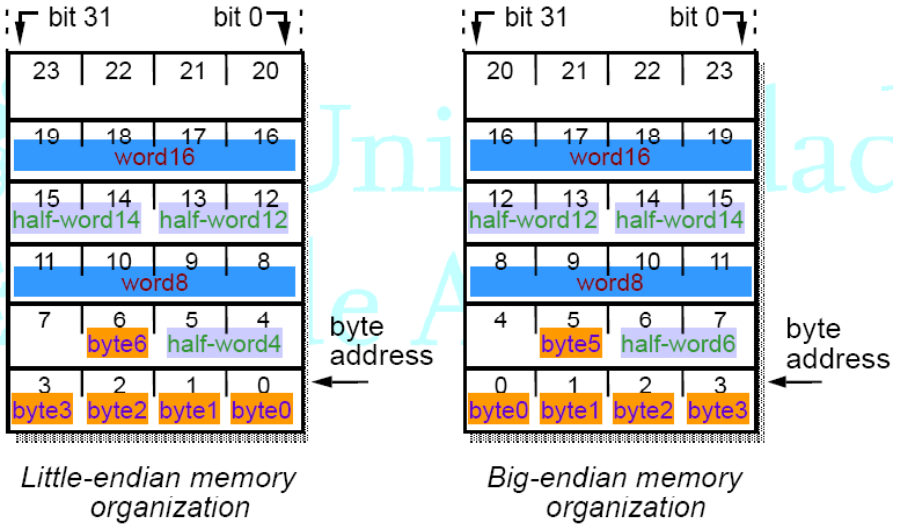
Sistemas electrónicos digitales

Tema 2:
La arquitectura del ARM



Índice del tema

- Evolución histórica
- Modos de funcionamiento
- Modelo de programación del ARM
  - ISA
  - Registros
  - Datos y organización de la memoria/registros
  - Juego de instrucciones
- Excepciones en el ARM
- Herramientas de desarrollo para ARM

Evolución histórica I

- 1983-1985: El primer procesador ARM se desarrolla en Acorn Computer Ltd., con la denominación Acorn RISC Machine → Advanced RISC Machine
- 1990: La colaboración entre Acorn y Apple finaliza con la creación de Advanced RISC Machines en UK
- 1991: Desarrollo del primer procesador RISC para aplicaciones empotradas (ARM6)
- 1992-1994: Empresas como Sharp y Samsung solicitan licencias de uso del ARM para aplicaciones empotradas
- 1993: Se crea el ARM7, enfocado a aplicaciones multimedia
- 1995: Desarrollo de la arquitectura Thumb y el core ARM8
- 1996-2000: Alcatel, Philips, Sony y Erickson solicitan licencia para usar el ARM en sus aplicaciones
- 2001-2006: El porcentaje del ARM en procesadores de aplicaciones empotradas, de 32-bits, alcanza el 80%

Evolución histórica II

- Bases de la arquitectura del ARM
  - Estado del arte:
    - Sun SPARC; Stanford MIPS
  - Basado en el procesador Berkeley RISC I:
  - Características heredadas:
    - Arquitectura del tipo *load-store*
    - Instrucciones de longitud fija, 32 bits
    - Instrucciones de tres direcciones
  - Características rechazadas:
    - Arquitectura basada en registros de ventanas
      - Elevado uso de área
    - Saltos retardados (*delayed branches*)
    - Ejecución de las instrucciones en un único ciclo de reloj
      - ARM → trata de minimizar el número de ciclos de ejecución

Evolución histórica: Versiones de la arquitectura

- Versión 1 (obsoleta): 26-bit *address*; datos tamaño byte y palabra; no incluye instrucciones de multiplicación ni soporta co-procesador; incluye llamadas al Sistema Operativo
- Versión 2 (obsoleta): 26-bit *address*; incluye instrucciones de multiplicación y da soporte a co-procesador; aumenta los registros internos (*banked registers*) en modo *fast-interrupt*
- Versión 3: Primer ARM diseñado por ARM *Limited* (1990); 32-bit *address*; registros CPSR y SPSR separados; añade dos nuevos modos de funcionamiento (*abort* y *undefined*)
- Versión 4: Añade datos *halfword*; instrucciones *load* y extensión de signo en byte y *halfwords*; se añade un nuevo modo (*system mode*)
  - Versión 4T: Añade juego de instrucciones comprimidas,16-bit Thumb
- Versión 5: Mejora la eficiencia de instrucciones comprimidas y las instrucciones de co-procesador; introduce *soft-breakpoint*

