**Assignment II**

**Problem Bank 16**

**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.

Submit the assignment by **5th September 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](mailto: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 (%)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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

x=x + y

x=x + z

x= x – 2

x=x + 1

end

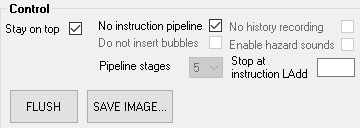
Compile the code and load it in CPU-OS simulator. Perform the following:

**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***

1. **Non-pipelined Processor:**

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



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

|  |
| --- |
| Solution: |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Clocks | Instruction Count | CPI | Speed up Factor |
| Non-Pipelined Processor |  |  |  |  |

1. What are the contents of General-purpose registers after the execution of the program?

|  |
| --- |
| Solution: |

1. **Pipelined processor:**

To use, enable pipelined processor, uncheck “No instruction pipeline” check box in control panel.

1. 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 |  |  |  |  |  |  |
| Uncheck “Do not insert bubbles” |  |  |  |  |  |  |

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

|  |
| --- |
| Solution: |

**Part II: Process Scheduling**

Consider the following source codes:

**program My\_Pgm**

**i = 2**

**for n = 1 to 10**

**x = i + n**

**n = n + 2**

**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:**

1. What is the order in which processes are executed?

|  |
| --- |
|  |

1. 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 |  |  |  |
| P2 | 3 |  |  |  |
| P3 | 4 |  |  |  |
| P4 | 6 |  |  |  |

**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:**

1. What is the order in which processes are executed?

|  |
| --- |
|  |

1. 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 |  |  |  |
| P2 | 3 |  |  |  |
| P3 | 4 |  |  |  |
| P4 | 6 |  |  |  |

**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:**

1. What is the order in which processes are executed?

|  |
| --- |
|  |

1. 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 |  |  |  |
| P2 | 3 |  |  |  |
| P3 | 4 |  |  |  |
| P4 | 6 |  |  |  |

