Assignment II

Problem Bank 22

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, RR, SJF)
- 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.

Submit the assignment by **3rd March 2021**, **through canvas only**. File submitted by any means outside CANVAS will not be accepted and marked.

In case of any issues, please drop an email to the course TAs, Ms. Michelle Gonsalves (michelle.gonsalves@wilp.bits-pilani.ac.in).

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	AMAL SHAJI	2020FC04101	33.34%
2	UMANG MISHRA	2020FC04103	33.34%
3	GNANESWAR S V V	2020FC04102	33.34%

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 pipeline2
        a=10
        b=20
        c=30
        b=a+b
        c=c+b
        a=a/3
        a=c
    end
```

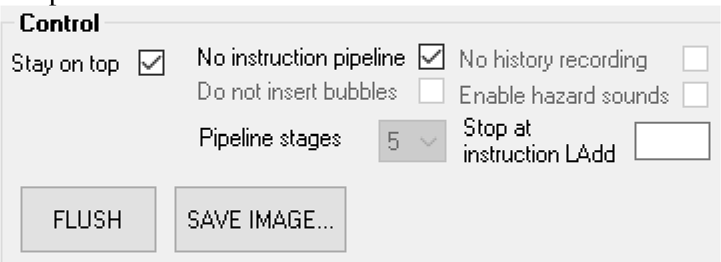
Compile the code and load it in CPU-OS simulator. Perform the following:

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.



Control

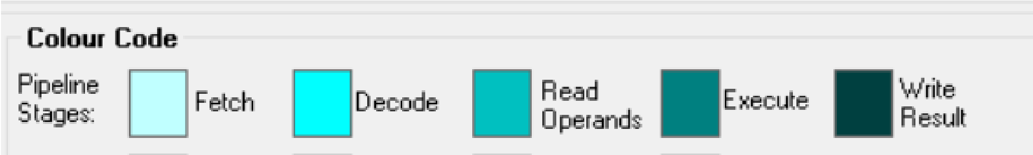
Stay on top ☑ No instruction pipeline ☑ No history recording □

Do not insert bubbles □ Enable hazard sounds □

Pipeline stages 5 Stop at instruction LAdd

FLUSH SAVE IMAGE...

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



Colour Code

Pipeline Stages: Fetch Decode Read Operands Execute Write Result

There are 5 stages in the non-pipelined processor. Consisting of

1. Fetch
2. Decode
3. Read Operands
4. Execute
5. Write Result

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

	Clocks	Instruction Count	CPI	Speed up Factor
Non Pipelined processor	122	21	5.81	0.86

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

R01 → 60
R02 → 30
R03 → 60
R04 → 3
R05 → 60

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	36	21	1.71	2.92	No	R01 → 0 R02 → 30 R03→ 0 R04 → 16 R05 → 0
Uncheck "Do not insert bubbles"	45	21	2.14	2.34	Yes	R01 → 60 R02 → 30 R03→ 60 R04 → 3 R05 → 60

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

Solution:

- The pipeline has a means of enabling operand forwarding, i.e. prior to previous instructions updating registers in a later stage, that contributes to reducing the CPI and improve speed factor.
- You may also try re-arranging the instructions if they cause any hazard.

Part II: Process Scheduling

Consider the following 4 source codes:

Source Code 1:

```
program My_Pgm1
    i = 1
    for n = 1 to 10
        x = i + n
    next
end
```

Source Code 2:

```
program My_Pgm2
    i = 10
    for n = 1 to 8
        x = i + n
    next
end
```

Source Code 3:

```
program My_Pgm3
    i = 10
    for n = 1 to 15
        x = i + n
    next
    end
```

Source Code 4:

```
    program My_Pgm4
        i = 10
        for n = 1 to 5
            x = i - n
        next
    end
```

Create 4 processes P1, P2. P3 and P4 from source codes 1, 2, 3and 4 respectively with following properties. Fill up the following table by considering **Log data only**:

Scheduling Algorithm: FCFS			
Process	Arrival Time		Waiting time
P1	0		0.17 sec
P2	0		16.37 sec
P3	0		30.07 sec
P4	0		51.97 sec
Average waiting time:		24.64 sec	
Scheduling Algorithm: Round Robin with time quantum 5			
Process	Arrival Time		Waiting time
P1	0		4.69 sec
P2	0		5.32 sec
P3	0		4.75 sec
P4	0		5.89 sec
Average waiting time:		49.66 sec	
Scheduling Algorithm: Shortest Job First (Pre-emptive)			
Process	Arrival Time	Priority	Waiting time
P1	0	1	0.16 sec
P2	2	2	14.5 sec
P3	6	1	35.04 sec
P4	4	2	26.04 sec
Average waiting time		18.93 sec	
Scheduling Algorithm: Shortest Job First (Non-Pre-emptive)			
Process	Arrival Time	Priority	Waiting time
P1	0	1	0.16 sec
P2	2	2	13.99 sec
P3	6	3	34.46 sec
P4	4	4	25.46 sec
Average waiting time: 18.52 sec			
Out of three cases, which one is better and why? Shortest Job First (Non-Pre-emptive and Pre-emptive) is the best as its average waiting time is much better than other algorithms. The SJF algorithm will stop processing with low			

priority and start executing high priority which thus allow better scheduling comparatively. The throughput is increased because more processes can be executed in less amount of time (shortest job). Also, The SJF is better if the process comes to processor simultaneously.

Part III: Multi-Threading

Consider the following source code

```
program ThreadTest
        total = 100
        sub thread1 as thread
            for i = 1 to 2
                total = total - i

            next
        end sub
        sub thread2 as thread
            for i = 3 to 4
                total = total + i
                call thread1
            next
        end sub
```

```
            call thread2
            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 3 ticks. Run the Process.

Answer the following questions:

a) What is the value of “Total”?

The value of the Total is 7.

b) How many processes and how many threads are created?

One process and two threads are created.

c) Identify the name of the processes and threads.

Process: P1T0

Thread: P1T0T1 and P1T0T2

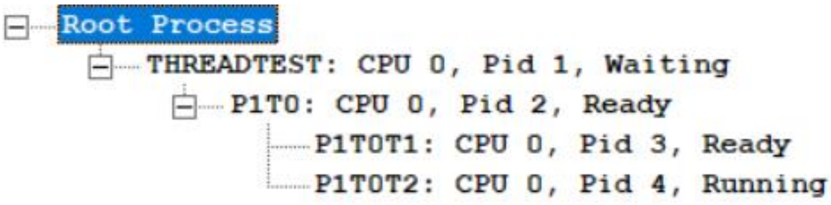
d) What is the PID and PPID of the processes and threads created?

Process: P1T0: PID is 2

Thread: P1T0T1: PID is 3 and PPID is 2

Thread: P1T0T2: PID is 4 and PPID is 2

e) Represent the parent and child relationship using tree representation.



```
Root Process
    THREADTEST: CPU 0, Pid 1, Waiting
        P1T0: CPU 0, Pid 2, Ready
            P1T0T1: CPU 0, Pid 3, Ready
            P1T0T2: CPU 0, Pid 4, Running
```