Hw2 : the Q&A puzzle

critical section problem– 在一段程式碼中，乍看之下為單一操作的指令，實際上在較低階的語言中為**複數操作**，例如

C 語言	組合語言
count++	register1 = count register1 = register1 + 1 count = register1

在系統實際運行時，無法確保 processA 在操作/變更共用變數時 processB 不會同時更動/存取共用變數，例如

P:Producer | C:Consumer	
P: register1 = count	{register1 = 5}
P: register1 = register1 + 1	{register1 = 6}
C: register2 = count	{register2 = 5}
C: register2 = register2 - 1	{register2 = 4}
P: count = register1	{count = 6}
C: count = register2	{count = 4}

因此我們必須管制 process 進入 CS(critical section)程式區段，以確保 process 在執行 CS 時會完整操作完該區段的指令，才讓其他的 process 進入各自的 CS。

Peterson's Solution 為 CS 問題的一個解法，具體作法如下

processA	**processB**

do{
flag[i] = true;
turn = j;
while(flag[j] && turn == j);
 	
Critical Section
 	
Flag[i] = false;
 	
Remainder Section
}while(true)
	do{
	flag[j] = true;
	turn = i;
	while(flag[i] && turn == i);
	
	Critical Section
	
	Flag[j] = false;
	
	Remainder Section
	}while(true)

其中:

flag - 執行 CS 的需求;

turn - 輪到 A/B 進入 CS;

藍字區域 - 表達自己有進入 CS 的需求，但禮讓對方，若對方無需求則可以先進入 CS。

紅字區域 - 表達自己已執行完 CS，無進入 CS 的需求。

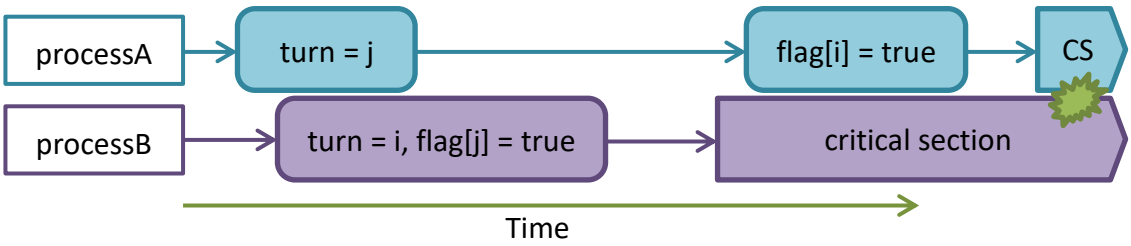
如此即可避免兩個 process 間的 Race Condition，同時也不會有效率問題或卡住的情況。

然而，在現代硬體上運行 Peterson's Solution 演算法可能會出現問題，為了最佳化系統效能，處理器或編譯器可能會**重新排序**操作順序

processA	**processB**

do{
turn = j;
flag[i] = true;
while(flag[j] && turn == j);
 	
Critical Section
 	
Flag[i] = false;
 	
Remainder Section
}while(true)
	do{
	turn = i;
	flag[j] = true;
	while(flag[i] && turn == i);
	
	Critical Section
	
	Flag[j] = false;
	
	Remainder Section
	}while(true)

橙色區域為被調換後的順序，可能會產生兩個 process 同時進入 CS 的情況



為了解決這個問題，我們可以使用記憶體屏障(Memory Barriers) 令**在其之前/後的指令**不會因為系統最佳化效能而亂序，使處理器/編譯器必須按照順序執行

processA	**processB**

do{
flag[i] = true;
memory_barrirer();
turn = j;
while(flag[j] && turn == j);
 	
Critical Section
 	
Flag[i] = false;
 	
Remainder Section
}while(true)
	do{
	flag[j] = true;
	memory_barrirer();
	turn = i;
	while(flag[i] && turn == i);
	
	Critical Section
	
	Flag[j] = false;
	
	Remainder Section
	}while(true)

其中綠色區域根據編譯器可分為下列幾種

gcc	
asm volatile("" ::: "memory");	__sync_synchronize
Microsoft Visual C++	
_ReadWriteBarrier()	MemoryBarrier()
Intel C++	
__memory_barrier()	