Context Switch

(\*)When µC/OS-III decides to run a different task (see ), it saves the current task’s Scheduling context, which typically consists of the CPU registers, onto the current task’s stack and restores the context of the new task and resumes execution of that task.

1.當kernel要執行high priority task時, kernel會將現在的task context; 包含CPU register; 存放到TCB的stack

(\*)Context switching adds overhead. The more registers a CPU has, the higher the overhead. The time required to perform a context switch is generally determined by how many registers must be saved and restored by the CPU.

1.context會有overhead, 此overhead是因為 cpu register要做copy紀錄

(\*)Task stack: 分為 function stack 和 cpu stack(cpu register stack)

Function stack是tack在做function call時的堆疊

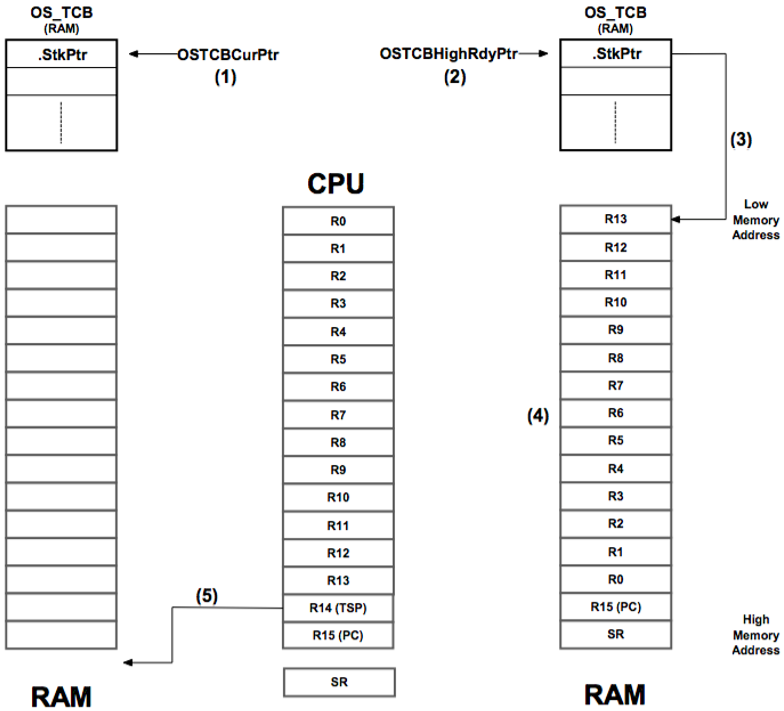
Cpu stack是 task 在做context switch或Interrupt時, CPU的register的儲存空間

(\*)the stack frame for a ready task is always setup to look as if an interrupt has just occurred and all processor registers were saved onto it.

(\*)There are two types of context switches: one performed from a task and another from an ISR

1. Task level 呼叫 OSCtxSw(), ISR context switch呼叫 OSIntCtxSw()

(\*) Variable and data structures prior to calling OSCtxSw(); 執行context switch前的準備動作



1. OSTCBCurPtr指向 OS\_TCB, 當下執行的task

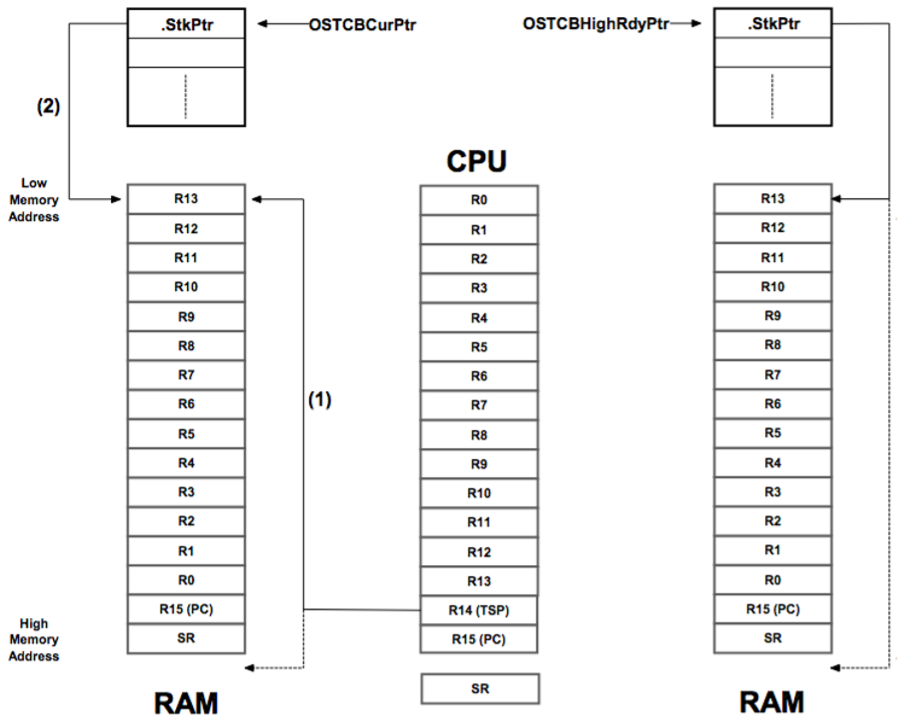
2. OSSched() 發現 OSTCBHighRdyPtr是最高權限的task

