Program: FCFS

import java.io.\*;

import java.util.Scanner;

public class FCFS {

public static void main(String[] args) {

int i,no\_p,bt[],TT[],WT[];

float avg\_wt=0,avg\_TT=0;

bt=new int[50];

TT=new int[50];

WT=new int[50];

WT[0]=0;

Scanner s=new Scanner(System.in);

System.out.println("Enter the number of processes:");

no\_p=s.nextInt();

System.out.println("Enter the burst time:");

for(i=0;i<no\_p;i++) {

System.out.println("\t P"+(i+1)+": ");

bt[i]=s.nextInt();

}

for(i=1;i<no\_p;i++) {

WT[i]=WT[i-1]+bt[i-1];

avg\_wt+=WT[i];

}

avg\_wt/=no\_p;

for(i=0;i<no\_p;i++)

{

TT[i]=WT[i]+bt[i];

avg\_TT+=TT[i];

}

avg\_TT/=no\_p;

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\tProcesses:");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Process\tBurst Time\tWaiting Time\tTurn Around Time");

for(i=0;i<no\_p;i++)

{

System.out.println("\tP"+(i+1)+"\t"+bt[i]+"\t\t "+WT[i]+"\t\t "+TT[i]);

}

System.out.println("\n----------------------------------------------------------------");

System.out.println("\nAverage waiting time : "+avg\_wt);

System.out.println("\nAverage Turn Around time :"+avg\_TT+"\n");

}

}

Output:

Enter the number of processes:

5

Enter the burst time:

P1:

2

P2:

3

P3:

5

P4:

7

P5:

11

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Processes:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Process Burst Time Waiting Time Turn Around Time

P1 2 0 2

P2 3 2 5

P3 5 5 10

P4 7 10 17

P5 11 17 28

----------------------------------------------------------------

Average waiting time : 6.8

Average Turn Around time :12.4

Program 2: Round Robin (Preemptive)

import java.util.Scanner;

public class RRP {

public static void main(String[] args) {

int n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem\_bt[];

float awt=0,atat=0;

bt=new int[50];

wt=new int[50];

tat=new int[50];

rem\_bt=new int[50];

Scanner s=new Scanner(System.in);

System.out.println("Enter the number of processes:");

n=s.nextInt();

System.out.println("Enter the burst time for each process");

for (i=0;i<n;i++)

{

System.out.print("P"+i+" = ");

bt[i] = s.nextInt();

rem\_bt[i] = bt[i];

}

System.out.print("Enter the quantum time: ");

qt = s.nextInt();

while(true)

{

for (i=0,count=0;i<n;i++)

{

temp = qt;

if(rem\_bt[i] == 0)

{

count++;

continue;

}

if(rem\_bt[i]>qt)

rem\_bt[i]= rem\_bt[i] - qt;

else

if(rem\_bt[i]>=0)

{

temp = rem\_bt[i];

rem\_bt[i] = 0;

}

sq = sq + temp;

tat[i] = sq;

}

if(n == count)

break;

}

System.out.print("--------------------------------------------------------------------------------");

System.out.print("\nProcess\t Burst Time\t Turnaround Time\t Waiting Time\n");

System.out.print("--------------------------------------------------------------------------------");

for(i=0;i<n;i++)

{

wt[i]=tat[i]-bt[i];

awt=awt+wt[i];

atat=atat+tat[i];

System.out.print("\n "+(i+1)+"\t "+bt[i]+"\t\t "+tat[i]+"\t\t"+wt[i]+"\n");

}

awt=awt/n;

atat=atat/n;

System.out.println("\nAverage waiting Time = "+awt+"\n");

System.out.println("Average turnaround time = "+atat);

}

}

Output:

Enter the number of processes:

5

Enter the burst time for each process

P0 = 1

P1 = 3

P2 = 5

P3 = 7

P4 = 9

Enter the quantum time: 3

--------------------------------------------------------------------------------

Process Burst Time Turnaround Time Waiting Time

--------------------------------------------------------------------------------

1 1 1 0

2 3 4 1

3 5 15 10

4 7 22 15

5 9 25 16

Average waiting Time = 8.4

Average turnaround time = 13.4

Program: SJF (preemptive)

import java.util.Scanner;

public class SJFP {

public static void main(String[] args) {

int bt[],process[],wt[],tat[],arr\_time[],ct[],i,j,n,total=0,total\_comp=0,pos,temp;

float wait\_avg,tat\_avg;

Scanner s=new Scanner(System.in);

System.out.print("Enter the number of processes:");

n=s.nextint();

process=new int[n];

bt=new int[n];

wt=new int[n];

tat=new int[n];

arr\_time=new int[n];

ct=new int[n];

