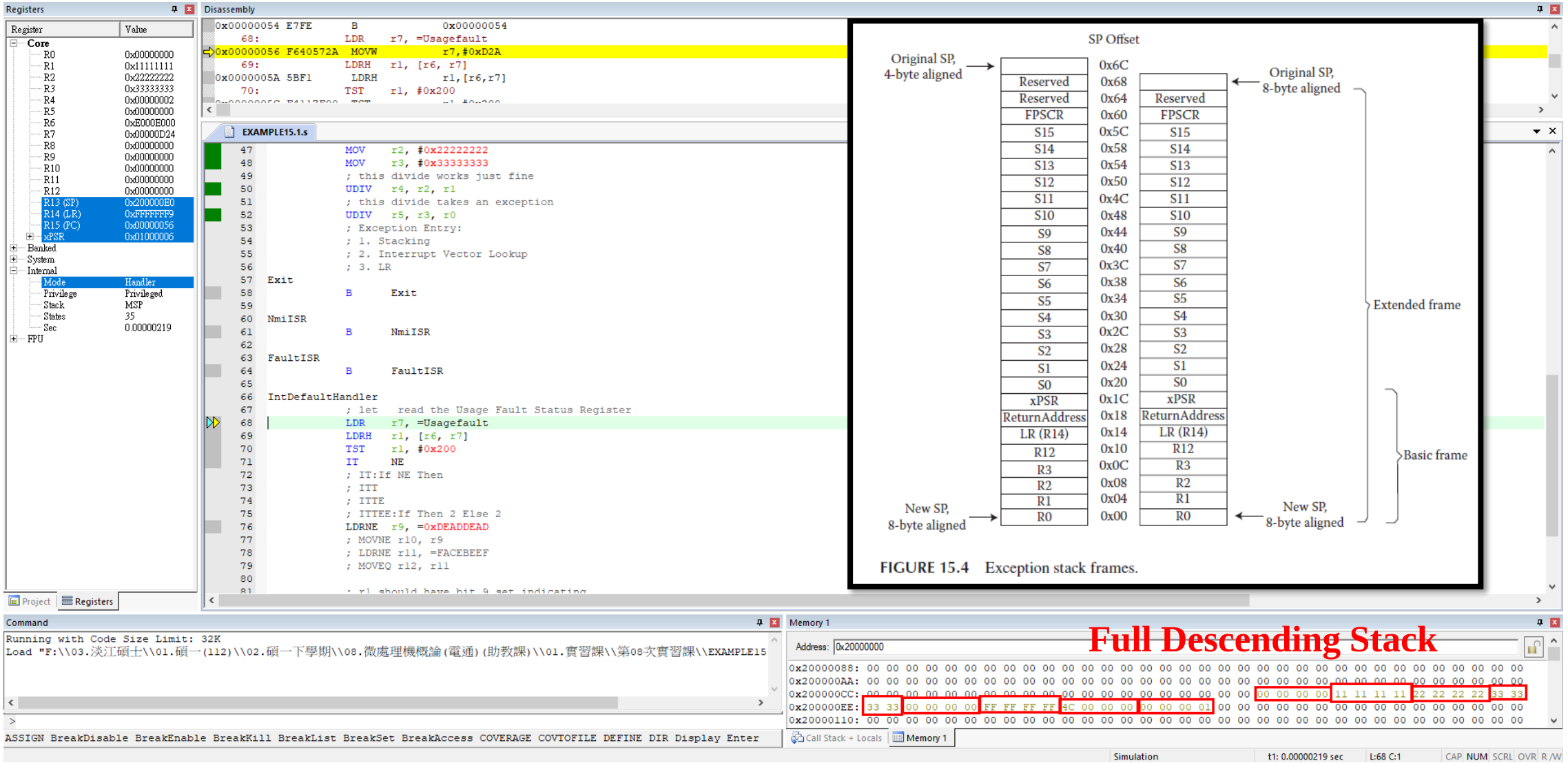
第10次實習課

學生：林培瑋

