NCKU Compiler-2025 Spring Homework 3

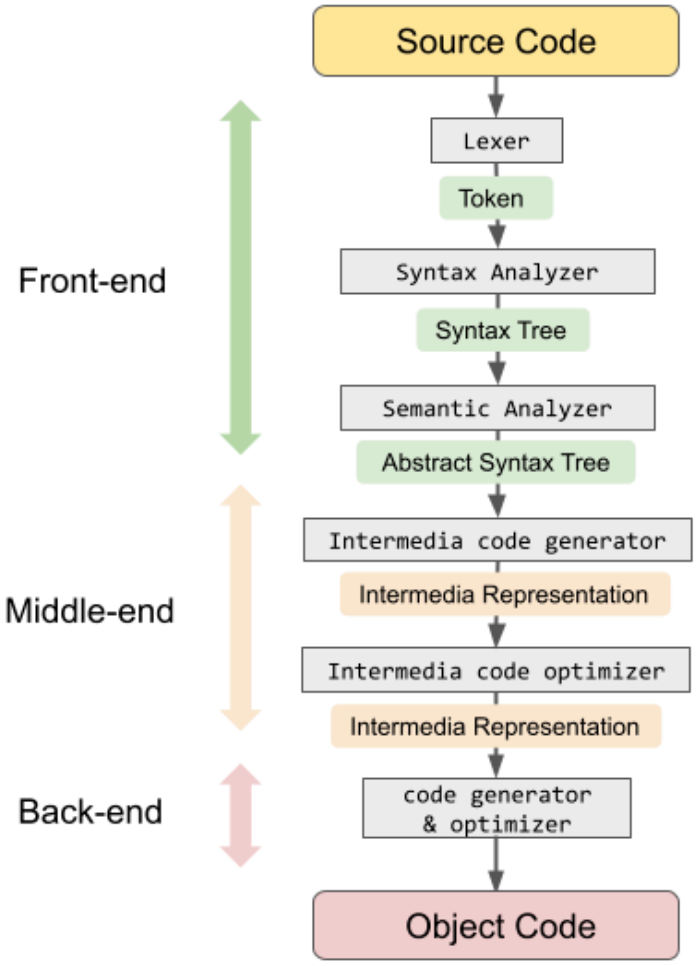
Java Assembly Code Generation

作業上傳

- 請將 `compiler.l`, `compiler.y`, `compiler_common.h`, `Makefile` 壓縮後上傳，壓縮檔的名稱為你的學號。

前言

Compiler 內部的步驟又分成前端、中端、後端



作業目標

這一個作業我們要產生 Java 的 Assembly Code，然後將其丟給 Java 的 Assmmbler (Jasmin) 產生出 Java 的 bytecode，完成 Compiler 的所有工作。

也就是說，完成了這一個作業之後，你所寫出來的東西，就是一個可以使用的 Compiler !

你需要做什麼?

在作業三，我們會將 Jasmin 所產生的 Java bytecode 丟給 JVM 執行，並使用執行的結果來檢查你的 Code Generator 是否正確。

接下來我們將會介紹一些 Jasmin 的語法，並提供範例參考，請你照著同樣的概念與邏輯完成這一個作業。

> 你會需要你作業二所寫的文法來協助你完成作業三，在作業二我們將語法解析順序輸出，在作業三我們要做的就是將這些照順序排好的步驟依序轉換成 Jasmin 的語法

> 請注意，我們不會列出所有 Jasmin 的語法，如果有想要使用的功能在這一份說明沒有列出來，請自行上網搜尋，此說明會盡可能的列出常用的當作範例。

基礎模板

```
.source Main.j
.class public Main
.super java/lang/Object

.method public static main([Ljava/lang/String;)V ; main function
.limit stack 100     ; 設定這個函式的最大堆疊大小
.limit locals 100    ; 設定區域變數大小(symbol table)
    ; ... main function asm code ...
    return
.end method
```

簡單來說，我們要把程式碼做的動作轉換成 Jasmin 的命令並寫至 Main.j，整個過程就很像在寫組語一樣。

> 更直白地說，就是你要把作業二使用語意動作輸出的那些順序，利用 `CODEGEN` 將 Jasmin 的指令當成字串寫入檔案。

生成 Main.j

範例程式有提供 macro 可以直接使用 `CODEGEN`

> Jasmin 的指令必須寫入 Main.j 這一個檔案內，可以在 compiler.y 的檔案中透過 `fprintf(yyout, "fmt", args...)` 完成，或是使用我們提供的macro (`CODEGEN("str")`)，可以自行選擇是否要使用。

