Chapter 12

輸入/輸出系統

章節目標

- 探討作業系統 I/O 子系統的結構
- 討論 I/O 硬體的原則和複雜性
- 解釋 I/O 硬體和軟體方面的效能

12.1 概 觀

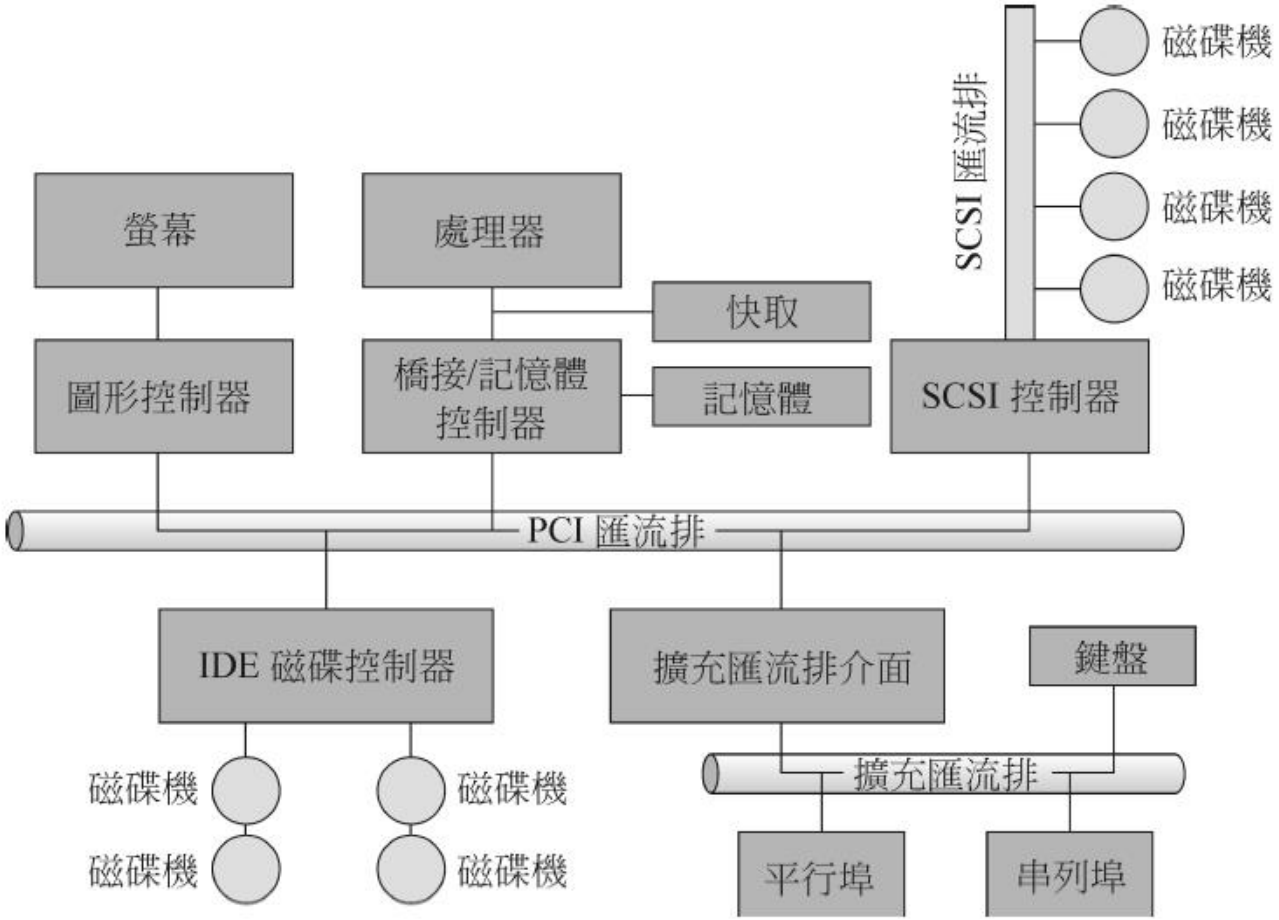
- 對於作業系統設計者而言，最關心的問題莫過於如何控制與電腦相連之裝置
  - 由於 I/O 裝置在功能與速度方面變化極大，因此需要許多不同的功能來加以控制
  - 這些方法構成核心的 I/O 子系統 (subsystem)，它將核心其它部份與管理 I/O 裝置可能涉及的複雜度區隔開來
- **裝置驅動程式** (device driver) 則代表一個與 I/O 子系統相通之統一裝置存取介面
  - 大部份與系統呼叫一樣，提供應用程式與作業系統間的標準介面

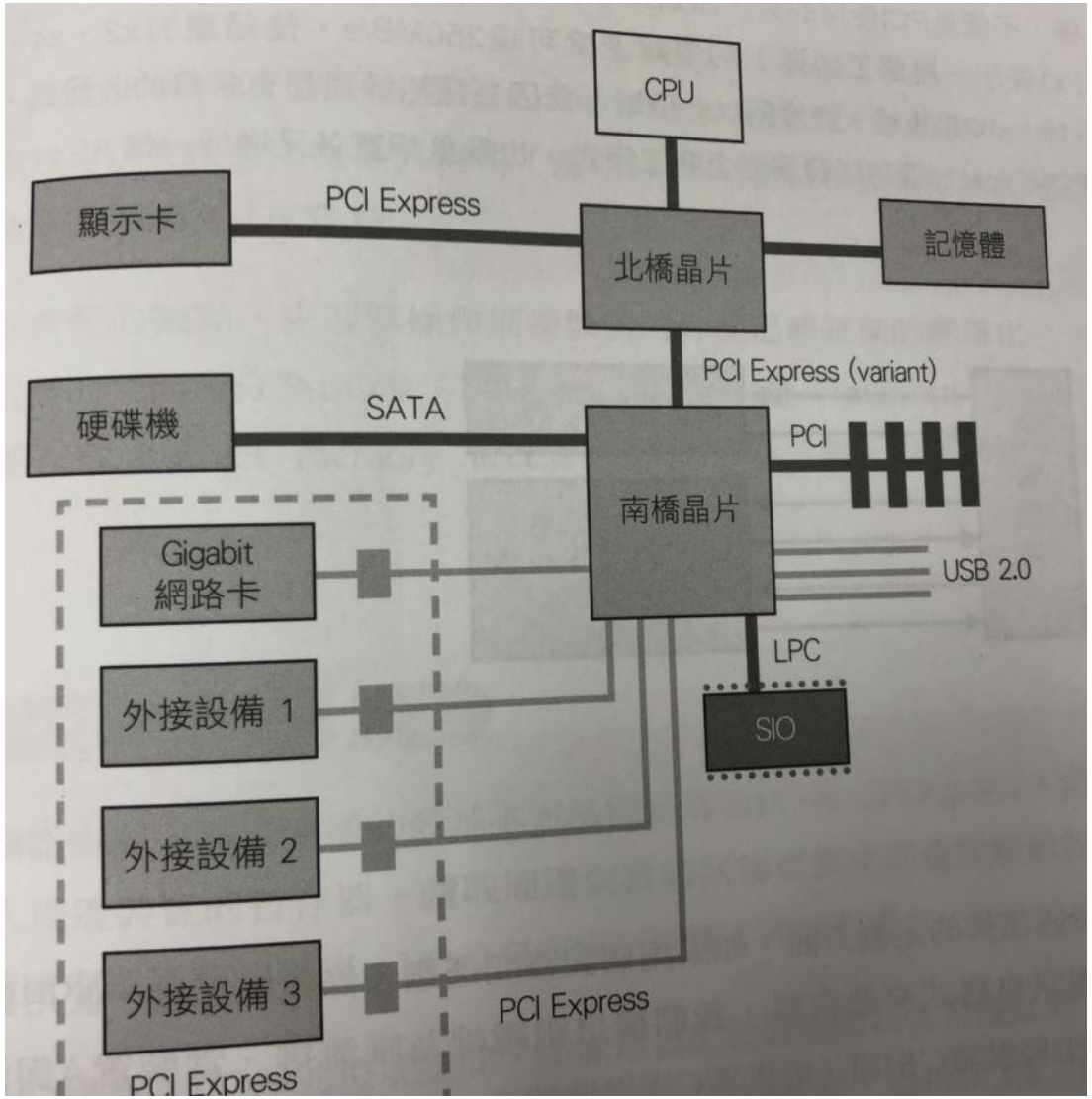
12.2 I/O 硬體

- 電腦負責操作許多不同種類的裝置，一般類型包含
  - 儲存裝置
  - 傳輸裝置
  - 人機介面裝置
- 裝置透過纜線或甚至在空中發送信號即可與電腦系統通信
  - 裝置和機器藉由所謂的連接點或**埠** (port)
  - **匯流排** (bus)：菊鏈 (daisy chain) 和共享直接存取
    - 典型的 PC 匯流排、PCI 匯流排 (PCI bus)
    - 擴充匯流排 (expansion bus) 連接慢速裝置

12.2 I/O 硬體

- **控制器** (controller) 為可操控連接埠、匯流排或裝置的電子零件組合
  - 光纖通道 (fibre channel, FC)
  - 主機匯流排轉接器(host bus adapter, HBA)
    - 連結到電腦的匯流排
  - 包含處理器、微程式碼及用來處理 FC 協定訊息的私有記憶體
    - 某些裝置擁有自己內建 (built-in) 的控制器


磁碟機
磁碟機
磁碟機
磁碟機
SCSI 匯流排
螢幕
處理器
快取
圖形控制器
橋接/記憶體
控制器
記憶體
SCSI 控制器
PCI 匯流排
IDE 磁碟控制器
擴充匯流排介面
鍵盤
磁碟機
磁碟機
磁碟機
磁碟機
擴充匯流排
平行埠
串列埠


CPU
顯示卡
PCI Express
北橋晶片
記憶體
PCI Express (variant)
硬碟機
SATA
PCI
南橋晶片
Gigabit
網路卡
USB 2.0
LPC
外接設備 1
SIO
外接設備 2
外接設備 3
PCI Express
PCI Express

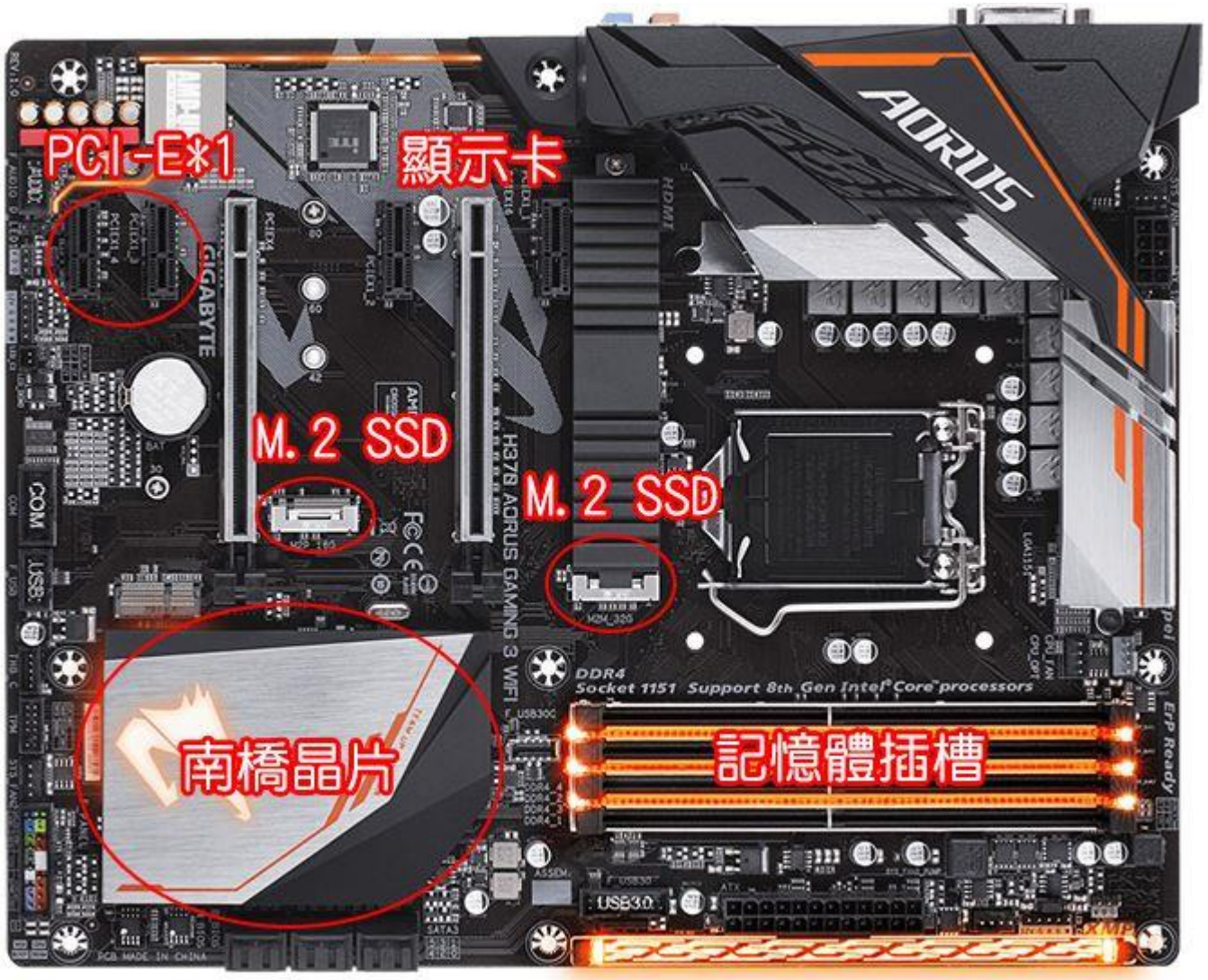
- PCI匯流排(PCI bus)
- 擴充匯流排(Expansion bus)
- SCCI匯流排
- PCI-X 提升到4.3G; PCI Express (PCIe)提升到16G

I/O 硬體(繼續)

- I/O指令控制裝置
- 裝置通常有暫存器可以讓裝置驅動程式放置指令、地址、寫入的資料、或指令執行後從暫存器讀取的資料
  - 資料輸入暫存器、資料輸出暫存器、狀態暫存器、控制暫存器
  - 通常 1-4 個位元組, 或FIFO緩衝區
- 裝置有地址，使用在
  - 直接 I/O 指令
  - **記憶體對映I/O**
    - 裝置資料和指令暫存器對映到處理器地址空間
    - 尤其是對於大型的地址空間 (繪圖)

在PC上的裝置I/O埠位址(部份資料)

I/O 位址範圍（十六進位）	裝　置
000-00F	DMA 控制器
020-021	中斷控制器
040-043	計數器
200-20F	遊戲控制器
2F8-2FF	串列埠（次要）
320-32F	硬碟控制器
378-37F	平行埠
3D0-3DF	圖形控制器
3F0-3F7	磁碟片裝置控制器
3F8-3FF	串列埠（主要）


PCI-E*1
顯示卡
M.2 SSD
M.2 SSD
南橋晶片
記憶體插槽
AORUS
GIGABYTE
H370 AORUS GAMING 3 WIFI
DDR4
Socket 1151 Support 8th Gen Intel® Core™ processors
USB30
PCB MADE IN CHINA

不同廠牌的CPU→插槽不同→會搭配不同的主機板

CPU插槽(CPU Socket)

主機板一定有個插槽放CPU，不同的主機板通常會有不同的CPU插槽型式，以支援不同的CPU，而即使插槽型式一樣，主機板也不一定都能支援，這跟CPU或主機板的世代交替，或是廠商自己劃分產品定位有關。

- Intel 主流為1151腳位 (Intel 是Core i
  - Intel Core i9-10900K
  - Intel Core i7-10700K
  - Intel Core i5-10600K
  - Intel Core i5-10400
  - Intel Core i3-10100

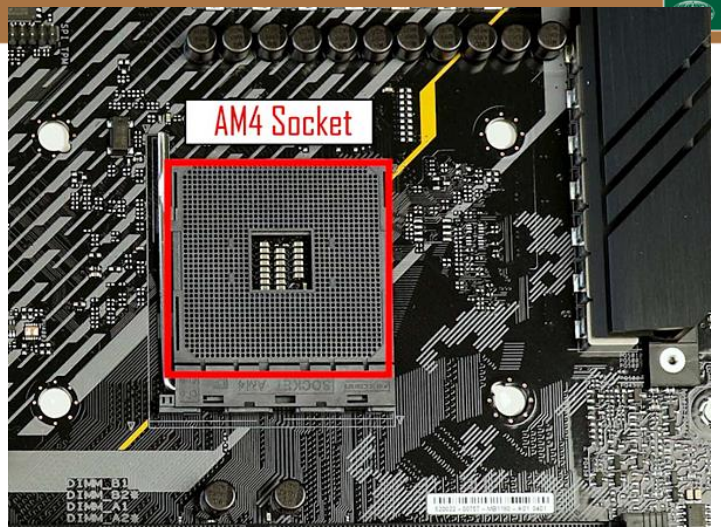
- AMD 主流為AM4 腳位 (AMD是Ryz
  - 12核3900XT 主頻4.1
  - 8核3800XT 主頻4.2
  - 6核3600XT 主頻4.0

