NAME: DATE VAISHNAVI BHALCHANDRA

DIV:A ROLL NO:30

SUB: SPOSL

Assignment No:1

Rainbow

PAGE:
DATE: / /

1. Pass-1 Assembler

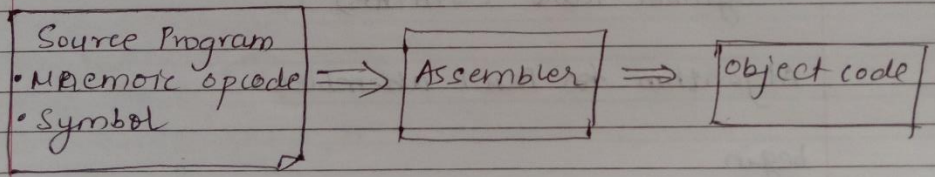
Aim: To implement Pass-1 Assembler

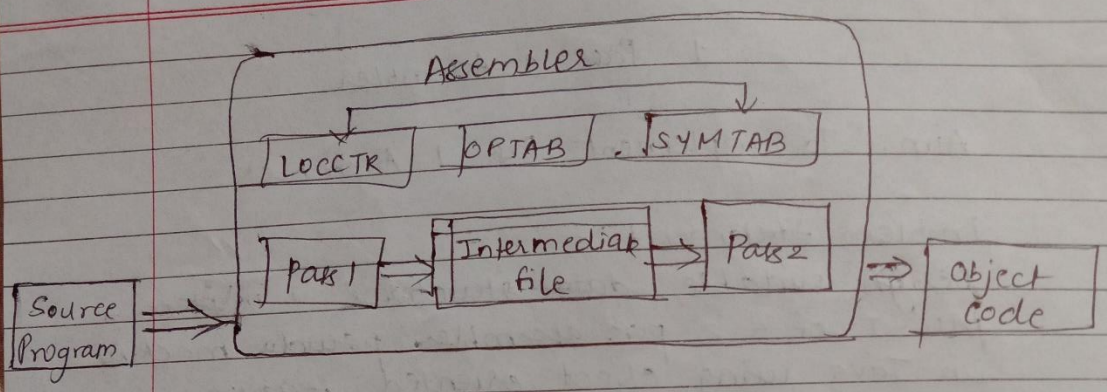
Problem Statement:-
Design suitable data structures & implement pass-I of a 2 pass assembler pseudo machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.

Theory:-

Assembly Language - It is a low-level programming language for a computer, or other programmable device, in which there is a very strong (one to one) correspondence.

Assembler :- It is converted into executable machine code by a utility program.





Assembler Directives :-

- Assembler directives are pseudo instructions
  - They will not be translated into machine instructions.
  - They only provide instruction to assembler.

Assembler directives :-

1. START
2. END
3. EQU

Three Main Data Structures

- Operation Code Table (OPTAB)
- Location Counter (LOCCTR)
- Symbol Table (SYMTAB)

Algorithm for Pass I assembler :-

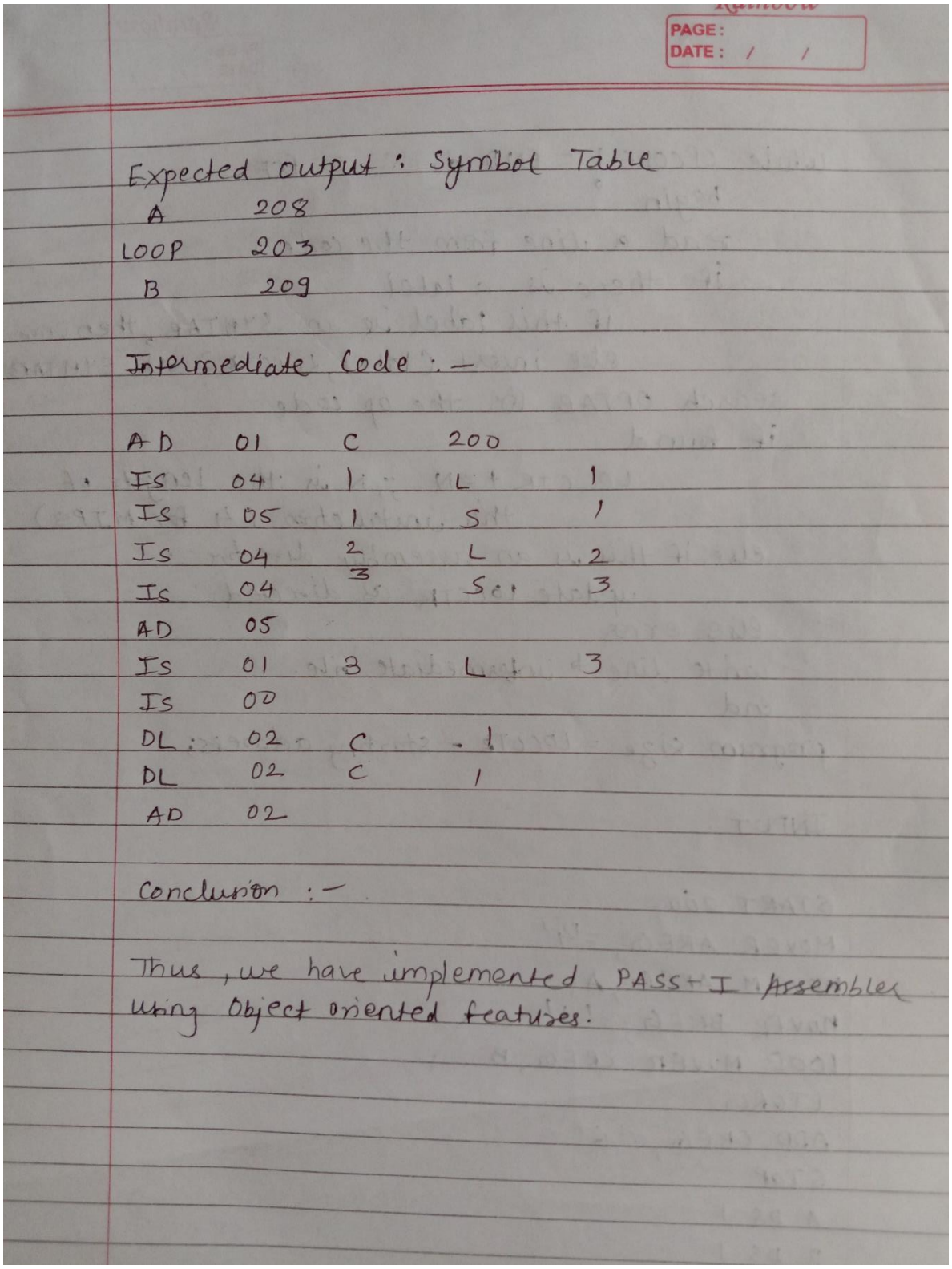
```
begin
    if starting address is given
        LOCCTR = starting address
    else
        LOCCTR = 0;
```

PAGE:
DATE: / /

while OPCODE != END do ;; or EOF
begin
read a line from the code
if there is a label
if this label is in SYMTAB, then error
else insert (label, LOCCTR) into SYMTAB
search OPTAB for the op code
if found
LOCCTR += N ;; N is the length of this instruction (4 for MIPS)
else if this is an assembly directive
update LOCCTR as directed
else error
write line to intermediate file.
end
program size = LOCCTR - starting address;

INPUT:

```
START 200
MOVER AREG, ='4'
MOVEM AREG, A
MOVER BREG, ='1'
LOOP MOVER CREG, B
LTORG
ADD CREG, ='6'
STOP
A DS 1
B DS 1
END
```



Expected output : Symbol Table

A	208
LOOP	203
B	209

Intermediate Code :-

AD	01	C	200	
Is	04	1	L	1
Is	05	1	S	1
Is	04	2	L	2
Is	04	3	S	3
AD	05			
Is	01	8	L	3
Is	00			
DL	02	C	1	
DL	02	C	1	
AD	02			

Conclusion :-

Thus, we have implemented PASS-I Assembler using Object oriented features.

