Exceptions

R. Ferrero

Politecnico di Torino

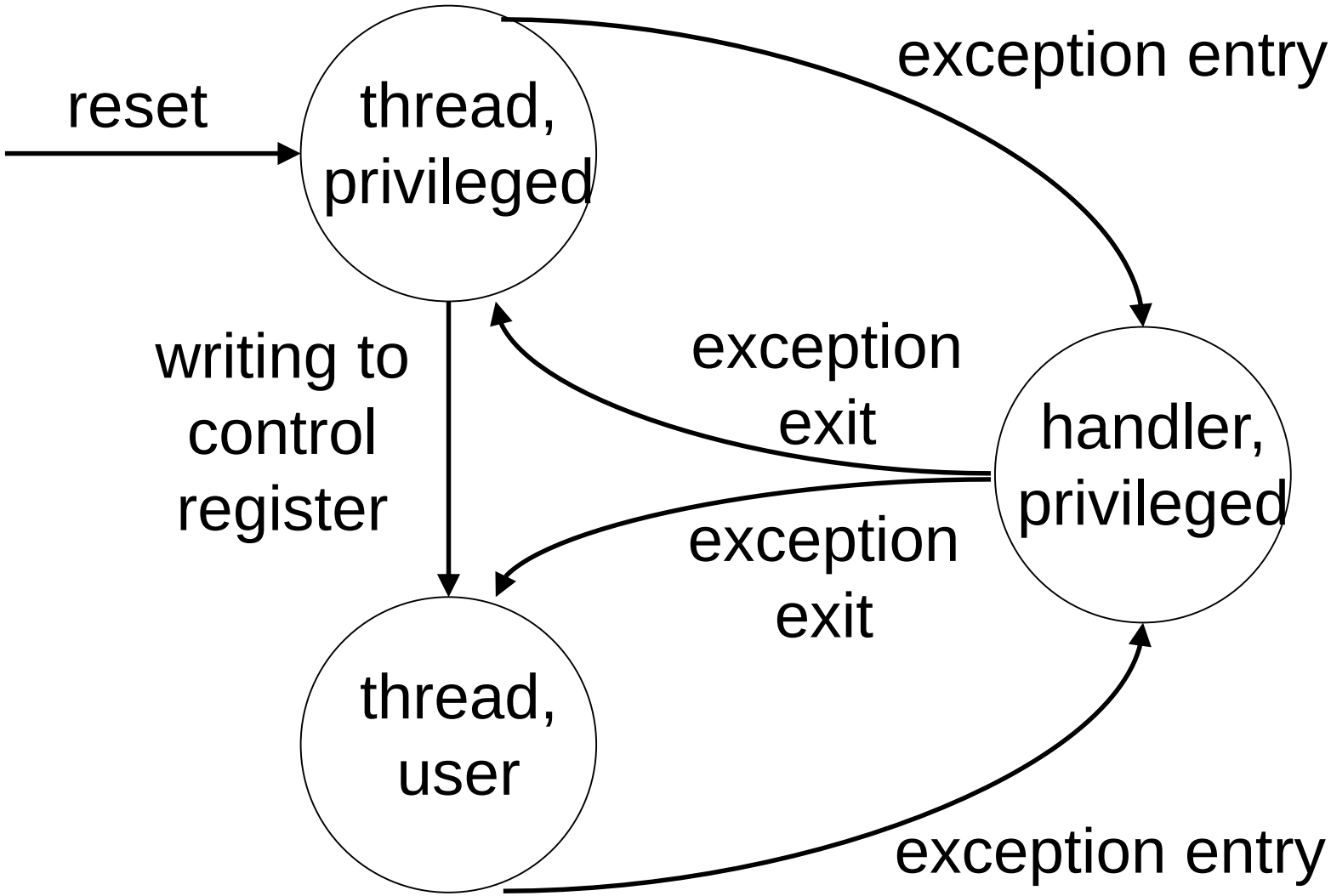
Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Processor modes

- Two operating modes:
  - thread mode: on reset or after completing the exception routine 线程模式，重置或者完成了异常处理例程之后
  - handler mode: when an exception occurs. 与处理模式
- Two access levels:
  - user level: limited access to resources 限制资源
  - privileged level: access to all resources. 所有资源
- Handler mode is always privileged. 与处理模式总是优先的

State diagram of processor modes



Control register

- CONTROL[0] changes the access level
  - 0: privileged level
  - 1: user level
- CONTROL[1] selects the current stack
  - 0: Main Stack Pointer (MSP)
  - 1: Process Stack Pointer (PSP)

四种状态

主堆栈指针

进程堆栈指针

	privileged level	user level
handler mode	CONTROL[0] = 0 CONTROL[1] = 0	not allowed
thread mode	CONTROL[0] = 0 CONTROL[1] = 0/1	CONTROL[0] = 1 CONTROL[1] = 0/1

Switching access level

- CONTROL[0] is writable only in privileged level.

