CH2、IO 與 HW

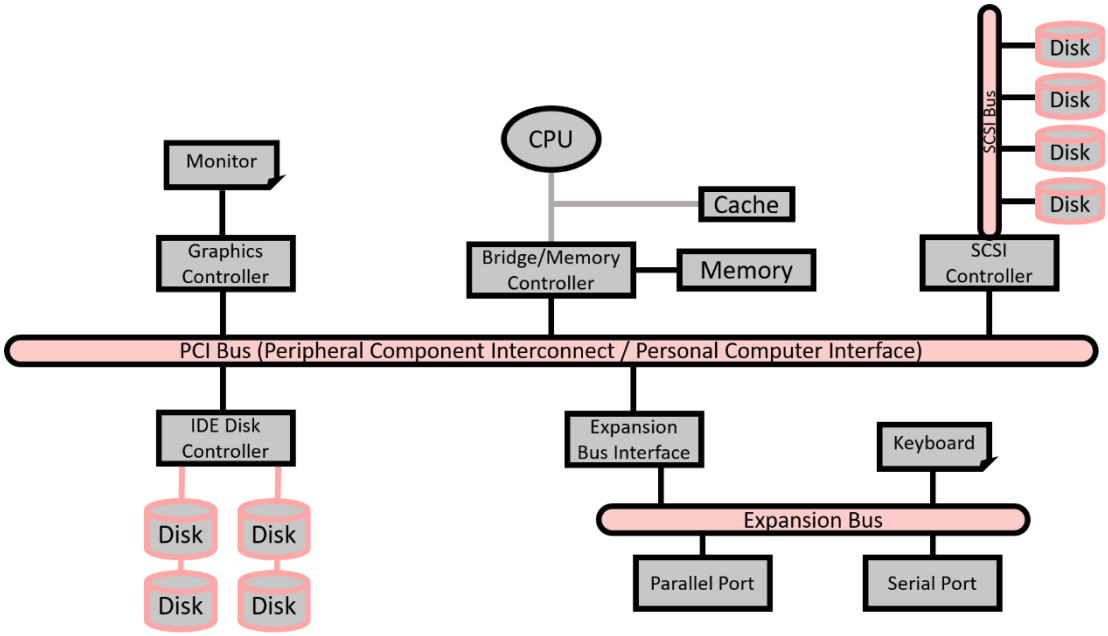
IO 運作與 HW resources Protection

目錄：

I/O 運作方式

IO 是指 Mem 到 Device 之間的傳輸(輸入、輸出)，常見的電腦硬體架構如下圖(基本上除了與 CPU 直接相接的線之外*(較灰色的線條)*，其餘黑色線條都可以算是 IO)

而又因為 IO 的速度，可分為北橋*(North Bridge、Memory hub)*與南橋*(South Bridge、IO hub)*，北橋連接速度較快的 CPU、Memory、High-speed Device*(ex：螢幕)*、南橋則為一般 IO*(ex：Network、USB、HDD...)*



常見的 IO 對應 Memory 位址如下*(利用更改 Memory 位址內之值，來控制 IO Device)*：

IO Address Range *(hexadecimal)*	Device
000-00F	DMA Controller
020-021	Interrupt Controller
040-043	Timer
200-20F	Game Controller
2F8-2FF	Serial Port (Secondary)

320-32F	Hard Disk Controller
378-37F	Parallel Port
3D0-3DF	Graphics Controller
3F0-3F7	Diskette-Drive Controller
3F8-3FF	Serial Port(Primary)

Polling I/O

(一)Def：又叫 Busy-waiting I/O 或 Programmed I/O

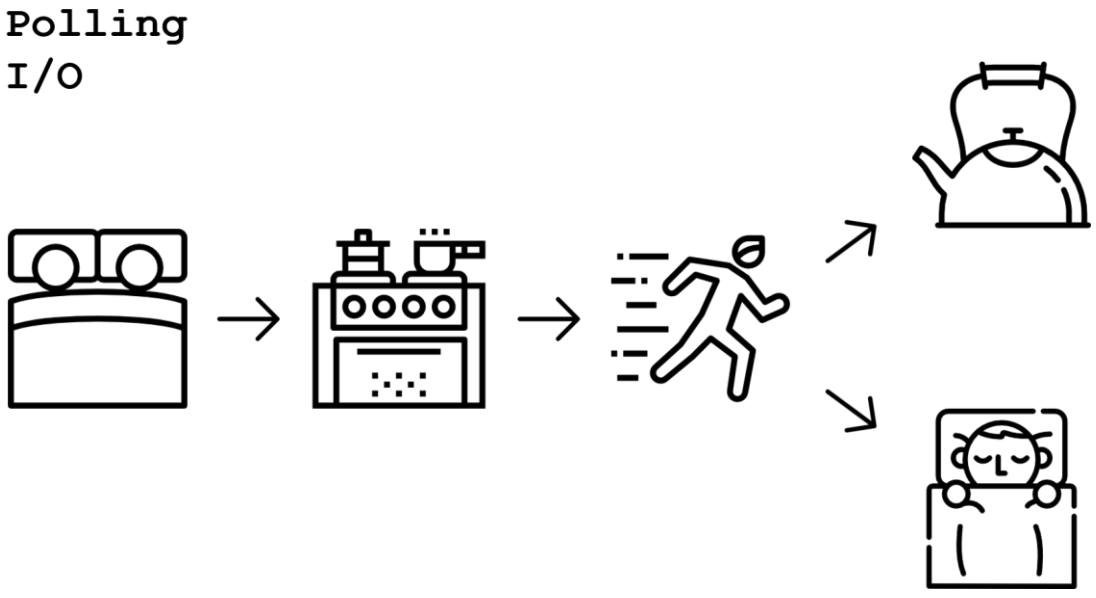
step 如下：

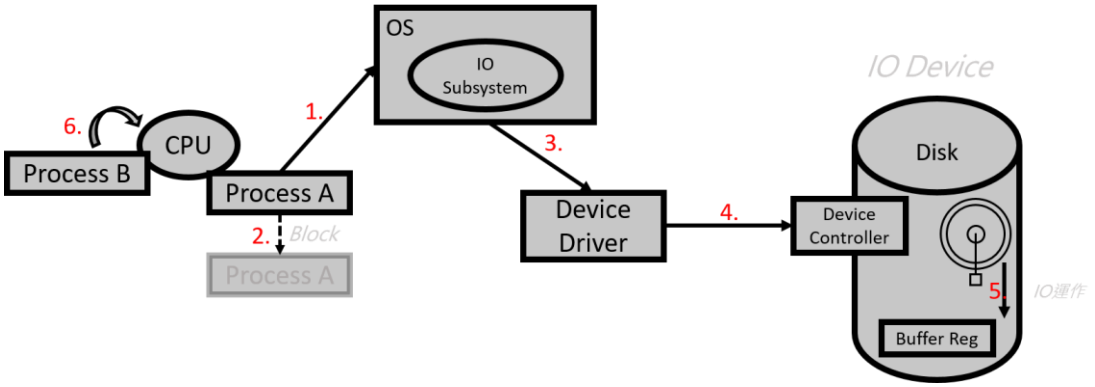
1. user process 發出 I/O request 給 OS
2. OS 收到請求後，(可能)會暫停此 process 執行並執行對應的 system call
3. kernel 的 I/O subsystem 會 pass 請求給 Device driver
4. Device Driver 依此請求設定對應的 I/O Commands 參數給 Device Controller
5. Device Controller 啟動監督 I/O Device 之 I/O 運作進行
6. 在此之時，OS(可能)將 CPU 切給另一個 process 執行
7. 然而，CPU 在執行 process 工作過程中都要不斷去 Polling Device Controller，以確定 I/O 運作是否完成或有 I/O error

比喻：老婆要求煮開水(最簡單的 Polling)

(前情提要：老婆要求簡單的煮開水，但家中煮水壺十分陽春，完全無功能)

1. *(早上起床，)**老婆**發出想喝咖啡的請求給我*
2. *我收到請求後，會暫停和**老婆一起躺在床上**的動作，並執行**煮開水的動作***
3. *我到廚房後，**手**打開**瓦斯爐開關***
4. ***瓦斯爐開關**會依照不同火力大小，打開**瓦斯爐***
5. ***瓦斯爐**啟動火焰，開始**燒水的動作***
6. *等水燒開的時候，我(可能)會去小孩的房間**叫小孩起床***
7. *然而，因為家裡的煮水壺不會自動提醒水已煮開，所以需要**不斷去廚房檢查水是否已燒開**，以確定不會燒完水的危險發生*





(二)缺點：CPU 耗費大量時間用於 polling I/O Device Controller 上，並未全用於 process execution 上，故 CPU utilization 低，throughput 不高

Interrupt I/O (中斷式)