Program:

1. Pass 1 Program:

```
import java.io.BufferedReader;
import java.io.*;
import java.io.IOException;
import java.util.*;
public class Pass1 { public static void main(String[] args) {
BufferedReader br = null;
 FileReader fr = null;
FileWriter fw = null;
BufferedWriter bw = null;
try {
String inputfilename = "/home/sagar-ravan/Desktop/Input.txt";
fr = new FileReader(inputfilename);
br = new BufferedReader(fr);
String OUTPUTFILENAME = "/home/sagar-ravan/Desktop/IC.txt";
fw = new FileWriter(OUTPUTFILENAME);
 bw = new BufferedWriter(fw);
Hashtable<String, String> is = new Hashtable<String, String>();
is.put("STOP", "00");
is.put("ADD", "01");
 is.put("SUB", "02");
is.put("MULT", "03");
is.put("MOVER", "04");
is.put("MOVEM", "05");
is.put("COMP", "06");
is.put("BC", "07");
is.put("DIV", "08");
is.put("READ", "09");
 is.put("PRINT", "10");
Hashtable<String, String> dl = new Hashtable<String, String>();
 dl.put("DC", "01"); dl.put("DS", "02");
Hashtable<String, String> ad = new Hashtable<String, String>();
ad.put("START", "01");

ad.put("END", "02");

ad.put("ORIGIN", "03");

ad.put("EQU", "04"); ad.put("LTORG", "05");
Hashtable<String, String> symtab = new Hashtable<String, String>();
Hashtable<String, String> littab = new Hashtable<String, String>();
ArrayList<Integer> pooltab = new ArrayList<Integer>();
String sCurrentLine; int locptr = 0; int litptr = 1; int symptr = 1; int pooltabptr = 1;
sCurrentLine = br.readLine();
String s1 = sCurrentLine.split(" ")[1];
if (s1.equals("START")) { bw.write("AD \t 01 \t");
```

```
String s2 = sCurrentLine.split(" ")[2];
 bw.write("C \t" + s2 + "\n");
locptr = Integer.parseInt(s2);
}
while ((sCurrentLine = br.readLine()) != null) { int mind_the_LC = 0; String type = null;
int flag2 = 0; // checks whether addr is assigned to current symbol
String s = sCurrentLine.split(" |\\,")[0]; // consider the first word in the
line
for (Map.Entry m : symtab.entrySet()) { // allocating addr to arrived
symbols if (s.equals(m.getKey())) {
m.setValue(locptr); flag2 = 1;
}
}
if (s.length() != 0 && flag2 == 0) { // if current string is not " " or
addr is not assigned,
// then the current string must be a new symbol.
symtab.put(s, String.valueOf(locptr));
symptr++;
}
int isOpcode = 0; // checks whether current word is an opcode or not
s = sCurrentLine.split(" |\\,")[1]; // consider the second word in the
line
for (Map.Entry m : is.entrySet()) { if (s.equals(m.getKey())) { bw.write("IS\t" +
m.getValue() + "\t"); // if match found in imperative stmt
type = "is"; isOpcode = 1;
}
}
for (Map.Entry m : ad.entrySet()) { if (s.equals(m.getKey())) { bw.write("AD\t" +
m.getValue() + "\t"); // if match
found in Assembler Directive type = "ad"; isOpcode = 1;
}
}
for (Map.Entry m : dl.entrySet()) { if (s.equals(m.getKey())) { bw.write("DL\t" +
m.getValue() + "\t"); // if match
found in declarative stmt type = "dl"; isOpcode = 1;
}
}
if (s.equals("LTORG")) { pooltab.add(pooltabptr);
for (Map.Entry m : littab.entrySet()) { if (m.getValue() == "") { // if addr is not assigned
to the
literal
m.setValue(locptr); locptr++; pooltabptr++; mind_the_LC = 1; isOpcode = 1;
}
}
```

```
}

if (s.equals("END")) { pooltab.add(pooltabptr);
for (Map.Entry m : littab.entrySet()) { if (m.getValue() == "") {
m.setValue(locptr); locptr++; mind_the_LC = 1;
}
}
}
if (s.equals("EQU")) { symtab.put("equ", String.valueOf(locptr));
}
if (sCurrentLine.split(" |\\,").length > 2) { // if there are 3 words s = sCurrentLine.split("
|\\,")[2]; // consider the 3rd word
// this is our first operand.

// it must be either a
Register/Declaration/Symbol if
(s.equals("AREG")) { bw.write("1\t");
isOpcode = 1;
} else if (s.equals("BREG")) {
bw.write("2\t"); isOpcode = 1;
} else if (s.equals("CREG")) {
bw.write("3\t"); isOpcode = 1;
} else if (s.equals("DREG")) {
bw.write("4\t"); isOpcode = 1;
} else if (type == "dl") { bw.write("C\t"
+ s + "\t");
} else { symtab.put(s, ""); // forward
referenced symbol }
}

if (sCurrentLine.split(" |\\,").length > 3) {
// if there are 4 words s =
sCurrentLine.split(" |\\,")[3]; // consider
4th word.
// this is our 2nd operand
// it is either a literal, or a symbol if
(s.contains("=")) { littab.put(s, "");
bw.write("L\t" + litptr + "\t");
isOpcode = 1; litptr++;
} else { symtab.put(s, ""); // Doubt : what
if the current symbol
is already present in SYMTAB?
// Overwrite?
```

```
b
w
.
w
r
i
t
e
(
"
S
\
t
"
+
s
y
m
p
t
r
+
"
\
t
"
)
;
s
y
```

```
mptr++;
}
}
bw.write("\n"); // done with a line.
if (mind_the_LC == 0) locptr++;
}
String f1 = "/home/sagar-
ravan/Desktop/SYMTAB.txt";
FileWriter fw1 = new FileWriter(f1);
BufferedWriter bw1 = new
BufferedWriter(fw1); for (Map.Entry m :
symtab.entrySet()) {
bw1.write(m.getKey()

+ "\t" + m.getValue() + "\n");
System.out.println(m.getKey() + " " +
m.getValue());
}
String f2 = "/home/sagar-
ravan/Desktop/LITTAB.txt";
FileWriter fw2 = new FileWriter(f2);
BufferedWriter bw2 = new
BufferedWriter(fw2); for (Map.Entry m :
littab.entrySet()) { bw2.write(m.getKey()
+ "\t" + m.getValue() + "\n");
System.out.println(m.getKey() + " " +
m.getValue());
}
String f3 = "/home/sagar-
ravan/Desktop/POOLTAB.txt";
FileWriter fw3 = new FileWriter(f3);
BufferedWriter bw3 = new
BufferedWriter(fw3);
for (Integer item : pooltab) {
bw3.write(item + "\n");
System.out.println(item);
}
bw.close(); bw1.close(); bw2.close();
bw3.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
```

Assignment No:2

PAGE:
DATE: / /

Name : Date Vaishnavi B -
Roll No - 30

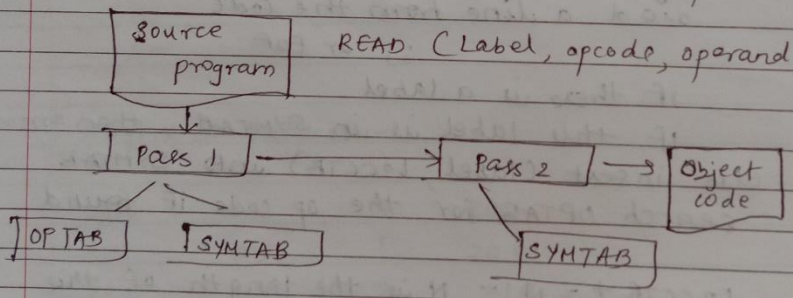
2. Pass-2 Assembler

Aim : To design data structure for Pass-2 Assembler

Problem Statement :-
Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features The output of assignment-I (intermediate file and symbol table) should be input for this assignment.

Theory :-

A Simple Two Pass Assembler Implementation



Data Structures :-

Location counter (LC) : points to the next location where the code will be placed.

DATE : / /

Op-code translation table :-

Symbol table (ST)

String storage buffer. (SSB)

Forward references table (FRT)

Algorithm :-

```
begin
    if starting address is given
        LOCCTR = starting address;
else
    LOCCTR = 0;
    while OPCODE != END do
    begin
    read a line from the code
                ;; or EOF
    if there is a label
    if this label is in SYMTAB, then error
else insert (label, LOCCTR) into SYMTAB
 search OPTAB for the op code if found

 LOCCTR += N ;; N is the length of this
 instruction (4 for MIPS)

 else if this is an assembly directive

 else error
```

PAGE:
DATE: / /

write line to intermediate file end

Program size = LOCCTR - starting address;

end.

Input :-

AD 01	C	200		
IS	04	1	L	1
IS	05	1	S	1
IS	04	2	L	2
IS	04	3	S	3
AD	05			
IS	01	3	L	3
IS	00			
DL	02	C	1	
DL	02	C	1	
AD	02			

Expected Output:-

200	04	1	204
201	05	1	208
202	04	2	210
203	04	3	209
204	00	0	004
205	00	0	006
206	01	3	205
207	00	0	000

Program:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader; import java.io.FileWriter;
import java.io.IOException; import
java.lang.reflect.Array; import
java.util.ArrayList; import
java.util.Hashtable; import java.util.Map;
public class Pass2 { public static void
main(String[] args) { try {
//1. Read Intermediate code file
 String f ="/home/sagar-ravan/Desktop/IC_new.txt";
 FileReader fw =new FileReader(f);
 BufferedReader IC_file=new BufferedReader(fw);

 //2.Read Symbol table file
 String f1="/home/sagar-ravan/Desktop/SYMTAB.txt";
 FileReader fs=new FileReader(f1);
 BufferedReader symtab_file=new BufferedReader(fs);
symtab_file.mark(500);

 //3.Read Literal table file
 String f2="/home/sagar-ravan/Desktop/LITTAB.txt";
 FileReader fl=new FileReader(f2);
 BufferedReader littab_file=new BufferedReader(fl);
littab_file.mark(500);


