0416005 張彧豪

編譯器 Project 4報告

1. 環境：

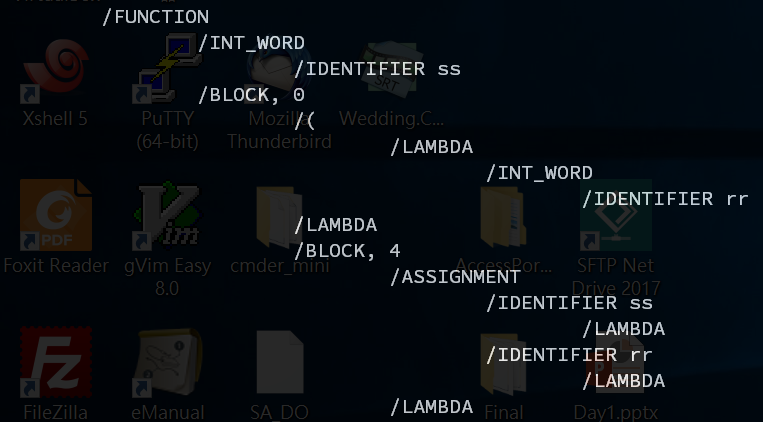
* OS：Ubuntu 16.04LTS
* gcc version：5.4.0

1. 檔案

* Scanner：scanner.l
* Parser：parser.y
* Abstract Syntax Tree：node.cpp, node.h
* Symbol Table：symbol\_table.cpp, symbol\_table.h
* Code Generation：CodeGen.cpp, CodeGen.h

1. 實作方法

這次的code generation是利用上一次實作symbol table時用到的abstract syntax tree，一樣是利用DFS的traversal方式去爬abstract syntax tree，並根據目前走到的node型別來generate出相對應的java byte code。範例如下圖所示



這是宣告函數敘述產生的abstract syntax tree，遇到function node時先決定其回傳type，並將綠框中的函數傳入參數走完，做完這兩件事情就可以把函數宣告的第一行byte code generate出來，之後把紅框部分的function body走完，在走的過程中，會利用到其他的node routine將相對應的byte code generate出來，最後再依照function的return type把

在這次的project裡，因為難度因素，我並沒有實作array的部分以及string的部分，但其他部分都有實作。

比較難實作的部份是字串以及印出的部份，字串的部份要一個特殊的type descriptor(java/lang/String)，而且當要把字串存到屬於字串型別的變數存到時，要先用ldc指令把字串塞到stack上，接著要使用astore存到相對應的local variable中，把字串的reference存到local variable中，會用astore的關係應該是因為是要存reference而不是值，這部份是嘗試很久後才試出來的。而print的實作則是要先把print對應到的static variable先放到stack上，再將要印出的東西放到stack上，最後填好正確的引數型別後呼叫print函數。

1. 心得

這次的project真的是非常困難，一開始其實完全不知道怎麼下手，看了另外一班的資料後又摸索了很久才發現可以利用上一次作業已經建好的abstract syntax tree來實作這次的code generation，在實作的過程中其實並不會覺得難到做不下去，但是需要一直測試一直測試才能繼續寫下去，實在有點辛苦，對於陣列的部份，因為實在想不到該怎麼實作，所以這次並沒有實作，不過有完成其他部份還是覺得很有成就感。