Sistemas Empotrados

Tema 4:
Excepciones e interrupciones

Lección 10:
Excepciones

Jesús González Peñalver

Contenidos

Concepto de excepción

Definición:

Cualquier evento que causa la detención de flujo normal de ejecución de instrucciones de un programa

Pueden suponer un cambio de modo de ejecución a un modo privilegiado

Clasificación:

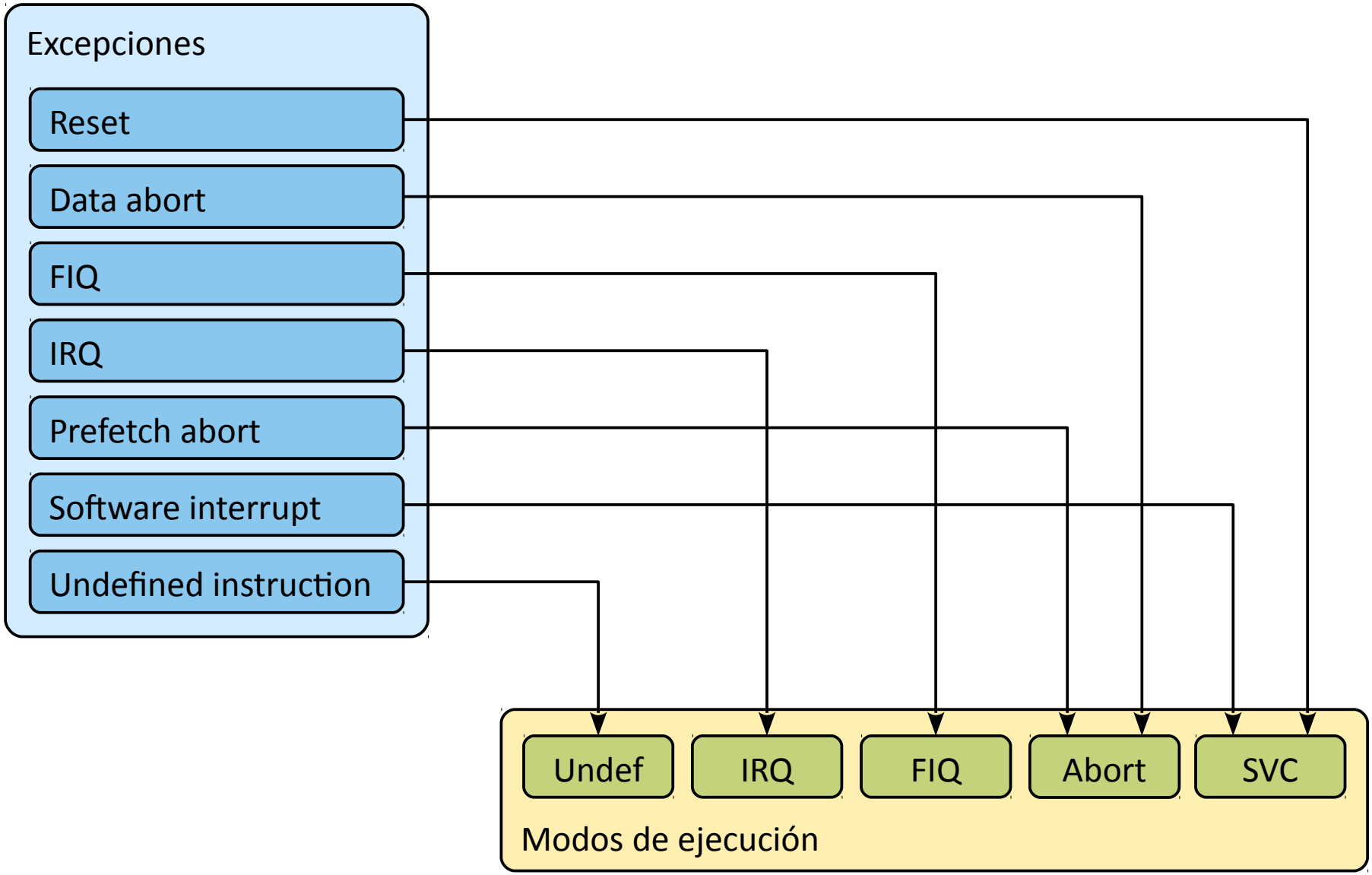
Condiciones de error: Recuperable o no recuperable

- **Instrucciones no definidas:** Puede que no sean errores realmente (Ej.: FP en software)
- **Data o prefetch aborts:** Puede que la dirección esté fuera del mapa de memoria o que la dirección no sea accesible por el proceso (MPU) o que haya un fallo de página (MMU)

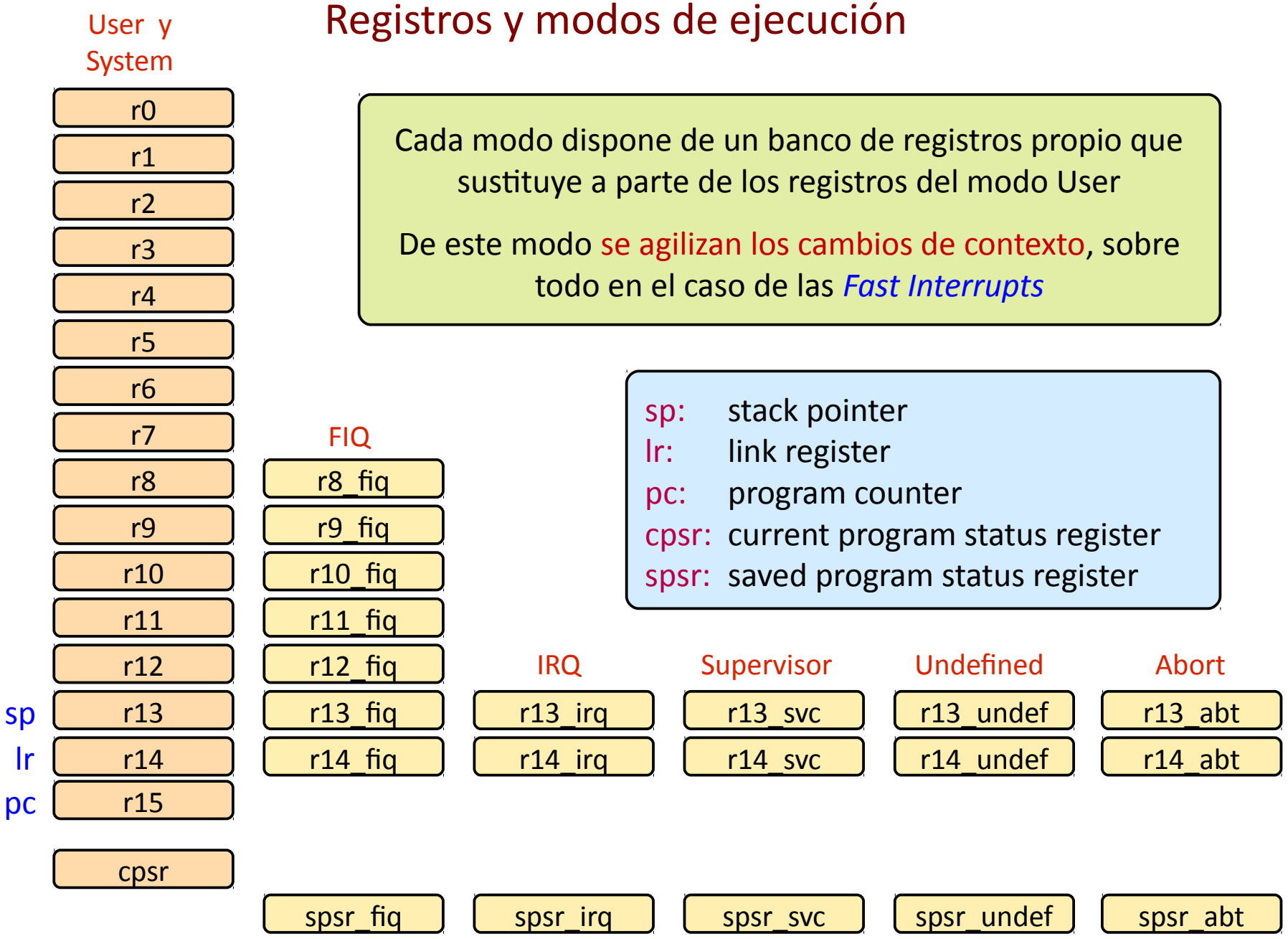
Interrupciones: Interrumpen la aplicación para prestar un servicio

