Курс по STM32

Лекция #6:

- Обработка исключений в Cortex-M0!
- Обработка исключений в Cortex-M0!!!
- Обработка исключений в Cortex-M0!!!!!
- Системный таймер в Cortex-M0.

11.03.2023 / 13.03.2023

Исключения и прерывания в Cortex-M0

Что такое прерывание?

1) Периферия генерирует запрос на прерывание – оповещает процессор.
2) Процессор сохраняет текущее состояние.
3) Процессор определяет адрес обработчика прерывания и исполняет его.
4) Процессор возвращается в исходное состояние.

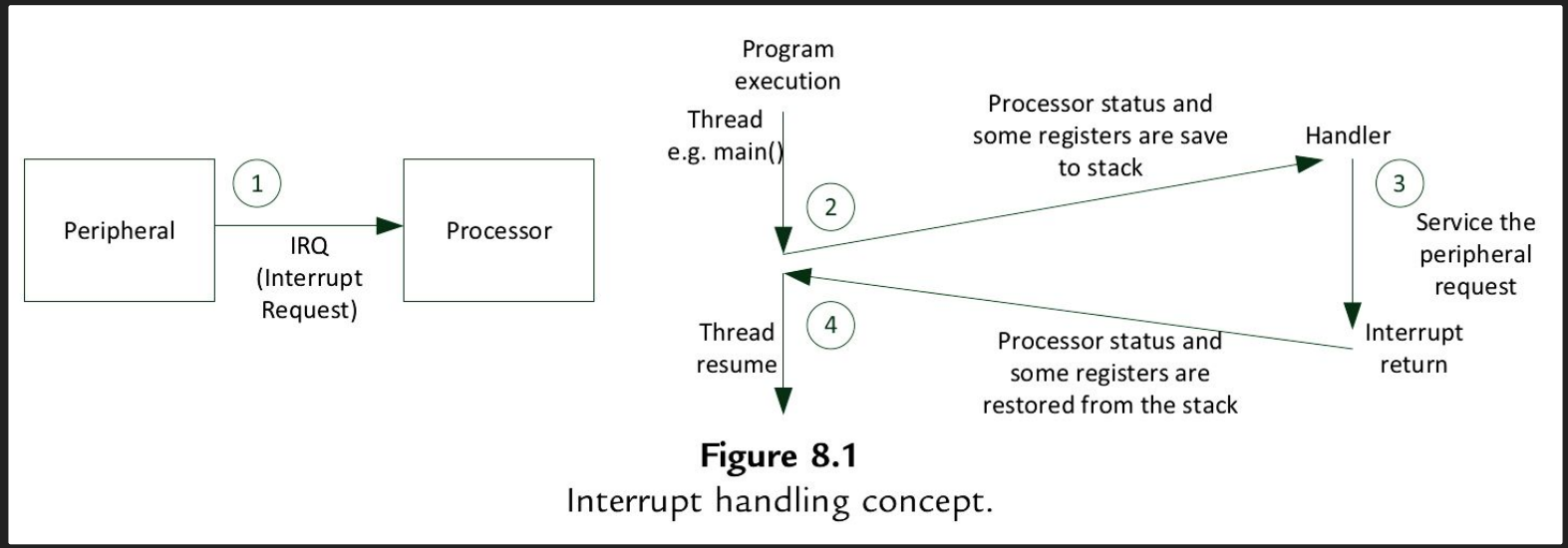


Figure 8.1
Interrupt handling concept.

(1) Контроллер прерываний

NVIC - Nested Vectored Interrupt Controller.

Функции NVIC:
- Включение/выключение отдельных прерываний.
- Приоритеты прерываний.
- Ручной вызов прерывания.

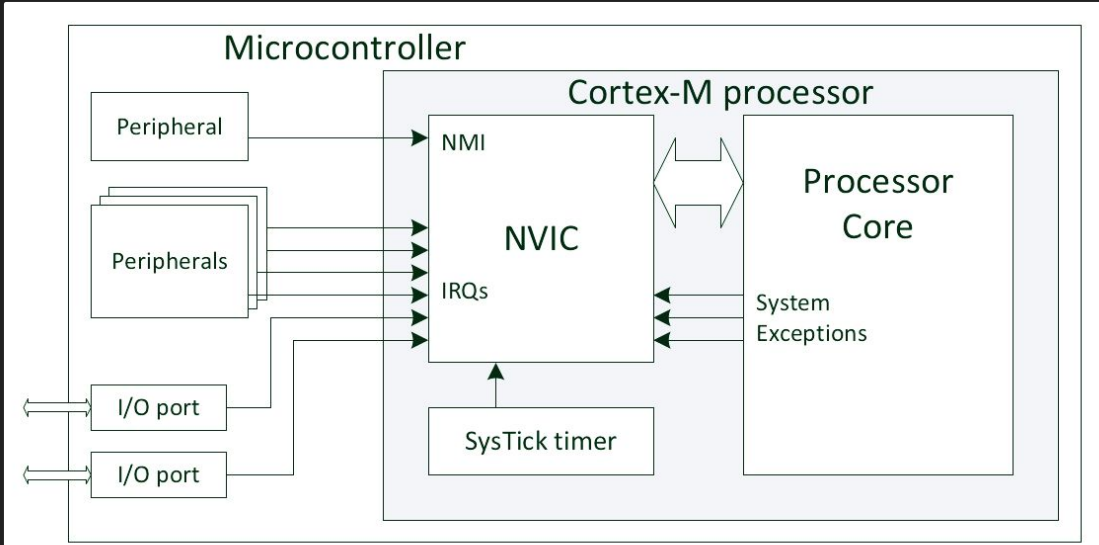


Figure 8.2

(1) Прерывания != исключения

Прерывание (interrupt) != исключение (exception)

Table 8.1: List of exceptions in the Cortex-M0 and Cortex-M0+ processors

Exception number	Exception type	Priority	Descriptions
1	Reset	-3 (Highest)	Reset
2	NMI	-2	Non-Maskable Interrupt
3	HardFault	-1	Fault handling exception
4–10	Reserved	NA	–
11	SVCall	Programmable	Supervisor call via SVC instruction
12–13	Reserved	NA	–
14	PendSV	Programmable	Pendable request for system service
15	SysTick	Programmable	System Tick Timer
16	Interrupt #0	Programmable	External Interrupt #0
17	Interrupt #1	Programmable	External Interrupt #1
…	…	…	…
47	Interrupt #31	Programmable	External Interrupt #31

(1) Какие бывают исключения?

Исключения в Cortex-M0:

- `Reset` – сброс микросхемы в начальное состояние.
- `Non-Maskable Interrupt` – беда: отказ питания, сторожевой таймер.
- `HardFault` – некорректная инструкция или адрес, …
- `SVCall` – Системный вызов, инструкция `SVC`.
- `PendSV` – отложенная системная процедура.
- `SysTick` – срабатывание системного таймера.
- `IRQ` – прерывания.

В Cortex-M4 более подробная обработка ошибок:

- `HardFault + BusFault, MemoryManagementFault, UsageFault.`

(1) Какие бывают прерывания?

Table 36. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
3	10	settable	FLASH	Flash global interrupt	0x0000 004C
4	11	settable	RCC_CRS	RCC and CRS global interrupts	0x0000 0050
5	12	settable	EXTI0_1	EXTI Line[1:0] interrupts	0x0000 0054
6	13	settable	EXTI2_3	EXTI Line[3:2] interrupts	0x0000 0058
7	14	settable	EXTI4_15	EXTI Line[15:4] interrupts	0x0000 005C
8	15	settable	TSC	Touch sensing interrupt	0x0000 0060
9	16	settable	DMA_CH1	DMA channel 1 interrupt	0x0000 0064
10	17	settable	DMA_CH2_3 DMA2_CH1_2	DMA channel 2 and 3 interrupts DMA2 channel 1 and 2 interrupts	0x0000 0068
11	18	settable	DMA_CH4_5_6_7 DMA2_CH3_4_5	DMA channel 4, 5, 6 and 7 interrupts DMA2 channel 3, 4 and 5 interrupts	0x0000 006C
12	19	settable	ADC_COMP	ADC and COMP interrupts (ADC interrupt combined with EXTI lines 21 and 22)	0x0000 0070

(1) Приоритеты исключений в Cortex-M0

Приоритеты контролируют:

- Очерёдность исполнения.
- Возможность вложенности.

Приоритеты бывают:

- Закреплённые.
- Устанавливаемые.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Implemented		Not implemented, read as zero					

Figure 8.3
A Priority Level Register with 2 bits implemented.

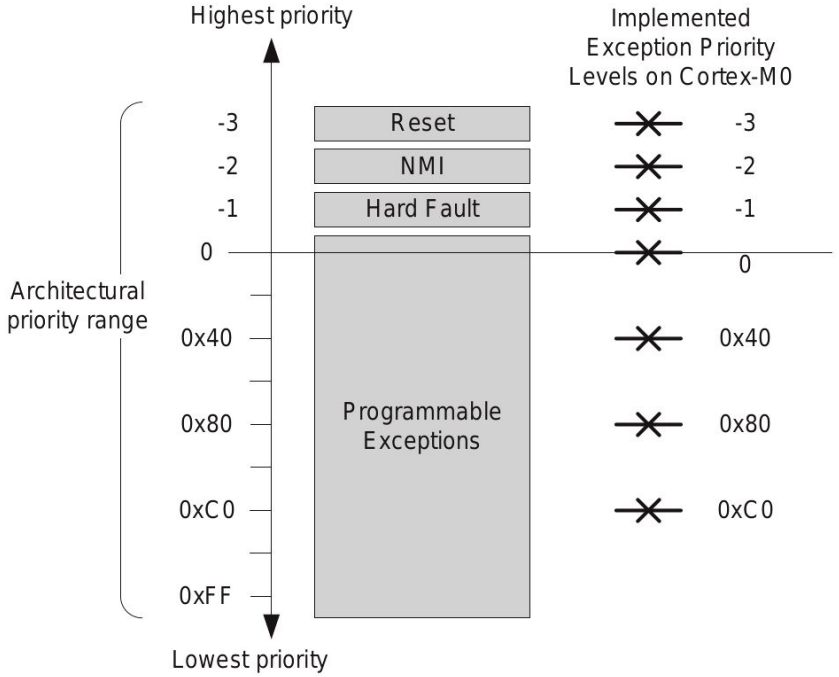


Figure 8.4
Available priority levels in the Cortex®-M0 and Cortex-M0+ Processors.

(2) Сохранение регистров на стеке

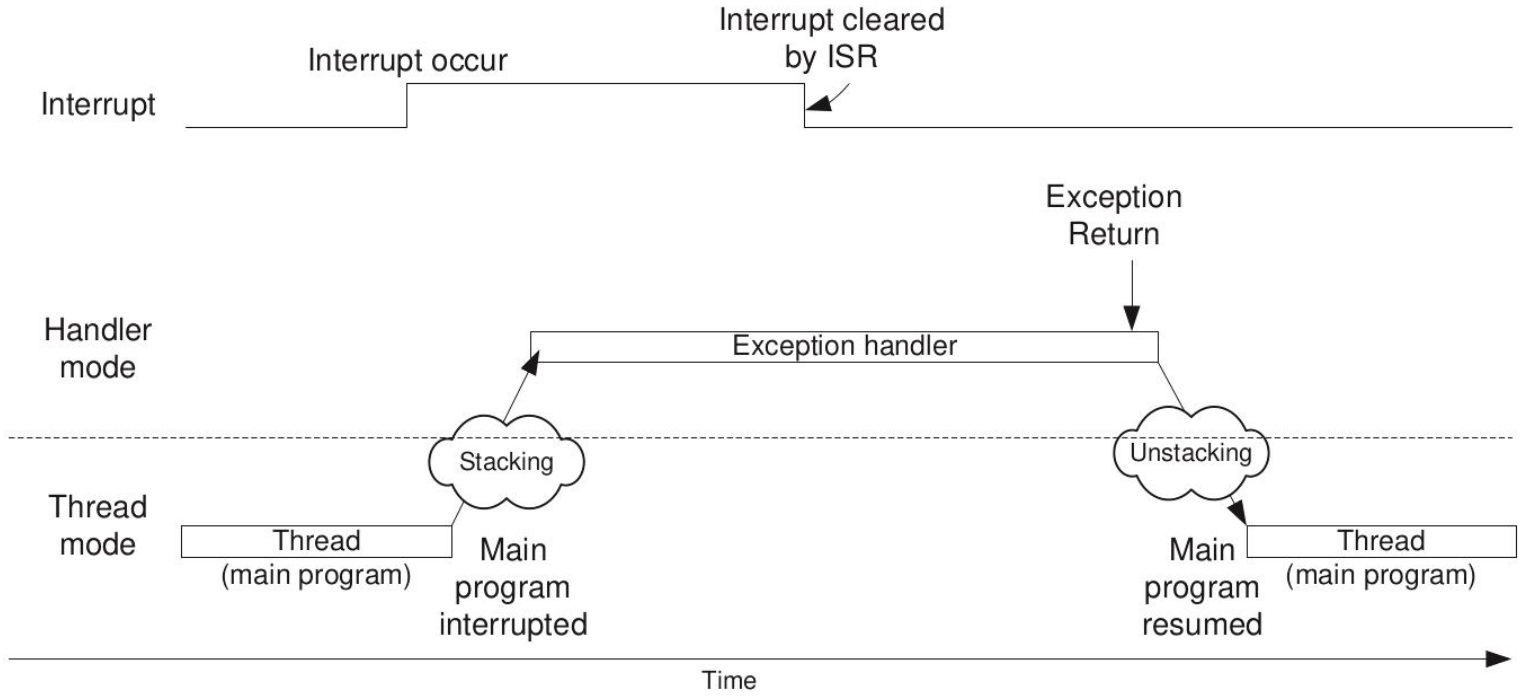


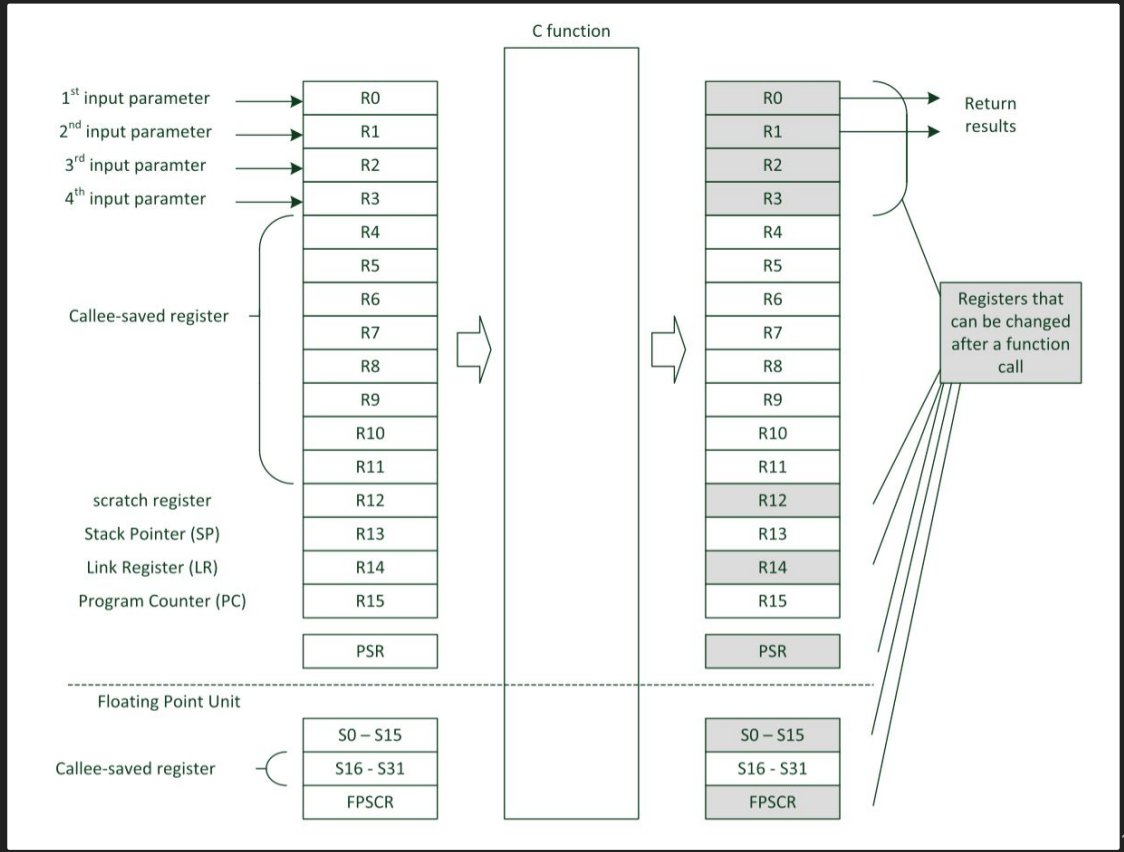
Figure 8.6

Stacking and unstacking of registers at exception entry and exit.

(2) Сохранение регистров на стеке

Stacking работает
как вызов функции
по calling convention!

Обработчиком исключения
может быть обычная
функция, написанная на си!



(2) Сохранение регистров на стеке

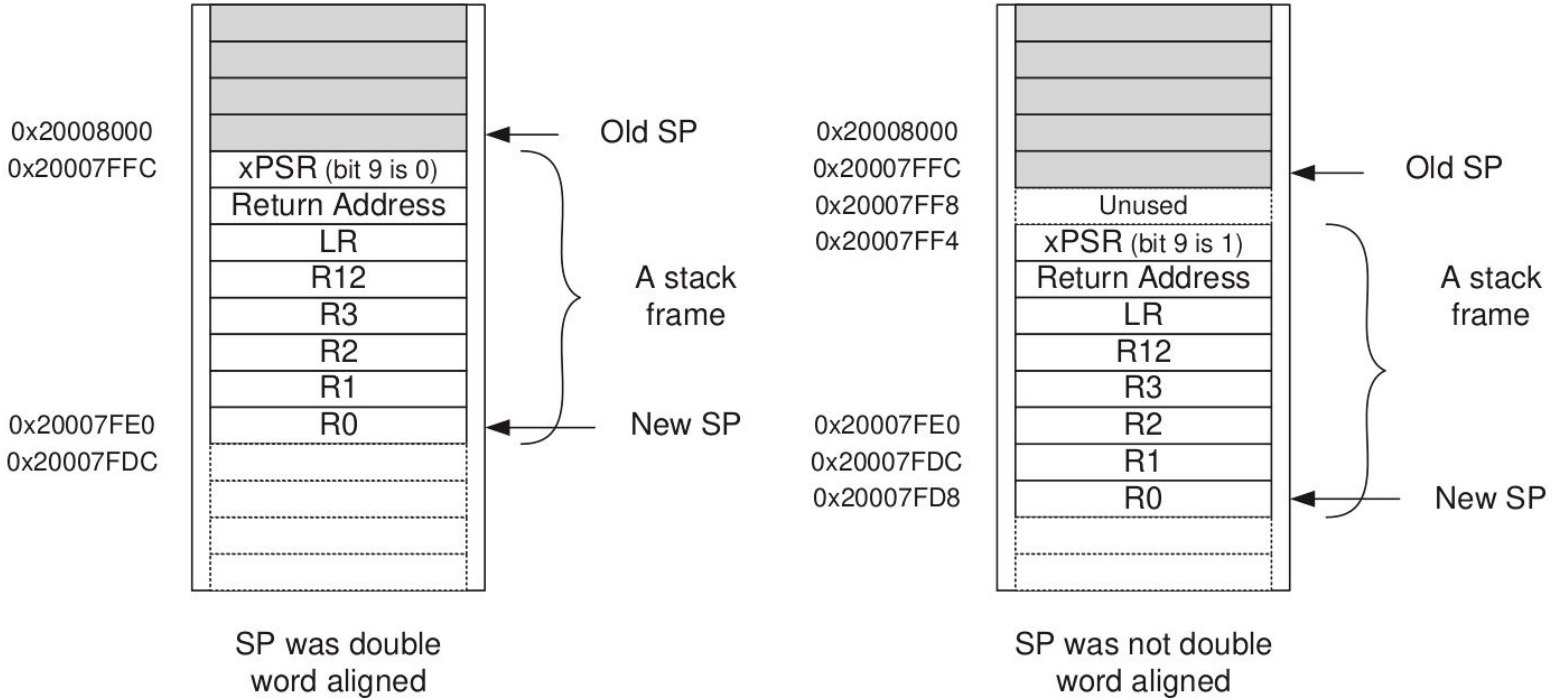
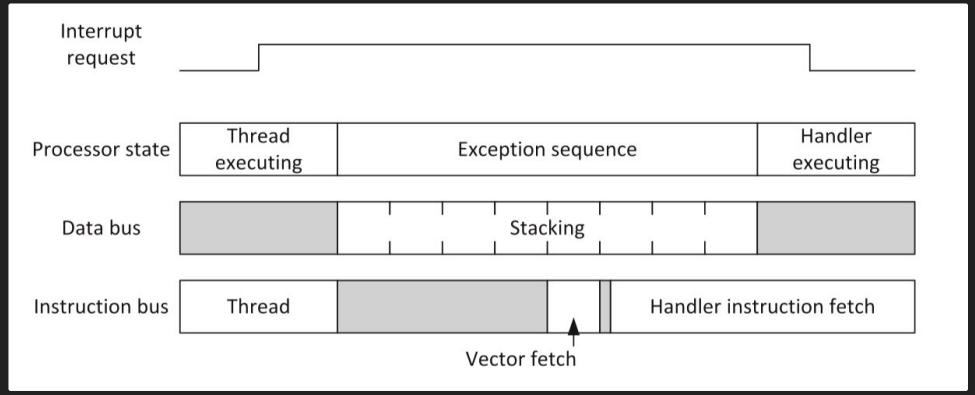


Figure 8.21
Stack frame and double word stack alignment.

(2) Поиск обработчика исключения




```
.section .vector_table
.word __stack_start          // Initial SP
.word __reset_handler        // Reset Handler
.word __exc_handler          // NMI Handler
.word __exc_handler          // Hard Fault Handler
.fill 7, 4, 0x00             // Reserved
.word __exc_handler          // SVCall
.fill 2, 4, 0x00             // Reserved
.word __exc_handler          // PendSV
.word systick_handler        // SysTick
```

0x0000003C	SysTick vector	15
0x00000038	PendSV vector	14
0x00000034	Not used	13
0x00000030	Not used	12
0x0000002C	SVC vector	11
0x00000028	Not used	10
0x00000024	Not used	9
0x00000020	Not used	8
0x0000001C	Not used	7
0x00000018	Not used	6
0x00000014	Not used	5
0x00000010	Not used	4
0x0000000C	HardFault vector	3
0x00000008	NMI vector	2
0x00000004	Reset vector	1
0x00000000	MSP initial value	0

Figure 8.5
Vector table.

(2) Оптимизация stacking-a: tail-chaining

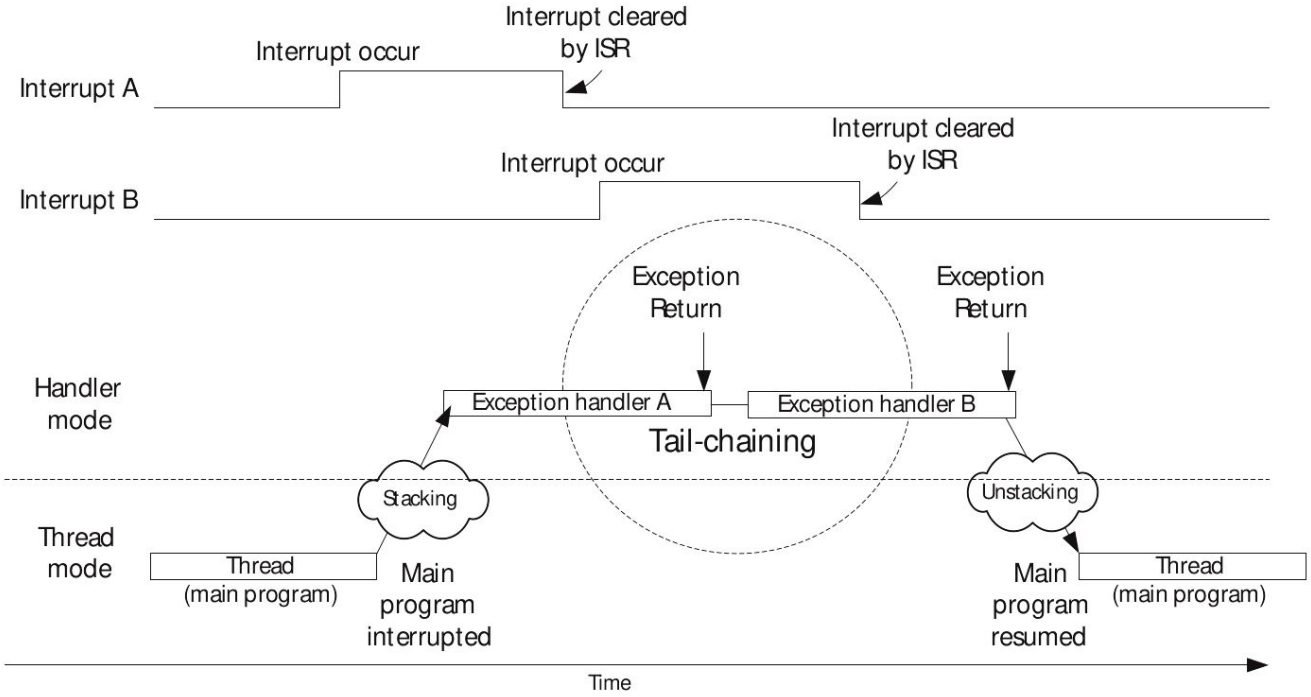


Figure 8.7
Tail chaining of interrupt service routines.

(2) Оптимизация stacking-a: late arrival

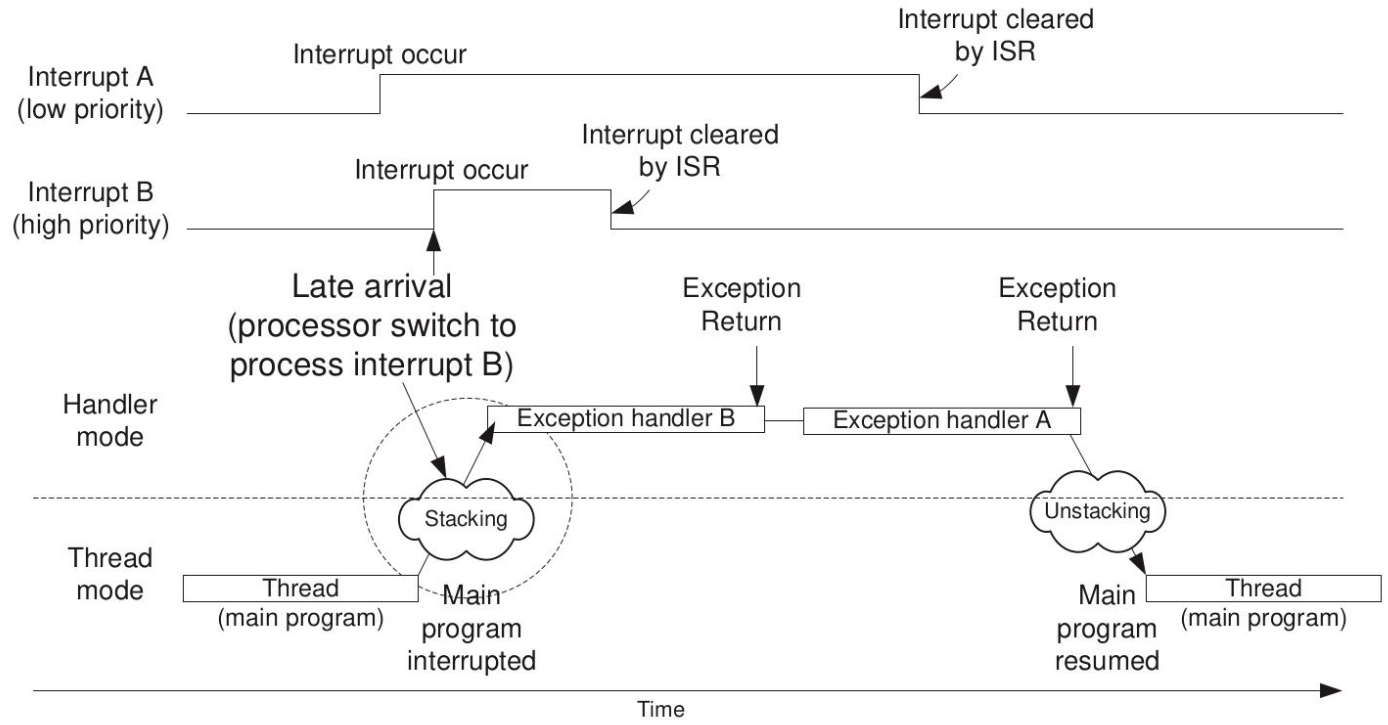


Figure 8.8
Late arrival optimization.

(3) Очистка бита IRQ Pending

Иногда бит “`Interrupt Pending Status X`” надо очищать вручную!

