第12次實習課

學生：林培瑋

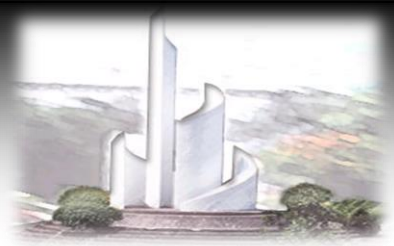
2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

HW2

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)



微處理機概論 – HW2

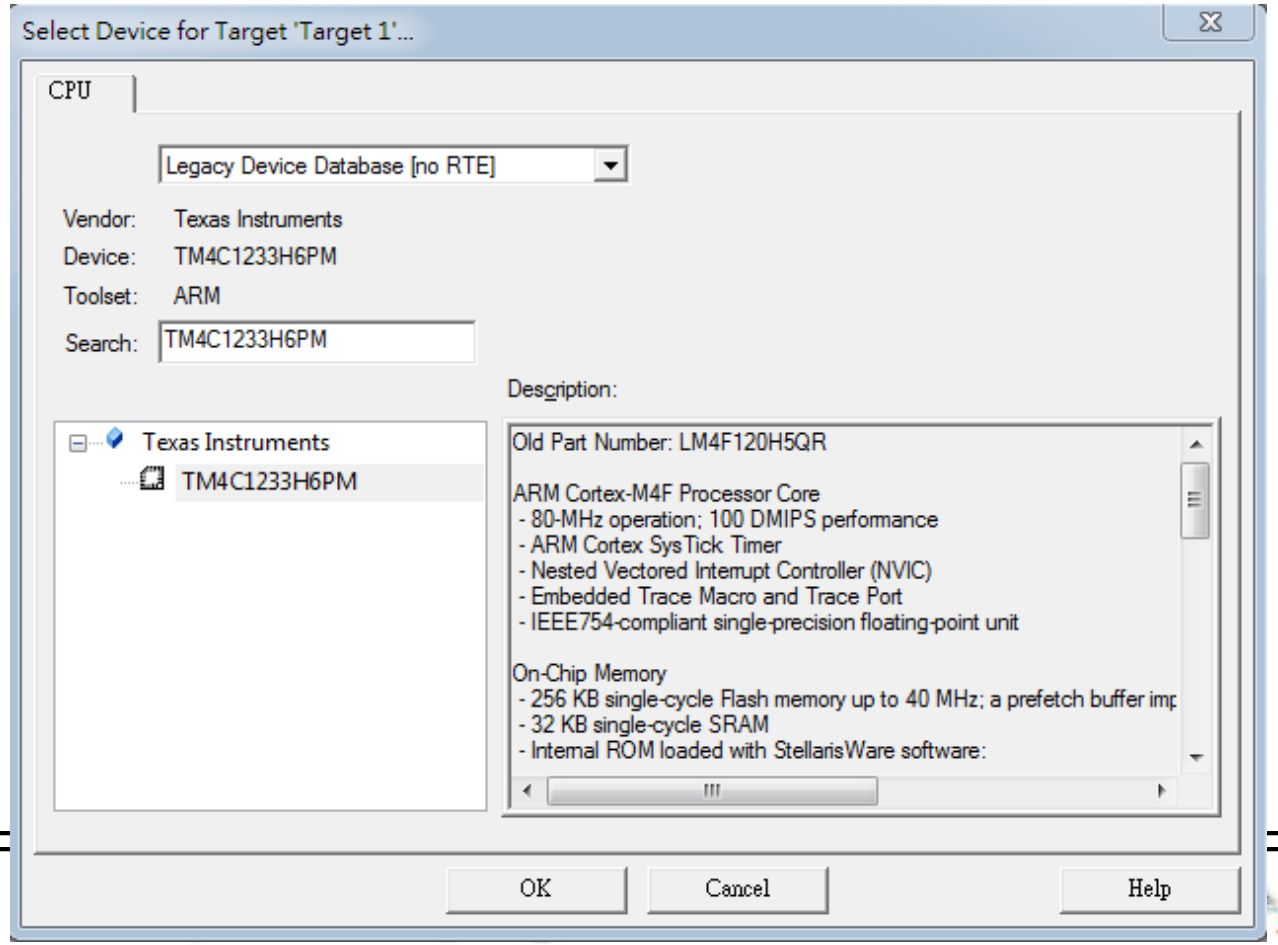
繳交期限：5/28 23:59

(用word檔上傳至iclass)

