College Of Engineering Trivandrum

SYSTEM SOFTWARE LAB

Exercise 10: Relocating Loader Implementation

Author:
Shuhaib Ibrahim
Roll No: 60
TVE18CS061

November 28, 2020

Contents

1 Aim

Implement a relocating loader

2 Algorithm

```
step 1: Start

convert(h[12]) //function to convert bitmap in hexadecimal to binary
step 2: bit <- ""
step 3: l <- strlen(h)
step 4: i <- 0
step 5: Repeat the following while i is less than l
        i)if h[i] is equal to 0, then
            - bit <- bit + "0000"
        ii)if h[i] is equal to 1, then
            - bit <- bit + "0001"
        iii)if h[i] is equal to 2, then
            - bit <- bit + "0010"
        iv)if h[i] is equal to 3, then
            - bit <- bit + "0011"
        v)if h[i] is equal to 4, then
            - bit <- bit + "0100"
        vi)if h[i] is equal to 5, then
            - bit <- bit + "0101"
        vii)if h[i] is equal to 6, then
            - bit <- bit + "0110"
        viii)if h[i] is equal to 7, then
            - bit <- bit + "0111"
        ix)if h[i] is equal to 8, then
            - bit <- bit + "1000"
        x)if h[i] is equal to 9, then
            - bit <- bit + "1001"
        xi)if h[i] is equal to A, then
            - bit <- bit + "1010"
        xii)if h[i] is equal to B, then
            - bit <- bit + "1011"
        xiii)if h[i] is equal to C, then
            - bit <- bit + "1100"
        xiv)if h[i] is equal to D, then
            - bit <- bit + "1101"
        xv)if h[i] is equal to E, then
            - bit <- bit + "1110"
        xvi)if h[i] is equal to 1111, then
            - bit <- bit + "0000"
        xvii)i <- i+1

main
step 6: fp <- open the input file objProgram.txt in read mode
step 7: line <- Read the first line from fp // header record
step 8: Read the starting address and store it in a variable start
step 9: locctr <- start
step 10: line <- Read the next line from fp
step 11: Repeat steps 12 to 21 while line[0] is not equal to ’E’
    step 12: i <- 9, j<-0
    step 13: recLength <- substring of line from index 9 to index 10
    step 14: iRecLength <- atoi(recLength)
    step 15: bitmap <- substring of line from index 12 to index 14
    step 16: convert(bitmap)
    step 17: instr <- "" // to store instruction
    step 18: bitCounter <- 0
    step 19: i <- 16, j <- 0
    step 20: Repeat the following while i is less than iRecLength
             i)opcode <- ""
             ii)operand <- ""
             iii)if line[i] is equal to '^',
                 - j <- j + strlen(instr)/2
                 - i <- i+1
                 - bitCounter <- bitCounter + 1
            ‘iv)Else,
                 - opcode <- substring of line from index i to index i+1
                 - i <- i+2
                 - operand <- substring of line from index i to the first occurance character '^'
                   from i excluding the character '^'
                 - i <- i+length of operand
                 - if bit[bitCounter] is equal to '0', then
                    a) Print locctr,"\t",opcode,operand
                - else
                    a) iOperand <- atoi(operand)
                    b) Print locctr,"\t",opcode,iOperand+start
                - instr <- ""
                - instr <- instr + opcode
                - instr <- instr + operand
                - len <- strlen(instr)
                - locctr <- locctr + len/2
    step 21: line <- Read the next line from fp
step 22: Close the file fp
step 23: Stop
```

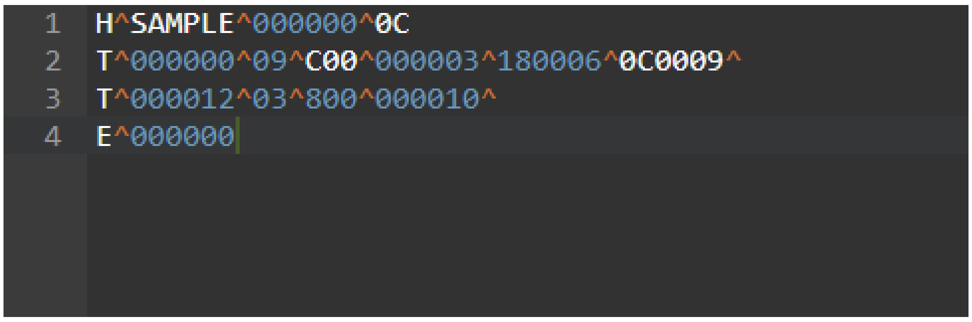
3 Program Code

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

char bit[12]; //bitmap converted to binary

void convert(char h[12])
{
    int i, l;
    strcpy(bit, "");
    l = strlen(h);
    for (i = 0; i < l; i++)
    {
        switch (h[i])
        {
            case '0':strcat(bit, "0000");
                     break;
            case '1':strcat(bit, "0001");
                     break;
            case '2':strcat(bit, "0010");
                     break;
            case '3':strcat(bit, "0011");
                     break;
            case '4':strcat(bit, "0100");
```

