LAPORAN RESMI PRAKTIKUM 12

“ *FCFS CPU Scheduling* ”

NAMA : FERI AFRIANTO

KELAS : D4 Teknik Informatika A

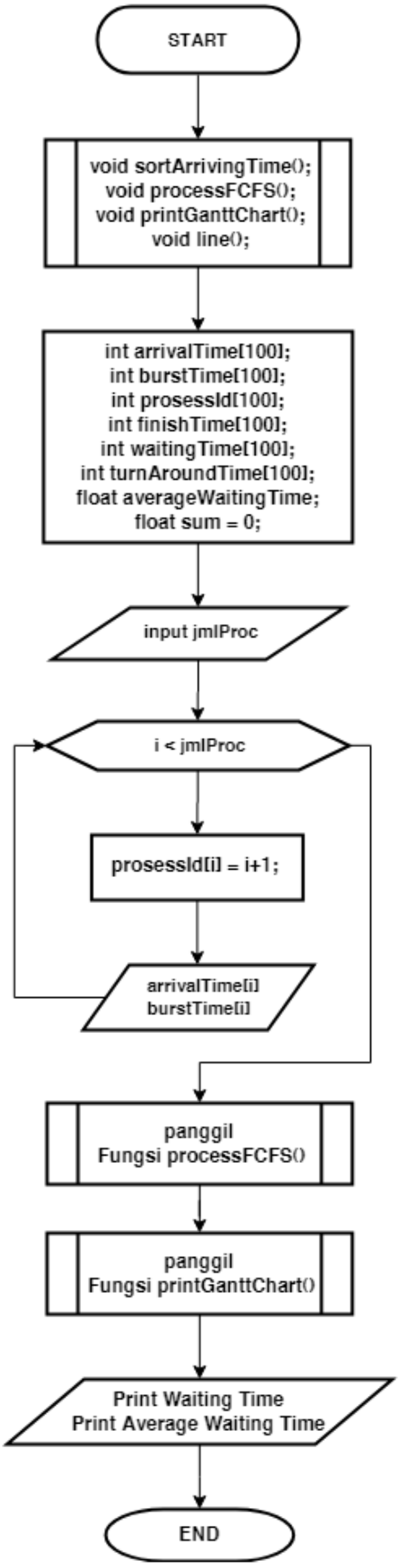
NRP : 2110191007

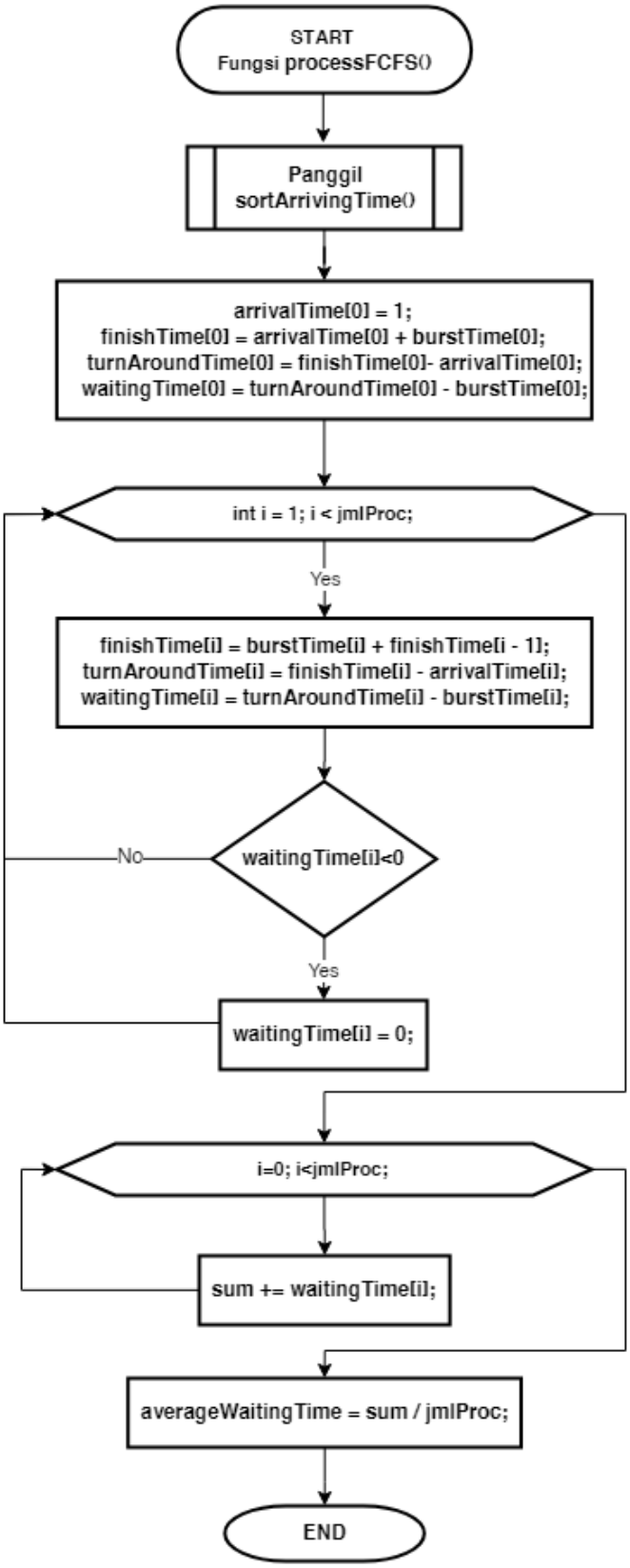
Politeknik Elektronika Negeri Surabaya

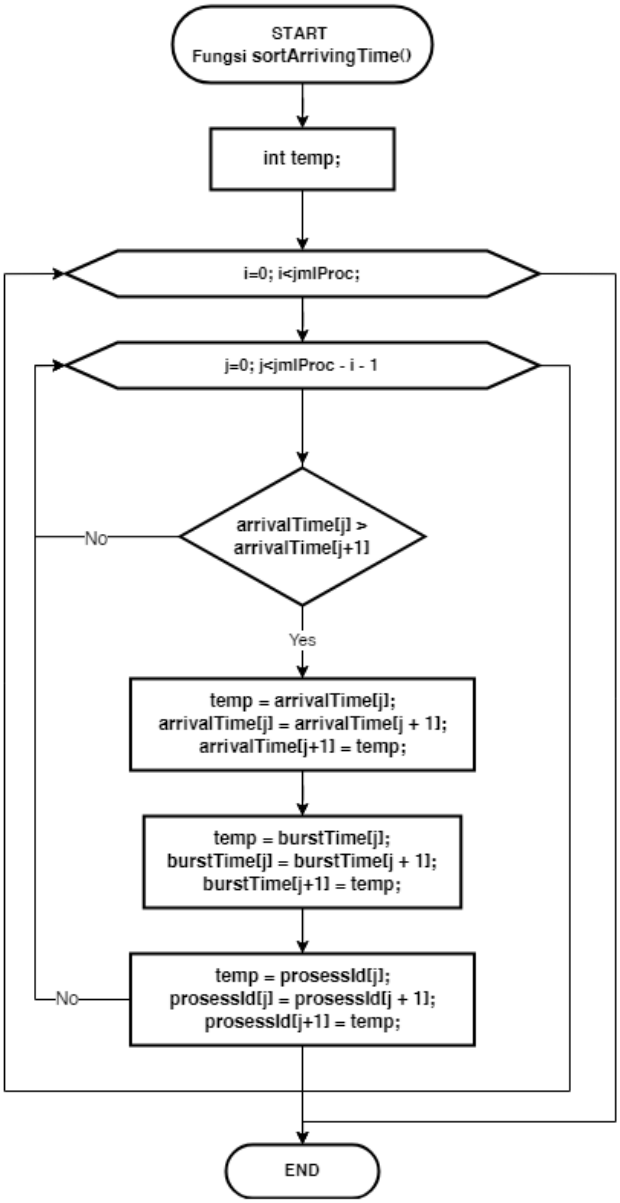
PRAKTIKUM 12

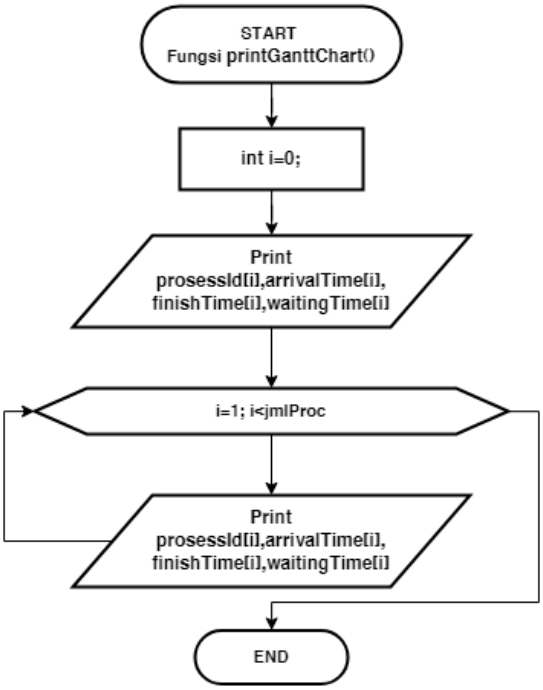
" *FCFS CPU Scheduling* "

1. FLOWCHART









2. LISTING PROGRAM

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

void sortArrivingTime();
void processFCFS();
void printGanttChart();
void line();

int jmlProc=0,i,j;
int arrivalTime[100];
int burstTime[100];
int prosessId[100];
int finishTime[100];
int waitingTime[100];
int turnAroundTime[100];
float averageWaitingTime;
float sum = 0;
```

