Sistemas Electrónicos Digitales

Tema #3

3. Arquitectura ARM

Temario

1. Introducción
2. GPIO: General Purpose Input/Output
3. Arquitectura Arm Cortex-M4
4. Interrupciones
5. C en ensamblador
6. Temporizadores (Timers)
7. Direct Memory Access
8. Comunicaciones Serie
9. Conversores A/D y D/A

INDICE ESPECÍFICO

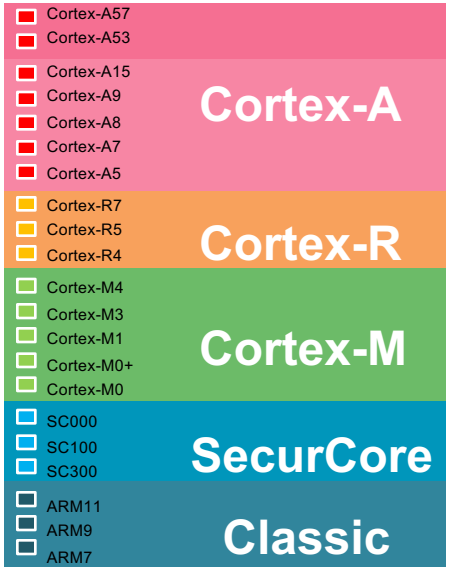
- **Arquitectura**
  - Evolución
  - Características
- **Registros**
- **Mapa de Memoria**
- **Conjunto de Instrucciones**

ARM Cortex-MXX

ARQUITECTURA

Familia de procesadores ARM

- **Cortex-A series (Application)**
  - Procesadores de **altas prestaciones**, capaces de ejecutar un Sistema Operativo completo como por ejemplo Linux
  - Entre sus aplicaciones están: smartphones, TV, libros digitales, etc.
- **Cortex-R series (Real-time)**
  - Altas prestaciones en aplicaciones de **tiempo real**
  - Gran fiabilidad
  - Entre sus aplicaciones están: sistemas de frenado para automóviles, sistemas de seguridad etc.
- **Cortex-M series (Microcontroller)**
  - Aplicaciones de tiempo determinista donde el precio es un factor importante
  - Entre sus aplicaciones tenemos: dispositivos de señal mixta (que incluyen parte analógica y digital), sensores inteligentes, sistemas empotrados en vehículos y transportes.
- **SecurCore series**
  - Aplicaciones de alta seguridad (security, not safety).
- **Nomenclatura de modelos anteriores.**
  - Familias ARM7, ARM9, ARM11

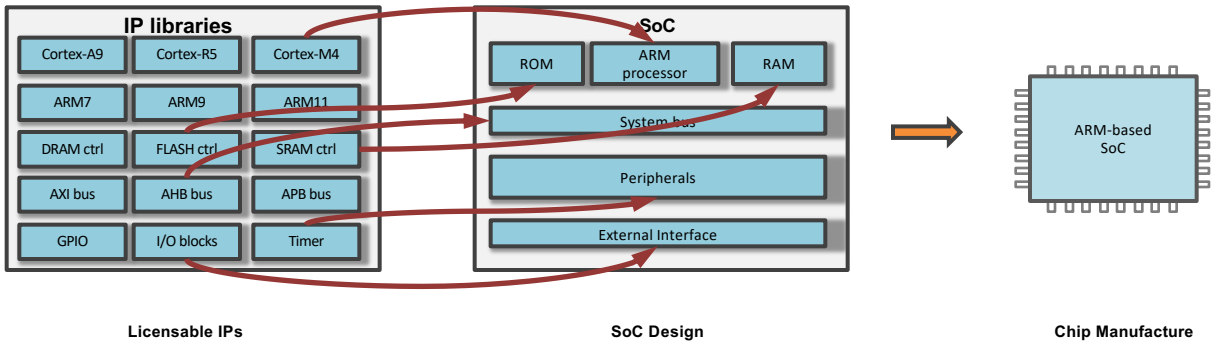


Diseño de un procesador basado en ARM (SoC)

El diseño de un Sistema en un Chip (SoC) requiere de los siguientes pasos:

- Seleccionar un conjunto de núcleos IP de ARM y/u otros IP
- Integrar los núcleos IP en el diseño de un chip
- Pedir a un fabricante de semiconductores que nos haga el chip

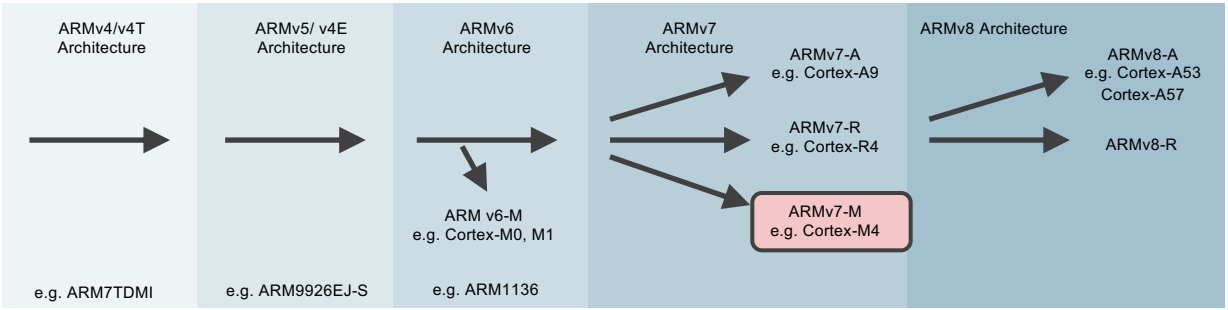
De esta forma el tiempo de puesta en mercado se reduce frente a un ASIC.



