Arquitectura del Microcontrolador I

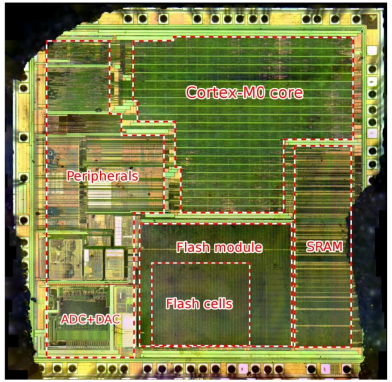
Temas

- Parte I
  - Introducción a la arquitectura
  - Procesador Cortex
  - Registros
  - Set de Instrucciones
  - Modelo de memoria

Introducción - Arquitectura

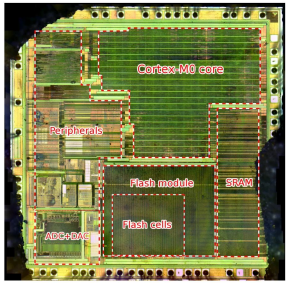
¿Qué es y por qué estudiarla?

- La “arquitectura” es una descripción de las estructuras que componen un sistema y su interacción.
- Se describe desde distintos niveles de abstracción y desde distintas “vistas” que en conjunto describen el sistema completo.
- Conocer la arquitectura permite:
  - Fase inicial: Tener los conocimientos básicos para trabajar con la herramienta
  - Fase avanzada: Obtener la mejor performance y depurar casos problemáticos



Arquitectura del **micro** **-procesador** **-controlador**

Arquitectura



Set de Instrucciones

- Modelo de programación
- Excepciones
- Modelo de memoria
- Arquitectura del set de Instrucciones (ISA)
- Estrategia de acceso a periféricos

Organización

- Registros del procesador
- Estructura del procesador
- Buses
- “Periféricos” del procesador
- Complementos de manejo de Interrupciones
- Estructura
- Buses
- Periféricos
- Acceso a memoria
- Reloj

Tecnología

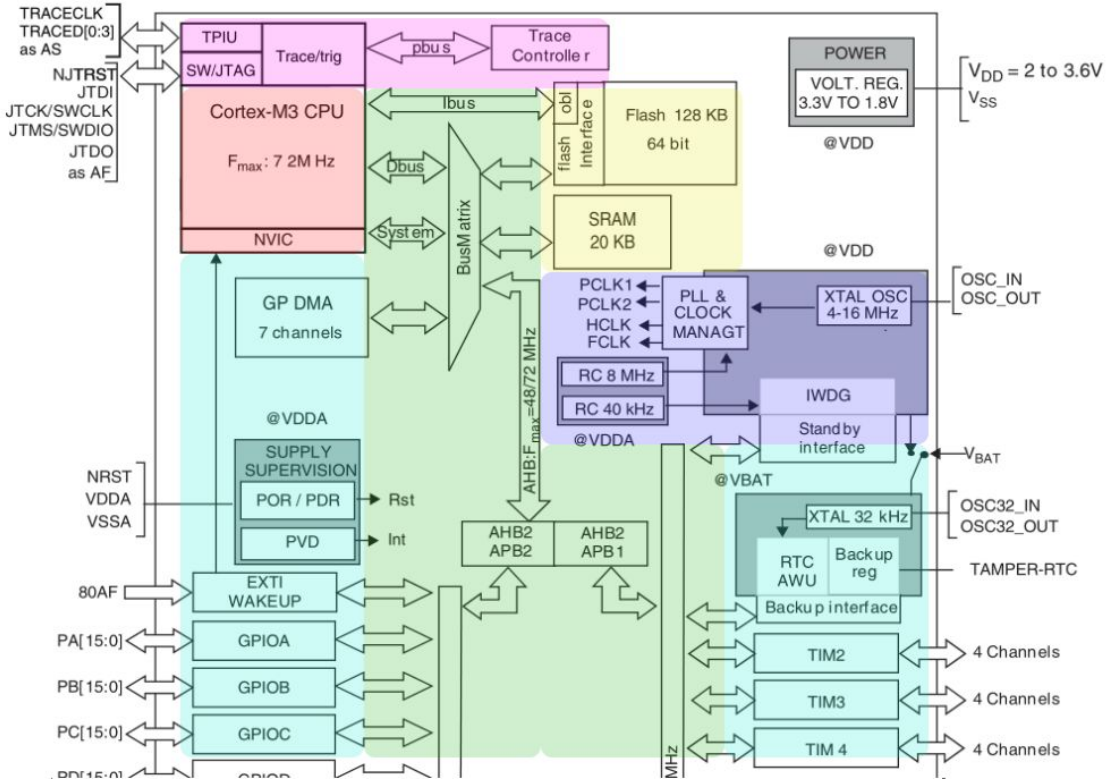
- Implementación micro(nano)electrónica
- Implementación de la memoria
- Características eléctricas

Y…. ¿a quién le importa?

- **Proc. ARM Cortex M3:** ¿Qué capacidades tiene mi µC?¿Hasta qué frecuencia de clock soporta?¿cuánto le toma ejecutar instrucciones?¿Puede trabajar con números reales?¿Es el más conveniente para aplicaciones de tiempo real?
- **Manejo de Excepciones:** Respuesta a eventos, posibilidad de implementar aplicaciones de tiempo real, velocidad de atención
- **Memoria:** Tamaño del programa que puedo cargar, tamaño de datos permanentes (elementos pre-programados para interfaces de usuario) y volátiles (procesamiento de señales)
- **Características de Debug:** El 90% del tiempo se utiliza: ¿qué información de debug puedo obtener? ¿Cómo? ¿A qué costo?
- **Buses:** ¿Limitación de velocidad de acceso? ¿Uso de periférico DMA?
- **Clock:** Tasas máximas, necesidades especiales según la precisión, costo
- **Alimentación:** Características de bajo consumo compatibles con el proyecto, facilidades en el manejo de baterías
- **Periféricos:** ¿Qué puede hacer?

Caso de estudio: STM32F103C8

- Proc. ARM Cortex M3
- Manejo de Excepciones
- Memoria
- Características de Debug
- Buses
- Temporización
- Alimentación
- Periféricos

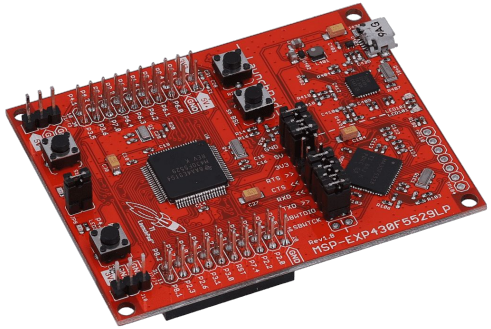


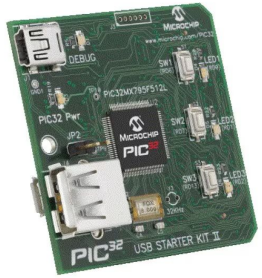
Procesador: Cortex de **ARM**

- ARM diseña micro**procesadores** y los licencia (no fabrica)
  - Por un lado, diseña arquitecturas de set de instrucciones, como la ARMv7-M
  - Por otro lado, diseña procesadores físicos o “cores” que pueden fabricarse
- ST le compra el diseño a ARM, le agrega buses y periféricos y así crea sus µCs
  - El diseño del procesador del **STM32F103** es un **Cortex-M3** de ARM
  - Por lo tanto, al describir la arquitectura, hablaremos en parte de los **Cortex-M3** y habrá parte de la documentación que corresponderá a los Cortex
- El desarrollo para un micro con core ARM tiene cierta “portabilidad” entre plataformas

ARM y otras arquitecturas

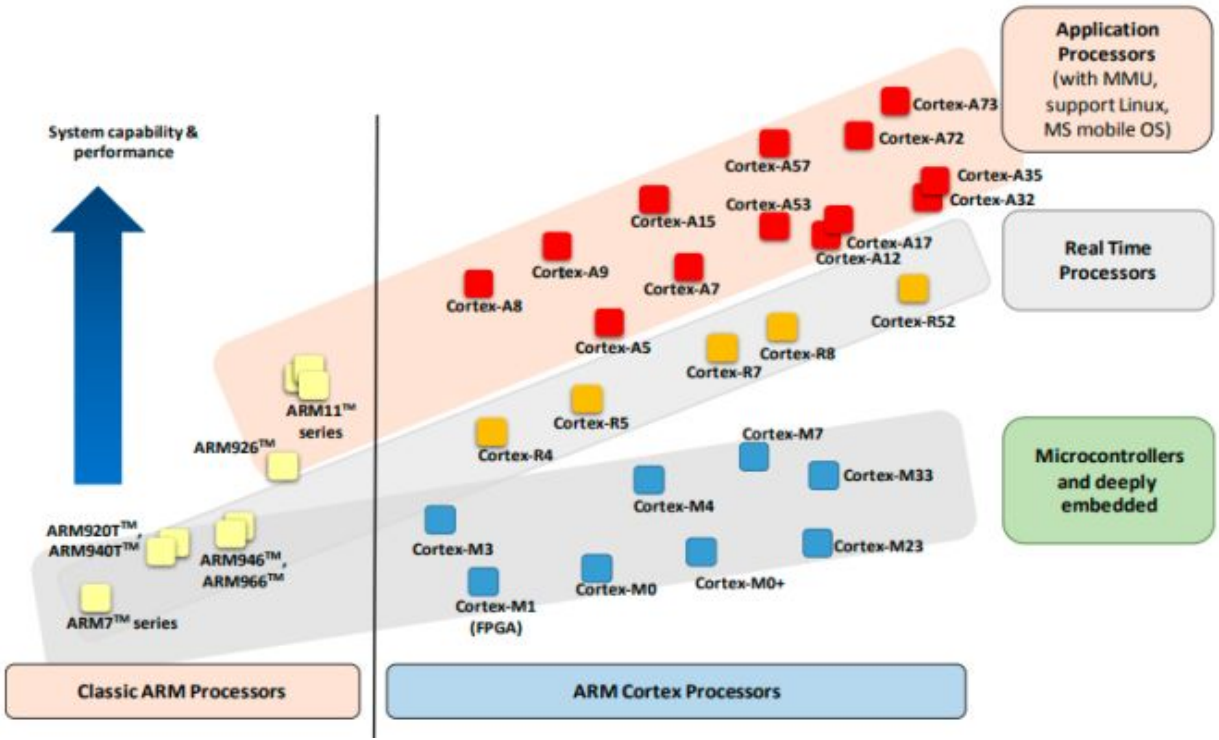
- Otros µCs
  - Basados en Cortex de ARM
    - STM32 (ST)
    - AVR32 (ATMEL)
    - MSP432 (TI)
    - LPC1700 (NXP)
  - Que no están basados en Cortex
    - ESP32 (Espressif)
    - PIC32 (Microchip)
    - DSPIC (Microchip)
    - C2000 (TI)
    - MSP430 (TI)





