



**FACULTAD
DE INGENIERIA**

Universidad de Buenos Aires

Arquitectura de Microprocesadores

Mg. Ing. Facundo Larosa

Carrera de Especialización en Sistemas Embebidos

Presentación



Alcance del curso

- Comprender arquitecturas (modelo del programador, set de instrucciones) de microprocesadores de uso común en sistemas embebidos y en sistemas de propósito general
 - Cortex M *
 - Cortex A
- Introducir a la programación de los microprocesadores usando su lenguaje nativo

Planificación

| Clase | Descripción | Modalidad |
|-------|--|------------------|
| 1 | ARMv7-M Modelo del programador | Teórica |
| 2 | ARMv7-M <i>Core peripherals</i> | Teórica |
| 3 | ARMv7-M: <i>Instruction set architecture</i> | Teórica/Práctica |
| 4 | ARMv7-M: Práctica | Práctica |
| 5 | ARMv7-M: Práctica | Práctica |
| 6 | Introducción a SIMD | Teórica |
| 7 | Práctica integradora | Teórica/Práctica |
| 8 | Examen teórico práctico | - |

Ver planificación con cronograma detallado en : <https://sites.google.com/site/arquitecturademicros/>

¿Qué necesitan para el cursado?

- Material de la asignatura
- Bibliografía del curso
 - YIU, Joseph , “The Definitive Guide to ARM Cortex M3 and M4 processors”
 - YIU, Joseph , “The Definitive Guide to ARM Cortex M0 and M0+ processors”
 - “ARMv7-M Architecture Reference Manual”, ARM Inc.
 - “ARMv7-AR Architecture Reference Manual”, ARM Inc.
 - “ARM Cortex R Architecture”, White Paper, ARM, Inc.
- PC con sistema operativo Linux Ubuntu con Eclipse instalado y configurado para utilizar la EduCIAA
- EduCIAA
- Directorio de trabajo incorporado al Eclipse:

<https://gitlab.com/AuP-UBA/arquitecturaDeMicroprocesadores.git>

ARM

ARM

Agenda

- 1) Introducción a los perfiles de ARM v7 (A-R-M)
- 2) Comparación entre familias
- 3) Cortex M3/M4
 - 1) Características generales
 - 2) Modos de operación
 - 3) Registros generales y especiales
 - 4) *FPU*
 - 5) *Stack*
 - 6) Secuencia de reset
 - 7) *Core peripherals*
 - 1) *NVIC*
 - 2) *Systick*

Introducción a los perfiles de ARM v7

- **Cortex A (Application)**

Procesadores de alto rendimiento orientados a la implementación de sistemas operativos en sistemas embebidos de alta performance



Introducción a los perfiles de ARM v7

- **Cortex R (Real time)**

Procesadores orientados a sistemas de tiempo real donde prima la necesidad de implementar soluciones de baja latencia y alta capacidad de procesamiento

Ejemplos: sistemas del automóvil (seguridad, frenado, etc.), sistemas médicos, industriales, etc.

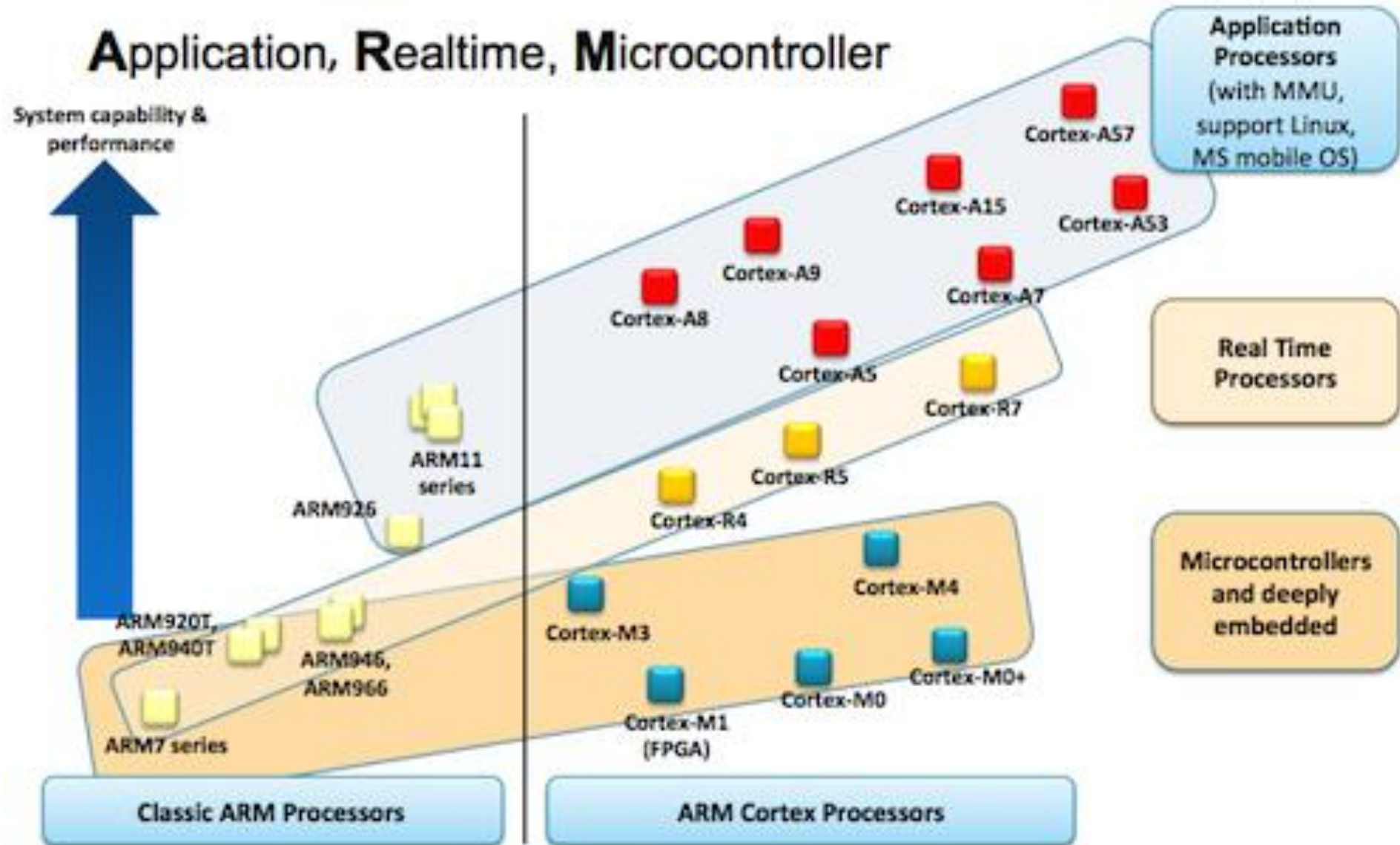
Introducción a los perfiles de ARM v7

- **Cortex M (Microcontrollers)**

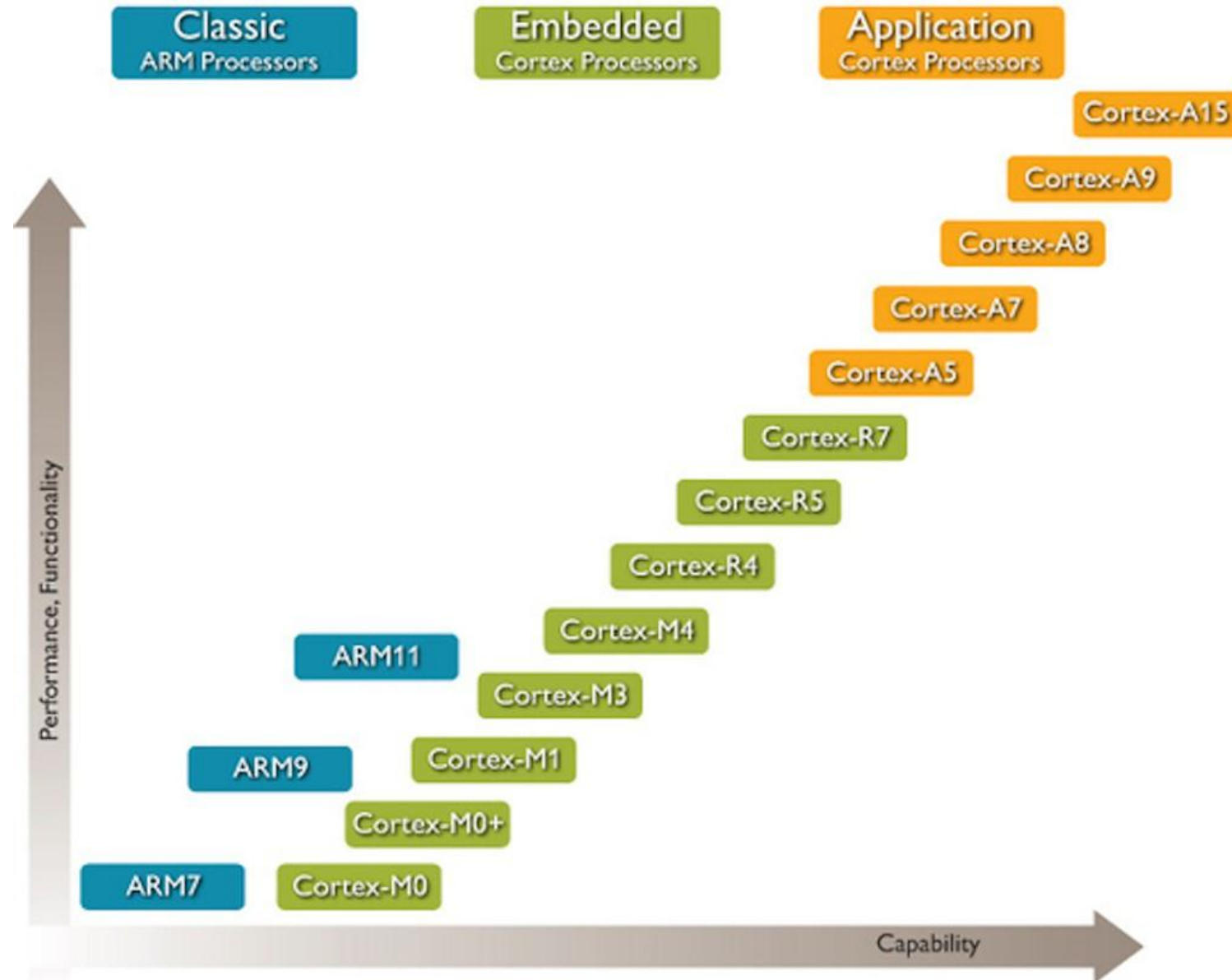
Procesadores orientados a dispositivos de consumo masivo y sistemas embebidos compactos (diseñados para alta densidad de código y programación en C)

- **Cortex M0 / M0+:** Implementación mínima para bajo consumo y bajo costo
- **Cortex M3 / M4 / M7:** Mayor performance, agregan funcionalidades para procesamiento digital de señales, unidad de protección de memoria, etc.

Familia ARM



Familia ARM



Cortex M : Comparación arquitecturas

| ARM Cortex-M | SysTick Timer | Bit-banding | Memory Protection Unit (MPU) | Tightly-Coupled Memory (TCM) | CPU cache | Memory architecture | ARM architecture |
|---------------------------|---------------|-------------------------|------------------------------|------------------------------|--------------------|---------------------|------------------|
| Cortex-M0 ^[1] | Optional* | Optional ^[9] | No | No | No ^[10] | Von Neumann | ARMv6-M |
| Cortex-M0+ ^[2] | Optional* | Optional ^[9] | Optional (8) | No | No | Von Neumann | ARMv6-M |

| | | | | | | | |
|--------------------------|----------|----------|----|----------|----|-------------|---------|
| Cortex-M1 ^[3] | Optional | Optional | No | Optional | No | Von Neumann | ARMv6-M |
|--------------------------|----------|----------|----|----------|----|-------------|---------|

| | | | | | | | |
|--------------------------|-----|-----------|--------------------|----------|----------------------------|---------|----------|
| Cortex-M3 ^[4] | Yes | Optional* | Optional (8) | No | No | Harvard | ARMv7-M |
| Cortex-M4 ^[5] | Yes | Optional* | Optional (8) | No | ⌚ Possible ^[11] | Harvard | ARMv7E-M |
| Cortex-M7 | Yes | No | Optional (8 or 16) | Optional | Optional | Harvard | ARMv7E-M |

Fuente: https://en.wikipedia.org/wiki/ARM_Cortex-M

Cortex M : Comparación extensiones

| ARM Cortex-M | Thumb | Thumb-2 | Hardware multiply | Hardware divide | Saturated math | DSP extensions | Floating-Point Unit (FPU) | ARM architecture |
|---------------------------|--------|---------|---------------------|-----------------|----------------|----------------|---------------------------|------------------|
| Cortex-M0 ^[1] | Most | Some | 32-bit result | No | No | No | No | ARMv6-M |
| Cortex-M0+ ^[2] | Most | Some | 32-bit result | No | No | No | No | ARMv6-M |
| Cortex-M1 ^[3] | Most | Some | 32-bit result | No | No | No | No | ARMv6-M |
| Cortex-M3 ^[4] | Entire | Entire | 32 or 64-bit result | Yes | Yes | No | No | ARMv7-M |
| Cortex-M4 ^[5] | Entire | Entire | 32 or 64-bit result | Yes | Yes | Yes | Optional: SP | ARMv7E-M |
| Cortex-M7 | Entire | Entire | 32 or 64-bit result | Yes | Yes | Yes | Optional: SP or SP & DP | ARMv7E-M |

Fuente: https://en.wikipedia.org/wiki/ARM_Cortex-M

Cortex M : Comparación ISA

| Instructions | Instruction size | Cortex M0 | Cortex M0+ | Cortex M1 | Cortex M3 | Cortex M4 | Cortex M7 |
|---|------------------|-----------|------------|-----------|-----------|-----------------|-----------------|
| ADC, ADD, ADR, AND, ASR, B, BIC, BKPT, BLX, BX, CMN, CMP, CPS, EOR, LDM, LDR, LDRB, LDRH, LDRSB, LDRSH, LSL, LSR, MOV, MUL, MVN, NOP, ORR, POP, PUSH, REV, REV16, REVSH, ROR, RSB, SBC, SEV, STM, STMIA, STR, STRB, STRH, SUB, SVC, SXTB, SXTB, SXTB, SXTB, TST, UXTB, UXTB, WFE, WFI, YIELD | 16-bit | Yes | Yes | Yes | Yes | Yes | Yes |
| BL, DMB, DSB, ISB, MRS, MSR | 32-bit | Yes | Yes | Yes | Yes | Yes | Yes |
| CBNZ, CBZ, IT | 16-bit | No | No | No | Yes | Yes | Yes |
| ADC, ADD, ADR, AND, ASR, B, BFC, BFI, BIC, CDP, CLREX, CLZ, CMN, CMP, DBG, EOR, LDC, LDMA, LDMD, LDR, LDRB, LDRBT, LDRD, LDREX, LDREXB, LDREXH, LDRH, LDRHT, LDRSB, LDRSBT, LDRSHT, LDRSH, LDRT, MCR, LSL, LSR, MLS, MCRR, MLA, MOV, MOVT, MRC, MRRC, MUL, MVN, NOP, ORN, ORR, PLD, PLDW, PLI, POP, PUSH, RBIT, REV, REV16, REVSH, ROR, RRX, RSB, SBC, SBFX, SDIV, SEV, SMLAL, SMULL, SSAT, STC, STMDB, STR, STRB, STRBT, STRD, STREX, STREXB, STREXH, STRH, STRHT, STRT, SUB, SXTB, SXTB, TBB, TBH, TEQ, TST, UBFX, UDIV, UMLAL, UMULL, USAT, UXTB, UXTB, WFE, WFI, YIELD | 32-bit | No | No | No | Yes | Yes | Yes |
| PKH, QADD, QADD16, QADD8, QASX, QDADD, QDSUB, QSAX, QSUB, QSUB16, QSUB8, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSAX, SHSUB16, SHSUB8, SMLABB, SMLABT, SMLATB, SMLATT, SMLAD, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD, SMLAWB, SMLAWT, SMLSD, SMLSDB, SMMLA, SMMLS, SMMUL, SMUAD, SMULBB, SMULBT, SMULTT, SMULTB, SMULWT, SMULWB, SMUSD, SSAT16, SSAX, SSUB16, SSUB8, SXTAB, SXTAB16, SXTAH, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSAX, UHSUB16, UHSUB8, UMAAL, UQADD16, UQADD8, UQASX, UQSAX, UQSUB16, UQSUB8, USAD8, USADA8, USAT16, USAX, USUB16, USUB8, UXTAB, UXTAB16, UXTAH, UXTB16 | 32-bit | No | No | No | No | Yes | Yes |
| VABS, VADD, VCMPE, VCVT, VCVTR, VDIV, VLDM, VLDR, VMLA, VMLS, VMOV, VMRS, VMSR, VMUL, VNEG, VNMLA, VNMLS, VMUL, VPOP, VPUSH, VSQRT, VSTM, VSTR, VSUB | 32-bit | No | No | No | No | Optional SP FPU | Optional SP FPU |
| VCVTA, VCVTM, VCVTN, VCVTP, VMAXNM, VMINNM, VRINTA, VRINTM, VRINTN, VRINTP, VRINTR, VRINTX, VRINTZ, VSEL | 32-bit | No | No | No | No | No | Optional DP FPU |

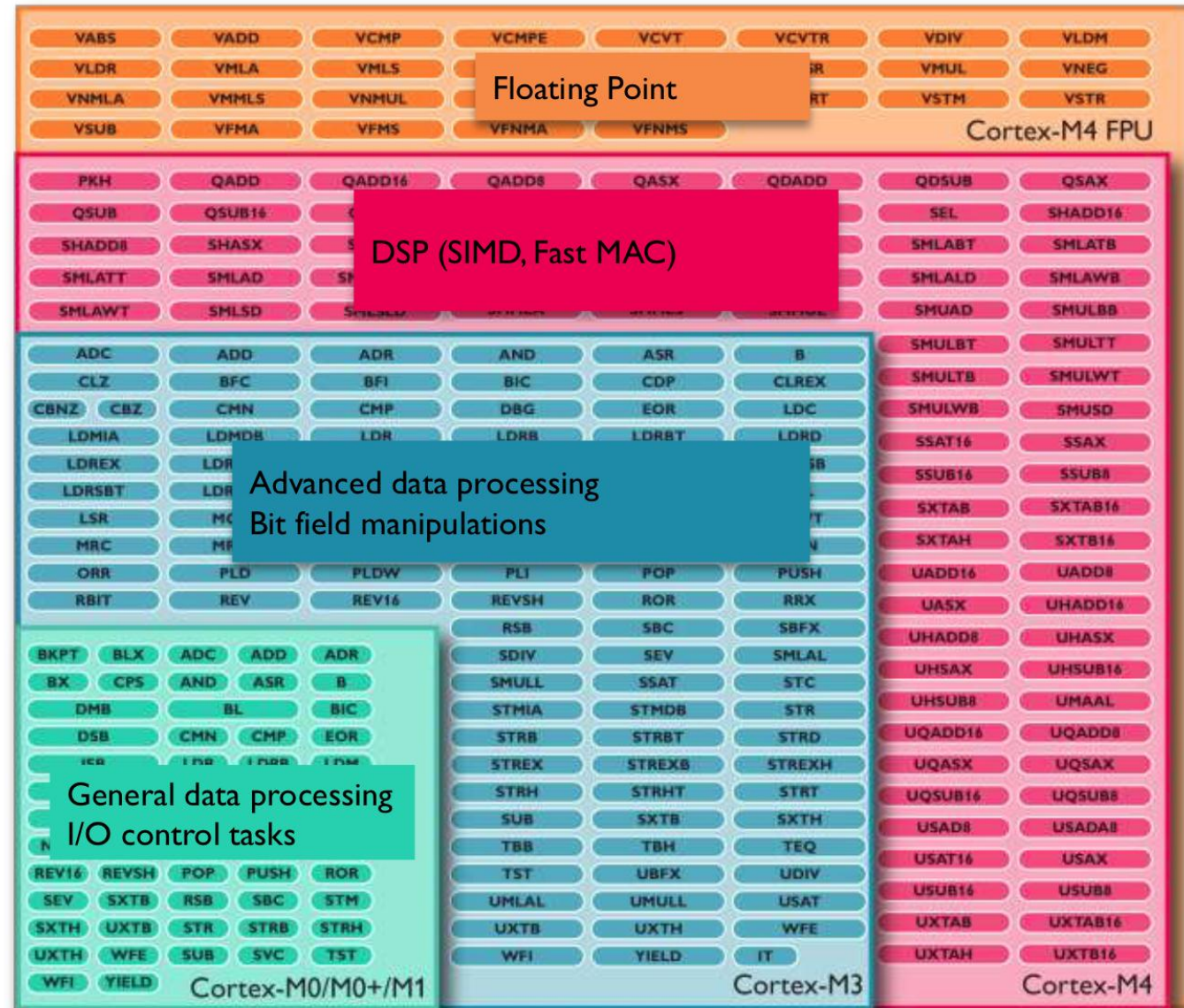
Fuente: https://en.wikipedia.org/wiki/ARM_Cortex-M

ARM Cortex-M Instruction Set Architecture

Cortex-M4

Cortex-M3

Cortex-M0/M0+

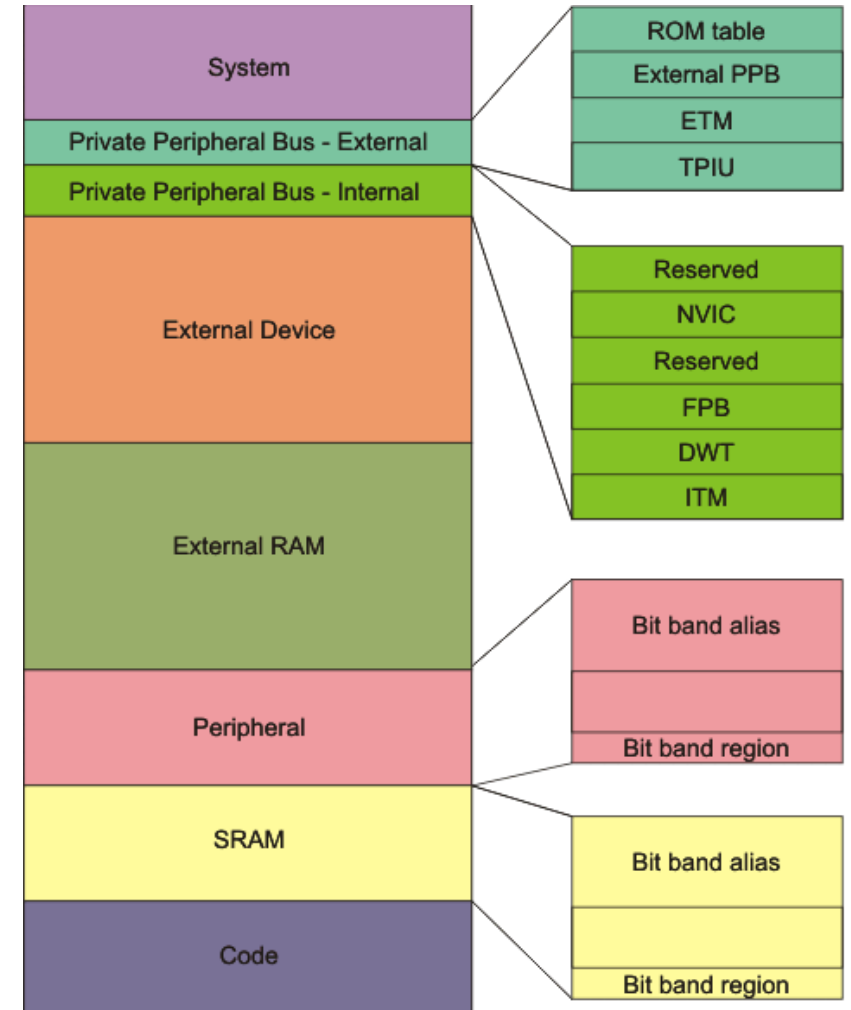


Cortex M3/M4: Características generales

- Diseño RISC (*Reduced Instruction Set Computers*)
- Arquitectura de 32 bits
 - Registros de 32 bits
 - Buses de 32 bits
 - Data path 32 bits
 - Aún así... capacidad de manejar 8 / 16 bits de forma eficiente
- Arquitectura Harvard
- Pipeline de tres etapas (*fetch, decode, execute*)

Cortex M3/M4: Características generales

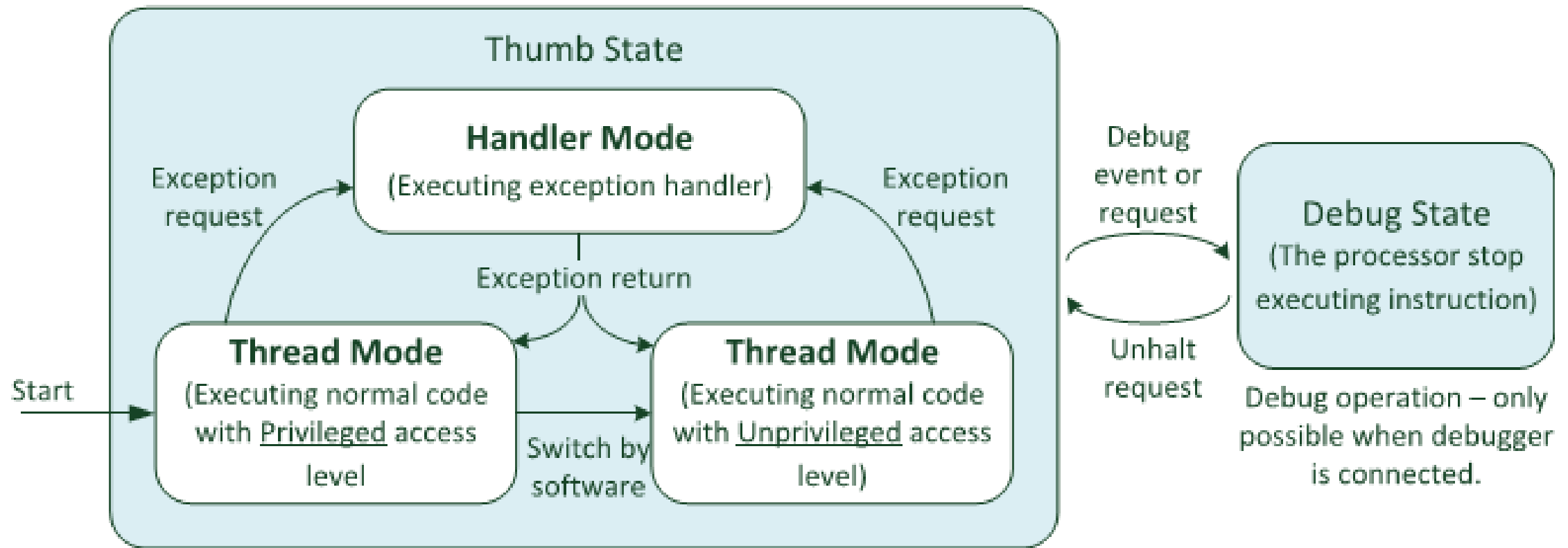
- Mapa de memoria plano de 4 GB
- Arquitectura Load-Store
- Set de instrucciones mixto de 16/32 bits (Thumb 2)
- Alta densidad de código
- Características de bajo consumo (<200µA/MHz, modos de bajo consumo)



Cortex M3/M4: Características generales

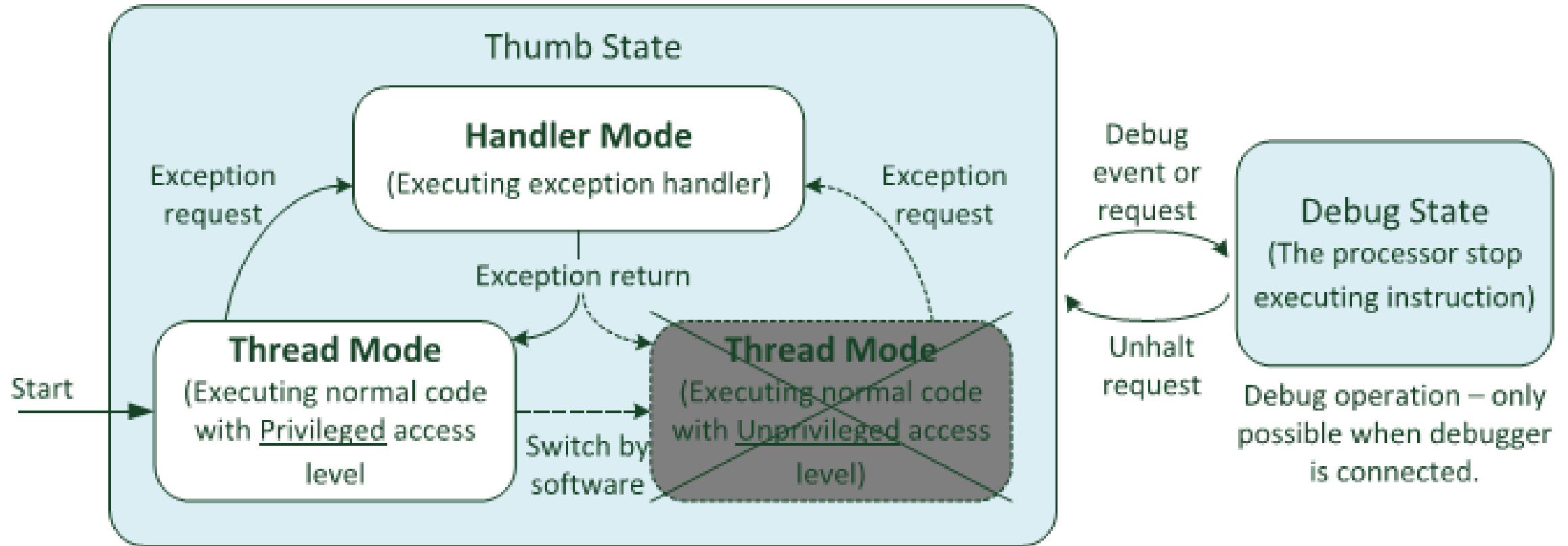
- Unidad de gestión de memoria (MMU) opcional
- Controlador de interrupciones: Nested vectored interrupt controller (NVIC)
- Soporte a SO embebidos:
 - SysTick
 - Banked stack pointers (MSP, PSP)
 - Niveles de privilegio
- Orientado a la programación en C
- Unidad de debugging

Cortex M3/M4: Modos de operación



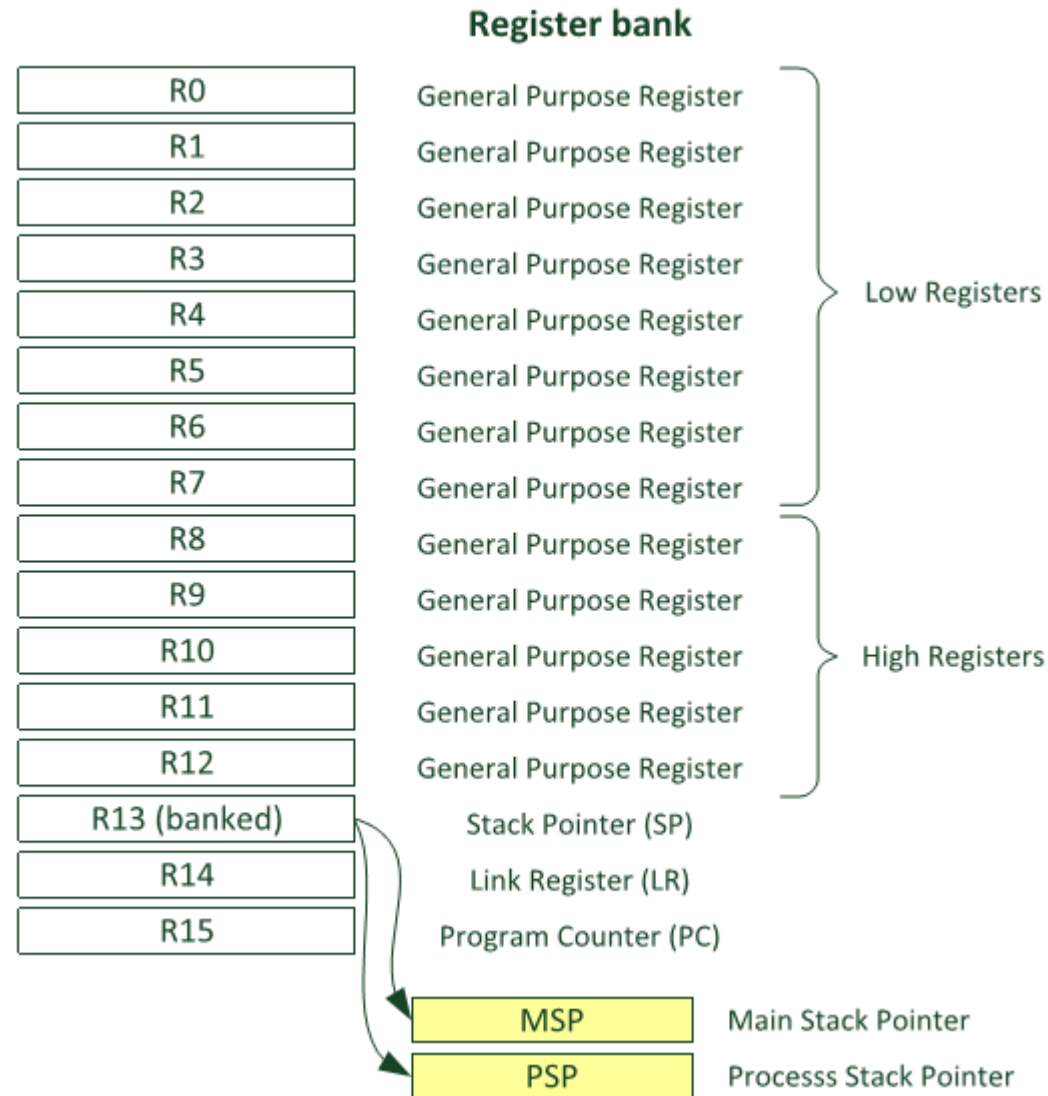
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Modos de operación



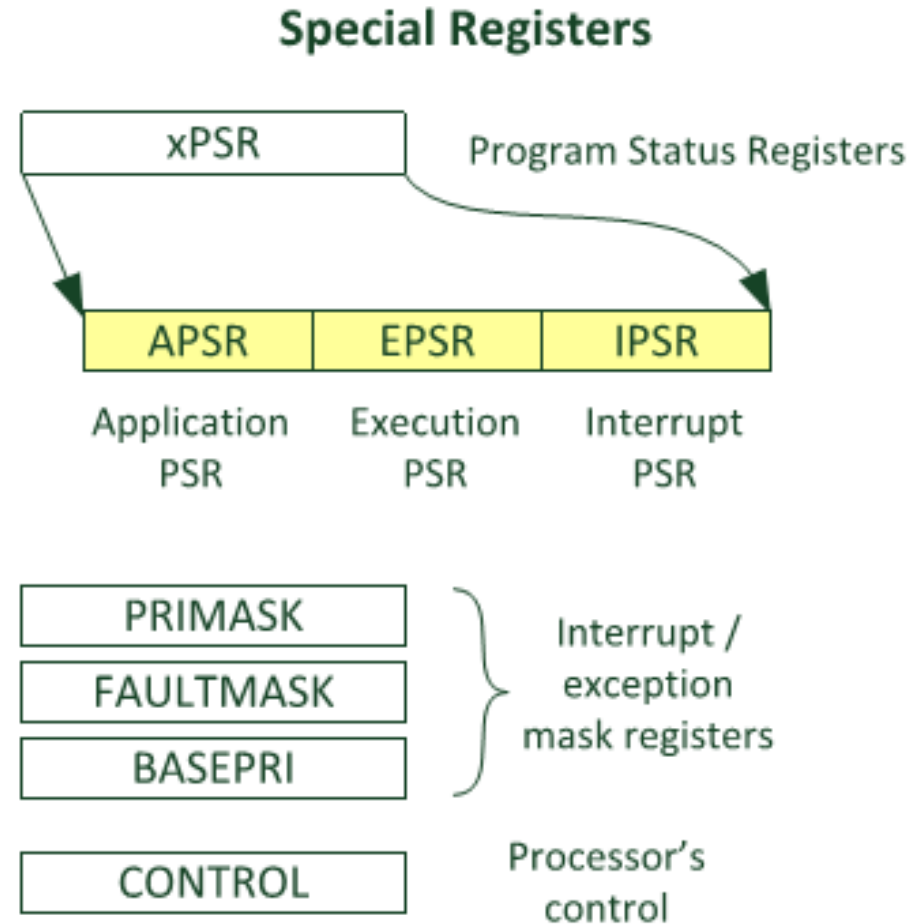
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Registros generales



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Registros especiales



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: PSR

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|--------|----|-------|-------|-------|--------|------------------|---|---|---|-----|
| | 31 | 30 | 29 | 28 | 27 | 26:25 | 24 | 23:20 | 19:16 | 15:10 | 9 | 8 | 7 | 6 | 5 | 4:0 |
| APSR | N | Z | C | V | Q | | | | GE* | | | | | | | |
| IPSR | | | | | | | | | | | | Exception Number | | | | |
| EPSR | | | | | | ICI/IT | T | | | | ICI/IT | | | | | |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

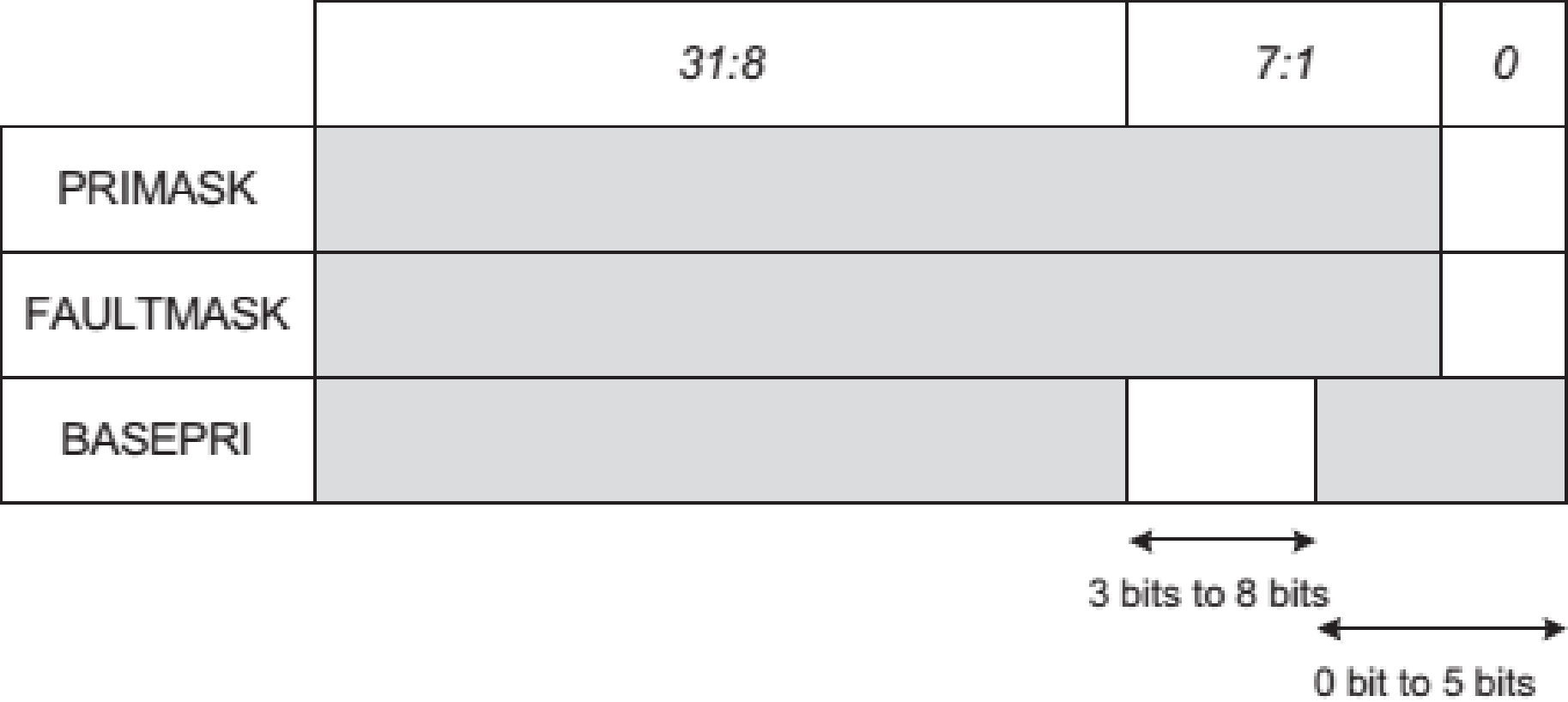
Cortex M3/M4: Ejecución condicional

Ejemplo de ejecución condicional

```
ITTE      NE
ANDNE     r0, r0, r1
ADDNE     r2, r2, #1
MOVEQ     r2, r3
```

¿Qué ventajas me aporta?

Cortex M3/M4: PRIMASK, FAULTMASK, BASEPRI



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Excepciones

| Table 4.9 Exception Types | | | | |
|---------------------------|------------------------|----------------|--------------|--|
| Exception Number | CMSIS Interrupt Number | Exception Type | Priority | Function |
| 1 | — | Reset | −3 (Highest) | Reset |
| 2 | −14 | NMI | −2 | Non-Maskable interrupt |
| 3 | −13 | HardFault | −1 | All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking |
| 4 | −12 | MemManage | Settable | Memory Management fault; caused by MPU violation or invalid accesses (such as an instruction fetch from a non-executable region) |
| 5 | −11 | BusFault | Settable | Error response received from the bus system; caused by an instruction prefetch abort or data access error |
| 6 | −10 | Usage fault | Settable | Usage fault; typical causes are invalid instructions or invalid state transition attempts (such as trying to switch to ARM state in the Cortex-M3) |
| 7–10 | — | — | — | Reserved |
| 11 | −5 | SVC | Settable | Supervisor Call via SVC instruction |
| 12 | −4 | Debug monitor | Settable | Debug monitor – for software based debug (often not used) |
| 13 | — | — | — | Reserved |
| 14 | −2 | PendSV | Settable | Pendable request for System Service |
| 15 | −1 | SYSTICK | Settable | System Tick Timer |
| 16–255 | 0–239 | IRQ | Settable | IRQ input #0–239 |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Instrucciones registros especiales

```
MRS    r0, BASEPRI    ; Read BASEPRI register into R0
MRS    r0, PRIMASK    ; Read PRIMASK register into R0
MRS    r0, FAULTMASK  ; Read FAULTMASK register into R0
MSR    BASEPRI, r0     ; Write R0 into BASEPRI register
MSR    PRIMASK, r0     ; Write R0 into PRIMASK register
MSR    FAULTMASK, r0   ; Write R0 into FAULTMASK register
```

Cortex M3/M4: Instrucciones registros especiales

```
x = __get_BASEPRI();    // Read BASEPRI register
x = __get_PRIMASK();    // Read PRIMASK register
x = __get_FAULTMASK();  // Read FAULTMASK register
__set_BASEPRI(x);        // Set new value for BASEPRI
__set_PRIMASK(x);        // Set new value for PRIMASK
__set_FAULTMASK(x);     // Set new value for FAULTMASK
__disable_irq();         // Set PRIMASK, disable IRQ
__enable_irq();          // Clear PRIMASK, enable IRQ
```

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: CONTROL

Cortex-M3
Cortex-M4

| | | | | |
|---------|------|---|-------|-------|
| | 31:3 | 2 | 1 | 0 |
| CONTROL | | | SPSEL | nPRIV |

Cortex-M4
with FPU

| | | | | |
|---------|------|------|-------|-------|
| | 31:3 | 2 | 1 | 0 |
| CONTROL | | FPCA | SPSEL | nPRIV |

ARMv6-M
(e.g. Cortex-M0)

| | | | | |
|---------|------|---|-------|-------|
| | 31:3 | 2 | 1 | 0 |
| CONTROL | | | SPSEL | nPRIV |

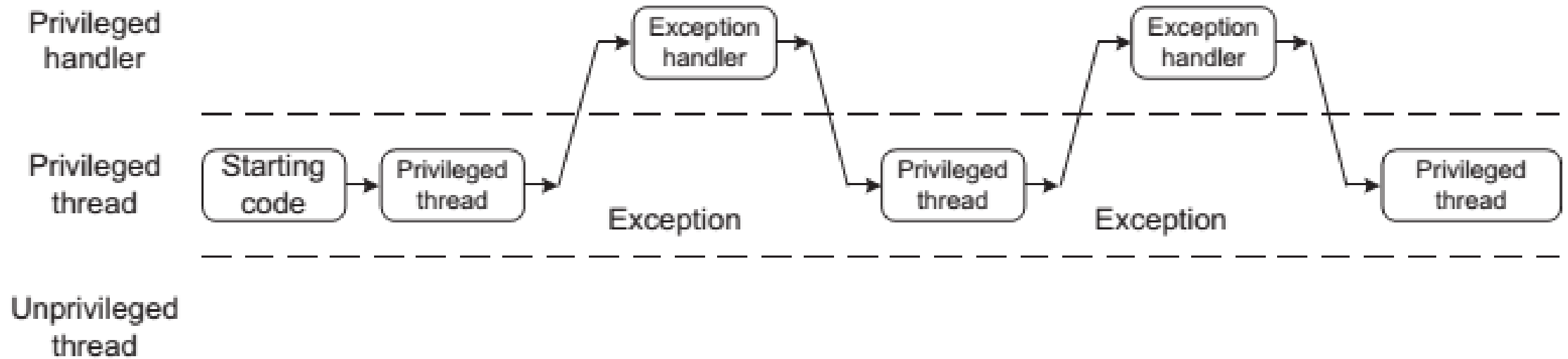
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Acceso al registro de control

```
MRS    r0, CONTROL ; Read CONTROL register into R0  
MSR    CONTROL, r0 ; Write R0 into CONTROL register
```

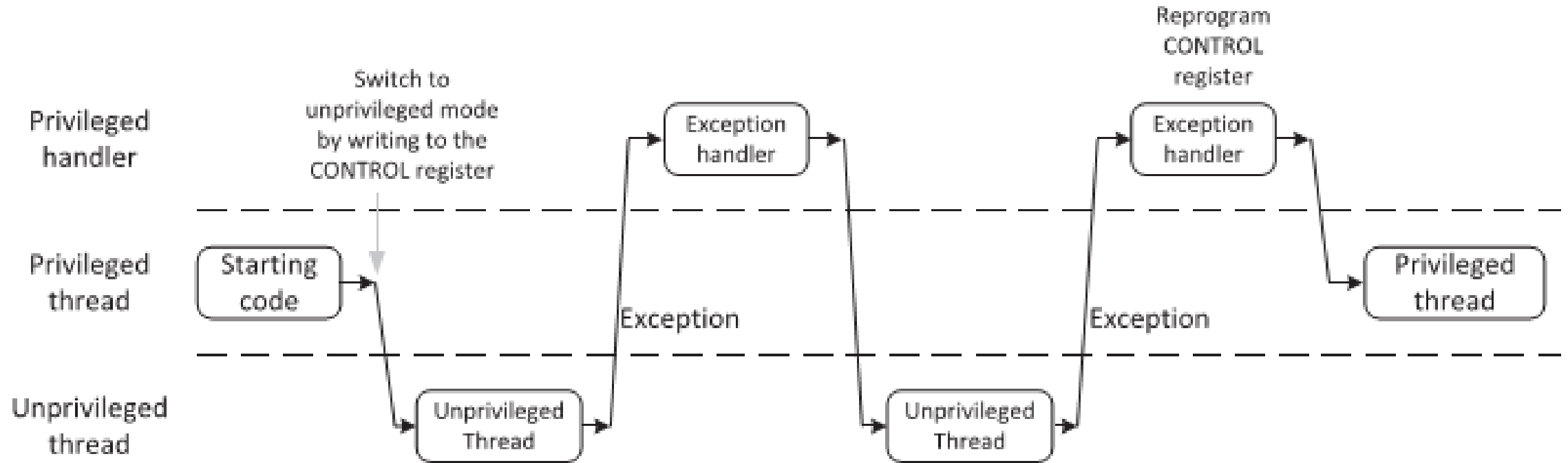
```
x = __get_CONTROL(); // Read the current value of CONTROL  
__set_CONTROL(x);    // Set the CONTROL value to x
```

Cortex M3/M4: Mode switching



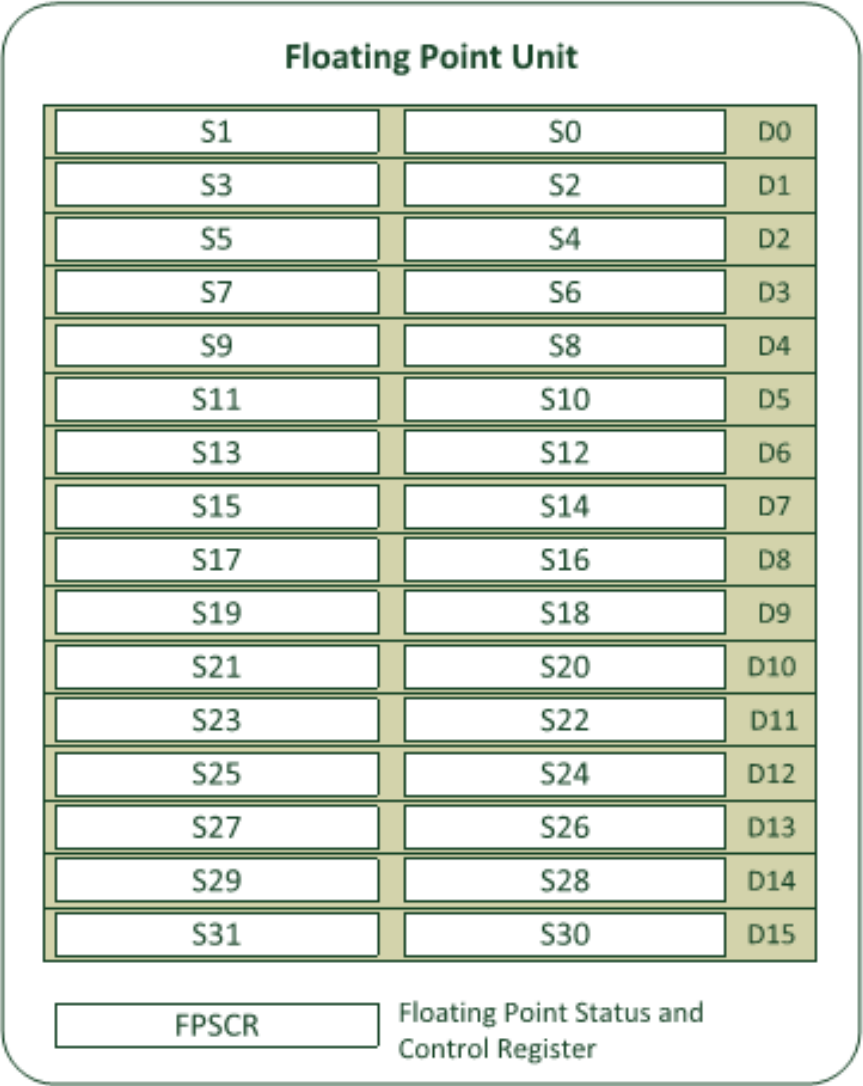
Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: Mode switching



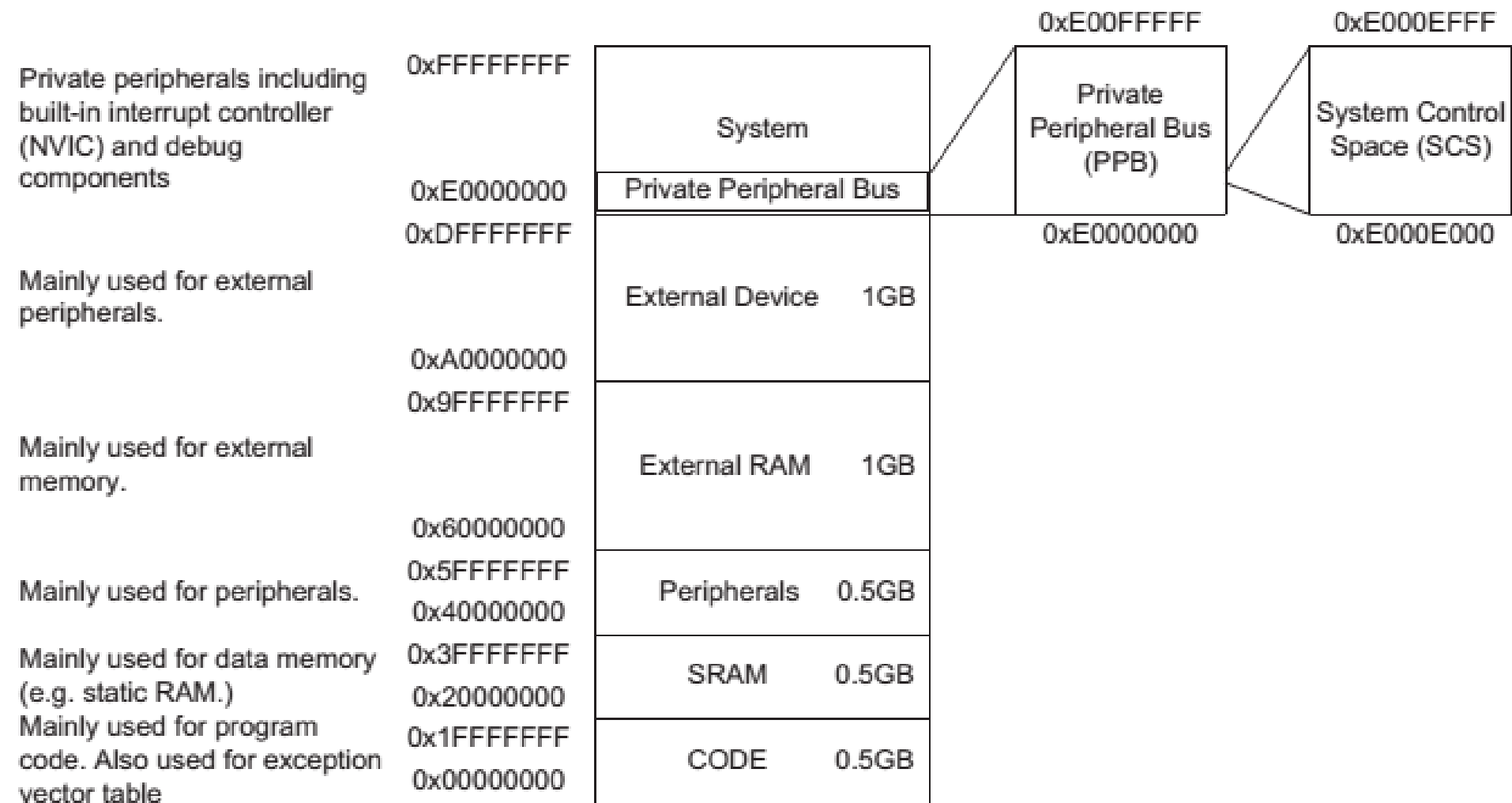
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M4F: FPU



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Mapa de memoria



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

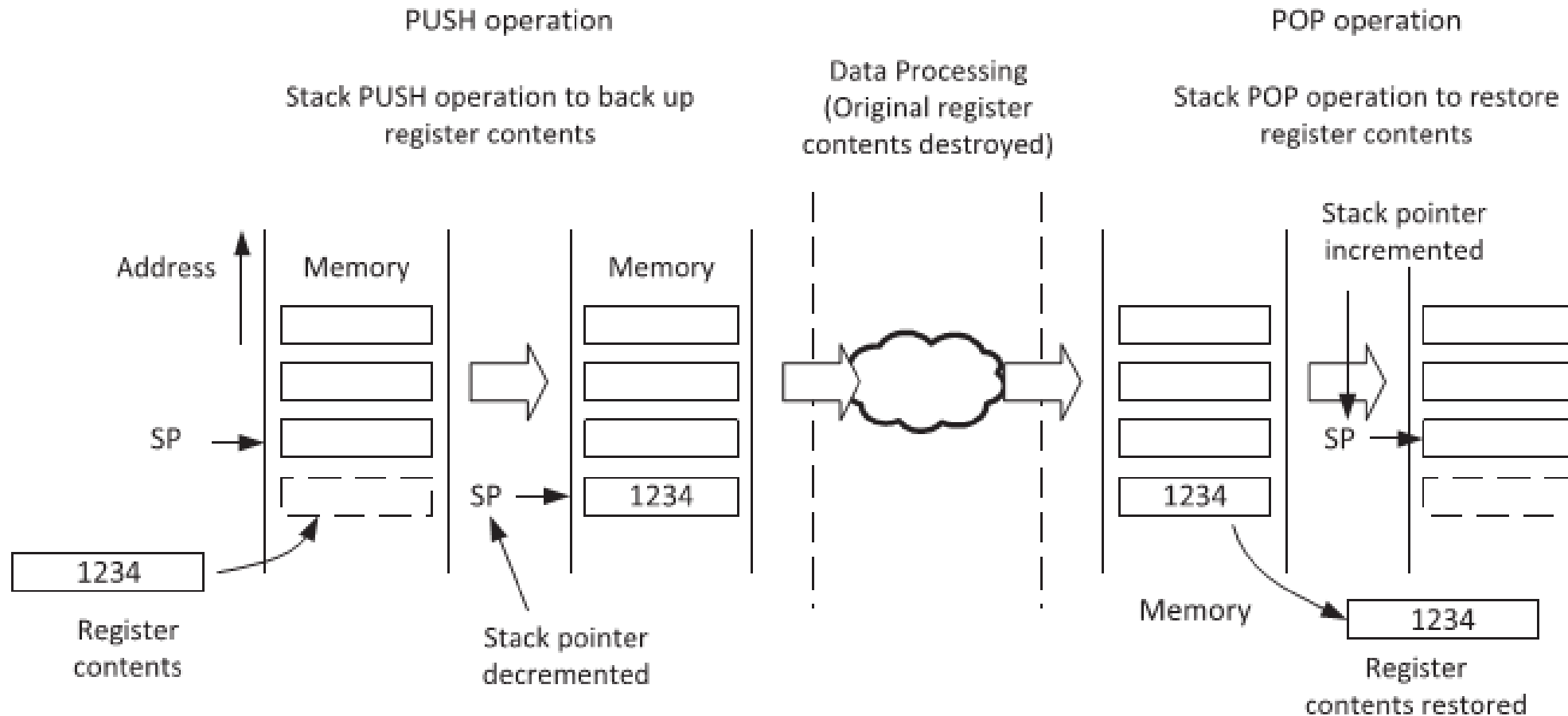
Cortex M3/M4: Stack (la “pila”)

¿Qué funciones cumple la pila?

- Pasar datos a funciones o subrutinas
- Guardar variables locales
- Salvaguardar el estado del procesador y de los registros de propósito general cuando ocurre una interrupción
- Guardar temporalmente el valor de registros previo a su reutilización

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”


Cortex M3/M4: Stacking



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: Stacking (código)

```
...  
; R4 = X, R5 = Y, R6 = Z  
BL    function1  
  
Subroutine  
function1  
    PUSH    {R4} ; store R4 to stack & adjust SP  
    PUSH    {R5} ; store R5 to stack & adjust SP  
    PUSH    {R6} ; store R6 to stack & adjust SP  
    ... ; Executing task (R4, R5 and R6  
        ; could be changed)  
    POP     {R4} ; restore R4 and SP re-adjusted  
    POP     {R5} ; restore R5 and SP re-adjusted  
    POP     {R6} ; restore R6 and SP re-adjusted  
    BX      LR   ; Return  
  
; Back to main program  
; R4 = X, R5 = Y, R6 = Z  
... ; next instructions
```

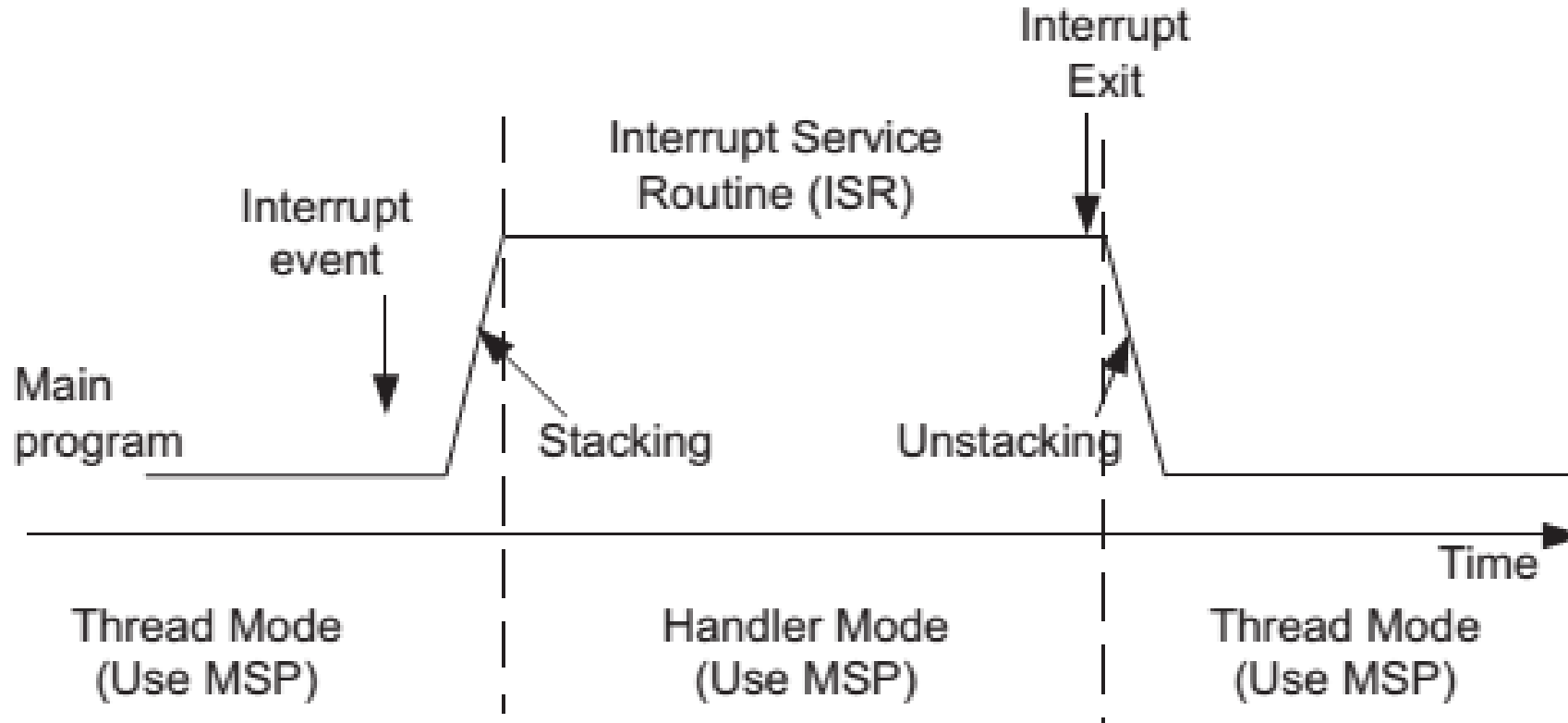


Cortex M3/M4: Stacking (código)

```
...  
; R4 = X, R5 = Y, R6 = Z  
BL    function1  
Subroutine  
function1  
    PUSH    {R4-R6, LR} ; Save registers  
                        ; including link register  
    ... ; Executing task (R4, R5 and R6  
        ; could be changed)  
    POP     {R4-R6, PC} ; Restore registers and  
                        ; return  
; Back to main program  
; R4 = X, R5 = Y, R6 = Z  
... ; next instructions
```

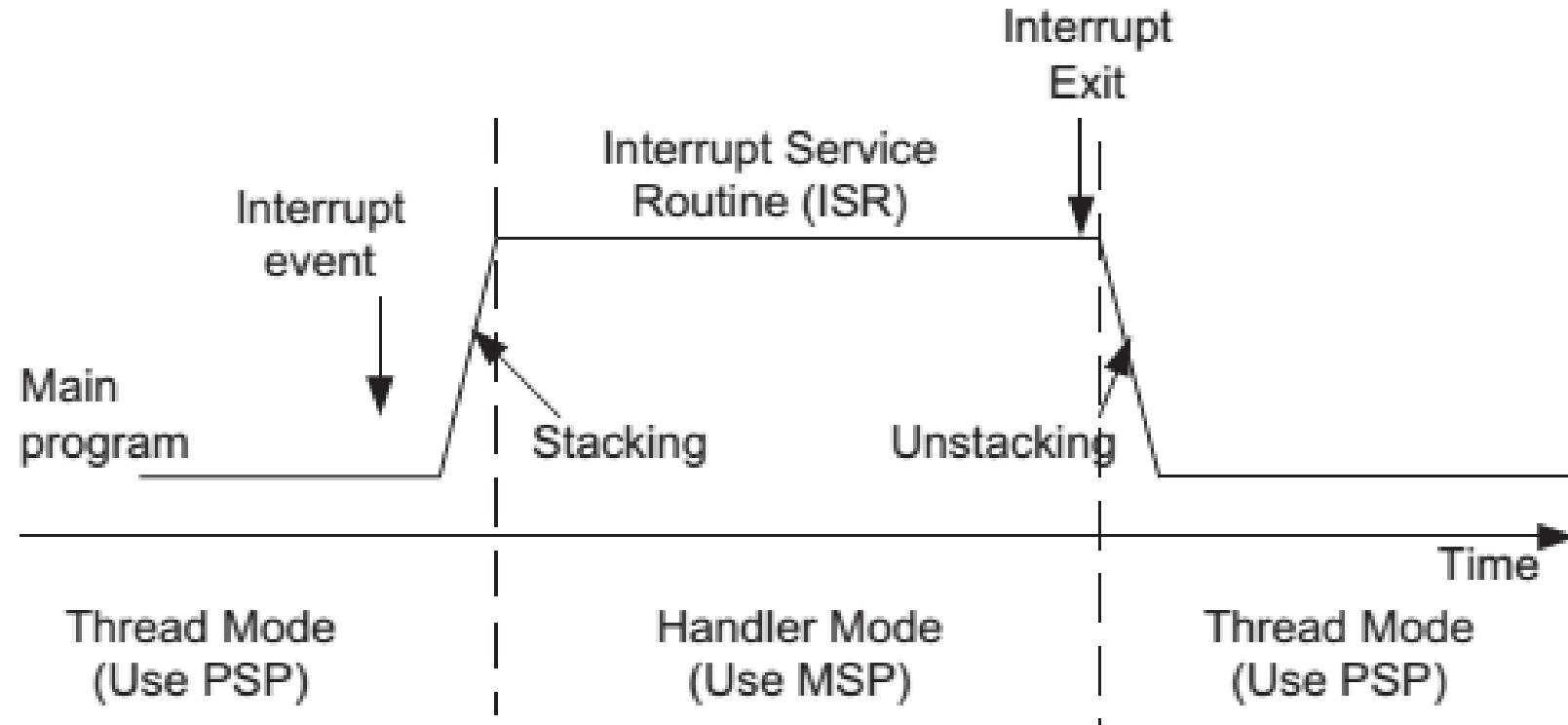
The diagram illustrates the execution flow of a branch with link (BL) instruction. It starts with a main program segment containing a BL instruction to call 'function1'. An arrow points from this instruction to the start of the 'function1' subroutine. Inside the subroutine, the 'PUSH' instruction saves registers R4-R6 and the link register (LR). The subroutine then executes its task. Finally, the 'POP' instruction restores the registers and the PC, and an arrow points back to the instruction following the BL in the main program, indicating the return path.

Cortex M3/M4: MSP, PSP



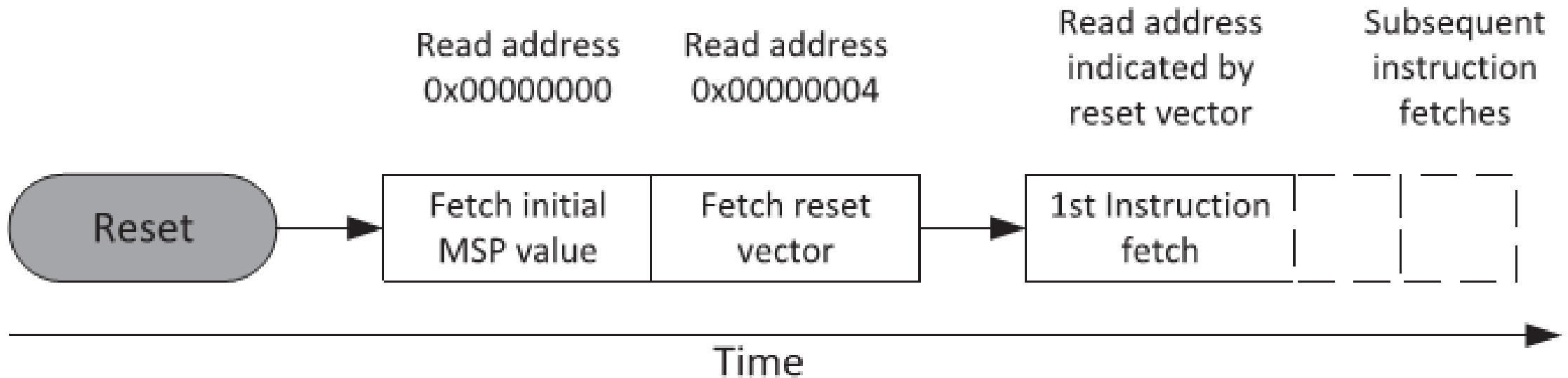
Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: MSP, PSP



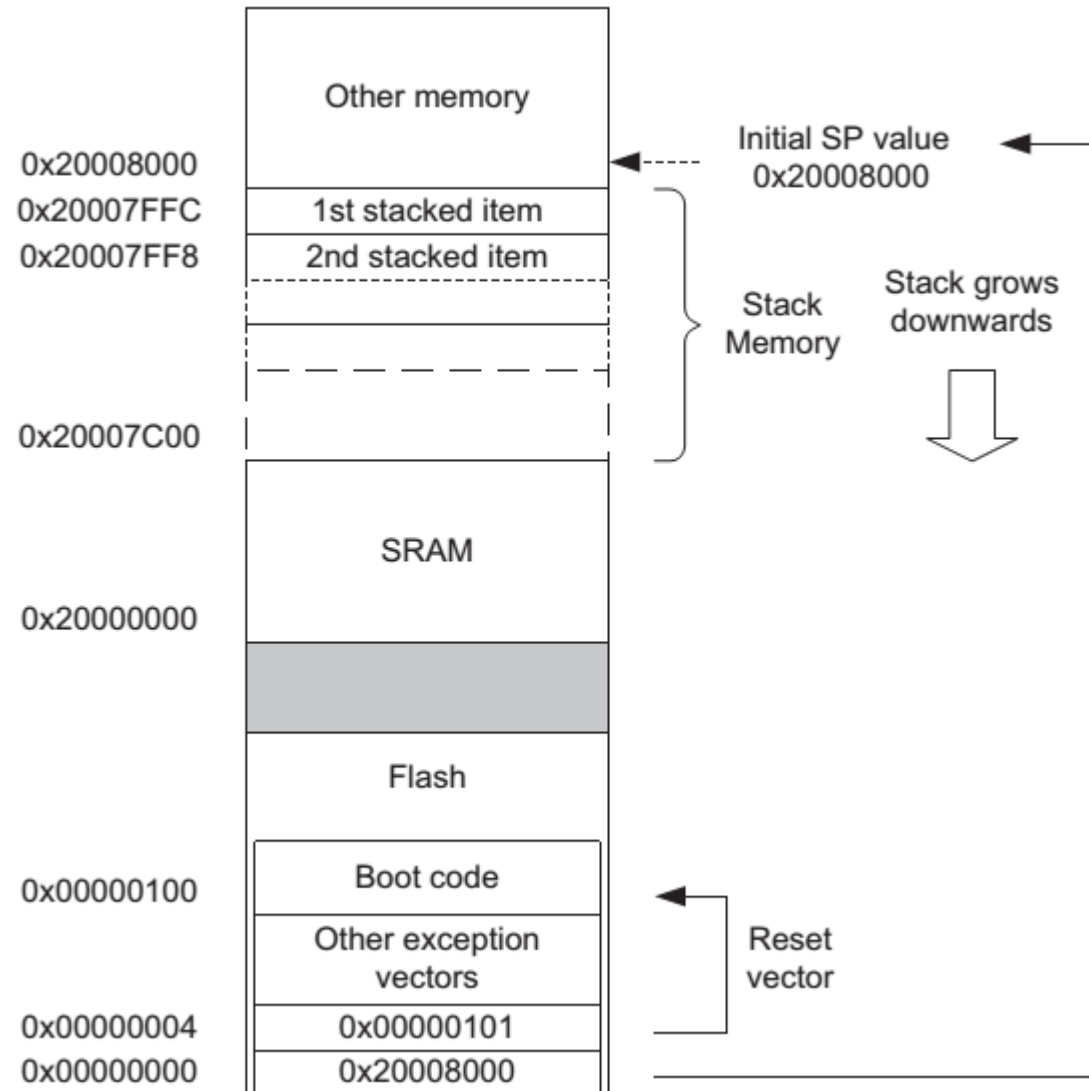
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Secuencia de reset



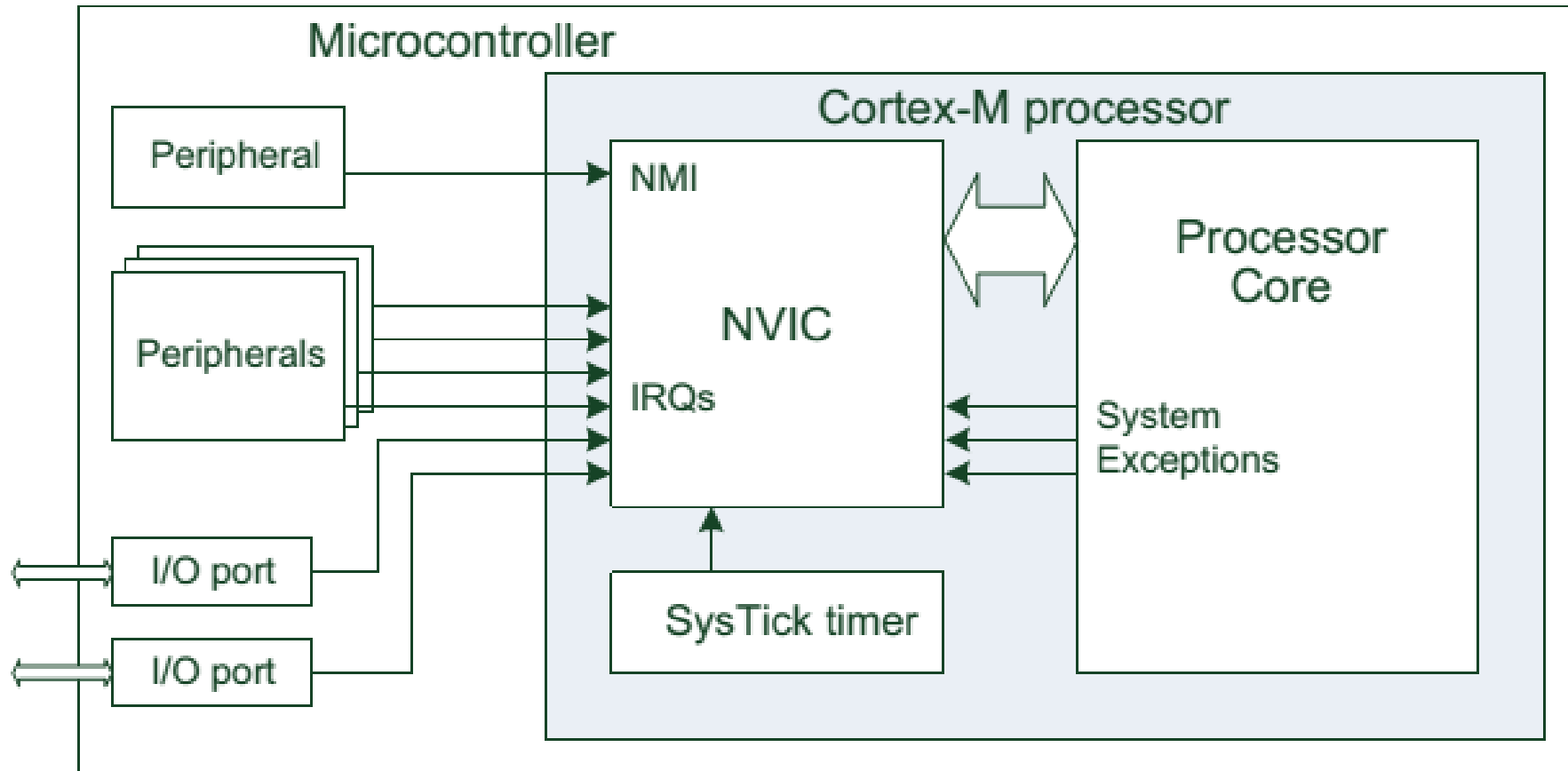
Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: Secuencia de reset



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: Core peripherals



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: NVIC

La familia Cortex M posee un controlador de interrupciones programable denominado “*nested vectored interrupt controller*” (NVIC).

Formalmente, el NVIC puede manejar hasta 255 excepciones (de las cuales las interrupciones son un caso particular):

- Reset (IRQ 1)
- Excepciones del *core* (IRQ2-15)
- Hasta 240 fuentes de interrupción externas (IRQ16-255)

Las primeras tres fuentes de IRQ tienen prioridades fijas y el resto pueden programarse hasta en 128 niveles de prioridad (no necesariamente implementados)

Cortex M3/M4: Excepciones

| Table 4.9 Exception Types | | | | |
|---------------------------|------------------------|----------------|--------------|--|
| Exception Number | CMSIS Interrupt Number | Exception Type | Priority | Function |
| 1 | — | Reset | −3 (Highest) | Reset |
| 2 | −14 | NMI | −2 | Non-Maskable interrupt |
| 3 | −13 | HardFault | −1 | All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking |
| 4 | −12 | MemManage | Settable | Memory Management fault; caused by MPU violation or invalid accesses (such as an instruction fetch from a non-executable region) |
| 5 | −11 | BusFault | Settable | Error response received from the bus system; caused by an instruction prefetch abort or data access error |
| 6 | −10 | Usage fault | Settable | Usage fault; typical causes are invalid instructions or invalid state transition attempts (such as trying to switch to ARM state in the Cortex-M3) |
| 7–10 | — | — | — | Reserved |
| 11 | −5 | SVC | Settable | Supervisor Call via SVC instruction |
| 12 | −4 | Debug monitor | Settable | Debug monitor – for software based debug (often not used) |
| 13 | — | — | — | Reserved |
| 14 | −2 | PendSV | Settable | Pendable request for System Service |
| 15 | −1 | SYSTICK | Settable | System Tick Timer |
| 16–255 | 0–239 | IRQ | Settable | IRQ input #0–239 |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Prioridades

| | | | | | | | |
|-------------|-------|-------|-----------------|-------|-------|-------|-------|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Implemented | | | Not Implemented | | | | |

| | | | | | | | |
|-------------|-------|-------|-------|-----------------|-------|-------|-------|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Implemented | | | | Not Implemented | | | |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Grupos de prioridades

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------------|-------|--------------|-----------------|-------|-------|-------|-------|
| Preempt priority | | Sub-priority | Not Implemented | | | | |

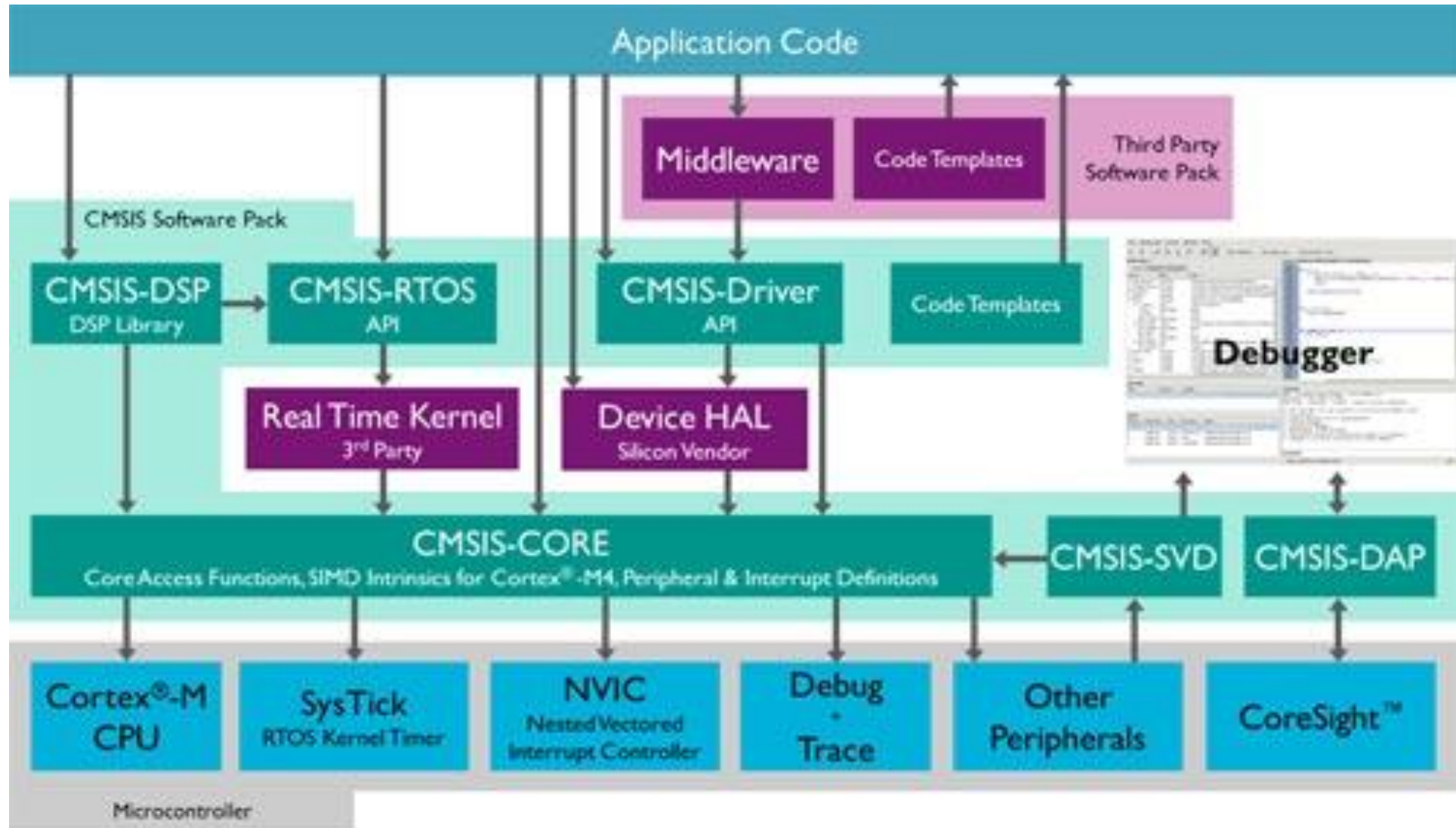
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: CMSIS



“The ARM® Cortex® Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the [Cortex-M processor](#) series and specifies debugger interfaces. Creation of software is a major cost factor in the embedded industry. By standardizing the software interfaces across all Cortex-M silicon vendor products, especially when creating new projects or migrating existing software to a new device, means significant cost reductions. The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware. It simplifies software re-use, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices.”

Cortex M3/M4: CMSIS



Fuente: <http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php>

Cortex M3/M4: Manejo de IRQs usando CMSIS

```
typedef enum IRQn
{
    /***** Cortex-M3 Processor Exceptions Numbers *****/
    NonMaskableInt_IRQn      = -14,    /*!< 2 Non Maskable Interrupt          */
    MemoryManagement_IRQn    = -12,    /*!< 4 Cortex-M3 Memory Management Interrupt */
    BusFault_IRQn            = -11,    /*!< 5 Cortex-M3 Bus Fault Interrupt      */
    UsageFault_IRQn          = -10,    /*!< 6 Cortex-M3 Usage Fault Interrupt    */
    SVCall_IRQn              = -5,     /*!< 11 Cortex-M3 SV Call Interrupt      */
    DebugMonitor_IRQn        = -4,     /*!< 12 Cortex-M3 Debug Monitor Interrupt */
    PendSV_IRQn              = -2,     /*!< 14 Cortex-M3 Pend SV Interrupt      */
    SysTick_IRQn             = -1,     /*!< 15 Cortex-M3 System Tick Interrupt   */
    /***** LPC17xx Specific Interrupt Numbers *****/
    WDT_IRQn                 = 0,      /*!< Watchdog Timer Interrupt          */
    TIMER0_IRQn              = 1,      /*!< Timer0 Interrupt                  */
    TIMER1_IRQn              = 2,      /*!< Timer1 Interrupt                  */
    TIMER2_IRQn              = 3,      /*!< Timer2 Interrupt                  */
    TIMER3_IRQn              = 4,      /*!< Timer3 Interrupt                  */
    UART0_IRQn               = 5,      /*!< UART0 Interrupt                   */
    UART1_IRQn               = 6,      /*!< UART1 Interrupt                   */
    UART2_IRQn               = 7,      /*!< UART2 Interrupt                   */
    UART3_IRQn               = 8,      /*!< UART3 Interrupt                   */
    PWM1_IRQn                = 9,      /*!< PWM1 Interrupt                    */
    I2C0_IRQn                = 10,     /*!< I2C0 Interrupt                    */
    I2C1_IRQn                = 11,     /*!< I2C1 Interrupt                    */
    I2C2_IRQn                = 12,     /*!< I2C2 Interrupt                    */
    SPI_IRQn                 = 13,     /*!< SPI Interrupt                     */
    SSP0_IRQn                = 14,     /*!< SSP0 Interrupt                    */
    SSP1_IRQn                = 15,     /*!< SSP1 Interrupt                    */
    PLL0_IRQn                = 16,     /*!< PLL0 Lock (Main PLL) Interrupt    */
    RTC_IRQn                 = 17,     /*!< Real Time Clock Interrupt         */
    EINT0_IRQn               = 18,     /*!< External Interrupt 0 Interrupt     */
    EINT1_IRQn               = 19,     /*!< External Interrupt 1 Interrupt     */
}
```

Fuente: https://www.keil.com/pack/doc/cmsis/Core/html/group___n_v_i_c__gr.html

Cortex M3/M4: Habilitar/Deshabilitar IRQ (general)

void __enable_irq (void)

The function enables interrupts and all configurable fault handlers by clearing PRIMASK. The function uses the instruction **CPSIE i**.

Remarks

- Can be executed in privileged mode only.

See Also

- `__disable_irq`; `__set_BASEPRI`; `__set_CONTROL`; `__set_PRIMASK`

void __disable_irq (void)

The function disables interrupts and all configurable fault handlers by setting PRIMASK. The function uses the instruction **CPSID i**.

Remarks

- Can be executed in privileged mode only.
- An interrupt can enter pending state even if it is disabled. Disabling an interrupt only prevents the processor from taking that interrupt.

See Also

- `__enable_irq`; `__set_BASEPRI`; `__set_CONTROL`; `__set_PRIMASK`

Cortex M3/M4: Habilitar IRQ (particular)

```
void NVIC_EnableIRQ ( IRQn_Type IRQn )
```

This function enables the specified device-specific interrupt *IRQn*. *IRQn* cannot be a negative value.

Parameters

[in] **IRQn** Interrupt number

Remarks

- The registers that control the enabling and disabling of interrupts are called SETENA and CLRENA.
- The number of supported interrupts depends on the implementation of the chip designer and can be read from the Interrupt Controller Type Register (ICTR) in granularities of 32:

ICTR[4:0]

- =0 - 32 interrupts supported
- =1 - 64 interrupts supported
- ...

See Also

- [NVIC_DisableIRQ](#); [SCnSCB_Type](#);
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Deshabilitar IRQ (particular)

```
void NVIC_DisableIRQ ( IRQn_Type IRQn )
```

This function disables the specified device-specific interrupt *IRQn*. *IRQn* cannot be a negative value.

Parameters

[in] **IRQn** Number of the external interrupt to disable

Remarks

- The registers that control the enabling and disabling of interrupts are called SETENA and CLRENA.

See Also

- [NVIC_EnableIRQ](#)
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Setear prioridad IRQ

```
void NVIC_SetPriority ( IRQn_Type IRQn,  
                      uint32_t  priority  
                      )
```

Sets the priority for the interrupt specified by *IRQn*. *IRQn* can specify any device-specific (external) interrupt, or core (internal) interrupt. The *priority* specifies the interrupt priority value, whereby lower values indicate a higher priority. The default priority is 0 for every interrupt. This is the highest possible priority.

The priority cannot be set for every core interrupt. HardFault and NMI have a fixed (negative) priority that is higher than any configurable exception or interrupt.

Parameters

[in] **IRQn** Interrupt Number
[in] **priority** Priority to set

Remarks

- The number of priority levels is configurable and depends on the implementation of the chip designer. To determine the number of bits implemented for interrupt priority-level registers, write *0xFF* to one of the priority-level register, then read back the value. For example, if the minimum number of 3 bits have been implemented, the read-back value is *0xE0*.
- Writes to unimplemented bits are ignored.
- **For Cortex-M0:**
 - Dynamic switching of interrupt priority levels is not supported. The priority level of an interrupt should not be changed after it has been enabled.
 - Supports 0 to 192 priority levels.
 - Priority-level registers are 2 bit wide, occupying the two MSBs. Each Interrupt Priority Level Register is 1-byte wide.
- **For Cortex-M3, Cortex-M4, and Cortex-M7:**
 - Dynamic switching of interrupt priority levels is supported.
 - Supports 0 to 255 priority levels.
 - Priority-level registers have a maximum width of 8 bits and a minimum of 3 bits. Each register can be further divided into preempt priority level and subpriority level.

See Also

- [NVIC_GetPriority](#); [NVIC_SetPriorityGrouping](#); [__set_BASEPRI](#);
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Setear grupos de prioridad

```
void NVIC_SetPriorityGrouping ( uint32_t PriorityGroup )
```

The function sets the priority grouping *PriorityGroup* using the required unlock sequence. *PriorityGroup* is assigned to the field PRIGROUP (register AIRCR[10:8]). This field determines the split of group priority from subpriority. Only values from 0..7 are used. In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS), the smallest possible priority group is set.

Parameters

[in] **PriorityGroup** Priority group

Remarks

- not for Cortex-M0, Cortex-M0+, or SC000.
- By default, priority group setting is zero.

See Also

- [NVIC_GetPriorityGrouping](#); [NVIC_SetPriority](#); [SCB_Type](#)
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Obtener grupos de prioridad

uint32_t NVIC_GetPriorityGrouping (void)

This function returns the priority grouping (flag PRIGROUP in AIRCR[10:8]).

Returns

Priority grouping field

Remarks

- not for Cortex-M0, Cortex-M0+, or SC000.
- By default, priority group setting is zero.

See Also

- [NVIC_SetPriorityGrouping](#); [NVIC_GetPriority](#); [SCB_Type](#)
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Codificar prioridad IRQ

```
uint32_t NVIC_EncodePriority ( uint32_t PriorityGroup,  
                             uint32_t PreemptPriority,  
                             uint32_t SubPriority  
                             )
```

This function encodes the priority for an interrupt with the priority group *PriorityGroup*, preemptive priority value *PreemptPriority*, and subpriority value *SubPriority*. In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`) the smallest possible priority group is set.

Parameters

- [in] **PriorityGroup** Priority group
- [in] **PreemptPriority** Preemptive priority value (starting from 0)
- [in] **SubPriority** Subpriority value (starting from 0)

Returns

Encoded priority for the interrupt

Remarks

- not for Cortex-M0, Cortex-M0+, or SC000.

See Also

- [NVIC_DecodePriority](#); [NVIC_SetPriority](#);
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Decodificar prioridad IRQ

```
void NVIC_DecodePriority ( uint32_t  Priority,  
                          uint32_t  PriorityGroup,  
                          uint32_t * pPreemptPriority,  
                          uint32_t * pSubPriority  
                        )
```

This function decodes an interrupt priority value with the priority group *PriorityGroup* to preemptive priority value *pPreemptPriority* and subpriority value *pSubPriority*. In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`) the smallest possible priority group is set.

Parameters

| | | |
|-------|--------------------------|---|
| [in] | Priority | Priority |
| [in] | PriorityGroup | Priority group |
| [out] | *pPreemptPriority | Preemptive priority value (starting from 0) |
| [out] | *pSubPriority | Subpriority value (starting from 0) |

Remarks

- not for Cortex-M0, Cortex-M0+, or SC000.

See Also

- [NVIC_EncodePriority](#); [NVIC_GetPriority](#); [NVIC_GetPriorityGrouping](#);
- [Cortex-M Reference Manuals](#)

Cortex M3/M4: Vector de interrupciones

| Memory Address | | Exception Number |
|----------------|----------------------|------------------|
| | | |
| | | |
| | | |
| | | |
| 0x0000004C | Interrupt#3 vector | 19 |
| 0x00000048 | Interrupt#2 vector | 18 |
| 0x00000044 | Interrupt#1 vector | 17 |
| 0x00000040 | Interrupt#0 vector | 16 |
| 0x0000003C | SysTick vector | 15 |
| 0x00000038 | PendSV vector | 14 |
| 0x00000034 | Not used | 13 |
| 0x00000030 | Debug Monitor vector | 12 |
| 0x0000002C | SVC vector | 11 |
| 0x00000028 | Not used | 10 |
| 0x00000024 | Not used | 9 |
| 0x00000020 | Not used | 8 |
| 0x0000001C | Not used | 7 |
| 0x00000018 | Usage Fault vector | 6 |
| 0x00000014 | Bus Fault vector | 5 |
| 0x00000010 | MemManage vector | 4 |
| 0x0000000C | HardFault vector | 3 |
| 0x00000008 | NMI vector | 2 |
| 0x00000004 | Reset vector | 1 |
| 0x00000000 | MSP initial value | 0 |

Note : LSB of each vector must be set to 1 to indicate Thumb state

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Vector de interrupciones

Caso de aplicación: EDU-CIAA-NXP (LPC 4337, M4F+M0)

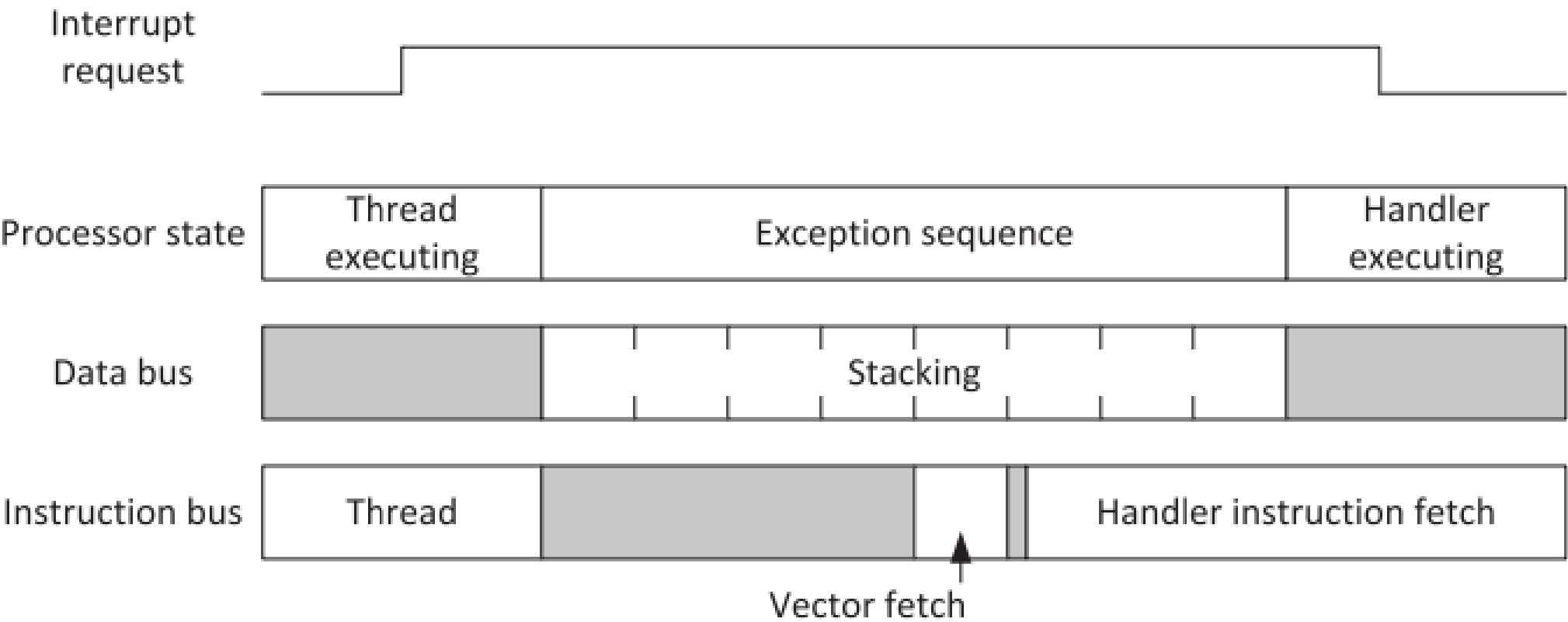
El vector de interrupciones está definido en:

modules/lpc4337_m4/base/src/cr_startup_lpc43xx.c

Las definiciones del CMSIS están contenidas en:

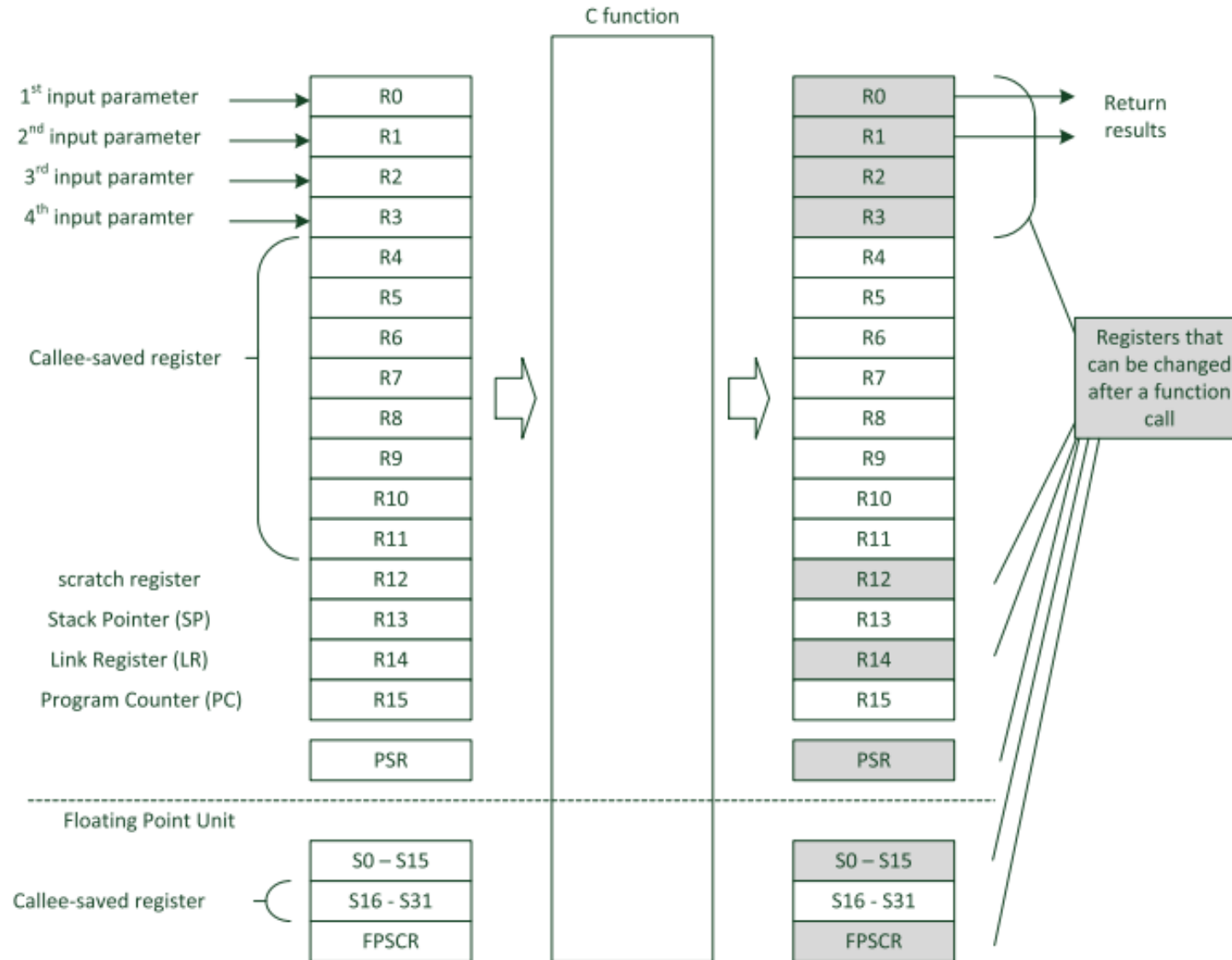
modules/lpc4337_m4/chip/inc/core_cm4.h

Cortex M3/M4: Stacking



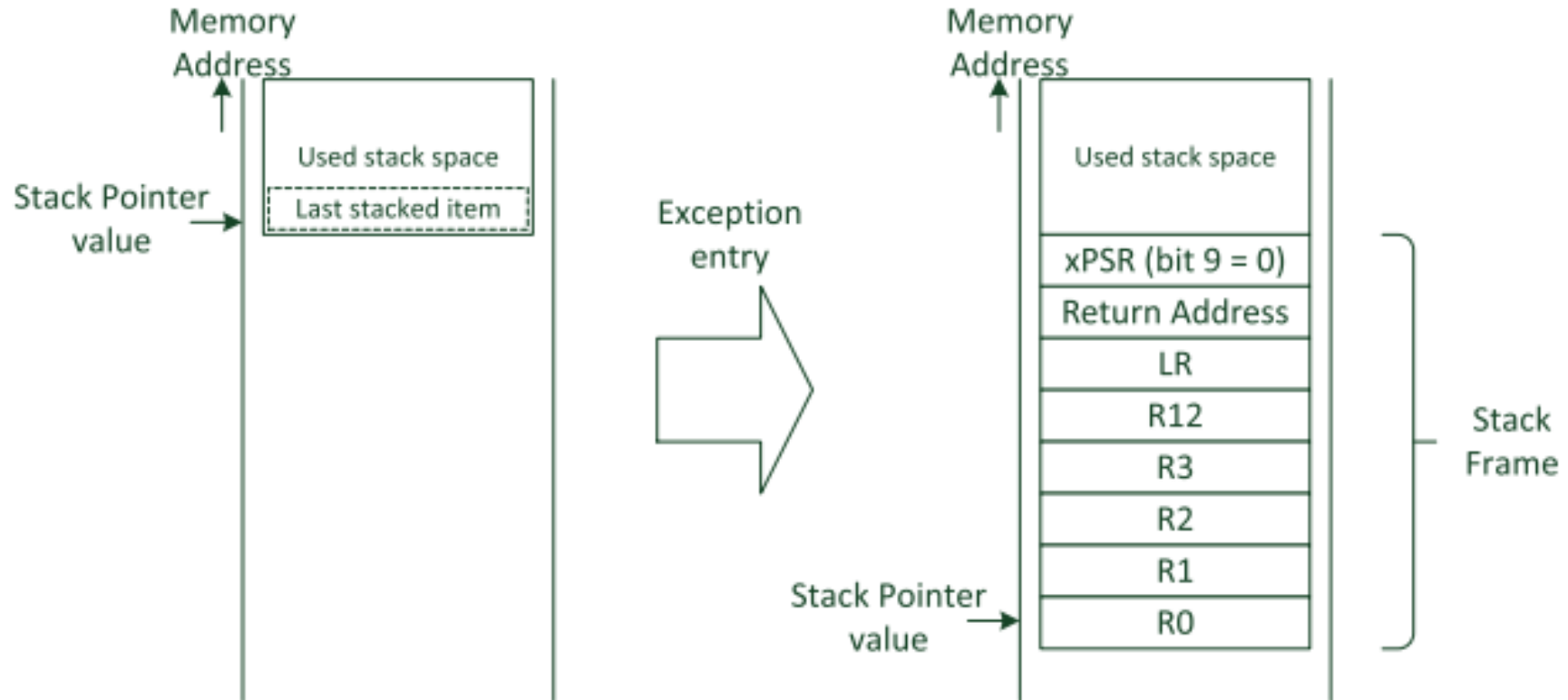
Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Stacking

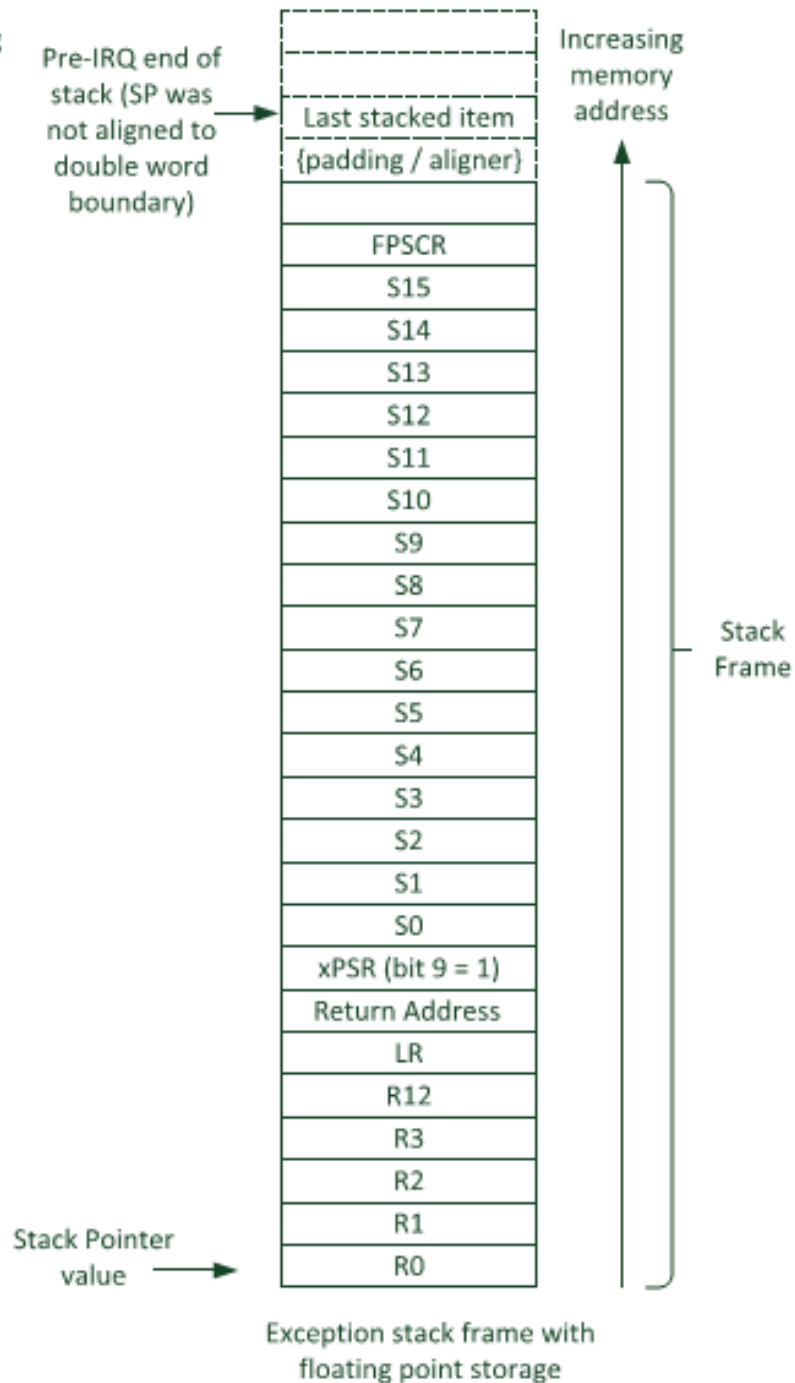
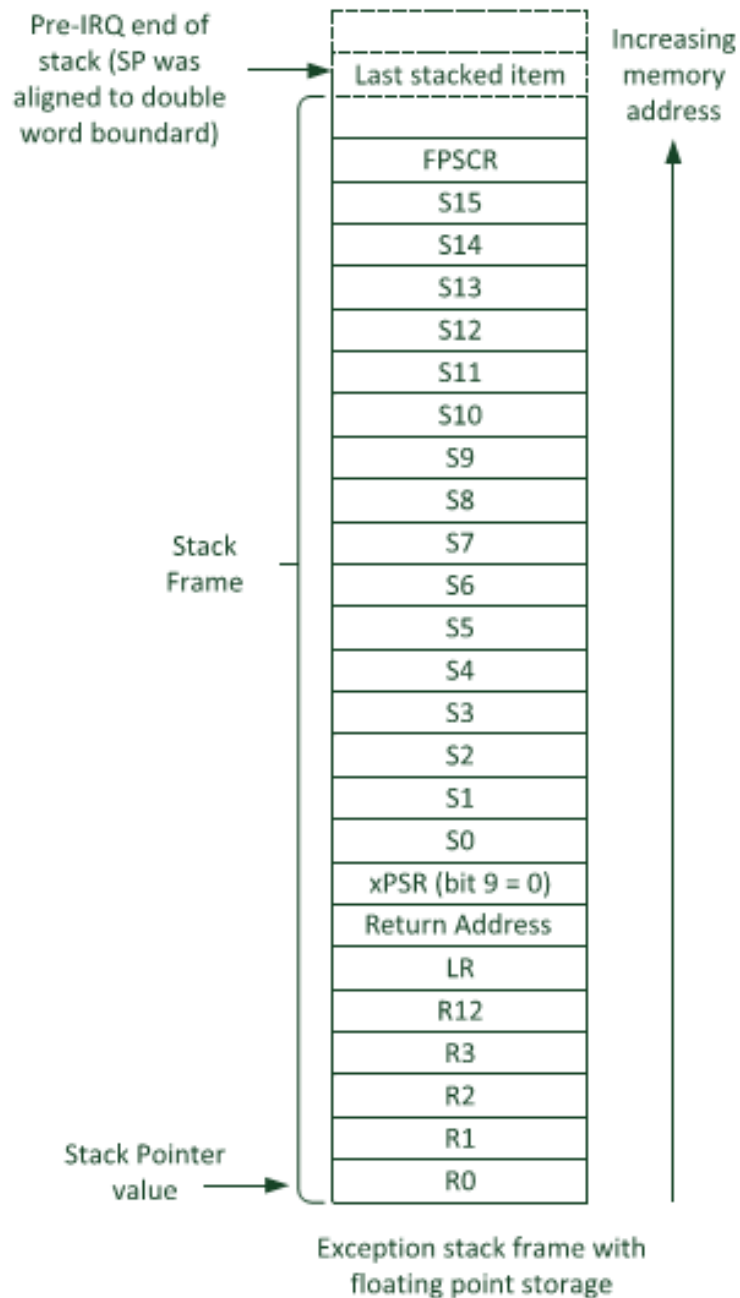


Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: Stacking



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"



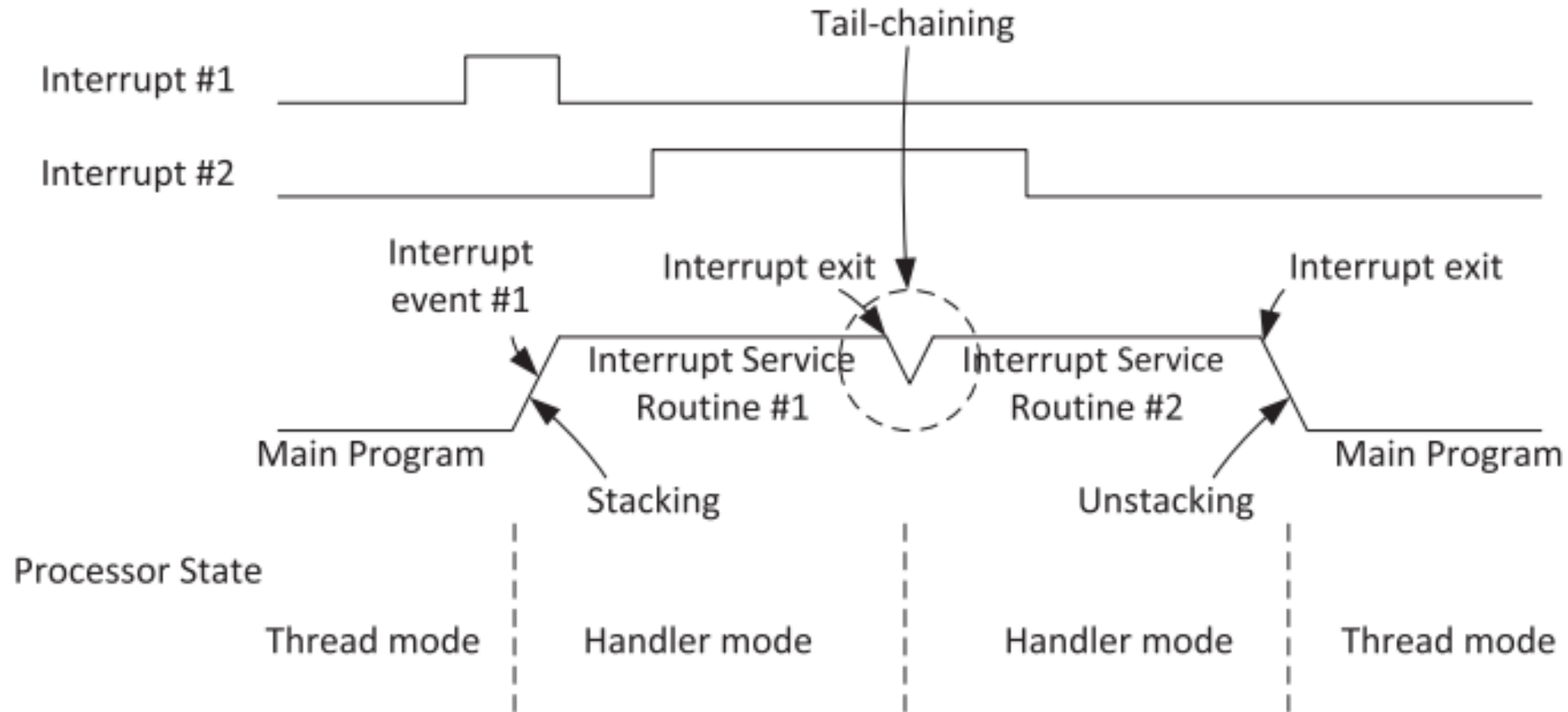
Cortex M3/M4: LR=EXEC_RETURN

| Bits | Descriptions | Values |
|-------|----------------------|---|
| 31:28 | EXC_RETURN indicator | 0xF |
| 27:5 | Reserved (all 1) | 0xEFFFFFFF (23 bits of 1) |
| 4 | Stack Frame type | 1 (8 words) or 0 (26 words). Always 1 when the floating unit is unavailable. This value is set to the inverted value of FPCA bit in the CONTROL register when entering an exception handler. |
| 3 | Return mode | 1 (Return to Thread) or 0 (Return to Handler) |
| 2 | Return stack | 1 (Return with Process Stack) or 0 (Return with Main Stack) |
| 1 | Reserved | 0 |
| 0 | Reserved | 1 |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Tail chaining

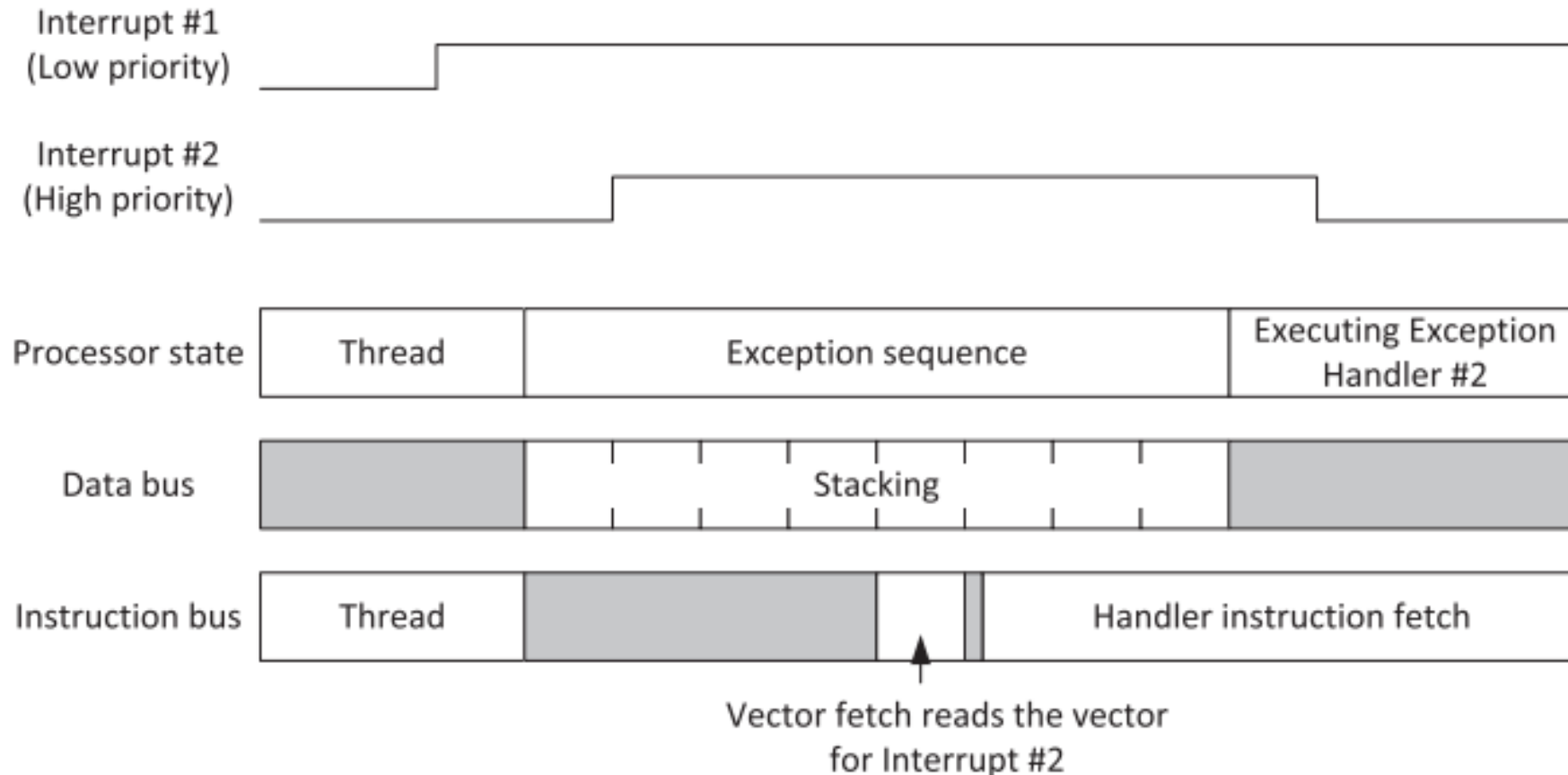
Cuando ocurren dos interrupciones anidadas de igual prioridad...



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: Late arrival

Cuando se está conmutando a una IRQ y ocurre otra de mayor prioridad



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: SysTick

- Es un contador descendente de 24 bits ligado a la arquitectura CM3
- Puede utilizarse para generar la excepción SysTick (#15)
- Puede sincronizarse con el reloj del procesador o por medio de una fuente externa (generalmente dentro del chip)

Cortex M3/M4: Configuración del systick (CMSIS)

uint32_t SysTick_Config (uint32_t ticks)

Initialises and starts the System Tick Timer and its interrupt. After this call, the SysTick timer creates interrupts with the specified time interval. Counter is in free running mode to generate periodical interrupts.

Parameters

[in] **ticks** Number of ticks between two interrupts

Returns

0 - success

1 - failure

Note

When **#define __Vendor_SysTickConfig** is set to 1, the standard function **SysTick_Config** is excluded. In this case, the file **device.h** must contain a vendor specific implementation of this function.

Cortex M3/M4: SysTick (ejemplo de configuración)

```
#include "LPC17xx.h"

uint32_t msTicks = 0; /* Variable to store millisecond ticks */

void SysTick_Handler(void)
{ /* SysTick interrupt Handler.
    msTicks++;
}

int main (void)
{
uint32_t returnCode;
/* Configure SysTick to generate an interrupt every millisecond */

returnCode = SysTick_Config(SystemCoreClock / 1000);
if (returnCode != 0) { /* Check return code for errors */
// Error Handling
}
while(1);
}
```

Cortex M3/M4: MPU

La unidad de protección de memoria (MPU=Memory Protection Unit) es un módulo opcional que tiene como función proteger el acceso a la memoria.

- Puede gestionar hasta 8 regiones de memoria (más una región de background)

Entre sus funciones:

- Prevenir el acceso de aplicaciones a zonas de memoria (stacks) de otras aplicaciones o del kernel
- Prevenir el acceso de aplicaciones a periféricos sin los permisos adecuados
- Evitar la ejecución de código desde zonas no permitidas (por ejemplo, RAM).

Cortex M3/M4: MPU

Si los permisos de la zona de memoria accedida son violados, puede ocurrir:

- Excepción HardFault
- Excepción MemManage

De acuerdo a la configuración vigente.

La MPU está por defecto deshabilitada. Si se la quiere utilizar se deberá configurar manualmente para el sistema en cuestión.

Cortex M3/M4: MPU->TYPE

Es un registro de lectura, permite averiguar si hay una MPU presente y sus características.

| Bits | Name | Type | Reset Value | Description |
|-------|----------|------|-------------|--|
| 23:16 | IREGION | R | 0 | Number of instruction regions supported by this MPU; because ARMv7-M architecture uses a unified MPU, this is always 0. |
| 15:8 | DREGION | R | 0 or 8 | Number of regions supported by this MPU; in the Cortex®-M3 and Cortex-M4 processors this is either 0 (MPU not present) or 8 (MPU present). |
| 0 | SEPARATE | R | 0 | This is always 0 as the MPU is unified. |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

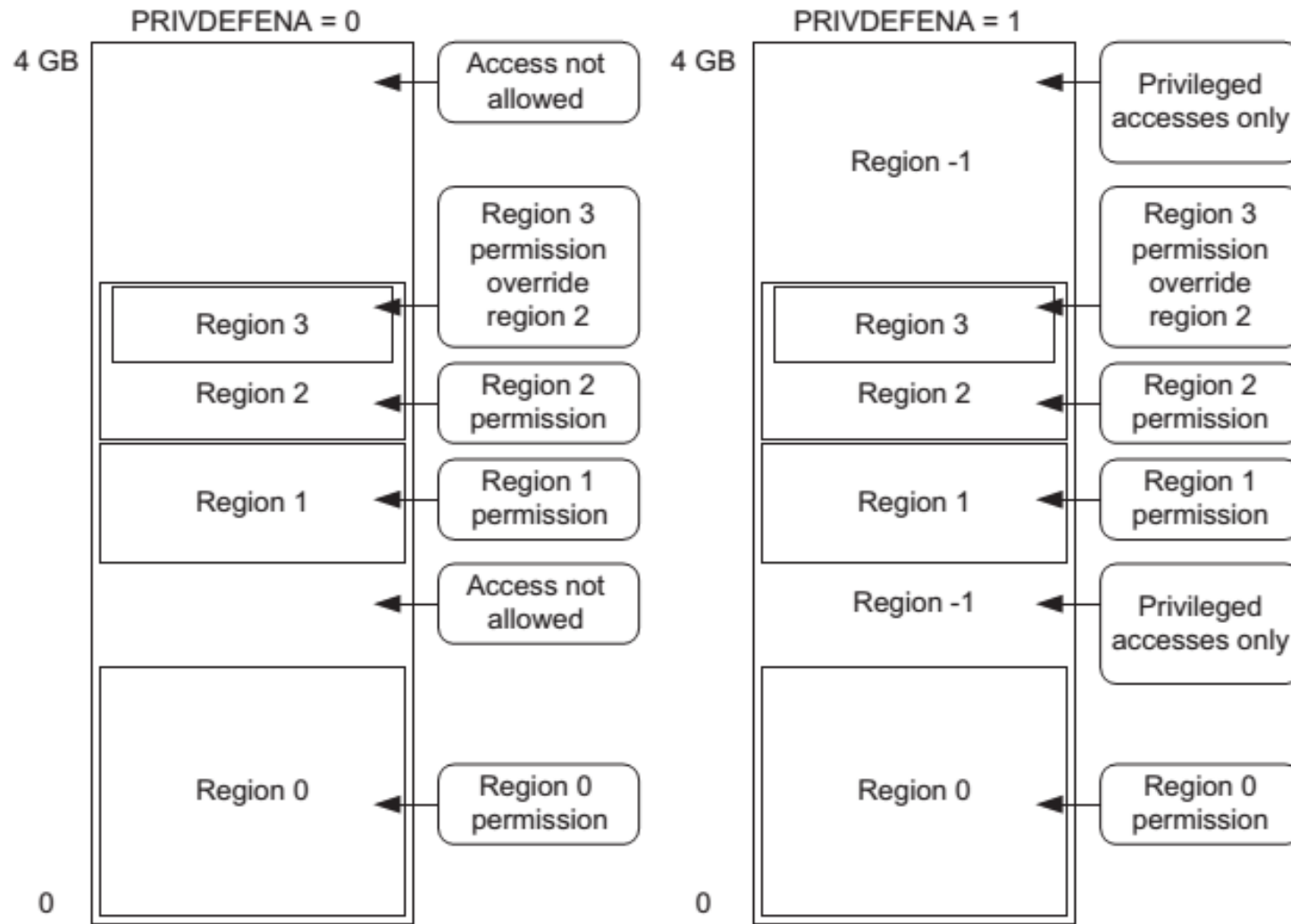
Cortex M3/M4: MPU->CTRL

Es el registro de control principal de la MPU.

| Bits | Name | Type | Reset Value | Description |
|------|------------|------|-------------|---|
| 2 | PRIVDEFENA | R/W | 0 | Privileged default memory map enable. When set to 1 and if the MPU is enabled, the default memory map will be used for privileged accesses as a background region. If this bit is not set, the background region is disabled and any access not covered by any enabled region will cause a fault. |
| 1 | HFNMENA | R/W | 0 | If set to 1, it enables the MPU during the HardFault handler and NMI handler; otherwise, the MPU is not enabled for the HardFault handler and NMI. |
| 0 | ENABLE | R/W | 0 | Enables the MPU if set to 1. |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Ejemplo de regiones



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: MPU->RNR , MPU->RBAR

Permite configurar que región de memoria estoy configurando.

| Bits | Name | Type | Reset Value | Description |
|------|--------|------|-------------|--|
| 7:0 | REGION | R/W | — | Select the region that is being programmed. Since eight regions are supported in the MPU, only bit[2:0] of this register is implemented. |

Permite configurar la dirección base de la región de memoria que estoy configurando.

| Bits | Name | Type | Reset Value | Description |
|------|--------|------|-------------|--|
| 31:N | ADDR | R/W | — | Base address of the region; N is dependent on the region size – for example, a 64 k size region will have a base address field of [31:16]. |
| 4 | VALID | R/W | — | If this is 1, the REGION defined in bit[3:0] will be used in this programming step; otherwise, the region selected by the MPU Region Number register is used. |
| 3:0 | REGION | R/W | — | This field overrides the MPU Region Number register if VALID is 1; otherwise, it is ignored. Since eight regions are supported in the Cortex®-M3 and Cortex-M4 MPU, the region number override is ignored if the value of the REGION field is larger than 7. |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: MPU->RASR

Es el registro de configuración de la región de memoria activa.

| Bits | Name | Type | Reset Value | Description |
|-------|----------------|------|-------------|--|
| 31:29 | Reserved | — | — | — |
| 28 | XN | R/W | — | Instruction Access Disable (1 = Disable instruction fetch from this region; an attempt to do so will result in a memory management fault). |
| 27 | Reserved | — | — | — |
| 26:24 | AP | R/W | — | Data Access Permission field |
| 23:22 | Reserved | — | — | — |
| 21:19 | TEX | R/W | — | Type Extension field |
| 18 | S | R/W | — | Shareable |
| 17 | C | R/W | — | Cacheable |
| 16 | B | R/W | — | Bufferable |
| 15:8 | SRD | R/W | — | Sub-region disable |
| 7:6 | Reserved | — | — | — |
| 5:1 | REGION SIZE | R/W | — | MPU Protection Region size |
| 0 | ENABLE | R/W | — | Region enable |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Permisos de la región

Bits [26:24] del registro MPU->RASR

| AP Value | Privileged Access | User Access | Description |
|----------|-------------------|---------------|---|
| 000 | No access | No access | No access |
| 001 | Read/Write | No access | Privileged access only |
| 010 | Read/Write | Read-only | Write in a user program generates a fault |
| 011 | Read/Write | Read/Write | Full access |
| 100 | Unpredictable | Unpredictable | Unpredictable |
| 101 | Read-only | No access | Privileged read only |
| 110 | Read-only | Read-only | Read-only |
| 111 | Read-only | Read-only | Read-only |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Tamaño de región

Bits [5:1] del registro MPU->RASR

| REGION Size | Size | | REGION Size | Size |
|-------------|----------|--|-------------|--------|
| b00000 | Reserved | | b10000 | 128 KB |
| b00001 | Reserved | | b10001 | 256 KB |
| b00010 | Reserved | | b10010 | 512 KB |
| b00011 | Reserved | | b10011 | 1 MB |
| b00100 | 32 Byte | | b10100 | 2 MB |
| b00101 | 64 Byte | | b10101 | 4 MB |
| b00110 | 128 Byte | | b10110 | 8 MB |
| b00111 | 256 Byte | | b10111 | 16 MB |
| b01000 | 512 Byte | | b11000 | 32 MB |
| b01001 | 1 KB | | b11001 | 64 MB |
| b01010 | 2 KB | | b11010 | 128 MB |
| b01011 | 4 KB | | b11011 | 256 MB |
| b01100 | 8 KB | | b11100 | 512 MB |
| b01101 | 16 KB | | b11101 | 1 GB |
| b01110 | 32 KB | | b11110 | 2 GB |
| b01111 | 64 KB | | b11111 | 4 GB |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Atributos de la región

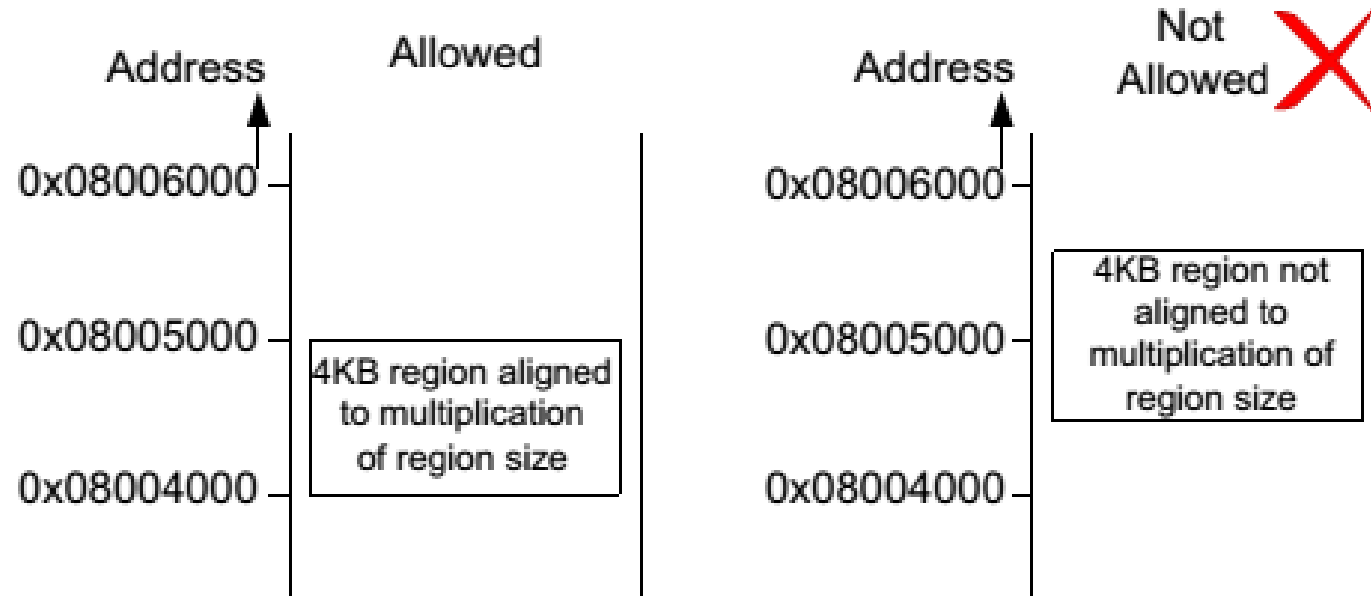
Bits [21:16] del registro MPU->RASR

| TEX | C | B | Description | Region Shareability |
|------|---|---|---|---------------------|
| b000 | 0 | 0 | Strongly ordered (transfers carry out and complete in programmed order) | Shareable |
| b000 | 0 | 1 | Shared device (write can be buffered) | Shareable |
| b000 | 1 | 0 | Outer and inner write-through; no write allocate | [S] |
| b000 | 1 | 1 | Outer and inner write-back; no write allocate | [S] |
| b001 | 0 | 0 | Outer and inner non-cacheable | [S] |
| b001 | 0 | 1 | Reserved | Reserved |
| b001 | 1 | 0 | Implementation defined | — |
| b001 | 1 | 1 | Outer and inner write-back; write and read allocate | [S] |
| b010 | 0 | 0 | Non-shared device | Not shared |
| b010 | 0 | 1 | Reserved | Reserved |
| b010 | 1 | X | Reserved | Reserved |
| b1BB | A | A | Cached memory; BB = outer policy, AA = inner policy | [S] |

Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Configurando la MPU...

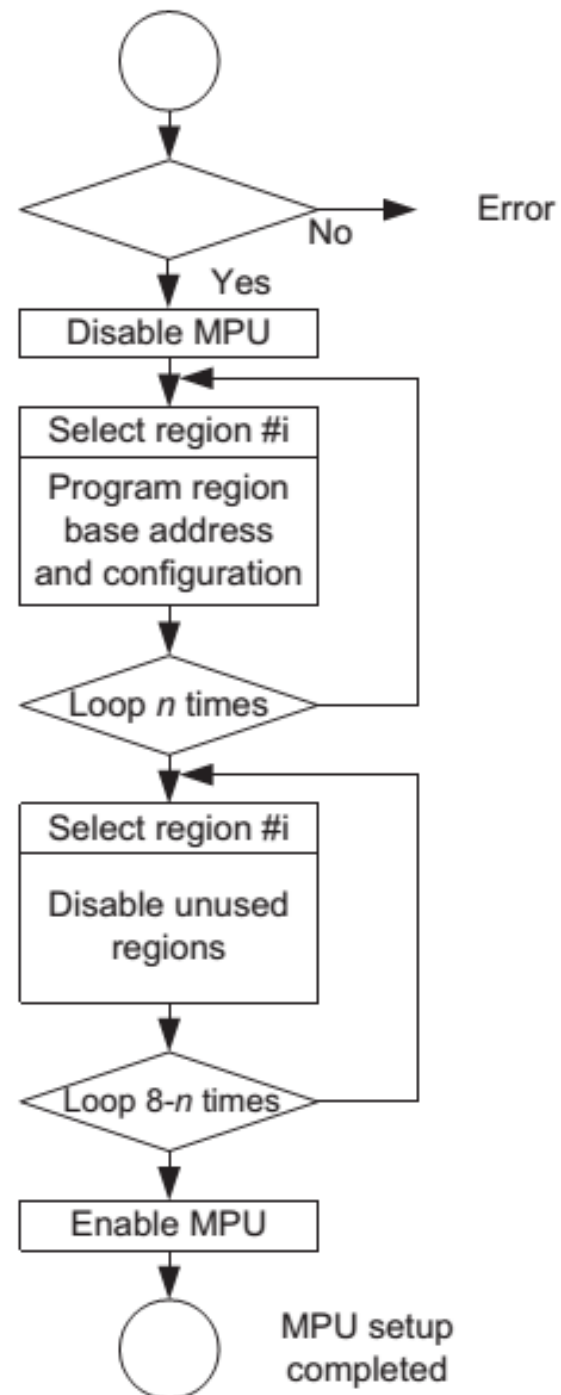
Las regiones de memoria deben estar alineadas a un múltiplo de su tamaño. Por ejemplo, si la región es de 4kB...



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Check MPU Type register
to see if MPU exist and
there are enough regions

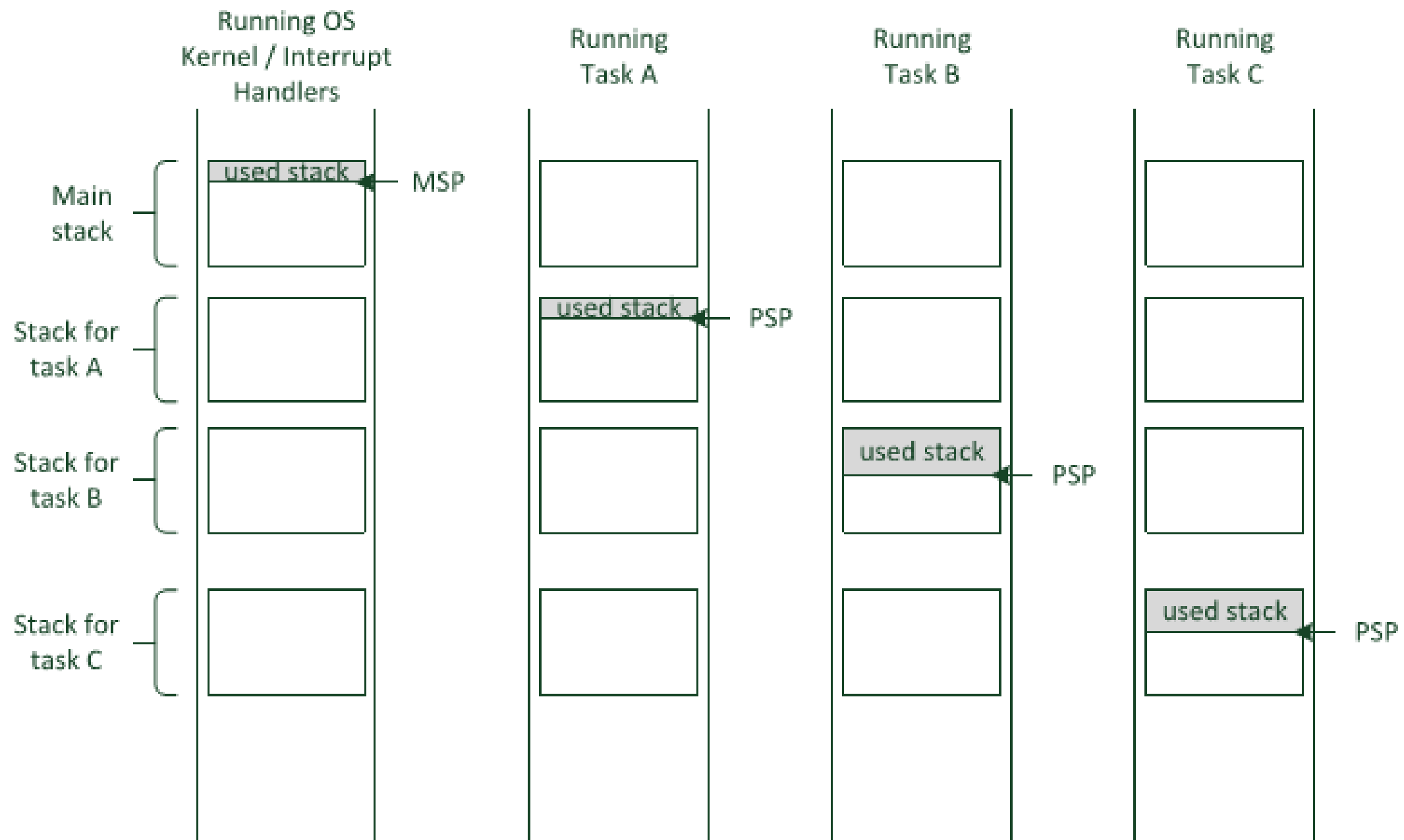
Region selection and
programming of region
registers can be combined
in one step



MPU setup
completed

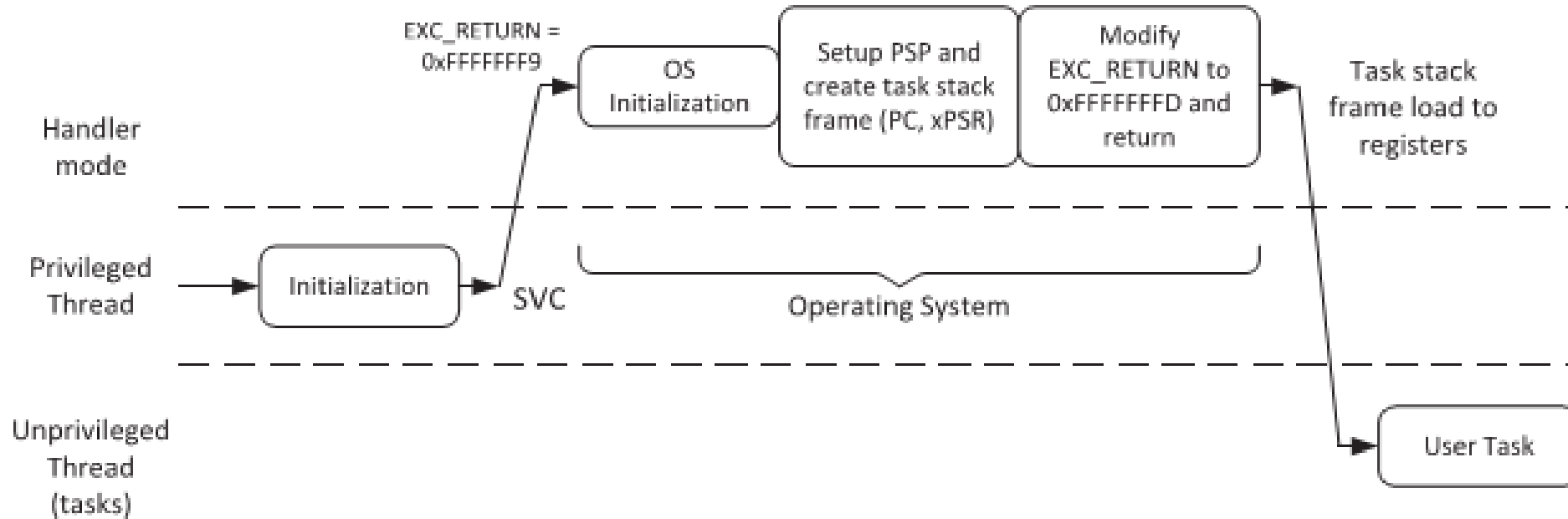
Características de soporte para SO embebidos

- *Timer SysTick*: Se incluye un timer dentro del core (utilizado normalmente para el scheduling). Esto mejora la portabilidad del kernel de una familia a otra.
- *Shadowed stack pointer*: Permite la conmutación por hardware de los punteros de pila para códigos corriendo en distintos niveles de privilegio.
- Excepciones *SVC* / *PendSV*: Son excepciones diseñadas exclusivamente para operaciones en SO tales como system calls y cambios de contexto.
- Implementación de niveles de privilegio
- Instrucciones para la implementación de semáforos y mutex



Cortex M3/M4: Armado de user stack

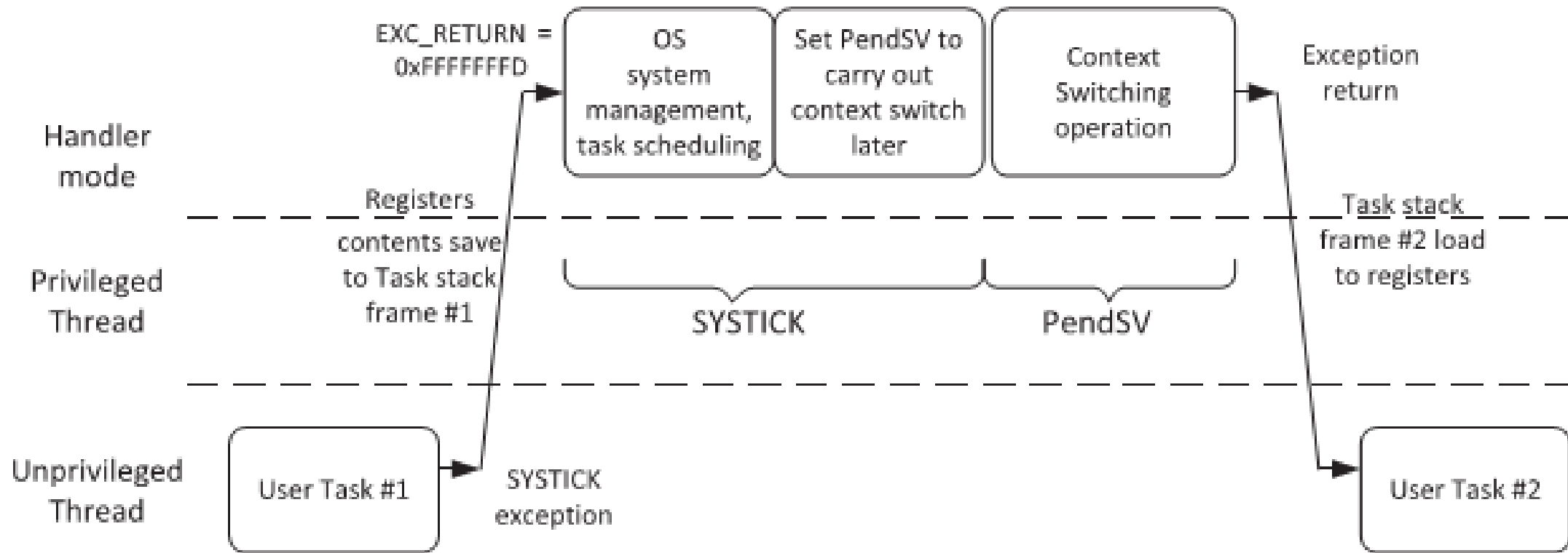
Ejemplo de inicialización de user stack



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Cortex M3/M4: Context switch

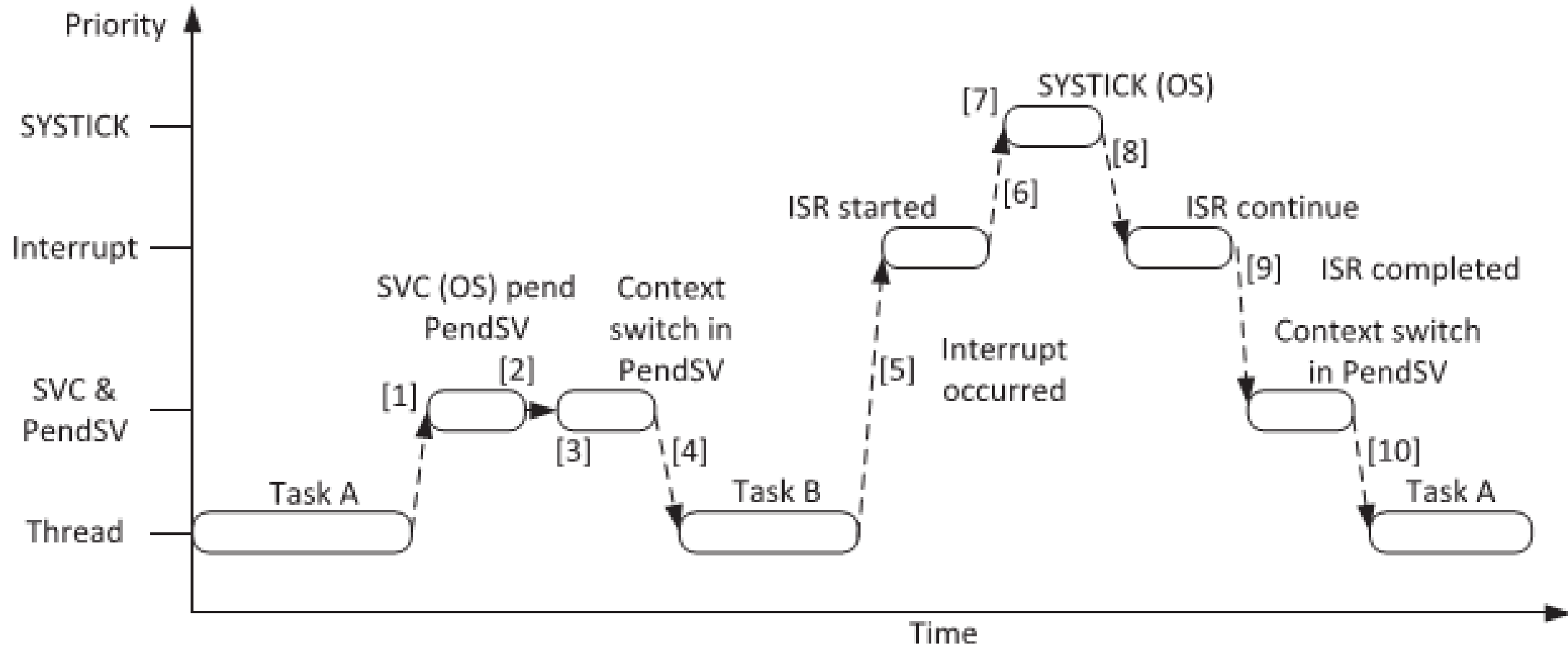
Ejemplo de inicialización de user stack



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: PendSV

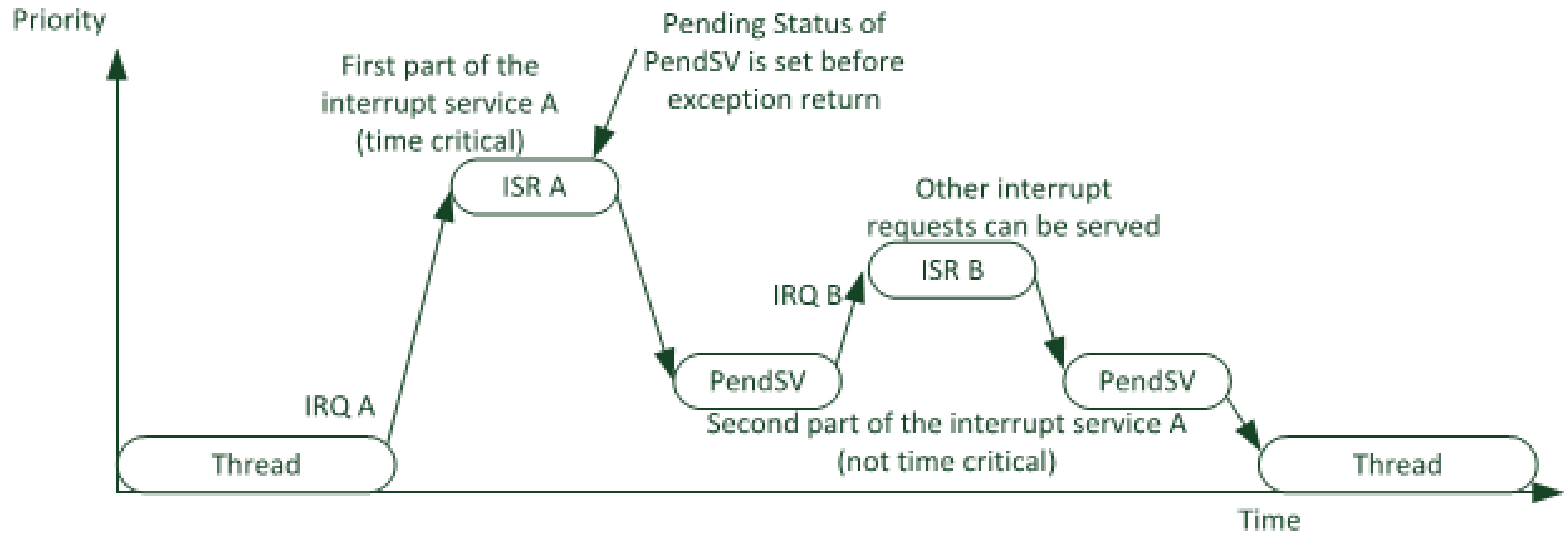
Uso para agendar cambio de contexto



Fuente: YIU, Joseph, "The Definitive Guide to ARM Cortex M3 and M4 processors"

Cortex M3/M4: PendSV

Uso para tareas “batch” dentro de una interrupción



Fuente: YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”

Bibliografía

- [1] YIU, Joseph , “The Definitive Guide to ARM Cortex M3 and M4 processors”
- [2] YIU, Joseph , “The Definitive Guide to ARM Cortex M0 and M0+ processors”
- [3]* “ARMv7-M Architecture Reference Manual”, ARM Inc.
- [4]* “ARMv7-AR Architecture Reference Manual”, ARM Inc.
- [5]* “ARM Cortex R Architecture”, White Paper, ARM, Inc.
- [6] www.infocenter.arm.com
- [7] www.keil.com

*Disponibles online en: *www.arm.com*