VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of the UGC Act, 1956)

Reg. No. : 22 BCE11 38

Final Assessment Test(FAT) - Nov/Dec 2024

Programme	B.Tech.	Semester	Fall Semester 2024-25
Course Code	BCSE307L	Faculty Name	Prof. Nagaraj S V
Course Title	Compiler Design	Slot	G2+TG2
		Class Nbr	CH2024250101295
Time	3 hours	Max. Marks	100

General Instructions

- Write only Register Number in the Question Paper where space is provided (right-side at the top) & do not write any other details.

Course Outcomes

CO1. Apply the skills on devising, selecting, and using tools and techniques towards compiler design

CO2. Develop language specifications using context free grammars (CFG).

CO3. Apply the ideas, the techniques, and the knowledge acquired for the purpose of developing software systems.

CO4. Constructing symbol tables and generating intermediate code.

CO5. Obtain insights on compiler optimization and code generation.

Section - I
Answer all Questions (7 × 10 Marks)

Q.No	Question	*M	CO	BL
		*M - Marks		

01.	a. Provide a detailed explanation of the compilation process using the given example to demonstrate the output of each phase of compilation for the input: (6 marks)
	while (i<=num) {
	sum= sum + i * 20;
	i++;
	}
	b. Construct Non-Deterministic Finite Automata for the Regular expression R = (0|1)* (00 |11) (0|1)* (4 marks)
		10	1	2

02.	Consider the following regular expression (RE) 'R'. Convert the regular expression into Deterministic Finite Automata (DFA) using direct method. (ab|c|a)(abc|c)*(bc)* a. Construct the syntax tree with first and last positions from the augmented regular expression (2 marks) b. Calculate the follow position (4 marks) c. Construct the minimized DFA (4 marks)	10	2	3
03.	Perform predictive parsing for the following grammar. expr → expr **or** term | term term → term **and** factor | factor factor → **not** factor | **(expr) | true | false** a) Compute the FIRST and FOLLOW functions (4 marks) b) Generate the parsing table (3 marks)	10	2	3

c) Show the actions of the parser for the input string: **not (true or false)** (3 marks)
Note: [All terminals are represented in bold face]

04. Consider the given grammar for arithmetic expression and answer the following — 10 3 3

S → S / T
S → T
T → R - T
T → R
R → num

a. Write down the semantic rules for evaluating the arithmetic expression (4 marks)
b. Draw the annotated parse tree for "132 / 4/ 8 -2 / 8 - 2 - 1" and print the results (3 marks)
c. Draw the dependency graph (3 marks)

05. Generate three address code for the following control statement and represent it in quadruples (4 marks), triples (3 marks) and indirect triples (3 marks) — 10 4 3

```
If((a==1 && a==0) || (a<b && a>c))
  y= 1
else
 y=0
```

06. Write the grammar and Syntax Directed Translation scheme (SDT) to generate three-address code for Boolean expression. — 10 4 4

((x >= y) or (a != b)) and ((c < d) or (e == f))

a. Generate the Syntax Directed Translation (SDT) production rules (3 marks)
b. Construct the parse tree (5 marks)
c. Generate the three-address code for the expression using backpatching (2 marks)

07. Evaluate the effectiveness of Automatic Parallelization and Cache Locality Optimization in handling complex, real-world applications with irregular data structures (e.g., graphs, sparse matrices) (6 marks). Analyze the challenges compilers face when automatically parallelizing such applications (4 marks) — 10 1 4

Section - II
Answer all Questions (2 × 15 Marks)