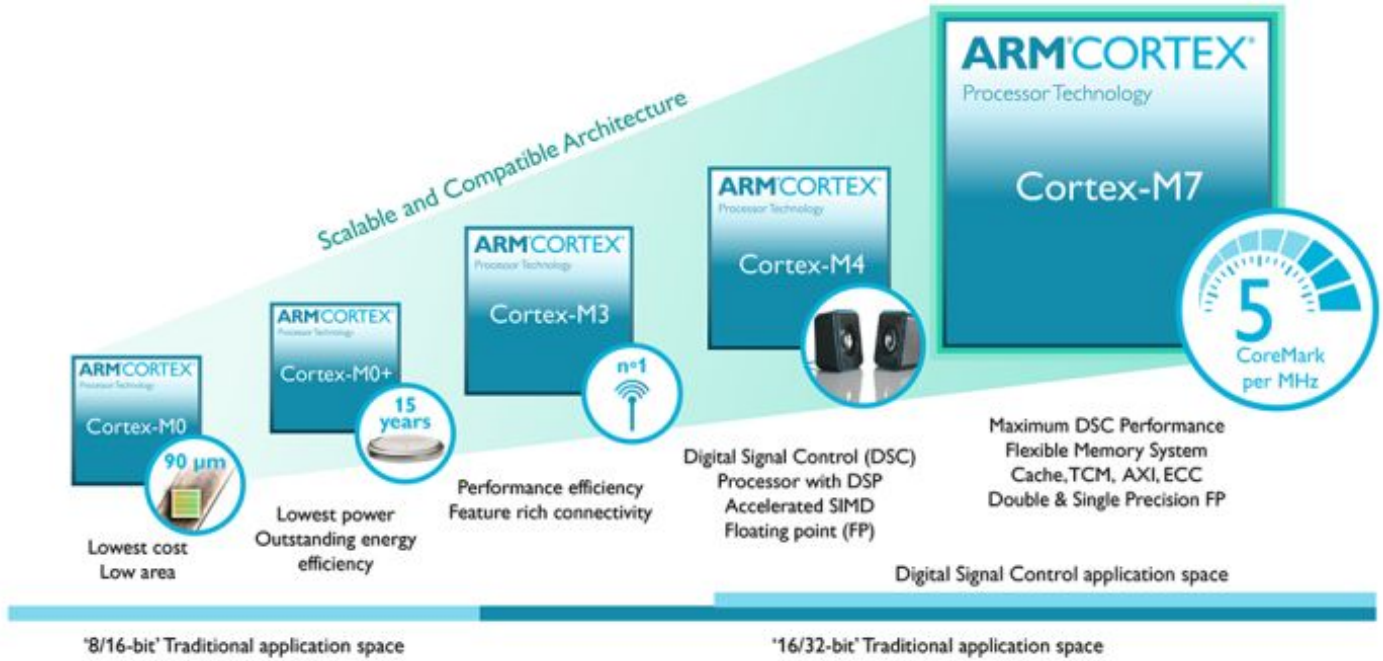
“Categoría” del Cortex-M3

- Al hablar de Cortex-M3 ya podemos posicionarnos en el rango de dispositivos



http://www.jkelec.co.kr/img/lecture/arm_arch/arm_arch_3.html

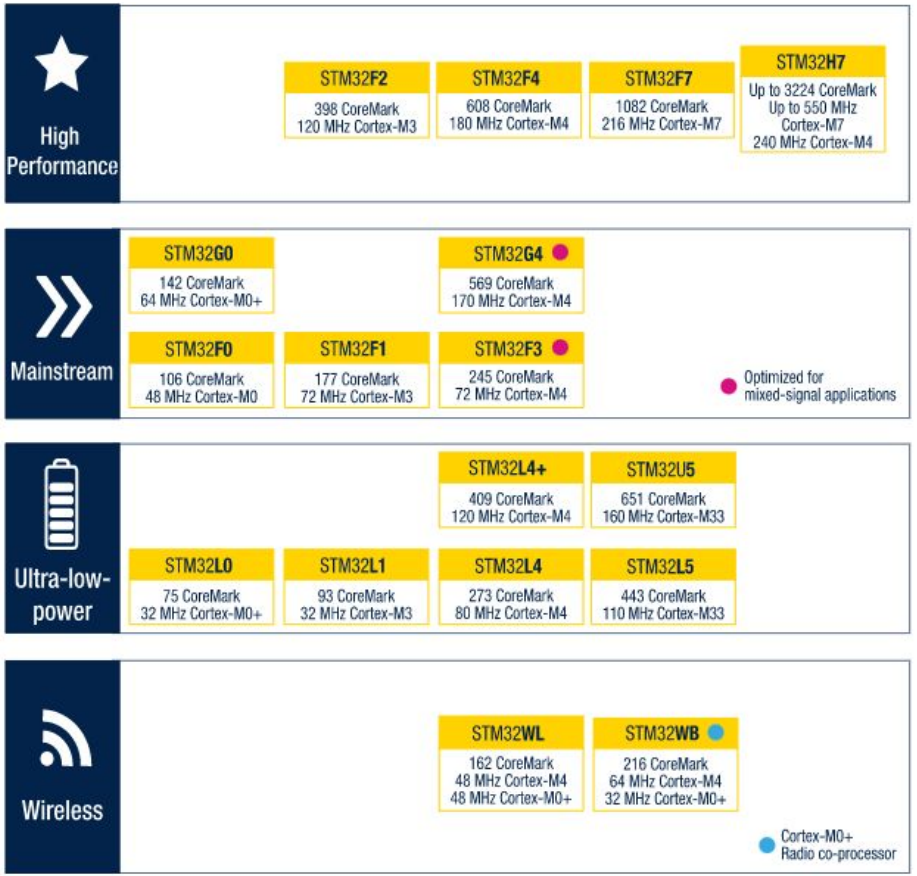
“Categoría” dentro de la familia Cortex



http://www.jkelec.co.kr/img/lecture/arm_arch/arm_arch_3.html

Métricas sobre la arquitectura

Asociado al microcontrolador



Comentario sobre benchmarks

- Histórico y vigente por la disponibilidad de evaluaciones: Dhrystone
- Un test como el D. mide “cuánto puede hacer” un sistema vs las mega-instrucciones por segundo (MIPS)
- La VAX 11/780 ejecutaba 1757 ciclos del test de Dhrystone por segundo
- Si un procesador ejecuta **N***1757 veces el test en 1s es una máquina de **N** DMIPS, y si lo hace con un clock de **M** MHz, es de **N/M** DMIPS/MHz

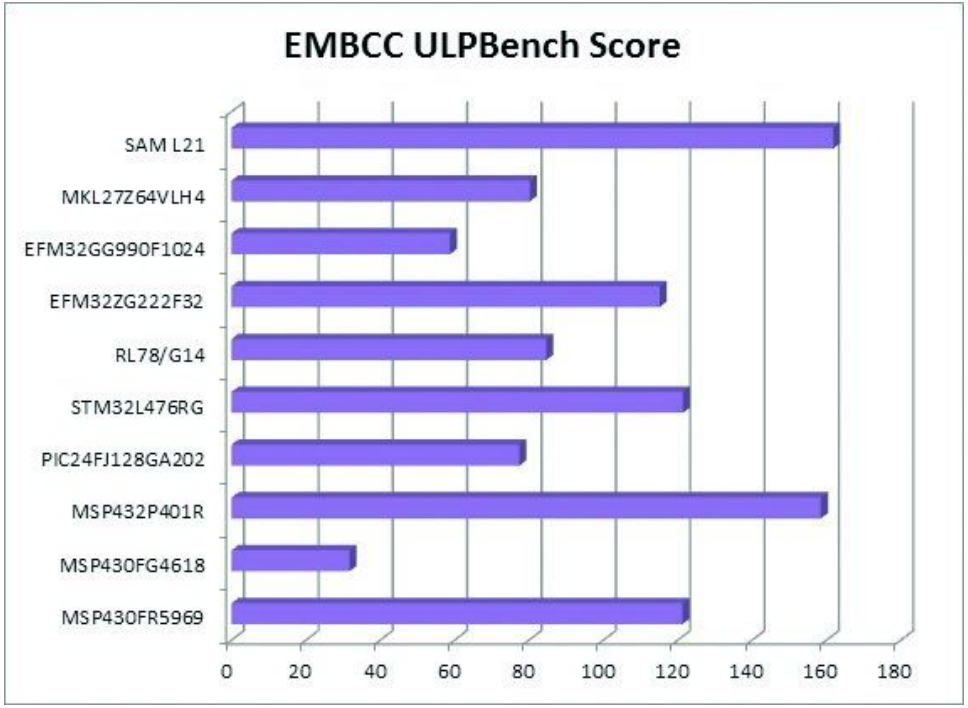
Benchmarks actuales

- Embedded Microprocessor Benchmark Consortium https://www.eembc.org/
  - *“EEMBC develops industry-standard benchmarks for the hardware and software used in autonomous driving, the Internet of Things, machine learning, and many other applications”*
- Por ejemplo:
  - **CoreMark** evalúa el CPU e incluye recorrer listas, operaciones matemáticas con matrices y procesamiento con máquinas de estado

Feature	Cortex-M0	Cortex-M0+	Cortex-M1	Cortex-M23	Cortex-M3	Cortex-M4	Cortex-M33	Cortex-M35P	Cortex-M55	Cortex-M7
Instruction Set Architecture	Armv6-M	Armv6-M	Armv6-M	Armv8-M Baseline	Armv7-M	Armv7-M	Armv8-M Mainline	Armv8-M Mainline	Armv8.1-M Mainline	Armv7-M
DMIPS/MHz*	0.87	0.95	0.8	0.98	1.25	1.25	1.5	1.5	1.6	2.14
CoreMark®/ MHz*	2.33	2.46	1.85	2.64	3.34	3.42	4.02	4.02	4.2	5.01

Benchmarks actuales

- **ULPBench**
  - Serie de tests que evalúan el consumo de energía apuntando a aplicaciones de consumo ultra-bajo.
  - Se consider ULP si dura más de 4 años ejecutando una función activa que incluye una serie determinada de pruebas 1 vez por segundo con una batería de 225 mAh.
  - Su primer test es Core Profile que mide la eficiencia del CPU, reloj de tiempo real, calendario, y modos de ahorro de energía.

