Topics :	Intermediate Code Generator
Objectives:	

The main goal of this lab is to study design issues to be considered for developing a Intermediate Code Generator such as the following:

•In the analysis-synthesis model of a compiler, the front end of a compiler translates a source program into an independent intermediate code, and then the back end of the compiler uses this intermediate code to generate the target code (which can be understood by the machine).

The benefits of using machine independent intermediate code are:

- Because of the machine independent intermediate code, portability will be enhanced.For ex, suppose, if a compiler translates the source language to its target machine language without having the option for generating intermediate code, then for each new machine, a full native compiler is required. Because, obviously, there were some modifications in the compiler itself according to the machine specifications.
- Retargeting is facilitated.

It is easier to apply source code modification to improve the performance of source code by optimizing the intermediate code. Now in this exercise we will use a given expression for generation of intermediate code.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char op[2],arg1[5],arg2[5],result[5];
void main()
{
 FILE *fp1,*fp2;
 fp1=fopen("input.txt","r");
 fp2=fopen("output.txt","w");
while(!feof(fp1))
 {

fscanf(fp1,"%s%s%s%s",op,arg1,arg2,result);
if(strcmp(op,"+")==0)
   {
fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nADD R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);
   }
if(strcmp(op,"*")==0)
   {
fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nMUL R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);
```

```
    }
if(strcmp(op,"-")==0)
    {
fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nSUB R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);
    }
if(strcmp(op,"/")==0)
    {
fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nDIV R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);
    }
if(strcmp(op,"=")==0)
    {
fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nMOV %s,R0",result);
    }
    }
fclose(fp1);
fclose(fp2);
getch();
  }
```

Input: (from file)

```
+ a b t1
* c d t2
- t1 t2 t
= t ?x
```

Output: (In file)

```
MOV R0,a
ADD R0,b
MOV t1,R0
MOV R0,c
MUL R0,d
MOV t2,R0
MOV R0,t1
SUB R0,t2
MOV t,R0
MOV R0,t
MOV x,R0
```