 //4.create littab array and hashtable for symbol table

String littab[][]=new String[10][2] ;
Hashtable<String, String> symtab = new Hashtable<String,
String>();
String str;
int z=0;
//5.Read LITTAB.txt
while ((str = littab_file.readLine()) != null) {
littab[z][0]=str.split("\\s+")[0]; //first word
```

```
littab[z][1]=str.split("\\s+")[1]; //second word z++;
}
//6.Read SYMTAB.txt
while ((str = symtab_file.readLine()) != null) {
symtab.put(str.split("\\s+")[0], str.split("\\s+")[1]); }
 //7.Read POOLTAB.txt
String f3 = "/home/sagar-ravan/Desktop/POOLTAB.txt";
FileReader fw3 = new FileReader(f3);
BufferedReader pooltab_file = new BufferedReader(fw3);
ArrayList<Integer> pooltab = new ArrayList<Integer>();
String t;
while ((t = pooltab_file.readLine()) != null) {
pooltab.add(Integer.parseInt(t));
}
int pooltabptr = 1;
int temp1 = pooltab.get(0); //dry run
int temp2 = pooltab.get(1);
//7.Read IC.txt
String sCurrentLine;
sCurrentLine = IC_file.readLine();
int locptr=0;
//locptr=Integer.parseInt(sCurrentLine.split("\\s+")[3]);
locptr=Integer.parseInt(sCurrentLine.split("\t")[3]);
while ((sCurrentLine = IC_file.readLine()) != null) {
System.out.print(locptr+"\t");
String s0 = sCurrentLine.split("\t")[0]; //contains
statement type
String s1 = sCurrentLine.split("\t")[1]; //contains
statement code
if (s0.equals("IS")) {
System.out.print(s1+"\t"); if
(sCurrentLine.split("\t").length == 5) {
System.out.print(sCurrentLine.split("\t")[2]
+ "\t");
//7.2 if third character is L
if (sCurrentLine.split("\t")[3].equals("L"))
{ int add =
Integer.parseInt(sCurrentLine.split("\t")[4]);
//machine_code_file.write(littab[add-1][1]);
System.out.print(littab[add-1][1]);
```

```
}
 //7.3 or if third character is S
if (sCurrentLine.split("\t")[3].equals("S"))
{ int add1 =
Integer.parseInt(sCurrentLine.split("\t")[4]);
//search for the 4th word in symbol
table int i = 1; String l1;
for (Map.Entry m : symtab.entrySet())
{
if (i == add1) {
System.out.print((String)
m.getValue());
}
i++;
}
}
} else {
System.out.print("0\t000");
}
}
//DRY RUN is a must
if (s0.equals("AD")) {
littab_file.reset();
if (s1.equals("05")) { //if it is
LTORG int j = 1; while (j < temp1) { littab_file.readLine();
}
while (temp1 < temp2) {
System.out.print("00\t0\t00" +
littab_file.readLine().split("'")[1]);
if(temp1<(temp2-1)){
locptr++;
System.out.println();
System.out.print(locptr+"\t");
}
temp1++;
} temp1 =
temp2;
pooltabptr++;
if (pooltabptr < pooltab.size()) {
temp2 = pooltab.get(pooltabptr); }
```

```
} int j =
1;
if (s1.equals("02")) { //if it is
"END" stmt
String s;
while ((s = littab_file.readLine()) != null)
{
if (j >= temp1)
System.out.print("00\t0\t00" +
s.split("")[1]); j++;
}
}
}
if(s0.equals("DL")&&s1.equals("01")){ //if it
is DC stmt
System.out.print("00\t0\
t00"+sCurrentLine.split("")[1]);
}
locptr++;
System.out.println();
}
IC_file.close();
symtab_file.close();
littab_file.close();
pooltab_file.close();
} catch (IOException e) {
e.printStackTrace();
}
}
```

Assignment No:3

Name: Vaishnavi Date
TE A-30
SPOSL

Rainbow
PAGE:
DATE: / /

3. Pass 1 - Macroprocessor

Aim:- To design Data Structure for Microprocessor

Problem statement:- Design suitable data structures and implement pass-I of a 2 pass macro processor using OOP features in java.

Theory:-

1. Macro processor (Definition)
A macro processor is a program that reads a file (or files) and scans them for certain keywords. When a keyword is found it is replaced by some text. The keyword/text combination is called a macro

Algorithm:- A one-pass macro processor the alternate between macro def & macro expansion algorithms.

Algorithm:-

```
begin {macro processor}
    EXPANDING := FALSE
        while OPCODE <> 'END' do
        begin
            GETLINE
            PROCESSLINE
        end {while}
end {macro procesor}
procedure PROCESSLINE
```

```
begin
    search NAMTAB for OPCODE
    if found then
        EXPAND
    else if OPCODE = 'MACRO' then
        DEFINE
    else write source line to expand file
    end {PROCESSLINE}
```

Algorithm :-

```
procedure EXPAND
    begin
    EXPANDING := TRUE
    get first line of macro def^n {prototype} from DEFTAB
set up arguments from macro invocation in ARGTAB
write macro invocation to expanded file as a comment
while not end of macro def^n do
    begin
        GETLINE
        PROCESSLINE
        end {while}
        EXPANDING := FALSE
    end {EXPAND}

procedure GETLINE
        begin
        IF EXPANDING then
begin get next line of macro def^n from DEFTAB
substitute arguments from ARGTAB for positional
notation
    end {if}
```

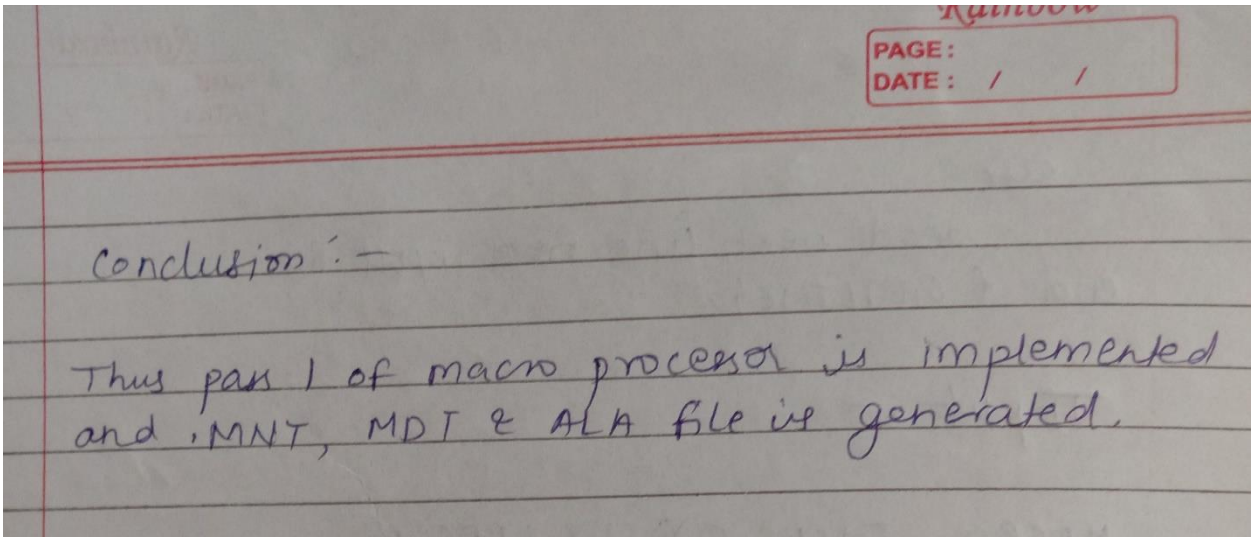
else
read next line from input file
end {GETLINE}

Input :-

```
MACRO  INCR &X &Y &REG1
       ADD  REG &Y
       MOVEM &REG1 &X
MEND
       START 100
       READ N1
       READ N2
       INCR N1 N2
       STOP
N1     DS1
N2     DS2
       END
```

C:\ABC > javac macro.java
C:\ABC > java macro

