SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing

Mode of Exam
OFFLINE
SET C

DEPARTMENT OF COMPUTING TECHNOLOGIES

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

Academic Year: 2021-2022 (Even)

Test: CLAT-2 **Date: 26-05-2022**
Course Code & Title: 18CSC205J: Operating Systems **Duration:** 2 Period
Year & Sem: 2022 & IV Semester **Max. Marks:** 50 Marks

Course Articulation Matrix: ***(to be placed)***

Part - A
(10 x 1 = 10 Marks)
Instructions: Answer all

Q. No	Question	Marks	BL	CO	PO	PI Code
1	b. Producer Consumer problem	1	1	2	3	3.5.1
2	b. there are n buffers (n being greater than one but finite)	1	1	2	3	3.5.1
3	d. writers	1	1	2	3	3.5.1
4	a. 5 philosophers and 5 chopsticks	1	3	2	3	3.7.1
5	a. count the number of empty and full buffers	1	3	2	3	3.5.1
6	c. Frame number with offset	1	1	3	1	1.6.1
7	c. segments are shared	1	1	3	1	1.6.1
8	b. Compile-time address binding	1	1	3	1	1.6.1
9	a. when the process completes its execution b. when the process waits for I/O operation	1	1	3	1	1.6.1
10	c. 4000 ms	1	1	3	1	1.6.1
	Part - B **(4 x 5 = 20 Marks)** **Instructions: Answer Any 5**					
11	**Give two reasons why this is a bad implementation for a lock:** **lock.acquire() { disable interrupts; }** **lock.release() { enable interrupts; }** **Answer:** There are a number of reasons why this is a bad implementation for a lock. (1) It prevents hardware events from occurring during the critical section, (2) User programs cannot use this lock, (3) It doesn't work for data shared between different processors.	5	4	2	3	3.6.2
12	**Design an algorithm for a monitor that**	5	4	2	3	3.6.1

<table>
<tr>
<td></td>
<td><b>implements an alarm clock that enables a calling program to delay itself for a specified number of time units (ticks). You may assume the existence of a real hardware clock that invokes a function tick() in your monitor at regular intervals.</b>

<b>Answer:</b>

monitor AlarmClock
{
condition c;
int cur_time=0;
void delay(int ticks)
{
while(cur_time < cur_time+ticks)
wait(c);
signal(c);
}
void tick()
{cur_time++;}
}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td><b>Consider a paging system with the page table stored in memory.</b>

<b>a. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?</b>
<b>b. If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there)</b>

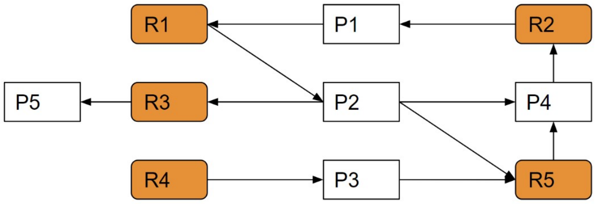
a. 400 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory.
b. Effective access time= 0.75 * (200 nanoseconds) + 0.25 * (400 nanoseconds) = 250 nanoseconds</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3.6.1</td>
</tr>
<tr>
<td>14</td>
<td><b>Consider the following segment table:</b>

<table>
<tr><td><b>Segment</b></td><td><b>Base</b></td><td><b>Length</b></td></tr>
<tr><td><b>0</b></td><td><b>219</b></td><td><b>600</b></td></tr>
<tr><td><b>1</b></td><td><b>2300</b></td><td><b>14</b></td></tr>
<tr><td><b>2</b></td><td><b>90</b></td><td><b>100</b></td></tr>
<tr><td><b>3</b></td><td><b>1327</b></td><td><b>580</b></td></tr>
<tr><td><b>4</b></td><td><b>1952</b></td><td><b>96</b></td></tr>
</table>
</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3.7.2</td>
</tr>
</table>

