Lab Performance Evaluation 2 Syllabus

Course Outcome

CO4: Designing back end of compiler (Intermediate Code Generation and Code Generation) using LEX and YACC.

Marks Distribution:

Lab Class	Questions	Question Types	Time	Marks	Difficulty	CO
Lab Class 3	Q1	x86 Assembly Program **Code Example: ASM (Link)**	17 min	60%	Easy	CO4
Lab Class 4	Q2 (a)	Intermediate Code Generation **Code Example: IR_and_CG_As_Compiler (Link)**	30 min	20%	Hard	
	Q2 (b)	Code Generation **Code Example: IR_and_CG_As_Compiler (Link)**		20%	Hard	

Code Repository:

1. **Lab Class 3+4:** https://github.com/nahin100/19-CSE4102/tree/main/Lab3and4
   Details:
   a. **ASM (Link): This was taught in Class 3 and is also included in LPE 2 Syllabus.**
   b. **IR_and_CG_As_Compiler (Link): This was taught in Class 4 and is also included in LPE 2 Syllabus.**
   c. **IR_and_CG_As_Interpreter (Link): This was not taught in Class and is not included in LPE 2 Syllabus. It will be covered in the last lab class before Lab Final.**
2. **Lab Class:** https://github.com/nahin100/17-CSE4102/tree/main/Lab5

Problem Sets:

1. **x86 Assembly Program:** Develop an equivalent x86 Assembly Program of following C Program:

   01.

```
int main()
{
   int V, I, R;
   printf("Current = ");
   scanf("%d", &I);
   printf("Resistance = ");
   scanf("%d", &R);

   V = I*R;
   printf("Voltage = %d", V);
   Return 0;
}
```

02.

```
int main()
{
   int a = 10; int count = 0;

   for(a=0; a<10; a++)
   {
     if(a==5) { count=count+1; }
     else if(a >= 7) { count = a++; }
     else { count = a--; }
   }
   Return 0;
}
```

2. **Intermediate Code Generation and Code Generation:** Consider following code snippets:

01.

```
INT num1 = input() + 100
INT num2 = input() - num1
output(num2)
```

02.

```
dim a as integer
dim b as integer
dim c as integer

a = input()
b = input()
output(a*b)
```

a. Generate Intermediate Code Generation from the given code snippet.
b. Generate Code Generation from the given code snippet.

Instructions for Question 1 and 2:

a. **For Question 1:** For example, for question 01, output will be

```
Current = 10
Resistance = 2
Voltage = 20
```

b. **For Question 2:** For a given code snippet, parser will generate intermediate code and assembly code. **Even hand written intermediate code and assembly code will carry marks.**

Regarding Extra Time:
There will be no extra time. Last time of Q1 (c) will be the last time for Lab Report Submission.

Grading Rubrics:

Question Type	A (100%)	B (80%)	C (60%)	D (40%)
Easy	The solution is completely correct.	A major part of the solution is correct.	A minor part of the solution is correct.	A very minor part of the solution is correct. Problem was understood and attempted.
Hard	The solution is completely correct.	A major part of the solution is correct.	-	-

Questions:
Every student will be given different question sets based on Roll number. Link to Google form will be given 1 minute before the lab test. Students will have to submit their answers to Google Classroom.

Upload Instructions:

1. **Separate Folders:** Create separate folders (also for Q2a and Q2b) for each question when uploading.

2. **Roll Number+Questions:** Add your Roll Number and paste given Questions to program files.

3. **Snapshots:** Take separate snapshots of the terminal which shows outputs [Run the program using command without adding ouput.txt: `a < input.txt`]. Do not fabricate the snapshots. If found, the student will get punished severely.

4. **Please rename your file/files with this format:** [Lab Performance Test No]_[Roll Number]_[Question No] (Example: `LPT1_1703060_Q2a`). Upload files to google classroom classwork.

   - **Question 1:** Submit ASM file and Terminal Screen shot.
   - **Question 2a:** Submit Flex file, Bison file, CodeGen file, Terminal Screen shot.
   - **Question 2b:** Submit Flex file, Bison file, CodeGen file, Terminal Screen shot.

