



# Bangladesh Agricultural University

Department of Bioinformatics Engineering

Course Code & Title: CSM 3222 Compiler Lab

Level-3, Semester-2, July-December/2024

Deadline: 07 February, 2026 11:59 PM    Lab 6 Tasks    Platform: Google Classroom

---

This lab assignment focuses on the construction of Three Address Code and Assembly for context-free grammars. All code should be uploaded to your GitHub repository in a folder named "Lab 6 (Code Generation)". Please provide the GitHub link in your report. For each task in your report, you must include the following sections:

- Objective
  - Grammar
  - Requirements
  - Installation and Set-up
  - Implementation with GitHub Link
  - Input and Output
  - Working Principles
- 

**Task 1** Write a program that reads statements (single or multiple) from input.txt, which may include arithmetic, assignment, and logical operations, and generates three-address and Assembly code for each statement. Your program should handle new lines, operator precedence, and all the operators: +, -, \*, /, \*\* (exponentiation), // (integer division), + =, - =, \* =, / =, % =, \*\* =, &&, ||, !.

Grammar:

$$\begin{aligned} \text{Program} &\rightarrow \text{StatementList} \\ \text{StatementList} &\rightarrow \text{Statement} \mid \text{StatementList NEWLINE Statement} \\ \text{Statement} &\rightarrow \text{ID ' = ' Expression} \mid \text{ID OpAssign Expression} \\ \text{OpAssign} &\rightarrow \text{' + = '} \mid \text{' - = '} \mid \text{' * = '} \mid \text{' / = '} \mid \text{' \% = '} \mid \text{' ** = '} \\ \text{Expression} &\rightarrow \text{Expression ' + ' Term} \mid \text{Expression ' - ' Term} \mid \text{Term} \\ \text{Term} &\rightarrow \text{Term ' * ' Factor} \mid \text{Term ' / ' Factor} \mid \text{Term ' // ' Factor} \mid \text{Factor} \\ \text{Factor} &\rightarrow \text{Factor ' * * ' Unary} \mid \text{Unary} \\ \text{Unary} &\rightarrow \text{' ! 'Unary} \mid \text{' - 'Unary} \mid \text{Primary} \\ \text{Primary} &\rightarrow \text{ID} \mid \text{NUM} \mid \text{' ('Expression')'} \\ \text{ID} &\rightarrow [a - zA - Z\_][a - zA - Z0 - 9]* \\ \text{NUM} &\rightarrow [0 - 9]+ \\ \text{NEWLINE} &\rightarrow \text{'\n'}$$



### Sample Input (input.txt):

```

1 a = 5 + 3
2 b += a * 2
3 c = !b || 0
4 d = a ** 2
5 e //= 3
6 f = (a + b) * (c - d)
  / e
7 g %= (f ** 2) + 1
8 h = !((a > b) && (c <
  d)) || e
9 i **= 2
10 j = i // (a + b * c)

```

### Sample Output:

```

1 t1 = 5 + 3
2 a = t1
3 t2 = a * 2
4 b = b + t2
5 t3 = !b
6 t4 = t3 || 0
7 c = t4
8 t5 = a ** 2
9 d = t5
10 t6 = e // 3
11 e = t6
12 t7 = a + b
13 t8 = c - d
14 t9 = t7 * t8
15 t10 = t9 / e
16 f = t10
17 t11 = f ** 2
18 t12 = t11 + 1
19 g = f % t12
20 t13 = a > b
21 t14 = c < d
22 t15 = t13 && t14
23 t16 = !t15
24 t17 = t16 || e
25 h = t17
26 t18 = i ** 2
27 i = t18
28 t19 = b * c
29 t20 = a + t19
30 t21 = i // t20
31 j = t21

```

```

1 MOV R0, #5
2 ADD R0, #3
3 MOV a, R0
4
5 MOV R0, a
6 MUL R0, #2
7 MOV R1, b
8 ADD R1, R0
9 MOV b, R1
10
11 MOV R0, b
12 NOT R0
13 OR R0, #0
14 MOV c, R0
15
16 MOV R0, a
17 POW R0, #2
18 MOV d, R0
19
20 MOV R0, e
21 IDIV R0, #3
22 MOV e, R0
23
24 MOV R0, a
25 ADD R0, b
26
27 MOV R1, c
28 SUB R1, d
29
30 MUL R0, R1
31 DIV R0, e
32 MOV f, R0
33
34 MOV R0, f
35 POW R0, #2
36 ADD R0, #1
37
38 MOV R1, f
39 MOD R1, R0
40 MOV g, R1
41
42 MOV R0, a
43 CMPGT R0, b
44
45 MOV R1, c
46 CMPLT R1, d
47
48 AND R0, R1
49 NOT R0
50 OR R0, e
51 MOV h, R0
52
53 MOV R0, i
54 POW R0, #2
55 MOV i, R0
56
57 MOV R0, b
58 MUL R0, c
59
60 MOV R1, a
61 ADD R1, R0
62
63 MOV R2, i
64 IDIV R2, R1
65 MOV j, R2