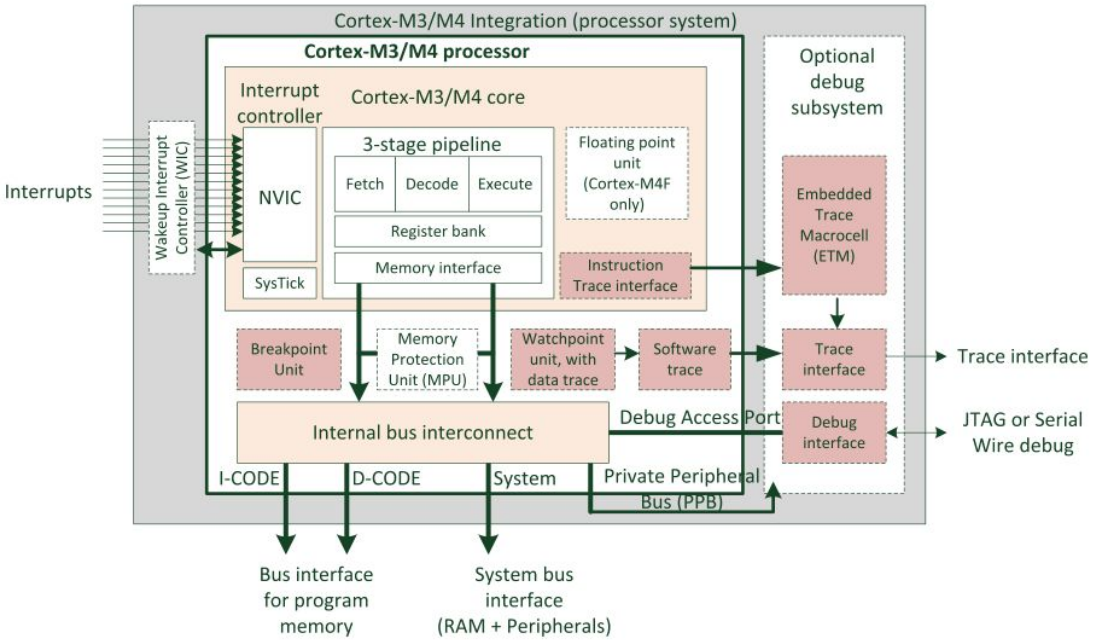


https://www.eembc.org/techlit/articles/elektroniknet_article_2015_07_21/index.html

Arquitectura del procesador Cortex M3

Arquitectura del Cortex-M3

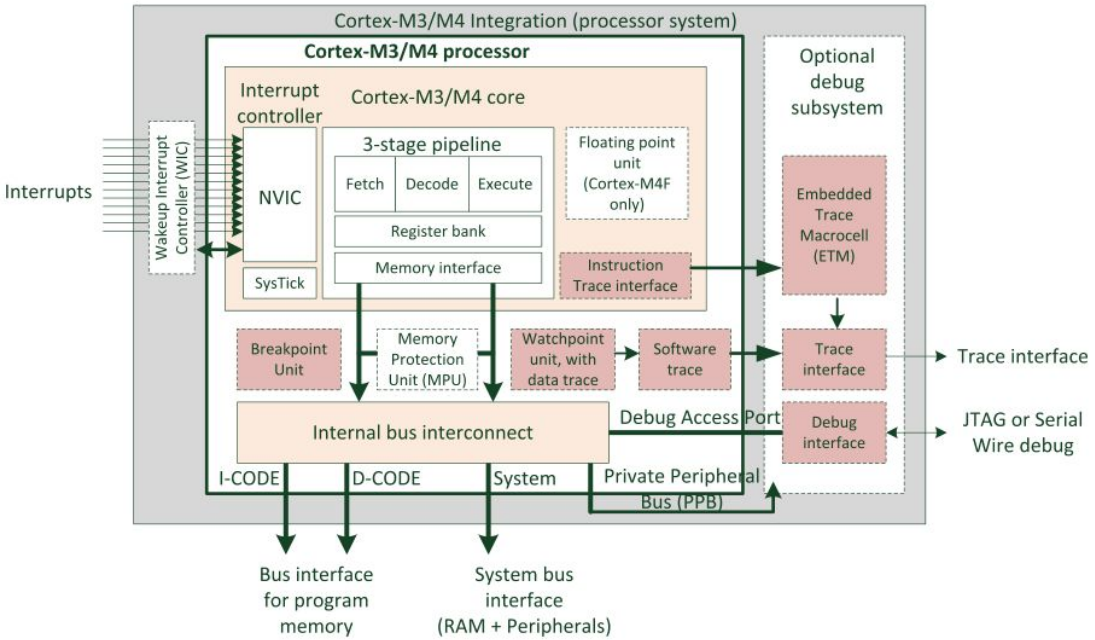
- Procesador de 32 bits: Es decir que sus registros internos y sus buses son de 32 bits y por lo tanto puede direccionar una memoria de 4 GB
- Arquitectura Harvard modificada
- Pipeline de 3 etapas



Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

Arquitectura del Cortex-M3

- Set de Instrucciones RISC
- En particular la arquitectura de instrucciones es **ARMv7-M**
- Es un set “reducido” de instrucciones que privilegia lograr un tamaño de código reducido y operación determinística por sobre la performance
- Utiliza instrucciones Thumb 2 en su mayoría de 16 bits y que pueden ejecutarse casi todas en el equivalente de 1 ciclo de reloj

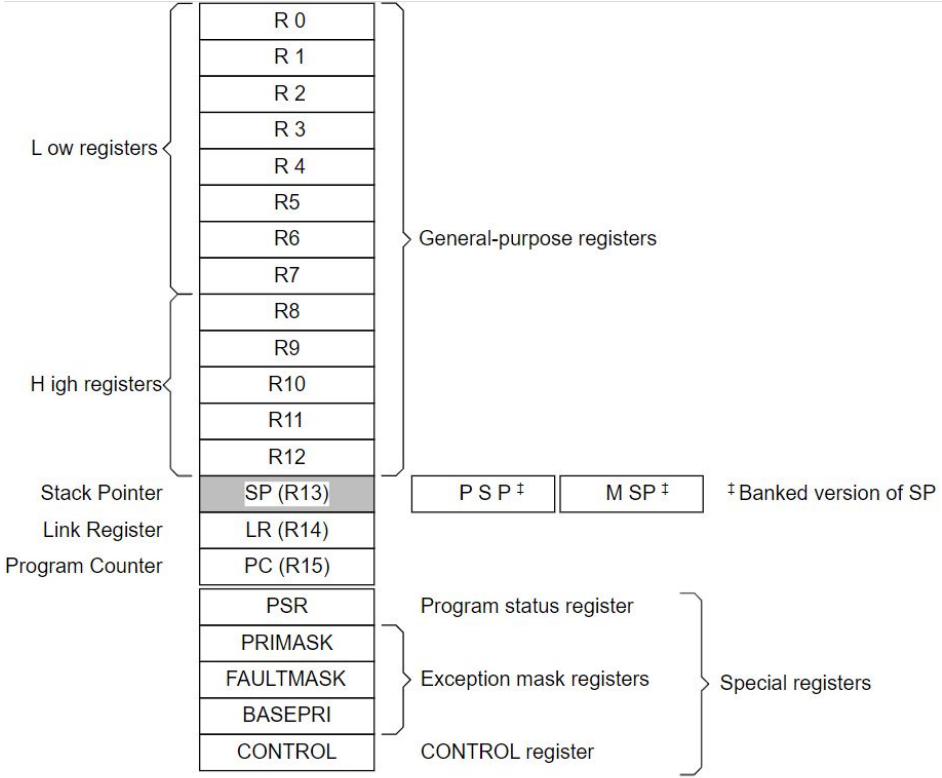


Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

Registros

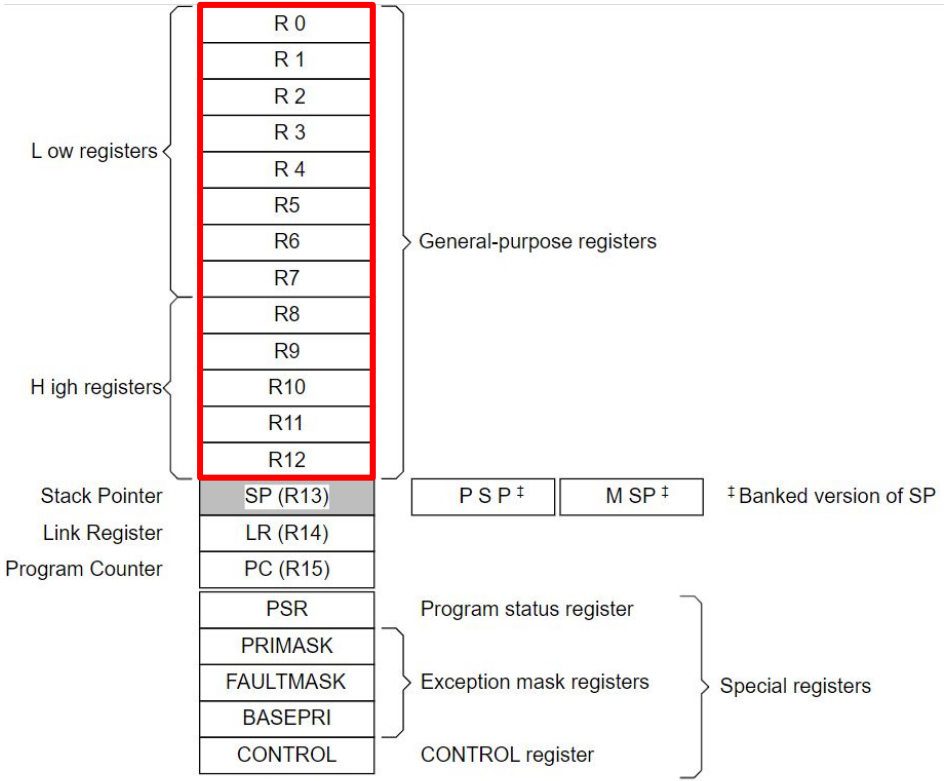
Registros del procesador

- Son el espacio “donde todo sucede”
- 16 registros + regs. especiales
- Debido al set RISC se usa arq. Load-Store y todas las instrucciones se desarrollan dentro de estos registros (salvo load y store)



