Pass 1 of 2-pass Assembler

/*Initially create 2 files **input.txt & optab.txt** in same directory, add the input After compiling pass1.c & executing a.out , **length.txt, intermediate.txt & symtab.txt** will be created and output will be displayed on terminal*/

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void passOne(char label[10], char opcode[10], char operand[10], char code[10], char
mnemonic[3]);

void display();

int main()

{

    // for reading from input

    char label[10], opcode[10], operand[10];

    // for reading from optab

    char code[10], mnemonic[3];

    // call the function

    passOne(label, opcode, operand, code, mnemonic);

    return 0;

}

void passOne(char label[10], char opcode[10], char operand[10], char code[10], char
mnemonic[3])

{

    int locctr, start, length;

    FILE *fp1, *fp2, *fp3, *fp4, *fp5;                              // file pointers
    // read mode
    fp1 = fopen("input.txt", "r");
    fp2 = fopen("optab.txt", "r");
    // write mode
    fp3 = fopen("symtab.txt", "w");
    fp4 = fopen("intermediate.txt", "w");
    fp5 = fopen("length.txt", "w");
    fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);              // read first line
    if (strcmp(opcode, "START") == 0)
    {
        // atoi() requires stdlib.h header file , it converts ASCII to integer
        start = atoi(operand);                                  // convert operand value from string to integer
and assign to start
        locctr = start;
        fprintf(fp4, "\t%s\t%s\t%s\n", label, opcode, operand);     // write to output file (additional
tab space as start will not have any locctr)
        fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);          // read next line
    }
    else
    {
        locctr = 0;
    }
    // iterate till end
    while (strcmp(opcode, "END") != 0)
    {
```

```
// transfer address and read line to output file
fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode, operand);
// make symtab file with values not starting with **
if (strcmp(label, "**") != 0)
{
    fprintf(fp3, "%s\t%d\n", label, locctr);
}
// read from optab (code and mnemonic value)
fscanf(fp2, "%s\t%s", code, mnemonic);
// traverse till the end of optab file
while (strcmp(code, "END") != 0)
{
    if (strcmp(opcode, code) == 0)
    {                       // if opcode in input matches the one in optab, increment locctr by 3
        locctr += 3;
        break;
    }
    // read next line
    fscanf(fp2, "%s\t%s", code, mnemonic);
}
// Searching opcode for WORD, RESW, BYTE, RESB keywords and updating locctr
// WORD -> add 3 to locctr
if (strcmp(opcode, "WORD") == 0)
{
    locctr += 3;
```

```
        }
                // RESW -> add 3*operand to locctr
        else if (strcmp(opcode, "RESW") == 0)
        {
            locctr += (3 * (atoi(operand)));                        // convert operand to integer and multiply
with 3
        }
                // BYTE -> add 1 to locctr
        else if (strcmp(opcode, "BYTE") == 0)
        {
            locctr++;
        }
                // RESB -> add operand to locctr
        else if (strcmp(opcode, "RESB") == 0)
        {
            locctr += atoi(operand);
        }
            // read next line
        fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
    }
    //  transfer last line to file
    fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode, operand);
    // Close all files
    fclose(fp4);
    fclose(fp3);
    fclose(fp2);

    fclose(fp1);

    // 8. display outputs

    display();

        // calculate length of program

    length = locctr - start;

    fprintf(fp5, "%d", length);

    fclose(fp5);

    printf("\nThe length of the code : %d\n", length);

}

void display()

{

    char str;

    FILE *fp1, *fp2, *fp3;

    // display content of Input Table

    printf("\nThe contents of Input Table :\n\n");

    fp1 = fopen("input.txt", "r");

    str = fgetc(fp1);

    while (str != EOF)

    {

        printf("%c", str);

        str = fgetc(fp1);

    }

    fclose(fp1);

    // display content of  Output Table
    printf("\n\nThe contents of Output Table :\n\n");
    fp2 = fopen("intermediate.txt", "r");
    str = fgetc(fp2);
    while (str != EOF)
    {
        printf("%c", str);
        str = fgetc(fp2);
    }
    fclose(fp2);
    // display content of  Symtable
    printf("\n\nThe contents of Symbol Table :\n\n");
    fp3 = fopen("symtab.txt", "r");
    str = fgetc(fp3);
    while (str != EOF)
    {
        printf("%c", str);
        str = fgetc(fp3);
    }
    fclose(fp3);
}
```

Create 2 input files

Input.txt and optab.txt

1. Input.txt

```
**      START       2000
**      LDA   FIVE
**      STA   ALPHA
**      LDCH  CHARZ
**      STCH  C1
ALPHA         RESW        2
FIVE    WORD        5
CHARZ         BYTE  C'Z'
C1      RESB  1
**      END   **
```

2. optab.txt

```
LDA 03
STA     0f
LDCH 53
STCH 57
END     *
```