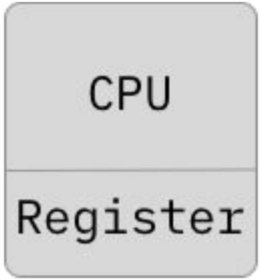
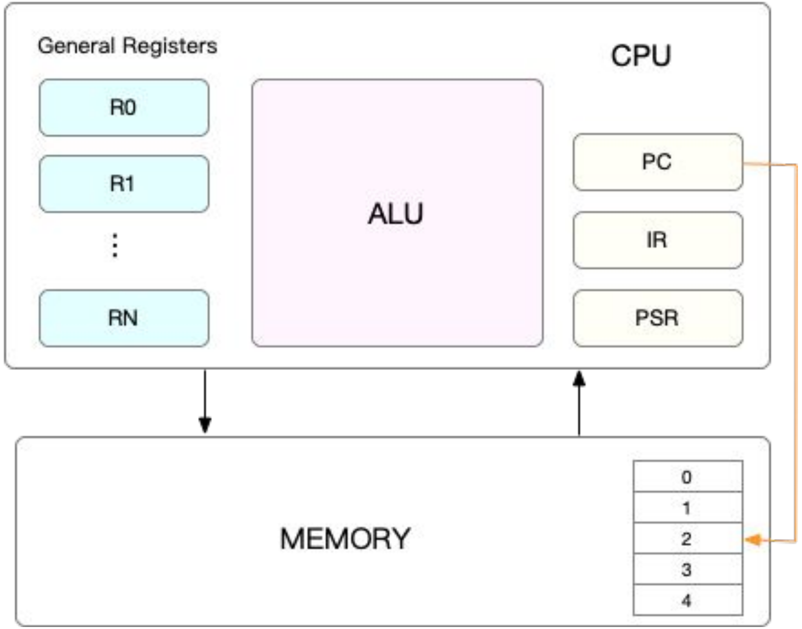
Context Switch

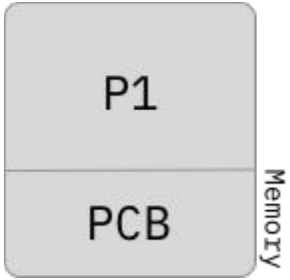
Ian.Wu


CPU
Register

CPU context


General Registers
CPU
R0
R1
RN
ALU
PC
IR
PSR
MEMORY
0
1
2
3
4

Process Controller Block



PCB 負責記錄 Process 相關資訊, 所以又被稱為 Process descriptor

負責記錄的資訊主要分為三種:

Process identification,

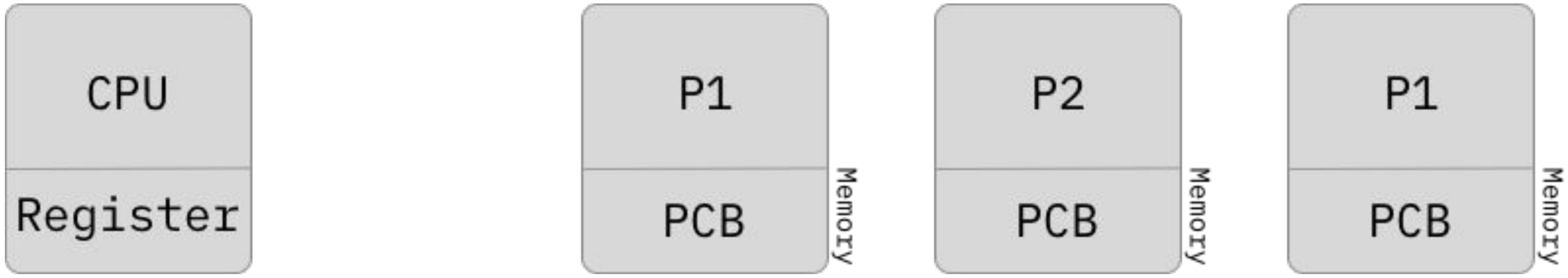
Process state,

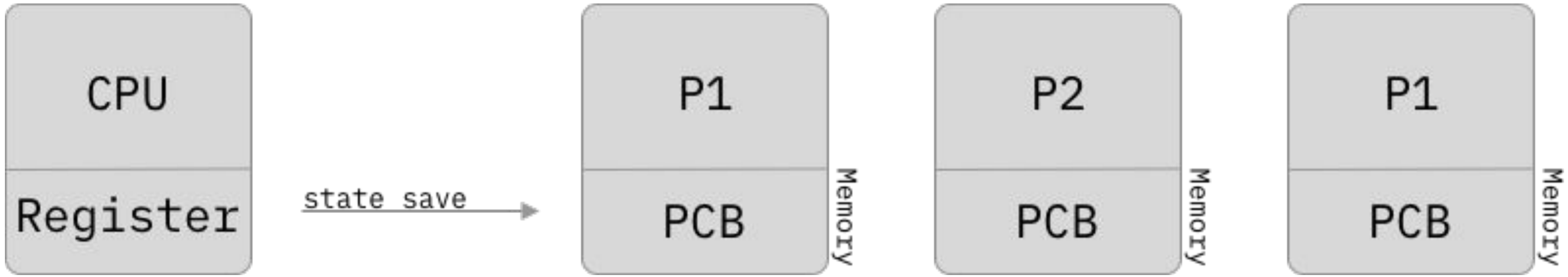
Process control

其中跟 context-switch 最相關的是

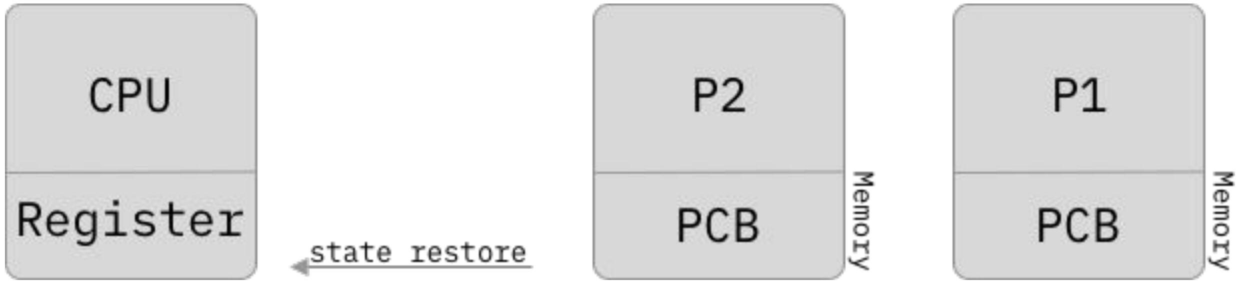
Process state

CPU 在發生 Process context-switch 的時候會把 Register 中的資訊記錄在 PCB

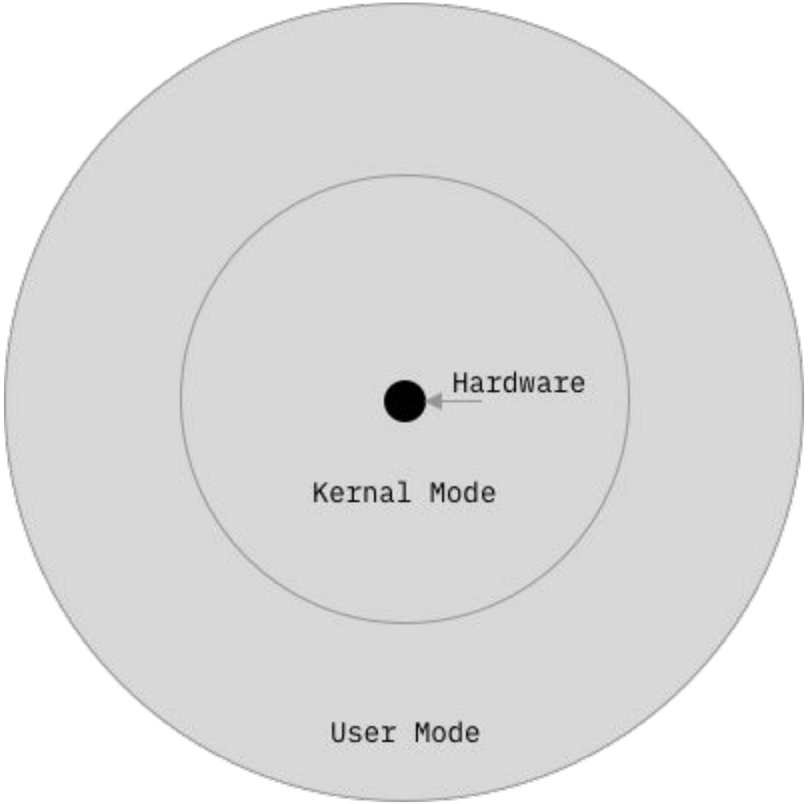

CPU
Register
P1
PCB
Memory
P2
PCB
Memory
P1
PCB
Memory



CPU
Register
state save
P1
PCB
Memory
P2
PCB
Memory
P1
PCB
Memory


CPU
Register
state restore
P2
PCB
Memory
P1
PCB
Memory

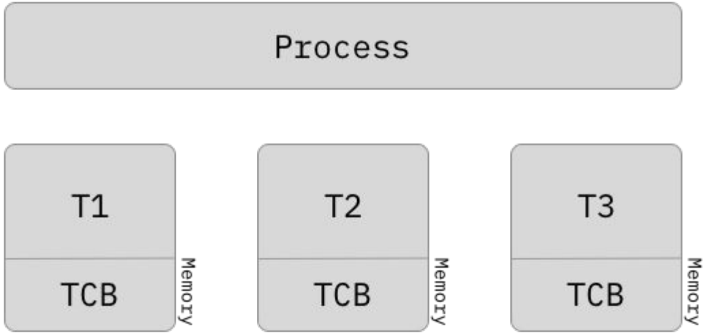
OS @ run (kernel mode)	**Hardware**	**Program (user mode)**
		Process A
		...
	timer interrupt	
	save regs(A) → k-stack(A)	
	move to kernel mode	
	jump to trap handler	
Handle the trap		
Call `switch()` routine		
save regs(A) → proc_t(A)		
restore regs(B) ← proc_t(B)		
switch to k-stack(B)		
return-from-trap (into B)		
	restore regs(B) ← k-stack(B)	
	move to user mode	
	jump to B's PC	
		Process B
		...



system call

system call 的起源跟世界第一臺超級電腦有關，Atlas，當時的程式有把硬體調用相關的 supervisor functions 以及計算用的 mathematical procedures 分開。例如：讀取磁帶的前 512 KiB，計算平方根。

OS @ run (kernel mode)	**Hardware**	**Program (user mode)**
Create entry for process list		
Allocate memory for program		
Load program into memory		
Setup user stack with argv		