//Burst time

System.out.println("\nEnter Burst time:");

for(i=0;i<n;i++)

{

System.out.print("\nProcess["+(i+1)+"]: ");

bt[i] = s.nextInt();;

process[i]=i+1; //Process Number

}

System.out.println("\nEnter arrival time:");

for(i=0;i<n;i++)

{

System.out.print("\nProcess["+(i+1)+"]: ");

arr\_time[i] = s.nextInt();

process[i]=i+1; //Process Number

}

//Sorting

for(i=0;i<n;i++)

{

pos=i;

for(j=i+1;j<n;j++)

{ if(bt[j]<bt[pos])

pos=j;

}

temp=bt[i];

bt[i]=burst\_time[pos];

burst\_time[pos]=temp;

temp=process[i];

process[i]=process[pos];

process[pos]=temp;

System.out.println("process"+process[i]);

}

//completion time new

for(i=1;i<n;i++)

{

ct[i]=0;

for(j=0;j<i;j++)

ct[i]+=bt[j];

total\_comp+=ct[i];

}

//First process has 0 waiting time

waiting\_time[0]=0;

//calculate waiting time

for(i=1;i<n;i++)

{

waiting\_time[i]=0;

for(j=0;j<i;j++)

waiting\_time[i]+=burst\_time[j];

total+=waiting\_time[i];

}

//Calculating Average waiting time

wait\_avg=(float)total/n;

total=0;

System.out.println("\nPro\_number\t Burst Time \tcompletion\_time\tWaiting Time\tTurnaround Time");

for(i=0;i<n;i++) {

tat[i]=burst\_time[i]+waiting\_time[i];

total+=tat[i];

System.out.println("\n"+process[i]+"\t\t "+burst\_time[i]+"\t\t"+completion\_time[i]+"\t\t"+waiting\_time[i]+"\t\t "+tat[i]);

}

//Calculation of Average Turnaround Time

TAT\_avg=(float)total/n;

System.out.println("\n\nAWT: "+wait\_avg);

System.out.println("\nATAT: "+TAT\_avg);

}

}

Program: SJF Non Preemptive

import java.util.Scanner;

class SJFNP {

int pid, burstTime, arrivalTime, waitingTime, turnaroundTime;

public SJFNProcess(int pid, int burstTime, int arrivalTime) {

this.pid = pid;

this.burstTime = burstTime;

this.arrivalTime = arrivalTime;

}

}

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of processes: ");

int n = sc.nextInt();

SJFNProcess[] processes = new SJFNProcess[n];

for (int i = 0; i < n; i++) {

System.out.print("Enter arrival time and burst time for process " + (i + 1) + ": ");

int at = sc.nextInt();

int bt = sc.nextInt();

processes[i] = new SJFNProcess(i + 1, bt, at);

}

// Sort processes by arrival time and burst time

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (processes[j].burstTime > processes[j + 1].burstTime) {

SJFNProcess temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

int totalTime = 0, totalWT = 0, totalTAT = 0;

for (SJFNProcess p : processes) {

p.waitingTime = totalTime - p.arrivalTime;

totalTime += p.burstTime;

p.turnaroundTime = p.waitingTime + p.burstTime;

totalWT += p.waitingTime;

totalTAT += p.turnaroundTime;

}

// Print Gantt Chart

System.out.println("Gantt Chart: ");

for (SJFNProcess p : processes) {

System.out.print("P" + p.pid + " ");

}

System.out.println("\n");

// Print process details

System.out.println("Process\tArrival\tBurst\tWaiting\tTurnaround");

for (SJFNProcess p : processes) {

System.out.println("P" + p.pid + "\t" + p.arrivalTime + "\t" + p.burstTime + "\t" + p.waitingTime + "\t" + p.turnaroundTime);

}

System.out.println("Average Waiting Time: " + (totalWT / (float) n));

System.out.println("Average Turnaround Time: " + (totalTAT / (float) n));

sc.close();

}

}

Output:

Enter number of processes: 5

Enter arrival time and burst time for process 1: 7 4

Enter arrival time and burst time for process 2: 5 3

Enter arrival time and burst time for process 3: 2 7

Enter arrival time and burst time for process 4: 3 5

Enter arrival time and burst time for process 5: 2 4

Gantt Chart:

P2 P1 P5 P4 P3

Process Arrival Burst Waiting Turnaround

P2 5 3 -5 -2

P1 7 4 -4 0

P5 2 4 5 9

P4 3 5 8 13

P3 2 7 14 21

Average Waiting Time: 3.6

Average Turnaround Time: 8.2

Program: Priortiy (Non preemptive)

import java.util.Scanner;

public class PNP {

public static void main(String args[]) {

Scanner s = new Scanner(System.in);

int x,n,p[],pp[],bt[],w[],t[],i;

float awt,atat;

