TITULO.

IMPLEMENTACIÓN DE UN COMPILADOR PARA LENGUAJE GAUSS.

Gustavo Arroyo Delgado
Centro Interdisciplinario de Investigación y Docencia
en Educación Técnica (CIIDET)
ciidet@ciateq.mx

RESUMEN.

La implementación de este compilador la realice durante un cuatrimestre como proyecto de una materia en el CINVESTAV-IPN en 1987. El trabajo consistió en traducir completamente el compilador de lenguaje PL/I a Pascal (Borland, 1985), el código fuente (PL/I) fue tomado completamente del libro "An implementation guide to compiler writing" de Jean-Paul Tremblay y Paul G. Sorenson. El compilador es totalmente didáctico y es utilizado por los alumnos del Departamento de Ciencias Computacionales de la Universidad de Saskatchewan en Canadá, según los autores del libro. La ventaja que se logro con este trabajo resulta obvia; el compilador escrito en Pascal es más accesible que escrito en PL/I; por lo tanto puede ser utilizado ampliamente como instrumento didáctico.

INTRODUCCIÓN.

En una materia de ingeniería en sistemas computacionales como es **Compiladores**, el objetivo es conocer la estructura de un compilador, la relación del lenguaje a traducir con la gramática que sé esta utilizando. Así como establecer un método de implementación. Esto significa; comprender primero la manera de representar y definir la gramática de un lenguaje por medio de una teoría que no es difícil pero si muy singular, posteriormente dar a conocer las bases para manejar cadenas de caracteres y aprender la representación de las operaciones que se pueden hacer con estas cadenas. Una vez visto lo anterior se procede a analizar casos prácticos de implementación de alguna de las fases del compilador.

He observado que en otras materias de la misma carrera de computación, en alguno de sus textos trae consigo un material didáctico que consiste de un programa en código fuente, el cual es muy susceptible de analizar. Reflexionando ¿el porque?, De este material didáctico resulta sencillo explicar que normalmente un programa de este tipo no es fácil diseñarlo he implementarlo en un semestre y es poco probable que en dos semestres se logre. Este material permite al alumno tener una idea más clara de lo que representa una implementación de este tipo. A su vez le permite analizarlo detalladamente, modificarlo y optimizarlo si su capacidad se lo permite. Estoy hablando del caso de Sistemas operativos.

En compiladores no he observado algún libro de texto que incluya un material como el caso descrito. Es importante hacer patente esta deficiencia. Pero no significa que una vez que se tenga este material el problema este resuelto. Es cuando comienza la labor más importante. La de hacer que al alumno aprenda con este nuevo material. Esto a su vez trae consigo una labor que se encuentra escondida, y es diseñar o modificar el contenido de la materia.

Hay un libro de texto de un compilador didáctico ***(Treblay, 1982)***. Este compilador esta escrito originalmente en lenguaje PL/I. Sin embargo como este lenguaje no es de uso cotidiano lo traduje completamente a Pascal durante un posgrado que hice en el CINVESTAV-IPN en 1987. Desde entonces se vislumbro la posibilidad de la utilidad de este trabajo. Sin embargo una cosa es el trabajo técnico y otra es el trabajo docente y todavía más el trabajo que implicaría la implementación de un de un nuevo proyecto curricular para esta materia. Pese a que han pasado diez años aún no he visto un compilador didáctico que se encuentre implementado tal como sucede con sistemas operativos como LINUX o MINIX.

Creo que es el momento de concluir ese trabajo. No tiene ningún sentido tener ese compilador didáctico archivado. Además estamos en el mejor lugar para plantear esta propuesta y en caso de ser aprobado llevar a cabo el trabajo.

DESCRIPCION DE LAS ACTIVIDADES REALIZADAS AL IMPLEMENTAR EL COMPILADOR GAUSS

El compilador esta originalmente escrito en lenguaje PL/I y se traslado a lenguaje pascal (Turbopascal). Se realizaron algunos cambios necesarios para lograr la correcta implementación del compilador, por ejemplo:

- Las constantes que representan tamaño de algunas tablas como: La tabla de símbolos y la tabla del código generado.

- Las constantes que representan el valor interno de los símbolos terminales y los no terminales, así como las constantes usadas para definir las instrucciones en la generación de código.

Estas constantes fueron definidas originalmente como macrodefiniciones, sin embargo, ya que en pascal no se puede hacer tal cosa, se declararon como tipos enumerados. Así cuando es necesario su valor ordinal se puede lograr fácilmente.

En el analizador léxico (scanner) también se hicieron algunos cambios importantes como:

El código que maneja la máquina donde se hizo la implementación original es EBCDIC, y en esta parte del compilador se encuentra una tabla donde se mapean los caracteres de tal código, pero como ahora se utilizará el código ASCII se mapeo la tabla de acuerdo al código actual y así el scanner cuando encuentra algún carácter se bifurca al estado correcto, marcado por el autómata finito determinístico al que simula el scanner.

