Tema 5:

Firmware y bootstraping

Programación de sistemas y dispositivos

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Contenidos

- ✓ Definición de conceptos.
- ✓ Funcionalidades del firmware.
- ✓ Firmware suites.
- ✓ Device drivers.
- ✓ Desarrollo de un bootloader elemental.

J.M. Mendías 2018

Definición de conceptos

- Firmware: software de muy bajo nivel que ofrece un interfaz entre el hardware y el software de aplicación/SO.
  - Reside en ROM y comienza a ejecutarse tras el encendido/reset del sistema.
  - El rango de funcionalidades que ofrece y el modo de acceso a ellas varía según los requisitos de la aplicación.
- Device driver: colección de rutinas software que facilitan el acceso a la funcionalidad de un dispositivo concreto de E/S
  - Configuran el dispositivo y efectúan físicamente la E/S.
- Startup code: porción de software que configura mínimamente el sistema para que pueda ejecutar una aplicación/SO.
  - Inicializa pilas, memoria y otros dispositivos básicos.
  - Cuando finaliza cede el control a la aplicación/SO.
- Bootloader: porción de software que carga y lanza la aplicación/SO en el hardware objetivo.
  - Carga en RAM la aplicación desde un soporte no-volátil, configura el sistema y cede el control a la aplicación/SO.

J.M. Mendías 2018

Firmware

funcionalidades

- Capa HAL (Hardware Abstraction Level) / BSP (Board Support Package)
  - Oculta el HW subyacente ofreciendo un conjunto homogéneo de interfaces de programación independientes de la plataforma.
- Configuración básica de la plataforma
  - Inicialización de los componentes HW esenciales para el arranque del sistema.
- Carga de la imagen de la aplicación/SO en RAM
  - Es dependiente del medio en donde esté almacenada la imagen (ROM, disco, red...),
    - Puede requerir drivers para su acceso e incluso la gestión de sistemas de ficheros.
  - Debe de tener en cuenta el formato de la imagen.
    - binario plano, ELF file (en cuyo caso hay que descifrar cabecera: dirección de ejecución, tipo, tamaño...), imagen comprimida, imagen cifrada, etc...
  - En los sistemas más simples la imagen se ejecuta directamente desde ROM.
- Cesión del control a la aplicación/SO
  - Actualizar las tablas de vectores de interrupción
    - para que apunten a las direcciones apropiadas de las ISR de la aplicación/SO
  - Actualizar el PC con la dirección de inicio de la aplicación/SO
  - Partes del firmware pueden quedar residentes (i.e drivers) para su uso por el SW.

Firmware
funcionalidades

- Identificación de la plataforma
  - En caso de que el ejecutable pueda trabajar sobre distintas plataformas, el firmware deberá descubrir el plataforma exacta sobre la que está operando.
  - Puede hacerse leyendo esa información de algún chip preprogramado, chequeando la existencia de ciertos periféricos, leyendo de ellos...
- Diagnosis de la plataforma
  - Identificación de las posibles malfunciones de los componentes HW.
- Creación de un interfaz de depuración interactivo (si procede)
  - Interfaz para la inserción de breakpoints, volcados y modificaciones de memoria, muestra de los contenidos de registros, desensamblado...
  - El envío de ordenes puede hacerse a través de un interprete de comandos o a través de un depurador corriendo en un host externo.
  - Este mismo interprete de comandos puede usarse cambiar las opciones por defecto de la aplicación / sistema operativo a cargar.
  - La comunicación se hace a través de una conexión serie, ethernet o JTAG.

Firmware suites

- ARM firmware suite
  - Soporta un variado número de plataformas y procesadores
  - µ-HAL: conjunto de device drivers de bajo nivel para operar con diferentes dispositivos usando un API estándar.
    - Inicialización del sistema.
    - Driver serie (por polling): método básico de comunicación con un host.
    - Soporte de leds: método básico de feedback.
    - Soporte de timers: interrupciones periódicas.
    - Controladores de interrupción.
    - Intérprete de comandos para monitorizar el proceso de aranque.
  - Angel: API que permite la comunicación entre la plataforma y un depurador corriendo en un host.
    - Depuración, carga de imágenes. etc.

- Red Hat Redboot
  - Soporta un amplio número de procesadores
  - Ofrece un interfaz para la depuración con GNU Debugger, una gestión de imágenes en Flash ROM y la carga y arranque de distintos SSOO empotrados.

