Pass 1 Assembler :-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 20
struct symbol {
   char sym[10];
   int addr;
} S[MAX];
struct litab {
   char lit[10];
   int addr;
} L[MAX];
void print_file(char *);
void print_symtab();
void print_littab();
char optab[][6] = {"STOP", "ADD", "SUB", "MULT", "MOVER", "MOVEM", "COMP", "BC", "DIV", "READ",
"PRINT"};
char regtab[][5] = {"AREG", "BREG", "CREG", "DREG"};
char adtab[][7] = {"START", "END", "ORIGIN", "EQU", "LTORG"};
char condtab[][4] = {"LT", "LE", "EQ", "GT", "GE", "ANY"};
FILE *fs, *ft;
char buffer[80], source[80], tok1[10], tok2[10], tok3[10], tok4[10];
int lc = 0, sc = 0, poolcnt = 0, litcnt = 0;
int pooltab[10];
int search_optab(char *s) {
   for (int i = 0; i < 11; i++) {
      if (strcmp(optab[i], s) == 0) {
         return i;
      }
   }
   return -1;
}

int search_regtab(char *s) {
   for (int i = 0; i < 4; i++) {
      if (strcmp(regtab[i], s) == 0) {
         return i + 1;
      }
   }
   return -1;
}
int search_condtab(char *s) {
   for (int i = 0; i < 6; i++) {
      if (strcmp(condtab[i], s) == 0) {
         return i + 1;
```

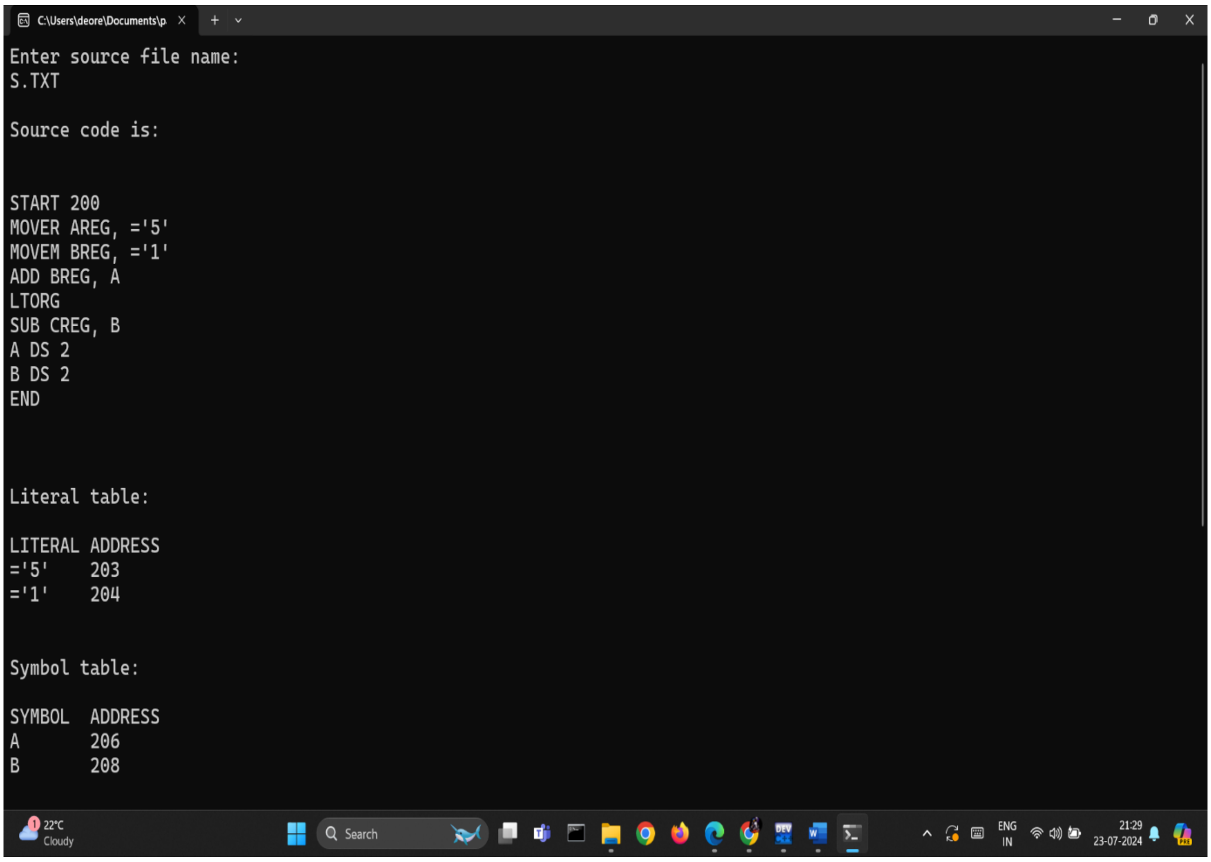
```
        }
    }
    return -1;
}
int search_adtab(char *s) {
    for (int i = 0; i < 5; i++) {
        if (strcmp(adtab[i], s) == 0) {
            return i + 1;
        }
    }
    return -1;
}
int search_symtab(char *s) {
    for (int i = 0; i < sc; i++) {
        if (strcmp(S[i].sym, s) == 0) {
            return i;
       } return -1;
}