```



**Task 2** Write a compiler frontend that reads arithmetic statements with math functions from input.txt (handle new lines). The statements can include basic operators +, -, \*, /, %, parentheses, and math functions like sqrt(), pow(), log(), exp(), sin(), cos(), tan(), abs(). Generate three-address code (TAC) + Assembly code for each statement.

Grammar:

$$\begin{aligned}
 \textit{Program} &\rightarrow \textit{StatementList} \\
 \textit{StatementList} &\rightarrow \textit{Statement} \mid \textit{StatementList NEWLINE Statement} \\
 \textit{Statement} &\rightarrow \textit{ID} ' = ' \textit{Expression} \\
 \textit{Expression} &\rightarrow \textit{Expression} ' + ' \textit{Term} \\
 &\mid \textit{Expression} ' - ' \textit{Term} \\
 &\mid \textit{Term} \\
 \textit{Term} &\rightarrow \textit{Term} ' * ' \textit{Factor} \\
 &\mid \textit{Term} ' / ' \textit{Factor} \\
 &\mid \textit{Term} ' \% ' \textit{Factor} \\
 &\mid \textit{Factor} \\
 \textit{Factor} &\rightarrow \textit{FunctionCall} \\
 &\mid ' ( \textit{Expression} ) ' \\
 &\mid \textit{ID} \\
 &\mid \textit{NUM} \\
 &\mid ' - ' \textit{Factor} \\
 \textit{FunctionCall} &\rightarrow \textit{sqrt} ' ( \textit{Expression} ) ' \\
 &\mid \textit{pow} ' ( \textit{Expression} ' , ' \textit{Expression} ) ' \\
 &\mid \textit{log} ' ( \textit{Expression} ) ' \\
 &\mid \textit{exp} ' ( \textit{Expression} ) ' \\
 &\mid \textit{sin} ' ( \textit{Expression} ) ' \\
 &\mid \textit{cos} ' ( \textit{Expression} ) ' \\
 &\mid \textit{tan} ' ( \textit{Expression} ) ' \\
 &\mid \textit{abs} ' ( \textit{Expression} ) ' \\
 \textit{ID} &\rightarrow [a-zA-Z][a-zA-Z0-9_]* \\
 \textit{NUM} &\rightarrow [0-9]^+ \\
 \textit{NEWLINE} &\rightarrow '\n'
 \end{aligned}$$



### Sample Input (input.txt):

```
1 a = 9
2 b = sqrt(a)
3 c = pow(a, 3)
4 d = log(b) + sin(a)
5 e = cos(c) * tan(d)
6 f = abs(-a + b) / exp
  (2)
```

### Sample Output:

```
1 a = 9
2 t1 = sqrt(a)
3 b = t1
4 t2 = pow(a, 3)
5 c = t2
6 t3 = log(b)
7 t4 = sin(a)
8 t5 = t3 + t4
9 d = t5
10 t6 = cos(c)
11 t7 = tan(d)
12 t8 = t6 * t7
13 e = t8
14 t9 = -a
15 t10 = t9 + b
16 t11 = abs(t10)
17 t12 = exp(2)
18 t13 = t11 / t12
19 f = t13
```

```
1 MOV R0, #9
2 MOV a, R0
3
4
5 MOV R0, a
6 SQRT R0
7 MOV b, R0
8
9 MOV R0, a
10 POW R0, #3
11 MOV c, R0
12
13 MOV R0, b
14 LOG R0
15
16 MOV R1, a
17 SIN R1
18
19 ADD R0, R1
20 MOV d, R0
21
22 MOV R0, c
23 COS R0
24
25 MOV R1, d
26 TAN R1
27
28 MUL R0, R1
29 MOV e, R0
30
31 MOV R0, a
32 NEG R0
33
34 ADD R0, b
35 ABS R0
36
37 MOV R1, #2
38 EXP R1
39
40 DIV R0, R1
41 MOV f, R0
```