Introducción a Cortex – M3

Juan Alarcón.
jalarcon@frba.utn.edu.ar

Agenda.

- Procesadores ARM.
- Características Generales.
- Arquitectura.
- Cortex. v7-A, v7-R, v7-M
- Cortex-M
- Cortex-M3. Generalidades.
- Thumb
- Modelo del núcleo del procesador
- Mapa de memoria. Bit Banding.
- Registros. Registros Especiales.
- Modos de Operación.
- Excepciones. NVIC.
- Set de Instrucciones.
- Secuencia de Reset.
- Ejemplos de código.

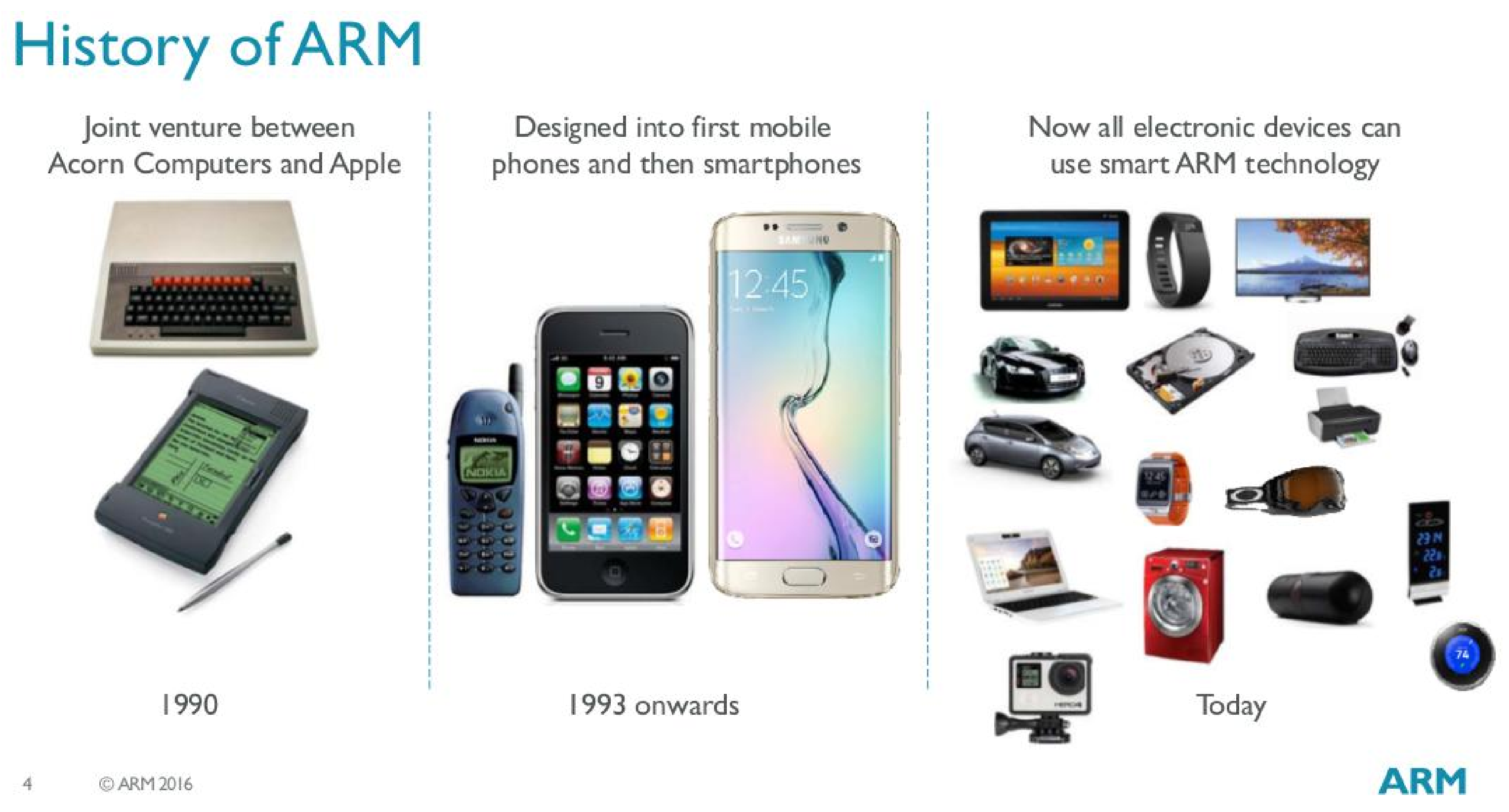
Procesadores ARM

- El nombre ARM proviene de ***Acorn RISC Machine*** renombrada en 1990 a ***Advanced RISC Machine***.
  - *Acorn fue la empresa que desarrolló el primer procesador ARM en 1985.*
  - *Hace enfásis en la arquitectura RISC (reduced instruction set computer) en vez de CISC (complex instruction set computer).*

Procesadores ARM.

- ***CISC.*** Un procesador con este tipo de diseño tiene instrucciones muy específicas, de ancho variable y que suelen ejecutarse en varios ciclos de reloj. Esto hace que el procesador se convierta en un hardware complejo.
- ***RISC.*** Se diseñan pocas instrucciones, de ancho fijo y que se ejecuten en un único ciclo de reloj. Se busca simplificar el procesador. Se pone la complejidad en el software

Procesadore ARM



History of ARM

Joint venture between Acorn Computers and Apple

Designed into first mobile phones and then smartphones

Now all electronic devices can use smart ARM technology

1990

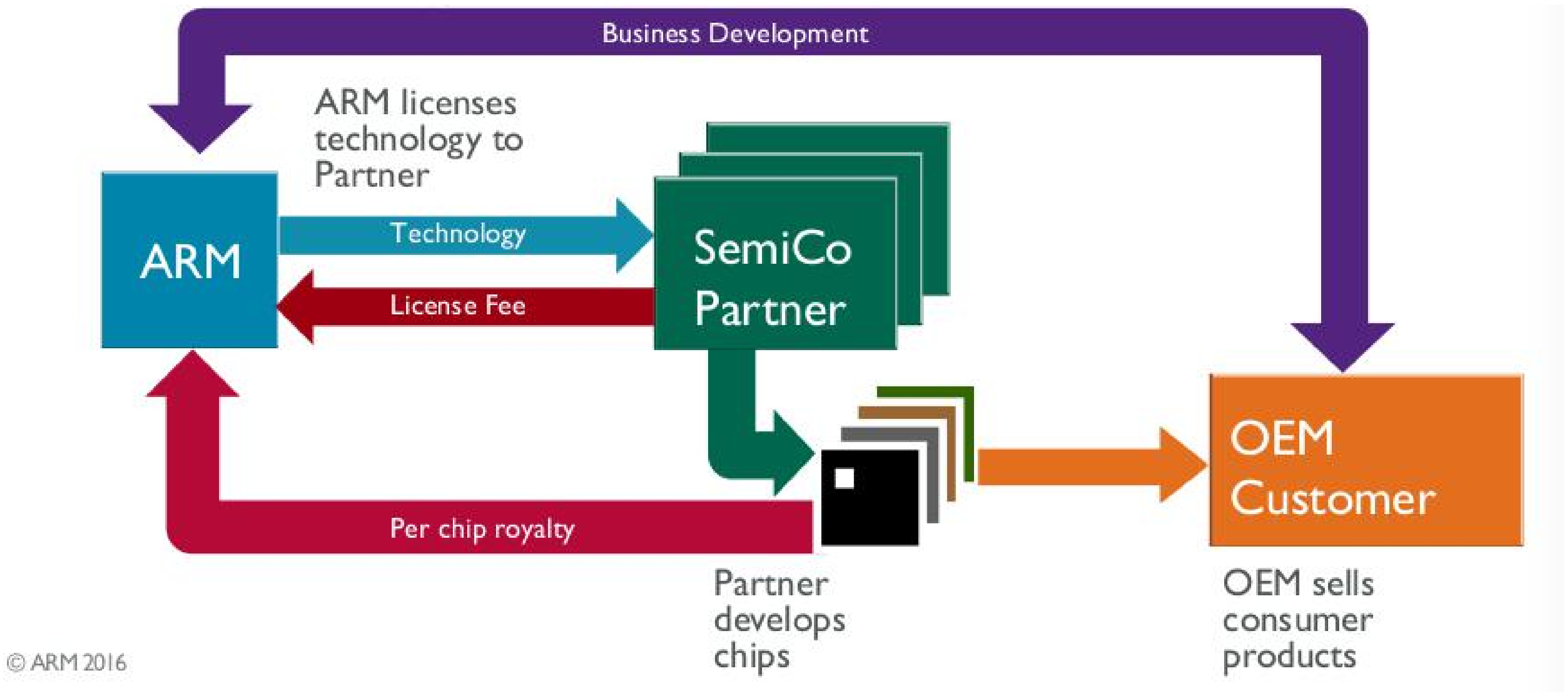
1993 onwards

Today

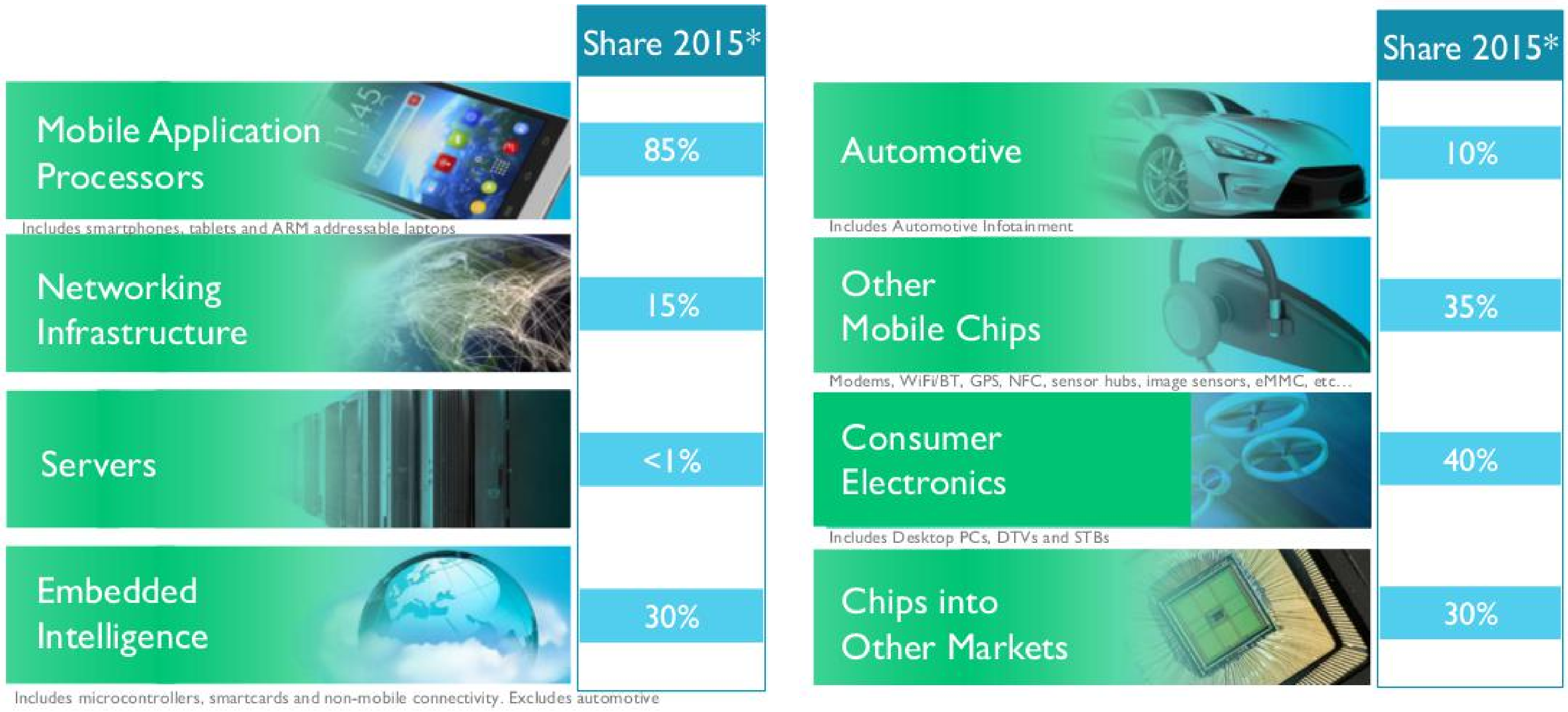
4 © ARM 2016

ARM

Procesadores ARM.



Procesadores ARM.



	Share 2015*
Mobile Application Processors Includes smartphones, tablets and ARM addressable laptops	85%
Networking Infrastructure	15%
Servers	<1%
Embedded Intelligence Includes microcontrollers, smartcards and non-mobile connectivity. Excludes automotive	30%

	Share 2015*
Automotive Includes Automotive Infotainment	10%
Other Mobile Chips Modems, WiFi/BT, GPS, NFC, sensor hubs, image sensors, eMMC, etc…	35%
Consumer Electronics Includes Desktop PCs, DTVs and STBs	40%
Chips into Other Markets	30%

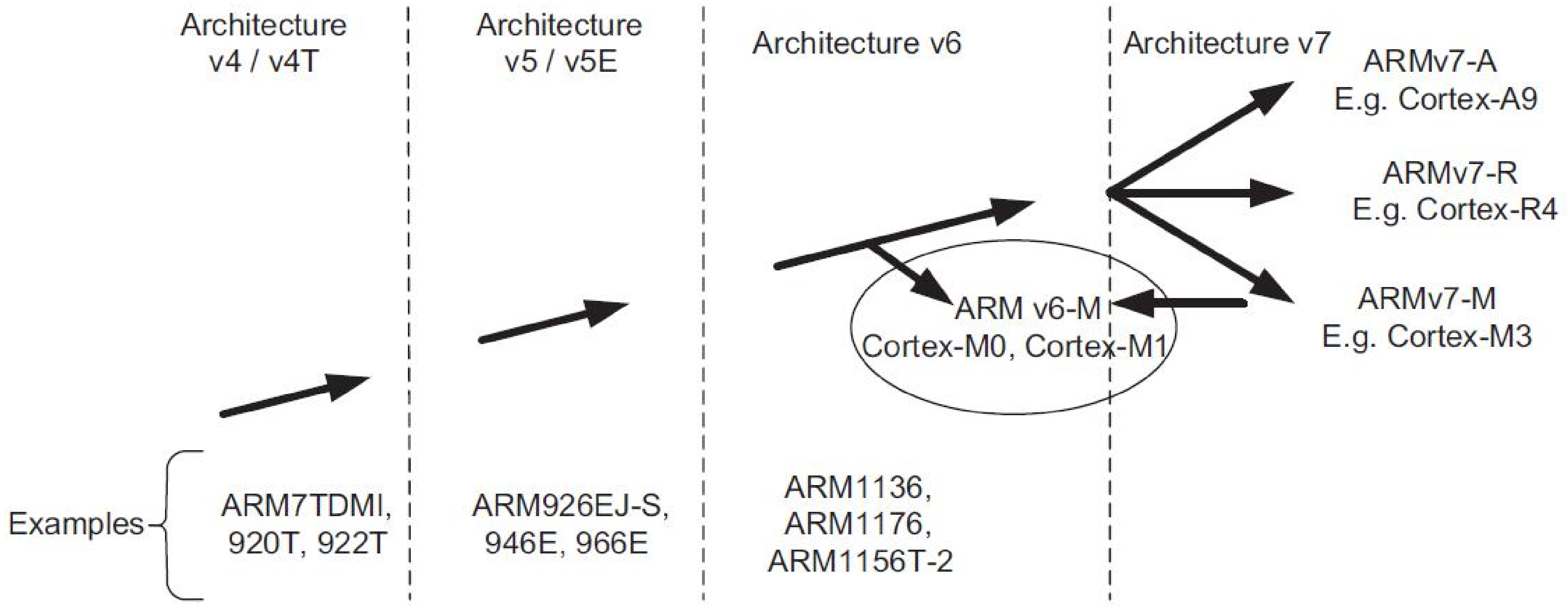
Procesadores ARM

- Procesadores de 32 bits (registros, bus de datos, bus de memoria).
- Procesadores con set de instrucciones RISC (reduced instruction set computer).
- Arquitectura Harvard (Cortex-M3). Arquitectura Von Neumann (Cortex-M0).
- Definen modos de operación (sistema – usuario definido por hardware).

Características Generales.

- Arquitectura Load/Store.
- Instrucciones de tamaño fijo.
- Máquina de 3 direcciones.
- La mayoría de las instrucciones son de un único ciclo de reloj.
- Ejecución condicional de instrucciones.
- Barrel shifter de 32 bits.

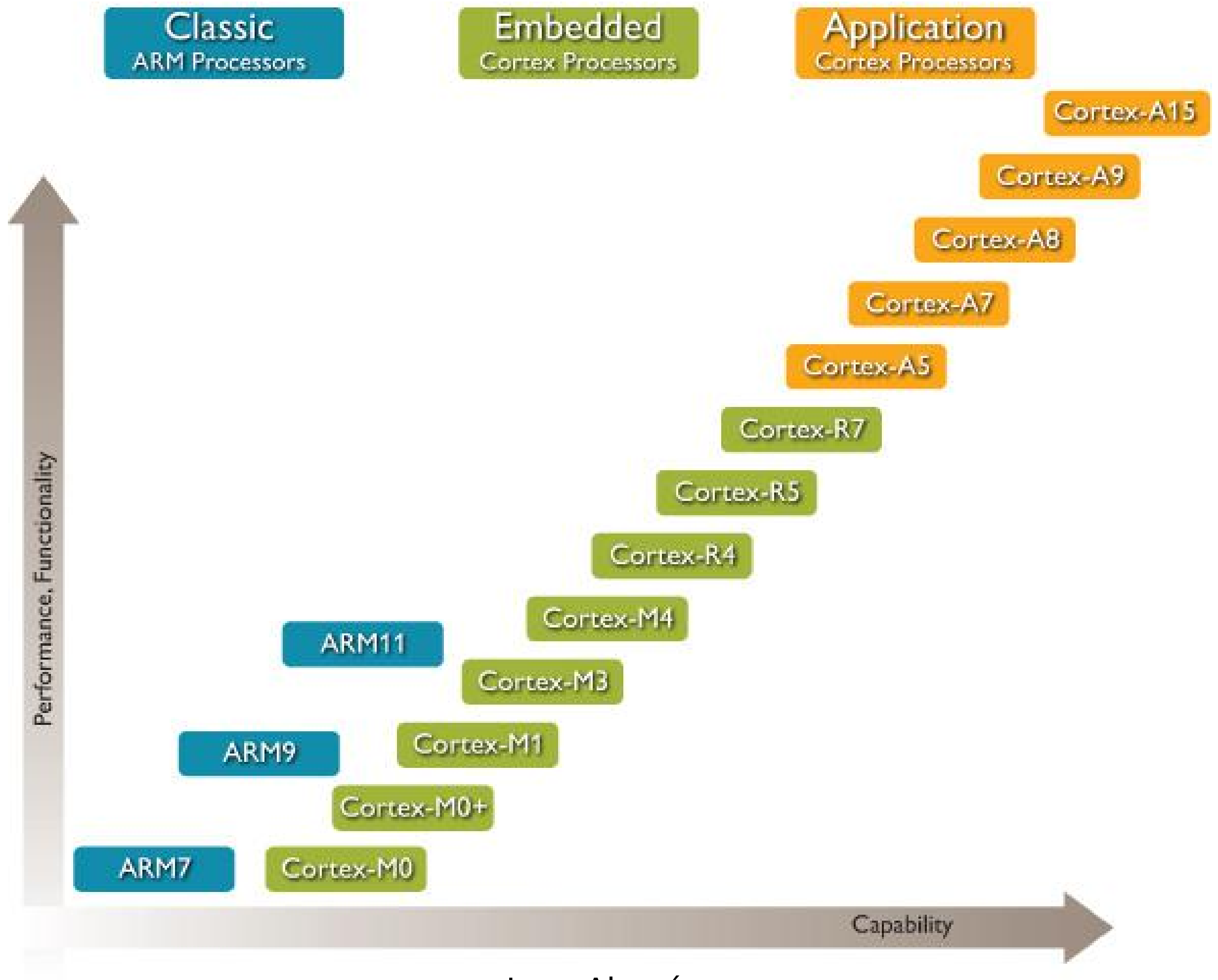
Arquitecturas. ARM



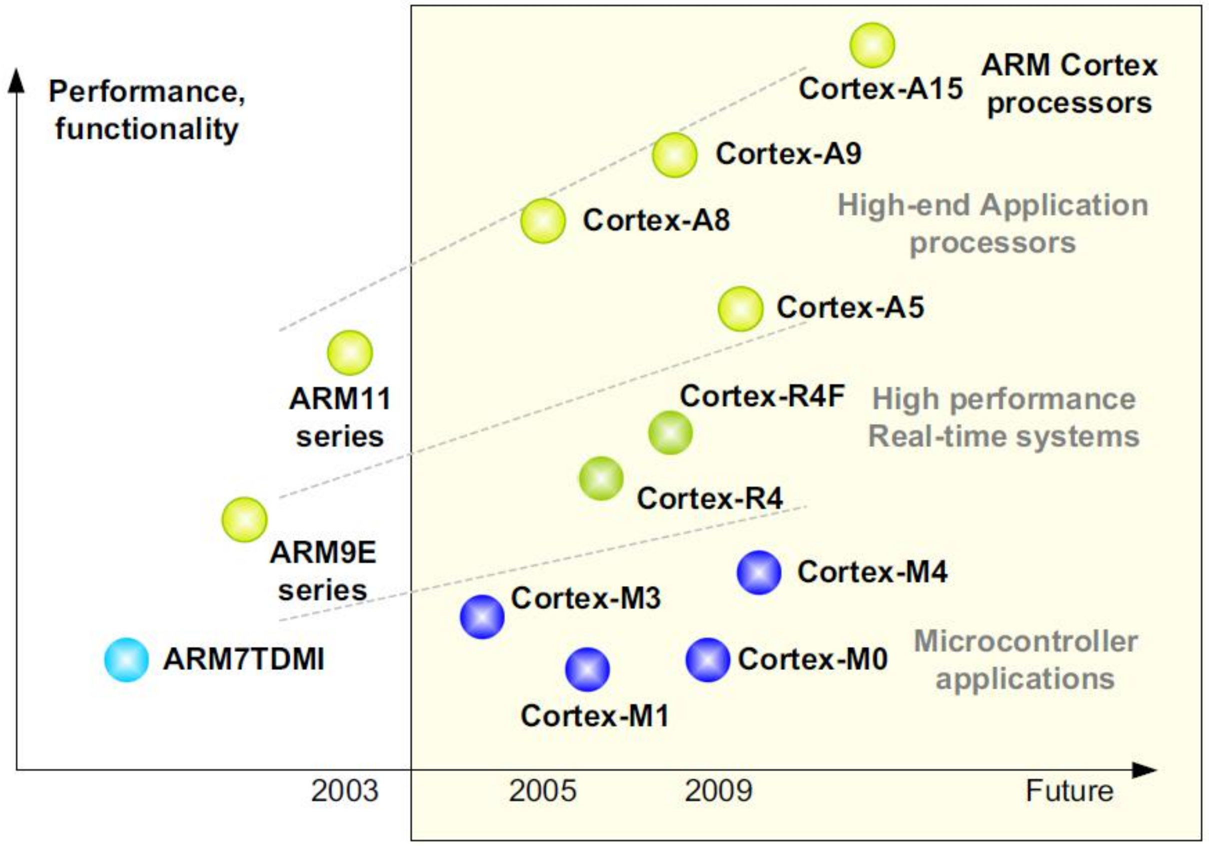
ARM - Cortex.