5. **Warning:**

   a. **Do not submit the `.exe` file. Google Drive may block the file and the zipped folder cannot be downloaded/examined by the examiner.**

   b. **Do not zip files using winrar or 7zip. Zip files using only the default windows zip file (.zip) feature (Instructions: Right Click on Folder -> Send to -> Compressed (zipped) folder).**

Tips:

1. Rather than writing everything from scratch, just write your codes within existing source code by editing them.

2. Ensure **Laptop Battery Backup + Internet**

3. **Use `mingw32-make` instead of `make` if you face any problem.**

Upload Lab Report Instructions:

1. **Use this Lab Report Template:** Link

2. **Please rename your lab report with this format:** [Lab Performance Test No]_[Roll Number]_Lab_Report (Example: LPT1_1703060_Lab_Report). Upload Lab Report to google classroom classwork.

3. **Lab Report Preparation:**
   - **Question:** Paste your question.
   - **Solution:** Paste contents of your source code. Bold out your own code.

- **Output:** Paste your output snapshot.

4. **Do not cheat in the lab report. Cheating will cause severe punishments.**

Academic Honesty Policy:

1. Do not cheat and be honest.
2. Do not share your answers.
3. *If it is found that someone cheated by copying someone's program file/snapshot, then the original author of the files (If identified) will get severe punishments.*
4. *Someone found guilty of cheating will have his/her test score reset and will have to retake all the lab tests on only the hardest question sets.*
5. *If someone is aware of someone's/organized group's cheating, he/she is welcomed to send (anonymous) mail to the teacher. Teacher will keep the sender's identity secret and reward that sender heavily with extra marks.*

Lab Performance Evaluation 1 Syllabus

Special Instructions:

1. **Students are allowed to open the following softwares: VSCode, NotePad, File Explorer and Snipping tool (to take screenshot) to write their code.**
2. **Students are allowed to open the following websites in their web browsers: google classroom and google form.**
3. **Students are expected to use their word processor programs like MS Word or LibreOffice to edit their lab report.**

Course Outcome

CO1: Understanding the practical approach of how a compiler works.

CO2: Understanding how LEX and YACC is used for lexical and syntax analysis.

CO3: Designing front end of compiler (Lexical Analysis, Syntax Analysis and Semantic Analysis) using LEX and YACC.

Marks Distribution:

Lab Class	Questions	Question Types	Time	Easy	Marks	CO
Lab Class 1	Q1	Stages of C compiler	8 min	Easy	100	CO1
Lab Class 2	Q1 (a)	Lexical Analysis	17 min	Easy	50	CO2
					30	CO3
	Q1 (b)	Syntax Hello, could we schedule a Zoom meeting today?	17 min	Easy	50	CO2
					40	CO3
	Q1 (c)	Semantic Analysis	20 min	Hard	30%	CO3

Reading Assignment (Optional):
Book 1: Compilers 2nd Ed - Principles, Techniques, & Tools - Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman - Pearson (2007)**.**

Topic Name	Book	Chapter	Topics
Lexical Analysis	Book 1	Chapter 3	3.5 (FLEX)
Syntax Analysis	Book 1	Chapter 4	4.9 (BISON)

Code Repository

1. **Lab Class 1 (17 Series):** https://github.com/nahin100/17-CSE4102/tree/main/Lab%201
2. **Lab Class 2 (17 Series):** https://github.com/nahin100/17-CSE4102/tree/main/Lab2
3. **Lab Class 3 (17 Series):** https://github.com/nahin100/17-CSE4102/tree/main/Lab3
4. **Lab Class 4 (17 Series) (Important):** https://github.com/nahin100/17-CSE4102/tree/main/Lab4
5. **Lab Class 1 (18 Series)**: https://github.com/nahin100/18-CSE4102

Video Link of CSE 4102 Extra Class 1 (Semantic Analysis):
https://www.youtube.com/watch?v=GcF2fwIgVHs

Problem Sets:

1. **Stages of C compiler:** Consider following code snippet:

```
#include<math.h>
#define INTEGER int

int main()
{
   INTEGER a=10;
   INTEGER b=20;
   return 0;
}
```

Show output files of all stages along with dumped object file generated by C compiler along with Makefile.

2. **LEX (FLEX) and YACC (BISON):** Consider following code snippets:

01.

```
float num = input("Enter a number: ")
if num > 0:
   print("Positive number")
elif num == 0:
   print("Zero")
else:
   print("Negative number")
```

02.

```
dim i as integer
For i = 1 To 9.9
    For j = 10 To 20
    Next j
Next i
```

03.

```
function isEven(n : int)
begin
   return n % 2.0 == 0;
end
```

a. Show Lexical Analysis on the given code snippet.
b. Show Syntax Analysis on the given code snippet.
c. Show Semantic Analysis on the given code snippet.