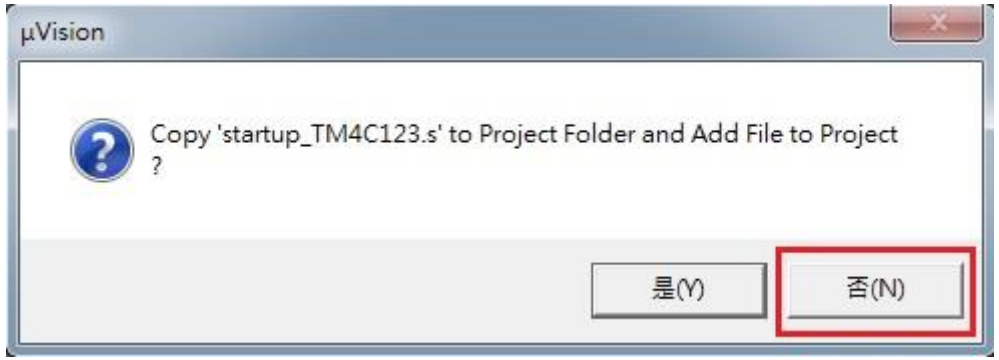
請提前上傳，以免iclass塞車

第一部分-作業題目

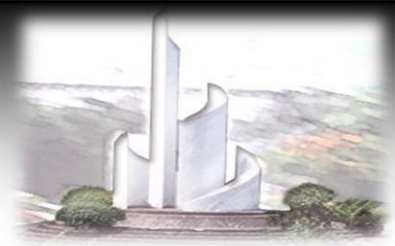
- 課本p.327-332 ex.15-1~15-3
- TM4C1233H6PM



- 不用copy ”.s” 檔，自己新增一個，把程式寫好。



EX.15-1 (1/4)



```
 1
 2 Stack        EQU     0x00000100
 3 DivbyZ       EQU     0xD14
 4 SYSHNDCTRL   EQU     0xD24
 5 Usagefault   EQU     0xD2A
 6 NVICBase     EQU     0xE000E000
 7
 8         AREA    STACK, NOINIT, READWRITE, ALIGN=3
 9 StackMem
10         SPACE   Stack
11         PRESERVE8
12
13         AREA    RESET, CODE, READONLY
14         THUMB
15
16 ; The vector table sits here
17 ; We'll define just a few of them and leave the rest at 0 for now
18
19         DCD     StackMem + Stack        ; Top of Stack
20         DCD     Reset_Handler           ; Reset Handler
21         DCD     NmiISR                  ; NMI Handler
22         DCD     FaultISR                ; Hard Fault Handler
23         DCD     IntDefaultHandler       ; MPU Fault Handler
24         DCD     IntDefaultHandler       ; Bus Fault Handler
25         DCD     IntDefaultHandler       ; Usage Fault Handler
26
27         EXPORT  Reset_Handler
28         ENTRY
29
```

PANEL
Parallel Architectures & NEtworks Lab.

TAMKANG UNIVERSITY

EX.15-1 (2/4)

```
29
30 Reset_Handler
31                 ; enable the divide-by-zero trap
32                 ; located in the NVIC
33                 ; base: 0xE000E000
34                 ; offset: 0xD14
35                 ; bit: 4
36                 LDR     r6, =NVICBase
37                 LDR     r7, =DivbyZ
38                 LDR     r1, [r6, r7]
39                 ORR     r1, #0x10       ; enable bit 4
40                 STR     r1, [r6, r7]
41
42                 ; now turn on the usage fault exception
43                 LDR     r7, =SYSHNDCTRL
44                 LDR     r1, [r6, r7]
45                 ORR     r1, #0x40000
46                 STR     r1, [r6, r7]
47
48                 ; try out a divide by 2 then a divide by 0!
49                 MOV     r0, #0
50                 MOV     r1, #0x11111111
51                 MOV     r2, #0x22222222
52                 MOV     r3, #0x33333333
53
54                 ; this divide works just fine
55                 UDIV    r4, r2, r1
56                 ; this divide takes an exception
57                 UDIV    r5, r3, r0
58
59 Exit           B       Exit
```

