Arquitectura de Microcontroladores

Carrera de Especialización en Sistemas Embebidos

Clase 1: Introducción a ARM - Cortex M

Contenidos de la clase

Sobre el curso:

1. Presentación de los docentes.
2. Alcance.
3. Reglamento.
4. Contenidos.
5. Bibliografía recomendada.

Sobre la clase:

6. Introducción a los perfiles de ARM y comparación entre familias
7. Arquitectura Cortex-M.
8. Características de los M3/M4

1. Presentación de los docentes

Seremos 2 docentes:

- Esp. Bioing. Denis Genero
  - Mail: denisjorgegenero@gmail.com
- Mg. Lic. Santiago Germino
  - Mail: sgermino@fi.uba.ar

¿Qué hay de ustedes? ¿Con qué placas cuentan?

2. Alcance

En este curso nos vamos a cubrir los siguientes aspectos:

- Comprender la arquitectura de los Cortex-M (ARMv7):
  - Modelo de programador para estas arquitecturas
  - Set de instrucciones
  - Introducción a la programación en Assembly
  - Introducción al uso de instrucciones específicas de la arquitectura (SIMD)

3. Reglamento

Para poder aprobar el curso se debe cumplir con 3 requisitos:

1. Asistencia ≥ 75% (Asistir a 6 de 8 clases) [A]
2. Realizar un examen **obligatorio** teórico-práctico (se realiza en la clase 8) [B]
3. Responder el cuestionario propuesto por la cátedra [C]

Cuestiones a tener en cuenta…

3. Reglamento

[A] En caso de no cumplir, se deberá hablar con las autoridades del posgrado.

[B] En caso **debidamente justificado** y con la **suficiente antelación**, se podrá pedir otra fecha. Se podrá **recuperar el examen**, solo en el caso de que el alumno **no alcance la nota de aprobación** de la materia (4) → Habrá penalización en este caso.

[C] El cuestionario se debe ir **resolviendo de forma progresiva** en un archivo Markdown (.md) que se irá actualizando en un repositorio de GitHub público. **No se darán puntos** si el repo registra menos de 4 commit. Entre los commit debe haber una separación de al menos 5 días.

3. Reglamento

Composición de la nota final:

$$N.F. = 0.1*P.C. + 0.2*R.C. + 0.7*N.E.$$

Donde:

- N.F. → Nota final (para aprobar NF ≥ 4)
- P.C. → Participación en clases
- R.C. → Resolución del cuestionario
- N.E. → Nota de examen

4. Contenidos

Clase	Descripción	Modalidad
1	ARMv7-M: Modelo de programación	Teórica
2	ARMv7-M: Core peripherals	Teórica
3	ARMv7-M: Instruction Set Architecture (ISA)	Teórica
4	ARMv7-M: Práctica	Práctica
5	ARMv7-M: Práctica	Práctica
6	Introducción a SIMD	Teórica
7	Práctica integradora	Práctica
8	Examen teórico-práctico	-

5. Bibliografía recomendad

Bibliografía para el curso:

- YIU, Joseph, “The Definitive Guide to ARM Cortex M3 and M4 processors”
- “ARMv7-M Architecture Reference Manual”, ARM Inc.

Habrá material complementario que se irán dando durante el cursado.

Si no hay consultas…

¡Comencemos con el curso!

6. Introducción a los perfiles de ARM

- **ARM** es una empresa que **no fabrica semiconductores**. Su modelo de negocios se basa en vender diseños de procesadores que los fabricantes compran e incorporan en sus diseños.
- Venden la propiedad intelectual y cobran regalías por unidades vendidas.
- Dependiendo de la aplicación que se quiera desarrollar, ARM ofrece distintas productos, conocidos como familias o perfiles:
  - Cortex A (Application)
  - Cortex R (Realtime)
  - Cortex M (Microcontroller)

6. Introducción a los perfiles de ARM

- **Cortex A (Application):** Son procesadores de alto **rendimiento** y están orientados a sistemas operativos embebidos de alta **performance** y alto nivel de **paralelismo**.
  - Varios núcleos (alta frecuencia)
  - “Mucha” memoria RAM
  - Memoria caché…

Nintendo switch:
- 4 Cortex A-57 @1GHz
- 4 Cortex A-53 @1,3 GHz
- 4 GB RAM LPDDR4
- Etc…

6. Introducción a los perfiles de ARM

- **Cortex R (Realtime):** Son procesadores orientados a sistemas de tiempo real donde es necesario implementar soluciones de baja **latencia**, alta **predictibilidad** y alta **capacidad de cómputo**.

  Suelen utilizarse en **sistemas críticos**. Ejemplo: sistemas del automóvil (control de tracción, frenos, etc.), dispositivos médicos críticos, industriales, etc…

¿Cuando un sistema se considera crítico?

6. Introducción a los perfiles de ARM

Memoria caché

- De “pequeño tamaño” y muy rápida, de 0 WS (Wait States).
- Hecha por “flip-flops”, es muy costosa, pero funciona a Fclk.
- Los datos que más frecuentemente se utilizan son almacenados allí.
- Las SRAMs, no son 0 WS, pero son mucho más económicas.

Si bien el uso de caché puede parecer muy buena, hay casos en los que no se aconseja su uso… En sistemas críticos no es recomendable, ya que afectan al **determinismo** (no se tiene certeza absoluta del tiempo de respuesta).

6. Introducción a los perfiles de ARM

- **Cortex M (Microcontroller):** Son procesadores orientados a dispositivos de consumo masivo y sistemas embebidos compactos. Están diseñados para alta densidad de código y ser **programados en C**.

