112 學年 第 2 學期 編譯器設計 小專題報告(3) 日期:2024/6/07

班級: 資工三 學號: B1043003 姓名: 陳麒安

1. 產生中間碼 (共 1 題，100 分，滿分 100 分)

請於 2024/6/07 晚上 12:00 前透過數位學習園區繳交

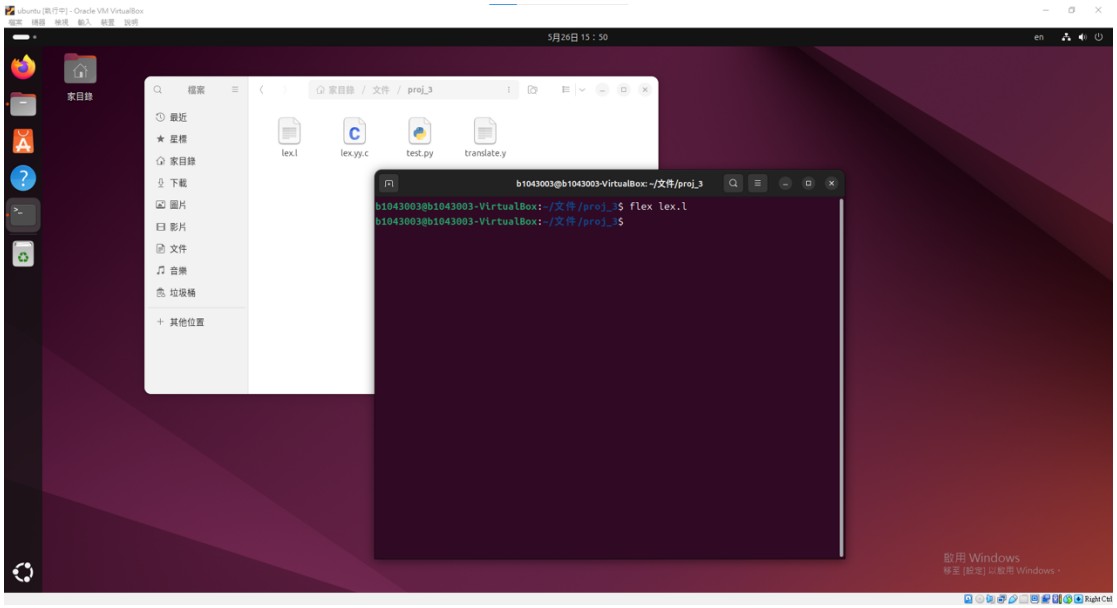
請以指定教材 4.9 小節內容及 Flex 與 YaCC 相關文件做為參考進行以下的實作，每小題須**貼關鍵程式碼，截圖呈現結果並文字說明。(注意: 請先使用 sudo hostname [學號]指令修改主機名稱為學號再截圖，否則不予計分。)**

A. 請以**小專題報告(2)**為例，產生及列印出三位址程式碼(Three-Address Code)。 (100 分)

Step 1. Lex.l 文件 :

```
%{
#include "translate.tab.h"   // 包含語法分析器生成的頭文件
#include <stdio.h>
%}

%%

[ \t\n]+                ;   // 匹配空白字符並忽略
print                   { return PRINT; }   // 匹配"print"關鍵字並返回 PRINT 標記
"("                     { return LPAREN; }   // 匹配左括號並返回 LPAREN 標記
")"                     { return RPAREN; }   // 匹配右括號並返回 RPAREN 標記
"+"                     { return PLUS; }      // 匹配加號並返回 PLUS 標記
"-"                     { return MINUS; }     // 匹配減號並返回 MINUS 標記
"*"                     { return MULT; }      // 匹配乘號並返回 MULT 標記
"/"                     { return DIV; }       // 匹配除號並返回 DIV 標記
"="                     { return ASSIGN; }   // 匹配等號並返回 ASSIGN 標記
[0-9]+                  { yylval.intval = atoi(yytext); return NUMBER; }   // 匹配數字並返回 NUMBER 標記，同時設置 yylval.intval 為匹配的整數值
[a-zA-Z][a-zA-Z0-9]* { yylval.strval = strdup(yytext); return IDENTIFIER; }   // 匹配標識符並返回 IDENTIFIER 標記，同時設置 yylval.strval 為匹配的字符串值
.                       { printf("Unexpected character: %s\n", yytext); exit(1); }
// 匹配其他字符，輸出錯誤消息並退出

%%

int yywrap() {
    return 1;
}
```

