PROGRAM CODE

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

typedef struct map {
	char* key;
	char* value;
} map;

map op[26];
char* ps_op[4];
map SYMTAB[100];
int SYMTAB_SIZE = 0;
int LOCCTR = 0;
char name[6];
int start = 0;
int end = 0;
int length = 0;

int containsKey(map arr[], int size, char* key) {
	if(key == NULL)
		return 0;

	for(int i = 0; i < size; i++)
		if(!strcmp(arr[i].key, key))
			return 1;

	return 0;
}

void updateKey(map arr[], int size, char* key, char* value) {
	if(key == NULL)
		return;

	for(int i = 0; i < size; i++)
		if(!strcmp(arr[i].key, key))
			arr[i].value = value;
}

char* getValue(map arr[], int size, char* key) {
	if(key == NULL)
		return NULL;

	for(int i = 0; i < size; i++)
		if(!strcmp(arr[i].key, key)) {
			char* temp = strdup(arr[i].value);
			return temp;
		}

	return NULL;
}

int containsElement(char* arr[], int size, char* key) {
	if(key == NULL)
		return 0;

	for(int i = 0; i < size; i++)
		if(!strcmp(arr[i], key))
```

```
				return 1;

		return 0;
}

char* intToHex(int num) {
		char* buff = malloc(7 * sizeof(char));
		sprintf(buff, "%X", num);
		return buff;
}

int hexToInt(char* buff) {
		int num;
		sscanf(buff, "%X", &num);
		return num;
}

char* stripX(char* token) {
		char* temp = strdup(token);

		for(int i = 0; i < strlen(token) - 1; i++) {
			if(temp[i] == ',' && temp[i+1] == 'X') {
				temp[i] = '\0';
				return temp;
			}
		}

		return temp;
}

void displayPgm(char* filename) {
		FILE* fp = fopen(filename, "r");
		char buff[64];

		printf("\nThe SIC program is given below:\n\n");
		while(1) {
			fgets(buff, 64, fp);
			if(feof(fp))
				break;
			fputs(buff, stdout);
		}

		fclose(fp);
}

void displayPass1() {
		FILE* fp = fopen("intm", "r");
		char buff[64];

		printf("\nPass 1:\n\n");
		while(1) {
			fgets(buff, 64, fp);
			if(feof(fp))
				break;
			fputs(buff, stdout);
		}

		fclose(fp);

		printf("\nSymbol Table:\n\n");

		for(int i = 0; i < SYMTAB_SIZE; i++)
			printf("%s\t%s\n", SYMTAB[i].key, SYMTAB[i].value);
```

```
}

void displayPass2() {
	FILE* fp = fopen("list", "r");
	char buff[64];

	printf("\nPass 2:\n\n");
	while(1) {
		fgets(buff, 64, fp);
		if(feof(fp))
			break;
		fputs(buff, stdout);
	}

	fclose(fp);

	fp = fopen("obj", "r");

	printf("\nObject code:\n\n");
	while(1) {
		fgets(buff, 64, fp);
		if(feof(fp))
			break;
		fputs(buff, stdout);
	}

	fclose(fp);
}

void pass1(char* filename) {
	displayPgm(filename);

	FILE* pgFile = fopen(filename, "r");
	FILE* intFile = fopen("intm", "w");
	char buff[64];

	while(1) {
		fgets(buff, 64, pgFile);

		if(feof(pgFile)) {
label:
			fclose(pgFile);
			fclose(intFile);
			displayPass1();
			return;
		}

		char* token = strtok(buff, "\t");
		if(token[strlen(token) - 1] == '\n')
			token[strlen(token) - 1] = '\0';

		int index = 0;
		char* temp = "";

		if(buff[0] == '\t')
			index = 1;
		else {
			temp = malloc(strlen(token) * sizeof(char));
			strcpy(temp, token);
		}

		while(token != NULL) {
			if(index == 0) {
```

```
		if(containsKey(SYMTAB, SYMTAB_SIZE, token)) {
			printf("\nLabel \"%s\" already used. Aborted.\n", token);
			fclose(pgFile);
			fclose(intFile);
			exit(0);
		}
		else {
			SYMTAB[SYMTAB_SIZE].key = malloc(strlen(token) * sizeof(char));
			strcpy(SYMTAB[SYMTAB_SIZE].key, token);
			SYMTAB[SYMTAB_SIZE].value = intToHex(LOCCTR);
			SYMTAB_SIZE++;
		}
	}
	else if(index == 1) {
		if(!strcmp(token, "START")) {

			token = strtok(NULL, "\t");
			if(token == NULL)
				break;
			else {
				LOCCTR = hexToInt(token);
				start = LOCCTR;
			}

			strcpy(name, temp);
			updateKey(SYMTAB, SYMTAB_SIZE, temp, intToHex(LOCCTR));