J.M. Mendías
2018

Device drivers

consideraciones de diseño

- El diseño de un device driver es modular y por capas:
  - Un módulo de un cierto nivel solo puede llamar a módulos de su mismo nivel o del inmediatamente inferior, a través del API que le ofrecen.
  - Suelen tener, al menos, 2 niveles:
    - HW dependent: accede físicamente a los elementos del dispositivo particular.
    - Hardware independent: representa de manera homogénea dispositivos de similares características (orientados a carácter, a bloque, etc.) y resuelve problemas de instanciación, serialización y sincronización de accesos en aplicaciones multi-hebra.

- Cada módulo de un device driver está formado por 4 elementos:
  - Rutinas de inicialización (públicas): inicializan el dispositivo y todas las estructuras de datos necesarias para su gestión. Son llamadas por la aplicación una única vez a su comienzo.
  - Rutinas de E/S (públicas): permiten que la aplicación interactué con el dispositivo de una manera abstracta.
  - Estructuras de datos globales (privadas): solo tiene acceso directo a ellas el driver, la aplicación accede a ellas a través de funciones públicas.
  - Software de soporte (privado): funciones que facilitan la programación del resto. Las RTI forman parte de ello.

J.M. Mendías
2018

Bootloader elemental

consideraciones de diseño (i)

- Un bootloader toma el control del sistema tras un encendido o un reset
- Tras el reset el microcontrolador de la placa Embest S3CEV40 se encuentra en el siguiente estado:
  - El PC = 0, luego el bootloader comienza a ejecutarse desde ROM.
  - El procesador está en estado ARM y en modo SVC.
  - Las líneas IRQ y FIQ están deshabilitadas.
  - El valor del resto de registros del procesador es desconocido.
  - Los registros de los controladores están inicializados al valor indicado por el fabricante:
    - La cache está deshabilitada.
    - El sistema funciona a una frecuencia de reloj de 8 MHz.
    - El sistema desconoce la organización física del mapa de memoria.
    - El watchdog está activado y contando.
    - Todas las líneas de interrupción del controlador de interrupciones están enmascaradas.

Bootloader elemental

consideraciones de diseño (ii)

- Un bootloader elemental para la placa Embest S3CEV40:
  - Programado en ensamblador y enlazado junto a la aplicación.
  - Asume ser (junto a la aplicación que arranca) una única imagen binaria plana ubicada al comienzo de la ROM.
  - Realiza una configuración mínima del microcontrolador delegando en la aplicación la inicialización completa del sistema y el feedback del proceso según sus necesidades.
    - Inicializa la tabla de vectores de excepción/interrupción.
    - Deshabilita el watchdog.
    - Configura el gestor de reloj.
    - Configura el controlador de memoria.
    - Inicializa los SP asociados a cada uno de los modos de ejecución privilegiados.
    - Habilita las líneas IRQ y FIQ del procesador.
  - Copia en RAM la imagen completa (bootloader + aplicación).
  - Cede el control a la aplicación (desarrollada en C).

Bootloader elemental

declaraciones

```
.include "s3c44b0x.asm"
.include "s3cev40.asm"
```
nemotécnicos de los registros de los controladores de dispositivos

```
.equ USRMODE, 0x10
.equ FIQMODE, 0x11
.equ IRQMODE, 0x12
.equ SVCMODE, 0x13
.equ ABTMODE, 0x17
.equ UNDMODE, 0x1b
.equ SYSMODE, 0x1f
```
nemotécnicos de los modos de operación del ARM7TDMI

```
.extern isrUNDEF
.extern isrSWI
.extern isrPABORT
.extern isrDABORT
...
.extern isrRTC
.extern isrADC
```
nombres de las RTE/RTI definidas en la aplicación C

```
.extern _image_start
.extern _image_end
```
dirección de comienzo/fin de la imagen (boot+app), definidas en el linker script

```
.extern main
```
punto de entrada a la aplicación a la que el bootloader cede el control

Bootloader elemental

vectores de excepción/interrupción

```
.section .text ..... comienza el código (sección .text)

0x0000  b bootloader ..... tras reset salta al comienzo del programa de arranque
        b isrUNDEF
        b isrSWI
        b isrPABORT
        b isrDABORT      } salta a las RTE
        nop ..... vector reservado
        b isrIRQ ..... solo aplica si las IRQ no están vectorizadas
0x001c  b isrFIQ ..... salta a la RTI-FIQ

0x0020  b isrUSB         } salta a las RTI externas
        b isrKEYPAD      } (solo aplica si las IRQ están vectorizadas)
        b isrTS
        b isrETHERNET
        b isrPB
        b isrTICK
        nop
        nop
        b isrZDMA0       } salta a las RTI internas
        ...              } (solo aplica si las IRQ están vectorizadas)
        nop
0x00c0  b isrADC
```

