

COMPILER DESIGN AND IMPLEMENTATION

Programming Assignment #1

Professor Chung Yung, National Dong Hwa University

SELF-STUDY MATERIAL

- ❖ Suggested Steps
- ❖ Programming Environment
- ❖ FLEX
- ❖ A Practical Flex Example
- ❖ Basics of Bison
- ❖ A Practical Bison Example

P1.1

SUGGESTED STEPS FOR DOING ASSIGNMENT #1

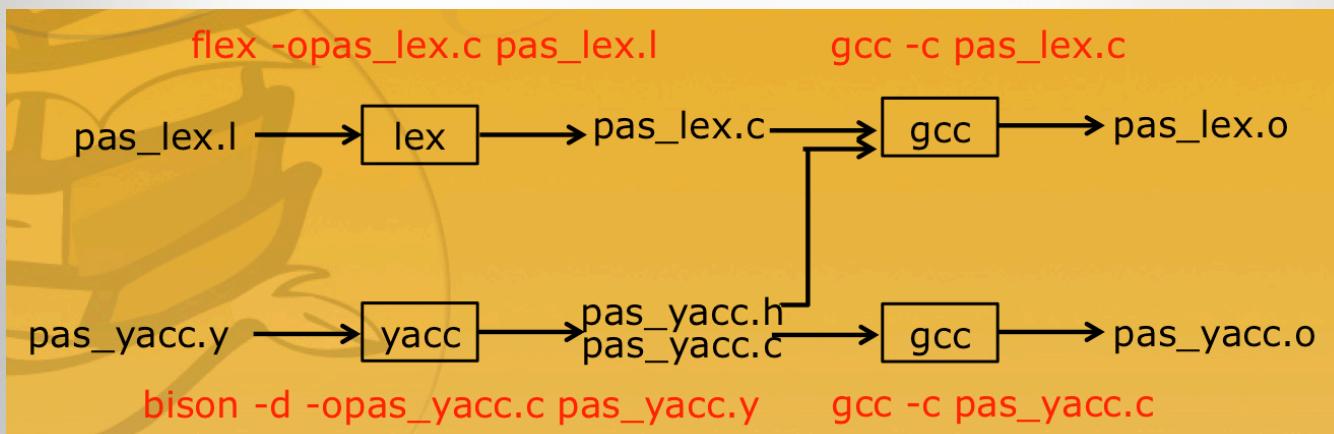
WHAT TO DO

- ❖ Write a lex specification **mytiny_lex.l** (see pas_lex.l).
- ❖ Write a yacc specification **mytiny_yacc.l** (see pas_yacc.y).
- ❖ Write a main function **main.c** (see main.c).
- ❖ Write a header file **mytiny.h** (see pascal.h)

