



ARMv7 - Gestión de Interrupciones

Alejandro Furfaro

20 de julio de 2020

1 Introducción

2 Sistema de Interrupciones ARMv7, perfiles A y R

3 Interrupciones en CORTEX-M

- Interrupciones en Microprocesadores
 - Generalidades del sistema de excepciones en ARM
- Gestión de Excepciones
 - Interrupciones
 - SOC SITARA AM335x: Controlador de Interrupciones
- Arquitectura ARMv7-M
 - el NVIC

Contenido

1 Introducción

- Interrupciones en Microprocesadores
- Generalidades del sistema de excepciones en ARM

2 Sistema de Interrupciones ARMv7, perfiles A y R

- Gestión de Excepciones
- Interrupciones
- SOC SITARA AM335x: Controlador de Interrupciones

3 Interrupciones en CORTEX-M

- Arquitectura ARMv7-M
- el NVIC



Acceso a los periféricos

- Nuestro scope en esta parte del estudio de un sistema de cómputo es el acceso a la Entrada Salida
- Para ello tenemos dos abordajes posibles
 - 1 Pooling: El procesador encuesta periódicamente dispositivo por dispositivo
 - 2 **Interrupción**: El procesador trabaja en procesamiento de datos y cada vez que un dispositivo tiene necesidad de entregar un dato al sistema o solicitarlo envía una señal eléctrica al procesador requiriendo servicio.
- Por lo general se opta por el segundo abordaje ya que el procesador se desentiende de vigilar la Entrada Salida y se concentra en las tareas que debe realizar. el pooling consume mucho procesamiento



Interrupciones en sistemas de cómputo modernos

No se limita al dominio de la Entrada Salida solamente

Los sistemas de computo modernos tienen por lo general un Sistema Operativo de gran porte (como Linux), o mas modestos como en el caso de los microcontroladores.

Pero por lo general salvo que sea un controlador mínimo, un sistema tiene un conjunto de actividades que desarrollará (tareas) y un Software de Administración (el Sistema Operativo).

El ojetivo de este tema es mostrar que las interrupciones no se limitan a acceder a un periférico, sino que también ayudan al sistema operativo en la administración, y permiten a las tareas a solicitar servicios al sistema operativo.



Fuentes de interrupción

3. *Internas.*

- Se conocen por lo general como excepciones
- son generadas por la propia CPU a consecuencia de una situación que le impide completar la ejecución de la instrucción en curso.
- Los ejemplos dependen de la CPU que se analice, pero por lo general encontrar un OPCODE indefinido en cualquier microprocesador generará una interrupción de este tipo, ya que al intentar decodificarla la CPU no puede generar ninguna acción interna.

cuando ocurre algo que no te permite ejecutar la instrucción en curso, se ejecuta una excepción. Por ejemplo cuando se quiere ejecutar una instrucción que no existe (por ej LDR r5, =0x23874)



Contenido

1 Introducción

- Interrupciones en Microprocesadores
- Generalidades del sistema de excepciones en ARM

2 Sistema de Interrupciones ARMv7, perfiles A y R

- Gestión de Excepciones
- Interrupciones
- SOC SITARA AM335x: Controlador de Interrupciones

3 Interrupciones en CORTEX-M

- Arquitectura ARMv7-M
- el NVIC



Manejo de Interrupciones en ARM

- Desafortunadamente la arquitectura ARMv7 no posee un sistema uniforme de interrupciones
- El sistema de interrupciones del CORTEX-M es bastante diferente del definido en la arquitectura para los CORTEX-A y CORTEX-R
- Tendremos que entender cada sistema por separado a fin de poder lidiar con cualquier tipo de sistema pudiendo cubrir el rango completo de aplicaciones posibles.
- Esta permanente diferenciación del Perfil M es el principal punto débil de v7.
- RISC-V es una definición única de arquitectura independientemente de la escala del sistema a desarrollar



Contenido

1 Introducción

- Interrupciones en Microprocesadores
- Generalidades del sistema de excepciones en ARM

2 Sistema de Interrupciones ARMv7, perfiles A y R

- Gestión de Excepciones
- Interrupciones
- SOC SITARA AM335x: Controlador de Interrupciones

3 Interrupciones en CORTEX-M

- Arquitectura ARMv7-M
- el NVIC



Gestión de Interrupciones y Excepciones

Interrupciones y Excepciones en Microprocesadores

Si analizamos el diagrama general de cualquier SoC (System on Chip), desde el mas simple que controle un par de sensores y active algunas señales sencillas, o un sistema Real Time para controlar un satélite en el espacio, o un procesador de los muchos que componen un sistema SMP de alta performance, o uno de los cores de un computador portátil, smartphone, etc, vienen acompañados inevitablemente un par de controladores muy cercanos a la CPU: El controlador de Memoria y el Controlador de Interrupciones.



Manejo de Excepciones

Definición de Excepción

Se produce una excepción en la CPU cuando se encuentra o aparece una condición que detiene la secuencia de ejecución normal de una instrucción.

Estas condiciones en un core ARM Pueden ser:

- Reset
- Fallo en una operación de acceso a memoria (Fetch, Read o Write). por ej cuando se quiere leer una posición de memoria y esta no es múltiplo de 4
- Lectura de un Opcode indefinido
- Ejecución de una interrupción de Software
- Activación de una línea de interrupción



Excepciones, Vectores, tipos, handlers

Procesamiento de una excepción

- 1 Se genera una excepción de cualquier tipo.
- 2 La CPU suspende la ejecución de la secuencia de instrucciones en curso.
- 3 De acuerdo a cual haya sido la excepción ejecuta una subrutina dedicada a implementar la función asociada a esa excepción.



Para ello se requiere que cada fuente de excepción pueda ser identificada unívocamente.

Debe disponer también de un mecanismo para que una vez identificada la interrupción se pueda acceder a la subrutina que ejecuta la función que corresponde a esa excepción o interrupción.



Excepciones y Modos en procesadores ARM

Cada excepción en un procesador ARM pone al core en un **Modo** específico de acuerdo con la siguiente tabla:


Excepción	Modo	Propósito Principal
Fast Interrupt Request	FIQ	fast interrupt request handling
Interrupt Request	IRQ	interrupt request handling
SWI and Reset	SVC	protected mode for operating systems
Prefetch Abort and Data Abort	abort	virtual memory and/or memory protection handling
Undefined Instruction	undefined	software emulation of hardware coprocessors

- Se puede poner al core en cualquiera de estos modos seteando apropiadamente los bits de **Modo** del registro **CPSR**. (En todos los modos excepto en Modo User)
- Los otros dos Modos, en que puede estar el procesador son *User* y *System*, y la única forma de que el core ingrese a uno de ellos es mediante escritura en los bits de **Modo** del **CPSR**.



Excepciones y Modos en procesadores ARM

Cuando una excepción cambia el Modo de un core ARM la operatoria es:

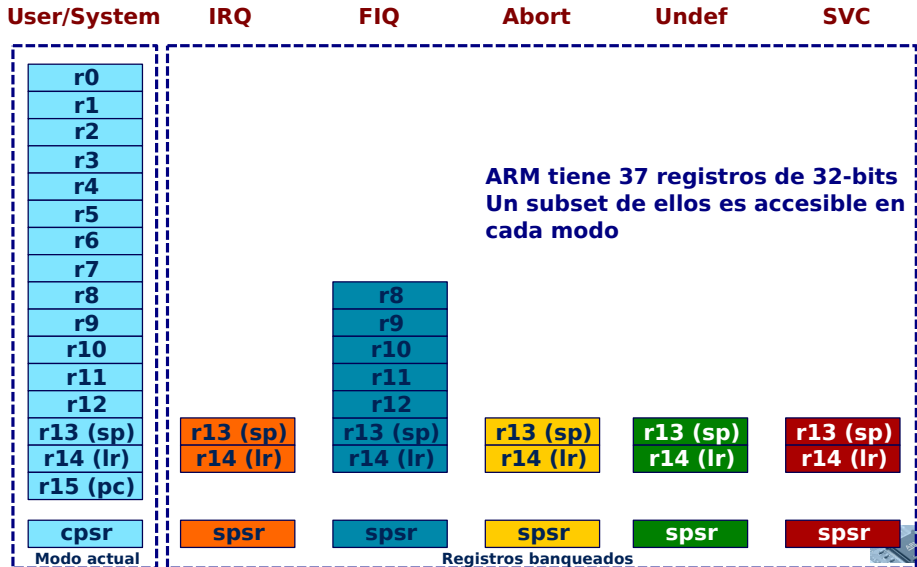
- Salva el registro **CPSR** en el registro **SPSR** correspondiente al modo de la excepción (osea el modo al que vamos a pasar) 
- Salva el **PC** en el **LR** del modo de la excepción
- Establece como **CPSR** al del modo de la excepción
- Establece en el **PC** la dirección del handler de la excepción.