		} else if(!strcmp(token, "END")) {

			token = strtok(NULL, "\t");
			if(token == NULL) {
				end = start;
				break;
			}
			else {
				if(token[strlen(token) - 1] == '\n')
					token[strlen(token) - 1] = '\0';

				char* value = getValue(SYMTAB, SYMTAB_SIZE, token);

				if(value != NULL)
					end = hexToInt(value);
				else {
					printf("\nLabel \"%s\" undefined. Aborted.\n", token);
					fclose(pgFile);
					fclose(intFile);
					exit(0);
				}
			}

			length = LOCCTR - start;
			goto label;

		} else if(containsKey(op, 26, token)) {

			char* tempToken = malloc(strlen(token) * sizeof(char));

			strcpy(tempToken, token);

			token = strtok(NULL, "\t");

			if(token == NULL) {
				char str[20];
```

```
}                       strcpy(str, intToHex(LOCCTR));
                        strcat(str, "\t");
                        strcat(str, tempToken);
                        strcat(str, "\n");
                        fputs(str, intFile);
                    } else {
                        char str[20];
                        strcpy(str, intToHex(LOCCTR));
                        strcat(str, "\t");
                        strcat(str, tempToken);
                        strcat(str, "\t");
                        strcat(str, token);
                        fputs(str, intFile);
                    }

                    LOCCTR += 3;

                } else if(containsElement(ps_op, 4, token)) {

                    char* tempToken = malloc(strlen(token) * sizeof(char));

                    strcpy(tempToken, token);

                    token = strtok(NULL, "\t");

                    char str[20];
                    strcpy(str, intToHex(LOCCTR));
                    strcat(str, "\t");
                    strcat(str, tempToken);
                    strcat(str, "\t");
                    strcat(str, token);
                    fputs(str, intFile);

                    if(!strcmp(tempToken, "WORD")) {
                        LOCCTR += 3;
                    } else if(!strcmp(tempToken, "BYTE")) {
                        if(token[0] == 'C')
                            LOCCTR += strlen(token) - 3;
                        else if(token[0] == 'X')
                            LOCCTR += (strlen(token) - 3) / 2;
                    } else if(!strcmp(tempToken, "RESW")) {
                        LOCCTR += 3 * atoi(token);
                    } else if(!strcmp(tempToken, "RESB")) {
                        LOCCTR += atoi(token);
                    }

                } else {
                    printf("\nInvalid opcode \"%s\". Aborted.", token);
                    fclose(pgFile);
                    fclose(intFile);
                    exit(0);
                }
            }

            token = strtok(NULL, "\t");
            index++;
        }
    }
}

void pass2(char* filename) {
    FILE* intFile = fopen("intm", "r");
    FILE* listFile = fopen("list", "w");
```

```
FILE* objFile = fopen("obj", "w");
char buff[64];

char* startStr = intToHex(start);

int len = strlen(startStr);
char temp[7] = "0";

for(int i = 6; i > len; i--) {
        strcat(temp, startStr);
        strcpy(startStr, temp);
        strcpy(temp, "0");
}

char* endStr = intToHex(end);

len = strlen(endStr);

for(int i = 6; i > len; i--) {
        strcat(temp, endStr);
        strcpy(endStr, temp);
        strcpy(temp, "0");
}

char* lenStr = intToHex(length);

len = strlen(lenStr);

for(int i = 6; i > len; i--) {
        strcat(temp, lenStr);
        strcpy(lenStr, temp);
        strcpy(temp, "0");
}

for(int i = strlen(name); i < 6; i++)
        strcat(name, " ");

int count = 0;
char str[100] = "H^";
char str1[100] = "";
char startRec[7] = "";
strcat(str, name);
strcat(str, "^");
strcat(str, startStr);
strcat(str, "^");
strcat(str, lenStr);
strcat(str, "\n");
fputs(str, objFile);

while(1) {
        fgets(buff, 64, intFile);

        if(feof(intFile))
                break;