int search_littab(char *s) {
    for (int i = pooltab[poolcnt]; i < litcnt; i++) {
        if (strcmp(L[i].lit, s) == 0) {
            return i;
        }
    }
    return -1;
}

void pass1() {
    int p, n, i = 0, j = 0, k = 0;

    fs = fopen(source, "r");
    if (fs == NULL) {
        printf("\nFile does not exist!\n");
        exit(0);
    }

    ft = fopen("id.txt", "w");
    if (ft == NULL) {
        printf("\nError creating intermediate file!\n");
        exit(0);
    }

    while (fgets(buffer, 80, fs)) {
        n = sscanf(buffer, "%s%s%s%s", tok1, tok2, tok3, tok4);
        switch (n) {
            case 1: // ltorg, end
                i = search_adtab(tok1);
                if (i == 2 || i == 5) {
                    for (j = pooltab[poolcnt]; j < litcnt; j++) {
                        L[j].addr = lc++;
                    }
                    lc--;
                    pooltab[++poolcnt] = litcnt;
                    fprintf(ft, "(AD, %02d)\n", i);
                    break;
                }
            case 2: // start
                i = search_adtab(tok1);
                if (i == 1) {
                    lc = atoi(tok2) - 1;
                    fprintf(ft, "(AD, %02d)  (C, %s)\n", i, tok2);
                    break;
                }
            case 3: // instructions
                i = search_optab(tok1);
                if (i >= 1 && i <= 8) {
                    tok2[strlen(tok2) - 1] = '\0';
                    k = search_regtab(tok2);

                    if (tok3[0] == '=') {
                        j = search_littab(tok3);
                        if (j == -1) {
                            strcpy(L[litcnt].lit, tok3);
                            fprintf(ft, "(IS, %02d)    %d    (L, %02d)\n", i, k, litcnt);
                            litcnt++;
                        } else {
                            fprintf(ft, "(IS, %02d)    %d    (L, %02d)\n", i, k, j);
                        }
                        break;
                    } else {
                        p = search_symtab(tok3);
                        if (p == -1) {
                            strcpy(S[sc].sym, tok3);
                            fprintf(ft, "(IS, %02d)    %d    (S, %02d)\n", i, k, sc);
                            sc++;
                        } else {
                            fprintf(ft, "(IS, %02d)    %d    (S, %02d)\n", i, k, p);
                        }
                        break;
                    }
                }

                if (strcmp(tok2, "DS") == 0) {
                    p = search_symtab(tok1);
                    if (p == -1) {
```

```
                    strcpy(S[sc].sym, tok1);
                    S[sc].addr = lc;
                    fprintf(ft, "(DL, 2)    (C, %s)\n", tok3);
                    sc++;
                } else {
                    S[p].addr = lc;
                    fprintf(ft, "(DL, 2)    (C, %s)\n", tok3);
                }
                lc = lc + atoi(tok3) - 1;
                break;
            }
        }
        lc++;
    }

    fclose(fs);
    fclose(ft);
}

void print_file(char *target) {
    FILE *fp = fopen(target, "r");
    if (fp == NULL) {
        printf("\nFile does not exist!\n");
        return;
    }
    printf("\n\n");
    while (fgets(buffer, 80, fp)) {
        printf("%s", buffer);
    }
    fclose(fp);
}

void print_littab() {
    printf("\nLITERAL\tADDRESS\n");
    for (int i = 0; i < litcnt; i++) {
        printf("%s\t%d\n", L[i].lit, L[i].addr);
    }
}

void print_symtab() {
    printf("\nSYMBOL\tADDRESS\n");
    for (int p = 0; p < sc; p++) {
        printf("%s\t%d\n", S[p].sym, S[p].addr);
    }
}

