Instituto Tecnológico de Costa Rica

Área de Ingeniería en Computadores

CE-3104: Lenguajes, Compiladores e Intérpretes

Profesor:

Marco Hernández Vázquez

Proyecto: AnimationLed

Integrantes:

Carlos Adrián Araya Ramirez

Michael Shakime Richards Sparks

José Andrés Solano Mora

Junio, 2021

Tabla de contenido

Diagrama de Arquitectura

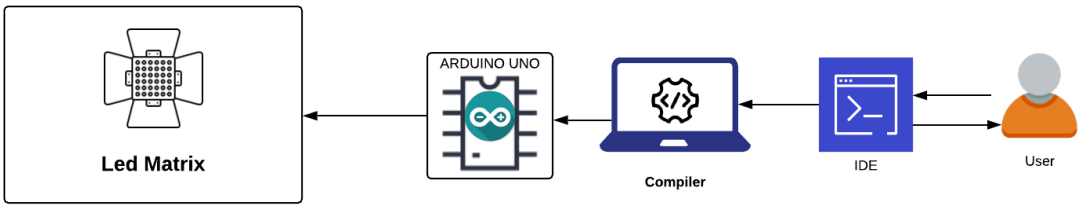

Ilustración 1 (Diagrama de Arquitectura)

Alternativas de solución

Comunicación en serie con el microcontrolador: para la comunicación desde Python al microcontrolador se presentaron varios inconvenientes, en un principio se estaban enviando las instrucciones en una lista que contenía el estado actual de la matriz después de ejecutar cada instrucción, y a raíz de este método surgió un problema que recortaba este insumo del microcontrolador provocando resultados inesperados en la matriz led.

Una de las soluciones que nos planteamos fue utilizar una comunicación en serie en la cual se mandaría cada instrucción una a una y se actualizaría la matriz en tiempo de ejecución del intérprete de Python, sin embargo, esto suponía otros problemas adicionales que nos hicieron desistir de esta alternativa.

Por un lado, este método iría en contra de los requisitos de implementación del compilador, ya que uno de los objetivos de la solución es lograr compilar el programa correctamente antes de ejecutarlo, y con esta alternativa se utilizarían hilos para manejar los delays y estos se ejecutarían en tiempo real hasta llegar al final del código.

Además, el tiempo de carga de datos del microcontrolador es aproximadamente de 1 segundo, esto no permitiría manejar tiempos en milisegundos y además podría causar problemas con la sincronización de las instrucciones.

Problemas conocidos

Blink: Se presentó un problema con la función “Blink”, pues esta función a nivel de compilación funciona, sin embargo, no fue posible implementarla a nivel del controlador de leds por problemas con la frecuencia de impresión de leds en la matriz. Pues sucede que las animaciones generales funcionan con un tiempo específico dentro del controlador e incorporar una animación que se ilustrase a una frecuencia diferente mientras otra animación ocurría era una tarea laboriosa por dos principales razones.

Primeramente, la impresión se maneja mediante una variable “current” que guarda el “frame” que debería imprimirse en determinado momento, de modo que era muy complicado decir cuando un led de ese “frame” debería estar encendida o apagada durante la ejecución del “Blink”.

Como segundo problema, para el manejo del tiempo “Delay” se utilizó un contador que mediante la función módulo (%), se podía ajustar dependiendo de la frecuencia deseada como unidad básica (milisegundos). El problema radica en que cuando el controlador encuentra una instrucción “Delay”, añade la cantidad en milisegundos a una variable “wait” que se encarga de hacer esperar al ciclo una determinada cantidad de vueltas antes de poder continuar con la animación, esto ocasionaba colisiones si se deseaba implementar otro tiempo de espera para la intermitencia del “Blink”.