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 CPU time---priority \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<n-1;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.println("-----------------------------------------------------------------------");

System.out.print("\n\nProcess \t\t |Burst Time \t\t |Wait Time \t\t Turn Time\t\t priority\n");

System.out.println("-----------------------------------------------------------------------");

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");

System.out.println("-----------------------------------------------------------------------");

awt/=n;

atat/=n;

System.out.print("\n Average Wait Time : "+awt);

System.out.print("\n Average Turn Around Time : "+atat);

}

}

Output:

Enter the number of process : 5

Enter CPU time---priority

Process[1]:2

3

Process[2]:4

5

Process[3]:6

7

Process[4]:8

9

Process[5]:10

11

-----------------------------------------------------------------------

Process |Burst Time |Wait Time Turn Time priority

-----------------------------------------------------------------------

5 | 10 | 0 |10 | 11

4 | 8 | 10 |18 | 9

3 | 6 | 18 |24 | 7

2 | 4 | 24 |28 | 5

1 | 2 | 28 |30 | 3

-----------------------------------------------------------------------

Average Wait Time : 16.0

Average Turn Around Time : 22.0

Program: Priority Preemptive

import java.util.Scanner;

public class Main {

static class Process {

int id, burst, priority, arrival, waitingTime, turnaroundTime;

int remainingBurst;

public Process(int id, int burst, int priority, int arrival) {

this.id = id;

this.burst = burst;

this.priority = priority;

this.arrival = arrival;

this.remainingBurst = burst;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of processes: ");

int n = sc.nextInt();

Process[] processes = new Process[n];

for (int i = 0; i < n; i++) {

System.out.print("Enter arrival time, burst time, and priority for process " + (i + 1) + ": ");

int at = sc.nextInt();

int bt = sc.nextInt();

int priority = sc.nextInt();

processes[i] = new Process(i + 1, bt, priority, at);

}

int completed = 0, time = 0;

String ganttChart = "";

while (completed < n) {

int idx = -1;

int highestPriority = Integer.MAX\_VALUE;

for (int i = 0; i < n; i++) {

if (processes[i].arrival <= time && processes[i].remainingBurst > 0 && processes[i].priority < highestPriority) {

highestPriority = processes[i].priority;

idx = i;

}

}

if (idx != -1) {

processes[idx].remainingBurst--;

ganttChart += "P" + processes[idx].id + " ";

if (processes[idx].remainingBurst == 0) {

completed++;

processes[idx].turnaroundTime = time + 1 - processes[idx].arrival;

processes[idx].waitingTime = processes[idx].turnaroundTime - processes[idx].burst;

}

} else {

ganttChart += "idle ";

}

time++;

}

// Print Gantt Chart

System.out.println("Gantt Chart: " + ganttChart);

// Calculate and print average waiting time and turnaround time

int totalWT = 0, totalTAT = 0;

System.out.println("Process\tArrival\tBurst\tPriority\tWaiting\tTurnaround");

for (Process p : processes) {

totalWT += p.waitingTime;

totalTAT += p.turnaroundTime;

System.out.println("P" + p.id + "\t" + p.arrival + "\t" + p.burst + "\t" + p.priority + "\t\t" + p.waitingTime + "\t" + p.turnaroundTime);

}

System.out.println("Average Waiting Time: " + (totalWT / (float) n));

System.out.println("Average Turnaround Time: " + (totalTAT / (float) n));

sc.close();

}

}

Output:

Enter number of processes: 5

Enter arrival time, burst time, and priority for process 1: 2 4 5

Enter arrival time, burst time, and priority for process 2: 3 6 4

Enter arrival time, burst time, and priority for process 3: 4 8 6

Enter arrival time, burst time, and priority for process 4: 5 10 3

Enter arrival time, burst time, and priority for process 5: 6 7 2

Gantt Chart: idle idle P1 P2 P2 P4 P5 P5 P5 P5 P5 P5 P5 P4 P4 P4 P4 P4 P4 P4 P4 P4 P2 P2 P2 P2 P1 P1 P1 P3 P3 P3 P3 P3 P3 P3 P3

Process Arrival Burst Priority Waiting Turnaround

P1 2 4 5 23 27

P2 3 6 4 17 23

P3 4 8 6 25 33

P4 5 10 3 7 17

P5 6 7 2 0 7

Average Waiting Time: 14.4

Average Turnaround Time: 21.4

Program: FIFO

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of frames: ");

int numberOfFrames = scanner.nextInt();

