Q4.2

When a program might suffer from **frequent page faults** or **has to wait for other system events**, a multithreaded solution would perform better even on a single-processor system.

Since a single-threaded process cannot perform useful work when a page fault takes place while in the multithreaded solution, another kernel can be switched in to use the interleaving time.

Q4.4

使用 multuprocessor system 和 single-processor system 對於 multithreaded 的 multiple user-level threads 來說，並沒有效能上的差異。

因為 multiple user-level threads 本身就無法在 multiprocessor 的系統中同時使用不同 process，也就是，從 OS 的角度只看到 single process，所以就算有其他 processor，OS 也不會 schedule 其他 thread 在不同 processor 上做別的工作

Q4.13

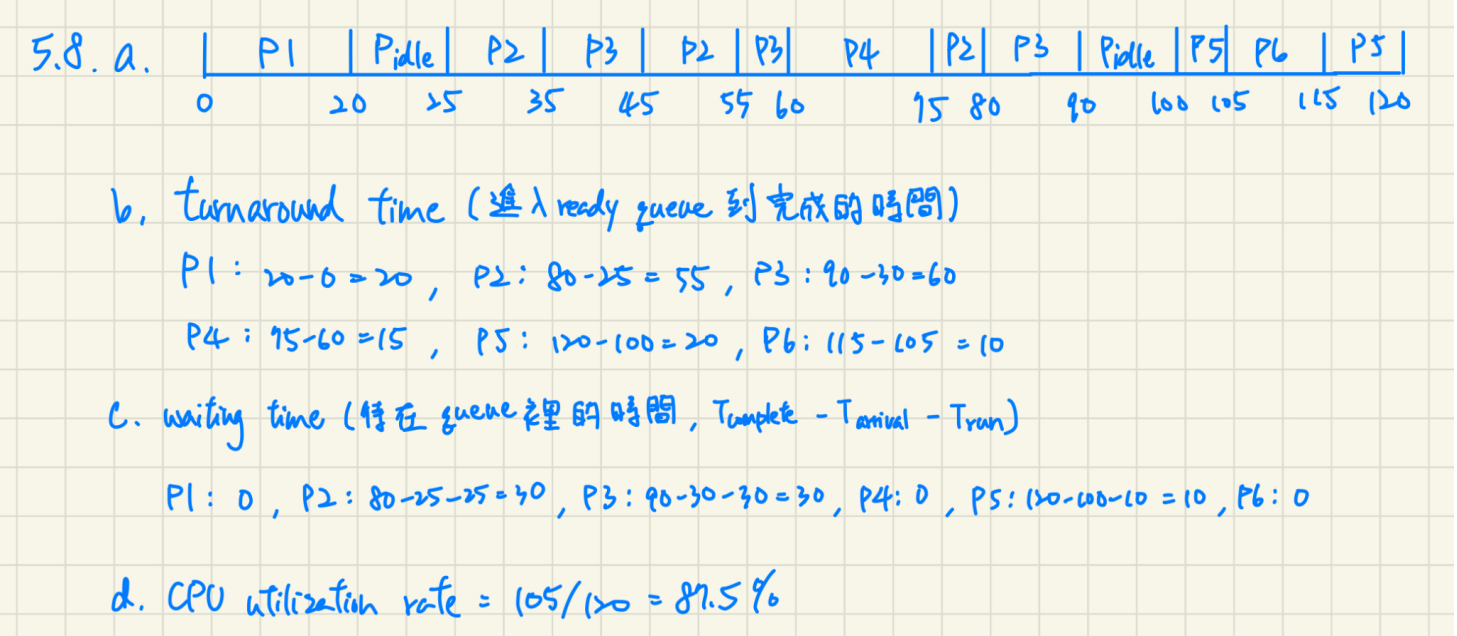
a. some of the processors would remain idle since the scheduler maps only kernel threads to processors and not user-level threads to processors.
b. it is possible that all of the processors might be utilized simultaneously. However, when a kernel-thread blocks inside the kernel(ex: page fault or while invoking system calls), the corresponding processor would remain idle.
c. a blocked kernel thread could be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.

Q5.6

CPU bound process.

Since they are rewarded with a longer time quantum and priority boost whenever they consume an entire time quantum. This scheduler does not penalize I/O-bound processes as they are likely to block for I/O before consuming their entire time quantum, but their priority remains the same.

Q5.8



Q5.10

a. FIFS: No. 因為按照進入queue的順序執行並不會將某個 job 一直排在最後而不執行
b. SJF: Yes. 若有一 job 時間一直大於存在 waiting queue 中其他 job 的時間，則不會挑選到其來執行
c. RR: No. 因為平等的執行每一個 job，並不會將某個 job 一直排在最後而不執行
d. Priority: Yes. 若有一 job priority 一直小於存在 waiting queue 中其他 job 的priority，則不會挑選到其來執行

Q5.15

a. FCFS: discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.
b. RR: treats all jobs equally(all jobs have equal burst time), so short jobs will be able to leave the system faster since they will finish first.
c. Multilevel feedback queues: work similar to the RR algorithm but also use FCFS. They discriminate favorably to short jobs.

Q6.4

If a user-level program is given the ability to disable interrupts, then it can disable the timer interrupt and prevent context switching from taking place, thereby allowing it to use the processor without letting other processes to execute.

Q6.10

This would be very similar to the changes made in the description of the semaphore. Associated with each mutex lock would be a queue of waiting processes. When a process determines the lock is unavailable, they are placed into the queue. When a process releases the lock, it removes and awakens the first process from the list of waiting processes.

Q6.11

1. short duration: spinlock.

There will be much less overhead if the process waits 3-4 cycles then performing two context switches (first for removing a blocked thread and placing a new thread to run, second for reversing this process).

2. long duration: mutex lock.

Reasons given for spinlock in the previous answer do not hold here.

3. a thread may be put to sleep while holding the lock: metex lock

The thread which can;t access the critical section will waste less cycles this way than if it was busy waiting.