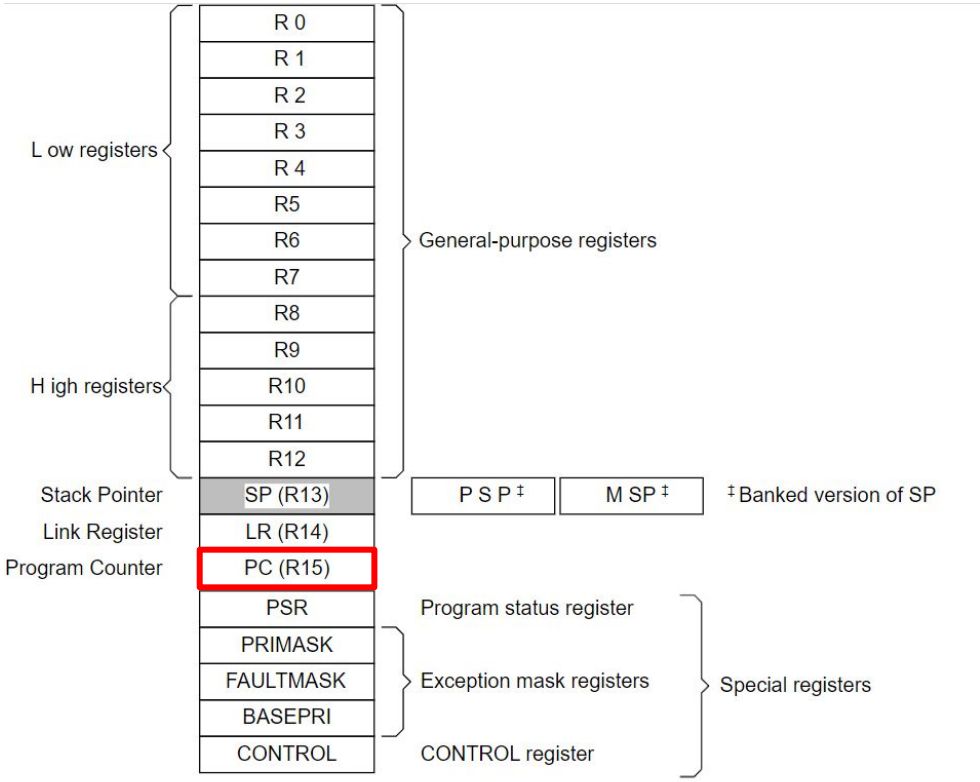
Registros de propósito general

- Se utilizan en todas las operaciones
- Debido al uso de instrucciones de 16 bits hay “poco espacio” sobrante en cada instrucción y algunas sólo pueden direccionar con 3 bits, y por eso la distinción de “low” y “high” registers
- A algunos registros se le asigna funcionalidad especial no por hardware sino por el estándar del compilador
  - R0 a R4 para pasaje de argumentos a subrutinas
  - R7 para puntero “modificable” al stack



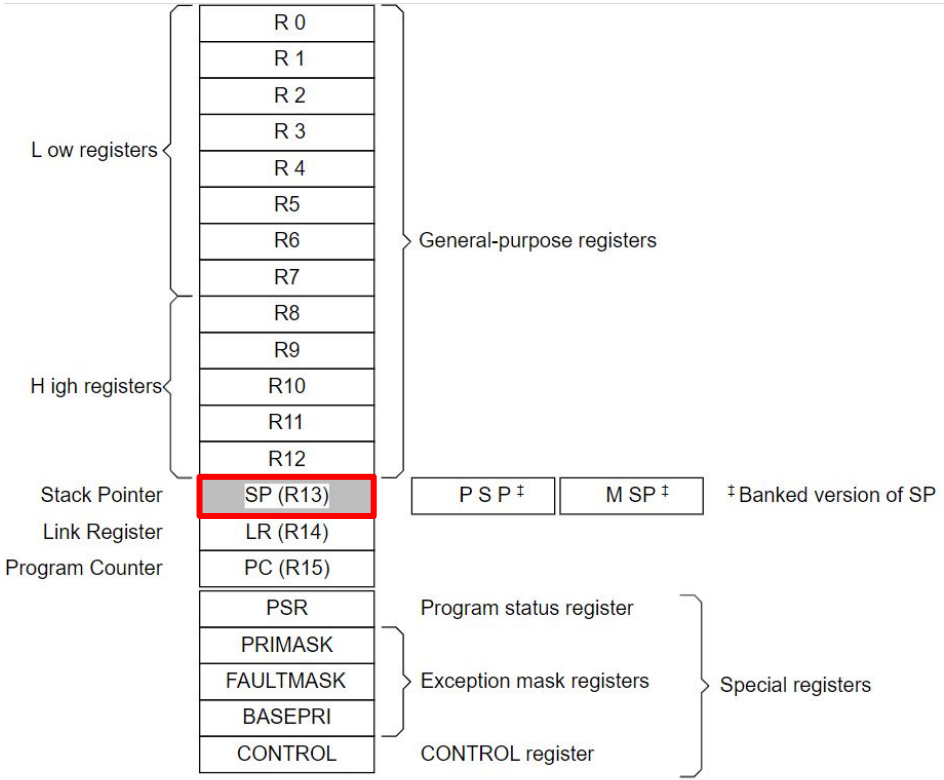
Program counter

- Se inicializa en la dir. 0x00000004 donde está la “tabla de vectores” <- el comienzo de todo! (en la 0 está la dirección del stack)
- Se puede alinear a 32 o 16 bits. Por lo tanto el LSB sería 0, pero debe escribirse siempre a 1 para indicar “thumb state”



Stack pointer

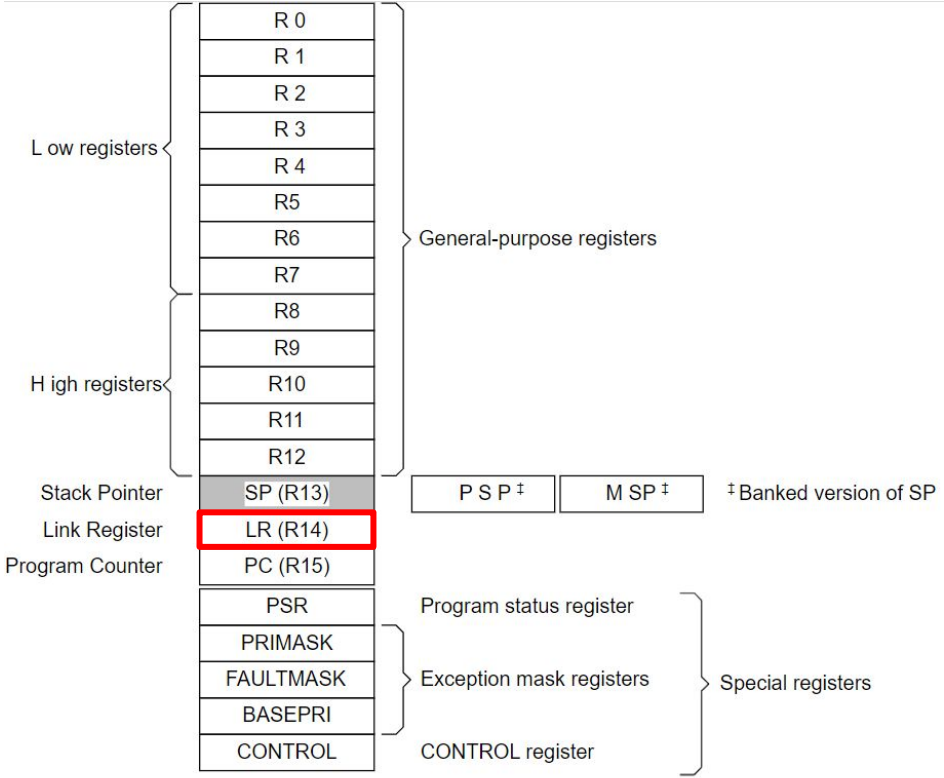
- El stack es la “memoria de trabajo” del programa y el lugar donde guardar el contenido de los registros al cambiar de contexto (saltar a otra función).
- El SP mantiene la dirección del segmento de stack que se está usando y permite rápidamente a través de instrucciones push y pop que lo modifican automáticamente cambiar de contexto.
- Es “full descending”: Al iniciar la ejecución se apunta **a la posición más alta** de la memoria RAM y se decrementa al hacer los push.
- Tiene 2 “bancos” o registros físicos. Según el estado de ejecución muestra el “Main SP” o el “Process SP” (el PSP se usa con OSs)
- Los últimos 2 bits son siempre 0 (push y pop son siempre de 32 bits)



ST PM0056 STM32F10xxx/20xxx/21xxx/L1xxxx
Cortex®-M3 programming manual

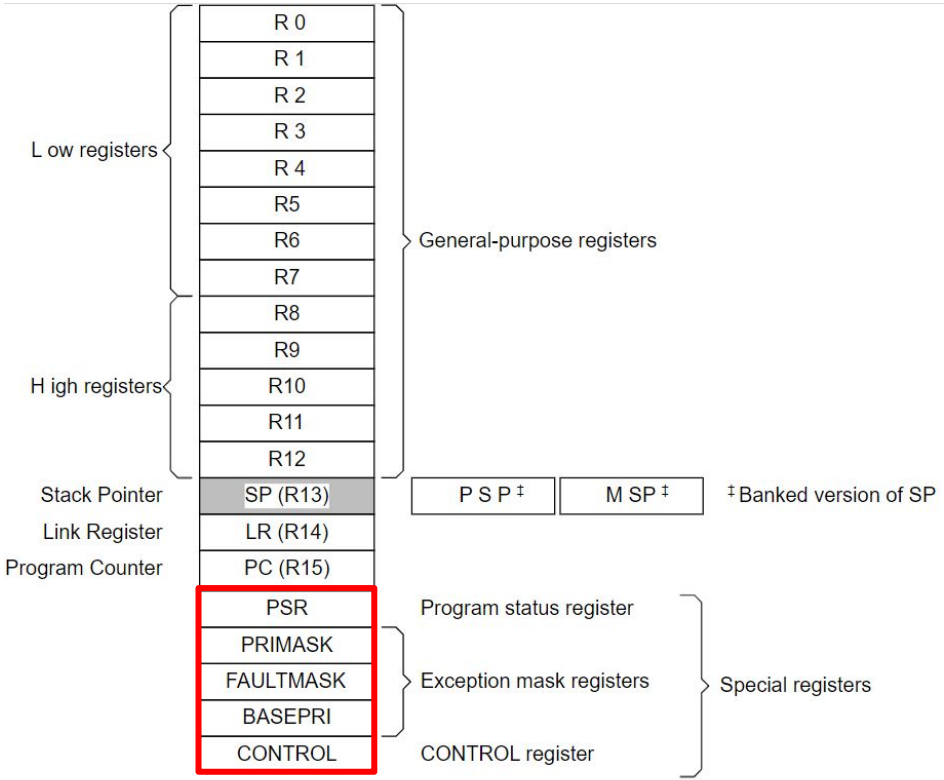
Link register

- Guarda el salto de retorno al cambiar de contexto con la instrucción “branch with link”
- Cuando se atiende una excepción almacena un código especial