Manejo de errores: debido a que no logramos conseguir el cuarto integrante en el grupo y la dificultad que impone la distancia para trabajar en la parte de electrónica, no nos alcanzó el tiempo para lograr manejar todos los errores de compilación. Es por esto que, a la hora de ejecutar ciertas instrucciones con errores, estos no se atrapan de la manera correcta y se cae el programa o no imprime correctamente la línea en la que se encuentra el error. Sin embargo, a pesar de este inconveniente, si el código no posee errores, no fallará a la hora de compilar.

Actividades realizadas por estudiante

Fecha	Horas invertidas	Estudiante	Actividad
18/05/2021	2	Jose Solano	Compra de componentes electronicos.
19/05/2021	3	Adrián Araya Shakime Richards	Investigación de interfaces en lex yacc.
21/05/2021	2	Adrián Araya	Búsqueda de potenciales tokens y creación de tabla de símbolos.
23/05/2021	7	Adrián Araya	Definición del análisis léxico.
24/5/2021	4	Shakime Richards	Definición de los establecimientos del parsing.
25/5/2021	2	Jose Solano	Investigación de matriz de leds en Arduino.
26/5/2021	3	Jose Solano	Armado del circuito.
27/05/2021	1	Shakime Richards	Definición de los BNF de la asignación de variables.
28/05/2021	2	Adrián Araya Shakime Richards	Definición de los BNF de las operaciones aritméticas.
29/05/2021	0,5	Adrián Araya	Definición de las BNF de las funciones temporizadoras para graficar en el Arduino.
	0,5	Shakime Richards	Definición de los BNF de la asignación de listas.
30/5/2021	2	Shakime Richards	Definición de los BNF de los índices de las listas y modificación de valores en una lista.
	0,5		Definición de los BNF de la inserción, eliminación y len en una lista.
1/6/2021	0,5	Adrián Araya	Definición y adaptar los BNF de los índices de las matrices y modificación de valores en una matriz.
	0,5	Adrián Araya Shakime Richards	Definición de los BNF de la inserción y eliminación en una matriz.

	0,5	Adrián Araya	Definición de los BNF de shapeC y shapeF.
	0,5		Definición de los BNF de las funciones booleanas para listas y matrices. (Neg, T, F)
	0,5		Definición del IF, FOR, PROCEDURE.
2/6/2021	10	Adrián Araya	Definición del IDE, funciones principales: open file, compile y compile and run.
3/6/2021	1	Jose Solano	Investigación sobre el uso de PySerial junto a Arduino.
	2		Definición de las funciones necesarias para la comunicación Python-Arduino.
5/6/2021	4	Shakime Richards	Implementación de las validaciones en la asignación de variables.
	1		Implementación de la función type.
	4		Implementación de las validaciones en los indices de fila y columna en las listas y matrices.
	5	Jose Solano	Definición de las funciones necesarias para el display de los leds mediante la estructura de matriz.
6/6/2021	4	Shakime Richards	Implementación de las validaciones en los indices de sublistas en las listas y matrices.
	2		Implementación y validaciones de la funcion insert.
	1		Implementación y validaciones de la funcion del.
	2		Implementación y validaciones de la funcion range.
	0,5		Implementación y validaciones de la funcion len.
7/6/2021	2	Shakime Richards	Implementación y validaciones de la funcion insert en matriz.
	2	Adrián Araya	Implementación y validaciones de la funcion del en matriz.