Recordar

ARM banca algunos registros en los diferentes modos de funcionamiento.

Los Modos de funcionamiento se definen en los 5 bits menos significativos del CPSR.

Recordatorio: Banqueo de Registros según modo

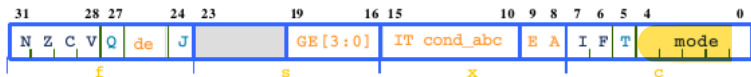


Recordatorio: Banqueo de Registros según modo

User/System	IRQ	FIQ	Abort	Undef	SVC
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8	r8	r8	r8	r8
r9	r9	r9	r9	r9	r9
r10	r10	r10	r10	r10	r10
r11	r11	r11	r11	r11	r11
r12	r12	r12	r12	r12	r12
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)
cpsr	spsr	spsr	spsr	spsr	spsr

Registros banqueados

Recordatorio: CPSR/SPSR, bits de Modo



Condition code flags

- N = **N**egative result from ALU
- Z = **Z**ero result from ALU
- C = ALU operation **C**arried out
- V = ALU operation **o**Verflowed

Sticky Overflow flag - Q flag

- Architecture 5TE and later only
- Indicates if saturation has occurred

J bit

- Architecture 5TEJ and later only
- J = 1: Processor in Jazelle state

Interrupt Disable bits

- I = 1: Disables IRQ
- F = 1: Disables FIQ

T Bit

- T = 0: Processor in ARM state
- T = 1: Processor in Thumb state
- Introduced in Architecture 4T

Mode bits

- Specify the processor mode

New bits in V6

- GE[3:0] used by some SIMD instructions
- E bit controls load/store endianness
- A bit disables imprecise data aborts
- IT [abcde] IF THEN conditional execution of Thumb2 instruction groups



Modos de operación

Modo	Mnemónico	Privilegiado	CPSR[4:0]
Abort	abt	si	10111
Fast interrupt request	fiq	si	10001
Interrupt request	irq	si	10010
Supervisor	svc	si	10011
System	sys	si	11111
Undefined	und	si	11011
User	usr	no	10000

Puede verse que excepto User el resto de los modos es Privilegiado. Lo cual significa que el kernel de un Sistema Operativo para un sistema basado en procesadores de ARM, se encargará de incluir todo el código que ejecute en estos modos.

Las tareas y procesos por otra parte, ejecutarán en Modo User.



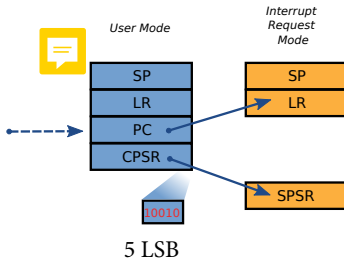
Vector de Interrupciones

Excepción/Interrupción	Mnemónico	Dirección	Dirección Alta
Reset	RESET	0x00000000	0xffff0000
Undefined instruction	UNDEF	0x00000004	0xffff0004
Software interrupt	SWI	0x00000008	0xffff0008
Prefetch abort	PABT	0x0000000c	0xffff000c
Data abort	DABT	0x00000010	0xffff0010
Reserved	-	0x00000014	0xffff0014
Interrupt request	IRQ	0x00000018	0xffff0018
Fast interrupt request	FIQ	0x0000001c	0xffff001c

Interrupt Vector

Reset
Undefined
SWI
Prefetch Abort
Data Abort
Reserved
IRQ
FIQ

esto lo hace el procesador automáticamente. nosotros solo debemos cargar la tabla



- Al generarse la interrupción se salva **CPSR** en el **SPSR** del modo al que se pasa.
- Siempre se usa **CPSR**. Solo cambia CPSR[4:0]
- LR=Dirección de retorno.
- Al saltar desde el Vector de Interrupciones el **PC** toma la dirección destino.

Tabla de vectores

- Por default se ubica en la dirección 0.
- CORTEX A y R permiten ubicarla en 0xFFFF0000. Puede resultar mas práctico cuando se tiene un sistema operativo.
- Cada entrada puede contener una instrucción que genere un branch.
Posibilidades:
 - `b address` (salto relativo al PC)
 - `ldr pc, [pc, #offset]`
 - `mov pc, #immediate` (dato inmediato al PC rotado a derecha)




Prioridad de las Excepciones

Excepción	Prioridad	Bit CPSR[I]	Bit CPSR[F]
Reset	1	1	1
Data abort	2	1	-
Fast interrupt request	3	1	1
Interrupt request	4	1	-
Prefetch abort	5	1	-
Software interrupt	6	1	-
Undefined instruction	6	1	-

- Esta tabla define en que orden se atienden las excepciones en caso de simultaneidad.
- **Reset** tiene máxima preeminencia por sobre las demás.
- En un **Reset** debe inicializarse el sistema completo, habilitar la memoria los caches y armar el vector de excepciones antes de habilitar **IRQ** y **FIQ**.
- Las primeras instrucciones de este handler deben garantizar que no ocurra ninguna excepción. Por eso deben estar deshabilitadas **IRQ** y **FIQ** (Flags **I** y **F** en '1').



Prioridad de las Excepciones

- **Data Abort** es generada por la M  Escenarios mas comunes:
 - Cuando se intenta acceder a una dirección de **memoria inválida** (si no hay memoria física asignada a esa dirección de memoria).
 - Cuando se intenta acceder a una dirección de memoria para la cual **no se dispone de los permisos necesarios**.
- Sin embargo no hay problemas en suspender la atención de un **Data Abort** para vectorizar el handler de una **FIQ**. Eventualmente una vez que termina el handler de **FIQ** se retorna al de **Data Abort** y se continúa con su resolución. FIQ tiene mas prioridad que Data abort
- Decimos que **Data Abort** soporta anidamiento con **FIQ**.



Prioridad de las Excepciones

- **FIQ**, se produce cuando un periférico lleva el terminal **FIQ** del procesador al estado *nFIQ*.
- El core deshabilita **FIQ** e **IRQ** cuando vectoriza **FIQ**.
Para que no entre constantemente
- **IRQ**, se produce cuando un periférico lleva el terminal **IRQ** del procesador al estado *nIRQ*.
- El core deshabilita **IRQ** cuando la vectoriza.

es recomendado deshabilitar las interrupciones cuando se entran a las interrupciones. Vemos que para IRQ y FIQ se deshabilita automáticamente. Cuando se sale del modo IRQ o FIQ se vuelven a habilitar. La idea de FIQ es que se ejecute rápidamente, por eso se deshabilita cualquier interrupción que pueda alargar el manejo de la interrupción.



Prioridad de las Excepciones

- Se genera **Prefetch Abort** cuando se intenta un Fetch de una instrucción en una dirección inválida de memoria.



- Esta excepción se genera cuando la instrucción responsable del Fallo ingresa a la etapa de ejecución del pipeline, siempre que no se hayan activado en el transcurso **FIQ**, y/o **IRQ** (recordar el carácter asincrónico y no determinístico de las interrupciones de hardware). Si esto hubiese ocurrido se atenderá la interrupción o ambas interrupciones (según corresponda) en el orden de prioridad establecido.
- Cuando se vectoriza **Prefetch Abort**, se deshabilita **IRQ**.
- El handler de **Prefetch Abort**, se suspende si ingresa **FIQ**, o **IRQ**, si se habilitó en algún momento durante la ejecución del handler.

entra al modo Abort



Prioridad en las Excepciones

- **SWI** se genera cuando se ejecuta la Instrucción **SWI** o **SVC** (**SWI** sigue implementada pero es considerada obsoleta por ARM desde 2007).
- Esta instrucción se ejecuta en código Modo User.
- Al ingresar al handler, el procesador establece en el registro CPSR el Modo Supervisor.



Prioridad en las Excepciones

- **Undefined Instruction** se genera en la fase de ejecución del pipeline cuando la instrucción no corresponde al set de instrucciones ARM Thumb o Thumb2 y no hay coprocesadores disponibles y si los hay tampoco corresponde a sus instrucciones.
- **Undefined Instruction** y **SWI** nunca pueden por razones obvias ocurrir al mismo tiempo. Por eso tienen la misma prioridad.