Tabla de vectores de excepción del ARM7TDMI (7 vectores + 1 reservado)

Tabla de vectores de IRQ del S3C44B0X (26 vectores + 15 reservados)

Bootloader elemental

configuración básica del microcontrolador

```
        bootloader:

0x00c4      ldr r0, =WTCON
            ldr r1, =0x0
            str r1, [r0]

0x00d0      ldr r0, =LOCKTIME
            ldr r1, =0xfff
            str r1, [r0]
            ldr r0, =PLLCON
            ldr r1, =0x38021
            str r1, [r0]
            ldr r0, =CLKSLOW
            ldr r1, =0x8
            str r1, [r0]
            ldr r0, =CLKCON
            ldr r1, =0x7ff8
            str r1, [r0]
```

deshabilita el watchdog

Configura el gestor de reloj

estabilización del PLL = 512 us

MCLK = 64 MHz

MCLK_SLOW = 500 KHz

modo NORMAL y reloj distribuido a todos los controladores

J.M. Mendías 2018

Bootloader elemental

autocopia en RAM

Configura el controlador de memoria

```
0x0100  ldr   r0, =memconf
        ldr   r1, =_image_start
        sub   r0, r0, r1
        ldmia r0, {r1-r13}
        ldr   r0, =BWSCON
        stmia r0, {r1-r13}
```

corrige la diferencia existente entre la dirección en ROM y la dirección en RAM establecida en el linker script

carga los registros de configuración utilizando ldmia/stmia según la recomendación del fabricante

```
0x0118  ldr   r0, =ROM_START_ADDRESS
        ldr   r1, =_image_start
        ldr   r2, =_image_end
remap_loop:
        cmp   r1, r2
        ldrcc r3, [r0], #4
        strcc r3, [r1], #4
        bcc   remap_loop
```

las direcciones de comienzo/fin de la imagen están definidas por el linker script

lee una palabra de ROM (post-incrementando RO)

escribe la palabra leída de RAM (post-incrementando R1)

Copia en RAM la imagen (boot+app) residente en ROM

```
0x0134  ldr pc, =ram_exec
```

esta instrucción salta a una dirección absoluta definida durante el proceso de enlace (que según lo indicado en el linker script asume que la imagen está ubicada en RAM)

Tiene como efecto que el bootloader pasa a ejecutarse desde RAM

J.M. Mendías 2018

Bootloader elemental

inicialización de las pilas del sistemas

```
ram_exec:
0138h   mrs r0, cpsr
        bic r0, r0, #0x1f

        orr r1, r0, #UNDMODE
        msr cpsr_c, r1
        ldr sp, =UNDSTACK

        orr r1, r0, #ABTMODE
        msr cpsr_c, r1
        ldr sp, =ABTSTACK

        orr r1, r0, #IRQMODE
        msr cpsr_c, r1
        ldr sp, =IRQSTACK

        orr r1, r0, #FIQMODE
        msr cpsr_c, r1
        ldr sp, =FIQSTACK

        orr r1, r0, #SVCMODE
        msr cpsr_c, r1
        ldr sp, =SVCSTACK
```

Inicializa los SP de cada uno de los modos de ejecución privilegiados

desde modo SVC cambia a modo UND e inicializa el SP_und

desde modo UND cambia a modo ABT e inicializa el SP_abt

desde modo ABT cambia a modo IRQ e inicializa el SP_irq

desde modo IRQ cambia a modo FIQ e inicializa el SP_fiq

desde modo FIQ cambia a modo SVC e inicializa el SP_svc

Bootloader elemental

cesión del control

```
0x017c    mrs  r0, cpsr
          bic  r0, r0, #0xc0
          msr  cpsr_c, r0
```

Habilita las líneas IRQ/FIQ del procesador permaneciendo en modo SVC

```
0x0188    bl   main
          b    .
```

El bootloader cede el control al programa principal en C

si retornara de main, ejecuta un bucle infinito

```
        memconf:
0x018c    .word 0x01000000 /* BWSCON   */
          .word 0x00000400 /* BANKCON0 */
          .word 0x00007ffc /* BANKCON1 */
          .word 0x00000000 /* BANKCON2 */
          .word 0x00007ffc /* BANKCON3 */
          .word 0x00000000 /* BANKCON4 */
          .word 0x00000000 /* BANKCON5 */
          .word 0x00018000 /* BANKCON6 */
          .word 0x00000000 /* BANKCON7 */
          .word 0x0086041b /* REFRESH  */
          .word 0x00000016 /* BANKSIZE */
          .word 0x00000020 /* MRSR6    */
          .word 0x00000000 /* MRSR7    */
```

constantes de configuración del controlador de memoria

```
0x01c4    .ltorg
```

Pool de literales usados en el código ensamblador

```
          .end
```

Fin del bootloader

Bootloader elemental

linker script

- El proyecto completo debe enlazarse según la ubicación final desde donde se ejecute (diferente de la ubicación en donde reside tras reset)

```
ENTRY( main )
SECTIONS
{
  . = 0x0C000000;

   _image_start = .;

  .text : { bootloader.o * (.text) }
  .rodata : { *(.rodata) }
  .data : { *(.data) }
  .bss : { *(.bss) }

   _image_end = .;

  end = .;
}
```

no es necesario, se conserva por si se quiere depurar la aplicación (no el bootloader) como proyecto volcado en RAM

dirección donde se ubicará el proyecto (comienzo de la RAM)

define la etiqueta que indica la dirección del inicio de la imagen

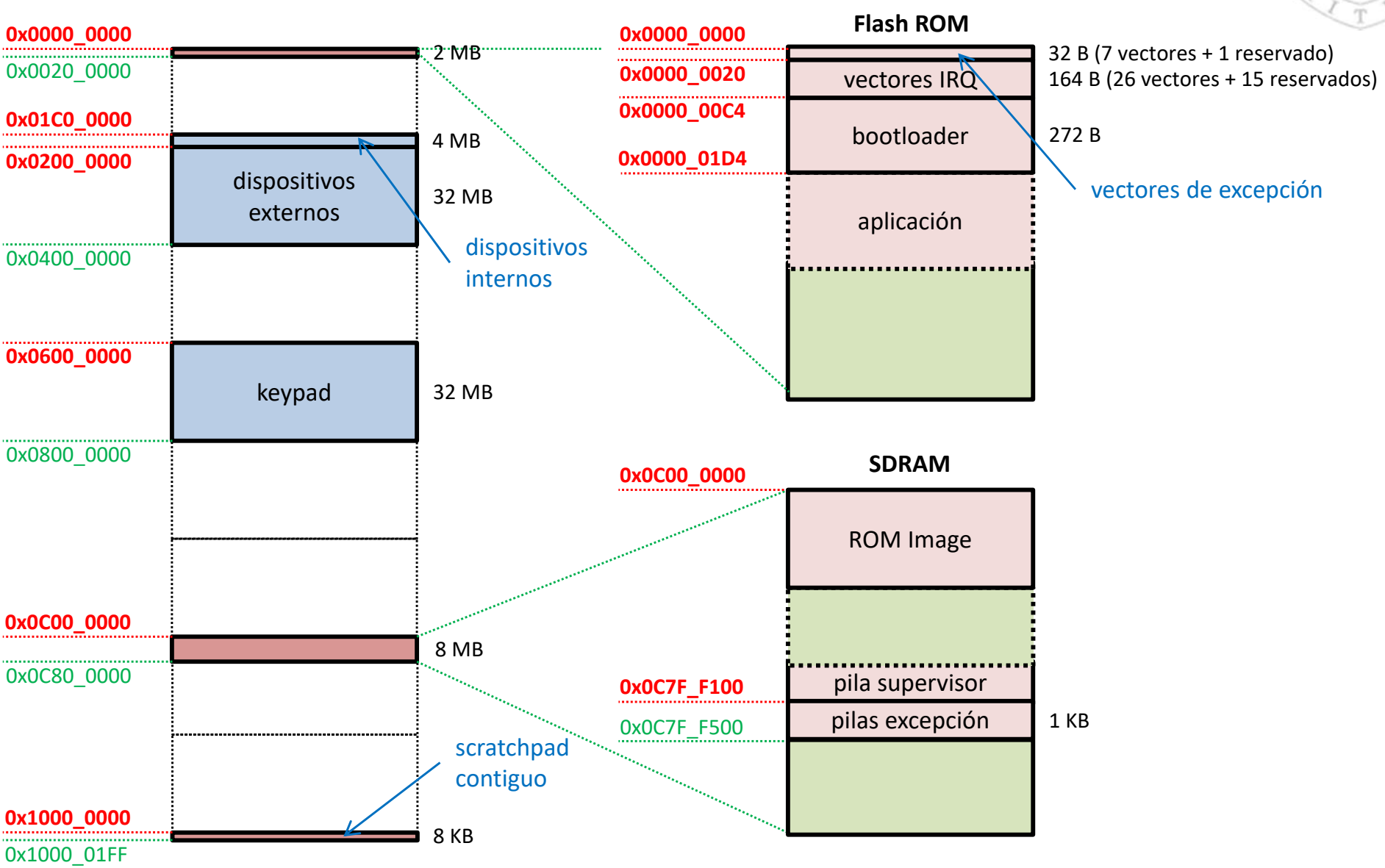
el bootloader debe ubicarse antes que el resto de código