  Los más conocidos son:

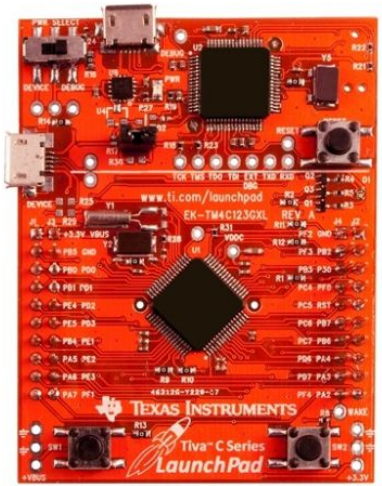
  ➢ **Cortex M0/M0+:** Pensados para una implementación mínima, de bajo consumo y bajo costo.
  ➢ **Cortex M3/M4/M7:** Agregan mayor performance, más funcionalidades (división por hardware), FPU, MPU, etc…

7. Arquitectura Cortex-M

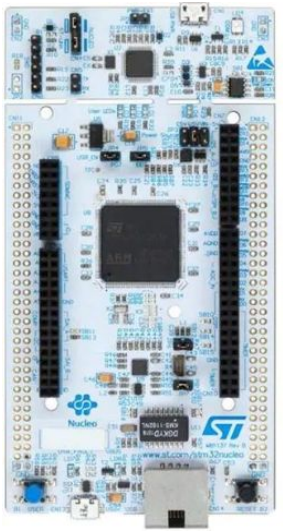
- **Cortex M (Microcontroller):** Ejemplos…

Raspberry pi pico
Cortex M0+

STM32 - Blue Pill
Cortex M3



Tiva C123 -LaunchPad
Cortex M4F



STM32 - Nucleo F676ZI
Cortex M7

7. Arquitectura Cortex-M

Comparación entre algunas arquitecturas:

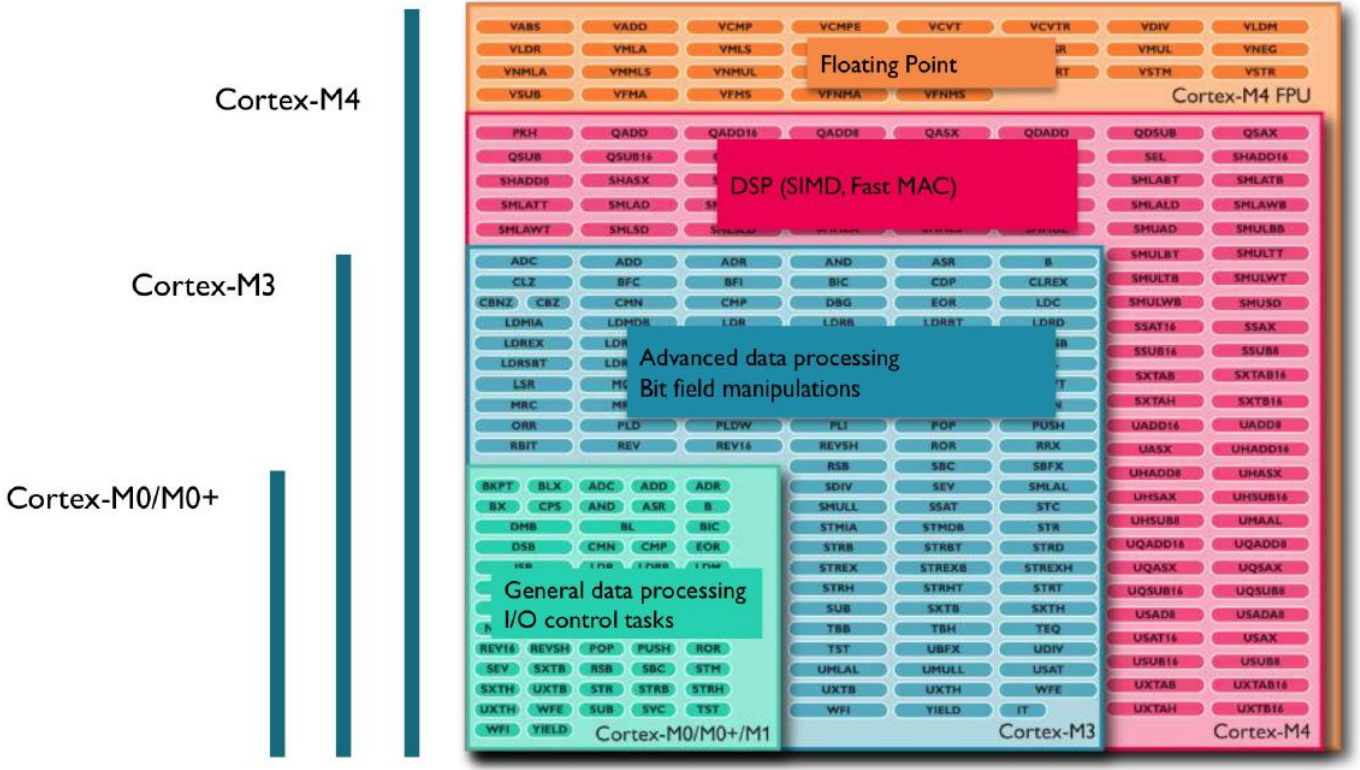
ARM Cortex-M	SysTick Timer	Bit-banding	Memory Protection Unit (MPU)	Tightly-Coupled Memory (TCM)	CPU cache	Memory architecture	ARM architecture
Cortex-M0[1]	Optional*	Optional[9]	No	No	No[10]	Von Neumann	ARMv6-M
Cortex-M0+[2]	Optional*	Optional[9]	Optional (8)	No	No	Von Neumann	ARMv6-M
Cortex-M3[4]	Yes	Optional*	Optional (8)	No	No	Harvard	ARMv7-M
Cortex-M4[5]	Yes	Optional*	Optional (8)	No	No	Harvard	ARMv7E-M
Cortex-M7	Yes	No	Optional (8 or 16)	Optional	Optional	Harvard	ARMv7E-M

7. Arquitectura Cortex-M

Comparación entre algunas arquitecturas:

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

7. Arquitectura Cortex-M



7. Arquitectura Cortex-M

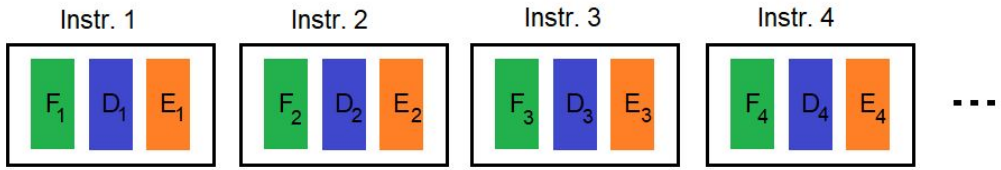
Instructions	Instruction size	Cortex M0	Cortex M0+	Cortex M1	Cortex M3	Cortex M4	Cortex M7
ADC, ADD, ADR, AND, ASR, B, BIC, BKPT, BLX, BX, CMN, CMP, CPS, EOR, LDM, LDR, LDRB, LDRH, LDRSB, LDRSH, LSL, LSR, MOV, MUL, MVN, NOP, ORR, POP, PUSH, REV, REV16, REVSH, ROR, RSB, SBC, SEV, STM, STMIA, STR, STRB,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

8. Características de los M3/M4

- RISC (Reduced Instruction Set Computers).
- Arquitectura Harvard.
- Arquitectura de 32 bits: tanto los registros como los buses de datos son de 32 bits.
- Manejo eficiente de datos de 8/16 bits.
- Pipeline de 3 etapas (fetch, decode, execute) → ¿Que es un pipeline?

8. Características de los M3/M4

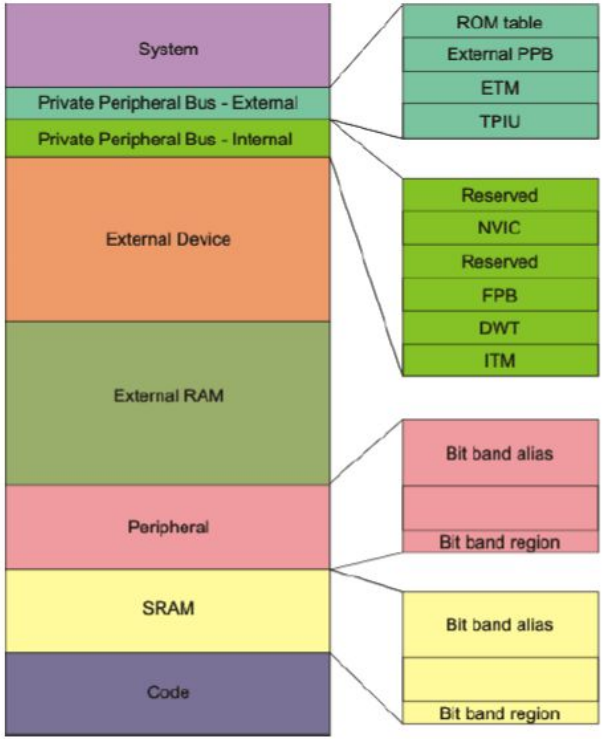
Un pipeline de 3 etapas funciona de la siguiente manera:



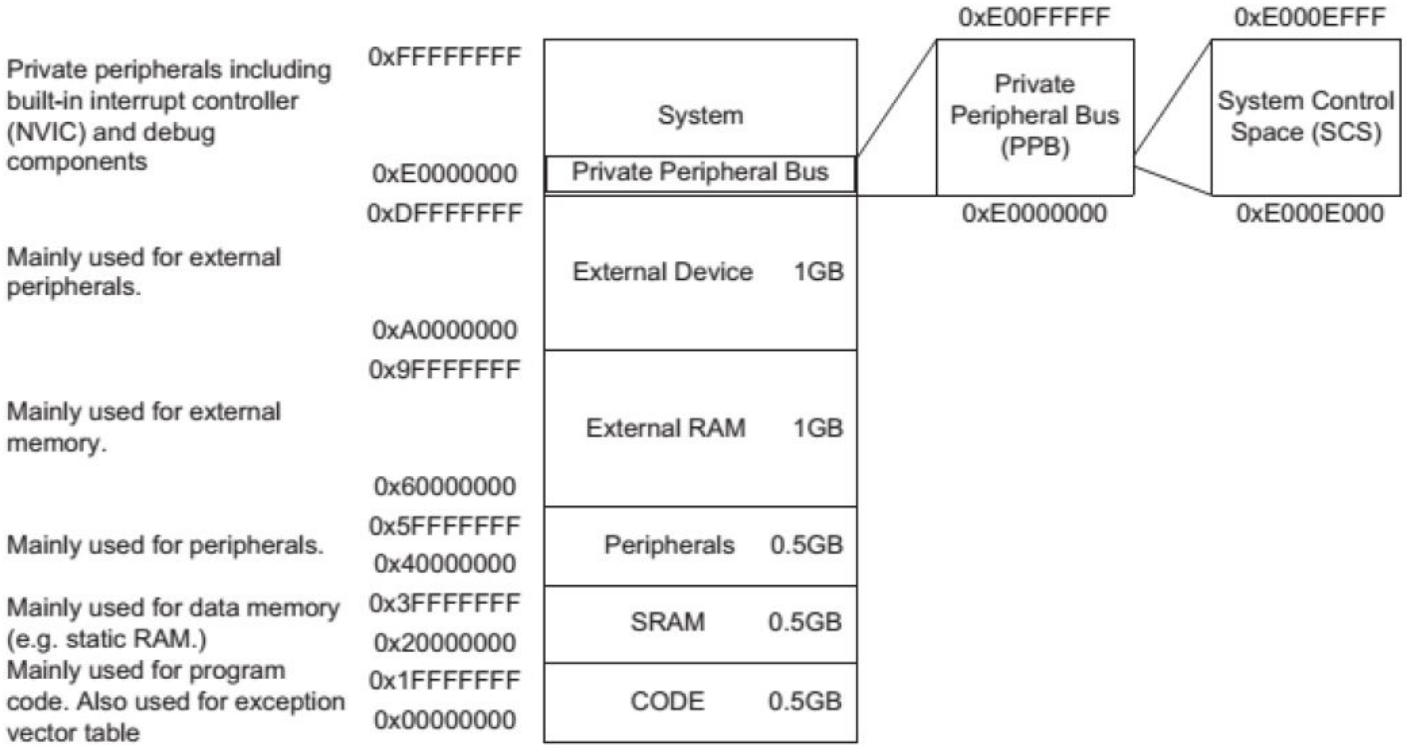
Clock	Fetch	Decode	Execute
1	F1	-	-
2	F2	D1	-
3	F3	D2	E1
4	F4	D3	E2