```
MACRO  INCR  &X  &Y  &REG1
  MOVER      &REG1  &X
  ADD        &REG1  &Y
  MOVEM      &REG1  &X
MEND
      START   100
      READ    N1
      READ    N2
      INCR    N1   N2
      STOP
N1    DS      1
N2    DS      2
      END
```



Program:

1. Pass 1 Macro Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
public class macroPass1 {
public static void main(String[] Args) throws IOException{
BufferedReader b1 = new BufferedReader(new FileReader("input.txt"));
FileWriter f1 = new FileWriter("intermediate.txt");
FileWriter f2 = new FileWriter("mnt.txt");
FileWriter f3 = new FileWriter("mdt.txt");
FileWriter f4 = new FileWriter("kpdt.txt");
HashMap<String,Integer> pntab=new HashMap<String,Integer>();
String s;
int paramNo=1,mdtp=1,flag=0,pp=0,kp=0,kpdtp=0;
while((s=b1.readLine())!=null){
```

```
String word[]=s.split("\\s"); //separate by space
if(word[0].compareToIgnoreCase("MACRO")==0){
flag=1;
if(word.length<=2){
f2.write(word[1]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdt
p+1))+"\n");
continue;
}
String params[]=word[2].split(",");
for(int i=0;i<params.length;i++){
if(params[i].contains("=")){
kp++;
String keywordParam[]=params[i].split("=");
pntab.put(keywordParam[0].substring(1,keywordParam[0].length()),para
mNo++);
if(keywordParam.length==2)
f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\t"+
keywordParam[1]+"\n");
else
f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\t"+
"-"+"\n");
}
else{
pntab.put(params[i].substring(1,params[i].length()),paramNo++);
pp++;
}
}
f2.write(word[1]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdt
p+1))+"\n");
kpdtp+=kp;
}
else if(word[0].compareToIgnoreCase("MEND")==0){
f3.write(s+'\n');
flag=pp=kp=0;
mdtp++;
```

```
paramNo=1;
pntab.clear();
}
else if(flag==1){
for(int i=0;i<s.length();i++){
if(s.charAt(i)=='&'){
i++;
String temp="";
while(!(s.charAt(i)==' '||s.charAt(i)==',')){
temp+=s.charAt(i++);
if(i==s.length())
break;
}
i--;
f3.write("#"+pntab.get(temp));
}
else
f3.write(s.charAt(i));
}
f3.write("\n");
mdtp++;
}
else{
f1.write(s+'\n');
}
}
b1.close();
f1.close();
f2.close();
f3.close();
f4.close();
}
}
```

Assignment No:4

DATE : / /

Vaishnovi B. Date
TE-A 30 SPOSL

4: Pass-2 Macroprocessor

Aim :- Design a MACRO PASS-2

Problem statement :- Write a Java program for pass-II of a 2-pass macro processor. The output of assignment-3 (MNT, MDT and file without any macro definitions) should be input for this assignment.

Theory :-

Basic tasks performed by Macro processor

a) Recognize macro def^n
b) Save the def^n
c) Recognize call
d) Expanded calls and substitute arguments.

- Pass I Macro Def^n
- Pass 2 Macro Calls and Expansion.

INPUT :-

```
MACRO
INCR1 &FIRST &SECOND = DATA9
 A    1&FIRST
 BL   2&SECOND
MEND  MACRO
INCR2 &ARG1, &ARG2 = DATA5
```

PAGE :
DATE : / /

```
L     3,&ARG1
ST    4,&ARG2
MEND
PRG2  START
USING *,BASE
INCR1 DATA1
INCR2 DATA3, DATA4
FOUR  DC  F4
FIVE  DC  F5
BASE  EQU 8
TEMP  DS  1F
DROP  8
END
```

Output :-

== Pass1 ==

ALA:

```
[&FIRST, &SECOND]
[&ARG1, &ARG2]
```

MNT:

```
[INCR1, 0]
[INCR2, 4]
```

MDT:

```
1       &FIRST, &SECOND = DATA9
A       1, #0
L       2, #1
MEN
```

```
D          &ARG1, &ARG2 = DATA5
INCR
 2
  L        3, #0
  ST       4, #1
  MEN
   D
```

== PASS 2 ==

MDT :

```
INCR       &FIRST, &SECOND =DATA9
 1
 A         1, #0
 L         2, #1
MEN
 D         &ARG1, &ARG2 = DATA5
INCR
 2
 L         3, #0
 ST        4, #1
MEN
 D
```

```
PRG2       STAR
             T          *, BASE
           USIN
             G
             A          1, DATA1
             L          2, DATA9
             L          3, DATA3
             ST         4, DATA4
```

FOUR	DC	F'4'
FIVE	DC	F'5'
BASE	EQU	8
TEMP	DS	1F
	DRO	
	P	
	END	

ALA:

[DATA1, DATA9]

[DATA3, DATA4]

conclusion:-

Thus pass II of Macro processor is implemented and ALA file is generated.

Program:

1. Pass 2 Macro Code:

```
import java.io.*;
import java.util.HashMap;
import java.util.Vector;
public class macroPass2 {
public static void main(String[] Args) throws IOException{
BufferedReader b1 = new BufferedReader(new
FileReader("intermediate.txt"));
BufferedReader b2 = new BufferedReader(new FileReader("mnt.txt"));
BufferedReader b3 = new BufferedReader(new FileReader("mdt.txt"));
BufferedReader b4 = new BufferedReader(new FileReader("kpdt.txt"));
FileWriter f1 = new FileWriter("Pass2.txt");
HashMap<Integer,String> aptab=new HashMap<Integer,String>();
HashMap<String,Integer> aptabInverse=new
HashMap<String,Integer>();
HashMap<String,Integer> mdtpHash=new HashMap<String,Integer>();
HashMap<String,Integer> kpdtpHash=new HashMap<String,Integer>();
HashMap<String,Integer> kpHash=new HashMap<String,Integer>();
HashMap<String,Integer> macroNameHash=new
HashMap<String,Integer>();
Vector<String>mdt=new Vector<String>();
Vector<String>kpdt=new Vector<String>();
String s,s1;
int i,pp,kp,kpdtp,mdtp,paramNo;
while((s=b3.readLine())!=null)
mdt.addElement(s);
while((s=b4.readLine())!=null)
kpdt.addElement(s);
while((s=b2.readLine())!=null){
String word[]=s.split("\t");
s1=word[0]+word[1];
```

```
macroNameHash.put(word[0],1);
kpHash.put(s1,Integer.parseInt(word[2]));

mdtpHash.put(s1,Integer.parseInt(word[3]));
kpdtpHash.put(s1,Integer.parseInt(word[4]));
}
while((s=b1.readLine())!=null){
String b1Split[]=s.split("\\s");
if(macroNameHash.containsKey(b1Split[0])){
pp= b1Split[1].split(",").length-b1Split[1].split("=").length+1;
kp=kpHash.get(b1Split[0]+Integer.toString(pp));
mdtp=mdtpHash.get(b1Split[0]+Integer.toString(pp));
kpdtp=kpdtpHash.get(b1Split[0]+Integer.toString(pp));
String actualParams[]=b1Split[1].split(",");
paramNo=1;
for(int j=0;j<pp;j++){
aptab.put(paramNo, actualParams[paramNo-1]);
aptabInverse.put(actualParams[paramNo-1],paramNo);
paramNo++;
}
i=kpdtp-1;
for(int j=0;j<kp;j++){
String temp[]=kpdt.get(i).split("\t");
aptab.put(paramNo,temp[1]);
aptabInverse.put(temp[0],paramNo);
i++;
paramNo++;
}
i=pp+1;
while(i<=actualParams.length){
String initializedParams[]=actualParams[i-1].split("=");
aptab.put(aptabInverse.get(initializedParams[0].substring(1,initializedPa
rams[0].length())),initializedParams[1
].substring(0,initializedParams[1].length()));
i++;
```

```
}
i=mdtp-1;
while(mdt.get(i).compareToIgnoreCase("MEND")!=0){
f1.write("+ ");
for(int j=0;j<mdt.get(i).length();j++){
if(mdt.get(i).charAt(j)=='#')
f1.write(aptab.get(Integer.parseInt("" +
mdt.get(i).charAt(++j))));
else
f1.write(mdt.get(i).charAt(j));
}
f1.write("\n");
i++;
}
aptab.clear();
aptabInverse.clear();
}
else
f1.write("+ "+s+"\n");
}
b1.close();
b2.close();
b3.close();
b4.close();
f1.close();
}
}
```

Assignment No:5

Vaishnavi B. Date
TE A - 30 SPOSL

5: LEX Program

Aim: Design Lex program for to generate token of given input file.

Problem Statement:-
Write a program using LEX specifications to implement lexical analysis Phase of compiler to generate tokens of subset of Java program

Pre-requisites:- LEX 110, LEX 120, LEX 130, LEX 140, LEX 160, 250

Software requirements

S.No	Facilities req	Quantity
1.	System	1
2.	O/S	Ubuntu
3.	S/W name	LEX tool (flex)

Objectives:-

1) To understand LEX concepts
2) To implement LEX Program
3) To study about LEX & JAVA
4) To know important about Lexical analyzer.

Theory :-

Regular Expression in LEX

A regular expressions is a pattern description using a Meta language. An expression is made up of symbols. Normal symbols are characters and numbers, but there are other symbols that have special meaning in LEX.

Programming in LEX :-

Programming in LEX can be divided into 3 steps :-

1) Specify the pattern - associated actions in a form
2) Run LEX over this file to generate C code for the
3) Compile and link the C code to produce the executable scanner

```
.... definitions ....
%  % rules
..... rules.....
%  %
.... subroutines...
```

Conclusion :-

Thus, we have studied lexical analyzer and implemented an applic for lexical analyzer to perform scan the program and generates token of subset of java.