Dirección de retorno. Valores del Link Register

Excepción	Dirección	Uso
Reset	-	Es un Reset. No está definido
Data abort	lr - 8	Apunta a la instrucción que causó el Abort
Fast interrupt request	lr - 4	Retorna a la instrucción siguiente a la que recibió FIQ
Interrupt request	lr - 4	Retorna a la instrucción siguiente a la que recibió IRQ
Prefetch abort	lr - 4	Retorna a la instrucción que recibió Prefetch Abort
Software interrupt	lr	Regresa a la instrucción siguiente a SWI
Undefined instruction	lr	Regresa a la instrucción siguiente a la no definida

- Tener presente que todas las excepciones ocurren durante la etapa de ejecución de la instrucción en el pipeline
- Lo mismo las Interrupciones de Hardware. Independientemente de cuando ingresan, se atienden luego de completar la ejecución de la instrucción en curso.
- Como el pipeline es de tres etapas, y el tamaño de ejecución es fijo, en la fase de ejecución en modo ARM el registro **PC**, está posicionado para fetchear dos instrucciones mas adelante.
- Por ello en **LR** se guarda el valor actual del **PC** mas 8.



Ejemplos de Código de retorno para IRQ y FIQ

handler:

```
/* handler code */
```

R14 es lr

```
...
```

```
subs pc, r14, #4 /* pc=r14-4 */
```

esto depende de que interrupcion ocurrio

otra forma:

handler:

```
sub r14, r14, #4 /* r14-=4 */
```

```
...
```

```
/* handler code */
```

```
...
```

```
movs pc, r14 /* return */
```

otra forma:

handler:

```
sub r14, r14, #4 /* r14-=4 */
```

```
stmfd r13!, {r0-r3, r14} /* Almacena contexto. (PUSH) r13=sp. */
... /* FD=Full Descending. Decrement Before.
```

```
*/
```

```
/* handler code */
```

```
...
```

```
ldmfd r13!, {r0-r3, pc} /* Recupera Contexto. (POP) r13=sp. LR->
PC. */
```



Contenido

1 Introducción

- Interrupciones en Microprocesadores
- Generalidades del sistema de excepciones en ARM

2 Sistema de Interrupciones ARMv7, perfiles A y R

- Gestión de Excepciones
- Interrupciones
- SOC SITARA AM335x: Controlador de Interrupciones

3 Interrupciones en CORTEX-M

- Arquitectura ARMv7-M
- el NVIC



Diseño del Sistema de Interrupciones

Guidelines para el diseñador de un sistema embebido

- 1 Seleccionar o diseñar un controlador de interrupciones adecuado para coleccionar las diferentes fuentes de interrupción de los diversos dispositivos de E/S, priorizarlas adecuadamente, y distribuirlas hacia IRQ o FIQ, y manejar un sistema de habilitación / deshabilitación particular para cada una.
 - Las interrupciones de propósito general van a IRQ (Ejemplo un timer que obra como base de tiempos para cambios de tarea)
 - Las interrupciones en donde es crítico el tiempo de respuesta van a FIQ. Ejemplo: DMA.
- 2 Diseñar un sistema de software compuesto por los diferentes handlers de cada dispositivo, acorde con los requerimientos de hardware.

Interrupt Latency

- Si lo que estamos diseñando es un sistema event driven, entonces la madre de todas las batallas se llama “*Interrupt Latency*”.
- *Latency* proviene de Late. Podríamos asemejarlo a retardo.
- Sin embargo en argento básico ... se dice latencia (como si latiera...)
- Nuestro trabajo cuando hay múltiples fuentes de interrupción es encontrar el mejor sistema de gestión de interrupciones que minimice este retardo



Abordajes para mínimo Interrupt Latency

Anidamiento :

Consiste en permitir que una interrupción pueda ser suspendida cuando otra requiere servicio de modo que ésta no deba esperar que se complete la anterior.

Funciona muy bien en sistemas de periféricos de similar complejidad y donde todos juegan con un sistema de prioridades mas o menos uniforme (plano)



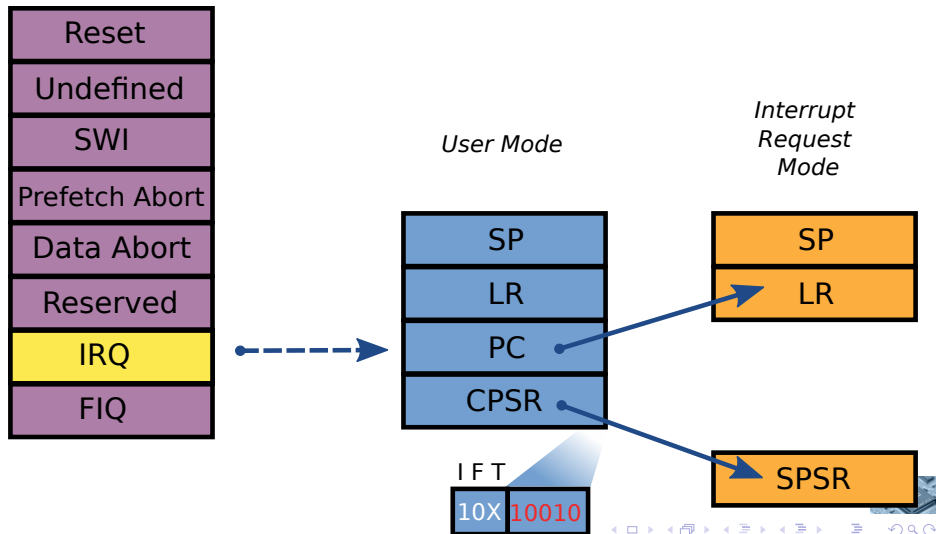
Priorización:

Consiste en asignar una jerarquía basada en prioridades de modo que cada interrupción pueda anidar el procesamiento únicamente de interrupciones de mayor prioridad.

Si los dispositivos tienen igual o menos prioridad no se cursa la nueva interrupción hasta que no finalice la interrupción en curso, así la nueva se encuentre habilitada.

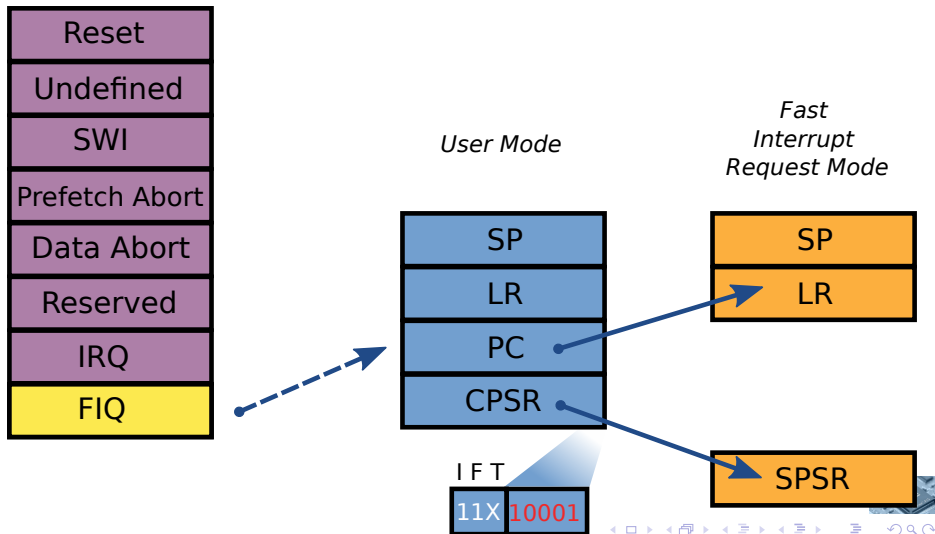
Secuencia de Atención de IRQ

Interrupt Vector



Secuencia de Atención de FIQ

Interrupt Vector



Habilitación de IRQ y FIQ

Para habilitación I=0 y F=0

IRQ_Enable :

```
mrs r1,cpsr /*mrs mover desde registro de sistema o
coprocesador a registro core*/
bic r1,r1,#0x80/* Bit Clear. Limpia el/los bits seteados
en el valor inmediato*/
msr cpsr,r1 /*msr, operac i3n inversa de mrs*/
```

