**Lab 5**

Tyler Holland (tyhollan)

1. I would expect the CPU to be idle 1/2 of the time. Since each program has its own section of memory, there is very little downtime to switch between processes. Because each program is blocked for 1/2 of its running time, I would expect that the CPU would be idle for the same ratio of time, since the I/O operations are distributed randomly with even probability.
2. 1. 20 = 8020 in physical memory
   2. 4100 = 4100 in physical memory
   3. 8300 = 24300 in physical memory
3. Bits in address: 9 (top level) + 11 (2nd level) + x (offset) = 32, x = 12. Since offset is 12 bits, pages are 4KB, and there are a total of 29+11 = 220 pages.
4. 1. It would replace page 0, because it is both not referenced and not modified, the lowest class in an NRU algorithm
   2. It would replace page 2, since it was loaded first. Page 2 was loaded at time 120, making it the first in, and currently oldest process which would be replaced with FIFO
   3. It would replace page 0. Page 0 was added at time 126, and last referenced at time 279, for a difference of 153 clock ticks. This is the page with the largest gap between add and last reference times, so LRU would replace it.
   4. It would replace page 0. Although page 2 is the oldest process, its R value is 1 so second chance will not replace it. The second oldest process is page 0, whose R value is 0 and therefore qualifies for replacing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Page Number** | **Page Frame 1** | **Page Frame 2** | **Page Frame 3** | **Page Frame 4** |
| 0 | **0** | - | - | - |
| 7 | 0 | **7** | - | - |
| 1 | 0 | 7 | **1** | - |
| 3 | 0 | 7 | 1 | **3** |
| 2 | **2** | 7 | 1 | 3 |
| 7 | 2 | 7 | 1 | 3 |
| 2 | 2 | 7 | 1 | 3 |
| 0 | 2 | **0** | 1 | 3 |
| 1 | 2 | 0 | 1 | 3 |
| 3 | 2 | 0 | 1 | 3 |
| **Final** | 2 | 0 | 1 | 3 |
| **Total Page Faults:** | **6** | Page fault = Bold number |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Page Number** | **Page Frame 1** | **Page Frame 2** | **Page Frame 3** | **Page Frame 4** |
| 0 | **0** | - | - | - |
| 7 | 0 | **7** | - | - |
| 1 | 0 | 7 | **1** | - |
| 3 | 0 | 7 | 1 | **3** |
| 2 | **2** | 7 | 1 | 3 |
| 7 | 2 | 7 | 1 | 3 |
| 2 | 2 | 7 | 1 | 3 |
| 0 | 2 | 7 | **0** | 3 |
| 1 | 2 | 7 | 0 | **1** |
| 3 | **3** | 7 | 0 | 1 |
| **Final** | 3 | 7 | 0 | 1 |
| **Total Page Faults:** | **8** | Page fault = Bold number |  |  |

1. Since we are doubling the size of memory, the mean interval between page faults should double. This means that if we got 15000 page faults in 60 seconds before, we should only get 7500 page faults with double the memory. This also means that it should take (2000 microseconds) \* (7500 page faults) less time to complete.

Math: (7500 \* 2000ms) = 15 seconds. 60s - 15s = 45s.

New Running Time with double memory: **45 seconds**