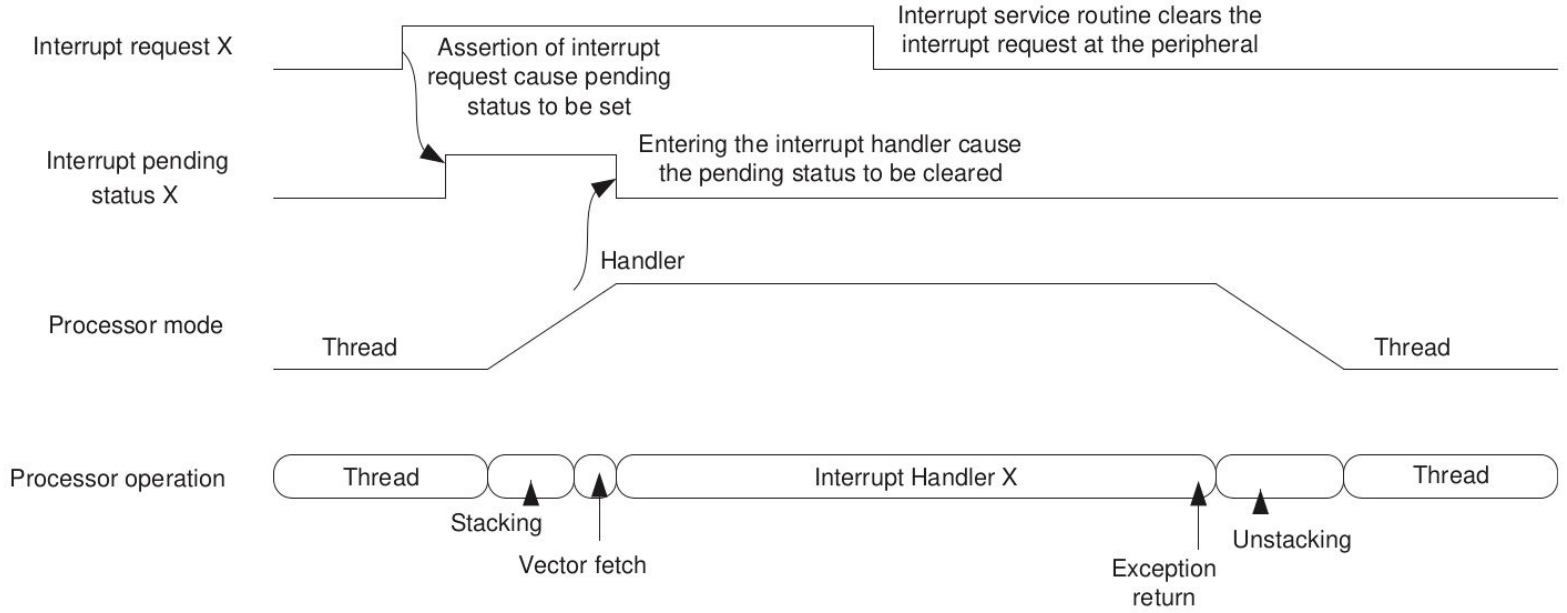


Figure 8.12
Simple case of interrupt activation and pending status behavior.

(3) Стек в прерываниях: тот же стек

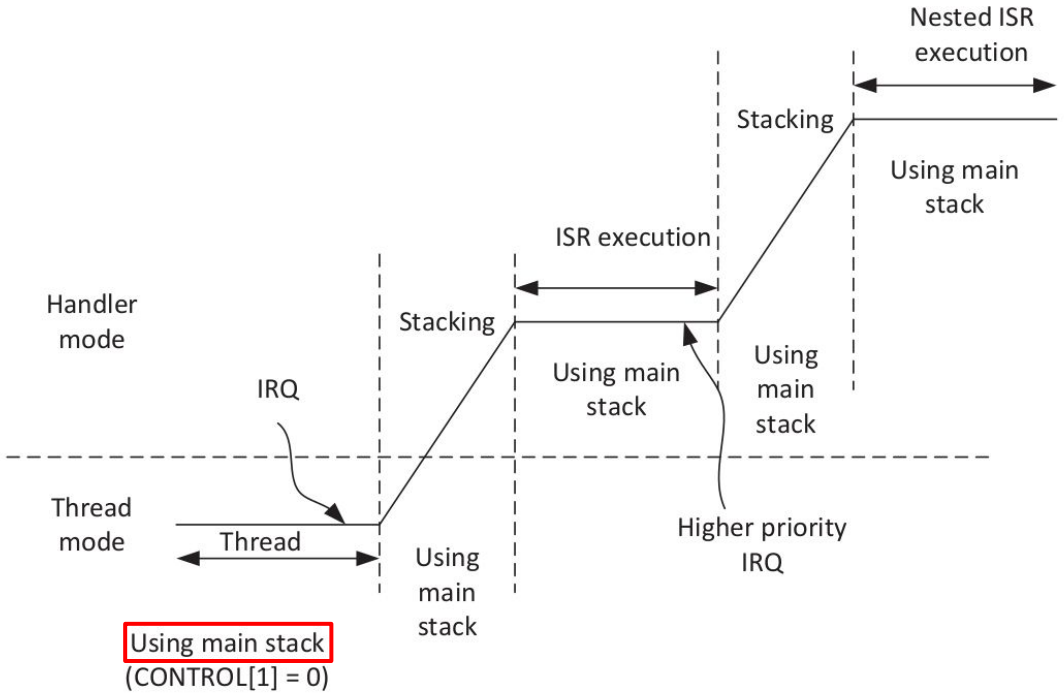


Figure 8.19

Exception stacking in nested interrupt with main stack used in the Thread mode.

(3) Стек в прерываниях: другой стек

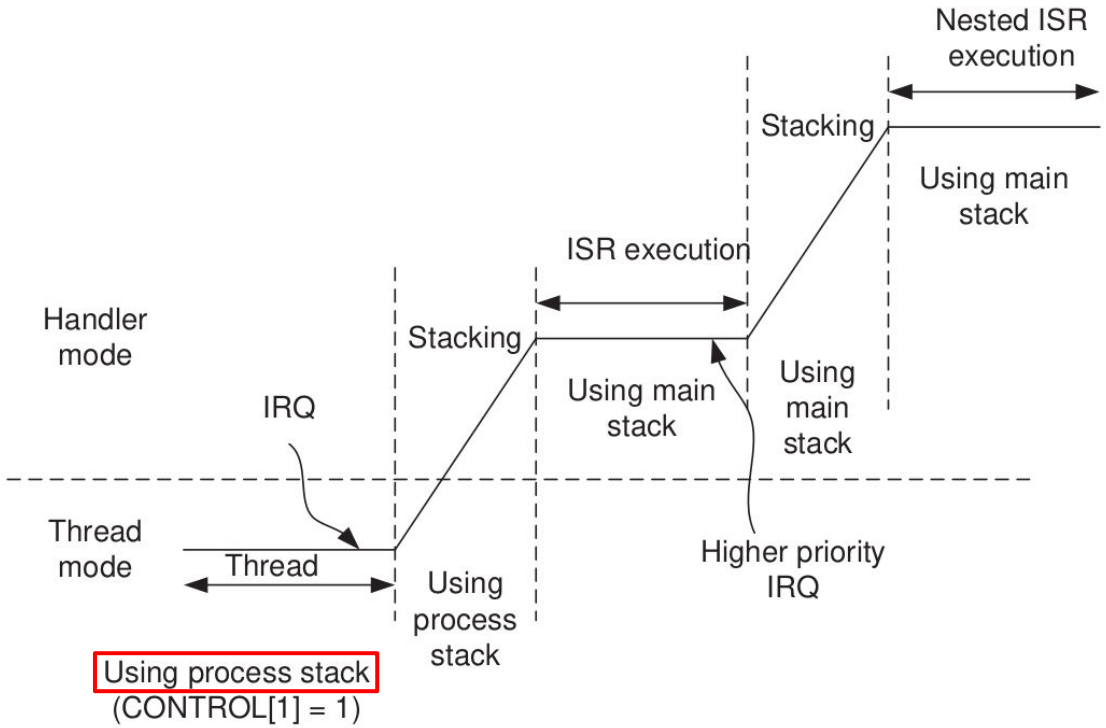


Figure 8.20

Exception stacking in nested interrupt with process stack used in the Thread mode.

(3) Стек в прерываниях: сравнение подходов

Тот же стек:

- Простота системы.
- Взаимозависимость “ОС” и ПО.

Разные стеки:

- Сложности при разработке.
- Независимость (почти) ОС и ПО.
- Возможность смены стека ПО.

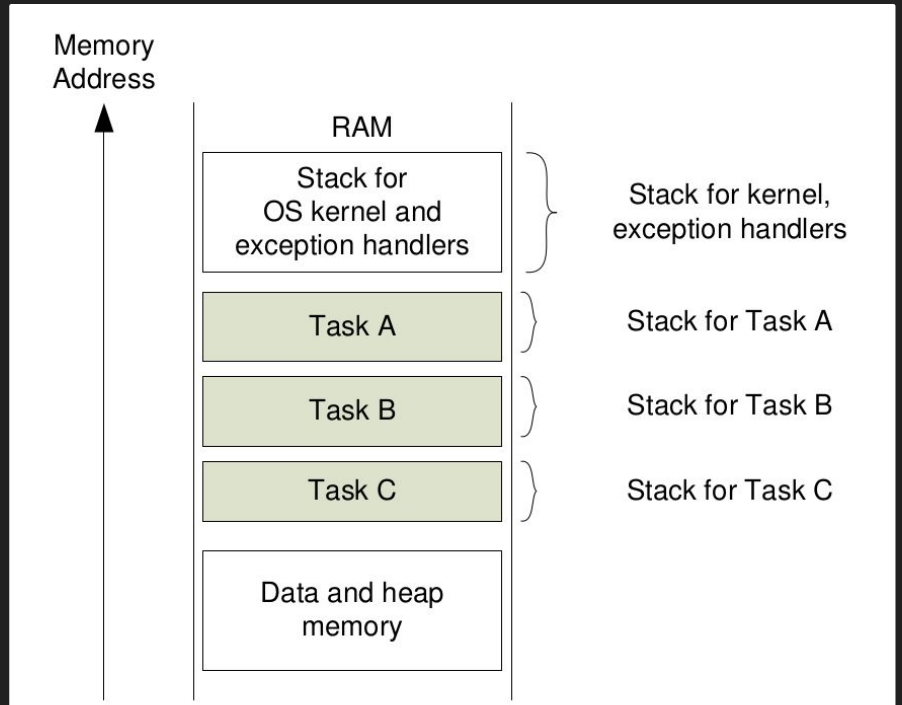


Figure 10.6
Separate memory ranges for OS and application tasks.

(3) Программное отключение исключений

```
; Запрет конфигурируемых исключений (PRIMASK = 1):
cpsid i
; Включение исключений (PRIMASK = 0):
cpsie i
```

Table 2-7 PRIMASK register bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	PRIMASK	0 = no effect 1 = prevents the activation of all exceptions with configurable priority.

(4) Выход из исключения: совместимость с си

```
; Вход в функцию:
push {r7, lr} ; lr = ???
; Выход из функции:
pop  {r7, pc}
```

Table 8.3: Valid EXC_RETURN values for the Cortex-M0 and Cortex-M0+ processors

EXC_RETURN	Condition
0xFFFFFFF1	Return to handler mode (nested exception case)
0xFFFFFFF9	Return to Thread mode and use the main stack for return
0xFFFFFFFD	Return to Thread mode and use the process stack for return

Table 8.2: Bit fields in the EXC_RETURN value

Bits	31:28	27:4	3	2	1	0
Descriptions	EXC_RETURN indicator	Reserved	Return mode	Return stack	Reserved	Processor state
Value	0xF	0xFFFFFF	1 (thread) or 0 (handler)	0 (main stack) or 1 (process stack)	0	1 (reserved)

(4) Восстановление регистров со стека

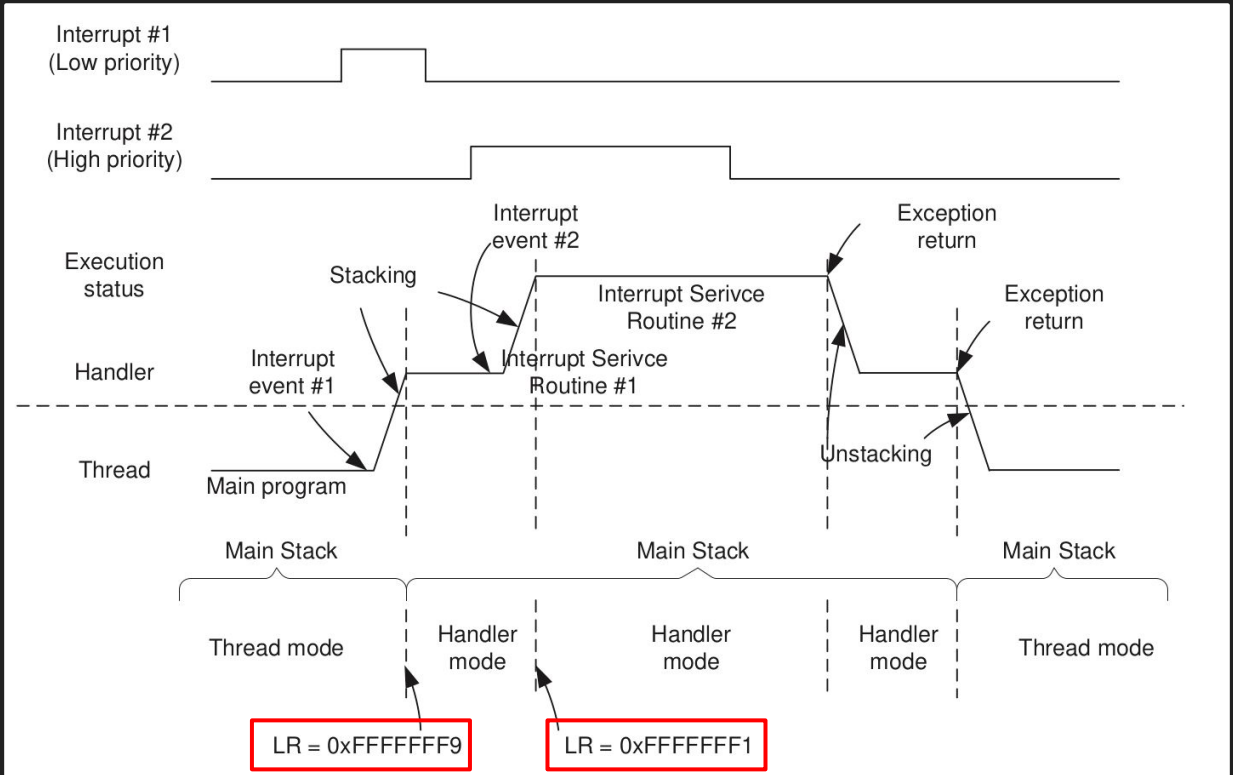
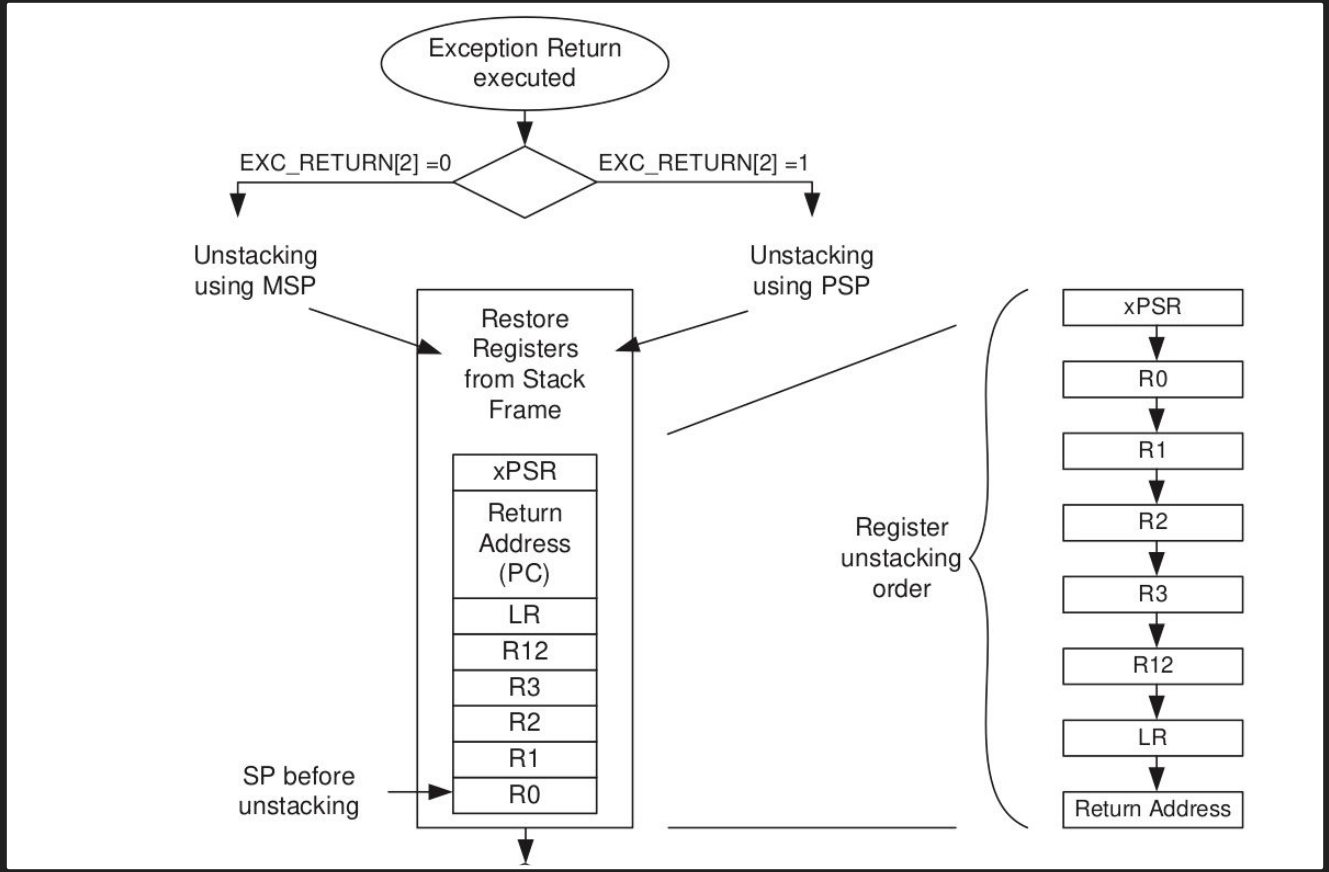


Figure 8.9

LR set to EXC_RETURN values at exceptions (main stack is used in Thread mode).

(4) Восстановление регистров со стека



(4) Восстановление регистров со стека

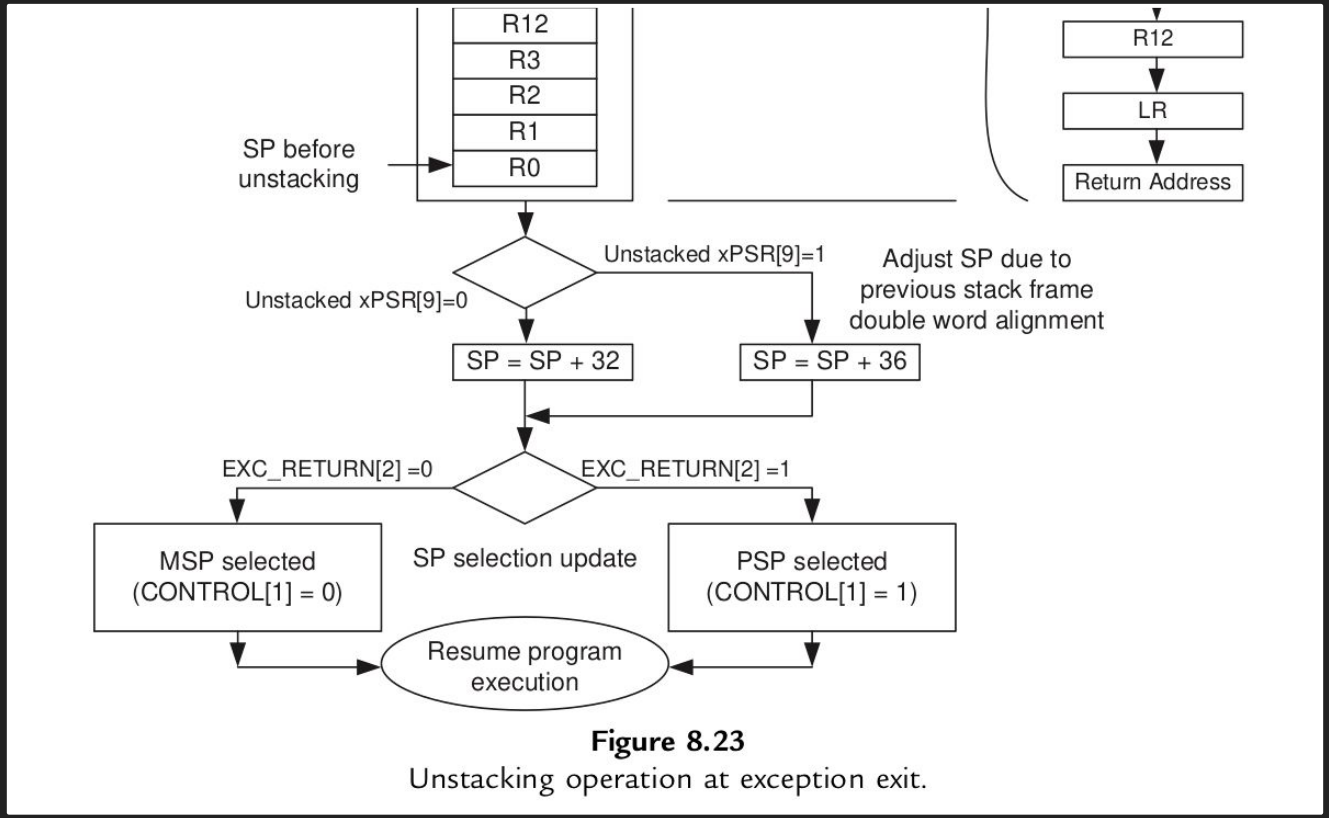
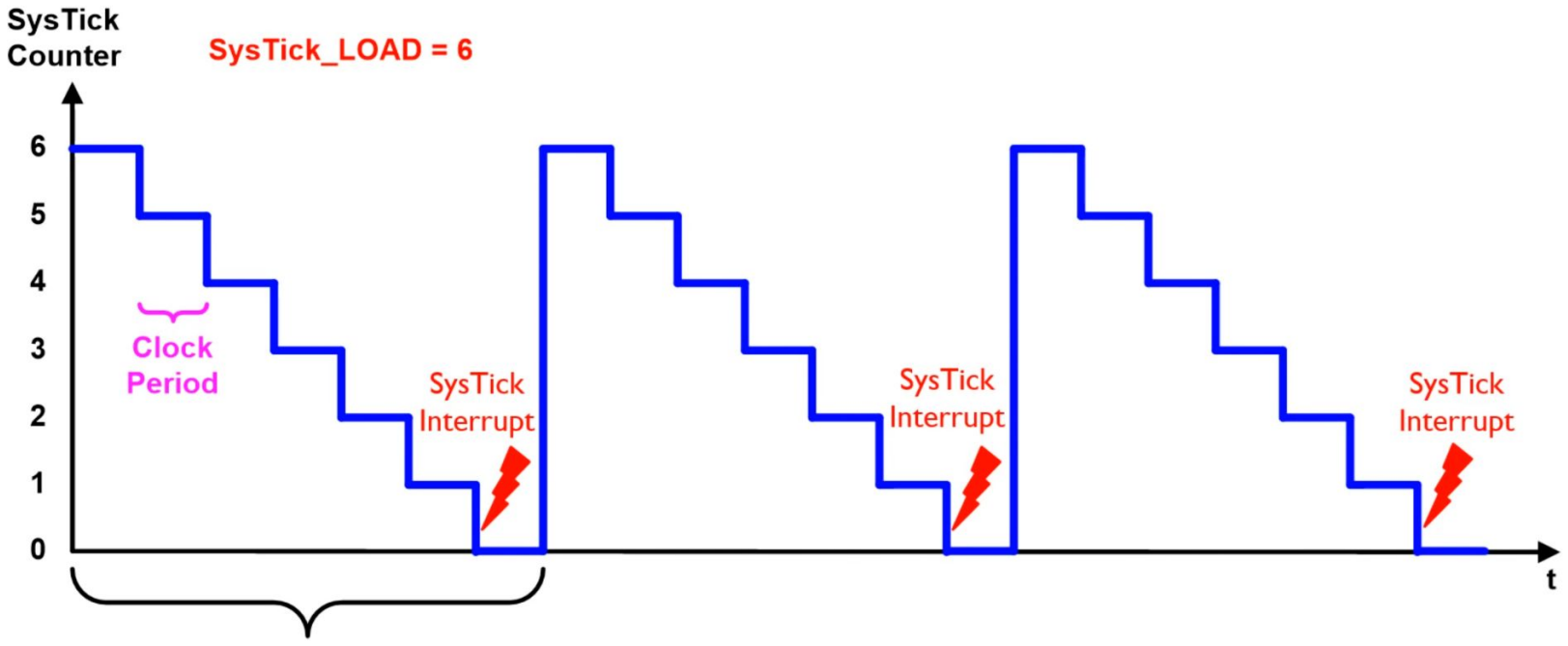


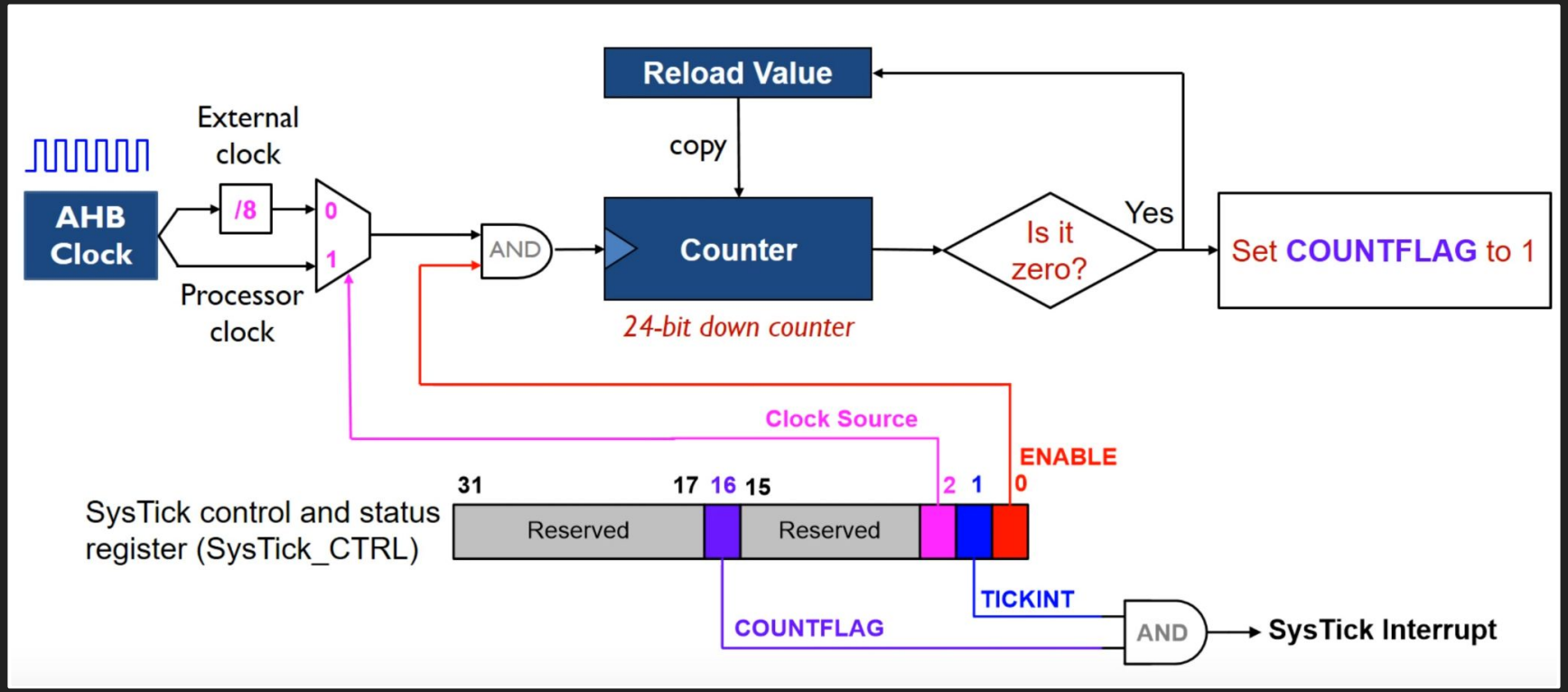
Figure 8.23
Unstacking operation at exception exit.

Системный таймер в Cortex-M0

Системный таймер: схема работы



Системный таймер: устройство



Эта и другие схемы взяты из [курса по МК от University of Maine](#).

Системный таймер: регистры

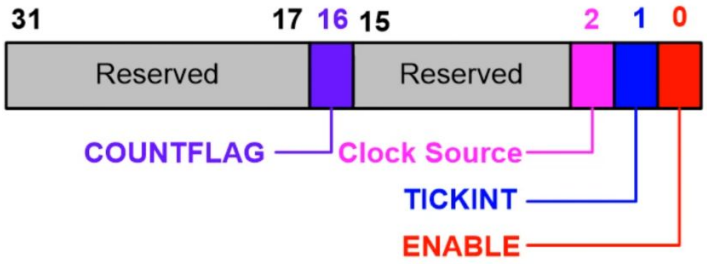
Описание системного таймера можно найти в документации на Cortex-M0 (docs/cortex_m0_gug.pdf).

Table 4-19 System timer registers summary

Address	Name	Type	Reset value	Description
0xE000E010	SYST_CSR	RW	-[a]	*SysTick Control and Status Register* on page 4-22
0xE000E014	SYST_RVR	RW	Unknown	*SysTick Reload Value Register* on page 4-23
0xE000E018	SYST_CVR	RW	Unknown	*SysTick Current Value Register* on page 4-23
0xE000E01C	SYST_CALIB	RO	-[a]	*SysTick Calibration Value Register* on page 4-24

Системный таймер: регистры

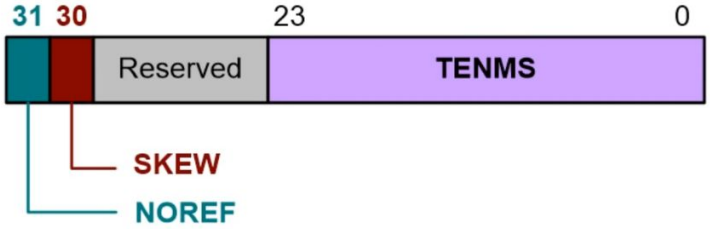
SysTick control and status register (SysTick_CTRL)



SysTick reload value register (SysTick_LOAD)

SysTick current value register (SysTick_VAL)

SysTick calibration register (SysTick_CALIB)



Спасибо за внимание!

Красивые иконки взяты с сайта handdrawngoods.com