FIQ_Enable

```
mrs r1,cpsr
bic r1,r1,#0x40
msr cpsr,r1
```



Habilitación - Deshabilitación de IRQ y FIQ

Código para deshabilitación

IRQ_Enable :

```
mrs r1,cpsr
orr r1,r1,#0x80
msr cpsr,r1
```

FIQ_Enable

```
mrs r1,cpsr
orr r1,r1,#0x40
msr cpsr,r1
```



Contenido

1 Introducción

- Interrupciones en Microprocesadores
- Generalidades del sistema de excepciones en ARM

2 Sistema de Interrupciones ARMv7, perfiles A y R

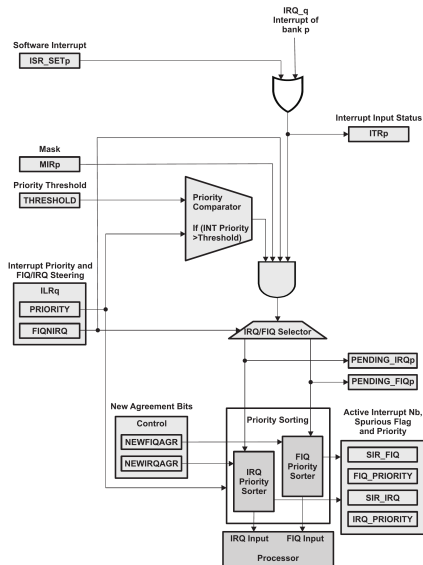
- Gestión de Excepciones
- Interrupciones
- SOC SITARA AM335x: Controlador de Interrupciones

3 Interrupciones en CORTEX-M

- Arquitectura ARMv7-M
- el NVIC



Ejemplo práctico de controlador de Interrupciones



Contenido

1 Introducción

2 Sistema de Interrupciones ARMv7, perfiles A y R

3 Interrupciones en CORTEX-M

- Interrupciones en Microprocesadores
 - Generalidades del sistema de excepciones en ARM
- Gestión de Excepciones
 - Interrupciones
 - SOC SITARA AM335x: Controlador de Interrupciones
- Arquitectura ARMv7-M
 - el NVIC



Características y diferencias con ARMv7



Características y diferencias con ARMv7

- Mercado Objetivo: microcontroladores



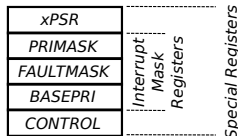
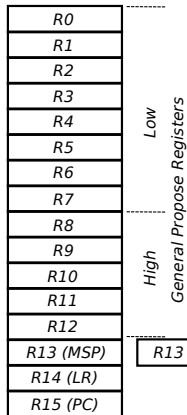
Características y diferencias con ARMv7

AHORA ESTA EN M

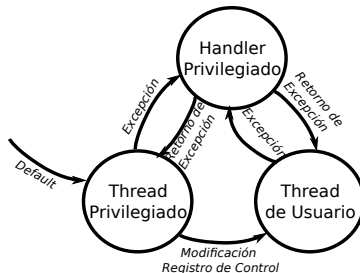
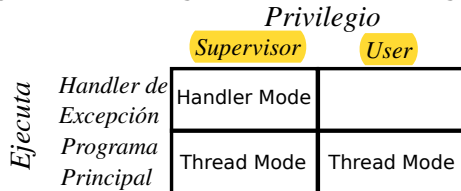
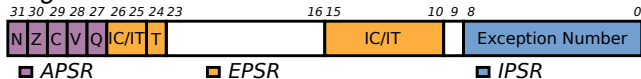
- Mercado Objetivo: microcontroladores
 - Facilidad de programación (se puede trabajar en C el 100 % del proyecto).
 - Muy pocas prestaciones comparativamente con las posibilidades de la arquitectura original.
- Cambios significativos en la ISA respecto del la ARMv7
 - No trabaja instrucciones en Modo ARM
 - Un solo set de registros, y solo dos modos de ejecución: Modo Thread y Modo Handler.
 - Aparece un segundo Registro de Estados, el **xPSR** cuyo bitmap no coincide con CPSR.
- Sistema de interrupciones completamente diferente.
 - El Vector de interrupciones incluye direcciones en lugar de instrucciones de salto.
 - Las excepciones salvan automáticamente en la pila los registros **r0-r3, r12, lr (r14), xPSR, y pc (r15)**.



Registros y Modos de operación



Registro xPSR



Cambios en el set de registros

- Como solo hay dos modos de operación los CORTEX-M tienen un solo banco de registros. **cada nivel de privilegio tiene su propia pila**
- Solo se bancaea en Stack Pointer (**r13**), para poder conmutar pila cuando se cambia de Modo Thread a Modo Handler.

MSP **Main Stack** Pointer. Utilizado por el Sistema Operativo y las interrupciones

PSP **Process Stack Pointer**. Utilizado por las aplicaciones.

- Registros Especiales en el core (no memory mapped):
 - Registros especiales de máscara: **PRIMASK**, **FAULTMASK**, **BASEPRI**.
 - Dos nuevos registros de Control de propósito especial.
 - Se acceden por medio de instrucciones especiales.

```
MRS <reg>, <special_reg> /* Lee Registro especial.*/  
MSR <special_reg>, <reg> /* Escribe Registro especial*/
```



Registros especiales del core

- Program Status Register. No se lo accede de manera explícita. Se lo salva en la pila (no está banqueado)
- El CPSR se descompone en tres registros de estado:
 - APSR Application Program Status Register.
 - IPSR Interrupt Program Status Register.
 - EPSR Exception Program Status Register.
- Si se acceden en forma colectiva como un solo ítem se los llama **xPSR**.

Bit	Descripción
N	Negative
Z	Zero
C	Carry/borrow
V	Overflow
Q	Sticky saturation flag
ICI/IT	Interrupt-Continuable Instruction (ICI) bits, IF-THEN instruction status bit
T	Thumb state, 1 siempre; limpiar este bit genera un fault exception.
Exception Number	Handler de excepción que se está ejecutando

PRIMASK, FAULTMASK, y BASEPRI

Nombre	Descripción
PRIMASK	Registro de 1-bit. Si es '1' habilita NMI y Hard Fault exception. El resto de las excepciones e interrupciones están deshabilitadas. Si es '0' (default), no aplica máscara alguna.
FAULTMASK	Registro de 1-bit. Si es '1', habilita solo a NMI. El resto de las interrupciones y faltas está deshabilitado. Si es '0' (default), no aplica máscara alguna.
BASEPRI	Registro de hasta 9 bits (dependiendo de ancho de bits implementado en el nivel de prioridades del NVIC). Define el nivel de prioridad de máscaras. Si es '1', deshabilita todas las interrupciones de igual o menor nivel (valor de prioridad mayor). Las interrupciones de mas alta proridad permanecen habilitadas. En '0' (default), no aplica máscara alguna.



PRIMASK, FAULTMASK, y BASEPRI

Nombre	Descripción
PRIMASK	Registro de 1-bit. Si es '1' habilita NMI y Hard Fault exception. El resto de las excepciones e interrupciones están deshabilitadas. Si es '0' (default), no aplica máscara alguna.
FAULTMASK	Registro de 1-bit. Si es '1', habilita solo a NMI. El resto de las interrupciones y faltas está deshabilitado. Si es '0' (default), no aplica máscara alguna.
BASEPRI	Registro de hasta 9 bits (dependiendo de ancho de bits implementado en el nivel de prioridades del NVIC). Define el nivel de prioridad de máscaras. Si es '1', deshabilita todas las interrupciones de igual o menor nivel (valor de prioridad mayor). Las interrupciones de mas alta proridad permanecen habilitadas. En '0' (default), no aplica máscara alguna.

FAULTMASK

Es aprovechada por el sistema operativo cuando maneja una situación de crash de una tarea. En este escenario, es imprescindible impedir que se produzcan nuevas faltas debidas al crash de la tarea mientras el sistema operativo la está limpiando.

Conclusión: La arquitectura no soporta el escenario de doble falta.



Registro de control

Nombre	Función
CONTROL[1]	Stack status: En modo Thread si es '1' utiliza stack alternativo (PSP), y si es '0' utiliza el Default Stack (MSP). En Modo Handler no hay stack alternativo, por lo tanto debe ser '0'.
CONTROL[0]	'0' Estado Privilegiado en Thread mode. '1' Estado User en Thread mode. En Modo Handler el procesador opera en Modo Privilegiado indefectiblemente (es kernel).



Registro de control

Nombre	Función
CONTROL[1]	Stack status: En modo Thread si es '1' utiliza stack alternativo (PSP), y si es '0' utiliza el Default Stack (MSP). En Modo Handler no hay stack alternativo, por lo tanto debe ser '0'.
CONTROL[0]	'0' Estado Privilegiado en Thread mode. '1' Estado User en Thread mode. En Modo Handler el procesador opera en Modo Privilegiado indefectiblemente (es kernel).