Evolución histórica: Familia de procesadores I

Ejemplo de procesadores y características generales

Core	Versión	Características
ARM1,2,3	v1,2,3	•No comerciales
ARM6, ARM60, ARM610, ARM7, ARM710, ARM7D, ARM7DI	v3	•Arquitectura Von Neumann •32-bit *address* •3 etapas de *pipeline* •Modos *und* y *abt*
StrongARM, SA-110, SA-1100 ARM8, ARM810	v4	•*Half word*, *signed halfword/byte* •*System mode* •*Long multiply instructions* •5 etapas de *pipeline* •ARM8: Von Neumann •StrongARM: Harvard
ARM7TDMI, ARM9TDMI	v4T	•*Thumb instruction set* •ARM7: Von Neumann; 3 etapas pipeline •ARM9: Harvard; 5 etapas pipeline

T: *Thumb Instruction Set* **D: *On-chip Debug***
M: *Enhanced Multiplier* **I: *Embedded ICE Logic***

Evolución histórica: Familia de procesadores II

Ejemplo de procesadores y características generales

Core	Versión	Características
ARM1020T	V5	•Mejora la eficiencia de las instrucciones comprimidas •Añade *Leading Zero Counter*
ARM9E-S, ARM10TDMI, ARM1020E	v5TE	•*Enhanced DSP Instructions* •*Saturated signed arithmetic*
ARM7EJ-S, ARM926EJ-S, ARM1026EJ-S	v5TEJ	•Extensión *Jazelle* optimiza la ejecución de aplicaciones Java en ARM
ARM11, ARM1136J-S	v6	•Diseño de muy bajo consumo •8 etapas de *pipeline* •Tecnología SIMD (*Single Instruction Multiple Data*)

J: *Jazelle*
S: *Synthesizable*
E: *Enhanced DSP Instructions*

Modos de funcionamiento

Modo	Abrev.	Descripción
User	(usr)	•Modo normal de ejecución de programas
FIQ	(fiq)	•*Fast Interrupt Request .* Empleado en transferencias de alta velocidad
IRO	(irq)	•*Interrupt Request.* Modo normal de gestión de interrupciones
Supervisor	(svc)	•Modo protegido para el sistema operativo
Abort	(abt)	•Usado cuando se aborta el ciclo de *Fetch* de instrucción o de datos
Undefined	(und)	•Usado para soportar emulación por software, principalmente emulación de co-procesador
System	(sys)	•Para ejecutar tareas del sistema operativo en modo privilegiado

