Code

// Linking Loader Pass 1

// Requires ll-input.txt as input

// Writes output in ESTAB.txt

// 1. Processing Header records

// 2. Processing Define records

// 3. Added ESTAB search to check for errors

/\*

Control | Symbol | Address | Length

Section | Name |

------------------------------

PROGA 500 63

PROGB 563 7F

PROGC 5E2 51

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define BIG 1024

#define SMOL 128

// Structure to hold components of Header Record

struct H\_ESTAB {

char csname[SMOL];

char extsym[SMOL];

int address;

int length;

};

// Structure to hold components of Define Record

struct D\_ESTAB {

char csname[SMOL];

char varname[SMOL];

int varaddress;

};

struct H\_ESTAB getHeaderTokens(char line[]) {

struct H\_ESTAB temp\_record;

char \*type;

type = strtok(line, "^");

strcpy(temp\_record.csname, strtok(NULL, "^"));

strcpy(temp\_record.extsym, "");

temp\_record.address = (int)strtol(strtok(NULL, "^"), NULL, 16);

temp\_record.length = (int)strtol(strtok(NULL, "^"), NULL, 16);

return temp\_record;

}

// Function returns 1 if CSNAME is found in ESTAB, else returns 0

int check\_H\_ESTAB\_Error(struct H\_ESTAB Hrecord, struct H\_ESTAB Hrecords[],

struct D\_ESTAB Drecords[], int Hcount, int Dcount) {

// rewind(fp2);

// No need to scan ESTAB file

// Search in Hrecords and Drecords Array instead

if (Hcount == 0) {

return 0;

}

for (int i = 0; i < Hcount; i++) {

if (!strcmp(Hrecord.csname, Hrecords[i].csname)) {

return 1;

}

}

if (Dcount == 0) {

return 0;

}

for (int i = 0; i < Dcount; i++) {

if (!strcmp(Hrecord.csname, Drecords[i].varname)) {

return 1;

}

}

return 0;

}

// Function returns 1 if variable is found in ESTAB, else returns 0

int check\_D\_ESTAB\_Error(struct D\_ESTAB Drecord, struct H\_ESTAB Hrecords[],

struct D\_ESTAB Drecords[], int Hcount, int Dcount) {

// rewind(fp2);

// No need to scan ESTAB file

// Search in Hrecords and Drecords Array instead

if (Dcount == 0) {

return 0;

}

for (int i = 0; i < Dcount; i++) {

if (!strcmp(Drecord.varname, Drecords[i].varname)) {

return 1;

}

}

if (Hcount == 0) {

return 0;

}

for (int i = 0; i < Hcount; i++) {

if (!strcmp(Drecord.varname, Hrecords[i].csname)) {

return 1;

}

}

return 0;

}

// Process input file

int Process(FILE \*fp1, FILE \*fp2, struct H\_ESTAB Hrecords[],

struct D\_ESTAB Drecords[], int PROGADDR) {

int line\_count = 0;

int Hcount = 0;

int Dcount = 0;

while (!feof(fp1)) {

char line[SMOL];

fgets(line, SMOL, fp1);

// If there is an empty line

if (line[0] == '\n') {

continue;

}

// fgets things

if (line[strlen(line)] == '\n') {

line[strlen(line) - 1] = '\0';

}

// If line is a Header Record

if (line[0] == 'H' && line[1] == '^') {

// Hrecords[Hcount] = getHeaderTokens(line);

struct H\_ESTAB Htemp = getHeaderTokens(line);

// Check for repeated name

int flag = check\_H\_ESTAB\_Error(Htemp, Hrecords, Drecords, Hcount, Dcount);

// If no error

if (flag == 0) {

// Copy Htemp to Hrecords Array

strcpy(Hrecords[Hcount].csname, Htemp.csname);

strcpy(Hrecords[Hcount].extsym, Htemp.extsym);

Hrecords[Hcount].address = Htemp.address;

Hrecords[Hcount].length = Htemp.length;

// Add PROGADDR from user to first record

if (Hcount == 0) {

Hrecords[Hcount].address += PROGADDR;

}

// Next records continue after previous ones

else if (Hcount > 0) {

Hrecords[Hcount].address +=

Hrecords[Hcount - 1].address + Hrecords[Hcount - 1].length;

