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

Reg. No. : 21BLC1122

Final Assessment Test (FAT) - May 2024

Programme	**B.Tech.**	Semester	**WINTER SEMESTER 2023 - 24**
Course Title	**COMPILER DESIGN**	Course Code	**BCSE307L**
Faculty Name	**Prof. Mercy Rajaselvi Beaulah P**	Slot	**A2+TA2**
		Class Nbr	**CH2023240502679**
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 (7 X 10 Marks = 70 Marks)

01. Consider the following grammar. The character R is the Non-terminal and the symbols **#** , **%** , **(** , **)** , ***a*** , ***b*** are Terminals. **[10]**

Grammar:

R → R # | # R | RR | R% | (R) | ***a*** | ***b***

(i) Check whether given grammar is LL(1) grammar or not. **(8 Marks)**

(ii) Justify your answer. **(2 Marks)**

02. A Context free grammar defining a programming block is given below. The uppercase strings represents the Non-terminals and characters in lowercase italics ***x*** , ***y*** and symbols { , } are terminals. **[10]**

Grammar:

BLOCK → { BODY }

BODY → { BODY STMT } | STMT

STMT → ***x*** | ***y***

(i) Generate the canonical LR(1) items and construct corresponding parsing table for the given grammar. **(7 marks)**

(ii) Illustrate the parsing action of the given programming construct using the parsing table and the stack. **(3 marks)**

Programming Construct : { { { ***x y*** } ***x*** } }

03. Consider the following Syntax directed translation scheme for the declaration of the variables of integer and double data type. The attributes of the grammar symbols ***'type'*** and ***'width'*** are used to hold the **datatype** and **memory size(bytes)** of the declared variables. The ***'m'*** and ***'n'*** represents temporary variables. The function ***Addsymtab*** is used to enter the datatype and **[10]**

storage size of the variable into the symbol table.The uppercase alphabets **D, T, P, A, A_1, V, V_1** represents the Non-terminals and the strings ***int, double, num, id*** and the symbols **; [ ] , Є** are terminals.

Grammar with SDT:

$D \rightarrow T$ { V.*type* = T.*type*; V.*width*= T.*width* } V ; D | Є

$T \rightarrow P$ { *m* = P.*type*; *n*= P.*width* }

A { T.*type* = A.*type*; T.*width*= A.*width* }

$P \rightarrow$ ***int*** { P.*type* = integer ; P.*width* = 4 }

$P \rightarrow$ ***double*** { P.type = double ; P.*width* = 8 }

$A \rightarrow$ **[** ***num*** **]** A_1 {A.*type*=array(num.value, A_1.*type*); A.*width* = num.value * A1.*width*}

$A \rightarrow$ Є { A.*type* = *m* ; A.*width* = *n*}

$V \rightarrow$ {V_1.*type* = V.*type*; V_1.*width* = V.*width*} V_1 , ***id*** { *Addsymtab*(id.entry,V.type,V.width)}

$V \rightarrow$ ***id*** { *Addsymtab*(id.entry, V.type, V.width) }

(i) Draw the annotated parse tree for the following declaration statement. **(6 marks)**

int a, b ; double [4] [3] *x* ;

(ii) Draw the corresponding dependency graph. **(2 marks)**

(iii)Show the status of the Symbol Table after parsing the given declaration statement. **(2 marks)**

04. Convert the following string into three address code using the given grammar(**4 Marks**) and represent them in quadruple, triple, and indirect triple(**6 Marks**). The symbols **@, #, $** are binary operators and **%** is unary operator. **[10]**

Grammar :

$W \rightarrow W$ @ X | X

$X \rightarrow X$ # Y | Y

$Y \rightarrow Y$ $ Z | Z

$Z \rightarrow$ % Z | *a* | *b* | *c* | *d*

String:

a # b $ % d $ c @ d # % a

05. Apply Back-patching in the generation of three address code for the following programming construct (**5 Marks**). Illustrate the Back-patching through annotated parse tree (**5 Marks**). **[10]**

```
while (i < 10 )
{
if (a < 6 ) && ( b > i)
  {
  a= a + i; i = i + 1;
  }
}
```

06. Consider the following three address code: **[10]**

```
t = a + b
s = c * d
```

```
u = b + a
v = d * c
x = t + u
w = v * s
z = x - w
p = z
```

(i) Construct the Direct Acyclic Graph and generate the optimized three address code. **(4 Marks)**
(ii) Generate the machine code for the above optimized three address code using the instruction sets of suitable microprocessor. **(4 Marks)**
iii) Construct the Activation tree for the following code. **(2 Marks)**

```
int main()
{
 int m1, m2;
 float avg;
 printf(" Enter the 1st mark\n");
 scanf("%d", &m1);
 printf(" Enter the 2nd mark\n");
 scanf("%d", &m2);
 avg = average(m1,m2);
 printf("average = %f",avg);
 }

 float average(int x, int y)
 { return sum(x,y)/2; }

 int sum(int a, int b)
   { return a+b; }
```

07. (i) Discuss the impact of Instruction scheduling and software pipelining in the design of compilers for parallel processing architecture. **(5 Marks)** **[10]**
(ii) Discuss the design aspects of vector operations supported by vectorizing compiler in achieving parallelism on a vector architecture. **(5 Marks)**

Section - II

Answer all questions (2 X 15 Marks = 30 Marks)

08. Consider the regular expression given below which specifies the definition of the floating-point number with exponent, **[15]**

(digit)$^+$. digit (E | e) (+ | -) **(digit)$^+$**

Where **digit** represents the numbers from **0-9** and '**.**' represents the **decimal point.**
(i) Compute the first position, last position and follow position for the given regular expression. **(6 Marks)**
(ii) Construct the deterministic finite automaton using the Direct method. **(6 Marks)**
(iii) Simulate the string "25.4E+18" using the constructed DFA. **(3 Marks)**

09. Consider the following code segment:- [15]

Code:

```
p = 3.14;
for( i = 0; i < 10; i++)
{
c = 2 * p * a[i];
a = p * a[i]^2;
}
```

(i) Convert the code segment into three address code. **(4 Marks)**

(ii) Identify the basic blocks and flow graph in the three address code. **(6 Marks)**

(iii) Describe suitable optimization techniques and apply them on the Basic blocks and generate the optimized three address code. **(5 Marks)**