> 作業二當中，我們透過語法樹以及語意動作輸出了語法被分析的順序，接下來我們就是直接透過這個順序轉換成 Jasmin 的命令做 Code Generator

我們以輸出當作例子，假設我們現在有一個程式片段如下：

```
println(30)
print("Hello")
```

那我們轉成 Jasmin 語法會變成：

```
getstatic java/lang/System/out Ljava/io/PrintStream;
ldc 30 ;
invokevirtual java/io/PrintStream/println(I)V

getstatic java/lang/System/out Ljava/io/PrintStream;
ldc "Hello World!\n" ;
invokevirtual java/io/PrintStream/print(Ljava/lang/String;)V
```

Jasmin 基本語法

Unary Operators

操作	Jasmin 語法 (for i32)	Jasmin 語法 (for f32)
+	無	無
-	ineg	fneg

Binary Operators

操作	Jasmin 語法 (for i32)	Jasmin 語法 (for f32)
+	iadd	fadd
-	isub	fsub
*	imul	fmul
/	idiv	fdiv
%	irem	浮點數不能做取餘數的操作

Bitwise Operators

操作	Jasmin 語法
&	iand
|	ior
^	ixor
<<	ishl
>>	iushr

> 接下來用一個範例來看看這些運算子結合在一起的應用。

現在我們有一段程式碼如下：

```
-5 + 3 * 2;
```

轉換成 Jasmin 的語法後，就會是：

```
ldc 5
ineg
ldc 3
ldc 2
imul
iadd
```

Store/Load Variables

操作	Jasmin 語法 (for i32)	Jasmin 語法 (for f32)
local -> stack	iload <addr>	fload <addr>
stack -> local	istore <addr>	fstore <addr>

> 接下來我們示範如何利用 Jasmin 語法將變數放入 stack 當中，並將變數儲存為 local variable。

- rust code

```
x = 9;
y = 4 + x;
z = "Hello";
```

- Jasmin code

```
ldc 9
istore 0     ; (可以想像成在宣告變數，並將 stack 最上方的東西儲存，所以這裡將 9 的值儲存到變
ldc 4
iload 0      ; (讀取變數 0 的值)
iadd         ; (將目前 stack 內的東西相加)
istore 1     ; (將相加的結果儲存到變數 1 (其實就是 y))
ldc "Hello"
astore 2     ; (將字串儲存到 2)
```

Jump Instruction

> goto 這一個語法可以直接跳轉到某個地方，通常會在 if/else 或是迴圈的地方使用到!
> ifxx 會pop掉stack最上面兩個，判斷是否要跳轉

Jasmin 語法	功用	Jasmin 語法	功用
goto <label>	直接跳		
ifeq <label>	如果是 0 就跳	if_icmpeq <label>	如果一樣就跳
ifne <label>	如果不是 0 就跳	if_icmpne <label>	如果不一樣就跳
iflt <label>	如果小於 0 就跳	if_icmplt <label>	如果小於就跳
ifle <label>	如果小於等於 0 就跳	if_icmple <label>	如果小於等於就跳
ifgt <label>	如果大於 0 就跳	if_icmpgt <label>	如果大於就跳
ifge <label>	如果大於等於 0 就跳	if_icmpge <label>	如果大於等於就跳

```
if( x == 10 ) {
    // do something...
}
else {
    // do the other thing...
}
```

> 這邊我們可以利用相減的結果是否為 0 的方式來判斷兩者是否相等。

> 原則上 Jasmin 的執行順序是由上到下，以下方的 code 做舉例：
>
> 當我們發現 x == 10 這件事情成立時，會跳到 L_cmp_0 把 true (iconst_1) 放進 stack
> 否則，我們會把 false (iconst_0) 放到 stack。
>
> 接下來我們就可以透過 stack 裡面是 true 還是 false 來判斷應該要執行哪一個部分，
> 如果是 true 那就沒有必要跳轉，執行完該執行的動作之後，再透過 goto 離開 label。
>
> 反之，如果 stack 裡面是 false，我們就直接跳到 L_if_false 去執行 else 想執行的動作。

- Jasmin Code

```
    iload 0         ; load x
    ldc 10          ; load integer 10
    isub
    ifeq L_cmp_0    ; jump to L_cmp_0 if x == 0
                    ; if not, execute next line
    iconst_0        ; false (if x != 0)
    goto L_cmp_1    ; skip loading true to the stack
                    ; by jumping to L_cmp_1
L_cmp_0:            ; if x == 0 jump to here
    iconst_1        ; true
L_cmp_1:
    ifeq L_if_false
                    ; -> do something
    goto L_if_exit
L_if_false:
                    ; -> do the other thing
L_if_exit:

```

Type Conversions

	int x	**long x**	**float x**	**double x**
(int) x		l2i	f2i	d2i
(long) x	i2l		f2l	d2l
(float) x	i2f	l2f		d2f
(double) x	i2d	l2d	f2d	

> 在 Jasmin 中，我們可以利用 i2f 與 f2i 來做 int to float 以及 float to int

我們一樣來看一個範例：

```
x = x + (int)6.28;
```

- Jasmin code

```
iload 0 (取得 0 的值)
ldc 6.28
f2i (轉型)
iadd (相加)
istore 0
```

Method Invocation

> 在這個部分我們要介紹如何用 invokevirtual 指令來呼叫 Function。

```
int foo(int x,int y) {
    return x + y;
}
int main(string argv[]) {
    int z = foo(3,4);
    cout << z << endl;
    return 0;
}
```

> function 的參數與回傳型態都會用 func_sig 來表示。
>
> 像是下方的 foo(II)I 括號內指的是有兩個整數參數，括號外指的是回傳的是整數型態。
> 詳細說明在參考資料 Java JNI types

> 注意，load跟store的地址會在每個function開始的地方重設，並且function的輸入會占用到address
> 例如 foo(II)I 0 跟 1 會是 function的輸入變數，local變數會從2開始

- Jasmin code

```
.method public static foo(II)I ; (定義 foo function)
    .limit stack 20
    .limit locals 20
     iload 0 ; (讀取第一個參數)
     iload 1 ; (讀取第二個參數)
     iadd
     ireturn ; (回傳整數)
.end method

.method public static main([Ljava/lang/String;)V
.limit stack 100
.limit locals 100
     ldc 3                       ; 將 3 放入 stack
     ldc 4                       ; 將 4 放入 stack
     invokestatic Main/foo(II)I  ; (呼叫 foo function)
     istore 1                    ; (將回傳結果儲存在 1)
     iload 1
     getstatic java/lang/System/out Ljava/io/PrintStream
     swap
     invokevirtual java/io/PrintStream/println(I)V ; (輸出)
     return
.end method
```

Creating Array

> 建立array會需要 newarray 或是 multianewarray，前者是一維陣列用的，後者是多維

```
let mut a: [i32; 3] = [10, 20, 30];
```

- Jasmin code

```
ldc 3            ; (3個元素)
newarray int     ; 創建3個元素的int陣列，並將array ref放進stack
dup              ; 複製array ref
ldc 0            ; (index 0)
ldc 10           ; (value 10)
iastore          ; [0]=10 (會pop掉stack最上面三個，也就是 index, value, array ref)
dup              ; 複製array ref
ldc 1
ldc 20
iastore          ; [1]=20
dup
ldc 2
ldc 30
iastore          ; [2]=30
astore 1         ; 儲存陣列到local變數
```

Tips

不知道怎麼寫的時候，可以創建一個Main.java檔案，例如

```
class Main {
    public static void main(String[] args) {
        int x = 9;
        int y = 4 + x;
        String z = "Hello";
    }
}
```

編寫完想要的程式後，執行

```
javac Main.java && javap -c -v Main
```

他就會吐出Java編譯器編譯出來的Java bytecode

```
public static void main(java.lang.String[]);
    descriptor: ([Ljava/lang/String;)V
    flags: (0x0009) ACC_PUBLIC, ACC_STATIC
    Code:
        stack=2, locals=4, args_size=1
         0: bipush        9
         2: istore_1
         3: iconst_4
         4: iload_1
         5: iadd
         6: istore_2
         7: ldc           #2                  // String Hello
         9: astore_3
        10: return
```

當然也可以看自己編譯出來的

```
javap -c -v ./build/out/Main.class
```

參考資源

- Jasmin instructions (看看就好，寫得沒有很清楚, 但ldc, getstatic, newarray 有特別用法，要看這裡): http://jasmin.sourceforge.net/instructions.html (http://jasmin.sourceforge.net/instructions.html)
- Java instruction table (精簡版指令介紹): https://en.wikipedia.org/wiki/Java_bytecode_instruction_listings

(https://en.wikipedia.org/wiki/Java_bytecode_instruction_listings)

- Java instruction set (比較詳細的指令介紹):
  https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html
  (https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html)
- Java JNI types (Function signature, instruction 會用到):
  https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/types.html
  (https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/types.html)