1.9:

Direct memory access is used for high speed I/O devices in order to avoid increasing the CPU’s execution load.

•1.9: (a) How does the CPU interface with the device to coordinate the transfer?

啟用DMA控制器，由CPU給予匯流排資料傳輸的權限與需要傳輸的資訊，DMA控制器不須經過CPU而能在設備與記憶體之間進行資料傳輸。

•1.9: (b) How does the CPU know when the memory operations are complete?

CPU會收到來自DMA控制器發出的interrupt信號，並且先執行interrupt需要處理的程序，再回到CPU原本的處理狀態。

•1.9: (c) The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.

DMA控制器在傳輸資料時，如果CPU也同樣需要訪問記憶體，會導致衝突產生，CPU需要等DMA控制器處理完資料傳輸後才能訪問記憶體，會造成降低系統性能的可能。

•2.7: What are the two models of interprocess communication? What are the strengths and weakness of the two approaches?

Shared memory 與 Message-passing。

Shared memory:

優點：在同一個機器上實作時會比Message-passing還要有效率的執行程式。

缺點：因為使用到同一個記憶體，需要注意到在記憶體操作上不能有同時被不同程序訪問的情形發生。

Message-passing:

優點：比Shared memory還更好實作，不需要特別考慮記憶體使用問題。

缺點：執行時間上比Shared memory還久，因為時間耗費在建立連線方面。

•2.10: What is the main advantage of the *microkernel* approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

在microkernel新增服務不需要修改kernel，大部分的操作都會在user mode完成而非kernel mode，所以較為安全，並有更簡單的kernel設計能有較可靠的操作系統，不容易出錯。User program跟system services使用進程間的訊息溝通，而訊息是由OS傳送的。而缺點就是頻繁的利用OS發出的訊息作為程序與服務的溝通機制，OS的負擔較重。

•3.1: Describe the differences among short-term, medium-term, and long-term scheduling.

short-term: 最小化程序的等待時間，來提高系統效率。

medium-term: 最大化系統吞吐量，提高內存利用率。

long-term: 平衡系統負載，避免系統過度擁塞，提高系統的可用性。

3.11:

What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.

•3.11: (a) Synchronous and asynchronous communication.

Synchronous communication：簡單理解與實作，通訊效率高，能保證訊息的準確與一致性，透過blocking控制程序的流程。但有可能因為blocking而發生鎖死，並且要求雙方在同一時間通訊，產生通訊延遲影響程序的執行速度。  
Asynchronous communication：不要求雙方同時參與通訊，程序能同時執行，能透過callback函式達到非同步blocking。但實作比同步複雜，需要考慮同步與非同步競爭的通訊問題。

•3.11: (b) Automatic and explicit buffering

Automatic buffering：不需要考慮與管理記憶體，能根據資料自動調整記憶體大小，低頻率的網路與記憶體訪問。但無法非常精確的使用記憶體，且自動處理記憶體可能導致預期外的錯誤。

Explicit buffering：由自己定義與控制記憶體使用，且能減少非必要的浪費。缺點是撰寫程序方面要花心思，隨時注意記憶體訪問與使用的不合法。

•3.11: (c) Send by copy and send by reference

Send by copy：

Send by reference：

•3.11: (d) Fixed-sized and variable-sized messages