Name: Brijesh Rameshbhai Rohit

Admission number: U19CS009

SYSTEM SOFTWARES – ASSIGNMENT - 05

Write a program to implement two pass assembler.

Code=>

PASS1:

```
// U19CS009
// BRIJESH ROHIT
// pass1

#include <stdio.h>
#include <stdlib.h>
#include "mnetype.h"
#include<string.h>

char label[20],mnem[20], opr1[20],sym[20],lit[20];
struct symbol_tab symtab[1000];
struct literal_tab littab[1000];
int sym_ct=0,lit_ct=0;

int islit(char * str)
{
    if(str[0]=='=')return 1;
    return 0;
}

void parse(char *inst)
{
    strcpy(label,"");
    strcpy(mnem,"");
    strcpy(opr1,"");
    strcpy(sym,"");
    strcpy(lit,"");
    int i=0,j=0;
    char word[5][20];
    char temp[20]="";
    for(j=0;j<strlen(inst);j++)
    {
        //printf("%d %c\n",j,inst[j]);
        if(inst[j]==':'||inst[j]==' '||inst[j]==',')
        {
            if((inst[j]==','&&inst[j+1]==' ')||(inst[j]==':'&&inst[j+1]=='
'))continue;

            strcpy(word[i],temp);
            strcpy(temp,"");
```

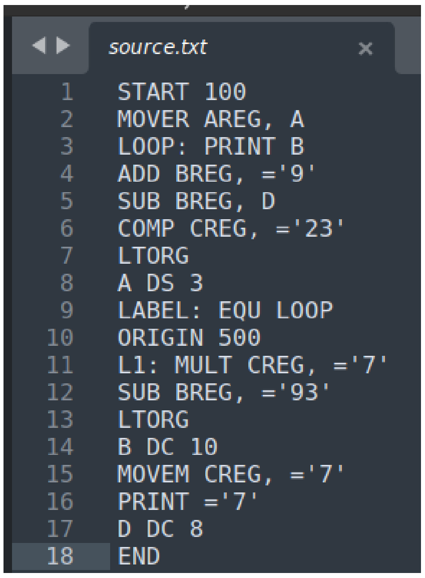
```c
            i++;
            continue;
        }
        else
        {
            char c[2];
            c[0]=inst[j];
            c[1]='\0';
            strcat(temp,c);
        }
    }
    strcpy(word[i],temp);
    i++;
    printf("%d\n",i);
    if(i==0)
    {
        return;
    }
    if(i==1)
    {
        strcpy(mnem,word[0]);
    }
    else if(i==2)
    {
        strcpy(mnem,word[0]);
        if(islit(word[1]))
        {
            strcpy(lit,word[1]);
        }
        else
        {
            strcpy(sym,word[1]);
        }

    }
    else if(i==3)
    {
        strcpy(mnem,word[0]);
        strcpy(opr1,word[1]);
        //strcpy(opr2,word[2]);
        if(islit(word[2]))
        {
            strcpy(lit,word[2]);
        }
        else
        {
            strcpy(sym,word[2]);
        }
    }
    else
    {
        strcpy(label,word[0]);
        strcpy(mnem,word[1]);
        strcpy(opr1,word[2]);

        if(islit(word[3]))
        {
            strcpy(lit,word[3]);
        }
        else
        {
            strcpy(sym,word[3]);
        }
    }
}

int insym(char *sym1)
{
    int i=0;
    while(i<sym_ct)
    {
        if(strcpy(symtab[i].symbol,sym1)==0)
        {
            return 1;
        }
        i++;
    }
    return 0;
}

int inlit(char *lit1)
{
    int i=0;
    while(i<lit_ct)
    {
        if(strcpy(littab[i].literal,lit1)==0)
        {
            return 1;
        }
        i++;
    }
    return 0;
}


int main()
{



    set();
    FILE *src,*inter;
    src=fopen("input.txt","r");
    inter=fopen("inter.txt","w");

    char inst[20];
    fgets(inst,20,src);

    printf("hell\n");
    int lc=100;
    parse("MOVER AREG, ='5'");
    char * intline="";
    int i=0;

    return 0;
}
```

