HW5 REPORT

王柏舜, 0716098

*Abstract*—這個作業希望我們透過從以軟體的修改來達到程式運行效率的提升以及增加counter 來計算context-switch overhead , synchronization overhead，再過程中需要我們閱讀理解freertos 的source code 。

# Thread management

Freertos 的thread 創建是透過xtaskcreate() 這個function 來完成的，並且在thread 創立完成後，需要呼叫vTaskStartScheduler()來選取要執行個task，再我看過source code 後我了解到task 是在freertos 中越來排程的單位，因此在呼叫xTaskCreate後這個fucntion會創立一個新的task並且將他加入ready queue中，這個task 會拿到一個新的 TCB object用來記錄task的名稱、priority、等等的其他細節，而這個task 會需要allocate memory ，這裡分配到的記憶體是由heap中的pvPortMalloc()來分配的，而task 被create 好了之後並不能馬上被使用的，他需要呼叫vTaskStartScheduler 才有辦法選task 來跑，這個function 一開始呼叫的時候，會先加入一個優先度最低的idle task，這個idle task 是為了卻保CPU再任何一個時間至少有一個task可以執行，在這個function的最後會呼叫xPortStartScheduler 來真的拿到真的要執行的task，這會再xPortStartScheduler，中呼叫xPortStartFirstTask() 完成，再理解完task的基本架構後我開始理解task priority 之間的關係，我從taskSELECT\_HIGHEST\_PRIORITY\_TASK 這個fucntion 了解，freertos的ready queue有很多個不同的level，每一個level 是一個獨立的queue ，要選priority最高的task 會從priority最高的那一層queue開始來看，如果那一層是empty的話那麼就會繼續往下一層priority 低一點的queue 來檢查，再來是要決定task切換的部分，再port.c裡面有一個function可以設定timer 發出interrupt的ticks，用來引發trap啟動ISR來執行task的切換，這個trap的過程會再統計overhead的時候才說明，這個部分我想把重點放在vtaskSwitchContext()上，呼叫這個function後會先判斷scheduler是不是suspend的狀態，如果不是的話就會執行task的切換，真的在做context switch的是，當作完前置作業後會呼叫xPortStartScheduler這個function來拿到下一個highest priority task來執行。

# measure context-switching overhead

## 計算規則

計算介於收到timer interrupt 到新的thread開始執行之間的cycle 數。

## 實作

再知道要計算context switch overhead 的時候我一開始想的是因為要context switch overhead 所以應該會有發出software interrupt的訊號出來表示現在跑的軟體有trap要把csr裡面新的task 執行到的program counter load 進來，還有當時的register 的狀態等等的資訊進來，因此我一開始抓了core\_top.v 裡面的irq\_taken\_i到ILA看是不是有這樣的interrupt 發生。以及對應的pc，但是在設定對應的pc當作trigger來看之後發現並沒有這樣的interrupt發生，所以有繼續看source code 看看是不是漏掉了甚麼，後來決定是用vtaskswitchcontext這個function進入點和return 的program counter 來計算，但是在更細部的看之後發現這是有問題的因為這個function並沒有含括最完整的部分，最後再portASM.s 裡面發現freertos\_risc\_v\_trap\_handler 、processed\_source 這兩個fucntion ，前者的功能是收到對應的interrupt後執行對應的interrupt service routine ，而後者是將新的task 要得cpu 狀態都設置好，因此最後我計算的cycle 數是從freertos\_risc\_v\_trap\_handler 的進入點，到processed\_source 的return 的program counter這個位置。

# measure sychronization overhead

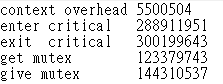
## 計算規則

這個部分總共有四個部分要算第一個部分是要計算進入critical section 需要幾個cycle，而第二個部分是要計算離開cirtical section 需要幾個cycle ，第三個部分是要計算拿到mutex 需要幾個cycle，第四個部分是要計算give mutex 需要幾個cycle。

## 實作

1. 計算進入critical section:在freertos 的實作中enter critical section 是利用 interrupt disabling 來完成的，再source code 的define中taskENTER\_CRITICAL 是由vtaskEnterCritical 這個function來完成，再這個function裡面會呼叫portDISABLE\_INTERRUPT這個function ，這個function 才是真正關掉interrupt的地方。在計算cycle 數的部分我選擇 vtaskEnterCritical 的進入點到 該function return 的program counter。
2. 計算離開 critical section: freertos 在這部分的實作方式和進入critical section 相同，由define 的關係可以知道最終interrupt enable 的地方在vtaskExitCritical。計算cycle 的部分是從 vtaskExitCritical 的進入點，到return 的 program counter。
3. 計算拿到mutex：拿到mutex 的實作是在xQueueSemaphoreTake 這個部分實作的，freertos 對於 semaphore 的實作是用 queue 來實作的在 xQueueSemaphoreTake 裡面會是都過 queue 的狀態以及 xTaskCheckTimeOut 的結果來決定是否能拿到mutex如果不能那就要被 block住。計算cycle 的部分我是從 xQueueSemaphoreTake 的進入點開始到其return 的program counter 間的 cycle 數。
4. 計算解開 mutex ：解開 mutex 的實作是在xQueueGenericSend 這個部分實作的，一樣是利用queue的狀態來決定如何操作，計算cycle 的部分我是從 xQueueGenericSend 的進入點到其 return 的 program counter 之間的cycle。

## 統計結果

統計結果的呈現我是利用ILA 以 program counter 的值當作trigger 來獲得，trigger 的條件為當 task 1 被 delete時，當時的統計數據，呈現的數值為整個程式跑完 overhead 的總額，結果為下圖。

## ISSUE

在計算 mutex 的部分因為在take , give mutex 的function 中途會把 interrupt enable 之後再把 interrupt disable 來讓其他的task 可以 give 或 taek semaphore 但這使的這裡的計算結果有可能不是最純正的mutex overhead 。

##### time quantum

我也有嘗試調整time quantum的大小來看task 間如何搶mutex 我是了10ms,5ms,1ms ，結果發現task 1 counter 的大小依序為1ms>10ms>5ms，而overhead 的結果沒有甚麼改變，這和我原本預期的結果不太一樣我以為time quantum 愈小task 之間的資源分配會稍微平均一點，但就看到的結果好像不是如此。The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g.” Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### optimization

這個部分我沒有成功做出來，我對於效能的評估是利用overhead 來當作標準，因為老師提供的source code 版本是general purpose 的，在trace code 的過程中，我有發現一些comment 的內容，我有看到像是在freertos config.h裡面加把原本 define 的值改成別的這樣就可以使用 port optimize 的funtion 版本，因為我有看到裡面有提到port optimize 是支援特定的 micro artectture 我有試圖去更改我的config 但是對於overhead 都沒有甚麼影響。