define la etiqueta que indica la dirección del fin de la imagen

J.M. Mendías 2018

Bootloader elemental

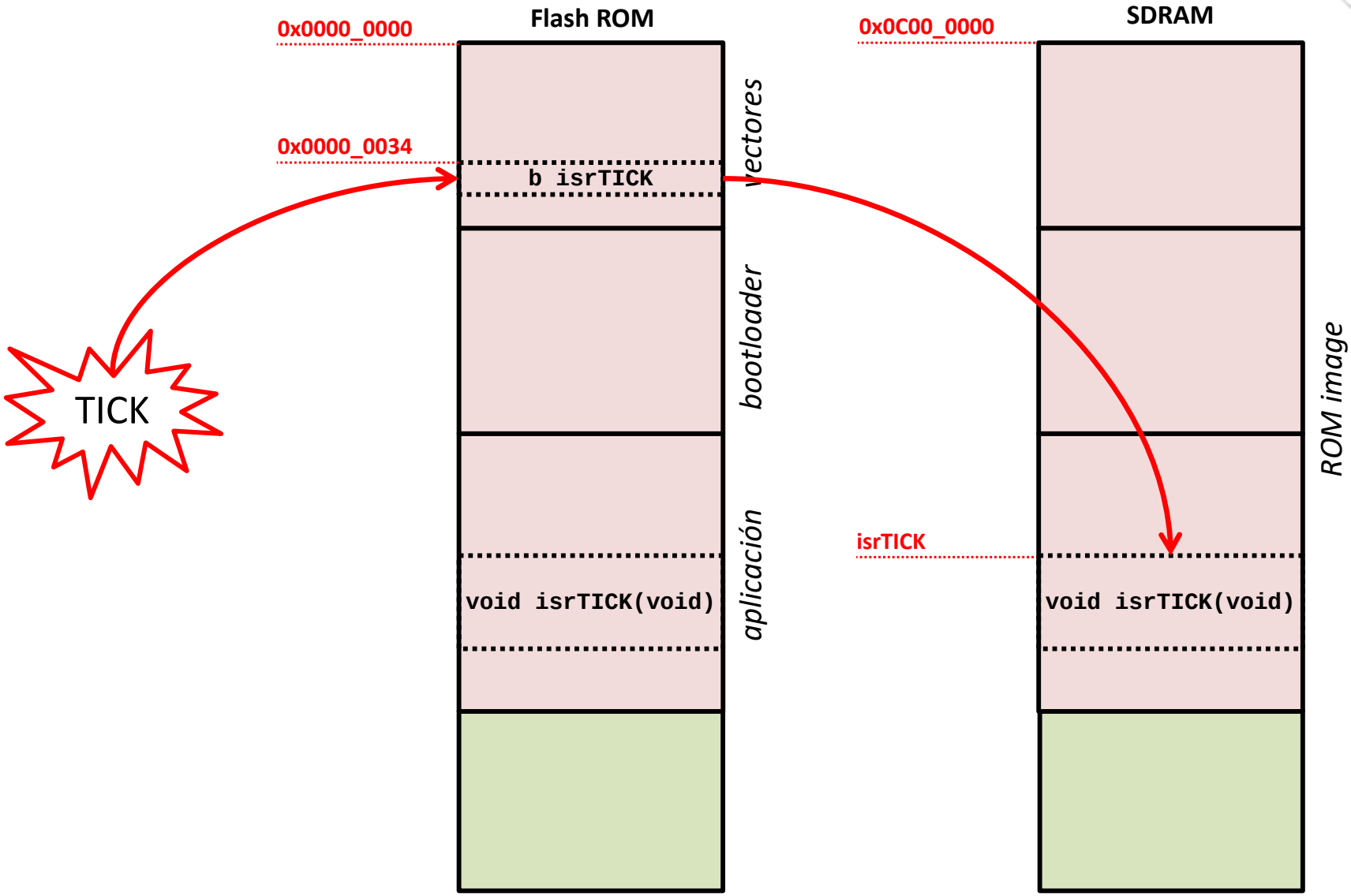
mapa de memoria



J.M. Mendías
2018

Bootloader elemental

comportamiento en interrupciones



J.M. Mendías 2018

Bootloader elemental

generalización

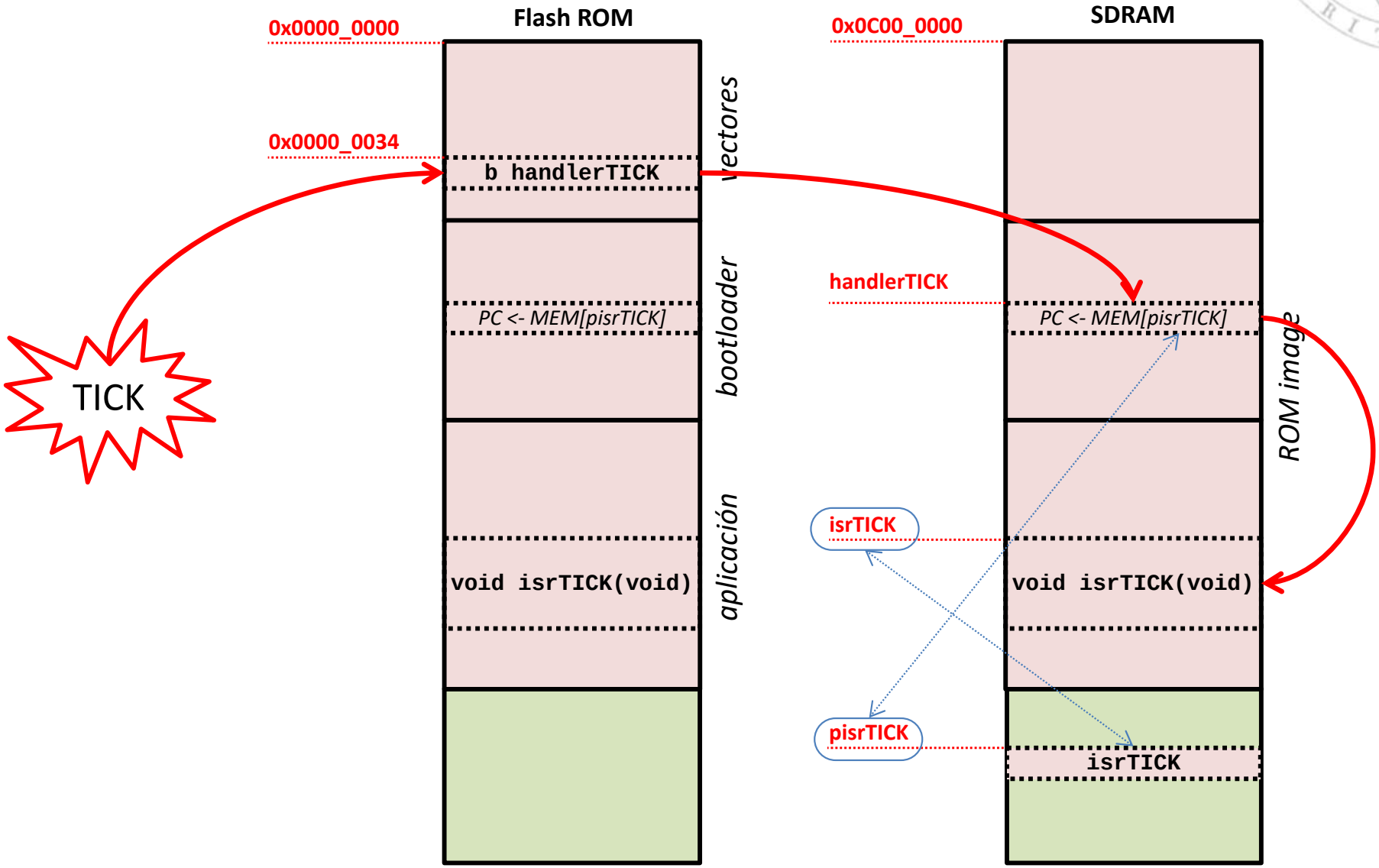
- El bootloader elemental presentado:
  - Instala en tiempo de compilación en la tabla de vectores ubicada en ROM las RTE/RTI definidas en la aplicación.
  - Es específico para cada aplicación
    - El número y nombre de las RTI activas puede cambiar de una aplicación a otra.
  - No permite el portado directo de los proyectos volcados en RAM
    - Ya que estas aplicaciones instalan las RTI en tiempo de ejecución en una tabla virtual de vectores en RAM (ante la imposibilidad de hacerlo en ROM).

