

Total No. of Questions : 10]

SEAT No. :

P1766

[5058]-406

[Total No. of Pages : 4

**T.E.(Information Technology)
SYSTEMS PROGRAMMING
(2012 Course)(Semester-II)**

Time :2½Hours]

[Max. Marks : 70

Instructions to the candidates:

- 1) Answer Q1 or Q2,Q3or Q4, Q5 or Q6 , Q 7 or Q 8, Q 9 or Q10.
- 2) Neat diagrams must be drawn wherever necessary.
- 3) Figures to the right indicate full marks.
- 4) Assume suitable data if necessary.

Q1) a) Show the contents of various tables alongwith stack organization generated by the II pass macro processor for the following code. Also give the code after expansion.

```
MACRO
EVAL & X,&Y,&Z
AIF (&Y EQ&Z). ONLY
LOAD &X
SUB &Y
ADD &Z
AGO .OVER
.OONLY LOAD &Z
.OVERMEND
MACRO
MAJOR &P,&Q,&R,&M,&N,&L
EVAL &P&Q&R
STORE &L
EVAL &M,&N,&N
MEND
START
MAJOR A,B,C,D,E,F
A DS 1
B DS 2
C DS 3
D DS 4
E DS 5
F DS 6
END
```

[8]

P.T.O.

- b) Define the following terms with examples. [2]
- | | |
|----------------|--------------------|
| i) Compiler | ii) Loader |
| iii) Assembler | iv) Macroprocessor |

OR

- Q2) a) Give the various data structures in the design of pass-1 and pass-2 of a Two-pass direct linking loader for the given example.

	Rel. Addr.
PGA START	0
ENTRY PGA1	10
EXTRN PGB	20
DC A(PGA), A(PGB+4)	30
PGA1 DC A(PGA1-PGA)	34
END	
PGB START	0
ENTRY PGB1	5
EXTRN PGA	10
PGB1 DC A(PGB1)	14
PGB2 DC A(PGB+4), A(PGB1-PGB)	18
PGB3 DC A(PGB-PGA-16)	22
END	

- b) Explain the different phases of a compiler. [2]

- Q3) a) Perform Pass I and Pass II for the given assembly language code and assume a hypothetical instruction set with each instruction of length 1

```

START 500
      MOVER AREG, LAB
      ADD BREG, LOOP
L1    DS 20
      LOAD AREG='5'
      ADD BREG='1'
BACK EQU L1
L2    SUB CREG, LAB
      LTORG
      MOVEM AREG, LAB
      ORIGIN L2
      SUB BREG='2'

LAB   DC 2
LOOP  DS 5
      STOP
      END

```

- b) Explain the concept of subroutine linkages in loaders and linkers. [2]

OR

- Q4) a) Write a note on overlay structure of loaders. [4]

- b) Convert the given regular expression of DFA: [6]

$$(a+b)^*(a+\epsilon)^*$$

- Q5) a) Differentiate between top down parser and bottom up parser. [4]

- b) For the given grammar, design predictive parser and show parsing table.
 $S \rightarrow +SS/*SS/a$ and parse the string $+*aaa$. Justify [8]

- c) Using the given table perform operator precedence parser for the expression $id+id*id$ [6]

	+	-	*	/	^	id	()	\$
+	>	>	<	<	<	<	<	>	>
-	>	>	<	<	<	<	<	>	>
*	>	>	>	>	<	<	<	>	>
/	>	>	>	>	<	<	<	>	>
^	>	>	>	>	>	<	<	>	>
id	>	>	>	>	>	>	>	>	>
(<	<	<	<	<	<	<	<	<
)	>	>	>	>	>	>	>	>	>
\$	<	<	<	<	<	<	<	<	<

OR

- Q6) a) Compare SLR and table driven parsing methods. [4]

- b) Define Handle and handle pruning w.r.t bottom up parser. For the grammar given, $S \rightarrow SS+/SS*/a$.

Identify the handles at each step and parse the string $aaa*a++$. [4]

- c) Design SLR parser for the given grammar. Also show the moves by the parser for input string $"a/(a+a)*a"$. [10]

$$S \rightarrow S+S/SS / (S) / S*S/a$$

- Q7) a) Define and explain annotated parse tree for the given grammar [8]

$$E \rightarrow E+T / T$$

$$T \rightarrow T*F / F$$

$$F \rightarrow id$$

Annotate the tree for $2+3*5$

- b) Which are the different types of intermediate code representations. Explain w.r.t the expression. [8]

$$v1=(v2-v3)*(v2+2*v3)$$

OR

- Q8) a) Draw the dependency graph of the expression in Q7a) and list down the synthesized and inherited attributes with definition. [8]

- b) Write the method of generating intermediate code for the expression [8]

If (condition) then $p=q+r$

Else $x=y+z$

- Q9) a) Obtain the TAC for the following code. [8]

for($i=1$; $i \leq 10$; $i++$)

$X[i][2*j-1]=Y[i][2*j-1]$

- b) Discuss code generation issues [4]

- c) Write a short note on activation record. [4]

OR

- Q10) a) Discuss with suitable example machine dependent code optimization. [8]

- b) Explain the following code optimization techniques with examples. [8]

- i) Removal of Loop Invariants
- ii) Elimination of common sub expressions
- iii) Dead Code Elimination
- iv) Copy Propagation

