Total No. of Questions :10]

SEAT No. :

**P2911**

[Total No. of Pages :6

# [4958]-1106
# T.E. (IT)
# SYSTEMS PROGRAMMING
# (2012 Course) (Semester - II) (314450)

*Time : 2½ Hours]* *[Max. Marks :70*

***Instructions to the candidates:***

1) ***Answer 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 side indicate full marks.***
4) ***Assume suitable data, if necessary.***

***Q1)*** a) For the following piece of assembly language code, show the contents of symbol table, literal table and pool table. Assume machines opcodes and size of instruction equal to one. **[6]**

```
      START 100
A     DC 10
      MOVER AREG, B
      MOVEM BREG, = '1'
      ADD AREG, = '2'
      SUB BREG, = '1'
B     LTORG
      PRINT C
      MOVER CREG, LOOP
C     EQU B+10
      ORIGIN 10
      MOVEM AREG, = '1'
      END.
```

b) Explain absolute loader scheme. What are the advantages and disadvantages of this scheme? **[4]**

OR

*P.T.O.*

*Q2)* a) For the following assembly language program show MNT, MDT, ALA and the expanded assembly language program. **[8]**

```
        MACRO
        XYZ      &A
        ST  1, &A
        MEND
        MACRO
        MIT      &Z
        MACRO
        &Z &W
        AR 4, &W
        XYZ      ALL
        MEND
        ST &Z, ALL
        MEND
PROG START
        USING *, 15
        MIT HELLO
        ST 2,3
        HELLO YALE
        YALE EQU 5
ALL DC  F '3'
        END.
```

b) List down the phases of a compiler. **[2]**

*Q3)* a) Give ESD, TXT and RLD cards for both PG1 and PG2 for the following assembly language program. **[6]**

| Rel. Addr. | | Source program | |
|---|---|---|---|
| 0 | PG1 | START | |
| | | ENTRY PG1ENT1, PG1ENT2 | |
| | | EXTRN PG2ENT1, PG2 | |
| 20 | ENT1 | ------ | |
| 30 | PG1ENT2 | ------ | |
| 40 | | DC | A(PG1ENT2) |
| 44 | | DC | A(PG1ENT1 +15) |
| 48 | | DC | A(PG1ENT2 - PG1ENT1 –3) |
| 52 | | DC | A(PG2) |
| 56 | | DC | A(PG2ENT1 + PG2 - PG1ENT1 +4) |
| | | END | |
| 0 | PG2 | START | |
| | | ENTRY PG2ENT1 | |
| | | EXTRN PG1ENT1, PG1ENT2 | |
| 16 | PG2ENT1 | ------ | |
| | | ------ | |
| 24 | | DC | A(PG1ENT2) |
| 28 | | DC | A(PG1ENT1) |
| 32 | | DC | A(PG1ENT2 – PG1ENT1 –3) |
| | | END | |

b) Explain different parameter passing methods used in macroprocessors. **[4]**

OR

*Q4)* a) Convert the given Regular expression to its equivalent DFA. **[6]**

(a.b)* + (a+c)*

b) Give the various data structures used in the lexical analysis phase of compilers. **[4]**

*Q5)* a) With a neat diagram explain the classification of parsers. **[6]**

b) Define table -driven predictive parser. For the following grammar **[8]**

$S \rightarrow aSbs \; / \; bSaS \; / \; \varepsilon$

Construct table-driven predictive parser and parse the string "ab".

c) Compare bottom up and top down parser. **[4]**

OR

*Q6)* a) Consider the following grammar **[10]**

$S \rightarrow CC$

$C \rightarrow cC \; / \; d$

Construct LALR parser and parse for the string "ccd".

b) Explain YACC file structure. **[4]**

c) Compare LALR and CLR parsers. **[4]**

*Q7)* a) Construct parse tree, syntax tree and annotated parse tree for 3*5+4. **[6]**

b) Explain type checking and its types. **[6]**

c) Generate three address code for **[4]**

```
while (a<b) do
{
    x=y+z;
}
```

OR

*Q8)* a) For the following statement, Generate intermediate code in the format: **[8]**

i) Postfix notation

ii) Quadruple

iii) Parse tree

iv) Triple

Temp = limit*(max-min)+3*limit*(max+min).

b) Translate the following C fragment into the three address code **[8]**

```
begin
    int add,i,j;
    int a[10][10],b[10][10];
    add=0;
    i=1;
    j=1;
    do
     begin
      add=add+a[i,j]*b[i,j];
          i++;
          j++;
      end;
    while(i<=20 && j<=20);
end;
```

***Q9)*** a) Discuss code generation issues. **[4]**

b) Discuss with suitable example machine dependent code optimization. **[8]**

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

OR

***Q10)*** a) Explain following machine independent optimization techniques: **[8]**

i) Loop invariation.

ii) Common sub-expression elimination.

iii) Dead code elimination.

iv) Strength reduction.

b) Compare machine dependent and independent optimization. **[4]**

c) Explain different storage allocation strategies. **[4]**