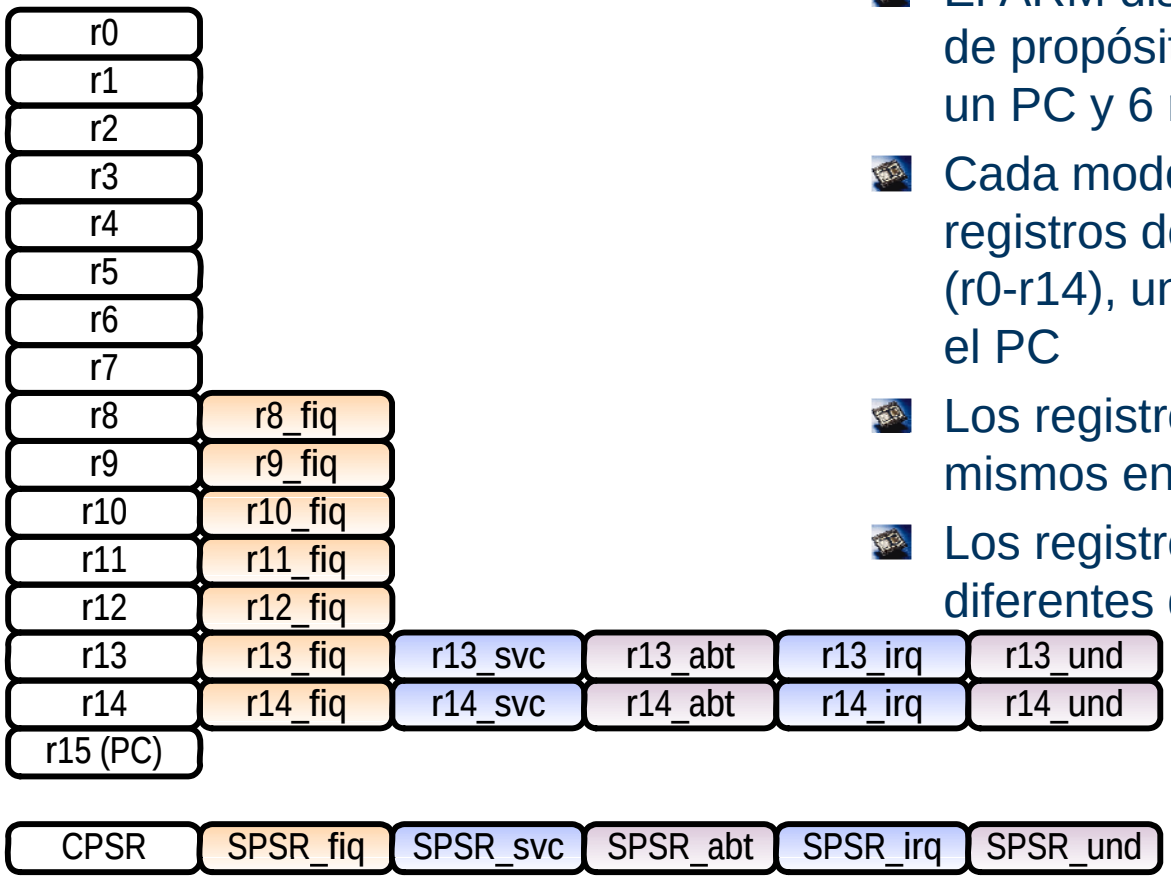
- El cambio de modos puede realizarse por software o por hardware
- El modo normal de ejecución es el modo usuario
- El modo *system* únicamente se soporta a partir de la versión 4

Modelo de programación del ARM: ISA

- Arquitectura del juego de instrucciones:
  - Conjunto de operaciones que el programador puede realizar para modificar el estado del procesador
  - El estado de un procesador se define por:
    - Datos almacenados en los registros visibles del procesador
    - Contenido de las memorias
    - Los registros no visibles o temporales no afectan al estado de un procesador
  - Cada instrucción provoca la transición entre dos estados
  - Únicamente se toman en consideración los valores almacenados en los registros visibles del procesador
  - En el ARM el estado del procesador, a nivel de usuario, dispone de 15 registros de propósito general de 32 bits, un contador de programa y un registro de estado (CPSR)