- **Llamadas a sistema:** Atienden peticiones de servicio de la aplicación
- **Interrupciones:** Atienden peticiones de servicio de los dispositivos

Excepciones y modos de ejecución asociados



Registros y modos de ejecución

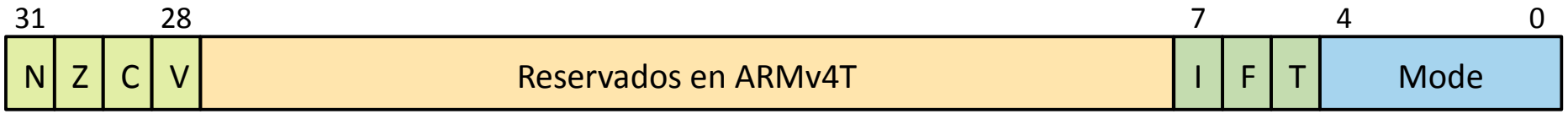


Cada modo dispone de un banco de registros propio que sustituye a parte de los registros del modo User

De este modo se agilizan los cambios de contexto, sobre todo en el caso de las *Fast Interrupts*

sp: stack pointer
lr: link register
pc: program counter
cpsr: current program status register
spsr: saved program status register

Los registros de estado (CPSR y SPSR)



Flags de condición — Solo lectura

N = Resultado Negativo en la ALU
Z = El resultado de la ALU es Zero
C = Se ha producido aCarreo en la ALU
V = Se ha producido oVerflow en la ALU

Máscaras de interrupción — Se pueden modificar en los modos privilegiados

I = 1, deshabilita las IRQ
F = 1, deshabilita las FIQ

Modo de ejecución — Solo lectura

T = 0, procesador en estado ARM
T = 1, procesador en estado Thumb

Bits de modo — Se pueden modificar en los modos privilegiados

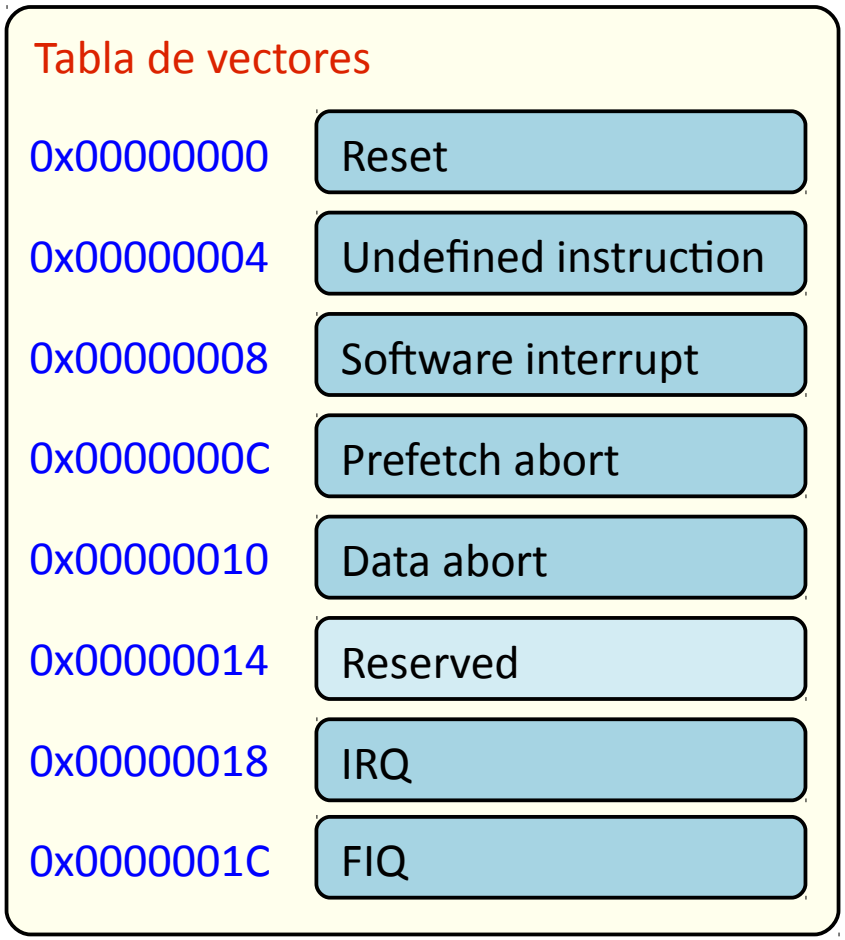
10000 : User
10001 : FIQ
10010 : IRQ
10011 : Supervisor
10111 : Abort
11011 : Undefined
11111 : System