仅仅可写，并不可读

- Switching to user level can occur in any privileged code (Thread or Handler mode).
  - Example:

```
MRS r8, CONTROL
ORR r8, r8, #1
MSR CONTROL, r8
```

或，有一个1为1　01=1，10=1，11=1，　00=0
与，同1为1　　01=0，10=0，11=1，　00=0
非，取反　　0=1 ，　1=0
异或，一个1为1　01=1，10=1，11=0，　00=0

move register to status

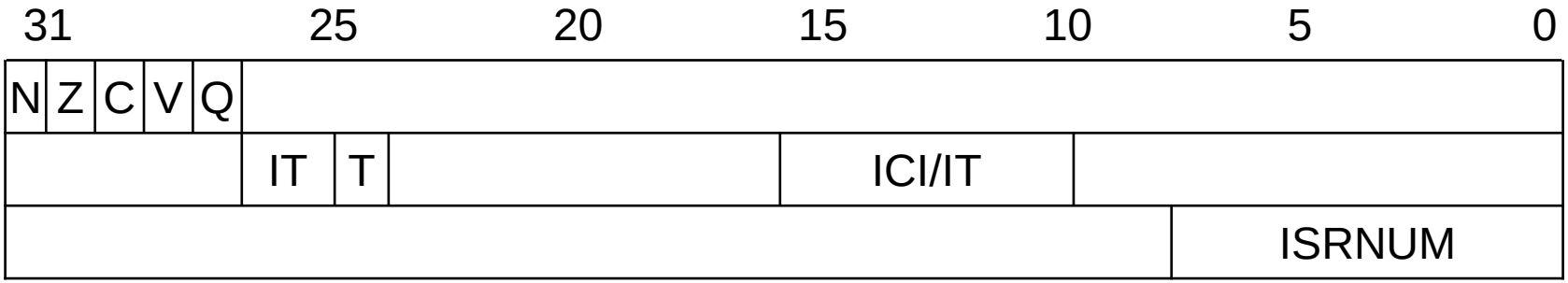
ORR按位或，把指定位置的bit置为1
BIC是先操作数2按位取反，再与操作数1进行与操作，放入寄存器中，就是把指定位置取反

- Switching to privileged level can occur only inside an exception handler (Handler mode).

Program Status Register

- It can be accessed as a combination of:
  - Application Program Status Register (APSR) 应用程序状态寄存器
  - Execution Program Status Register (EPSR) 执行程序状态寄存器
  - Interrupt Program Status Register (IPSR) 中断程序状态寄存器
- When an exception occurs, IPSR stores the exception number.

当一个异常出现的时候，IPSR存储异常数字



Exception sequence

程序员职能

Programmer's duty

- *relocate vector table**
- *set up priority level**
- enable the interrupt (only usage, bus and memory)
- implement exception handler

实现异常处理

* *optional task*

处理器职能

Processor's duty

- stack registers
- fetch vector table
- update registers
- execute exception handler
- unstack registers
- update NVIC registers

重定位向量表

设置优先级

使能够中断（仅，总线和内存）

堆叠寄存器

取向量表

更新寄存器

执行异常处理

取消堆叠寄存器

更新NVIC寄存器

嵌套中断向量控制器

Nested Vectored Interrupt Controller

- It is involved in the management of internal exceptions and up to 240 external interrupts.

设及内部异常的管理，以及高达240种外部中断

- It contains control registers
  - interrupt processing 中断运算
  - SYSTICK timer 时钟计时器
  - debugging 调试
- Registers are accessible as memory-mapped devices, starting from 0xE000E000.

寄存器可以作为内存映射设备来访问，并且从0xE000E000开始

- Most of registers are accessible only in privileged mode.