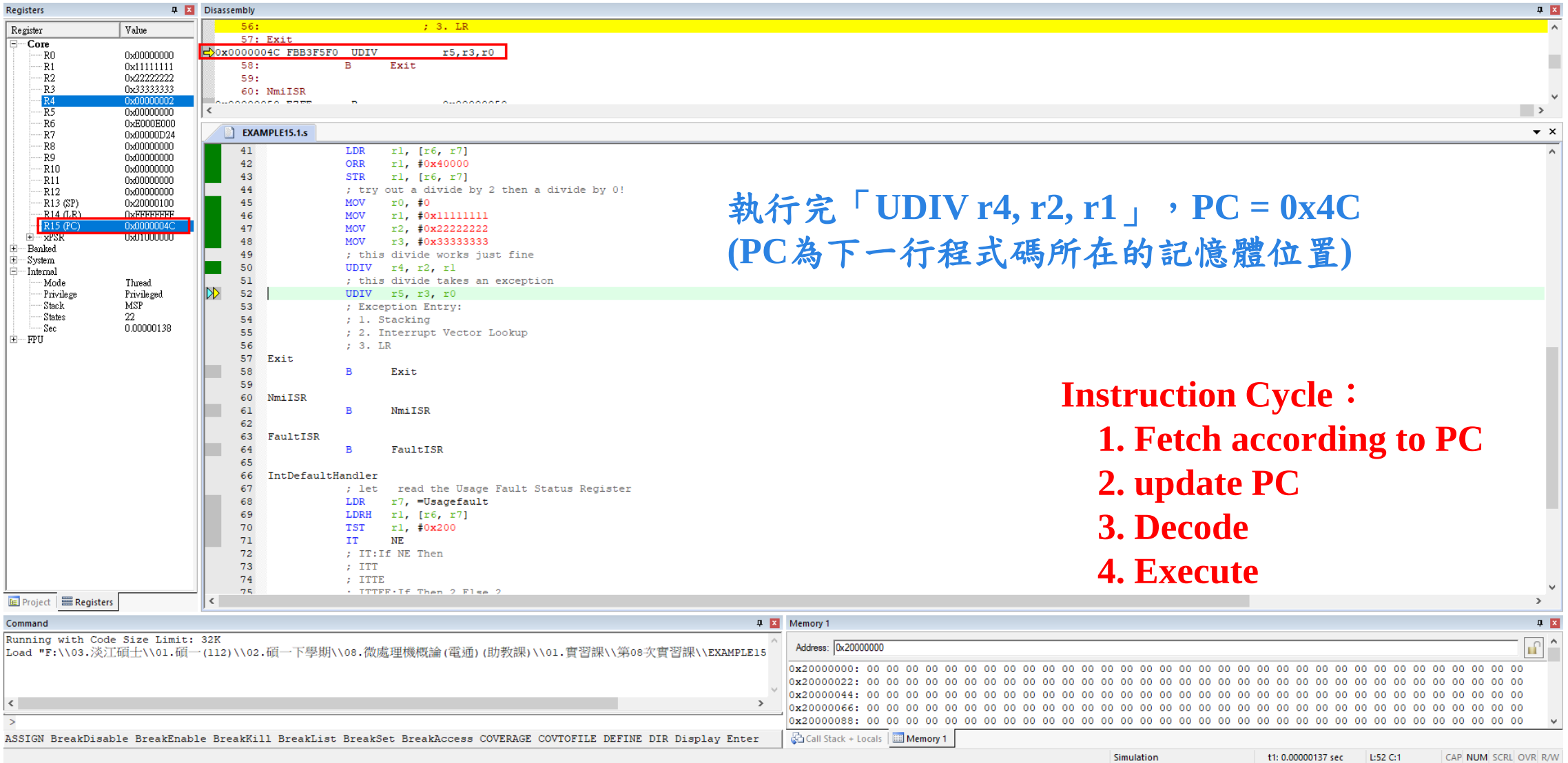
2024 Advanced Mixed-Operation System (AMOS) Lab.

