Assignment II

Problem Bank 14

Assignment Description:

The assignment aims to provide deeper understanding of Pipelining Architecture, Scheduling and Multithreading using CPU- OS Simulator. The assignment has three parts.

- Part I deals with Pipeline Architecture
- Part II deals with Scheduling algorithm(FCFS, SJF and RR)
- Part III deals with Multithreading

Submission:

You will have to submit this documentation file and the name of the file should be GROUP-NUMBER.pdf. For Example, if your group number is 1, then the file name should be GROUP-1.pdf.

File submitted by any means outside CANVAS will not be accepted and marked.

In case of any issues, please drop an email to your respective course TAs.

Caution!!!

1. Assignments are designed for individual groups which may look similar and you may not notice minor changes in the assignments. Hence, refrain from copying or sharing documents with others. Any evidence of such practice will attract severe penalty.
2. **Marks will not be awarded for individual submissions**

Evaluation:

- The assignment carries 12 marks
- Grading will depend on
    - Contribution of each student in the implementation of the assignment
    - **Plagiarism or copying will result in -12 marks**

**************************FILL IN THE DETAILS GIVEN BELOW****************

Assignment Set Number:

Group Name:

Contribution Table:

Contribution (This table should contain the list of all the students in the group. Clearly mention each student's contribution towards the assignment. Mention "No Contribution" in cases applicable. If the contribution is equal the write 100%)

Sl. No.	Name (as appears in Canvas)	ID NO	Contribution (%)
1	CHANDUPATLA ANIRUDH REDDY	2022da04387	100%
2	BHARATH S	2022da04409	100%
3	DISYA RAKSHITA S	2022da04467	100%
4	MEDAPATI JAYANAGA SURESH REDDY	2022da04445	100%

Resource for Part I, II and III:

- Use following link to login to "eLearn" portal.
  - https://elearn.bits-pilani.ac.in
- Click on "My Virtual Lab – CSIS"
- Using your canvas credentials login in to Virtual lab
- In "BITS Pilani" Virtual lab click on "Resources". Click on "Computer Organization and software systems" course. Refer to LabCapsule 4, LabCapsule 5, LabCapsule 6.

Part I: Pipeline Processor

Consider the following program:

```
program pipeline1
      x=10
      y=20
      z=30
      y=z - y
      y=z + y
      y=y+1
      z=z-x
end
```

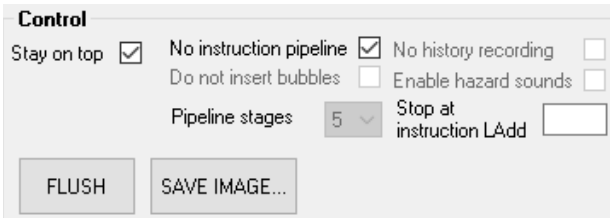
Compile the code and load it in CPU-OS simulator. Perform the following:

Execute the above program using non-pipelined processor and pipelined processor and answer the following questions.

Note: Every time flush the pipeline before running the code

A) Non-pipelined Processor:

To enable non-pipelined processor, check "No instruction pipeline" check box in control panel.



a) How many stages are there in non-pipelined processor? List them.

Solution:
- Fetch
- Decode
- Read Operands
- Execute
- Write Results

b) Fill in the following after executing of above program using non-pipelined processor.

	Clocks	Instruction Count	CPI	Speed up Factor
Non-Pipelined Processor	**127**	**23**	**5.52**	**0.91**

c) What are the contents of General-purpose registers after the execution of the program?

Solution:
R01: 10, R02: 41, R03: 20 ,R04: 20, R05: 20

GENERAL PURPOSE CPU REGISTERS

Reg	Val (D)	C	Val (D)
R00	0		
R01	10		
R02	41		
R03	20		
R04	20		
R05	20		
R06	0		
R07	0		
R08	0		
R09	0		
R10	0		

B) Pipelined processor:
To use, enable pipelined processor, uncheck "No instruction pipeline" check box in control panel.

a) Fill in the following table with respect to pipelined processor execution of the above program:

Pipelined processor conditions	**Clocks**	**Instruction Count**	**CPI**	**Speed up Factor**	**Data hazard (Yes/No)**	**Contents of registers used by the program**
Check "Do not insert bubbles" check box	**39**	**23**	**1.7**	**2.94**	**No**	GENERAL PURPOSE CPU REGISTERS R00: 0, R01: 0, R02: 50, R03: 51, R04: 51, R05: 51, R06: 0, R07: 0 **R02: 50, R03: 51 R04: 51, R05: 51**
Uncheck "Do not insert bubbles"	**46**	**23**	**2**	**2.5**	**Yes**	GENERAL PURPOSE CPU REGISTERS R00: 0, R01: 10, R02: 41, R03: 20, R04: 20, R05: 20, R06: 0, R07: 0 **R01: 10, R02: 41 R03: 20, R04: 20 R05: 20**

b) Is there a way to improve the CPI and Speed up factor? If so give the solution.