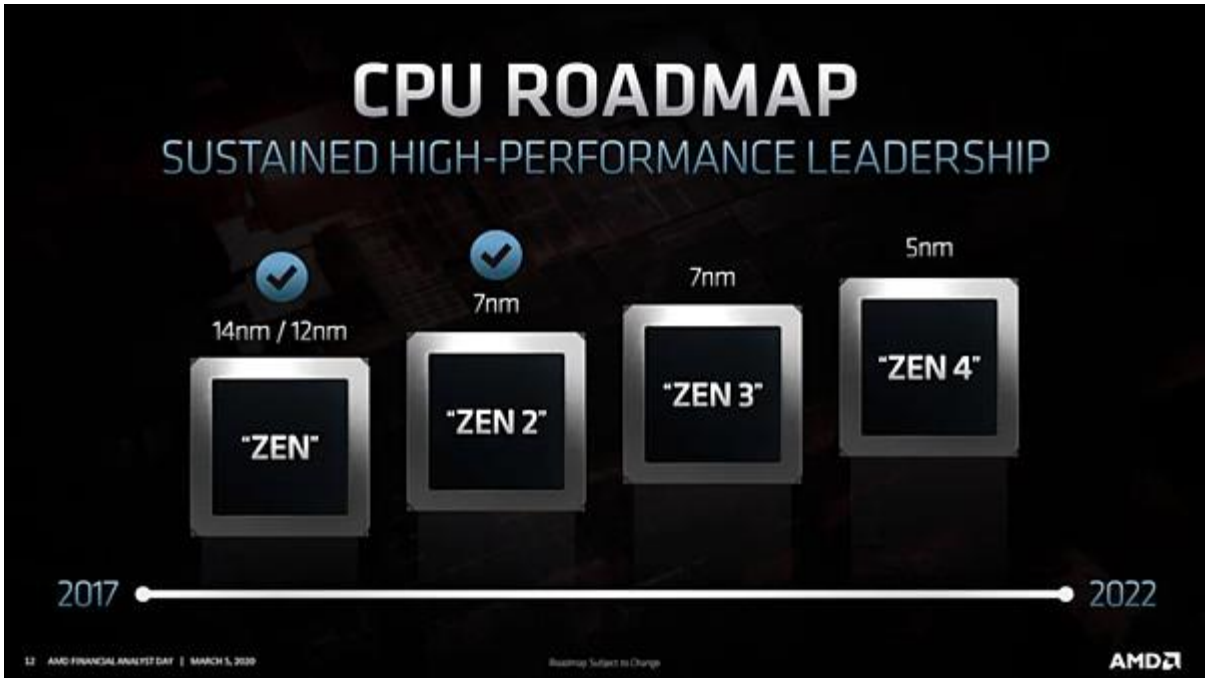
Desktop Platform – Consumer Roadmap (LGA)

		2019 Q2	2019 Q3	2019 Q4	2020 Q1 / Q2
CORE i9 9th Gen	P2K	i9-9900K	i9-9900K	i9-9900K	Comet Lake-S
	P2	i9-9900	i9-9900	i9-9900	
CORE i7 9th Gen	P1K	i7-9700K	i7-9700K	i7-9700K	
	P1	i7-9700	i7-9700	i7-9700	
CORE i5 9th Gen	MS2K	i5-9600K	i5-9600K	i5-9600K	
	MS2	i5-9600	i5-9600	i5-9600	
	MS1	i5-9500	i5-9500	i5-9500	
	MS0	i5-9400	i5-9400	i5-9400	
CORE i3 9th Gen	T3K	i3-9350K	i3-9350K	i3-9350K	
	T3	i3-9320	i3-9320	i3-9320	
	T2	i3-9300	i3-9300	i3-9300	
	T1	i3-9100	i3-9100	i3-9100	

CFL-S Refresh | Comet Lake-S | Unlocked


RYZEN
AMD Ryzen 4000
AMD
RYZEN
7


AM4 Socket

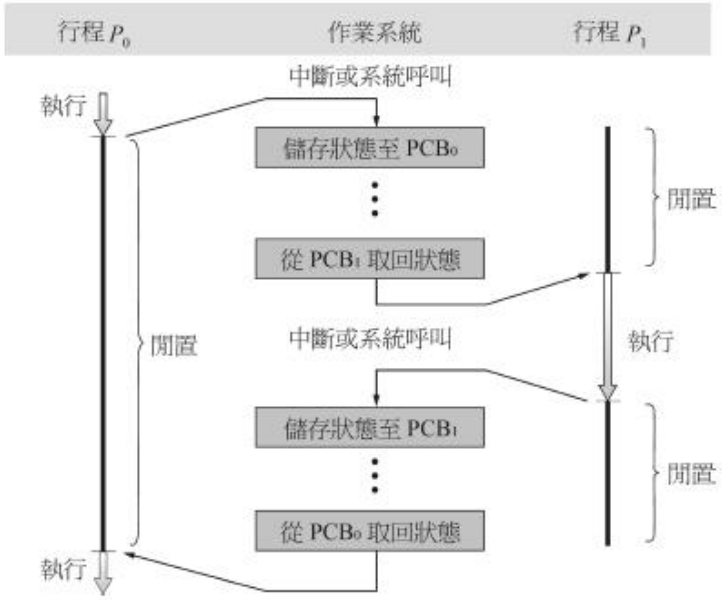

CPU ROADMAP
SUSTAINED HIGH-PERFORMANCE LEADERSHIP
14nm / 12nm
"ZEN"
7nm
"ZEN 2"
7nm
"ZEN 3"
5nm
"ZEN 4"
2017
2022
AMD

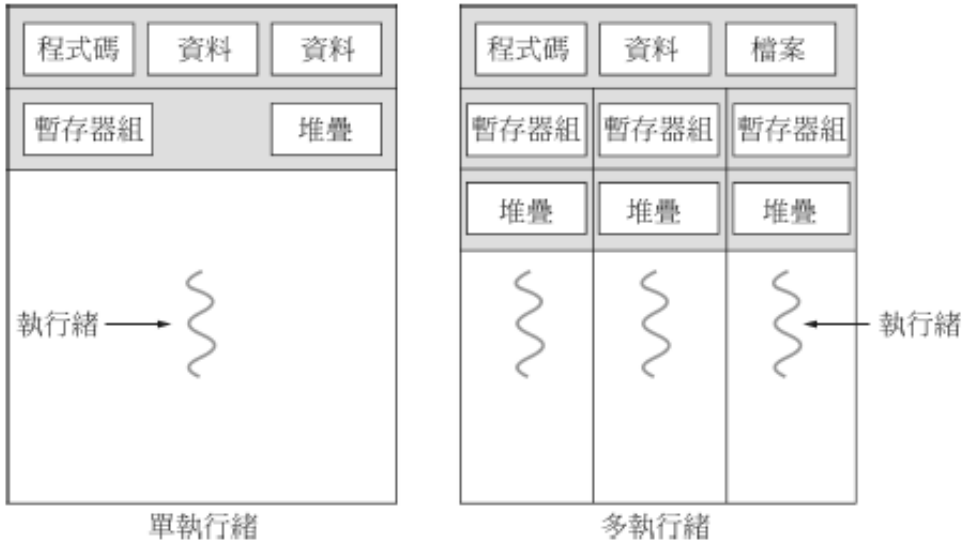
CPU散熱

塔扇

下吹式(原廠風扇)

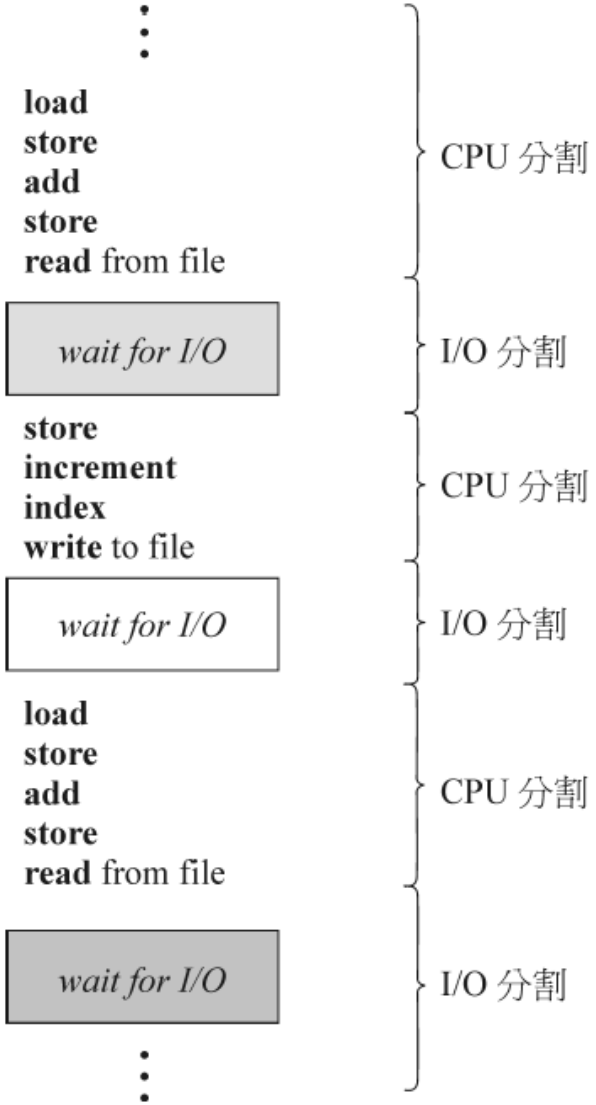
- CPU在行程間來回運轉狀況





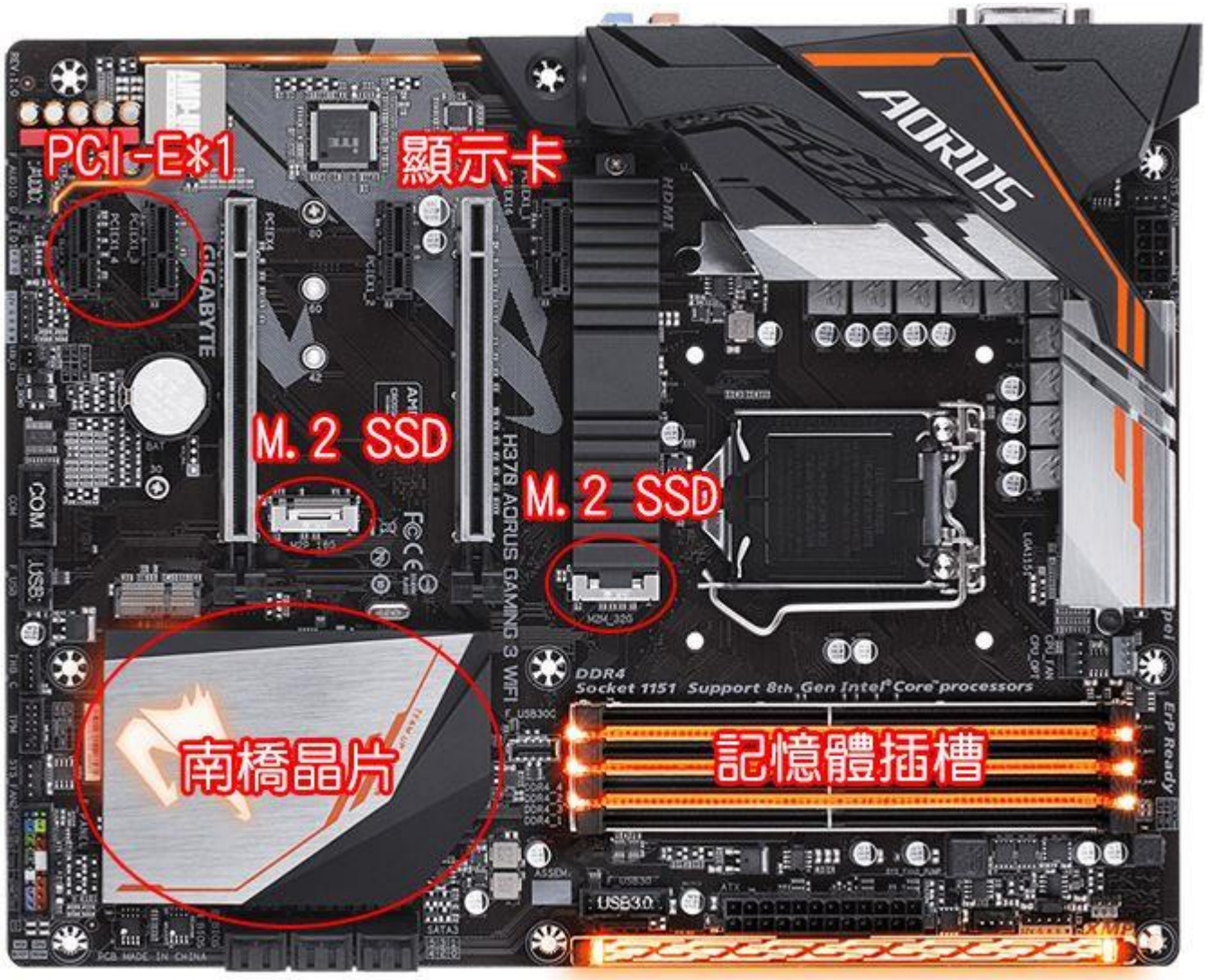
- **單執行緒和多執行緒的行程**

排班演算法最佳化的原則



- 最佳的CPU使用率
- 最大產量
- 最短回復時間
- 最短等候時間
- 最短反應時間

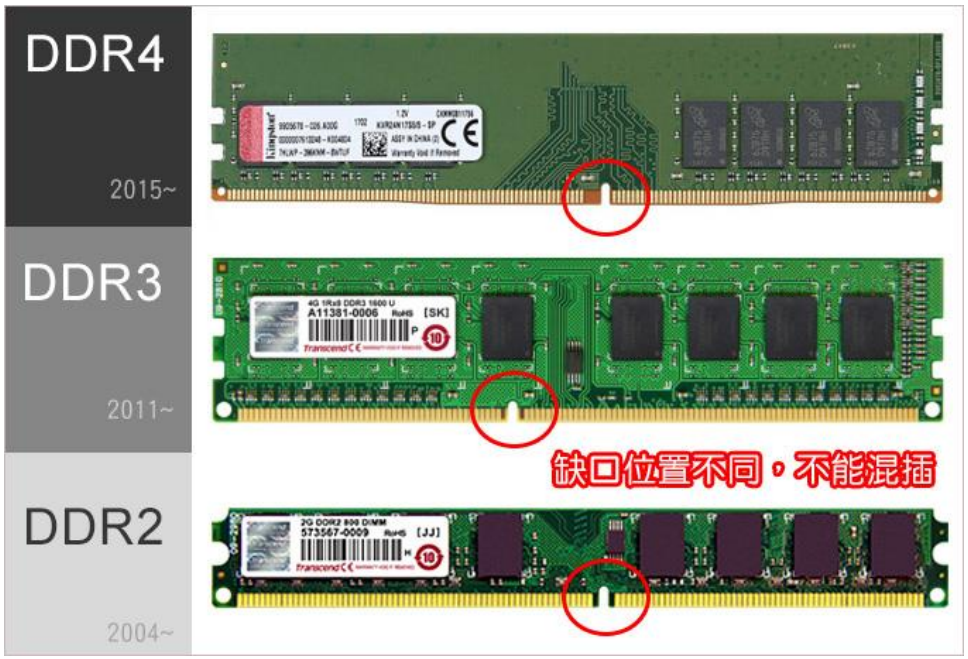
- 先來先做(FCFS)排班演算法
- 最短工作先做(SJF)排班演算法
- 優先權排班演算法
- 依序循環(RR)演算法

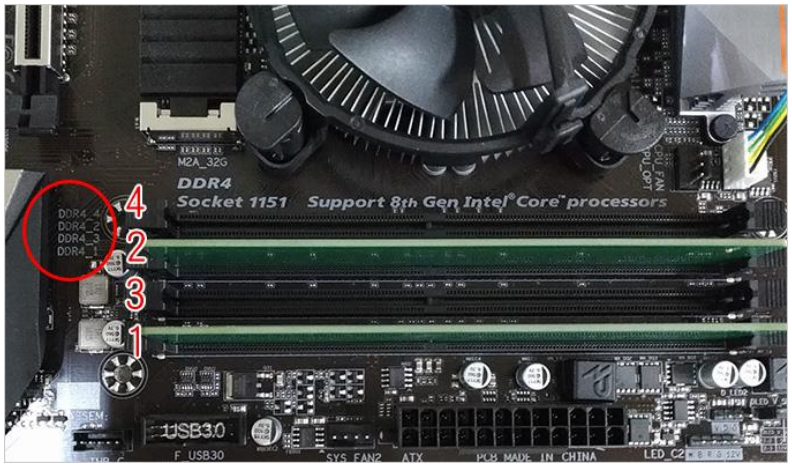

PCI-E*1
顯示卡
M.2 SSD
M.2 SSD
南橋晶片
記憶體插槽
AORUS
GIGABYTE
H370 AORUS GAMING 3 WIFI
DDR4
Socket 1151 Support 8th Gen Intel® Core™ processors
USB30
PCB MADE IN CHINA

