Department of

Computer Science & Engineering

LAB MANUAL

SYSTEM PROGRAMMING LAB

B.Tech–IV Semester

KCT College OF ENGG AND TECH.

VILLAGE FATEHGARH

DISTT.SANGRUR

BTCS 409 SYSTEM PROGRAMMING LAB

1. Create a menu driven interface for
    a) Displaying contents of a file page wise
    b) Counting vowels, characters, and lines in a file
    c) Copying a file
2. Write a program to check blance parenthesis of a given program. Also generate the error report.
3. Write a program to create symbol table for a given assembly language program.
4. Write a program to create symbol table for a given high-level language program.
5. Implementation of single pass assembler on a limited set of instructions.
6. Exploring various features of debug command.
7. Use of lax and YACC tools.

INDEX

Practical 1

Aim: Write a program to create, read and write into a file having record of students.

Theory:

Opening a file

To open a file we use fopen()

$filename='c:\file.txt':

$fp = fopen($filename."r"):

the fopen() function normally uses two parameters. the first is the file which we want to open. this can be a file on your system. in this example we are opening a text file on a windows machine. the second parameter is the "mode". we can open a file for reading and writing. in the above example "r" means reading.

Reading info from a file:

After we have opened the file, we will probably want to do something with it. to read the contents of a file into a variable we use fread().

$contents=fread($fp, filesize($filename));

fclose($fp);

Now fread() also uses two parameters. the first parameter is the variable to which we assigned the fopen() function, the second is the number of bytes we want to read up to in the file. in this case we want to read the entire file, so we use the filesize() function to get the size of the file specified in the $filename variable. thus it reads the entire file in.

Let us assume c: file.txt contains:

line1

line2

line 3

line4

line5

So now $contents would contain:

$contents="line1\nline2\nline3\nline4\nline5":

If we printed the variable out the output would be:

line1

line2

line3

line4

line5

The contents of the file are read into a string, complete with newline characters(\n. we can then process this string however we like.

Write to a file:

We can also write data into a file once we have opened it. to do this we use the fputs() function.

$filename = 'c:\file.txt':

$fp = fopen($filename,"a");

$string="\nline5";

$write=fputs($fp.$string);

fclose($fp);

firstly we open the file. notice the "a" parameter? that means "open the file for writing only, and place the file pointer at the end of the file".

we then use fputs() to write to the file. we then close the file. so now what will the file contain.

line1

line2

line3

line4

modes:

there are various modes you can use to open a file, they are listed on the php.net fopen() function page, but i'll stick them up on here too.

- 'r'-open for reading only, place the place the file pointer at the beginning of the file.
- 'r+'- open for reading and writing ; place the file pointer at the beginning of the file.
- 'w' – open for writing only, place the file pointer at the beginning of the file and truncate the file to zero length. if the file does not exist, attempt to create it.

- 'a' – open for writing only, place the file pointer at the end of the file. if the file does not exist, attempt to create it.
- 'a+' – open for reading and writing ; place the file pointer at the end of the file. if the file does not exist , attempt to create it.

VIVA VOICE:

1. What is the use of fseek() function?
2. What is the difference between getw() and getc() function?
3. What is the use of ftell() function?
4. How we are creating file?
5. What is the difference between fscanf() and fread()?
6. What is the difference between fprintf() and fwrite()?
7. What are the different techniques of file handling?

Practical 2

Aim: Write a program to count the numbers of vowels and consonants in a given string.

```
#include<stdio.h>
//#include<conio.h>
#include<string.h>
void main()
{
   char str[50];
   int vovels=0,consonants=0,len,i;
   printf("Enter the String:\n");
   gets(str);
   len=strlen(str);
   for(i=0;i<len;i++)
   {
        if((str[i]>64&&str[i]<91)||(str[i]>96&&str[i]<123))          //Firstly checking the
character is alphabet or not
 {
if(str[i]=='a'||str[i]=='A'||str[i]=='e'||str[i]=='E'||str[i]=='i'||str[i]=='I'||str[i]=='o'||str[i]=='O'||str[i]
=='u'||str[i]=='U')            //checking if it is vowel or not
  vovels++;
    else
  consonants++;
 }
   }
   printf("Number of vovels = %d and consonants = %d ",vovels,consonants);
  //getch();
}
```