*M - Marks

Q.No	Question	*M	CO	BL
08.	Show the following grammar is SLR (1). A → Train TRAIN → DEPARTURE ARRIVAL DEPARTURE → "The train departs from" CITY "at" TIME ARRIVAL → "and arrives in" CITY "at" TIME CITY→ "Bangalore" | "Mumbai" TIME → "6:00 AM" | "2:00 PM" a. Generate the LR (0) canonical collection for the given grammar (5 marks) b. Construct the SLR (1) parsing table (5 marks) c. Parse the string below: (5 marks) The flight departs from Bangalore at 6:00 AM and arrives in Mumbai at 2:00 PM [Note: words given within double quotes to be taken as terminals e.g. "and arrives in" is a single terminal, words mentioned in uppercase are non-Terminals e.g. TIME is a non-terminal]	15	2	4
09.	a. Construct basic blocks (2.5 marks), control flow graph (2.5 marks), Dominator tree (1.5 marks) and natural loops (1.5 marks) i. a = 10 ii. b = 20 iii. t1 = a + b iv. if t1 > 15 goto L1 v. d = a / b vi. e = d + t1 vii. goto L2	15	5	4

viii. L1: d = a * b
ix. e = d - t1
x. L2: return 0

b. Construct Directed Acyclic Graph (DAG) and optimal target code for the expression (7 marks)

x=((a+b)/(b-c)) – (a+b) * (b-c)

BL-Bloom's Taxonomy Levels - (1.Remembering, 2.Understanding, 3.Applying, 4.Analysing, 5.Evaluating, 6.Creating)

⇔⇔⇔

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of the UGC Act, 1956)

Reg. No. : 21BCE1808

Final Assessment Test(FAT) - Nov/Dec 2024

Programme	**B.Tech.**	Semester	**Fall Semester 2024-25**
Course Code	**BCSE307L**	Faculty Name	**Prof. Mercy Rajaselvi Beaulah P**
Course Title	**Compiler Design**	Slot	**D1+TD1**
		Class Nbr	**CH2024250102280**
Time	**3 hours**	Max. Marks	**100**

General Instructions

- Write only Register Number in the Question Paper where space is provided (right-side at the top) & do not write any other details.

Course Outcomes

1. Apply the skills on devising, selecting, and using tools and techniques towards compiler design
2. Develop language specifications using context free grammars (CFG).
3. Apply the ideas, the techniques, and the knowledge acquired for the purpose of developing software systems.
4. Constructing symbol tables and generating intermediate code.
5. Obtain insights on compiler optimization and code generation

Section - I

Answer all Questions (7 × 10 Marks) *M - Marks

Q.No	Question	*M	CO	BL
01.	Construct a deterministic finite automaton for the following regular expression using the direct method. Draw the syntax tree and compute nullable, firstpos, lastpos (**4 marks**), and followpos (**2 marks**). Illustrate the steps of generating DFA states (**3 marks**) and represent the DFA using a transition table (**1 mark**). Regular expression: **a* (a|b)$^+$ (cd)* (b|c|d)$^+$**	10	1	4
02.	A context-free grammar G for declaring variables and array variables is given below. As per the grammar, more than one variable can be declared in a single line with a semicolon as a delimiter. The non-terminals are **D, T, B, C** and the terminals are ***id***, ***int***, ***float***, **;** , **[** , **]** , ***num.*** Grammar G: D → T ***id*** ; D | T ***id*** ; T → B | B C B → ***int*** | ***float*** C → **[*num*]** | **[*num*]** C i) Generate the canonical collection of sets of LR(0) item. **(4 marks)** ii) Construct Simple LR parsing table for the given grammar. **(3 marks)** iii) Illustrate the parsing action of the declaration "***float x*; *int* [5][5] *y*;" using the parsing table and the stack. **(3 marks)**	10	2	5