Diferencias entre Procesadores ARM y Arquitecturas ARM

Son dos conceptos distintos. La confusión surge porque ambos usan el prefijo ARM:

- **Arquitectura ARM**
  - Describe los detalles de instrucciones, modelo de programación, interrupciones, mapa de memoria …
  - Esta documentada en el manual de referencia de la arquitectura
- **Procesador ARM**
  - Desarrollado en base a una de las arquitecturas ARM
  - Contiene detalles de implementación real, como diagramas de tiempos, mapa concreto de memoria …
  - Su documentación principal es la hoja de características (processor's Technical Reference Manual)



As of Dec 2013

Familia de procesadores ARM Cortex-M

Processor	ARM Architecture	Core Architecture	Thumb®	Thumb®-2	Hardware Multiply	Hardware Divide	Saturated Math	DSP Extensions	Floating Point
Cortex-M0	ARMv6-M	Von Neumann	Most	Subset	1 or 32 cycle	No	No	Software	No
Cortex-M0+	ARMv6-M	Von Neumann	Most	Subset	1 or 32 cycle	No	No	Software	No
Cortex-M1	ARMv6-M	Von Neumann	Most	Subset	3 or 33 cycle	No	No	Software	No
Cortex-M3	ARMv7-M	Harvard	Entire	Entire	1 cycle	Yes	Yes	Software	No
Cortex-M4	ARMv7E-M	Harvard	Entire	Entire	1 cycle	Yes	Yes	Hardware	Optional

Características del Procesador Cortex-M4

- **Procesador RISC** (Reduced Instruction Set Computing) de 32-bits
- **Arquitectur Harvard**
  - Buses de datos y de instrucciones separados
- **Conjunto de instrucciones**
  - Incluye las instrucciones Thumb®-1 (16-bit) y Thumb®-2 (16/ 32-bit)
- **Pipeline** de 3 estados
  - **Fech** (carga de instrucción); **Decode** (decodificación); **Execute** (se leen los operandos de los registros, se operan con al ALU, se actualizan los registros)
  - Con **predicción de salto** (branch speculation): en instrucciones de salto condicional, se carga la dirección de la instrucción para la que la condición es verdadera antes de la ejecución de la comprobación de condición, y si la comprobación es falsa se descarta.
- **Ejecución eficiente**
  - 1.25 – 1.95 DMIPS/MHz (Dhrystone Million Instructions Per Second / MHz)
- **Tipo de interrupciones** disponibles
  - Non-maskable Interrupt (NMI): + 1 hasta 240 interrupciones
  - 8 a 256 niveles de prioridad

Características del Procesador Cortex-M4 (II)

- **Modo de reposo (Sleep Modes)**
  - Hasta 240 interrupciones de salida de reposo (Wake-up Interrupts)
  - Incluye instrucciones WFI (Wait For Interrupt) y WFE (Wait For Event) y "Sleep On Exit"
  - Señales "Sleep" y "Deep Sleep"
  - Modo de Retención (opcional) con "ARM Power Management Kit"
- **Instrucciones avanzadas**
  - Divisor Hardware (2-12 Ciclos)
  - 16, 32-bit MAC, doble 16-bit MAC todo en un ciclo de ejecución
  - Aritmética SIMD de 8, 16-bits
- **Depuración**
  - Puertos opcionales para depuración : JTAG y Serial-Wire Debug (SWD)
  - Hasta 8 puntos de ruptura (Breakpoints) y 4 de observación (Watchpoints)
- **Memory Protection Unit (MPU)**
  - Característica opcional. Proporciona 8 regiones MPU con subregiones y "background region"

Diagrama de bloques Cortex-M4

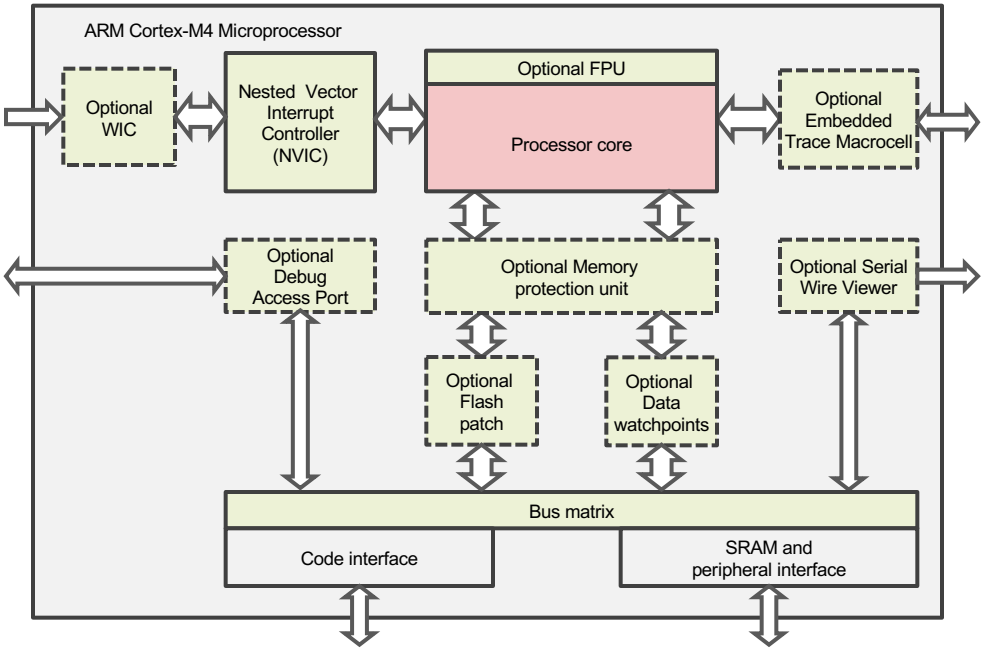


Diagrama de bloques Cortex-M4 (II)

- **Núcleo del procesador (core)**
  - Contiene los registros, ALU, buses internos y lógica de control
  - Los registros son de 32-bit, hay 16 en total, tanto de propósito general como del uso específico
- **Procesamiento paralelo (pipeline stages)**
  - Tiene tres estado de proceso: lectura (fetch), decodificación (decode), y ejecución (execution)
  - Algunas instrucciones pueden necesitar varios ciclos para la ejecución: el proc. paralelo se retrasa
  - El procesamiento paralelo ha de reiniciarse si aparece una instrucción de salto
  - Se pueden leer dos instrucciones en un ciclo (instrucciones de 16 bits)

