

ASSEMBLER(VARIANT-I)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

char variant1[50][50];
int varcnt=0;

char optab[15][8]={"STOP","ADD","SUB","MUL","MOVER","MOVEM","COMP","BC","DIV",
                  "READ","PRINT","DS","DC","START","END"};

char regtab[4][5]={"AREG","BREG","CREG","DREG"};
char reloptab[6][4]={"LT","LE","EQ","GT","GE","ANY"};

char instruction[30],t1[10],t2[10],t3[10],t4[10];
char fname[30];

int start,flag;
int op,r,index;
int lc,pc,n;

int symcnt;
struct symtab
{
    char symbol[10];
    int address,define,used,length,value;
}symtab[40];

int searchoptab(char t[])
{
    int i;
    for(i=0;i<15;i++)
    {
        if(stricmp(optab[i],t)==0) //strcasecmp for linux
            return i;
    }
    return -1;
}
```

```
int searchregtab(char t[])
{
    int i;
    for(i=0;i<4;i++)
    {
        if(stricmp(regtab[i],t)==0)
            return i+1;
    }
    return -1;
}
int searchreloptab(char t[])
{
    int i;
    for(i=0;i<6;i++)
    {
        if(stricmp(reloptab[i],t)==0)
            return i+1;
    }
    return -1;
}
int searchsymtab(char t[]) //search symbol and if not found add symbol
{
    int i;
    for(i=0;i<symcnt;i++)
    {
        if(stricmp(symtab[i].symbol,t)==0)
            return i;
    }
    i=symcnt;
    strcpy(symtab[i].symbol,t);
    symcnt++;
    return i;
}
int issymbol(char sym[])
{
    if(searchoptab(sym)==-1 && searchregtab(sym)==-1 && searchreloptab(sym)==-1)
        return 1;
    else
    {
        //error: keywords cant be used as a symbol
        return 0;
    }
}
```

```
void updatesymbol(char sym[]) //update used of symbol according to index
{
    index=searchsymtab(sym);
    symtab[index].used=1;
}

void definesymbol(char sym[])
{
    index=searchsymtab(sym); //get the index number
    if(symtab[index].define==0)
    {
        symtab[index].define=1;
        symtab[index].address=lc;
    }
    //else error: redeclaration of symbol
    if(op==11) //ds
    {
        int i,l;
        l=symtab[index].length=atoi(t3);
        symtab[index].value=0;
        sprintf(variant1[varcnt++],"(DL,02) (C,%d) ",l);
        for(i=0;i<l;i++)
            lc++;
        lc--;
    }
    else if(op==12) //dc
    {
        symtab[index].length=1;
        symtab[index].value=atoi(t3);
        sprintf(variant1[varcnt++],"(DL,01) (C,%d) ",symtab[index].value);
    }
}
```

```
void process1(char t1[])
{
    op=searchoptab(t1);
    if(op==0) //stop without label
    {
        sprintf(variant1[varcnt++], "(IS,00)");
    }
    else if(op==14) //end without name
    {
        lc--;
        flag=1;
        sprintf(variant1[varcnt++], "(AD,02)");
        return;
    }
    else if(op==13) //start without number
    {
        lc=-1;
        sprintf(variant1[varcnt++], "(AD,01)");
    }
    //else error: invalid opcode
}
```



```
void process2(char t1[],char t2[])
{
    r=0;
    op=searchoptab(t1);
    if(op==9 || op==10) //read or write
    {
        if(issymbol(t2))
            updatesymbol(t2);
        sprintf(variant1[varcnt++],"(IS,%d) (S,%d)",op,index);
    }
    else if(op==13) //start 200
    {
        start=lc=atoi(t2);
        sprintf(variant1[varcnt++],"(AD,01) (C,%d)",start);
        lc--;
    }
    else if(op==14) // end functionname
    {
        index=searchsymtab(t2);
        if(symtab[index].define==1)
            pc=symtab[index].address;
        //else error: symbol is used but not defined
    }
    else // label with stop
    {
        if(issymbol(t1))
        {
            op=searchoptab(t2);
            if(op==0)
            {
                definesymbol(t1);
                process1(t2);
            }
            //else error: invalid opcode
        }
    }
}
```

```
void process3(char t1[],char t2[],char t3[])
{
    op=searchoptab(t1);
    if((op>0 && op<7) || op==8) // add sub mul mover movem comp or div
    {
        r=searchregtab(t2);
        if(r>0 && r<5)
        {
            if(issymbol(t3))
                updatesymbol(t3);
            sprintf(variant1[varcnt++],"(IS,%d) %d (S,%d)",op,r,index);
        }
        //else error: invalid register
    }
    else if(op==7) // bc
    {
        r=searchreloptab(t2);
        if(r>=1 && r<=6)
        {
            if(issymbol(t3))
                updatesymbol(t3);
            sprintf(variant1[varcnt++],"(IS,%d) %d (S,%d)",op,r,index);
        }
        //else error: invalid register opcode
    }
    else //label with read or write or symbol ds or dc
    {
        if(issymbol(t1))
        {
            op=searchoptab(t2);
            if(op==9 || op==10) //read or write
            {
                definesymbol(t1);
                process2(t2,t3);
            }
            else if(op==11 || op==12) //ds or dc
                definesymbol(t1);
        }
    }
}
```

```
void process4(char t1[],char t2[],char t3[],char t4[])
{
    if(issymbol(t1))
    {
        op=searchoptab(t2);
        if(op>0 && op<9)
        {
            definesymbol(t1);
            process3(t2,t3,t4);
        }
        //else error: invalid opcode
    }
}
```

```
int main(int argc,char *argv[])
{
    FILE *fp;
    int i;

    if(argc==2)
        strcpy(fname,argv[1]);
    else
    {
        printf("Enter a source file name: ");
        scanf("%s",&fname);
    }

    fp=fopen(fname,"r");
    if(fp==NULL)
    {
        printf("File is not found");
        return 0;
    }
}
```

```

else
{
    while(!feof(fp) && flag==0)
    {
        fgets(instruction,40,fp);
        n=sscanf(instruction,"%s %s %s %s",t1,t2,t3,t4);
        switch(n)
        {
            case 1 : process1(t1);
                     break;
            case 2 : process2(t1,t2);
                     break;
            case 3 : process3(t1,t2,t3);
                     break;
            case 4 : process4(t1,t2,t3,t4);
                     break;
            default: printf("\nInvalid instruction");
        }

        lc++;
    }
    fclose(fp);
}

printf("Symbol table\n");
printf("\nSymbol Table\nSymbol\taddress\tdefine\tused\tlength\tvalue\n");
for(i=0;i<symcnt;i++)
    printf("%s\t%d\t%d\t%d\t%d\t%d\n",symtab[i].symbol,symtab[i].address,
    symtab[i].define,symtab[i].used,symtab[i].length,symtab[i].value);

printf("\n\nVariant I\n");
for(i=0;i<varcnt;i++)
    printf("%s\n",variant1[i]);

return 0;
}

```