0416005 張彧豪

編譯器 Project 4 報告

一、環境：

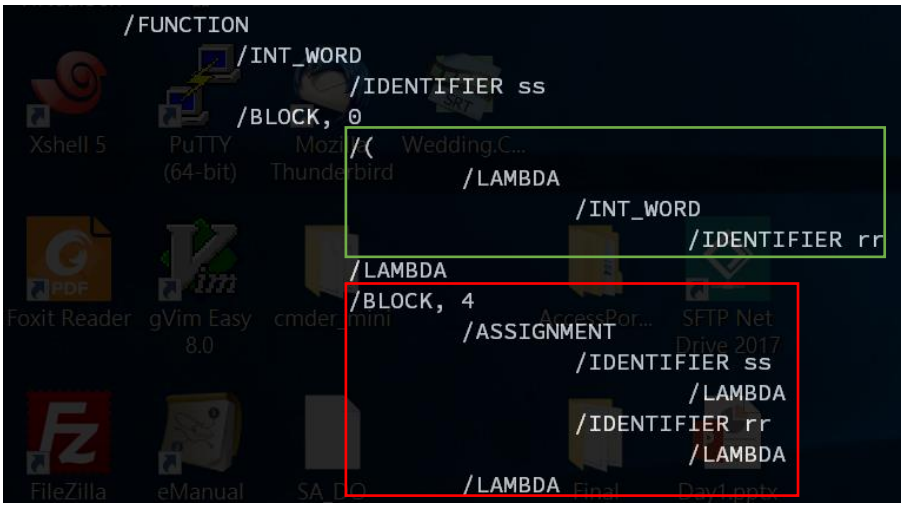
- OS：Ubuntu 16.04LTS
- gcc version：5.4.0

二、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

三、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。

四、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。