- Los procesadores Cortex son procesadores ARM (http://www.arm.com) de arquitectura v7.
- La arquitectura v7 tiene a su vez tres versiones.
  - v7-A. Application.
  - v7-R. Real-Time.
  - v7-M. Microcontroller.

Procesadores. ARM



Procesadores ARM.



Cortex. Versiones

- ***ARMv7-A. Application***. Procesadores que requieren ejecutar aplicaciones complejas de alto rendimiento, tales como Linux, Symbian, Android. Requieren unidad de administración de memoria (MMU). Por ejemplo celulares.

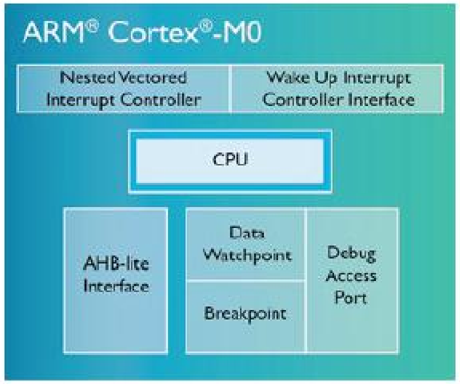
- ***ARMv7-R. Real-Time***. Procesadores orientados a aplicaciones de alto rendimiento, alta confiabilidad y tiempo real (dar una respuesta en un período garantizado de tiempo). Por ejemplo controladores de disco duros, sistemas

- ***ARMv7-M. Microcontroller***. Procesadores de bajo costo , bajo consumo y baja latencia de interrupciones. Procesadores orientados a usos de microcontroladores.

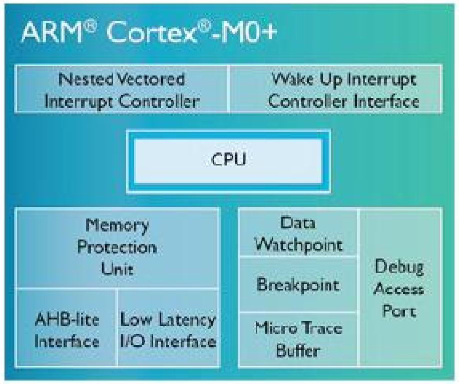
Cortex-M.

- ***Cortex-M0.*** Familia de microcontroladores muy pequeños, implementados en la arquitectura ARMv6-M. (LPC1114).
- ***Cortex-M0+***. Familia de microcontroladores muy pequeños y de mejor eficiencia energética que los M0 (LPC812).
- ***Cortex-M1.*** Familia de microcontroladores implementados como soft-cores (IP-cores en FPGA). Pertenecen a la arquitectura ARMv6-M. Por ejemplo Altera ARM Cortex-M1.
- ***Cortex-M3.*** Microcontroladores de arquitectura ARMv7-M. Agregan multiplicación por hardware en un ciclo de reloj, división por hardware y matemática saturada. LPC1769.
- ***Cortex-M4.*** Microcontroladores de arquitectura ARMv7-M, similar a los Cortex-M3 pero con extensiones para DSP. Multiplicación y suma de 32 bits en un único ciclo de reloj. LPC4300 .
- ***Cortex-M7***. Son la familia de mayor rendimiento de los Cortex M. Tienen pipeline de 6 etapas. Instrucciones SIMD y bus de 64bits.

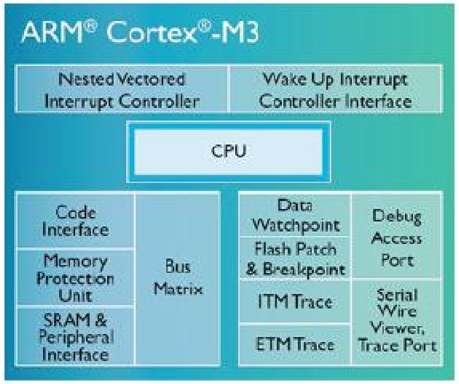
Cortex-M. Aplicaciones



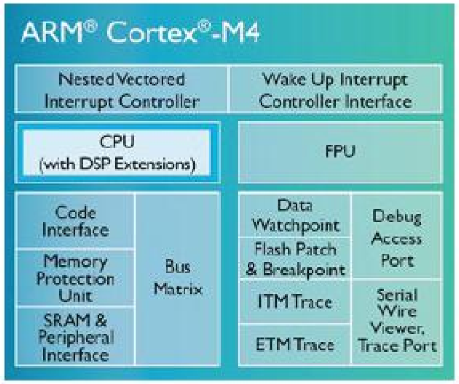
Lowest cost
Low power



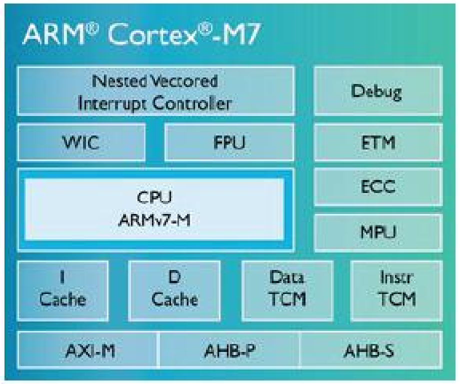
Lowest power
Outstanding energy efficiency



Performance efficiency
Feature rich connectivity


ARM® Cortex®-M4
Nested Vectored Interrupt Controller
Wake Up Interrupt Controller Interface
CPU (with DSP Extensions)
FPU
Code Interface
Memory Protection Unit
SRAM & Peripheral Interface
Bus Matrix
Data Watchpoint
Flash Patch & Breakpoint
ITM Trace
ETM Trace
Debug Access Port
Serial Wire Viewer, Trace Port

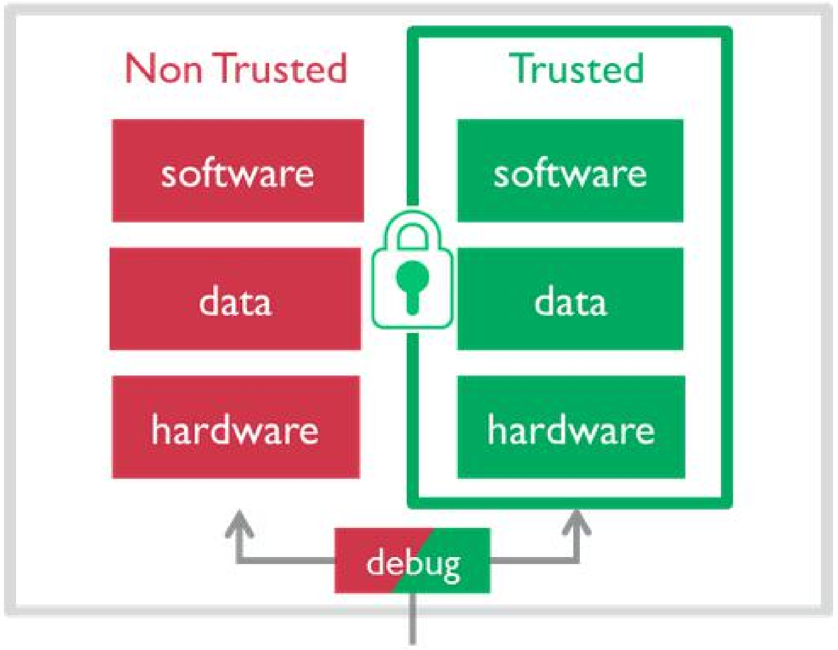
Digital Signal Control (DSC)
Processor with DSP
Accelerated SIMD
Floating point (FP)



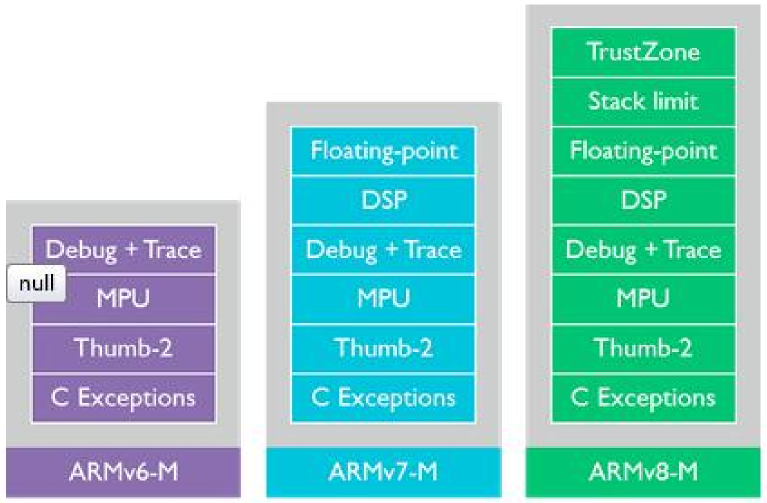
Maximum DSC Performance
Flexible Memory System
Cache, TCM, AXI, ECC
Double & Single Precision FP

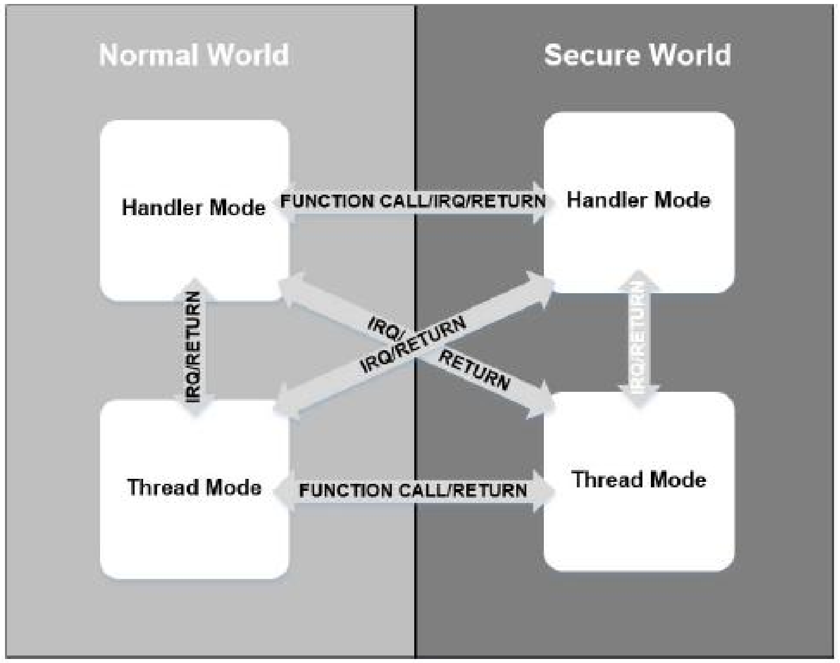
Cortex-M. ARMv8

- ***Cortex-M23.*** Procesador más pequeño de con CPU ARMv8 (comparable con los M0/M0+)
- ***Cortex-M33.*** Procesador con extensiones para DSP + FPU de arquitectura ARMv8. (comparable con los M4/M7).



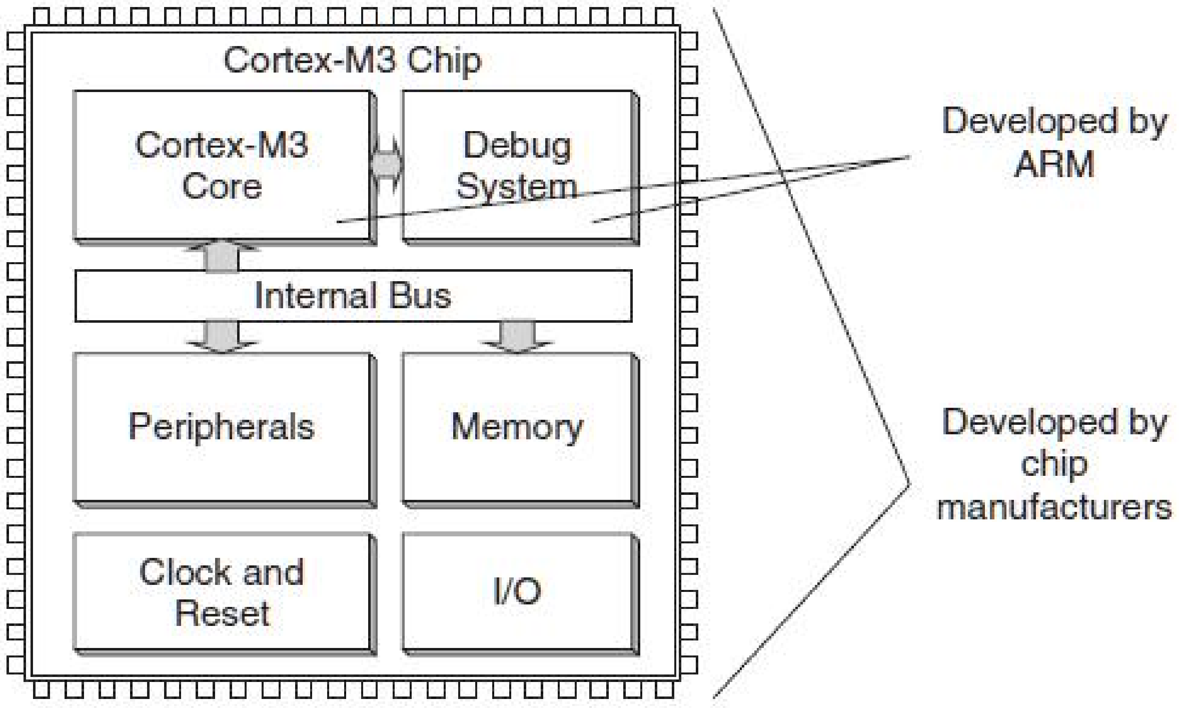
Cortex-M. ARMv8. TrustZone





Juan Alarcón.
jalarcon@frba.utn.edu.ar

Cortex-M3. Generalidades.

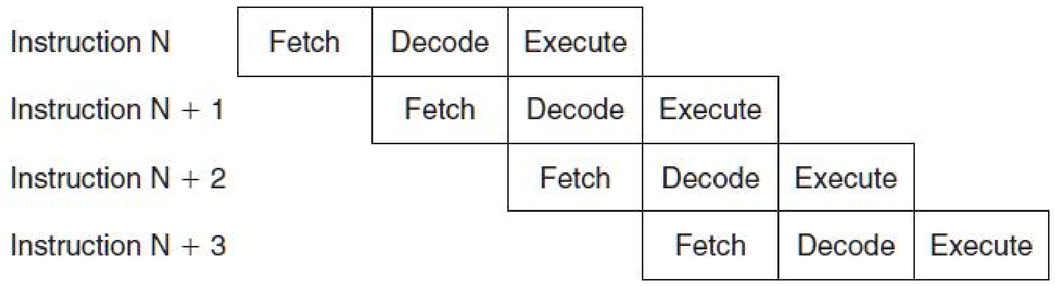


Cortex-M3. Generalidades.

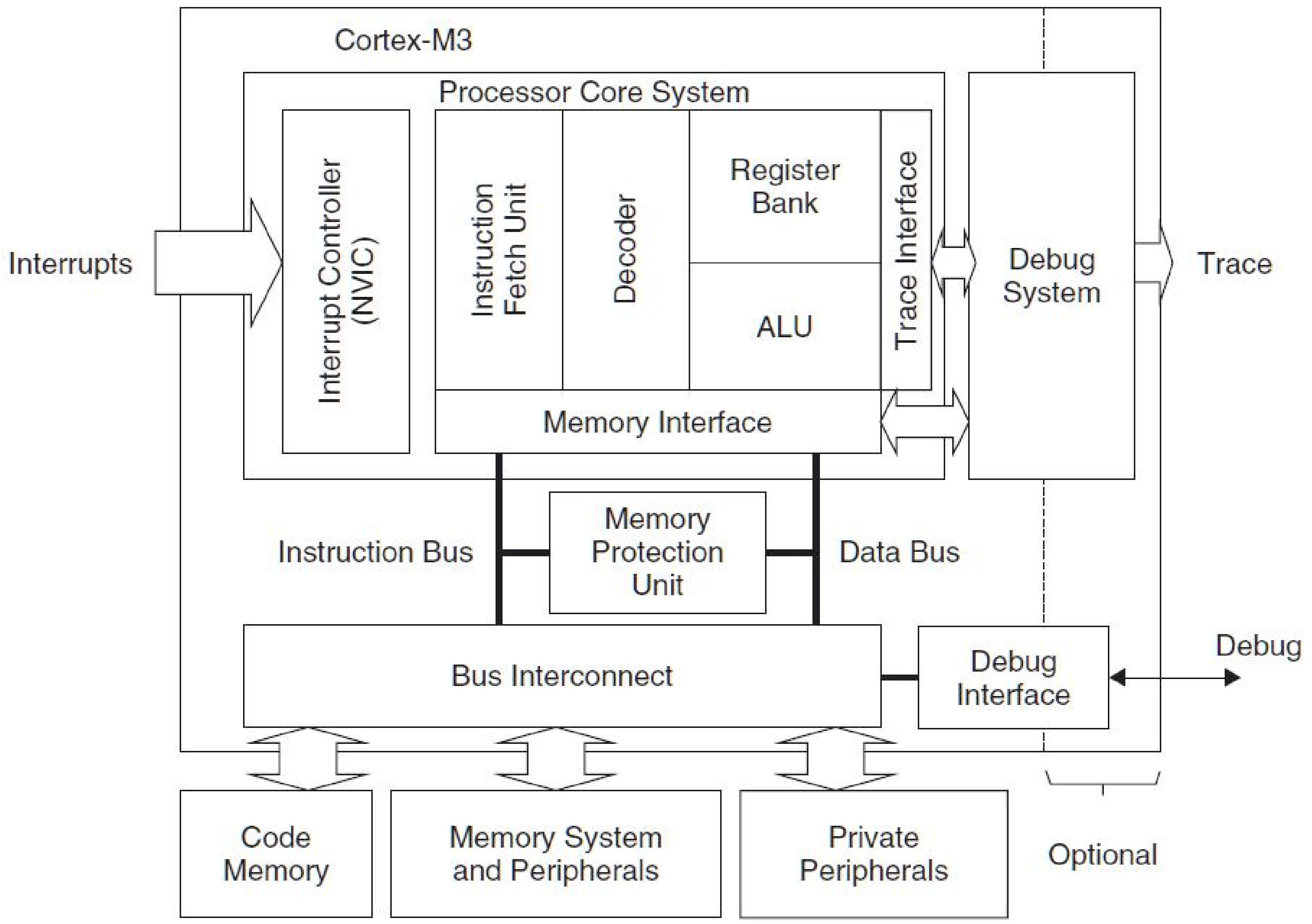
- Procesador
  - Set de instrucciones Thumb-2.
  - Pipeline de 3 etapas
  - Arquitectura Harvard con búsqueda en memoria de código y en memoria de datos simultánea.
  - Multiplicación de 32 bits en un solo ciclo de reloj.
  - División de 32 bits por hardware en 2-12 ciclos de reloj.
  - Modo Handler y Modo Thread.
  - Soporte de hasta 240 interrupciones por medio del NVIC.
  - Latencia de interrupciones de 12 ciclos de reloj.
- Registros
  - 13 registros de propósito general de 32 bits.
  - Link Register (LR)
  - Program Counter (PC)
  - Program Status Register (xPSR)
  - Dos registros de puntero a pila (SP)

