Informe:

El procesador ARM Cortex-M3

DECLASSIFIED

- Procesador Cortex-M3
  - core Cortex-M3
    - Harvard
    - 3-stage pipeline + branch speculation
    - arquitectura ARMv7-M
    - 1,25 DMIPS/MHz
    - Thumb-2
    - single-cycle hardware multiplier
    - hardware divide
  - NVIC determinístico, baja latencia, 240 IRQs, 255 prioridades*
  - Bit banding
  - Sleep modes
  - PMU (PMSA) opcional
  - Embedded trace opcional

Postulados axiomáticos

Con cualquier herramienta se puede hacer cualquier trabajo

Ej: se puede poner un clavo con un destornillador, golpeándolo de culata. Se puede poner un tornillo con un martillo, entrándolo a los golpes.

Con la herramienta adecuada se hace el trabajo de forma más eficiente

Ej: con el destornillador, el tornillo entra más fácil y forma la rosca, siendo más difícil sacarlo. Con el martillo, el clavo entra más fácil.

Corolario

Con cualquier micro se puede realizar cualquier tarea. Cualquier micro se puede programar en C. Un micro con un set de instrucciones "C-friendly" o "C-compatible" va a permitir que el compilador genere código más compacto que se ejecute de forma más eficiente y ocupe menos espacio de memoria.

¿Qué
<insertar epíteto>
es
un set de instrucciones
"C-compatible" ?

- En C, todas las operaciones se realizan sobre enteros (a menos que se especifique lo contrario) => 32-bits
  - Promoción
- Desplazamientos a izquierda y derecha
- Bitfields
- Variables globales:
  - direccionamiento directo
    - el dato se encuentra en una posición de memoria, especificada por su dirección absoluta (o una reducción unívoca de ésta) en la instrucción
- Punteros:
  - direccionamiento indirecto
    - el dato se encuentra en una posición de memoria, cuya dirección está contenida en un registro, que oficia de puntero, referido en la instrucción.

- El puntero puede opcionalmente incrementarse o decrementarse antes o después del acceso a memoria
  - indirecto pre- o post-incrementado y pre- o post-decrementado, respectivamente.
- Arrays:
  - direccionamiento indexado
    - similar al indirecto, se agrega un desplazamiento respecto al valor indicado por el registro que oficia de base.
      - Dicho desplazamiento puede estar contenido o referenciado (un registro) en la instrucción
        - Hilando fino, el direccionamiento indirecto entonces, sería un caso particular del indexado en el que constante = 0 y no se utiliza registro.
- Variables locales -> registros => gran cantidad de registros
- Set de instrucciones ortogonal

Modos de direccionamiento en ARM

- Variación del modo indexado.
  - ARM propone un desplazamiento con signo y la modificación del registro base por el desplazamiento
    - pre-incrementado y pre-decrementado -> pre-indexado
    - post-incrementado y post-decrementado -> post-indexado.

Offset

- direccionamiento indexado puro, sin modificaciones en el valor del índice.

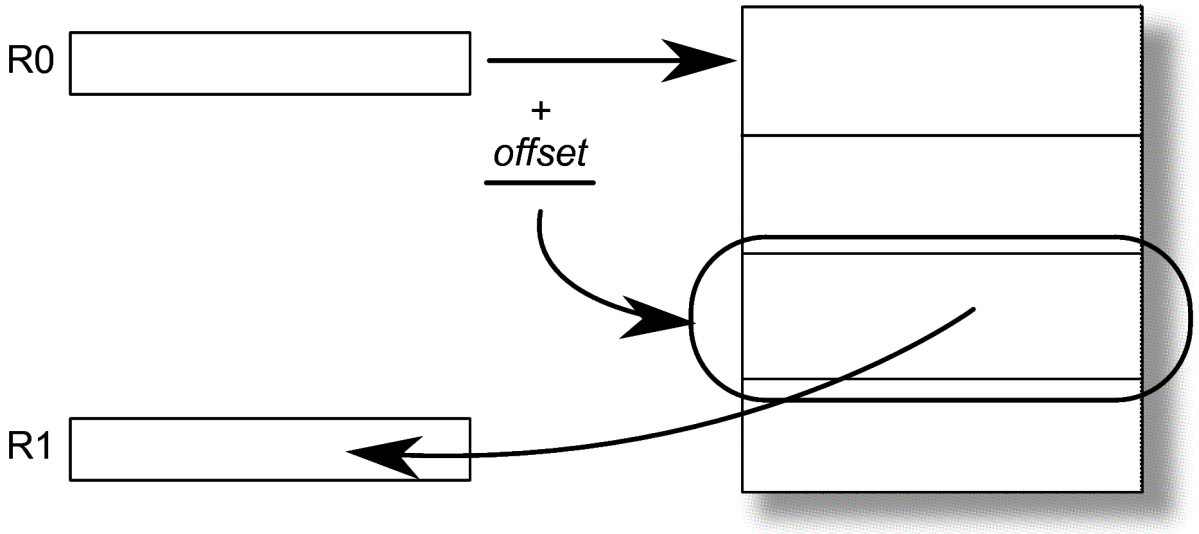
El offset a su vez puede ser:

Inmediato

el desplazamiento respecto de la base es una constante, y se encuentra dentro de la instrucción (no requiere bytes adicionales).

- miembro de una estructura
- parámetros de una función pasados en el stack.

```
LDR R1 [R0, #offset]
```



Registro

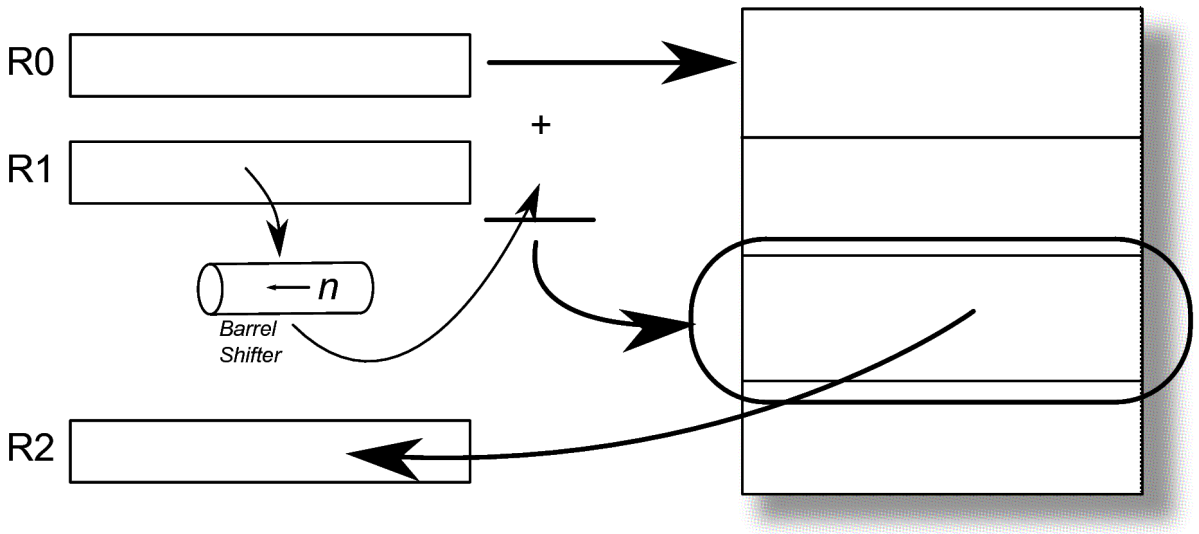
el desplazamiento respecto de la base se halla especificado por un registro.

- elementos de un array.

Escalado

barrel shifter, tamaños comunes de elementos de un array, siempre que sean potencias de 2.

```
LDR R2 [R0, R1, LSL #n]
```



Pre-Indexado

el desplazamiento indicado se suma o resta al registro base *antes* de realizar la operación, alterando su valor y permitiendo su utilización iterada en varios accesos dentro de, por ejemplo, un array.

Post-Indexado

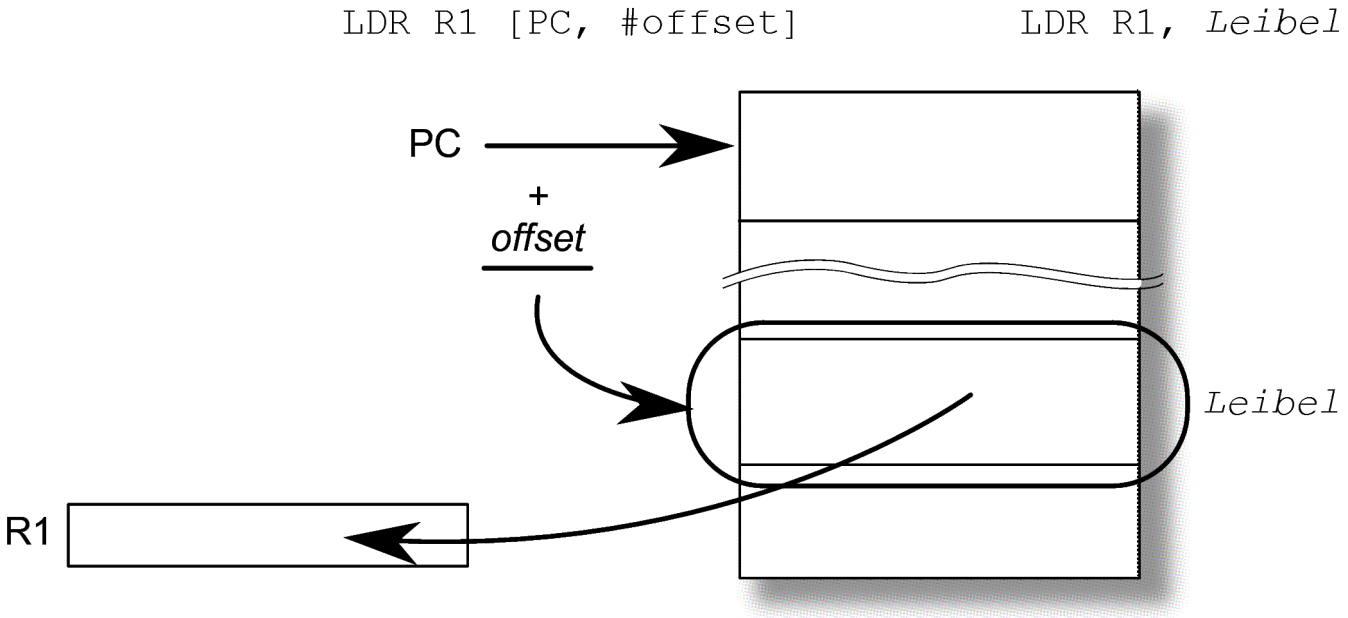
el desplazamiento indicado se suma o resta al registro base *después* de realizar la operación, con igual finalidad.