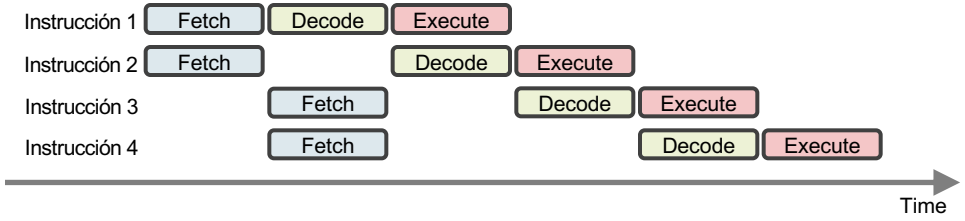


Diagrama de bloques Cortex-M4 (III)

- **Nested Vectored Interrupt Controller (NVIC)**
  - Hasta 240 señales de solicitud de interrupción y una interrupción no enmascarable (NMI)
  - Maneja automáticamente interrupciones anidadas, como comparación de prioridades entre las solicitudes de interrupción y el nivel de prioridad actual
- **Wakeup Interrupt Controller (WIC)**
  - Para aplicaciones de baja potencia, el microcontrolador puede entrar en modo de suspensión apagando la mayoría de los componentes.
  - Cuando se detecta una solicitud de interrupción, el WIC puede informar a la unidad de administración de energía para encender el sistema.
- **Memory Protection Unit (optional)**
  - Se utiliza para proteger el contenido de la memoria, ej. Hacer que algunas regiones de memoria sean de solo lectura o impedir que las aplicaciones de usuario accedan a datos privilegiados

Arquitectura ARM

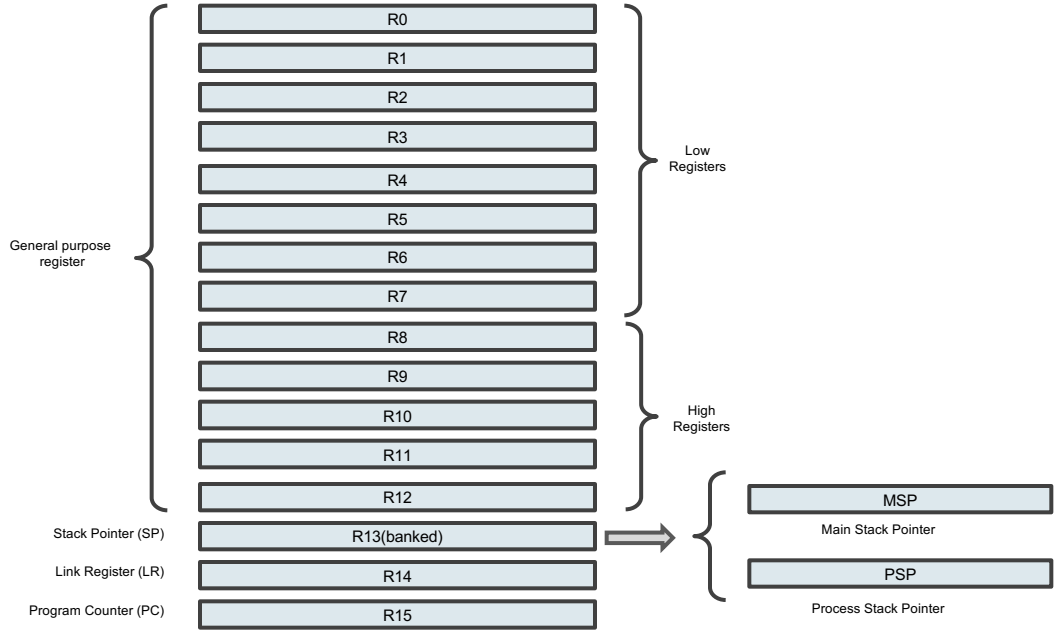
REGISTROS EN ARM CORTEX-M4

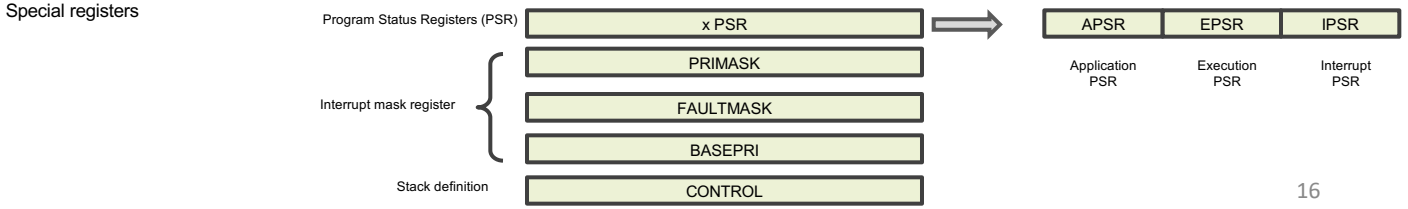
Registros en Cortex-M4

- **Registros del procesador.**
  - Registros internos que se usan para almacenar y procesar datos temporales dentro del núcleo del procesador.
  - Todos los registros están dentro del núcleo del procesador y por lo tanto se puede accede a ellos muy rápidamente.
  - Arquitectura "**Load-store**"
    - Para procesar datos de la memoria primero se cargan desde la memoria a los registros. Se procesan dentro del núcleo del procesador usando solo los datos de los registros y se escriben los resultados a memoria, si hay algún resultado.
- **Registros para Cortex-M4**
  - Banco de Registros
    - 16 registros de 32-bit (13 son de propósito general)
  - Registros de uso específico

Registros en Cortex-M4 (II)

- **Esquema con los registros del procesador.**

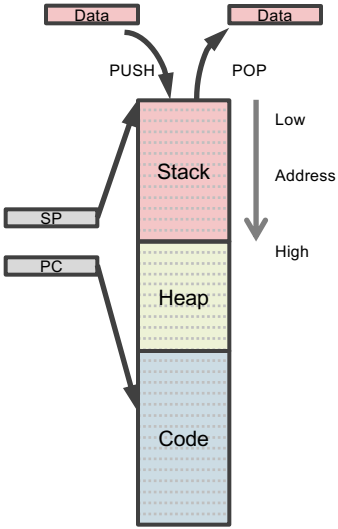




Registros en Cortex-M4 (III)

