**Compilador**

**FASE 0**

Command Line Interface

* Para solucionar el problema de ejecutar la última opción repetida, la solución fue recorrer el arreglo args de “derecha a izquierda” auxiliándonos de una variable tipo boolean para cada opción, que por defecto está seteada como true. Una vez que encontró la opción, setea la variable auxiliar correspondiente a la opción como false, el cual impedirá que entre a ejecutar nuevamente la opción si en caso se volviera a encontrar en el arreglo.
* Esta solución tiene un problema, que si se ejecutan de una vez las opciones, se ejecutarán en el orden invertido al que se ingresaron en el comand line interface. Para solucionar esto, se crearon linkedlists, para guardar la opción, y el parámetro. Se va agregando al final de la linked list, de modo que al terminar obtenemos una linked list con las ultimas opciones y en el orden correcto.

-o outname

* Para la fase 0 se instancia la clase scanner con el nombre del archivo de salida “outname”.
* Se instancian todas las demás clases con el objeto de la clase anterior, siguiendo el orden Scanner, Parser, Ast, Semantic, Irt, Codegen.
* En cada clase se escribe en el archivo stage: [stage]
* Al recibir en la clase, el objeto de la clase anterior, se guardan los campos del objeto entrante en los campos de la clase actual, definiendo nuevamente los campos. Esto con la finalidad de escribir en el archivo. (Al terminar la implementación nos parecio, que era innecesario hacer nuevamente la creación de los campos para escribir en el archivo el stage)
* Debido a que, para que el archivo se pudiese seguir escribiendo, debía continuar abierto, por lo que se cerró hasta el final, en codegen.

-target <stage>

* Se agregó en el constructor del scanner un espacio para target, el cual cuando se instanciara con la función –o esta quemado como “end”, pero para la función target, en vez de enviar “end” se envía una variable target. En la clase se recibirá esta variable y luego se hará la verifiación si target es igual a la etapa en la que se encuentra, si es así se cierra el archivo con .close() Esto se hizo porque para seguir escribiendo en el archivo, el archivo debía de estar abierto, y solo hasta la clase a la que quisiéramos llegar debería de cerrarse y así fue como se implementó.

-opt

* Para esta función se entendió que al ejecutarla en el command line solamente se crearía una instancia de la clase (correspondiente, según el parámetro, algebraic o constant) y que se escribiría en el archivo de salida “optimizing: constant folding ” o “optimizin: algebraic simplification”
* Encontramos un posible error en ejecución. Suponiendo que en el comand line ejecutemos –target stage –opt optimization filename. El compilador escribirá un archivo con nombre namefile para la opción stage como primera ejecución, luego se ejecutará opt y también escribirá un archivo con el mismo nombre. La duda aquí es, si el programa debería de sobre escribir el archivo, osea se crea el anterior y en el momento de que se ejecute la otra opción debería de escribirse uno nuevo (encima de ese archivo) o si el archivo queda abierto para que se continue escribiendo para cada opción que se ejecute.
* Actualmente está implementado para que sobreescriba el archivo.

Scanner

* Hace falta corregir los errores de los testcases de string 1, string 2, string 3.
* Cuando ingresan 0x es un numero hex invalido, toma el 0 como entero y la x como variable
* Char permite tener más de un carácter

04/09/14

Scanner

Testcases

* Char5
* Char6
  + 0xA?
* Char8
  + 0xA?
* Char9

Ast

* Visitor de field
* Visitor de method
* Como recorrer árbol ast