System.out.print("Enter the number of pages: ");

int numberOfPages = scanner.nextInt();

System.out.print("Enter the page reference string (space-separated): ");

int[] pageReferenceString = new int[numberOfPages];

for (int i = 0; i < numberOfPages; i++) {

pageReferenceString[i] = scanner.nextInt();

}

int[] frames = new int[numberOfFrames];

Arrays.fill(frames, -1);

int pageFaults = 0;

int currentIndex = 0;

for (int page : pageReferenceString) {

boolean pageHit = false;

for (int frame : frames) {

if (frame == page) {

pageHit = true;

break;

}

}

if (!pageHit) {

frames[currentIndex] = page;

currentIndex = (currentIndex + 1) % numberOfFrames;

pageFaults++;

}

System.out.print("Frames: ");

for (int frame : frames) {

System.out.print(frame + " ");

}

System.out.println();

}

System.out.println("Total Page Faults: " + pageFaults);

System.out.println(" Page Faults ratio: " + pageFaults +":"+numberOfPages);

scanner.close();

}

}

Output:

Enter the number of frames: 3

Enter the number of pages: 7

Enter the page reference string (space-separated): 1 2 3 2 1 4 5

Frames: 1 -1 -1

Frames: 1 2 -1

Frames: 1 2 3

Frames: 1 2 3

Frames: 1 2 3

Frames: 4 2 3

Frames: 4 5 3

Total Page Faults: 5

Page Faults ratio: 5:7

Program: Optimal Page Replacement Algorithm

import java.util.\*;

public class OPRA {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Get inputs for pages and number of frames

System.out.println("Enter number of frames: ");

int frames = sc.nextInt();

System.out.println("Enter number of pages: ");

int pagesCount = sc.nextInt();

int[] pages = new int[pagesCount];

System.out.println("Enter the page reference string: ");

for (int i = 0; i < pagesCount; i++) {

pages[i] = sc.nextInt();

}

// Initialize frames and other variables

int[] frameArray = new int[frames];

Arrays.fill(frameArray, -1);

int pageFaults = 0;

for (int i = 0; i < pagesCount; i++) {

int currentPage = pages[i];

if (!isInFrames(frameArray, currentPage)) {

pageFaults++;

int replaceIndex = findOptimalIndex(frameArray, pages, i + 1);

frameArray[replaceIndex] = currentPage;

}

// Print the current state of frames

System.out.print("Frames after inserting page " + currentPage + ": ");

printFrames(frameArray);

}

System.out.println("Total page faults: " + pageFaults);

}

// Check if a page is already in frames

private static boolean isInFrames(int[] frames, int page) {

for (int frame : frames) {

if (frame == page) {

return true;

}

}

return false;

}

// Find the optimal index for replacement

private static int findOptimalIndex(int[] frames, int[] pages, int start) {

int farthest = start;

int index = -1;

for (int i = 0; i < frames.length; i++) {

int j;

for (j = start; j < pages.length; j++) {

if (frames[i] == pages[j]) {

if (j > farthest) {

farthest = j;

index = i;

}

break;

}

}

if (j == pages.length) {

return i; // Page not found in the future, replace this frame

}

}

return (index == -1) ? 0 : index;

}

// Print the frames

private static void printFrames(int[] frames) {

for (int frame : frames) {

if (frame == -1) {

System.out.print("[ ] ");

} else {

System.out.print("[" + frame + "] ");

}

}

System.out.println();

}

}

Output:

Enter number of frames:

5

Enter number of pages:

8

Enter the page reference string:

1 2 3 2 1 4 5 6

Frames after inserting page 1: [1] [ ] [ ] [ ] [ ]

Frames after inserting page 2: [1] [2] [ ] [ ] [ ]

Frames after inserting page 3: [1] [2] [3] [ ] [ ]

Frames after inserting page 2: [1] [2] [3] [ ] [ ]

Frames after inserting page 1: [1] [2] [3] [ ] [ ]

Frames after inserting page 4: [4] [2] [3] [ ] [ ]

Frames after inserting page 5: [5] [2] [3] [ ] [ ]

Frames after inserting page 6: [6] [2] [3] [ ] [ ]

Total page faults: 6

Program: LRU

import java.util.\*;

public class Main {

public static void main(String args[]) {

Scanner sc= new Scanner(System.in);

System.out.println("Enter the number of frames:");

int nof=sc.nextInt();

System.out.println("Enter the number of pages:");

int nop=sc.nextInt();

System.out.println("Enter the page referenced string:");

int[] prs=new int[nop];

for (int i=0;i<nop;i++) {

prs[i]=sc.nextInt();