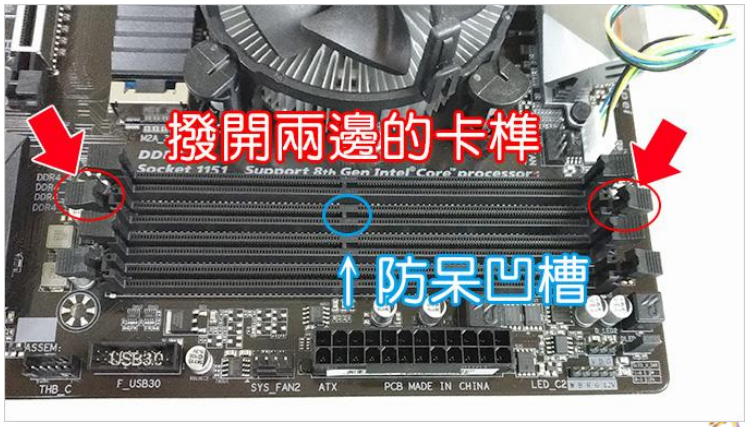
記憶體 (RAM)

記憶體插槽(DRAM Slot)

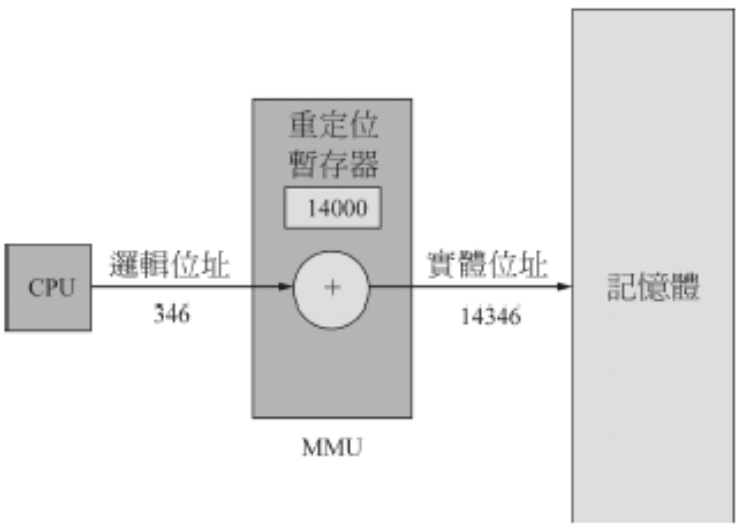
這長條狀的插槽就是插記憶體用的，一般主機板會有2～4條，或更多，緊鄰著CPU和北橋晶片。

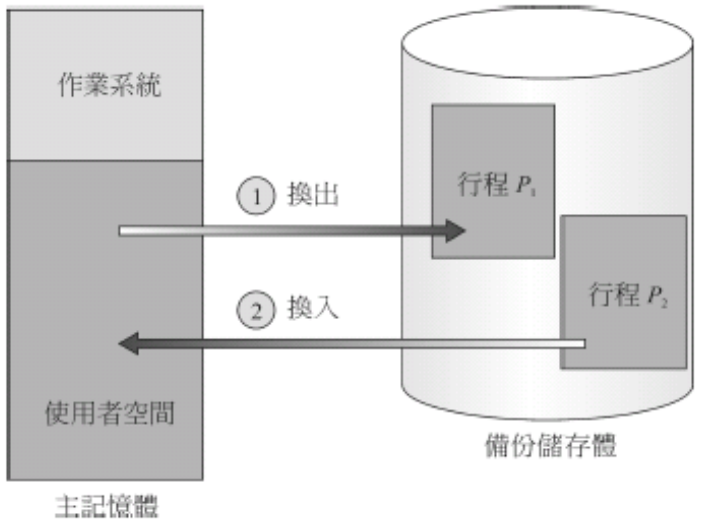


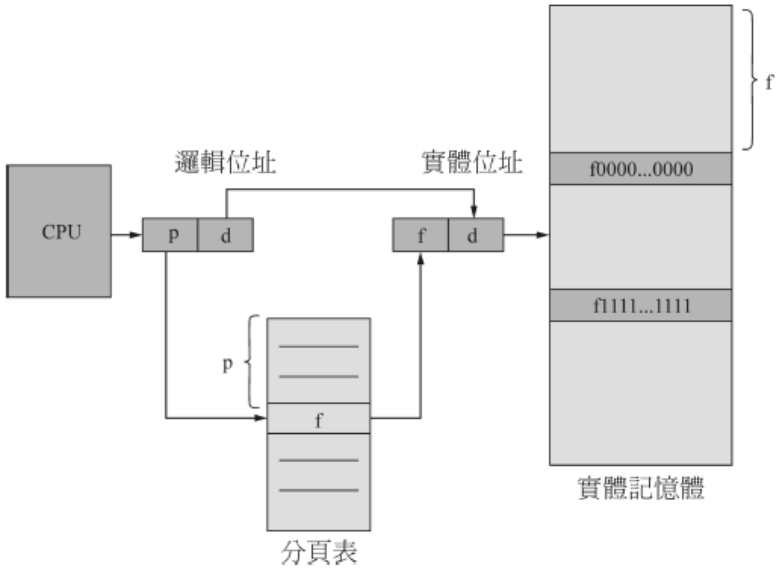


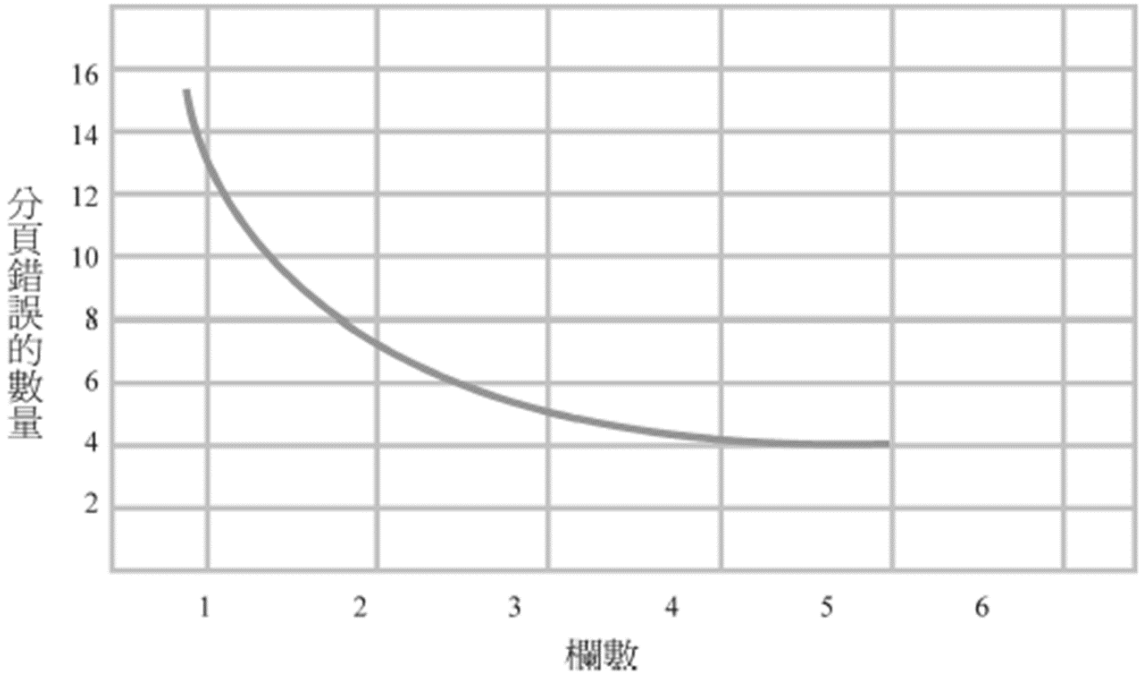


記憶體管理策略


重定位
暫存器
14000
CPU
邏輯位址
346
+
實體位址
14346
記憶體
MMU


作業系統
使用者空間
主記憶體
1 換出
2 換入
行程 P_1
行程 P_2
備份儲存體

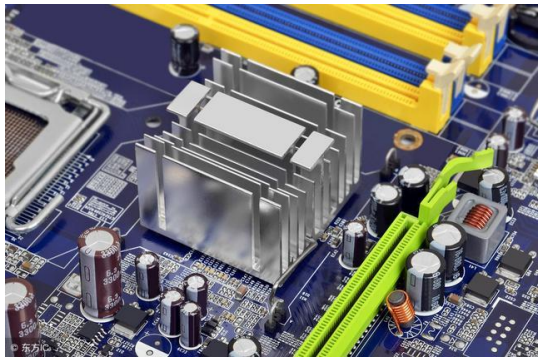

CPU
邏輯位址
實體位址
p
d
f
d
p
f
f0000...0000
f1111...1111
f
分頁表
實體記憶體

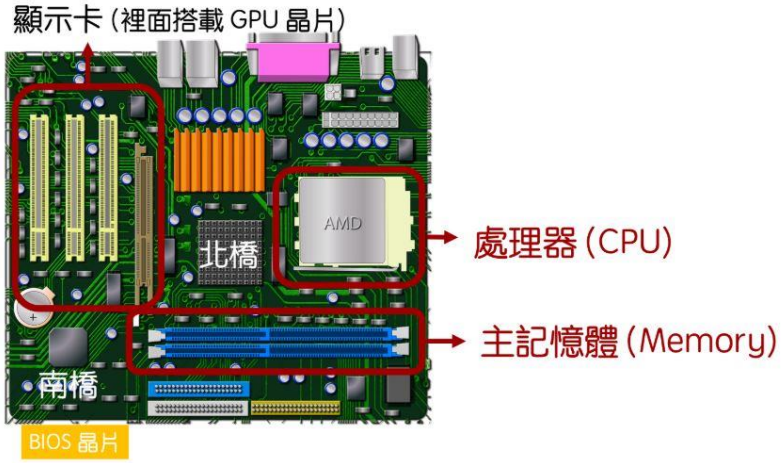


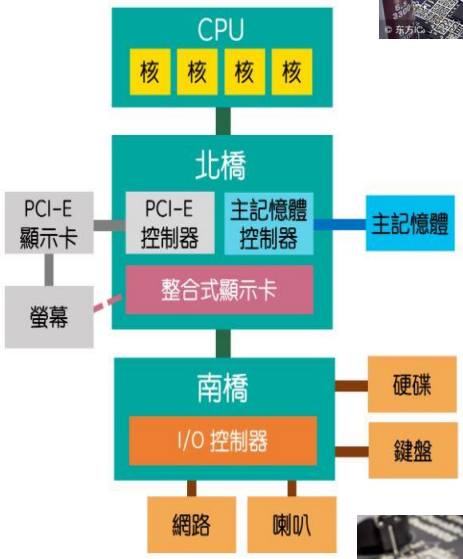
- FIFO演算法
- 最佳頁(Optimal)替換演算法
- LRU演算法

北橋晶片(Northbridge)

北橋晶片負責與CPU的聯繫並控制內存AGP數據在北橋內部傳輸，提供對CPU的類型和主頻、系統的前端總線頻率、內存的類型和最大容量、AGP插槽、ECC糾錯等支持，整合型晶片組的北橋晶片還集成了顯示核心。北橋起到的作用非常明顯，在電腦中起著主導的作用，所以人們習慣的稱為主橋（Host Bridge）

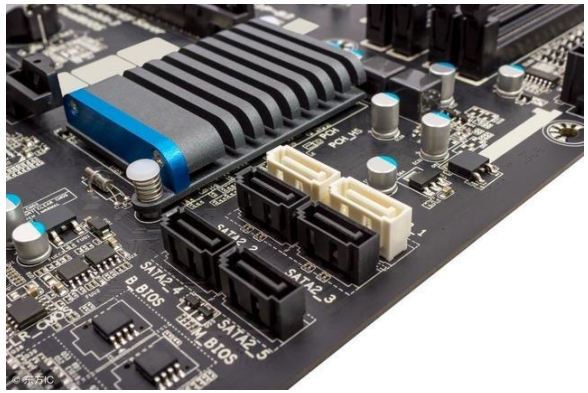






南橋晶片(Southbridge)

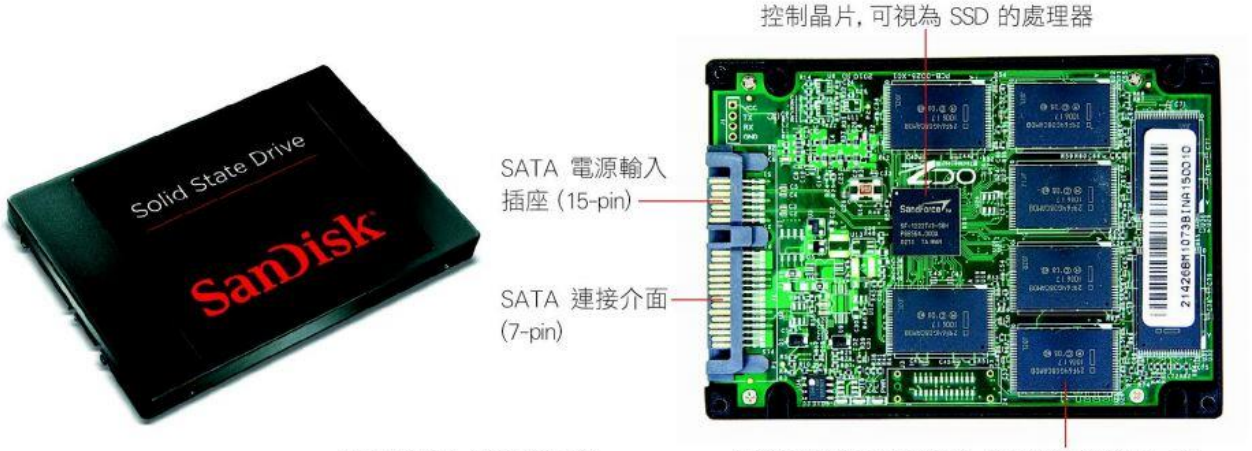
南橋是主機板上的老二，和北橋互連並連接其他週邊，我們熟知的主機板「功能」大多來自南橋，比如USB、網路、音效、SATA/IDE硬碟，都是從南橋連出來的。它也是一顆晶片，照片中看起來好像比北橋還大，那只是晶片製程和封裝的型式不同。



輔助記憶體 - 電腦上的儲存設備

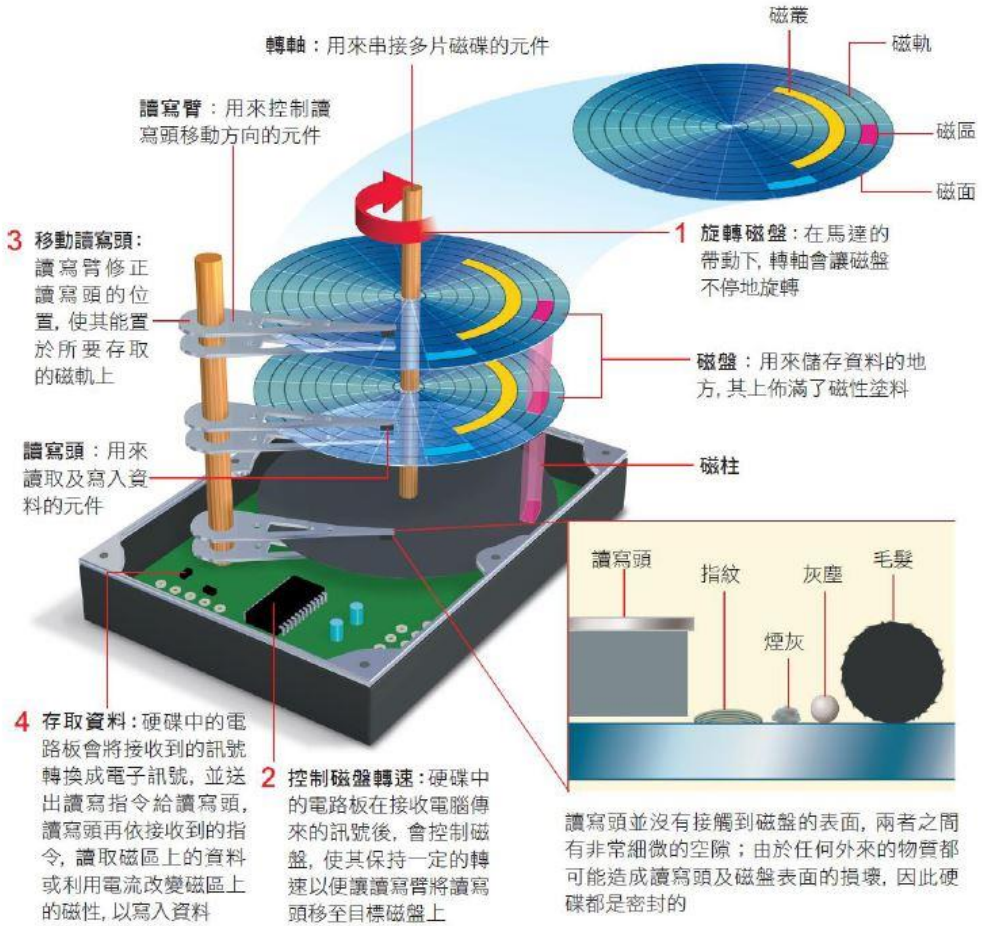
硬碟機－ HDD 與 SSD

圖表 2-27 HDD 硬碟機



圖表 2-28 SSD 硬碟機

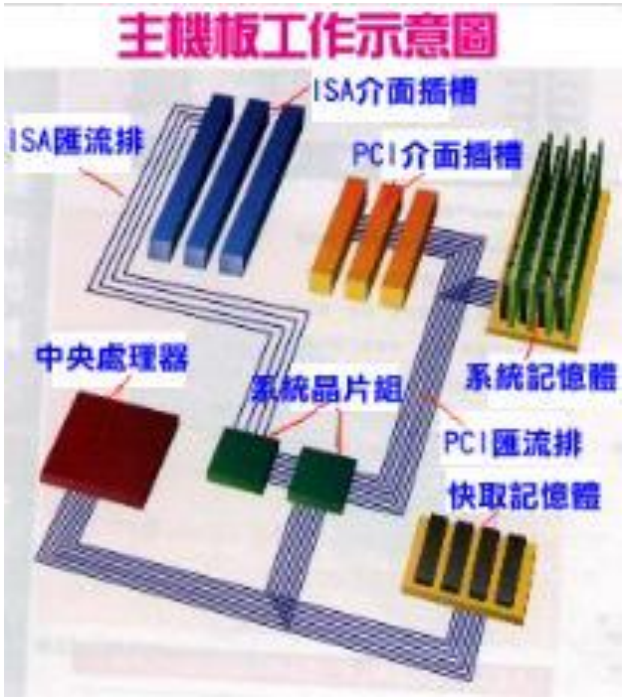
磁碟排班



- FCFS排班演算法
- 最短尋找時間先做(SSTF)
- SCAN演算法
- C-SCAN演算法
- LOOK
- C-LOOK演算法

匯流排(BUS)