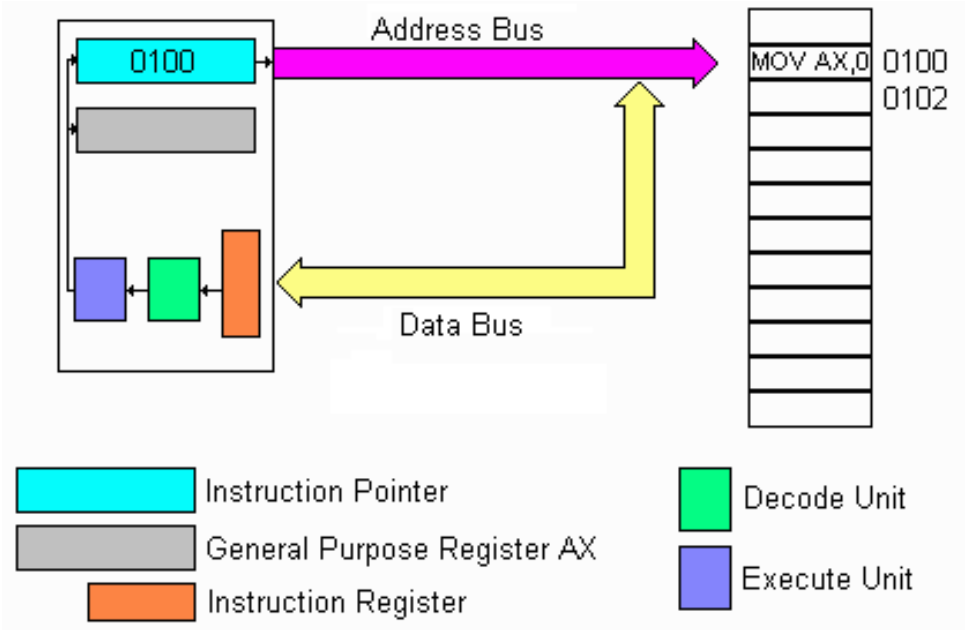
Descripción de los registros del procesador.

- **R0 – R12: Registros de propósito general**
  - Los registros (R0 – R7) se pueden utilizar en cualquier instrucción
  - Los registros (R8 – R12) no pueden utilizarse siempre.
    Por ejemplo, no son accesibles para algunas instrucciones Thumb (16-bit)
- **R13: Puntero de Pila (SP)**
  - Guarda el valor actual de la cima de pila
  - Se usa para guardar el contexto de un programa cuando se cambia a otra tarea
  - Cortex-M4 tiene dos SPs:
    + el SP principal, usado en aplicaciones que requieren acceso privilegiado, como por ejemplo en el kernel de un sistema operativo, o en el manejo de excepciones,
    + el SP de proceso, usado en aplicaciones básicas (que no manejan excepciones)
- **Contador de Programa (PC)**
  - Contiene la dirección de la instrucción actual
  - Automáticamente se incrementa en 4 al finalizar cada instrucción (para instrucciones de 32-bit), excepto si la instrucción provoca un salto
  - Una instrucción de salto, tal como la llamada a una función, cambia expresamente el PC a una dirección específica, y guarda el valor del PC (dirección de retorno) al Link Register (LR)



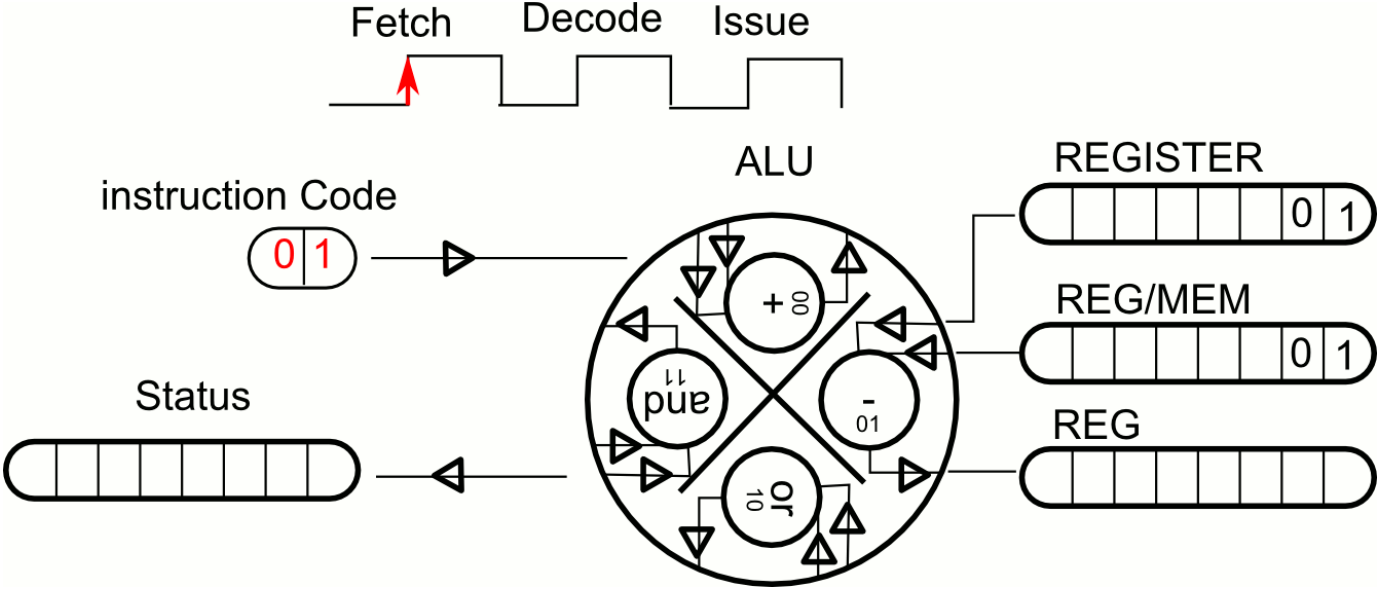
Registros en Cortex-M4 (IV)

- **Ejemplo de ejecución de una instrucción**



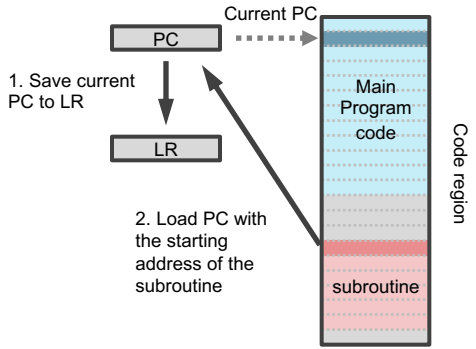
Registros en Cortex-M4 (IVb)

- **Ejemplo de ejecución de una instrucción**

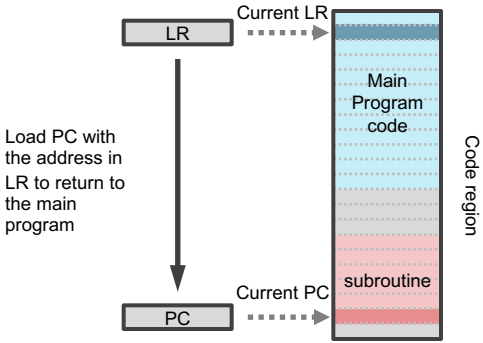


Registros en Cortex-M4 (V)

- **R14: Link Register (LR)**
  - LR se usa para almacenar la dirección de retorno de una subrutina o una llamada a una función
  - El contador de programa (PC) se cargará con el valor de LR una vez la función finaliza



Call a subroutine



Return from a subroutine to the main program

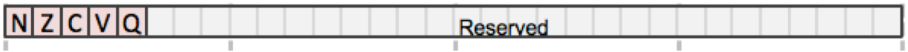
Registros en Cortex-M4 (VI)

- **xPSR, es el registro de estado (Program Status Register)**
  - Nos proporciona información sobre la ejecución del programa y sobre las banderas de la ALU
  - Lo podemos dividir en tres zonas
    - Application PSR (APSR)
    - Interrupt PSR (IPSR)
    - Execution PSR (EPSR)

Registros en Cortex-M4 (VII)

PARTES DEL REGISTRO DE ESTADO:

- **APSR**

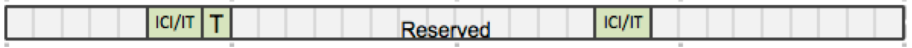


  - N: negative flag – indicador de resultado negativo: vale “1” si el resultado de la ALU es negativo
  - Z: zero flag – indicador de cero: se pone a “1” si el resultado de la ALU es cero
  - C: carry flag – indicador de acarreo: se pone a “1” si se produce desbordamiento en una operación sin signo
  - V: overflow flag – indicador de desbordamiento: se pone a “1” si se produce desbordamiento en una operación con signo
  - Q: sticky saturation flag – indicador de saturación: se pone a “1” si se produce saturación en una instrucción de aritmética saturada, o desbordamiento en ciertas instrucciones de multiplicación

- **IPSR**

  - ISR number – número de subrutina de interrupción que se está ejecutando en un momento dado

- **EPSR**



  - T: Thumb state – siempre a “1” ya que Cortex-M4 solo funciona en estado “Thumb”
  - IC/IT: Interrupt-Continuable Instruction (ICI) bit, IF-THEN instruction status bit

Registros en Cortex-M4 (VIII)

REGISTROS DE MÁSCARA DE INTERRUPCCIÓN:

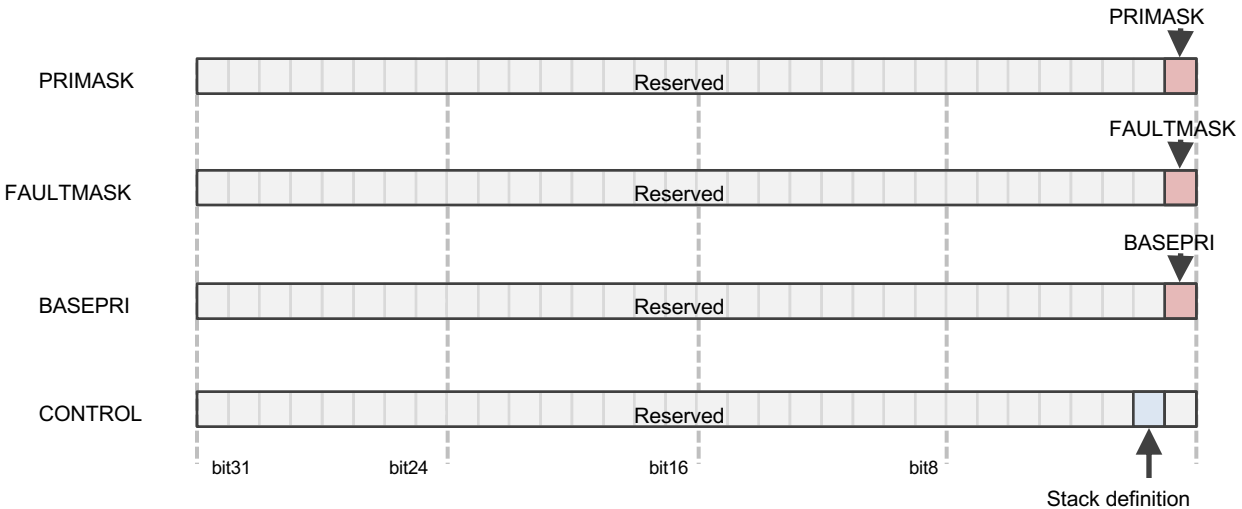
Son registros en los que sólo el bit menos significativo tiene información (**1-bit registers**)

- **PRIMASK**
  - Si el bit está a uno, se bloquean todas las interrupciones, excepto las no-enmascarables (NMI) y la excepción de fallo de hardware
- **FAULTMASK**
  - Si el bit está a uno, se bloquean todas las interrupciones, excepto las no-enmascarables (NMI)
- **BASEPRI**
  - Si el bit está a uno, se bloquean todas las interrupciones de la misma o inferior prioridad

REGISTRO DE CONTROL:

- **Es un registro con un sólo bit de información: bit de definición de la pila (stack definition)**
  - Si el bit está a uno, se usa la pila de proceso y por tanto el puntero de la pila de proceso (PSP)
  - Si el bit está a cero, se usa la pila principal y por tanto el puntero de la pila principal (MSP)

Registros en Cortex-M4 (IX)



