

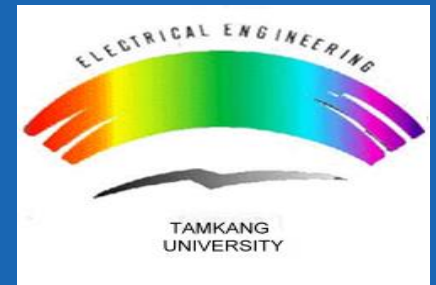
第10次實習課

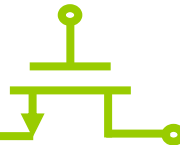
學生：林培瑋

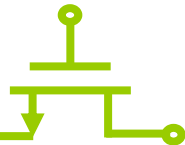
2024 Advanced Mixed-Operation System (AMOS) Lab.



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







執行完「UDIV r4, r2, r1」，PC = 0x4C
(PC為下一行程式碼所在的記憶體位置)

Instruction Cycle :

1. Fetch according to PC
2. update PC
3. Decode
4. Execute

Exception Entry : 2. Interrupt Vector Table Lookup

Registers

Register	Value
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D24
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000056
xPSR	0x01000006

Disassembly

```

0x00000054 E7FE B 0x00000054
68: LDR r7, =Usagefault
0x00000056 F640572A MOVW r7, #0xD2A
69: LDRH r1, [r6, r7]
0x0000005A 5BF1 LDRH r1, [r6, r7]
70: TST r1, #0x200

```

EXAMPLE15.1.s

```

47: MOV r2, #0x22222222
48: MOV r3, #0x33333333
49: ; this divide works just fine
50: UDIV r4, r2, r1
51: ; this divide takes an exception
52: UDIV r5, r3, r0
53: ; Exception Entry:
54: ; 1. Stacking
55: ; 2. Interrupt Vector Lookup
56: ; 3. LR
57: Exit
58: B Exit
59: Privileged
60: NmiISR
61: B NmiISR
62: FaultISR
63: B FaultISR
64: IntDefaultHandler
65: ; let read the Usage Fault Status Register
66: LDR r7, =Usagefault
67: LDRH r1, [r6, r7]
68: TST r1, #0x200
69: IT NE
70: ; IT: If NE Then
71: ; ITT
72: ; ITTE
73: ; ITTEE: If Then 2 Else 2
74: LDRNE r9, =0xDEADDEAD
75: MOVNE r10, r9
76: ; MOVNE r10, r9
77: LDRNE r11, =FACEBEEF
78: ; MOVNE r11, =FACEBEEF
79: MOVEQ r12, r11
80: ; r1 should have bit 9 set indicating
81:

```

TABLE 15.1
Exception Types and Vector Table

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

TABLE 15.4
Usage Fault Status Register (Offset 0xD2A)

Bit	Name	Reset Value	Description
9	DIVBYZERO	0	Indicates a divide by zero has occurred (only if DIV_0_TRP is also set)
8	UNALIGNED	0	An unaligned access fault has occurred
7:4	—	—	—
3	NOCP	0	Indicates a coprocessor instruction was attempted
2	INVPC	0	An invalid EXC_RETURN value was used in an exception
1	INVSTATE	0	An attempt was made to switch to an invalid state
0	UNDEFINSTR	0	Processor tried to execute an undefined instruction

若沒有設定 DIV_0_TRP，則不會進入中斷處理程式，且 Usage Fault Status Register 狀態不會改變。

執行完「UDIV r5, r3, r0」，PC = 0x56 (若未發生中斷，PC = 0x50)，因為發生中斷，系統由 Vector Number 計算出 $6 \times 4 = 24$ ，找到 Vector 的值為 0x56。

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08次實習課\EXAMPLE15

```

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

Memory 1

Address: 0x20000000

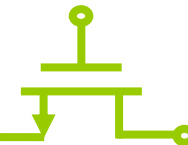
```