	2		Implementación de las funciones y validaciones de shapeC y shapeF.
8/6/2021	1	Shakime Richards	Implementación de las validaciones para las operaciones aritméticas.
9/6/2021	1	Jose Solano	Realización de pruebas de leds con matrices de python.
	4		Implementación de primeras animaciones leds con instrucciones de prueba.
10/6/2021	2	Adrián Araya Shakime Richards	Implementación de las validaciones en las funciones del DELAY.
	2		Implementación de las validaciones en las funciones del BLINK.
	2		Implementación de las validaciones en las funciones del PRINTLED.
	2		Implementación de las validaciones en las funciones del PRINTLEDX.
11/6/2021	1	Adrián Araya	Implementación de las validaciones en las funciones booleanas para listas y matrices. (Neg, T, F)
	2	Adrián Araya	Implementación del IF.
	3	Jose Solano	Optimización del controlador de leds en Arduino y ajustes del ciclo de animación
12/6/2021	2	Adrián Araya	Implementación del FOR.
	3		Implementación del PROCEDURE.
13/6/2021	6	Jose Solano	Reestructuración del envío y almacenamiento de instrucciones en el controlador de Arduino.
14/6/2021	1	Adrián Araya	Combinación del IDE junto con el compilador.

16/6/2021	3	Jose Solano	Pruebas y ajuste del tiempo de delay en la animación mostrada.
19/6/2021	9	Adrián Araya Shakime Richards	Corrección de errores de las estructuras de datos y testeo del para verificar validaciones extra.
20/6/2021	4	Adrián Araya Shakime Richards	Preparación en conjunto en el análisis semántico.
23/6/2021	4	Adrián Araya Shakime Richards	Implementación de la notificación de errores.

Problemas encontrados

- **Declaración de variables:** En un principio se optó por la implementación del análisis semántico dentro del mismo análisis sintáctico, esto ocasionaba que cada vez que el parser encontraba una instrucción de declaración inmediatamente instanciaba la variable. Al inicio tenía sentido, sin embargo, cuando se comenzaron a implementar los bloques que encapsulan de instrucciones propias, como el IF, FOR y PROCEDURE, se detectó que las declaraciones se ejecutaban siempre, sin importar si se cumplía la condición para entrar en el bloque, por ejemplo, en un IF, si no se cumplía la condición igualmente ejecutaba todas las instrucciones que tenía dentro, ya que el parser lee todos los tokens generados por el análisis léxico sin importar las reglas semánticas.

  Esto se solucionó separando el análisis sintáctico del análisis semántico, de esta manera, primero el parser genera una lista con los BNF de cada instrucción y el análisis semántico se encarga de verificar si es posible ejecutar cada instrucción según las reglas especificadas.

  Asimismo, se mantuvieron algunas validaciones sencillas dentro del análisis sintáctico, por ejemplo, recibir un booleano en el espacio donde se espera un entero o el ID de una variable; ingresar un string erróneo (“hora”) en las funciones de temporizador, entre otros.
- **Almacenamiento de instrucciones:** Se presentó un problema a la hora de enviarle las instrucciones de ejecución del programa al controlador de la matriz de leds implementada en Arduino. Pues se pensaba manejar una matriz tridimensional en donde se fuesen almacenando matrices de dos dimensiones, cada una representaba la distribución de los leds

iluminados en el dispositivo en un momento determinado. Sin embargo, esta solución resultó ser poco eficiente ya que la estructura planteada consumiría demasiada memoria para las variables globales del programa en Arduino UNO, dejando muy poco espacio para variables locales, esto causaría que no se recibiesen correctamente los datos y, por consiguiente, los leds no se iluminarían de forma correcta.

Para solucionar este problema se tuvo que reestructurar el almacenamiento de las instrucciones de matrices a String, de modo que ahora, en vez de existir una matriz tridimensional, se tenía un arreglo de Strings, cada String almacenaba los bits que se deberían encender en el dispositivo de leds. Esto solucionó en gran medida el problema, pues aumentó considerablemente la cantidad de instrucciones que se podían almacenar y por otro lado también se facilitó el manejo y recorrido de las mismas.

- **Buffer de envío insuficiente:** Otro problema relacionado con el envío de instrucciones al controlador de Arduino fue el tamaño del buffer, pues los datos de la matriz se enviaban mediante Pyserial en un String que contenía cada bit que debería ser iluminado o apagado, esto con 1's y 0's dentro del String, de modo que, si se enviaban tres instrucciones, se hubiese tratado de un String de 192 caracteres (64 bits por cada matriz). Esta estrategia podía funcionar con pocos pasos dentro del programa y con un almacenamiento que diera abasto, pero conforme se realizaron pruebas los problemas fueron más evidentes. De modo que decidimos compactar el envío de la información haciendo uso del sistema hexadecimal.

