EMBEDDED SYSTEMS

BASED ON CORTEX-M4 AND THE RENESAS SYNERGY PLATFORM

2020
PROF. DOUGLAS RENAUX, PHD
PROF. ROBSON LINHARES, DR.
UTFPR / ESYSTECH

RENESAS ELECTRONICS CORPORATION

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.3 – CORTEX-M4 ISA – REGISTERS

Floating Point Registers:

- The Cortex-M4 with optional floating-point extension, implements 32 32-bit floating point registers named S0 to S31.
- These can be combined two by two forming 16 double precision (64-bit) floating point registers named D0 to D15. D0 is formed by S1:S0 (the concatenation of the registers S1 and S0 where S1 is the most significant word, i.e. the leftmost word).

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Before presenting the instruction set, lets examine:

- Assembly syntax
- 3-operand instructions
- Conditional instructions
- Instructions that affect the flags

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

The most common instruction formats are:

label: MNEMONIC Destination, Operand1, Operand2 ;comment

label: MNEMONIC Destination, Operand2 ;comment

Examples:

```
fmt1: ADD  R2, R4, R5          ;R2 = R4 + R5
fmt2: ADD  R2, R4              ;R2 = R2 + R4
```

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Suggested assembly source file layout:

col 1 col 5 col 13 col 21

```
fmt1: ADD   R2, R4, R5    ;R2 = R4 + R5
```

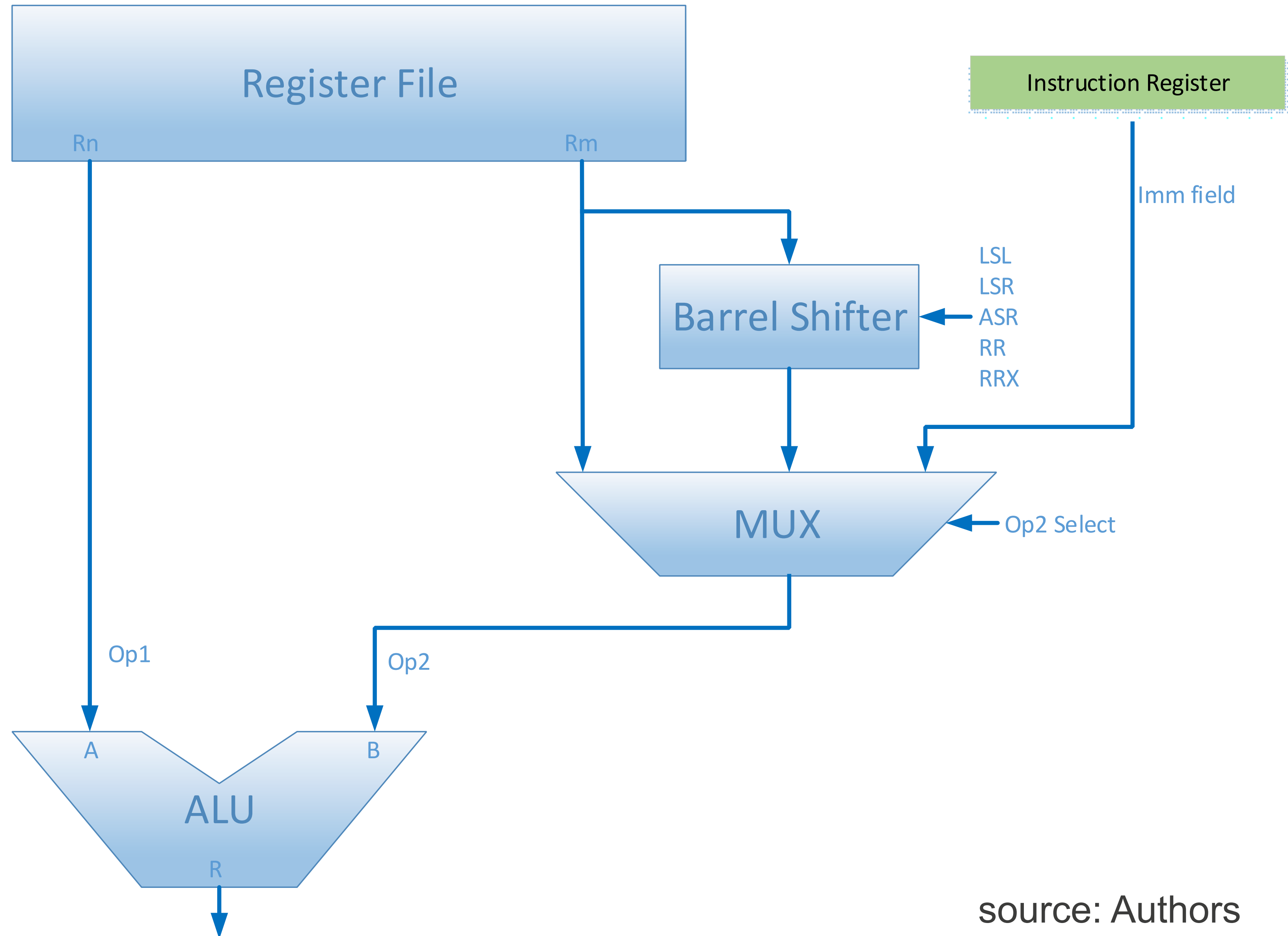
By setting the TABs to 4 spaces, these positions can be easily obtainable.

Only labels should begin on column 1.

Only mnemonics are compulsory. Labels, operands and comments are optional, although they are all very frequent.

© 2020 Renesas Electronics Corporation. All rights reserved.

For an instruction with two operands, the first operand (Op1) is always a register. The second operand (Op2) may be a register, or a register that had its contents shifted or rotated, or an immediate value coded in the instruction.



source: Authors

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Examples of Operand 2

```
ADD  R2, R4, R5            ;Operand 2 is a register (R5)
ADD  R2, R4, R5,LSL #2     ;Operand 2 is a shifted register
                           ;R5 is shifted left by 2 bits
                           ;this corresponds to multiplying its value by 4
ADD  R2, R4, #0xFF         ;Operand 2 is an immediate value
                           ;the hexadecimal value 0xFF
```

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

In the Cortex-M4 instruction set, the programmer explicitly controls if the result of an instruction should affect the flags: N,Z,C,V.

Most instructions have a variant with the letter S appended to the mnemonic. The S variant means: “set the flags”.

```
ADD  R2, R4, R5        ;the result of this addition does
                       ;not affect the flags.
ADDS R2, R4, R5        ;the result of this addition
                       ;affects the N,Z,C and V flags.
```

© 2020 Renesas Electronics Corporation. All rights reserved.

3.4 – INSTRUCTION SET

Many Cortex-M4 instruction can be **conditional**, meaning that the instruction only executes if the flags are in a given state. Except for branch instructions, an instruction must be in an IT block to be conditional. The condition is specified by two letters appended after the mnemonic (see condition table on next slide).

```
ADD     R2,R4,R5        ;the result of this addition does
                        ;not affect the flags.
ADDS    R2,R4,R5        ;the result of this addition
                        ;affects the N,Z,C and V flags.
ITT     EQ              ;start of an IT block with 2 instructions
ADDEQ   R2,R4,R5        ;if Z is set, execute the ADD
ADDSEQ  R2,R4,R5        ;if Z is set, execute the ADD and change
                        ;flags according to result of this instruction
```

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Condition codes mnemonics suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned
CC or LO	C = 0	Lower, unsigned
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned
LS	C = 0 or Z = 1	Lower or same, unsigned
GE	N = V	Greater than or equal, signed
LT	N != V	Less than, signed
GT	Z = 0 and N = V	Greater than, signed
LE	Z = 1 and N != V	Less than or equal, signed
AL	Can have any value	Always. This is the default when no suffix is specified.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Code examples for conditional instructions.

```
        cmp     R12,R10       ;compare the unsigned values in R12 and R10, change flags
        beq     op1           ;branch to op1 if the values of R12 and R10 are equal
        ite     hi            ;two-instruction IT block with HI condition
        addhi   R12,R12,#1    ;if R12 > R10 then increment R12
        addls   R10,R10,#1    ;else increment R10
op1:
        ...
```

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

An immediate value is a constant whose value is encoded in the instruction. Hence, a limited range of values is allowed.

The notation for immediate values is #value.

The notation for negative values is `#-15`.

The notation for hexadecimal values is `#0xFA0`.

Example:

```
ADD    R2,R4,#5      ;R2 = R4 + 5
```

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

- Cortex-M4 instruction codes are either 16-bit or 32-bit.
- 16-bit instructions are called narrow and may have a .N suffix.
- 32-bit instructions are called wide and may have a .W suffix.
- Some mnemonics may be coded either in narrow or wide format, for example:

```
0x37a: 0x1840          ADDS.N    R0, R0, R1     //16-bit code
0x37c: 0xeb10 0x0001   ADDS.W    R0, R0, R1     //32-bit code
```

- 16-bit and 32-bit instruction code can be freely intermixed in a program.
- All instructions must be halfword aligned, i.e. must be stored on an even address.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

- Hence, PC will never hold an odd address => bit 0 of PC is always 0.
- When writing a 32-bit value to PC, bit 0 is ignored => can be used for other purpose.
- In other ARM processors, use bit 0 for interworking (i.e. change of instruction set).
- On Cortex-M, bit 0 must be a 1. This value is stored to the T flag in XPSR.
- Instructions that can be used for interworking (i.e. write to T flag):
  - BX
  - BLX
  - pop {PC}
- Instructions that have as destination register the PC, cause a branch
  - MOV PC, LR
  - ADD PC, PC,R1
- There are restrictions on which instructions may write to PC.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Arithmetic instructions

Instruction	Description	Action
ADD Rd, Rn, Op2	Add a register to Operand2	Rd = Rn + Op2
ADC Rd, Rn, Op2	Add a register to Operand2 and to Carry	Rd = Rn + Op2 + CY
SUB Rd, Rn, Op2	Subtract from a register the Operand2	Rd = Rn - Op2
SBC Rd, Rn, Op2	Subtract from a register the Operand2 and the Borrow (negation of Carry)	Rd = Rn - Op2 - /CY
RSB Rd, Rn, Op2	Subtract from Operand2 a register	Rd = Op2 - Rn
RSC Rd, Rn, Op2	Subtract from Operand2 a register and the Borrow	Rd = Op2 - Rn - /CY
MOV Rd, Op2	Move to Rd from Operand2 (put a copy of Operand2 into Rd)	Rd = Op2
MVN Rd, Op2	Move to Rd /Operand2	Rd = /Op2
MOVT Rd,<imm16>	Move to Rd[31:16] from imm16. Lower bits of Rd are unaffected	Rd[31:16] = imm16 Rd[15:0] unchanged

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Compare and Test

Instruction	Description	
CMP Rn, Op2	Compare: Subtract from Rn the Operand2, discard result, change flags	
CMN Rn, Op2	Compare negative: Add Rn to Operand2, discard result, change flags	
TST Rn, Op2	Test: Rn AND Operand2, discard result, change flags	
TEQ Rn, Op2	Test equivalence: Rn EOR Operand2, discard result, change flags	

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Logical

Instruction	Description	Action
AND Rd, Rn, Op2	AND: bitwise logical AND a register to Operand2	Rd = Rn AND Op2
ORR Rd, Rn, Op2	OR: bitwise logical OR a register to Operand2	Rd = Rn OR Op2
EOR Rd, Rn, Op2	Exclusive OR: bitwise logical XOR a register to Operand2	Rd = Rn XOR Op2
ORN Rd, Rn, Op2	OR NOT: bitwise logical OR a register to NOT(Operand2)	Rd = Rn OR /Op2

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Shift Instructions

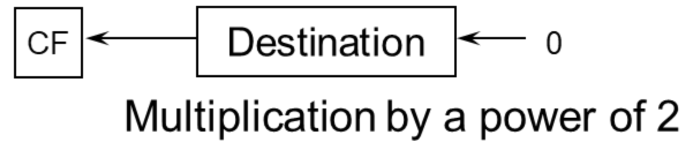
Instruction	Description	#imm Sh range
ASR Rd, Rn, Sh	Arithmetic Shift Right (preserves signal)	1..32
LSL Rd, Rn, Sh	Logical Shift Left	0..31
LSR Rd, Rn, Sh	Logical Shift Right	1..32
ROR Rd, Rn, Sh	Rotate Right	0..31
RRX Rd, Rn	Rotate Right Extended	
Remark: Sh is either	the lower 8 bits of a register (value from 0..255)	
	a 5-bit immediate value representing either 1..32 or 0..31	

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

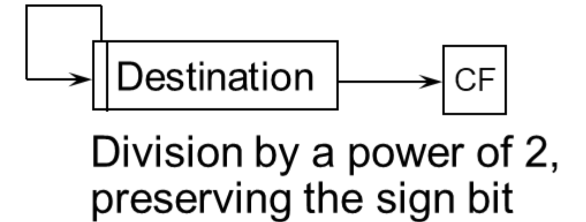
3.4 – INSTRUCTION SET

LSL : Logical Left Shift



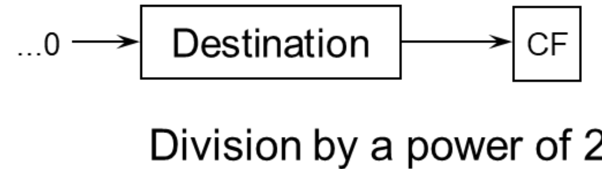
Multiplication by a power of 2

ASR: Arithmetic Right Shift



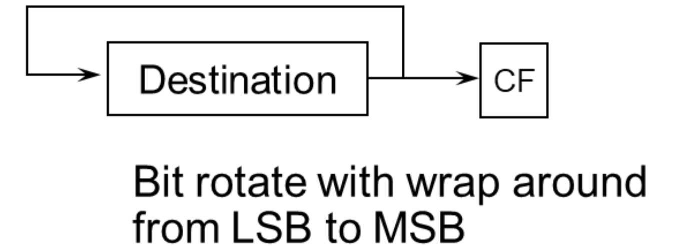
Division by a power of 2, preserving the sign bit

LSR : Logical Shift Right



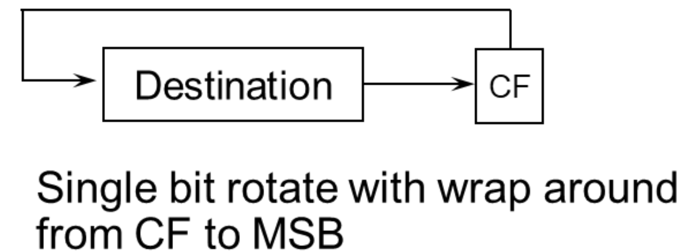
Division by a power of 2

ROR: Rotate Right



Bit rotate with wrap around from LSB to MSB

RRX: Rotate Right Extended



Single bit rotate with wrap around from CF to MSB

source: ARM
The ARM Architecture

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Shift Operators to be used in Operand2

Operator	Description	#imm Sh range
ASR Sh	Arithmetic Shift Right (preserves signal)	1..32
LSL Sh	Logical Shift Left	0..31
LSR Sh	Logical Shift Right	1..32
ROR Sh	Rotate Right	0..31
RRX	Rotate Right Extended	
Remark: Sh is either	the lower 8 bits of a register (value from 0..255)	
	a 5-bit immediate value representing either 1..32 or 0..31	
Usage:	R4, LSL #3 (Operand2 is R4 << 3)	

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Multiply

Instruction	Description	Action
Instructions that multiply 32-bit by 32-bit resulting 32-bit with wrapping (LSW is preserved and higher bits are discarded)		
MUL Rd, Rm, Rs	Multiply	Rd = Rm * Rs
MLA Rd, Rm, Rs, Rn	Multiply and accumulate	Rd = Rm * Rs + Rn
MLS Rd, Rm, Rs, Rn	Multiply and subtract	Rd = Rm * Rs - Rn
Long multiplication: multiply 32-bit by 32-bit resulting 64-bit		
UMULL RdLo, RdHi, Rm, Rs	Unsigned long multiply	RdHi:RdLo = unsigned(Rm*Rs)
UMLAL RdLo, RdHi, Rm, Rs	Unsigned long multiply and accumulate	RdHi:RdLo = unsigned(RdHi:RdLo + Rm*Rs)
UMAAL RdLo, RdHi, Rm, Rs	Unsigned long multiply and accumulate double	RdHi:RdLo = unsigned(RdHi+RdLo + Rm*Rs)
SMULL RdLo, RdHi, Rm, Rs	Signed long multiply	RdHi:RdLo = signed(Rm*Rs)
SMLAL RdLo, RdHi, Rm, Rs	Signed long multiply and accumulate	RdHi:RdLo = signed(RdHi:RdLo + Rm*Rs)

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Divide

Instruction	Description	Action
UDIV Rd, Rn, Rm	Unsigned divide	Rd = Rn / Rm
SDIV Rd, Rn, Rm	Signed divide	Rd = Rn / Rm

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Bit field operations

A bit field is a sequence of bits in a register.

A bit field is characterized by two values:

- Width: the number of bits in the bit field (1..32);
- lsb: the position of the least significant bit in the bitfield (0..31).

Instruction	Description	Action
BFC Rd,#<lsb>,#<width>	Bit field clear	clear Rd[(width+lsb-1)..lsb], others unchanged
BFI Rd, Rn,#<lsb>,#<width>	Bit field insert. Copy the <width> LSb of Rn to Rd	Rd[(width+lsb-1)..lsb] = Rn[(width-1)..0]
SBFX Rd, Rn,#<lsb>,#<width>	Signed bit field extract.	Copy bitfield from Rn to LSb of Rd and sign extend.
UBFX Rd, Rn,#<lsb>,#<width>	Unsigned bit field extract.	Copy bitfield from Rn to LSb of Rd and zero extend.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Memory access instructions

Instruction type	Description
LDRB	Load byte. Read a byte from memory and store in the LSB of a register.
LDRH	Load half word. Read a half-word from memory and store in the lower half-word of a register.
LDR	Load register. Read a word from memory and store in a register.
LDRD	Load double. Read a double word form memory and store in two registers.
STRB	Store byte. Store the LSB of a register into memory.
STRH	Store half-word. Store the lower half of a register into memory.
STR	Store register. Store a register into memory.
STRD	Store double. Store the two registers into memory.
LDM	Load multiple. Read several (up to 16) registers from memory.
STM	Store multiple. Store several (up to 16) registers into memory.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

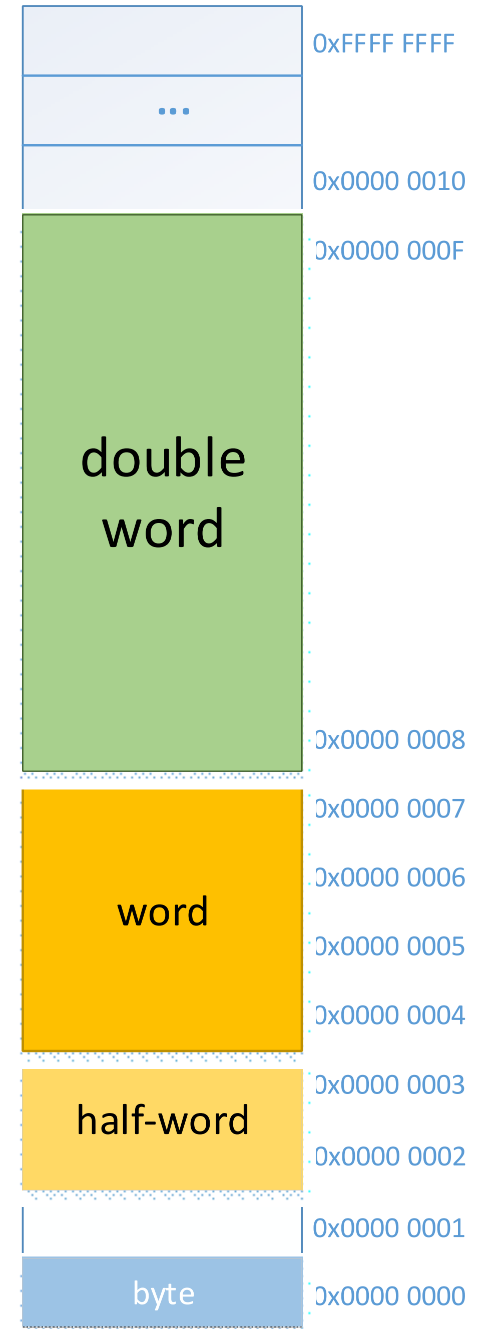
3.4 – INSTRUCTION SET

Memory access instructions – addressing

For the purposes of addressing, memory is just a very large vector of bytes. For Cortex-M, it is a vector with 4G entries.

The four data types that can be accesses with LDR/STR instructions are shown here. In a little-endian memory system, when a data type occupies more than 1 byte in memory, its address is the address of the LSB (Least Significant Byte).

shown: byte at 0x0, half-word at 0x2, word at 0x4, and double-word at 0x8



source: Authors

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

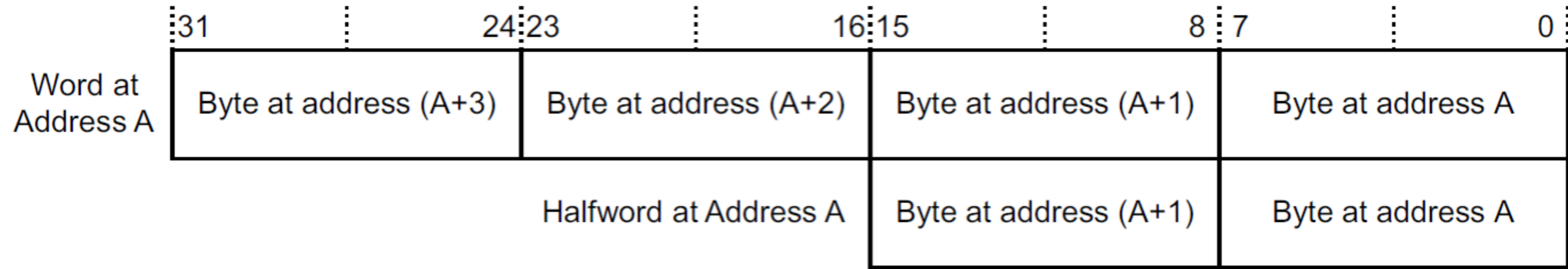


Figure A3-1 Little-endian byte format

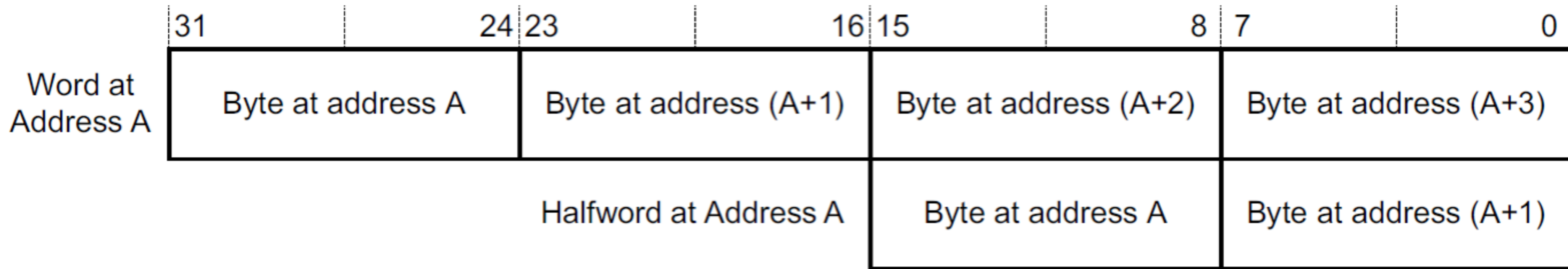


Figure A3-2 Big-endian byte format

source: DDI0403E.B ARMv7-M Architecture Reference Manual

© 2020 Renesas Electronics Corporation. All rights reserved.

3.4 – INSTRUCTION SET

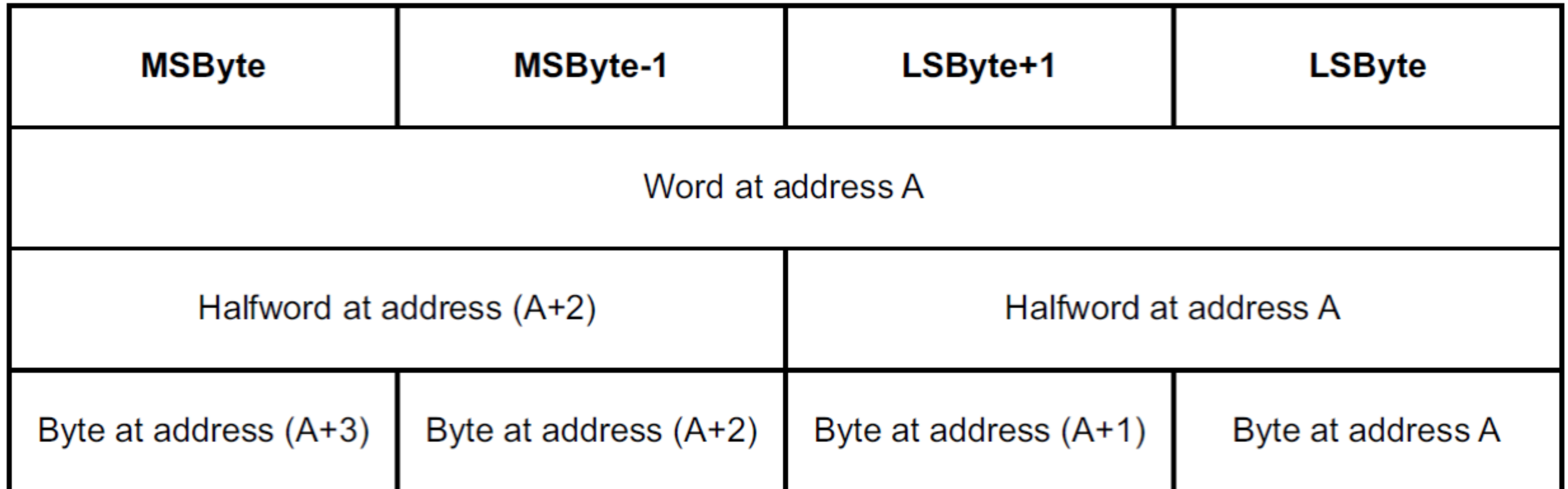


Figure A3-3 Little-endian memory system

source: DDI0403E.B ARMv7-M Architecture Reference Manual

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Memory access instructions – addressing modes

Indexing Mode	Example	Action	Change in base register
Pre-index with writeback (!)	LDR R0,[R1,#4] !	R0 = [R1 + 4] (R0 gets the contents of memory location at address R1+4)	R1 = R1 + 4
	LDR R0,[R1,R2] !	R0 = [R1+R2]	R1 = R1 + R2
	LDR R0,[R1,R2,LSL #2] !	R0 = [R1 + (R2 << 2)]	R1 = R1 + R2 << 2
Pre-index	LDR R0,[R1,#4]	R0 = [R1 + 4]	no change
	LDR R0,[R1,R2]	R0 = [R1+R2]	no change
	LDR R0,[R1,R2,LSL #2]	R0 = [R1 + (R2 << 2)]	no change
Pre-index	LDR R0,[R1],#4	R0 = [R1]	R1 = R1 + 4
	LDR R0,[R1],R2	R0 = [R1]	R1 = R1 + R2
	LDR R0,[R1],R2,LSL #2	R0 = [R1]	R1 = R1 + R2 << 2

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Execution flow control instructions

Instruction	Usage	Branch Range
B.N <label>	16-bit Branch to target address.	-256 to 254 bytes
B.W <label>	32-bit Branch to target address.	+/–1 MB
CBNZ <label> *CBZ* <label>	Compare and Branch on Nonzero. Compare and Branch on Zero.	0-126 B
BL <label>	Call a subroutine.	+/–16 MB
BLX <register>	Call a subroutine, optionally change instruction set.	Any
BX <register>	Branch to target address, optionally change instruction set.	Any
TBB	TBB: Table Branch, byte offsets.	0-510 B
TBH	TBH: Table Branch, halfword offsets.	0-131070 B

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.4 – INSTRUCTION SET

Miscellaneous instructions

Instruction	Usage
CPSID	Change Processor State, Disable Interrupts.
CPSIE	Change Processor State, Enable Interrupts.
DMB	Data Memory Barrier.
DSB	Data Synchronization Barrier.
ISB	Instruction Synchronization Barrier.
MRS	Move to Register from Special Register.
MSR	Move to Special Register from Register.
NOP	No Operation.
SVC	Supervisor Call.
WFI	Wait for Interrupt.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS

The normal flow of execution of a program is to execute the next instruction in memory, unless a Branch, Subroutine Call or Return is executed. Hence, a human processor could execute the same program in the same order.

An exception if a break in this normal flow of execution. Such a break can be caused by:

- Hardware interrupt,
- Fault (e.g. memory access error, divide by 0, invalid instruction code),
- Software generate exception.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS

Exceptions occur **asynchronousl**y, this is, at any point of the execution. They may occur many time and on successive executions of the program they usually occur at different places of this program.

When an exception occurs it must be **serviced**. Meaning that a software routine must either respond to the interrupt request or take steps to resolve or mitigate the fault.

This routine is called: **exception handler routine, interrupt service routine,** or **interrupt handler**.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – INTERRUPTS

Hardware Interrupts are one of the kind of exceptions.

Interrupts are an efficient way for a peripheral to inform the processor that it requires servicing. If interrupts were not available, the processor would have to periodically poll the peripherals (thus termed **polling**) to check if service is required. Polling is an inefficient technique.

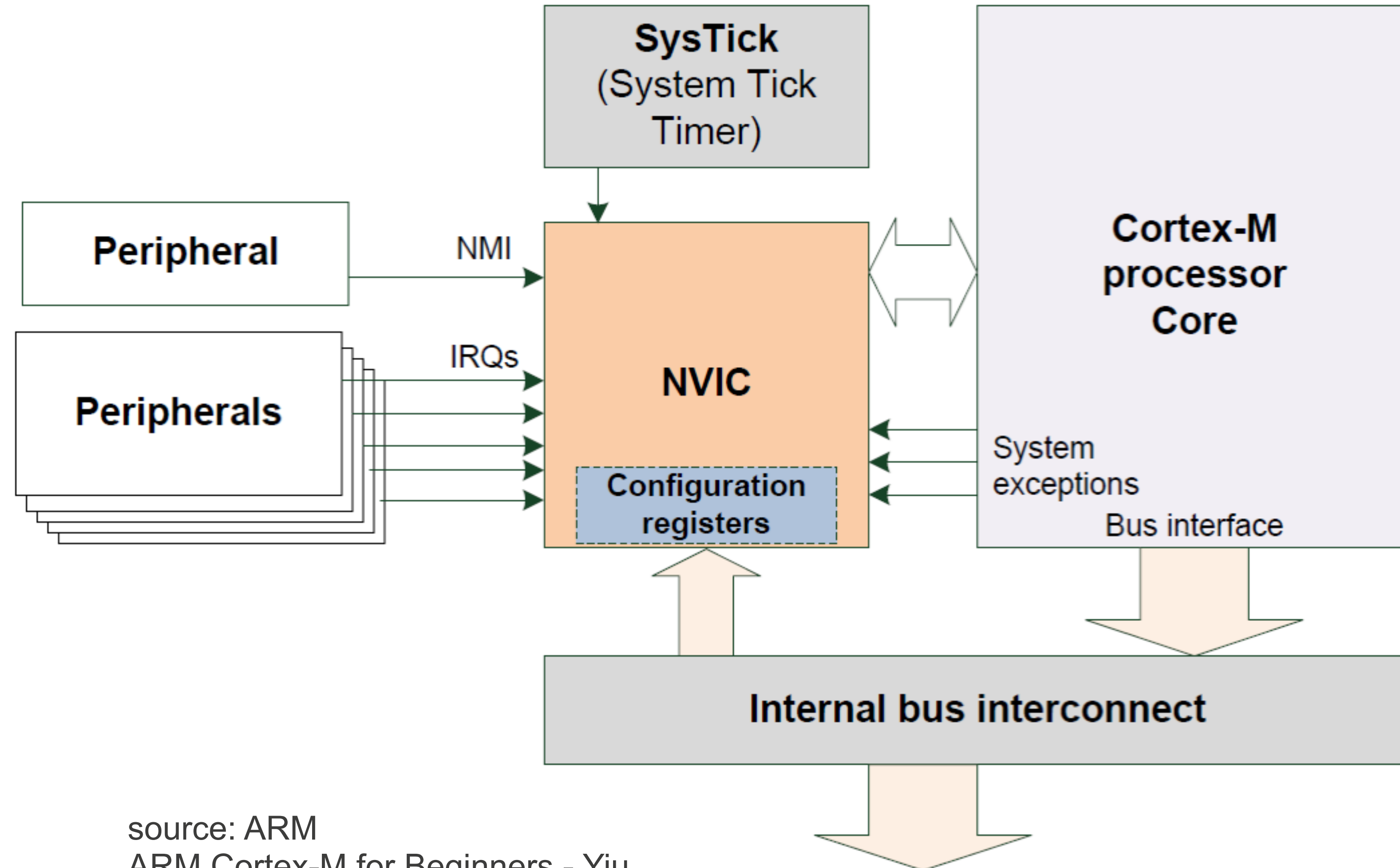
© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – INTERRUPTS

Basic concepts of interrupts:

1. Peripheral sends an Interrupt Request (IRQ) to an Interrupt Controller
2. Interrupt Controller selects the highest priority non-masked interrupt request and informs the core.
3. If the priority of the IRQ is sufficiently high, when the instruction currently in execution finishes then the IRQ is serviced.



source: ARM
ARM Cortex-M for Beginners - Yiu

© 2020 Renesas Electronics Corporation. All rights reserved.

3.5 – EXCEPTIONS – INTERRUPTS – DETAILED PROCESS

1- An external device, such as a peripheral, requests an interrupt (IRQ) by signaling to the interrupt controller.

The input lines of the interrupt controller (240 in the NVIC of a Cortex-M4) can be either level sensitive of edge sensitive.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – INTERRUPTS – DETAILED PROCESS

2- Upon receiving an IRQi (hardware signal on input i of the interrupt controller - IC) then the IC performs two checks:

a) if input i is masked or not;

b) if there is another request (IRQj on input j) already being sent to the processor.

If IRQi is not masked and if its priority is higher than IRQj's priority (or no request is currently being sent to the processor)

then IRQi is forwarded to the processor.

© 2020 Renesas Electronics Corporation. All rights reserved.

3.5 – EXCEPTIONS – INTERRUPTS – DETAILED PROCESS

3- The processor, upon receiving an IRQ verifies if its priority is sufficiently high:

a) PRIMASK and FAULTMASK, when set, impose a priority level of 0 or -1 respectively. Hence, when FAULTMASK is set, all exceptions from 3 on are masked.

b) If an exception is active (being serviced) then only a higher priority exception may preempt its handler.

If both these conditions are met, servicing starts at the end of the current instruction.

Exception number[a]	IRQ number[a]	Exception type	Priority	Vector address or offset[b]	Activation
1	-	Reset	-3, the highest	0x00000004	Asynchronous
2	-14	NMI	-2	0x00000008	Asynchronous
3	-13	HardFault	-1	0x0000000C	-
4	-12	MemManage	Configurable[c]	0x00000010	Synchronous
5	-11	BusFault	Configurable[c]	0x00000014	Synchronous when precise, asynchronous when imprecise
6	-10	UsageFault	Configurable[c]	0x00000018	Synchronous
7-10	-	Reserved	-	-	-
11	-5	SVCall	Configurable[c]	0x0000002C	Synchronous
12-13	-	Reserved	-	-	-
14	-2	PendSV	Configurable[c]	0x00000038	Asynchronous
15	-1	SysTick	Configurable[c]	0x0000003C	Asynchronous
16	0	Interrupt (IRQ)	Configurable[d]	0x00000040 [e]	Asynchronous

source: ARM
Cortex™-M4 Devices Generic User Guide

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – INTERRUPTS – DETAILED PROCESS

4- Eight registers are pushed onto the Stack

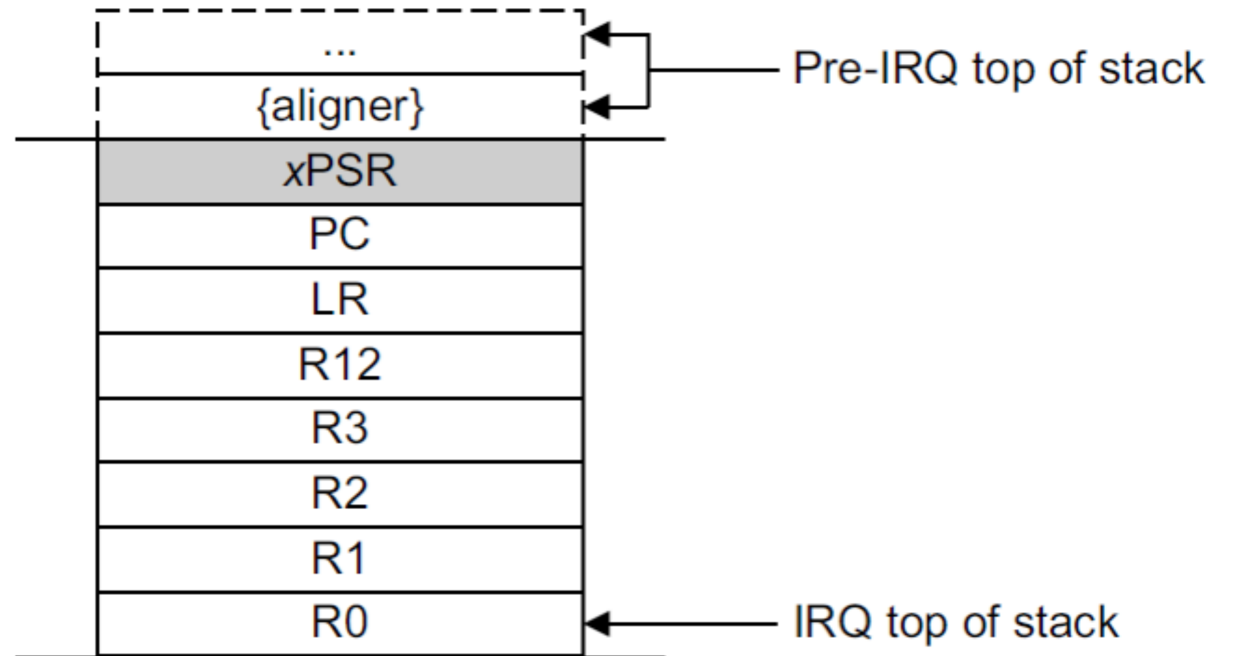
The Interrupt changes state to

Active.

Since R0-R3 and R12 are stacked,

any C procedure following ATPCS

can be registered as a handler.



Exception frame without
floating-point storage

source: ARM
Cortex™-M4 Devices Generic User Guide

© 2020 Renesas Electronics Corporation. All rights reserved.

3.5 – EXCEPTIONS – INTERRUPTS – DETAILED PROCESS

5- Register LR is loaded with one of the EXC_RETURN values, depending on the current state of the processor. Note that EXC_RETURN represents memory addresses in a region where code is not allowed.

EXC_RETURN[31:0]	Description
0xFFFFFFF1	Return to Handler mode, exception return uses non-floating-point state from the MSP and execution uses MSP after return.
0xFFFFFFF9	Return to Thread mode, exception return uses non-floating-point state from MSP and execution uses MSP after return.
0xFFFFFFFD	Return to Thread mode, exception return uses non-floating-point state from the PSP and execution uses PSP after return.
0xFFFFFFE1	Return to Handler mode, exception return uses floating-point-state from MSP and execution uses MSP after return.
0xFFFFFFE9	Return to Thread mode, exception return uses floating-point state from MSP and execution uses MSP after return.
0xFFFFFFED	Return to Thread mode, exception return uses floating-point state from PSP and execution uses PSP after return.

source: ARM
Cortex™-M4 Devices Generic User Guide

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – INTERRUPTS – DETAILED PROCESS

6- The processor reads from the vector table the initial address of the handler for the Interrupt. This value is loaded to PC and the execution of the handler starts.

Important: since the handler is Thumb-2 code the addresses in the vector table must have its LSb set to 1.

Exception number	IRQ number	Offset	Vector
16+n	n	0x0040+4n	IRQn
. . .		. . . 0x004C	. . .
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

source: ARM
Cortex™-M4 Devices Generic User Guide

Figure 2-2 Vector table

© 2020 Renesas Electronics Corporation. All rights reserved.

3.5 – EXCEPTIONS – SERVICING

Exception Servicing

The handler must service the interrupt request, in this process at least three actions must be taken:

1. The interrupt request signal must be deactivated, otherwise the processor would continuously be servicing this interrupt.
2. Any volatile data (such as a byte that arrived on the UART and is available in the Receiving Register) must be saved.
3. Apart from the registers saved in Step 4 of the entry process, any other register must be saved by the handler before modifying it and these registers must be restored before returning to the interrupted code.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – RETURN

Exception Return

The instruction that causes the return from the handler to the interrupted code is either:

```
BX LR
```

or

```
POP {...,PC}   // if this instruction is used the the entry of the handler must be PUSH {...,LR}
```

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTIONS – RETURN

Exception Return

What happens when a value such as 0xFFFF FFF1 is loaded to the PC?

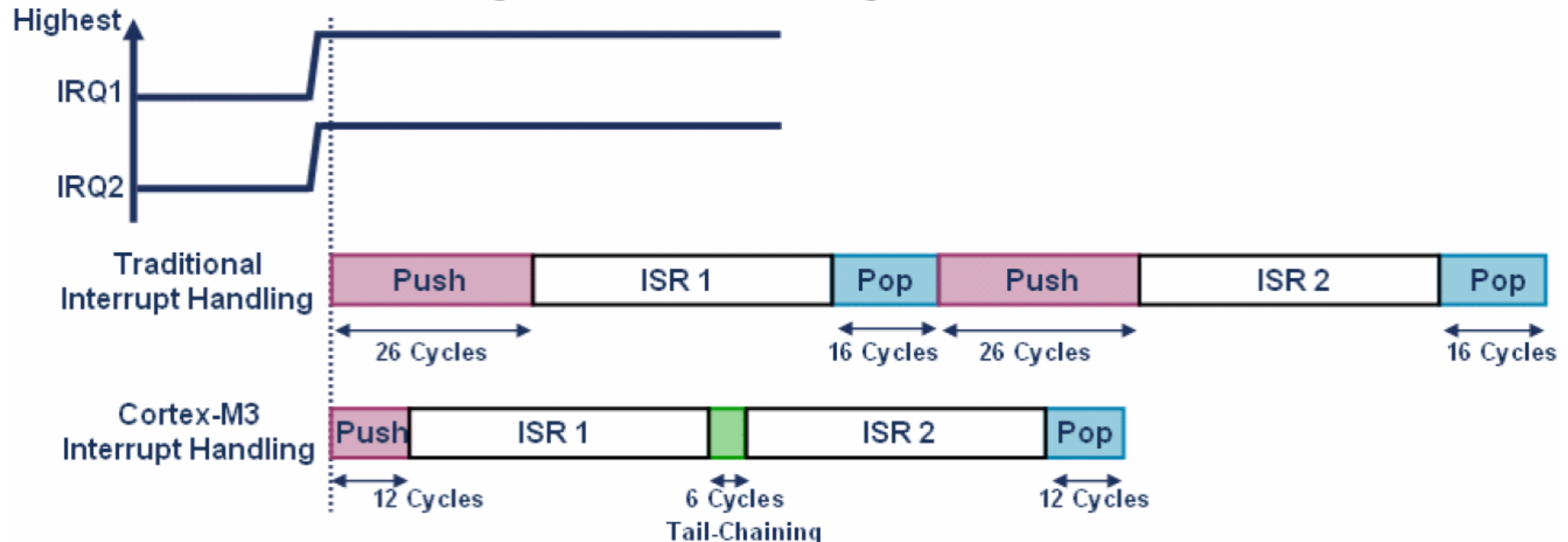
Being an invalid code address, the processor detects that this is an EXC_RETURN code and proceeds with the actions described in the table in the slide Step 5 of the Interrupt Entry.

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTION HANDLING

Tail Chaining: optimization that avoids registers pop followed by registers push when one exception is handled right after another.



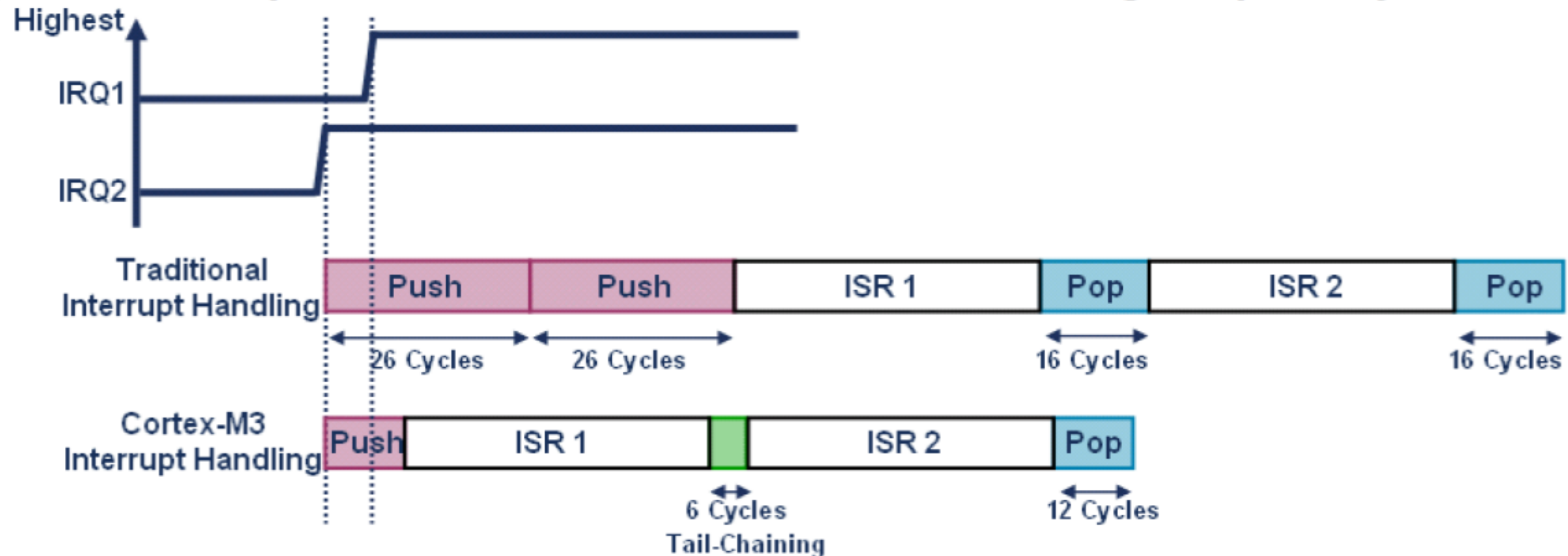
source: ARM
An Introduction to the ARM Cortex-M3 Processor - Shyam Sadasivan

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

3.5 – EXCEPTION HANDLING

Late arrival: optimization where a higher priority interrupt is serviced first even if it arrives while a prior lower priority interrupt is already in the stage of pushing registers.



source: ARM
An Introduction to the ARM Cortex-M3 Processor - Shyam Sadasivan

© 2020 Renesas Electronics Corporation. All rights reserved.

BIG IDEAS FOR EVERY SPACE RENESAS

Renesas.com

© 2020 Renesas Electronics Corporation. All rights reserved.