**Assignment II**

**Problem Bank 37**

**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 **3rd March 2022,** 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

z=z+0

x=x + y

x=x + z

x= x /2

x=x + 1

end

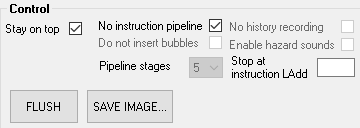
Compile the code and load it in CPU-OS simulator. Perform the following:

**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: 5 Stages   * Fetch * Decode * Read Operand * Execute * Write Result |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Clocks | Instruction Count | CPI | Speed up Factor |
| Non-Pipelined Processor | 154 | 27 | 5.7 | 0.88 |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Solution:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | R00 | R01 | R02 | R03 | R04 | R05 | | 0 | 31 | 20 | 30 | 31 | 31 |   All other registers R06-R31 have 00 |

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 | 46 | 27 | 1.7 | 2.94 | No | |  |  | | --- | --- | | R00 | 0 | | R01 | 20 | | R02 | 0 | | R03 | 30 | | R04 | 21 | | R05 | 21 | |
| Uncheck “Do not insert bubbles” | 60 | 27 | 2.22 | 2.25 | Yes | |  |  | | --- | --- | | R00 | 0 | | R01 | 31 | | R02 | 20 | | R03 | 30 | | R04 | 31 | | R05 | 31 | |

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

|  |
| --- |
| *Solution:*  Adding more stages to the pipeline can help increase the CPI and speedup factor. Also, this program has a relatively small set of instructions. For large number of instructions, we can really expect the ideal speedup which is more than what we are currently getting. |

**Part II: Process Scheduling**

Consider the following source codes:

**Source Code 1:**

program My\_Pgm1

i = 1

for n = 1 to 10

x = i + n

n = n + 1

next

end

**Source Code 2:**

program My\_Pgm2

i = 1

for n = 1 to 12

x = i + n

n = n + 1

next

end

**Source Code 3**:

program My\_Pgm3

i = 3

for n = 1 to 10

x = i + n

n = n + 2

next

end

**Source Code 4**:

program My\_Pgm4

i = 3

for n = 1 to 10

x = i + n

n = n + 3

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 as the program name given above, 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?

|  |
| --- |
| P1 P2 P3 P4 |

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 | 88 | 0.33 | 67 |
| P2 | 3 | 103 | 86.1 | 79 |
| P3 | 3 | 73 | 189.52 | 55 |
| P4 | 6 | 58 | 260.02 | 43 |

**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?

|  |
| --- |
| P1 P2 P3 P4 |

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 | 88 | 0.34 | 67 |
| P2 | 3 | 103 | 85.85 | 79 |
| P3 | 3 | 73 | 190.02 | 55 |
| P4 | 6 | 58 | 260.25 | 43 |

**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?

|  |
| --- |
| P1 P2 P2 P4 P4 P4 P4 P4 P4 P4 P4 P4 P4 P4 P4 P4 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1 P3 |

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 | 278 | 8.86 | 3 |
| P2 | 3 | 180 | 2.72 | 3 |
| P3 | 3 | 73 | 275.82 | 55 |
| P4 | 6 | 64 | 0.24 | 3 |

**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

|  |  |  |  |
| --- | --- | --- | --- |
|  | **FCFS** | **SJF** | **RR** |
| **Overall Avg  Elapsed Time (s)** | 80.50 | 80.50 | 148.75 |
| **Overall Avg  Waiting Time (s)** | 133.99 | 134.12 | 71.91 |
| **Overall Avg  Burst Period (ticks)** | 61.00 | 61.00 | 16.00 |

If we compare for the given example from the table above,

* + - 1. FCFS and SJF seem to give approximately the same performance.
      2. Round Robin differs significantly from the other two algorithms in terms of all the three parameters.
      3. Round Robin
         1. Ensures *all processes get some CPU utilization* once in a while. Hence overall avg. waiting time of processes is quite low.
         2. However, there’s a lot of context switching happening in this algorithm because of which avg. overall time needed for an individual process to complete increases somewhat.
      4. In general *Round Robin is a better scheduling algorithm* provided the quantum is not too small (in which case the time taken for context switching will adversely impact the overall elapsed time and slow down the processes).