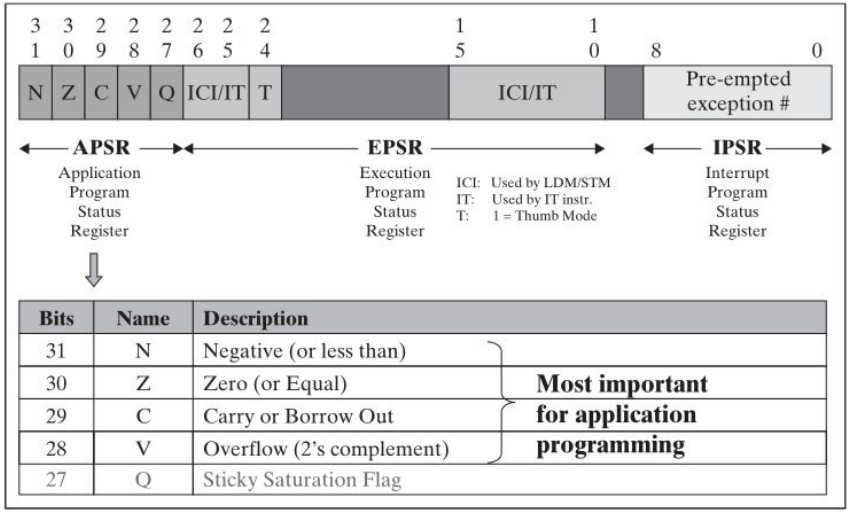
Registros Especiales

- Sirven para conocer y controlar el estado de ejecución del código
- Se accede con instrucciones
  - MRS <reg>, <special_reg>;
    - Read special register into register
  - MSR <special_reg>, <reg>;
    - write to special register
  - Estas instrucciones pueden “bloquearse” en un modo privilegiado y de esa manera se implementa un rudimentario control de acceso



Program Status Register

- Application PSR:
  - Contiene los resultados de las operaciones y es utilizado constantemente en cualquier aplicación
- Execution PSR:
  - Le sirve al procesador para asentar el estado durante instrucciones que conllevan varios pasos secuenciales (load/store múltiple e if/then)
- Interrupt PSR
  - Se relaciona con el manejo de excepciones



Bits	Name	Description	
31	N	Negative (or less than)	Most important for application programming
30	Z	Zero (or Equal)	
29	C	Carry or Borrow Out	
28	V	Overflow (2's complement)	
27	Q	Sticky Saturation Flag	

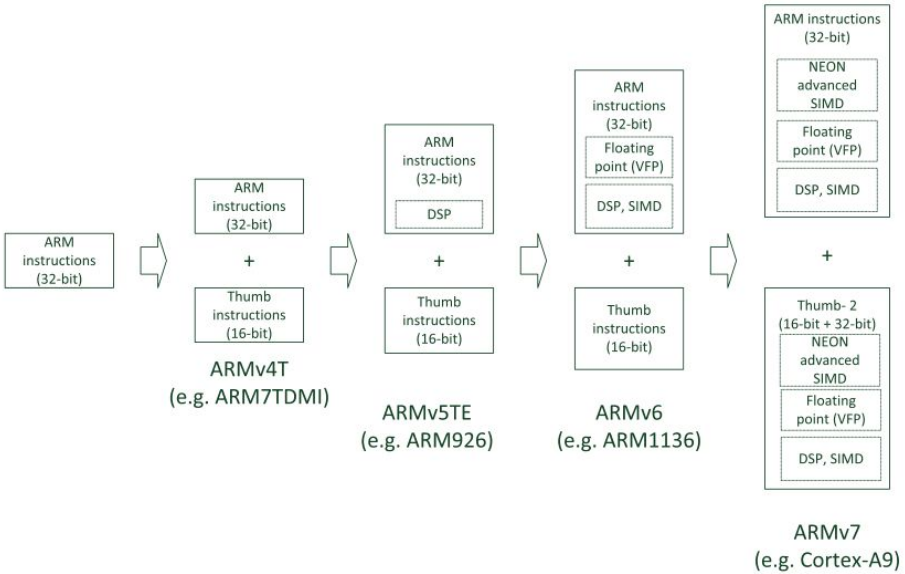
FIGURE 5-12 Bit fields within the Program Status Register.

D.W. Lewis - Fundamentals of Embedded Software with the Arm Cortex M3 (2012)

Set de Instrucciones

Set de Instrucciones thumb-2

- Instruction Set Architecture (ISA)
  ARMv7-M
  - Documentación: ARMv7-M Reference Manual
  - https://developer.arm.com/documentation/ddi0403
- Son de tipo Thumb 2, es decir que mezclan 16 y 32 bits.
- La arq. -M que privilegia tamaño y determinismo, sólo opera en modo Thumb y prevalecen las instrucciones de 16 bits. Por compatibilidad (o portabilidad) se mantiene el bit de modo thumb.
- Al ingresar al procesador, todas las instrucciones se expanden a 32 bits



Manejo de datos

- ISA ARMv7-M soporta los siguientes tipos en memoria:
  - Byte 8 bits
  - Halfword “media palabra” 16 bits
  - Word “palabra ”32 bits
- Existen instrucciones para los siguientes tipos de datos en los registros:
  - Punteros de 32 bits
  - Enteros con o sin signo de 32 bits
  - Enteros sin signo de 16 o 8 bits almacenados rellenando con 0s
  - Enteros con signo de 16 o 8 bits almacenados con extensión de signo
  - Enteros de 64 bits con o sin signo almacenados en 2 registros

Direccionamiento

- El direccionamiento de memoria en las instrucciones puede darse a través de varios métodos, entre ellos:
  - El contenido de registros (incluyendo a veces un shift opcional)
  - Valores constantes
  - Valores relativos al PC (un método que el compilador usa comúnmente)
  - Valores relativos al SP
- El formato es Little Endian y aunque puede ser configurable, es el modo más común y así lo encontraremos en la documentación de ST

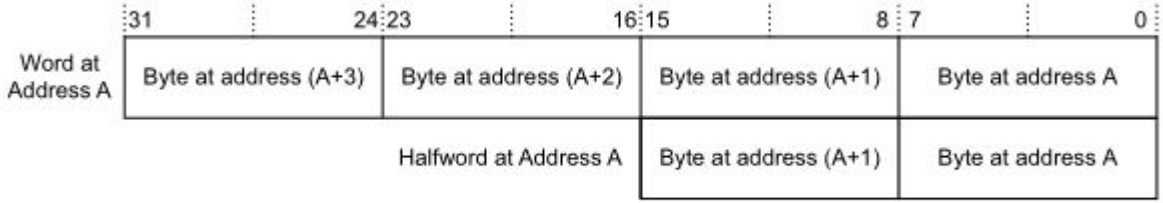


Figure A3-1 Little-endian byte format

Operaciones y formato de las instrucciones

- El CPU puede realizar varias operaciones de bajo nivel en paralelo
- Las instrucciones tienen operandos Rm, Rn y destino Rd, y algunas pueden contener valores constantes de hasta 8 bits
- Existe un “extensor de signo” para expandir los operandos a 32 bits
- También un barrel shifter que permite hacer el desplazamiento que se quiera en 1 ciclo (que además puede ayudar a la ALU a realizar multiplicaciones). Algunas instrucciones pueden incluir desplazamientos.
- Un “memory address calculator” ayuda a calcular los direccionamientos y un sistema de incremento permite ir avanzando sin recalcular
- Todas las instrucciones incluyen ejecución condicional según los resultados del APSR

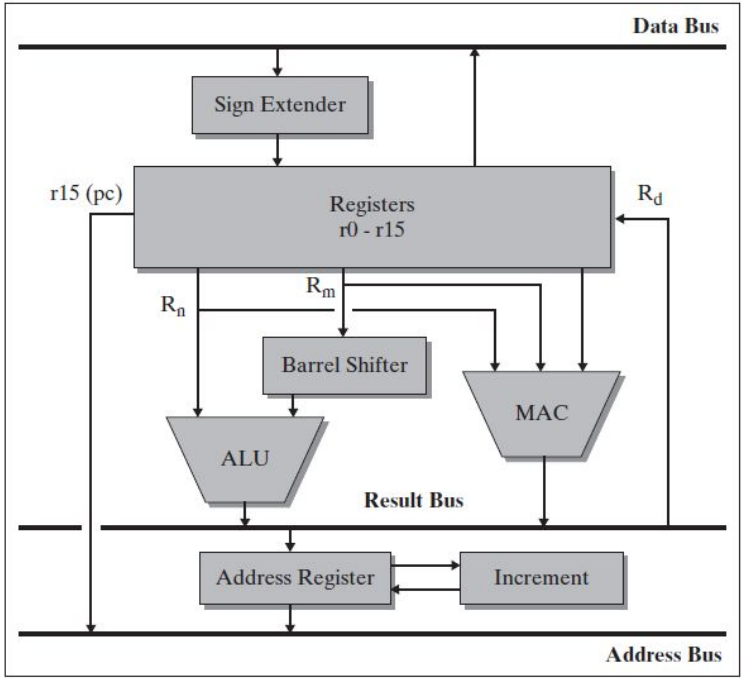


FIGURE 5-11 Internal organization of the ARM Cortex-M3 v7M CPU.

D.W. Lewis - Fundamentals of Embedded Software with the ARM Cortex M3 - Pearson (2012)

Impacto en el alto nivel

Barrel Shifter Integrado

- El desplazador (barrel shifter) permite realizar shifts y rotaciones en paralelo con otras instrucciones sin penalización en ciclos.
  - Optimiza operaciones como `x << n`, `x >> n`, `x & (1 << n)`, evitando ciclos adicionales.
  - Hace que cálculos como `x * 8` (equivalente a `x << 3`) sean eficientes.
  - Permite que algunas operaciones bit a bit se realicen en una sola instrucción ensamblador.