Solution:
Yes, By enabling pipeline the speed up factor(higher is better) went from 0.91 to 2.94 and CPI(lower is better) went from 5.52 to 1.7

Part II: Process Scheduling

Consider the following source codes:

```
program My_Pgm
        read(i)
        for n = 1 to 10
                x = i + n
        next
end
```

Compile the above source code and load it in the main memory.

We are now going to use the OS simulator to run this code. To enter the OS simulator:

1) Click on the OS O… button in the current window. The OS window opens.
2) You should see an entry, titled LoopTest, in the PROGRAM LIST view.
3) Now that this program is available to the OS simulator, we can create as many instances, i.e. processes, of it as we like. You do this by clicking on the CREATE NEW PROCESS button.

PART-II_A

- Select the **First-Come-First-Served (FCFS)** option in the SCHEDULER/Policies view
- Time slice should be considered as **seconds.**
- Create four processes P1, P2, P3 and P4 from source code respectively (Use the Priority drop-down list in the PROGRAM LIST / Process View): **3, 2, 4**,**1**
- Slide the Speed selector half-way down and then hit the START button.
- **Arrival delay** should be considered in **seconds** in the OS simulator

Now, give answer for the following:

a) What is the order in which processes are executed?

P1 , P2 ,P3 ,P4

b) What is the ***Elapsed time , Average Process Waiting Time*** and ***Average Burst Period*** and of each process? (To see this, Click on VIEWS button available on the left of your OS control, the click VIEW LOG)

Process	***Arrival Time/Delay***	***Elapsed Time (sec)***	***Average Process Waiting Time (sec)***	***Average Burst Period***
P1	0	114	0.33	88
P2	3	112	111.72	88
P3	4	112	222.9	88
P4	6	112	333.1	88

PART-II_B

- Select the **Shortest Job First (SJF)** option in the SCHEDULER/Policies view
- Select the Priority (static) as **Pre-emptive** option in the SCHEDULER/Policies view
- Time slice should be considered as **seconds.**
- Create four processes P1, P2, P3 and P4 from source codes respectively (Use the Priority drop-down list in the PROGRAM LIST / Process View): **3, 2, 4,1**
- Slide the Speed selector half-way down and then hit the START button.
- **Arrival delay** should be considered in **seconds** in the OS simulator

Now, give answer for the following:

a) What is the order in which processes are executed?

P1 ,P2, P3, P4

b) What is the ***Elapsed time , Average Process Waiting Time*** and ***Average Burst Period*** and of each process? (To see this, Click on VIEWS button available on the left of your OS control, the click VIEW LOG)

Process	***Arrival Time/Delay***	***Elapsed Time (sec)***	***Average Process Waiting Time (sec)***	***Average Burst Period***
P1	0	114	0.33	88
P2	3	112	111.71	88
P3	4	112	223.18	88
P4	6	112	333.37	88

PART-II_C

- Select the **Round Robin (RR) with 5 seconds as time slice** option in the SCHEDULER/Policies view.
- Select the Priority (static) as **Pre-emptive** option in the SCHEDULER/Policies view
- Time slice should be taken in terms of **seconds** instead of **ticks**
- Create four processes P1, P2, P3 and P4 from source codes respectively (Use the Priority drop-down list in the PROGRAM LIST / Process View): **3, 2, 4,1**
- Slide the Speed selector half-way down and then hit the START button.
- **Arrival delay** should be considered in **seconds** in the OS simulator

Now, give answer for the following:

a) What is the order in which processes are executed?