Cortex-M3. Pipeline

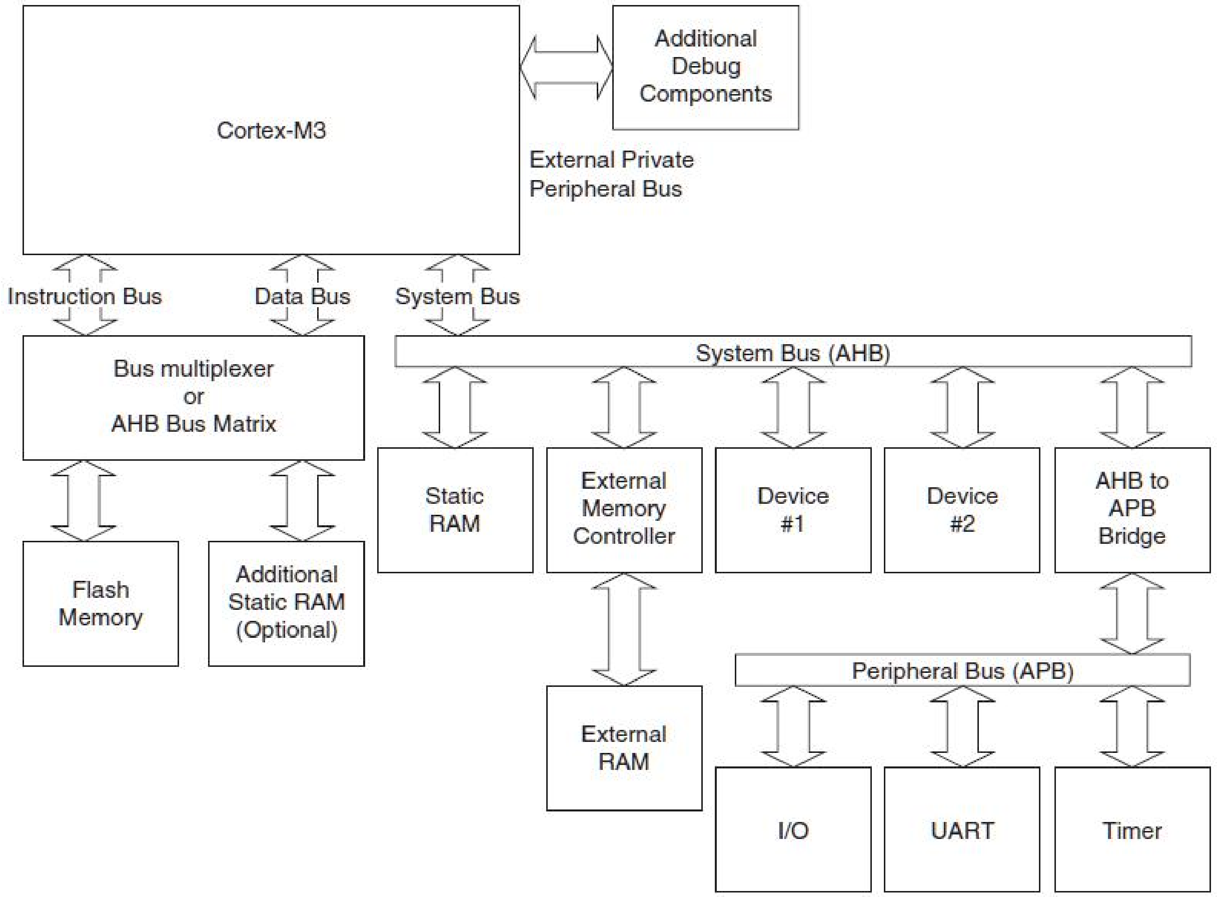
- Para poder alcanzar altas velocidades de clock se divide lógica combinacional en bloques de lógica más pequeña separada por FF, minimizando los caminos críticos.
- Los procesadores Cortex implementan esta técnica en un pipeline de 3 etapas.



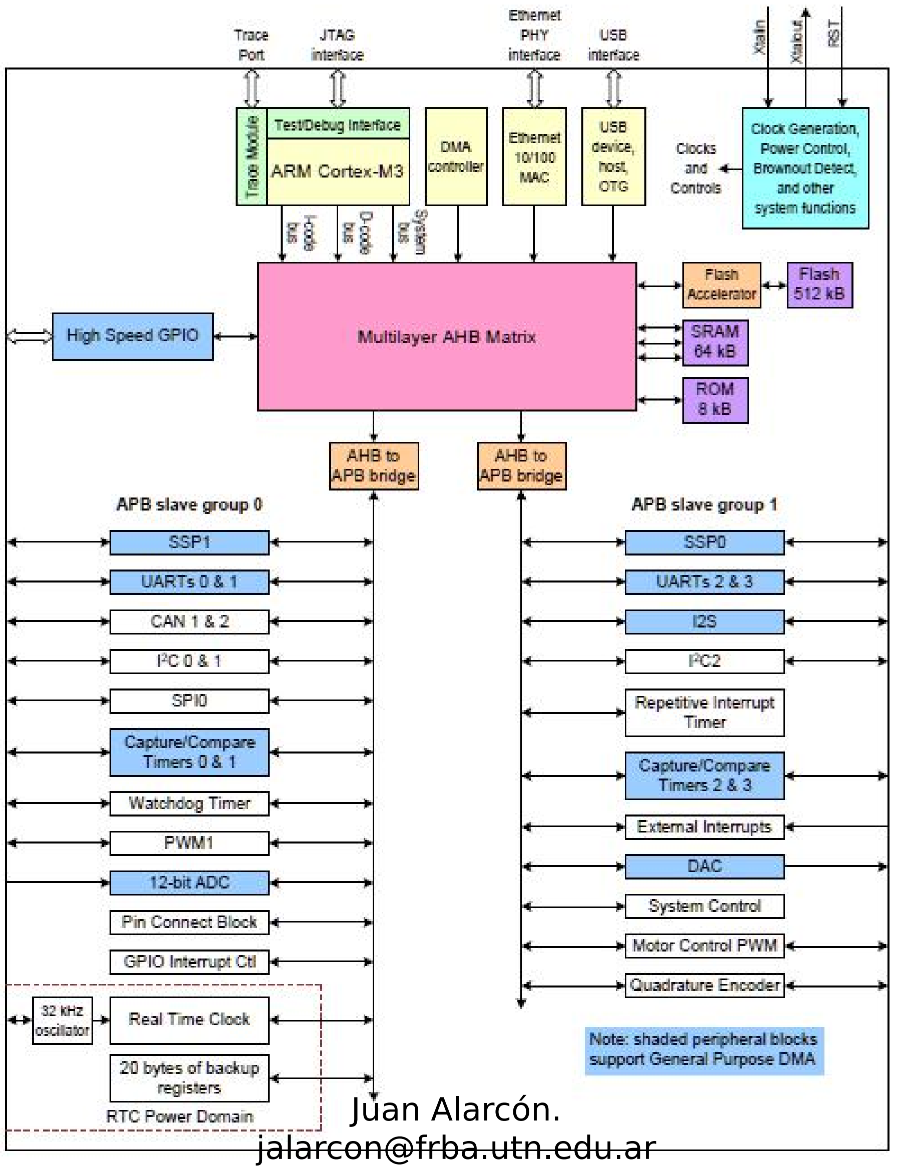
Núcleo del Cortex-M3.



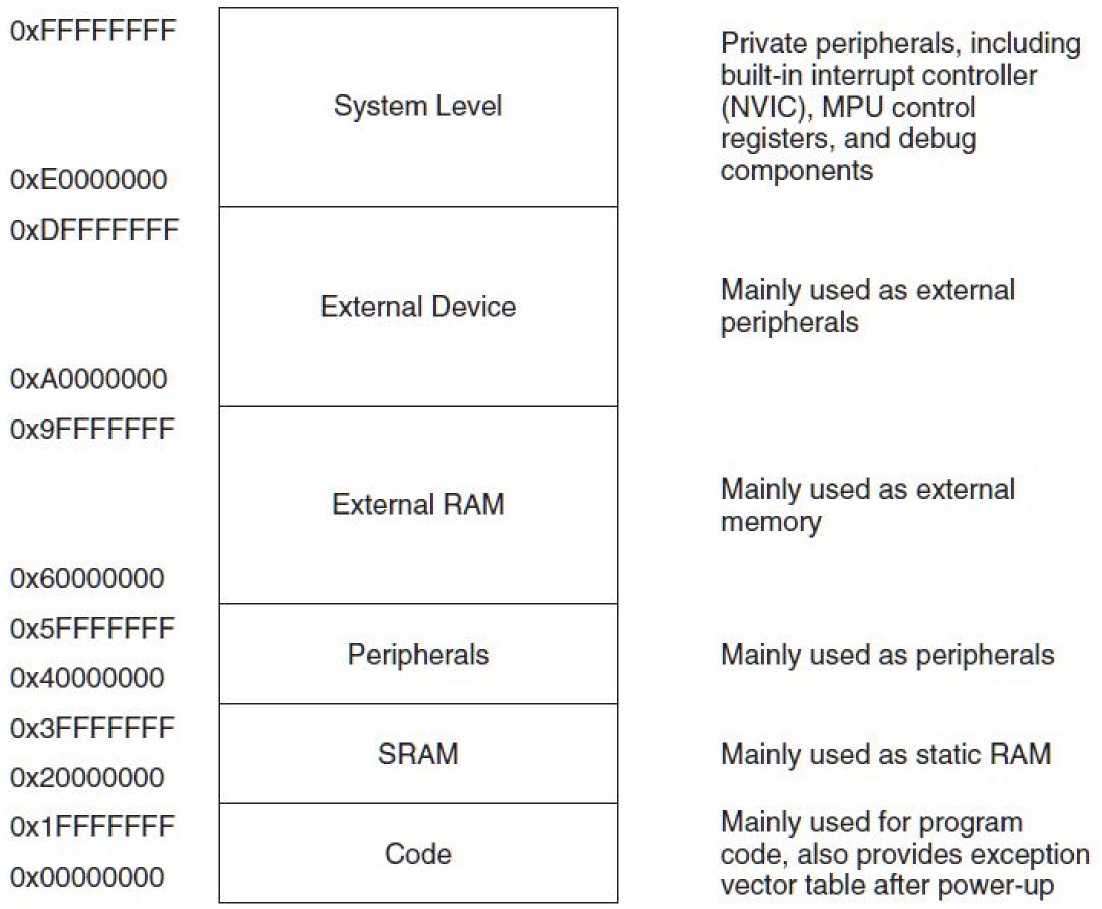
Buses. Cortex – M3.



Ejemplo.Cortex-M3



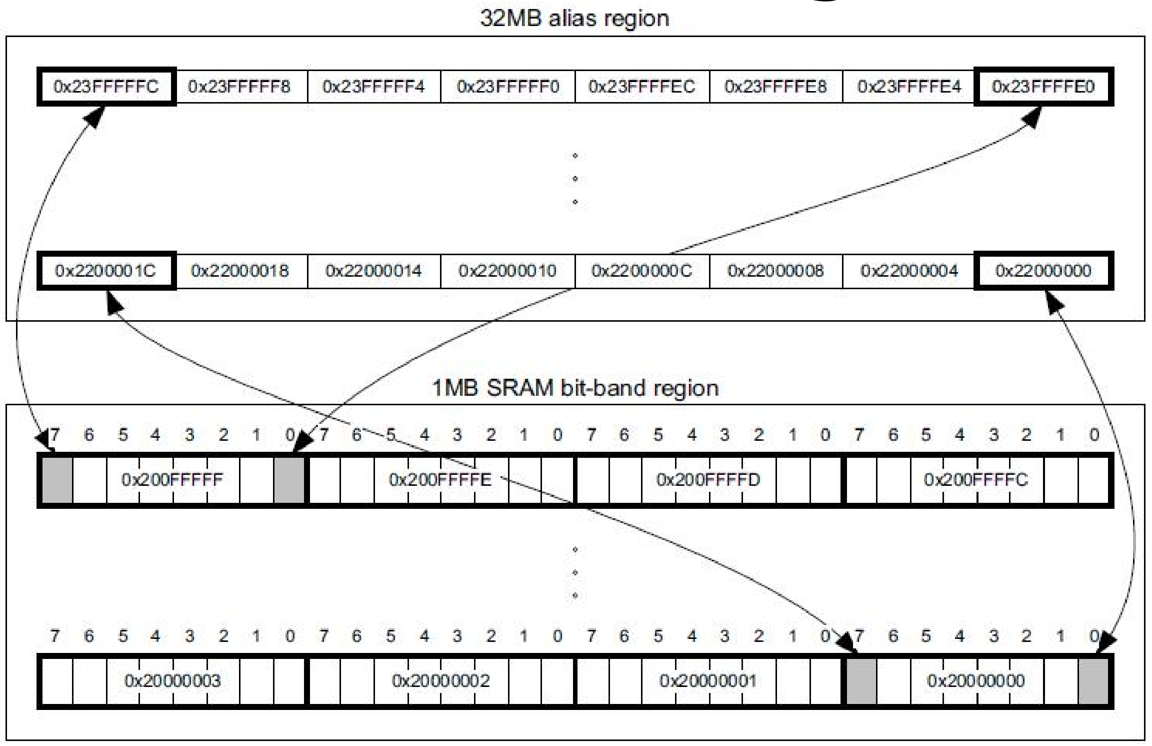
Mapa de memoria.



0xFFFFFFFF
0xE0000000
0xDFFFFFFF
0xA0000000
0x9FFFFFFF
0x60000000
0x5FFFFFFF
0x40000000
0x3FFFFFFF
0x20000000
0x1FFFFFFF
0x00000000
System Level
External Device
External RAM
Peripherals
SRAM
Code
Private peripherals, including built-in interrupt controller (NVIC), MPU control registers, and debug components
Mainly used as external peripherals
Mainly used as external memory
Mainly used as peripherals
Mainly used as static RAM
Mainly used for program code, also provides exception vector table after power-up

- Los Cortex-M3 tienen un mapa de memoria predefinido.
- El procesador direcciona 4Gb.
- Se definen zonas de acceso a nivel de bits (bit banding).

Bit banding.

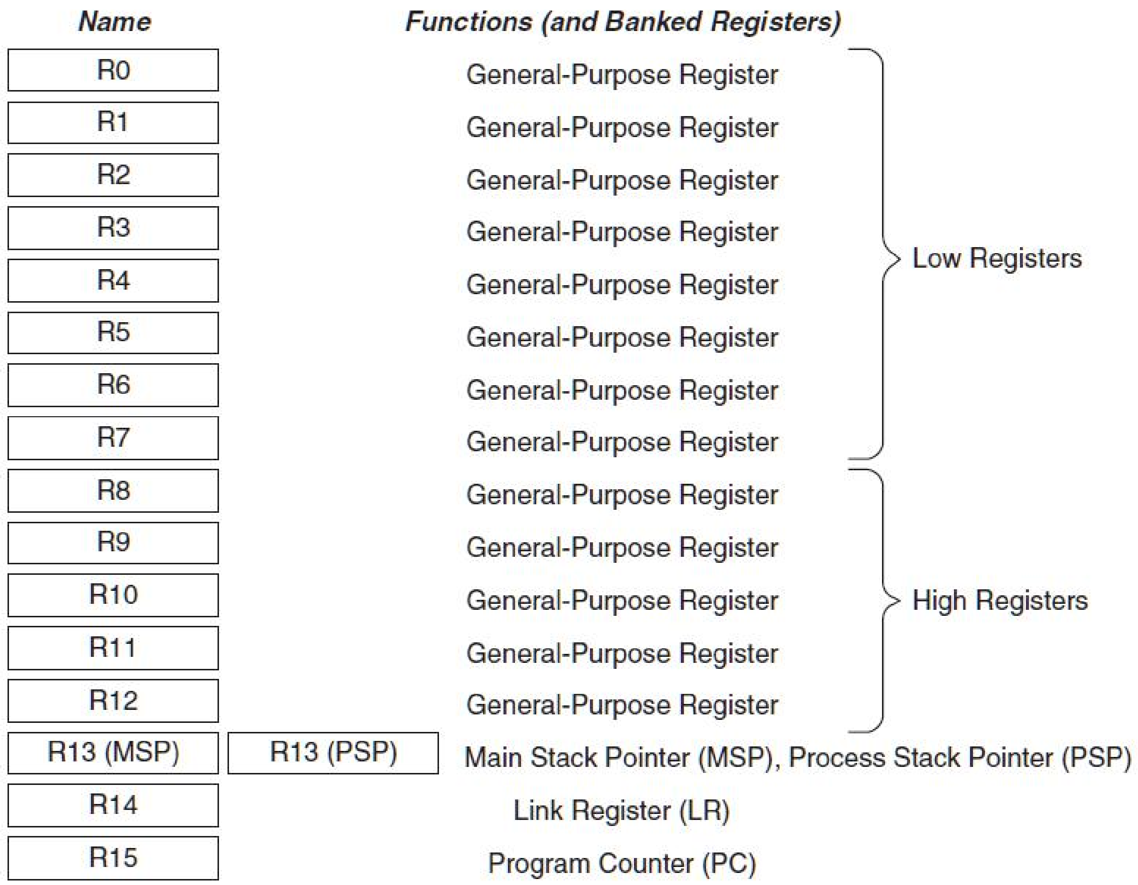


```
bit_word_offset = (byte_offset x 32) + (bit_number × 4)

bit_word_addr = bit_band_base + bit_word_offset
```

- Dos zonas de 1MB que son direccionables a nivel de bits por medio de dos zonas de “alias” de 32MB.

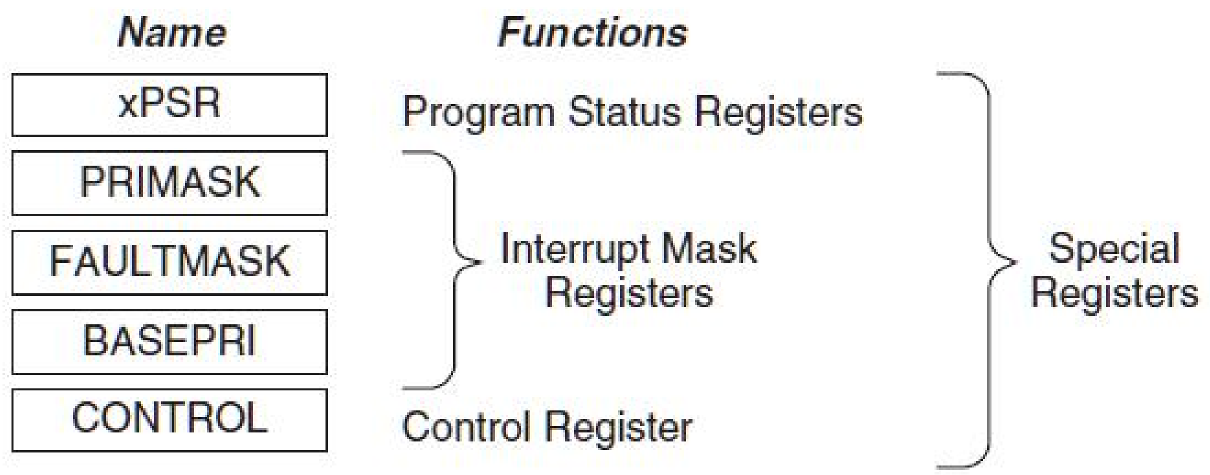
Registros.