EX.15-1(3/4)

```
60
61 NmiISR          B        NmiISR
62 FaultISR        B        FaultISR
63 IntDefaultHandler
64
65                 ; let's read the Usage Fault Status Register
66
67                 LDR      r7, =Usagefault
68                 LDRH     r1, [r6, r7]
69                 TEQ      r1, #0x200
70                 IT       EQ
71                 LDREQ    r9, =0xDEADDEAD
72                 ; r1 should have bit 9 set indicating
73                 ; a divide-by-zero has taken place
74 done             B        done
75                 ALIGN
76
77                 END
78
```

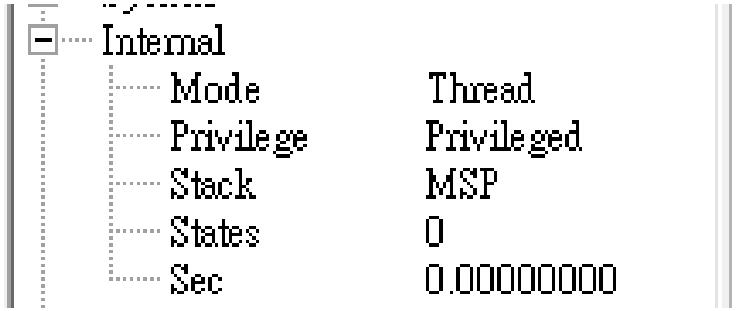
EX.15-1 (4/4)

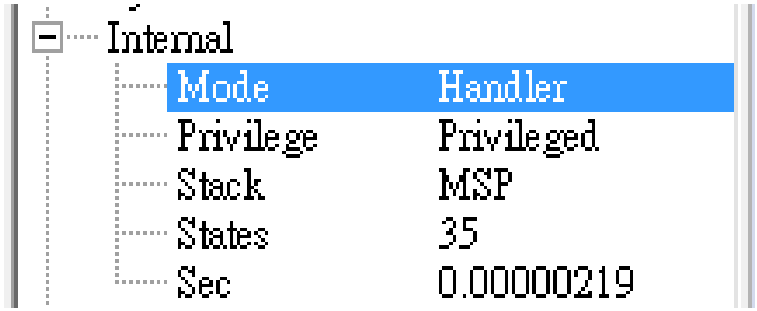
Register	Value
Core	
R0	0x00000000
R1	0x00000200
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D2A
R8	0x00000000
R9	0xDEADDEAD
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000066
xPSR	0x41000006

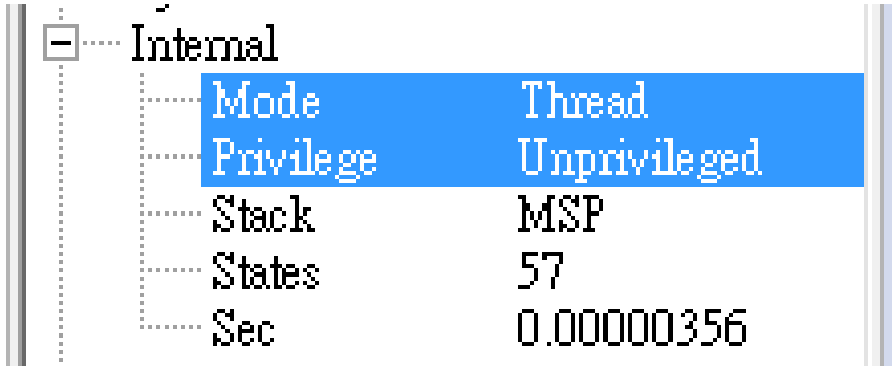
EX.15-2 (1/2)

```
62 IntDefaultHandler
63
64                  ; let's read the Usage Fault Status Register
65
66                  LDR     r7, =Usagefault
67                  LDRH    r1, [r6, r7]
68                  TEQ     r1, #0x200
69                  IT      EQ
70                  LDREQ   r9, =0xDEADDEAD
71                  ; r1 should have bit 9 set indicating
72                  ; a divide-by-zero has taken place
73
74                  ; switch to user Thread mode
75                  MRS     r8, CONTROL
76                  ORR     r8, r8, #1
77                  MSR     CONTROL, r8
78                  BX      LR
79
80                  ALIGN
81
82                  END
83
```