Cualquier valor diferente provocará resultados impredecibles

Bits reservados — No se deben modificar

Se leen como cero

La tabla de vectores

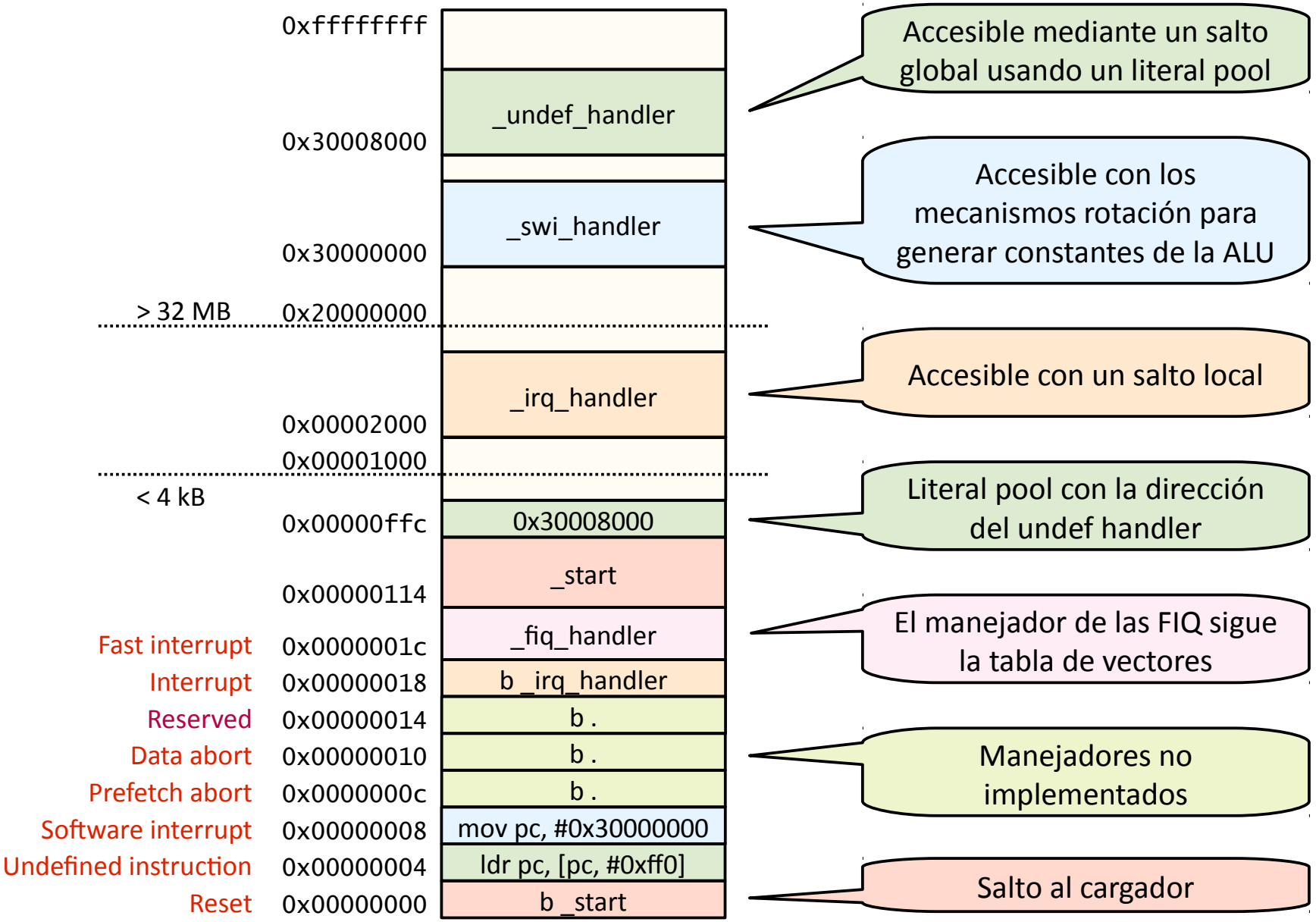


Cada vector es una instrucción ejecutable, normalmente un salto al manejador correspondiente

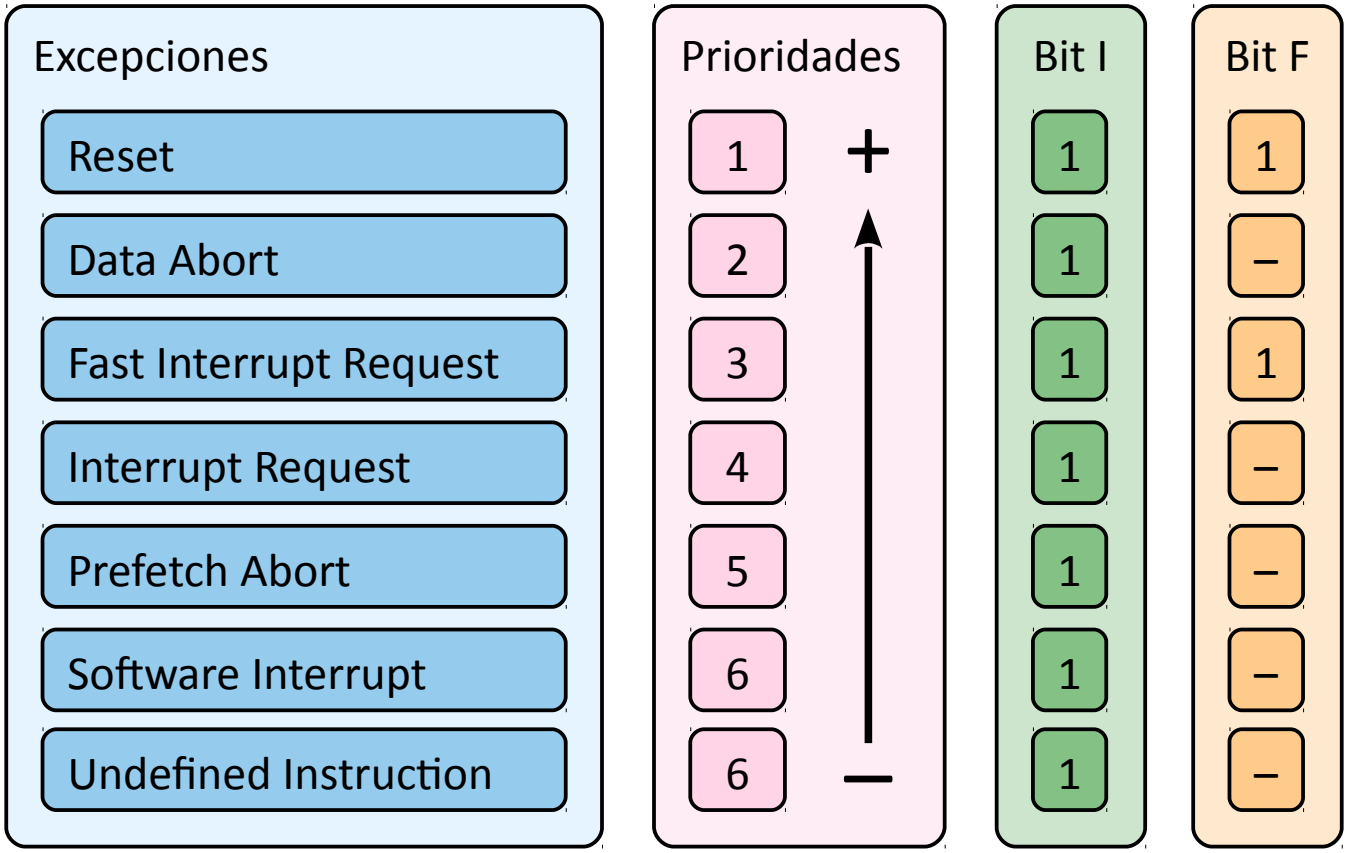
El vector de las FIQ se ha situado el último para poder comenzar directamente en la dirección 0x0000001C su manejador.