8. Características de los M3/M4

- Mapa de memoria plano de 4 GB
- Arquitectura Load-Store
- Set de instrucciones mixto (16/32 bits) → Thumb y Thumb2
- Alta densidad de código
- Características de bajo consumo
- NVIC (Nested Vectored Interrupt Controller)



8. Características de los M3/M4



8. Características de los M3/M4

- Soporte para SO embebido
  - MPU → Protección de regiones de memoria.
  - Systick → Timer para base de tiempo (otorga **portabilidad** de código)
  - Banked stack pointers:
    - Main Stack Pointer (MSP)
    - Process Stack Pointer (PSP)
  - Niveles de privilegios…

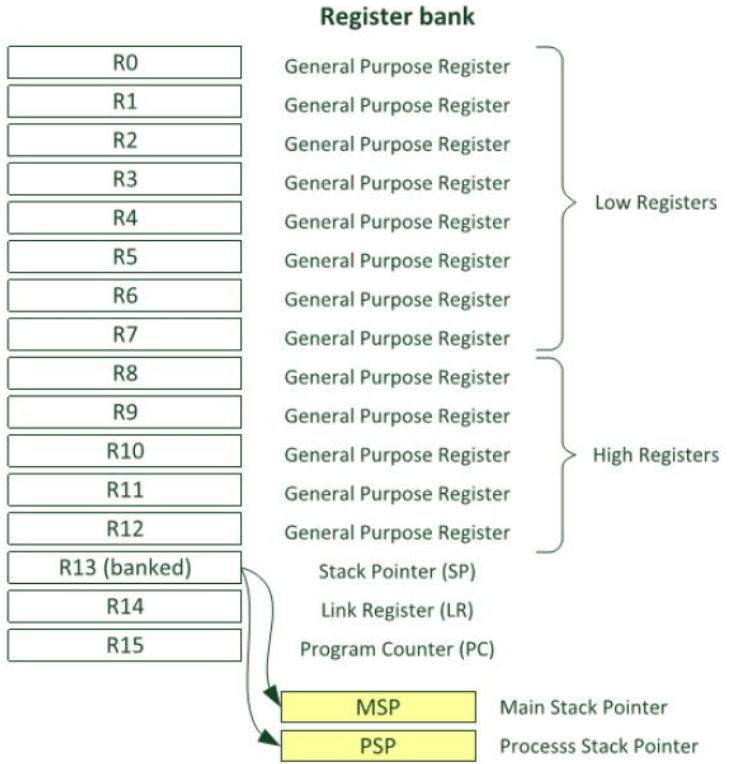
M3/M4: Registros

Disponen de registros de uso general.

¿Qué es el Stack Pointer?

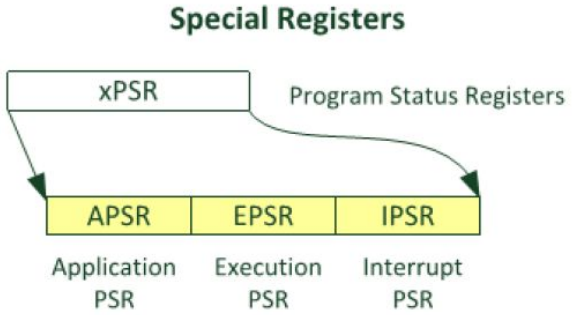
¿Para qué se utiliza el Link Register?

¿Qué función cumple el Program Counter?



M3/M4: Registros

Registros especiales



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

M3/M4: Registros

Table 4.2 Bit Fields in Program Status Registers

Bit	Description
N	Negative flag
Z	Zero flag
C	Carry (or NOT borrow) flag
V	Overflow flag
Q	Sticky saturation flag (not available in ARMv6-M)
GE[3:0]	Greater-Than or Equal flags for each byte lane (ARMv7E-M only; not available in ARMv6-M or Cortex®-M3).
ICI/IT	Interrupt-Continuable Instruction (ICI) bits, IF-THEN instruction status bit for conditional execution (not available in ARMv6-M).
T	Thumb state, always 1; trying to clear this bit will cause a fault exception.
Exception Number	Indicates which exception the processor is handling.

M3/M4: Excepciones e interrupciones

¿Qué diferencia hay entre una excepción y una interrupción?

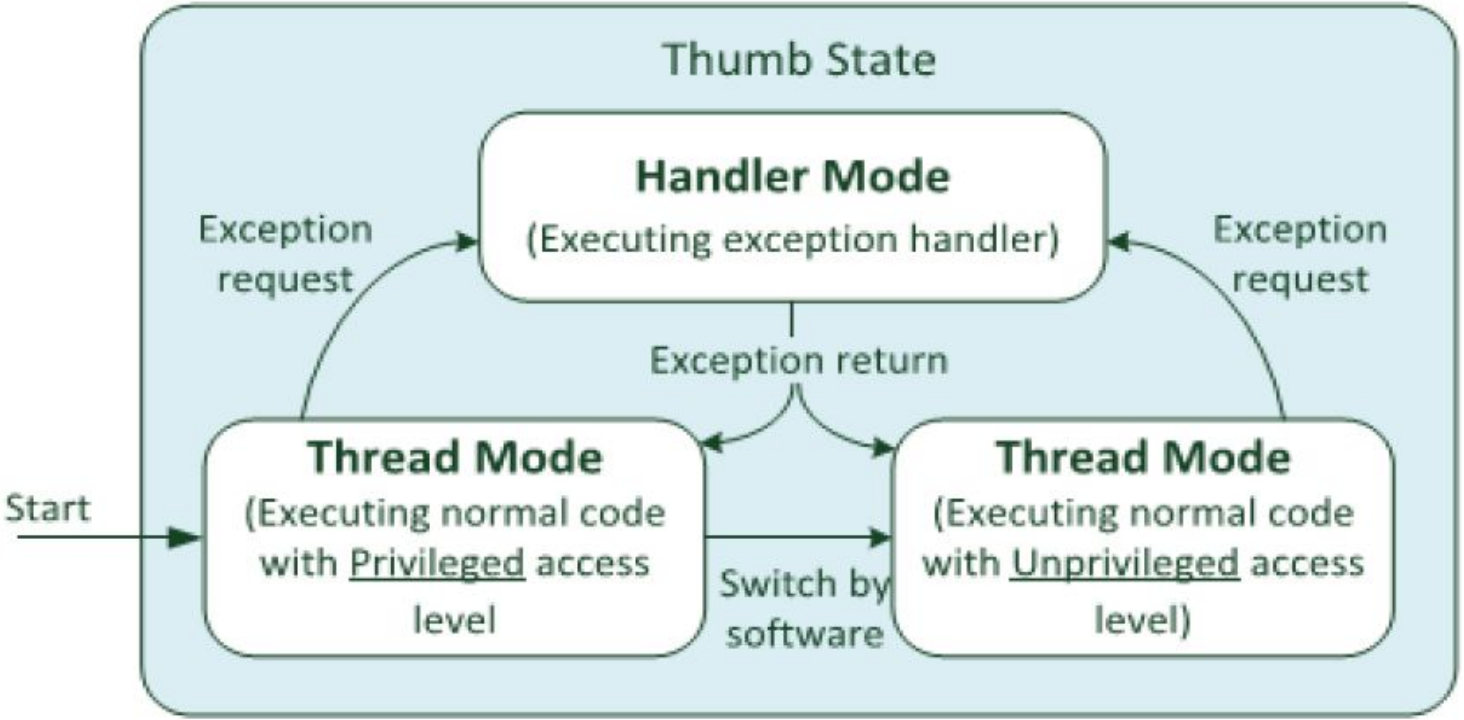
Para responder esta pregunta, aún debemos seguir indagando en la arquitectura…