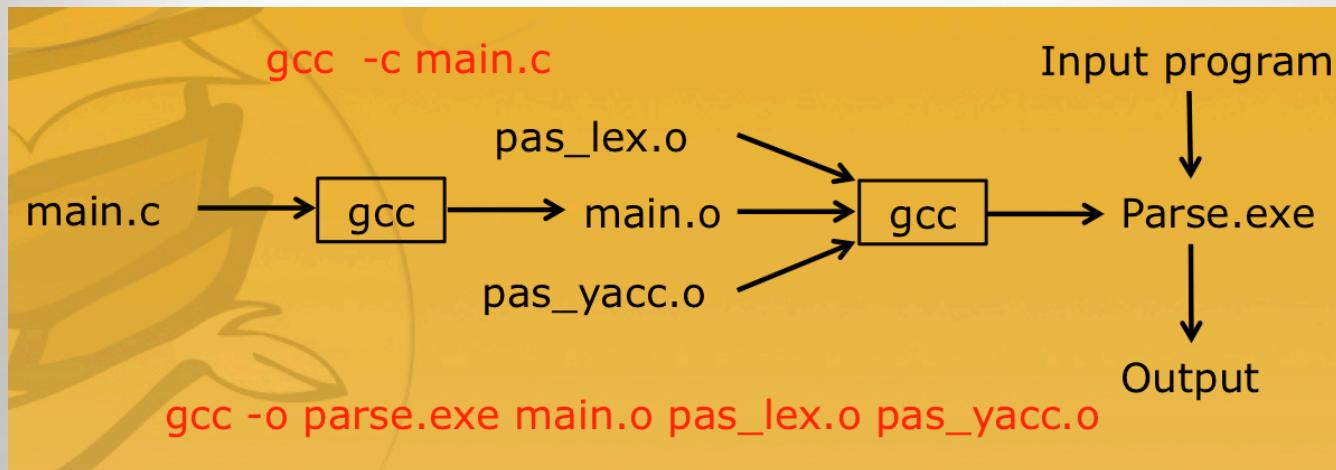
SUGGESTED PROCESS

1. Use bison to compile **mytiny_yacc.y** into **mytiny_yacc.c**.
The **-d** switch produces a header file **mytiny_yacc.h**.
2. Use flex to compile **mytiny_lex.l** into **mytiny_lex.c**.
3. Use gcc to compile **mytiny_lex.c** into **mytiny_lex.o**,
mytiny_yacc.c into **mytiny_yacc.o**, and **main.c** into **main.o**.
4. Use gcc to link **main.o**, **pas_lex.o**, and **pas_yacc.o** into
parse.exe.

GENERATING C CODE



BUILDING EXECUTABLE



P1.2

PROGRAMMING ENVIRONMENT

SOFTWARE

Software needed

- ❖ Flex 2.5.4
- ❖ Bison 2.4.1
- ❖ MinGW 4.6.2

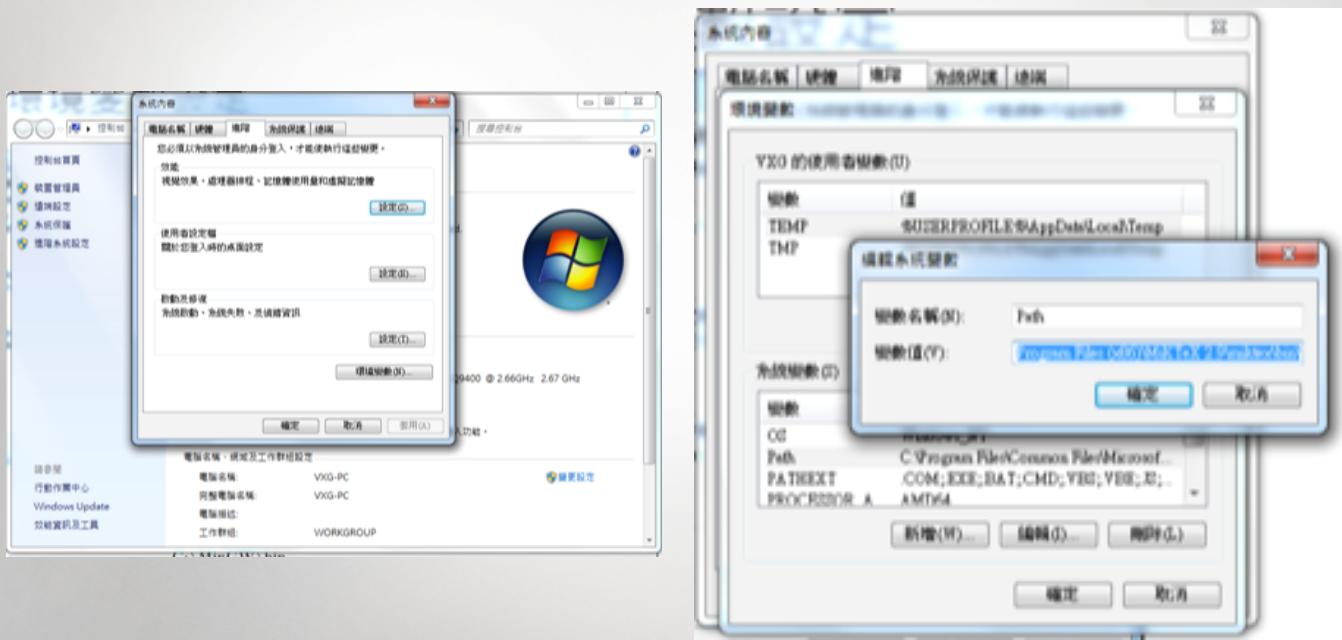
Notes

- ❖ DO NOT install both MinGW and DevC.
- ❖ Check and set up environment variables.
- ❖ Rebooting after installation is suggested.