匯流排是在主機板 (Mother Board) 上的一部份，主要是用來裝設各種介面卡 (Interface Card) ，並傳輸各種資料 (Data) 。



匯流排的種類與頻寬大小？

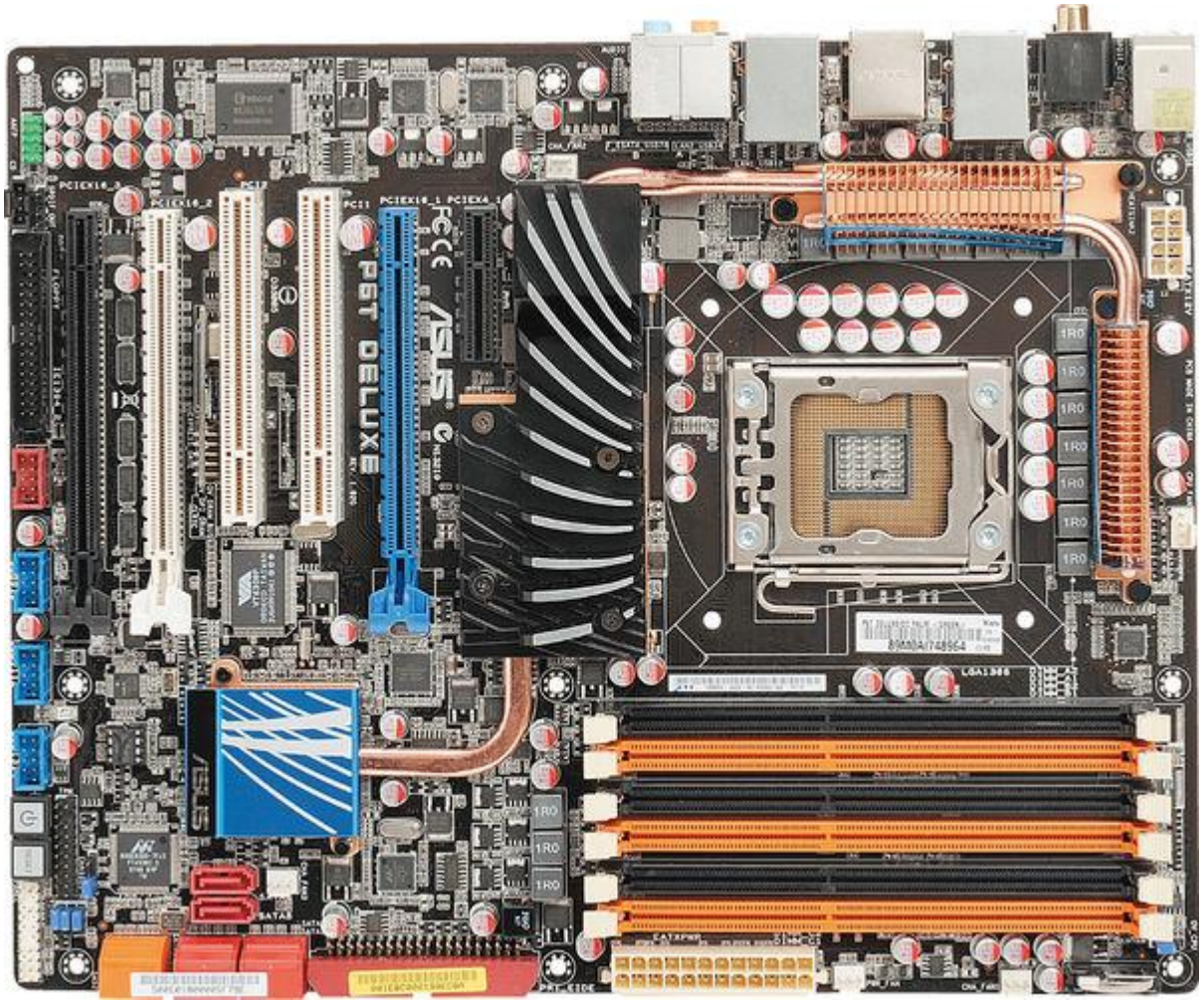
- Processor bus：(系統匯流排)

又稱前置匯流排(Front Side Bus)，是CPU資料進出的專用通道，由北橋晶片控制運作，資料寬度64bits，目前工作頻率仍以100或133MHz為主，但新技術已將頻率提升至266或甚至400MHz。

- Memory bus：(記憶體匯流排)

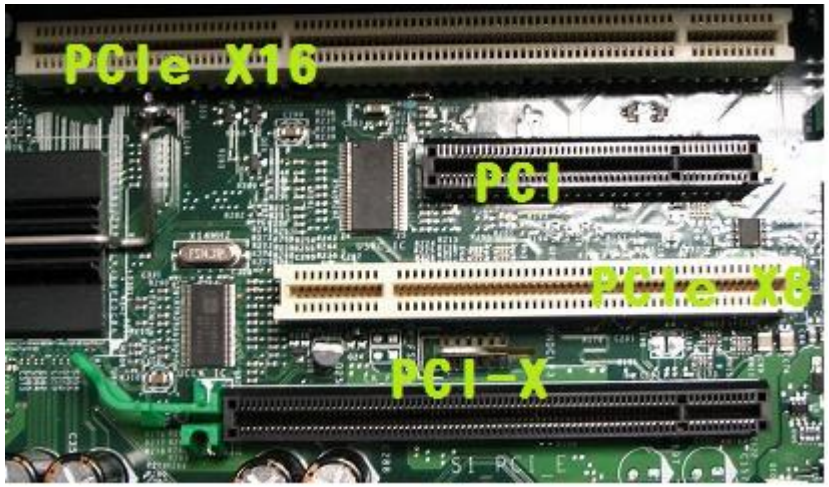
由北橋晶片控制運作，資料寬度64bits，工作頻率必須與系統匯流排保持一致(例如100或133MHz等)，才

Intel 於 1992 年提出了被稱為 PCI (Peripheral Component Interconnect) 的介面標準的一部分，之後數家科技公司共同成立了 PCI-SIG 並在 1993 年正式發布了 PCI 2.0 標準 (首個完整制定各項 PCI 插槽、介面卡等規範的版本)

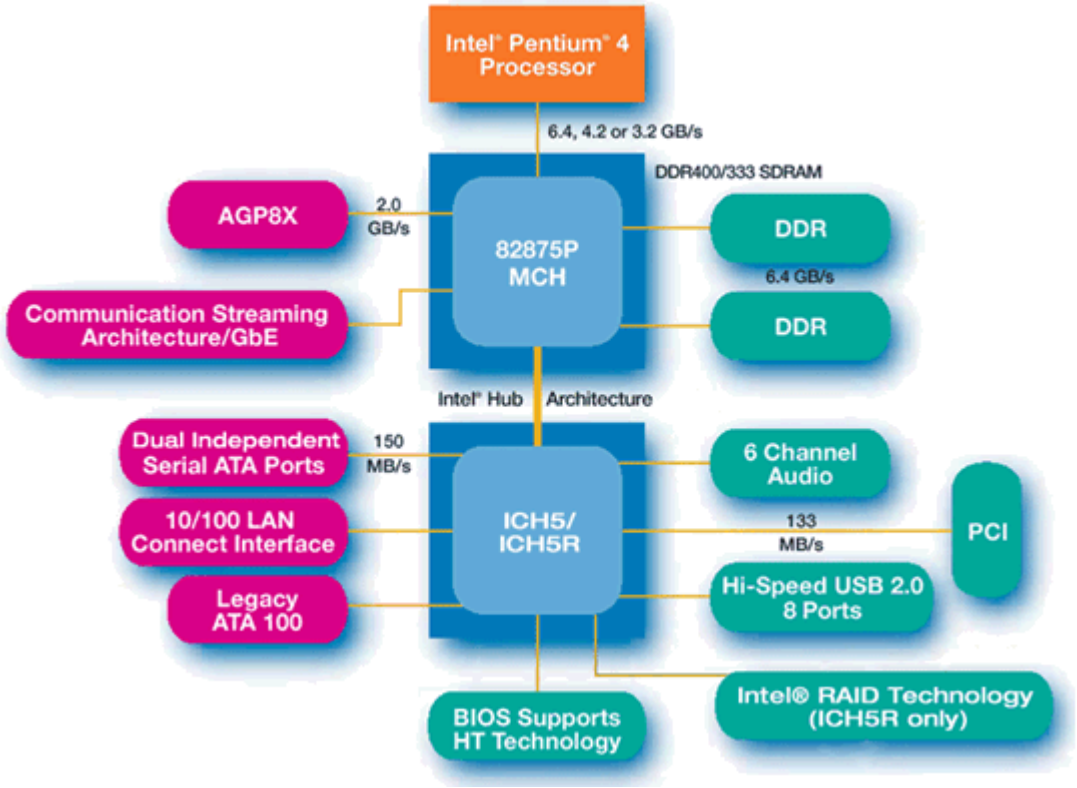


PCI Express x16 插槽

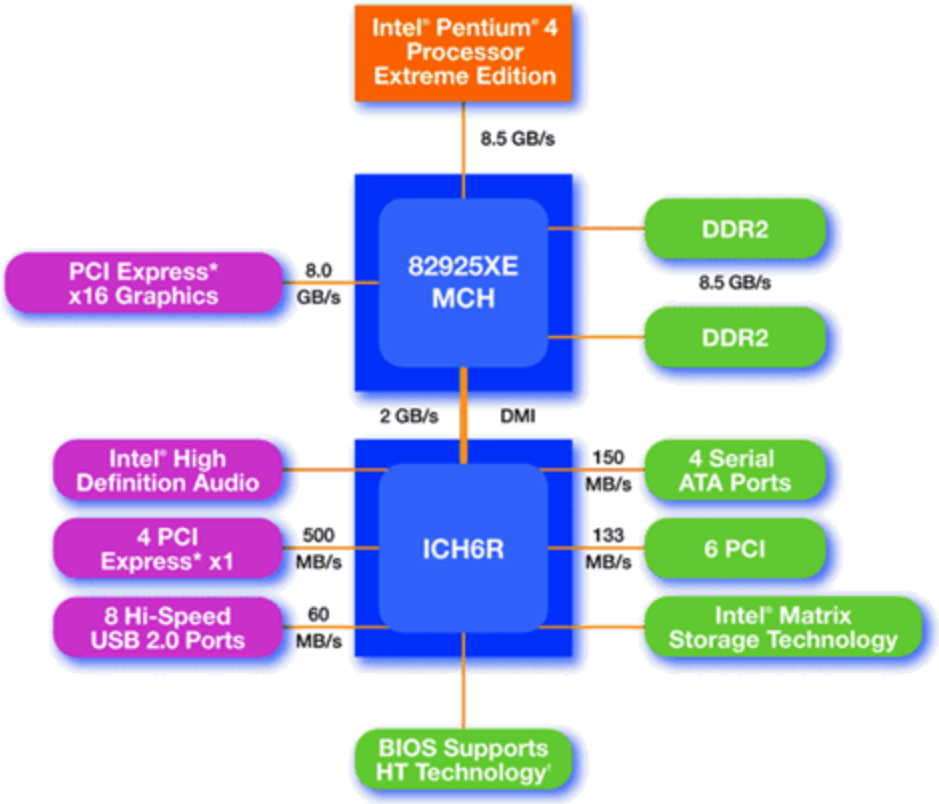
- PCI (Peripheral Component Interconnect) 是 一種匯流排介面, PCI Express (簡稱 PCIe) 是 全新一代的匯流排標準, 而 PCIe x16 插槽則 是顯示卡的專屬介面。
- PCIe 主要的特點是採單 / 雙向傳輸的通道設 計, 減少共用匯流排架構中相互干擾的問題, 頻寬也大為提升, 可在電腦開機的情況下進行 維護, 因此已逐漸取代 AGP 及 PCI 介面, 成 為顯示卡介面的新標準。



具備 PCI 匯流排的 2002 年電腦架構

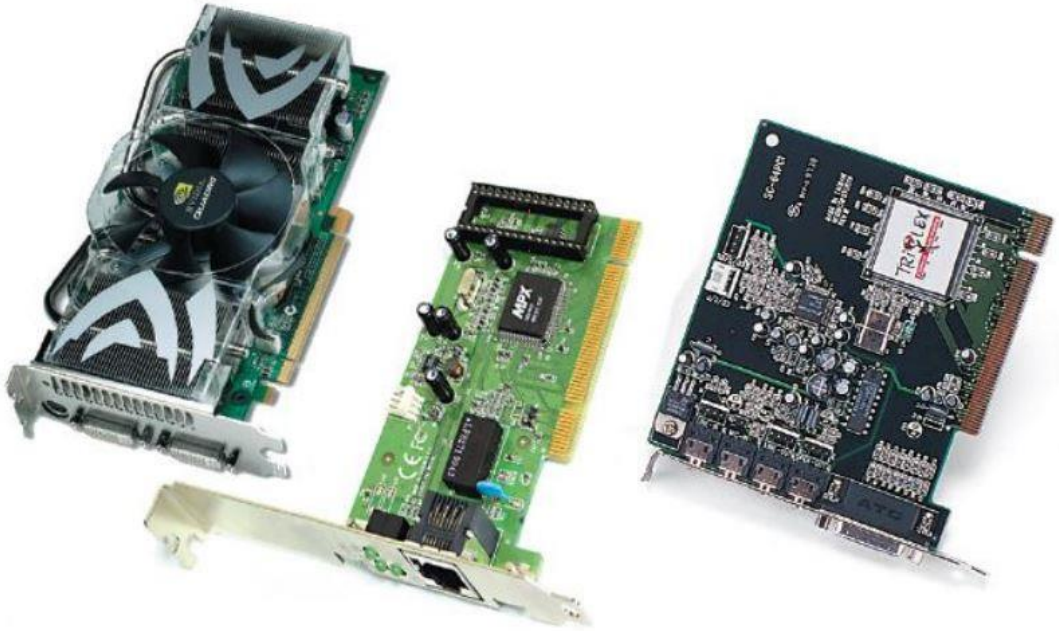


PCI Express 透過常見的匯流排架構統一 I/O 系統。

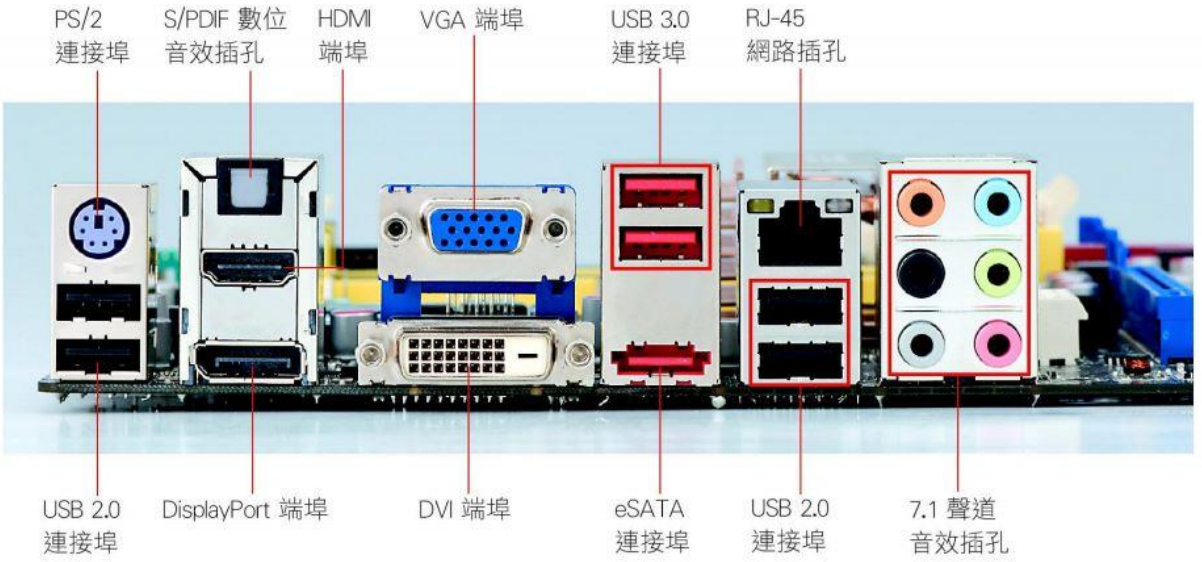


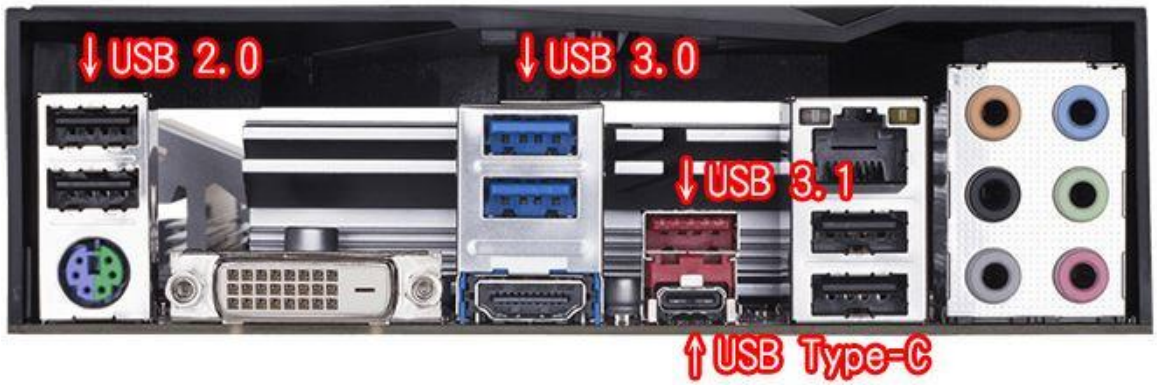
介面卡

介面卡（Interface Card）是安裝在主機板擴充槽上的一種「電子電路板」。雖然主機板大都已內建各種顯示、音訊、網路等功能，若想讓電腦加強特定功能，就可以再另外安裝介面卡。