M3/M4: Excepciones

Table 4.9 Exception Types

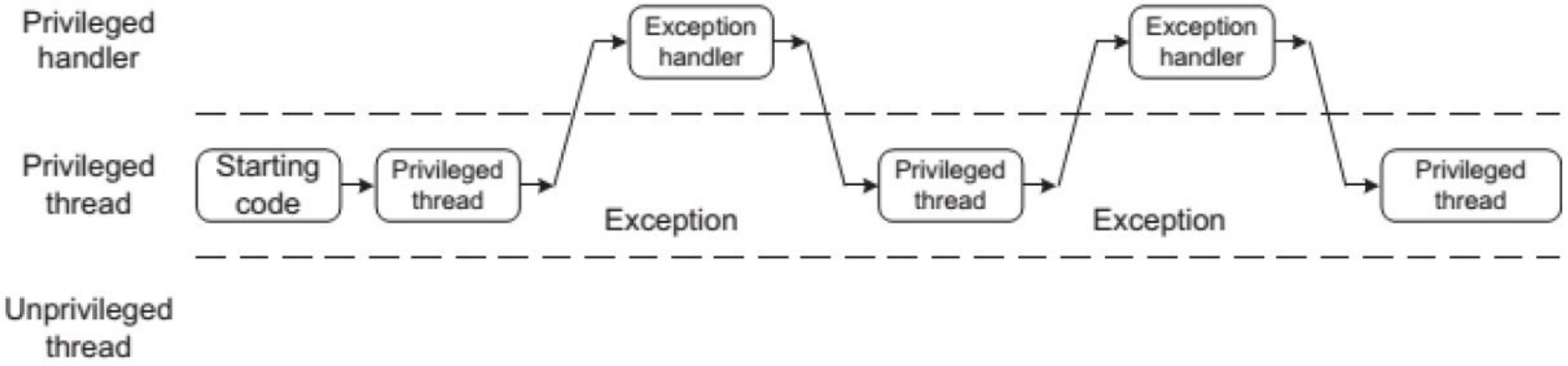
Exception Number	CMSIS Interrupt Number	Exception Type	Priority	Function
1	—	Reset	−3 (Highest)	Reset
2	−14	NMI	−2	Non-Maskable interrupt
3	−13	HardFault	−1	All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking
4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

M3/M4: Modos de operación



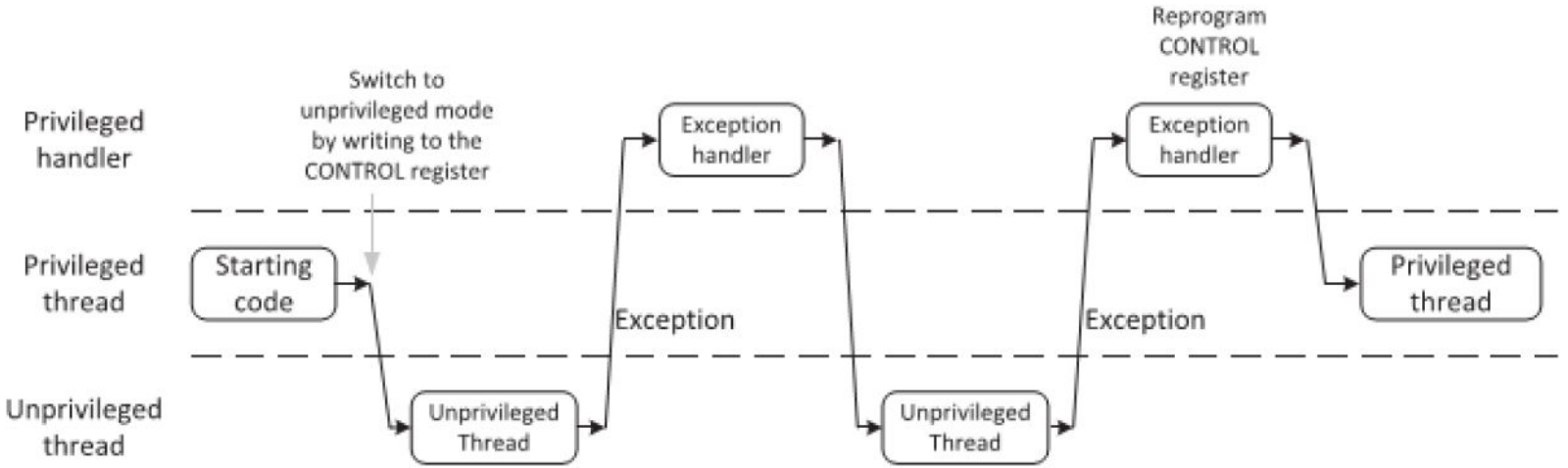
M3/M4: Context switching

En baremetal...