**PART-II\_D**

1. 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

**Part III: Multi-Threading**

Consider the following source code

program ThreadTest

total = 0

sub thread1 as thread

for i = 1 to 3

total = total + i

next

end sub

sub thread2 as thread

for i = 3 to 5

call thread1

total = total \*i

next

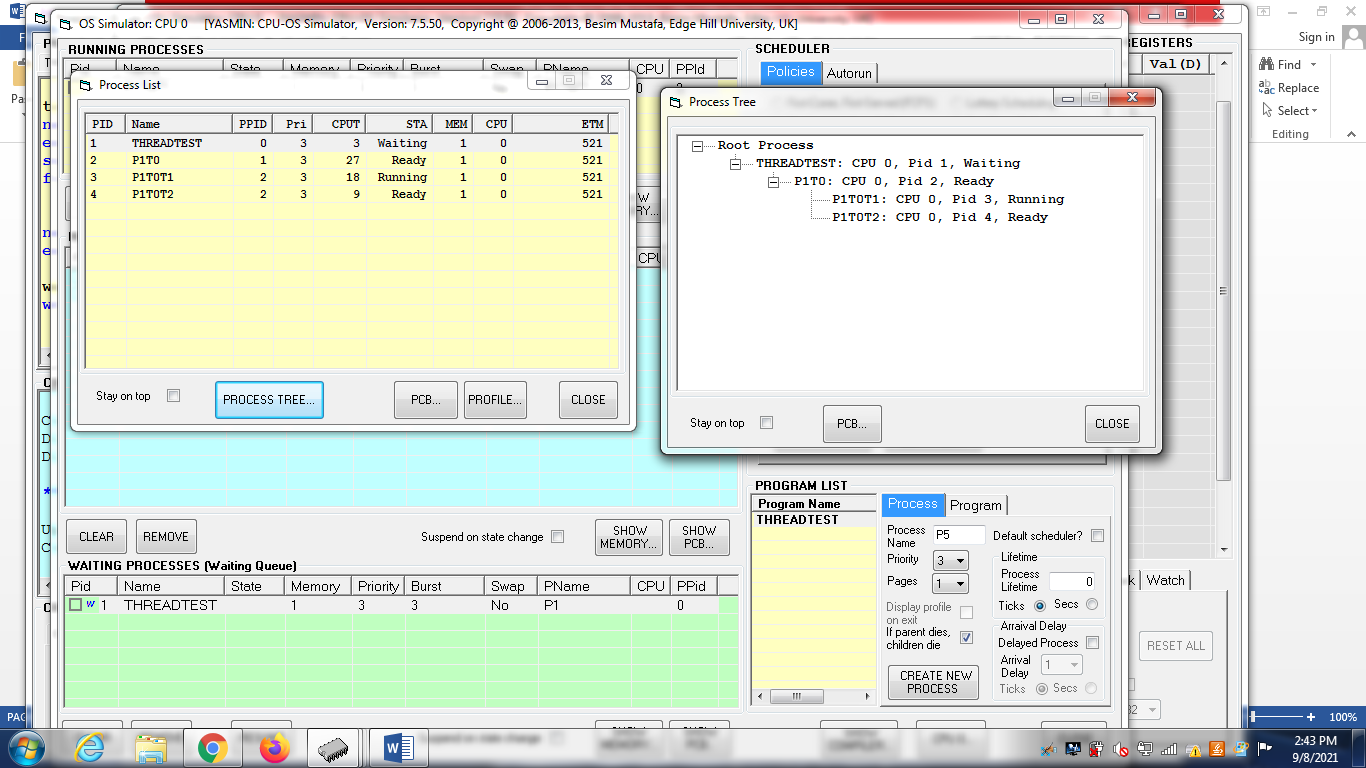
end sub

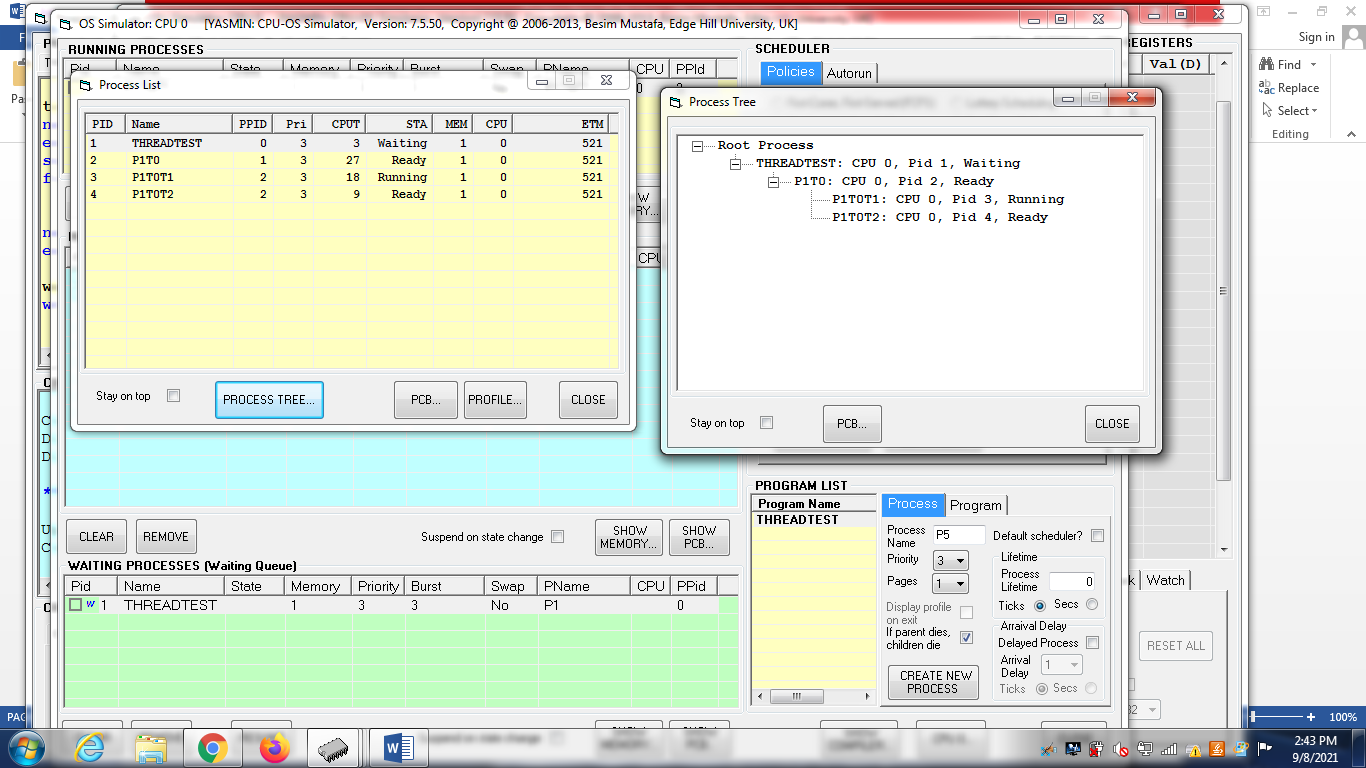
call thread2

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 5 seconds. Run the Process.

**Answer the following questions:**

1. What is the value of “Total”?

Total = 1

1. How many processes and how many threads are created?

Process – 1

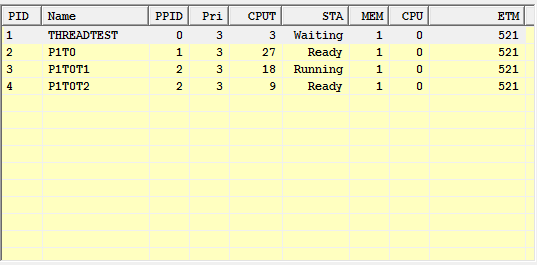
Thread - 1

1. Identify the name of the processes and threads.

P1 – THREADTEST

Thread – P1T0

1. What is the PID and PPID of the processes and threads created?



P1 is the process, THREADTEST

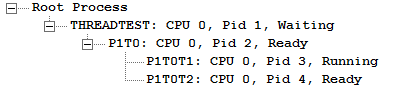
PID – 1 and PPID - 0

Here P1T0 thread is corresponding to the subtask

PID – 2 and PPID – 1

P1 is the parent of P1T0

1. Represent the parent and child relationship using tree representation



In the process tree, P1T0 is the child of P1 where, P1 is the THREADTEST. In the Round Robin Scheduling process, OS control is resumed and the execution begins. Each process and the thread gets CPU for 5 time limits after which it goes back to the ready queue.