PASS2:

```
// U19CS009
// BRIJESH ROHIT
//pass 2

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

// to extract the numeric val depending on the type of mnemonic used.
char *extract(char *str)
{
    //- MEANING NOT REQUIRED THUS 00.
    if (strcmp(str, "-") == 0)
        return "00";

    // IF THE STR IS NUMERIC THEN RETURN THE STRING ITSELF.
    if (str[0] >= '0' && str[0] <= '9')
    {
        return str;
    }

    char *final = malloc(sizeof(char) * 4);
    strcpy(final, "");
    int i = 0;
    for (i = 0; i < strlen(str); i++)
    {
        if (str[i] == ',')
            break;
    }
    i++;
    int j = 0;
    while (i < strlen(str))
    {
        char ch[2] = {str[i], '\0'};
        strcat(final, ch);
        i++;
    }
```

```c

    // IF STR[0]=='S' THEN SEARCHING 'final' FROM THE SYMBOL TABLE TO RETURN
ADDRESS.
    if (str[0] == 'S')
    {
        FILE *f = fopen("symbol.txt", "r");
        int n = atoi(final);
        int x;
        char t1[8], t2[8];
        while (fscanf(f, "%d%s%s", &x, t1, t2) != EOF)
        {
            if (n == x)
            {
                strcpy(final, t2);
                break;
            }
        }
        fclose(f);
    }

    // IF STR[0]=='L' THEN SEARCHING 'final' FROM THE LITERAL TABLE TO RETURN
ADDESSS.
    if (str[0] == 'L')
    {
        FILE *f = fopen("literal.txt", "r");
        int n = atoi(final);
        int x;
        char t1[8], t2[8];
        while (fscanf(f, "%d %s %s", &x, t1, t2) != EOF)
        {
            if (n == x)
            {
                strcpy(final, t2);
                break;
            }
        }
        fclose(f);
    }

    return final;
}

int main()
{
    char lc[8], mnem[8], reg[8], symlit[8];
    FILE *fin, *fsym, *flit, *fout;

    // OPENING FILES.
    fin = fopen("inter.txt", "r");
    fout = fopen("obcode.txt", "w");

    fscanf(fin, "%s%s%s", mnem, reg, symlit);

    char *s = extract(mnem);
    char *s1 = extract(reg);
    char *s2 = extract(symlit);
    // printf("%s %s %s\n",s,s1,s2);
    fprintf(fout, "%s %s %s\n", s, s1, s2);

    // AD,02 STANDS FOR 'END'.
    while (strcmp(mnem, "AD,02") != 0)
    {

        fscanf(fin, "%s %s %s %s", lc, mnem, reg, symlit);

        char op[8], op1[8], op2[8];

        // FOR ASSEMBLER DIRECTIVES
        if (mnem[0] == 'A' && mnem[1] == 'D')
        {
            s2 = extract(symlit);
            fprintf(fout, "%s) 00 00 %s\n", lc, s2);
        }

        // FOR IMPERATIVE STATEMENTS
        else if (mnem[0] == 'I' && mnem[1] == 'S')
        {
            s = extract(mnem);
            s1 = extract(reg);
            s2 = extract(symlit);
            fprintf(fout, "%s) %s %s %s\n", lc, s, s1, s2);
        }

        // FOR DECLARATIVE STATEMENTS
        else if (mnem[0] == 'D' && mnem[1] == 'L')
        {
            //DECLARATIVE STORAGE STATEMENT
            if(mnem[3]=='0'&&mnem[4]=='1')
            {
                fprintf(fout, "%s) 00 00 000\n", lc);
            }
            else{
                s2 = extract(symlit);
            fprintf(fout, "%s) 00 00 %s\n", lc, s2);
            }

        }
    }
    // CLOSING THE FILES
    fclose(fout);
    fclose(fin);

    return 0;
}
```

