Compilador MFJG

Martin Purita, Legajo 51187
Facundo Menzella, Legajo 51533
Julián Gutiérrez Ferrara, Legajo 51141
Gustavo Del Guidice, Legajo 51289

Objectivo

Desarrollar un compilador que sea capaz de factorizar un número primo. Este compilador también funcionaría para implementar otras funciones como obtener el número de Fibonacci.

El Problema

Al avanzar con el desarrollo de la segunda parte, nos dimos cuenta que la gramática que habíamos definido incialmente
Validaba muchas cosas que seria mas cómodo verificar en la semántica, Esto se debió a que todavía no poseaiamso los conocimientos sobre la estructura semántica al momento de confeccionar la gramática

La solución

Se realizo un cambio drástico en la misma, adicionalmente y en consideración del objetivo de nuestro compilador, se simplifico de forma tal que solo admita una función principal, y una lista de números a factorizar en primos por medio de la palabra reservada "*decompose*" de nuestra sintaxis.

La gramática

La nueva gramática se define de la siguiente forma

G = (Vn, Vt, S, P) donde:

Vn = {S, LEFT_PARENTHESIS, RIGHT_PARENTHESIS, LEFT_BRACE , RIGHT_BRACE , SEMI_COLON, RETURN, DECOMPOSE, TYPE,CONST,ID};

Vt = {a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,0,1,2,3,4,5,6,7,8,9,{,},(,)};

S símbolo inicial;

P producciones;

La semántica

Como se menciono anteriormente, se paso gran parte de las validaciones de la gramatica aquí, internamente al compilar se ejecuta la descomposición en primos y queda en a.out un archivo de c con una lista de printfs con los resultados.Esta es la representación en C de un programa de nuestra sintaxis, elegimos este método porque es mas sencillo que soportar salida estándar, y mucho mas eficiente a la hora de ejecutarse mas de una vez. El método de entrada debe ser manual por medio del programa a compilar, ya que no soportamos entrada estándar.

Ref. Externas

En busca de una forma eficiente de descomponer números primos se utilizo una implementación con cache[1] , la primera ejecución genera un archivo adicional primebits de alrededor de 167mb con todos los números primos que entran en un int de 64 bits, por lo que la primer ejecución toma alrededor de 1 minuto, y luego se reduce a 1 segundo.

Pendientes

Como mejoras al TPE seria bueno soportar BIG INTEGER como parámetro en la descomposición, ya que por ahora solo soportamos int de 64bits, adicionalmente seria bueno también que el código equivalente a nuestra sintaxis sea en ASM en vez de en C, para mayor velocidad.

Código fuente

El código fuente se puede ver en el repositorio de GitHub: https://github.com/MartuPuri/TLA/ encontraran también un archio de README con el modo de uso del mismo.

[1] http://rosettacode.org/wiki/Prime_decomposition#Using_GNU_Compiler_Collection_gcc_extensions