

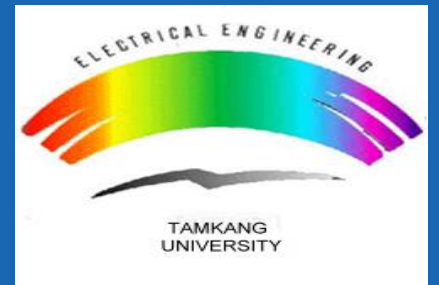
第11次實習課

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.)

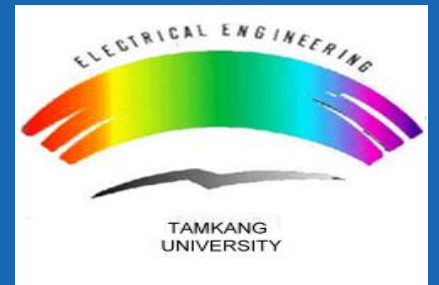


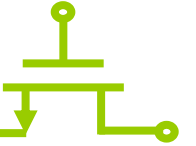
Hard fault

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.)





A hard fault can occur when the processor sees an error during exception processing, or when another fault such as a **usage fault is disabled**.¹ In our example code, if we disable usage faults and then rerun the code, you will notice that the processor takes a hard fault when the UDIV instruction is attempted, rather than a usage fault. You can also see hard faults when there is an **attempt to access the System Control Space in an unprivileged mode**;² for example, if you attempt to write a value to one of the NVIC registers in Thread mode, the processor will take an exception.

TABLE 15.1

Exception Types and Vector Table

Exception Type	Exception Number	Priority	Vector Address	Caused by...
—	—	—	0x00000000	Top of stack
Reset	1	– 3 (highest)	0x00000004	Reset
NMI	2	– 2	0x00000008	Non-maskable interrupt
Hard fault	3	– 1	0x0000000C	All fault conditions if the corresponding fault is not enabled
Mem mgmt fault	4	Programmable	0x00000010	MPU violation or attempted access to illegal locations
Bus fault	5	Programmable	0x00000014	Bus error, which occurs during AHB transactions when fetching instructions or data
Usage fault	6	Programmable	0x00000018	Undefined instructions, invalid state on instruction execution, and errors on exception return
—	7–10	—	—	Reserved
SVcall	11	Programmable	0x0000002C	Supervisor Call
Debug monitor	12	Programmable	0x00000030	Debug monitor requests such as watchpoints or breakpoints
—	13	—	—	Reserved
PendSV	14	Programmable	0x00000038	Pendable Service Call
SysTick	15	Programmable	0x0000003C	System Tick Timer
Interrupts	16 and above	Programmable	0x00000040 and above	Interrupts

1. Usage fault is disabled (Keil Tool)



Registers

Register	Value
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x0000D14
R8	0x00000000
R9	0x00000000
R10	0xFACEBEEF
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000100
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000040
xPSR	0x01000000

Banked

System

Internal

Mode	Thread
Privileged	Privileged
Stack	MSP
States	42
Sec	0.00000262

FPU

EXAMPLE15.1.s

```

27 Reset_Handler
28 ; enable the divide-by-zero trap
29 ; located in the NVIC
30 ; base: 0xE000E000
31 ; offset: 0xD14
32 ; bit: 4
33 ; ADDR r11, StackMem
34
35 ;MRS      r8, CONTROL
36 ;ORR      r8, #1
37 ;MSR      CONTROL, r8
38
39
40 LDR      r6, =NVICBase
41 LDR      r7, =Divby2
42 ; read(access) an NVIC register
43 ORR      r1, [r6, r7]
44 ; enable bit 4
45 STR      r1, [r6, r7]
46 ; write(access) an NVIC register
47
48
49 ; now turn on the usage fault exception
50 ;LDR      r7, =SYSHNDCTRL ; p. 163
51 ;LDR      r1, [r6, r7]
52 ;ORR      r1, #0x40000
53 ;STR      r1, [r6, r7]
54
55 ; try out a divide by 2 then a divide by 0!
56 MOV      r0, #0
57 MOV      r1, #0x11111111
58 MOV      r2, #0x22222222
59 MOV      r3, #0x33333333
60
61 ; this divide works just fine
62 UDIV     r4, r2, r1
63 ; this divide takes an exception
64 UDIV     r5, r3, r0
65 ; Exception Entry
66 ; 1. Stacking
67 ; 2. Interrupt Vector Lookup
68 ; 3. LR
69
70 Exit
71 B        Exit
72
73 NmiISR
74 B        NmiISR
75
76 FaultISR
77 LDR      r1, =0xFACEBEEF
78 BK      LR
79 B        FaultISR
80
81 IntDefaultHandler
            
```