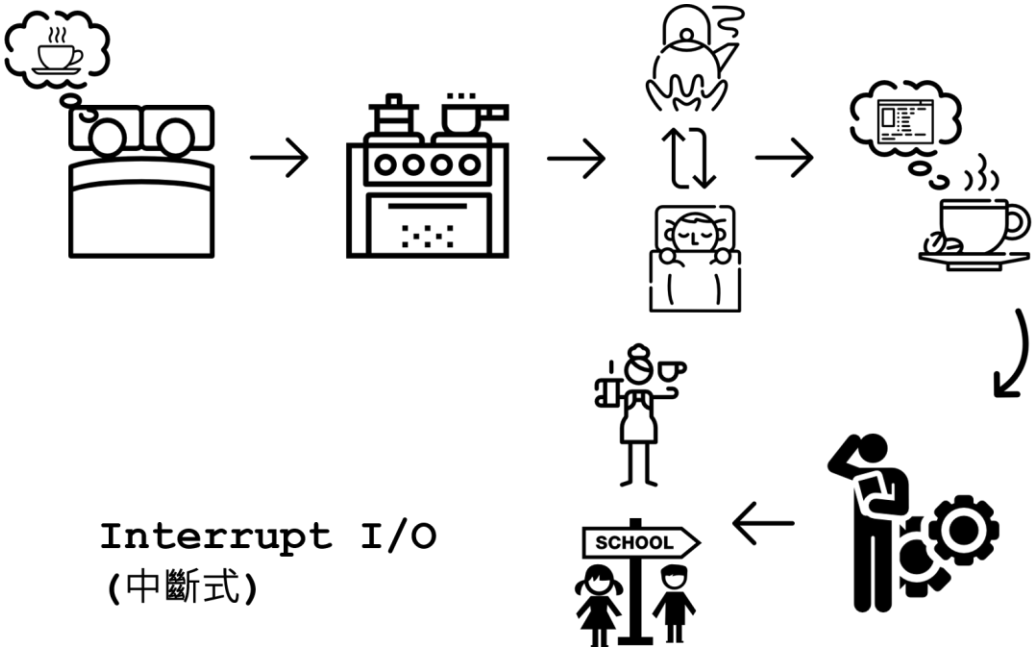
(一)Def：step 如下(1-6 同 polling)：

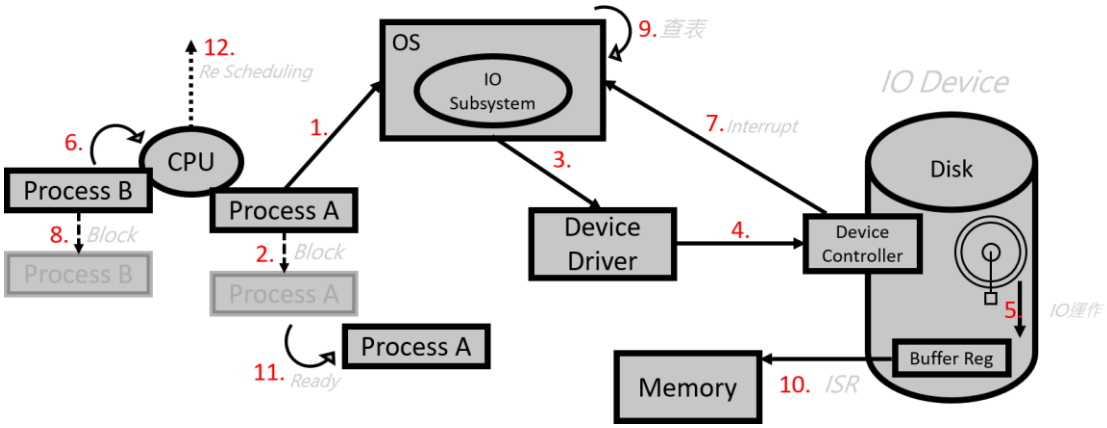
1. *user process 發出 I/O request 給 OS*
2. *OS 收到請求後，(可能)會暫停此 process 執行並執行對應的 system call*
3. *kernel 的 I/O subsystem 會 pass 請求給 Device driver*
4. *Device Driver 依此請求設定對應的 I/O Commands 參數給 Device Controller*
5. *Device Controller 啟動監督 I/O Device 之 I/O 運作進行*
6. *在此之時，OS(可能)將 CPU 切給另一個 process 執行*
7. 當 I/O 運作完成，Device Controller 會發出"I/O Completed" Interrupt 通知 OS (CPU)
8. OS 收到中斷後，(可能)會先暫停目前 Process 的執行
9. OS 必須查詢 Interrupt Vector，確認中斷發生，同時也要找到該中斷之服務處理程式(ISR：Interrupt Service Route 的位址)
10. Jump to ISR 位址，執行 ISR
11. ISR 完成後，return control to kernel，kernel 也許作一些通知工作
12. 恢復(resume)原先中斷之前的工作或交由 CPU Scheduler 決定

比喻：老婆要求煮咖啡(較進階的 Interrupt)

(前情提要：老婆要求較困難的泡咖啡，而家中煮水壺有進步，煮開時會發出通知聲)

1. *(早上起床，)**老婆**發出想喝咖啡的請求給我*
2. *我收到請求後，會暫停和**老婆一起躺在床上**的動作，並執行**煮開水的動作***
3. *我到廚房後，**手**打開**瓦斯爐開關***
4. ***瓦斯爐開關**會依照不同火力大小，打開**瓦斯爐***
5. ***瓦斯爐**啟動火焰，開始**燒水的動作***
6. *等水燒開的時候，我(可能)會去小孩的房間**叫小孩起床***
7. *當水煮開後，煮水壺會**自動發出通知聲**，來通知**我***
8. ***聽到**水煮開的**通知聲**後，我會先暫停叫小孩的動作，到廚房關水壺*
9. *到了廚房，**確認開水是否已經煮開**，而後查詢老婆想要咖啡之**配方***
10. *執行該咖啡**配方***
11. *完成咖啡後，**控制權回到我**(原本控制權是在配方上)*
12. ***回到**老婆身邊給咖啡、或執行更緊急的事(ex：小孩上學要遲到了)*





Interrupt	Address	Type	Function
00h	0000:0000h	Processor	Divide By Zero
01h	0000:0004h	Processor	Single Step
02h	0000:0008h	Processor	Non-maskable Interrupt (NMI)
03h	0000:000Ch	Processor	Breakpoint Instruction
04h	0000:0010h	Processor	Overflow Instruction
05h	0000:0014h	BIOS/Software	Print Screen
05h	0000:0014h	Hardware	Bounds Exception (80286, 80386)
06h	0000:0018h	Hardware	Invalid Op Code (80286, 80386)
07h	0000:001Ch	Hardware	Math Coprocessor Not Present
08h	0000:0020h	Hardware	Double Exception Error (80286, 80386) (AT Only)
08h	0000:0020h	Hardware	System Timer -IRQ 0
09h	0000:0024h	Hardware	Keyboard -IRQ 1
09h	0000:0024h	Hardware	Math Coprocessor Segment Overrun (80286, 80386) (AT Only)
0Ah	0000:0028h	Hardware	IRQ 2 -Cascade from Second programmable Interrupt Controller
0Ah		Hardware	Invalid Task Segment State (80286, 80286) (AT Only)
0Ah		Hardware	IRQ 2 -General Adapter Use (PC Only)
0Bh	0000:002Ch	Hardware	IRQ 3 -Serial Communications (COM 2)
0Bh		Hardware	Segment Not Present (80286, 80386)
0Ch	0000:0030h	Hardware	IRQ 4 -Serial Communications (COM 1)
0Ch		Hardware	Stack Segment Overflow (80286, 80386)
0Dh	0000:0034h	Hardware	Parallel Printer (LPT 2) (AT Only)
0Dh		Hardware	IRQ 5 -Fixed Disk (XT Only)
0Dh		Software	General Protection Fault (80286, 80386)
0Eh	0000:0038h	Software	IRQ 6-Diskette Drive Controller
0Eh		Software	Page Fault (80386 Only)
0Fh	0000:003Ch	Software	IRQ 7 -Parallel printer (LPT 1)