大部分寄存器只能通过高优先级来访问

Stacking registers

这些寄存器保存在当前使用的栈中

- r0-r3, r12, LR, PC and PSR are saved in the currently used stack (MSP or PSP)

在处理异常的过程中，主堆栈指针总是在使用

- Then, MSP is always used during exception
- PC and PSR are stacked first, so instruction fetch and PSR update can be started early.

PC和PSR总是排在前面的，所以取指和更新PSR可以开始的早一些

Register	r0	r1	r2	r3	r12	LR	PC	PSR
Stacking order	3	4	5	6	7	8	1	2
Offset in stack	32	28	24	20	16	12	8	4

都是按字对齐的，寄存器位置是按照序号来的

取中断向量表

Fetching interrupt vector table

嵌套中断向量控制器（NVIC）把异常编码放到数据总线上

- NVIC puts the exception code on the data bus

CPU把这个编码当作索引去访问中断向量表中的向量

- The CPU uses the code as an index to access a vector named Interrupt Vector Table (IVT).
- Each entry of the IVT manages an exception: it contains either one ARM instruction or an addresses in memory.

中断向量表中的每个条目管理着一个异常
它包含一个ARM指令或者一个内存中的地址

- In the v7-M architecture, there are addresses of the exception service procedures.

在这个架构中，有一些异常服务程序的地址

Vector table

Exception type	Priority	Position	Vector address
(Top of Stack)		0	0x00000000
Reset	-3	1	0x00000004
NMI	-2	2	0x00000008
Hard fault	-1	3	0x0000000C
Memory management fault	programmable	4	0x00000010
Bus fault	programmable	5	0x00000014
Usage fault	programmable	6	0x00000018
Reserved		7-10	
Supervisor call	programmable	11	0x0000002C
Debug monitor	programmable	12	0x00000030
Reserved		13	
PendSV	programmable	14	0x00000038
SysTick	programmable	15	0x0000003C
External interrupts	programmable	16 - 255	≥0x00000040

Register updates

下面的寄存器将会被更新，当进入一个异常处理时

- The following registers are updated when entering the exception handler:
  - `SP` (either `MSP` or `PSP`): new value after stacking
  - `PSR` (`IPSR` only): exception number 异常数字
  - `PC`: address of the exception handler 异常处理的地址
  - `LR`: exception return information EXC_RETURN 异常返回信息
  - some NVIC registers: pending status of exception is cleared, active bit of exception is set.

一些嵌入中断向量控制器：清除异常挂起状态，激活异常位

EXC_RETURN

- Bit 0: processor state. 0 -> ARM, 1 -> Thumb
  处理器状态
- Bit 1: reserved, it must be 0
  保留位，必须为0
- Bit 2: return stack. 0 -> MSP, 1 -> PSP
  返回栈。0是主存堆栈指针，1是进程堆栈指针
- Bit 3: return mode. 0 -> handler, 1 -> thread
  返回模式。0是处理模式，1是线程模式
- Bits 4-31: each bit must be 1
  第4-31位必须都是1
- Allowed values of EXC_RETURN are:
  - 0xFFFFFFF1: return to handler mode
  - 0xFFFFFFF9: return to thread mode with MSP
  - 0xFFFFFFFD: return to thread mode with PSP

Exception exits

异常退出

1. The interrupt return sequence is triggered

中断返回序列被触发

   - `BX LR` 返回中断点
   - `PUSH {…, LR}` and `POP {…, PC}` 把当前寄存器里的内容压入栈中，并弹出栈中的内容到寄存器中
   - `STM xx, {…, LR}` and `LDM xx, {…, PC}` 把当前寄存器列表存入到xx为基址的内存中，或把以xx为基址的内容载入到当前寄存器列表中
2. Unstacking: registers pushed to the stack are restored in the same order 寄存器压入栈中，并以相同的序列存储
3. NVIC register update 嵌入中断向量控制器 寄存器更新
   - the active bit of the exception is cleared

异常激活位被清除

   - if the input of an external interrupt is still asserted, the pending bit is set again, so handler is reentered

如果一个外部中断的输入依然存在，那么挂起位会被重置，所以处理会重返

Exception handler

处理者可能会采取的操作

- Possible operations performed by the handler:
  - reading a fault status register (FSR) to know the cause of usage, bus, memory management fault

读取故障寄存器，获取总线，内存管理故障的原因

  - reading BFAR or MMFAR to get the memory location involved in a precise fault (bus or memory)

读取总线故障地址寄存器和内存管理寄存器来获取内存位置（涉及一个精确的错误）

  - reading the stacked program counter to retrieve the offending instruction

读取堆栈程序计数器来检索这个错误的指令

- Value of bits can be checked through a mask.