disable usage fault

Vector Table

```

; The vector table sits here
; We 1 define just a few of them and leave the rest at 0 for now
*
DCD      StackMem + Stack ; Top of Stack
DCD      Reset_Handler    ; Reset Handler
DCD      NmiISR            ; NMI Handler
DCD      FaultISR          ; Hard Fault Handler
DCD      IntDefaultHandler ; MPU Fault Handler
DCD      IntDefaultHandler ; Bus Fault Handler
DCD      IntDefaultHandler ; Usage Fault Handler
EXPORT   Reset_Handler
            
```

Command

Running with Code Size Limit: 32K

Load "F:\03.淡江碩士\01.碩一(112)\02.碩一下學期\08.微處理機概論(電通)(助教課)\01.實習課\第11次實習課\EXAMPLE15

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter

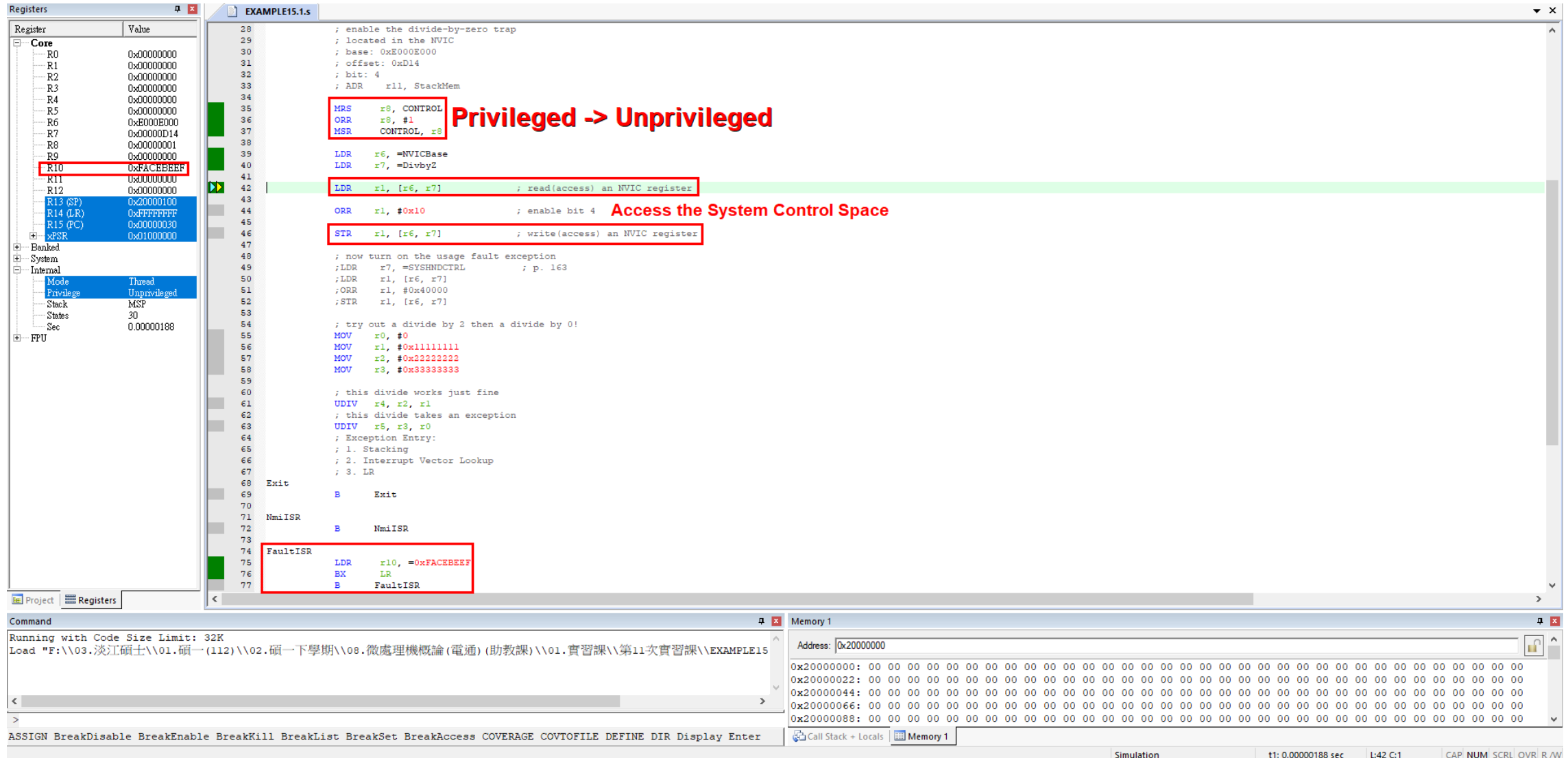
Memory 1

Address: 0x20000000

0x20000000:	00 00
0x20000022:	00 00
0x20000044:	00 00
0x20000066:	00 00
0x20000088:	00 00

Call Stack + Locals Memory 1

Simulation t1: 0.00000262 sec L60 C:1 CAP. NUM SCRL OVR. R/W

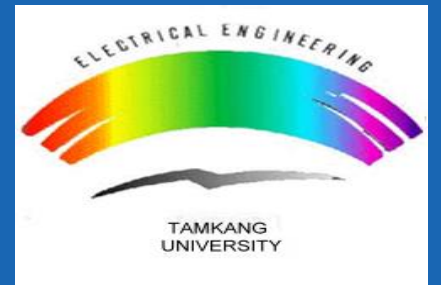


Priority

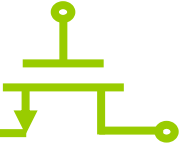
2024 Advanced Mixed-Operation System (AMOS) Lab.



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



p.330 TABLE 15.1 Priority



FIFO queue(一般型, 先進先出)

CPU ← NVIC peripheral chips

TABLE 15.1 若使用FIFO queue, 重開機則會等待一段時間才去做。

Exception Types and Vector Table

Exception Type	Exception Number	Priority	Vector Address	Caused by...
—	—	—	0x00000000	Top of stack
Reset	1	-3 (highest)	0x00000004	Reset
NMI	2	-2	0x00000008	Non-maskable interrupt 不可忽略
Hard fault	3	-1	0x0000000C	All fault conditions if the corresponding fault is not enabled
Mem mgmt fault	4	Programmable	0x00000010	MPU violation or attempted access to illegal locations
Bus fault	5	Programmable	0x00000014	Bus error, which occurs during AHB transactions when fetching instructions or data EX: install driver