03.	The grammar G given below generates declarations for a single numerical identifier; these declarations involve three different, independent properties of numbers namely scale, precision and base. **Grammar G:** stmt → ***declare*** variable optionList variable → ***x*** optionList → optionList option | ε option → scale | precision | base scale → ***fixed*** | ***floating*** precision → ***single*** | ***double*** base → ***binary*** | ***decimal*** In the grammar, the terminals are ***declare, x, fixed, floating, single, double, binary, decimal,*** and non-terminals are **stmt, variable, optionList, option, scale, precision, base.** Compute the first and follow of each non-terminals **(4 marks).** Construct a predictive parsing table **(3 marks)** and check whether the declaration **"*declare x single floating*"** is accepted or not. **(3 marks)**	10	2	4
04.	i) Consider the grammar G = **({S, A, Sign}, S, { , , - , + , *num*}, P)** with **P,** a set of productions. The **S** is the start symbol and has a synthesized attribute **val.** The **A** is a non-terminal representing a sequence of numbers and has an inherited attribute **sign_val** and a synthesized attribute **val.** The **Sign** is a non-terminal representing either a positive or negative sign. Grammar G: S → Sign A Sign → + Sign → - A → A_1 , ***num*** A → ***num*** Write the syntax-directed translation scheme (SDT) for the above grammar using the attributes and temporary variables to compute the maximum or minimum among a set of numbers based on the positive or negative sign. If the sign is positive, then **S.val** should contain the maximum of the given set of numbers. If the sign is negative then **S.val** should contain the minimum of the given set of numbers. **(6 marks)** e.g If set of numbers are " - 8, 9, 3", then S.val = 3, If set of numbers are "+ 8, 9, 3", then S.val = 9 ii) Parse the given set of numbers **"+ 5, 2, 9, 6"** and compute the **S.val** using the SDT. Draw the annotated parse tree. **(4 marks)**	10	3	4
05.	The symbols ¬, ∧, ∨, ⇒, ⇔ (negation, conjunction, disjunction, implies and if and only if) are operators of propositional logic. Convert the following propositional logic expressions into three address codes using the given grammar **(4 marks)** and represent them in quadruples, triples, and indirect triples. **(6 marks)** **Grammar G:** S → S ⇔ W | W W → W ⇒ X | X X → X ∨ Y | Y Y → Y ∧ Z | Z Z → ¬ Z | ***a*** | ***b*** | ***c*** | ***d*** **Propositional logic Expressions:** ***a*** ⇒ ***b*** ∨ ¬ ***d*** ⇔ ***c*** ∧ ***d*** ⇔ ¬ ***a***	10	4	3

06.	Generate the intermediate code using Back-patching for the following programming construct **(5 marks)**. Illustrate the Back-patching through an annotated parse tree **(5 Marks)**.
	Programming construct:
	if(year>=2020 && year <=2023 &&month=="May" || month=="June" && item=="book")
	{
	i=0;
	max = 0;
	while (i<200)
	{
	if (max<sale[i])
	max=sale[i];
	i = i+1;
	}
	max_sale=max;
	}
		10	4	5

07.	Discuss the design aspects of parallel machines which support parallelizing compilers with respect to instruction streams **(5 marks)** and architecture **(5 marks)**	10	5	3

Section - II

Answer all Questions (2 × 15 Marks) *M - Marks

Q.No	Question	*M	CO	BL
08.	i) Generate basic blocks and flow graph for the following three address codes.**(4 + 2 marks)** 10 i=0 20 a=n-3 30 if i <a goto 50 40 goto 170 50 b=i-4 60 x=a+2 70 p=x 80 c=p+b 90 d=m[c] 100 e=d-2 110 f=i-4 120 g=p+f 130 m[g]=e 140 i=i+1 150 a=n-3 160 if i<a goto 50 170 ---- ii) Apply suitable optimization techniques and generate the optimized basic blocks. **(5 marks)** iii) Optimize the following code using direct acyclic graph. **(4 marks)** x = a + b y = c * d v = b + a w = d * c z = w + x t = v + y s = z / t	15	5	4