ENVIRONMENT VARIABLES

- ❖ Check whether **C:\MinGW\bin** is in **PATH**. If not, add it into **PATH**, using ; as separators.
- ❖ In case of using **DevC**, check for **C:\Dev-Cpp\bin** instead.

IN WINDOWS



TEST INSTALLATION

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\fdas>gcc --version
gcc (GCC) 4.6.2
Copyright (C) 2011 Free Software Foundation, Inc.
本程式是自由軟體；請參看來源程式碼的版權宣告。本軟體沒有任何擔保；  
包括沒有適銷性和某一專用目的下的適用性擔保。

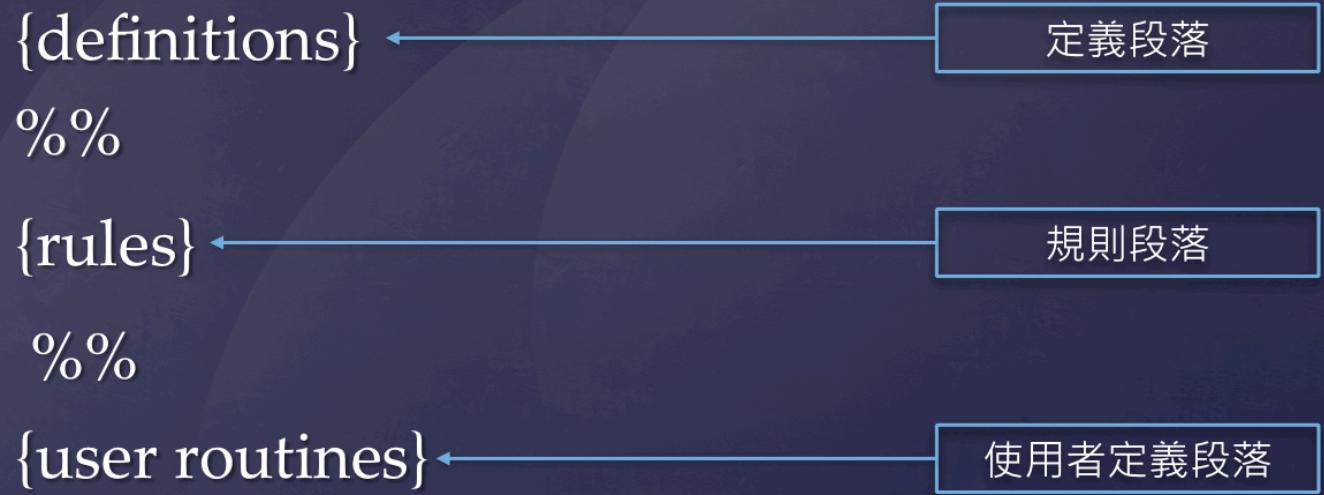
C:\Documents and Settings\fdas>flex -V
flex version 2.5.4