補充：中斷向量表

IRQ 0	System Timer
IRQ 1	Keyboard
IRQ 2	Cascaded with IRQ 9
IRQ 3	Default COM2 and COM4
IRQ 4	Default COM1 and COM3
IRQ 5	LPT2
IRQ 6	Floppy Drive Controller
IRQ 7	LPT1
…	…

IO 裝置對應代碼：最初因為電腦周邊並不多，所以只設計了 8 個，後來不夠用而又增加了一組，但是第一組的最後一個要作為與第二組溝通使用，所以實際上只有 15 個而已。
IRQ=Interrupt request

中斷處理的方式：
查表即先確認中斷號*(ex：00h，即了解發生了"Divide By Zero"的中斷)*，再至中斷表中找到對應該『中斷處理的程式碼』所存放的起始位址*(ex：處理"Divide By Zero"的中斷處理碼，起始位址存在 0000:0000h)*，而後跳至該起始位址*(0000:0000h)*，並執行中斷處理程式碼(ISR)

(二)優點：CPU 不須耗費時間用於 polling I/O Device，而是可以用於 Process execution 上，所以 CPU utilization 提高 throughput
(三)缺點：
1. Interrupt 之處理仍需耗費 CPU 時間
   如果 I/O 運作時間 < Interrupt 處理時間，則使用 Interrupt I/O 就不划算，所以 polling I/O 仍有其必要性
2. 若中斷發生頻率太高、會佔用幾乎所有的 CPU time，則系統效能會很差
3. CPU 仍需耗費一些時間用於監督 Device 與 Memory 之間傳輸的過程

[補充]若 ISR 正在執行時，又有其他中斷發生，則該如何處理？
法一：Disable 其他中斷(但不適用 Real-Time System、Time Sharing System)
法二：Interrupt 有優先權高低之分

比喻：
使命必達、勇往之前
老婆最大，其他人先別吵

DMA (Direct Memory Access)

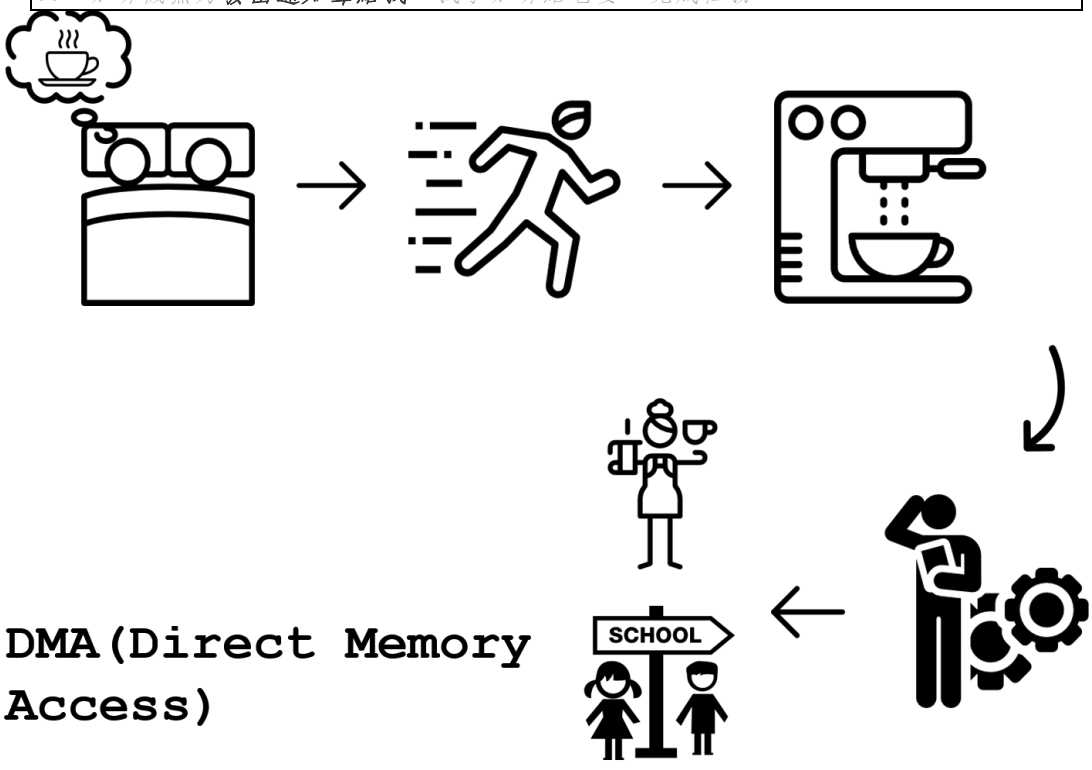
(一)Def：DMA Controller 負責 I/O Device 與 Memory 之間的 Data transfer 工作過程中無需 CPU 之多與監督，所以 CPU 有更多時間用於 process execution 上