Expansor de Signo

- Un bloque de hardware que expande valores de 8 y 16 bits a 32 bits sin necesidad de instrucciones adicionales.
  - Hace eficientes las conversiones de tipos (`char -> int`, `short -> int`).
  - Evita el uso de instrucciones extra para expandir el signo.

```
int8_t a = -5;
int32_t b = a;  // Expansión de signo eficiente
```

Impacto en el alto nivel

Multiplicación por Hardware

- El Cortex-M3 tiene multiplicación de 32 bits por hardware en solo 1 ciclo.

```
int32_t resultado = a * b;  // Se ejecuta en un solo ciclo (más load y store)
```

Instrucciones de División por Hardware

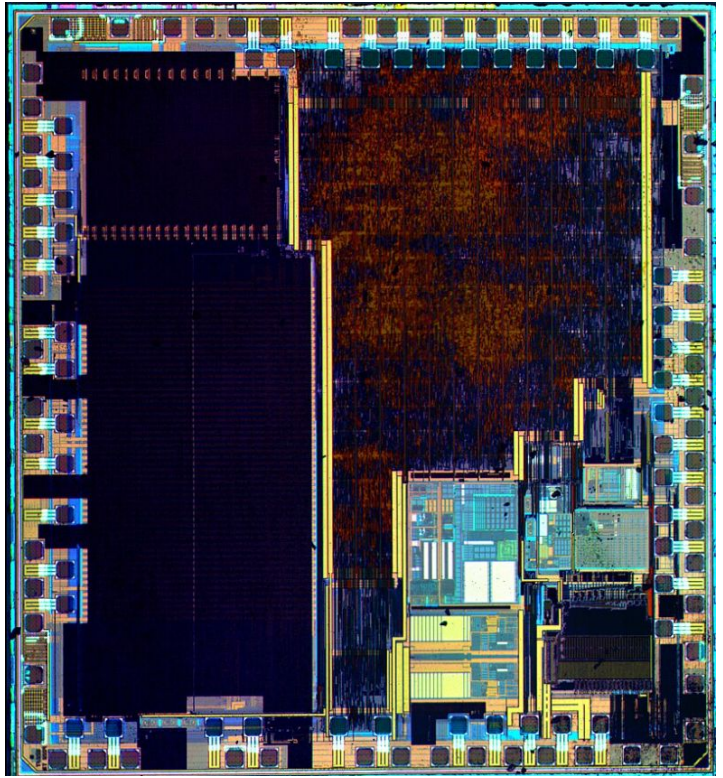
- El Cortex-M3 tiene división entera por hardware, aunque toma más ciclos (2 a 12).
- Permite ejecutar divisiones en hardware sin librerías externas.

```
int32_t resultado = a / b;
```

Instrucciones de Exclusión Mutua (LDREX/STREX)

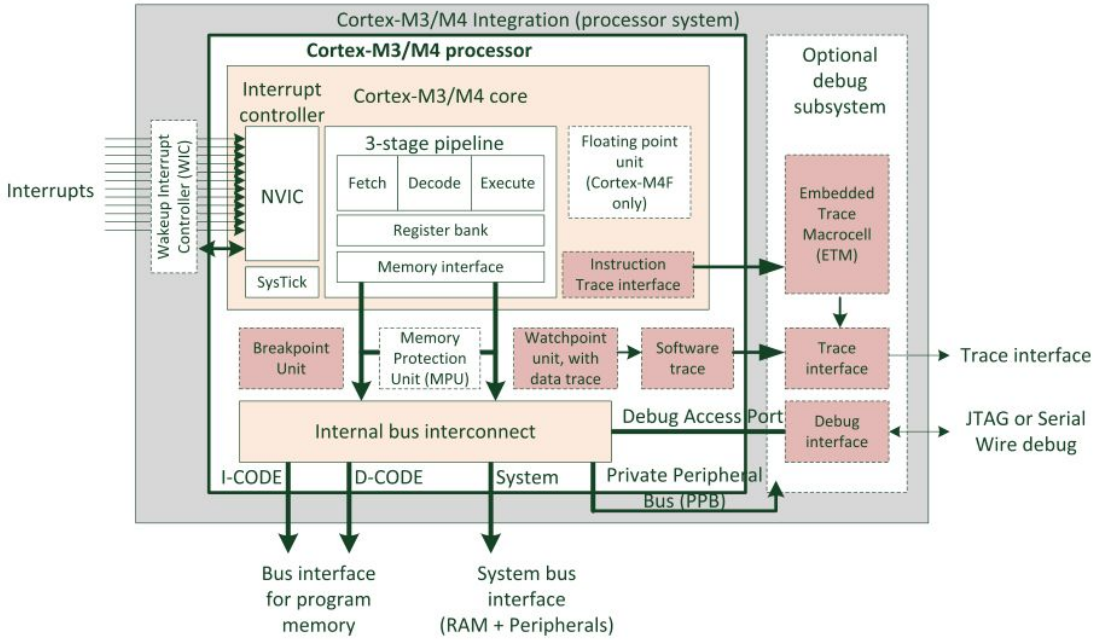
- Instrucciones `LDREX` y `STREX` permiten operaciones atómicas en memoria.
  - Facilita la implementación eficiente de semáforos y locks

Memorias y modelo de memoria



Acceso a memoria y periféricos

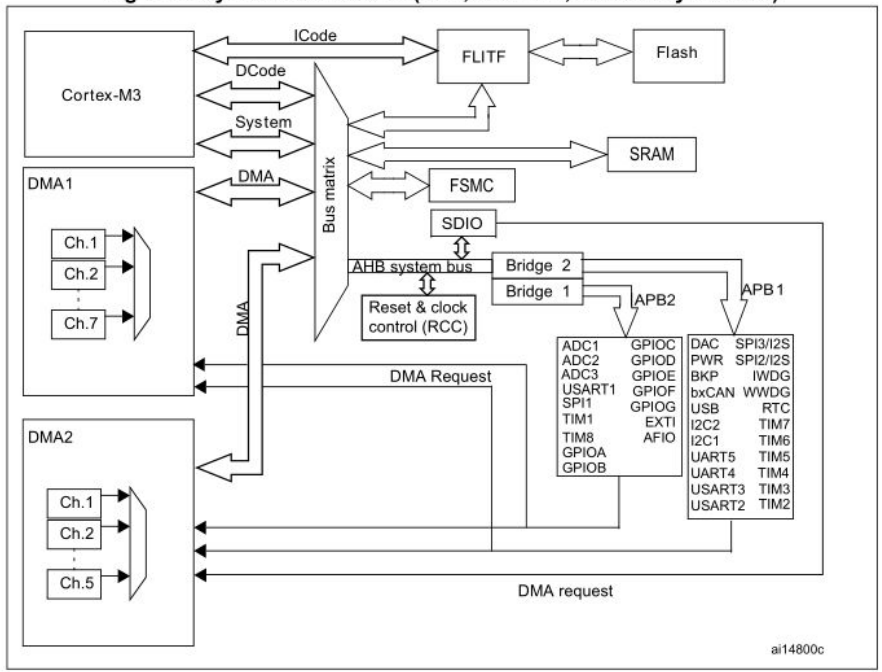
- Arquitectura Harvard modificada
- Buses que tiene disponible el procesador:
  - I-CODE
  - D-CODE
  - System
  - Debug Access Port
  - Private Peripheral



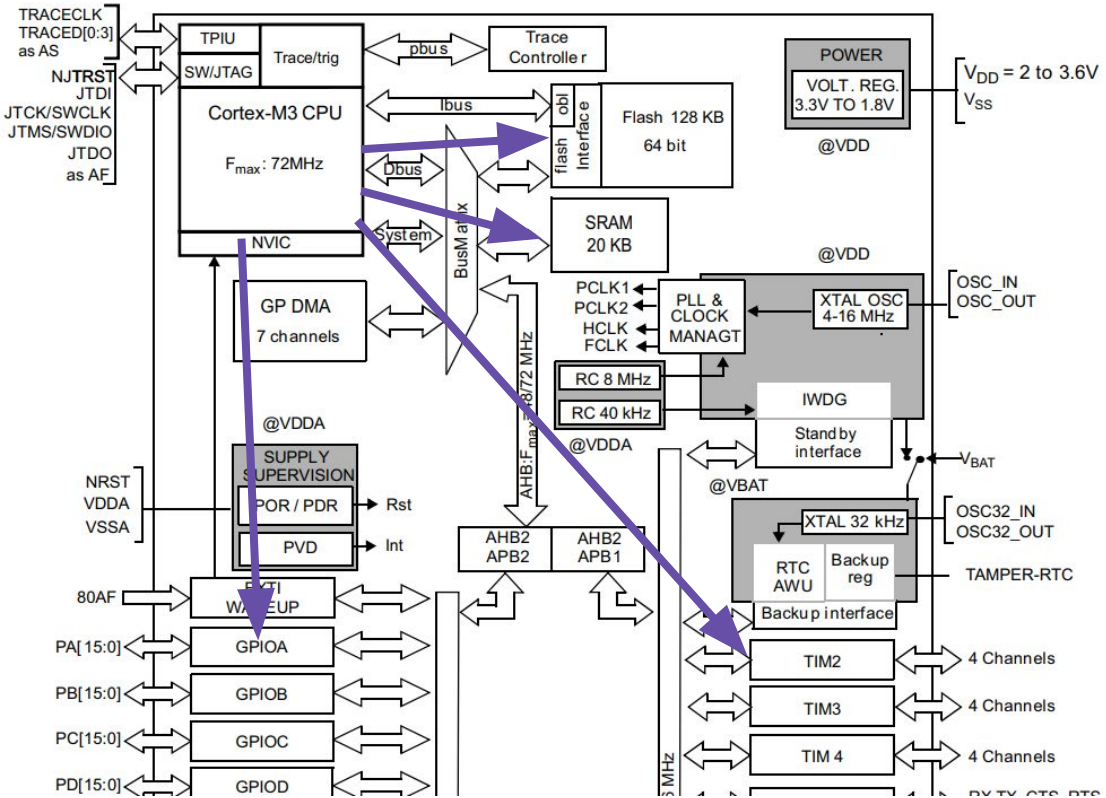
Acceso a memoria y periféricos

- Acceso a memoria de programa (Flash)
  - Por ICode para instrucciones, DCode para datos
  - FLIFT es un caché de prefetch de 2x64 bits, permite escribir y borrar la Flash de a 16 bytes (reprogramar)
  - Acceso a través del DAP a la memoria flash para la programación
- Acceso a la memoria de datos (RAM) y periféricos
  - Bus System tipo AHB
  - **Periféricos en buses APB1, APB2**
- Un arbitrador da prioridad a los distintos maestros
  - Maestros: Cortex, DMA
  - Esclavos: Memorias, Periféricos

Figure 1. System architecture (low-, medium-, XL-density devices)



Vista del sistema desde el procesador



Direcciones engañapichanga

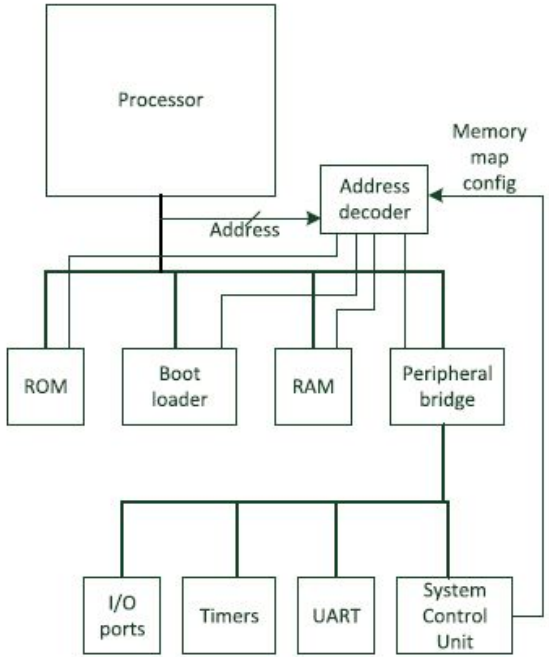


FIGURE 6.19

A simplified memory system with configurable memory map

Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

Modelo de memoria - Cortex M

- Los procesadores Cortex-M ven **todo** como una posición de memoria **en un mapa lineal de 4 GB** (el máximo direccionable con 32 bits)
- No son direcciones “reales”, pero pueden verse como si lo fueran para usar el dispositivo
- Las direcciones de los segmentos son un estándar dentro de la familia Cortex-M para poder portar el código