Modelo de programación del ARM: Registros I

Registros en el ARM



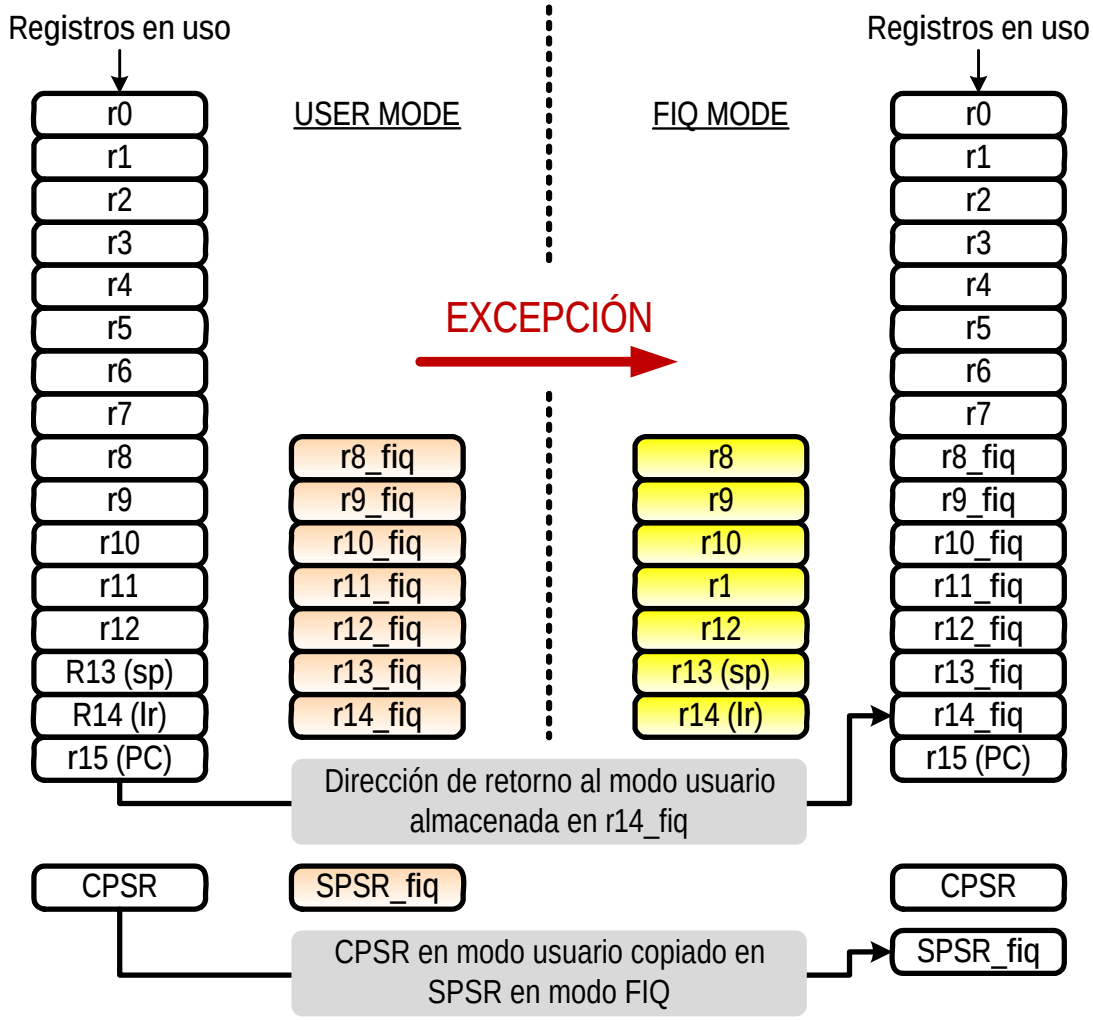
- El ARM puede funcionar en seis modos diferentes
- El ARM dispone de 37 registros de propósito general, incluyendo un PC y 6 registros de estado
- Cada modo dispone de 15 registros de propósito general (r0-r14), un registro de estado y el PC
- Los registros r0 a r7 son los mismos en todos los modos
- Los registros r8 a r14 son diferentes dependiendo del modo de funcionamiento del procesador

Modelo de programación del ARM: Registros II

- r15 es usado como *Program Counter* (PC). Contiene la dirección de la siguiente instrucción
  - Se puede usar como cualquier registro de propósito general, salvo en contadas ocaciones
  - La dirección almacenada debe ser múltiplo de 4 (*word-alignment)*
  - Su lectura devuelve el valor de la dirección actual + 8
- r13 es usado como *Stack Pointer* (SP). Cada modo dispone de su propio puntero de pila, lo que indica el uso de una pila dedicada para cada modo de funcionamiento (almacenamiento de los registros de usuario)
- r14 es usado como *Link Register* (LR). Realiza dos funciones:
  - Cuando ocurre una excepción, se almacena la dirección de retorno en el registro r14. El retorno de excepción se realiza mediante la operación r15 ← r14 (Ensamblador: `MOV r15, r14`)
  - En las llamadas a subrutina actúa de igual manera, manteniendo la dirección de retorno de la rutina