- ARM *no posee direccionamiento directo*
  - cualquier referencia a una posición absoluta de memoria debe hacerse cargando primero un registro con el valor de la dirección de memoria y luego accediendo mediante ese registro.
    - Es poco común acceder en una función a posiciones fijas de memoria que no tengan una referencia base (estén dentro de una estructura o un array).
    - Aún en I/O, los accesos no suelen ser sólo uno, sino varios, de modo que el utilizar un registro base resulta(ría) más eficiente a la larga

obvia e “inmediata” pregunta:

¿Cómo cargamos una dirección absoluta de memoria, es decir, un valor fijo en un registro?

- Técnicamente, ARM no posee direccionamiento inmediato.
  - Existen constantes inmediatas embebidas en la instrucción
    - no cubren el rango completo de 32-bits por razones obvias.
  - Se aloja dicho valor en alguna posición de memoria fija cercana y se accede con los modos de direccionamiento vistos, usando el PC como registro base y un offset fijo.



Barrel Shifter

Circuito secuencial capaz de desplazar una palabra binaria un número determinado de bits en un ciclo de clock.

Acelera notablemente las operaciones de normalización y la generación de potencias de 2.

ARM incorpora un barrel shifter en uno de los operandos de la ALU, de modo que en las operaciones de procesamiento es posible afectar uno de los operandos por un desplazamiento lógico o aritmético, o incluso una rotación.

El barrel shifter también es aplicable a las operaciones de carga, de modo de soportar el direccionamiento offset con registro escalado

Modos de operación, privilegios, stacks

El procesador puede operar en uno de dos modos:

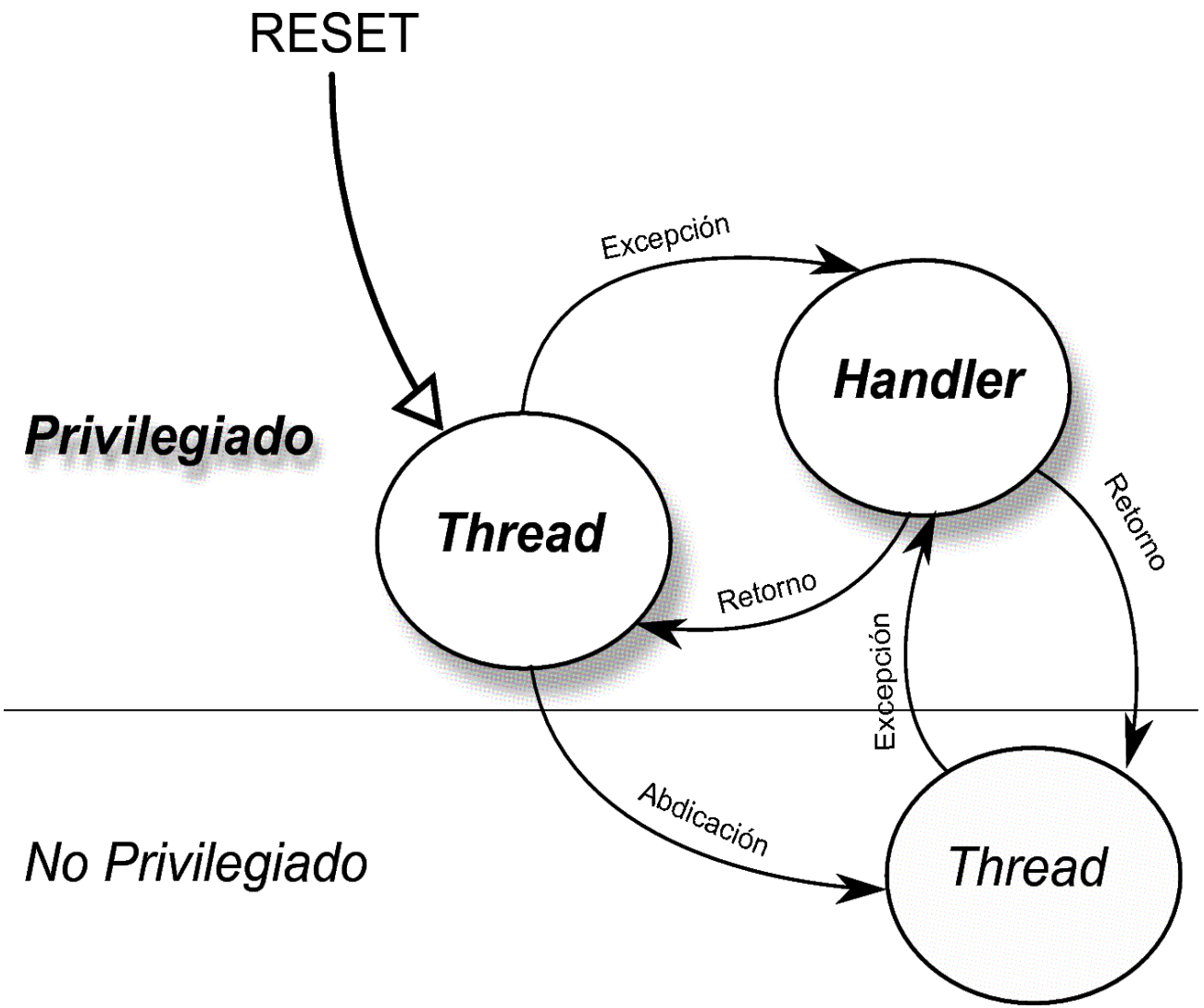
- THREAD
  - modo normal de operación, en el que el micro inicia al reset, y al que se vuelve luego de terminar de procesar excepciones.
- HANDLER
  - modo en el que el procesador está procesando una excepción.

El modo HANDLER es un modo *privilegiado*, mientras que en el modo THREAD, el procesador tiene a su vez dos niveles de privilegio:

- PRIVILEGIADO, en el que puede acceder a todos los recursos
- NO-PRIVILEGIADO, en el que se limitan ciertos recursos “sensibles” para la operación del sistema.

En el modo HANDLER, el procesador utiliza el stack principal.

En el modo THREAD, el procesador puede utilizar un *stack alternativo* si el sistema se configura para esto.


RESET
Excepción
Handler
Privilegiado
Thread
Retorno
Retorno
Excepción
Abdicación
No Privilegiado
Thread

Registros

Registros

Registros alternativos según el modo de trabajo

Thread

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13/MSP
R14/LR
R15/PC

R13/PSP

El registro SP puede configurarse para ser diferente en modo THREAD, de modo de poder separar stacks y aumentar la resistencia a errores de programación, aún sin hardware de control de acceso a memoria como una MPU.

Program Status Registers El estado del procesador puede observarse en el PSR (CURRENT PROGRAM STATUS REGISTER). Este registro es único y contiene a su vez tres registros específicos:

APSR (APPLICATION PROGRAM STATUS REGISTER)

Contiene los códigos de condición: N, Z, V y C, dados por el resultado de las operaciones lógicas y aritméticas y el flag Q (STICKY SATURATION), resultado de overflow en operaciones de saturación

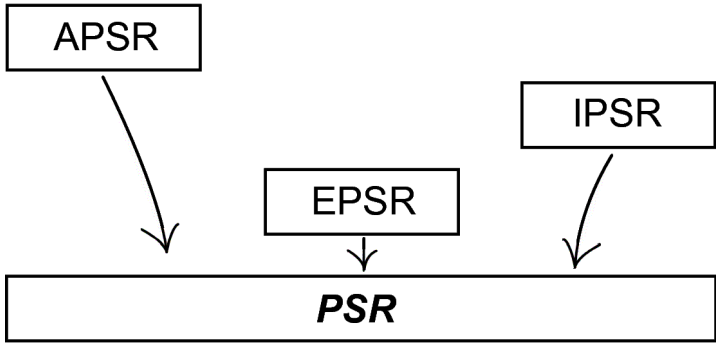
IPSR (INTERRUPT PROGRAM STATUS REGISTER)

Cuando el procesador está atendiendo una excepción, contiene el número de ésta. Caso contrario, contiene el valor cero.

EPSR (EXECUTION PROGRAM STATUS REGISTER)

Contiene el estado parcial de operaciones múltiples, de modo que puedan ser interrumpidas.

sólo el APSR es accesible en el modo THREAD NO-PRIVILEGIADO



Registros ***accesibles*** según el modo de trabajo

Thread/Handler	Thread (no-privilegiado)
APSR	***APSR***
IPSR	
EPSR	
PSR	

El acceso a estos registros se realiza mediante instrucciones que mueven de y a registros generales: MRS y MSR.

Registros especiales

Disponemos de tres registros para inhabilitar excepciones:

PRIMASK

permite inhabilitar todas las excepciones, excepto NMI y HARDFAULT.

FAULTMASK

permite inhabilitar todas las excepciones, excepto NMI.

BASEPRI

permite excepciones sólo de una prioridad mayor al valor escrito en este registro.

CONTROL

permite seleccionar el stack pointer y el nivel de privilegio.

Estos registros sólo son accesibles en modo privilegiado, mediante instrucciones especiales como los PSR.

Set de instrucciones

Entre los grupos de instrucciones tenemos:

- BRANCH, permiten cambios en el flujo de programa
- DATA PROCESSING, realizan operaciones sobre los registros
- DATA TRANSFER o LOAD/STORE, realizan transferencia de datos entre memoria y registros.

Instrucciones de transferencia de control (branch)

Branch

el salto relativo tradicional
puede ser condicional
amplio número de condiciones dado por cuatro bits.
offset de 24-bit con signo
rango de 32MB en ambos sentidos
salto de mayor distancia o transferencia a posición absoluta ?
debe cargarse directamente el PC.

Branch with Link

llamada a subrutina
la dirección de retorno se guarda en el LINK REGISTER (R14)
permite realizar llamadas rápidas
Si se desea anidar llamadas?
deberá salvarse el contenido de LR en el stack.

CBZ (COMPARE AND BRANCH IF ZERO)

si el registro referido *es* igual a cero, realiza un salto relativo a la posición indicada

CBNZ (COMPARE AND BRANCH IF NOT ZERO)

si el registro referido *no es* igual a cero, realiza un salto relativo a la posición indicada

Obtener una dirección o desplazamiento de una tabla

Un registro apunta a la tabla y otro provee el índice.