}

LinkedList<Integer> frames=new LinkedList<>();

int pagefaults=0;

for (int page : prs) {

if(!frames.contains(page)) {

if(frames.size()>=nof) {

frames.removeFirst();

}

frames.addLast(page);

pagefaults++;

}

else {

frames.remove(frames.indexOf(page));

frames.addLast(page);

}

System.out.println("Frames:");

for(int frame:frames){

System.out.println(frame+"");

}

System.out.println();

}

System.out.println("Total Page Faults:"+pagefaults);

sc.close();

}

}

Output:

Enter the number of frames:

5

Enter the number of pages:

15

Enter the page referenced string:

7 0 1 2 3 1 0 3 0 4 2 3 0 3 1

Frames:

7

Frames:

7

0

Frames:

7

0

1

Frames:

7

0

1

2

Frames:

7

0

1

2

3

Frames:

7

0

2

3

1

Frames:

7

2

3

1

0

Frames:

7

2

1

0

3

Frames:

7

2

1

3

0

Frames:

2

1

3

0

4

Frames:

1

3

0

4

2

Frames:

1

0

4

2

3

Frames:

1

4

2

3

0

Frames:

1

4

2

0

3

Frames:

4

2

0

3

1

Total Page Faults:6

Program Pass 1 Macro

import java.util.\*;

import java.io.\*;

class Pass1 {

static string mnt[]=new string[5][3];

static string ala[]=new string[10][2];

static string mdt[]=new string[20][1];

static int mntc=0,mdtc=0,alac=0;

public static void main(String args[]){

pass1();

System.out.println("Macro Name Table(MNT)");

display(mnt,mntc,3);

System.out.println("ALA for pass1");

display(ala,alac,2);

System.out.println("Macro Definition Table(MDT)");

display(mdt,mdtc,1);

}

static void pass1(){

int index=0,i;

String s,prev=”'',substring;

try {

BufferedReader inp=new BufferedReader(new FileReader('input.txt'));

while((s=inp.readLine())!=NULL) {

if (s.equalsIgnoreCase("MACRO")) {

prev=s;

for(;!(s=inp.readLine()).equalsIgnoreCase("MEND");mdtc++,prev=s){

if (prev.equalsIgnoreCase("MACRO")){

StringTokenizer st= new StringTokenizer(s);

String str[]=new String(st.countTokens());

for(i=0;i<str.length;i++) {

str[i]=st.nextToken();

}

mnt[mntc][0]=(mntc+1)+"";

mnt[mntc][1]=str[0];

mnt[mntc++][2]=(++mdtc)+"";

st=new StringTokenizer(str[1,","]);

String string[]=new String[st.countTokens()];

for (i=0;i<string.length;i++) {

string[i]=st.nextToken();

ala[alac][0]=alac+"";

index=string[i].indexOf("=");

if (index!=-1) {

ala[alac++][1]=string[i].substring(0,index);

}

else{

ala[alac++][1]=string[i];

}

}

}

else{

index=s.indexOf("&");

substring=s.substring(index);

for(i=0;i<alac;i++) {

if(ala[i][1].equals(substring)) {

s=s.replaceAll(substring,"#"+ala[i[0]]);

}

}

}

mdt[mdtc-1][0]=s;

}

mdt[mdtc-1][0]=s;

}

}

}

catch(FileNotFoundException ex) {

System.out.println("Unable to find file");

} catch(IOException e) {

e.printStackTrace();

}

}

static void display(String a[][],int n,int m) {

int i,j;

for(i=0;i<n;i++) {

for (j=0;j<m;j++) {

System.out.println(a[i][j]+"");

}

System.out.println();

}

}

}

INPUT from Input.txt

MACRO

ADDITION &arg1,&arg2,&arg3

MOV ax,&arg1

ADD ax,&arg2

ADD ax,&arg3

MEND

ADDITION 34,45,44

END

Output:

Macro Name Table

1 ADDITION 1

ALA for Pass 1

0 &arg1

1 &arg2

2 &arg3

MDT

ADDITION &arg1,&arg2,&arg3

MOV ax,#0

ADD ax,#1

ADD ax,#2

MEND

Program: Pass 2 macro

import java.io.\*;

import java.util.\*;

class Macropass2 {

public static void main(String args[]) {

pass2();

System.out.println("Argument List Array(ALA) for pass 2");

display(Pass1.ala,Pass1.alac,2);

System.out.println("All tables displayed here whereas the expanded output is stored in the file pass2\_output.txt");

}

static void pass2() {

int alap=0,index,mdtp,flag=0,i,j;

String s,temp;

try{

BufferedReader inp=new BufferedReader(new FileReader(new FileReader("pass1\_output.txt")));

File op=new File("pass2\_output.txt");

if(!op.exists()) {

op.createNewFile();