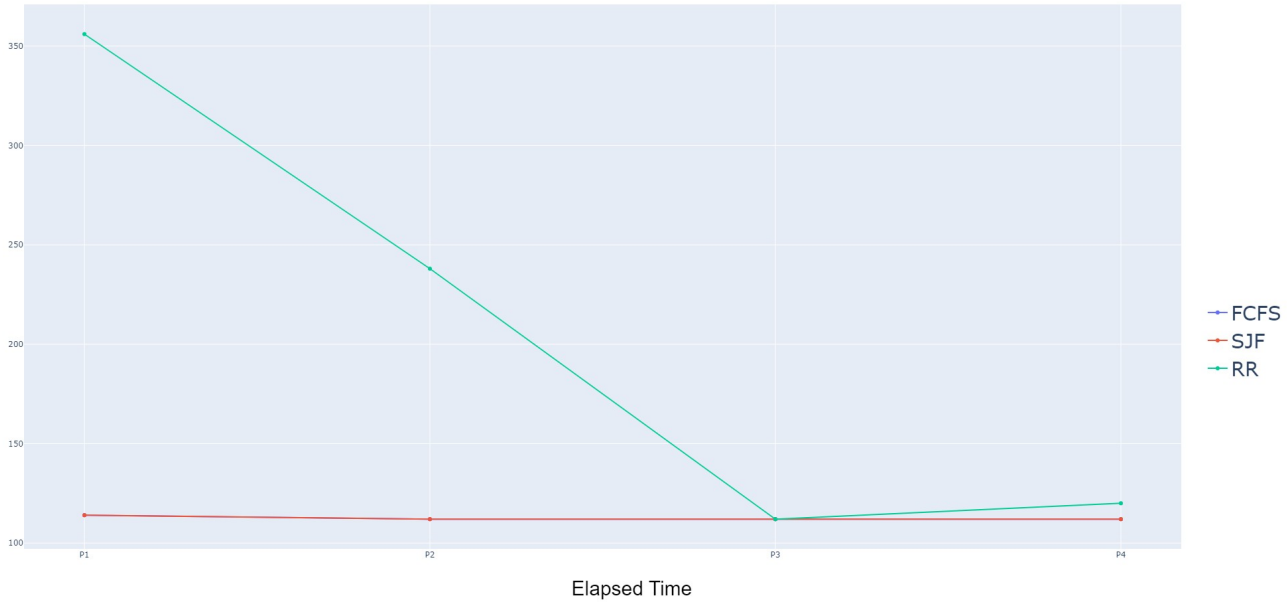
P4, P2, P1, P3

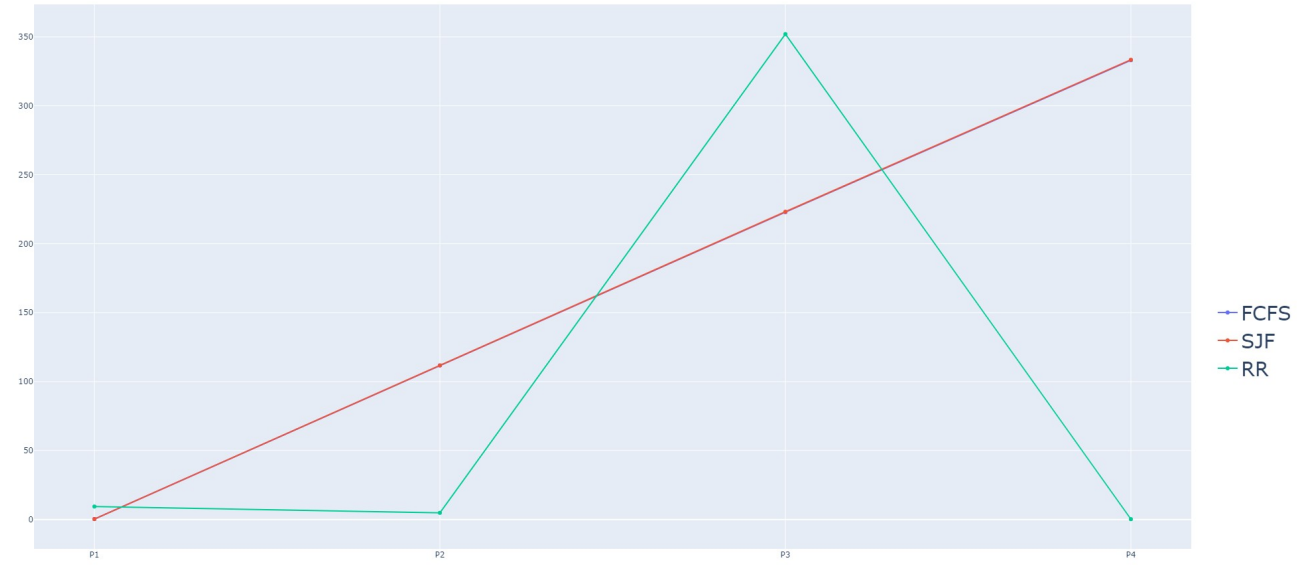
b) What is the ***Elapsed time , Average Process Waiting Time*** and ***Average Burst Period*** and of each process? (To see this, Click on VIEWS button available on the left of your OS control, the click VIEW LOG)

