

Total No. of Questions : 10]

SEAT No. :

P3985

[Total No. of Pages : 5

[5353]-597

T.E. (IT) (Semester - II)
SYSTEMS PROGRAMMING
(2015 Pattern)

Time : 2.30 Hours]

[Max. Marks : 70

Instructions to the candidates:

- 1) Solve Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8, Q.9 or Q.10.
- 2) Neat diagrams must be drawn wherever necessary.
- 3) Figures to the right indicate full marks.
- 4) Assume suitable data if necessary.

- Q1)** a) Explain with example the need of pool table in assembler. [4]
b) With a neat diagram explain how any input string is processed by LEX n YACC. [6]

OR

- Q2)** a) Explain with example need of TII in single pass assembler. [4]
b) Explain different parameter passing methods used in macroprocessors. [6]

- Q3)** For the following assembly language program show MNT, MDT, stack organization and the expanded code for the given assembly language program [10]

```
MACRO
XYZ    &A
MOVER AREG,&A
MEND
MACRO
MIT    &Z
MACRO
&Z&W
ADD BREG,&W
XYZ ALL
MEND
MOVER &Z,ALL
MEND
START
MIT HELLO
ADD AREG, BREG
HELLO YALE
YALE EQU$
ALL DC 3
END
```

P.T.O.

OR

Q4)

Program deck for PGA: ESD cards

TXT cards:

[10]

Card Ref. No.	Name	Id	Type	Rel. Addr	Length
1	PGA	01	SD	0	38
2	PGA1	-	LD	34	--
3	PGB	02	ER	--	--
3	PGC	03	ER	--	--
3	PGC2	04	ER	--	--

Card Ref. No.	Relative Address	Contents
4	30-33	0, 0+4
5	34-37	34, 0

RLD cards:

Card Ref. No.	ESD ID	length	Flag	Relative address
4	1	4	+	30
4	2	4	+	30
5	4	4	+	34
5	3	4	-	34

Program deck for PGB: ESD cards

TXT cards:

Card Ref. No.	Name	Id	Type	Rel. Addr	Length
7	PGB	01	SD	0	26
8	PGB1	-	LD	14	--
9	PGA	02	ER	--	--
9	PGC1	03	ER	--	--

Card Ref. No.	Relative Address	Contents
10	14-17	-4, 14
11	18-21	4, 14
12	22-25	-16

RLD cards:

Card Ref. No.	ESD ID	length	Flag	Relative address
10	3	4	+	14
10	1	4	+	14
11	1	4	+	18
11	1	4	+	18
11	1	4	-	18
12	3	4	+	22
12	1	4	+	22
12	2	4	-	22

Program deck for PGC: ESD cards

TXT cards:

Card Ref. No.	Name	Id	Type	Rel. Addr	Length
14	PGC	1	SD	0	20
15	PGC1	--	LD	12	--
15	PGC2	--	LD	16	--

CR no.	Rel. Addr.	Contents
16	8-11	16,16
17	12-15	8.16
18	16-19	4

RLD cards:

Card Ref. No.	ESD ID	length	Flag	Relative address
17	1	4	+	12
17	1	4	+	12
18	1	4	+	16
18	1	4	+	16
18	1	4	-	16

For the given card information for program segments PGA, PGB and PGC generate GEST, LESA and final code allocation in main memory using DLL.

Q5) a) $S \rightarrow S + S / S - S / (S) / S * S / a$ [6]

Remove ambiguity and left recursion from the given grammar.

b) Check whether the unambiguous grammar generated from the grammar in Q5a) is LL? [6]

c) Explain YACC file structure. [6]

OR

Q6) a) Consider the following grammar [10]

$S \rightarrow L = R / R$

$L \rightarrow *R / id$

$R \rightarrow L$

Construct LALR parser and parse for the string "id = *id".

b) Write a short note on Recursive Descent Parser. [4]

c) Differentiate between SLR and CLR parsers. [4]

Q7) a) Construct syntax tree and DAG for the given expression: [4]

$X = (b * -c) + y + (b * -c) / z$

b) Define the following: [8]

i) Syntax Directed Definition

ii) Syntax Directed Translation

iii) Synthesized Attributes

iv) Inherited Attributes

- c) Generate three address code for [4]

For(i=0;i<=10;i++)

x=y+z;

OR

- Q8)** a) Explain stack and heap storage allocation strategies. [6]

- b) Translate the following C code fragment into three address code (TAC). Assume integer size of 4 bytes; [8]

```
int sum=0,i,j;
```

```
int A[10][10],B[10][10],C[10][10],X[10];
```

```
i=1;
```

```
j=1;
```

```
while (i<10 && j<=20)
```

```
{
```

```
    Sum += X[i];
```

```
    C[i][i] = A[i][j] + B[i][j];
```

```
        i++;
```

```
        j++;
```

```
}
```

- c) Explain implicit and explicit type conversion. [2]

- Q9)** a) Explain the need of Flow graph and show the same for the example in Q10a. [4]

- b) Compare machine dependent and independent optimization. [8]

- c) Discuss machine dependent issues for code generation. [4]

OR

- Q10) a) Optimize the given quick sort code using peephole optimization techniques. [8]

```
i = m - 1
j = n
t1 = 4 * n
v = a[t1]
i = i + 1
t2 = 4 * I
t3 = a[t2]
if t3 < v goto (5)
j = j - 1
t4 = 4 * j
t5 = a[t4]
if t5 > v goto (9)
if i >= j goto (23)
t6 = 4 * I
x = a[t6]

t7 = 4 * I
t8 = 4 * j
t9 = a[t8]
a[t7] = t9
t10 = 4 * j
a[t10] = x
goto (5)
t11 = 4 * I
x = a[t11]
t12 = 4 * i
t13 = 4 * n
t14 = a[t13]
a[t12] = t14
t15 = 4 * n
a[t15] = x
```

- b) Discuss code generation issues. [4]
c) Explain dynamic code generation algorithm. [4]