- **CONTROL[1]**: Es siempre '0' en el Modo Handler. Su estado es relevante en Modo Thread, en el que solo puede escribirse si el core está en Estado Privilegiado. En Modo Handler o en Modo Thread Estado User no está permitido escribir este bit.



Registro de control

Nombre	Función
CONTROL[1]	Stack status: En modo Thread si es '1' utiliza stack alternativo (PSP), y si es '0' utiliza el Default Stack (MSP). En Modo Handler no hay stack alternativo, por lo tanto debe ser '0'.
CONTROL[0]	'0' Estado Privilegiado en Thread mode. '1' Estado User en Thread mode. En Modo Handler el procesador opera en Modo Privilegiado indefectiblemente (es kernel).

- **CONTROL[1]**: Es siempre '0' en el Modo Handler. Su estado es relevante en Modo Thread, en el que solo puede escribirse si el core está en Estado Privilegiado. En Modo Handler o en Modo Thread Estado User no está permitido escribir este bit.
- **CONTROL[0]**: Solo se puede modificar en estado Privilegiado. Una vez que ingresa al estado User, la única forma de volver al estado Privilegiado, es que a expensas de una interrupción en Modo Handler se escriba este bit.



Contenido

1 Introducción

- Interrupciones en Microprocesadores
- Generalidades del sistema de excepciones en ARM

2 Sistema de Interrupciones ARMv7, perfiles A y R

- Gestión de Excepciones
- Interrupciones
- SOC SITARA AM335x: Controlador de Interrupciones

3 Interrupciones en CORTEX-M

- Arquitectura ARMv7-M
- el NVIC



Sistema de interrupciones CORTEX-M



Sistema de interrupciones CORTEX-M

- Los ARMv7-M tienen 15 excepciones (que se numeran de 1 a 15), y 240 interrupciones externas (se numeran de 0 a 239).



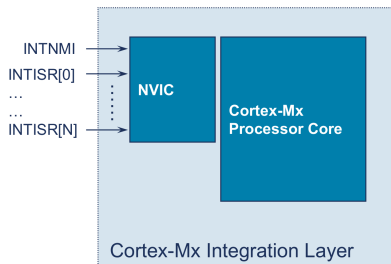
Sistema de interrupciones CORTEX-M

- Los ARMv7-M tienen 15 excepciones (que se numeran de 1 a 15), y 240 interrupciones externas (se numeran de 0 a 239).
- ARMv7-M incluye un controlador de interrupciones predeterminado, el **NVIC** (Nested Virtualized Interrupt Controller).



Sistema de interrupciones CORTEX-M

- Los ARMv7-M tienen 15 excepciones (que se numeran de 1 a 15), y 240 interrupciones externas (se numeran de 0 a 239).
- ARMv7-M incluye un controlador de interrupciones predeterminado, el **NVIC** (Nested Virtualized Interrupt Controller).

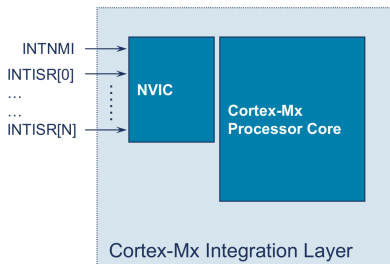


- A diferencia de CORTEX-A y R que no incluyen más que el core, los CORTEX-M incluyen un controlador de interrupciones.



Sistema de interrupciones CORTEX-M

- Los ARMv7-M tienen 15 excepciones (que se numeran de 1 a 15), y 240 interrupciones externas (se numeran de 0 a 239).
- ARMv7-M incluye un controlador de interrupciones predeterminado, el **NVIC** (Nested Virtualized Interrupt Controller).



- A diferencia de CORTEX-A y R que no incluyen más que el core, los CORTEX-M incluyen un controlador de interrupciones.
- El número de la excepción que se está atendiendo se mantiene en el registro especial **IPSR**, o en el campo de bits **VECTACTIVE** del Interrupt Control State Register del **NVIC**.



Lista de Excepciones Cortex-M

Nro	Tipo	Prioridad	Descripción
1	Reset	-3	Reset
2	NMI	-2	1 Non Maskable Interrupt (externa)
3	Hard Fault	-1	Todas las condiciones de falta cuyo handler no esté habilitado.
4	MemManage Fault	Programable	Fallo de Manejo de Memoria. Violación de la MPU o acceso a direcciones no permitidas.
5	Bus Fault	Programable	Error de bus. Cuando el bus ABH recibe un error de un slave (corresponde a Prefetch Abort y Data Abort).
6	Usage Fault	Programable	Excepciones en el código de un programa o en el acceso a un coprocesador.
7-10	Reservados	NA	-
11	SVCall	Programable	Llamada a Servicios del Sistema.
12	Debug Monitor	Programable	Debug Monitor (breakpoints, watchpoints, Requerimientos externos al debugger)
13	Reservado	NA	-
14	PendSV	Programable	Requerimiento pendiente de un system device
15	SYSTICK	Programable	System Tick Timer
16	External INT#0	Programable	-
17	External INT#1	Programable	-
...
255	External INT#239	Programable	-

tiene la opción de agregarle mas de 200 excepciones.



Sistema de interrupciones CORTEX-M



Sistema de interrupciones CORTEX-M

- El NVIC se introduce en el CORTEX-M3, por lo tanto los números de interrupción externa del NVIC coinciden con los números de excepción.



Sistema de interrupciones CORTEX-M

- El NVIC se introduce en el CORTEX-M3, por lo tanto los números de interrupción externa del NVIC coinciden con los números de excepción.
- Sin embargo en la medida en que la familia CORTEX-M evolucionó, los nuevos procesadores fueron incorporando nuevos periféricos internos que empezaron a consumir las primeras líneas del NVIC.



Sistema de interrupciones CORTEX-M

- El NVIC se introduce en el CORTEX-M3, por lo tanto los números de interrupción externa del NVIC coinciden con los números de excepción.
- Sin embargo en la medida en que la familia CORTEX-M evolucionó, los nuevos procesadores fueron incorporando nuevos periféricos internos que empezaron a consumir las primeras líneas del NVIC.
- Conclusión: la correspondencia de los terminales de interrupción de un CORTEX-M con respecto a los números de excepción externa depende finalmente de cada versión de SoC.



Sistema de interrupciones CORTEX-M



Sistema de interrupciones CORTEX-M

- En los microcontroladores previos al CORTEX-M3 cada dispositivo que envía una señal de interrupción la debe sostener hasta que ésta sea atendida (líneas activas por nivel).



Sistema de interrupciones CORTEX-M

- En los microcontroladores previos al CORTEX-M3 cada dispositivo que envía una señal de interrupción la debe sostener hasta que ésta sea atendida (líneas activas por nivel).
- A partir del CORTEX-M3 si una interrupción no puede ser atendida debido a que el core está procesando en este momento una interrupción de mayor prioridad, quedará pendiente (excepto algunas de las excepciones) y su requerimiento se mantiene en el Pending Status Register.



Sistema de interrupciones CORTEX-M

- En los microcontroladores previos al CORTEX-M3 cada dispositivo que envía una señal de interrupción la debe sostener hasta que ésta sea atendida (líneas activas por nivel).
- A partir del CORTEX-M3 si una interrupción no puede ser atendida debido a que el core está procesando en este momento una interrupción de mayor prioridad, quedará pendiente (excepto algunas de las excepciones) y su requerimiento se mantiene en el Pending Status Register.
- Esto permite liberar a los periféricos de la responsabilidad de sostener el nivel de la señal de interrupción cuando esta es demorada en su atención.



Prioridades de Interrupciones en CORTEX-M



Prioridades de Interrupciones en CORTEX-M

- Indican cual interrupción debe ser atendida primero y cuando.



Prioridades de Interrupciones en CORTEX-M

- Indican cual interrupción debe ser atendida primero y cuando.
- A menor valor numérico mayor prioridad.



Prioridades de Interrupciones en CORTEX-M

- Indican cual interrupción debe ser atendida primero y cuando.
- A menor valor numérico mayor prioridad.
- RESET, NMI, Hard Fault, en ese orden son las de mayor prioridad.



Prioridades de Interrupciones en CORTEX-M

- Indican cual interrupción debe ser atendida primero y cuando.
- A menor valor numérico mayor prioridad.
- RESET, NMI, Hard Fault, en ese orden son las de mayor prioridad.
- Para el resto cada versión de CORTEX-M dispone de registros en los que se puede programar prioridades por grupos de líneas.



Prioridades de Interrupciones en CORTEX-M

- Indican cual interrupción debe ser atendida primero y cuando.
- A menor valor numérico mayor prioridad.
- RESET, NMI, Hard Fault, en ese orden son las de mayor prioridad.
- Para el resto cada versión de CORTEX-M dispone de registros en los que se puede programar prioridades por grupos de líneas.
- Las variantes dependen de la cantidad de bits mas significativos de cada registro.



Definición de prioridades



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***
- Implica que una interrupción de mayor prioridad se adelanta (***preemp***) siempre a una de menor prioridad.



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***
- Implica que una interrupción de mayor prioridad se adelanta (***preemp***) siempre a una de menor prioridad.
- Ya vimos cuales son las de prioridad fija. Queda por ver como asignar prioridades a las restantes (Número 4 a 255)



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***
- Implica que una interrupción de mayor prioridad se adelanta (***preemp***) siempre a una de menor prioridad.
- Ya vimos cuales son las de prioridad fija. Queda por ver como asignar prioridades a las restantes (Número 4 a 255)
- CORTEX-M soporta hasta 256 niveles de prioridad programables, y hasta 128 niveles de “preemption”.



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***
- Implica que una interrupción de mayor prioridad se adelanta (***preempt***) siempre a una de menor prioridad.
- Ya vimos cuales son las de prioridad fija. Queda por ver como asignar prioridades a las restantes (Número 4 a 255)
- CORTEX-M soporta hasta 256 niveles de prioridad programables, y hasta 128 niveles de “preemption”.
- La cantidad de niveles de prioridad que pueden programarse, termina dependiendo de la implementación.



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***
- Implica que una interrupción de mayor prioridad se adelanta (***preemp***) siempre a una de menor prioridad.
- Ya vimos cuales son las de prioridad fija. Queda por ver como asignar prioridades a las restantes (Número 4 a 255)
- CORTEX-M soporta hasta 256 niveles de prioridad programables, y hasta 128 niveles de “preemption”.
- La cantidad de niveles de prioridad que pueden programarse, termina dependiendo de la implementación.
- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.



Definición de prioridades

- El escenario de anidamiento de estos procesadores se basa en un concepto denominado ***preemption***
- Implica que una interrupción de mayor prioridad se adelanta (***preemp***) siempre a una de menor prioridad.
- Ya vimos cuales son las de prioridad fija. Queda por ver como asignar prioridades a las restantes (Número 4 a 255)
- CORTEX-M soporta hasta 256 niveles de prioridad programables, y hasta 128 niveles de “preemption”.
- La cantidad de niveles de prioridad que pueden programarse, termina dependiendo de la implementación.
- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.
- Invocando al API también provista por el fabricante la prioridad se establece independientemente del layout de los bits implementados.



Definición de prioridades



Definición de prioridades

- La especificación establece 60 registros para programación de Prioridad a cada tipo de interrupción.



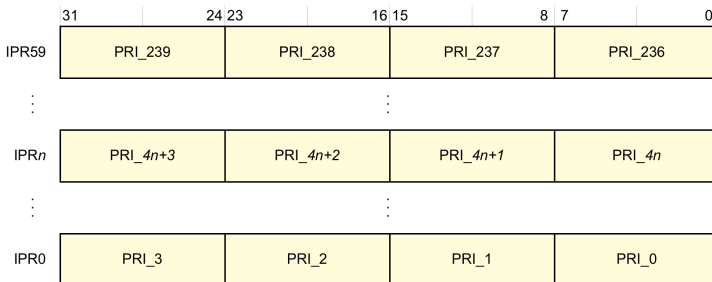
Definición de prioridades

- La especificación establece 60 registros para programación de Prioridad a cada tipo de interrupción.
- El layout es el siguiente



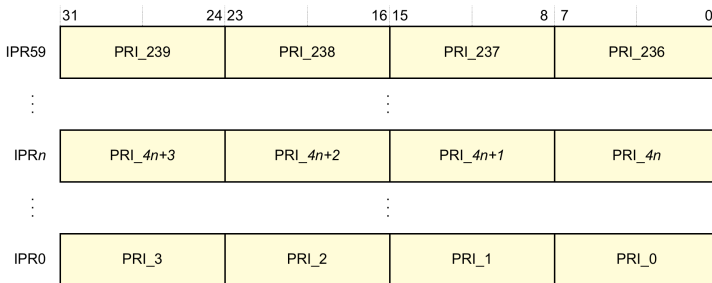
Definición de prioridades

- La especificación establece 60 registros para programación de Prioridad a cada tipo de interrupción.
- El layout es el siguiente



Definición de prioridades

- La especificación establece 60 registros para programación de Prioridad a cada tipo de interrupción.
- El layout es el siguiente

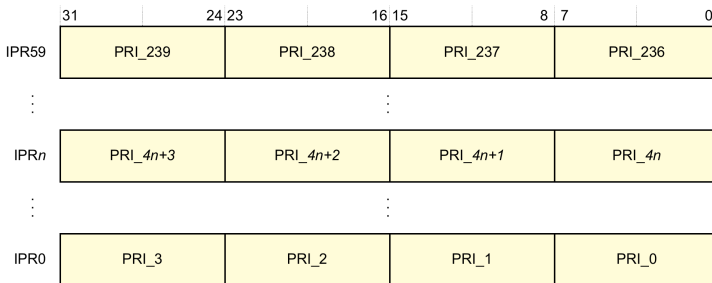


- Cada Registro setea la prioridad de 4 líneas de interrupción externas. El acceso para INT#n es de acuerdo a:



Definición de prioridades

- La especificación establece 60 registros para programación de Prioridad a cada tipo de interrupción.
- El layout es el siguiente



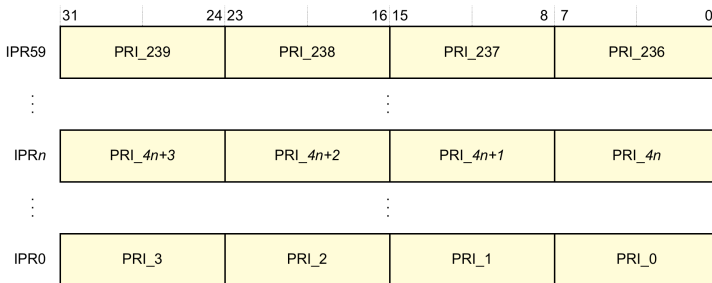
- Cada Registro setea la prioridad de 4 líneas de interrupción externas. El acceso para INT#n es de acuerdo a:

$$IPR_i = INT\#n / 4$$



Definición de prioridades

- La especificación establece 60 registros para programación de Prioridad a cada tipo de interrupción.
- El layout es el siguiente



- Cada Registro setea la prioridad de 4 líneas de interrupción externas. El acceso para INT#n es de acuerdo a:

$$IPR_i = INT\#n / 4$$

$$byte\#n = IPR_i \bmod 4$$



Definición de prioridades



Definición de prioridades

- En cada byte dentro de cada registro de Prioridad generalmente no se utilizan los 8 bits sino una cantidad de bits mas significativos, y los menos significativos se dejan en cero.



Definición de prioridades

- En cada byte dentro de cada registro de Prioridad generalmente no se utilizan los 8 bits sino una cantidad de bits mas significativos, y los menos significativos se dejan en cero.
- De este modo no se tiene en todos los dispositivos los 256 niveles de prioridad diferentes.



Definición de prioridades

- En cada byte dentro de cada registro de Prioridad generalmente no se utilizan los 8 bits sino una cantidad de bits mas significativos, y los menos significativos se dejan en cero.
- De este modo no se tiene en todos los dispositivos los 256 niveles de prioridad diferentes.
- La cantidad de bits empleados depende del SoC, es decir del fabricante.



Definición de prioridades

- En cada byte dentro de cada registro de Prioridad generalmente no se utilizan los 8 bits sino una cantidad de bits mas significativos, y los menos significativos se dejan en cero.
- De este modo no se tiene en todos los dispositivos los 256 niveles de prioridad diferentes.
- La cantidad de bits empleados depende del SoC, es decir del fabricante.
 - Si se utilizan los bits 7-5, se tiene solo 8 niveles de prioridad: 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, y 0xE0.