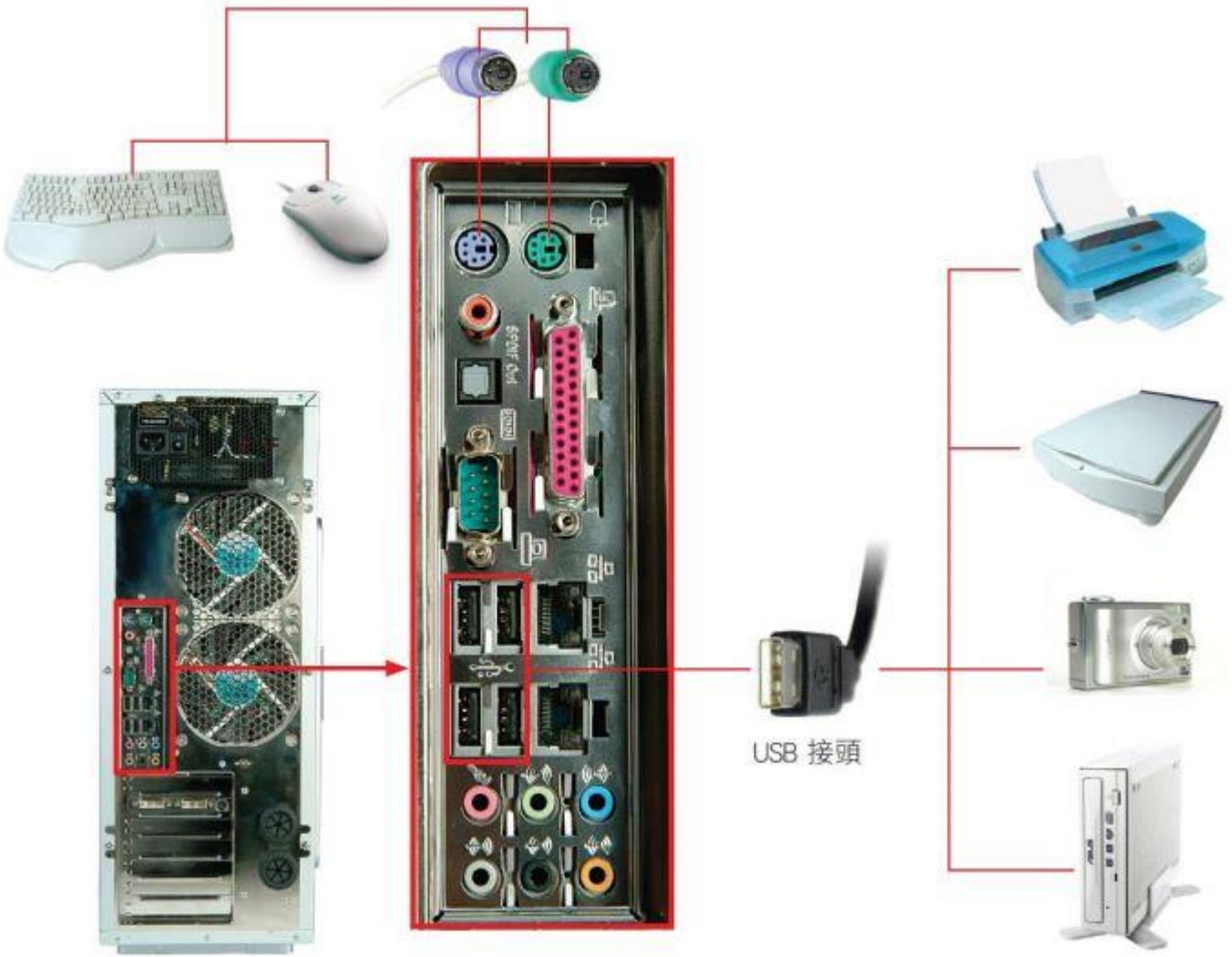


IO支援





版本	速度稱號	最大傳輸頻寬	速度 (理論值)
USB 3.0	SuperSpeed (超高速)	5 Gbps	500 MB / Sec
USB 2.0	Hi-Speed (高速)	480 Mbps	60 MB / Sec
USB 1.1	Full Speed (全速)	12 Mbps	1.5 MB / Sec
USB 1.0	Low Speed (低速)	1.5 Mbps	187.5 KB / Sec


USB 接頭

輪詢

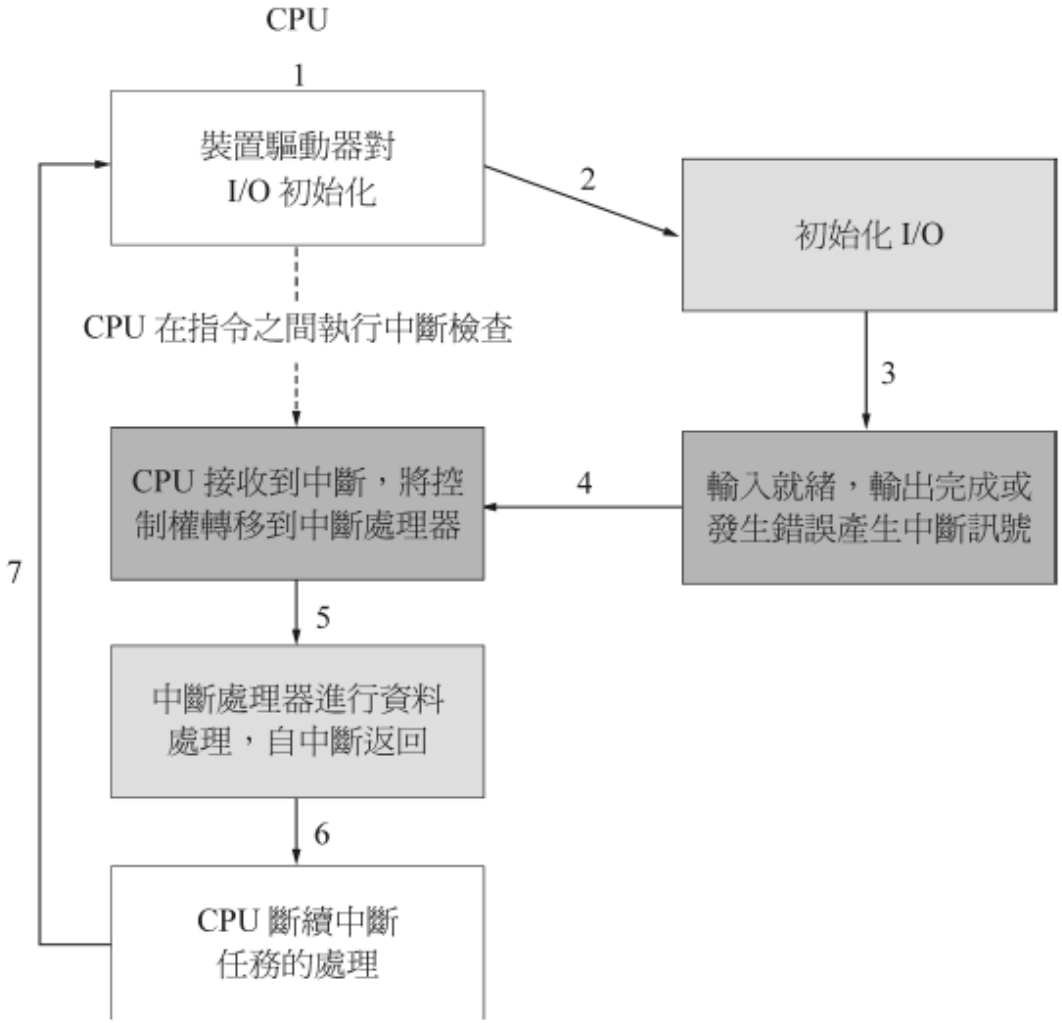
- 對於每一位元
  - 重複讀取狀態暫存器的忙碌位元，直到內容是0為止
  - 主機設定在指令暫存器中的寫入位元，並將位元組寫入資料輸出暫存器
  - 主機設定指令就緒位元
  - 控制器設定忙碌位元，執行傳輸
  - 當傳輸完成，控制器清除忙碌位元、錯誤位元、指令就緒位元

- 上面的步驟1是等待裝置I/O的忙碌等待週期
  - 如果裝置的速度夠快，這個方法就足夠了
  - 如果裝置速度慢時就沒有效率
  - CPU 轉換到其它的工作?
    - 如果主機未能即時讀取資料，將會造成資料滿溢/遺失

中 斷

- 輪詢可以在三個指令週期完成
  - 讀取狀態、利用邏輯-和(logical-and)運算抽取狀態位元、若不為零時產生分支
  - 如果狀態位元經常不是0時，如何更有效率?
- I/O 裝置觸發**中斷要求管線**
  - 會在每執行一個指令後檢查
- **中斷處理程式(Interrupt handler)** 接收中斷
  - **可遮罩(Maskable)** 以忽視或延遲某些中斷
- 中斷向量用來分配中斷到正確的**中斷處理程式**
  - 在開始與結束時執行內容轉換
  - 根據優先權Based on priority
  - 有些是**不可遮罩**
  - 如果相同的中斷號碼有超過一個裝置連接時，使用中斷串鏈

中斷-驅動I/O迴圈週期


CPU
1
裝置驅動器對
I/O 初始化
2
初始化 I/O
CPU 在指令之間執行中斷檢查
3
CPU 接收到中斷，將控
制權轉移到中斷處理器
4
輸入就緒，輸出完成或
發生錯誤產生中斷訊號
7
5
中斷處理器進行資料
處理，自中斷返回
6
CPU 斷續中斷
任務的處理

12.2.1 記憶體映射 I/O

- 裝置控制暫存器映射到處理器的位址空間
- 圖型控制器具有用於基本控制操作的 I/O 埠
  - 但是該控制器具有較大的記憶體映射區域，用於保存螢幕的內容

圖 12.2　在 PC 上的裝置 I/O 埠位址 (部份資料)

I/O 位址範圍 (十六進位)	裝置
000–00F	DMA 控制器
020–021	中斷控制器
040–043	計數器
200–20F	遊戲控制器
2F8–2FF	串列埠 (輔助)
320–32F	硬碟控制器
378–37F	平行埠
3D0–3DF	圖型控制器
3F0–3F7	磁碟片裝置控制器
3F8–3FF	串列埠 (主要)

12.2.2 輪　詢

1. 主機將重複讀取 busy 位元，直到位元被清除
2. 主機設定在 command 暫存器中的 write 位元，並將位元組寫入 data-out 暫存器
3. 主機設定 commandy-ready 位元
4. 當控制器發現 commandy-ready 位元已經設定完成，即設定 busy 位元
5. 控制器讀取指令暫存器，並發現 write 指令，則自 data-out 暫存器讀取位元組，並且執行必須的裝置 I/O 處理
6. 控制器清除 commandy-ready 位元，並清除在狀態暫存器中的 error 位元，表示裝置 I/O 已經成功，再清除 busy 位元以便表示動作完成

12.2.2 輪　詢

- 在步驟 1 之中，主機處於**忙碌等待** (busy-waiting) 或**輪詢** (polling) 的狀態
- 如果控制器與裝置速度夠快，這個方法就夠了，但是也可能因為主機切換到另一個工作上，而使得等待時間
- 如何知道控制器何時變為閒置狀態呢？
- 主機必須提供裝置快速的服務，否則可能造成資料遺失

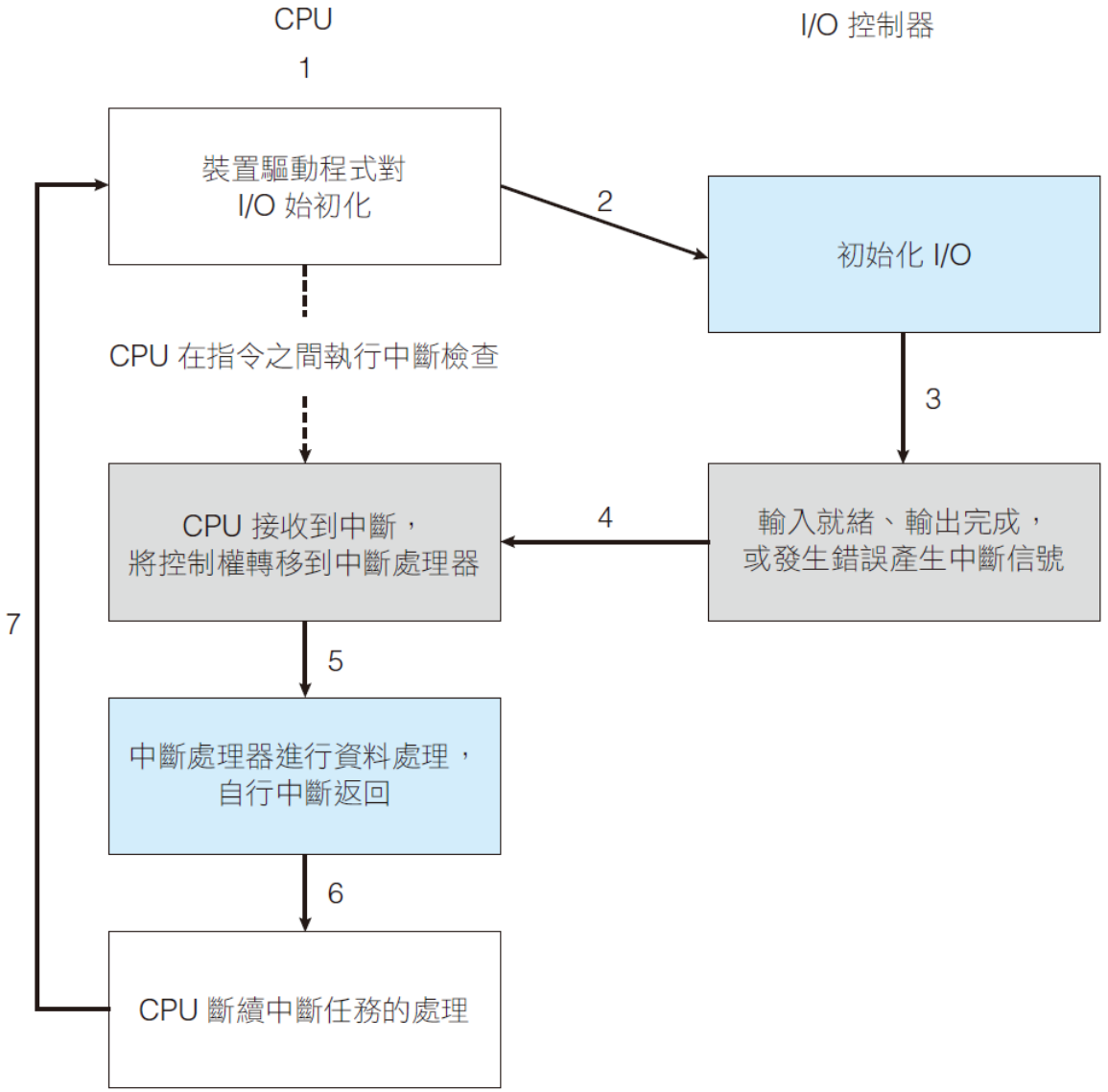
12.2.3 中　斷

- **中斷要求管線** (interrupt-request line) 的纜線，CPU 會在每執行一個指令後檢查
- 大部份的 CPU 有兩種中斷要求管線：
  - 一種是無遮罩中斷 (nonmaskable interrupt)，保留給像不可回復記憶體錯誤之事件使用
  - 第二種為遮罩 (maskable) 中斷：
    - CPU 可以在執行不可被中斷的關鍵指令之前先關閉這類中斷，像裝置控制器都使用遮罩中斷來要求服務