09. i) Compute next use and liveness information for the following block of code. **(4 marks)** | 15 | 5 | 4

code:

```
k= a+ b
m= k- c
n = k * b
p = m +n
q = a / k
```

ii) A microprocessor X has three registers R_0, R_1, and R_2. Generate the machine code utilizing the available registers for the code provided in section(i) based on the next use and live information. Show the status of the address descriptor and register descriptor. **(6 marks)**

iii) Construct the Activation tree and show the status of the control stack for the following code. **(5 marks)**

```
int BinarySearch(int arr[],int num,int first,int last){
int mid;
if(first <= last){
      mid = first+( last - first)/2;
      if(arr[mid]==num)
return mid;
    else if(arr[mid]>num)
return BinarySearch(arr, num, first, mid-1);
    else return BinarySearch(arr, num, mid+1, last);
      }
return -1;
}
void main(){
 int beg, mid, end , num, index;
 int arr[]={12, 23, 34, 45, 56, 67, 78, 89, 91, 95};
 beg=0; end=9;
 num= 5;
 index= BinarySearch(arr,num,beg,end);
 printf("\n %d", index);
}
```

BL-Bloom's Taxonomy Levels - (1.Remembering, 2.Understanding, 3.Applying, 4.Analysing, 5.Evaluating, 6.Creating)

VIT Vellore Institute of Technology

Reg. No. : 21BCE1771

Final Assessment Test (FAT) - July/August 2023

Programme	B.Tech.	Semester	Fall Inter Semester 22-23
Course Title	COMPILER DESIGN	Course Code	BCSE307L
Faculty Name	Prof. Mercy Rajaselvi Beaulah P	Slot	F2+TF2
		Class Nbr	CH2022232500872
Time	3 Hours	Max. Marks	100

Section 1 (10 X 10 Marks)
Answer All questions

01. Convert the regular expression (a + b)* a b (a + b)* into Deterministic Finite Automata(DFA) using direct method. **[10]**

(i) Build the syntax tree and the relevant first and last position for the given regular expression **(5 Marks)**

(ii) Calculate the follow position and draw the final DFA diagram for the given regular expression. **(5 Marks)**

02. Consider the grammar G with the following productions, **[10]**

S → A
A → BC | DBC
B → Bb | ε
C → c | ε
D → a | d

Construct the non-recursive LL(1) predictive parsing table for the given grammar G. **(8 marks)**

Show the stack status, input and shift/reduce action used for parsing the string "abbc" **(2 marks)**

03. Consider the grammar G with the following productions, **[10]**

N → L & L
L → L B
L → B
B → 0
B → 1

Construct the Canonical Left to Right (CLR) parser with lookahead of 1 parsing table for the given grammar G. **(8 marks)**

Show the stack status, input and shift/reduce action used for parsing the string "101&10" **(2 marks)**

04. (i) Develop the grammar and syntax directed definition to convert any prefix expression into postfix expression. **(5 Marks)** **[10]**

(ii) Illustrate the parse tree for the prefix expression "+ a / * b c - d e" and show the corresponding semantic translations for postfix conversion. **(5 Marks)**

05. Construct the parse tree and the corresponding three address code for the following program. **[10]**

while (A < C and B > D) do
{
if A = 1 then C = C + 1
else

while A <= D
do A = A + B

06. Consider the following expression [10]

x = a / b + c - d * e – f

Discover the following representations.

(i) Abstract syntax tree **(2 Marks)**

(ii) Three address code **(2 Marks)**

(iii) Quadruple representation **(2 Marks)**

(iv) Triple representation **(2 Marks)**

(v) Indirect triple representation **(2 Marks)**

07. Categorize the basic blocks from the following three address code. **(5 Marks)** [10]

Also discover the control flow graph(CFG) and identify the loops present in the CFG. **(5 Marks)**

1. if (A < C) goto (3)
2. goto (15)
3. if (B > D) goto (5)
4. goto (15)
5. if (A = 1) goto (7)
6. goto (10)
7. T1 = c + 1
8. c = T1
9. goto (1)
10. if (A <= D) goto (12)
11. goto (1)
12. T2 = A + B
13. A = T2
14. goto (10)
15. T3 = A - B
16. T4= C - D

08. (i) Illustrate the Directed Acyclic Graph (DAG) for the following three address code. **(5 Marks)** [10]

a = b + c
t1 = a x a
b = t1 + a
c = t1 x b
t2 = c + b
a = t2 + t2

(ii) Consider the following expression

s = a + a * (b - c) + (b-c) * d

For the given expression find the following representations,

a. Expression grammar **(1 Marks)**

b. Parse tree **(2 Marks)**

c. Optimized code **(2 Marks)**

09. (i) When instructions are independent, is it possible to change the evaluation order during code generation?. Justify your answer with an example. **(5 Marks)** [10]