O/P:

```
Enter the String:
welcome to programmingspark
Number of vovels = 8 and consonants = 17 _
```

Practical 3

Aim: Program to count the number of characters, words of a given input string.

```
#include <stdio.h>

void main()
{
  int countch=0;
  int countwd=1;

  printf("Enter your sentence in lowercase: ");
  char ch='a';
  while(ch!='\r')
  {
    ch=getche();
    if(ch==' ')
      countwd++;
    else
      countch++;
  }

  printf("\n Words = ",countwd);

  printf("Characters = ",countch-1);

  getch();

}
```

O/P: Enter your sentence in lowercase: i want help
Words=3
Characters=9

Practical 4

Aim: creation of symbol table in assembly language.

To write a c program to understand the working function of assembler in first pass creating symbol table where the table where the tables are entered in the first pass along with the corresponding address.

Algorithm:

Step 1: start the program execution.

Step 2: create a structure for opcode table and assign the values.

Step 3: create a structure for symbol table and assign the values.

Step 4: create a structure for intermediate code table and assign the values.

Step 5: write the opcode in separate file and machine code in another separate file.

Step 6: open the opcode file and compare it with the given machine code and then generate opcode for corresponding source code.

Step 7: check the forward reference in intermediate code and print the corresponding jump statement address.

Step 8: compare machine code with the opcode. If any jump statement with backward reference is present, then print backward reference address.

Step 9: for symbol table, print the symbol and address of the symbol.

Step 10: stop the program execution.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
struct table
{
char var[10];
int value;
};
struct table tbl[20];
int i,j,n;
void create();
void modify();
int search(char variable[],int n);
void insert();
void display();
void main()
{
int ch,result=0;
char v[10];
clrscr();
do
{
printf("enter your choice:\n1.create\n2.insert\n3.modify\n4.search\n5.display\n6.exit");
scanf("%d",&ch);
switch(ch)
```

```
{
case 1:
create();
break;
case 2;
insert();
break;
case 3:
modify();
break;
case 4:
printf("enter the variable to be searched\n");
scanf("%s",&v);
result=search(v,n);
if(result==0)
printf("the variable does not belong to the table\n");
else
printf("the location of variable is %d. the value of %s is
%d",result,tbl[result].var,tbl[result].value);
break;
case 5:
display();
break;
case 6;
exit(1);
}
}
```

```
while(ch!=6)
getch();
}
void create()
{
printf("enter the number of entries\n");
scanf("%d",&n);
printf("enter the variable and the value:\n");
for(i=1; i<=n; i++)
{
scanf("%s%d",tbl[i].var,&tbl[i].value);
check:
if(tbl[i].var[0]>='0' && tbl[i].var<='9')
{
printf("the variable should start with an alphabet\n enter the correct variable name \n");3
scanf("%s%d",tbl[i].var,&tbl[i].value);
goto check;
}
check1:
for (j=1; j<1; j++)
{
if(strcmp(tbl(i).var,tbl[j].var==0)
{
printf("the variable already exists. \nenter another variable\n");
scanf("%s%d",tbl[i].var,&tbl[i].value);
goto check 1;
}
```

```
}
}
printf("the table after creation is \n");
display();
}
void insert()
{
if(i>=20)
printf("can not insert. table is full");
else
{
n++;
printf("enter the variable and value \n");
scanf("%s%d",tbl[n].var,&tbl[n].value);
check:
if(tbl[i].var[0]>='0' && tbl[i].var[0]<='9')
{
printf("the variable should start with alphabet \n enter the correct variable name \n");
scanf("%s%d",tbl[i].var, & tbl[i].value);
goto check;
}
check 1:
for (j=1; j<n; j++)
{
if(strcmp(tbl[j].var,tbl[i].var)==0)
{
printf("the variable already exist \n enter another variable \n");
```

```
scanf("%s%d",tbl[i].var, &tbl[i].value);
goto check 1;
}
}
printf("the table after insertion is \n");
display();
}
}
void modify(0
{
char variable[10];
int result = 0;
printf("enter the variable to be modified \n");
scanf("%s", & variable);
result = search(variable, n);
if(result==0)
printf("%s does not belong to the table", variable);
else
{
printf("the current value of the variable %s is %d, enter the new variable and its value",
tbl[result].var, tbl[result].value);
scanf("%s%d", tbl[result].var, &tbl[result].value);check:
if(tbl[i].var[0]>='0' && tbl[i].var[0] <= '9')
{
printf("the variable should start with alphabet \n enter the correct variable name\n");
scanf("%s%d", tbl[i].var, &tbl[i].value);
goto check;
```