Esto, junto con que el modo FIQ tiene más registros replicados, agiliza el cambio de contexto, lo que minimiza la latencia de las FIQ

Ejemplo de mapa de memoria con manejadores de excepción



Prioridades de las excepciones



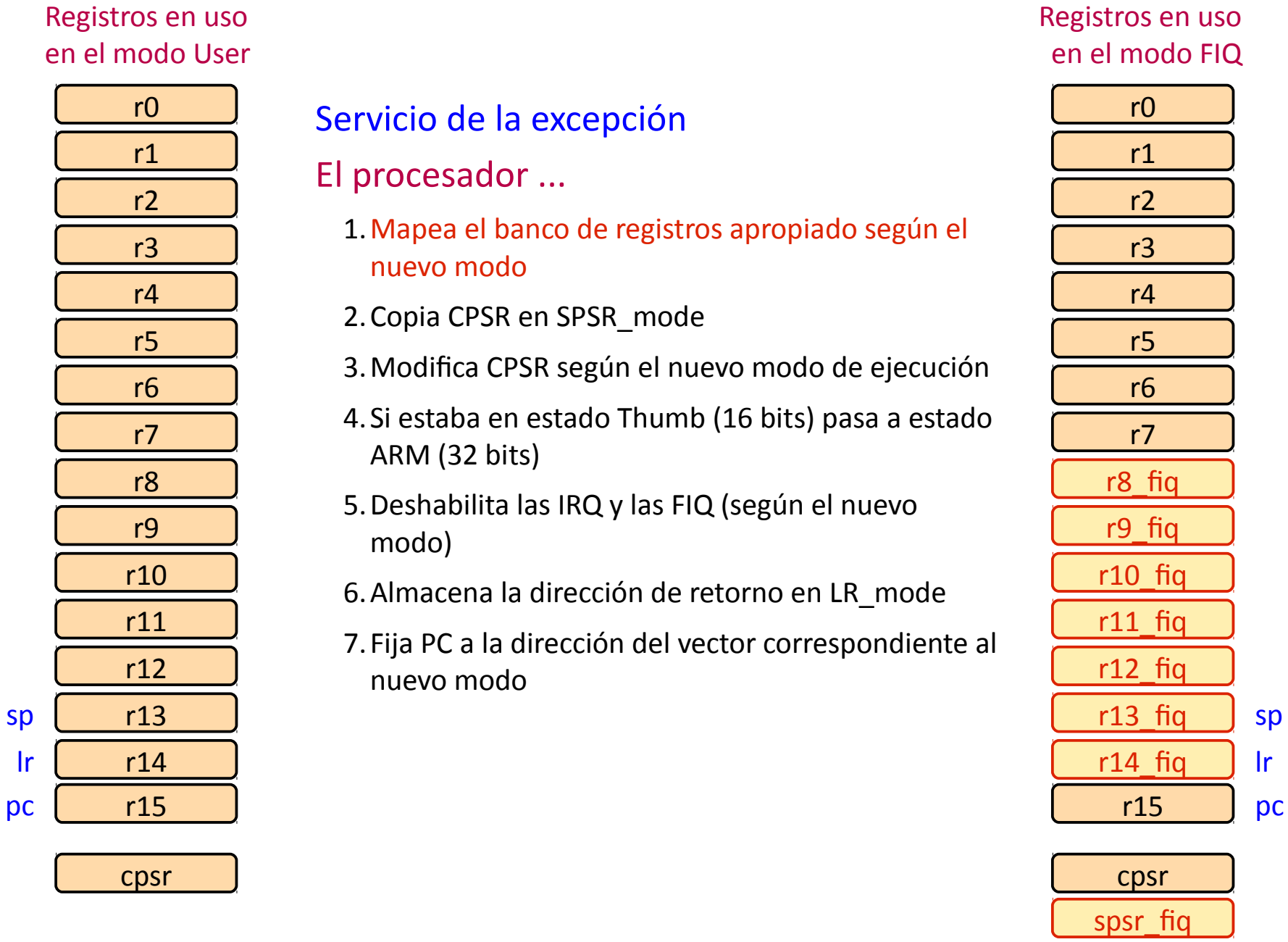
Las interrupciones software y las excepciones por una instrucción no definida tienen la misma prioridad, ya que no pueden ocurrir simultáneamente

1: Deshabilita

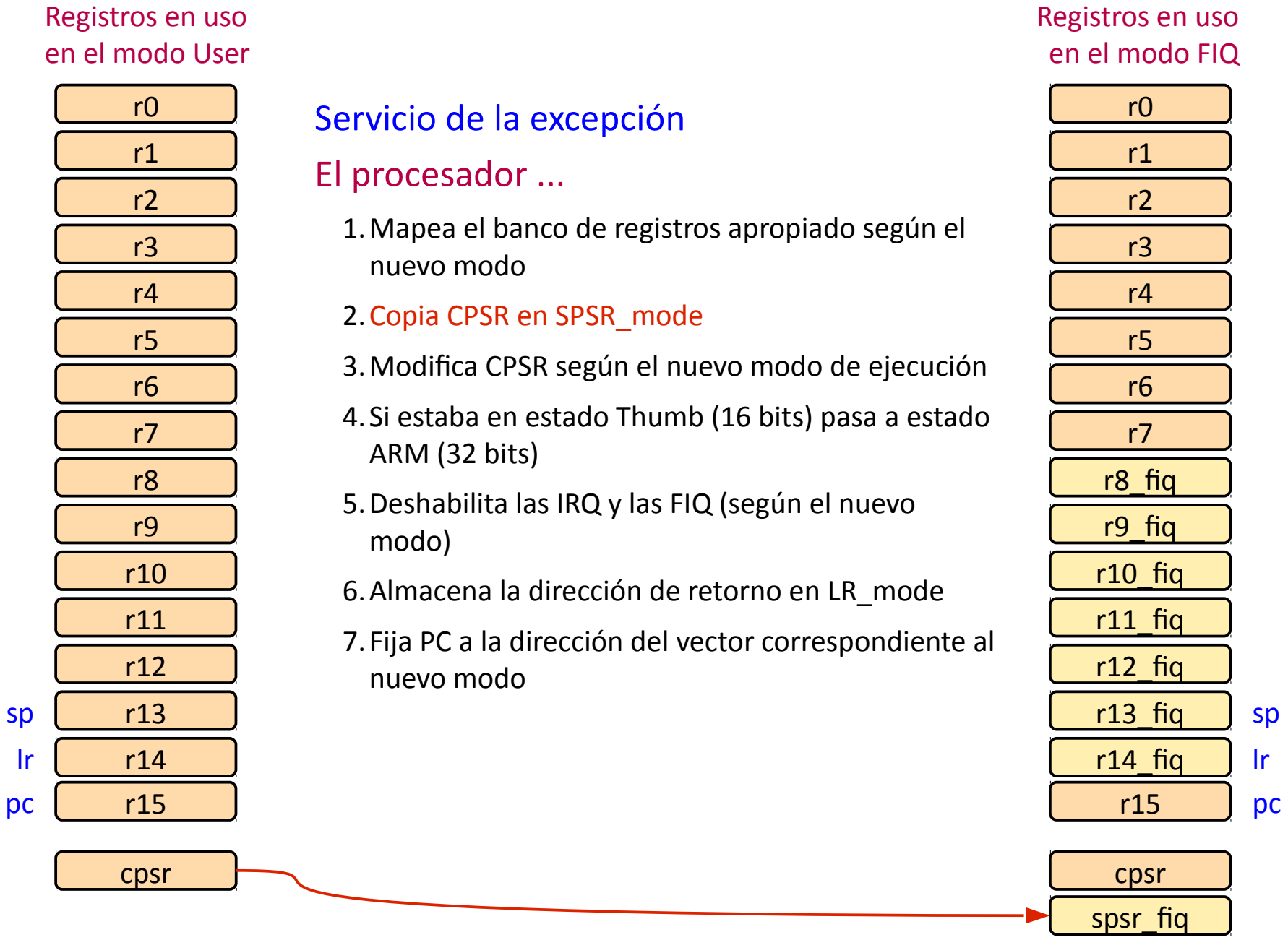
–: No afecta

Contenidos

Gestión de excepciones



Gestión de excepciones



Gestión de excepciones

Registros en uso en el modo User

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
sp r13
lr r14
pc r15

cpsr

Servicio de la excepción

El procesador ...

