Theory Assignment 3

By

Keyur Patel

40154883

A report submitted in partial fulfilment of the requirements of COMP346

Concordia University

March, 2022

**Answer 1:**

i.

a) A relocatable program is a program that can be executed without any changes, and from any address from the memory.

b) If the data has a relative address, then it is relocatable. However, if it an absolute address (cannot be changed), then it would not be relocatable.

c) If the programs were not relocatable, this would mean that many programs would be running in the same memory location. This would cause the execution of the programs to be slow. Then, they would find an open memory address and be executed there.

ii.

Small sizes pages means that there will be less wasted memory versus the large page sizes. The waste of memory in the large page size may cause internal fragmentation. There will be more page faults for smaller page size compared to larger page sizes.

iii.

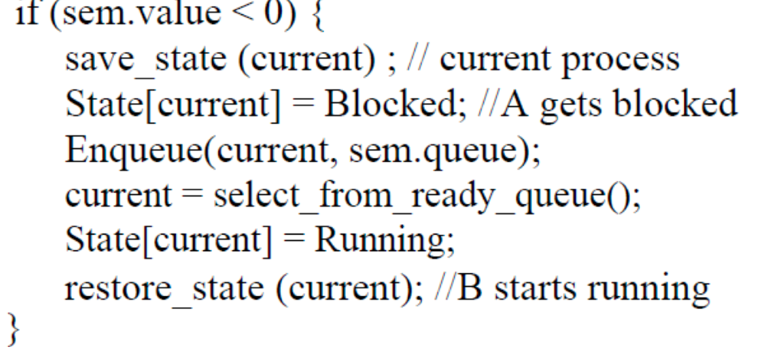
There will be no external fragmentation. It also allows for easy swapping between disk and memory because both frames and pages are of the same size.

iv.

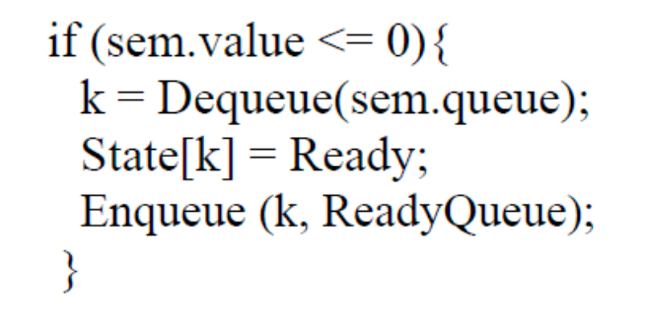
There is no internal fragmentation. In terms of the paging mechanism, there will be less memory space occupied in the segmentation mechanism.

**Answer2:**

1. Critical section for wait():



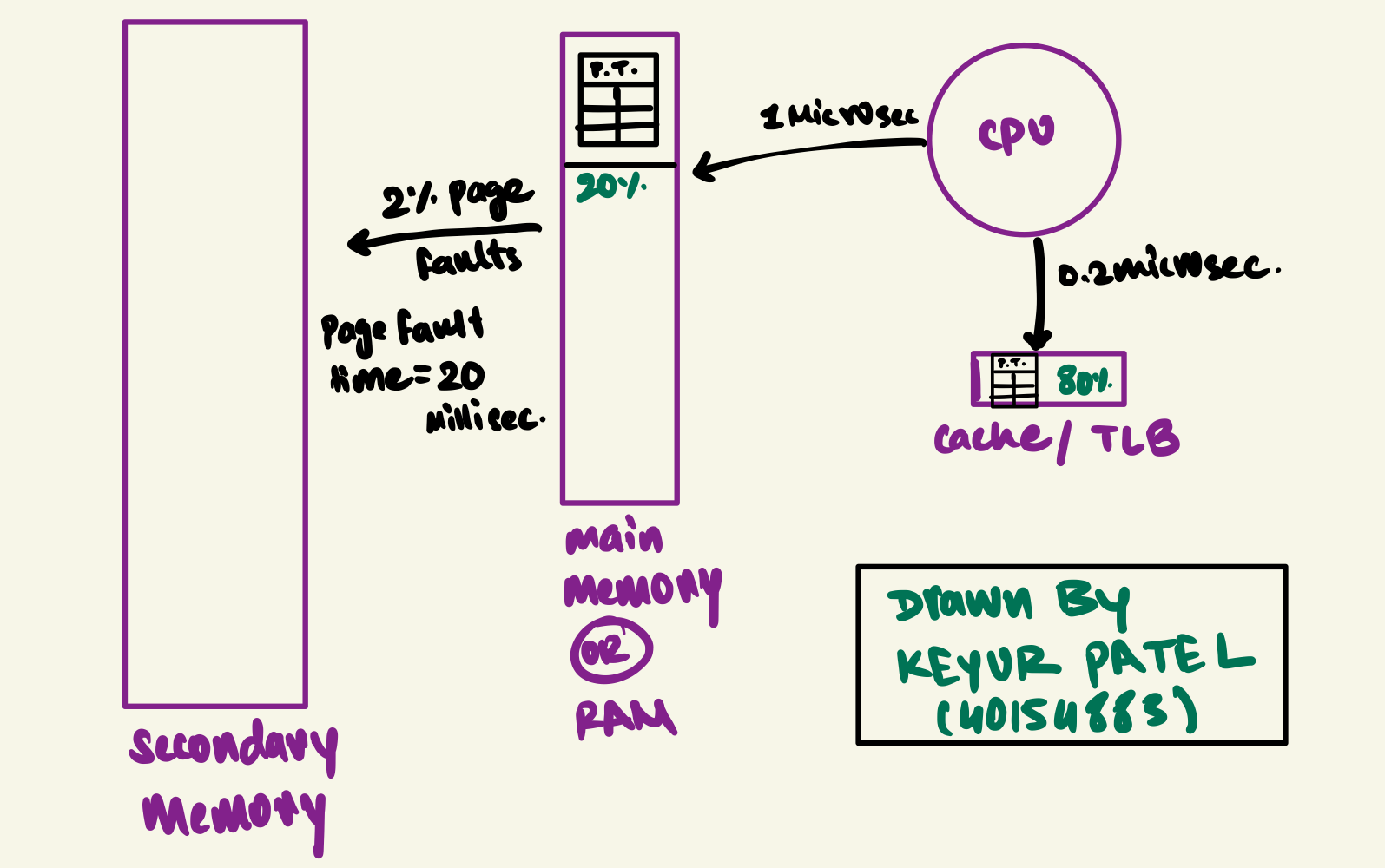
Critical section for signal():