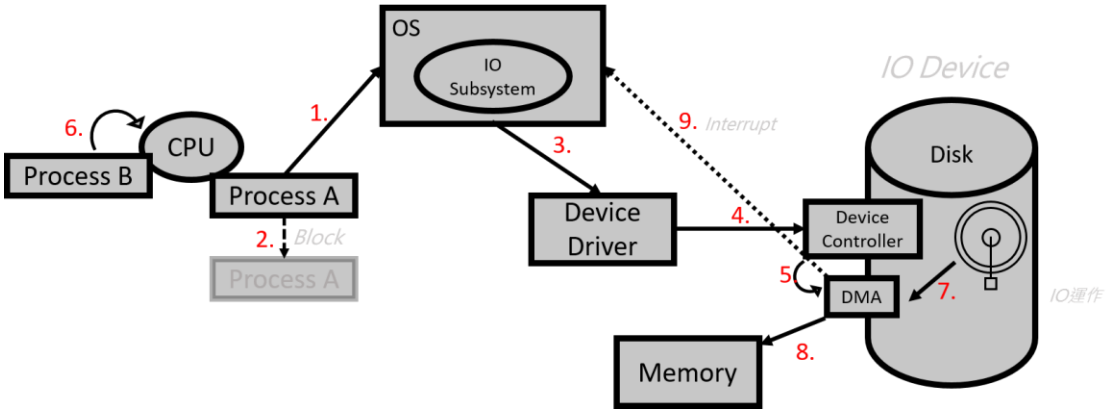
1. *user process 發出 I/O request 給 OS*
2. *OS 收到請求後，(可能)會暫停此 process 執行並執行對應的 system call*
3. *kernel 的 I/O subsystem 會 pass 請求給 Device driver*
4. *Device Driver 依此請求設定對應的 I/O Commands 參數給 Device Controller*
5. *Device Controller 啟動 DMA 以進行資料的傳輸*
6. *在此之時，OS(可能)將 CPU 切給另一個 process 執行*
7. *Device Controller 將資料傳至 DMA Controller*
8. *DMA Controller 將資料傳送至 Memory 目標位址*
9. *完成傳輸後，發出中斷以通知 kernel 資料傳送完成*

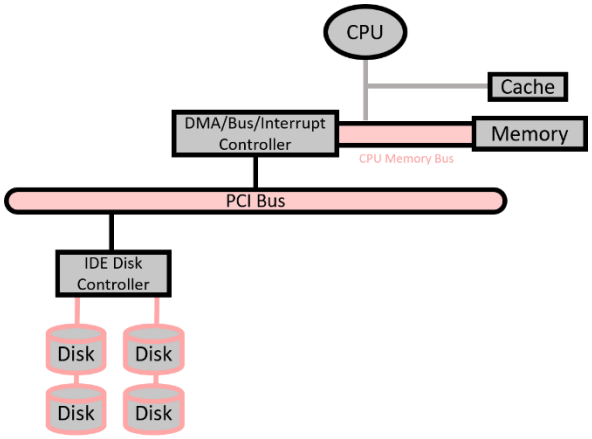
比喻：老婆要求煮咖啡，但已買咖啡機(可視為較進階的 Interrupt)

(前情提要：老婆要求較困難的泡咖啡，但家裡已經買了新的咖啡機)

1. *(早上起床，)**老婆**發出想喝咖啡的請求給我*
2. *我收到請求後，會暫停和**老婆一起躺在床上**的動作，並執行**泡咖啡的動作***
3. *我到廚房後，因為有了**咖啡機**，所以只要將老婆想要的**咖啡種釋資訊**，**用手按下**對應的**按鍵***
4. *按鍵會啟動咖啡機的**電腦***
5. *主機確認好後，**機器本體**會開始進行泡咖啡的動作*
6. *等機器泡咖啡的過程，我(可能)會去小孩的房間叫小孩起床*
7. *咖啡機本體運作中*
8. *咖啡機完成咖啡，**倒入杯子***
9. *咖啡機煮好**發出通知聲給我**，我拿咖啡給老婆、完成任務*







(二)優點：

1. CPU utilization 更高
2. 適合用在 Blocked transfer oriented I/O Device*(不適用在 Byte-transfer oriented I/O Device)*上(代表中斷發生頻率不致過高。ex：Disk)

(三)缺點：引用 DMA Controller 會增加 HW 設計複雜度(Complicated the HW design)

原因：DMA Controller 會與 CPU 爭搶 Memory、Bus 之資源使用權，若 DMA 佔用了 Memory、Bus 時，CPU 要被迫等待

[補充]

1. DMA Controller 採用"cycle stealing"的技術*(或 Interleaving[恐])* 與 CPU 輪番(交錯)使用 Memory 和 Bus。
   如果 CPU 因 DMA Controller 發生 conflict(同時都要 Memory/Bus)則會給 DMA Controller 較高的優先權
   人如果要喝水，可以自行選擇喝多喝少、且能夠觀查哪個時間裝水不會影響到咖啡機的運作，故水流的優先權會給予咖啡機較高的使用優先權
2. 通常系統會給予"對該資源需求量、頻率較小"的對象有較高的優先權，會獲得平均等待時間較小、產出較高。
   人可能隨時會突然口渴想喝水，但咖啡機需要有指令才會運作，且時間是相對可預測的，所以優先權較高
3. 機器指令 Stages：IF(Instruction Fetch)、DE(Decode)、FO(Fetch Operands)、EX(Execution)、WM(Write Memory)