Instructions for Question 2:

a. **For question a:** For given input, the lexical analyzer will reply `‘input -> Token Name’` for the correct inputs.

For example, for question 03, output will be

```
function -> FUNCTION
isEven -> ID
( -> LP
n -> ID
: -> COLON
int -> INT_TYPE
) -> RP
begin -> BEG
return -> RET
n -> ID
% -> MOD
2.0 -> FLOAT_NUM
== -> EQUAL
0 -> INT_NUM
; -> SEMI
end -> END
```

b. **For question b:** For given input, the parser will reply `‘Parsing Finished’` for the correct code snippet.

```
Parsing Finished
```

c. **For question c:** Student will need to perform following semantic checkings:

□ Checking whether a variable is declared before use.

```
a = 10; //variable is not declared but used
```

□ Checking whether a variable is declared more than once.

```
int a;
int a = 10; //same variable is declared more than once
```

□ Perform type checking of variable

```
int a = 10.0;
//float number is used instead of integer
```

□ Perform type checking of expression

```
float b = 10.0;
char c = ‘c’;
int a = b+c;
//type of b and c do not match type of a
```

For example, for question 03, output will be

```
In line 3, n with type int does not match with type
float.
```

Regarding Extra Time:

There will be no extra time. Last time of Q1 (c) will be the last time for Lab Report Submission.

Grading Rubrics:

A (100%)	B (80%)	C (60%)	D (40%)
The solution is completely correct.	A major part of the solution is correct.	A minor part of the solution is correct.	A very minor part of the solution is correct. Problem was understood and attempted.

Questions:

Every student will be given different question sets based on Roll number. Link to Google form will be given 1 minute before the lab test. Students will have to submit their answers to Google Classroom.

Upload Instructions:

1. **Separate Folders:** Create separate folders (also for Q2a and Q2b) for each question when uploading.
2. **Roll Number+Questions:** Add your Roll Number and paste given Questions to program files.
3. **Snapshots:** Take separate snapshots of the terminal which shows outputs [Run the program using command without adding ouput.txt: `a < input.txt`]. Do not fabricate the snapshots. If found, the student will get punished severely.
4. **Please rename your file/files with this format:** [Lab Performance Test No]_[Roll Number]_[Question No] (Example: `LPE1_1703060_Q2a`). Upload files to google classroom classwork.

- **Question 1:** Submit both output files and Makefile.
- **Question 2:**
  a. **Tokenize:** Submit Flex file, Makefile, input and output text files.

b. **Parsing:** Submit Flex file (Different from the Flex file submitted for Tokenization), Bison file, Makefile, input and output text files.

5. **Warning:**

   a. **Do not submit the `.exe` file. Google Drive may block the file and the zipped folder cannot be downloaded/examined by the examiner.**

   b. **Do not zip files using winrar or 7zip. Zip files using only the default windows zip file (.zip) feature (Instructions: Right Click on Folder -> Send to -> Compressed (zipped) folder).**

Tips:

1. Rather than writing everything from scratch, just write your codes within existing source code by editing them.

2. Ensure **Laptop Battery Backup + Internet**

3. **Use `mingw32-make` instead of `make` if you face any problem.**

Upload Lab Report Instructions:

1. **Use this Lab Report Template: Link**

2. **Please rename your lab report with this format:** [Lab Performance Test No]_[Roll Number]_Lab_Report (Example: LPT1_1703060_Lab_Report). Upload Lab Report to google classroom classwork.

3. **Lab Report Preparation:**
   - **Question:** Paste your question.
   - **Solution:** Paste contents of your source code. Bold out your own code.
   - **Output:** Paste your output snapshot.

4. **Do not cheat in the lab report. Cheating will cause severe punishments.**

Academic Honesty Policy:

1. Do not cheat and be honest.

2. Do not share your answers.

3. *If it is found that someone cheated by copying someone's program file/snapshot, then the original author of the files (If identified) will get severe punishments.*

4. *Someone found guilty of cheating will have his/her test score reset and will have to retake all the lab tests on only the hardest question sets.*

5. *If someone is aware of someone's/organized group's cheating, he/she is welcomed to send (anonymous) mail to the teacher. Teacher will keep the sender's identity secret and reward that sender heavily with extra marks.*