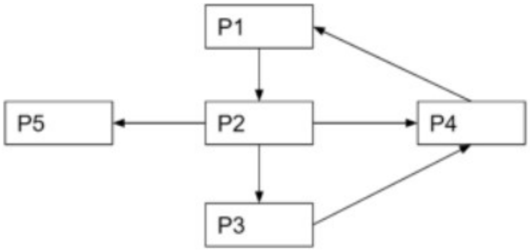
	What are the physical addresses for the following logical addresses? **a. 0,430** **b. 1,10** **c. 2,500** **d. 3,400** **e. 4,112** Answer: a. 219+430 = 649 b. 2300 + 10 = 2310 c. illegal reference, trap to operating system d. 1327+ 400= 1727 e. illegal reference, trap to operating system					
15	**In a paging scheme, virtual address space is 4 KB and page table entry size is 8 bytes. What should be the optimal page size?** Given- • Virtual address space = Process size = 4 KB • Page table entry size = 8 bytes We know- Optimal page size = (2 x Process size x Page table entry size)$^{1/2}$ = (2 x 4 KB x 8 bytes)$^{1/2}$ = (2^{16} bytes x bytes)$^{1/2}$ = 2^{8} bytes = 256 bytes Thus, Optimal page size = 256 bytes.	**5**	**4**	**3**	**2**	**2.6.2**
Part – C **(2 x 10 = 20 Marks)** **Instructions: Answer All**						
16.a	**Analyse and develop a graph to detect the possibility of deadlock in the following scenario which has only one instance in each resource. And**	10	4	2	3	3.6.1

<table>
<tr>
<td></td>
<td>

illustrate the methods that helps to recover from the detected deadlock.



Answer:



P1 => P2 => P3 => P4, It is forming a cycle hence its in a deadlock state.

Recovery Mechanism

Process termination

Kill a Process

Kill all the process

Resource Preemption

Preemption and RollBack

</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>

16.b

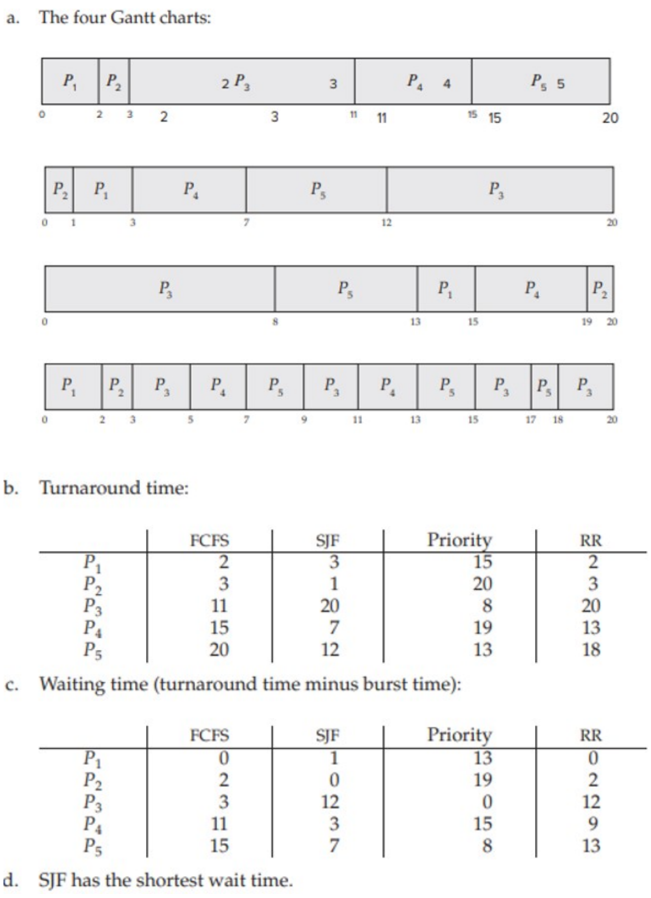
</td>
<td>

Consider the following set of processes, with the length of the CPU burst given in milliseconds:

<table>
<tr><th>Process</th><th>Burst Time</th><th>Priority</th></tr>
<tr><td>P1</td><td>2</td><td>2</td></tr>
<tr><td>P2</td><td>1</td><td>1</td></tr>
<tr><td>P3</td><td>8</td><td>4</td></tr>
<tr><td>P4</td><td>4</td><td>2</td></tr>
<tr><td>P5</td><td>5</td><td>3</td></tr>
</table>

The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.
a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time of each process for

</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3.6.2</td>
</tr>
</table>

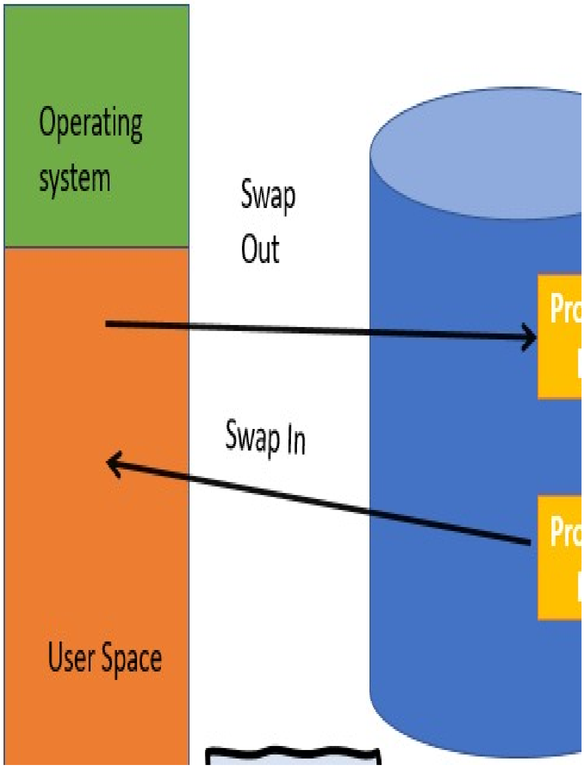
<table>
<tr><td></td><td><b>each of these scheduling algorithms?</b>
<b>d. Which of the algorithms results in the minimum average waiting time (over all processes)?</b>
<b>Answer :</b>

a. The four Gantt charts:

P1
P2
2 P3
3
P4
4
P5
5
0
2
3
2
3
11
11
15
15
20
P2
P1
P4
P5
P3
0
1
3
7
12
20
P3
P5
P1
P4
P2
0
8
13
15
19
20
P1
P2
P3
P4
P5
P3
P4
P5
P3
P5
P3
0
2
3
5
7
9
11
13
15
17
18
20

b. Turnaround time:
<table>
<tr><th></th><th>FCFS</th><th>SJF</th><th>Priority</th><th>RR</th></tr>
<tr><td>P_1</td><td>2</td><td>3</td><td>15</td><td>2</td></tr>
<tr><td>P_2</td><td>3</td><td>1</td><td>20</td><td>3</td></tr>
<tr><td>P_3</td><td>11</td><td>20</td><td>8</td><td>20</td></tr>
<tr><td>P_4</td><td>15</td><td>7</td><td>19</td><td>13</td></tr>
<tr><td>P_5</td><td>20</td><td>12</td><td>13</td><td>18</td></tr>
</table>
c. Waiting time (turnaround time minus burst time):
<table>
<tr><th></th><th>FCFS</th><th>SJF</th><th>Priority</th><th>RR</th></tr>
<tr><td>P_1</td><td>0</td><td>1</td><td>13</td><td>0</td></tr>
<tr><td>P_2</td><td>2</td><td>0</td><td>19</td><td>2</td></tr>
<tr><td>P_3</td><td>3</td><td>12</td><td>0</td><td>12</td></tr>
<tr><td>P_4</td><td>11</td><td>3</td><td>15</td><td>9</td></tr>
<tr><td>P_5</td><td>15</td><td>7</td><td>8</td><td>13</td></tr>
</table>
d. SJF has the shortest wait time.</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>17.a</b></td><td><b>Memory management is in urge to enable more usable memory than held by the computer hardware. There are times when physical memory will be allocated and a process needs additional memory, in such cases describe how these issues can be handled and help the memory management for back storage for further execution, also claim the benefits of the techniques.</b>

<b>Swapping</b> is a method in which the process should be swapped temporarily from the main memory to the backing store. It will be later brought back into the memory for continue execution.