在撰寫完 lex.l 文件後執行 flex lex.l 得到 lex.yy.c 檔案，如圖一。



圖一、執行 flex lex.l 畫面

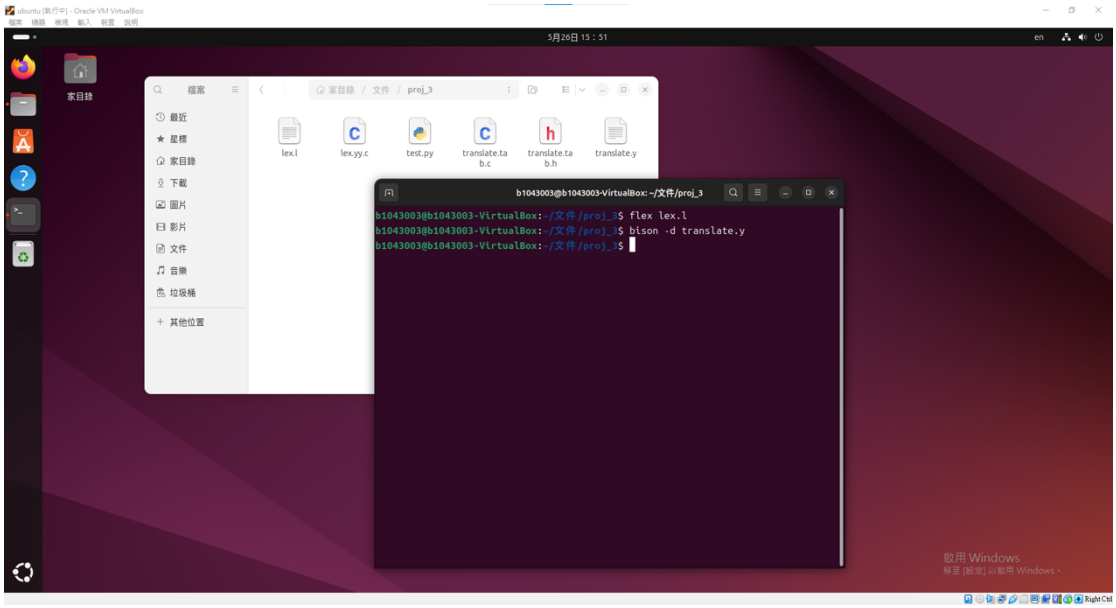
Step 2. Translate.y 文件 ：

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int tempVarCount = 0;   // 用於產生臨時變量的計數器

void yyerror(const char *s);   // 語法錯誤處理函數聲明
int yylex();   // 詞法分析器函數聲明

// 生成三位址碼的函數，接受操作符、兩個操作數和結果作為參數
void generate_code(const char* op, const char* arg1, const char* arg2, const char*
result) {
    if (arg2 == NULL) {
        printf("%s = %s\n", result, arg1);
    } else {
        printf("%s = %s %s %s\n", result, arg1, op, arg2);
    }
}

// 生成打印語句的函數，接受要打印的變量名作為參數
void generate_print(const char* var) {
    printf("print %s\n", var);
}

%}

%union {
```

```
    int intval;     // 整數類型的共用體成員
    char* strval; // 字符串類型的共用體成員
}

%token <intval> NUMBER     // 定義標記 NUMBER 的屬性為 intval
%token <strval> IDENTIFIER    // 定義標記 IDENTIFIER 的屬性為 strval
%type <strval> expression term factor statement    // 定義規則的返回類型為 strval

%token PLUS MINUS MULT DIV ASSIGN PRINT LPAREN RPAREN    // 定義操作符和關鍵字的標記

%%

program: statements    // 程序由語句序列組成
        ;

statements: statements statement    // 一個或多個語句組成語句序列
          | statement
          ;

statement: IDENTIFIER ASSIGN expression {    // 賦值語句，生成賦值三位址碼
                generate_code("=", $3, NULL, $1);
            }
          | PRINT LPAREN IDENTIFIER RPAREN {    // 打印語句，生成打印三位址碼
                generate_print($3);
            }
          ;

expression: expression PLUS term {    // 表達式的加法運算，生成加法三位址碼
                char temp[20];
                sprintf(temp, "t%d", tempVarCount++);
                generate_code("+", $1, $3, temp);
                $$ = strdup(temp);
            }
          | expression MINUS term {    // 表達式的減法運算，生成減法三位址碼
                char temp[20];
                sprintf(temp, "t%d", tempVarCount++);
                generate_code("-", $1, $3, temp);
                $$ = strdup(temp);
            }
          | term {    // 單個項
                $$ = $1;
            }
```