	CPU 會 Memory Access	DMA 會用到 Memory
IF	會	衝突(Conflict)
DE	不會	沒問題
FO	可能	沒問題或衝突
EX	不會	沒問題
WM	可能	沒問題或衝突

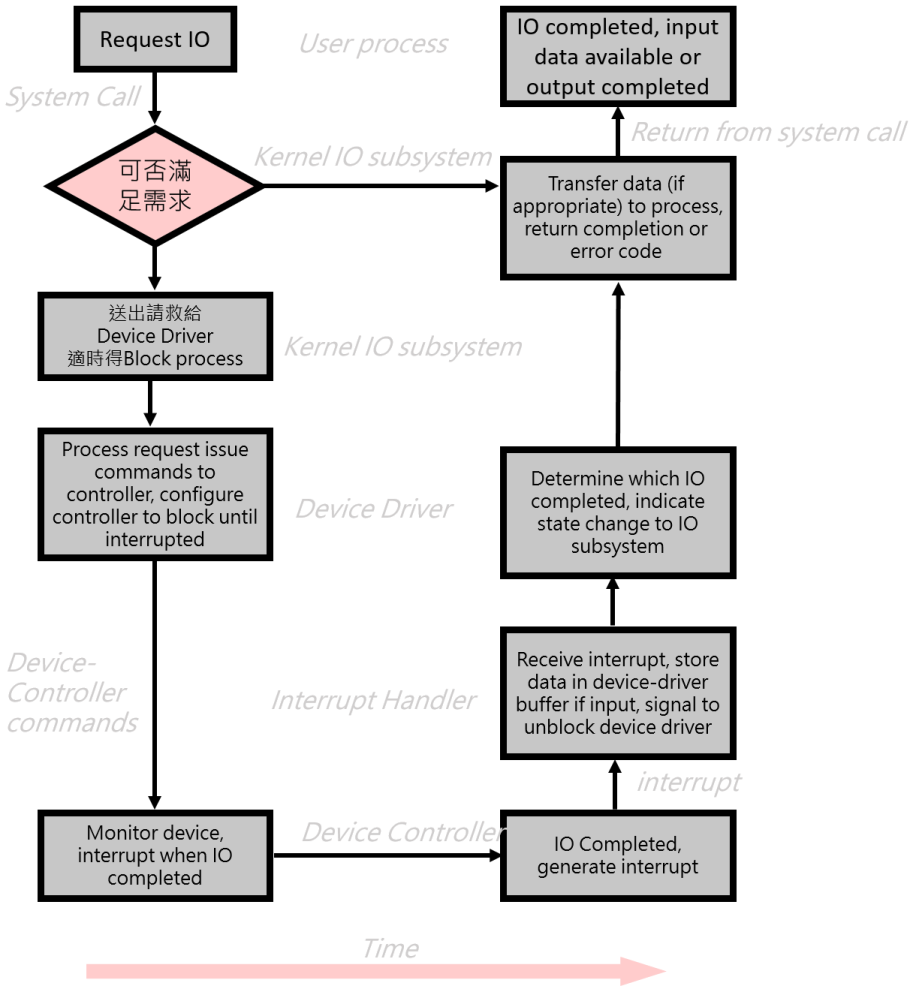
CPU 設定 DMAC(Controller)的資料有：

1. IO Commands
2. Device data source/destination 位址
3. Memory 之 source/destination 位址
4. Timer：用以確認 transfer length*(依長度設定一 Timer 數值，開始傳輸即 Timer 倒數，倒數至 0 就表示傳輸完成)*

咖啡機比喻：

1. *選擇咖啡種類並按下咖啡機*
2. *咖啡機相關資料(廚房可能有很多台)*
3. *開水來源的管線、最終要倒入的地方(杯子)*
4. *依杯子大小設定時間(倒入時間較長則份量較多、時間較短則份量較少)*

Life cycle of I/O Request (via Interrupted I/O)



Blocking and Non-blocking I/O

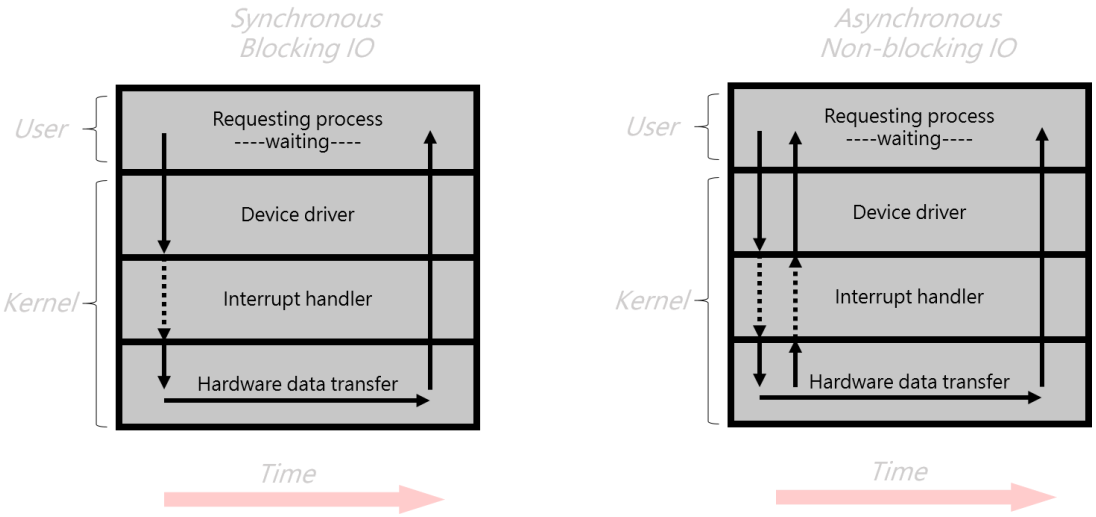
1. Blocking I/O：process suspended until I/O completed
   甲、優點：Easy to use and understand
   乙、Insufficient for some needs
2. Non-blocking I/O：控制權馬上還給 user process after I/O request.
   I/O calls returns as much as available. (ex：user interface data copy. Returns quickly with count of Bytes read or write.)
3. Asynchronous-I/O：Process runs while I/O executes. Difficult to use. I/O subsystem signals process when I/O completed.