Name		*Functions (and Banked Registers)*	
R0		General-Purpose Register	Low Registers
R1		General-Purpose Register	
R2		General-Purpose Register	
R3		General-Purpose Register	
R4		General-Purpose Register	
R5		General-Purpose Register	
R6		General-Purpose Register	
R7		General-Purpose Register	
R8		General-Purpose Register	High Registers
R9		General-Purpose Register	
R10		General-Purpose Register	
R11		General-Purpose Register	
R12		General-Purpose Register	
R13 (MSP)	R13 (PSP)	Main Stack Pointer (MSP), Process Stack Pointer (PSP)	
R14		Link Register (LR)	
R15		Program Counter (PC)	

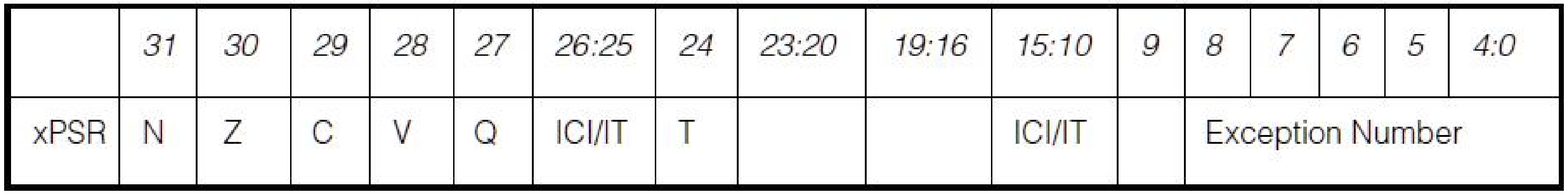
- Algunas de las instrucciones Thumb de 16 bits sólo pueden acceder a los registros bajos (R0-R7).
- MSP es el puntero a la pila para las excepciones y modo privilegiado.
- PSP puntero a la pila del código de aplicación.
- LR. Cuando se llama a una subrutina, LR contiene la dirección de retorno.

Registros Especiales.

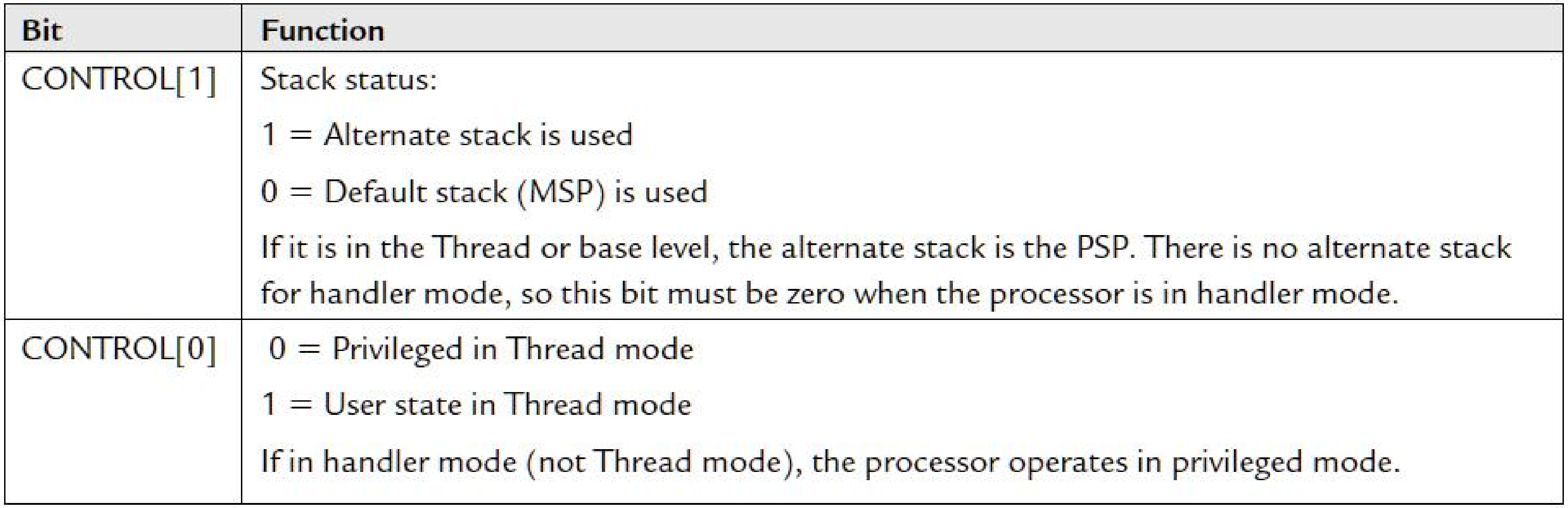


Register	Function
xPSR	Provide ALU flags (zero flag, carry flag), execution status, and current executing interrupt number
PRIMASK	Disable all interrupts except the nonmaskable interrupt (NMI) and HardFault
FAULTMASK	Disable all interrupts except the NMI
BASEPRI	Disable all interrupts of specific priority level or lower priority level
CONTROL	Define privileged status and stack pointer selection

Registros Especiales.

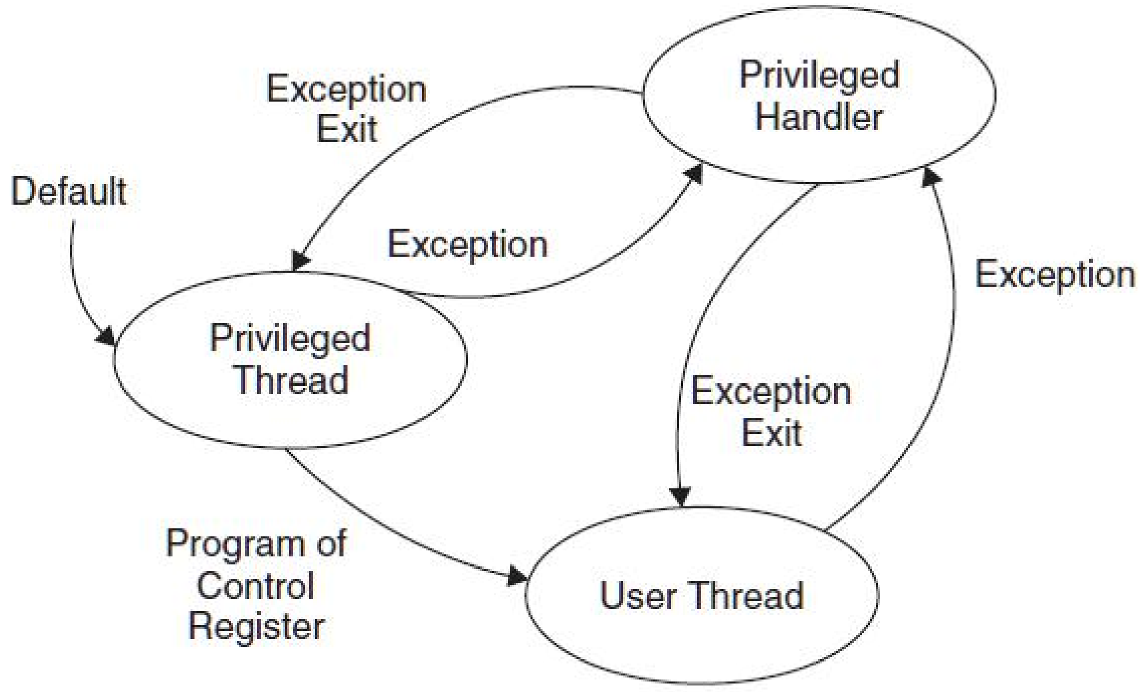


	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
xPSR	N	Z	C	V	Q	ICI/IT	T			ICI/IT		Exception Number				



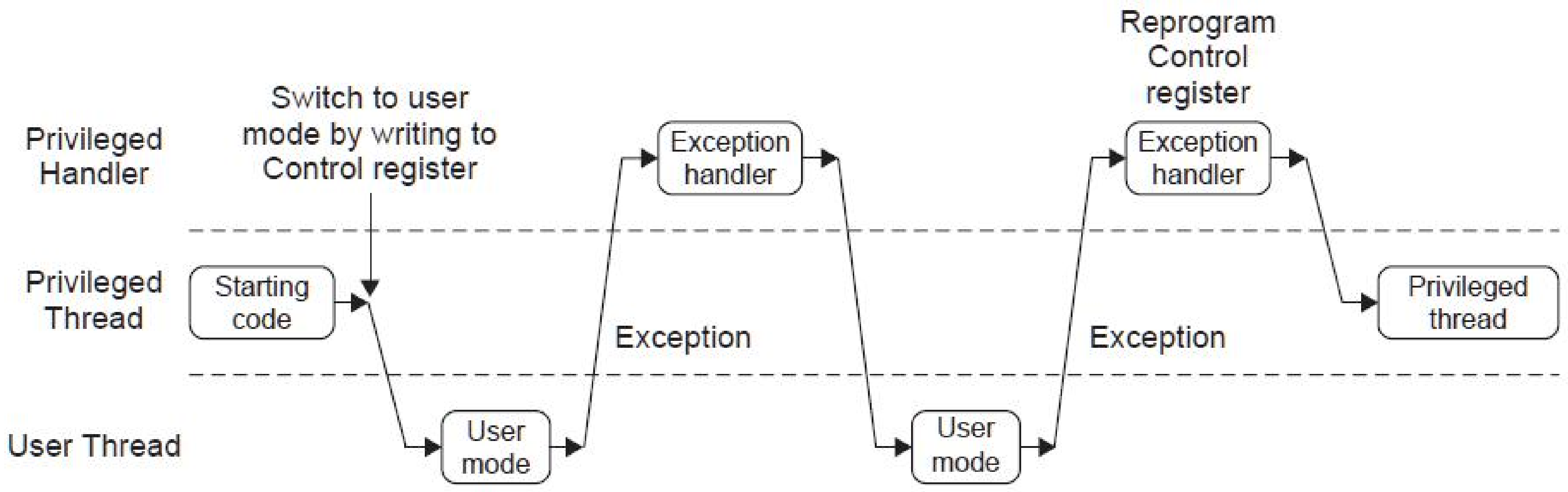
Bit	**Function**
CONTROL[1]	Stack status: 1 = Alternate stack is used 0 = Default stack (MSP) is used If it is in the Thread or base level, the alternate stack is the PSP. There is no alternate stack for handler mode, so this bit must be zero when the processor is in handler mode.
CONTROL[0]	0 = Privileged in Thread mode 1 = User state in Thread mode If in handler mode (not Thread mode), the processor operates in privileged mode.

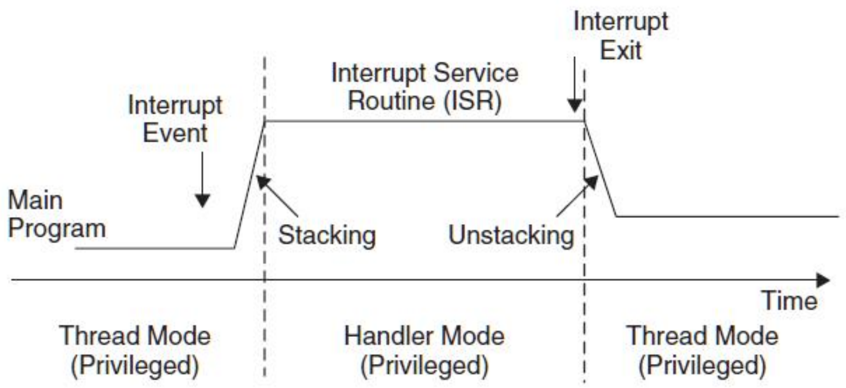
Modos de operación

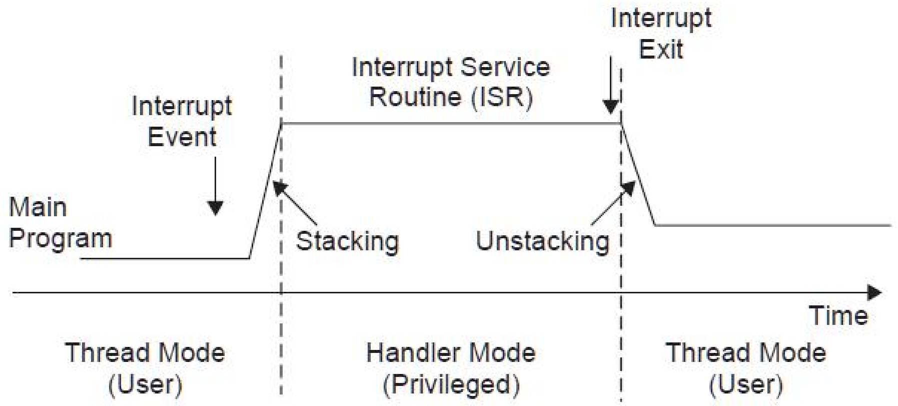


	Privileged	***User***
When running an exception	Handler Mode	■
When running main program	Thread Mode	Thread Mode

Modos de operación (2).



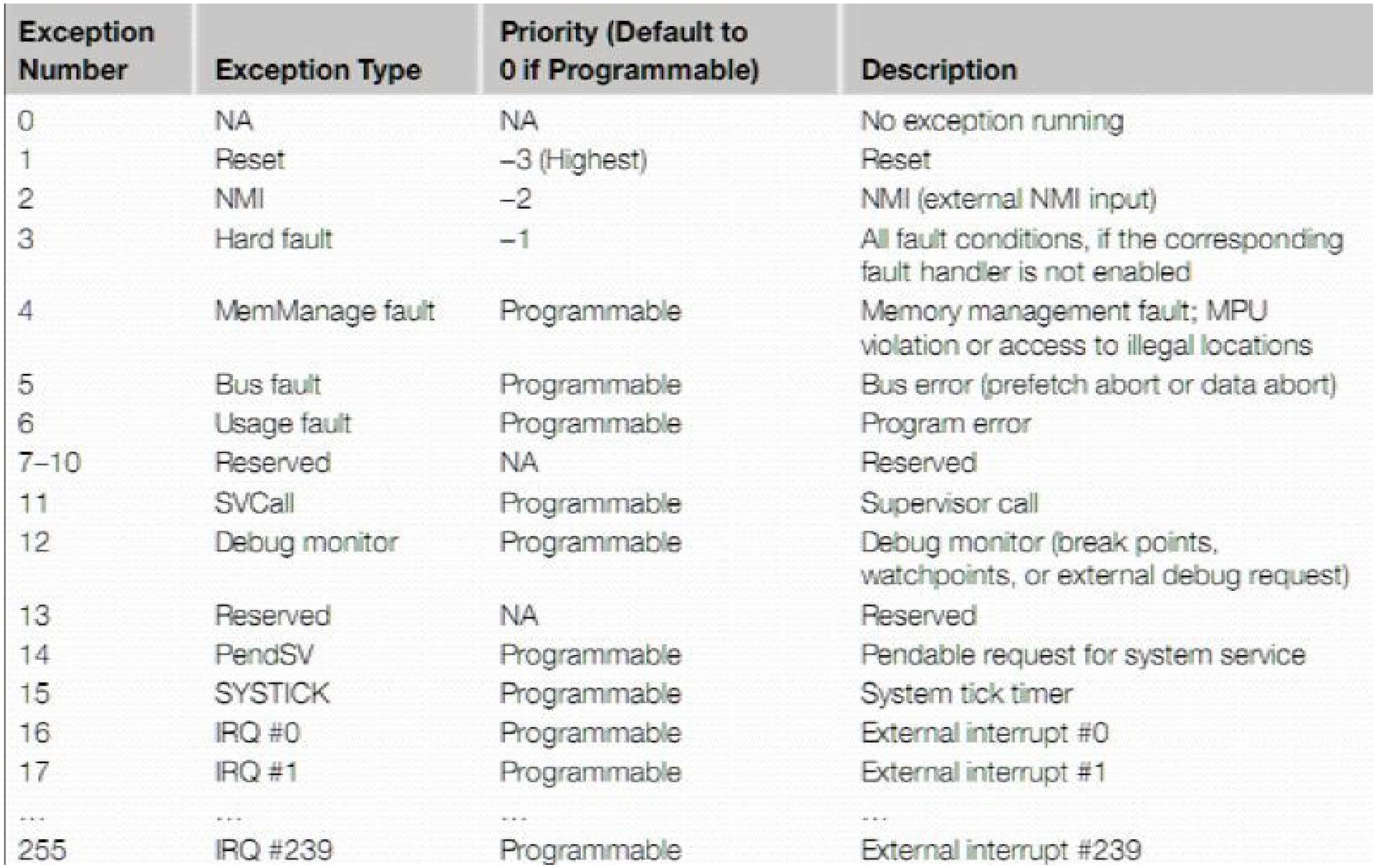




Excepciones. Interrupciones.

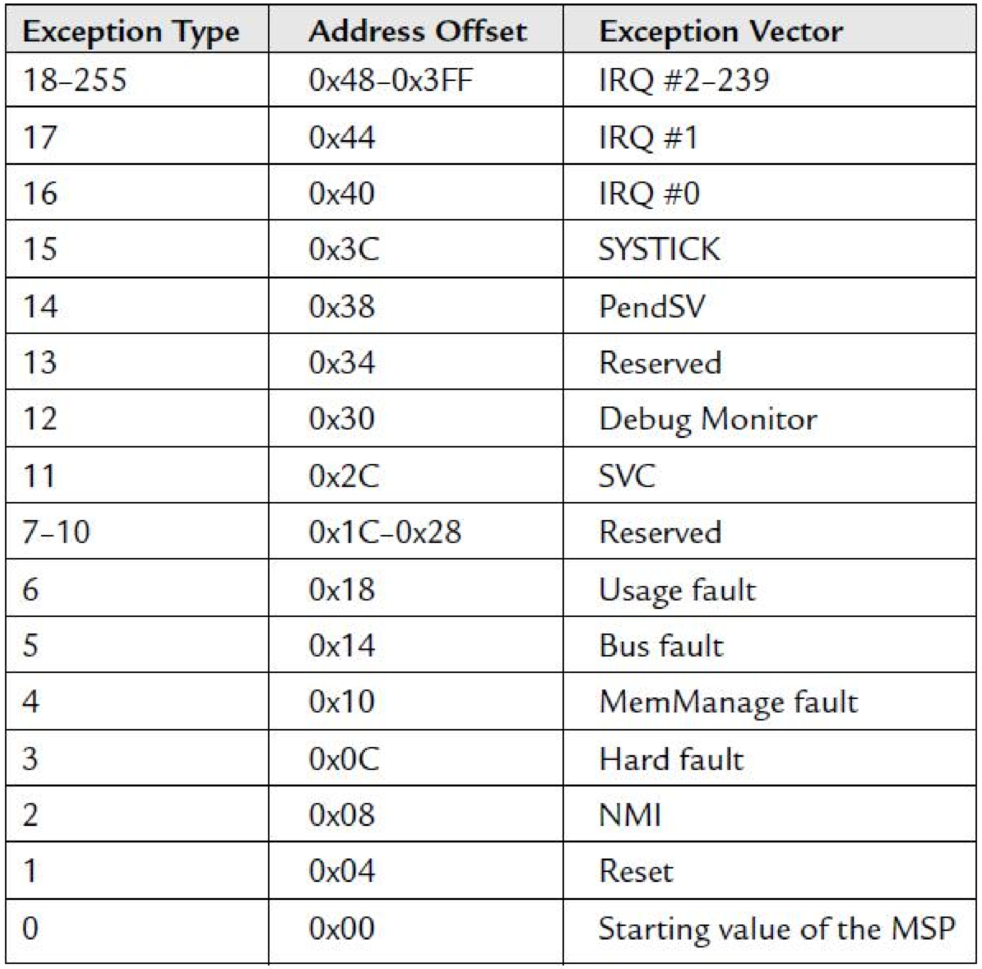
- **Interrupción.** Evento ***relacionado con el hardware*** que altera el flujo normal del programa en ejecución.
- **Excepción.** Evento que altera el flujo normal de un programa en ejecución. El intento de acceder a memoria no válida es un caso típico de excepción.
- Todos los Cortex-M3 tienen el ***mismo*** controlador de excepciones e interrupciones el ***NVIC –***Nested Vectored Interrupt Controller(Controlador de Interrupciones Anidadas y Vectorizadas).
- El vector de excepciones e interrupciones admite 256 entradas. Las primeras 16 son estándar de la arquitectura, y las restantes 240 son específicas del microcontrolador, definadaspor el fabricante.

Excepciones.