```
        ;

term: term MULT factor {   // 項的乘法運算，生成乘法三位址碼
                char temp[20];
                sprintf(temp, "t%d", tempVarCount++);
                generate_code("*", $1, $3, temp);
                $$ = strdup(temp);
            }
    | term DIV factor {   // 項的除法運算，生成除法三位址碼
                char temp[20];
                sprintf(temp, "t%d", tempVarCount++);
                generate_code("/", $1, $3, temp);
                $$ = strdup(temp);
            }
    | factor {   // 單個因子
                $$ = $1;
            }
    ;

factor: NUMBER {   // 數字因子，直接返回其字符串形式
                char temp[20];
                sprintf(temp, "%d", $1);
                $$ = strdup(temp);
            }
    | IDENTIFIER {   // 變量因子，直接返回其字符串形式
                $$ = $1;
            }
    ;
%%

void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}

int main() {
    yyparse();   // 語法分析
    return 0;
}
```

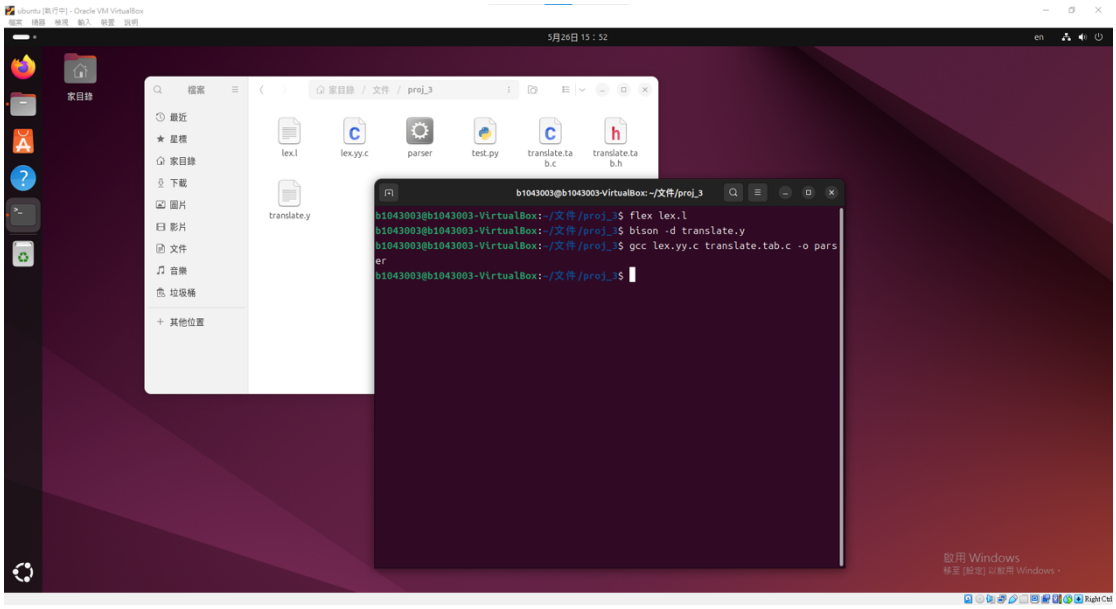
在撰寫完 translate.y 文件後執行 bison -d translate.y 得到 translate.tab.c 和 translate.tab.h 檔案，如圖二。



圖二、執行 bison -d translate.y 畫面

Step 3. 使用 gcc 編譯檔案成可執行文件：

在經過上述步驟得到 lex.yy.c 和 translate.tab.c 文件後執行 gcc lex.yy.c translate.tab.c -o parser 的到一個名為 parser 的文件，如圖三。



圖三、執行 gcc lex.yy.c translate.tab.c -o parser 畫面

Step 4. 使用 gcc 編譯檔案成可執行文件：

最後將要測試的 Python 檔案輸入進 parser 中，可得到三位址程式碼 (Three-Address Code)，如圖四。

輸入之 Python 文件內容：

```
a = 3 + 5
b = a * 2
c = b - 4
```

```
d = c / 2
e = d + a
print(a)
print(b)
print(c)
print(d)
print(e)
```

```
b1043003@b1043003-VirtualBox:~/文件/proj_3$ ./parser<test.py
t0 = 3 + 5
a = t0
t1 = a * 2
b = t1
t2 = b - 4
c = t2
t3 = c / 2
d = t3
t4 = d + a
e = t4
print a
print b
print c
print d
print e
```

圖四、執行 ./parser<test.py 畫面

由輸出之三位只程式碼中可看出，程式會先計算 a, b, c, d, e 的結果，過程如下:

t0 = 3 + 5： 計算 3 + 5，結果是 8，並將其賦值給 t0。
a = t0： 將 t0 的值賦給 a，即 a = 8。
t1 = a * 2： 計算 a * 2，結果是 8 * 2 = 16，並將其賦值給 t1。
b = t1： 將 t1 的值賦給 b，即 b = 16。
t2 = b - 4： 計算 b - 4，結果是 16 - 4 = 12，並將其賦值給 t2。
c = t2： 將 t2 的值賦給 c，即 c = 12。
t3 = c / 2： 計算 c / 2，結果是 12 / 2 = 6，並將其賦值給 t3。
d = t3： 將 t3 的值賦給 d，即 d = 6。
t4 = d + a： 計算 d + a，結果是 6 + 8 = 14，並將其賦值給 t4。
e = t4： 將 t4 的值賦給 e，即 e = 14。

在計算完後使用 print 印出 a, b, c, d, e 的值。