```
                      break;
            case '5':strcat(bit, "0101");
                      break;
            case '6':strcat(bit, "0110");
                      break;
            case '7':strcat(bit, "0111");
                      break;
            case '8':strcat(bit, "1000");
                      break;
            case '9':strcat(bit, "1001");
                      break;
            case 'A':strcat(bit, "1010");
                      break;
            case 'B':strcat(bit, "1011");
                      break;
            case 'C':strcat(bit, "1100");
                      break;
            case 'D':strcat(bit, "1101");
                      break;
            case 'E':strcat(bit, "1110");
                      break;
            case 'F':strcat(bit, "1111");
                      break;
        }
    }
}

void main()
{
    FILE * fp;

    int i, j, k;
    int start, add, locctr; //add strores strating  address of text record
    int iRecLength, iOperand, len;

    char line[50], staddr[10], recLength[2], opcode[3], operand[10], instr[20];

    char bitmap[12];//bitmap in hexadecimal

    int bitCounter;

    fp = fopen("objProgram.txt", "r");

    fscanf(fp, "%s", line);//header record

    printf("Enter the starting address : ");
    scanf("%x",&start);
    printf("\n");

    add=start;
    locctr=start;

    fscanf(fp, "%s", line);
    while(line[0]!='E')
    {
        add=locctr;

        for(i=9, j=0; i<11; i++, j++)
            recLength[j]=line[i];
        recLength[j]='\0';

        iRecLength=atoi(recLength);

        for(i=12, j=0; i<15; i++, j++)
            bitmap[j]=line[i];
        bitmap[j]='\0';

        convert(bitmap);

        strcpy(instr,"");

        bitCounter=0;
        for(i=16 ,j=0; j<iRecLength;)
        {
            strcpy(opcode,"");
            strcpy(operand,"");

            if(line[i]=='^')
            {
                j+=strlen(instr)/2;;
                i++;
                bitCounter++;
            }
            else
            {
                opcode[0]=line[i++];
                opcode[1]=line[i++];

                opcode[2]='\0';

                operand[0]=line[i++];
                operand[1]=line[i++];
                operand[2]=line[i++];
                operand[3]=line[i++];

                operand[4]='\0';

                if(bit[bitCounter]=='0')
                    printf("%x\t %s%s\n",locctr,opcode,operand);
                else
                {
                    iOperand=atoi(operand);
                    printf("%x\t %s%d\n",locctr,opcode,iOperand+start);
                }

                strcpy(instr, "");

                strcat(instr,opcode);
                strcat(instr,operand);

                len=strlen(instr);
                locctr+=(len/2);

            }
        }
```

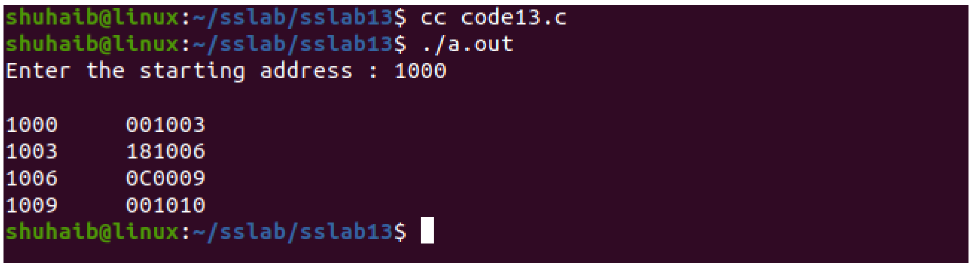
```
        fscanf(fp, "%s", line);
    }
    fclose(fp);
}
```

4 Input Files and Output



```
H^SAMPLE^000000^0C
T^000000^09^C00^000003^180006^0C0009^
T^000012^03^800^000010^
E^000000
```

Figure 1: objProgram.txt - conatins object program



```
shuhaib@linux:~/sslab/sslab13$ cc code13.c
shuhaib@linux:~/sslab/sslab13$ ./a.out
Enter the starting address : 1000

1000     001003
1003     181006
1006     0C0009
1009     001010
shuhaib@linux:~/sslab/sslab13$
```

Figure 2: Output

5 Result

Program to Implement an relocating loader was successfully implemented and output was obtained using C programming language