位的数值可以通过 屏蔽 检查

```
LDR r0, =FSRaddress
LDR{size} r1, [r0]
TEQ r1, #mask
```

Fault status registers (FSR)

FSP中的位表示故障原因

- Bits in a FSR indicate the cause of the fault
- Bits can be read or cleared by writing 1.

位 可以通过写1来读取或者清楚

Register	Size	Address
Memory Management Fault Status register 存储管理错误状态寄存器	byte	0xE000ED28
Bus Fault Status register	byte	0xE000ED29
Usage Fault Status register 总线错误状态寄存器	half word	0xE000ED2A
Hard Fault Status register 使用错误状态寄存器	word	0xE000ED2C
Debug Fault Status register 硬件错误状态寄存器	word	0xE000ED30
Auxiliary Fault Status register 调试错误状态寄存器	word	0xE000ED3C

辅助错误状态寄存器

Other useful registers

Register	Type	Size	Address
Configuration Control Register 配置控制寄存器	R/W	word	0xE000ED14
System Handler Control and State Register 系统处理控制和状态寄存器	R/W	word	0xE000ED24
Memory Management Fault Address Register (MMFAR) 内存管理错误地址寄存器	R	word	0xE000ED34
Bus Fault Address Register (BFAR) 总线错误地址寄存器	R	word	0xE000ED38

System exception types

- Reset 重启
- Non maskable interrupt (NMI) 不可屏蔽中断
- Hard fault 硬件错误
- Memory management fault 内存管理错误
- Bus fault 总线错误
- Usage fault 用途错误
- Supervisor call (SVC) 管理程序调入
- Debug monitor 调试监控
- Pendable Service Call 可挂起服务调入
- System Tick Timer (SYSTICK) 系统节拍定时器

Non maskable interrupt (NMI) 非屏蔽中断

- The use of NMI depends on the SoC design.
- For example, NMI can be connected to: 依赖于系统芯片设计
  - watchdog timer 监视时钟
  - voltage-monitoring block, which warns the CPU when the voltage drops below a certain level.
    电压检测块，当电压降低至某一水平时，给CPU发出警告
- The NMI exception can be activated any time, even right after the core exits reset.
  NMI异常可能在任何时间出现，即使在核心退出重置之后
- It has a fixed priority levels, equal to -2.
  有固定的优先级，值为-2
  - the negative number indicates that it is of higher priority than all other exceptions, except Reset.

这个负数说明了，他的优先级高于其他任何异常，重启除外

Hard Fault Status register

硬件错误状态寄存器

Bit	Possible causes of hard fault
31	Debug event (breakpoint or watchpoint).
30	A bus fault, memory management fault, or usage fault occurred, but the corresponding handler is disabled or cannot be started because another exception with same or higher priority is running, or because exception mask is set.
1	Failed interrupt vector fetch during exception processing (due to bus fault or incorrect vector table offset setup).

调试事件（断点或者观察点）

在异常处理期间，不能成功取出 中断向量（由于总线故障 或者 向量表的偏移量设置错误）

总线错误，内存管理错误，用途错误出现，但是相应的处理者死机或者不能启动，因为其他具有相同或者更高优先级异常正在处理，或者因为设置了异常屏蔽

Hard fault example

- Attempting to switch to ARM state cause a usage fault. 尝试转换到ARM状态，会导致用途错误
- If usage fault it is not enabled, an hard fault is generated. 如果用途错误未启用，那么硬件故障就会出现

```
ADRL r0, label
BLX r0
```

系统处理程序控制

System Handler Control

Bit	Type	Description
18	R/W	Usage fault handler enable
17	R/W	Bus fault handler enable
16	R/W	Memory management fault handler enable
15	R/W	SVC pended (replaced by higher-priority exception)
14	R/W	Bus fault pended
13	R/W	Memory management fault pended
12	R/W	Usage fault pended
11	R/W	Read as 1 if SYSTICK exception is active
10	R/W	Read as 1 if PendSV exception is active
8	R/W	Read as 1 if debug monitor exception is active
7	R/W	Read as 1 if SVC exception is active
3	R/W	Read as 1 if usage fault exception is active
1	R/W	Read as 1 if bus fault exception is active
0	R/W	Read as 1 if memory management fault is active