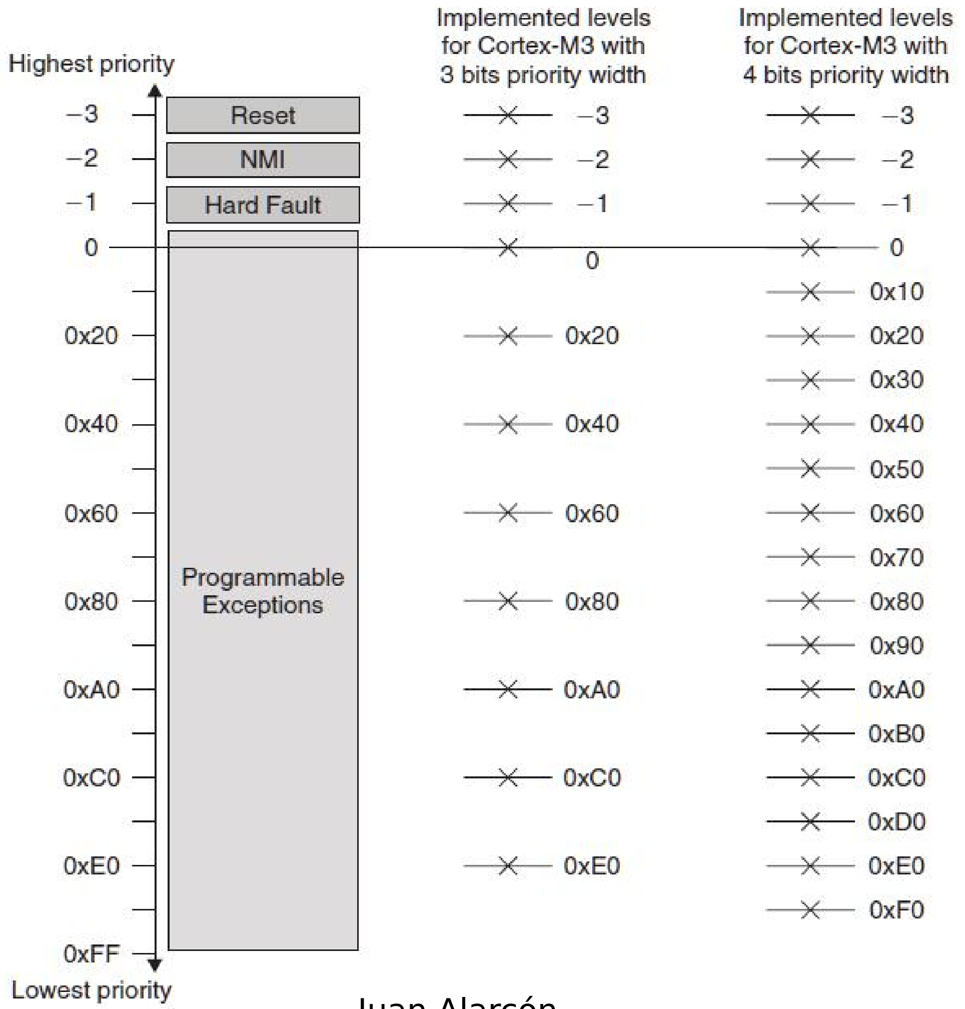
Exception Number	Exception Type	Priority (Default to 0 if Programmable)	Description
0	NA	NA	No exception running
1	Reset	–3 (Highest)	Reset
2	NMI	–2	NMI (external NMI input)
3	Hard fault	–1	All fault conditions, if the corresponding fault handler is not enabled
4	MemManage fault	Programmable	Memory management fault; MPU violation or access to illegal locations
5	Bus fault	Programmable	Bus error (prefetch abort or data abort)
6	Usage fault	Programmable	Program error
7–10	Reserved	NA	Reserved
11	SVCall	Programmable	Supervisor call
12	Debug monitor	Programmable	Debug monitor (break points, watchpoints, or external debug request)
13	Reserved	NA	Reserved
14	PendSV	Programmable	Pendable request for system service
15	SYSTICK	Programmable	System tick timer
16	IRQ #0	Programmable	External interrupt #0
17	IRQ #1	Programmable	External interrupt #1
...	...	...	...
255	IRQ #239	Programmable	External interrupt #239

Vector de Excepciones.

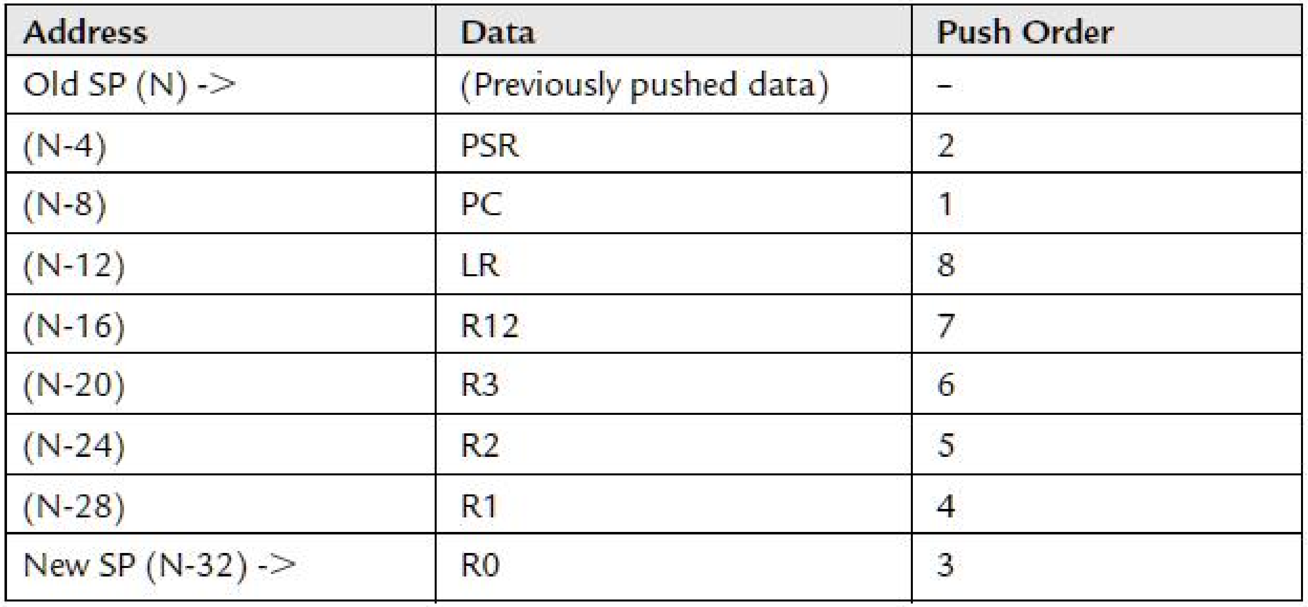


Exception Type	Address Offset	Exception Vector
18–255	0x48–0x3FF	IRQ #2–239
17	0x44	IRQ #1
16	0x40	IRQ #0
15	0x3C	SYSTICK
14	0x38	PendSV
13	0x34	Reserved
12	0x30	Debug Monitor
11	0x2C	SVC
7–10	0x1C–0x28	Reserved
6	0x18	Usage fault
5	0x14	Bus fault
4	0x10	MemManage fault
3	0x0C	Hard fault
2	0x08	NMI
1	0x04	Reset
0	0x00	Starting value of the MSP

Prioridades.



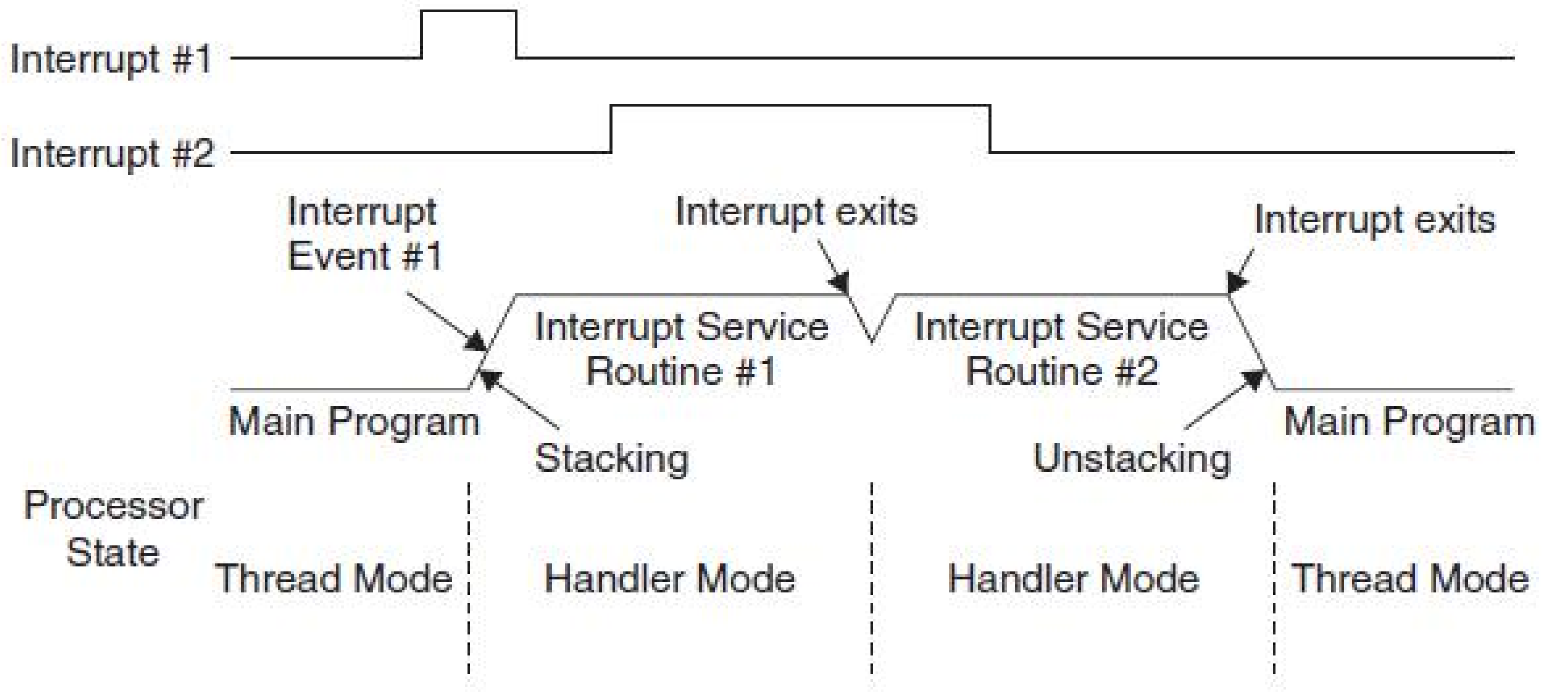
Stacking



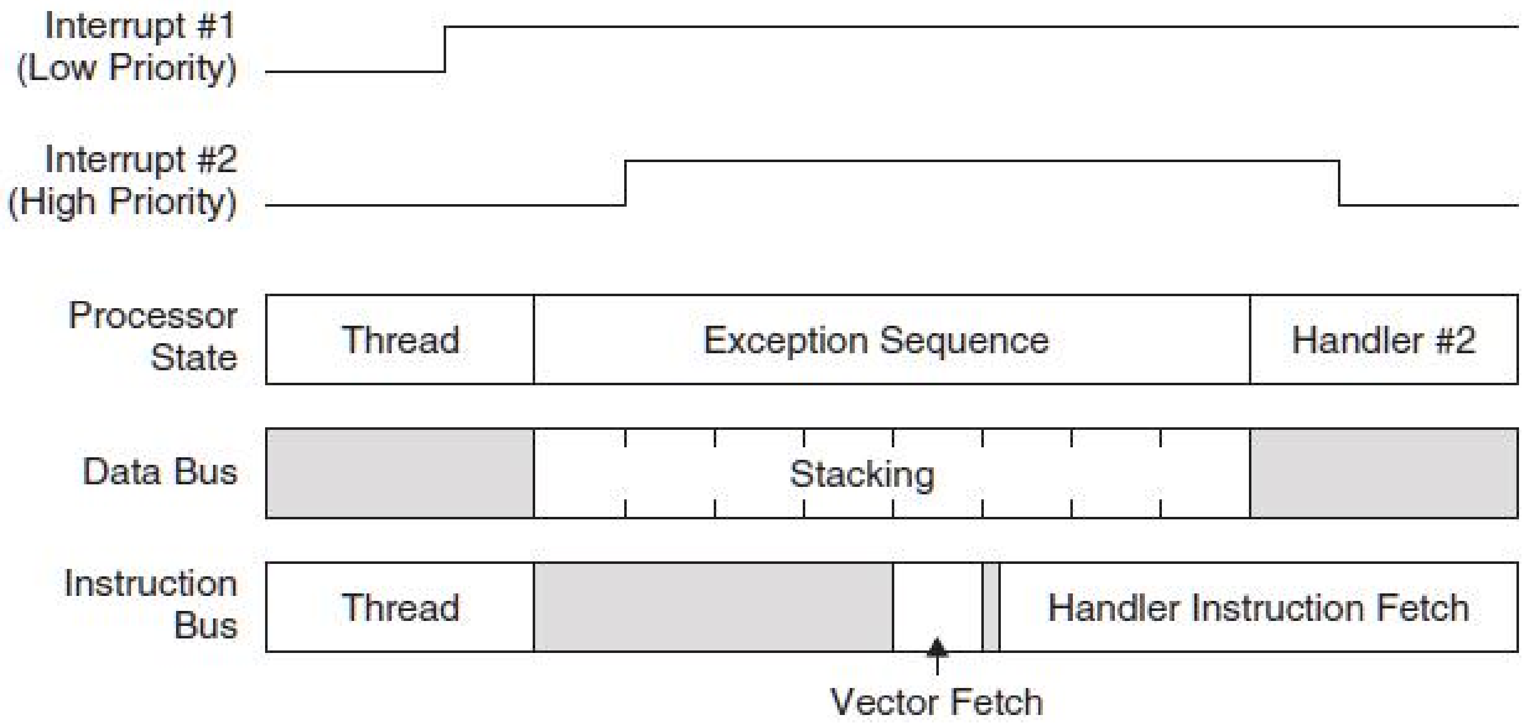
Address	Data	Push Order
Old SP (N) ->	(Previously pushed data)	-
(N-4)	PSR	2
(N-8)	PC	1
(N-12)	LR	8
(N-16)	R12	7
(N-20)	R3	6
(N-24)	R2	5
(N-28)	R1	4
New SP (N-32) ->	R0	3

- Cuando ocurre una excepción se envían a la pila los registros PC, PSR, R0-R3, R12 y LR.
- Si el procesador está en modo privilegiado usa el MSP si está en modo usuario usa el PSP.

Tail Chaining



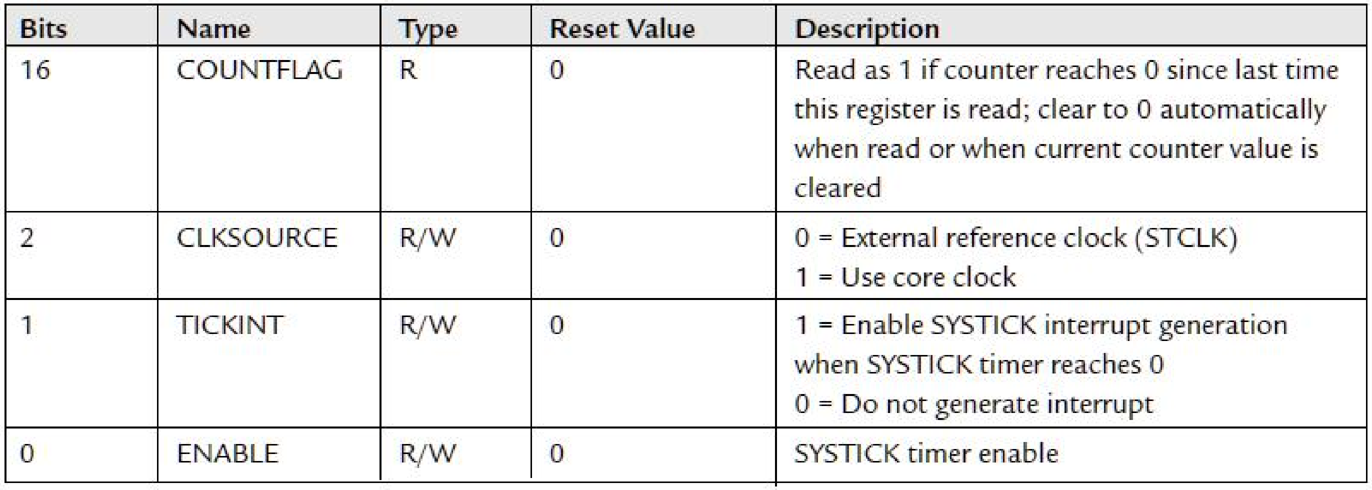
Late Arrivals



Juan Alarcón.
jalarcon@frba.utn.edu.ar

Systick.

- El Systick es un timer sencillo integrado con el NVIC.
- Es un contador descendente de 24 bits.
- Se lo va a utilizar generalmente como “ticks” del sis



Bits	Name	Type	Reset Value	Description
16	COUNTFLAG	R	0	Read as 1 if counter reaches 0 since last time this register is read; clear to 0 automatically when read or when current counter value is cleared
2	CLKSOURCE	R/W	0	0 = External reference clock (STCLK) 1 = Use core clock
1	TICKINT	R/W	0	1 = Enable SYSTICK interrupt generation when SYSTICK timer reaches 0 0 = Do not generate interrupt
0	ENABLE	R/W	0	SYSTICK timer enable

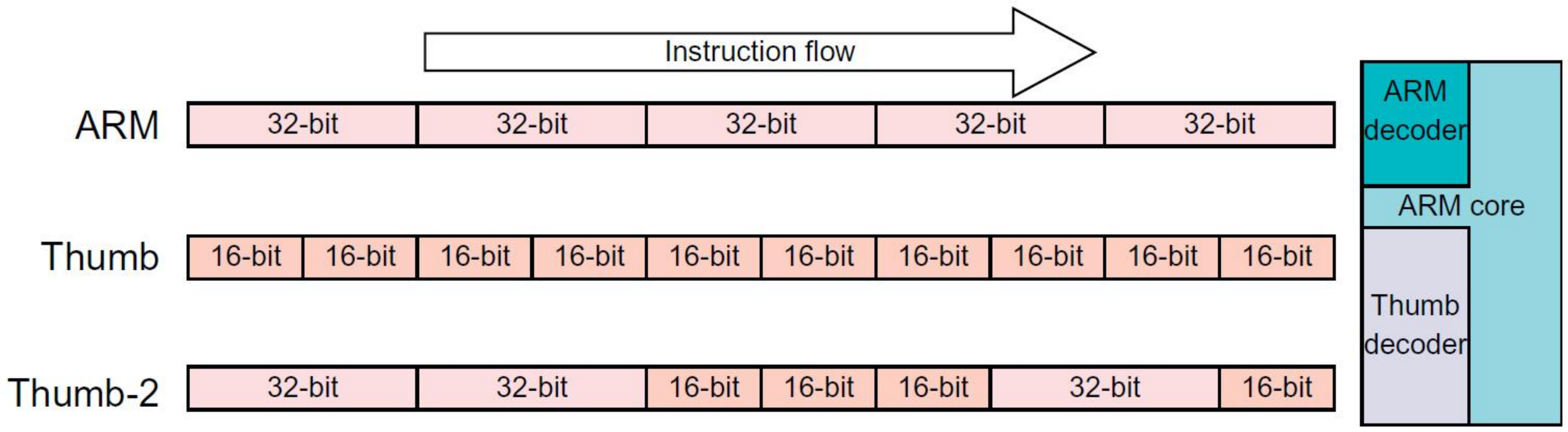
Systick (2)

Bits	Name	Type	Reset Value	Description
23:0	RELOAD	R/W	0	Reload value when timer reaches 0

Bits	Name	Type	Reset Value	Description
23:0	CURRENT	R/Wc	0	Read to return current value of the timer. Write to clear counter to 0. Clearing of current value also clears COUNTFLAG in SYSTICK Control and Status Register

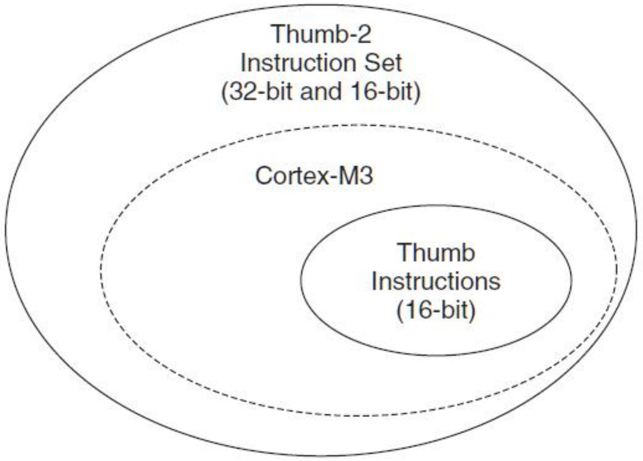
Bits	Name	Type	Reset Value	Description
31	NOREF	R	–	1 = No external reference clock (STCLK not available) 0 = External reference clock available
30	SKEW	R	–	1 = Calibration value is not exactly 10 ms 0 = Calibration value is accurate
23:0	TENMS	R/W	0	Calibration value for 10 ms.; chip designer should provide this value via Cortex-M3 input signals. If this value is read as 0, calibration value is not available

Set de Instrucciones.