```
int main()
{
    line();
    puts("|     FCFS CPU-SCHEDULING  |");
    line();
    printf("| Jumlah Proses : ");
    scanf("%d",&jmlProc);
    line();
    for(i=0; i<jmlProc; i++)
    {
        prosessId[i] = i;
        printf("| P%d\n",i);
        printf("| Arriving time : ");
        scanf("%d",&arrivalTime[i]);
        printf("| Burst time : ");
        scanf("%d",&burstTime[i]);
        puts("------------------------");

    }
    processFCFS();
    printGanttChart();
    line();
    printf("| Waiting Time         : %.2f\n",sum);
    printf("| Average Waiting Time : %.2f\n",averageWaitingTime);
    line();
    return 0;
}
void line()
{
    puts("----------------------------");
}
void sortArrivingTime()
{
    int temp;
    for(i=0; i<jmlProc; i++)
    {
        for(j=0; j<jmlProc - i - 1; j++)
        {
            if(arrivalTime[j] > arrivalTime[j+1])
            {
                temp = arrivalTime[j];
                arrivalTime[j] = arrivalTime[j + 1];
                arrivalTime[j+1] = temp;

                temp = burstTime[j];
                burstTime[j] = burstTime[j + 1];
                burstTime[j+1] = temp;

                temp = prosessId[j];
                prosessId[j] = prosessId[j + 1];
                prosessId[j+1] = temp;
            }

        }
    }
}
```

```
void processFCFS()
{
    sortArrivingTime();
    arrivalTime[0] = 1;
    finishTime[0] = arrivalTime[0] + burstTime[0];
    turnAroundTime[0] = finishTime[0]- arrivalTime[0];
    waitingTime[0] = turnAroundTime[0] - burstTime[0];
    for (int i = 1; i < jmlProc; i++)
    {
        finishTime[i] = burstTime[i] + finishTime[i - 1];
        turnAroundTime[i] = finishTime[i] - arrivalTime[i];
        waitingTime[i] = turnAroundTime[i] - burstTime[i];
        if (waitingTime[i]<0)
            waitingTime[i] = 0;
    }
    for (i=0; i<jmlProc; i++)
    {
        sum += waitingTime[i];
    }

    averageWaitingTime = sum / jmlProc;
}
void printGanttChart()
{
    int i=0;
    printf("| P%d -> Start: %d  End : %d Waiting Time :
%d\n",prosessId[i],arrivalTime[i],finishTime[i],waitingTime[i]);
    for(i=1; i<jmlProc; i++)
    {
        printf("| P%d -> Start: %d  End : %d Waiting Time :
%d\n",prosessId[i],finishTime[i-1],finishTime[i],waitingTime[i]);
    }
}
```

3. OUTPUT PROGRAM

➔ **Kasus 1**

```
----------------------------
|     FCFS CPU-SCHEDULING  |
----------------------------
| Jumlah Proses : 4
----------------------------
| P0
| Arriving time : 3
| Burst time : 3
-----------------------
| P1
| Arriving time : 2
| Burst time : 2
-----------------------
| P2
| Arriving time : 1
| Burst time : 1
-----------------------
| P3
| Arriving time : 4
| Burst time : 2
-----------------------
| P2 -> Start: 1  End : 2 Waiting Time : 0
| P1 -> Start: 2  End : 4 Waiting Time : 0
| P0 -> Start: 4  End : 7 Waiting Time : 1
| P3 -> Start: 7  End : 9 Waiting Time : 3
----------------------------
| Waiting Time          : 4.00
| Average Waiting Time : 1.00
----------------------------
```

➔ **Kasus 2**

```
----------------------------
|     FCFS CPU-SCHEDULING  |
----------------------------
| Jumlah Proses : 4
----------------------------
| P0
| Arriving time : 0
| Burst time : 1
-----------------------
| P1
| Arriving time : 2
| Burst time : 2
-----------------------
| P2
| Arriving time : 9
| Burst time : 1
-----------------------
| P3
| Arriving time : 5
| Burst time : 2
-----------------------
| P0 -> Start: 1  End : 2 Waiting Time : 0
| P1 -> Start: 2  End : 4 Waiting Time : 0
| P3 -> Start: 4  End : 6 Waiting Time : 0
| P2 -> Start: 6  End : 7 Waiting Time : 0
----------------------------
| Waiting Time         : 0.00
| Average Waiting Time : 0.00
----------------------------
```

4. KESIMPULAN

Algoritma ini merupakan algoritma penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma ini setiap proses yang berada pada status ready dimasukkan kedalam FIFO queue atau antrian dengan prinsip first in first out, sesuai dengan waktu kedatangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi.