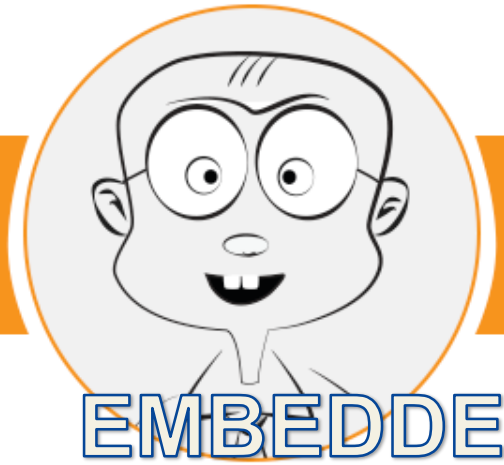


EMBEDDED SYSTEM COURSE

LECTURE 5: EXCEPTION & INTERRUPT

Learning Goals

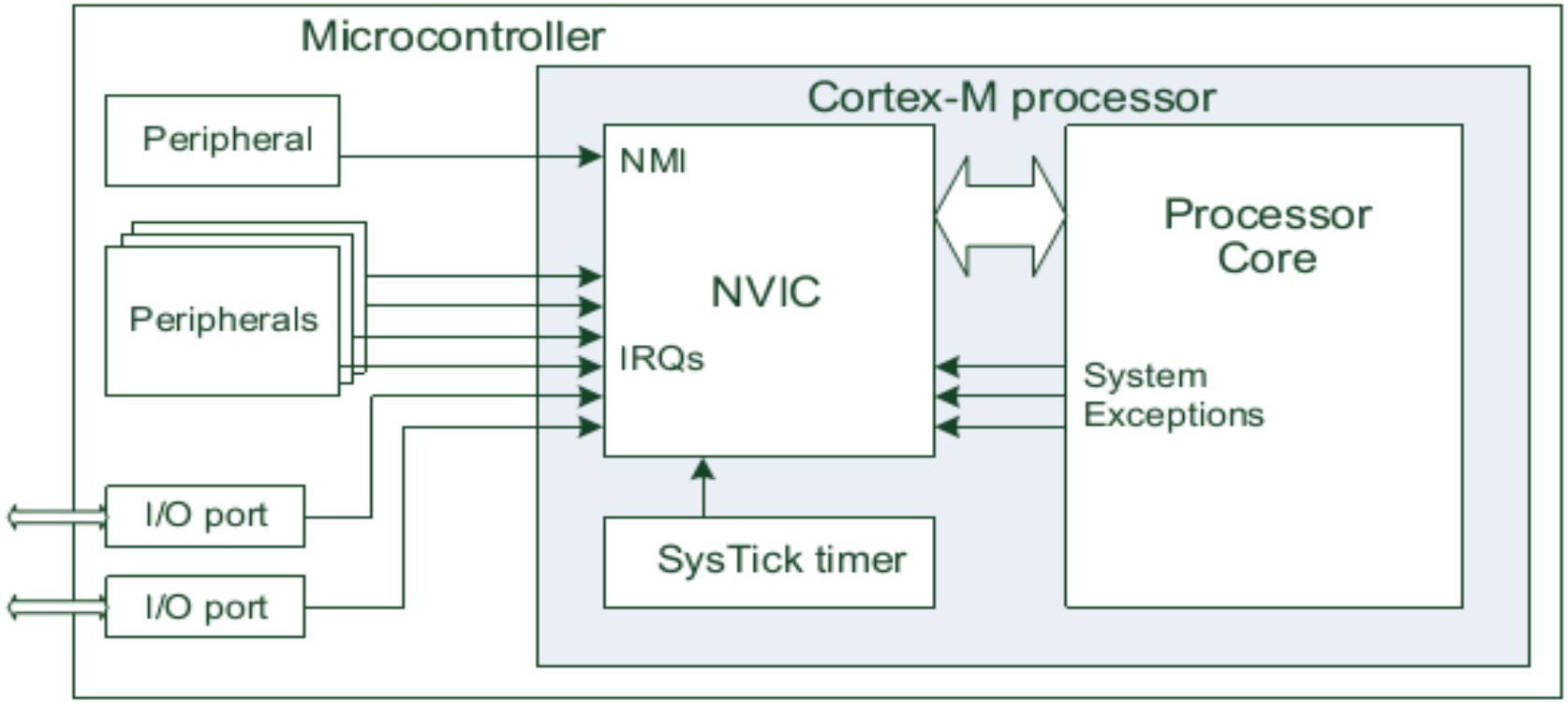
- Describe detail about the Cortex-M interrupt handling (to be more specific: NVIC controller).
- Describe about the exception entry/return sequence and how to optimize the interrupt latency.

Table of contents

Table of contents

Overview of interrupt management

- Exception sources



Various sources of exceptions in a typical microcontroller

Overview of interrupt management

- System Exceptions and Interrupt

Excp. Number	Exception Type	Priority
1	Reset	-3 (highest)
2	NMI	-2
3	Hard Fault	-1
4	MemManage Fault	Programmable
5	Bus Fault	Programmable
6	Usage Fault	Programmable
7-10	Reserved	NA
11	SVC	Programmable
12	Debug Monitor	Programmable

Excp. Number	Exception Type	Priority
13	Reserved	NA
14	PendSV	Programmable
15	SysTick	Programmable
16	Interrupt #0	Programmable
17	Interrupt #1	Programmable
…		
47	Interrupt #31	Programmable
…		
255	Interrupt #239	Programmable

Overview of interrupt management

Exception Properties

- Exception number: Identification for the exception
- Vector address: Exception entry point in memory
- Priority level: Determines the order in which multiple pending exceptions are handled

Table of contents

Vector Table

- ***Vector Table:***
  - After powerup, vector table is located at 0x00000000
  - Vector table contains:
    - Initial value for Main Stack Pointer
    - Handler vector addresses

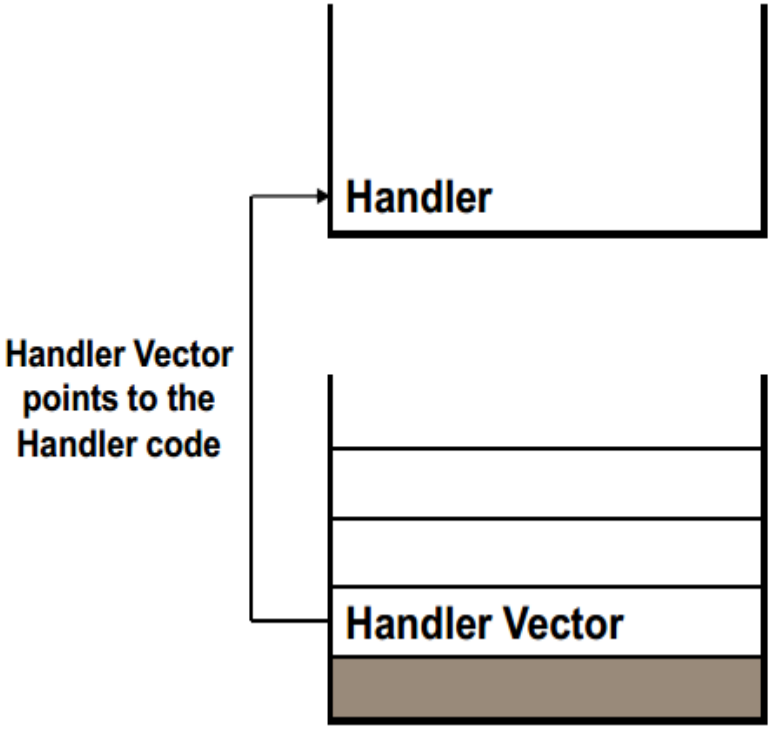
Address	Exception Number	Value (Word Size)
0x00000000	-	MSP initial value
0x00000004	1	Reset vector
0x0000008	2	NMI handler starting address
0x000000C	3	Hard fault handler starting address
……	…….	Other handler starting address

Vector Table

Vector Table Usage:

In the case of an exception, the core:

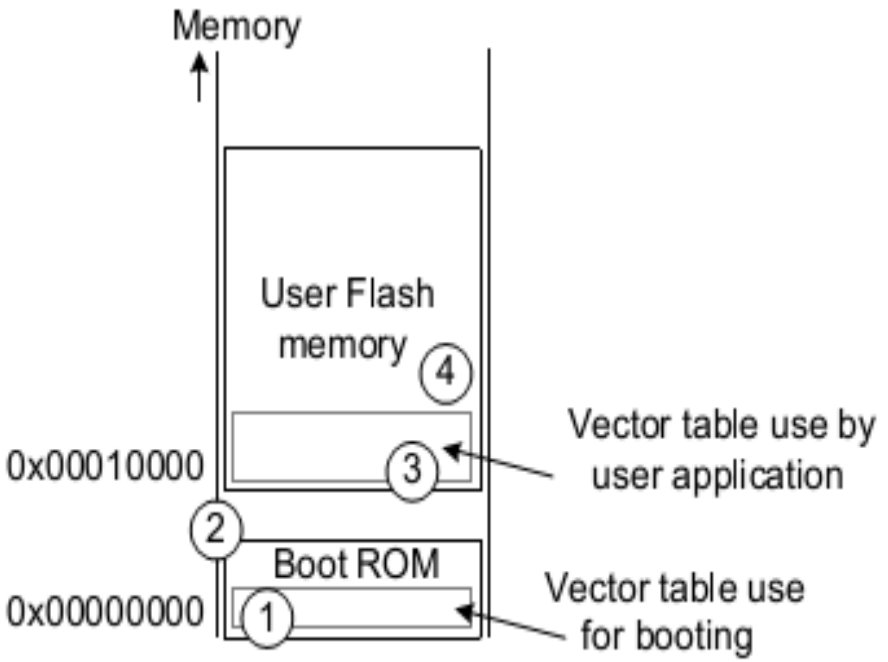
- Reads the vector handler address for the exception from the vector table
- Branches to the handler



Vector Table & Vector Table Relocation

Vector Table Relocation:

- Vector table can be relocated to change interrupt handlers at run-time
- Some cases using vector table relocation feature:
  - Boot loader
  - Applications load to RAM
  - Dynamic changing of Vector



Vector Table

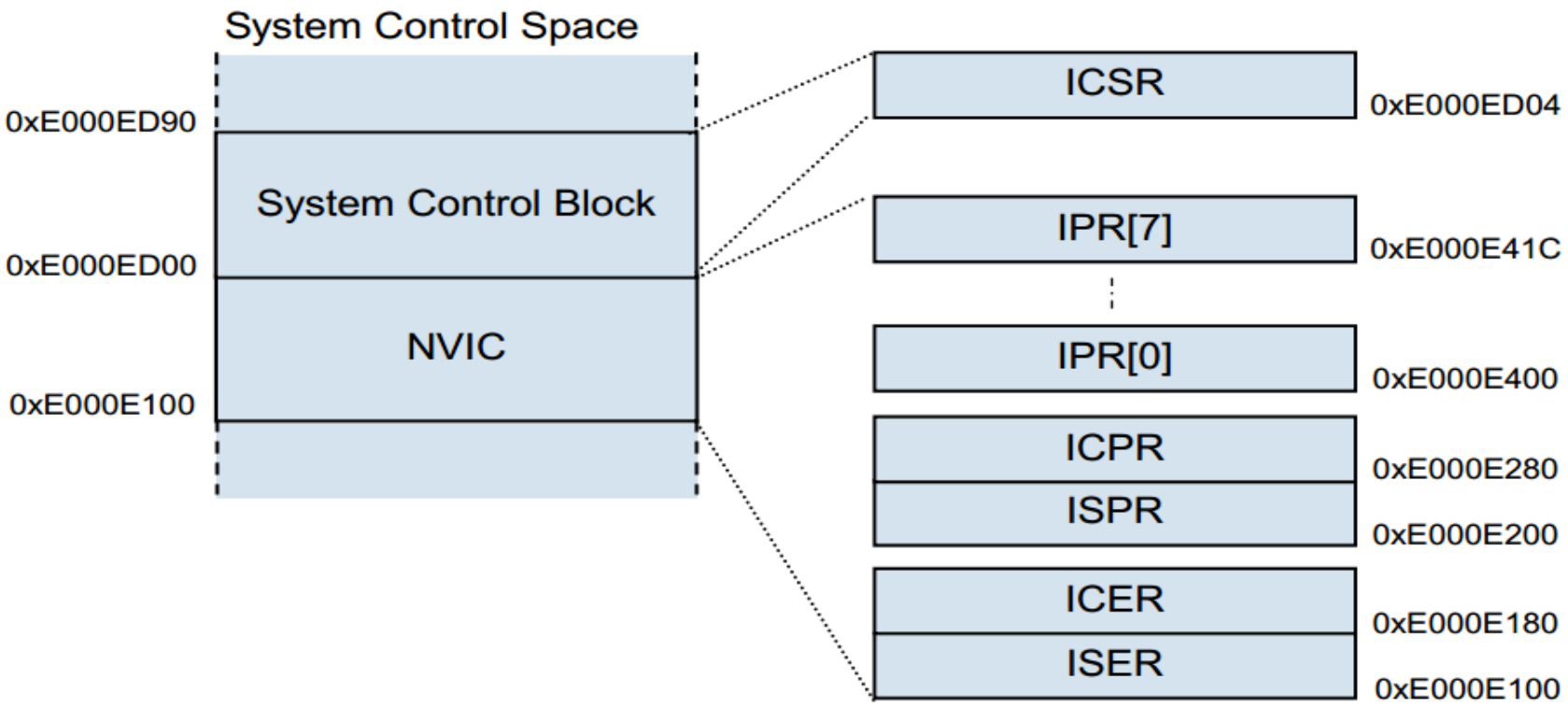
Vector Table Implementation (Startup.s & linker file):

```
; Vector Table Mapped to Address 0 at Reset

                AREA    RESET, DATA, READONLY
                EXPORT  __Vectors
                EXPORT  __Vectors_End
                EXPORT  __Vectors_Size

__Vectors       DCD     __initial_sp   ; Top of Stack
                DCD     Reset_Handler  ; Reset Handler
                DCD     NMI_Handler    ; NMI Handler
                DCD     HardFault_Handler   ; Hard Fault Handler
                DCD     0   ; Reserved
                DCD     0   ; Reserved
                DCD     0   ; Reserved
                DCD     0   ; Reserved
```

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************

LR_IROM1 0x00000000 0x00020000  {    ; load region size_region
  ER_IROM1 0x00000000 0x00020000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
```

Table of contents

Details of NVIC & SCB registers



SCB & NVIC Register Set for Cortex M0

Details of NVIC & SCB registers

NVIC registers

- ***Interrupt Priority Level Register (IPR0 – IPR7):***

Address	Name	Type	Reset Value	Descriptions
0xE000E400	PRIORITY0	R/W	0x00000000	Priority level for interrupt 0 to 3 [31:30] Interrupt priority 3 [23:22] Interrupt priority 2 [15:14] Interrupt priority 1 [7:6] Interrupt priority 0

Address	Name	Type	Reset Value	Descriptions
0xE000E404	PRIORITY1	R/W	0x00000000	Priority level for interrupt 4 to 7
0xE000E408	PRIORITY2	R/W	0x00000000	Priority level for interrupt 8 to 11
0xE000E40C	PRIORITY3	R/W	0x00000000	Priority level for interrupt 12 to 15
0xE000E410	PRIORITY4	R/W	0x00000000	Priority level for interrupt 16 to 19
0xE000E414	PRIORITY5	R/W	0x00000000	Priority level for interrupt 20 to 23
0xE000E418	PRIORITY6	R/W	0x00000000	Priority level for interrupt 24 to 27
0xE000E41C	PRIORITY7	R/W	0x00000000	Priority level for interrupt 28 to 31

Details of NVIC & SCB registers

NVIC registers

- ***Interrupt Clear Pending Register***

Address	Name	Type	Reset Value	Descriptions
0xE000E280	CLRPEND	R/W	0x00000000	Clear pending for interrupt 0 to 31; write 1 to clear bit to 0, write 0 has no effect Bit[0] for Interrupt #0 (exception #16) … Bit[31] for Interrupt #31 (exception #47) Read value indicates the current pending status

- ***Interrupt Set Pending Register***

Address	Name	Type	Reset Value	Descriptions
0xE000E200	SETPEND	R/W	0x00000000	Set pending for Interrupts 0 to 31; write 1 to set bit to 1, write 0 has no effect. Bit[0] for Interrupt #0 (exception #16) Bit[1] for Interrupt #1 (exception #17) … Bit[31] for Interrupt #31 (exception #47) Read value indicates the current pending status

Details of NVIC & SCB registers

NVIC registers

- ***Interrupt Set Enable Register***

Address	Name	Type	Reset Value	Descriptions
0xE000E100	SETENA	R/W	0x00000000	Set enable for Interrupts 0 to 31; write 1 to set bit to 1, write 0 has no effect Bit[0] for Interrupt #0 (exception #16) Bit[1] for Interrupt #1 (exception #17) … Bit[31] for Interrupt #31 (exception #47) Read value indicates the current enable status

- ***Interrupt Clear Enable Register***

Address	Name	Type	Reset Value	Descriptions
0xE000E180	CLRENA	R/W	0x00000000	Clear enable for Interrupts 0 to 31;write 1 to clear bit to 0, write 0 has no effect Bit[0] for Interrupt #0 (exception #16) …

Details of NVIC & SCB registers

SCB registers

- ***Interrupt control and state register***
  - Set and clear the pending status of system exceptions including SysTick, PendSV, and NMI.
  - Determine the currently executing exception/interrupt number by reading VECTACTIVE
- ***Vector table offset Register***

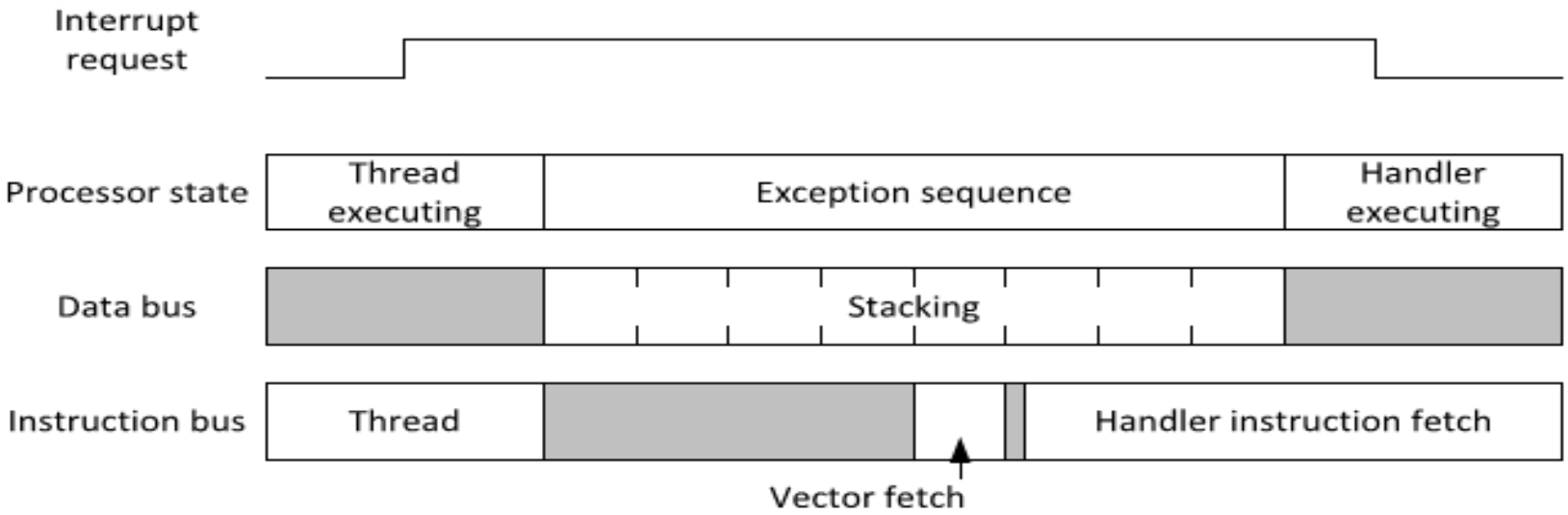
Bits	Name	Type	Reset Value	Description
31:30	**Reserved**	**-**	**-**	**Not implemented, tied to 0**
29	TBLBASE	R/W	0	Table base in Code (0) or RAM (1)
28:7	TBLOFF	R/W	0	Table offset value from Code region or RAM region

Table of contents

Exception Sequence

An exception entrance sequence contains:

1. Stacking of a number of registers
2. Fetching the exception vector
3. Fetching the instructions for the exception handler to be executed.
4. Update NVIC and core registers



Exception Sequence

Exception handler execution:

1. The processor is in Handler mode

2. Nested Interrupt: If a higher-priority exception arrives, the currently executing handler will be suspended and pre-empted

Exception Sequence

Exception return:

Instructions that can be Used for Triggering Exception Return	
Return Instruction	**Description**
BX <*reg*>	If the EXC_RETURN value is still in the LR when the exception handler ends, we can use the "*BX LR*" instruction to perform the exception return.
POP {PC}, or POP {....., PC}	Very often the value of LR is pushed to the stack after entering the exception handler. We can use the POP instruction, either a single register POP or a POP operation with multiple registers including the PC, to put the EXC_RETURN value to the Program Counter (PC). This will cause the processor to perform the exception return.
Load (LDR) or Load multiple (LDM)	It is possible to produce an exception return using the LDR or LDM instructions with PC as the destination register.

Exception Sequence

Exception return:

- Unstacking: Restore Registers in the stack memory
- Update NVIC & core Registers
- Fetch the instructions of the previously interrupted program.

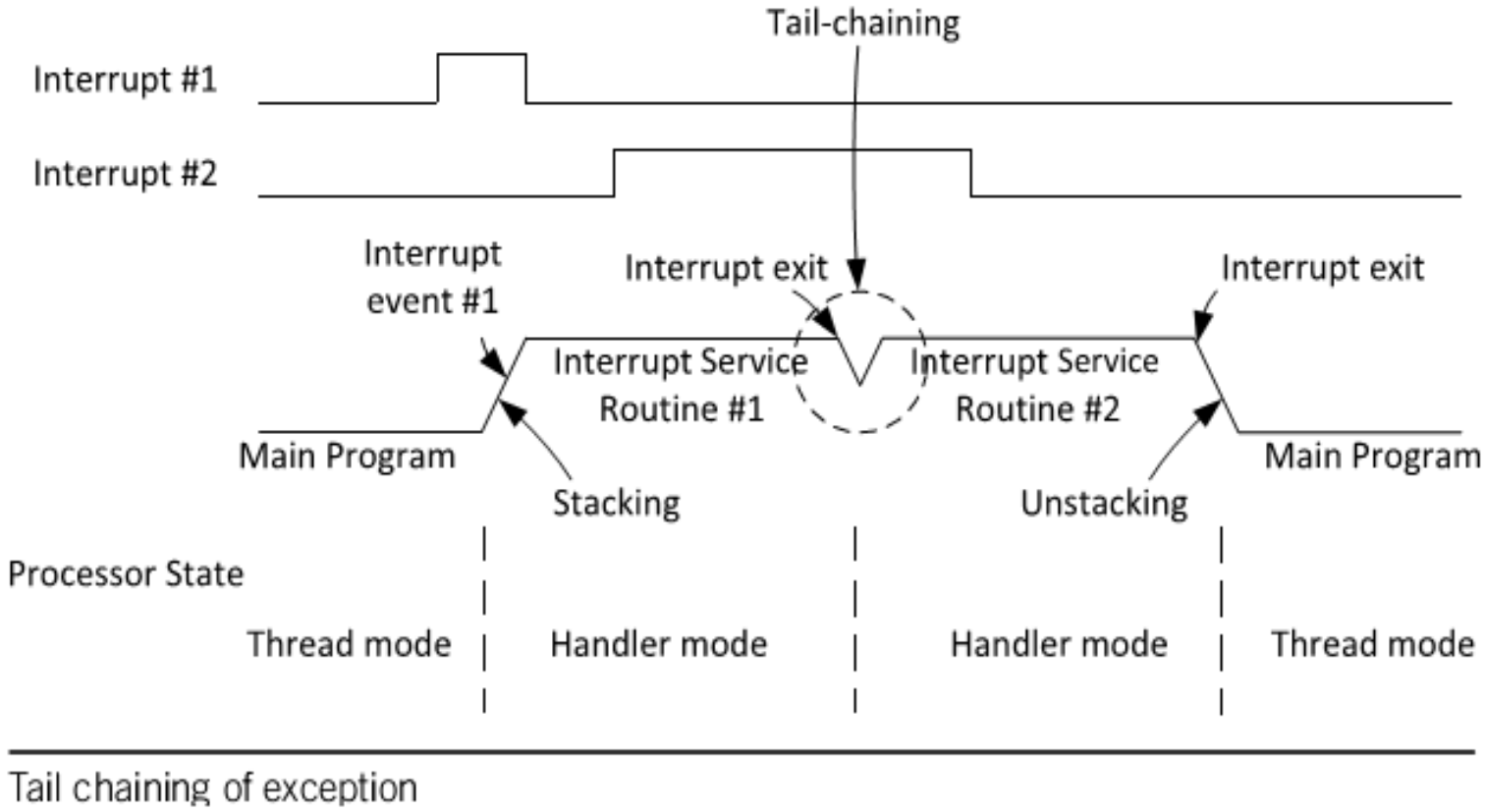
Table of contents

Exception handling optimization

- Tail chaining
- Late arrival
- Pop preemption
- Lazy stacking (Only support for Cortex M4 with FPT)

Exception handling optimization

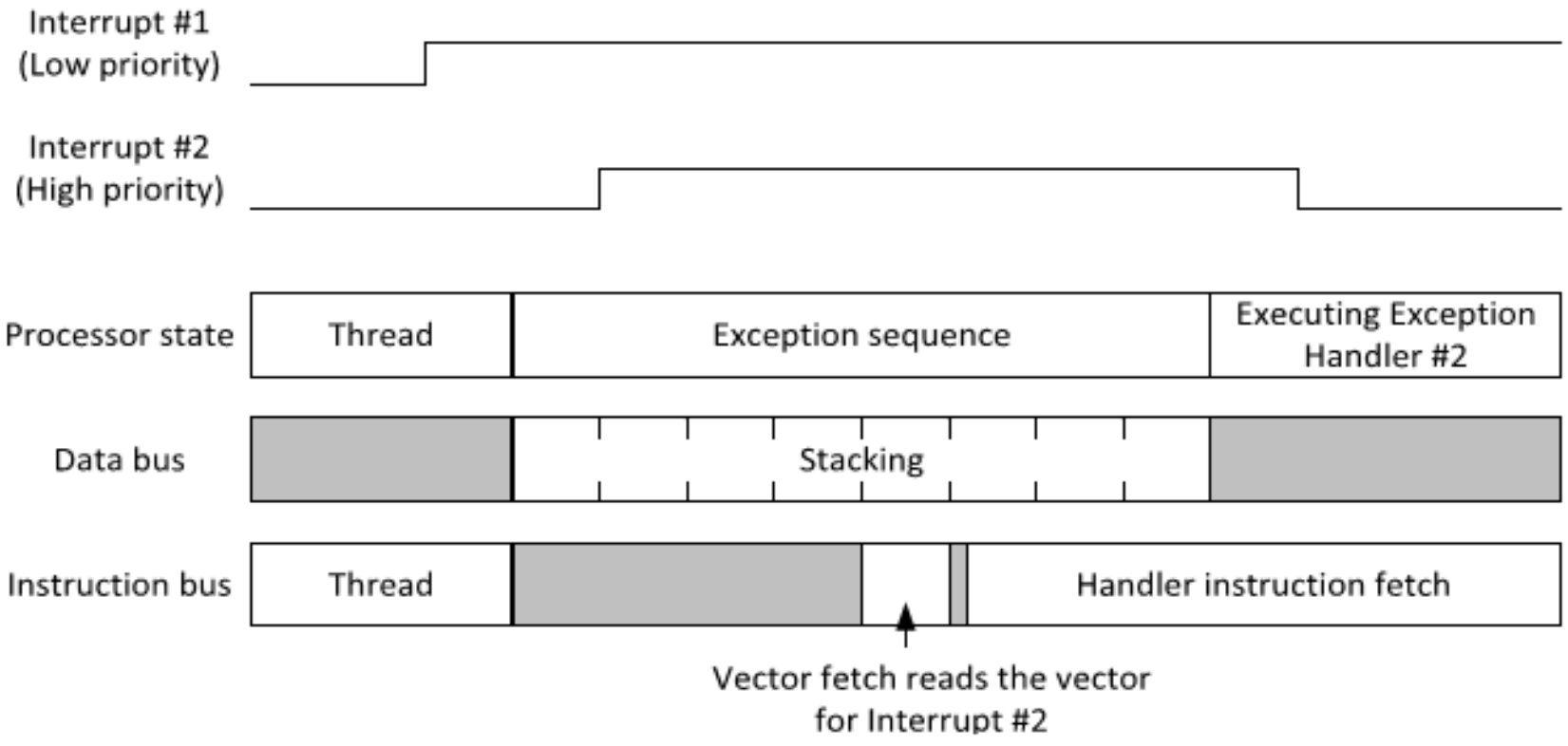
Tail chaining:



Tail chaining of exception

Exception handling optimization

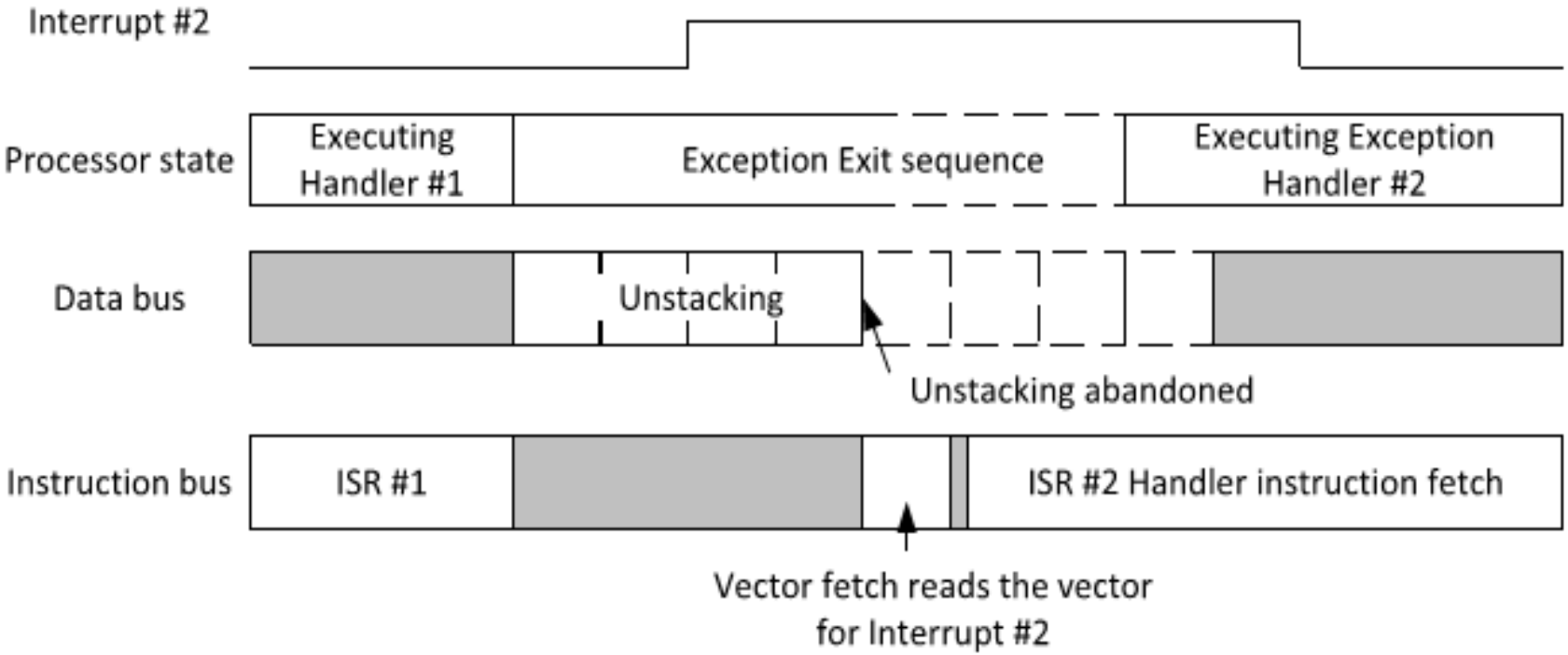
Late arrival:



Late arrival exception behavior

Exception handling optimization

Pop preemption:



Pop pre-emption behavior

Table of contents

Summary

- Cortex-M provides a Nested Vectored Interrupt Controller (NVIC) for interrupt handling.
- Exception entry sequence: Staking, Update registers, Fetch vector number & instruction of exception handler.
- Exception return sequence: Unstacking, Fetch & execute from the restored return address
- Exception handling optimization: Tail chaining, Late arrival, pop preemption.

Question & Answer

Thanks for attention !

Copyright

- This course including **Lecture Presentations**, **Quiz**, **Mock Project**, **Syllabus**, **Assignments**, **Answers** are copyright by FPT Software Corporation.
- This course also uses some information from external sources and non-confidential training document from Freescale, those materials comply with the original source licenses.