

Compiler Construction

Programming Assignment 2

Syntactic and Semantic Definitions for μGo

NCKU
National Cheng Kung University

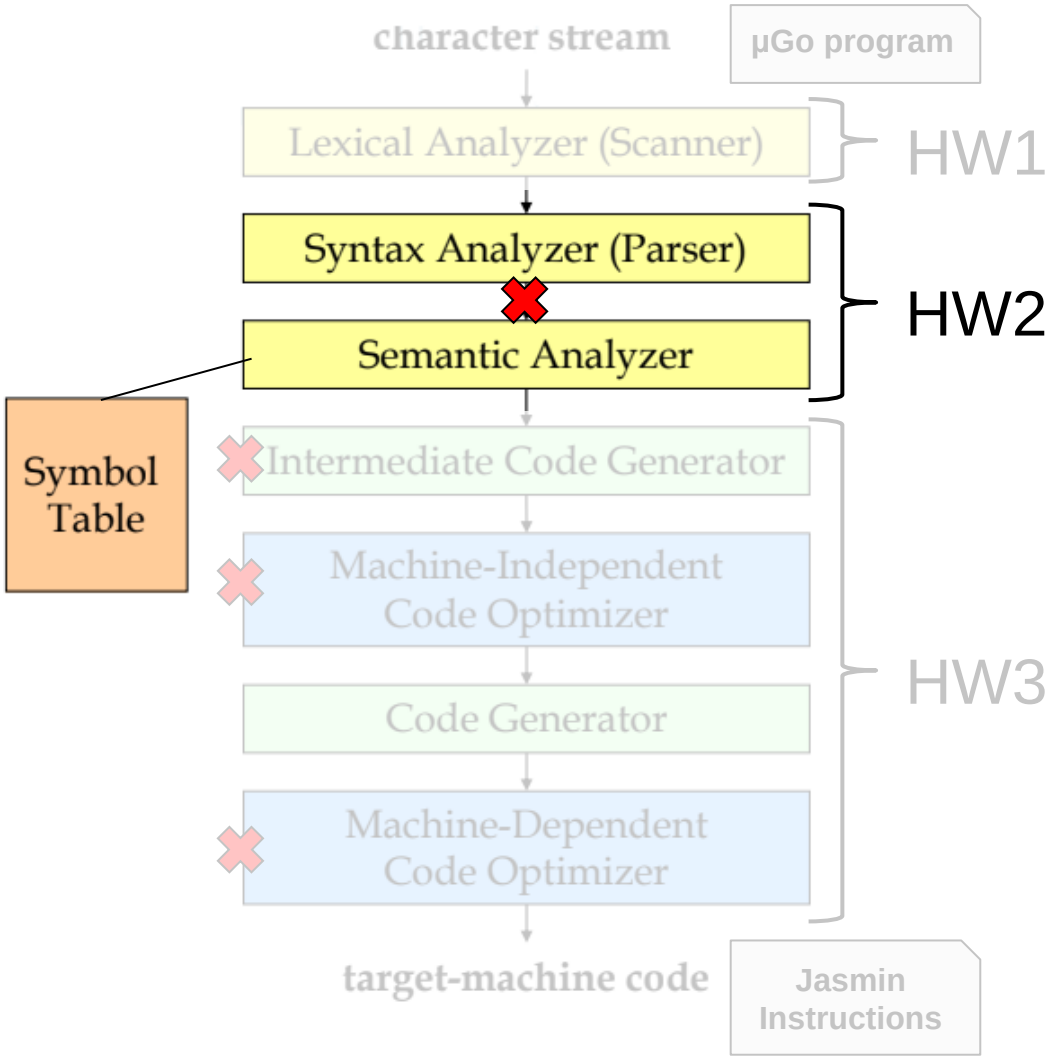


What to do in this Assignment?

- Write an *LALR(1)* parser for μGO using Lex and Yacc.
- The parser supports print I/O, arithmetic operations, and some programming language basic concepts.
- The spec of μGO is available for your reference.
- You need to design grammar for your own parser by following the given spec.
- You also need to check semantic correctness by implementing *symbol table*.