开关：用途错误处理

开关：总线错误处理

开关：内存管理错误处理

挂起：管理程序挂起（替换为更高优先级的异常）

挂起：总线故障挂起

挂起：内存管理故障挂起

挂起：用途错误

读取1：激活：系统节拍定时器异常

读取1：激活：可挂起系统调用异常

读取1：激活：调试监控异常

读取1：激活：管理程序异常

读取1：激活：用途错误异常

读取1：激活：总线故障异常

读取1：激活：内存管理故障异常

Enabling faults

```
LDR r2, =SystemHandlerControl
LDR r1, [r2]
; enabling usage fault
ORR r1, #0x40000
; enabling bus fault
ORR r1, #0x20000
; enabling memory mngm. fault
ORR r1, #0x20000
STR r1, [r2]
```

或运算　使R1的第19位置为1

或运算，使R1的第18位置为1

#0x10000

存入，SystemHandlerControl的内存块中

Usage Fault Status Register

用途错误状态寄存器

当CPU进行除0操作时，会导致故障或者停止

Bit	Possible causes of usage fault
9	Division by zero and DIV_0_TRP is set. 除以0，或者设置了DIV_0_TRP
8	Unaligned memory access attempted and UNALIGN_TRP is set 尝试访问未对齐内存，或者设置了UNALIGN_TRP
3	Attempt to execute a coprocessor instruction 尝试执行一个协处理器指令
2	Invalid EXC_RETURN during exception return. 无效的EXC_RETURN，在异常返回期间 Invalid exception active status. 无效的异常激活状态 Invalid value of stacked IPSR (stack corruption). IPSR栈的无效值（栈溢出） Invalid ICI/IT bit for current instruction. 对于当前值，无效的ICT/IT位
1	Branch target address to PC with LSB equals 0. 在LSB=0 时，转移目标地址到PC中
0	Use of not supported (undefined) instruction. 使用了不支持的指令

Usage fault examples (1)

- switch to ARM state

```
ADRL r0, label
BLX r0
```

- coprocessor instruction

```
LDC p1, c0, [r1]
```

LDC和STC皆为协处理器指令

- undefined instruction:

```
DCD 0xe7f0def0
```

within the executed code.

Usage fault examples (2)

- division by zero

```
LDR r4, =ConfigurationControl
LDR r3, [r4]
ORR r3, #0x10
STR r3, [r4]
MOV r0, #0
MOV r1, #0x11111111
UDIV r2, r1, r0
```

或运算 第5位 置为 1

Usage fault examples (3)

- unaligned access

```
        LDR r4, =ConfigurationControl
        LDR r3, [r4]
        ORR r3, #0x8
        STR r3, [r4]
        LDR r1, =myVar1
        LDR r3, [r1]  ;unaligned access?
        LDR r2, =myVar2
        LDR r4, [r2]  ;unaligned access
stop    B stop
myVar1  DCB 0x11
myVar2  DCB 0x22
```

或运算　第4位置为1

ORR指令集为Thumb
LDR指令集为ARM

所以出现了对齐问题

Configuration Control Register

配置控制寄存器

Bit	Description
9	Force exception stacking start in double word aligned address. 强制异常进栈，开始于双字对齐地址
8	Ignore data bus fault during hard fault and NMI. 在硬中断和非屏蔽中断时，忽略总线错误
4	Trap on divide by 0 落入除以0的陷阱
3	Trap on unaligned accesses 落入非对齐访问的陷阱
1	If set to 1, allow user code to write to Software Trigger Interrupt register 如果设置为1，则允许用户代码写入软件触发中断寄存器
0	If set to 1, allows exception handler to return to thread state at any level by controlling return value.

如果设置为1，则允许异常处理程序通过控制返回值，在任何等级下返回到线程状态，

Bus Fault Status Register

Bit	Possible causes of bus fault
7	Indicates that the Bus Fault Address Register (BFAR) contains a valid bus fault address 指明总线故障地址寄存器包含一个有效的总线故障地址
4	Stacking error: stack pointer is corrupted, or stack size became too large (reaching an undefined memory region), or PSP is used but not initialized 栈错误：栈指针被破坏，或者栈尺寸过大（到达一个未定义的内存边界），或者PSP被使用但是没有初始化
3	Unstacking error or stack pointer changed during exception 非堆栈错误，或者堆栈指针改变，在异常期间
2	Imprecise data access violation: device not initialized, access of privileged-only device in user mode, incorrect transfer size for the device. 不精确的数据访问冲突：设备没有初始化，私有设备访问，设备的不正确的转换尺寸
1	Precise data access violation: BFAR indicates fault address 精确的数据访问冲突：BFAR指示错误的地址
0	Instruction access violation: branch to invalid region, invalid exception return code, invalid entry in interrupt vector table, access to NVIC in user mode.

