**EXPERIMENT -5**

**Pass 2 Of Two Pass Assembler**

## Aim

To implement pass 2 of a two pass assembler

**Program**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct

{

    int address;

    char label[50];

    char opcode[50];

    char operand[50];

} inter\_file;

int program\_len = 0;

struct

{

    char opcode[50];

    int value;

} optab[1000];

int optab\_length = 0;

struct

{

    char label[50];

    int value;

} symtab[1000];

int sym\_length = 0;

int searchOpcode(char \*opcode)

{

    for (int i = 0; i < optab\_length; i++)

    {

        if (strcmp(optab[i].opcode, opcode) == 0)

            return i;

    }

    return -1;

}

int searchSymtab(char \*label)

{

    for (int i = 0; i < sym\_length; i++)

    {

        if (strcmp(symtab[i].label, label) == 0)

            return i;

    }

    return -1;

}

void writeTextRecord(FILE \*temp\_ptr, FILE \*record\_ptr, int \*obj\_len, int \*starting\_addr)

{

    char str[100];

    fclose(temp\_ptr);

    fprintf(record\_ptr, "\nT^%06X^%02X", \*starting\_addr, \*obj\_len);

    temp\_ptr = fopen("op2/temp.txt", "r");

    fscanf(temp\_ptr, "%s", str);

    fprintf(record\_ptr, "%s", str);

    fclose(temp\_ptr);

}

void checkTextRecordLimit(FILE \*temp\_ptr, FILE \*record\_ptr, int \*obj\_len, int \*starting\_addr, int offset, int new\_starting\_addr, int \*obj\_staring\_addr)

{

    if (\*starting\_addr == -1)

        \*starting\_addr = new\_starting\_addr;

    if (\*obj\_staring\_addr == -1)

        \*obj\_staring\_addr = new\_starting\_addr;

    if ((\*obj\_len + offset) \* 2 > 60)

    {

        writeTextRecord(temp\_ptr, record\_ptr, obj\_len, starting\_addr);

*// Re-Initialize*

        \*obj\_len = 0;

        temp\_ptr = fopen("op2/temp.txt", "w");

        \*starting\_addr = new\_starting\_addr;

    }

}

void main()

{

    FILE \*inter\_ptr, \*record\_ptr, \*pgmlen\_ptr, \*optab\_ptr, \*symtab\_ptr, \*temp\_ptr;

    inter\_ptr = fopen("op/intermediate\_code.txt", "r");

    pgmlen\_ptr = fopen("op/program\_length.txt", "r");

    symtab\_ptr = fopen("op/symtab.txt", "r");

    optab\_ptr = fopen("input/optab.txt", "r");

    record\_ptr = fopen("op2/record.txt", "w");

    temp\_ptr = fopen("op2/temp.txt", "w");

    printf("Reading Symtab\n");

    while (fscanf(symtab\_ptr, "%s%X", symtab[sym\_length].label, &symtab[sym\_length].value) != EOF)

        sym\_length++;

    printf("Reading Optab\n");

    while (fscanf(optab\_ptr, "%s%X", optab[optab\_length].opcode, &optab[optab\_length].value) != EOF)

        optab\_length++;

    printf("Reading program length\n");

    fscanf(pgmlen\_ptr, "%X", &program\_len);

    int obj\_len = 0, starting\_addr = -1, obj\_staring\_addr = -1;

    printf("Reading intermediate file\n");

    printf("Generating record file\n");

    while (fscanf(inter\_ptr, "%X%s%s%s", &inter\_file.address, inter\_file.label, inter\_file.opcode, inter\_file.operand) != EOF)

    {

        if (strcmp(inter\_file.opcode, "START") == 0)

        {

            fprintf(record\_ptr, "H^%6.6s^%06s^%06X", inter\_file.label, inter\_file.operand, program\_len);

        }

        else if (searchOpcode(inter\_file.opcode) != -1)

        {

            checkTextRecordLimit(temp\_ptr, record\_ptr, &obj\_len, &starting\_addr, 3, inter\_file.address, &obj\_staring\_addr);

            int op\_pos = searchOpcode(inter\_file.opcode);

            int sym\_pos = searchSymtab(inter\_file.operand);

            if (sym\_pos != -1)

            {

                fprintf(temp\_ptr, "^%02X%X", optab[op\_pos].value, symtab[sym\_pos].value);

            }

*//Index Addressing*

            else if (strstr(inter\_file.operand, ",X"))