Project Outline





Assignment Requirements

- Each test case is 10pt and the total score is 110pt.
- You can judge your code locally with the attached judger.

```
$ ls
judge/  common.h  compiler_hw2.l  compiler_hw2.y  judge.conf  Makefile
$ python3 judge/judge.py
====================+============================================================
            Sample | Accept
====================+============================================================
    in01_arithmetic | ✔
====================+============================================================
    in02_precedence | ✔
====================+============================================================
        in03_scope | ✔
====================+============================================================
        in04_array | ✔
====================+============================================================
    in05_assignment | ✔
====================+============================================================
    in06_conversion | ✔
====================+============================================================
           in07_if | ✔
====================+============================================================
          in08_for | ✔
====================+============================================================
    in09_type_error | ✔
====================+============================================================
in10_variable_error | ✔
====================+============================================================
       in11_monster | ✔
====================+============================================================
Correct rate: 100.0%
Obtained/Total scores: 110.0/110
```

```
// "Hard Coding" will get 0pt.
main() {
    result = read(answer_file);
    print(result);
}
```

https://github.com/aben20807/local-judge/tree/master/examples/student



Scoping

- What will this program print with proper scoping?

```
var x int32 = 10
{
  var x int32 = 5
  x++
  println(x)
}
println(x)
{
  x++
  println(x)
}
println(x)
```

- Output

```
6
10
11
11
```



Scoping (cont.)

```
var x int32 = 10
{
  var x int32 = 5
  x++
  println(x)
}
println(x)
{
  x++
  println(x)
}
println(x)
```

Scope Block A

Scope Block B

Scope Block A

Scope Block C

Scope Block A

Scope B's x

Access scope A's x



Scoping (cont.)

- A scope block is a set of statements enclosed within left and right braces ({ and }).
- A variable declared in a block is accessible in the block and all the inner blocks of that block, but not accessible outside the block.
- Different inner scope block in same scope block can't see each other.
- You can declare variable with same name in different scope.



Symbol table

- Fields

Index	Name	Type	Address	Lineno	Element type
0	x	int32	0	1	-
1	y	float32	1	2	-
2	z	array	2	3	int32

- Functions
  - **create_symbol**: Create a symbol table when entering a new scope.
  - **insert_symbol**: Insert entries for variables declarations.
  - **lookup_symbol**: Look up entries in the symbol table.
  - **dump_symbol**: Dump all contents in the symbol table of current scope and its entries when exiting a scope.

Note:

- Function names and their parameters can be properly defined by yourself.



Symbol table

```
1 var x int32 = 10
2 var y float32
3
4 {
5   var x float32 = 3.14
6 }
7 var z [87]int32
```

Insert x into symbol table (scope level: 0)

Insert y into symbol table (scope level: 0)

Insert x into symbol table (scope level: 1)

Insert z into symbol table (scope level: 0)



Symbol table (cont.)

```
1 var x int32 = 10
2 var y float32
3
4 {
5   var x float32 = 3.14
6 }
7 var z [87]int32
```

Index: unique in each symbol table
Address: unique in whole program

Dump scope level 1's symbol table:

Index	Name	Type	Address	Lineno	Element type
0	x	float32	2	5	-

Dump scope level 0's symbol table:

Index	Name	Type	Address	Lineno	Element type
0	x	int32	0	1	-
1	y	float32	1	2	-
2	z	array	3	7	int32



Handle semantic error

```
1 var x int32
2
3 var z float32
4 var x int32
5 y = 8
```