指令访问冲突：转移到无效区域，无效异常返回码，无效进入中断向量表，在用户模型中，访问NVIC

Bus fault examples (1)

- PSP is used but not initialized

```
MRS r8, CONTROL
ORR r8, r8, #2
MSR CONTROL, r8
PUSH {r1}
```

- To avoid the bus fault, before the `PUSH` add

```
LDR SP, =initial_psp
```

应先存寄存器SP中的值到栈中

with this previous memory allocation:

```
    AREA PSTACK, READWRITE, ALIGN=3
    SPACE   Stack_Size
initial_psp
```

Bus fault examples (2)

- access to a privileged memory location

```
MRS r8, CONTROL ;to user level
ORR r8, r8, #1
MSR CONTROL, r8
LDR r0, =0xE000ED92
LDR r1, [r0]
```

- the data access violation is precise: the fault address can be read in BFAR

```
LDR r7,=BusFaultAddressRegister
LDR r2, [r7]
```

内存管理状态寄存器

MemManage Status Register

Bit	Possible causes of memory management fault
7	Indicates that the Memory Manage Address Register (MMAR) contains a valid fault addressing value
4	Stacking error: stack pointer is corrupted, or stack size became too large (reaching a region not defined by the MPU or disallowed in the MPU configuration)
3	Unstacking error or MPU configuration changed by exception handler.
1	Violation to memory access protection, which is defined by MPU setup. For example, user application trying to access privileged-only region.
0	Violation to memory access protection; branch to nonexecutable regions; invalid exception return code; invalid entry in exception vector table.

说明MMSAR包含了一个有效的故障地址值

栈错误：栈指针被损毁，或者栈的尺寸过大（到达通过MPU未定义的边界，或者不被MPU的配置所允许）

非栈错误或者通过异常处理程序 改变了MPU的配置

内存访问保护冲突，被MPU设置所定义。例如，用户应用尝试访问私有区域

内存访问保护冲突；转移到不可执行区域；无效的异常返回码；在异常向量表无效进入

Memory management fault example

内存管理故障例子

- branch to nonexecutable region

```
LDR r0, =0x40000001
BX r0
```

- 0x40000000 is the starting address of the peripherals region. 这个地址时外围设备的起始地址区域
- the least significant bit of the address must be 1 to avoid a usage fault.

地址最低位的bit值必须位1来避免用途错误

Supervisor call 管理程序调用

- A supervisor call exception is generated as:

  `SVC #immediate` 管理程序调用异常生成如下：

- E.g.: the operating system provides access to hardware through a system function call `SVC`
  操作系统通过叫SVC的系统功能来访问硬件
- When the software exception handler in the operating system is executed, it provides the service requested by the user application.
  当软件异常处理程序在操作系统中被执行，他提供了用户应用程序要求的服务
- Benefits:
  - robustness: access to hardware is under OS control
    健壮性：在OS的控制下访问硬件
  - portability: the user application does not need to know the programming details of the hardware.
    可移植性：用户程序不需要知道硬件的编程细节

Handler of supervisor call

管理程序调用的程序

- The handler can determine the immediate data value in the SVC instruction as follows:

处理程序可以决定在SVC指令中当前的数据值，如下：

1. determining the stack used for stacking registers

确定用于栈寄存器的堆栈

2. reading the stacked PC 读取栈指针PC
3. reading the instruction at the address pointed by the stacked PC 读取栈指针PC指向的地址
4. masking out the unneeded bits (operating code)

掩盖不需要的Bits（操作码）

- The SVC instruction is encoded in 16 bits, and the immediate value is stored in the least significant byte.