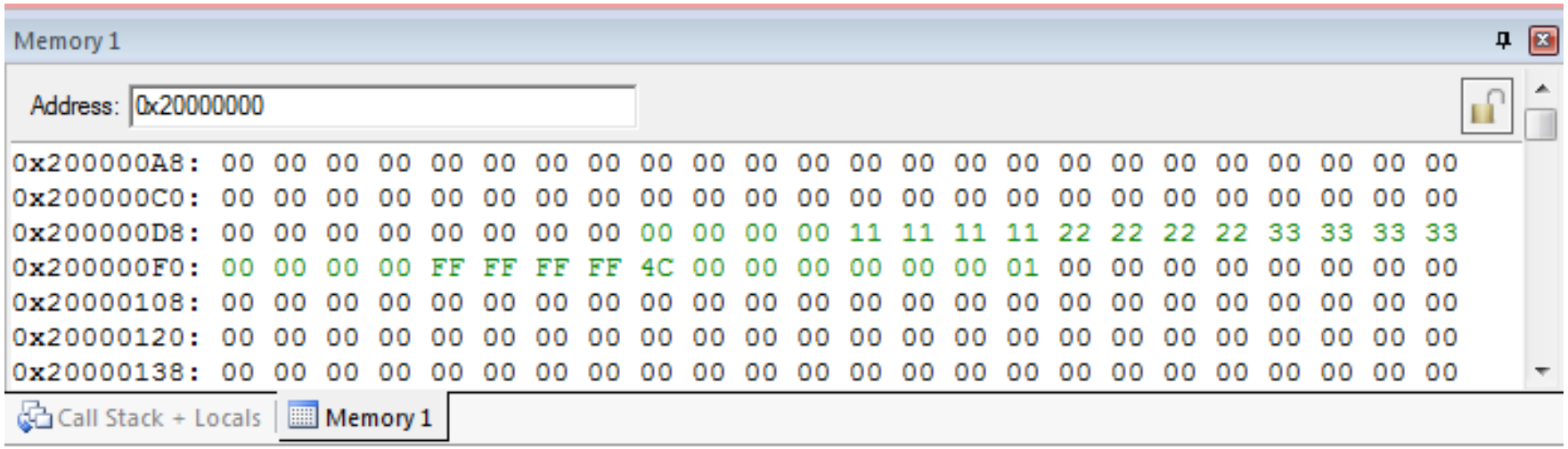
EX.15-2 (2/2)


Internal
Mode Thread
Privilege Privileged
Stack MSP
States 0
Sec 0.00000000

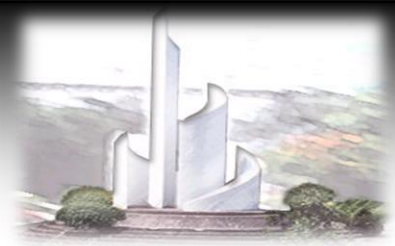

Internal
Mode Handler
Privilege Privileged
Stack MSP
States 35
Sec 0.00000219


Internal
Mode Thread
Privilege Unprivileged
Stack MSP
States 57
Sec 0.00000356

EX.15-3


Memory 1
Address: 0x20000000
0x200000A8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000D8: 00 00 00 00 00 00 00 00 00 00 00 00 11 11 11 11 22 22 22 22 33 33 33 33
0x200000F0: 00 00 00 00 FF FF FF FF 4C 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00
0x20000108: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000138: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals
Memory 1

第一部分



- 執行課本EX.15-1~15-3程式。
- 截圖請截完整的畫面圖。
- EX.15-1要有暫存器完整的結果。

  (如課堂上所提示的，程式第69行之TEQ

  需自行調整成使用TST來達成才正確)
- EX.15-2要能看出mode的變化。
- EX.15-3要能看得到stacking與unstacking效果(含顯示register window中相關暫存器值與memory window中堆疊值的對應)。

第二部分

(1) 加入以下兩字串宣稱

divide_by_0 DCB “DIVIDE-BY-ZERO Event”, 0

not_divide_by_0 DCB “No DIVIDE-BY-ZERO Event”, 0

查看usage fault status register後，如發生除以零錯誤，則將字串divide_by_0寫入從位址0x20000120開始的記憶體空間，反之，則將字串not_divide_by_0寫入從位址0x20000130開始的記憶體空間。（務必說明exception entry三步驟及exception exit兩步驟。）