La solución fue simple pero laboriosa, pues se construyó una estructura específica para enviar y recibir la información:

"(P66812400007E0000TS0002P66816600423C0000)"

El String se encuentra delimitado por paréntesis en ambos extremos para facilitar la lectura dentro del controlador. La letra "P" es un identificador de que los siguientes 16 caracteres conforman la matriz de 64 bits a graficar en hexadecimal. La letra "T" es el identificador de los delays a realizar, lo sigue una letra ("N" para milisegundos, "S" para segundos y "M" para minutos), acompañada de la cantidad (4 dígitos).

Conclusiones

El estudio del proceso de compilación requiere un conocimiento profundo de los elementos que constituyen un lenguaje de programación, así como en múltiples áreas como expresiones regulares y estructuras de datos. Además, es necesaria una gran capacidad de análisis y síntesis, debido al tiempo que requiere validar cada uno de los posibles casos que se presenten, pues al ser código fuente, depende meramente de la imaginación del usuario. Por tanto, se concluye, que un compilador depende implícitamente de las limitaciones del lenguaje que este representa y por tanto todo compilador es propenso a tener mínimo una falencia técnica.

Durante el proceso de realización de este proyecto fue primordial la constancia y capacidad de autoaprendizaje para identificar los formalismos y las técnicas adecuadas de la teoría para implementar las diferentes fases del compilador, sumada la orientación del profesor.

El envío posterior del código y su final interpretación en la parte de hardware simbolizan lo aprendido en el curso de Compiladores e Intérpretes, así como el resultado obtenido después de un desarrollo exponencial de conocimiento que abarca desde el inicio de la teoría relacionada al reconocimiento de patrones del lenguaje, para luego formalizar estos patrones de manera lógica en estructuras de datos y así finalmente generar un lenguaje de “bajo nivel” que es interpretado por la matriz de leds.

Recomendaciones

- Antes de iniciar el desarrollo de un compilador es muy importante considerar primero la tabla de símbolos y generar cuidadosamente las gramáticas, debido a que a medida que crecen las declaraciones, se va convirtiendo en un proceso más automatizado y puede ser muy engorroso encontrar un error.
- De ser posible, buscar una matriz de leds más compacta en el sentido de los cables, por ejemplo, una de pins. La matriz que se utilizó en este proyecto tenía 16 pins, por lo que al inicio se complicó un poco la manipulación y la construcción del circuito.

Bibliografía

[1] J. Engelsma, "Part 01: Tutorial on lex/yacc", *Youtube.com*, 2021. [Online]. Available: https://www.youtube.com/watch?v=54bo1qaHAfk. [Accessed: 26- Jun- 2021].

[2] D. M. Beazley, "PLY (Python Lex-Yacc)", *Dabeaz.com*, 2021. [Online]. Available: https://www.dabeaz.com/ply/ply.html. [Accessed: 26- Jun- 2021].

[3] "Graphical User Interfaces with Tk — Python 3.9.5 documentation", *Docs.python.org*, 2021. [Online]. Available: https://docs.python.org/3/library/tk.html. [Accessed: 26- Jun- 2021].

[4] " Programming 8x8 LED Matrix", Create.arduino.cc, 2021. [Online]. Available: https://create.arduino.cc/projecthub/SAnwandter1/programming-8x8-led-matrix-23475a [Accessed: 26- Jun- 2021].

[5] "pySerial's documentation", Pythonhosted.org, 2021. [Online]. Available: https://pythonhosted.org/pyserial/ [Accessed: 26- Jun- 2021].