- Para disponer de un bootloader genérico y compatible con otros labs:
  - Cada RTE deberá tener asociado un prólogo (handler) instalado en tiempo de compilación en la tabla de vectores ubicada en ROM.
  - La función de cada prólogo es saltar a la correspondiente RTE/RTI
    - asumiendo que la aplicación ha instalado previamente y en tiempo de ejecución la RTE/RTI en una tabla virtual de vectores ubicada en RAM a partir de 0xc7fff00 (misma ubicación que en proyectos volcados en RAM).
  - Adicionalmente, se consigue un grado adicional de versatilidad (las RTI pueden cambiarse en tiempo de ejecución) a costa de sobrecargar el tiempo de respuesta.

J.M. Mendías
2018

Bootloader elemental

generalización



PSyD bootloader

declaraciones

```
.include "s3c44b0x.asm"
.include "s3cev40.asm"
```
nemotécnicos de los registros de los controladores de dispositivos

```
.equ USRMODE, 0x10
.equ FIQMODE, 0x11
.equ IRQMODE, 0x12
.equ SVCMODE, 0x13
.equ ABTMODE, 0x17
.equ UNDMODE, 0x1b
.equ SYSMODE, 0x1f
```
nemotécnicos de los modos de operación del ARM7TDMI

*Macro cuyo objetivo es PC = *pISR ≡ pISR()*

```
.macro HANDLER pISR
  sub   sp,  sp, #4
  stmfd sp!, {r0}
  ldr   r0,  =\pISR
  ldr   r0,  [r0]
  str   r0,  [sp,#4]
  ldmfd sp!, {r0, pc}
.endm
```
reserva espacio en pila para almacenar la dirección de la ISR

apila R0 (que actuará como registro de trabajo en el handler)

carga en R0 el contenido de la dirección indicada como argumento y lo apila (dirección de inicio de la ISR almacenada en la tabla virtual de vectores)

restaura R0 y salta a la ISR (desapilando ambos de la pila)

```
.extern _image_start
.extern _image_end
```
dirección de comienzo/fin de la imagen (boot+app), definidas en el linker script

```
.extern main
```
punto de entrada a la aplicación a la que el bootloader cede el control

PSyD bootloader

vectores de excepción/interrupción

```
.section .text ..... comienza el código (sección .text)

0x0000  b bootloader ..... tras reset salta al comienzo del programa de arranque
        b handlerUndef
        b handlerSWI
        b handlerPabort
        b handlerDabort      } salto a las RTE
        nop ..... vector reservado
        b handlerIRQ ..... solo aplica si las IRQ no están vectorizadas
0x001c  b handlerFIQ ..... salta a la RTI-FIQ

0x0020  b handlerUSB
        b handlerKEYPAD
        b handlerTS
        b handlerETHERNET    } salta a las RTI externas
        b handlerPB          (solo aplica si las IRQ están vectorizadas)
        b handlerTICK
        nop
        nop
        b handlerZDMA0
        ...
        nop                  } salta a las RTI internas
0x00c0  b handlerADC         (solo aplica si las IRQ están vectorizadas)
```

Tabla de vectores de excepción del ARM7TDMI (7 vectores + 1 reservado)

Tabla de vectores de IRQ del S3C44B0X (26 vectores + 15 reservados)

PSyD bootloader

prólogos para rutinas de tratamiento

```
0x00c4   handlerUndef:     HANDLER pISR_UNDEF
         handlerSWI:       HANDLER pISR_SWI
         handlerPabort:    HANDLER pISR_PABORT
         handlerDabort:    HANDLER pISR_DABORT
         handlerIRQ:       HANDLER pISR_IRQ
         handlerFIQ:       HANDLER pISR_FIQ

0x154    handlerUSB:       HANDLER pISR_USB
         handlerKEYPAD:    HANDLER pISR_KEYPAD
         handlerTS:        HANDLER pISR_TS
         handlerETHERNET:  HANDLER pISR_ETHERNET
         handlerPB:        HANDLER pISR_PB
         handlerTICK:      HANDLER pISR_TICK
         handlerZDMA0:     HANDLER pISR_ZDMA0
         ...
         handlerRTC:       HANDLER pISR_RTC
0x03ac   handlerADC:       HANDLER pISR_ADC
```