}

BufferedWriter output=new BufferedWriter (new FileWriter(op.getAbsoluteFile()));

for(;(s=inp.readLine())!=null;flag=0) {

StringTokenizer st=new StringTokenizer(s);

String str[]=new String(st.countTokens());

for(i=0;i<str.length;i++) {

str[i]=st.nextToken();

for(j=0;j<Pass1.mntc;j++) {

if(str[0].equals(Pass1.mnt[j][1])) {

mdtp=Integer.parseInt(Pass1.mnt[j][2]);

st=new StringTokenizer(str[1],",");

String arg[]=new String[st.countTokens()];

for(i=0;i<arg.length;i++) {

arg[i]=st.nextToken();

Pass1.ala[alap++][1]=arg[i];

}

for(i=mdtp;!(Pass1.mdt[i][0].equalsIgnoreCase("MEND"));i++) {

index=Pass1.mdt[i][0].indexOf("#");

temp=Pass1.mdt[i][0].substring(0,index);

temp+=Pass1.ala[Integer.parseInt(""+Pass1.mdt[i][0].charAt(index+1))][1];

output.write(temp);

output.newLine();

}

flag=1;

}

}

if (flag==0) {

output.write(s);

output.newLine();

output.close();

}

}

}

catch(FileNotFoundException ex) {

System.out.println("Unable to find file");

}

catch (IOException e) {

e.printStackTrace();

}

}

static void display(String a[],int n,int m) {

int i,j;

for(i=0;i<n;i++) {

for(j=0;j<m;j++) {

System.out.println(a[i][j]+"");

}

}

System.out.println();

}

}

}

Input:

PRG2 START

USING \*,BASE

INCR1 DATA1,DATA2

INCR2 DATA3,DATA4

FOUR DC F’4’

FIVE DC F’5’

BASE EQU 8

TEMP DS 1F

DROP 8

END

ST 4,DATA4

FOUR DC F’4’

FIVE DC F’5’

BASE EQU 8

TEMP DS 1F

DROP 8

END

Output:

ALA

1. DATA 1
2. DATA 2
3. DATA 3
4. DATA 4

PRG2 START

USING \*,BASE

A 1, DATA 1

L 2 , DATA 2

L 3, DATA 3

Program: Pass 2 Assembler

import java.io.\*;

public class assemblerPasstwo {

public static void main(String args[]){

BufferedReader inputReader=new BufferedReader(new FileReader("intermediate.txt"));

BufferedReader symReader=new BufferedReader(new FileReader("SYMTAB.txt"));

BufferedReader litReader=new BufferedReader(new FileReader("LITTAB.txt"));

BufferedReader outputWriter=new BufferedWriter(new FileWriter("machine\_code.txt"));

String inputLine;

String symLine=null;

String litLine=null;

while((inputLine=inputReader.readLine())!=null){

String[] tokens=inputLine.split("\\)\\(");

StringBuilder.outputLine=new StringBuilder();

for(String token:tokens) {

token=token.replaceAll("[\\(\\)]","");

if (token.startsWith("S,")) {

if (symLine==null) {

symLine=symReader.readLine();

}

if(symLine!=null) {

String[] symTokens=symLine.split("\t");

if(symTokens.length>1) {

int address=Integer.parseInt(symTokens[1]);

token=token.replace("S,","");

token=getFormattedmachineCode(token,address);

}

}

}else if(token.startsWith("L,")) {

if(litLine==null) {

litLine=litReader.readLine(); }

if(litLine!=null) {

String[] litTokens=litLine.split("\t");

if(litTokens.length>1) {

int address=Integer.parseInt(litTokens[1]);

token=token.replace("L,","");

token=getFormattedmachineCode(token,address);

}

}

}

outputLine.append(token.replace(",","")).append("");

}

outputLine=new StringBuilder(outputLine.toString().replaceAll("[A-Za-z]",""));

outputWriter.write(outputLine.toString().trim());

outputWriter.newLine();

}

inputReader.close();

symReader.close();

litReader.close();

outputWriter.close();

}

private static String getFormattedmachineCode(String instruction,int address) {

String[] parts=instruction.split(",");

String opcode=parts[0];

String[] operands=parts.length>1?parts[1].split(""):new String[0];

String registerNumbers="";

for(String operand:operands){

int regIndex=Integer.parseInt(operand);

registerNumbers+=getRegisterNumber(regIndex);

}

return opcode+""+registerNumbers+""+address;

}

private static String getRegisterNumber(int index) {

String[] regNumbers={"01","02","03","04"};

if(index>=0 && index<regNumbers.length) {

return regNumbers[index];

}

return "";

}

}

Output:

04 01 205

05 01 214

04 02 206

00 00 005

00 00 002

01 01 210

02 02 211

07 02 202

00 00 001

00 00 002

03 03 215

00 00 000

00 00 004

Program: Assembler pass 1

import java.io.\*;

import java.util.Scanner;

import java.util.StringTokenizer;

class AssemberPassOne {

static Scanner in=new Scanner(System.in);

static String is[]={

"STOP","ADD","SUB","MULT","MOVER","MOVEM","COMP","BC","DIV","READ","PRINT"};

static String ad[]={"START","END","LTORG","ORIGIN","EQU"};

static String d1[]={"DC","DS"};

static String cc[]={"LT","LE","EQ","GT","GE","ANY"};

static int symCounter=0;

static int litCounter=0;

static String sym[][]=new String[100][2];

static String lit[][]=new String[100][2];

static String ptab[][]=new String[100][2];

public static void main(String args[]) throws Exception {

int locate=0;

int litcount=0;

BufferedReader reader=new BufferedReader(new FileReader("input.asm"));

BufferedWriter writer=new BufferedWriter(new FileWriter("intermediate.txt"));

BufferedWriter writer1=new BufferedWriter(new FileWriter("SYMTAB.txt"));

BufferedWriter writer2=new BufferedWriter(new FileWriter("LITTAB.txt"));

String st;

String y,prev=null;

int stp=0;

String ans;

int k=0;

String buffer="";

String buffer1="";

String buffer2="";

while ((st=reader.readLine())!=null) {

int isFlag=0;

k++;

StringTokenizer splitted=new StringTokenizer(st);

ans="";

while(splitted.hasMoreTokens()){

y=splitted.nextToken();

if(y.equals("START")) {

locate=Integer.parseInt(splitted.nextToken());

ans="(AD 01)(C,"+locate")";

break;

}

else{

if(searchis(y)){

if(y.equals("STOP")) {

stp=1;

}

ans+="(IS,"+Integer.toString(indexis(y))+")";

isFlag=1;

locate+=1;

}

else if(searchad(y)) {

if(y.equals("LTORG")) {

locate+=litcount;

ans="(AD,05) \n";

while(litcount>0) {

lit[litcounter-litcount][1]=Integer.toString(locate-litcount);

int len=lit[litcounter-litcount][0].length();

String temp=lit[litcounter-litcount][0].substring(2,len-1);

ans+="(DL,02)(C,"+temp+")";

litcount--;

if(litcount!=0)

ans+="\n";

}

}

if(y.equals("ORIGIN")) {

y=splitted.nextToken();

String[] words=y.split("\\+");

int location=Integer.parseInt(sym[indexsym(words[0])][1]);

locate=location+Integer.parseInt(words[1]);

ans="(AD,03)(S,"+Integer.toString(indexsym(words[0])+1)+")+"+words[1];

}

if(y.equals("END") && litcount!=0) {

locate+=litcount;

ans="(AD,02) \n";

while(litcount>0) {

lit[litcounter-litcount][1]=Integer.toString(locate-litcount);

int len=lit[litcounter-litcount][0].length();

String temp=lit[litcounter-litcount][0].substring(2,len-1);

ans+="(DL,02)(C,"+temp+")\n";

}

}

if(y.equals("EQU")) {

int temp=indexsym(splitted.nextToken());

y=prev;

sym[indexsym(y)][1]=sym[temp][1];

ans="";

}

}

else if(searchd1(y)) {

if(y.equals("DS")) {

ans="";

ans+="(DL,01)(C,"+splitted.nextToken()+")";

}

if(y.equals("DC")) {

ans="";

ans+="(DL,2)(C,"+splitted.nextToken()+")";

}

locate+=1;

}

else{

prev=y;

char[] x=y.toCharArray();

if(x[0]=='=') {

int z=litcounter;

ans+="(L,"+(z+1)+")";

lit[litcounter++][0]=y;

litcount++;

}

else if(y.equals("AREG")) {

ans+="(R,1)";

}

else if(y.equals("BREG")) {

ans+="(R,2)";

}

else if(y.equals("CREG")) {

ans+="(R,3)";

}

else if(y.equals("DREG")) {

ans+="(R,4)";

}

else if(searchcc(y)) {

ans+="("+Integer.toString(indexcc(y)+1)+")";

}

else{

if(!searchsym(y) && isFlag==0 && stp==0) {

sym[symcounter][0]=y;

sym[symcounter++][1]=Integer.toString(locate);

ans+="(S,"+Integer.toString(indexsym(y)+1)+")";

if(splitted.hasMoreTokens()) {

ans="";