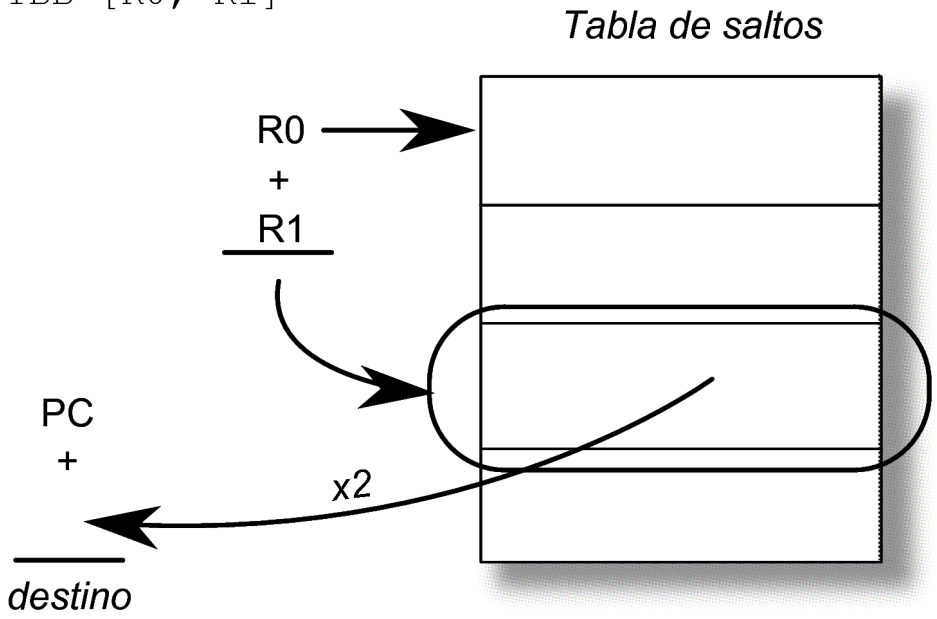
TBB (TABLE BRANCH BYTE)

para una tabla con elementos de 8-bits, provee saltos de hasta 2 × 255 bytes

TBH (TABLE BRANCH HALF-WORD)

para una tabla con elementos de 16-bits, provee saltos de hasta 2 × 65535 bytes

```
TBB [R0, R1]
```



Instrucciones de procesamiento de datos (data processing)

aritméticas y lógicas

- uno de los operandos siempre es un registro, y el otro puede ser un valor inmediato o un registro, modificado a través del barrel shifter.
- el resultado siempre es un registro, y no necesariamente debe ser uno de los operandos.
- dado que el PC es un registro más, es posible realizar operaciones sobre éste.

de comparación

poseen igual estructura que las anteriores, pero no tienen resultado numérico, es decir, no alteran ningún registro.

de desplazamiento

permiten modificar cualquier registro mediante el barrel shifter, pudiendo alojar el resultado en cualquier otro registro, incluido el PC.

de multiplicación

con hardware dedicado, 16x16=16 ó 32; 32x32=32 ó 64, etc.

de división

SDIV y UDIV, división con signo y sin signo, respectivamente, *mediante hardware.*

Es posible configurar al core para que genere una excepción USAGEFAULT ante una división por cero.

de saturación

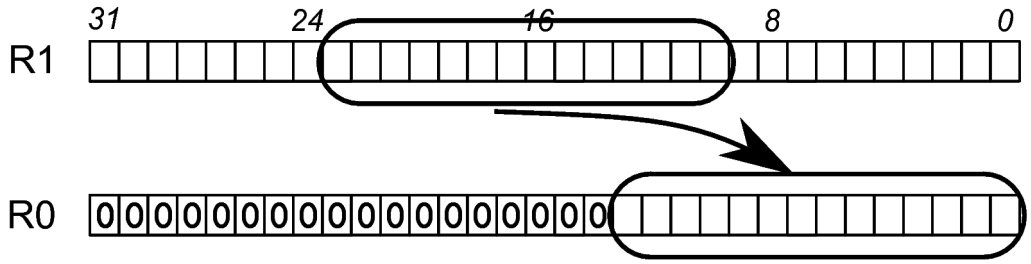
permiten reducir un valor de 32-bits a un formato más chico, por ejemplo 16-bits, saturando el resultado dentro del rango permitido

Instrucciones de procesamiento de bitfields

UBFX (UNSIGNED BITFIELD EXTRACT)

toma una cantidad de bits de un registro, a partir de una posición, y la coloca en otro registro comenzando por el bit menos significativo. El resto de los bits es puesto a cero. Equivale a extraer o desempaquetar una variable de bits para luego operar sobre ella

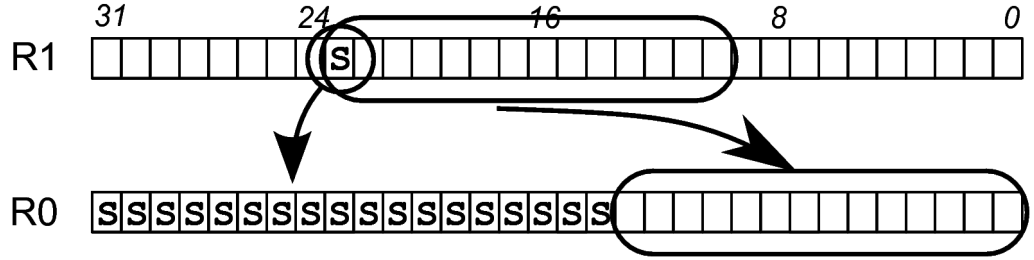
```
UBFX R0, R1, #10, #14
```



SBFX (SIGNED BITFIELD EXTRACT)

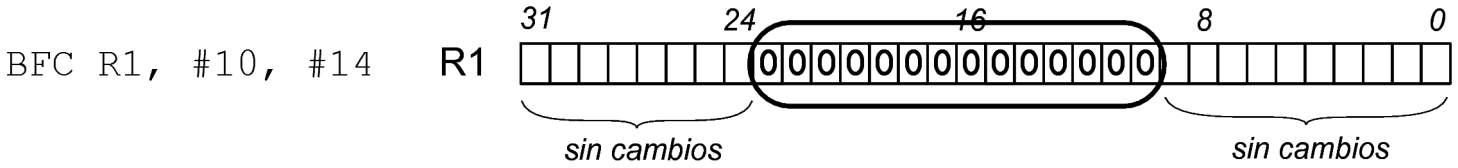
realiza la misma operación que UBFX, pero acompañada de una extensión de signo. Es decir, considera al bitfield como una variable con signo y lo extiende a los bits no utilizados.

```
SBFX R0, R1, #10, #14
```



BFC (BITFIELD CLEAR)

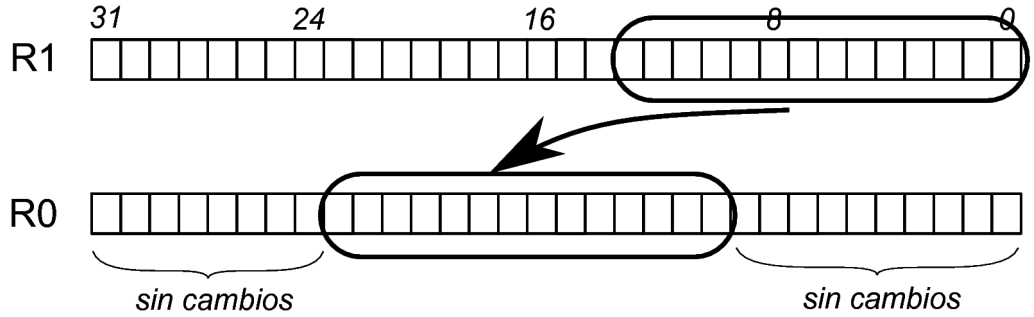
coloca al valor cero una variable bitfield. Es decir, sin alterar los demás bits, pone en cero un grupo contiguo de bits comenzando en una determinada posición.



BFI (BITFIELD INSERT)

Copia un grupo contiguo de bits de un registro, desde el menos significativo, colocándolos en una determinada posición en el registro destino, sin alterar los demás bits.

```
BFI R0, R1, #10, #14
```



Instrucciones de transferencia de datos (data transfer, load/store)

LDR y STR

- mueven datos de R a memoria y viceversa

Múltiples

LDM y STM

- transferencia de varios registros en una sola instrucción
- la arquitectura permite que se puedan interrumpir en medio del proceso
  - para esto existe un campo en el registro EPSR (EXECUTION PROGRAM STATUS REGISTER) que permite mantener el estado parcial.

Instrucciones de ejecución condicional

THUMB-2, bloque IT.

Este bloque de instrucciones se puede interrumpir.

Ejemplo:
algoritmo de Euclides para obtener el mayor número entero que divide a dos números dados:

```
int gcd (int i, int j){
    while (i != j)
        if (i > j)
            i -= j;
        else
            j -= i;
    return i;
}
```

```
            ; B start
loop:
            ITE    GT
                    ; conditional block, IF(GT), ELSE (!GT => LE)
            SUBGT  Ri, Ri, Rj           ; if "GT", i = i-j;
            SUBLE  Rj, Rj, Ri           ; else (if "LE"), j = j-i;
start:      CMP    Ri, Rj               ; set flags NZVC
            BNE    loop                 ; if "NE" (Z=0), then loop
```

- Thumb-2
  - cinco instrucciones en el loop, = líneas de código C
    - 10 bytes (caben en opcodes de 16-bits THUMB-2)
    - 7 ciclos (estimado, 4 + 3 branch (purga pipeline*)

(la velocidad de ejecución varía según las características del core, Cortex-M3 *especula* pero *no predice*)

Extensiones

La arquitectura ARMV7-M incorpora dos extensiones posibles.

El procesador CORTEX-M3 puede incorporar opcionalmente, a criterio del fabricante del micro, una de estas extensiones:

periférico en el SYSTEM CONTROL SPACE denominado MPU (MEMORY PROTECTION UNIT).

La MPU implementa la arquitectura PMSAv7 (PROTECTED MEMORY SYSTEM ARCHITECTURE).

control de acceso a varias zonas de memoria

no contiene tablas de páginas ni traslación de direcciones.

regiones desde 32 bytes hasta el total de la memoria (4GB)

Acceso a memoria

El CORTEX-M3 es un procesador con arquitectura de bus Harvard

Esto significa que posee dos buses de direcciones y datos para poder acceder a dos espacios de memoria a la vez

La mayoría de los lenguajes de programación no fueron diseñados para esta distinción

La arquitectura ARMV7-M define un mapa de memoria unificado.

¿cómo podemos pasar de un esquema a otro sin una visita al psiquiatra?