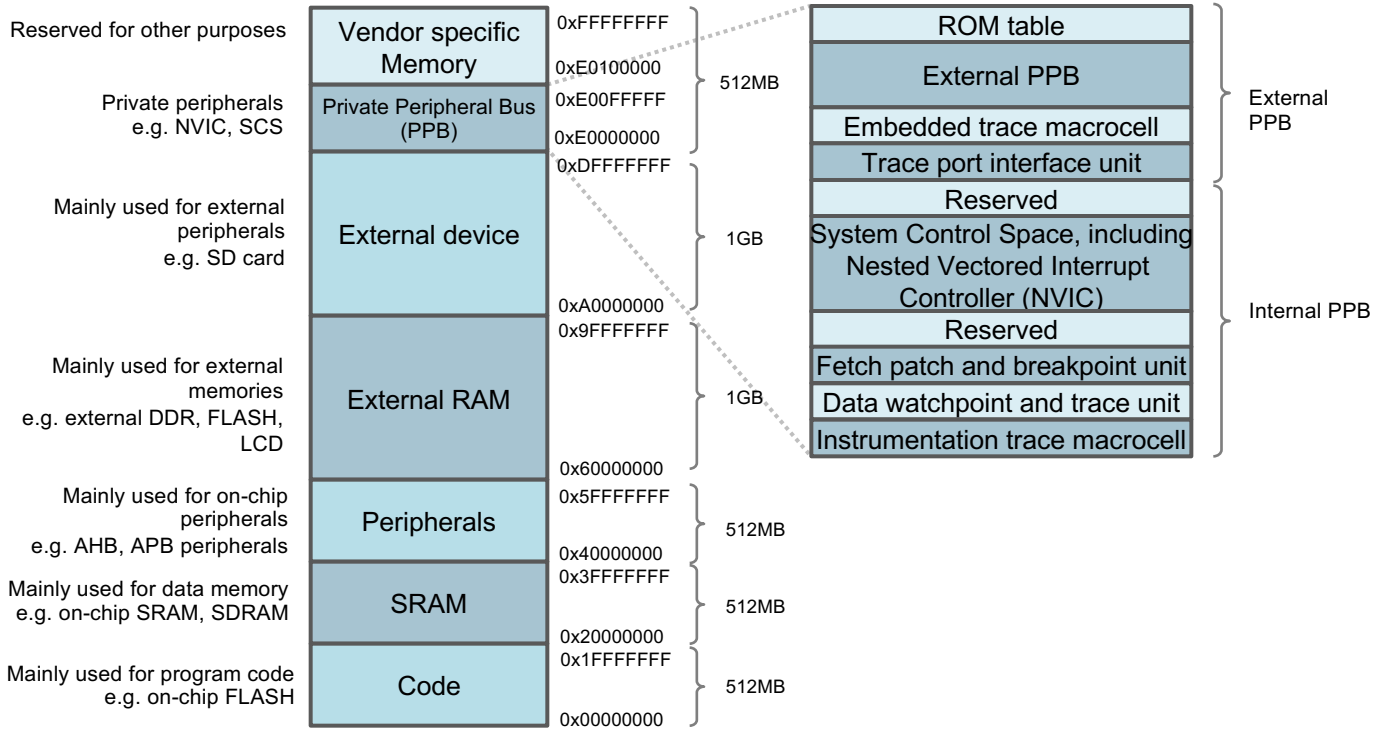
Arquitectura ARM

Mapa de memoria

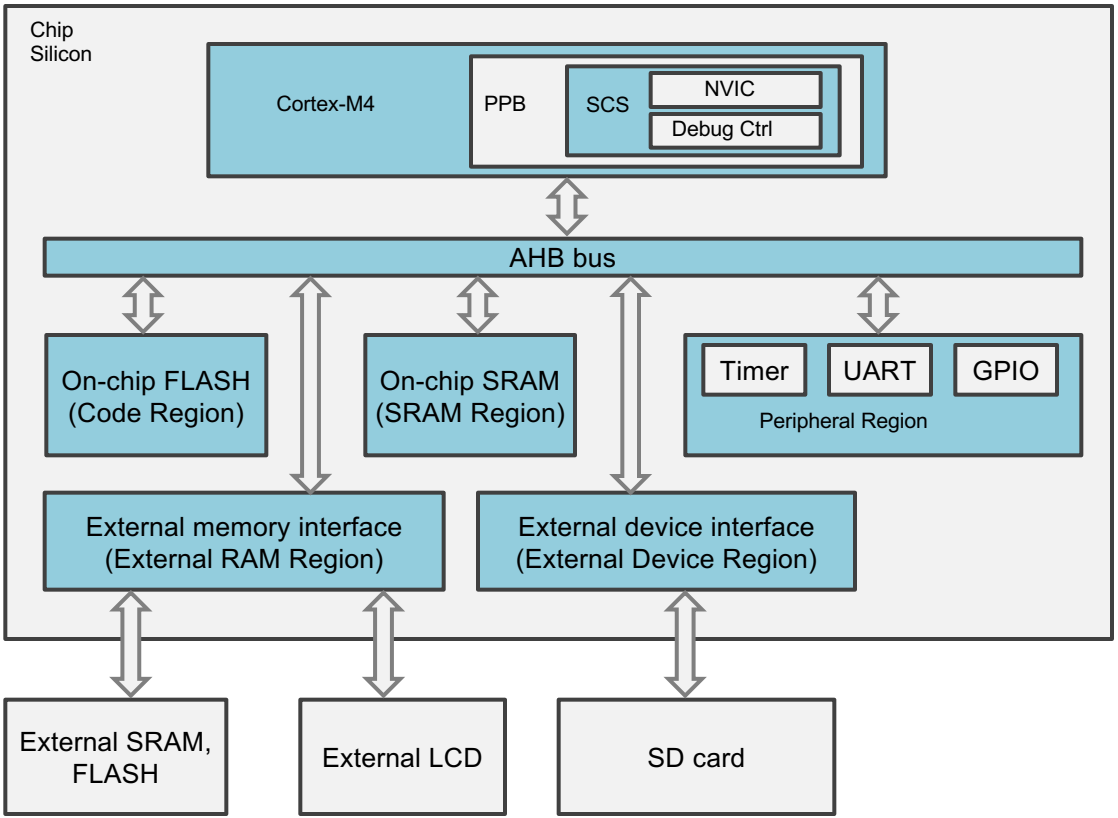
Cortex-M4: mapa de memoria (repaso)