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

```

Simulation t1: 0.00000219 sec L:68 C:1 CAP NUM SCRL OVR R/W

Exception Entry : 3. LR = EXC_RETURN(1/2)



15.4 STACK POINTERS

There are two stack pointers available to programmers, the **Main Stack Pointer (MSP)** and the **Process Stack Pointer (PSP)**, both of which are called register r13; the choice of pointer depends on the mode of the processor and the value of CONTROL[1]. If you happen to have an operating system running, then the kernel should use the MSP. Exception handlers and any code requiring privileged access *must* use the MSP. Application code that runs in Thread mode should use the PSP and create a process stack, preventing any corruption of the system stack used by the operating system. Simpler systems, however, such as those without any operating system may choose to use the MSP alone, as we'll see in the examples in this chapter. The topic of the inner working of operating systems literally fills textbooks, but a good working knowledge of the subject can be gleaned from (Doepner 2011).

EXC_RETURN[31:0]	State	Return to	Using Stack Pointer
0xFFFFFE1	Floating-point	Handler mode	MSP
0xFFFFFE9	Floating-point	Thread mode	MSP
0xFFFFFED	Floating-point	Thread mode	PSP
0xFFFFF1	Non-floating-point	Handler mode	MSP
0xFFFFF9	Non-floating-point	Thread mode	MSP
0xFFFFFD	Non-floating-point	Thread mode	PSP



Nested Interrupts

Nested Vectored Interrupt Controller(NVIC)

1. Stacking

2. Interrupt Vector Table Lookup

While the processor is storing critical information on the stack, it also reads the address of the exception handler in the vector table. In our previous example, the processor is about to take a usage fault exception, so the address found at memory location 0x00000018 would be used. The processor will also store one more value for us, called **EXC_RETURN**, in the Link Register. This 32-bit value describes which stack to use upon exception return, as well as the mode from which the processor left before the exception occurred. Table 15.2 shows all the values currently used on the Cortex-M4—most are reserved. Notice also from our previous example that the EXC_RETURN value was 0xFFFFF9, since the floating-point unit was not enabled at the time we took the exception, and we wish to return to Thread mode.

PC ←

6x4=24

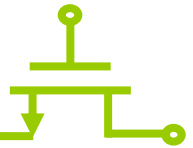
3. LR = EXC_RETURN ←

TABLE 15.2 註記要回到哪一種狀態

EXC_RETURN Value for the Cortex-M4 with Floating-Point Hardware

EXC_RETURN[31:0]	State	Return to	Using Stack Pointer
0xFFFFFE1	Floating-point	Handler mode	MSP
0xFFFFFE9	Floating-point	Thread mode	MSP
0xFFFFFED	Floating-point	Thread mode	PSP
0xFFFFF1	Non-floating-point	Handler mode	MSP
0xFFFFF9	Non-floating-point	Thread mode	MSP
0xFFFFFD	Non-floating-point	Thread mode	PSP

Exception Entry : 3. LR = EXC_RETURN(2/2)



Registers

Register	Value
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xB000E000
R7	0x0000D24
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000056
xPSR	0x01000006

Core

Banked

System

Internal

FPU

Mode

Privilege

Stack

States

Sec

Handler

Privileged

MSP

35

0.00000219

EXAMPLE15.1.s