Buses internos y BUS MATRIX

bus ICODE

accesos a la memoria de código en búsquedas de opcodes y vectores de excepciones. Es un bus AHB-LITE de 32-bits.

bus DCODE

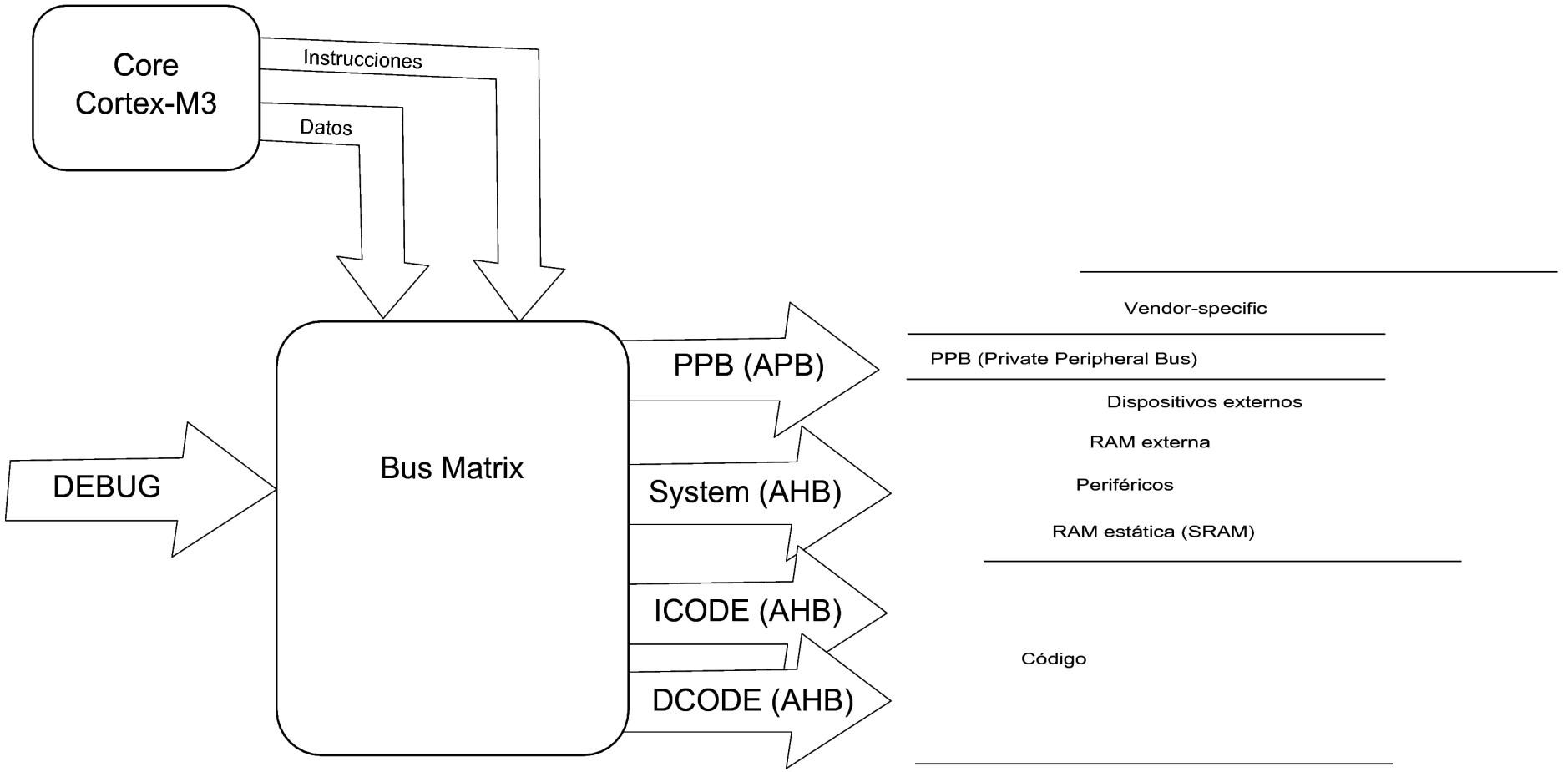
accesos de lectura/escritura a la memoria de código. Es un bus AHB-LITE de 32-bits.

bus PPB

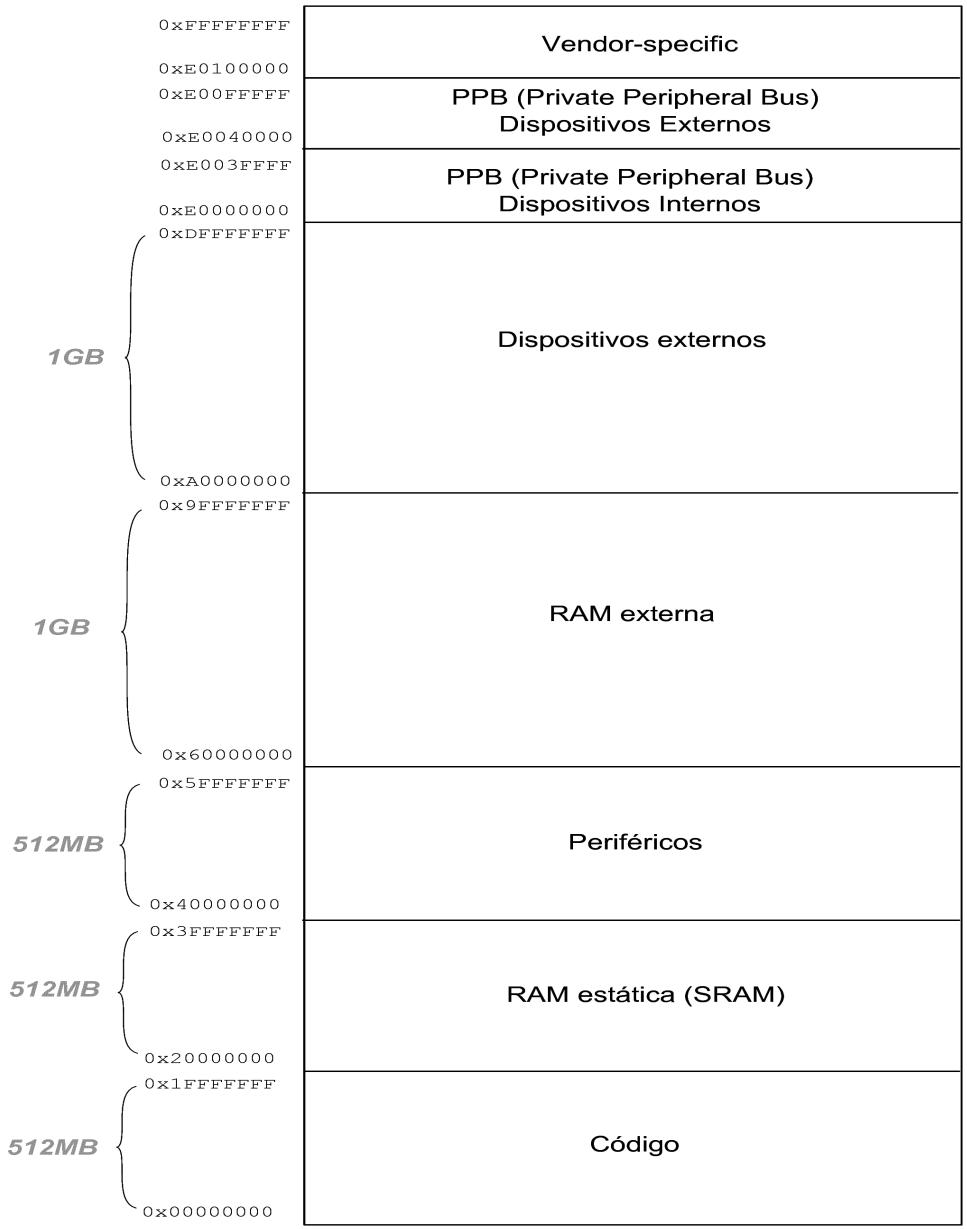
accesos de lectura/escritura al área de PPB. Es un bus APB de 32-bits

bus SYSTEM

accesos al resto del mapa de memoria. Es un bus AHB-LITE de 32-bits.


Core
Cortex-M3
Instrucciones
Datos
DEBUG
Bus Matrix
PPB (APB)
System (AHB)
ICODE (AHB)
DCODE (AHB)
Vendor-specific
PPB (Private Peripheral Bus)
Dispositivos externos
RAM externa
Periféricos
RAM estática (SRAM)
Código