cada HANDLER ocupa 6 palabras

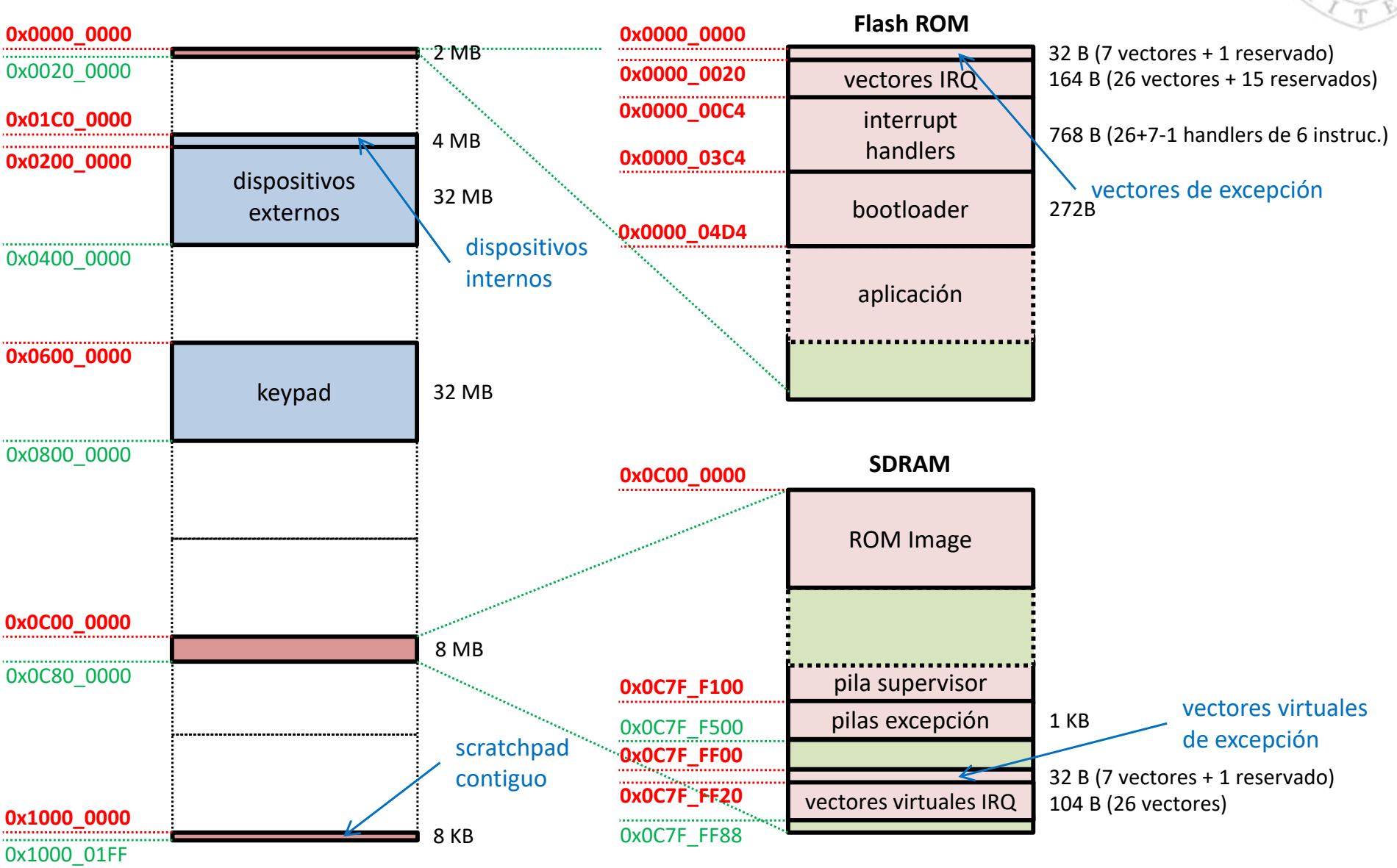
prólogos para RTE

prólogos para RTI-IRQ

J.M. Mendías 2018

PSyD bootloader

mapa de memoria



Acerca de *Creative Commons*

- Licencia CC (Creative Commons)
  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/