Fill kernel stack with reg/PC		
return-from-trap		
	restore regs	
	(from kernel stack)	
	move to user mode	
	jump to main	
		Run main()
		...
		Call system call
		trap into OS
	save regs	
	(to kernel stack)	
	move to kernel mode	
	jump to trap handler	
Handle trap		
Do work of syscall		
return-from-trap		
	restore regs	
	(from kernel stack)	
	move to user mode	
	jump to PC after trap	
		...
		return from main
		trap (via `exit()`)



Thread context-switch

Thread context-switch 為了更新 Thread 狀態到 TCB(Memory), 這也會產生硬體讀寫

會發生 context-switch 的情況

- 當 Process/Thread 被分配到的時間被耗盡
  - 屬於 non voluntary context switches
- 當 Process/(Thread?) 資源不足而要求更多資源
  - 例如 Memory 不夠，需要 allocate 更多內存。
- sleep()
- Higher Priority Process
- Hardware Interrupt
  - 文章提到兩次很相似的概念，我有點分不清。而且 interrupt process 不就是 higher priority process 嗎？

Summary

- Process/Thread context-switch 保存跟恢復狀態需要讀寫硬體(Memory)
- 我們可以藉由工具觀測 context-switch 以及 interrupt 來判斷 CPU 的使用狀況, 如果系統資源耗費在不必要的讀寫硬體, 會排擠到真正有需要的程式運行

我的問題

- Process context-switch 跟 Thread context-switch 相比就只是記憶體大塊一點的讀寫硬體?
- context-switch 跟 system call 的差別是什麼？
- 我的測試環境 OSX, VirtualBox, Debian 10 沒辦法模擬出 interrupt 的情境，有其他人可以模擬出 interrupt 的情境嗎？

Reference

https://en.wikipedia.org/wiki/Atlas_(computer)

http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-mechanisms.pdf