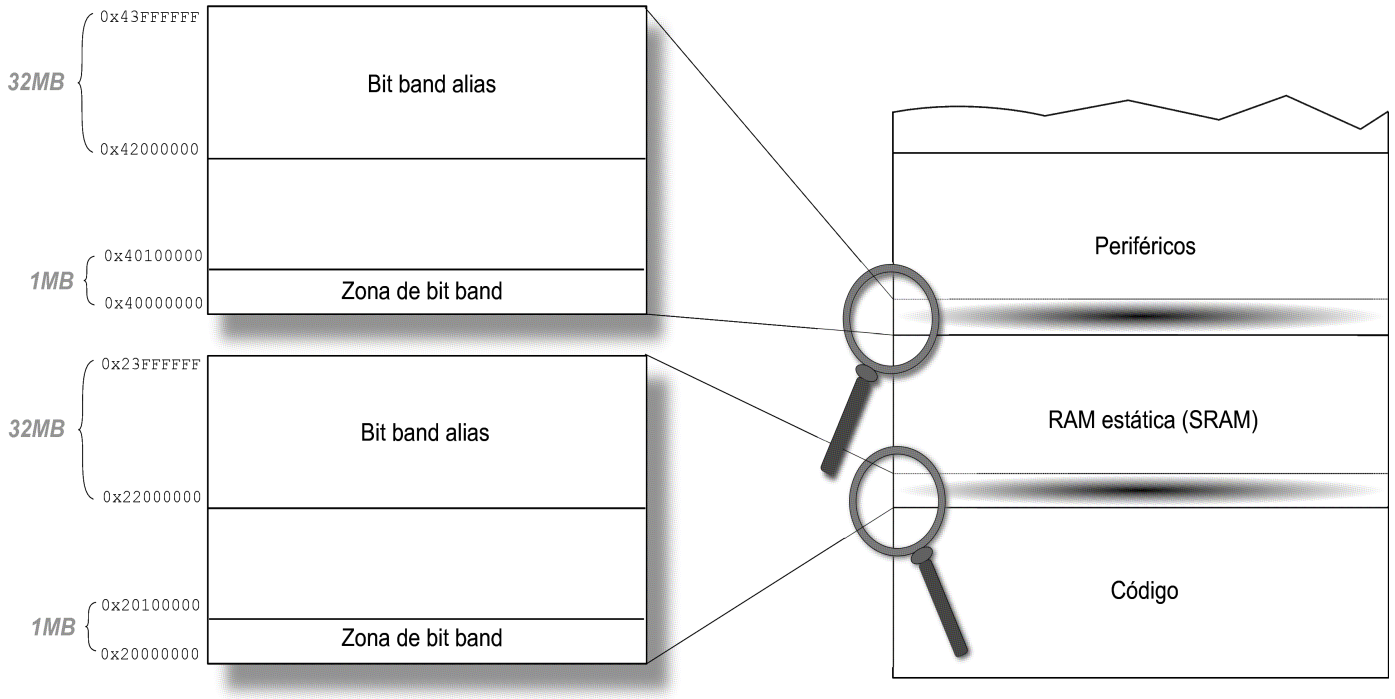
Mapa de memoria

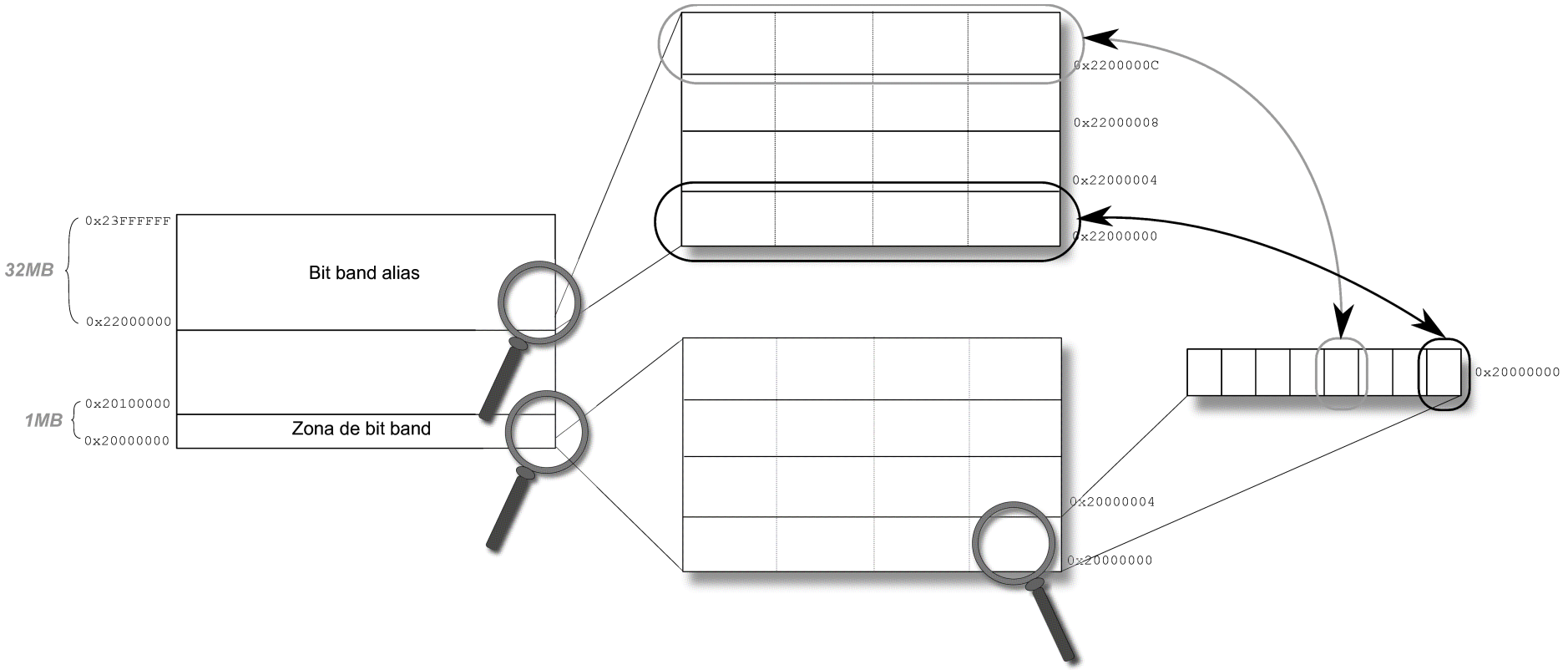


Bit banding

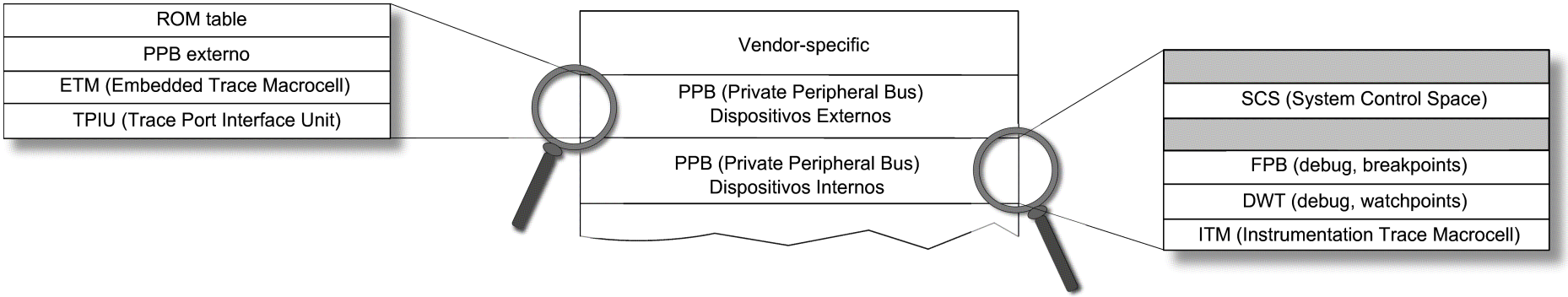
Acceso a RAM y periféricos como bits o como bytes/words

Misma variable en diferentes modos

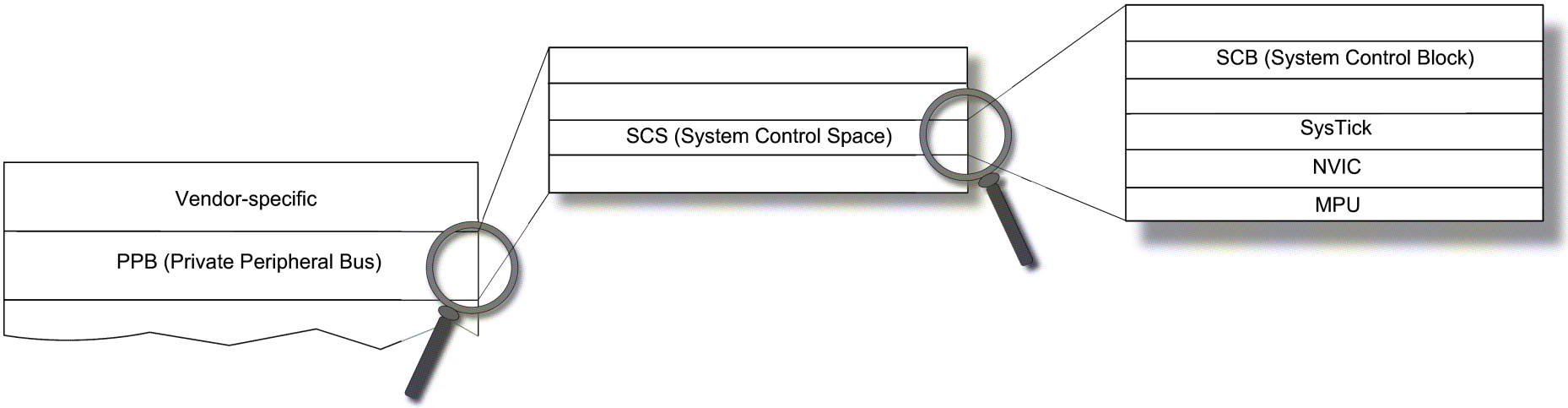



0x23FFFFFF
32MB
Bit band alias
0x22000000
0x20100000
1MB
Zona de bit band
0x20000000
0x2200000C
0x22000008
0x22000004
0x22000000
0x20000004
0x20000000
0x20000000

área(s) PPB:



espacio SCS:



System Control Space

El SYSTEM CONTROL SPACE se encuentra en el área de SISTEMA del mapa de memoria, dentro de la región PRIVATE PERIPHERAL BUS

SCB (SYSTEM CONTROL BLOCK)
el controlador de excepciones y demás yerbas
SYSTICK
el timer del sistema
NVIC (NESTED VECTORED INTERRUPT CONTROLLER)
el controlador de interrupciones vectorizadas
MPU
la unidad de protección de memoria, si existe.

System Control Block

provee el manejo de funciones clave para la operación de un procesador basado en esta arquitectura, fundamentalmente en el manejo de excepciones.

- Control de reset por software
- Manejo de la tabla o vector de excepciones
- Manejo del sistema de excepciones en sí
  - habilitación e inhabilitación
  - estado (inactiva, pendiente, activa)
  - prioridades de las excepciones del sistema
- Definición del esquema de prioridades y grupos de prioridades
- Power management (control de consumo de energía)

SysTick

- contador descendente auto-recargable de 24-bits.

Permite contar con una base de tiempo portable de muy simple utilización

El acceso a sus registros puede hacerse solamente en un nivel privilegiado, lo que lo hace ideal para utilizarlo como base de tiempo para un RTOS.

El clock empleado puede obtenerse de uno de los clocks del core, o de una referencia, dependiendo de la implementación particular.

Excepciones, interrupciones

En ARMV7-M las interrupciones son separadas y anidadas gracias al SCB y al NVIC

La dirección del handler para atender una excepción se obtiene leyendo una posición (asociada con la excepción) dentro del vector de excepciones.

Modelo de manejo de excepciones

Ante una excepción, el procesador suspende su actividad y guarda los registros R0 a R3, R12, PC, PSR y LR en el stack

Formato: especificación de AAPCS (PROCEDURE CALL STANDARD FOR THE ARM ARCHITECTURE) dentro de EABI (EMBEDDED APPLICATION BINARY INTERFACE)

Al momento de ingresar al exception handler, el LR se carga con un valor especial que no corresponde a una posición válida en el mapa de memoria; de modo que ante una instrucción de retorno de subrutina el procesador identifica que se trata en realidad de un retorno de excepción y procede a recuperar del stack los registros salvados.

Un exception handler, entonces, es una función más en C o C++, dado que no requiere de ningún manejo especial de registros del procesador.

Excepciones del sistema

RESET

la más alta prioridad (-3), fija y permanente

NMI

de prioridad siguiente al reset (-2), también fija y permanente,

HARDFAULT

excepción genérica para atender cualquier otro tipo de falta o un escalado de éstas. Su prioridad es -1, fija y permanente.

MEMMANAGE

error de protección de memoria, en sistemas con MPU.

BUSFAULT

errores de acceso a memoria para instrucciones o datos.

USAGEFAULT

respuesta a fallos relacionados con la ejecución

SVCALL

respuesta a una instrucción SVC (llamadas al OS)

DEBUGMONITOR

soporte para el debugging

Interrupciones internas

PENDSV

se genera por software y es parte del proceso de atención de una llamada al OS mediante una instrucción SVC

SYSTICK

generada por el timer del mismo nombre

Interrupciones externas

NVIC (NESTED VECTORED INTERRUPT CONTROLLER)

Modelo de manejo de prioridades

Cada excepción tiene un nivel de prioridad, y sólo aquéllas de mayor prioridad pueden interrumpir a otra que esté siendo atendida. Los números más bajos tienen precedencia, es decir, una prioridad más alta.

Las excepciones RESET, NMI y HARDFAULT tienen las prioridades más altas, fijas e inamovibles, -3, -2 y -1 respectivamente.

El resto de las excepciones tiene su prioridad configurable por software mediante registros en el SCB o el NVIC, comenzando con el valor 0 (la prioridad más alta).

Excepciones, interrupciones, *detalle*

Ante una excepción, el procesador suspende su actividad y realiza la siguiente serie de operaciones:

1. Guardar los registros R0 a R3, R12, LR, PSR y PC en el stack. Esto ocupa el bus SYSTEM
2. Leer el vector de interrupciones en la posición indicada por la dirección base de éste y el número de excepción, mediante el bus ICODE. Dado que los pasos “1” y “2” ocurren en espacios separados, *se realizan simultáneamente*.
3. Actualizar el PC
4. Llenar la pipeline, también en paralelo con el paso “1”, al ocurrir los accesos a memoria en buses separados.
5. Colocar un valor especial: EXC_RETURN en el LR.