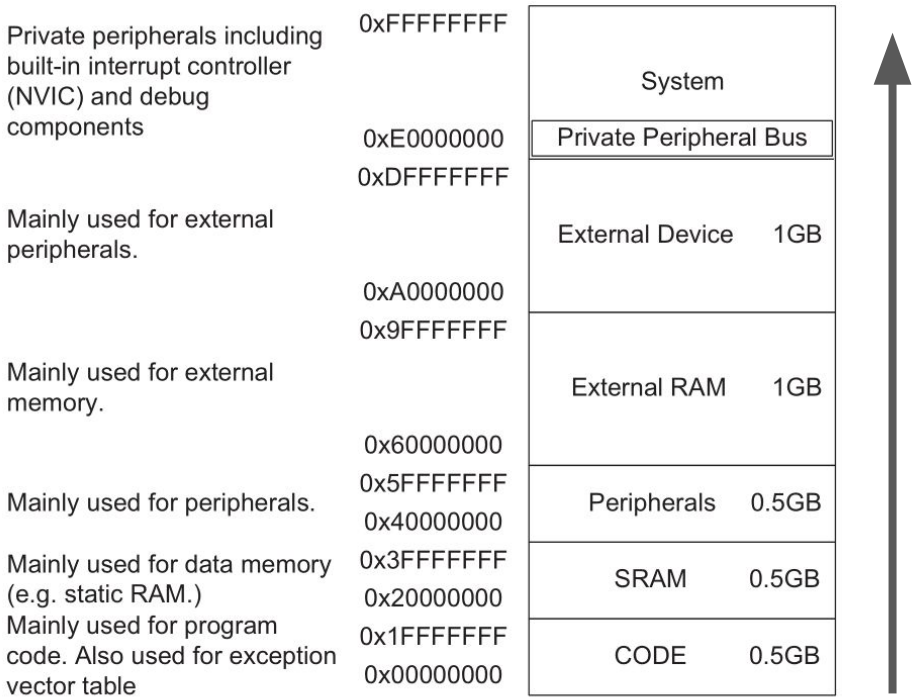
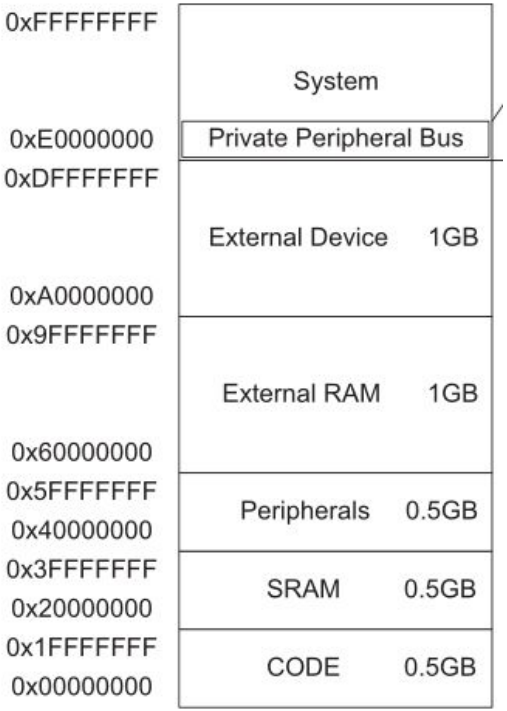


FIGURE 4.18

Memory map

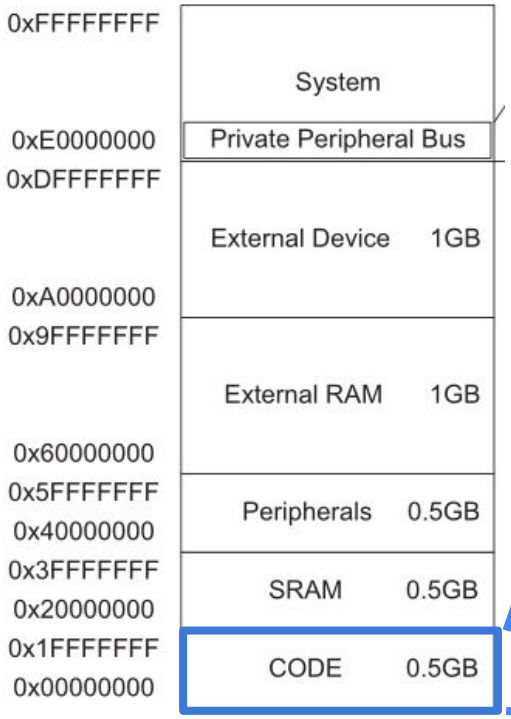
Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

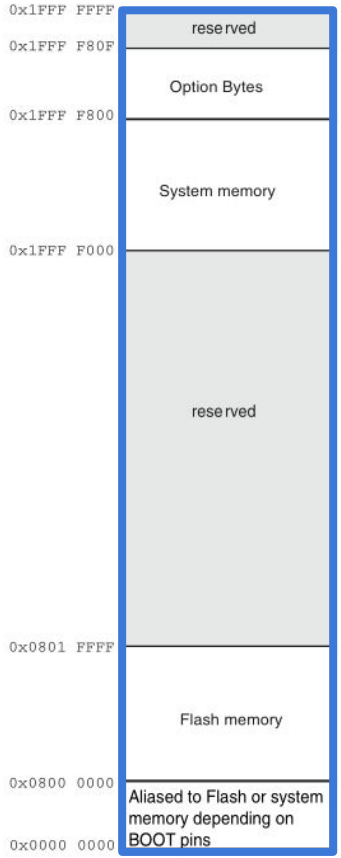
Modelo de memoria - **STM32**



- Analizaremos este mapa de memoria “genérico” de los Cortex M estudiando qué segmentos están implementados en el microcontrolador stm32f103c8
- Para eso, recorreremos los segmentos desde la primera a la última posición

Memoria de código





16 B en FLASH
(descriptos en PM0057)

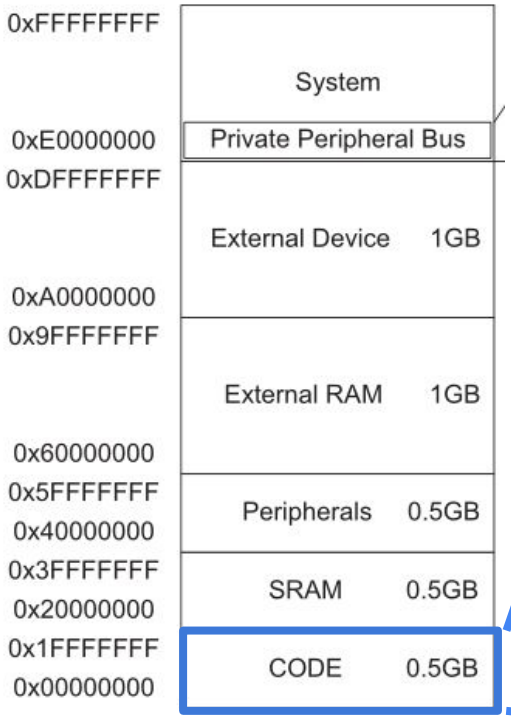
2 kB en ROM

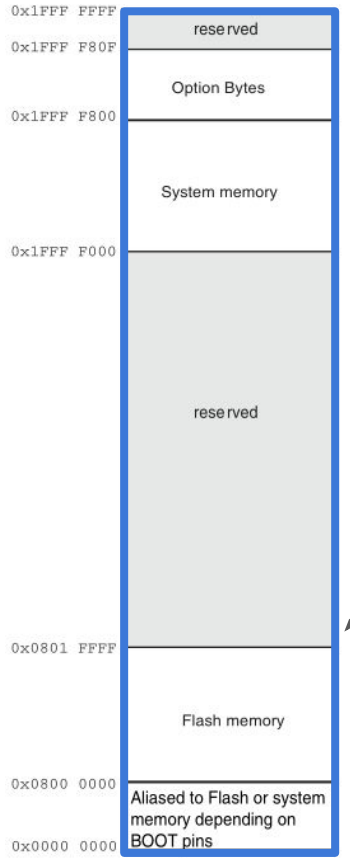
64 kB en FLASH

no existe!

Datasheet - STM32F103x8 STM32F103xB
Doc ID 13587 Rev 12, ST

Memoria de código





Valores programables para controlar comportamiento al encender el dispositivo, relativo a protección de memoria, watchdog, eventos de reset para modos de bajo consumo

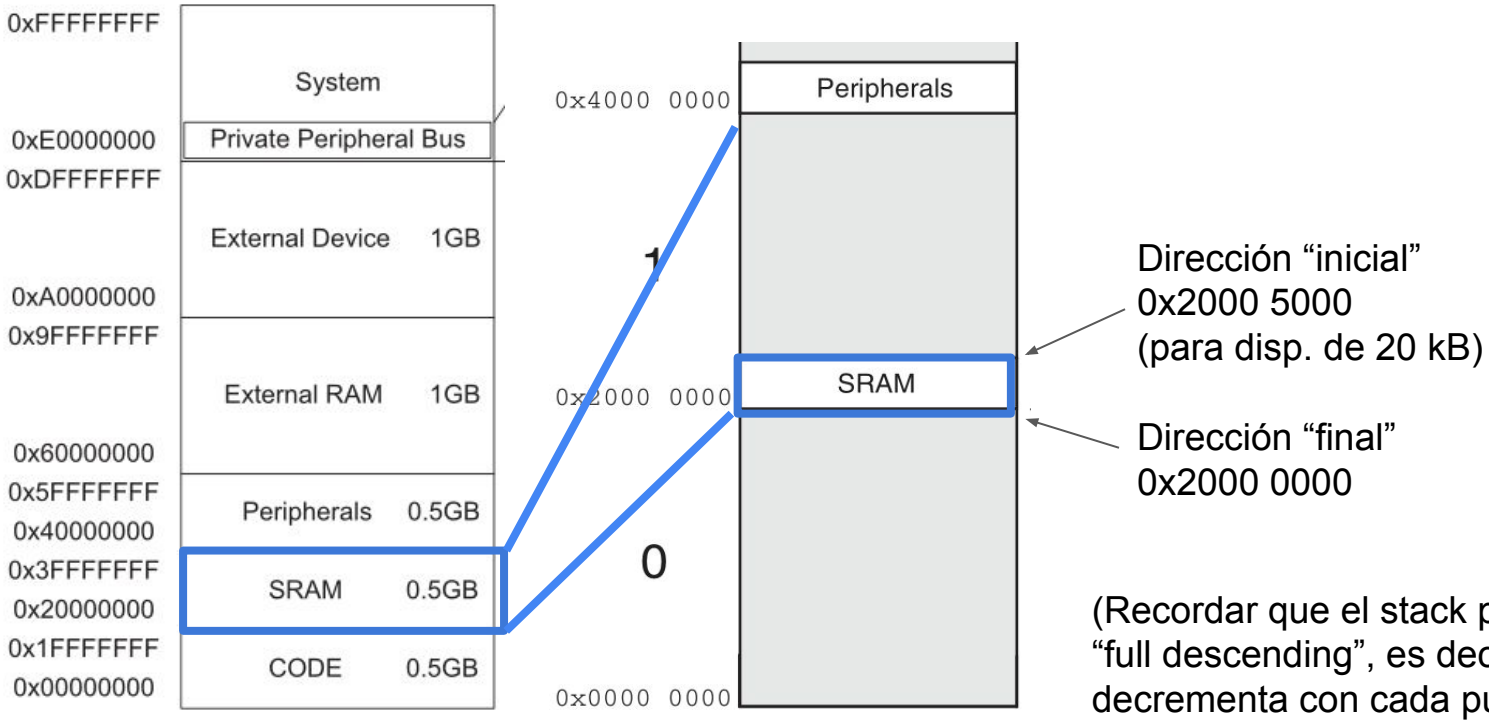
Bootloader pre-programado por ST en ROM con el código para reescribir la flash a través de algún puerto (e.g. UART)