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

Exception Entry：1. Stacking



Exception Entry：2. Interrupt Vector Table Lookup



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

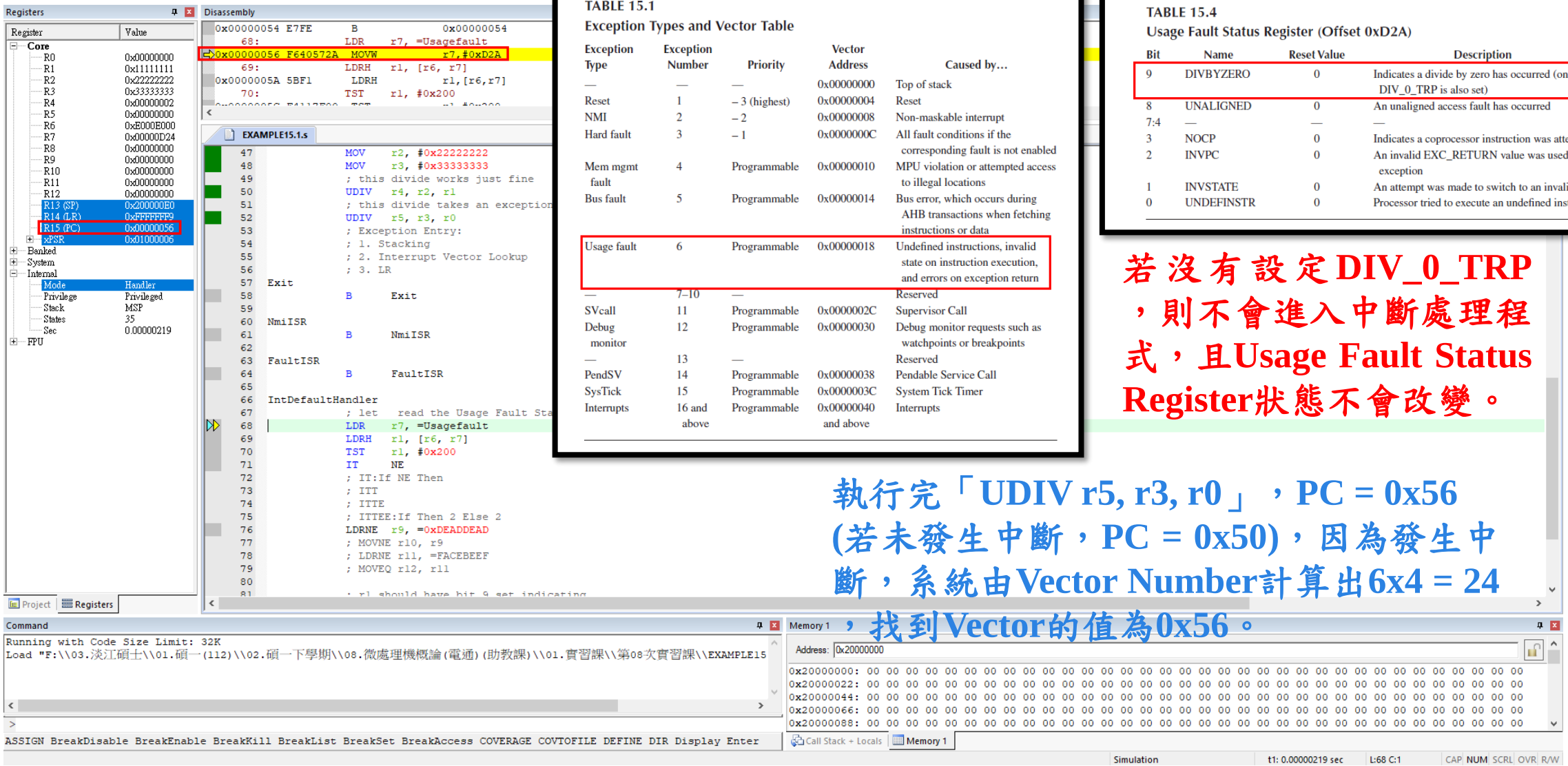


TABLE 15.1
Exception Types and Vector Table

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

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6x4 = 24，找到Vector的值為0x56。

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

15.4 STACK POINTERS

使用者、應用程式 — 系統

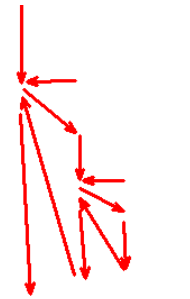
There are two stack pointers available to programmers, the Main Stack Pointer (MSP) and the Process Stack Pointer (PSP), both of which are called register r13; the choice of pointer depends on the mode of the processor and the value of CONTROL[1]. If you happen to have an operating system running, then 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 (Doeppner 2011).