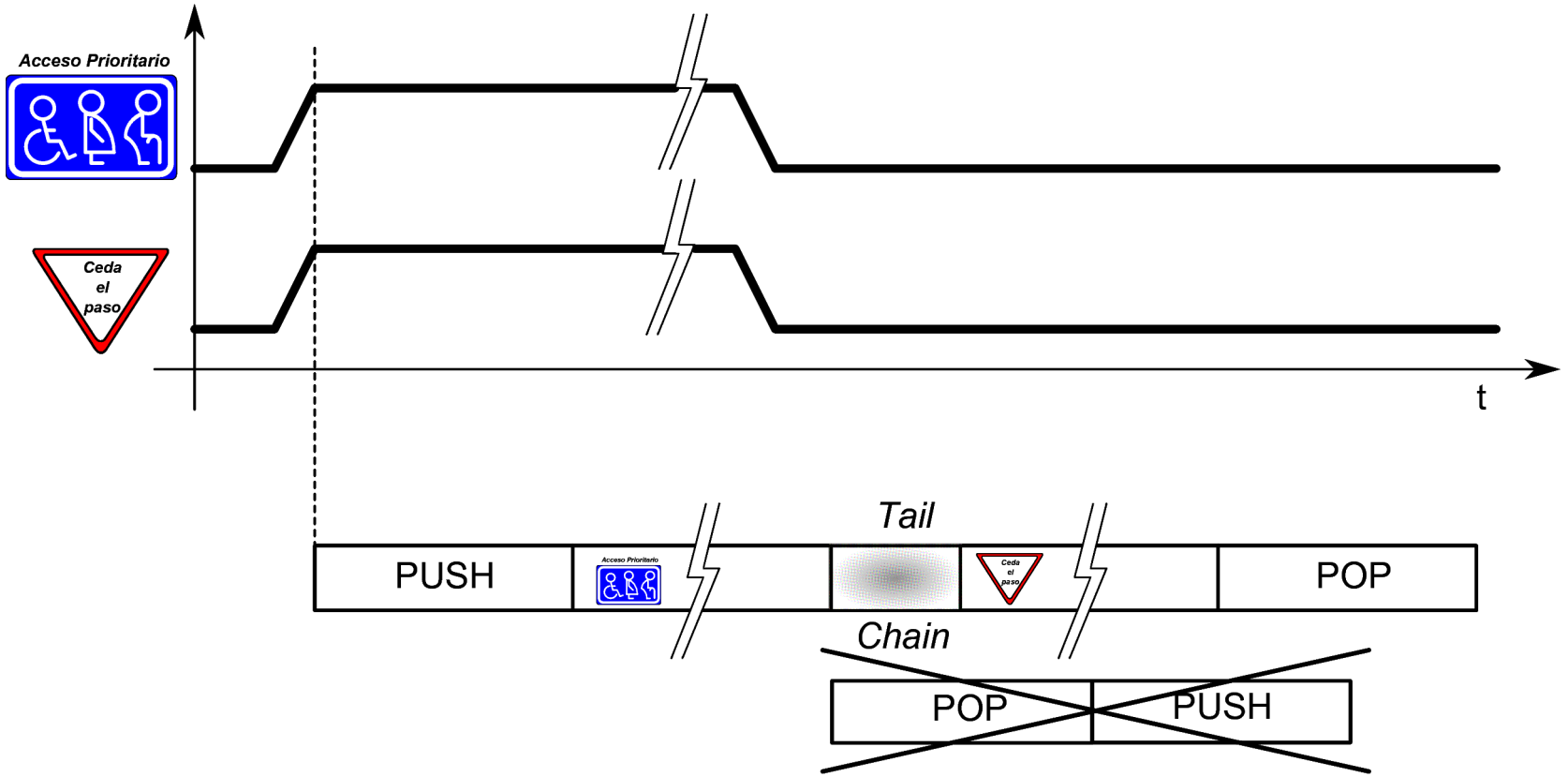
El proceso de atención de una excepción, es decir, desde que se la detecta hasta que se comienza a procesarla, demora unos 12 ciclos de clock.

Terminado el procesamiento de la excepción, el procesador se encontrará con una instrucción de retorno de función, por lo que copiará el contenido de LR al PC. El valor especial cargado al iniciar la excepción en el LR, al ser visto en el PC se lee como "esto es un retorno de excepción", por lo que el procesador inicia la secuencia correspondiente, que consta de los siguientes pasos:

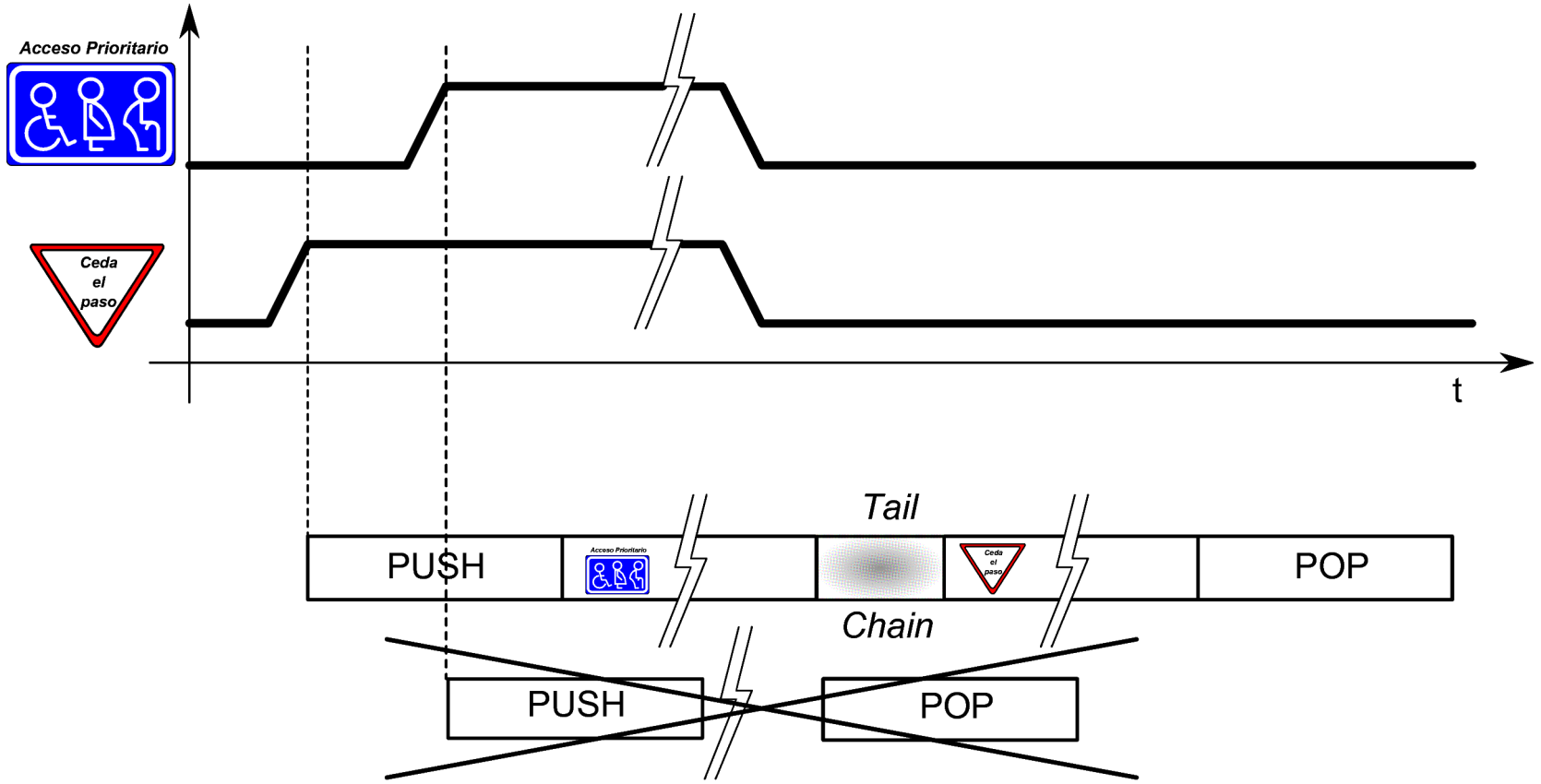
1. Recuperar los ocho registros del stack
2. Observar si debe volverse a otra excepción o al programa principal
3. Elegir el stack correspondiente y actualizar el SP. Si retorna a una excepción, utilizará MSP. Si retorna al programa principal (modo THREAD), será el MSP o el PSP según cómo esté configurado.

Algunas de las acciones enumeradas (en ambos casos) pueden ser evitadas o reiniciadas

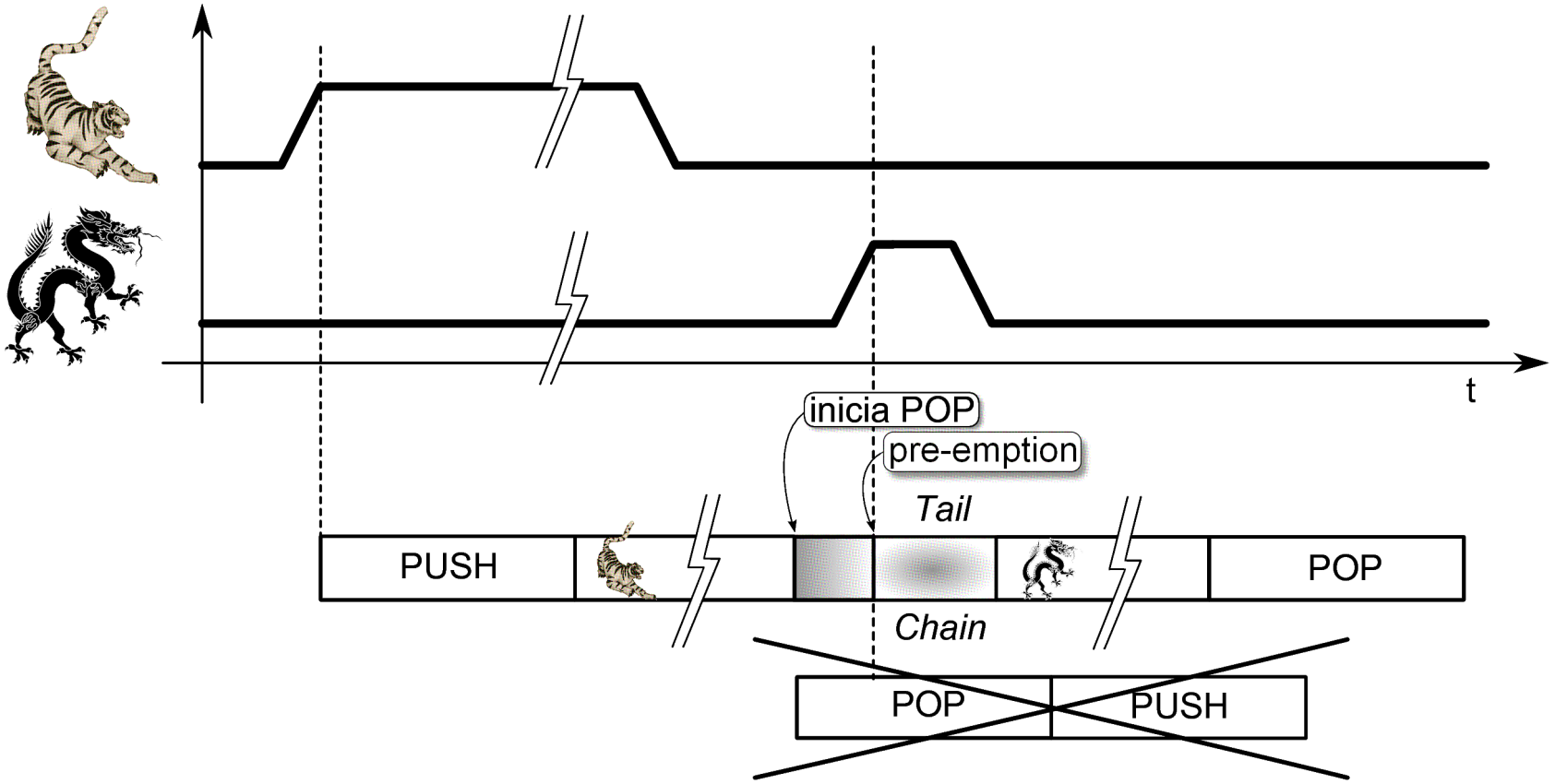
Tail chaining (encadenado a la cola)