Para este uC que tiene 64 kB de Flash = 2^16 kB se utiliza desde la posición 0 a la 0x100000 - 1 del mapa de memoria

Cuando se accede a la dirección 0x00000000 en realidad se redirige el acceso a la posición de la FLASH o del bloque System

Datasheet - STM32F103x8 STM32F103xB
Doc ID 13587 Rev 12, ST

SRAM



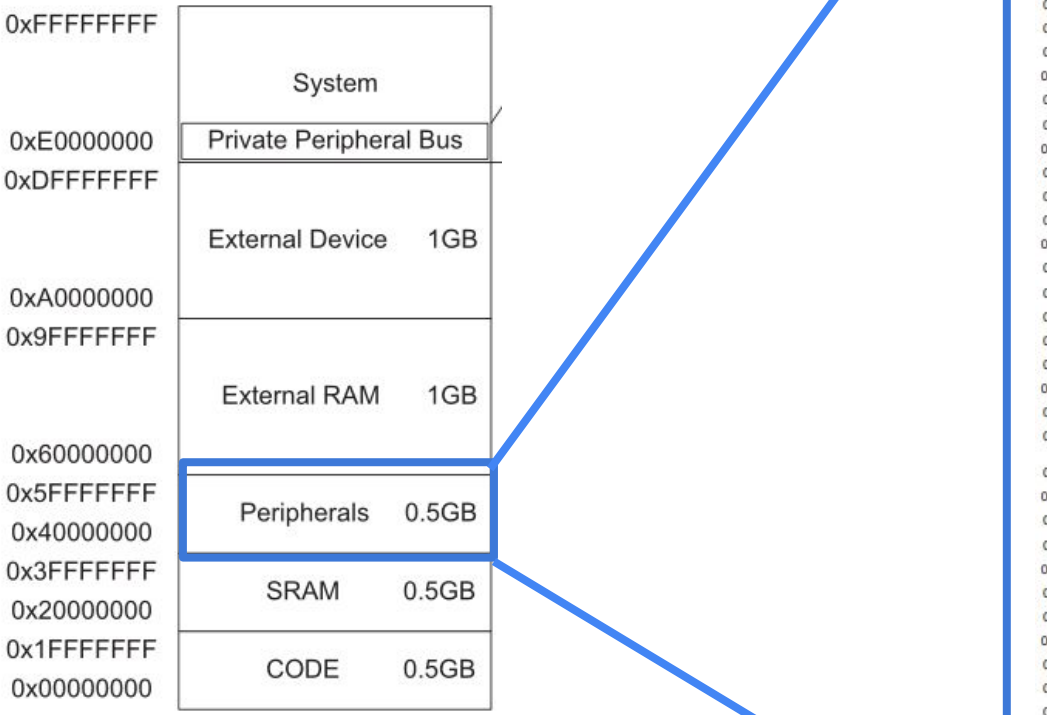
Dirección “inicial”
0x2000 5000
(para disp. de 20 kB)

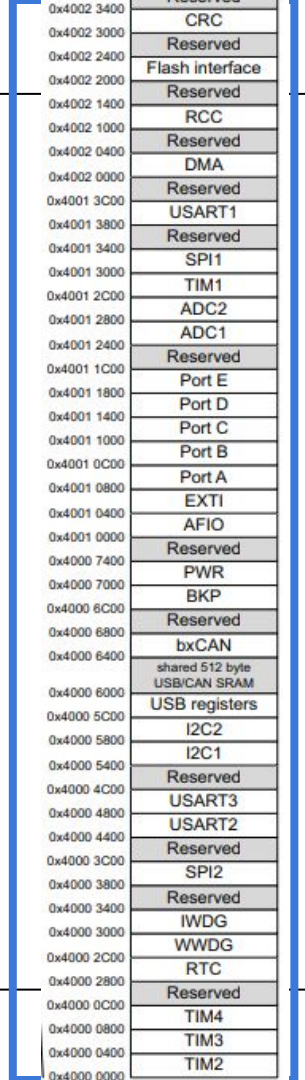
Dirección “final”
0x2000 0000

(Recordar que el stack pointer es “full descending”, es decir que se decrementa con cada push)

Datasheet - STM32F103x8 STM32F103xB
Doc ID 13587 Rev 12, ST

Periféricos

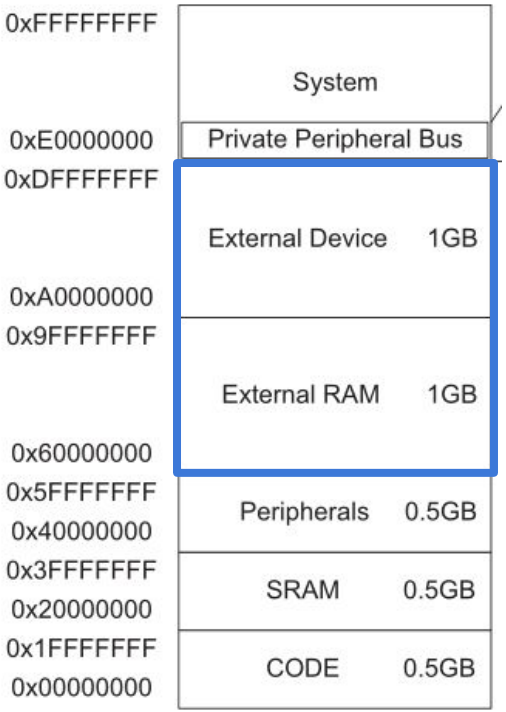




Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

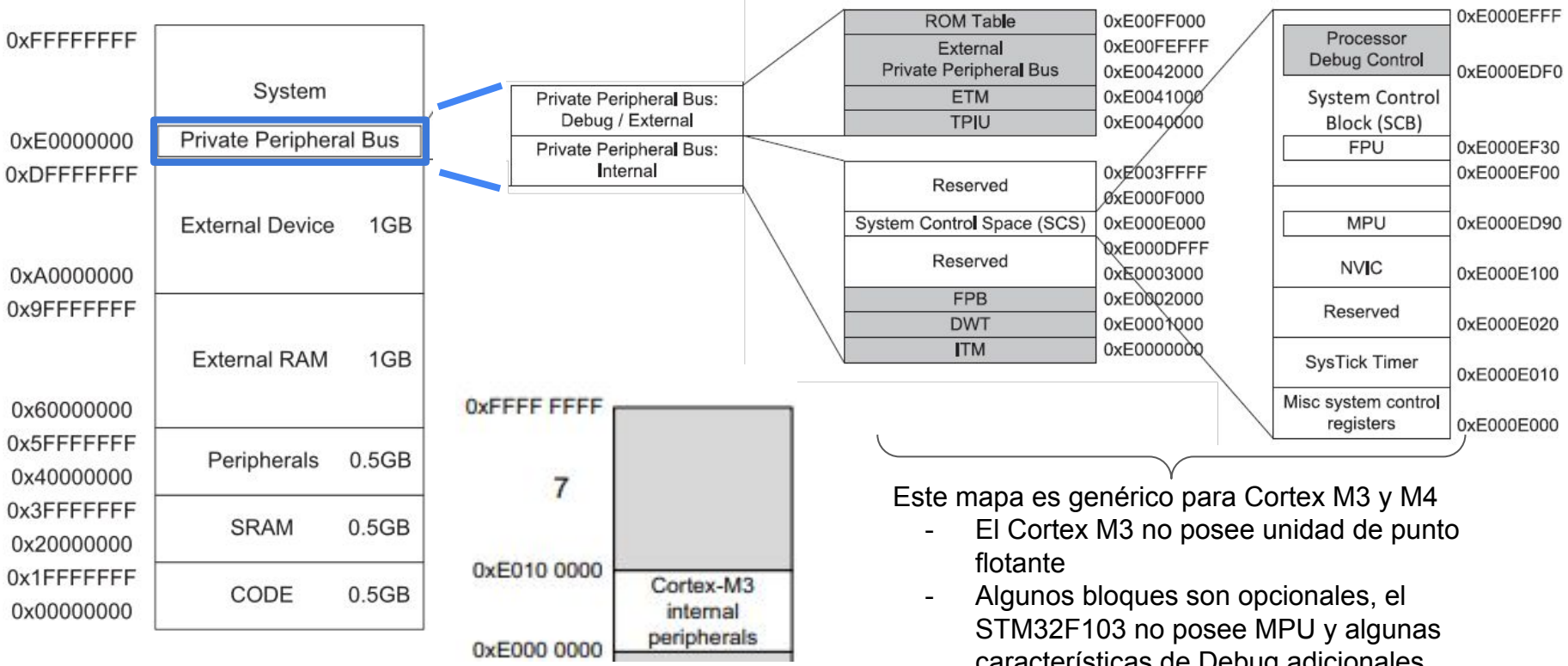
Datasheet - STM32F103x8 STM32F103xB Doc ID 13587 Rev 12, ST

Externos



Sirve para integrar memoria o periféricos externos en “igualdad de condiciones” con los internos. Requiere periféricos especializados (que el STM32f103 no tiene) y buses rápidos en general con un conteo de pines alto

Segmento privado del Cortex M3



Este mapa es genérico para Cortex M3 y M4

- El Cortex M3 no posee unidad de punto flotante
- Algunos bloques son opcionales, el STM32F103 no posee MPU y algunas características de Debug adicionales