(ii) Specify two or more roles of runtime organization in code generation phase. **(5 Marks)**

10. Analyse the difference between scalar processor and vector processor with respect to compiler design aspects. Illustrate with relevant examples. **[10]**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of the UGC Act, 1956)

Reg. No. : 21BA1735

Final Assessment Test (FAT) - May 2024

Programme	**B.Tech.**	Semester	**WINTER SEMESTER 2023 - 24**
Course Title	**COMPILER DESIGN**	Course Code	**BCSE307L**
Faculty Name	**Prof. S Venkatraman**	Slot	**B2+TB2**
		Class Nbr	**CH2023240501784**
Time	**3 Hours**	Max. Marks	**100**

General Instructions:

- Write only Register Number in the Question Paper where space is provided (right-side at the top) & do not write any other details.

Section - I

Answer all questions (5 X 10 Marks = 50 Marks)

01. a) Show that the grammar **[10]**

stmt → **if** expr **then** stmt | **if** expr **then** stmt **else** stmt | **other**

is ambiguous. Note: stmt is non-terminal and **other** is terminal.

Do this by demonstrating that the statement

if E1 **then if** E2 **then** S1 **else** S2

has two dissimilar parse trees [5marks]

b) Rewrite the given grammar so that there is no ambiguity. [5 marks]

02. Construct the SLR[1] parsing table for the grammar given below showing all required steps. **[10]**

S → b A i B

A → ε

B → r C

C → d

Note: S, A,B, and C are non-terminals. b, i, r and d are terminals.

03. Consider the following grammar **[10]**

expr → expr + term | expr - term | term

term → term * factor | term / factor | factor

factor → digit | (expr)

digit → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

a) For this grammar draw a table whose left-hand-side consists of productions and whose right-hand-side consists of associated semantic rules [5 marks]

b) build the annotated parse tree for (1+2)/3-4*5 by showing all steps [5 marks]

04. Consider the following program with a global variable and a function call. **[10]**

```
int num;
int Pinky(int a, int b)
{
 int c;
```

```
c = a + b;
return c;
}
void main()
{
num = Pinky(4, 10);
Print(num);
}
```

a) For the given program write three address code instructions [5 marks]
b) write the corresponding assembly language code assuming a suitable hypothetical assembly language. Include comments so that the assembly language code instructions are explanatory.[5 marks]

05. a) Consider a program for sorting numbers in a high-level language of your choice such as C. Using this program, discuss and illustrate mechanisms by which we may achieve parallelism on modern micro-processors that support multi-core architectures. [5 marks]. **[10]**
b) Provide an example of a program where parallelism is hard to achieve, with justification[5 marks].

Section - II
Answer all questions (2 X 15 Marks = 30 Marks)

06. Convert the regular expression (a|b)*abbc(a|b)* into a minimized deterministic finite automata using the direct method while indicating all steps with appropriate explanation during the conversion process. **[15]**

07. By means of suitable scenarios, describe the following code optimization techniques with appropriate code segments: (3 marks each) **[15]**
i) Constant folding
ii) Constant propagation
iii) Algebraic simplification
iv) Operator strength reduction
v) Common sub-expression elimination

Section - III
Answer all questions (1 X 20 Marks = 20 Marks)

08. a) Convert the following expression into a three address code and then construct quadruples, triples, and indirect triples. (10 marks) **[20]**

```
int e=5;
int low=0;
int high=n-1;
while(low<=high) { int mid=(low+high)/2; if(a_mid==e) {p=q+r;} else { if(a_mid>e) { low =
mid+1; } else { high=mid-1; } } }
```

b) Convert the following expression into a three address code and then construct quadruples, triples, and indirect triples. (10 marks)
a := b * - c + b * - c