ASSEMBLER DESIGN

&

SYMBOL TABLE CONSTRUCTION

ON A C++ PLATFORM

Presented by :

Sambhav R Jain

S Vignesh

Thanujan

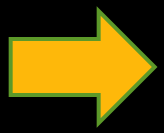
Department of Electricals & Electronics Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI

Steps

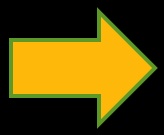
source.ass

```
ORG 1000H
LDA data1
JZ LOCNa
MVI data3 , 05
STA data2
JZ LOCNb
ADD data2
INR A
JMP LOCNb
HLT C
LOCNa LDA data3
LOCNb SUB B
data1 EQU 12H
data2 EQU 24H
data3 EQU 56H
A EQU 22
B EQU 23
C EQU 24
END
```



object.txt

```
1000 data1
1003 LOCNa
1006 data3
1008 data2
100b LOCNb
100e data2
100f A
1010 LOCNb
1013 C
1014 data3
1017 B
```



values.txt

```
LOCNa line 10
LOCNb line 11
data1 12H
data2 24H
data3 56H
A 22
B 23
C 24
```

FORMAT of Source File

The '*source.ass*' should comply to the following specifications

- Comments should follow after ';'
- The first line should be of the form "ORG 1000H" which would specify the starting address
- MVI command should be as "MVI A , 08" with at least one space between every word
- All EQU commands should be placed at the end of the file
- Keywords should be separated from symbol names by at least one space

Step1: source.ass → newsrc.txt

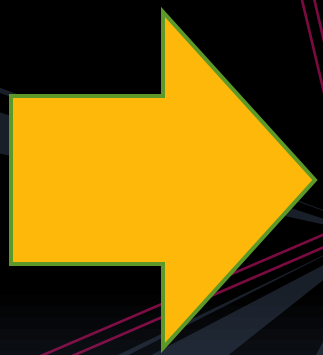
```
infile.open("source.ass");
outfile.open("newsrc.txt");
conv();
```

```
char* z=new char[50];
while(!infile.eof())
{
infile.getline(z,50,'\n');
for(int i=0;;i++)
{
if((z[i]==';')||(z[i]=='\0'))
break;
else
{
if(((z[i]==' ')||(z[i]=='\t'))&&((z[i+1]==' ')||(z[i+1]=='\0')||(z[i+1]==';')||(z[i+1]==
='\t')))
continue;
outfile<<z[i];
}
}
outfile<<'\n';
}
```

This function converts the given "source.ass" to "newsrc.txt" after removing comments and redundant spaces in the source file

source.ass

```
ORG    1000H          ; origin
LDA    data1          ; load
JZ     LOCNa          ; jump on zero
MVI    data3 , 05     ; move immediate
STA    data2
JZ     LOCNb          ; jump on zero
ADD    data2
INR    A
JMP    LOCNb          ; jump
HLT    C
LOCNa  LDA    data3
LOCNb  SUB    B
data1  EQU 12H
data2  EQU 24H
data3  EQU 56H
A      EQU 22
B      EQU 23
C      EQU 24
END
```



newsrc.txt

```
ORG 1000H
LDA data1
JZ LOCNa
MVI data3 , 05
STA data2
JZ LOCNb
ADD data2
INR A
JMP LOCNb
HLT C
LOCNa LDA data3
LOCNb SUB B
data1 EQU 12H
data2 EQU 24H
data3 EQU 56H
A EQU 22
B EQU 23
C EQU 24
END
```

Step2: newsrc.txt → object.txt

```
infile.open("newsrc.txt");
outfile.open("object.txt");
char hexadrs[4];    //To store the 4-bit starting address
p=new char[3];
infile>>p;
if(strcmp(p,"ORG")!=0)
{
cout<<"origin of the code not defined! TERMINATING!!";
return 0;
}
else
infile>>hexadrs[3]>>hexadrs[2]>>hexadrs[1]>>hexadrs[0];
adrs=hexadec(hexadrs);
outfile.setf(ios::hex,ios::basefield);
outfile<<adrs<<' ';
```

Check if origin is defined

Step2: newsrc.txt → object.txt

```
while(!infile.eof())
{
p=new char[10];
infile>>p;          //Extracts the assembler-specific keywords
int h=hash(p);
delete p;
……
}
```

Calls the hash table to return a unique hash value for each particular keyword

Hash Function

```
int hash(char* a)
{
if((strcmp(a,"LDA")==0)||(strcmp(a,"STA")==0)||(strcmp(a,"JMP")==0)
||(strcmp(a,"JZ")==0))
return 3;
else if(strcmp(a,"MVI")==0)
return 2;
else
if((strcmp(a,"ADD")==0)||(strcmp(a,"INR")==0)||(strcmp(a,"SUB")==0)
||(strcmp(a,"HLT")==0))
return 1;
else
return 0;
}
```