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
0xFFFFFFE1	Floating-point	Handler mode	MSP
0xFFFFFFE9	Floating-point	Thread mode	MSP
0xFFFFFFED	Floating-point	Thread mode	PSP
0xFFFFFFF1	Non-floating-point	Handler mode	MSP
0xFFFFFFF9	Non-floating-point	Thread mode	MSP
0xFFFFFFFD	Non-floating-point	Thread mode	PSP

EXC_RETURN[31:0]	State	Return to	Using Stack Pointer
0xFFFFFFE1	Floating-point	Handler mode	MSP
0xFFFFFFE9	Floating-point	Thread mode	MSP
0xFFFFFFED	Floating-point	Thread mode	PSP
0xFFFFFFF1	Non-floating-point	Handler mode	MSP
0xFFFFFFF9	Non-floating-point	Thread mode	MSP
0xFFFFFFFD	Non-floating-point	Thread mode	PSP



Nested Vectored Interrupt Controller(NVIC)

1. Stacking

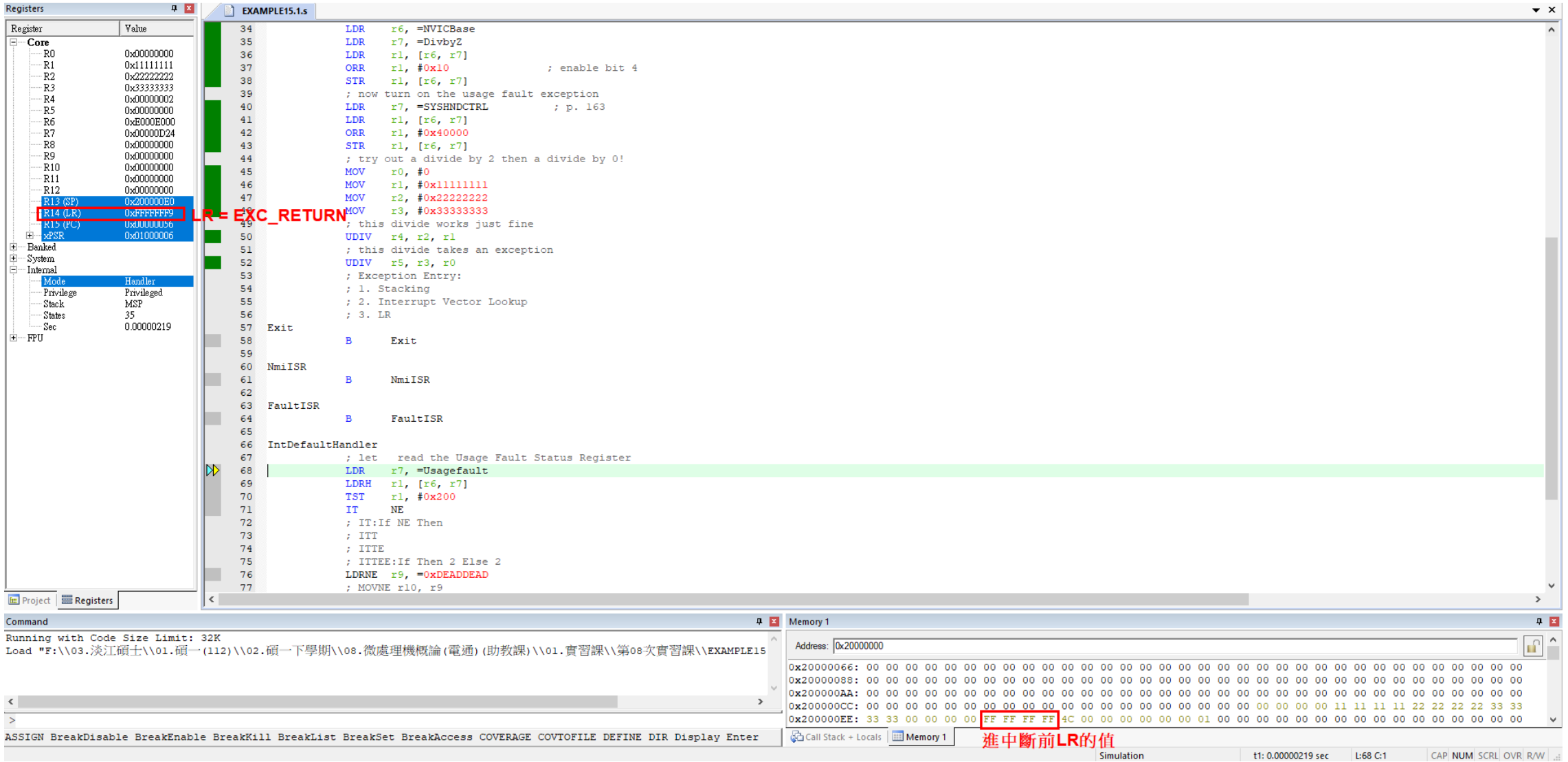
2. Interrupt Vector Table Lookup

PC ← 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

6x4=24 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

3. LR = EXC_RETURN ← 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 0xFFFFFFF9, since the floating-point unit was not enabled at the time we took the exception, and we wish to return to Thread mode.