12.2.3 中 斷

- 中斷向量方法的目的在於讓中斷處理器不需搜尋所有中斷發生的可能來源
  - 即可決定到底哪一個裝置需要服務
- **中斷串鏈** (interrupt chaining)，在中斷向量裡的每個元素都指向由中斷處理器組成之串列前端
  - 當中斷發生時，則會一個個呼叫對應串列內的處理器，直到可回應要求之中斷器找到為止

圖 12.3　中斷-驅動 I/O 週期



12.2.3 中 斷

- 大部份的 CPU 有兩種中斷要求管線：
  - 一種是**無遮罩中斷** (nonmaskable interrupt)，保留給像不可回復記憶體錯誤之事件使用
  - 第二種為**遮罩** (maskable) 中斷：
    - CPU 可以在執行不可被中斷的關鍵指令之前先關閉這類中斷，像裝置控制器都使用遮罩中斷來要求服務

12.2.3 中　斷

- **中斷向量**方法的目的在於讓中斷處理器不需搜尋所有中斷發生的可能來源，即可決定到底哪一個裝置需要服務
- **中斷串鏈** (interrupt chaining)，在中斷向量裡的每個元素都指向由中斷處理器組成之串列前端
  - 當中斷發生時，則會一個個呼叫對應串列內的處理器，直到可回應要求之中斷器找到為止

圖 12.5 Intel Pentium 處理器事件向量表

向量值	種類
0	除法錯誤
1	偵錯例外
2	空中斷
3	中斷點
4	INTO－偵測溢位
5	邊界範圍例外
6	無效 opcode
7	裝置未就緒
8	雙重錯誤
9	雙處理器段溢位 (保留)
10	錯誤工作狀態區段
11	區段不存在
12	堆疊錯誤
13	一般保護
14	分頁錯誤
15	(Intel 保留，未使用)
16	浮點數錯誤
17	對位檢查
18	機器檢查
19–31	(Intel 保留，未使用)
32–255	遮罩中斷

12.2.3 中　斷

- 中斷機制也製作系統的**中斷優先權層** (interrupt priority level)
  - 此機制使得 CPU 能夠在不關閉所有中斷的前提下，延遲處理低優先權的中斷，並使高優先權中斷能夠先於低優先權中斷執行。
- 中斷機制也用來處理許多不同的**例外** (exception)
- 分頁錯誤是引發中斷的異常
- 當系統呼叫執行陷阱指令時，中斷硬體將儲存使用者程式碼的狀態，切換至核心模式，再分派至實作要求服務之核心常式

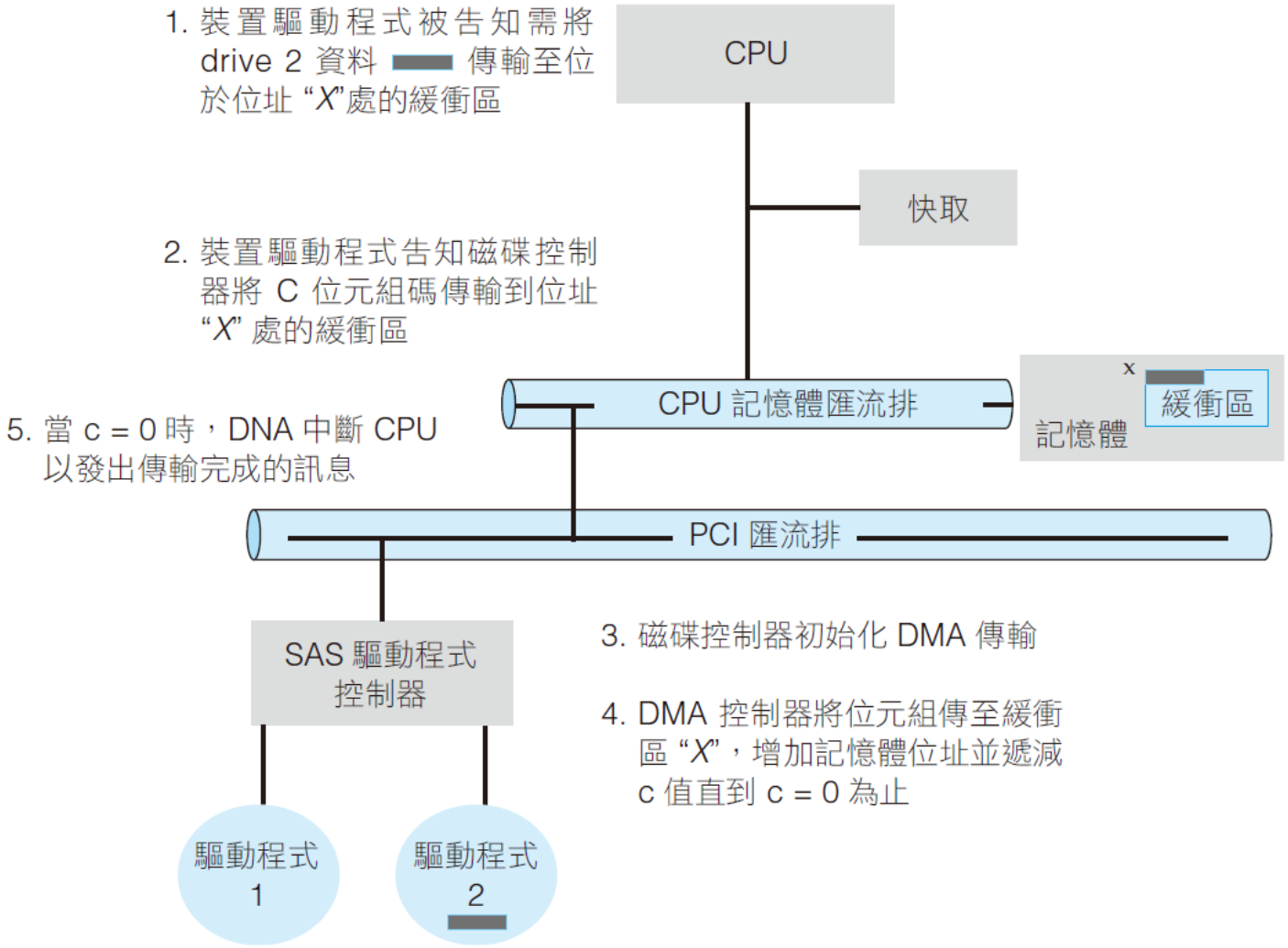
12.2.4 直接記憶體存取

- 需要做大量傳輸的裝置，若使用昂貴及一般用途的處理器來監看狀態位元，以及將資料以一次一位元組大小的方式送入控制暫存器之中，似乎有點浪費——避免加重主要 CPU 的負擔
  - **直接記憶體存取** (direct-memory-access, DMA) 控制器
  - DMA 傳輸做初始化時，主機會將 DMA 指令區塊寫入記憶體
  - 包含指向傳輸來源的指標、指向傳輸目的地的指標與傳輸位元組數目
- DMA 控制器會直接操作記憶體匯流排，將位址置於匯流排上，不需主要 CPU 的輔助即可執行傳送工作

12.2.4 直接記憶體存取

- DMA 控制器會中斷 CPU，這個過程參考圖 12.6
  - 需注意的是，當 DMA 控制器取得記憶體匯流排時，CPU 暫時阻止存取主記憶體，而只能存取快取中的資料項；
  - **週期偷取** (cycle stealing) 可能會降低 CPU 的運算速度，但將資料傳輸的工作交給 DMA 控制器處理，一般的確可改善整個系統的執行效率
- **虛擬記憶體存取** (direct virtual memory access, DVMA)，使用需經過虛擬至實體記憶體位址轉換的虛擬位址

圖 12.6 DMA 傳輸之步驟



12.3 應用 I/O 介面

- 裝置驅動程式層的目的在於隱藏各裝置控制器間之相異點，不讓核心的 I/O 子系統觸及
- 正如 I/O 系統呼叫起裝置行為以少量且一般類別包裝，以達到隱藏應用程式硬體相異點的目的

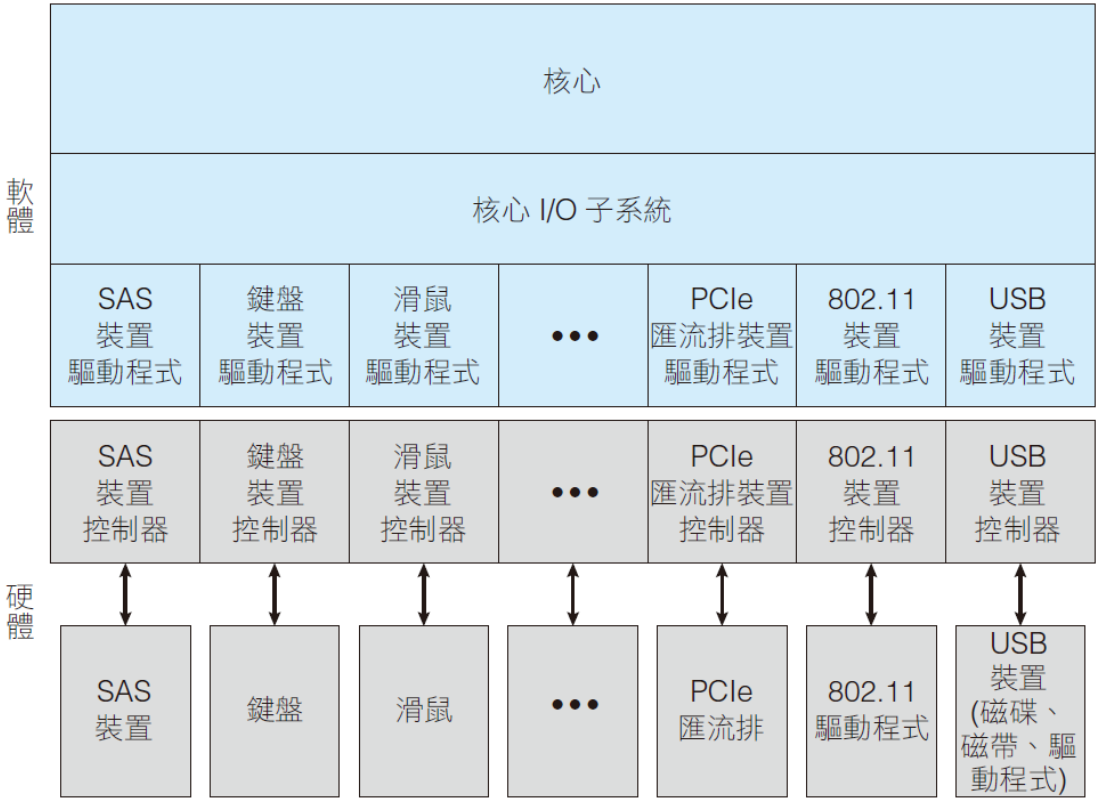


圖 12.7 核心 I/O 結構

12.3 應用 I/O 介面

- 字元串列或區塊
- 循序或隨機存取
- 同步或非同步
- 共用或指定
- 指令操作速度
- 唯讀、唯寫

種類	變動 (異)	範例
資料傳輸模式	字元 區塊	終端機 磁碟機
存取方式	順序的 隨機的	數據機 CD-ROM
傳輸排班	同步 非同步	磁帶 鍵盤
共用	指定 可共用	磁帶 鍵盤
裝置速度	潛伏的 搜尋時間 傳輸速率 操作之間的延遲	
I/O 方向	只能讀取 只能寫入 讀－寫	CD-ROM 圖型控制器 磁碟

圖 12.8 I/O 裝置之特色

12.3 應用 I/O 介面

- 裝置類別的介面卻已標準化
- 區塊 I/O、字元串列 I/O、記憶體映射檔案存取，以及網路插座
- 大部份的作業系統也有**跳脫** (escape)
  - 或後門 (back door)
- 在 UNIX 中，ioctl() 系統呼叫讓應用程式可以存取任何裝置驅動程式上

12.3.1 區塊與字元裝置

- **區塊裝置介面** (block-device interface) 包括所有存取磁碟機
  - read() 與 write()
  - seek()
- **原始 I/O** (raw I/O)
- **直接 I/O** (direct I/O)

12.3.1 區塊與字元裝置

- 記憶體映射檔案存取將檔案映射至記憶體之系統呼叫時，將傳回一串包含檔案複製內容之字元串列的虛擬記憶體位址
  - 真正的資料傳輸只在當需要存取記憶體映像時才會執行
  - 因為這部份的傳輸與用來處理分頁需求 (demand-page) 虛擬記憶體存取使用到的機制相同，記憶體映射 I/O 即已足夠
- 字元串列介面 (character-stream interface)
  - 基本系統呼叫可以讓應用程式 get() 或 put() 一個字元
  - 鍵盤、滑鼠及數據機之類的輸入裝置相當方便

12.3.2 網路裝置

- 許多如 UNIX 與 Windows 的作業系統之中，稱為網路**插座** (socket) 介面
- select() 的功能
  - 呼叫 select() 功能免除輪詢與忙碌等待
  - 這些原本為網路 I/O 必須處理的時間
  - 這些功能將網路基本功能隱藏起來，對於建立那些將使用網路硬體與協定堆疊之分散式應用程式而言大有助益
- UNIX 之中，有半雙工管線 (halfduplex pipe)、全雙工 (full-duplx) FIFO、全雙工 STREAMS、訊息佇列及插座

12.3.3 時鐘與計時器

- 大部份電腦都有提供三種基本功能的時鐘與計時器：
  - 記錄目前時間 (current time)
  - 記錄經過時間 (elapsed time)
  - 設定計時器，可在時間 T 啟動操作 X

12.3.4 阻隔與非阻隔 I/O

- **阻塞** (blocking)
  - 馬上懸置應用程式的執行，此應用程式將自作業系統的執行佇列移至等待佇列
  - 在系統呼叫完成後，應用程式將移回執行佇列，取得由系統呼叫傳回之資料值，並轉變為可繼續執行狀態 (resume execution)
  - 當它恢復執行時，將接收系統呼叫傳回的數值
  - 比較容易瞭解

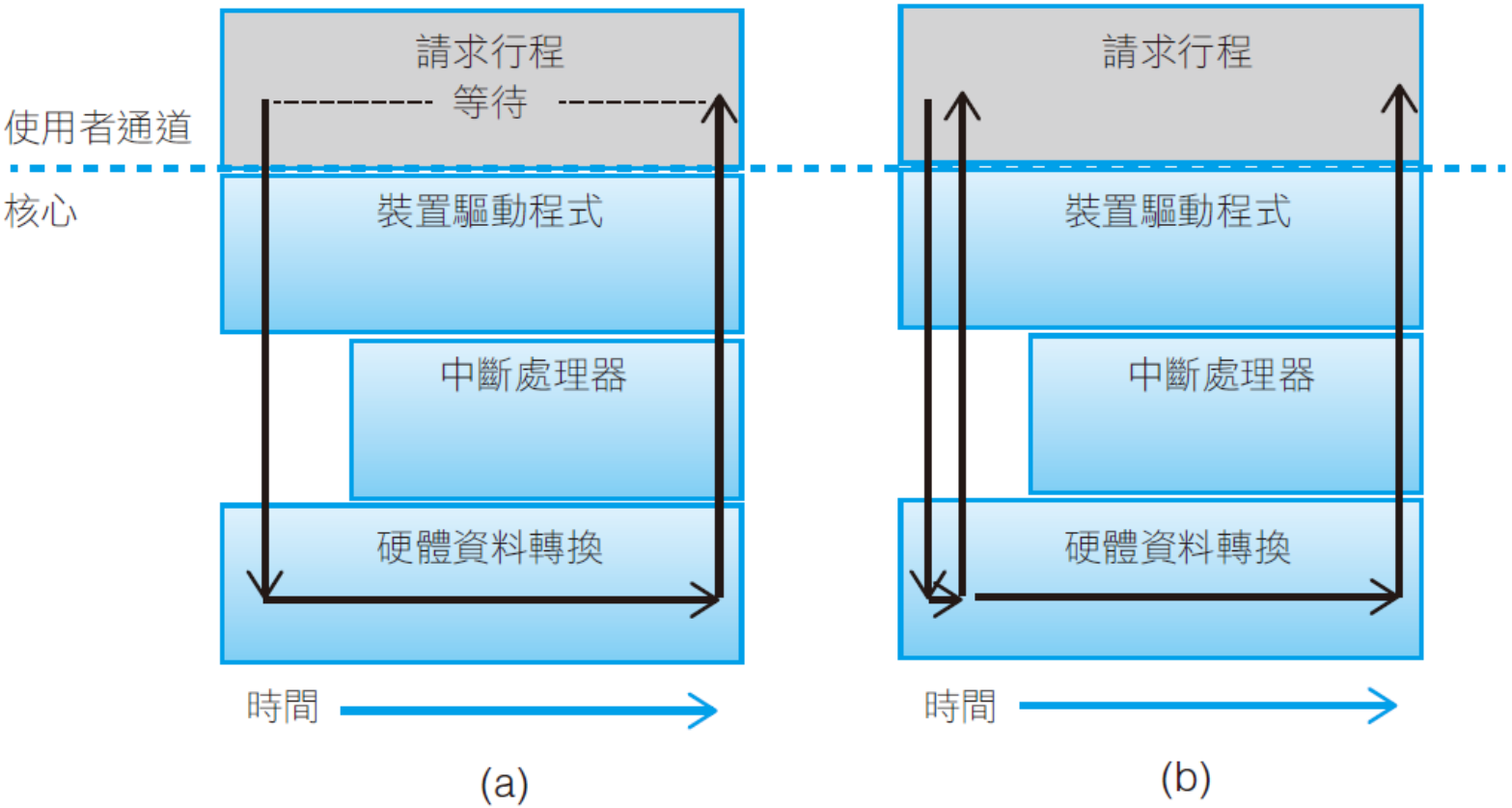
12.3.4 阻隔與非阻隔 I/O

- **非阻隔** (nonblocking)
  - 應用程式設計者可重疊 (overlap) 使用 I/O 的方法之一是，撰寫一個多執行緒應用程式
  - 某些執行緒可以在其它程序進行時執行阻隔系統呼叫，而其它的則會繼續執行
  - 某些作業系統提供非阻隔 I/O 系統呼叫
  - 一非阻隔呼叫並不會中斷正處於執行狀態的應用程式
  - 相反地，它將快速的傳回一個資料值，指出已經有多少位元組傳輸成功之資訊