        char* token = strtok(buff, "\t");
        char* addr = strdup(token);

        token = strtok(NULL, "\t");
        if(token != NULL && token[strlen(token) - 1] == '\n')
                token[strlen(token) - 1] = '\0';

        char instr[15] = "";
```

```
		char* opcode = getValue(op, 26, token);

		char* tempToken = strdup(token);

		token = strtok(NULL, "\t");
		if(token != NULL && token[strlen(token) - 1] == '\n')
			token[strlen(token) - 1] = '\0';

		if(opcode != NULL) {
			strcat(instr, opcode);

			if(token == NULL) {
				strcat(instr, "0000");
			} else if(!containsKey(SYMTAB, SYMTAB_SIZE, stripX(token))) {
				printf("\nLabel \"%s\" not declared. Aborted.", token);
				fclose(objFile);
				fclose(listFile);
				fclose(intFile);
				exit(0);
			} else if(strstr(token, ",X")) {
				strcat(instr, intToHex(hexToInt(getValue(SYMTAB, SYMTAB_SIZE,
stripX(token))) + pow(2, 15)));
			} else {
				strcat(instr, getValue(SYMTAB, SYMTAB_SIZE, token));
			}
		} else {
				if(!strcmp(tempToken, "BYTE")) {
					if(tempToken[0] == 'X') {
						for(int i = 2; i < strlen(token) - 1; i++) {
							instr[i-2] = token[i];
						}
					}
					else if(tempToken[0] == 'C') {
						for(int i = 2; i < strlen(token) - 1; i++) {
							char buff[2];
							sprintf(buff, "%X", token[i]);
							strcat(instr, buff);
						}
					}
				} else if(!strcmp(tempToken, "WORD")) {
					strcat(instr, intToHex(atoi(token)));

					len = strlen(instr);
					for(int i = 6; i > len; i--) {
						strcat(temp, instr);
						strcpy(instr, temp);
						strcpy(temp, "0");
					}
				}
		}

		strcpy(str, "");
		strcat(str, addr);
		strcat(str, "\t");
		strcat(str, instr);
		strcat(str, "\n");
		fputs(str, listFile);

		if(strlen(instr) == 0 || count >= 30) {
			if(strlen(str1) != 0) {
				char* recLen = intToHex(count);

				if(strlen(recLen) == 1) {
```

```
                    strcat(temp, recLen);
                    strcpy(recLen, temp);
                    strcpy(temp, "0");
                }

                strcpy(str, "T^");
                strcat(str, startRec);
                strcat(str, "^");
                strcat(str, recLen);
                strcat(str, str1);
                strcat(str, "\n");
                fputs(str, objFile);

                strcpy(str1, "");
                count = 0;

                strcpy(startRec, addr);

                len = strlen(startRec);
                for(int i = 6; i > len; i--) {
                    strcat(temp, startRec);
                    strcpy(startRec, temp);
                    strcpy(temp, "0");
                }
            }
        }

        if(strlen(startRec) == 0) {
            strcpy(startRec, addr);

            len = strlen(startRec);
            for(int i = 6; i > len; i--) {
                strcat(temp, startRec);
                strcpy(startRec, temp);
                strcpy(temp, "0");
            }
        }

        if(strlen(instr) != 0) {
            strcat(str1, "^");
            strcat(str1, instr);
            count += strlen(instr) / 2;
        }
    }

    strcpy(str, "E^");
    strcat(str, endStr);
    strcat(str, "\n");
    fputs(str, objFile);

    fclose(objFile);
    fclose(listFile);
    fclose(intFile);

    displayPass2();
}

void initialise() {
    op[0].key = "LDA";
    op[1].key = "LDX";
    op[2].key = "LDL";
    op[3].key = "STA";
    op[4].key = "STX";
```