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



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

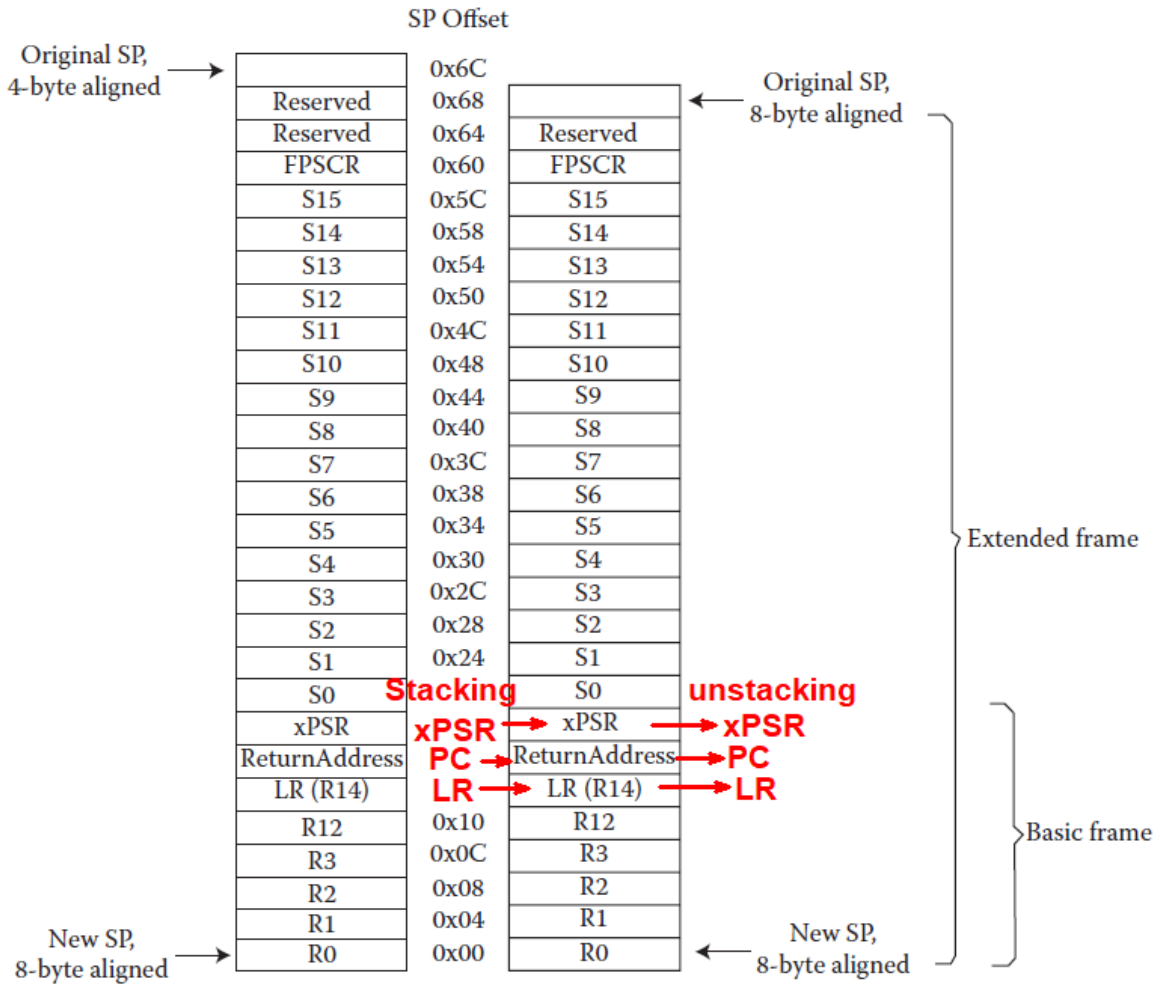


FIGURE 15.4 Exception stack frames.

Hard fault

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 !!