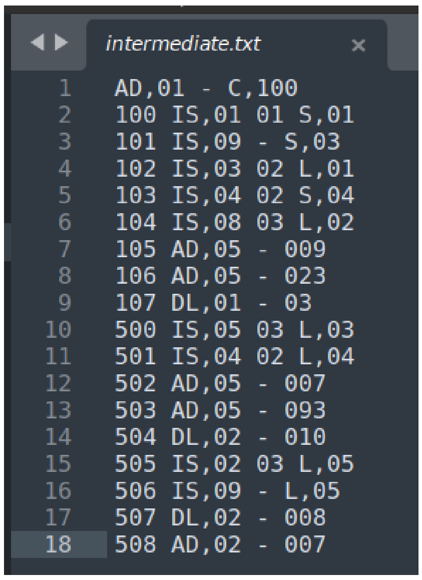
SOURCE CODE :


source.txt

```
START 100
MOVER AREG, A
LOOP: PRINT B
ADD BREG, ='9'
SUB BREG, D
COMP CREG, ='23'
LTORG
A DS 3
LABEL: EQU LOOP
ORIGIN 500
L1: MULT CREG, ='7'
SUB BREG, ='93'
LTORG
B DC 10
MOVEM CREG, ='7'
PRINT ='7'
D DC 8
END
```

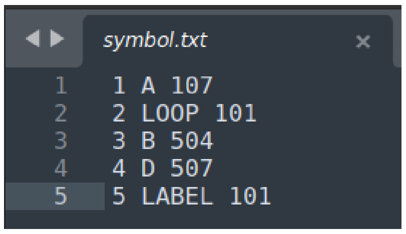
INPUT=>

INTERMEDIATE CODE :


intermediate.txt

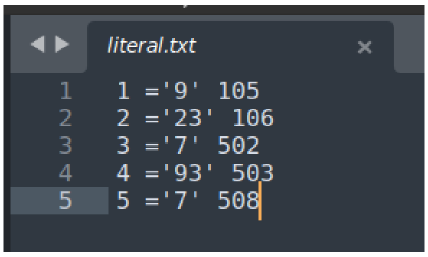
```
AD,01 - C,100
100 IS,01 01 S,01
101 IS,09 - S,03
102 IS,03 02 L,01
103 IS,04 02 S,04
104 IS,08 03 L,02
105 AD,05 - 009
106 AD,05 - 023
107 DL,01 - 03
500 IS,05 03 L,03
501 IS,04 02 L,04
502 AD,05 - 007
503 AD,05 - 093
504 DL,02 - 010
505 IS,02 03 L,05
506 IS,09 - L,05
507 DL,02 - 008
508 AD,02 - 007
```

SYMBOL TABLE :


symbol.txt

```
1 A 107
2 LOOP 101
3 B 504
4 D 507
5 LABEL 101
```

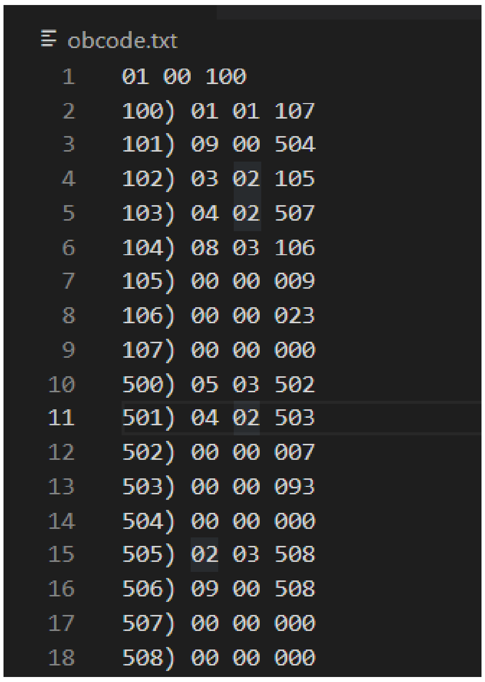
LITERAL TABLE :



```
literal.txt
1 ='9' 105
2 ='23' 106
3 ='7' 502
4 ='93' 503
5 ='7' 508
```

OUTPIT=>

OBJECT CODE :



```
obcode.txt
01 00 100
100) 01 01 107
101) 09 00 504
102) 03 02 105
103) 04 02 507
104) 08 03 106
105) 00 00 009
106) 00 00 023
107) 00 00 000
500) 05 03 502
501) 04 02 503
502) 00 00 007
503) 00 00 093
504) 00 00 000
505) 02 03 508
506) 09 00 508
507) 00 00 000
508) 00 00 000
```