C:\Documents and Settings\fdas>
```

P1.2

FLEX

FLEX SPECIFICATION



Char	Meaning
.	Any character except for '\n'
-	The range of characters, e.g., a-z
*	Repeat for 0 or more times
+	Repeat for 1 or more times
?	Appear either 0 or 1 time
^	Negation
A B	A or B
"sth."	Exactly the characters appeared in parentheses (sth.)

P1.3

A PRACTICAL FLEX EXAMPLE

FLEX COMPILATION

- ❖ **flex -opas_lex.c pas_lex.l**
- ❖ Translate **pas_lex.l** into **pas_lex.c**.
(Specify the filename of your own.)
- ❖ There is **NO** space between **-o** and
pas_lex.c.

FOR YOUR ASSIGNMENT

- ❖ Write your own mytiny.lex according to the description of mytiny lexicons.
- ❖ **flex -o mytiny_lex.c mytiny_lex.l**

GCC COMPILEMENT

- ❖ **gcc -c -o pas_lex.o pas_lex.c**
- ❖ **gcc -c -o main.o main.c**
- ❖ **gcc -o scanner.exe pas_lex.o main.o**
- ❖ There **IS** a space between **-o** and **pas_lex.o**.

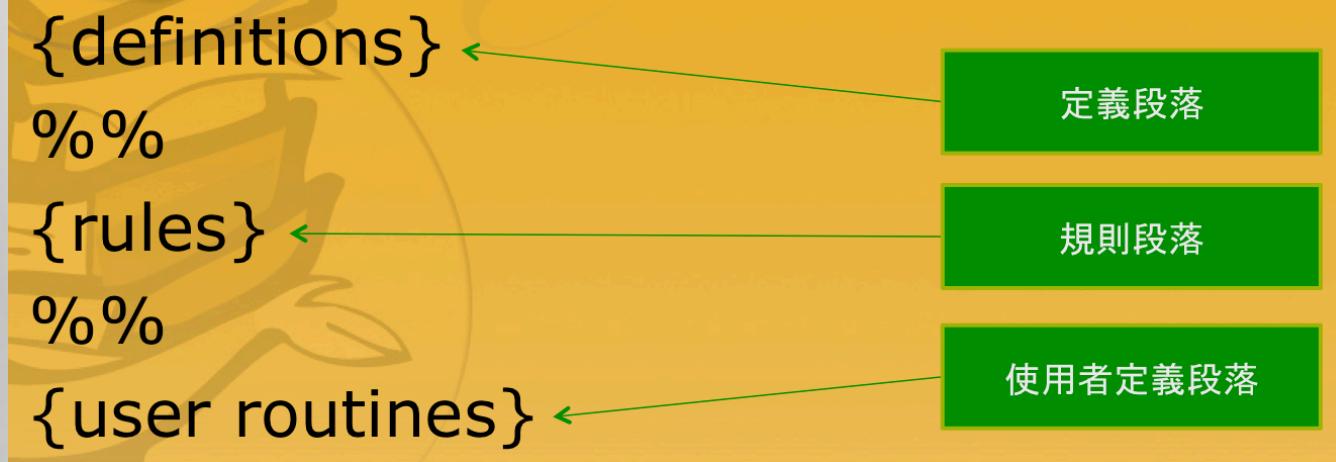
EXECUTION

- ❖ **scanner.exe input.pas**
- ❖ **input.pas** can be some other test file names given by TA.
- ❖ According the environment setup, you may need to use command as follows in some cases.
./scanner.exe input.pas

P1.5

BASICS OF BISON

A BISON SPECIFICATION



EXAMPLE: DEFINITIONS

```
1  %{
2      #include <stdio.h>
3      #include <stdlib.h>
4      #include <string.h>
5      #include "pascal.h"
6  %}
7  %token PROG PROC BG END INT BOOL TRUE
8  %token LP RP LSP RSP FALSE CC
9  %token DOT SEMI VAR ARRAY OF DOTDOT
10 %token IF THEN READ WRITE WHILE DO
11 %token ELSE ASSIGN COMMA COLON ID NUM
12 %left OR AND
13 %left NOT
14 %left EQ NE LT GT LE GE
15 %left ADD MINUS
16 %left DIV TIMES
17 %expect 1
18
19 %%
```

定義段落

EXAMPLE: RULE

規則段落

```
19 %%  
20 prog : PROG ID SEMI block DOT {printf("prog => PROG ID SEMI block DOT \n*****");}  
21 | {printf("*****Parsing failed! \n");}  
22 ;  
23  
24 block : vardecs prodecs stmts {printf("block=>vardecs prodecs stmts \n");}  
25 ;  
26  
27 vardecs : VAR vardec SEMI morevd {printf("vardecs => VAR vardec SEMI morevd \n");}  
28 | {printf("vardecs => Null \n");}  
29 ;  
30  
31 morevd : vardec SEMI morevd {printf("morevd => vardec SEMI morevd \n");}  
32 | {printf("morevd => Null \n");}  
33 ;  
34  
35 vardec : ID moreid COLON type {printf("vardec => ID moreid COLON type\n");}  
36 ;  
37  
38 moreid : COMMA ID moreid {printf("moreid => COMMA ID moreid \n");}  
39 | {printf("moreid => Null \n");}  
40 ;  
41
```

P1.6

A PRACTICAL BISON EXAMPLE

COMMANDS USED

❖ Bison