```
        op[5].key = "STL";
        op[6].key = "ADD";
        op[7].key = "SUB";
        op[8].key = "MUL";
        op[9].key = "DIV";
        op[10].key = "COMP";
        op[11].key = "TIX";
        op[12].key = "JEQ";
        op[13].key = "JGT";
        op[14].key = "JLT";
        op[15].key = "J";
        op[16].key = "AND";
        op[17].key = "OR";
        op[18].key = "JSUB";
        op[19].key = "RSUB";
        op[20].key = "LDCH";
        op[21].key = "STCH";
        op[22].key = "RD";
        op[23].key = "WD";
        op[24].key = "TD";
        op[25].key = "STSW";

        op[0].value = "00";
        op[1].value = "04";
        op[2].value = "08";
        op[3].value = "0C";
        op[4].value = "10";
        op[5].value = "14";
        op[6].value = "18";
        op[7].value = "1C";
        op[8].value = "20";
        op[9].value = "24";
        op[10].value = "28";
        op[11].value = "2C";
        op[12].value = "30";
        op[13].value = "34";
        op[14].value = "38";
        op[15].value = "3C";
        op[16].value = "40";
        op[17].value = "44";
        op[18].value = "48";
        op[19].value = "4C";
        op[20].value = "50";
        op[21].value = "54";
        op[22].value = "D8";
        op[23].value = "DC";
        op[24].value = "E0";
        op[25].value = "E8";

        for(int i = 0; i < 4; i++)
                ps_op[i] = malloc(4 * sizeof(char));

        ps_op[0] = "WORD";
        ps_op[1] = "BYTE";
        ps_op[2] = "RESW";
        ps_op[3] = "RESB";
}

void main() {
        char* filename = malloc(20 * sizeof(char));

        initialise();
```

```
	printf("Enter filename of SIC program: ");
	fgets(filename, 20, stdin);

	for(int i = 0; i < 20; i++)
		if(filename[i] == '\n')
			filename[i] = '\0';

	pass1(filename);
	pass2(filename);
}
```

OUTPUT

```
Enter filename of SIC program: program

The SIC program is given below:

PG1     START   1000
        LDX     ZERO
        STX     INDEX1
        STX     INDEX2
LOOP    LDA     DATA1,X
        AND     MAX
        COMP    MAX
        JEQ     L1
        J       L2
L1      LDX     INDEX1
        LDA     DATA1,X
        LDX     INDEX2
        STA     DATA2,X
        LDA     INDEX2
        ADD     THREE
        STA     INDEX2
L2      LDA     INDEX1
        ADD     THREE
        STA     INDEX1
        LDX     INDEX1
        COMP    LENGTH
        JLT     LOOP
        RSUB
ZERO    WORD    0
THREE   WORD    3
LENGTH          WORD    15
MAX     WORD    8388608
DATA1   RESW    5
INDEX1          RESW    5
DATA2   RESW    1
INDEX2          RESW    1
        END

Pass 1:

1000    LDX     ZERO
1003    STX     INDEX1
1006    STX     INDEX2
1009    LDA     DATA1,X
100C    AND     MAX
100F    COMP    MAX
1012    JEQ     L1
```

```
1015    J       L2
1018    LDX     INDEX1
101B    LDA     DATA1,X
101E    LDX     INDEX2
1021    STA     DATA2,X
1024    LDA     INDEX2
1027    ADD     THREE
102A    STA     INDEX2
102D    LDA     INDEX1
1030    ADD     THREE
1033    STA     INDEX1
1036    LDX     INDEX1
1039    COMP    LENGTH
103C    JLT     LOOP
103F    RSUB
1042    WORD    0
1045    WORD    3
1048    WORD    15
104B    WORD    8388608
104E    RESW    5
105D    RESW    5
106C    RESW    1
106F    RESW    1
```

Symbol Table:

```
PG1     1000
LOOP    1009
L1      1018
L2      102D
ZERO    1042
THREE   1045
LENGTH          1048
MAX     104B
DATA1   104E
INDEX1          105D
DATA2   106C
INDEX2          106F
```

Pass 2:

```
1000    041042
1003    10105D
1006    10106F
1009    00904E
100C    40104B
100F    28104B
1012    301018
1015    3C102D
1018    04105D
101B    00904E
101E    04106F
1021    0C906C
1024    00106F
1027    181045
102A    0C106F
102D    00105D
1030    181045
1033    0C105D
1036    04105D
1039    281048
103C    381009
103F    4C0000
```

```
1042    000000
1045    000003
1048    00000F
104B    800000
104E
105D
106C
106F
```

Object code:

```
H^PG1   ^001000^000000
T^001000^1E^041042^10105D^10106F^00904E^40104B^28104B^301018^3C102D^04105D^00904E
T^00101E^1E^04106F^0C906C^00106F^181045^0C106F^00105D^181045^0C105D^04105D^281048
T^00103C^12^381009^4C0000^000000^000003^00000F^800000
E^001000
```