int main() {
    printf("\nEnter source file name: \n");
    scanf("%s", source);

    printf("\nSource code is: \n");
    print_file(source);

    pass1();

    printf("\n\nLiteral table: \n");
    print_littab();

    printf("\n\nSymbol table: \n");
    print_symtab();

    printf("\nIntermediate code is: \n");
    print_file("id.txt");

    return 0;
}
```

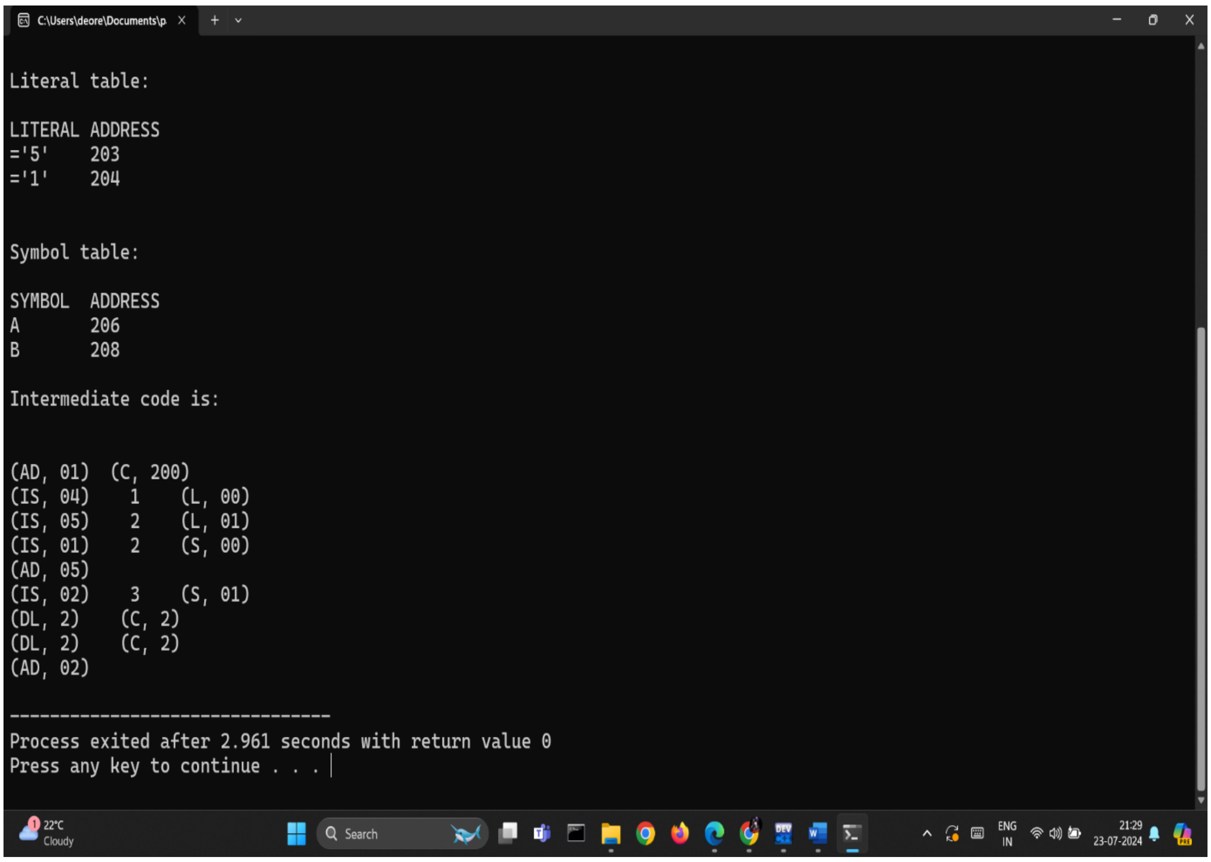
S.txt

```
START 200
MOVER AREG, ='5'
MOVEM BREG, ='1'
ADD BREG, A
LTORG
SUB CREG, B
A DS 2
B DS 2
END
```

Output :-



```
Enter source file name:
S.TXT

Source code is:


START 200
MOVER AREG, ='5'
MOVEM BREG, ='1'
ADD BREG, A
LTORG
SUB CREG, B
A DS 2
B DS 2
END



Literal table:

LITERAL ADDRESS
='5'    203
='1'    204


Symbol table:

SYMBOL  ADDRESS
A       206
B       208
```



```
Literal table:

LITERAL ADDRESS
='5'    203
='1'    204


Symbol table:

SYMBOL  ADDRESS
A       206
B       208

Intermediate code is:


(AD, 01)  (C, 200)
(IS, 04)    1    (L, 00)
(IS, 05)    2    (L, 01)
(IS, 01)    2    (S, 00)
(AD, 05)
(IS, 02)    3    (S, 01)
(DL, 2)    (C, 2)
(DL, 2)    (C, 2)
(AD, 02)

--------------------------------
Process exited after 2.961 seconds with return value 0
Press any key to continue . . .
```

Pass 2 Assembler :-

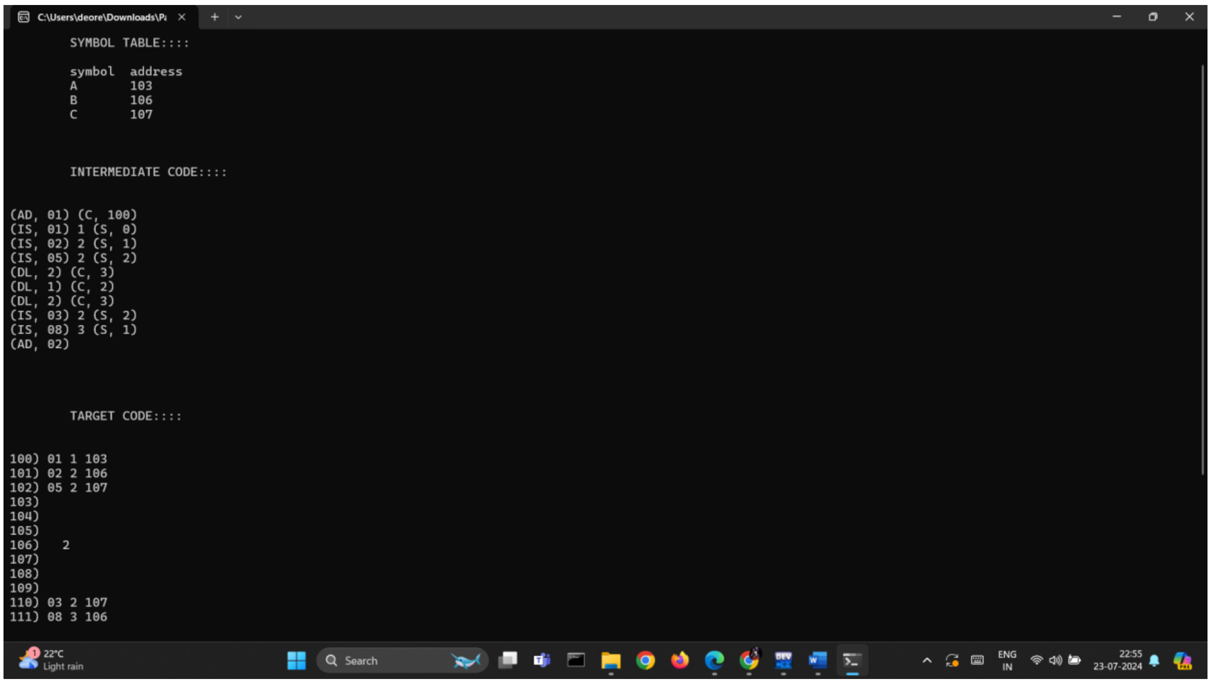
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 20

struct symbol {
   char sym[10];
   int addr;
} S[MAX] = {
   {"A", 103},
   {"B", 106},
   {"C", 107}
};

char optab[][6] = {"STOP", "ADD", "SUB", "MULT",
               "MOVER",  "MOVEM",  "COMP",  "BC",  "DIV",  "READ",
"PRINT"};
char regtab[][5] = {"AREG", "BREG", "CREG", "DREG"};
char condtab[][4] = {"LT", "LE", "EQ", "GT", "GE", "ANY"};
char adtab[][7] = {"START", "END", "ORIGIN", "EQU", "LTORG"};

FILE *fs, *ft;
char  source[20],  buffer[80],  tok1[10],  tok2[10],  tok3[10],  tok4[10],
tok5[10];
int lc, ec = 0, sc = 3;

int search_optab(char *s) {
   for (int i = 0; i < 11; i++) {
       if (strcmp(optab[i], s) == 0)
           return i;
   }
   return -1;
}

int search_symb(char *s) {
   for (int i = 0; i < sc; i++) {
       if (strcmp(S[i].sym, s) == 0)
           return i;
   }
   return -1;
}

int search_regtab(char *s) {
   for (int i = 0; i < 4; i++) {
       if (strcmp(regtab[i], s) == 0)
           return i + 1;
   }
   return -1;
}

int search_condtab(char *s) {
   for (int i = 0; i < 6; i++) {
       if (strcmp(condtab[i], s) == 0)
           return i + 1;
   }
   return -1;
}

int search_adtab(char *s) {
   for (int i = 0; i < 5; i++) {
       if (strcmp(adtab[i], s) == 0)
           return i + 1;
   }
   return -1;
}

void print_file(char *target) {
   FILE *fp = fopen(target, "r");
   if (fp == NULL) {
       printf("\nError In Opening File\n");
       exit(0);
   }
   printf("\n\n");
   while (fgets(buffer, 80, fp)) {
       printf("%s", buffer);
   }
   fclose(fp);
}

void passtwo() {
   int i, j, k, n;

   if ((fs = fopen("ic.txt", "r")) == NULL) {
       printf("\n\nError In Opening ic.txt\n");
       exit(0);
   }
   if ((ft = fopen("target.txt", "w")) == NULL) {