第二部分

(2) 展示模式轉換

(a) from privileged thread mode to unprivileged thread mode

(b) from privileged thread mode to privileged handler mode

(c) from privileged handler mode to privileged thread mode

(d) from privileged handler mode to unprivileged thread mode

第二部分

(3) 將usage fault改成hard fault。

- Hard fault更改條件及說明可查看課本P.335的第3行~第9行，上面寫出有兩種方法能產生Hard fault。

第二部分

- 程式碼一定要有一個判斷是否為Hard fault的基準。例:在15.1程式碼中FaultISR為Hard Fault Handler (圖1)，因此需要能產生Hard Fault，r10才會改變(圖2)。

```
FaultISR

        LDR     R10,=0x77777777
```

ore

R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000000
R5	0x00000000
R6	0xE000E000
R7	0x00000D24
R8	0x00000000
R9	0x00000000
R10	0x77777777
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000C0
R14 (LR)	0xFFFFFFF1
R15 (PC)	0x0000004E
xPSR	0x01000003

第二部分

- 需將兩種方法的結果都呈現出來，所以題(3)會有兩組呈現結果的截圖與兩份程式碼，**且需能執行完FaultISR後回到主程式**。

第三部分

- **寫一個程式含三個副程式分別達成以下三個目的並於主程式分別呼叫此三個副程式來達成測試：**

（副程式結果可置於自選的暫存器中，但需在截圖中框出輸出之暫存器）

- 1. 設定priority group number (用empty ascending stack暫存恢復)

(假設AIRCR暫存器位址為 0x400000C8)

(呼叫前欲設定的priority group number需先存入自選的暫存器中）

- 2. 讀取priority group number (用full ascending stack暫存恢復)

(假設AIRCR暫存器位址為 0x400000C8)

- 3. 計算某一IRQ的pre-emption priority及sub priority

(用empty descending stack暫存恢復)

(呼叫前假設的width of the interrupt priority register 、priority group number 、與該IRQ的interrupt priority register內容值需先存入自選的暫存器中）

第四部分

- **寫一個程式含四個副程式(**皆用full descending stack暫存恢復)**分別用下列四種方式獲取**width of an interrupt priority(請假設寬度為6)，**並於主程式分別呼叫此四個副程式來達成測試：**

  1. using LSR with test pattern LSR
  2. using LSL with test pattern LSL
  3. using LSR without test pattern LSR
  4. using LSL without test pattern LSL

  (可用位址為 0x4000008C 的interrupt priority register來獲取，副程式結果可置於自選的暫存器中，但需在截圖中框出輸出之暫存器)

- 繳交內容：每部分按照結報格式編寫完整（含Keil Tool 視窗中程式與執行結果—含暫存器、記憶體、堆疊及題目要求之變化等等相關截圖，每個截圖務必註解學號與英文姓名，否則不計分），把結報word檔上傳iclass對應作業位置。
- 第三、四部分晶片皆使用**LPC2104**執行即可
- 繳交期限： 5/28 23:59
- Word檔名：

  微處理機概論_學號_姓名_HW2

Priority

2024 Advanced Mixed-Operation System (AMOS) Lab.

Tamkang University
Department of Electrical and Computer Engineering
No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

Priority value of an interrupt

Priority value of an IRQ(Interrupt Request) is stored in a corresponding 8-bit Interrupt Priority Register (Priority Level Register)in NVIC.

00000000~11111111　　256 interrupt priority values(programmable) ➔ programmable priority values >= 0

<=16 interrupts	4 bits are enough
16 priority values	4 bits of Interrupt Priotity Register 0000~1111 (4 implemented bits, 4 not implemented bits-always 0)
<=8 interrupts	3 bits are enough
8 priority values	3 bits of Interrupt Priotity Register 000~111 (3 implemented bits, 5 not implemented bits-always 0)

MSB(Most Significant Bit) first implemented
LSB(Least Significant Bit) first implemented

Use MSB or LSB first?