- El procesador Cortex-M4 tiene **4 GB de espacio de direcciones de memoria**
- Este espacio se divide en **regiones**
  - Cada región se da para un **uso recomendado**
  - Facilita al programador de software la **portabilidad** de los programas entre diferentes dispositivos
- Sin embargo, el uso real del mapa de memoria también puede ser definido de manera **flexible** por el usuario (excepto algunas direcciones de memoria fija, como el bus periférico interno privado PPB)

Cortex-M4: mapa de memoria (II) (repaso)

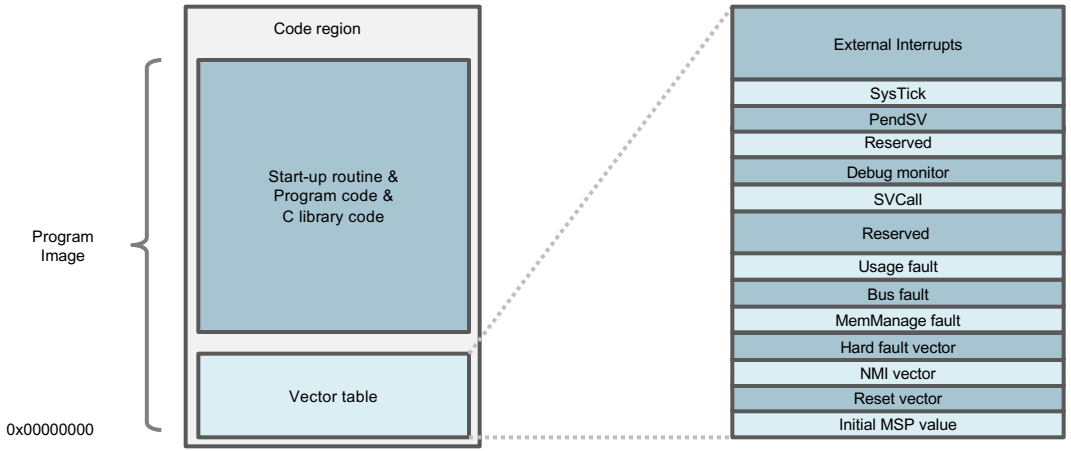


Cortex-M4: ejemplo mapa de memoria (repaso)

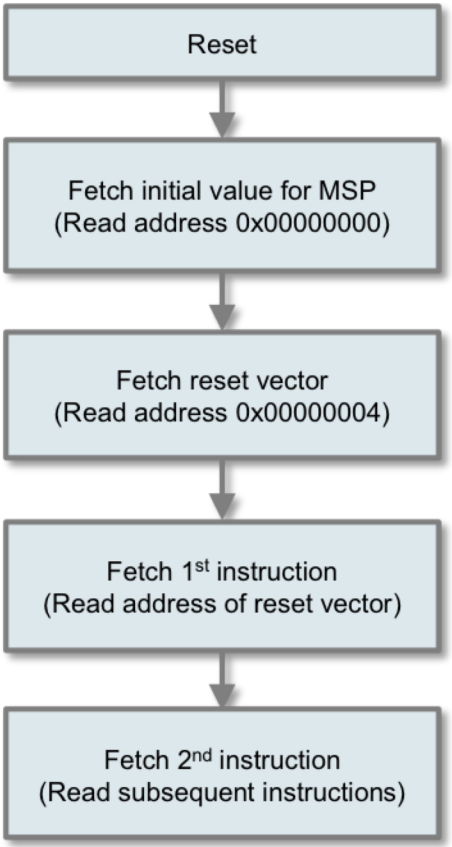


Cortex-M4 Imagen del Programa (Program Image)

- La imagen del programa en Cortex-M4 contiene:
  - Tabla de vectores: incluye las direcciones de inicio de las excepciones (vectores) y el valor del main stack point (MSP);
  - Rutina en C de puesta en marcha;
  - Código de programa: código de aplicación y datos;
  - Código de biblioteca C: códigos de programa para funciones de biblioteca C.



Cortex-M4 Imagen del Programa (II)

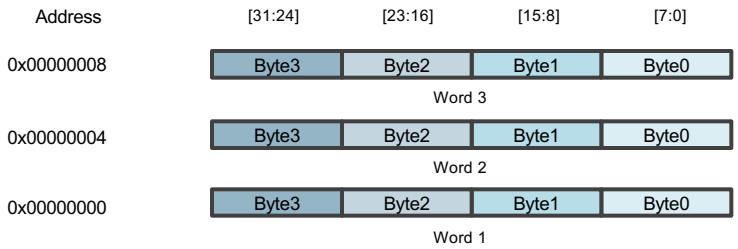


Después de reiniciar, el procesador:

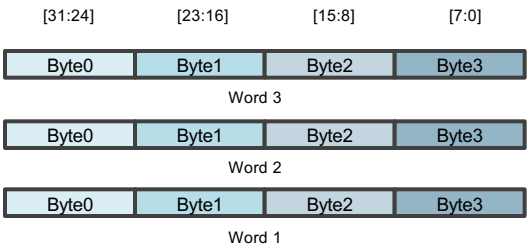
- Primero lee el valor inicial de MSP;
- Luego lee el vector de reset;
- Bifurca al inicio de la dirección de ejecución del programa (reset handler);
- Posteriormente ejecuta las instrucciones del programa.

Cortex-M4 “Endianness”

- Endian se refiere al orden de los bytes almacenados en la memoria.
  - **Little endian**: el byte más bajo de un tamaño de palabra se almacena en los bits 0 a 7
  - **Big endian**: el byte más bajo de un tamaño de palabra se almacena en el bit 24 al bit 31
- Cortex-M4 soporta tanto little endian como big endian
- Sin embargo, “Endianness” solo existe en el nivel de hardware.