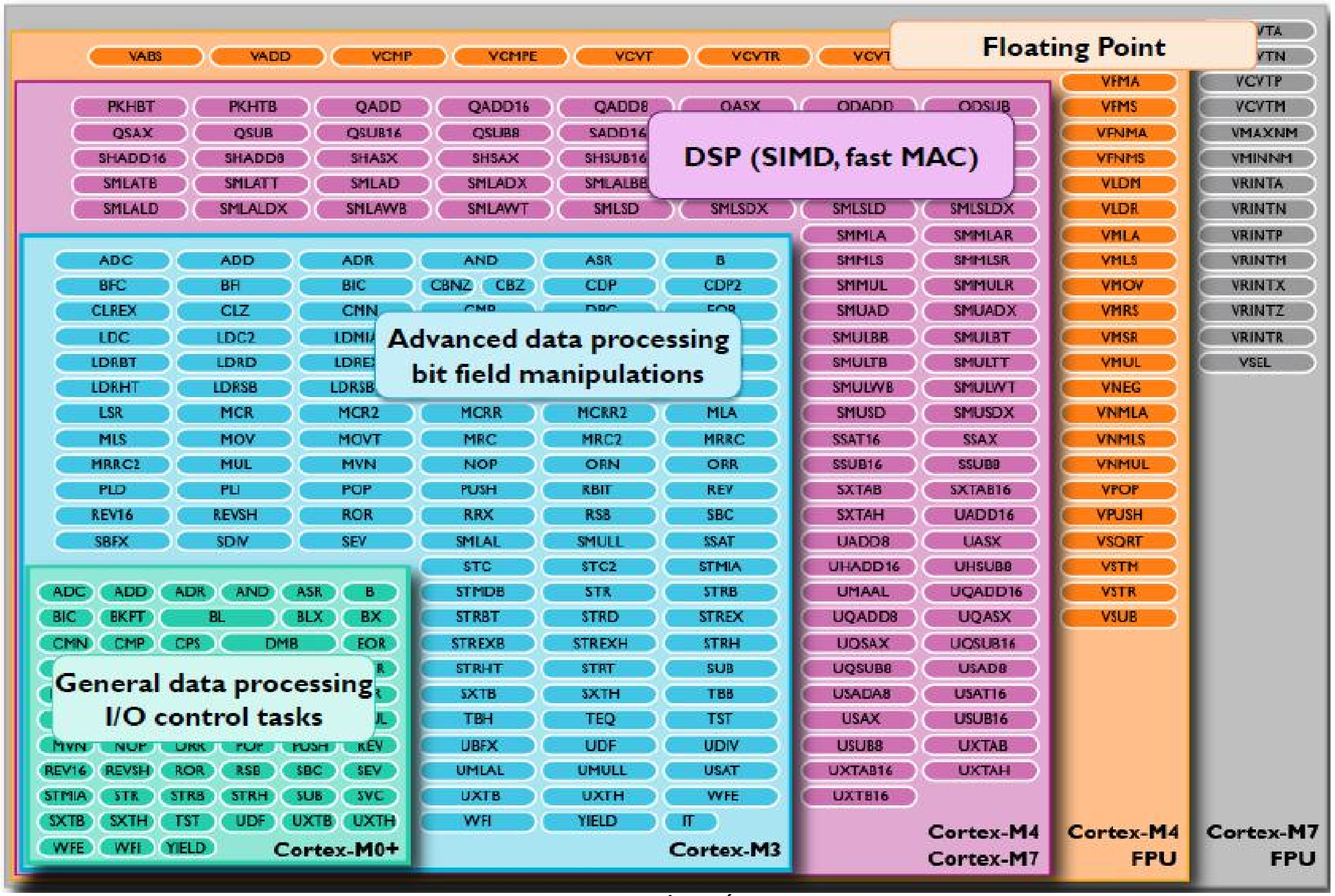


Set de Instrucciones Thumb-2

- Thumb-2. es un set de instrucciones de las instrucciones más comunes de ARM de 32bits en 16 bits.
- En ejecución el código de operación se "descomprimen" a instrucciones de 32 bits, siendo transparente al procesador.
- El uso del set de instrucciones Thumb-2 permite un mejor aprovechamiento de la memoria flash.
- Thumb-2, utiliza un conjunto de instrucciones mixto de 32bits y 16bits, descomprimiendo las de 16 bits a 32bits.
- El Cortex-M3 utiliza un subconjunto de las instrucciones del Thumb-2-

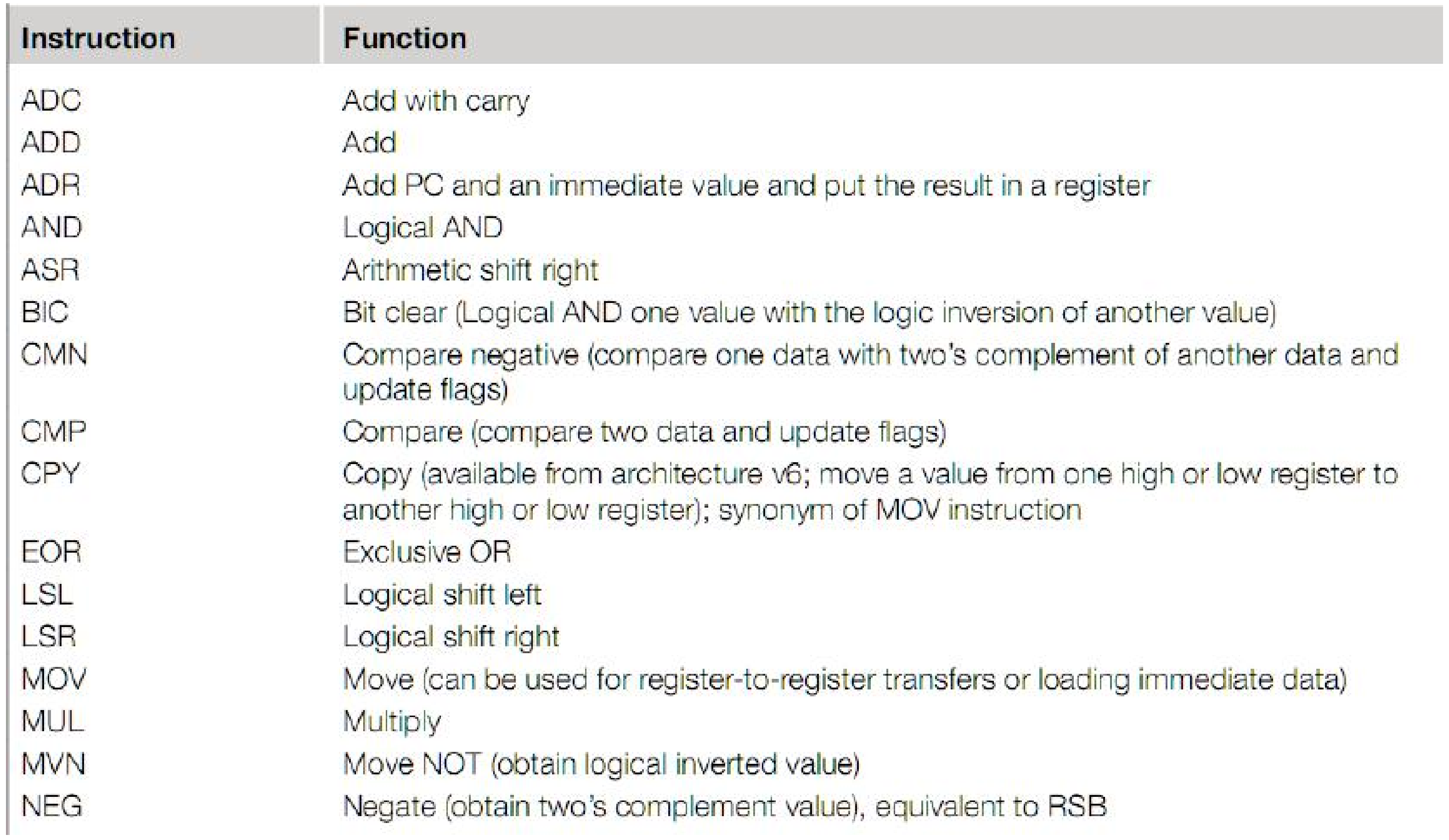

Thumb-2 Instruction Set (32-bit and 16-bit)
Cortex-M3
Thumb Instructions (16-bit)

Set de instrucciones Thumb-2.



Floating Point
VABS VADD VCMP VCMPE VCVT VCVTR VCVT
VFMA VFMS VFNMA VFNMS VLDM VLDR VMLA VMLS VMOV VMRS VMSR VMUL VNEG VNMLA VNMLS VNMUL VPOP VPUSH VSQRT VSTM VSTR VSUB
VTA VTN VCVTP VCVTM VMAXNM VMINNM VRINTA VRINTN VRINTP VRINTM VRINTX VRINTZ VRINTR VSEL
DSP (SIMD, fast MAC)
PKHBT PKHTB QADD QADD16 QADD8 QASX QDADD QDSUB
QSAX QSUB QSUB16 QSUB8 SADD16
SHADD16 SHADD8 SHASX SHSAX SHSUB16
SMLATB SMLATT SMLAD SMLADX SMLALBB
SMLALD SMLALDX SMLAWB SMLAWT SMLSD SMLSDX SMLSLD SMLSLDX
SMMLA SMMLAR SMMLS SMMLSR SMMUL SMMULR SMUAD SMUADX SMULBB SMULBT SMULTB SMULTT SMULWB SMULWT SMUSD SMUSDX SSAT16 SSAX SSUB16 SSUB8 SXTAB SXTAB16 SXTAH UADD16 UADD8 UASX UHADD16 UHSUB8 UMAAL UQADD16 UQADD8 UQASX UQSAX UQSUB16 UQSUB8 USAD8 USADA8 USAT16 USAX USUB16 USUB8 UXTAB UXTAB16 UXTAH UXTB16
Advanced data processing
bit field manipulations
ADC ADD ADR AND ASR B
BFC BFI BIC CBNZ CBZ CDP CDP2
CLREX CLZ CMN CMP DBG EOR
LDC LDC2 LDMIA
LDRBT LDRD LDREX
LDRHT LDRSB LDRSB
LSR MCR MCR2 MCRR MCRR2 MLA
MLS MOV MOVT MRC MRC2 MRRC
MRRC2 MUL MVN NOP ORN ORR
PLD PLI POP PUSH RBIT REV
REV16 REVSH ROR RRX RSB SBC
SBFX SDIV SEV SMLAL SMULL SSAT
STC STC2 STMIA
STMDB STR STRB
STRBT STRD STREX
STREXB STREXH STRH
STRHT STRT SUB
SXTB SXTH TBB
TBH TEQ TST
UBFX UDF UDIV
UMLAL UMULL USAT
UXTB UXTH WFE
WFI YIELD IT
General data processing
I/O control tasks
ADC ADD ADR AND ASR B
BIC BKPT BL BLX BX
CMN CMP CPS DMB EOR
MVN NOP ORR POP PUSH REV
REV16 REVSH ROR RSB SBC SEV
STMIA STR STRB STRH SUB SVC
SXTB SXTH TST UDF UXTB UXTH
WFE WFI YIELD
Cortex-M0+
Cortex-M3
Cortex-M4
Cortex-M7
Cortex-M4
FPU
Cortex-M7
FPU

Set de Instrucciones Thumb-2



Instruction	Function
ADC	Add with carry
ADD	Add
ADR	Add PC and an immediate value and put the result in a register
AND	Logical AND
ASR	Arithmetic shift right
BIC	Bit clear (Logical AND one value with the logic inversion of another value)
CMN	Compare negative (compare one data with two's complement of another data and update flags)
CMP	Compare (compare two data and update flags)
CPY	Copy (available from architecture v6; move a value from one high or low register to another high or low register); synonym of MOV instruction
EOR	Exclusive OR
LSL	Logical shift left
LSR	Logical shift right
MOV	Move (can be used for register-to-register transfers or loading immediate data)
MUL	Multiply
MVN	Move NOT (obtain logical inverted value)
NEG	Negate (obtain two's complement value), equivalent to RSB

Set de Instrucciones Thumb-2

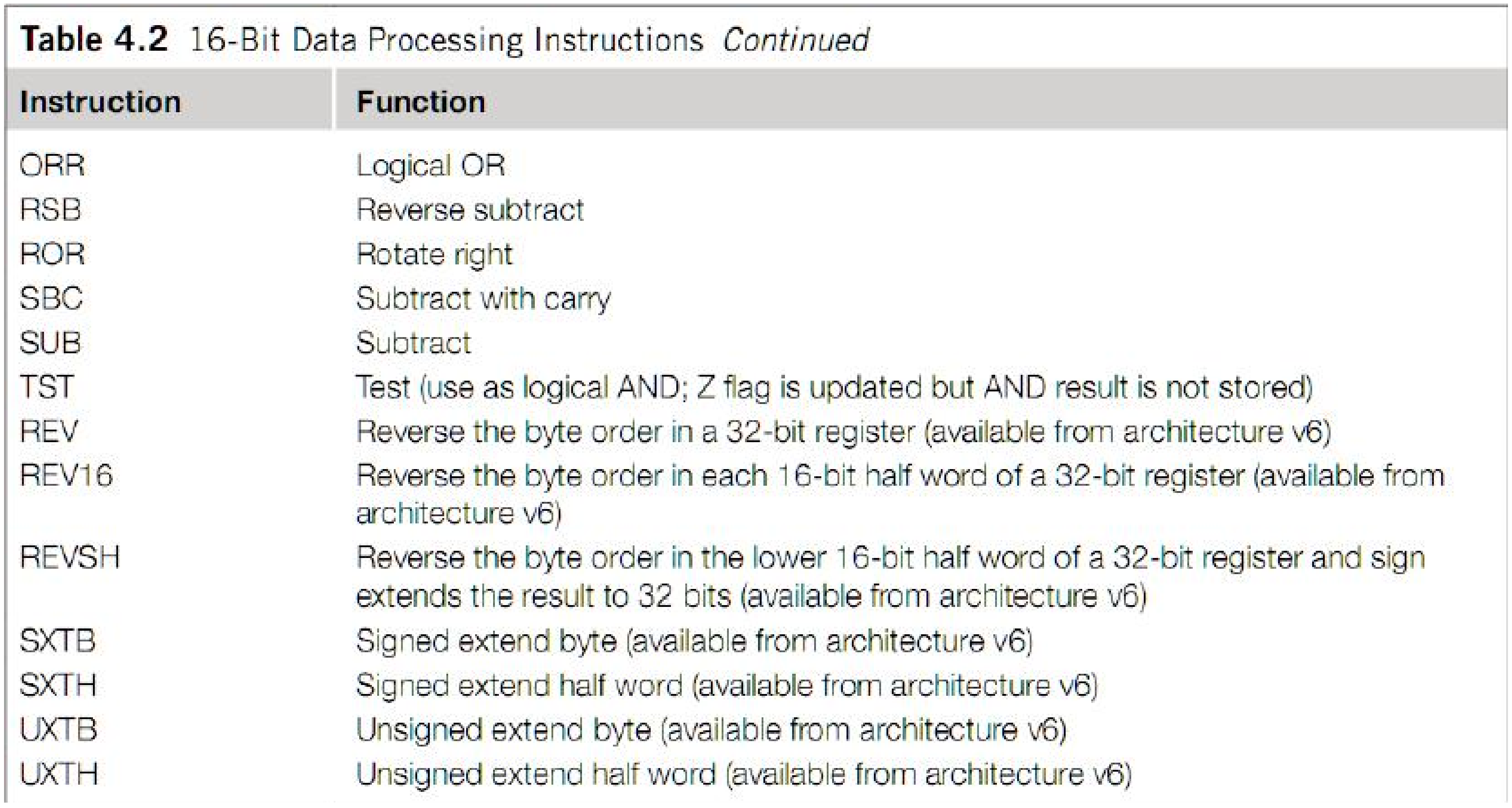
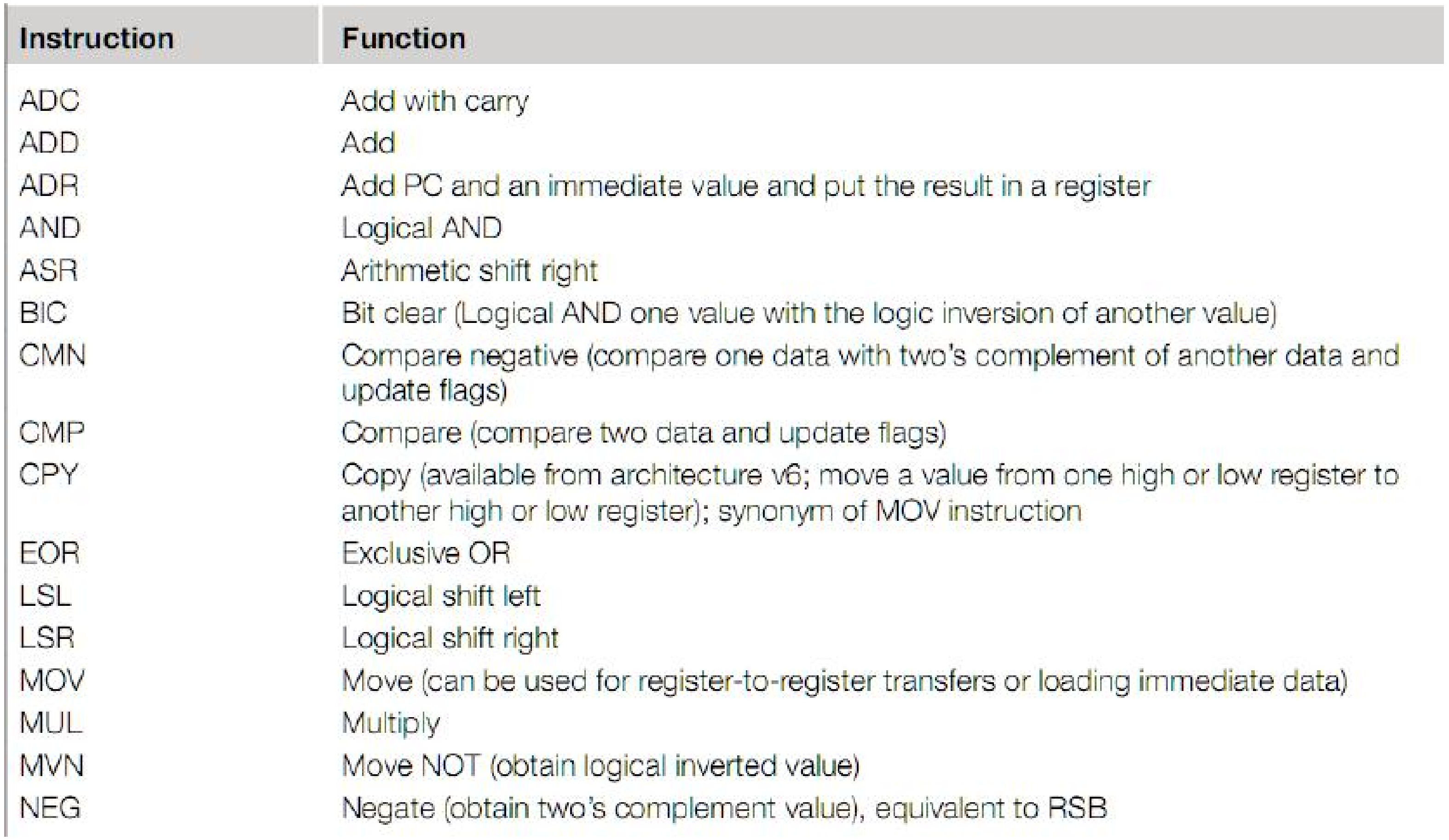


Table 4.2 16-Bit Data Processing Instructions *Continued*

Instruction	**Function**
ORR	Logical OR
RSB	Reverse subtract
ROR	Rotate right
SBC	Subtract with carry
SUB	Subtract
TST	Test (use as logical AND; Z flag is updated but AND result is not stored)
REV	Reverse the byte order in a 32-bit register (available from architecture v6)
REV16	Reverse the byte order in each 16-bit half word of a 32-bit register (available from architecture v6)
REVSH	Reverse the byte order in the lower 16-bit half word of a 32-bit register and sign extends the result to 32 bits (available from architecture v6)
SXTB	Signed extend byte (available from architecture v6)
SXTH	Signed extend half word (available from architecture v6)
UXTB	Unsigned extend byte (available from architecture v6)
UXTH	Unsigned extend half word (available from architecture v6)

Set de Instrucciones Thumb-2



Instruction	Function
ADC	Add with carry
ADD	Add
ADR	Add PC and an immediate value and put the result in a register
AND	Logical AND
ASR	Arithmetic shift right
BIC	Bit clear (Logical AND one value with the logic inversion of another value)
CMN	Compare negative (compare one data with two's complement of another data and update flags)
CMP	Compare (compare two data and update flags)
CPY	Copy (available from architecture v6; move a value from one high or low register to another high or low register); synonym of MOV instruction
EOR	Exclusive OR
LSL	Logical shift left
LSR	Logical shift right
MOV	Move (can be used for register-to-register transfers or loading immediate data)
MUL	Multiply
MVN	Move NOT (obtain logical inverted value)
NEG	Negate (obtain two's complement value), equivalent to RSB