3. OSTCBHighRdyPtr->StkPtr指向之前存好的 cpu register stack

4.

5.回到OSTCBCurPtr, 他會儲存之前的function stack pointer (R14); current task的當下stack的位址(R14)紀錄. 此Stack是存放在RAM中. 是curTask的 calling stack frame. 非CPU register stack.

(\*) Operations performed by OSCtxSw(); 開始執行context switch



1. OSCtxSw() 執行, 一開始會儲存status register 和 program counter 到task’s stack(屬於cpu register stack)

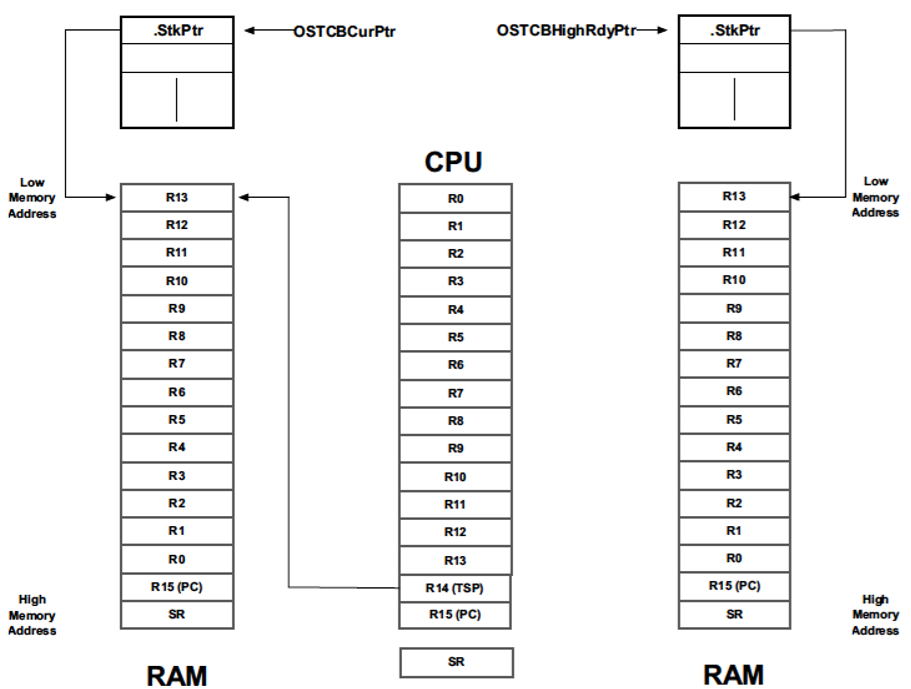
2. OSCtxSw()會儲存CPU的 stack pointer至 OS\_TCB中的 stkPtr; 儲存cpu register中的stack pointer 至RAM中; CPU 的 R14存到 RAM的R13, 當curTask resume的時候將R13在存放回 CPU的R14 , R13的值同時也會存到StkPtr中

3. OSCtxSw() 會將另外一個 task(OSTCBHigh)的StkPtr回存到 cpu register R14; R14 = OSTCBHighRdyPtr->StkPtr

4. OSCtxSw()會將OSTCBHigh的 cpu stack的其他內容回存CPU 的register中

(\*)ISR level scheduler (ISR 造成的context switch), 呼叫OSIntCtSw(), 此函式是在OSIntExit()中被呼叫; OSIntExit()是 IST level的scheduler.

1. 當發生interrupt的時候, 會先將cpu register複製一份到 cpu stack.



2. Load CPU stack pointer, 此stack pointer是新的 ready-to-run的task sp(function stack). 到 cpu register中 R14 = OSTCBHighRdyPtr->StkPtr.

3. 將新的CPU register 填上ready-to-run task的 cpu stack 資訊; OSIntCtxSw()將 task 的 cpu register stack取出 , 填回CPU register