1. Mapea el banco de registros apropiado según el nuevo modo
2. Copia CPSR en SPSR_mode
3. Modifica CPSR según el nuevo modo de ejecución
4. Si estaba en estado Thumb (16 bits) pasa a estado ARM (32 bits)
5. Deshabilita las IRQ y las FIQ (según el nuevo modo)
6. Almacena la dirección de retorno en LR_mode
7. Fija PC a la dirección del vector correspondiente al nuevo modo

Registros en uso en el modo FIQ

r0
r1
r2
r3
r4
r5
r6
r7
r8_fiq
r9_fiq
r10_fiq
r11_fiq
r12_fiq
r13_fiq sp
r14_fiq lr
r15 pc

cpsr
spsr_fiq

Gestión de excepciones

Registros en uso en el modo User

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
sp r13
lr r14
pc r15

cpsr

Servicio de la excepción

El procesador ...

1. Mapea el banco de registros apropiado según el nuevo modo
2. Copia CPSR en SPSR_mode
3. Modifica CPSR según el nuevo modo de ejecución
4. Si estaba en estado Thumb (16 bits) pasa a estado ARM (32 bits)
5. Deshabilita las IRQ y las FIQ (según el nuevo modo)
6. Almacena la dirección de retorno en LR_mode
7. Fija PC a la dirección del vector correspondiente al nuevo modo

Registros en uso en el modo FIQ

r0
r1
r2
r3
r4
r5
r6
r7
r8_fiq
r9_fiq
r10_fiq
r11_fiq
r12_fiq
r13_fiq sp
r14_fiq lr
r15 pc

cpsr
spsr_fiq

Gestión de excepciones

Registros en uso en el modo User

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
sp r13
lr r14
pc r15

cpsr

Servicio de la excepción

El procesador ...

1. Mapea el banco de registros apropiado según el nuevo modo
2. Copia CPSR en SPSR_mode
3. Modifica CPSR según el nuevo modo de ejecución
4. Si estaba en estado Thumb (16 bits) pasa a estado ARM (32 bits)
5. Deshabilita las IRQ y las FIQ (según el nuevo modo)
6. Almacena la dirección de retorno en LR_mode
7. Fija PC a la dirección del vector correspondiente al nuevo modo

Registros en uso en el modo FIQ

r0
r1
r2
r3
r4
r5
r6
r7
r8_fiq
r9_fiq
r10_fiq
r11_fiq
r12_fiq
r13_fiq sp
r14_fiq lr
r15 pc

cpsr
spsr_fiq

Gestión de excepciones

Registros en uso en el modo User

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
sp r13
lr r14
pc r15

cpsr

Servicio de la excepción

El procesador ...

1. Mapea el banco de registros apropiado según el nuevo modo
2. Copia CPSR en SPSR_mode
3. Modifica CPSR según el nuevo modo de ejecución
4. Si estaba en estado Thumb (16 bits) pasa a estado ARM (32 bits)
5. Deshabilita las IRQ y las FIQ (según el nuevo modo)
6. Almacena la dirección de retorno en LR_mode
7. Fija PC a la dirección del vector correspondiente al nuevo modo

Registros en uso en el modo FIQ

r0
r1
r2
r3
r4
r5
r6
r7
r8_fiq
r9_fiq
r10_fiq
r11_fiq
r12_fiq
r13_fiq sp
r14_fiq lr
r15 pc

cpsr
spsr_fiq

Gestión de excepciones

Registros en uso en el modo User

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
sp r13
lr r14
pc r15

cpsr

Servicio de la excepción

El procesador ...

1. Mapea el banco de registros apropiado según el nuevo modo
2. Copia CPSR en SPSR_mode
3. Modifica CPSR según el nuevo modo de ejecución
4. Si estaba en estado Thumb (16 bits) pasa a estado ARM (32 bits)
5. Deshabilita las IRQ y las FIQ (según el nuevo modo)
6. Almacena la dirección de retorno en LR_mode
7. Fija PC a la dirección del vector correspondiente al nuevo modo

Registros en uso en el modo FIQ

r0
r1
r2
r3
r4
r5
r6
r7
r8_fiq
r9_fiq
r10_fiq
r11_fiq
r12_fiq
r13_fiq sp
r14_fiq lr
0x0000001C pc

cpsr
spsr_fiq

La tabla de manejadores de excepción (crt0.s)

```
_start:
        b       _setup_vectors
```

```
@ Tabla de vectores que se copiará a la RAM
        .globl _vector_table
_vector_table:
        ldr     pc, [pc, #24]   @ Soft reset
        ldr     pc, [pc, #24]   @ Undefined
        ldr     pc, [pc, #24]   @ SWI
        ldr     pc, [pc, #24]   @ Prefetch abort
        ldr     pc, [pc, #24]   @ Data abort
        nop                     @ Reserved
        ldr     pc, [pc, #24]   @ IRQ
        ldr     pc, [pc, #24]   @ FIQ
```

```
@ Tabla de direcciones absolutas de manejadores
        .word   _soft_reset_handler
        .word   _undef_handler
        .word   _swi_handler
        .word   _pabt_handler
        .word   _dabt_handler
        nop
        .word   _irq_handler
        .word   _fiq_handler
```

```
@ Manejadores por defecto
_soft_reset_handler:
        b       .
_undef_handler:
        b       .
_swi_handler:
        b       .
_pabt_handler:
        b       .
_dabt_handler:
        b       .
_irq_handler:
        b       .
_fiq_handler:
        b       .

@ Copiamos las tablas de vectores y de
@ manejadores al principio de la RAM
        .globl _setup_vectors
_setup_vectors:
        sub     r8, pc, #(8+.-_vector_table)
        ldr     r9, =_ram_base_boot
        ldmia   r8!, {r0-r7}
        stmia   r9!, {r0-r7}
        ldmia   r8!, {r0-r7}
        stmia   r9!, {r0-r7}
```

Se debe usar una dirección relativa. Todavía no hemos remapeado la memoria

Definida por el linker

La tabla de vectores es una tabla de saltos globales a las direcciones de los manejadores

Las tablas de vectores y manejadores deben copiarse al principio de la RAM antes de remapear

La tabla de manejadores de excepción (crt0.s)