Usage fault	6	Programmable	0x00000018	Undefined instructions, invalid state on instruction execution, and errors on exception return get an interrupt number (exception number)
—	7–10	—	—	Reserved install program in main memory vector
SVcall	11	Programmable	0x0000002C	Supervisor Call
Debug monitor	12	Programmable	0x00000030	Debug monitor requests such as watchpoints or breakpoints
—	13	—	—	Reserved
PendSV	14	Programmable	0x00000038	Pendable Service Call
SysTick	15	Programmable	0x0000003C	System Tick Timer
Interrupts	16 and above (16~255, 240 interrupts)	Programmable	0x00000040 and above	Interrupts

priority queue 1. who will be served first
2. who will be interrupted (preempted 搶佔)

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

00000000~11111111

256 interrupt priority values

16 priority values

4 bits of Interrupt Priority Register 0000~1111 (4 implemented bits, 4 not implemented bits-always 0)

8 priority values

3 bits of Interrupt Priority Register 000~111 (3 implemented bits, 5 not implemented bits-always 0)

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

MSB first vs. 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。

p.330 TABLE 15.1 Priority



❖ LSB :

- 16 priority values → 4 bits
- 0000**1011** (value 11, **lower** level)
- 0000**0111** (value 7, **higher** level)
- reconfigure to 8 priority values only → 3 bits
- 00000**011** (value 2, **higher** level) → **priority inversion**
- 00000**111** (value 7, **lower** level)

❖ MSB :

- 16 priority values → 4 bits
- **1011**0000 (value 0xB0, **lower** level)
- **0111**0000 (value 0x70, **higher** level)
- reconfigure to 8 priority values only → 3 bits
- **101**00000 (value 0xA0, **lower** level) → **No priority inversion**
- **011**00000 (value 0x60, **higher** level)

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

Q&A

Thanks for your attention !!