Program:

1. Code b2.l:

```
%{
 FILE* yyin;
%}
DATATYPE "int"|"char"|"float"|"double"
KEYWORDS "class"|"static"
DIGIT [0-9]
NUMBER {DIGIT}+
TEXT [a-zA-Z]
IDENTIFIER {TEXT}({DIGIT}|{TEXT}|"_")*
ACCESS "public"|"private"|"protected"
CONDITIONAL "if"|"else"|"else if"|"switch"
LOOP "for"|"while"|"do"
FUNCTION
{ACCESS}{DATATYPE}{IDENTIFIER}"("({DATATYPE}{IDENTIFIER})*")
"
%%
[ \n\t]+ ;
{ DATATYPE} {printf("%s == DATATYPE\n",yytext); }
{ KEYWORDS} {printf("%s == KEYWORDS\n",yytext); }
{ NUMBER} {printf("%s == NUMBER\n",yytext); }
{ IDENTIFIER} {printf("%s == IDENTIFIER\n",yytext); }
{ CONDITIONAL} {printf("%s == CONDITIONAL\n",yytext); }
{ FUNCTION} {printf("%s == FUNCTION\n",yytext); }
. ;
```

```
%%
int yywrap(){
}
int main(int argc,char* argv[]){
yyin= fopen(argv[1],"r");
 yylex();
fclose(yyin);
 return 0;
}
```

2. Demo.java Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
public class demo
{
public static void main(String[] args)throws Exception
{ int hit=0; int miss=0;
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter total no of frames");
int noFrames=Integer.parseInt(br.readLine());
int[] frames=new int[noFrames];
int[] lruTime=new int[noFrames];
System.out.println("Enter total no of pages");
int totalPages = Integer.parseInt(br.readLine());
```

```
for(int i=0;i<totalPages;i++){
System.out.println("Enter page value");
int page= Integer.parseInt(br.readLine());
int searchIndex=isPresent(frames, page );
if(searchIndex!=-1){
// page fonud
hit++; lruTime[searchIndex]=i;
System.out.println("Page
Hit");
}
else{
System.out.println("Page Miss");
miss++;
// page not found
int emptyindex=isEmpty(frames); if(emptyindex!=-
1){
// if frame is empty
frames[emptyindex]=page;
lruTime[emptyindex]=i;
}
else{
//user lru algo to find replace location
int minLocationIndex=lru(lruTime);
System.out.println("Replace "+
frames[minLocationIndex]);
```

```
frames[minLocationIndex]=page;
lruTime[minLocationIndex]=i;
}
}
}
System.out.println("Total page hit" + hit);
System.out.println("Total Page miss " + miss);
System.out.println(Arrays.toString(frames));
}
public static int lru(int[] lruTime){ int min = 9999; int
index = -1; for(int
i=0;i<lruTime.length;i++){
if(min>lruTime[i]){
min=lruTime[i];
index=i;
}
}
return index;
}
public static int isEmpty(int[] frames){
for(int i=0;i<frames.length;i++)
{ if(frames[i]==0){
return i;
}
}
```

```
return -1;
}
public static int isPresent(int[] frames, int search){
for(int i=0;i<frames.length;i++){
if(frames[i]==search)
return i;
}
return -1;
}
}
```

Assignment No:6

Vaishnavi B. Date
TE A - 30 SPOSL

6. LEX Program

Aim: Design LEX program to count no of words lines and characters of given input

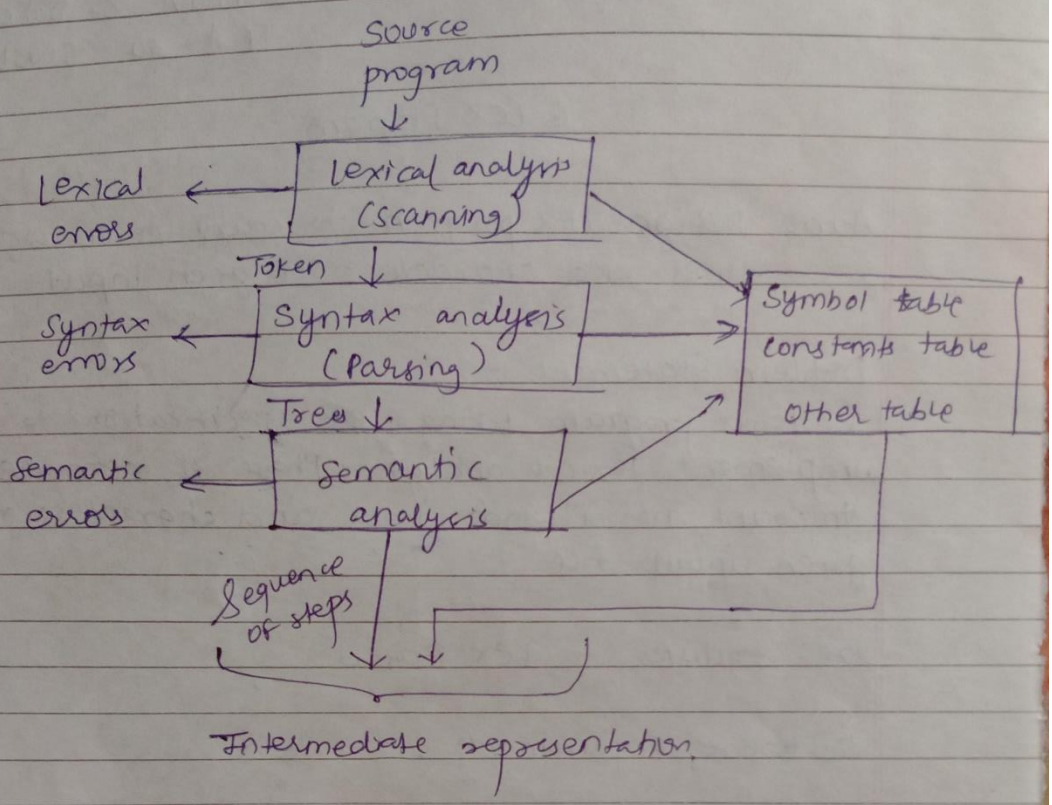
Problem Statement :-
Write a program using Lex specifications to implement lexical analysis phase of compiler to count no. of words, lines and characters of given input file

Pre-requisites :- LEX Basics

Software Requirements :-

S.No	Facilities req	Quantity
1	System	1
2	OIS	Ubuntu Kylin
3	S/W name	LEX tool (flex)

Objectives :-
1) To understand LEX concepts
2) To implement LEX program for no's of count.
3) To study about Lex & Java
4) To know important about lexical analyzer.



Conclusion :-

Thus, we have studied lexical analyzer & implemented an applic^n for lexical analyse to count total no of words, chard and line etc..

Program:

1. Code b3.l:

```
%{
int no_line=0; int
no_space=0; int
no_char=0; int
no_words=0;
#include<string.h>
%}
%%
([ a-zA-Z])+ {no_words++; no_char+=strlen(yytext); }
[ " "] {no_space++; }
[ "\n"] {no_line++; }
. ;
%%
int yywrap(){
}
int main(int argc,char* argv[]){
yyin=fopen("test.txt","r");
 yylex();
 printf("Total Spaces %d\n",no_space);
printf("Total Words %d\n",no_words);
```

```
printf("Total Line %d\n",no_line);
no_char+=no_space;
 printf("Total Char %d\n",no_char);
 fclose(yyin);
}
```

2. text.txt File:

// Content of text.txt File

The earliest foundations of what would become computer science predate the invention of the modern digital computer. Machines for calculating fixed numerical tasks such as the abacus have existed since antiquity, aiding in computations such as multiplication and division. Algorithms for performing computations have existed since antiquity, even before the development of sophisticated computing equipment.Computer science, the study of computers and computing, including their theoretical and algorithmic foundations, hardware and software, and their uses for processing information. The discipline of computer science includes the study of algorithms and data structures, computer and network design, modeling data and information processes, and artificial intelligence. Computer science draws some of its foundations from mathematics and engineering and therefore incorporates techniques from areas such as queueing theory, probability and statistics, and electronic circuit design. Computer science also makes heavy use of hypothesis testing and experimentation during the conceptualization, design, measurement, and refinement of new algorithms, information structures, and computer architectures.

Assignment No:7

7. LEX & YACC Program

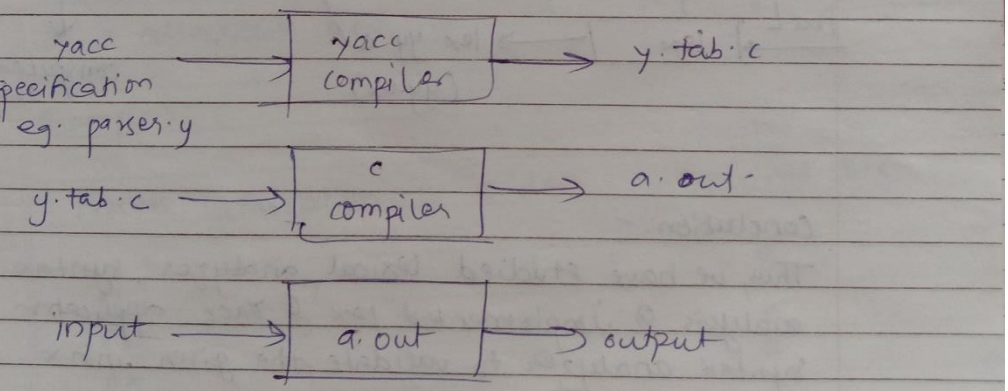
Aim :- Design Lex & Yacc program to validate type and syntax of variable declaration in Java.

Problem statement :-
Write a program using Yacc specifications to implement lexical analysis phase of compiler to validate type & syntax of variable decl in Java.

Prerequisites :- LEX 110, LEX 120, LEX 130, LEX 140, LEX 160, 2

Theory :-

YACC (Yet another compiler-compiler) is a computer program for the UNIX operating system developed by Stephen C. Johnson.



DATE : / /

Structure of a yacc file :-

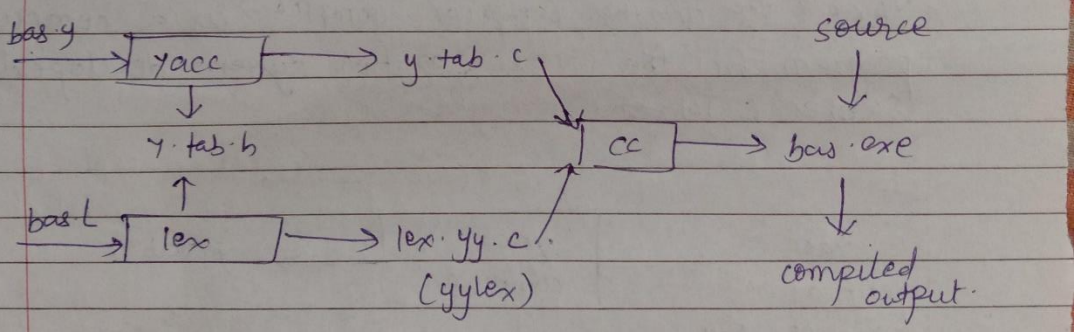
A yacc file looks much like a lex file

.... def ....
%%
.... rules ....
%%
.... code - -

Applications :-

YACC is used to generate parsers, which are an integral part of compiler.

Design (architecture) :-



Conclusion :-

Thus we have studied lexical analyzer, syntax analysis & implemented lex & Yacc application for syntax analyzer to validate the given infix expression.

Program:

1. Code b5.l:

```
%{
 #include<stdio.h>
 int simple=0;
%}
%%
[ \t\n][aA][nN][dD][ \t\n] {simple=1; }
[ \t\n][bB][uU][tT][ \t\n] {simple=1; }
[ \t\n][oO][rR][ \t\n] {simple=1; }
. ;
%%
int yywrap(){
}
int main(){
 printf("Enter sentence: \n");
yylex(); if(simple==1){
 printf("compound\n\n");
 }
else{
 printf("simple\n\n");
 }
return 0;
}
```

Assignment No:8

8. Job Scheduling Algorithm:-

Aim :- Implement Job Scheduling algorithm
1.) FCFS
2.) Shortest Job first
3.) Priority.
4.) Round Robin

Problem Statement :- Write a Java program (using OOP features) to implement following scheduling algorithm FCFS, SJF (Preemptive). Priority (Non-Preempre) and Round Robin (Preemptive)

Theory:-

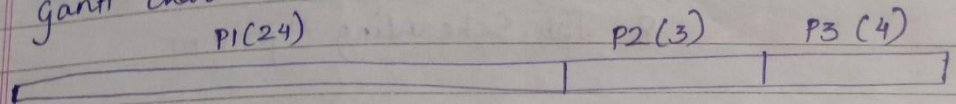
1.) First come first serve (FCFS)

This is the simplest CPU scheduling algorithm. The Process that request the CPU first, is the one which it is allocated first.

Example:-

Process	Duration	Order	Arrival time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Gantt chart



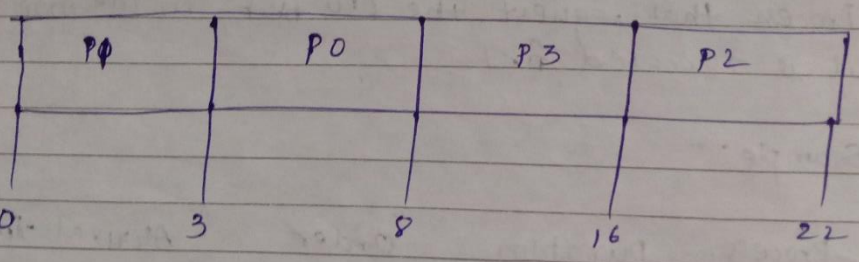
P1 waiting time : 0
P2 waiting time : 24
P3 waiting time : 27

The AWT :-
(0+24+27)/3 = 17

2) Shortest Job first :-

This algorithm associates with it the length of the next CPU burst

Process	Arrival time	Execute time	Service time
P0	0	5	0
P1	1	3	3
P2	2	8	8
P3	3	6	16



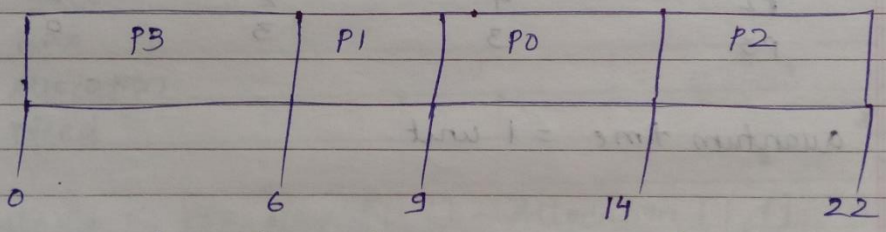
Shortest Remaining Time (SRT):-

It is the preemptive version of the SJN algorith

3) Priority Based Scheduling :-

- Priority scheduling is a non-preemptive algorithm and one of the most common algorithm.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.

Process	Arrival time	Execute time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



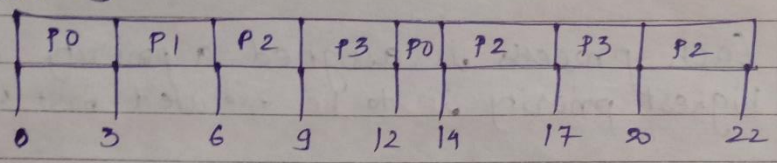
Wait time

Process	Wait time : Service - Arrival time
P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

4) Round Robin Scheduling

It is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.

Quantum = 3



Example :-

Process	Duration	order	Arrival time
P1	3	1	0
P2	4	2	0
P3	3	3	0

quantum time = 1 unit

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2

P1 W.T = 4
P2 W.T = 6
P3 W.T = 6

AWT = (4+6+6)/3 = 5.33

Program:

1. FCFS Program:

```
// Java program for implementation of FCFS
// scheduling import
java.text.ParseException; class
FCFS {
// Function to find the waiting time for all
// processes
static void findWaitingTime(int processes[], int n,
int bt[], int wt[]) {
// waiting time for first process is 0
wt[0] = 0;
// calculating waiting time for (int
i = 1; i < n; i++) { wt[i] = bt[i - 1]
+ wt[i - 1]; }
}
// Function to calculate turn around time
static void findTurnAroundTime(int processes[], int n,
int bt[], int wt[], int tat[]) {
// calculating turnaround time by adding
// bt[i] + wt[i]
for (int i = 0; i < n; i++) {
tat[i] = bt[i] + wt[i];
}
}
```

```
//Function to calculate average time
static void findavgTime(int processes[], int n, int bt[]) {
int wt[] = new int[n], tat[] = new int[n]; int
total_wt = 0, total_tat = 0;
//Function to find waiting time of all processes
findWaitingTime(processes, n, bt, wt);
//Function to find turn around time for all processes
 findTurnAroundTime(processes, n, bt, wt, tat);
//Display processes along with all details
System.out.printf("Processes \t Burst time \t Waiting" +" time Turn around time\n");
// Calculate total waiting time and total turn
// around time for (int i = 0; i < n; i++) {
total_wt = total_wt + wt[i]; total_tat =
total_tat + tat[i]; System.out.printf(" %d
", (i + 1));
System.out.printf(" %d ", bt[i]);
System.out.printf(" %d", wt[i]);
System.out.printf(" %d\n", tat[i]);
}
float s = (float)total_wt /(float) n;
int t = total_tat / n;
System.out.printf("Average waiting time = %f", s);
System.out.printf("\n");
System.out.printf("Average turn around time = %d ", t);
}
```

```
// Driver code
public static void main(String[] args) throws ParseException {
//process id's int processes[] =
{1, 2, 3,4,5}; int n =
processes.length;
//Burst time of all processes int
burst_time[] = {4,3,1,2,5};
findavgTime(processes, n, burst_time);
}
}
```

2. Shortest Job First Program:

```
import java.util.*;
public class SJF { public static void
main(String args[])
{
Scanner sc = new Scanner(System.in); System.out.println ("enter no of
process:"); int n = sc.nextInt(); int pid[] = new int[n]; int at[] = new int[n]; //
at means arrival time int bt[] = new int[n]; // bt means burst time int ct[] =
new int[n]; // ct means complete time int ta[] = new int[n]; // ta means turn
around time int wt[] = new int[n]; //wt means waiting time int f[] = new
int[n]; // f means it is flag it checks process is completed or not int st=0,
tot=0; float avgwt=0, avgta=0;
for(int i=0;i<n;i++)
{
```