**Part III: Multi-Threading**

Consider the following source code

program ThreadTest

total = 0

sub thread1 as thread

for i = 4 to 8

total = total + i

next

end sub

sub thread2 as thread

for i = 1 to 4

total = total + 10 + i

next

end sub

call thread2

call thread1

wait

writeln (“Total =”, total)

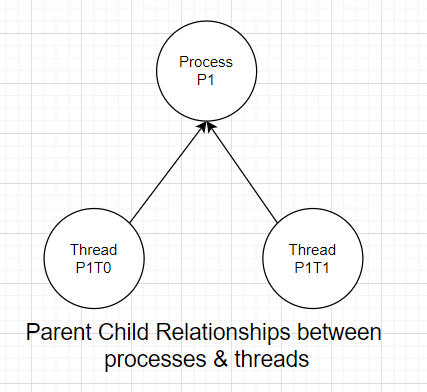
end

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 = 30
2. How many processes and how many threads are created?   
   Processes = 1, Threads = 2
3. Identify the name of the processes and threads.   
   Process = {P1}, Threads = {P1T0 P1T1}
4. What is the PID and PPID of the processes and threads created?

|  |  |  |
| --- | --- | --- |
|  | PID | PPID |
| Process P1 | 1 | 0 |
| Thread P1T0 | 2 | 1 |
| Thread P1T1 | 3 | 1 |

1. Represent the parent and child relationship using tree representation

**PART-IV:Deadlock**

program DeadlockP1

resource(0, allocate)

wait(3)

resource(1, allocate)

for n = 1 to 20

next

end

program DeadlockP2

resource(1, allocate)

wait(3)

resource(2, allocate)

for n = 1 to 20

next

end

program DeadlockP3

resource(2, allocate)

wait(3)

resource(3, allocate)

for n = 1 to 20

next

end

program DeadlockP4

resource(3, allocate)

wait(3)

resource(0, allocate)

for n = 1 to 20

next

end

* The above source code creates a program which attempts to allocate two resources for itself.
* After the first allocation it waits for 3 seconds and tries to allocate another resource.
* Finally, it counts from 1 to 20 in a loop and then terminates

**Problem Statement:**

* Create Four processes from the above source code .
* Allocate the resources for each process as below:
  + P1🡪 R0 and R1
  + P2🡪 R1 and R2
  + P3🡪 R2 and R3
  + P4🡪 R3 and R0
* Save each program into local directory
* Compile each one of the four-source code.
* Load in memory the four pieces of code generated.
* Now switch to the OS simulator.
* Create a single instance of each of the programs. You can do this by double-clicking on each of the program names in the PROGRAM LIST frame under the Program Name column.
* In the SCHEDULER frame select Round Robin (RR) scheduling policy with no priority in the Policies tab.
* Select the Views tab and click on the VIEW RESOURCES… button.
* In the OS Control tab use the START button to start the OS scheduler and observe the changing process states for few seconds.

1. How to avoid the deadlock by releasing either resource or a process (Justify your answer wrt cost effective)

Releasing any one process or resource will bring us back out of the deadlock into a safe state.  
for eg. Consider I release resource R0 from the hands of P1, then

* P4 can run till completion and it will release resource R0 & R3.
* Once R3 is released, P3 can get R3 and it will run to completion
* Once P3 runs to completion then it will release R2 & R3
* Then P2 can get R2, run to completion and release R2 & R1.
* Then P1 can claim both R1 & R0 and run to completion and all processes will be successfully completed safely.

1. Draw a Resource Allocation Graph.