Linker script

```
MEMORY
{
  ram   : org = 0x00000000, len = 0x00040000
  flash : org = 0x01000000, len = 0x00100000
}

SECTIONS
{
    .startup :  { ... } > flash

    .vectors :  {
        . += 0x20 ;
        _excep_handlers = . ;
        . += 0x20 ;
    } > ram
    _ram_base_boot = 0x00300000 ;

    .text :     { ... } > ram AT > flash
    _text_flash_start = LOADADDR(.text);

    .rodata :   { ... } > flash

    .data :     { ... } > ram AT > flash
    _data_flash_start = LOADADDR(.data);

    .bss :      { ... } > ram

    _ram_limit = ORIGIN(ram) + LENGTH(ram);
    _stack_size = 0x800;
    .stack _ram_limit - _stack_size : { ... }
}
```

Reservamos espacio para la tabla de vectores y la tabla de manejadores

Dirección de inicio de la RAM antes del remapeo

Acceso a la tabla de manejadores de excepción

Tipos de excepción

```
typedef enum {
        excep_reset = 0,
        excep_undef,
        excep_swi,
        excep_pabt,
        excep_dabt,
        excep_rsv,
        excep_irq,
        excep_fiq,
        excep_max
} excep_t;
```

Prototipo para los manejadores

```
typedef void (* excep_handler_t) (void);
```

Tabla de manejadores

```
extern volatile excep_handler_t  _excep_handlers[excep_max];
```

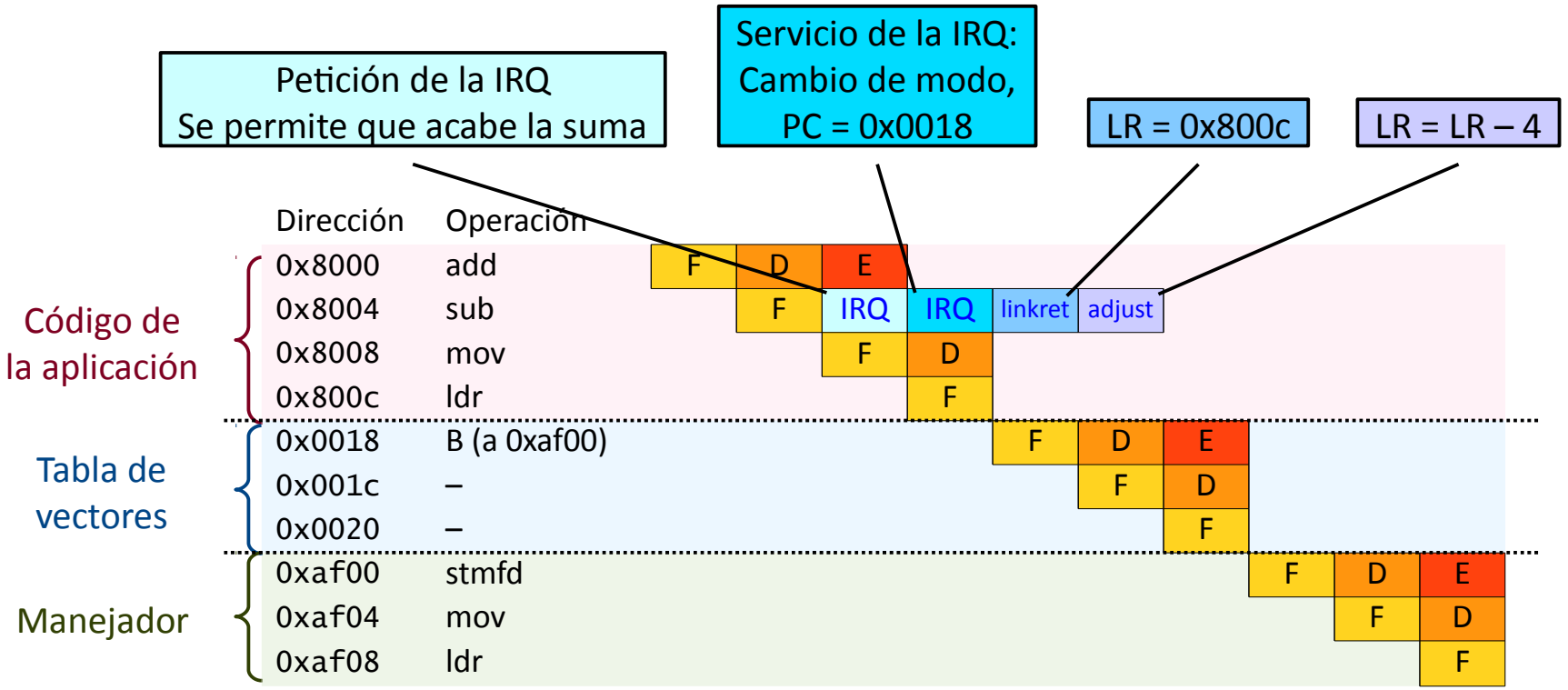
Fijar un manejador

```
inline void excep_set_handler (excep_t excep, excep_handler_t handler) {
        _excep_handlers[excep] = handler;
}
```

Obtener un manejador

```
inline excep_handler_t excep_get_handler (excep_t excep) {
        return _excep_handlers[excep];
}
```

La dirección de retorno del manejador de una excepción



Las excepciones son tratadas en el procesador como saltos con enlace (BL)

Problema:

Según el tipo de excepción, puede que el contenido de LR no apunte a la instrucción correcta

Ejemplo:

En este caso, LR apunta a la instrucción mov, en vez de a la instrucción sub

Solución:

El manejador deberá corregir el registro LR para que se pueda retornar a la instrucción correcta

Ajuste de la dirección de retorno de la excepción

Excepción	Dirección	Uso
Reset	–	No definido tras un reset
Data Abort	LR – 8	Apunta a la instrucción que causó la excepción
Fast Interrupt Request	LR – 4	Dirección de retorno de la FIQ
Interrupt Request	LR – 4	Dirección de retorno de la IRQ
Prefetch Abort	LR – 4	Apunta a la instrucción que causó la excepción
Software Interrupt	LR	Apunta a la siguiente instrucción tras la instrucción SWI
Undefined Instruction	LR	Apunta a la siguiente instrucción tras la instrucción no definida

