *y.tab.c* puesto que puede generarse a partir del *.l* y del *.y* respectivamente.
- Deberá contener un fichero *Makefile* compatible con la herramienta make que para el objetivo ***all*** genere el ejecutable de nombre ***alfa***.
- El nombre del fichero *.zip* será***:***
- Para entregas individuales:

Apellido1_Apellido2_Nombre_sintactico.zip

- Para entregas en pareja:

Apellido1Estudiante1_Apellido1Estudiante2_sintactico.zip

Los apellidos de los elementos de la pareja serán en orden alfabético. Los nombres no deben contener espacios, acentos ni eñes.

MUY IMPORTANTE: UN COMPILADOR CON CONFLICTOS SE CONSIDERARÁ SUSPENSO.