Late arrival (arribo tardío)



POP pre-emption



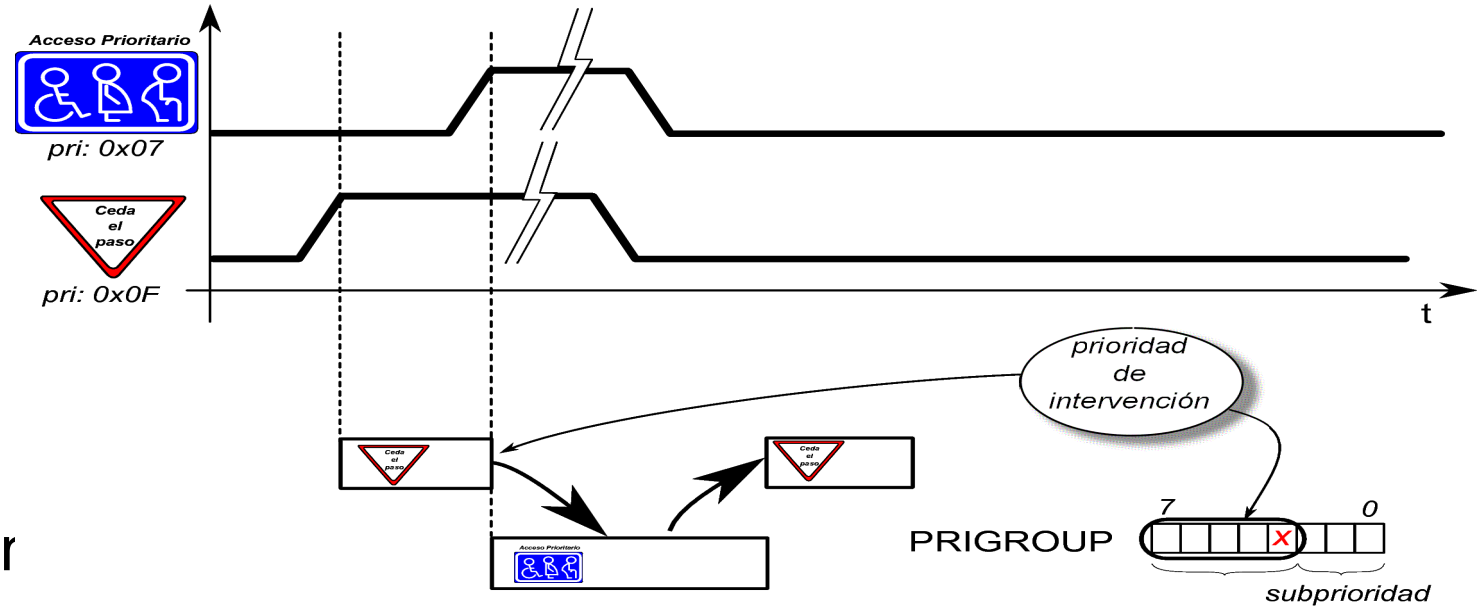
Control de prioridades

8 a 256 niveles de prioridad

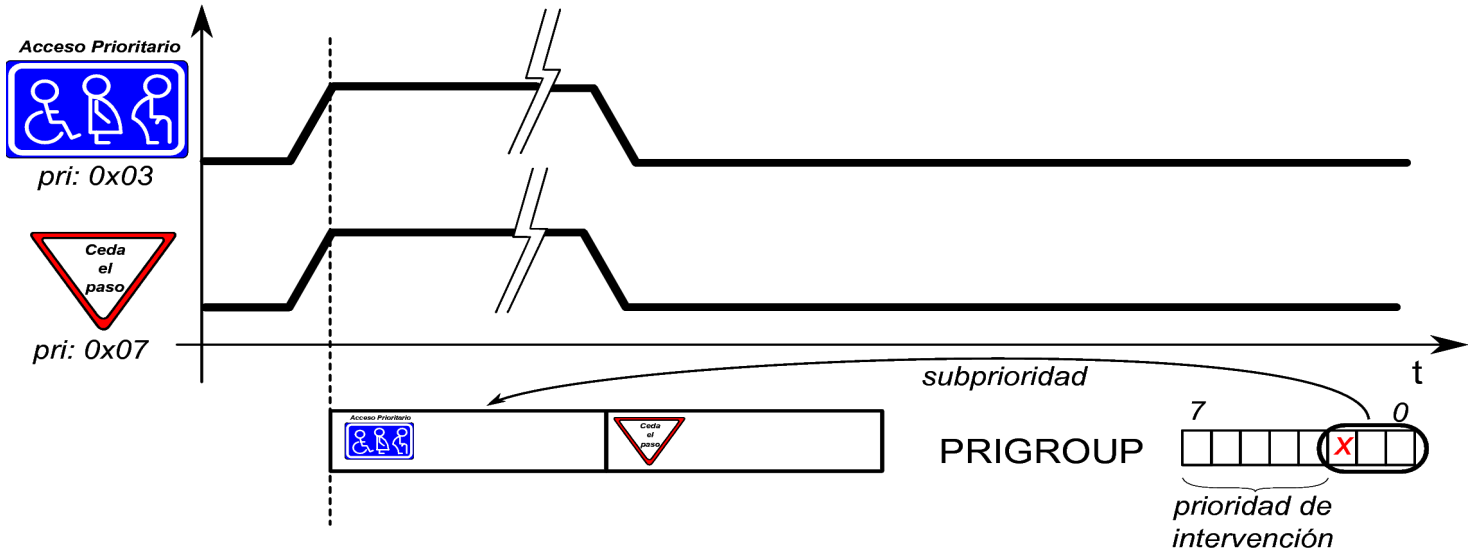
AIRCR (APPLICATION INTERRUPT AND RESET CONTROL REGISTER)

- campo PRIGROUP
  - nivel de prioridad de intervención (*pre-emption priority*)
  - subprioridad (*subpriority*)

Prioridad de ir

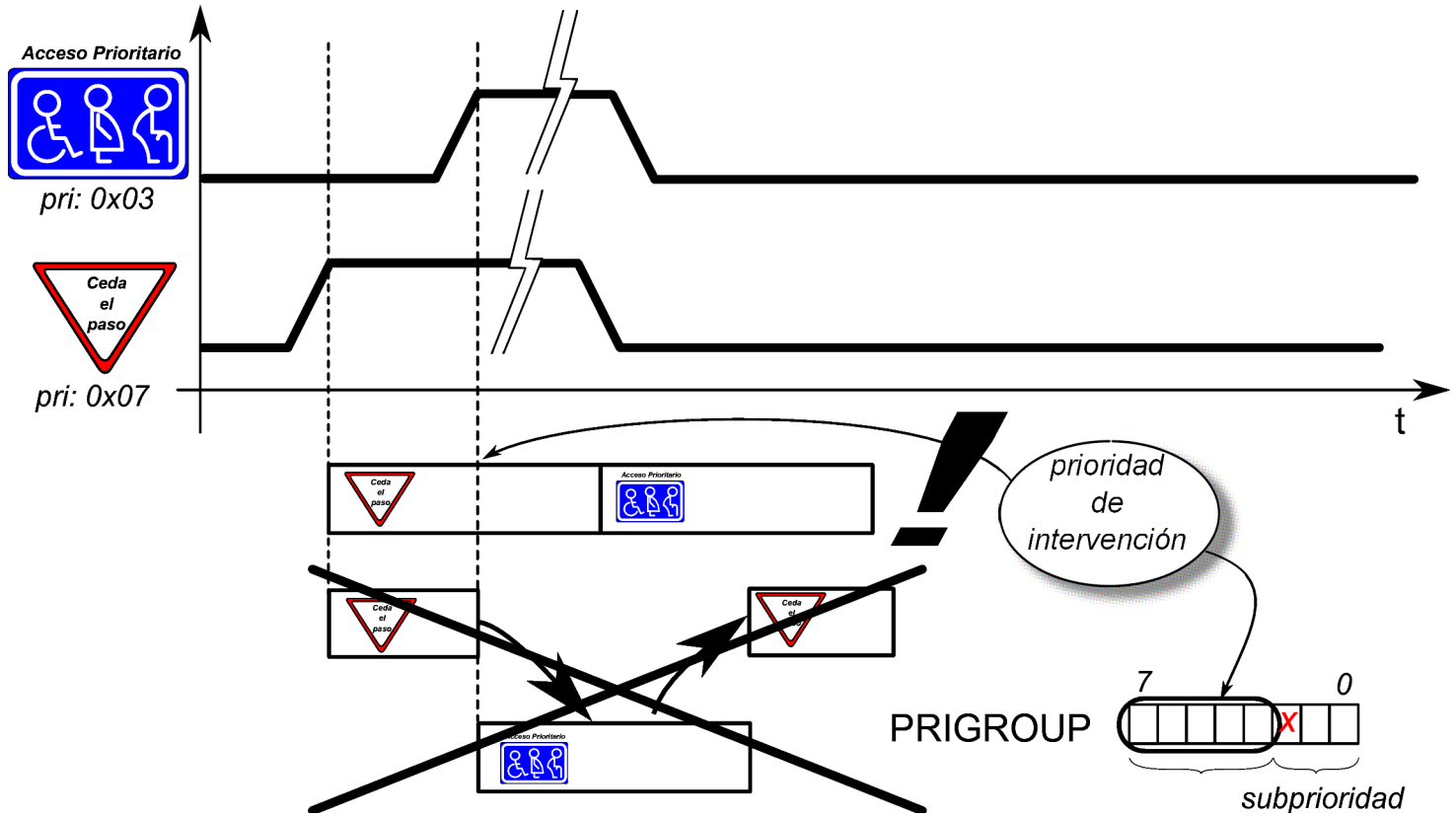


Subprioridad



CUIDADO COMO ASIGNAMOS LAS PRIORIDADES

En el ejemplo (mal hecho): Si bien la prioridad de la excepción crítica es superior, ambas pertenecen al mismo grupo de prioridad de intervención y la excepción crítica no interrumpe a la otra excepción en curso. Sí se la atiende primero en caso de estar las dos pendientes



Power Management (Sleep)

WFE (WAIT FOR EVENT)

WFI (WAIT FOR INTERRUPT)

El procesador suspende su clock y permanece en estado latente hasta tanto requiera reanudar su actividad

La arquitectura prevé tres opciones:

DORMIR-AHORA (SLEEP-NOW)

el procesador ingresa inmediatamente en modo bajo consumo

DORMIR-AL-SALIR (SLEEP-ON-EXIT)

el procesador ingresará en modo bajo consumo al terminar de atender la última excepción pendiente. Esto se realiza sin extraer los registros de la pila, por lo que el procesador queda en un estado de atender casi inmediatamente cualquier otra excepción.