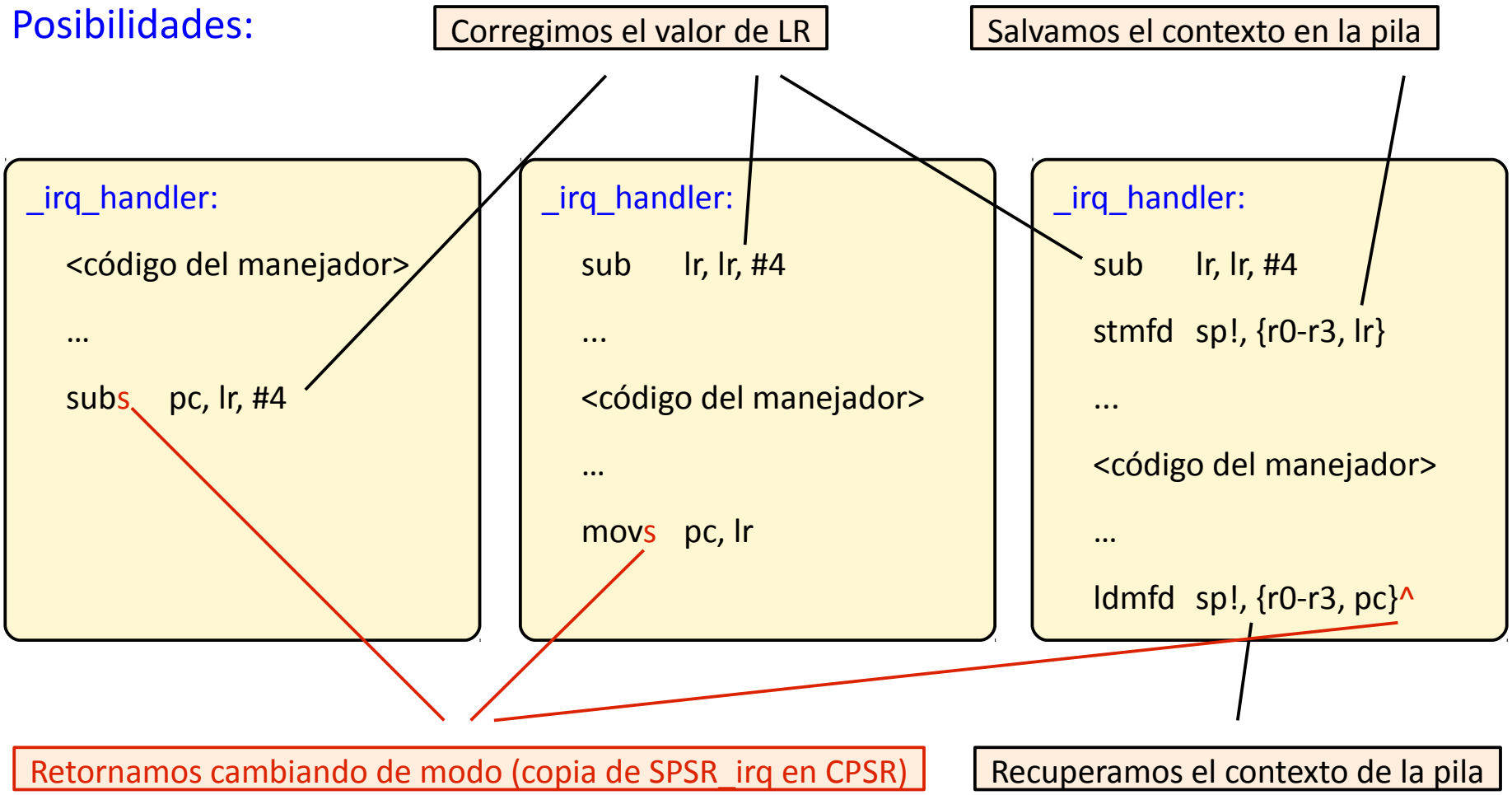
Retorno del manejador de una excepción

Antes de retornar, el manejador debe...

Cambiar al modo de ejecución previo a la excepción (copiar SPSR_mode en CPSR)

Mover a PC la dirección de retorno (obtenida a partir de LR_mode)

Posibilidades:



Retorno del manejador de una excepción en C (GNU)

El compilador puede realizar los ajustes anteriores por nosotros

Sólo hay que indicar, al declarar la función, que se trata de un manejador de excepciones, junto con el tipo de excepción

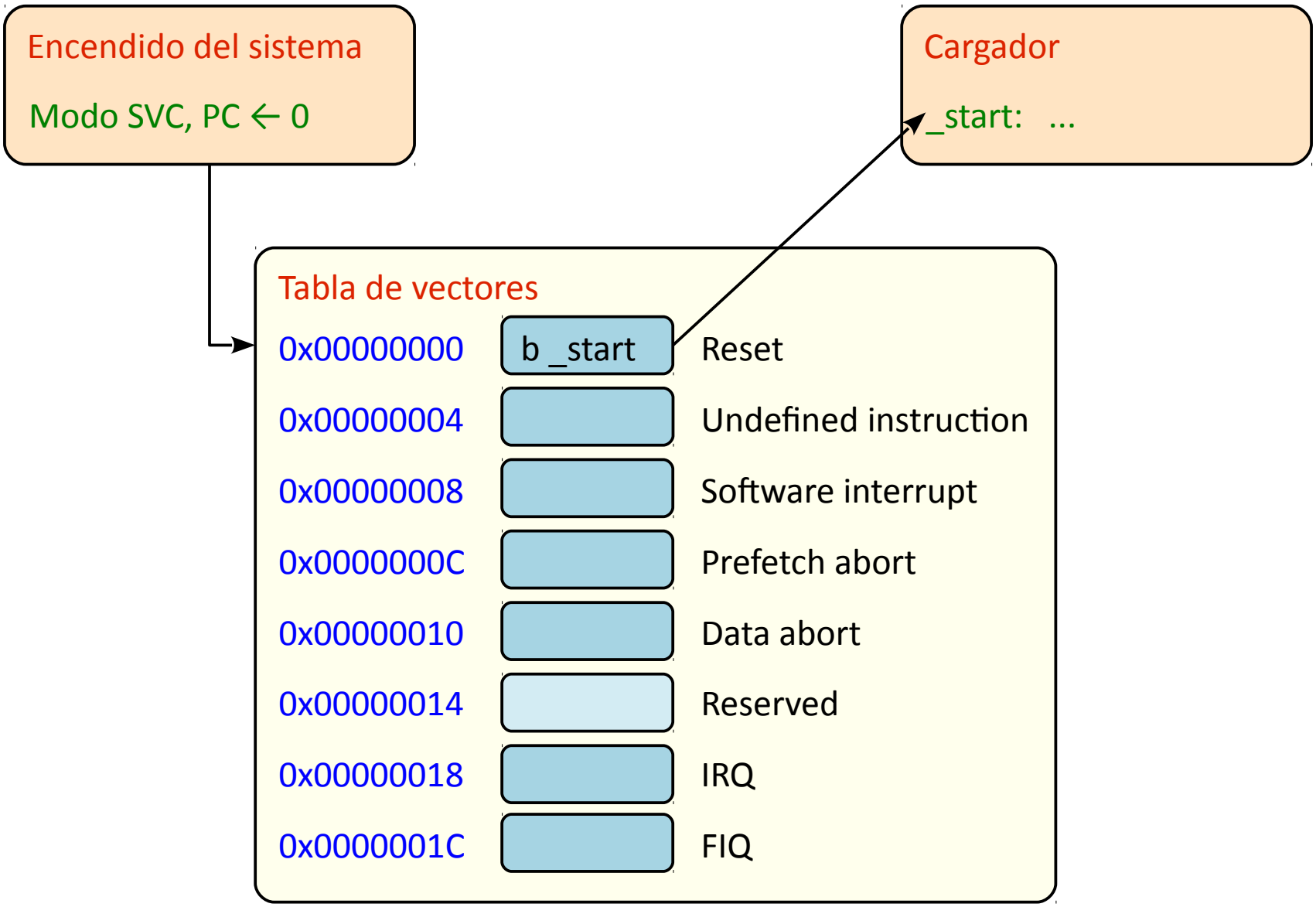
```
__attribute__ ((interrupt ("<tipo>")))
void my_handler (void)
{
        /*
         * Código del manejador
         */
}
```

El valor de `<tipo>` indica el tipo de excepción.

