MISNO:

# COLLEGE OF ENGINEERING, PUNE

**(An Autonomous Institute of Govt. of Maharashtra)**

**END SEMESTER EXAMINATION**

**Final.B.Tech**

**Computer Engineering**

CT-14001 – **COMPILER CONSTRUCTION**

Academic Year: 2014- 15 — Timing: 3 hrs

Semester -VII — Max. Marks: 60

Date: 24/11/2014 — Time: 14:00-17:00

Instructions:
1. Figures to the right indicate the full marks(for part-A).
2. Mobile phones and programmable calculators are strictly prohibited.
3. Writing anything on question paper is not allowed.
4. Exchange/Sharing of anything like stationery, calculator is not allowed.
5. Assume suitable data if necessary.
6. Write your MIS Number on Question Paper

## PART-A

1 What is front-end and back-end of the compiler? 2

2 If the following descriptions define a regular language then write the regular expression. Otherwise indicate that the language is not regular. 2

(i) All strings of 0's and 1's representing the binary numbers which are power of 2.

(ii) All strings of 0's and 1's that do not have more than 3 consecutive 1's in it.

3 Consider the following C code fragment: 4

```
int a[10][10];
i=0;
while(i<10)
{
      a[i][i]=i;
       i++;
}
```

(a) Design its three-address representation.

(b) Construct the flow graph of TAC you designed above.

4 What are the contents of activation record? 2

## PART-B ( 5 X 10 M = 50 M)

1. Explain the working of compiler by drawing its block diagram.

2. Generate code (target) for the following C statements, assume 3 registers are available, and mention the code generation algorithm used for generating the code.

a/ ( b + c) – d * (e + f)

3 Given the grammar

S -> aABe

A -> Abc

A -> b

B -> d

form its SLR parsing table

4 Using the translation scheme given below, generate the 3-address code for the following program fragment:

```
while( a<c and b>d) do
        if a=1 then c=c+1
        else
            while a<=d do
                a=a+3
```

Assume additional data if necessary.

| **Production rule** | **Semantic action** |
|---|---|
| E-> E1 and M E2 | backpatch(E1.truelist, M.quad)<br>E.truelist= E2.truelist<br>E.falselist=merge(E1.falselist,E2.falselist) |
| E-> id1 relop id2 | E.truelist=makelist(nextquad)<br>E.falselist=makelist(nextquad+1)<br>gen(if id1 relop id2 goto___)<br>gen(goto___) |
| S-> if E then M1 S1 N else M2 S2 | backpatch(E.truelist, M1.quad)<br>backpatch(E.falselist, M2.quad) |
| S->while M1 E do M2 S1 | backpatch(S1.nextlist, M1.quad)<br>backpatch(E.truelist, M2 .quad)<br>S.nextlist=E.falselist<br>gen(goto__, M1.quad) |
| N->ε | gen(goto___) |
| M->ε | M.quad= nextquad |

5 Explain each of the following with proper example:-

(i) Copy propagation

(ii) Dead-code elimination

(iii) Strength reduction

(iv) Loop in-variant code motion

(v) Common sub-expression elimination