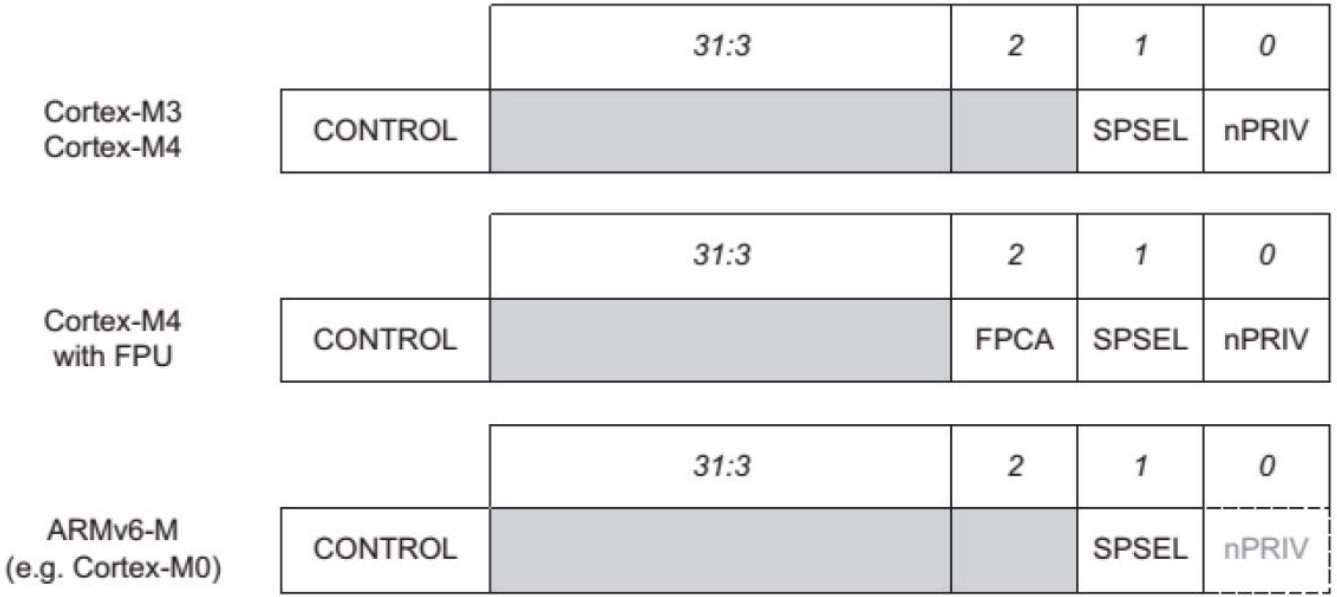
M3/M4: Context switching

Con un RTOS…




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

M3/M4: Context switching



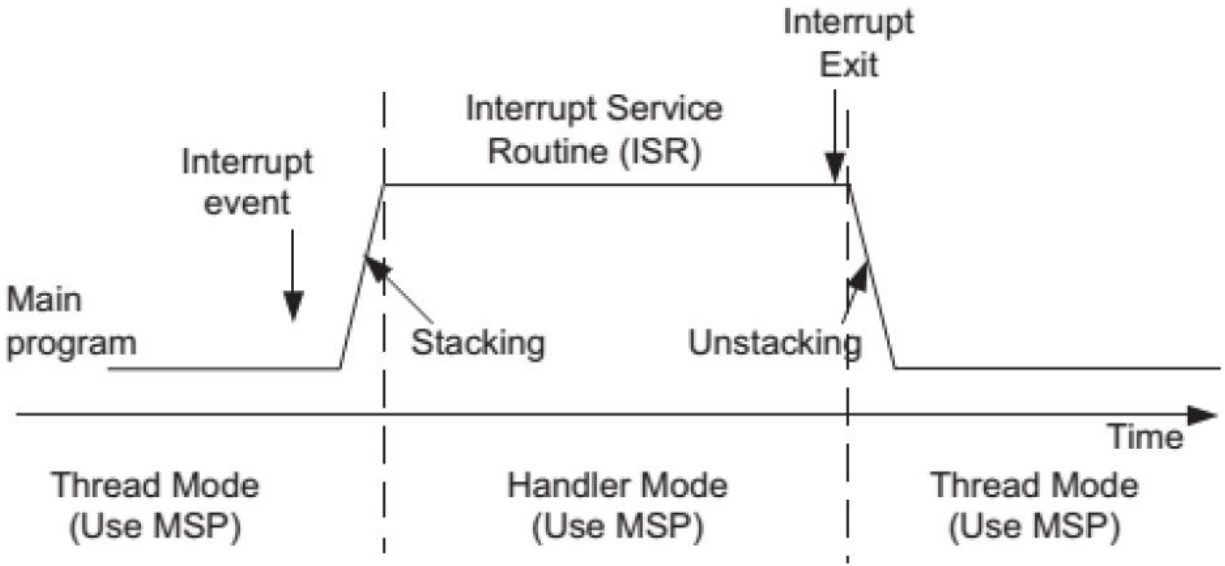
nPRIV = 0 → Privilegiado

SPSEL = 0 → MSP

CONTROL tiene un **modelo ortogonal** → Se pueden utilizar 3 de las 4 combinaciones de los bits SPSEL y nPRIV. ¿Cuál no se podría usar?