```
bison -d -o pas_yacc.c pas_yacc.y
```

❖ Flex

```
flex -opas_lex.c pas_lex.l
```

❖ GCC

```
gcc -c pas_lex.c
```

```
gcc -c pas_yacc.c
```

```
gcc -c main.c
```

```
gcc -o parse.exe pas_lex.o main.o pas_yacc.o
```

EXECUTION

❖ To run your parser:

parser.exe input.pas

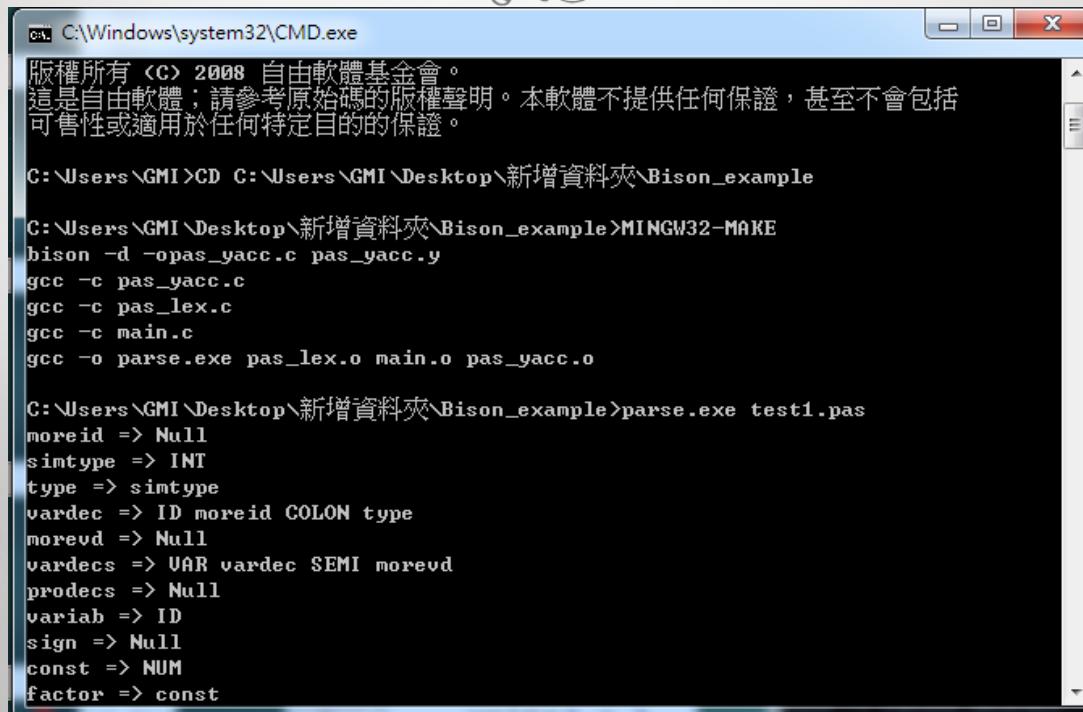
FOR YOUR ASSIGNMENT

❖ Write a mytiny.y according to the MyTiny grammar.

❖ Build a mytiny.exe and run

./mytiny.exe test.t

BUILDING EXECUTABLE

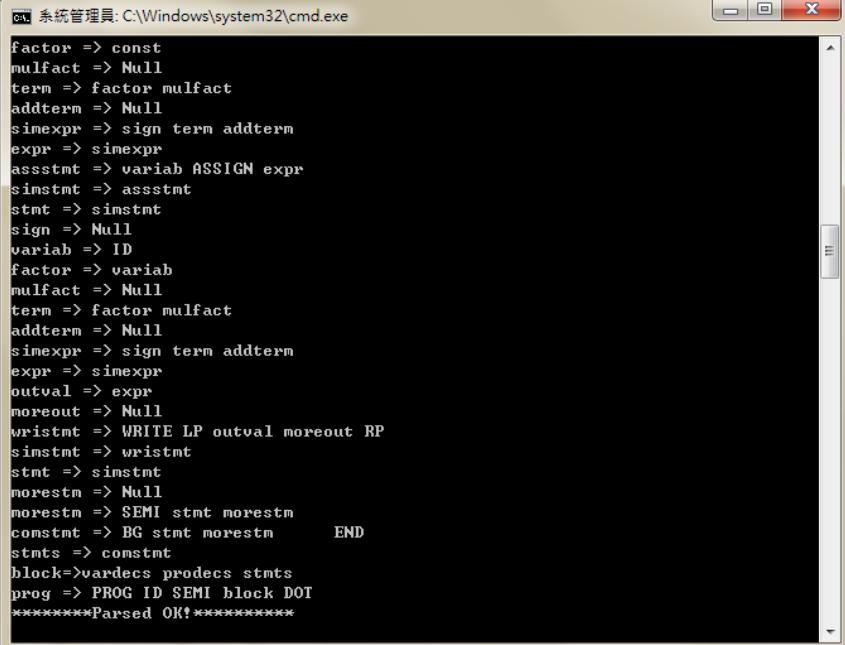


A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\CMD.exe". The window displays the following text:

```
版權所有 <C> 2008 自由軟體基金會。  
這是自由軟體；請參考原始碼的版權聲明。本軟體不提供任何保證，甚至不會包括  
可售性或適用於任何特定目的的保證。  
  
C:\Users\GMI>CD C:\Users\GMI\Desktop\新增資料夾\Bison_example  
  
C:\Users\GMI\Desktop\新增資料夾\Bison_example>MINGW32-MAKE  
bison -d -opas_yacc.c pas_yacc.y  
gcc -c pas_yacc.c  
gcc -c pas_lex.c  
gcc -c main.c  
gcc -o parse.exe pas_lex.o main.o pas_yacc.o  
  
C:\Users\GMI\Desktop\新增資料夾\Bison_example>parse.exe test1.pas  
moreid => Null  
simtype => INT  
type => simtype  
vardec => ID moreid COLON type  
morevd => Null  
vardecs => VAR vardec SEMI morevd  
prodecs => Null  
variab => ID  
sign => Null  
const => NUM  
factor => const
```

EXECUTING YOUR PROGRAM

```
1 program test1;
2 var a: integer;
3 begin
4     a := 3;
5     write( a )
6 end.
```



The screenshot shows a Windows command prompt window titled "系统管理员: C:\Windows\system32\cmd.exe". The window displays the results of a parser for a simple program. The parser's internal state is shown at the top, followed by the tokens and their corresponding grammar rules. The final message "*****Parsed OK!*****" indicates successful parsing.

```
factor => const
multfact => Null
term => factor multfact
addterm => Null
simexpr => sign term addterm
expr => simexpr
assstmt => variab ASSIGN expr
simstmt => assstmt
stmt => simstmt
sign => Null
variab => ID
factor => variab
multfact => Null
term => factor multfact
addterm => Null
simexpr => sign term addterm
expr => simexpr
outval => expr
moreout => Null
wristmt => WRITE LP outval moreout RP
simstmt => wristmt
stmt => simstmt
morestm => Null
morestm => SEMI stmt morestm
constmt => BG stmt morestm      END
stmts => constmt
block=>vardecs prodecst statms
prog => PROG ID SEMI block DOT
*****Parsed OK!*****
```

REFERENCES

- ❖ <http://www.mingw.org/>
- ❖ <http://flex.sourceforge.net/>
- ❖ <http://www.gnu.org/software/bison/>
- ❖ <http://sourceforge.net/projects/gnuwin32>

Now,

START DOING YOUR OWN ASSIGNMENT