```
System.out.println ("enter process " + (i+1) + " arrival time:");
at[i] = sc.nextInt();
System.out.println ("enter process " + (i+1) + " brust
time:"); bt[i] = sc.nextInt(); pid[i] = i+1; f[i] = 0;
}
boolean a = true;
while(true)
{ int c=n, min=999; if (tot == n) // total no of process = completed process loop
will
be terminated break;
for (int i=0; i<n; i++)
{
/*
* If i'th process arrival time <= system time and its flag=0 and
burst<min
* That process will be executed first
*/ if ((at[i] <= st) && (f[i] == 0) &&
(bt[i]<min))
{ min=bt[i]; c=i;
}
}
/* If c==n means c value can not updated because no process arrival time<
system time so we increase the system time */
if (c==n) st++;
else
{
```

```
ct[c]=st+bt[c];
st+=bt[c];
ta[c]=ct[c]-at[c];
wt[c]=ta[c]-
bt[c]; f[c]=1;
tot++;
}
}
System.out.println("\npid arrival brust complete turn waiting");
for(int i=0;i<n;i++)
{ avgwt+= wt[i];
avgta+= ta[i];
System.out.pri
ntln(pid[i]+"\t
"+at[i]+"\t"+bt
[i]+"\t"+ct[i]+
"\t"+ta[i]+"\
t"+wt[i]);
}
System.out.println ("\naverage tat is "+ (float)(avgta/n));
System.out.println ("average wt is "+ (float)(avgwt/n));
sc.close();
}
}
```

3. Priority Program:

```
import java.util.Scanner;
public class Priority {
public static void main(String args[]) {
Scanner s = new Scanner(System.in);
int x,n,p[],pp[],bt[],w[],t[],awt,atat,i;
p = new int[10]; pp =
new int[10]; bt = new
int[10]; w = new
int[10]; t = new int[10];
//n is number of process
//p is process
//pp is process priority
//bt is process burst time
//w is wait time
// t is turnaround time
//awt is average waiting time
//atat is average turnaround time
System.out.print("Enter the number of process : ");
n = s.nextInt();
System.out.print("\n\t Enter burst time : time priorities \n");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]:");
bt[i] = s.nextInt();
pp[i] = s.nextInt();
```

```
p[i]=i+1;
}
//sorting on the basis of priority for(i=0;i<n1;i++)
{
for(int j=i+1;j<n;j++)
{
if(pp[i]<pp[j])
{ x=pp[i];
pp[i]=pp[j];
pp[j]=x;
x=bt[i];
bt[i]=bt[j];
bt[j]=x;
x=p[i];
p[i]=p[j];
p[j]=x; }
} }
w[0]=0;
awt=0;
t[0]=bt[0];
atat=t[0];
for(i=1;i<n;i++)
{ w[i]=t[i-1];
awt+=w[i];
t[i]=w[i]+bt[i];
```

```
atat+=t[i];
}
//Displaying the process
System.out.print("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time Priority \n");
for(i=0;i<n;i++)
System.out.print("\n "+p[i]+"\t\t "+bt[i]+"\t\t "+w[i]+"\t\t"+t[i]+"\t\t "+pp[i]+"\n");
awt/=n; atat/=n;
System.out.print("\n Average Wait Time : "+awt);
System.out.print("\n Average Turn Around Time : "+atat);
}
}
```

4. Round Robin Program:

```
import java.io.*;
class RoundR {
public static void main(String args[])throws IOException
{
DataInputStream in=new DataInputStream(System.in);
int i,j,k,q,sum=0;
System.out.println("Enter number of process:");
int n=Integer.parseInt(in.readLine()); int
bt[]=new int[n]; int wt[]=new int[n]; int
tat[]=new int[n]; int a[]=new int[n];
System.out.println("Enter brust Time:");
for(i=0;i<n;i++)
{
```

```
System.out.println("Enter brust Time for "+(i+1));
bt[i]=Integer.parseInt(in.readLine());
}
System.out.println("Enter Time quantum:");
q=Integer.parseInt(in.readLine());
for(i=0;i<n;i++) a[i]=bt[i]; for(i=0;i<n;i++)
wt[i]=0; do {
for(i=0;i<n;i++)
{
if(bt[i]>q)
{
bt[i]-=q;
for(j=0;j<n;j++) {
if((j!=i)&&(bt[j]!=0))
wt[j]+=q; }
}
else {
for(j=0;j<n;j++) {
if((j!=i)&&(bt[j]!=0))
wt[j]+=bt[i];
}
bt[i]=0;
} } sum=0;
for(k=0;k<n;k++)
sum=sum+bt[k];
```

```
}
while(sum!=0);
for(i=0;i<n;i++)
tat[i]=wt[i]+a[i];
System.out.println("process\t\tBT\tWT\tTAT");
for(i=0;i<n;i++)
{
System.out.println("process"+(i+1)+"\t"+a[i]+"\t"+wt[i]+"\t"+tat[i]);
}
float avg_wt=0;
float avg_tat=0;
for(j=0;j<n;j++)
{
avg_wt+=wt[j];
}
for(j=0;j<n;j++)
{
avg_tat+=tat[j];
}
System.out.println("average waiting time"+(avg_wt/n)+"\n Average turn around
time"+(avg_tat/n));
}
}
```

Assignment No:9

9. Bankers Algorithm

Aim:- Bankers Algorithm for deadlock detection and avoidance

Problem statement:- write a Java program to implement Banker's Algorithm.

Theory :-

Following Data structures are used to implement the Banker's Algorithm:-

1. Available :-
2. Max
3. Allocation
4. Need

Need [i,j] = Max [i,j] - Allocation [i,j]

* Safety Algorithm:-

1) Let Work and finish be vectors of length 'm' & 'n' respectively.
Initialize : Work = Available
Finish [i] = false; for i = 1, 2, 3, 4.... n

2) Find an i such that both
   a) finish [i] = false
   b) Need$_i$ <= Work
   if no such i exists goto step (4)

3) Work = Work + Allocation [i]
   Finish [i] = true
   goto step (2)

4) if Finish [i] = true for all i
   then the system is in a safe state.

question 1: What will be content of Need matrix

- Need [i, j] = Max [i, j] - Allocation [i, j]

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Program:

1. Banker's Algorithm Program:

```
import java.util.Scanner; public class Bankers{
private int need[][],allocate[][],max[][],avail[][],np,nr;
private void input(){
Scanner sc=new Scanner(System.in);
System.out.print("Enter no. of processes and resources : ");
np=sc.nextInt(); //no. of process nr=sc.nextInt(); //no. of
resources need=new int[np][nr]; //initializing arrays
max=new int[np][nr]; allocate=new int[np][nr]; avail=new
int[1][nr];
System.out.println("Enter allocation matrix -->");
for(int i=0;i<np;i++) for(int j=0;j<nr;j++)
allocate[i][j]=sc.nextInt(); //allocation matrix
System.out.println("Enter max matrix -->");
for(int i=0;i<np;i++) for(int j=0;j<nr;j++)
max[i][j]=sc.nextInt(); //max matrix
System.out.println("Enter available matrix -->");
for(int j=0;j<nr;j++) avail[0][j]=sc.nextInt();
//available matrix
sc.close();
}
private int[][] calc_need(){ for(int
i=0;i<np;i++) for(int j=0;j<nr;j++)
//calculating need matrix
```