b) Let sem.value = 1. If we go through the wait method, the if statement is not executed and it automatically goes to enable interrupts. After this, it would go to the signal() method, where it would first disable interrupts and then change the value of sem.value to 2. This will then skip the if statement and then enable interrupt. This will just signal without execute the wait since there is nowhere in the code where the value of the sem is reset to 0.

c) No it will not be executed. We need to be able to switch interrupts on a hardware level, which is used to make the operation atomic. Hence, if B is disabled, we would not have synchronisation and it will not be atomic.

**Answer 3:**

****

**Case 1:**

Hit in TLB = TLB hit rate \*(TLB access time + memory access time) = 0.8(0.2+1)= 0.96 microseconds.

**Case 2:**

Miss in TLB but hit in Main Memory = Page Hit Rate\*TLB miss rate\*(TLB access time + memory access time for Page table + memory access time for page )

= 0.98\*0.2(0.2+1+1) = 0.4312 microseconds.

**Case 3:**

Miss in TLB, miss in Main Memory but hit in Secondary memory by Page Fault.

= Page Miss Rate\*TLB miss rate\*(Page Fault time + TLB access time + memory access time for Page table + memory access time for page)

= 0.02\*0.2(2000+0.2+1+1) = 0.04 (2002.2) = 80.088 microseconds.

**Total effective memory access time:** Case1+Case2+Case3

= (0.96 + 0.4312 +80.088) = 81.48 microseconds.

**Answer 4:**

**a)**

**Least Recently Used Memory Representation**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FRAME | **0** | **1** | **2** | **0** | **1** | **2** | **0** | **1** | **2** | **3** | **6** | **7** | **6** | **7** | **0** | **1** | **2** | **3** | **4** |
| **0** | 0\* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3\* | 3 | 3 | 3 | 3 | 0\* | 0 | 0 | 3\* | 3 |
| **1** |  | 1\* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6\* | 6 | 6 | 6 | 6 | 1\* | 1 | 2 | 4\* |
| **2** |  |  | 2\* | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7\* | 7 | 7 | 7 | 7 | 2\* | 2 | 2 |

LRU Page Fault = 11

**b)**

**Belady Optimal Algorithm Memory Representation**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FRAME | **0** | **1** | **2** | **0** | **1** | **2** | **0** | **1** | **2** | **3** | **6** | **7** | **6** | **7** | **0** | **1** | **2** | **3** | **4** |
| **0** | 0\* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2\* | 2 | 2 |
| **1** |  | 1\* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7\* | 7 | 7 | 7 | 1\* | 1 | 1 | 1 |
| **2** |  |  | 2\* | 2 | 2 | 2 | 2 | 2 | 2 | 3\* | 6\* | 6 | 6 | 6 | 6 | 6 | 6 | 3\* | 4\* |

Belady Optimal Algorithm Page Fault = 10

**c)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FRAME | **0** | **1** | **2** | **0** | **1** | **2** | **0** | **1** | **2** | **3** | **6** | **7** | **6** | **7** | **0** | **1** | **2** | **3** | **4** |
| **0** | 0\* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6\* | 6 | 6 | 6 | 6 | 6 | 2\* | 2 | 2 |
| **1** |  | 1\* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 7\* | 7 | 7 | 7 | 7 | 7 | 3\* | 3 |
| **2** |  |  | 2\* | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |  |  | 0\* | 0 | 0 | 0 | 4\* |
| **3** |  |  |  |  |  |  |  |  |  | 3\* | 3 | 3 | 3 |  |  | 1\* | 1 | 1 | 1 |

**Working Set Model** with a window sizeΔ=3

Working Set Model Page Fault = 11

**Answer 5 :**

1. The biggest advantage of this implementation would be that the speed of accessing the memory will get faster i.e., memory access time will decrease.
2. But the disadvantage would be that more memory will be consumed on the CPU to use page table compared to what we are spending just to store it on CPU.