Modelo de programación del ARM: Ejemplo

Ejemplo del uso de los registros en modo FIQ



Modelo de programación del ARM: CPSR y SPSR

Formato del CPSR y del SPSR

- Códigos de condición (ALU *status*):
  - N: Negativo; Z: Cero; C: *Carry out*; V: Overflow
  - Q: *Sticky Overflow* (Core 5TE)
- Máscaras de interrupción:
  - I: A “1” deshabilita IRQ
  - F: A “1” deshabilita FIQ
- Bit T: Selección del conjunto de instrucciones Thumb
  - 0: Instrucciones ARM
  - 1: Ejecución Thumb
  - En versiones sin T, habilita el modo de traza
- Modo: Codifica el modo de operación

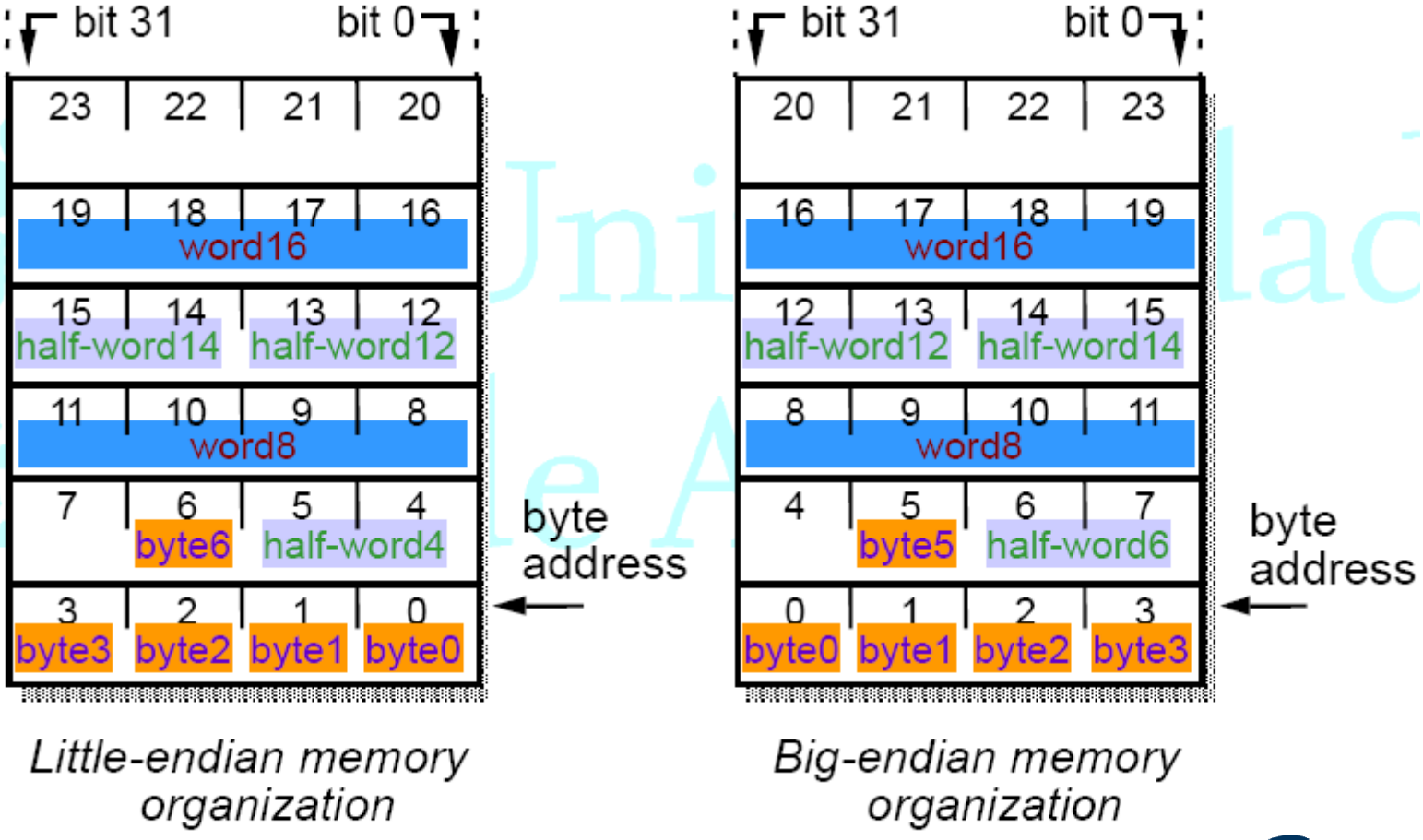
CPSR[4:0]	Modo	Reg.
10000	User	user
10001	FIQ	_fiq
10010	IRQ	_irq
10011	SVC	_svc
10111	Abort	_abt
11011	Undef	_und
11111	System	user