```
}
}
printf("the table after modification is \n");
display();
}
int search (char variable{}, int n)
{
int flag;
for (i = 1; i<=n;i++)
{
if (strcmp(tbl[i].var, variable)==0){
flag = 1;
break;
}
}
if (flag==1)
return i;
else
return 0;
}
void display()
{
printf("variable \t value \n");
for(i = 1; i<=n; i++)
printf("%s\t\t%d\n",tbl[i].var,tbl[i].value);
}
```

Output:

Enter ur choice:

1. Create
2. Insert
3. Modify
4. Search
5. Display
6. Exit

1.

Enter the number of entries

2.

Enter the variable and the value

A 26

B 42

The table after creation is

Variable	Value
A	25
B	42w

Enter ur choice:

1. Create
2. Insert
3. Modify
4. Search
5. Display
6. Exit

2.

Enter the variable and value

D 10

The table after insertion is

Variable	Value
A	26
B	42
D	10

Enter ur choice:

1. Create
2. Insert
3. Modify
4. Search
5. Display
6. Exit

3

Enter the variable to be modified

D

The current value of variable D is 10, Enter the new variable and its value

C

20

The table after modification is

Variable	Value
A	26
B	42
C	20

Enter ur choice:

1. Create
2. Insert

3. Modify

4. Search

5. Display

6. Exit

Enter the variable to be searched

A

The location of variable is 1. The value of A is 26

Enter ur choice: 1. Create

2. Insert

3. Modify

4. Search

5. Display

6. Exit

5

Variable	Value
A	26
B	42
C	20

Enter ur choice:

1. Create

2. Insert

3. Modify

4. Search

5. Display

6. Exit

Result:

Thus the program has been done and the output has been verified.

Practical 5

IMPLEMENTATION O SINGLE PASS ASSEMBLER

AIM: To write a C program to implement single pass assembler.

ALGORITHM:

Step 1: Start the program execution.

Step 2: Assembler simply generate object code as it scans the source code.

Step 3: If the instruction operand is a symbol text, has not yet been defined, the operands address is omitted.

Step 4: When nearly half the program translation is over, some of the forward reference problem are existed.

Step 5: Combine the process to the end of the program to fill forward reference property.

Step 6: At the end of the program, the symbol table entries with 'x' are undefined.

Step 7: Stop the program execution.

Program

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main
{
file *f1, *f2;
char ch, str[30], str1[10], str2[30], cstr[15];
int i, j, num, q, r;
clrscr();
printf("enter your assembly instructions \n");
f1 = fopen("asin","w");
while(1)
{
ch = getchar();
if (ch=='*')
break;
fputec(ch, f1);
}
fclose(f1);
f1 = fopen("asin", "r");
f2 = fopen("asout", "w");
while (1)
{
fgets(str ,25, f1);
strncpystr1, str, 3;
str [] = '\0';
```

```
j = 0;
for (i = 3, i<strlen(str); i++)
{
str2[j] = str[i];
j++;
}
str[j] = '\0';
if ((strcmp(str1, "1da"))==0)
{
fputs("3a\t",f2);
fputs(str2, f2);
}
else if ((strcmp(str 1, "mov")) ==0)
{
fputs("47 \n",f2);
}
else if ((strcmp(str1,"add"))==0)
{
fputs("80 \n",f2);
}
else if((strcmp(str1, "sub"))==0
{
fputs("90 \n", f2);
}
else if ((strcmp(str1, "hlt"))==)
{
fputs("76 \n", f2);
```

```
break:
)
else if((strcmp(str1, "sta"))==0)
{
fputs("32 \t", f2);
num = atoi(str2)
q = num/100;
r = num%100;
if(r==0)
fputs("00 \t",f2);
else
fputs(itoa(r,cstr,10),f2);
fputs("\t",f2);
fputs(itoa(q, cstr,10),f2);
fputs("\n",f2);
}
else
{
fputs("error\n",f2);
}
}
fclose(f1);
fclose(f2);
f2 = fopen("asout", "r");
printf("\n the object code contents \n");
ch = fgetc(f2);
while(ch! = eof)
```

```
{
putchar(ch);
ch = fgetc(f2);
}
fclose(f2);
getch();
}
```

Output:

Input

Enter your assembly instructions

Lda 5000

Sub z

Sta 9988

Hlt

Output

The object code contents

3a 00 50

90

32 88 99

76

Result

Thu the program has been done and the output has been verified.