**Answer 6:**

1. An advantage of the global page replacement algorithm would be that as the working set of the process grows the number of framed allocated can be increased and vice versa true for process shrinking.
2. But the disadvantage is this algorithm replaces the pages regard to the process it belongs which will eventually lead to thrashing. Thrashing is more of a problem with the global page replacement algorithm over the local.

**Answer 7:**

i) TPF > TFS : We would have less page faults than service page faults, which would lead to the best multiprogramming result.

ii) TPF < TFS : We would have more page faults than service page faults, which would lead to the worst multiprogramming result.

iii) TPF = TFS: We would have a relatively stable system based on the number of faults being relatively close to each other.

**Answer 8:**

Advantages: There will be no need for the File I/O to manually close the file.

Disadvantages: There are a very high changes of privacy leak and deadlock since the files will not go though File I/O.

**Answer 9:**

1. In a Non Pre-emptive Scheduling , once a process is scheduled for execution or given to CPU for execution, the given process keeps the CPU until it finishes its execution or by switching to waiting state.

Whereas, in Pre-emptive Scheduling , once a process is scheduled it switches from running to ready state or from waiting to ready state.

Strict non-pre-emptive scheduling is unlikely to be used in a computer system as it will not provide priority, fairness and most importantly better system performance compared to pre-emptive.

1. If the quantum size is low it would result in more contact switching and low overall system performance will go down.

If the quantum size is high, then it would result in less context switching, which would lead to increase in response time and eventually the degree of multiprogramming will go down.

**Answer 10:**

The first benefit of the queue is that it allows processes to switch between queues. Another benefit is that the higher the priority of the queue priority, the more CPU time is available. The longer a process takes to complete , the lower its priority, allowing other processes to run concurrently on the CPU. As a result, it will prevent hunger on our program, but it will also destroy fairness because low-priority processes may run concurrently with high-priority process.

**Answer 11:**

**a)**

FCFS scheduling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P0(20) | P1(35) | P2(56) | P3(63) | P4(75) |

Non-pre-emptive SJF Scheduling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P3(7) | P4(19) | P1(34) | P0(54) | P2(75) |

Non-pre-emptive priority scheduling: lower number means higher priority

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1(15) | P4(27) | P0(47) | P2(68) | P3(75) |

Pure Round-Robin scheduling with the quantum = 3 : Find the highest possible multiple of 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Service time** | **Highest multiple of 3** | **Remainder** |
| **P0** | 20 | 18 | 2 |
| **P1** | 15 | 15 | 0 |
| **P2** | 21 | 21 | 0 |
| **P3** | 7 | 6 | 1 |
| **P4** | 12 | 12 | 0 |

**b) Waiting Time: -**

FCFS Scheduling: - P0 = 0, P1 = 20, P2 = 35, P3 = 56, P4 = 63

Non-preemptive SJF scheduling: - P0 = 34, P1 = 19, P2 = 54, P3 = 0, P4 = 7

Non-preemptive priority scheduling: - P0 = 27, P1 = 0, P2 = 47, P3 = 68, P4 = 15

Round-Robin scheduling: - P0 = 69, P1 = 57, P2 = 72, P3 = 71, P4 = 51

**c)Response Time: -**

FCFS Scheduling: - P0 = 0, P1 = 20, P2 = 35, P3 = 56, P4 = 63

Non-preemptive SJF scheduling: - P0 = 34, P1 = 19, P2 = 54, P3 = 0, P4 = 7

Non-preemptive priority scheduling: - P0 = 27, P1 = 68, P2= 47, P3 = 75, P4 = 15

Round-Robin scheduling: - P0 = 0, P1 = 3, P2 = 6, P3 = 9, P4 = 12

**d)Turn-Around Time: -**

FCFS Scheduling: - P0 = 20, P1 = 35, P2 = 56, P3 = 63, P4 = 75

Non-preemptive SJF scheduling: - P0 = 53, P1 = 34, P2 = 75, P3 = 7, P4 = 19

Non-preemptive priority scheduling: - P0 = 47, P1 = 15, P2= 68, P3 = 75, P4 = 27

Round-Robin scheduling: - P0 = 71, P1 = 60, P2 = 75, P3 = 72, P4 = 54