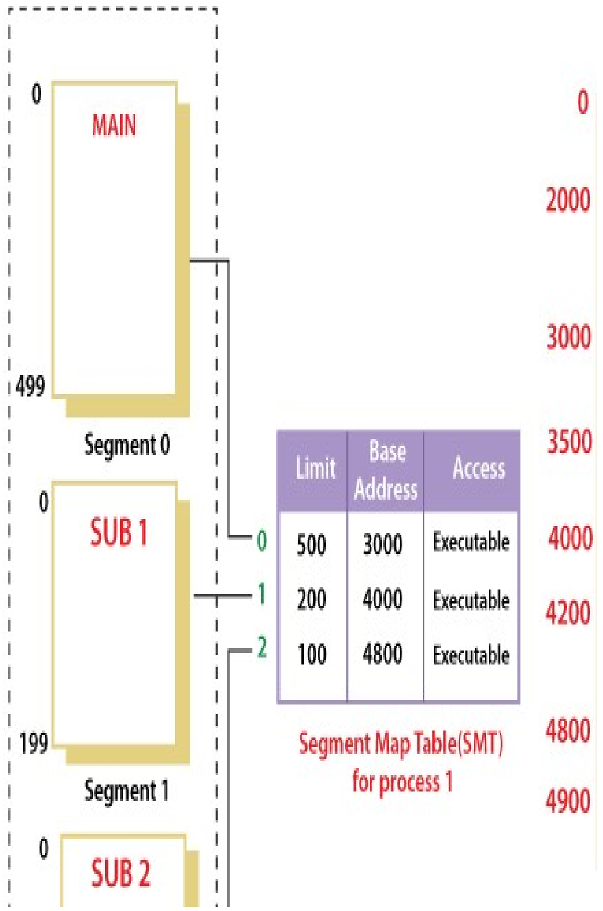
Backing store is a hard disk or some other secondary storage device that should be big enough in order to accommodate copies of all memory images for all users. It is also capable of offering direct access to</td><td><b>10</b></td><td><b>3</b></td><td><b>3</b></td><td><b>1</b></td><td><b>1.7.1</b></td></tr>
</table>

	these memory images.  Operating system Swap Out Swap In User Space Pro Pro Benefits of Swapping Here, are major benefits/pros of swapping: - It offers a higher degree of multiprogramming. - Allows dynamic relocation. For example, if address binding at execution time is being used, then processes can be swap in different locations. Else in case of compile and load time bindings, processes should be moved to the same location. - It helps to get better utilization of memory. - Minimum wastage of CPU time on completion so it can easily be applied to a priority-based scheduling method to improve its performance.					
17.b	**Suppose a 16 bit address is used with 4 bits for the segment number and 12 bits for the segment offset so the maximum segment size is 4096 and the maximum number of segments that can be refereed is 16.Elloborate how the Translation of Logical address into physical address been**	**10**	**3**	**3**	**1**	**1.7.1**

mapped by segment table method.

When a program is loaded into memory, the segmentation system tries to locate space that is large enough to hold the first segment of the process, space information is obtained from the free list maintained by memory manager. Then it tries to locate space for other segments. Once adequate space is located for all the segments, it loads them into their respective areas.

The operating system also generates a segment map table for each program.



Limit	Base Address	Access
500	3000	Executable
200	4000	Executable
100	4800	Executable

Segment Map Table(SMT)
for process 1

With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.

The **Segment number** is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid. In the case of valid addresses, the base address of the segment is added to the offset to get the physical

	address of the actual word in the main memory. The above figure shows how address translation is done in case of segmentation					

Question Paper Setter

Approved by Audit Professor/
Course Coordinator