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:

a) Objective

b) Grammar

c) Requirements

d) Installation and Set-up

e) Implementation with GitHub Link

f) Input and Output

g) 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:

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

Sample Input (input.txt):

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

Sample Output:

```
t1 = 5 + 3
a = t1
t2 = a * 2
b = b + t2
t3 = !b
t4 = t3 || 0
c = t4
t5 = a ** 2
d = t5
t6 = e // 3
e = t6
t7 = a + b
t8 = c - d
t9 = t7 * t8
t10 = t9 / e
f = t10
t11 = f ** 2
t12 = t11 + 1
g = f % t12
t13 = a > b
t14 = c < d
t15 = t13 && t14
t16 = !t15
t17 = t16 || e
h = t17
t18 = i ** 2
i = t18
t19 = b * c
t20 = a + t19
t21 = i // t20
j = t21
```

```
MOV R0, #5
ADD R0, #3
MOV a, R0

MOV R0, a
MUL R0, #2
MOV R1, b
ADD R1, R0
MOV b, R1

MOV R0, b
NOT R0
OR  R0, #0
MOV c, R0

MOV R0, a
POW R0, #2
MOV d, R0

MOV R0, e
IDIV R0, #3
MOV e, R0

MOV R0, a
ADD R0, b

MOV R1, c
SUB R1, d

MUL R0, R1
DIV R0, e
MOV f, R0

MOV R0, f
POW R0, #2
ADD R0, #1

MOV R1, f
MOD R1, R0
MOV g, R1

MOV R0, a
CMPGT R0, b

MOV R1, c
CMPLT R1, d

AND R0, R1
NOT R0
OR  R0, e
MOV h, R0

MOV R0, i
POW R0, #2
MOV i, R0

MOV R0, b
MUL R0, c

MOV R1, a
ADD R1, R0

MOV R2, i
IDIV R2, R1
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:

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

Sample Input (input.txt):

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

Sample Output:

```
a = 9
t1 = sqrt(a)
b = t1
t2 = pow(a, 3)
c = t2
t3 = log(b)
t4 = sin(a)
t5 = t3 + t4
d = t5
t6 = cos(c)
t7 = tan(d)
t8 = t6 * t7
e = t8
t9 = -a
t10 = t9 + b
t11 = abs(t10)
t12 = exp(2)
t13 = t11 / t12
f = t13
```

```
MOV R0, #9
MOV a, R0


MOV R0, a
SQRT R0
MOV b, R0

MOV R0, a
POW R0, #3
MOV c, R0

MOV R0, b
LOG R0

MOV R1, a
SIN R1

ADD R0, R1
MOV d, R0

MOV R0, c
COS R0

MOV R1, d
TAN R1

MUL R0, R1
MOV e, R0

MOV R0, a
NEG R0

ADD R0, b
ABS R0

MOV R1, #2
EXP R1

DIV R0, R1
MOV f, R0
```