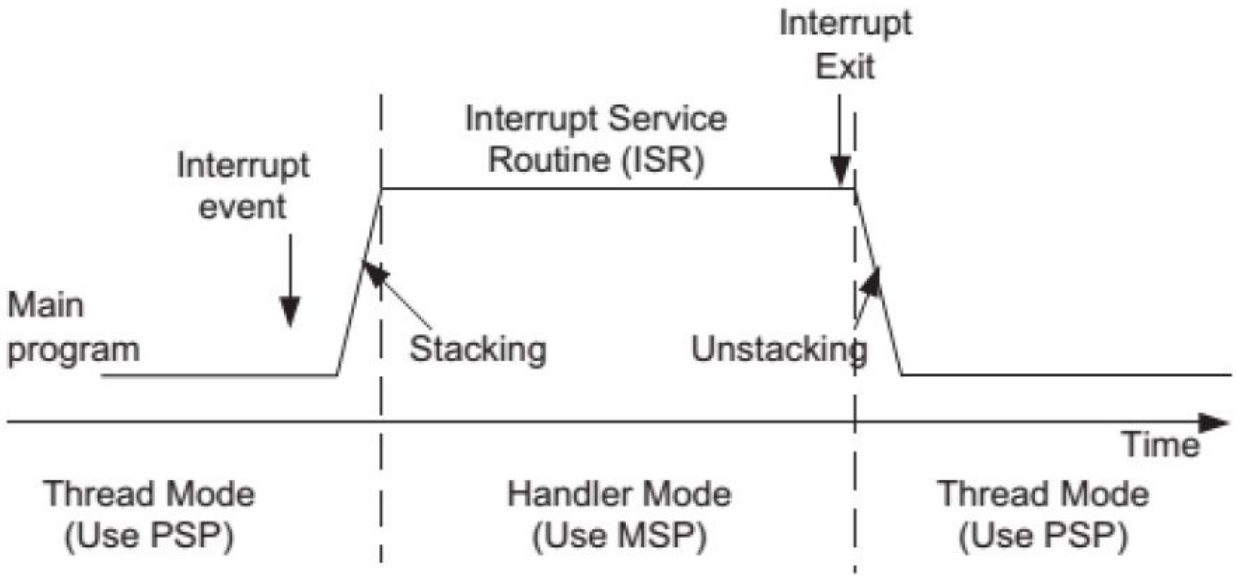
M3/M4: Context switching

En modo privilegiado, el Stack Pointer no cambia



M3/M4: Context switching

En modo no privilegiado automáticamente se cambia para separar contextos…

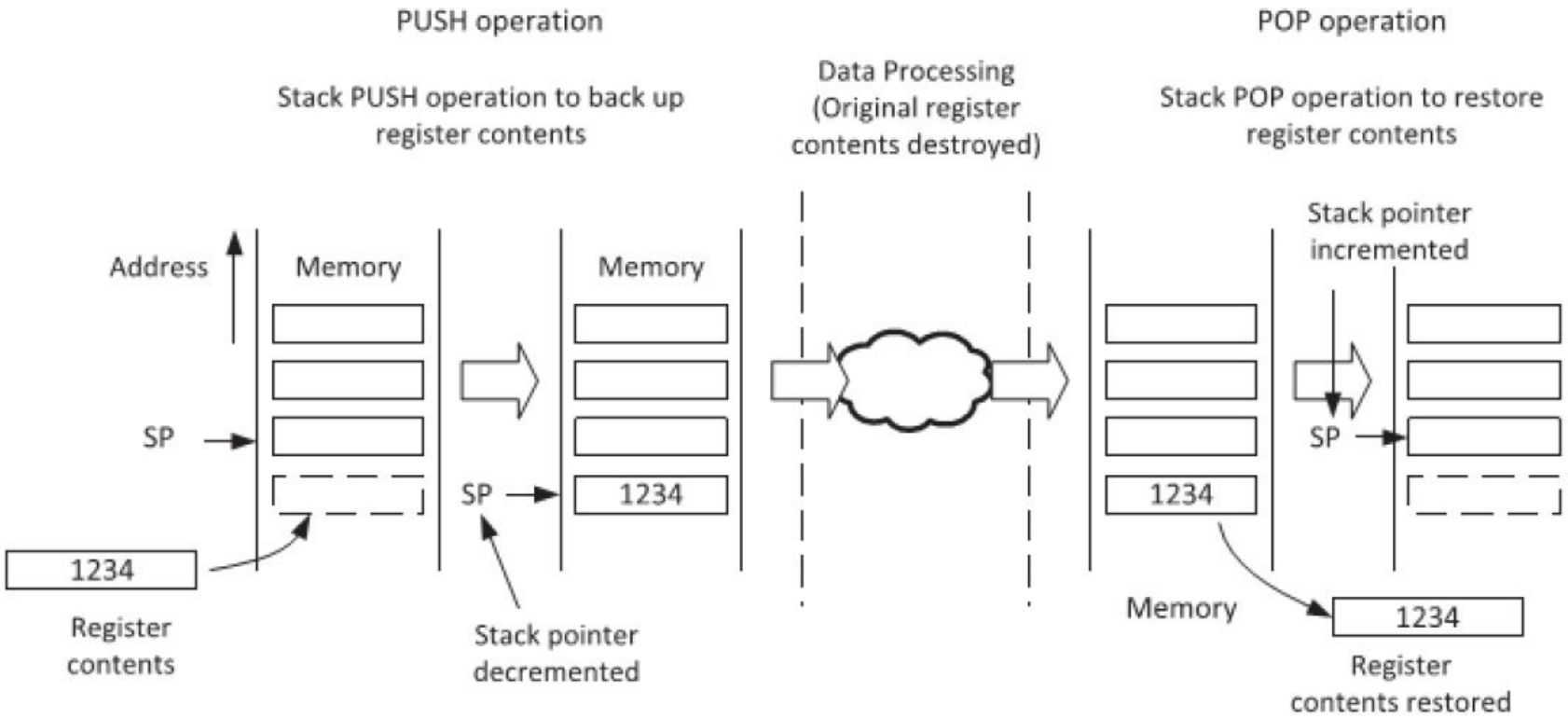


M3/M4: Stack

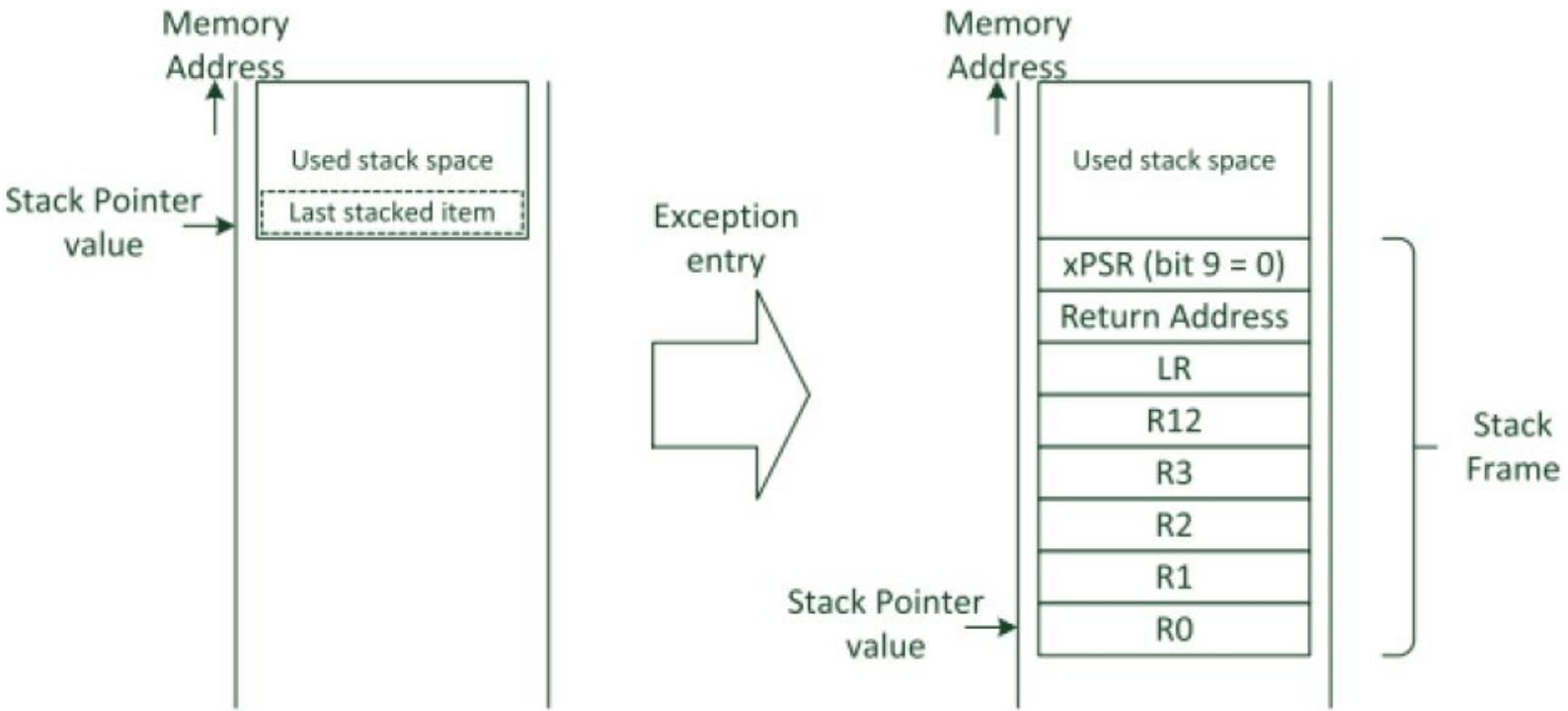
¿Qué funciones cumple el stack o pila?

- Guardar variables locales
- Pasar datos a funciones o subrutinas
- Guardar el estado del procesador y de los registros de propósito general cuando ocurre una interrupción → **stacking**

M3/M4: Stacking



M3/M4: Stacking



M3/M4: Stacking

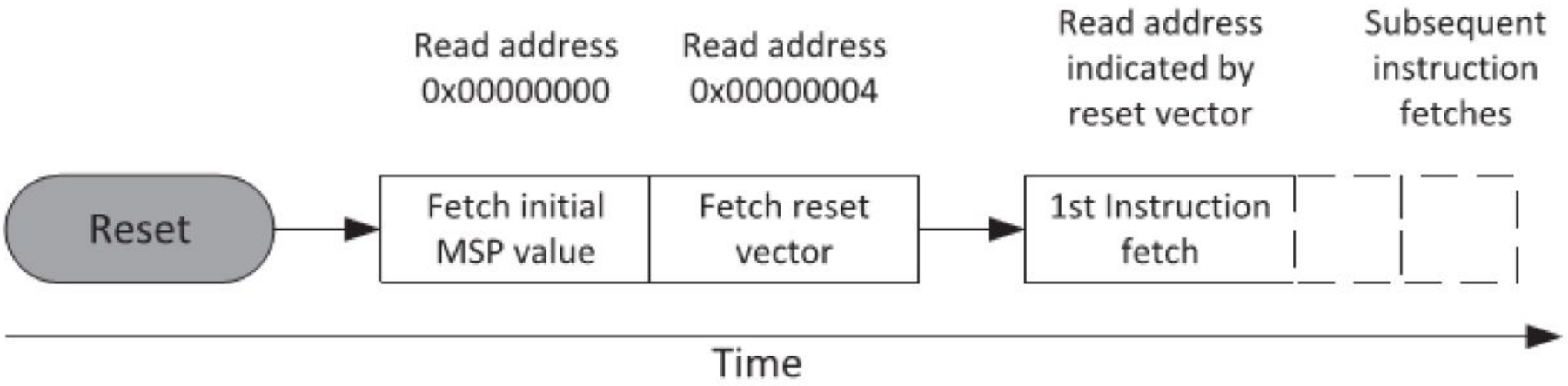
```
    ...
; R4 = X, R5 = Y, R6 = Z
BL    function1
```

Subroutine

```
function1
    PUSH    {R4-R6, LR} ; Save registers
                        ; including link register
    ... ; Executing task (R4, R5 and R6
        ; could be changed)
    POP     {R4-R6, PC} ; Restore registers and
                        ; return
```

```
; Back to main program
; R4 = X, R5 = Y, R6 = Z
... ; next instructions
```

M3/M4: Secuencia de Reset



M3/M4: Secuencia de Reset

Lo primero que se hace es inicializar el Stack Pointer y luego el Program Counter se dirige al vector de Reset. Este dirige al PC a la memoria de código y comienza la ejecución del programa