Set de Instrucciones Thumb-2

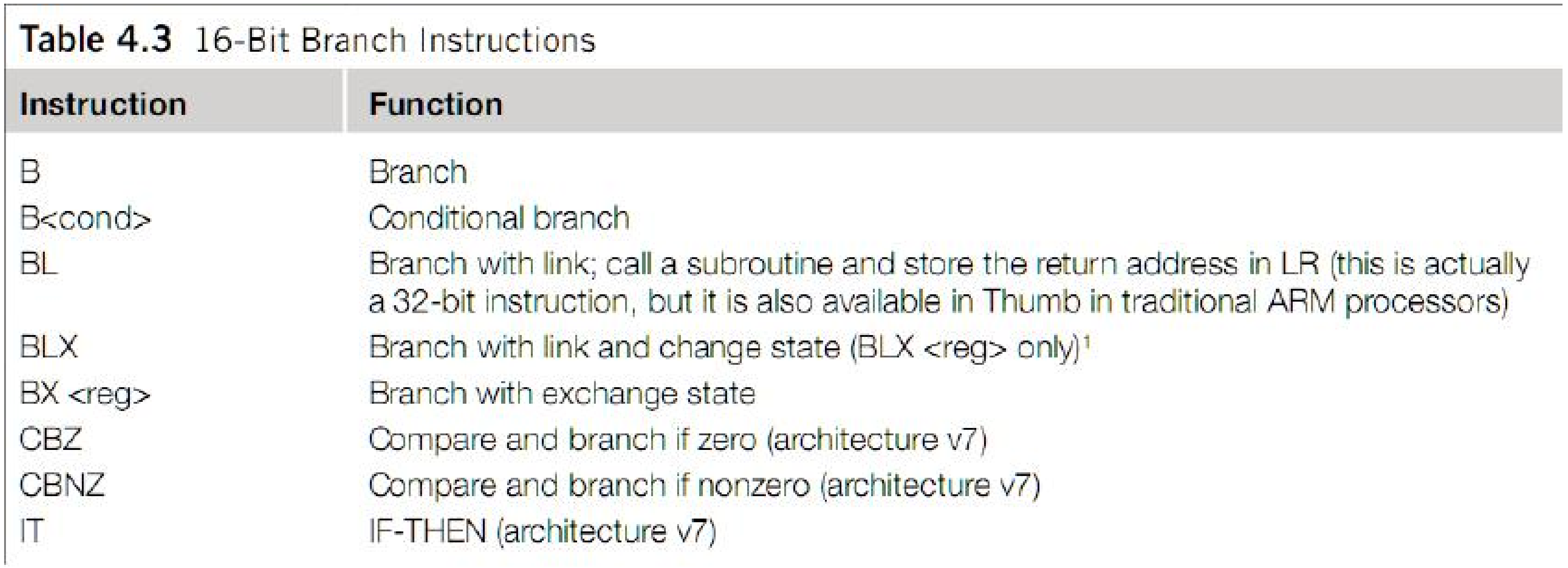


Table 4.3 16-Bit Branch Instructions

Instruction	Function
B	Branch
B<cond>	Conditional branch
BL	Branch with link; call a subroutine and store the return address in LR (this is actually a 32-bit instruction, but it is also available in Thumb in traditional ARM processors)
BLX	Branch with link and change state (BLX <reg> only)[1]
BX <reg>	Branch with exchange state
CBZ	Compare and branch if zero (architecture v7)
CBNZ	Compare and branch if nonzero (architecture v7)
IT	IF-THEN (architecture v7)

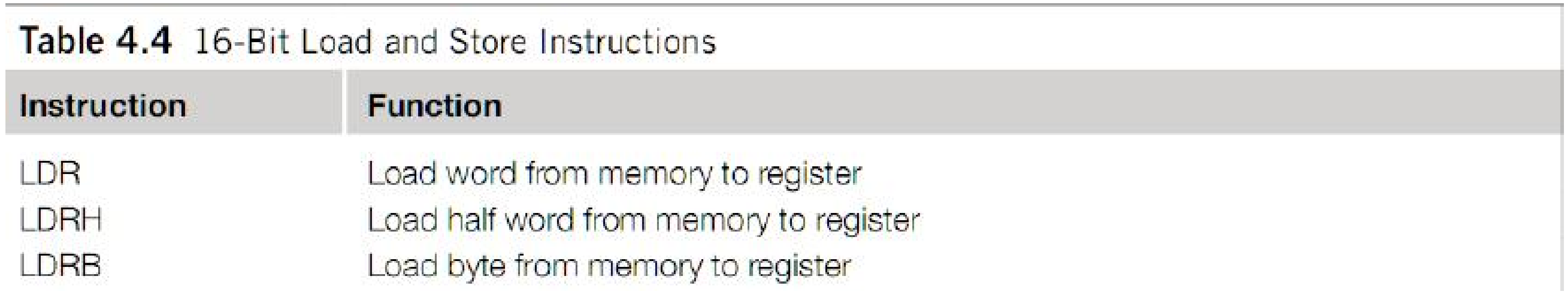


Table 4.4 16-Bit Load and Store Instructions

Instruction	Function
LDR	Load word from memory to register
LDRH	Load half word from memory to register
LDRB	Load byte from memory to register

Set de Instrucciones Thumb-2

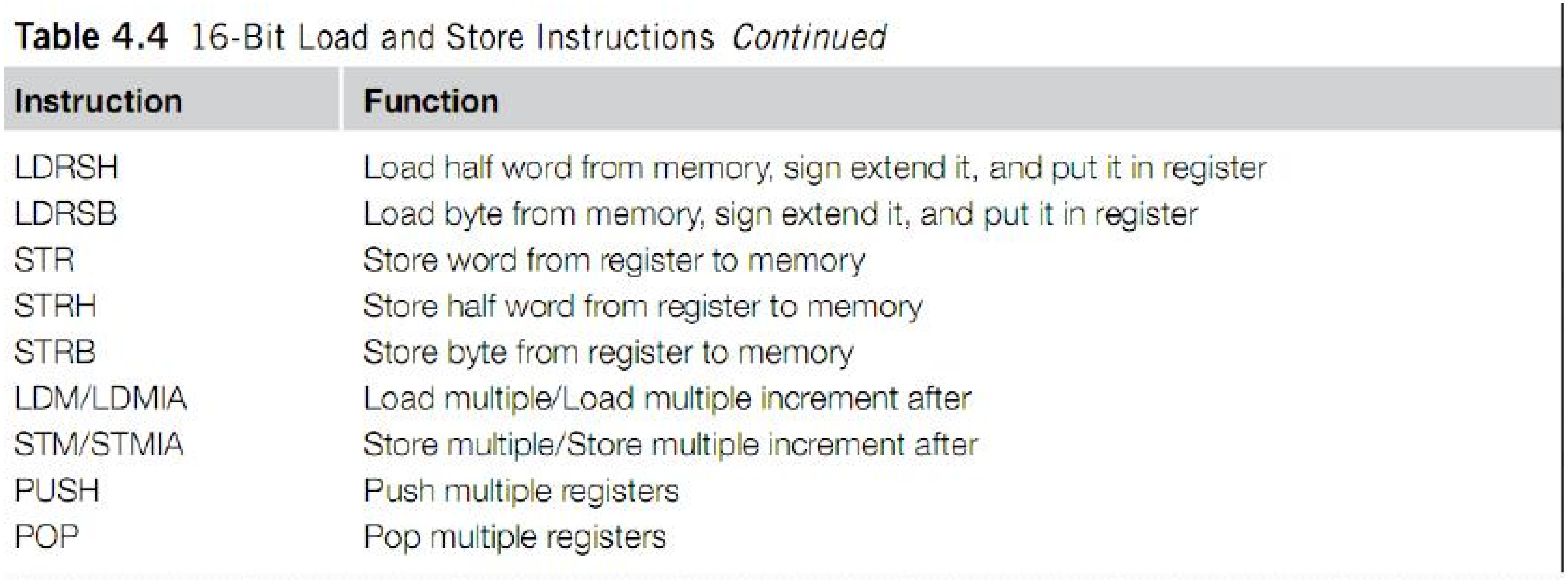


Table 4.4 16-Bit Load and Store Instructions *Continued*

Instruction	Function
LDRSH	Load half word from memory, sign extend it, and put it in register
LDRSB	Load byte from memory, sign extend it, and put it in register
STR	Store word from register to memory
STRH	Store half word from register to memory
STRB	Store byte from register to memory
LDM/LDMIA	Load multiple/Load multiple increment after
STM/STMIA	Store multiple/Store multiple increment after
PUSH	Push multiple registers
POP	Pop multiple registers

Set de Instrucciones Thumb-2

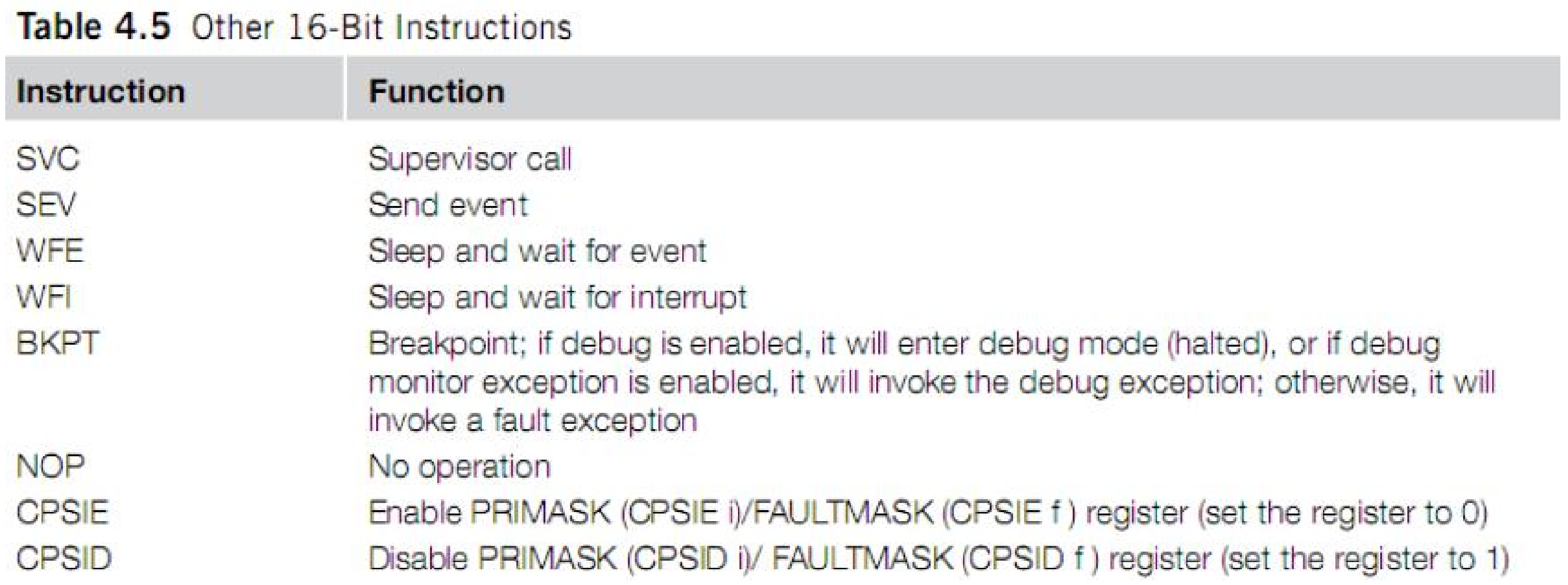


Table 4.5 Other 16-Bit Instructions

Instruction	Function
SVC	Supervisor call
SEV	Send event
WFE	Sleep and wait for event
WFI	Sleep and wait for interrupt
BKPT	Breakpoint; if debug is enabled, it will enter debug mode (halted), or if debug monitor exception is enabled, it will invoke the debug exception; otherwise, it will invoke a fault exception
NOP	No operation
CPSIE	Enable PRIMASK (CPSIE i)/FAULTMASK (CPSIE f) register (set the register to 0)
CPSID	Disable PRIMASK (CPSID i)/ FAULTMASK (CPSID f) register (set the register to 1)

Set de Instrucciones Thumb-2

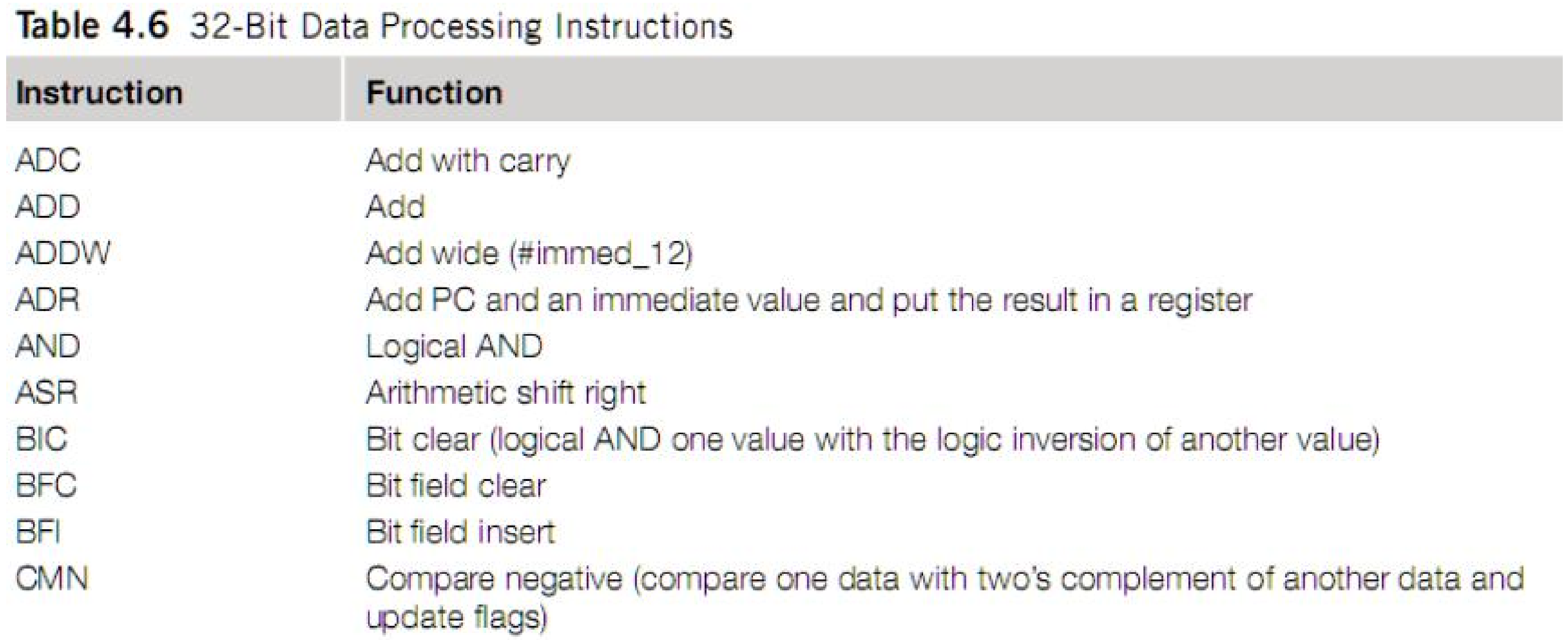


Table 4.6 32-Bit Data Processing Instructions

Instruction	Function
ADC	Add with carry
ADD	Add
ADDW	Add wide (#immed_12)
ADR	Add PC and an immediate value and put the result in a register
AND	Logical AND
ASR	Arithmetic shift right
BIC	Bit clear (logical AND one value with the logic inversion of another value)
BFC	Bit field clear
BFI	Bit field insert
CMN	Compare negative (compare one data with two's complement of another data and update flags)

Set de Instrucciones Thumb-2

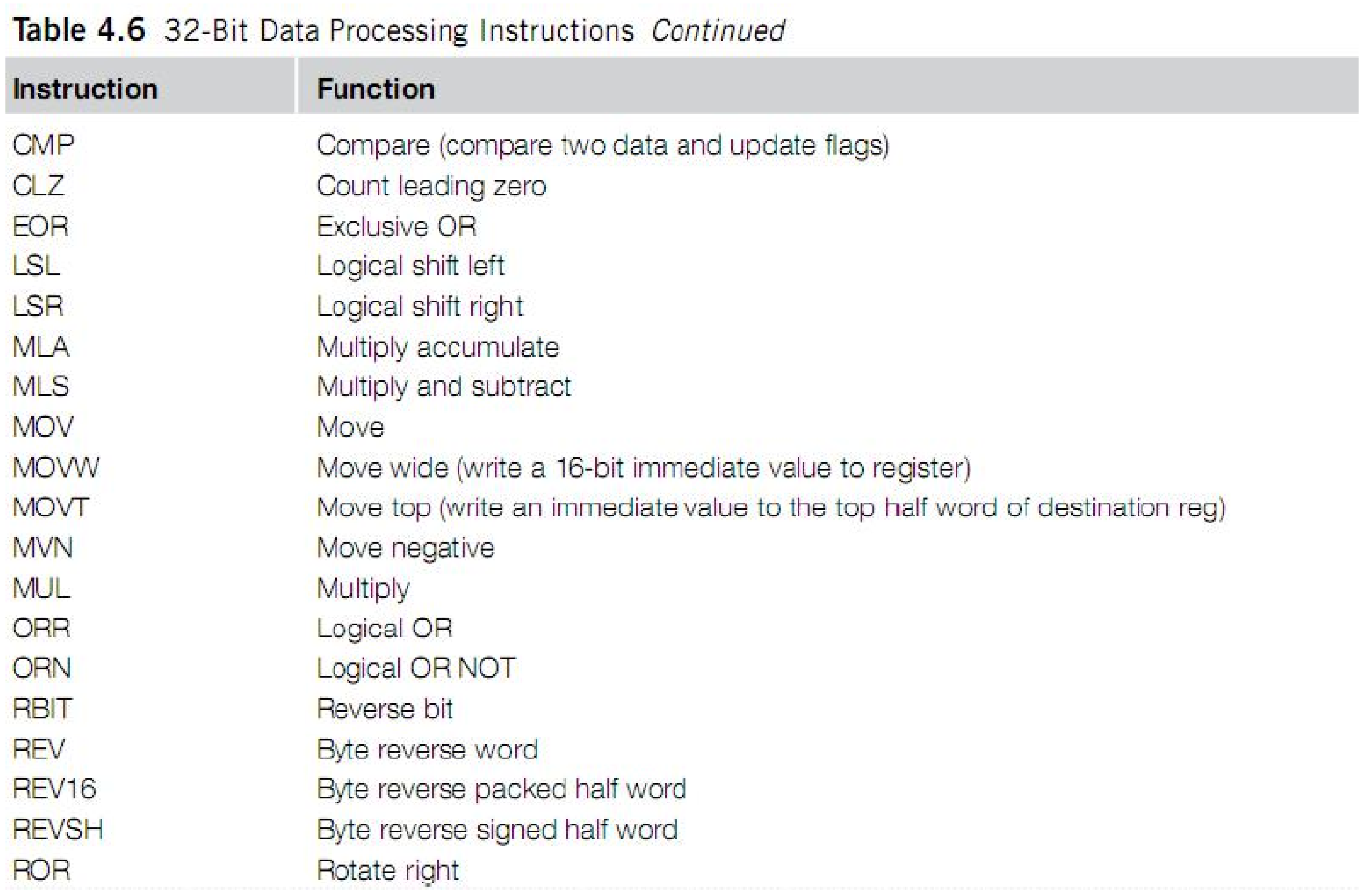
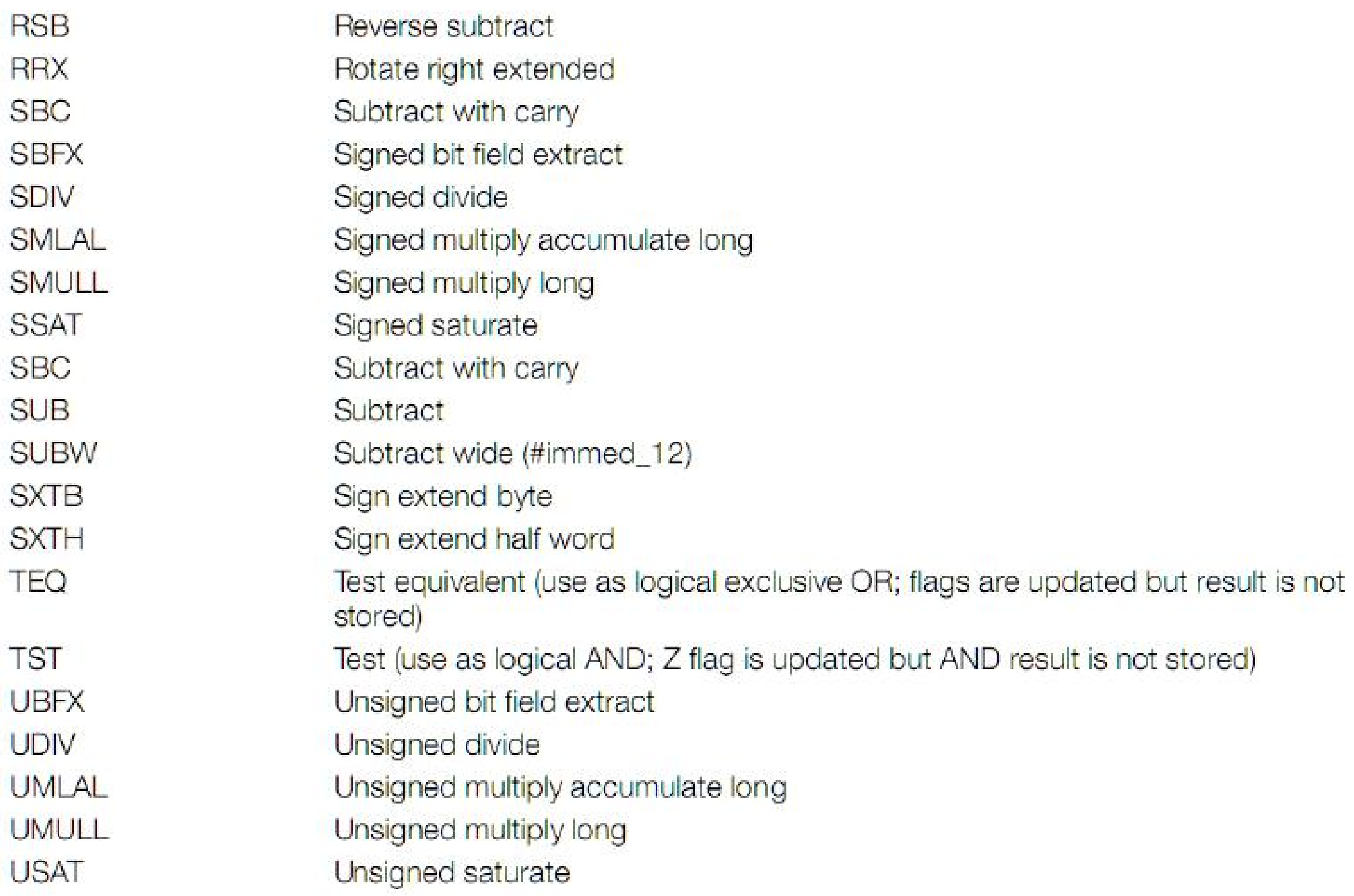