Modelo de programación del ARM: Tipos de datos

- El ARM puede operar con seis tipos de datos:
  - *Byte*: palabra de 8 bits (con o sin signo)
  - *Halfword*: palabra de 16 bits (con o sin signo)
  - *Word*: Palabra de 32 bits (con o sin signo)
- Representación de números negativos en complemento a 2
  - Sin signo: 0 .. 2^N-1
  - Con signo: -2^{N-1} .. 2^{N-1}-1
- Operaciones de almacenamiento en registros de *byte/ halfword/word*:
  - Posibilidad de extensión de signo en *byte* y *halfword*
- Instrucciones:
  - Todas de 32 bits. En modo Thumb se agrupan de dos en dos

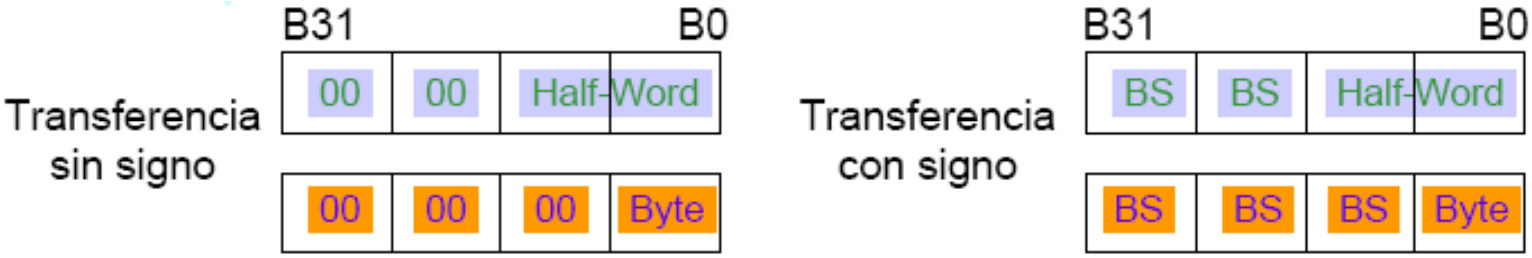
Modelo de programación del ARM: Org. de la Memoria

- Direccionamiento al *byte* y alineamiento de *words* y *halfwords*. Por defecto se organiza en *little-endian*



Modelo de programación del ARM: Org. de los reg.

- Organización de los datos en registros:
  - La transferencia de datos en tamaños inferiores a la palabra (32 bits) se almacenan en la parte inferior del registro. Si no se especifica nada, la parte alta del registro se rellena con ceros
  - Existen instrucciones específicas de transferencia de *bytes* y *halfwords* con signo. En este caso la parte alta del registro será una extensión del signo del operando



Modelo de programación del ARM: Juego de instrucc.

- Arquitectura *load/store*
- Clasificación:
  - Instrucciones de procesamiento de datos
  - Instrucciones de transferencia de datos (*load/store*)
  - Instrucciones de control de ejecución
- Principales características:
  - Instrucciones de 3 direcciones
  - Ejecución condicional de todas las instrucciones
  - Transferencias de múltiples registros
  - Operación de desplazamiento más ALU en un único ciclo
  - Juego de instrucciones abierto mediante el uso de un juego de instrucciones de co-procesador → nuevos registros y tipos de datos
  - Instrucciones Thumb → aumentan la "densidad del código"

Excepciones en el ARM: Definición y clasificación

- Las excepciones se utilizan para gestionar todos aquellos eventos no esperados durante la ejecución de un programa, tales como interrupciones o fallos de memoria
- Cada modo está asociado a una excepción, excepto *system mode*
  - *System mode:* Hay que modificar el CPSR en modo supervisor
- Clasificación de las excepciones:
  - Generadas por software:
    - Interrupciones por software; instrucciones no definidas; *Prefetch aborts* debido a un error en memoria al buscar un código de operación
  - Generadas como efecto secundario en la ejecución de una instrucción:
    - *Data aborts* debido a un error en memoria durante el acceso a lectura/escritura de un dato
  - Generadas externamente:
    - Reset, IRQ y FIQ

Excepciones en el ARM: Tratamiento de una excepc.