12.3.4 阻隔與非阻隔 I/O

- 非同步系統呼叫不需等待 I/O 完成，即可立刻回傳，執行緒可繼續執行
  - 而 I/O 在工作完成後，會透過幾種方法通知執行緒，可以經由設定執行緒中變數位址空間
  - 或是透過驅動信號、軟體中斷
  - 或是在執行緒線性控制流動 (linear control flow) 以外執行的回呼常式 (call-back routine)

圖 12.9　雙 I/O 的方式



(a) 同步和 (b) 非同步

12.3.5 向量 I/O

- 允許一個系統呼叫可以對多個位置執行 I/O 操作
  - UNIX 的系統呼叫 ready 接受一個多緩衝區的向量，並從來源讀取資料到該向量，將該向量寫入目的地
- 沒有向量 I/O，資料可能先需要以正確的順序被傳送到一個較大的緩衝區，然後才傳送，這非常沒有效率
  - 有些版本的分散—集中提供單元性，以確保沒有中斷被執行
    - 以免如果其它執行緒也對這些緩衝區執行 I/O 時造成資料毀損
  - 利用分散—集中的 I/O 特性，以增加產量和降低系統額外負擔

12.4 核心 I/O 子系統

- I/O 排班
- 緩　衝
- 快　取
- 排存和裝置預約
- 錯誤處理
- I/O 保護
- 電源管理

12.4.1 I/O 排班

- 找出好的執行順序以便執行
  - 允許執行行程公平地享有裝置存取權力
  - 降低 I/O 完成指令的平均等待時間

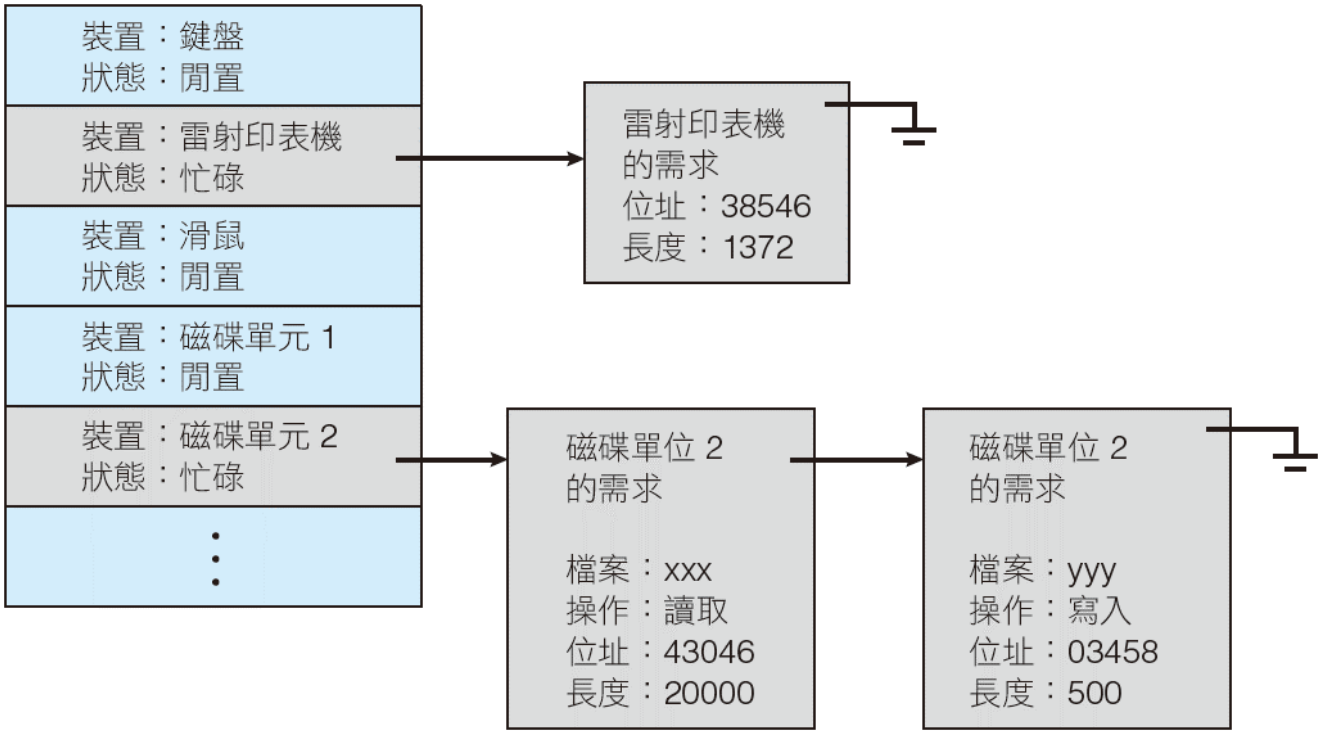
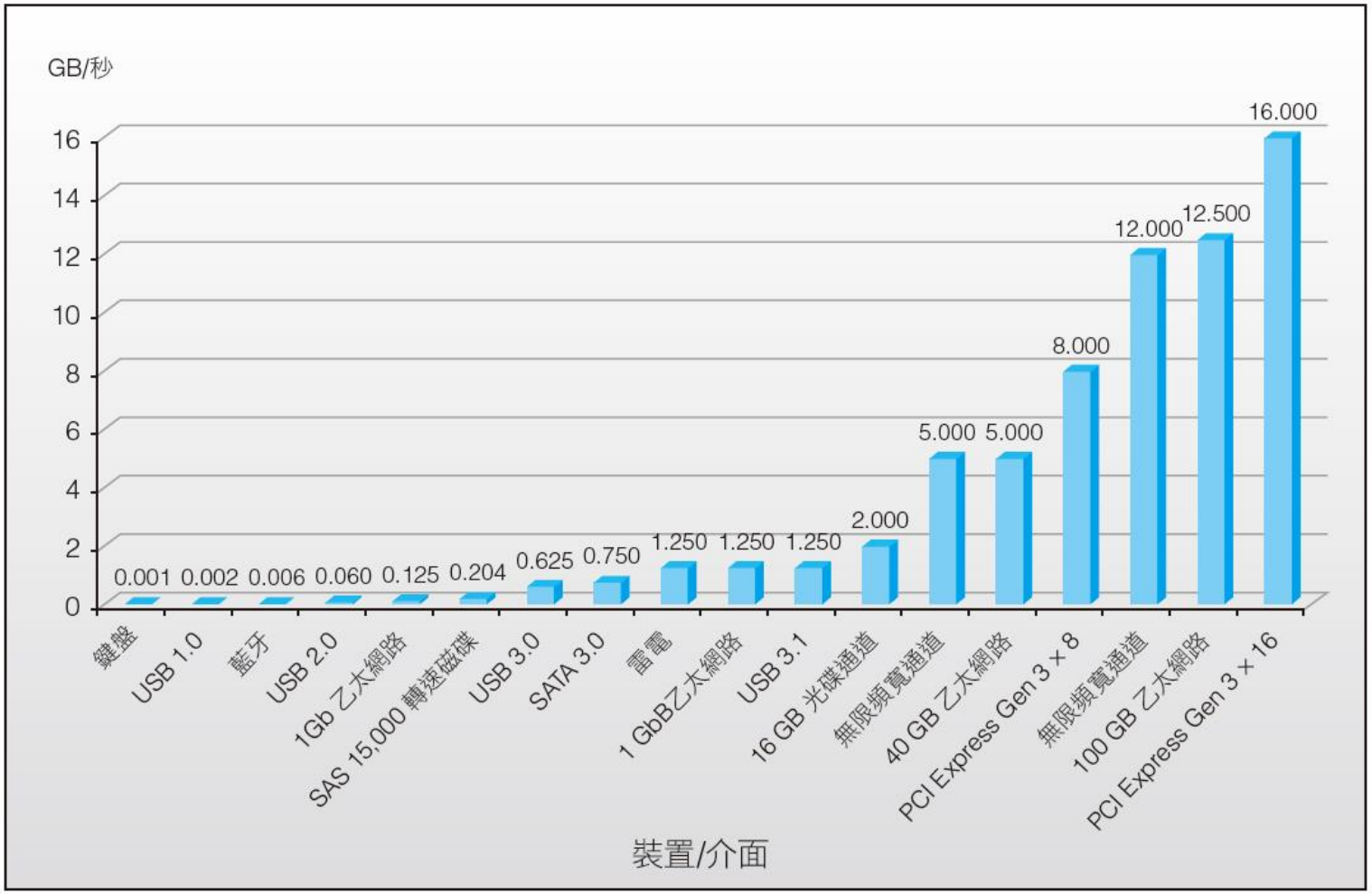


圖 12.10 裝置狀態表

12.4.2 緩　衝

- 在兩個裝置或裝置與應用程式之間傳輸資料時，可以儲存資料的記憶體區域
- 使用緩衝的原因有三
  - 一是為了解決生產者與消費者的資料串列在速度上不相等的問題
    - **雙緩衝** (double buffering) 方法讓生產者與消費者之間的資料脫鉤，並放鬆它們之間的時間需求限制
  - 用在不同資料傳輸大小的裝置之間做調整
  - 用來提供應用 I/O 的複製語法
    - 採用**複製語法** (copy semantic) 時，寫入磁碟之資料版本保證是應用系統呼叫時的資料版本，與應用在緩衝器中任何後來的改變無關

圖 12.11　一般電腦和資料中心 I/O 裝置與介面的速度



12.4.3 快　取

- 一個持有資料複製的快速記憶體，存取已經快取的複製會比存取原有的資料來得有效率
- 持有一個存在於其它位置之項目的複製在快速儲存體
- 有時候同一個記憶體範圍可以讓這兩種目同時使用

12.4.4 排存和裝置預約

- spool 用來保留裝置 (例如印表機) 輸出的緩衝區，不能接收交錯的資料串列
  - 一部印表機一次只能為一個工作服務，但是許多應用可能同時想要列印它們的輸出，而且不能將它們的輸出混在一起
- 裝置預約讓處理程序配置一個閒置裝置，當不需要時再將此裝置釋回避免死結 (deadlock) 的情況發生

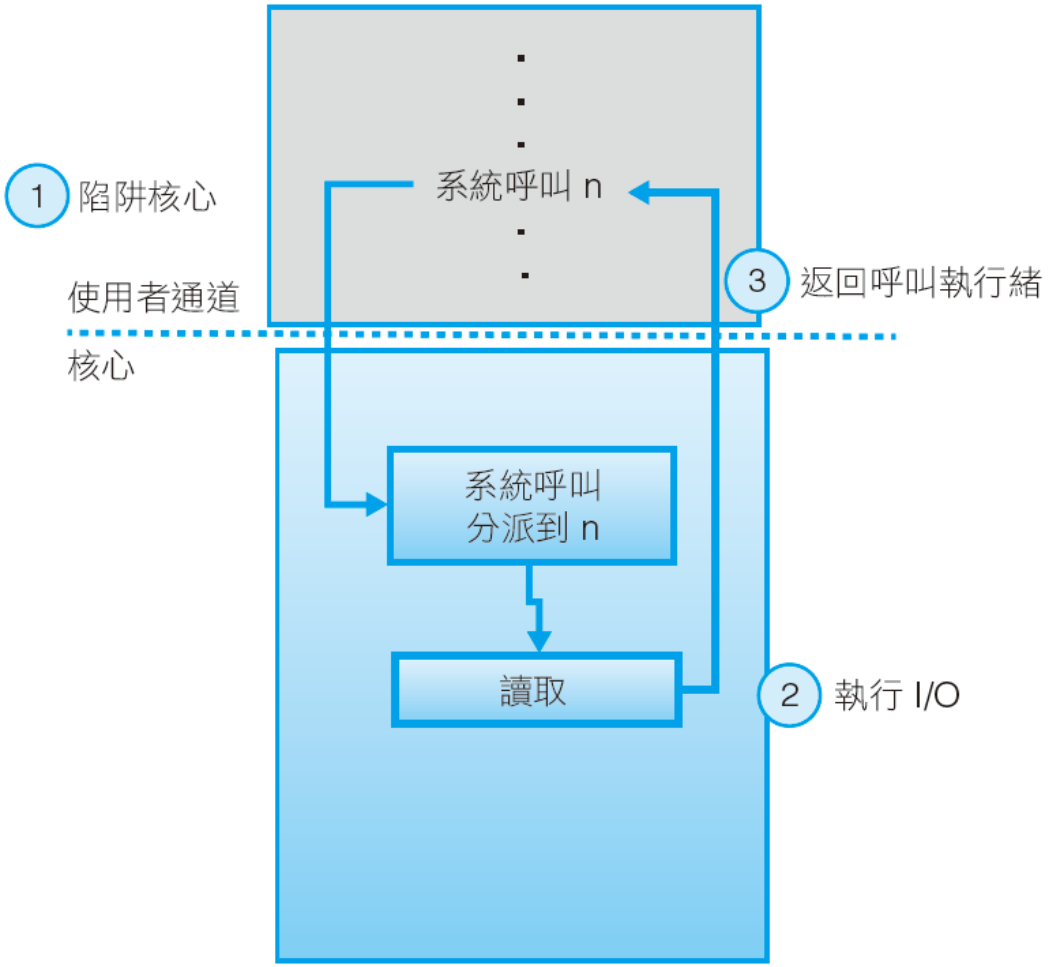
12.4.5 錯誤處理

- 使用保護記憶體的作業系統，能夠防止許多型態的硬體與應用程式錯誤，因此真正的系統錯誤並非一般小型機械誤失引起
- SCSI 裝置發生失誤時，會由 SCSI 協定回報三種層次的細節：
  - **感應鍵值** (sense key) 辨別失誤的一般性質，例如硬體錯誤或非法要求
  - **額外感應碼** (additional sense code) 可以說明像無效指令參數或自我測試錯誤之類的錯誤類別
  - **額外感應碼修飾子** (additional sense-code qualifier) 則可提供更詳細的資料

12.4.6 I/O 保護

- 使用者意外地或有目的地藉由使用非法 I/O 指令企圖打斷正常的系統操作
- 執行 I/O，使用者程式執行系統呼叫，要求作業系統代表使用者程式執行 I/O
- 任何記憶體映射和 I/O 埠記憶體位置必須由記憶體保護系統保護，以防止使用者存取