       printf("\n\nError In Opening target.txt\n");
       fclose(fs);
       exit(0);
   }
   lc = 0;
   while (fgets(buffer, 80, fs)) {
       n = sscanf(buffer, "%s%s%s%s%s", tok1, tok2, tok3, tok4, tok5);
```

```
switch (n) {
    case 4:
        if (strcmp(tok1, "(AD,") == 0) {
            tok4[strlen(tok4) - 1] = '\0';
            lc = atoi(tok4) - 1;
            break;
        }
        if (strcmp(tok1, "(DL,") == 0) {
            tok2[strlen(tok2) - 1] = '\0';
            tok4[strlen(tok4) - 1] = '\0';
            i = atoi(tok2);
            j = atoi(tok4);
            if (i == 2) {  // A DS 2
                for (k = 0; k < j; k++) {
                    fprintf(ft, "%d)\n", lc++);
                }
                lc--;
            } else {  // ONE DC 1
                fprintf(ft, "%d)   %d\n", lc, j);
            }
        }
        break;

    case 5:
        tok2[strlen(tok2) - 1] = '\0';
        tok5[strlen(tok5) - 1] = '\0';
        i = atoi(tok2);
        j = atoi(tok3);
        k = atoi(tok5);
        if (strcmp(tok4, "(S,") == 0) {
            fprintf(ft, "%d) %02d %d %03d\n", lc, i, j, S[k].addr);
        }
        break;
        }  // switch
        lc++;
    }  // while
    fclose(fs);
    fclose(ft);
}
void print_symb() {
    printf("\n\tsymbol\taddress\n");
    for (int p = 0; p < sc; p++) {
        printf("\t%s\t%d\n", S[p].sym, S[p].addr);
    }
}

int main() {
    printf("\n\n\tSYMBOL TABLE::::\n");
    print_symb();

    printf("\n\n\n\tINTERMEDIATE CODE::::\n");
    print_file("ic.txt");

    passtwo();

    printf("\n\n\tTARGET CODE::::\n");
    print_file("target.txt");

    return 0;
}
```

IC.txt

```
(AD, 01) (C, 100)
(IS, 01) 1 (S, 0)
(IS, 02) 2 (S, 1)
(IS, 05) 2 (S, 2)
(DL, 2) (C, 3)
(DL, 1) (C, 2)
(DL, 2) (C, 3)
(IS, 03) 2 (S, 2)
(IS, 08) 3 (S, 1)
(AD, 02)
```

OUTPUT :-



```
        SYMBOL TABLE::::

        symbol  address
        A       103
        B       106
        C       107



        INTERMEDIATE CODE::::


(AD, 01) (C, 100)
(IS, 01) 1 (S, 0)
(IS, 02) 2 (S, 1)
(IS, 05) 2 (S, 2)
(DL, 2) (C, 3)
(DL, 1) (C, 2)
(DL, 2) (C, 3)
(IS, 03) 2 (S, 2)
(IS, 08) 3 (S, 1)
(AD, 02)



        TARGET CODE::::


100) 01 1 103
101) 02 2 106
102) 05 2 107
103)
104)
105)
106)   2
107)
108)
109)
110) 03 2 107
111) 08 3 106
```