Step2: newsrc.txt → object.txt

```
while(!infile.eof())
{
.....
char* z;       //Retrieve and
store symbols
    switch(h)
    {
       case 3:
          adrs+=3;    //Since 3-
byte instructions
          z=new char[10];
          infile>>z;
          outfile<<z<<endl;
          outfile<<adrs<<' ';
          delete z;
          break;
        case 2:
          adrs+=2;    //Since 2-
byte instructions
          z=new char[10];
          infile>>z;
          outfile<<z<<endl;
          outfile<<adrs<<' ';
          infile>>z>>z
          delete z;
          break;
        case 1:
          adrs+=1;    //Since 1-
byte instructions
          z=new char[10];
          infile>>z;
          outfile<<z<<endl;
          outfile<<adrs<<' ';
          delete z;
    }
  }
```

```
1000 data1
1003 LOCNa
1006 data3
1008 data2
100b LOCNb
100e data2
100f A
1010 LOCNb
1013 C
1014 data3
1017 B
```

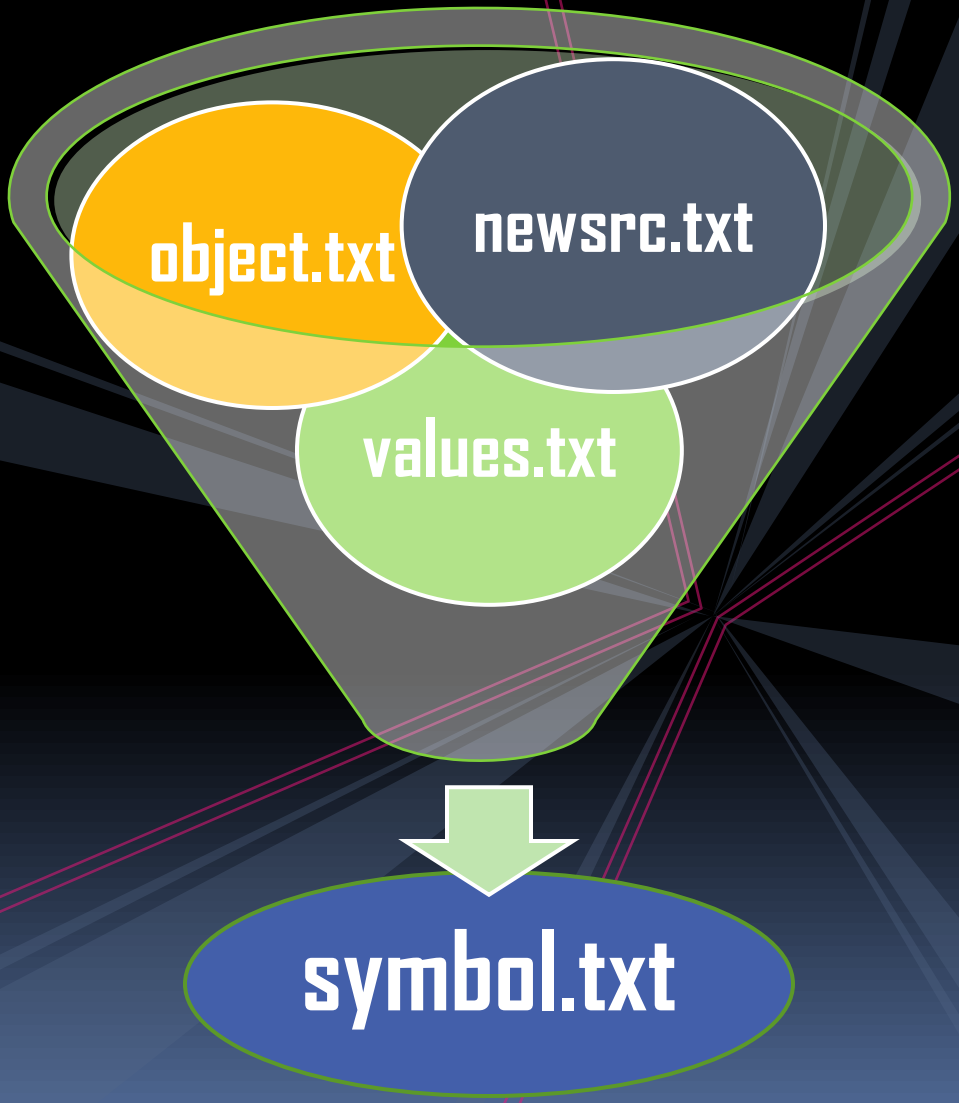
Step3: newsrc.txt → values.txt

```
infile.open("newsrc.txt");
outfile.open("values.txt");
value();
```

```
char* z=new char[30];
int i=0,s=0,n=0,m=0;
while(!infile.eof())
{
s=0;
infile.getline(z,30,'\n');
for(n=0;n<strlen(z);n++)
{
if(z[n]==' ')
s++;
}
if(s==2)
{
n=0;
outfile<<'\n';
while(z[n]!=' ')
{
outfile<<z[n];
n++;
}
outfile<<' ';
if((z[n+1]=='E')&&(z[n+2]=
='Q')&&(z[n+3]=='U'))
{
for(m=n+5;m<strlen(z);m++)
outfile<<z[m];
}
else
outfile<<"line"<<' '<<i;
}
i++;
}
```

```
LOCNa line 10
LOCNb line 11
data1 12H
data2 24H
data3 56H
A 22
B 23
C 24
```

Final steps



Symbol Table

SYMBOL	VALUE/ADDRESS	REFERENCED at
data1	12H	1000
LOCNa	1014	1003
data3	56H	1006,1014
data2	24H	1008,100e
LOCNb	1017	100b,1010
A	22	100f
C	24	1013
B	23	1017

THANK YOU