}

// Writing in ESTAB file

printf("%s\t%s\t%04X\t%04X\n", Hrecords[Hcount].csname,

Hrecords[Hcount].extsym, Hrecords[Hcount].address,

Hrecords[Hcount].length);

fprintf(fp2, "%s\t%s\t%04X\t%04X\n", Hrecords[Hcount].csname,

Hrecords[Hcount].extsym, Hrecords[Hcount].address,

Hrecords[Hcount].length);

Hcount++;

} else if (flag == 1) {

printf("\nError at line %d: %s is repeated\n", line\_count,

Htemp.csname);

}

}

// If line is a Define Record

else if (line[0] == 'D' && line[1] == '^') {

// printf("Here is a Define\n");

char holder[SMOL];

holder[0] = '\0';

int toggle = 0;

for (int i = 2; i < strlen(line) + 1; i++) {

char temp = line[i];

if (line[i] == '^' || line[i] == '\0') {

struct D\_ESTAB Dtemp;

// strcat(holder,"\0");

if (toggle == 0) {

// printf("Here is a variable\n");

strcpy(Dtemp.varname, holder);

toggle = 1;

}

else if (toggle == 1) {

Dtemp.varaddress = (int)strtol(holder, NULL, 16);

// int h = Hcount - 1;

Dtemp.varaddress += Hrecords[Hcount - 1].address;

toggle = 0;

// Check for repeated name

int flag =

check\_D\_ESTAB\_Error(Dtemp, Hrecords, Drecords, Hcount, Dcount);

// If no error

if (flag == 0) {

strcpy(Drecords[Dcount].varname, Dtemp.varname);

Drecords[Dcount].varaddress = Dtemp.varaddress;

// Write Define variables in ESTAB

printf("\t%s\t%04X\n", Drecords[Dcount].varname,

Drecords[Dcount].varaddress);

fprintf(fp2, "\t%s\t%04X\n", Drecords[Dcount].varname,

Drecords[Dcount].varaddress);

Dcount++;

} else if (flag == 1) {

printf("\nError at line %d: %s is repeated\n", line\_count,

Dtemp.varname);

}

}

holder[0] = '\0';

continue;

}

char temp\_star[2] = {temp, '\0'}; /\* gives {value of temp, \0} \*/

strcat(holder, temp\_star);

}

}

line\_count++;

}

return line\_count;

}

int main() {

FILE \*fp1 = fopen("input.txt", "r"); // Open input file in read mode

FILE \*fp2 = fopen("ESTAB.txt", "w"); // Open output file in write mode

if (fp1 == NULL) {

perror(" => Error opening the Input read file");

return -1;

}

if (fp2 == NULL) {

printf(" => Error opening the ESTAB write file");

return -2;

}

struct H\_ESTAB Hrecords[BIG];

struct D\_ESTAB Drecords[BIG];

int PROGADDR;

printf("Enter the starting location to load the program: ");

scanf("%X", &PROGADDR);

printf("\nCS\_NAME SYM\_NAME ADDR\tLEN\n");

printf("---------------------------------------------\n");

int line\_count = Process(fp1, fp2, Hrecords, Drecords, PROGADDR);

printf("---------------------------------------------\n");

printf("\n Number of lines in Input file = %d\n", line\_count);

fclose(fp1);

fclose(fp2);

printf("\n --Files closed-- \n\n");

return 0; // Program executed successfully

}

input.txt

H^PROGA^000000^000063

D^LISTA^000040^ENDA^000054

R^LISTB^ENDB^LISTC^ENDC

T^0020^0A^171033^141033^465555^678909^568787^345678

T^0054^0B^181340

M^0023^05^+^LISTB

M^0050^06^-^LISTB

E^0020

H^PROGB^000000^00007F

D^LISTB^000030^ENDB^000044

R^LISTA^ENDA^LISTC^ENDC

T^0021^0A^171033^169033^069420^678909^568787^345678

T^0055^0B^181340

M^0023^05^+^LISTA

M^0050^06^-^LISTA

E

H^PROGC^000000^000069

D^LISTC^000020^ENDC^000034

R^LISTA^ENDA^LISTB^ENDB

T^0022^0A^171033^169033^089095^678909^568787^345678

T^0056^0B^181340

M^0023^05^+^LISTB

M^0050^06^-^LISTB

E

END

ESTAB

PROGA 2000 0063

LISTA 2040

ENDA 2054

PROGB 2063 007F

LISTB 2093

ENDB 20A7

PROGC 20E2 0069

LISTC 2102

ENDC 2116