MP2 Bonus Student ID：B09902046 Name：何翊豪

I. Report: 4.4.1 Pros and Cons of Demand Paging (5%)

1. Memory Utilization：Memory can be utilized more efficiently, because OS only load the pages which are demanded. (Benefit)

2. External fragmentation：Since now we have a fixed size page, during swapping or loading pages, every page can be exactly fitted. Hence, there is no external fragmentation anymore. (Benefit)

3. Input/Output for demand paging：Because now we use demand paging, we don't have to load every page from disk while IO anymore, we can have less IO and only need demand page. (Benefit)

4. Size of physical memory：Since we only load the pages demanded, we can run processes with bigger size than the actual physical memory size. Thus, the size of physical memory is no longer a primary restriction to the size and number of processes we can run. (Benefit)

5. Share the pages with demand paging：Due to demand paging, we can easily share pages among different processes (map to same page), so we can reduce the actual memory we need to avoid having two identical pages. (Benefit)

6. Overheads due to interrupts, page tables access, and memory access time in demand paging：If there's a page fault happened, we need to trap in OS, bring the demand page back into the physical memory. Thus, page faults are the main reason why demand paging increase the overhead. There still have some other overheads such as TLB access time and memory access time. (Drawback)

II. Report: 4.4.2 Effective Memory Access Time Analysis (5%)

Question 1：

By the formula of single level paging EMAT, we know that when we miss the hit (miss ratio), we need to take 2 times of memory access time, one for page table, and the other for accessing the page.

Thus, if we have k level paging, it means when we miss the hit, we need to take k+1 times of memory access time, because we need to access k page tables.

Therefore, the formula of "Multi-level Paging" becomes：

$$EMAT = P \times (TLB\ access\ time + memory\ access\ time) + \\ (1 - P) \times (TLB\ access\ time\ + (\boldsymbol{k+1}) \times memory\ access\ time)$$

(Next Page)

Question 2：

By using the formula derived by Question 1, we can let the maximum memory access time be $x\ nanoseconds$, and have：

$$180 = 0.8 \times (20 + x) + (1 - 0.8) \times (20 + (3 + 1)x)$$

=> $$180 = 16 + 0.8x + 0.2 \times (20 + 4x)$$

=> $$180 = 16 + 0.8x + 4 + 0.8x$$

=> $$180 = 20 + 1.6x$$

=> $$160 = 1.6x$$

=> $$x = 100$$

Hence, the maximum memory access time could be design in this system is **100 nanoseconds**.