LSB ➔ 00000000~00000111
MSB ➔ 00000000~11100000

如果從16 ➔ 8，
考慮Modularity(有彈性)、Reconfiguration(可再造的)，
決定哪一個是最佳的。

- **LSB**：
  - 16 priority values ➔ 4 bits
  - IRQ1 00001011 (value 11, **lower** priority level)
  - IRQ2 00000111 (value 7, **higher** priority level)

  ➔reconfigure to 8 priority values only(Reconfiguration, software portability) ➔ 3 bits
  - IRQ1 00000011 (value 3, **higher** priority level) ➔ **priority inversion**
  - IRQ2 00000111 (value 7, **lower** priority level)
- **MSB**：
  - 16 priority values ➔ 4 bits
  - IRQ1 10110000 (value 0xB0, **lower** priority level)
  - IRQ2 01110000 (value 0x70, **higher** priority level)

  ➔reconfigure to 8 priority values only ➔ 3 bits
  - IRQ1 10100000 (value 0xA0, **lower** priority level) ➔ **No priority inversion**
  - IRQ2 01100000 (value 0x60, **higher** priority level)

Interrupt Priority Registers-Cortex-M3

- https://developer.arm.com/documentation/dui0552/a/cortex-m3-peripherals/nested-vectored-interrupt-controller/interrupt-priority-registers

Interrupt Priority Registers

The NVIC_IPR0-NVIC_IPR59 registers provide an 8-bit priority field for each interrupt and each register holds four priority fields. These registers are byte-accessible. See the register summary in Table 4.2 for their attributes. Each register holds four priority fields as shown:

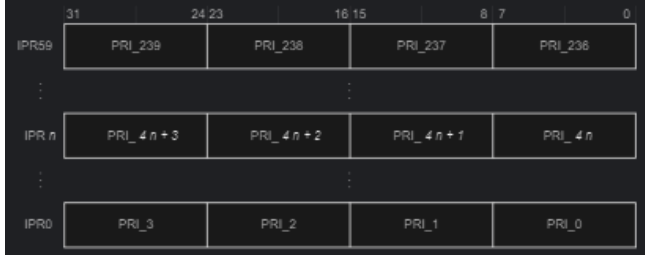


Table 4.9. IPR bit assignments

Bits	Name	Function
[31:24]	Priority, byte offset 3	Each implementation-defined priority field can hold a priority value, 0-255. The lower the value, the greater the priority of the corresponding interrupt. Register priority value fields are 8 bits wide, and un-implemented low-order bits read as zero and ignore writes.
[23:16]	Priority, byte offset 2	
[15:8]	Priority, byte offset 1	
[7:0]	Priority, byte offset 0	

Priority Grouping

1. Who will be served first
2. Who will be interrupted

➔為了節省查考時間，只使用其中的一些bits。

- 8-bit Interrupt Priority Register
  - group (**preemption**) priority field (upper field)
    - Interrupt with **higher/lower or the same** group priority **can/cannot preempt** a **lower/higher or the same** priority interrupt
    - Interrupt with **higher** group priority can be served first
  - Subpriority (within the group) field (lower field)
    - To decide who will be served first when multiple pending interrupts **have the same group priority** (the multiple pending interrupts are in the same group)
  - Hardware priority
    - To decide who will be served first when multiple pending interrupts **have the same group priority and the same subpriority**
    - The interrupt with **the lowest IRQ number** is processed first ➔連接NVIC腳位的IRQ編號
  - https://developer.arm.com/documentation/dui0552/a/the-cortex-m3-processor/exception-model/interrupt-priority-grouping

Application Interrupt and Reset Control Register

- https://developer.arm.com/documentation/dui0552/a/cortex-m3-peripherals/system-control-block/application-interrupt-and-reset-control-register?lang=en

[10:8]	PRIGROUP	R/W	Interrupt priority grouping field is implementation defined. This field determines the split of group priority from subpriority, see *Binary point*.

Binary point

The PRIGROUP field indicates the position of the binary point that splits the PRI_*n* fields in the Interrupt Priority Registers into separate *group priority* and *subpriority* fields. Table 4.18 shows how the PRIGROUP value controls this split. Implementations having fewer than 8-bits of interrupt priority treat the least significant bits as zero.

Table 4.18. Priority grouping

x：Group priority；y：Subpriority

	Interrupt priority level value, PRI_*N*[7:0]			Number of	
PRIGROUP	Binary point [a]	Group priority bits	Subpriority bits	Group priorities	Subpriorities
0b000	bxxxxxxx.y	[7:1]	[0]	128	2
0b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
0b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
0b011	bxxxx.yyyy	[7:4]	[3:0]	16	16
0b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
0b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
0b110	bx.yyyyyyy	[7]	[6:0]	2	128
0b111	b.yyyyyyyy	None	[7:0]	1	256

[a] PRI_*n*[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.

分割點介於
Group number
~ Group number+1
之間

Q&A

Thanks for your attention !!