```
need[i][j]=max[i][j]-allocate[i][j];
return need; } private
boolean check(int i){
//checking if all resources for ith process can be allocated
for(int j=0;j<nr;j++) if(avail[0][j]<need[i][j]) return
false;
return true; } public void isSafe(){ input();
calc_need(); boolean done[]=new
boolean[np]; int j=0; while(j<np){ //until all
process allocated boolean allocated=false;
for(int i=0;i<np;i++) if(!done[i] &&
check(i)){ //trying to allocate for(int
k=0;k<nr;k++)
avail[0][k]=avail[0][k]-need[i][k]+max[i][k];
System.out.println("Allocated process : "+i);
allocated=done[i]=true; j++; } if(!allocated)
break; //if no allocation
} if(j==np) //if all processes are
allocated System.out.println("\nSafely
allocated"); else
System.out.println("All proceess cant be allocated safely");
}
public static void main(String[] args) {
new Bankers().isSafe();
}}
```

Assignment No:10

Assignment No - 10

Aim - Implement UNIX system calls like for process management.

Problem statement -
To write a program to implement UNIX system calls like for process management.

Pre-requisites -
1. Explain concepts of system call.
2. Explain state diagram working of new process

Software Requirements -

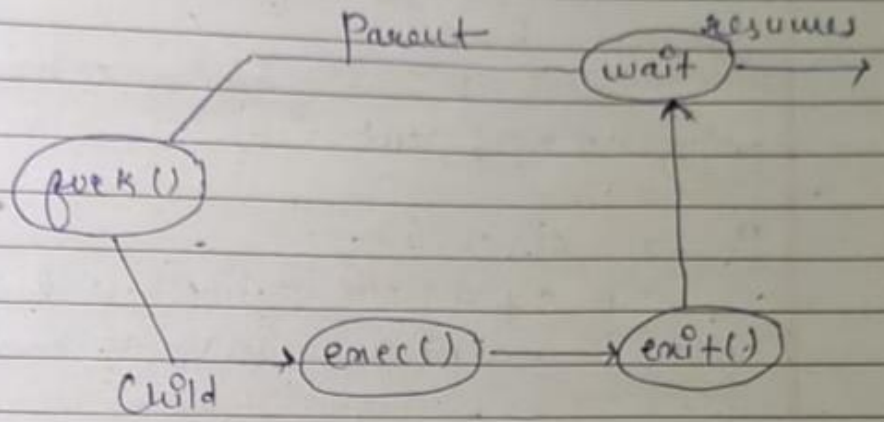
Sr No	Facilities required	Quantity
1	System	1
2	o/s	Ubuntu Kylin
3	s/w name	C Turbo or GCC

Hardware Requirements -
No

Objectives -
1. To understand UNIX system call.
2. To understand Concept of process management
3. Implementation of some system call of OS

UNIX system calls -

- PS command
- fork command.
- join command.
- Exec() command.
- Wait() command.



Conclusion -

Thus, the process system call program is implemented & studied various system calls.

Program:

1. Code:

Problem Statement : Write a C program to create a child process using fork system call. Display

Status of running processes used in child process(EXEC) & terminate child process before

completion of parent task(wait).

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{ pid_t pid , ppid , p_status ;

int status ;

printf("parent process created \n");

pid = fork();

if(pid ==0)

{

printf("child created succesfull\n");

printf("child process id : %d \n",

pid); sleep(10); printf("child after

sleep \n");

execlp("/bin/ps","ps",NULL);

printf("child terminating\n");

exit(0);

}
```

```
else
{ printf("parent still executing"); p_status
= wait(&status); printf("status :
%d \n",status); printf("p_status
:%d \n",p_status); sleep(10);
printf("parent after sleep\n");
ppid = getppid();
printf("parent process id : %d\n",ppid);
printf("parent terminating\n");
exit(0);
}
return 0;
}
```

Assignment No:12

12. Page Replacement Algorithm

Aim:- Implementing Page replacement algorithm
1) LRU
2) Optimal

Problem statement :- To write java program to implement LRU & Optimal algorithm for page Replacement

Theory:-
There are several algorithms to achieve.
1) Last Recently Used (LRU)
2) Optimal

1] LRU Page Replacement :-
If we use the recent past as an approximation of the future then we will replace the page that has not been used for the longest period of time. This approch is called as least recently. Use (LRU) algorithm.
LRU replacement associates with each page must be replaced LRU choose that page that has not been used for the longest period of time.

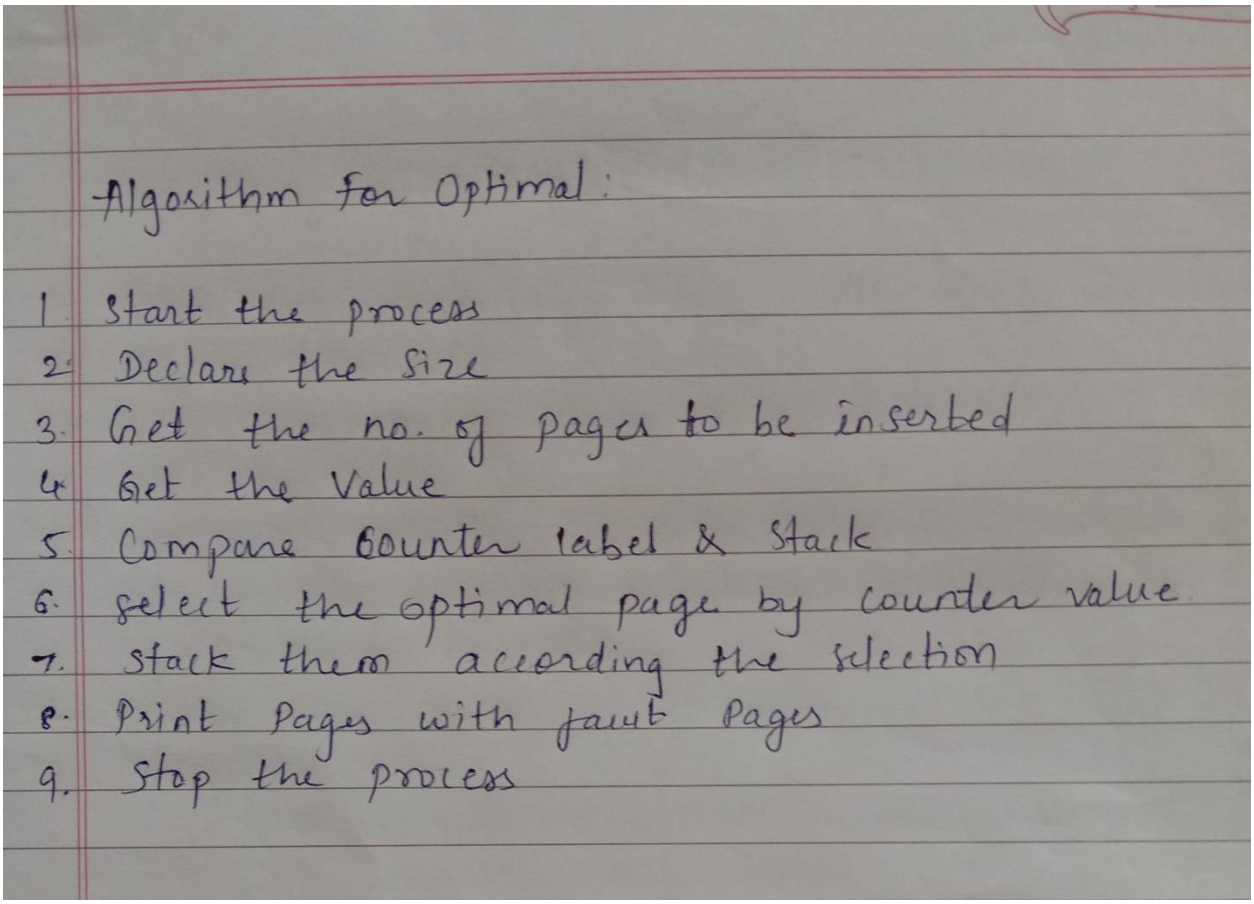
2] Optimal Page Replacement:-
The algorithm has lowest page fault rate of all algorithms. This algorithm state that: Replace the page which will not be used for longest

period of time i.e future knowledge of reference string is required.
- Often called Balady's Min Basic idea: Replace the page that will not be referenced for the longest time.
- Impossible to implement

Algorithm for LRU:-

Let capacity be the no. of pages that memory can hold. Let set be the current set of pages in memory.

1. Start traversing pages
i) If set holds less pages than capacity.
a) Insert page into the set one by one until the size of set reaches capacity or all page requests are processed.
b) Simultaneously maintain the recent occured index of each page in a map called indexes.
c) Increment page fault.
ii) Else
If current page is present in set, do nothing
Else
a) Find the page in the set that was least recently used. We find it using index array. We basically need to replace the page with minimum index.
b) Replace found page with current page.
c) Increment Page faults.
d) Update index of current Page



Program:

1. LRU (Last Recently Used) Program:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
public class LRU
{
public static void main(String[] args)throws Exception {
int hit=0;
```

```
int miss=0;
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter total no of frames");
int noFrames=Integer.parseInt(br.readLine());
int[] frames=new int[noFrames];
int[] lruTime=new int[noFrames];
System.out.println("Enter total no of pages");
int totalPages = Integer.parseInt(br.readLine());
for(int i=0;i<totalPages;i++){
System.out.println("Enter page value");
int page= Integer.parseInt(br.readLine());
int searchIndex=isPresent(frames, page );
if(searchIndex!=-1){
// page fonud
hit++; lruTime[searchIndex]=i;
System.out.println("Page
Hit");
}
else{
System.out.println("Page Miss");
miss++;
// page not found
int emptyindex=isEmpty(frames);
if(emptyindex!=-1){
```

```
// if frame is empty
frames[emptyindex]=page;
lruTime[emptyindex]=i;
}
else{
//user lru algo to find replace location int minLocationIndex=lru(lruTime);
System.out.println("Replace "+
frames[minLocationIndex]);
frames[minLocationIndex]=page;
lruTime[minLocationIndex]=i;
}
}
}
System.out.println("Total page hit" + hit);
System.out.println("Total Page miss " + miss);
System.out.println(Arrays.toString(frames));
}
public static int lru(int[] lruTime){ int min = 9999; int
index = -1; for(int
i=0;i<lruTime.length;i++){
if(min>lruTime[i]){
min=lruTime[i];
index=i;
}
}
```

```
return index;
}
public static int isEmpty(int[] frames){
for(int i=0;i<frames.length;i++)
{ if(frames[i]==0){
return i;
}
}
return -1;
}
public static int isPresent(int[] frames, int search){
for(int i=0;i<frames.length;i++){
if(frames[i]==search)
return i;
}
return -1;
}
}
```