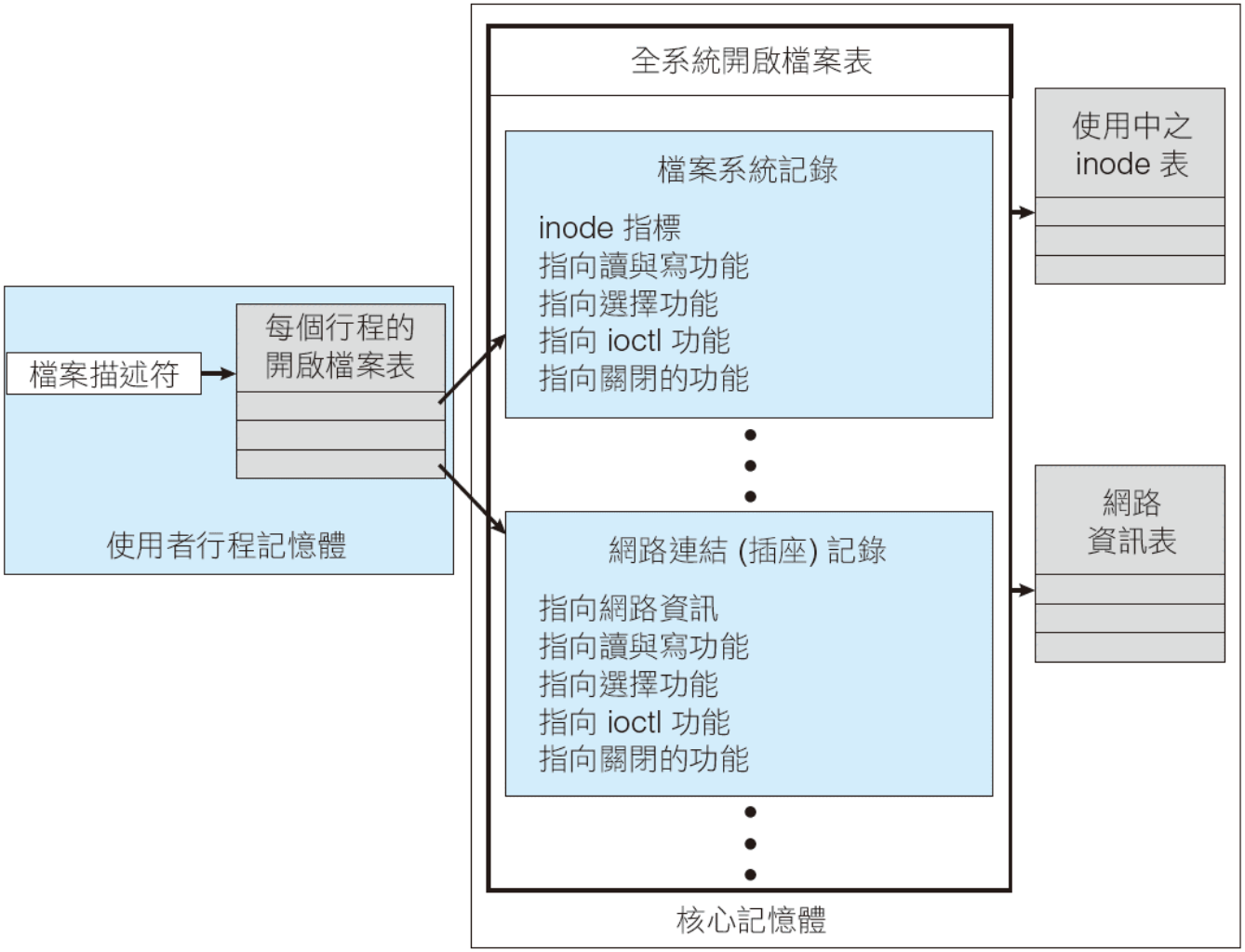
圖 12.12　系統使用者呼叫來執行 I/O



12.4.7 核心資料結構

- 系統核心需要保有關於 I/O 元件使用的狀態資訊
- 核心使用許多類似的結構來追蹤網路連接、字元裝置通信及其它 I/O 活動
- 要讀取使用者檔案時，核心需要在決定是否執行磁碟 I/O 前，先對緩衝區快取進行檢測
- 某些作業系統更廣泛地使用物件導向方法
  - Windows 在 I/O 上使用訊息傳達 (message-passing) 的方法
  - I/O 要求轉換成為一個訊息，經由核心傳送給 I/O 管理者，然後再傳送給裝置驅動程式，而在每個停留點都可能改變訊息內容

圖 12.13　UNIX I/O 核心結構



12.4.8 電源管理

- 用電並產生熱能會讓電腦零件在高溫中發生故障，因此冷卻也是其中的考量之一
- 作業系統於電力使用上扮演著重要的角色 (並因此進行熱能處理和冷卻)
  - 在雲端運算環境中，可以經由監控和管理工具來調整負載
  - 進而從系統中轉移出所有使用者行程，來讓這些系統閒置
  - 近而能夠關閉它們的電源，直到再次需要使用負載為止
- 在行動計算中，電源管理成為作業系統的高度優先

12.4.8 電源管理

- Android 如何關閉手機的各個元件？
- 如何知道何時安全關閉快閃儲存器，以及如何能在關閉整個 I/O 子系統電源知道如何進行呢？
  - 元件層級的電源管理方式
- 瞭解元件間關係
  - 建立一個代表電話實體設備拓撲的裝置樹
  - I/O 子系統的快存和 USB 儲存器為系統匯流排的子節點，它將依序連結到 CPU
  - 驅動程式能追蹤元件是否正使用中
    - 如果未使用的元件則會將其關閉
    - 如果整個裝置樹中的所有元件都未被使用，系統可能會進入電源陷縮狀態

12.4.8 電源管理

- 喚醒鎖核心將會阻止系統進入電源陷縮狀態
  - 例如，在 Android Market 更新應用程式時，將會保持喚醒鎖來確保在更新完成之前系統不會進入睡眠狀態
    - 一旦完成後，Android Market 將釋放喚醒鎖，從而使系統電源陷縮狀態

12.5 轉換 I/O 要求為硬體操作指令

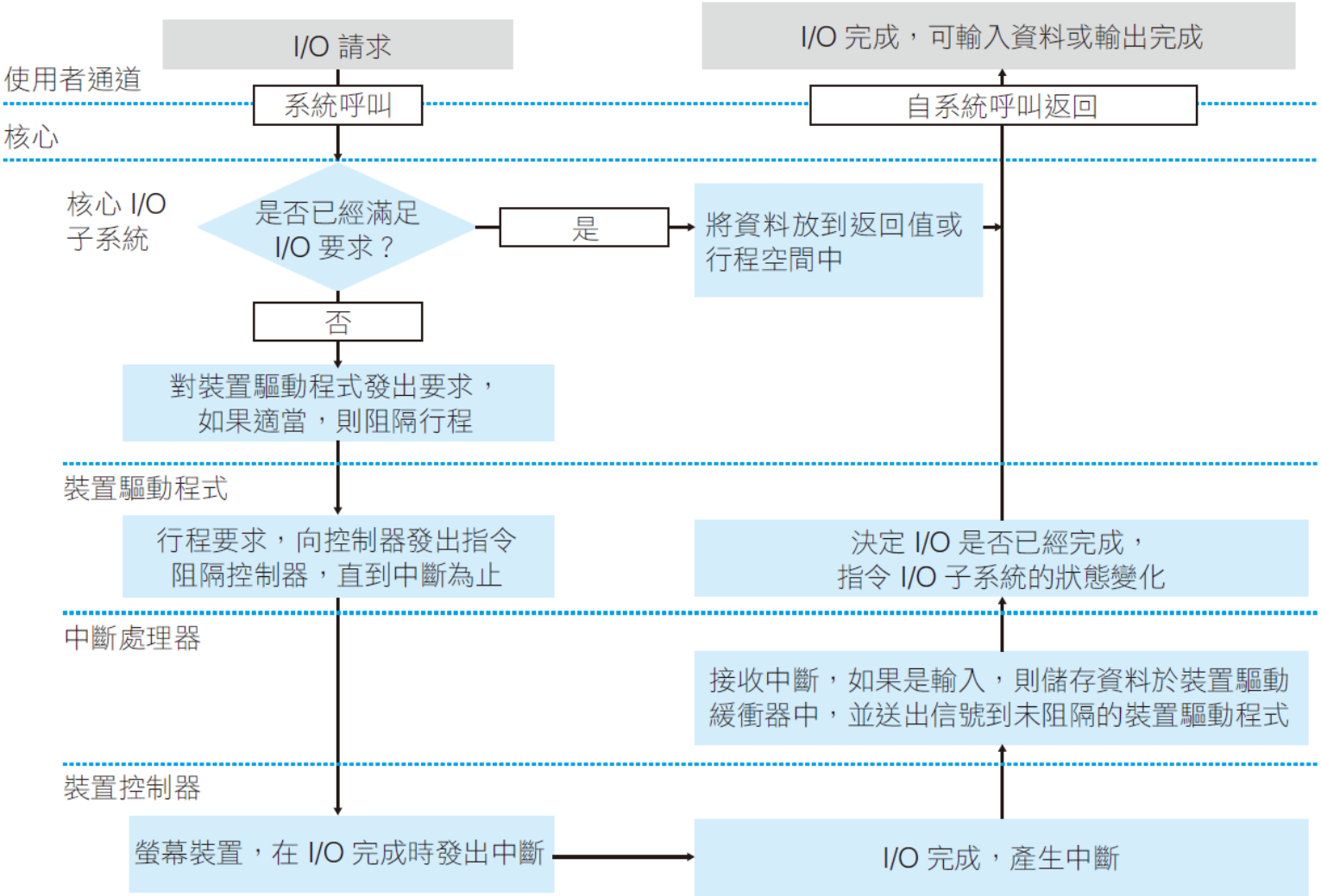


圖 12.14 I/O 請求的生命週期

12.6 STREAMS

- UNIX System V (以及許多後續的 UNIX 版本) 讓應用程式能夠動態組合驅動程式之程式碼的管線
  - 所謂的資料串列是一個在裝置驅動程式與使用者行程之間的全雙工連接
- **STREAMS** 包含
  - **資料串列標頭** (stream head)
  - 可控制裝置的**驅動程式尾端** (driver end)
  - 零或多個存在它們之間的資料**串列模組** (stream module)

12.6 STREAMS

- 每個模組都包含一組佇列——一個讀取佇列和一個寫入佇列
  - 息傳送被用來在佇列間傳輸資料
- 是非同步 (或非阻隔)，除非當使用者行程和資料串列標頭溝通時

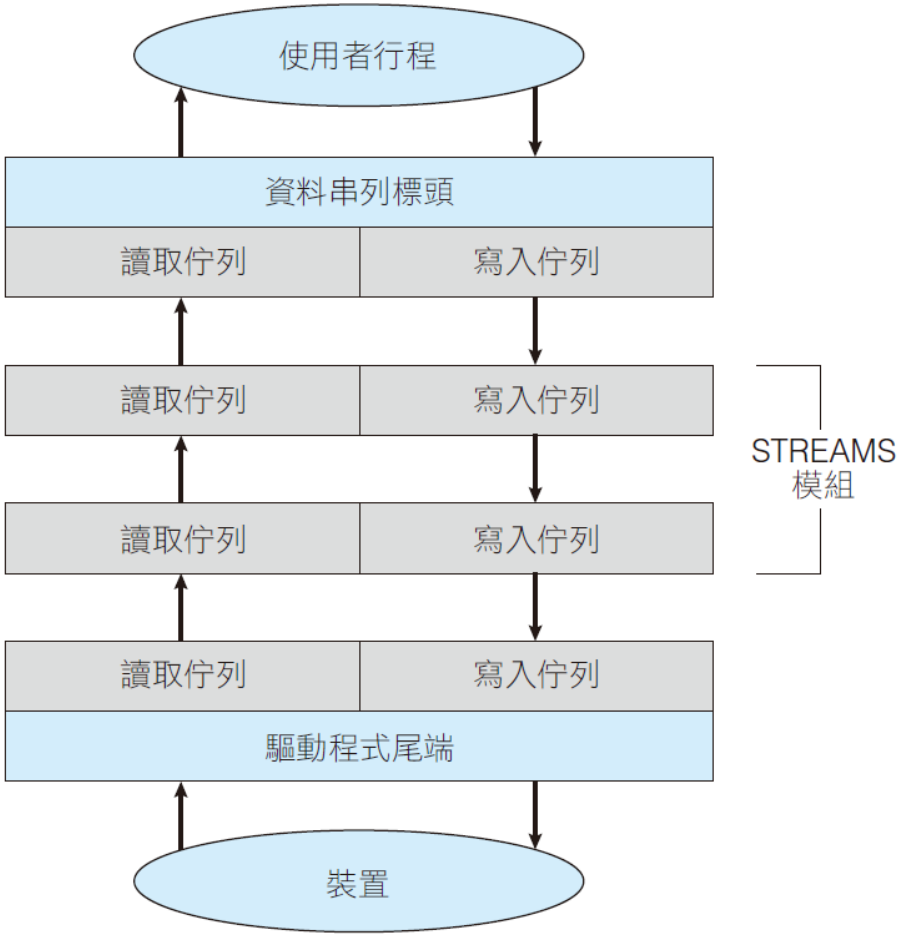
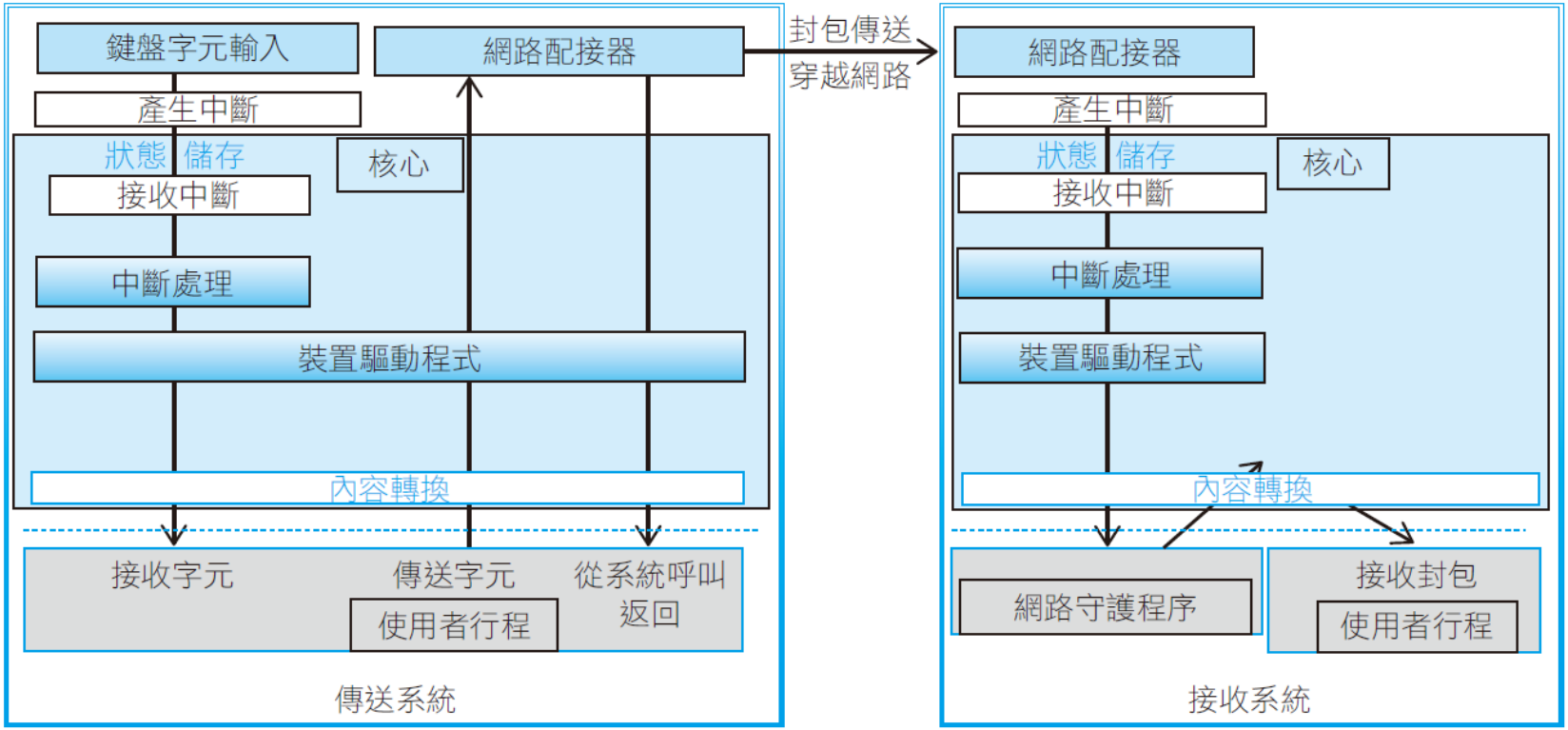


圖 12.15 STREAMS 的結構

12.7 效 能

- I/O 是影響系統效能的主要因素。
- 在執行裝置驅動程式碼及當行程被阻隔與非阻隔時，公平有效地安排其執行順序，將會為 CPU 帶來極大負擔
- I/O 在核心中斷處理器控制方面較無效率
- 當控制器與實體記憶體之間進行資料複製，與核心緩衝區與應用程式資料空間兩處進行資料複製時，會降低記憶體匯流排的速度
- 網路上的交通量也可能造成較高的內容轉換率 (context-switch rate)

圖 12.16　計算機通信


鍵盤字元輸入
網路配接器
產生中斷
狀態 儲存
核心
接收中斷
中斷處理
裝置驅動程式
內容轉換
接收字元
傳送字元
使用者行程
從系統呼叫返回
傳送系統
封包傳送
穿越網路
網路配接器
產生中斷
狀態 儲存
核心
接收中斷
中斷處理
裝置驅動程式
內容轉換
網路守護程序
接收封包
使用者行程
接收系統

12.7 效 能

- 減少內容轉換次數
- 減少在裝置與應用程式之間傳輸時，必須複製的記憶體資料次數
- 藉由使用大量傳輸、智慧型控制器與輪詢 (如果可以將忙碌等待最小化)，降低中斷發生頻率
- 藉由使用 DMA—認知控制器或通道，以降低來自 CPU 簡單資料複製次數的負擔，進而增加並行
- 將處理的基本功能移到硬體之中，使它在裝置控制器之中的操作可以與 CPU 及匯流排的操作同時進行
- 平衡 CPU、記憶體子系統、匯流排與 I/O 之效能，因為任一區域的額外負擔都可能造成其它部份閒置

圖 12.17　裝置功能進程