Blocking 與 Non-Blocking 的差別如同廚房是否擁擠：Blocking 為擁擠的公共廚房，要等到水煮開才進去泡咖啡，若水尚未煮開，進去不但無法做事，而且會惹人厭；Non-Blocking 則是如同在自家的廚房，是否待在廚房等水開、或者到別的地方(ex 小孩房間叫小孩起床)皆可

差異：

1. Asynchronous：整個 I/O 完成，才通知 process
2. Non-blocking：I/O 完成 Data return ASAP(有多少傳多少)

Asynchronous 就如同一般煮水壺，水開時才提醒；Non-Blocking 則是像有溫度提醒的煮水壺，會時時提醒目前溫度為何



(一)Interrupt 種類：

甲、分類一：

1. External Interrupt：CPU 以外的週邊設備控制卡…等，所發出的。(ex：”I/O Completed”、”I/O error”、”monitor”)
2. Internal Interrupt：CPU 執行 process 過程中，遭過重大錯誤而引發。(ex：Divide-by-zero、執行非法的特殊指令…等)
3. Software Interrupt：user process 在執行中，若需要 OS 提供服務時，必須發出此中斷，目的是通知 OS，請它執行對應的服務請求。(ex：I/O request)

乙、分類二：Interrupt 和 Trap 2 種

Interrupt	Trap
HW generated change of control flow	SW generated interrupt
例：設備發出：I/O-Completed、I/O-error、Machine-check、Time-out by Timer	用途有 2： 1. Catch the arithmetic error (重大錯誤)。例：Divide by zero、非法指令(記憶體) 2. user process 執行需要 OS 提供服務時，也會發出 Trap 通知 OS。例：I/O request

丙、分類三：2 種(背後哲學，Interrupt 也有優先權之分)

1. Maskable Interrupt：此類中斷發生時，可延後處理，不一定要馬上處理(SW Interrupt 優先權較低)
2. Non-maskable Interrupt：此類中斷必須立刻處理。ex：Internal Interrupt、I/O error

HW Resources Protection(基礎建設)

Dual mode operation(雙核模式)：

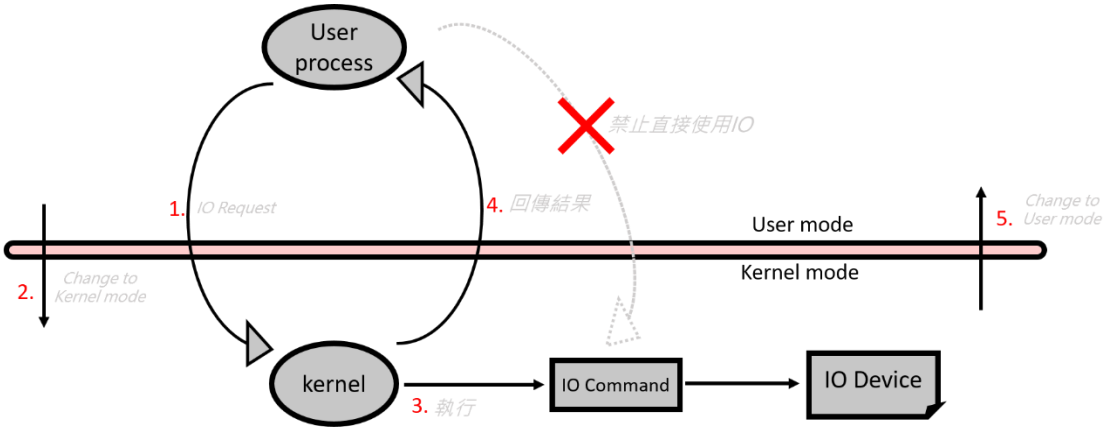
(一)Def：

甲、kernel mode(又稱 System mode、supervisor mode、*monitor mode(已刪)*、privilege mode)：此刻是 kernel 取得系統控制權(取得 CPU 執行)、允許 privilege instruction 在 kernel mode 下執行。Mode bit=0

乙、user mode：代表 user process 取得 CPU 執行在此 mode。不允許執行特權指令。Mode bit=1

比喻：銀行櫃台 (或者更接近的古蘭閣 XD)

銀行為了確保財富的安全，設立了嚴密的櫃台，來分開大廳與金庫。user mode 就如同大廳，客人/使用者可以隨意進出與使用；但 kernel mode 就如同櫃台後面的金庫，只有銀行職員可以進出與取用物品，例如珠寶。以此來確定不會有外人的入侵、偷竊或破壞。