Process	***Arrival Time/Delay***	***Elapsed Time (sec)***	***Average Process Waiting Time (sec)***	***Average Burst Period***
P1	0	356	9.41	3
P2	3	238	4.85	3
P3	4	112	352	88
P4	6	120	0.25	3

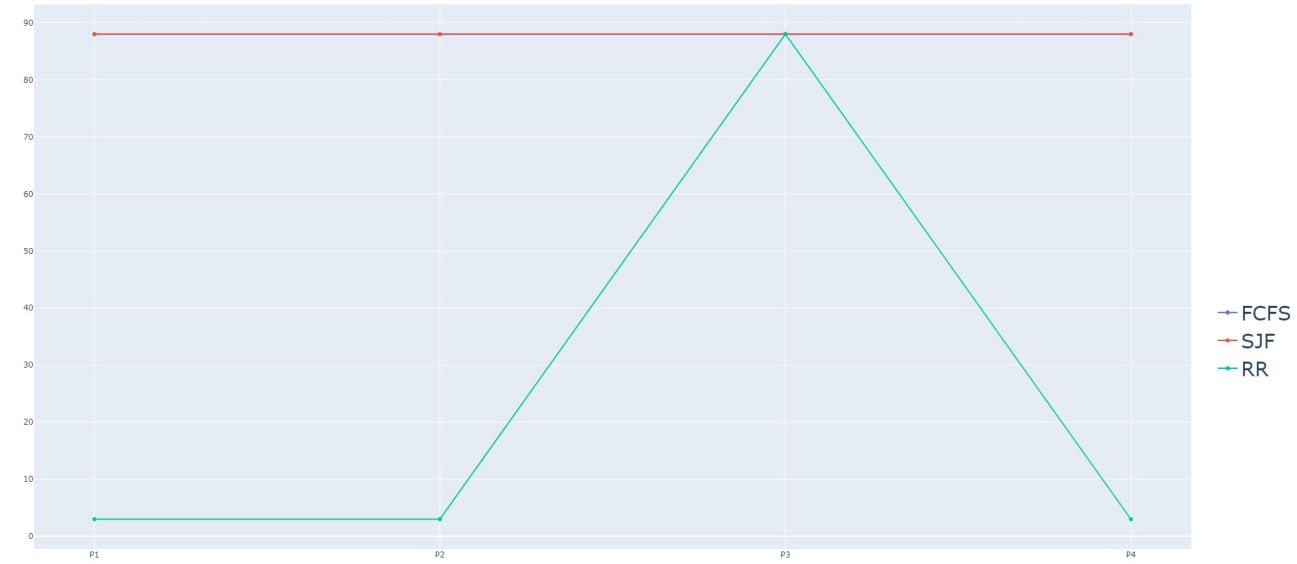
PART-II_D

a) Plot a graph from the results obtained by FCFS, SJF and Round Robin scheduling and explain which algorithm is better among these with proper justification





Average Process Waiting Time



Average Burst Period

- For Elapsed time FCFS & SJF are better since they took less time from the start while Round robin took some extra time at start

- For Average Process Waiting Time FCFS & SJF are better since they are constantly increasing in linear manner , while Round Robin is having too high or too low unpredictable waiting times
- For Average Burst Period Round Robin is better since it had less burst period for P1,P2, & P4 while FCFS & SJF had large burst period for all processes

Part III: Multi-Threading

Consider the following source code

```
program ThreadTest
        total = 0
        sub thread1 as thread
            for i = 1 to 10
                total = total + i
            next
        end sub
        sub thread2 as thread
            call thread1
            for i = 11 to 20
                total = total + i
            next
        end sub
        sub thread3 as thread
            call thread2
            for i = 21 to 30
                total = total + i
            next
        end sub
        call thread3
        wait
        writeln("Total =", total)
    end
```

Compile the above source code and load it in the main memory. Create a single process, choose RR scheduling algorithm with time quantum of 5 seconds with no-priority. Run the Process.

Answer the following questions:

a) What is the value of "Total" ? (value of "Total" is 55)

b) How many processes and how many threads are created? (1 process and 3 threads)

c) Identify the name of the processes and threads.

name of process : "THREADTEST" ,

name of threads: "P1T3", "P1T3T4", "P1T3T4T5"

d) What is the PID and PPID of the processes and threads created?

Process Name	PID	PPID
THREADTEST	1	0
P1T3	2	1
P1T3T4	3	2
P1T3T4T5	4	3

e) Represent the parent and child relationship using tree representation