            {

                char label[40] = "";

                strncpy(label, inter\_file.operand, strlen(inter\_file.operand) - 2);

                int sym\_pos = searchSymtab(label);

                if (sym\_pos != -1)

                {

                    int value = symtab[sym\_pos].value + 0x8000;

                    fprintf(temp\_ptr, "^%02X%X", optab[op\_pos].value, value);

                }

                else

                {

                    printf("ERROR: SYMBOL NOT FOUND IN SYMTAB: %s\n", inter\_file.operand);

                    fprintf(temp\_ptr, "^%02X%04X", optab[op\_pos].value, 0x0);

                }

            }

            else

            {

                if (strcmp(inter\_file.operand, "\*\*") != 0)

                    printf("ERROR: SYMBOL NOT FOUND IN SYMTAB: %s\n", inter\_file.operand);

                fprintf(temp\_ptr, "^%02X%04X", optab[op\_pos].value, 0x0);

            }

            obj\_len += 3;

        }

        else if (strcmp(inter\_file.opcode, "BYTE") == 0)

        {

            float temp\_len = 0;

            for (int i = 0; i < strlen(inter\_file.operand); i++)

            {

                if (inter\_file.operand[i] == 'C' || inter\_file.operand[i] == 'X' || inter\_file.operand[i] == '\'')

                    continue;

                temp\_len += 1;

            }

            checkTextRecordLimit(temp\_ptr, record\_ptr, &obj\_len, &starting\_addr, temp\_len / 2, inter\_file.address, &obj\_staring\_addr);

            fprintf(temp\_ptr, "%s", "^");

            for (int i = 0; i < strlen(inter\_file.operand); i++)

            {

                if (inter\_file.operand[i] == 'C' || inter\_file.operand[i] == 'X' || inter\_file.operand[i] == '\'')

                    continue;

                if (inter\_file.operand[0] == 'X')

                    fprintf(temp\_ptr, "%C", inter\_file.operand[i]);

                else if (inter\_file.operand[0] == 'C')

                    fprintf(temp\_ptr, "%X", inter\_file.operand[i]);

            }

            obj\_len += temp\_len / 2;

        }

        else if (strcmp(inter\_file.opcode, "WORD") == 0)

        {

            checkTextRecordLimit(temp\_ptr, record\_ptr, &obj\_len, &starting\_addr, 3, inter\_file.address, &obj\_staring\_addr);

            fprintf(temp\_ptr, "^%06X", strtol(inter\_file.operand, NULL, 10));

            obj\_len += 3;

        }

    }

    if (obj\_len != 0)

        writeTextRecord(temp\_ptr, record\_ptr, &obj\_len, &starting\_addr);

    fprintf(record\_ptr, "\nE^%06X", obj\_staring\_addr);

    fclose(inter\_ptr);

    fclose(record\_ptr);

    fclose(pgmlen\_ptr);

    fclose(optab\_ptr);

    fclose(symtab\_ptr);

    fclose(temp\_ptr);

    printf("Removing temp file\n");

    remove("op2/temp.txt");

    printf("Assembler: PASS 2 done...\n");

}

**Input**

**symtab.txt**

BUFTOREC        3000

WRREC           3000

WLOOP           3003

OUTPUT          3015

ZERO            3016

BUFFER          3019

LENGTH          4019

**intermediate\_code.txt**

3000   BUFTOREC  START     3000

3000   WRREC     LDX       ZERO

3003   WLOOP     TD        OUTPUT

3006   \*\*        JEQ       WLOOP

3009   \*\*        LDCH      BUFFER,X

300C   \*\*        WD        OUTPUT

300F   \*\*        TIX       LENGTH

3012   \*\*        JLT       WLOOP

3015   OUTPUT    BYTE      X'05'

3016   ZERO      WORD      0

3019   BUFFER    RESB      4096

4019   LENGTH    RESW      1

401C   \*\*        END       WRREC

**optab.txt**

LDA 00

LDX 04

ADD 18

COMP 28

DIV 24

JEQ 30

JGT 34

JLT 38

LDCH 50

MUL 20

RD  D8

STA 0C

STCH 54

STX 10

SUB 1C

TD  E0

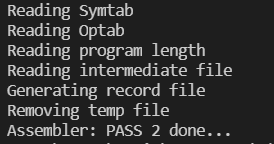
TIX 2C

WD  DC

**program\_length.txt**

101C

**Output**



**record.txt**

H^BUFTOR^003000^00101C

T^003000^19^043016^E03015^303003^50B019^DC3015^2C4019^383003^05^000000

E^003000