Table 4.6 32-Bit Data Processing Instructions *Continued*

Instruction	Function
CMP	Compare (compare two data and update flags)
CLZ	Count leading zero
EOR	Exclusive OR
LSL	Logical shift left
LSR	Logical shift right
MLA	Multiply accumulate
MLS	Multiply and subtract
MOV	Move
MOVW	Move wide (write a 16-bit immediate value to register)
MOVT	Move top (write an immediate value to the top half word of destination reg)
MVN	Move negative
MUL	Multiply
ORR	Logical OR
ORN	Logical OR NOT
RBIT	Reverse bit
REV	Byte reverse word
REV16	Byte reverse packed half word
REVSH	Byte reverse signed half word
ROR	Rotate right

Set de Instrucciones Thumb-2



RSB	Reverse subtract
RRX	Rotate right extended
SBC	Subtract with carry
SBFX	Signed bit field extract
SDIV	Signed divide
SMLAL	Signed multiply accumulate long
SMULL	Signed multiply long
SSAT	Signed saturate
SBC	Subtract with carry
SUB	Subtract
SUBW	Subtract wide (#immed_12)
SXTB	Sign extend byte
SXTH	Sign extend half word
TEQ	Test equivalent (use as logical exclusive OR; flags are updated but result is not stored)
TST	Test (use as logical AND; Z flag is updated but AND result is not stored)
UBFX	Unsigned bit field extract
UDIV	Unsigned divide
UMLAL	Unsigned multiply accumulate long
UMULL	Unsigned multiply long
USAT	Unsigned saturate

Set de Instrucciones Thumb-2

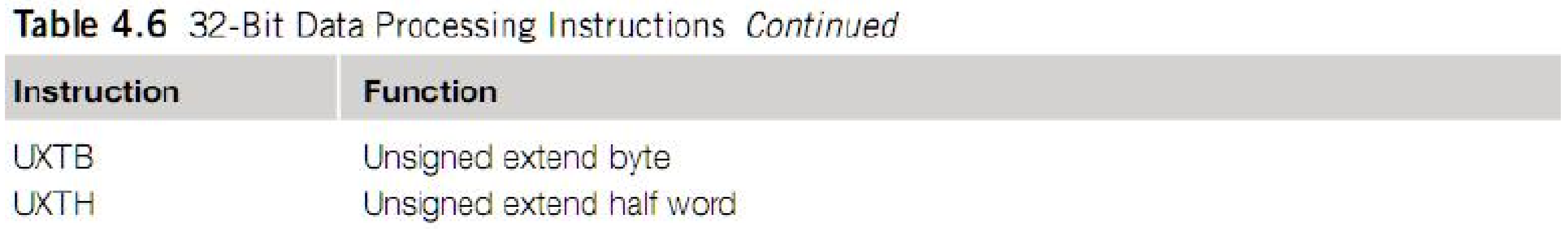


Table 4.6 32-Bit Data Processing Instructions *Continued*

Instruction	Function
UXTB	Unsigned extend byte
UXTH	Unsigned extend half word

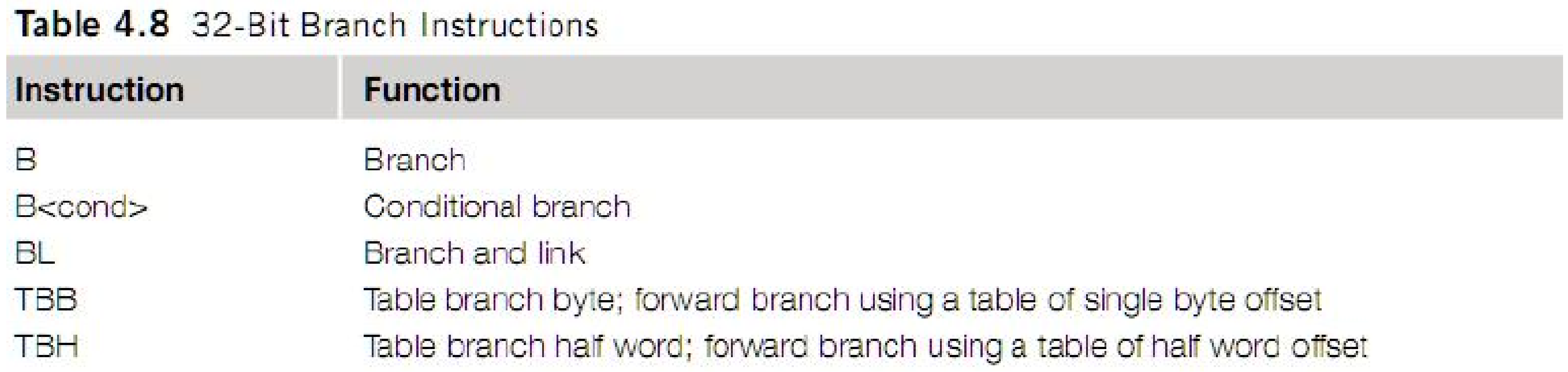


Table 4.8 32-Bit Branch Instructions

Instruction	Function
B	Branch
B<cond>	Conditional branch
BL	Branch and link
TBB	Table branch byte; forward branch using a table of single byte offset
TBH	Table branch half word; forward branch using a table of half word offset

Set de Instrucciones Thumb-2

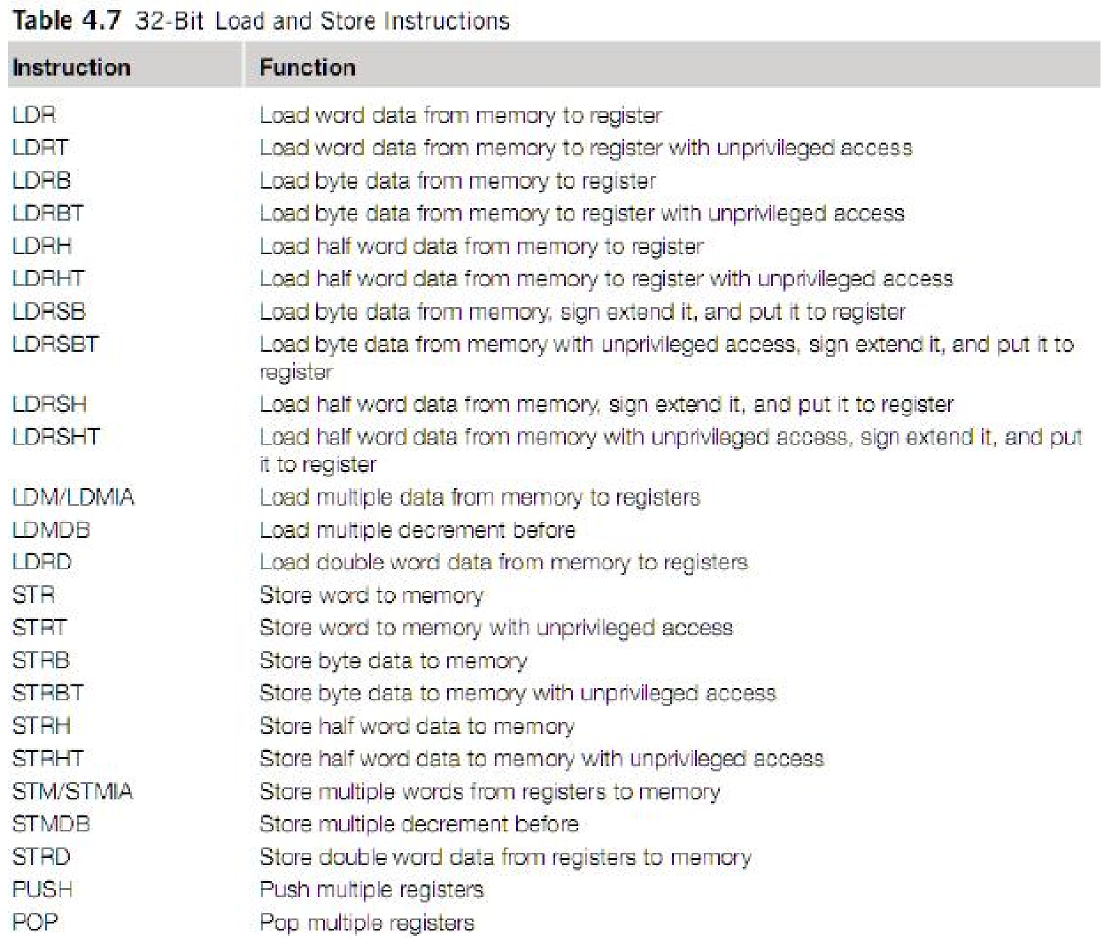


Table 4.7 32-Bit Load and Store Instructions

Instruction	Function
LDR	Load word data from memory to register
LDRT	Load word data from memory to register with unprivileged access
LDRB	Load byte data from memory to register
LDRBT	Load byte data from memory to register with unprivileged access
LDRH	Load half word data from memory to register
LDRHT	Load half word data from memory to register with unprivileged access
LDRSB	Load byte data from memory, sign extend it, and put it to register
LDRSBT	Load byte data from memory with unprivileged access, sign extend it, and put it to register
LDRSH	Load half word data from memory, sign extend it, and put it to register
LDRSHT	Load half word data from memory with unprivileged access, sign extend it, and put it to register
LDM/LDMIA	Load multiple data from memory to registers
LDMDB	Load multiple decrement before
LDRD	Load double word data from memory to registers
STR	Store word to memory
STRT	Store word to memory with unprivileged access
STRB	Store byte data to memory
STRBT	Store byte data to memory with unprivileged access
STRH	Store half word data to memory
STRHT	Store half word data to memory with unprivileged access
STM/STMIA	Store multiple words from registers to memory
STMDB	Store multiple decrement before
STRD	Store double word data from registers to memory
PUSH	Push multiple registers
POP	Pop multiple registers

Set de Instrucciones Thumb-2

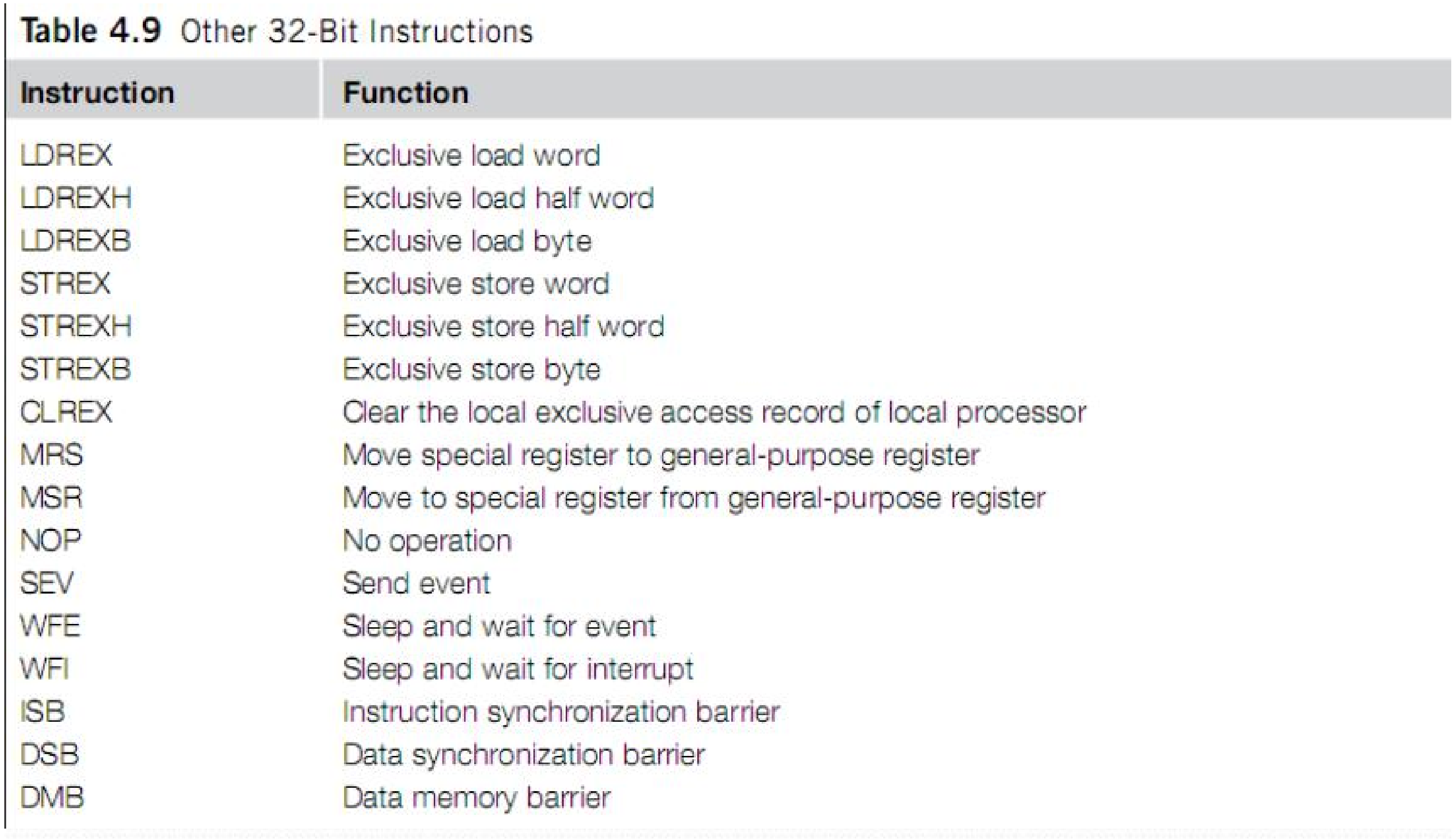
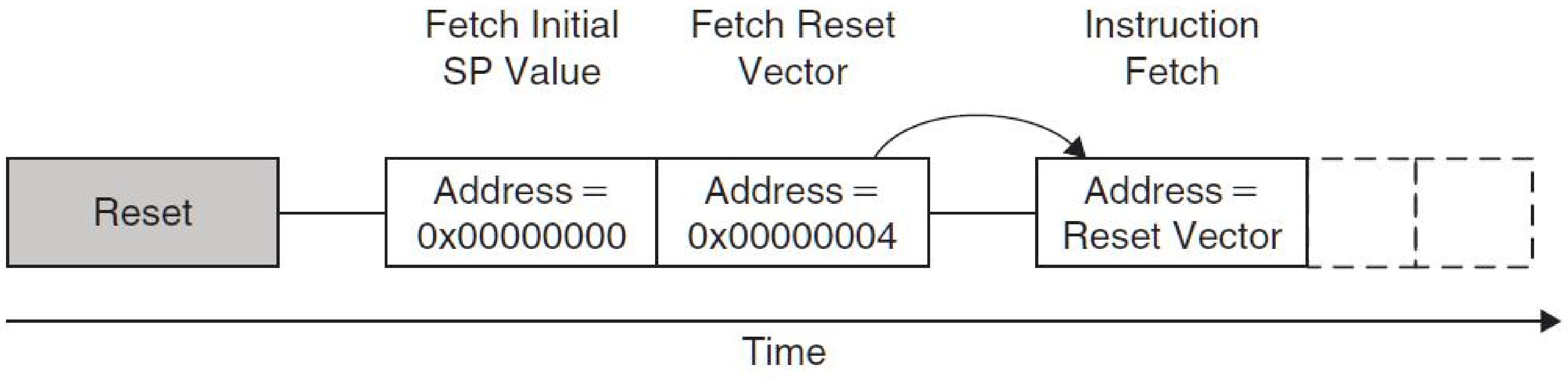


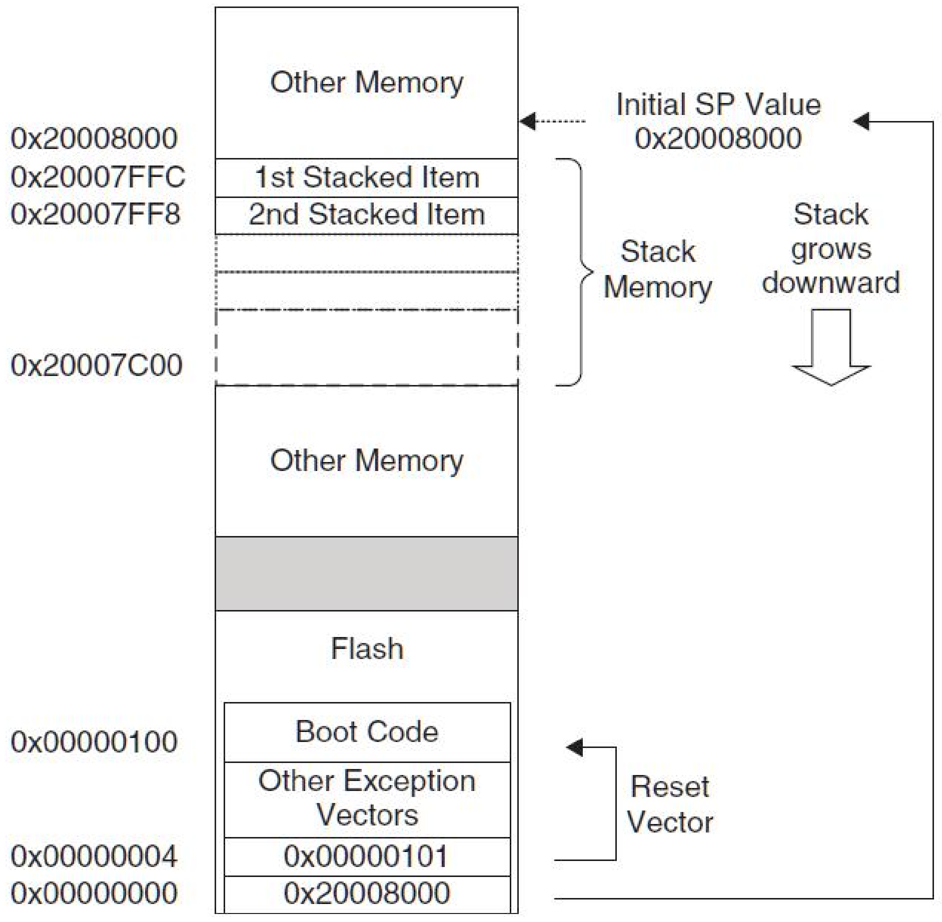
Table 4.9 Other 32-Bit Instructions

Instruction	Function
LDREX	Exclusive load word
LDREXH	Exclusive load half word
LDREXB	Exclusive load byte
STREX	Exclusive store word
STREXH	Exclusive store half word
STREXB	Exclusive store byte
CLREX	Clear the local exclusive access record of local processor
MRS	Move special register to general-purpose register
MSR	Move to special register from general-purpose register
NOP	No operation
SEV	Send event
WFE	Sleep and wait for event
WFI	Sleep and wait for interrupt
ISB	Instruction synchronization barrier
DSB	Data synchronization barrier
DMB	Data memory barrier

Secuencia de Reset.



Secuencia de Reset.



Ejemplos de Código.

```
int cubo (int nro)
{
    return nro*nro*nro;
}


MOV     R1,R0
MUL     R0,R1,R1
MULS    R0,R1,R0
BX LR
```

```
int dividepor7 (int nro)
{
    return nro/7;
}


MOV     R1,R0
MOVS    R0, #0x07
SDIV    R0,R1,R0
BX LR
```

Bibliografía.

- The Definitive Guide to the ARM Cortex-M3, Second Edition – Joseph Yiu – Newnes – 2009.
- The Definitive Guide to the ARM Cortex-M0. Joseph Yiu. Elsevier. 2011.
- The Designer’s Guide to the Cortex-M Processor Family. Trevor Martin. Elsevier. 2013
- LPC17xx User manual.
- ARM®v7-M Architecture Reference Manual.
- Cortex™-M3 Revision: r1p1 Technical Reference Manual.