Operator precedence parsing

Aim: Write a program to implement operator precedence parsing.

Algorithm:

Step 1: Start

Step 2: Declare the prototypes for functions.

Step 3: Enter the String like id*id + id

Step 4: Read the string and analyze tokens, identifiers, variables.

Step 5: Display the operator precedence table.

Step 6: Stop

Practical 6

Aim: For checking the operator precedence.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char *p;
e();
t();
main()
{
int i, j = 0, n b[10], k = 0;
char [10], c[10];
clrscr();
printf("enter the string \n");
scanf("%s", a);
for (i = 0, j = 0; i<strlen(); i++)
{
switch(a[i])
{
case '+':
case '-':
c[j] = a[i];
b[j] = 1;
j++;
break;
case '*':
case '/':
```

```
c[j]=  a[i];
b[j] = 2;
j++;
break;
case'^':
c[j] = a[i];
b[j] = 3;
j++;
break;
default:
if(k==0)
{
k = 1;
c[j] = a[i];
b[j] = 4;
j++;
}
}
c[j] = '$';
c[j] = 0;
j+;
printf(\n\n");
printf(\n\t-----------------------------------------------");
printf("\n\n");
for(i = 0; i<j; i++)
printf("\t%c",c[i]);
printf('\t");
```

```
for (i = 0; i<j; i++)
{
printf("\n\t-------------------------------------------------");
printf("\n\n%c",c[i]);
for(n=0; n<j;n++)
{
if(b[i]<b[n])
printf("\t<");
if(b[i>b[n])
printf("\t>");
if(b[i]==b[n])
printf("\t=");
}
printf("\t");
}
printf("\n\t-------------------------------------------------");
p = a;
if(e())
printf("\n \n string parsed");
else
printf("\n string not parsed");
getch();
return 0;
}
int e()
{
if(*p=='i')
```

```
{
p++;
t();
t();
}
else
return 0;
}
int t()
{
if(*p==nul)
return 1;
else(if*p=='+'||*p=='*')
{
p++;
if(*p=='i')
{
p++;
}
else
{
return 0;
}
}
else
return 0;
}
```

Output

Enter the string

| + | * i

---

i + * $

---

i = > > >

+ < = < >

---

* < > = >

$ < < < =

String parsed

Practical 7

Lexical Analyzer

Aim: To write a program for diving the given input program into lexemes.

Algorithm:

Step 1: Start

Step 2: Read the file and open the file as read mode.

Step 3: Read the string for token identifiers, variables.

Step 4: Take parenthesis also a token.

Step 5: Parse the string

Step 6: Stop

Practical 7

Aim: Lexical analyzer

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int i,j,k,p,c;
char s[120], r[100];
char par[6] = {'(',')','{','}','[',']'};
char sym[9] = {'.',';',':',',','<','?','$','#'};
char key[9][10]={"main","if","else","switch","void","do","while","for","return"};
char dat[4][10] = {"int","float","char","double"};
char opr[5]=:'*','+','-','/','^'};
file *fp;
clrscr();
printf("\n\n\t enter the file name");
scanf("%s",s);
fp = fopen(s,"r");
c=0;
do
{
fscanf(fp, "%s",r);
getch()]
for(i = 0; i<6: i++)
if(strchr(r,par[i])!=null)
printf("\n paranthesis: %c",path[i]);
```

```
for (i = 0; i<9; i++)
if (strchr(r, sym[i]0! = null)
printf("\n symbol: %c", sym[i]);
for(i=+;i<9;i++)
ifstrstr(r,key[i])!=null)
printf("\n keyword: %s", key[i]);
for (i=0;i<4;i++)
if((strstr(r,dat[i])&&(!strstr(r,"printf")))!=null)
{
printf("\n data type: %s", dat[i]);
fscaf(fp "%s", r);
printf("\n identifiers: %s", r);
}
for (i=0; i<5; i++)
if (strchr(r, opr[i])!=null)
printf("\n operator: %c", opr[i]);
p = c;
c = ftel(fp);
}
```