Posibilidades: `IRQ`, `FIQ`, `SWI`, `ABORT` y `UNDEF`

Contenidos

Reset


Encendido del sistema
Modo SVC, PC ← 0
Cargador
_start: ...
Tabla de vectores
0x00000000
b _start
Reset
0x00000004
Undefined instruction
0x00000008
Software interrupt
0x0000000C
Prefetch abort
0x00000010
Data abort
0x00000014
Reserved
0x00000018
IRQ
0x0000001C
FIQ

Instrucción no definida

Posibilidades

Se captan 32 bits de memoria que no codifican una instrucción válida

La instrucción es para un coprocesador que no está presente o no responde (ej. FP)

Utilidad

Permite usar instrucciones máquina en nuestro programa que puede que no estén soportadas en nuestro procesador

Si no existe un coprocesador para ejecutar estas instrucciones de forma eficiente, el manejador de la excepción llamará a una subrutina que lo hará mediante software

Ejemplo: Soporte de instrucciones FP sin FPU

El compilador genera instrucciones para la FPU cuando encuentra operaciones FP

Si el sistema cuenta con FPU, la ejecución será muy eficiente

Si no, se generará una excepción, y el manejador llamará a una función que calculará el resultado correcto

El programa funcionará correctamente, aunque no haya FPU, solo que la ejecución será más lenta

Ejemplo: Nos inventamos la instrucción memset

Fija un búfer a un valor determinado

Sólo usamos operandos en registro para simplificar el ejemplo

Sintaxis

memset{<cond>} rd, rn, rm

rd: Puntero al búffer, rn: tamaño del búfer, rm: valor

Ejemplo de uso

```
adr    r4, buffer  ; puntero al búfer
ldr    r5, =256    ; tamaño del búfer
ldr    r6, = 0     ; valor inicial

.word  0x77f54006  ; memset r4,r5,r6
```

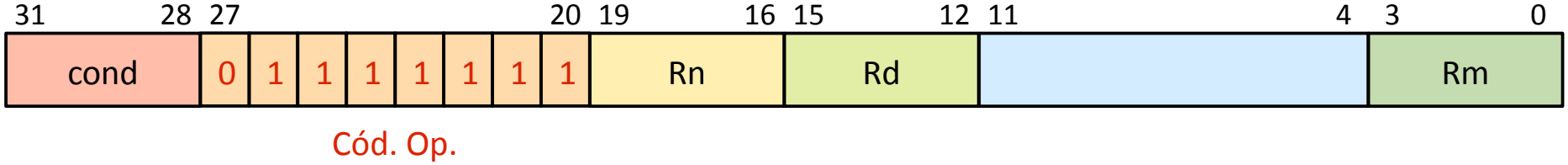
Manejador

```
undef_handler:

  stmfd sp!, {r0-r12,lr}    ; salvamos el contexto
  mrs   r0, spsr            ; r0 <- spsr
  str   r0, [sp, #-4]!      ; salvamos spsr en la pila

  ldr   r0, [lr, -#4]       ; r0 <- instr. no definida
  bic   r2, r0, #0xf00fffff ; aislamos el codop
  teq   r2, #07f00000       ; comprobamos el codop
  bleq  memset_instr        ; ejecutamos la instrucción

  ldr   r1, [sp], #4        ; r1 <- spsr
  msr   spsr_cxsf, r1       ; Restauramos spsr
  ldmfd sp!, {r0-r12, pc}^  ; Retornamos del manejador
```



Aborts

Posibilidades

El procesador intenta acceder a un dato (*data abort*) o a una instrucción (*prefetch abort*) mediante una dirección que no está en el mapa de memoria de direcciones válidas para el proceso que se está ejecutando

Utilidad

Con la presencia de MPU o MMU permite implementar esquemas de protección de memoria o memoria virtual

Si no hay MMU o MPU, o bien el proceso está intentando acceder a una dirección que sólo se puede acceder en un modo privilegiado, se trata de un error fatal → Reset

Ejemplo: Implementación de memoria virtual

Un proceso intenta acceder a una dirección virtual de memoria

La MMU comprueba que la dirección física correspondiente no está en la RAM

Genera una excepción (*prefetch* o *data abort*), según el tipo de acceso

El manejador trae la página correspondiente a memoria RAM y retorna a la instrucción que causó la excepción (apuntada por lr – 8 en los *data aborts* o lr – 4 en los *prefetch aborts*)

Interrupción software

Posibilidades

Se usan para que un proceso de usuario realice una petición de un servicio al sistema operativo. El identificador del servicio de codifica en los bits menos significativos (24 bits en estado ARM, 8 bits en estado thumb). Los parámetros del servicio se pasan en los registros, de r0 en adelante.

Utilidad

Fuerza un cambio al modo privilegiado SVC, lo que permite acceder a recursos no accesibles directamente por los procesos (pantallas, dispositivos de almacenamiento, etc.). El SO se encarga de gestionar dichos recursos mediante el uso de colas, etc.

Ejemplo: E/S estándar (printf, scanf, etc.)

El SO define STDIN, STDOUT y STDERR con los descriptores estándar 0, 1 y 2. Cualquier proceso que quiera acceder a dichos dispositivos debe hacerlo a través del SO

```
hello_str: .ascii "Hello, World!\n"

mov   r0, #1          @ Descriptor de fichero para STDOUT
adr   r1, hello_str   @ Puntero a la buffer
mov   r2, #14         @ Tamaño del buffer
mov   r12, #4         @ Write es el servicio 4 del kernel
swi   0x80            @ Llamada al kernel de linux
```

Interrupciones

Posibilidades

En los cores de ARM hay dos tipos IRQ (*Interrupt Request*) y FIQ (*Fast Interrupt Request*)

Como en el sistema habrá más de dos fuentes de interrupción, el manejador correspondiente (IRQ o FIQ) deberá identificar la fuente de la petición de interrupción antes de proceder a su servicio, teniendo en cuenta las prioridades asignadas a cada dispositivo

La presencia de un controlador de interrupciones HW facilita la identificación de la fuente, minimizando la latencia de la ISR

Utilidad

Permiten una gestión eficaz de los dispositivos de E/S

Ejemplo: Llegada de un paquete de red al sistema

La interfaz de red realiza una petición de interrupción al procesador

El procesador deja de ejecutar el proceso actual y atiende la petición, almacenando los datos en el búfer correspondiente

Una vez que los datos ha sido correctamente recibidos, se reanuda la ejecución del proceso interrumpido

Lecturas recomendadas

Gestión de excepciones:

A. N. Sloss, D. Symes, C. Wright. *ARM System Developer's Guide*. Morgan Kaufmann, 2004. Capítulo 9

W. Hohl. *ARM Assembly Language. Fundamentals and Techniques*. CRC Press, 2009. Capítulo 11

M. Samek. *Building Bare-Metal ARM Systems with GNU: Part 6 General Description of Interrupt Handling*. Embedded.com, 2007.

M. Samek. *Building Bare-Metal ARM Systems with GNU: Part 7 Interrupt Locking and Unlocking*. Embedded.com, 2007.

M. Samek. *Building Bare-Metal ARM Systems with GNU: Part 8 Low Level Interrupt Wrapper Functions*. Embedded.com, 2007.

M. Samek. *Building Bare-Metal ARM Systems with GNU: Part 9 C-level ISRs and other ARM Exceptions*. Embedded.com, 2007.