error:4: x redeclared in this block. previous declaration at line 1

error:5: undefined: y

Submission

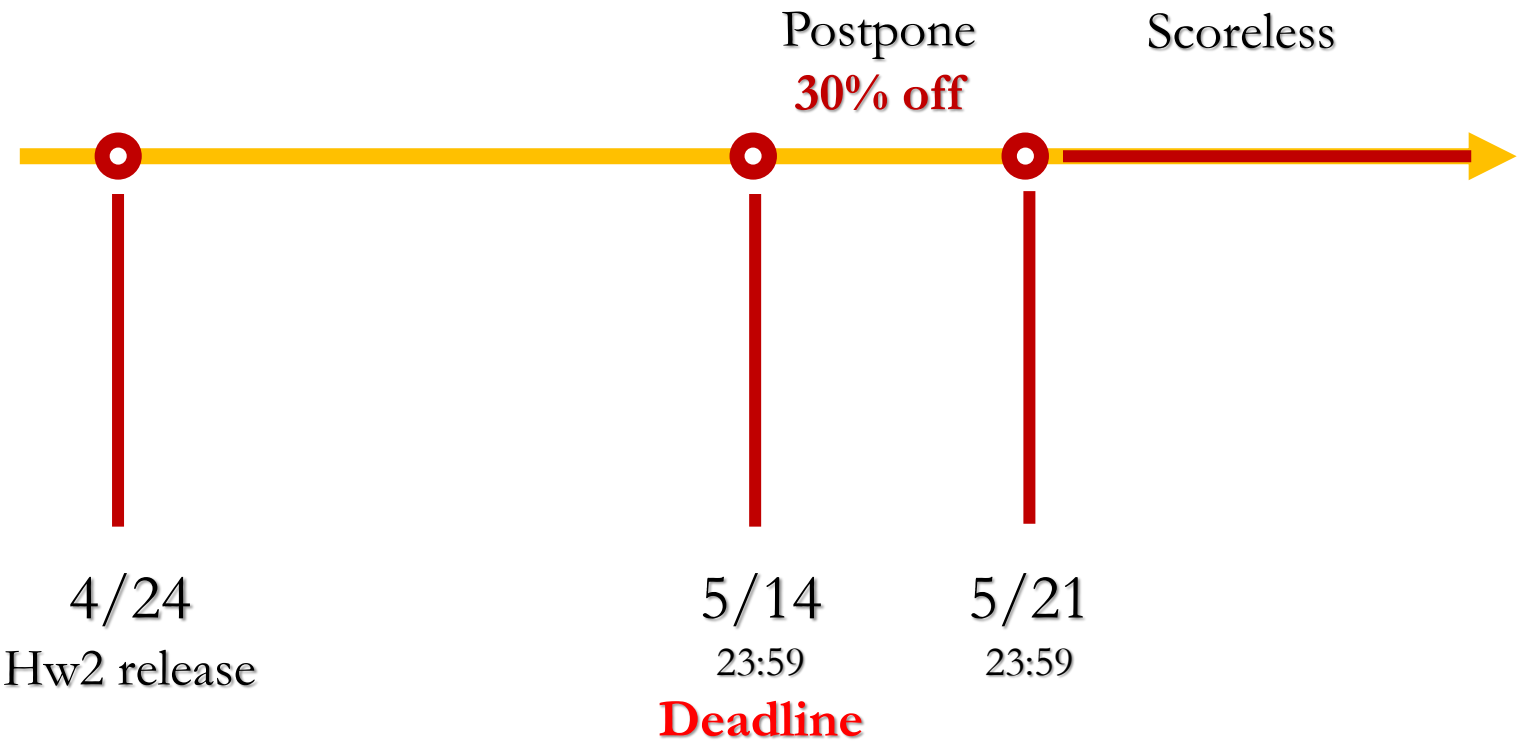
- Upload your homework to Moodle.
- The expected arrangement of your codes:
  - Only **.zip** and **.rar** types of compression are allowed.
  - The directory should be organized as:

```
Compiler_StudentID_HW2.zip/
└── Compiler_StudentID_HW2/
    ├── compiler_hw2.l
    ├── compiler_hw2.y
    ├── common.h
    └── Makefile
```

  - You will lose 10pt if your programs were uploaded in incorrect format!!!



Deadline


Postpone
30% off
Scoreless
4/24
Hw2 release
5/14
23:59
Deadline
5/21
23:59



How to Mail TAs

- Send mail to asrlab@csie.ncku.edu.tw, not any TA's mail!!
- Email subject starts with "[Compiler2020]"



QUESTIONS ?