Little endian 32-bit memory



Big endian 32-bit memory

Operaciones “Bit-band”

- La operación de bit-band permite que **una sola operación de carga / almacenamiento acceda a un solo bit en la memoria**, por ejemplo, para cambiar un solo bit de un dato de 32 bits:
- **Operación normal sin bit-band** (lectura-modificación-escritura):
  - Leer el valor de los datos de 32 bits
  - Modificar un solo bit del valor de 32 bits (otros bits sin cambios)
  - Escribir el nuevo valor en la dirección
- **Operación de bit-band**:
  - Escribir directamente un solo bit (0 o 1) en la "dirección de alias de banda de bits" de los datos
- **Dirección de alias de bit-band**
  - Cada dirección de alias de bit-band se asigna a una dirección de datos real
  - Al escribir en la dirección de alias de bit-band, solo se modificará un bit de los datos

Ejemplo de Operaciones “Bit-band”

- Por ejemplo, para establecer el bit [3] de una palabra en la dirección 0x20000000:

```
;Read-Modify-Write Operation

LDR     R1, =0x20000000  ;Setup address
LDR     R0, [R1]         ;Read
ORR.W   R0, #0x8         ;Modify bit
STR     R0, [R1]         ;Write back
```

```
;Bit-band Operation

LDR     R1, =0x2200000C  ;Setup address
MOV     R0, #1           ;Load data
STR     R0, [R1]         ;Write
```

- Operación de lectura-modificación-escritura
  - Lea la dirección de datos reales (0x20000000)
  - Modifica el bit deseado (mantiene otros bits sin cambios)
  - Escribe los datos modificados de nuevo
- Operación de banda de bits
  - Establece directamente el bit escribiendo "1" en la dirección 0x2200000C, que es la dirección de alias del cuarto bit de los datos de 32 bits en 0x20000000
    - En efecto, esta instrucción única se asigna a 2 transferencias de bus: lee datos de 0x20000000 al búfer, y luego escribe a 0x20000000 desde el búfer con el bit [3] establecido

Arquitectura ARM

Conjunto de Instrucciones para el procesador ARM Cortex-M4

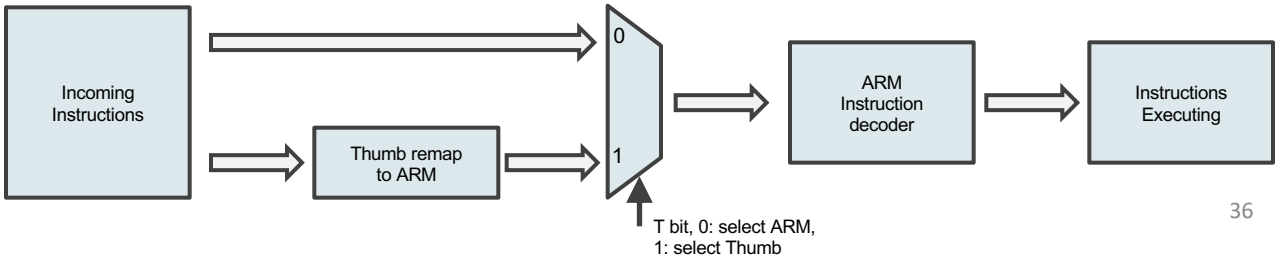
Conjunto de Instrucciones ARM y Thumb®

- **Conjunto de instrucciones ARM inicial**
  - Conjunto de instrucciones de 32 bits, llamado instrucciones ARM.
  - Potente y buen rendimiento.
  - Memoria de programa más grande en comparación con los procesadores de 8 y 16 bits.
  - Mayor consumo de energía

- **Conjunto de instrucciones Thumb-1**
  - Conjunto de instrucciones de 16 bits, utilizado por primera vez en el procesador ARM7TDMI en 1995
  - Proporciona un subconjunto de las instrucciones ARM, lo que proporciona una mejor densidad de código en comparación con la arquitectura RISC de 32 bits
  - El tamaño del código se reduce en un ~ 30%, pero el rendimiento también se reduce en un ~ 20%

Conjunto de Instrucciones ARM y Thumb® (II)

- **Mezcla de conjuntos de instrucciones ARM y Thumb-1**
  - Se juntan los beneficios de las instrucciones ARM de 32 bits (alto rendimiento) y Thumb-1 de 16 bits (alta densidad de código)
  - Se utiliza un multiplexor para cambiar entre dos estados: el “estado ARM” (32 bits) y el “estado Thumb” (16 bits)
- **Conjunto de instrucciones Thumb-2**
  - Consta de instrucciones Thumb de 32 bits y conjuntos de instrucciones Thumb-1 originales de 16 bits.
  - En comparación con el conjunto de instrucciones ARM de 32 bits, el tamaño del código se reduce en aproximadamente un 26%, mientras se mantiene un rendimiento similar
  - Capaz de manejar los requisitos de procesamiento en un único estado



Conjunto de Instrucciones ARM CORTEX M4

Procesadores ARM CORTEX M4:

- Arquitectura ARMv7-M
- Soporta el conjunto de instrucciones Thumb-2 de 32 bits
- Puede manejar todos los requerimientos de procesado en un único estado (estado Thumb)
- Comparado con los procesadores ARM que usan dos estados:
  - No se precisa multiplexación. Se mejora el tiempo de ejecución y el espacio dedicado a las instrucciones.
  - No se precisa separar el código en ficheros distintos con instrucciones ARM de 32 bits y Thumb-1 de 16 bits. El desarrollo y mantenimiento del software se simplifica.
  - Es más fácil obtener una ejecución eficiente y de buenas prestaciones.

Conjunto de Instrucciones ARM CORTEX M4 (II)

- **Sintaxis del código ensamblador ARM:**

  Label mnemonic operando1, operando2, … ; Comentarios

  - Label se usa como referencia de una dirección;
  - Mnemonic es el nombre de la instrucción;
  - Operando1 es el destino de la operación;
  - Operando2 es usualmente el origen de la operación;
  - Los comentarios se escriben después de un " ; ", y no afectan al programa;
  - Por ejemplo

    MOVS R3, #0x11 ;Guardar 0x11 en el registro R3to

  - El código ensamblador puede compilarse con ARM assembler (armasm) o herramientas de ensamblador de las disponibles en el mercado (e.g. GNU tool chain). Cuando se usan estas otras, la sintaxis de las etiquetas (labels) y comentarios puede ser ligeramente diferente.

Recursos recomendados

- Architecture Reference Manual:
  http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0403c/index.html

- Cortex-M4 Technical Reference Manual:
  http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439d/DDI0439D_cortex_m4_processor_r0p1_trm.pdf

- Cortex-M4 Devices Generic User Guide:
  http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf

Fin