```

34      LDR    r6, =NVICBase
35      LDR    r7, =Divby2
36      LDR    r1, [r6, r7]
37      ORR    r1, #0x10          ; enable bit 4
38      STR    r1, [r6, r7]
39      ; now turn on the usage fault exception
40      LDR    r7, =SYSHNDCTRL    ; p. 163
41      LDR    r1, [r6, r7]
42      ORR    r1, #0x40000
43      STR    r1, [r6, r7]
44      ; try out a divide by 2 then a divide by 0!
45      MOV    r0, #0
46      MOV    r1, #0x11111111
47      MOV    r2, #0x22222222
48      MOV    r3, #0x33333333
49      ; this divide works just fine
50      UDIV   r4, r2, r1
51      ; this divide takes an exception
52      UDIV   r5, r3, r0
53      ; Exception Entry:
54      ; 1. Stacking
55      ; 2. Interrupt Vector Lookup
56      ; 3. LR
57      Exit
58      B      Exit
59
60      NmiISR
61      B      NmiISR
62
63      FaultISR
64      B      FaultISR
65
66      IntDefaultHandler
67      ; let read the Usage Fault Status Register
68      LDR    r7, =Usagefault
69      LDRH   r1, [r6, r7]
70      TST    r1, #0x200
71      IT     NE
72      ; IT:If NE Then
73      ; ITT
74      ; ITTE
75      ; ITTEE:If Then 2 Else 2
76      LDRNE  r9, =0xDEADDEAD
77      MOVNE  r10, r9

```

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08次實習課\EXAMPLE15

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

Memory 1

Address: 0x20000000

0x20000066:	00 00
0x20000088:	00 00
0x200000AA:	00 00
0x200000CC:	00 00
0x200000EE:	33 33 00

Call Stack + Locals

Memory 1

進中斷前LR的值

Exception Exit : 1. PC = EXC_RETURN



上學期組語副函式EX :

BL TEST ; LR = return address

TEST

...

BX LR ; PC = LR = return address

15.5.2 EXIT

Returning from exceptions might be one of the few processes that is easier to do on a Cortex-M4 than on the ARM7TDMI, since the processor does most of the work for us. If we are in Handler mode and we wish to return to the main program, one of the following instructions can be used to load the EXC_RETURN value into the Program Counter:

BX LR (PC = LR = EXC_RETURN)

- A LDR or LDM instruction with the PC as the destination
- A POP instruction that loads the PC
- A BX instruction using any register

Exception Exit : 2. Unstacking

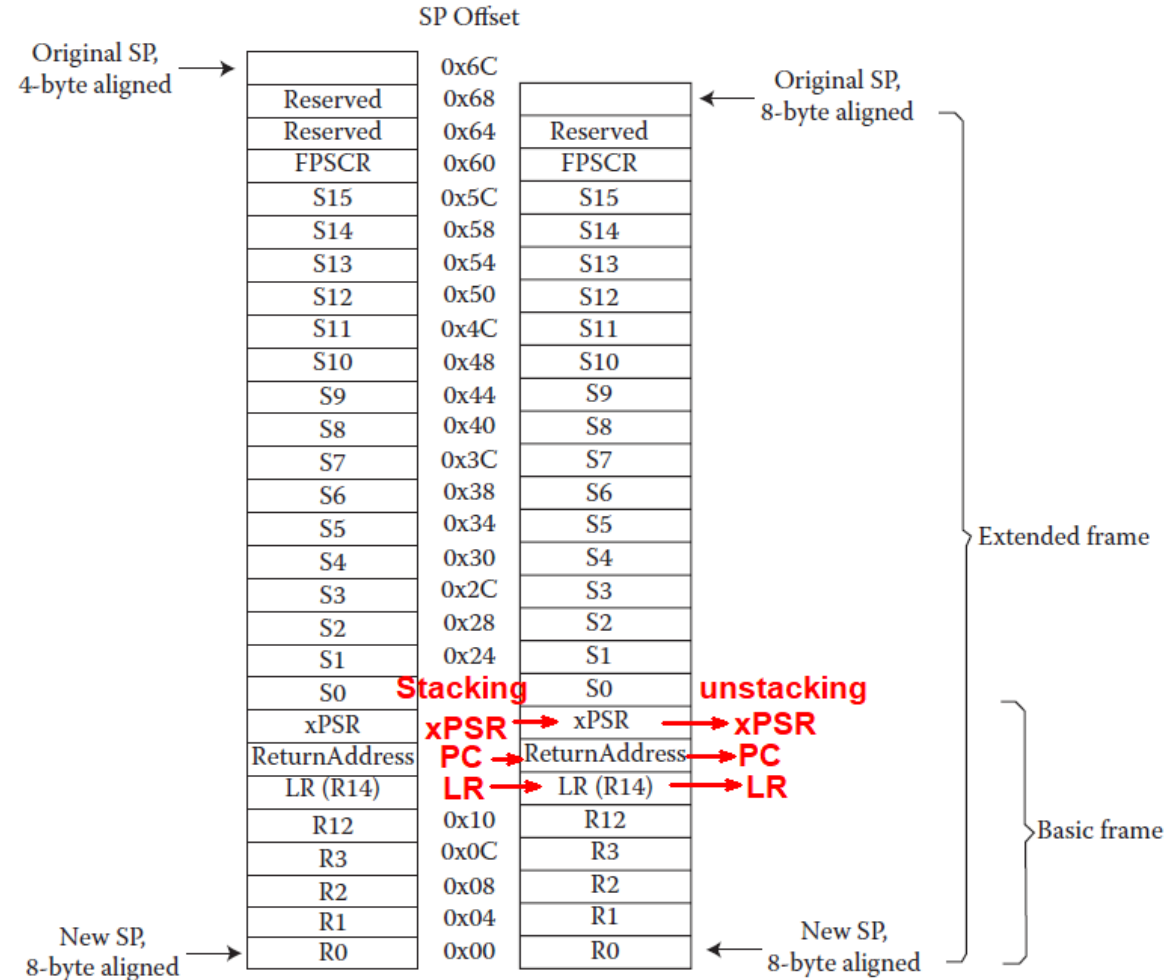
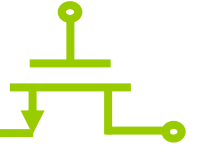
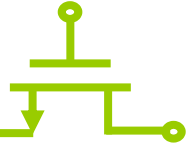


FIGURE 15.4 Exception stack frames.



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

Q&A

Thanks for your attention !!