ALGORITHM for onePass

1. Till end of file

Read label, opcode, operand

if opcode == "START" then

set startAddress = operand

set loccter = hexToDecimal(operand)

Write the loccter, label, opcode, operand to intermediate file

Read label, opcode, operand

if label is not null

Add the label and loccter to the symbol table

Write the loccter, label, opcode, operand to intermediate file

if opcode == WORD then loccter += 3

else if opcode == RESW then loccter += atoi(operand)*3

else if opcode == BYTE then loccter += strlen(operand)

else if opcode == RESW then loccter += atoi(operand)

else loccter += 3

PROGRAM

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
struct SYMTAB {
        char symbol[30];
        char address[30];
} s[100];
```

```
int scount = 0;
int hexToDecimal (char acHex[]){
        int len, i, temp;
        int dec = 0;
        len = strlen(acHex);
        for (i=0; i<len; i++){
            switch (acHex[i]){
                case '0': temp = 0;
                          break;
                case '1': temp = 1;
                          break;
                case '2': temp = 2;
                          break;
                case '3': temp = 3;
                          break;
                case '4': temp = 4;
                          break;
                case '5': temp = 5;
                          break;
                case '6': temp = 6;
                          break;
                case '7': temp = 7;
                          break;
                case '8': temp = 8;
                          break;
                case '9': temp = 9;
                          break;
```

```
case 'A':
case 'a': temp = 10;
          break;

case 'B':
case 'b': temp = 11;
          break;

case 'C':
case 'c': temp = 12;
          break;

case 'D':
Case 'd': temp = 13;
          break;

case 'E':
case 'e': temp = 14;
          break;

case 'F':
case 'f': temp = 15;
          break;
}
    dec = dec + temp * pow(16, len-i-1);
}
return dec;
}
void onePass (){
    char label[30], opcode[30], operand[30], address[30];
    char loc[30];
    int flag, j;
```

```
int startAddress, locctr=0, i;
FILE *fp, *f1;
fp = fopen("source.txt", "r");
f1 = fopen("intermediate.txt", "w");
fscanf(fp, "%s %s %s", label, opcode, operand);
while(!feof(fp)){
    if(strcmp(opcode, "START")==0){
        startAddress = atoi(operand);
        locctr = hexToDecimal(operand);
        fprintf(f1, "%x\t%s\t%s\t%s\n", locctr, label, opcode, operand);
        fscanf(fp, "%s%s%s", label, opcode, operand);
    }
    if(strcmp(label, "**")!=0){
        strcpy(s[scount].symbol, label);
        sprintf(address, "%x", locctr);
        strcpy(s[scount++].address, address);
    }
    fprintf(f1, "%x\t%s\t%s\t%s\n", locctr, label, opcode, operand);
    if(strcmp(opcode, "WORD")==0){
        locctr += 3;
    }
    else if(strcmp(opcode, "RESW")==0){
        locctr += atoi(operand)*3;
    }
    else if(strcmp(opcode, "RESB")==0){
        locctr += atoi(operand);
```

```
}
else if (strcmp(opcode, "BYTE")==0){
        locctr += strlen(operand);
}
else {
     locctr += 3;
}
fscanf(fp,"%s%s%s", label, opcode, operand);
}
fclose(f1);
fclose(fp);
printf("Symbol Table\n");
for(i=0; i<scount; i++){
     printf("%s\t %s\n", s[i].symbol, s[i].address);
}
printf("Intermediate Code of Assembler\n");
f1 = fopen("intermediate.txt", "r");
fscanf(f1,"%s %s %s %s", loc, label, opcode, operand);
while(!feof(f1)){
     printf("%s\t%s\t%s\t%s\n", loc, label, opcode, operand);
     fscanf(f1,"%s %s %s %s", loc, label, opcode, operand);
}
fclose(f1);
}
void main(){
     onePass();
}
```