SVC指令是16位编码，并且立即数存储在最近的标志字节中

Implementation of SVC handler

```
;Test bit 2 of EXC_RETURN in LR
TST LR, #0x4
ITE EQ
MRSEQ r0, MSP
MRSNE r0, PSP
;get stacked PC from stack
LDR r1, [r0, #24]
;get immediate from instruction
LDRB r0, [r1, #-2]
```

测试LR的第4位是否为1，不为1的话CPSR中的标志位Z=1，如果为1的话，Z=0

IT：IF-THEN 接下来的一条指令条件执行
ITE：IF-THEN-THEN 接下来的两条指令条件执行
ITT：
ITTE：
ITTEE：

SYSTICK exception 系统节拍定时器异常

- A 24-bit timer is used to generate interrupts 一个24位的定时器用来产生异常的
  - internal timer: core free-running clock 内部定时器：核心不同步的时钟
  - external timer: external reference clock (STCLK) 外部定时器：外部参考时钟
- Possible uses: 可能的使用
  - alarm timer 报警计时
  - timing measurement 时间测量
  - context switch among multiple tasks 多个任务之间的上下文切换
- Controlled by four registers. 由四个寄存器控制

SYSTICK registers

系统节拍定时器寄存器

SYSTICK Register	Size	Address
Control and Status Register	32 bits	0xE000E010
Reload Value Register	24 bits	0xE000E014
Current Value Register	24 bits	0xE000E018
Calibration Value Register	32 bits	0xE000E01C

控制和状态寄存器

重载值寄存器

当前值寄存器

校准值寄存器

- SYSTICK Reload Value Register stores the value to reload when timer reaches 0.

时钟重载值寄存器，存储定时器到达0时，要重载的值

- SYSTICK Current Value Register stores the current value of the timer. Writing any number clears its content.

系统节拍定时器 当前值寄存器存储当前定时器下的。写任何数字来清除它的内容

SYSTICK Control and Status Register

系统节拍定时器 控制和状态寄存器

Bit	Type	Description
16	R	Read as 1 if counter reaches 0 since last time this register is read; clear to 0 when read or when current counter value is cleared
2	R/W	1 = use processor free running clock 0 = use external reference clock (STCLK)
1	R/W	1 = generate interrupt when timer reaches 0 0 = Do not generate interrupt
0	R/W	1 = enable SYSTICK Timer 0 = disable SYSTICK Timer

如果从最后一次这个寄存器读取，计数器达到0，那么读取值位1；当读取或者当前计数器的值被清空，那么清除并置0

1：用处理器的不同步时钟　　0：用额外的参考时钟

1：当计时器到达0时，生成中断　　0：不生成中断

1： 开启时钟节拍定时器

0：关闭时钟节拍定时器

SYSTICK Calibration Value Register

时钟节拍定时器 校准值寄存器

Bit	Type	Description
31	R	1 = external reference clock not available 1：额外参考时钟不可用 0 = external reference clock available
30	R	0：外部的参考时钟可用 1 = calibration value is not exactly 10 ms 0 = calibration value is accurate
23:0	R/W	1：校正值不能超过精准的10ms　0：校准值是准确的 Calibration value for 10 ms. If this value is read as 0, calibration value is not available.

校准值为10m，如果这个值是0，那么校准值不可用

SYSTICK configuration

时钟节拍定时器配置

1. Stop timer counter to prevent interrupt triggered accidentally. 停止时钟计数器，防止意外中断触发
   - write 0 to the Control and Status register
2. Set the desired interval between interrupts 在中断和状态寄存器中写入0
   - write a new value to the Reload Value register. 设置所需的中断间隔 写一个新的值到重载值寄存器中
3. Reset the SYSTICK timer
   - write any value to the Current Value register 重置系统节拍定时器的计数器 写入任何值到当前值寄存器
4. Start the SYSTICK timer 开始时系统节拍定时器
   - write proper value to Control and Status register.

写入适合的值到控制和状态寄存器

Code for SYSTICK configuration

系统节拍的配置代码

```
LDR r0, =SYScontrolAndStatusReg
MOV r1, #0
STR r1, [r0]      ; step 1
LDR r0, =SYSreloadValueReg
LDR r1, =1023     ; example
STR r1, [r0]      ; step 2
LDR r0, =SYScurrentValueReg
STR r1, [r0]      ; step 3
LDR r0, =SYScontrolAndStatusReg
MOV r1, #7
STR r1, [r0]      ; step 4
```

把系统控制和状态寄存器 置为0

把系统重载值寄存器置为1023

把系统当前值寄存器置为1023

把系统控制和状态寄存器置为7