Privileged instruction(特權指令)

(一)Def：任何可能會造成系統重大危害的指令，可設為特權指令。一旦 user mode 下執行，會發出 Trap 通知 OS，將此 user process 終止

例：

1. Turn-off(Disable) interrupt
2. Clear Memory
3. I/O Instruction (for I/O protection)
4. Timer 值 set/ change (for CPU protection)
5. Base/Limit register 修改/set for memory protection
6. change mode from user to kernel mode

> **解釋(銀行櫃台)：**
> 1. *Turn-Off Interrupt：**休市**，可能有更重要的事，例如**盤點***
> 2. *Clear Memory 即**清理金庫***
> 3. *IO 如同櫃台後方或金庫內的**機器***
> 4. *Timer 即**時鐘**，隨意更改時間，會影響到股市*
> 5. *Memory Base & Limit 如同**設定金庫大小***
> 6. *Change mode 如同**上班**：從大廳進到櫃台，反之則可以(下班)*

例 2[Ex]p3-42

33.

1. set value of Timer
2. Read the clock
3. clear Memory
4. Turn off interrupt
5. switch from user to monitor mode

kernel mode：1、3、4、5

34.

1. change to user mode
2. change to kernel mode
3. Read from monitor memory
4. Write into monitor memory
5. Fetch an instruction from monitor memory
6. Turn on timer interrupt
7. Turn off timer interrupt

kernel mode：2、3、4、7

為何構成保護基礎：

HW Resources Protection：

I/O：

(一)目的：由於 I/O 運作較為複雜，為了降低 user process 操控 I/O 之複雜度及避免 user process 對 I/O-Devices 之不當操，所以才有 I/O protection

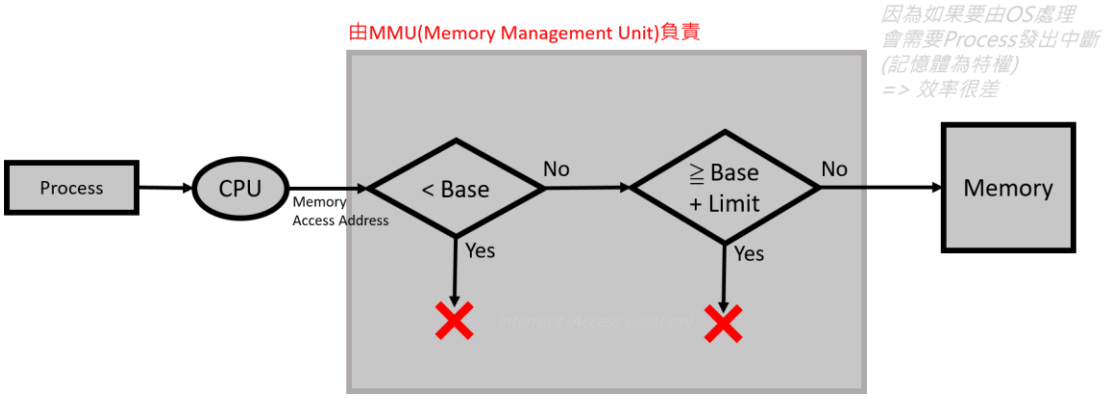
(二)作法：將所有 I/O 指令設為特權，配合 dual-mode，一律讓 user process 委託 kernel 執行 I/O 操作

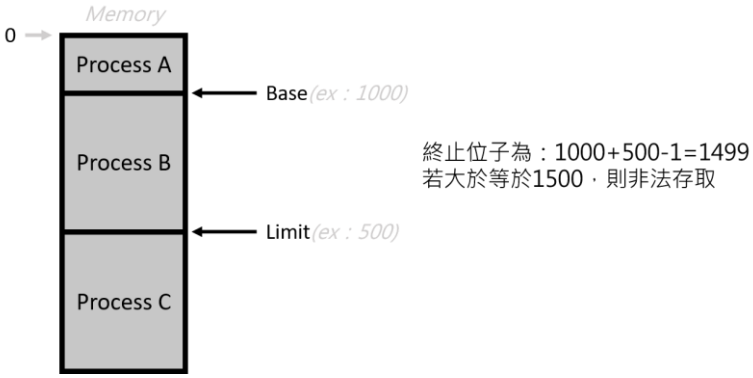
Memory

(一)目的：防止 user process 存取其他 user processes 之 memory area 及 kernel memory area

(二)作法：(以 Contiguous Allocation 為例)針對每一 process，kernel 提供一套 Register 叫 Base/Limit Register；Base 記錄 process 起始位置，而 Limit 記錄 process 大小，將來，Process 執行，會進行下列的 checking

此外，Base/Limit 值之修改須設為特權指令





CPU：

(一)目的：防止 user process 無限期/長期佔用 CPU，而不釋放

(二)作法：利用 Timer 實施，同時 OS 會規定 process 使用 CPU 最大配額值(Max time Quantum)當 process 取得 CPU 後，Timer 初始值即設為 Max Time Quantum 值，隨著 process 執行時間增加，Timer 值會逐步遞減，直至 Timer 發出 Time-out 中斷，通知 OS 便可強迫此 process 放掉 CPU。此外，Timer 值之修改也需設為特權指令

例：

A ____ can be used to prevent a user program from never returning control to the operating system ?

A) portal

B) program counter

C) firewall

D) timer

D