Definición de prioridades

- En cada byte dentro de cada registro de Prioridad generalmente no se utilizan los 8 bits sino una cantidad de bits mas significativos, y los menos significativos se dejan en cero.
- De este modo no se tiene en todos los dispositivos los 256 niveles de prioridad diferentes.
- La cantidad de bits empleados depende del SoC, es decir del fabricante.
 - Si se utilizan los bits 7-5, se tiene solo 8 niveles de prioridad: 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, y 0xE0.
 - Si se utilizan los bits 7-4, se tiene 16 niveles de prioridad: 0x00, 0x10, 0x20, 0x30, 0x40, ... etc... 0xE0, y 0xF0.



Definición de prioridades

- En cada byte dentro de cada registro de Prioridad generalmente no se utilizan los 8 bits sino una cantidad de bits mas significativos, y los menos significativos se dejan en cero.
- De este modo no se tiene en todos los dispositivos los 256 niveles de prioridad diferentes.
- La cantidad de bits empleados depende del SoC, es decir del fabricante.
 - Si se utilizan los bits 7-5, se tiene solo 8 niveles de prioridad: 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, y 0xE0.
 - Si se utilizan los bits 7-4, se tiene 16 niveles de prioridad: 0x00, 0x10, 0x20, 0x30, 0x40, ... etc... 0xE0, y 0xF0.
- Por este motivo cada fabricante provee una solución bastante transparente, basada en la especificación de ARM CMSIS.



Definición de prioridades



Definición de prioridades

- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.



Definición de prioridades

- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.
- El API también provista por el fabricante **CMSIS** (Cortex Microcontroller Software Interface Standard), incluye funciones que permiten operar los registros de prioridad de forma transparente al layout de los bits implementados.



Definición de prioridades

- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.
- El API también provista por el fabricante **CMSIS** (Cortex Microcontroller Software Interface Standard), incluye funciones que permiten operar los registros de prioridad de forma transparente al layout de los bits implementados.

```
__STATIC_INLINE void NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)
{
    if (IRQn < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = (SCB->SHP[_SHP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
    else {
        NVIC->IP[_IP_IDX(IRQn)] = (NVIC->IP[_IP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
}
```



Definición de prioridades

- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.
- El API también provista por el fabricante **CMSIS** (Cortex Microcontroller Software Interface Standard), incluye funciones que permiten operar los registros de prioridad de forma transparente al layout de los bits implementados.

```
__STATIC_INLINE void NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)
{
    if (IRQn < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = (SCB->SHP[_SHP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
    else {
        NVIC->IP[_IP_IDX(IRQn)] = (NVIC->IP[_IP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
}
```

- En el caso de una UART por ejemplo se invoca del siguiente modo:



Definición de prioridades

- El fabricante provee la macro `__NVIC_PRIO_BITS` dentro del código empaquetado para permitir el manejo en C del dispositivo.
- El API también provista por el fabricante **CMSIS** (Cortex Microcontroller Software Interface Standard), incluye funciones que permiten operar los registros de prioridad de forma transparente al layout de los bits implementados.

```
__STATIC_INLINE void NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)
{
    if (IRQn < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = (SCB->SHP[_SHP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
    else {
        NVIC->IP[_IP_IDX(IRQn)] = (NVIC->IP[_IP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
}
```

- En el caso de una UART por ejemplo se invoca del siguiente modo:

```
NVIC_SetPriority(UART0_IRQn, (1 << __NVIC_PRIO_BITS) - 1);
```



Mapa de Registros del NVIC

Address	Name	Type	Required privilege	Reset value
0xE000E100- 0xE000E11C	NVIC_ISER0- NVIC_ISER7	RW	Privileged	0x00000000
0xE000E180- 0xE000E19C	NVIC_ICER0- NVIC_ICER7	RW	Privileged	0x00000000
0xE000E200- 0xE000E21C	NVIC_ISPR0- NVIC_ISPR7	RW	Privileged	0x00000000
0xE000E280- 0xE000E29C	NVIC_ICPR0- NVIC_ICPR7	RW	Privileged	0x00000000
0xE000E300- 0xE000E31C	NVIC_IABR0- NVIC_IABR7	RW	Privileged	0x00000000
0xE000E400- 0xE000E4EF	NVIC_IPR0- NVIC_IPR59	RW	Privileged	0x00000000
0xE000EF00	STIR	WO	Configurable ^a	0x00000000



CMSIS: Funciones de acceso a los registros

CMSIS function	Description
<code>void NVIC_EnableIRQ(IRQn_Type IRQn)^a</code>	Enables an interrupt or exception.
<code>void NVIC_DisableIRQ(IRQn_Type IRQn)^a</code>	Disables an interrupt or exception.
<code>void NVIC_SetPendingIRQ(IRQn_Type IRQn)^a</code>	Sets the pending status of interrupt or exception to 1.
<code>void NVIC_ClearPendingIRQ(IRQn_Type IRQn)^a</code>	Clears the pending status of interrupt or exception to 0.
<code>uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)^a</code>	Reads the pending status of interrupt or exception. This function returns non-zero value if the pending status is set to 1.
<code>void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)^a</code>	Sets the priority of an interrupt or exception with configurable priority level to 1.
<code>uint32_t NVIC_GetPriority(IRQn_Type IRQn)^a</code>	Reads the priority of an interrupt or exception with configurable priority level. This function return the current priority level.



Habilitación deshabilitación



Habilitación deshabilitación

- **NVIC_ISER0 – NVIC_ISER7**: Interrupt Set Enable Registers, son registros en los que cada bit obra como máscara de una interrupción interna para su habilitación. Se activa escribiendo '1'. Escribir '0' no tiene efecto.



Habilitación deshabilitación

- **NVIC_ISER0 – NVIC_ISER7**: Interrupt Set Enable Registers, son registros en los que cada bit obra como máscara de una interrupción interna para su habilitación. Se activa escribiendo '1'. Escribir '0' no tiene efecto.
- **NVIC_ICER0 – NVIC_ICER7**: Interrupt Clear Enable Registers, cumplen la misma función que los anteriores pero para deshabilitar interrupciones individualmente. Se desactiva una interrupción escribiendo '1' en el bit correspondiente, y no tiene efecto escribir '0'.



Habilitación deshabilitación

- **NVIC_ISER0 - NVIC_ISER7**: Interrupt Set Enable Registers, son registros en los que cada bit obra como máscara de una interrupción interna para su habilitación. Se activa escribiendo '1'. Escribir '0' no tiene efecto.
- **NVIC_ICER0 - NVIC_ICER7**: Interrupt Clear Enable Registers, cumplen la misma función que los anteriores pero para deshabilitar interrupciones individualmente. Se desactiva una interrupción escribiendo '1' en el bit correspondiente, y no tiene efecto escribir '0'.

Cuando se leen se obtiene el estado de máscara de las 32 líneas externas de interrupción controladas por el registro: '0' INT#n deshabilitada, '1' INT#n habilitada



Manejo de pendientes



Manejo de pendientes

- **NVIC_ISPR0 – NVIC_ISPR7**: Interrupt Set Pending Registers, tiene por objeto setear una línea de interrupción como pendiente, escribiendo '1' en el bit correspondiente del registro correspondiente. Escribir '0' no tiene efecto alguno.



Manejo de pendientes

- **NVIC_ISPR0 – NVIC_ISPR7**: Interrupt Set Pending Registers, tiene por objeto setear una línea de interrupción como pendiente, escribiendo '1' en el bit correspondiente del registro correspondiente. Escribir '0' no tiene efecto alguno.
- **NVIC_ICPR0 – NVIC_ICPR7**: Interrupt Clear Pending Registers, tiene la misma función que el grupo de registros anterior pero para limpiar el estado pendiente de una línea de interrupción. También se limpia el estado escribiendo '1' y no tiene efecto escribir '0'.



Manejo de pendientes

- **NVIC_ISPR0 – NVIC_ISPR7**: Interrupt Set Pending Registers, tiene por objeto setear una línea de interrupción como pendiente, escribiendo '1' en el bit correspondiente del registro correspondiente. Escribir '0' no tiene efecto alguno.
- **NVIC_ICPR0 – NVIC_ICPR7**: Interrupt Clear Pending Registers, tiene la misma función que el grupo de registros anterior pero para limpiar el estado pendiente de una línea de interrupción. También se limpia el estado escribiendo '1' y no tiene efecto escribir '0'.

Cuando se leen se obtiene el estado de pendiente de las 32 líneas externas de interrupción controladas por el registro: '0' INT#n no pendiente, '1' INT#n pendiente



Líneas de Interrupción activas



Líneas de Interrupción activas

- **NVIC_IABR0 – NVIC_IABR7**: Interrupt Active Bit Registers, sirven para indicar cuales son las líneas de Interrupción que están activas.



Líneas de Interrupción activas

- **NVIC_IABR0 – NVIC_IABR7**: Interrupt Active Bit Registers, sirven para indicar cuales son las líneas de Interrupción que están activas.
- Es muy importante definir este estado ya que de acuerdo al fabricante del SoC puede haber mas o menos dispositivos conectados y mas o menos interrupciones externas útiles por lo tanto.



Líneas de Interrupción activas

- **NVIC_IABR0 – NVIC_IABR7**: Interrupt Active Bit Registers, sirven para indicar cuales son las líneas de Interrupción que están activas.
- Es muy importante definir este estado ya que de acuerdo al fabricante del SoC puede haber mas o menos dispositivos conectados y mas o menos interrupciones externas útiles por lo tanto.
- '1' indica Línea activa, '0' indica línea inactiva.



Generando Interrupciones por Software



Generando Interrupciones por Software

- **STIR**: Software Trigger Interrupt Register



Generando Interrupciones por Software

- **STIR**: Software Trigger Interrupt Register
- Es un registro Write Only. Solo es relevante el byte menos significativo. En él se escribe el valor de **INTid** (0 a 239), para identificar el número de interrupción que se cursará por software al procesador.



Generando Interrupciones por Software

- **STIR**: Software Trigger Interrupt Register
- Es un registro Write Only. Solo es relevante el byte menos significativo. En él se escribe el valor de **INTid** (0 a 239), para identificar el número de interrupción que se cursará por software al procesador.
- Terminada la escritura se dispara la interrupción con el número correspondiente.

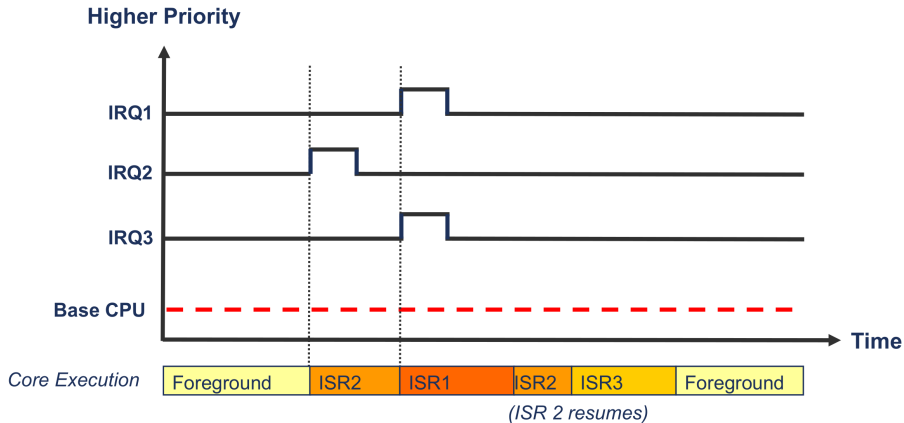


Generando Interrupciones por Software

- **STIR**: Software Trigger Interrupt Register
- Es un registro Write Only. Solo es relevante el byte menos significativo. En él se escribe el valor de **INTid** (0 a 239), para identificar el número de interrupción que se cursará por software al procesador.
- Terminada la escritura se dispara la interrupción con el número correspondiente.
- Por ejemplo: Si **INTid** = 3, entonces se genera IRQ3



Preemption en el manejo de excepciones



Ejemplo de tres Excepciones



Tabla de vectores

Address		Vector #
0x40 + 4*N	External N	16 + N
...
0x40	External 0	16
0x3C	SysTick	15
0x38	PendSV	14
0x34	Reserved	13
0x30	Debug Monitor	12
0x2C	SVC	11
0x1C to 0x28	Reserved (x4)	7-10
0x18	Usage Fault	6
0x14	Bus Fault	5
0x10	Mem Manage Fault	4
0x0C	Hard Fault	3
0x08	NMI	2
0x04	Reset	1
0x00	Initial Main SP	N/A



Tabla de vectores

Address		Vector #
0x40 + 4*N	External N	16 + N
...
0x40	External 0	16
0x3C	SysTick	15
0x38	PendSV	14
0x34	Reserved	13
0x30	Debug Monitor	12
0x2C	SVC	11
0x1C to 0x28	Reserved (x4)	7-10
0x18	Usage Fault	6
0x14	Bus Fault	5
0x10	Mem Manage Fault	4
0x0C	Hard Fault	3
0x08	NMI	2
0x04	Reset	1
0x00	Initial Main SP	N/A

- Es una tabla de direcciones.



Tabla de vectores

Address		Vector #
0x40 + 4*N	External N	16 + N
...
0x40	External 0	16
0x3C	SysTick	15
0x38	PendSV	14
0x34	Reserved	13
0x30	Debug Monitor	12
0x2C	SVC	11
0x1C to 0x28	Reserved (x4)	7-10
0x18	Usage Fault	6
0x14	Bus Fault	5
0x10	Mem Manage Fault	4
0x0C	Hard Fault	3
0x08	NMI	2
0x04	Reset	1
0x00	Initial Main SP	N/A

- Es una tabla de direcciones.
- Entrada 0 = MSP inicial.



Tabla de vectores

Address		Vector #
0x40 + 4*N	External N	16 + N
...
0x40	External 0	16
0x3C	SysTick	15
0x38	PendSV	14
0x34	Reserved	13
0x30	Debug Monitor	12
0x2C	SVC	11
0x1C to 0x28	Reserved (x4)	7-10
0x18	Usage Fault	6
0x14	Bus Fault	5
0x10	Mem Manage Fault	4
0x0C	Hard Fault	3
0x08	NMI	2
0x04	Reset	1
0x00	Initial Main SP	N/A

- Es una tabla de direcciones.
- Entrada 0 = MSP inicial.
- En RESET se carga MSP de allí.



Tabla de vectores

Address		Vector #
0x40 + 4*N	External N	16 + N
...
0x40	External 0	16
0x3C	SysTick	15
0x38	PendSV	14
0x34	Reserved	13
0x30	Debug Monitor	12
0x2C	SVC	11
0x1C to 0x28	Reserved (x4)	7-10
0x18	Usage Fault	6
0x14	Bus Fault	5
0x10	Mem Manage Fault	4
0x0C	Hard Fault	3
0x08	NMI	2
0x04	Reset	1
0x00	Initial Main SP	N/A

- Es una tabla de direcciones.
- Entrada 0 = MSP inicial.
- En RESET se carga MSP de allí.
- Resto de la tabla puede reallocarse.



conclusiones



conclusiones

- Tengo la sensación de haber expuesto dos arquitecturas diferentes.



conclusiones

- Tengo la sensación de haber expuesto dos arquitecturas diferentes.
- En lo referente a interrupciones CORTEX-M parece otro procesador.



conclusiones

- Tengo la sensación de haber expuesto dos arquitecturas diferentes.
- En lo referente a interrupciones CORTEX-M parece otro procesador.
- Evidentemente CORTEX-M se orienta a desarrollo de aplicaciones relativamente simples con un time to market muy corto



conclusiones

- Tengo la sensación de haber expuesto dos arquitecturas diferentes.
- En lo referente a interrupciones CORTEX-M parece otro procesador.
- Evidentemente CORTEX-M se orienta a desarrollo de aplicaciones relativamente simples con un time to market muy corto
- Por eso trae una cantidad de cuestiones resueltas.



conclusiones

- Tengo la sensación de haber expuesto dos arquitecturas diferentes.
- En lo referente a interrupciones CORTEX-M parece otro procesador.
- Evidentemente CORTEX-M se orienta a desarrollo de aplicaciones relativamente simples con un time to market muy corto
- Por eso trae una cantidad de cuestiones resueltas.
- Justamente las mas relevantes para aprender arquitectura de un procesador.



conclusiones

- Tengo la sensación de haber expuesto dos arquitecturas diferentes.
- En lo referente a interrupciones CORTEX-M parece otro procesador.
- Evidentemente CORTEX-M se orienta a desarrollo de aplicaciones relativamente simples con un time to market muy corto
- Por eso trae una cantidad de cuestiones resueltas.
- Justamente las mas relevantes para aprender arquitectura de un procesador.
- Por tal motivo estamos hemos incluido CORTEX-A en la cual haremos mayor incapié durante esta parte del curso.