En este mismo procedimiento esta simulado un estatuto *case* con *goto's* ya que en PL/I no existe este estatuto pero al traducirlo a pascal no hay necesidad de simularlo ya que aquí si se encuentra implementado el *case*. En la rutina *llparse* existía originalmente una tabla constante llamada *ll1parse* la cual contiene la tabla del analizador sintáctico LL(1) (*parse*) tal como se

genero de la gramática para lenguaje gauss, sin embargo se tomo la decisión de guardar esta tabla en un archivo, ya que es una tabla bastante grande y si se declara como constante genera mucho código objeto, además si se deja tal y como esta se satura el segmento de código objeto de turbo pascal antes de compilar completamente el programa por lo que marcaba un error de sobreflujo de memoria. En vista del problema se opto por declararla como una variable y cargar la tabla desde disco, como si se tratara de un archivo de datos, de este modo toma espacio de la memoria (*heap*). El programa que genera el archivo que contiene la tabla del *parse* es llamado: PARSETAB.PAS y el archivo que genera: *parsetab.doc*. En general en el compilador escrito en PL/I se declaran los procedimientos tan pronto como se necesiten, o sea que no sigue una estructura de alcance de procedimientos tal como sucede en pascal, por lo tanto al traducirlo a pascal se tuvo que adecuar al formato de un programa pascal. A continuación se detalla la forma como esta declarados los procedimientos en PL/I:

compile
- *init*
  - *scanner*
    - *get_char*
      - *option*
    - *get_num*
    - *is_kywd*
  - *ll_parse*
    - *s_push*
    - *s_pop*
    - *err_hand*
      - *fix_up*
    - *stack_pr*
  - *lookup*
  - *sym_rem*
  - *action*
    - *store_lit*
    - *template*
    - *relation*
    - *code_fix*
    - *bin_op*
    - *split*
    - *unary_op*
  - *execution_phase*
  - *sem_err*
  - *error*
  - *list_err*
  - *llerror*
  - *token_pr*

Ahora detallaremos como están declarados los mismo procedimientos pero en lenguaje pascal:

compile
init
error
token_pr
llerror
list_err
sem_err
lookup
sym_rem
action
store_lit
template
code_fix
relation
bin_op
split
unary_op
format_lit
scanner
get_char
option
get_num
is_kywd
llparse
s_push
s_pop
err_hand
fixup
stack_pr

DISTRIBUCIÓN DE LOS PROGRAMAS DEL CODIGO FUENTE
El compilador cuenta con los siguientes archivos de código fuente:

Gauss.pas.- Programa principal del compilador que tiene incluido los siguientes:

Procedu1.pas
Procedu2.pas
Procedu3.pas
Scanner.pas
Llparse.pas
Llparse1.pas

El programa gauss.pas hace uso de un archivo de datos llamado:

Parsetab.dat

El programa gauss.pas produce un archivo de datos llamado:

Code.out

El programa interp.pas correspondiente al programa principal del interprete el cual contiene incluidos los siguientes programas:

Resint.pas
Inter1.pas

COMO SE OPERA EL COMPILADOR GAUSS

El programa gauss puede editarse en cualquier procesador de textos para compilarlo más tarde. La forma como se debe correr el compilador es la siguiente:

C:> gauss < *nombre*

Donde:

nombre.- Es el archivo que contiene el programa gauss a compilar.

Para visualizar mejor los procesos que realiza el compilador existen varias opciones de compilación, las cuales siempre se declaran en la primer línea del programa a compilar. Algunas de las opciones posibles son:

scanner_debug:

Esta opción esta diseñada para ayudar a depurar el scanner. Cada vez que el scanner es llamado son desplegados el valor del interno del token y el tipo de token que es invocado.

ll1_debug :

La cual causa que sea desplegado en la pantalla :

El contenido del stack del parse
El token de entrada
Las acciones del parse (una expansión, un pop, un error, un accept)

code_debug :

Cuando esta opción es habilitada se despliega el código generado.

PROPUESTA.

La propuesta concreta es publicar la traducción del libro “An implementation Guide to Compiler Writing” el cual un 50 %, aproximadamente, es código fuente en lenguaje PL/I. La nueva edición contaría con este mismo código fuente pero ahora en Pascal. Además puede incluirse un disquete con el mismo código fuente, así mismo se agregarían las actividades realizadas durante la implementación.

CONCLUSION.

Desde el punto de vista didáctico, no basta con crear material de este tipo para una asignatura, es necesario elaborar actividades de aprendizaje con dicho material para materias muy especificas en ingeniería. Ya que muchas de las veces existe el material pero, o no se usa o no se sabe que existe. Consecuentemente el potencial de estas herramientas se esta desperdiciando inútilmente.

BIBLIOGRAFIA.

TREMBLAY, Jean-Paul, Paul G. Sorensen. "THE THEORY AND PRACTICE OF COMPILER WRITING. Primera Edición. Edit. Mc Graw-Hill. Singapore, 1985.

TREMBLAY, Jean-Paul, Paul G. Sorensen. "AN IMPLEMENTATION GUIDE TO COMPILER WRITING. Edit. Mc Graw-Hill. Estados Unidos, 1982.

BORLAND International Inc., "TURBO PASCAL v 3.0 REFERENCE MANUAL". Scotts Valley, CA. Estados Unidos, 1985.