}

else if(!searchsym(y) && isFlag==1 && stp==0) {

sym[symcounter++][0]=y;

ans+="(S,"+Integer.toString(indexsym(y)+1)+")";

}

else if(searchsym(y) && isFlag==0){

sym[indexsym(y)][1]=Integer.toString(locate);

ans+="(S,"+Integer.toString(indexsym(y)+1)+")";

if(splitted.hasMoreTokens())

ans="";

prev=y;

}

else {

if(!splitted.hasMoreTokens()) {

ans+="(S,"+Integer.toString(indexsym(y)+1)+")";

continue;

}

}

}

}

}

ans=ans+"\n";

buffer+=ans;

}

System.out.println(buffer+"\n");

String ans1="";

for(int i=0;i<symcounter;i++) {

ans1+=sym[i][0]+"\t";

ans1+=sym[i][1]+"\t";

}

buffer1+=ans1;

System.out.println();

String ans2="";

for(int i=0;i<litcounter;i++) {

ans2+=lit[i][0]+"\t";

ans2+=lit[i][1]+"\t";

}

buffer2+=ans2;

System.out.println();

writer.write(buffer);

writer.write(buffer1);

writer.write(buffer2);

reader.close();

writer.close();

writer1.close();

writer2.close();

System.out.println("Program finished.....");

}

public static boolean search(String s) {

boolean flag=false;

int i=0;

while(i<11) {

if(is[i].equals(s)) {

flag=true;

break;

}

i++;

}

return flag;

}

public static booleansearchad(String s) {

boolean flag=false;

int i=0;

while(i<5) {

if(ad[i].equals(s)) {

flag=true;

break;

}

i++;

}

return flag;

}

public static boolean searchad1(String s) {

bollean flag=false;

int i=0;

while(i<2) {

if (d1[i].equals(s)) {

flag=true;

break;

}

i++;

}

return flag;

}

public static boolean searchsym(String s) {

boolean flag=s.equals("BREG")|| s.equals("AREG") || s.equals("CREG") || s.equals("DREG") || s.equals(",") || s.equals("LE")|| s.equals("LT")

|| s.equals("ANY")||s.equals("EQ") || s.equals("GT") || s.equals("GE") ;

int i=0;

while(i<symcounter) {

if(sym[i][0].equals(s)) {

flag=true;

break;

}

i++;

}

return flag;

}

public static boolean searchcc(String s) {

boolean flag=false;

int i=0;

while(i<6) {

if (cc[i].equals(s)) {

flag=true;

break;

}

i++;

}

return flag;

}

public static int indexsym(String s) {

int c=0,i=0;

while (i<symcounter) {

if (sym[i][0].equals(s)) {

c=i;

break;

}

i++;

}

return i;

}

public static int indexlit(String s) {

int c=0,i=0;

while(i<litcounter) {

if(lit[i][0].equals(s)) {

c=i;

break;

}

i++;

}

return i;

}

public static indexis(String s) {

int i=0;

while(i<l1) {

if(is[i].equals(s)) {

break;

}

i++;

}

return i;

}

public static int indexad(String s) {

int i=0;

while(i<5) {

if (ad[i].equals(s)) {

break;

}

i++;

}

return i;

}

public static int indexd1(String s) {

int i=0;

while (i<2) {

if(d1[i].equals(s)) {

break;

}

i++;

}

return i;

}

public static int indexcc(String s) {

int i=0;

while(i<6) {

if(cc[i].equals(s)) {

break;

}

i++;

}

return i;

}

}

}

}

Input:

START 200

MOVER AREG,=’5’

MOVEM AREG,X

L1 MOVER BREG,=’2’

ORIGIN L1+3

LTORG

NEXT ADD AREG,=’1’

SUB BREG,=’2’

BC LT,BACK

LTORG

BACK EQU L1

ORIGIN NEXT+5

MULT CREG,=’4’

STOP

X DS 1

END

OUTPUT:

(AD,01)(C,200)

(IS,4)(R,1)(L,1)

(IS,5)(R,1)(S,1)

(IS,4)(R,2)(L,2)

(AD,03)(S,2)+3

(AD,05)

(DL,02)(C,5)

(DL,02)(C,2)

(IS,1)(R,1)(L,3)

(IS,2)(R,2)(L,4)

(IS,7)(1)(S,4)

(AD,05)

(DL,02)(C,2)

(AD,03)(S,3)+5

(IS,3)(R,3)(L,5)

(IS,0)

(DL,1)(C,1)

(AD,02)

(DL,02)(C,4)

Literal Table

=’5’ 205

=’2’ 206

=’1’ 210

=’2’ 211

=’4’ 215

Symbol Table

X 214

L1 202

NEXT 207

BACK 202