SUEÑO-PROFUNDO (DEEP-SLEEP)

funciona de forma concurrente con las opciones anteriores, en cuanto coloca al procesador en un estado de bajo consumo del que tardará un tiempo mayor en volver.

clock gating

Exactamente qué sucede cuando se ingresa en uno de estos modos, depende de la implementación particular del micro por parte del fabricante

El procesador posee dos clocks: HCLK y FCLK

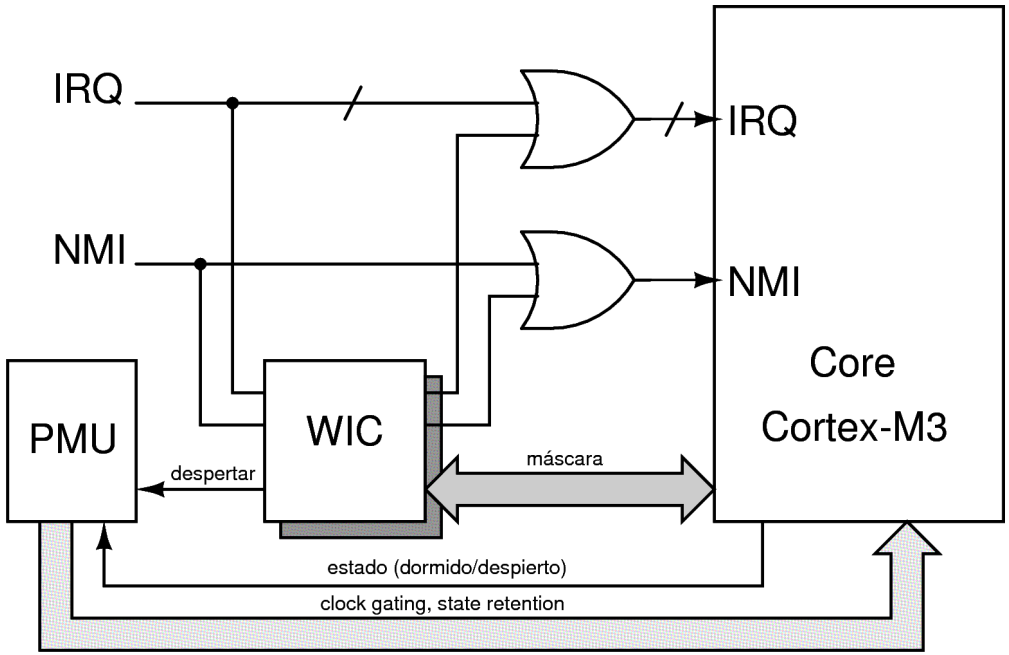
Deben mantenerse el SCB y NVIC clockeados para poder detectar las excepciones y salir del modo bajo consumo.

Esto es resuelto por un bloque adicional introducido en la revisión r2: el WIC.

El timer SYSTICK se encuentra en el SCS. Tiene la opción de recibir el FCLK o una referencia externa. Esto es a criterio del fabricante del chip, por lo que es muy común que la única opción de clock del SYSTICK sea el FCLK. *Esto significa que este timer dejará de contar si se suspende este clock*

WIC (WAKEUP INTERRUPT CONTROLLER)

- bloque adicional y opcional introducido en la revisión r2
- copia funciones de SCB+NVIC de forma rudimentaria



tecnología SRPG (STATE RETENTION POWER GATING)

PMU (POWER MANAGEMENT UNIT)

Debugging

JTAG

SWD, SWV (SERIAL WIRE DEBUG, SERIAL WIRE VIEW)

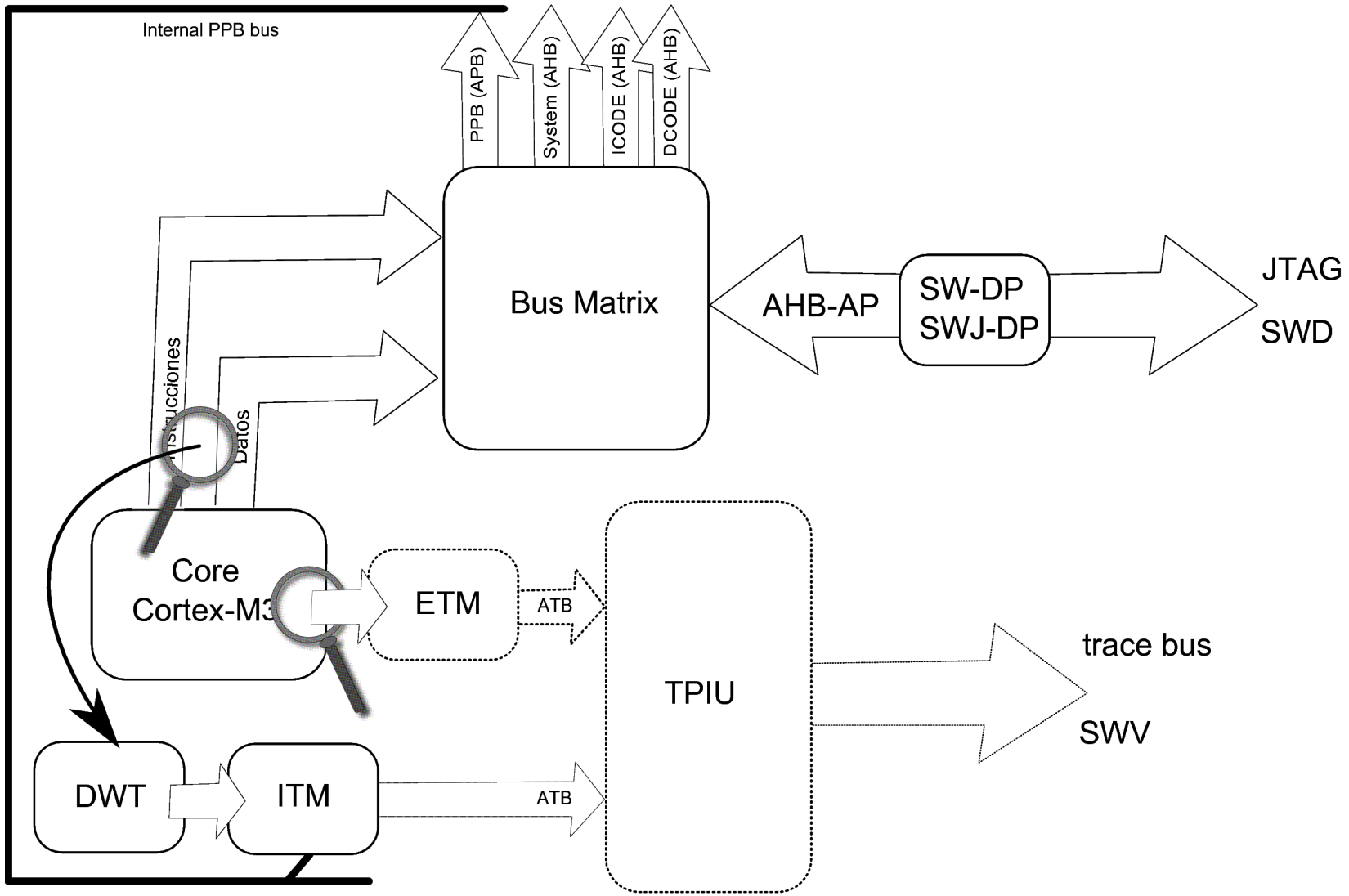
EMBEDDEDICE

ETM (EMBEDDED TRACE MACROCELL)

DP (DEBUG PORT)

SW-DP
un DP conteniendo sólo SWD/SWV

SWJ-DP
un DP combinado JTAG y SWD/SWV


Internal PPB bus
PPB (APB)
System (AHB)
ICODE (AHB)
DCODE (AHB)
Bus Matrix
AHB-AP
SW-DP
SWJ-DP
JTAG
SWD
Instrucciones
Datos
Core
Cortex-M3
ETM
ATB
TPIU
trace bus
SWV
DWT
ITM
ATB

Herramientas de desarrollo

IDEs

El entorno de desarrollo en un Cortex-M3 no es una elección del fabricante, sino del desarrollador

(Si bien existen algunos casos particulares que son todo lo contrario, siempre existe la posibilidad de recurrir a un proveedor no atado con el fabricante o incluso gratuito o hasta Open Source)

- **Keil** "una empresa del grupo ARM"
- **IAR** Embedded Workbench for ARM (EWARM)
- **CooCox**
  - basado en Eclipse
- **GNU**

Programador-Debugger

- conector de .1” con 20 pines, compatible JTAG.
- standard familia Cortex: 10 pines de paso .05”.

Algunos micros poseen ambas interfaces: JTAG y SWD.

- Segger J-Link funciona con IAR, Keil, CooCox
- Keil uLink funciona con IAR, Keil, CooCox
- otros fabricantes de JTAG
- del fabricante
- embedded en los devkits *

Alternativas Open Source

- Colink-EX funciona bajo CooCox, drivers para IAR y Keil

Trace ? Requiere ETM en micro y JTAG c/trace, CARO

CMSIS

- Archivos generales para soportar core y compilador
  - *core_cm3.h*
- Archivos particulares para soportar el micro
  - *system_<micro>.c*
    - contiene *SystemInit()*, rutina que se encarga de inicializar el clock
  - *startup_<micro>.s* o *startup_<micro>.c*
    - contiene vectores de interrupción y reset. Este último llama a *SystemInit()* y luego salta a ejecutar el código del usuario en *main()*.
- *<micro>.h*

Sergio R. Caprile

Ing. en Electrónica, UTN FRA

LDIR R&D

Senior R&D Engineer

Cika Electrónica S.R.L.

http://www.cika.com

http://www.scaprile.ldir.com.ar