- Cuando ocurre una excepción:
  - Se finaliza la instrucción en curso (excepto en reset)
  - Se salva el estado de la máquina:
    - Se copia el PC en r14_mode (excepto en reset)
    - Se copia CPSR en SPSR_mode (excepto en reset)
  - Se cambia al modo de operación correspondiente
  - Se actualiza el CPSR:
    - Se actualizan los bits de modo del CPSR (en reset a modo supervisor)
    - Se activa el modo ARM si es necesario (en reset, siempre)
    - Se desactivan las interrupciones IRQ (siempre) y FIQ (sólo en modo FIQ o en caso de reset)
  - Re realiza el mapeado al banco de registros correspondiente
  - Se actualiza el PC con el valor del vector correspondiente a la fuente de interrupción

Excepciones en el ARM: Tabla de vectores de excepc.

Tabla de vectores de excepción:

Exception	Mode	Vector Normal	Vector High
Reset	svc	0x00000000	0xFFFF0000
Undefined instruction	und	0x00000004	0xFFFF0004
Software interrupt	svc	0x00000008	0xFFFF0008
Prefetch abort (instruction fetch mem. fault)	Abort	0x0000000C	0xFFFF000C
Data abort (data access memory fault)	Abort	0x00000010	0xFFFF0010
IRQ (normal interrupt)	IRQ	0x00000018	0xFFFF0018
FIQ (fast interrupt)	FIQ	0x0000001C	0xFFFF001C

- En la dirección de cada vector se debe localizar un salto a la dirección de comienzo de la Rutina de Servicio de la Excepción, excepto en el caso de FIQ que, al ser la última, puede localizarse el código de la RSE con el consiguiente ahorro de tiempo

Excepciones en el ARM: Rutina de SE I

- La ejecución de una Rutina de Servicio de Excepción debe contemplar:
  - Almacenar en la pila todos aquellos registros de usuario que se vayan a modificar, excepto en FIQ, donde se recomienda el uso del banco de registros r8-r12
  - Los punteros de pila se deben inicializar en modo supervisor (por ejemplo tras un reset) durante el arranque del sistema
- El retorno de una Rutina de Servicio de Excepción debe contemplar:
  - Restaurar los registros de usuario modificados
  - Restaurar el CPSR a partir del SPSR
  - Restaurar el valor del PC a su valor antes de la ejecución
- Las dos últimas operaciones han de realizarse simultáneamente

Excepciones en el ARM: Rutina de SE II

- Existen instrucciones que permiten restaurar la dirección de retorno, así como el CPSR
  - Retorno de interrupción por software o no definida
    - `MOVS pc, r14` ; La “S” en el código de operación fuerza ; la copia del SPSR_svc al CPSR
  - Retorno de una interrupción IRQ, FIQ o *Prefetch abort*:
    - `SUBS pc, r14, #4` ; Retorna a la siguiente instrucción a ; ejecutar
  - Retorno de una excepción *Data abort*:
    - `SUBS pc, r14, #8` ; Permite volver a ejecutar la instrucción ; que dio lugar a la interrupción
  - El primer tipo no necesita modificación del contenido de r14 al ser excepciones “esperadas” por lo que el registro r14 contendrá la dirección válida de retorno

Herramientas de desarrollo para el ARM

- Necesidad de plataformas de desarrollo cruzadas (*cross-development*).
  - C compiler: soporta ANSI C; genera código Thumb; soporte de librerías de funciones
  - Genera ARM *object format*; puede enlazarse con el código generado por el compilador
  - Linker: enlaza varios módulos generando un objeto ".aif" o ".axf". Sustituye las referencias simbólicas (*address*)
  - ARMsd *(symbolic debugger):*
    - *Development board*
    - ARMulator: *Instruction, Cycle, Timming-accurate*