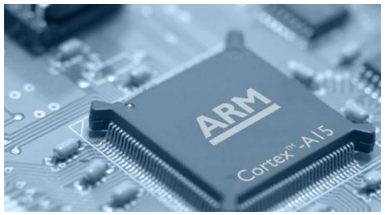


Procesadores ARM

Implementación de MMU para multitasking

Alejandro Furfaro

15 de mayo de 2023

Temario

Temario

Modos

Modos

La arquitectura ARMv7 propone diferentes sistemas de Manejo de Memoria, de acuerdo con el tipo de aplicación para cada SoC.

Modos

La arquitectura ARMv7 propone diferentes sistemas de Manejo de Memoria, de acuerdo con el tipo de aplicación para cada SoC.

En general, para aplicaciones donde existen restricciones severas de temporización, que requieren sistemas Real Time, se emplea una Unidad que permite asignar regiones de memoria de tamaño variable a las diferentes tareas, y que le puede asignar atributos a cada región.

Modos

La arquitectura ARMv7 propone diferentes sistemas de Manejo de Memoria, de acuerdo con el tipo de aplicación para cada SoC.

En general, para aplicaciones donde existen restricciones severas de temporización, que requieren sistemas Real Time, se emplea una Unidad que permite asignar regiones de memoria de tamaño variable a las diferentes tareas, y que le puede asignar atributos a cada región.

Para poder hostear un SO Multitasking con asignación de memoria y tares dinámica, implementa una típica unidad de Paginación que permite mapear direcciones virtuales en direcciones físicas.

Modos

Modos

El perfil Cortex-M, como ya sabemos se aplica a aplicaciones de rango bajo a medio (dependiendo del SOC) pero podrían resolver problemas hasta sin un sistema operativo, sino mas bien un pequeño administrador de lazo cerrado. Obviamente disponen de una modesta Unidad de Protección de Regiones: MPU.

Modos

El perfil Cortex-M, como ya sabemos se aplica a aplicaciones de rango bajo a medio (dependiendo del SOC) pero podrían resolver problemas hasta sin un sistema operativo, sino mas bien un pequeño administrador de lazo cerrado. Obviamente disponen de una modesta Unidad de Protección de Regiones: MPU.

El Perfil Cortex-R, es exactamente igual a Cortex-A, excepto justamente en este aspecto. Al estar destinado a aplicaciones de Tiempo Real que a la vez requieran de un procesador robusto, se diferencia del Cortex-A en la funcionalidad de su MMU. El Cortex-R posee una MPU, que permite asignar estáticamente las áreas de memoria a sus procesos en tiempo de inicialización, y no cambiar en todo el ciclo de trabajo. Es decir Aplicaciones Real Time de misión critica y gama alta.

Modos

El perfil Cortex-A, por su parte se pensó para aplicaciones de gama media y alta que permitan ejecución dinámica de tareas y alocación dinámica de memoria. Estas dos funciones no pueden implementarse sin capacidades de Memoria Virtual.
De los dos métodos posibles para gestionar bloques de memoria, ARM, empresa formada en los años 80, encontró la discusión concluida por el peso de la evidencia tecnológica: Paginación. Y llegaría a desarrollar procesadores aptos para alojamiento dinámico de memoria y tareas bastante tiempo después. De modo que obviamente utilizan Paginación como método.

Temario

MMU y MPU

MMU y MPU

- La única diferencia entre el perfil R respecto del perfil A es el soporte de Hardware de Administración de Memoria.

MMU y MPU

- La única diferencia entre el perfil R respecto del perfil A es el soporte de Hardware de Administración de Memoria.
- ARM denomina **MPU** (**M**emory **P**rotection **U**nit) y **MMU** (**M**emory **M**anagement **U**nit) a las unidades de administración de memoria del Perfil R y del Perfil A respectivamente.

MMU y MPU

- La única diferencia entre el perfil R respecto del perfil A es el soporte de Hardware de Administración de Memoria.
- ARM denomina **MPU** (**M**emory **P**rotection **U**nit) y **MMU** (**M**emory **M**anagement **U**nit) a las unidades de administración de memoria del Perfil R y del Perfil A respectivamente.
- Lo cierto es que ambas unidades Administran (o Gestionan) la Memoria, de modo que el término **MMU** aplica a ambas, en base al estricto significado de sus siglas: **M**emory **M**anagement **U**nit.

MMU y MPU

- La única diferencia entre el perfil R respecto del perfil A es el soporte de Hardware de Administración de Memoria.
- ARM denomina **MPU** (**M**emory **P**rotection **U**nit) y **MMU** (**M**emory **M**anagement **U**nit) a las unidades de administración de memoria del Perfil R y del Perfil A respectivamente.
- Lo cierto es que ambas unidades Administran (o Gestionan) la Memoria, de modo que el término **MMU** aplica a ambas, en base al estricto significado de sus siglas: **M**emory **M**anagement **U**nit.
- Lo que ARM denomina **MMU** es en verdad una Unidad de Mapeo de Memoria por Paginación, con capacidad de alocación dinámica, y soporte a Memoria Virtual.

MMU y MPU

- La única diferencia entre el perfil R respecto del perfil A es el soporte de Hardware de Administración de Memoria.
- ARM denomina **MPU** (**M**emory **P**rotection **U**nit) y **MMU** (**M**emory **M**anagement **U**nit) a las unidades de administración de memoria del Perfil R y del Perfil A respectivamente.
- Lo cierto es que ambas unidades Administran (o Gestionan) la Memoria, de modo que el término **MMU** aplica a ambas, en base al estricto significado de sus siglas: **M**emory **M**anagement **U**nit.
- Lo que ARM denomina **MMU** es en verdad una Unidad de Mapeo de Memoria por Paginación, con capacidad de alocación dinámica, y soporte a Memoria Virtual.
- Lo que ARM denomina **MPU**, como veremos someramente, es una **MMU** sin capacidad de administración de memoria dinámica, ni soporte a Memoria Virtual, y que al final del día es una Unidad de administración rígida de Segmentos.

Administración de Memoria en Cortex-R y Cortex-A

Administración de Memoria en Cortex-R y Cortex-A

- Cuando tenemos un procesador ARMv7 Cortex-A o Cortex-R, para habilitar el soporte de Hardware de la Administración de memoria, es necesario habilitar el Coprocesador 15.

Administración de Memoria en Cortex-R y Cortex-A

- Cuando tenemos un procesador ARMv7 Cortex-A o Cortex-R, para habilitar el soporte de Hardware de la Administración de memoria, es necesario habilitar el Coprocesador 15.
- En cada caso la arquitectura de Sistemas provee diferentes registros para configurar el Hardware.

Administración de Memoria en Cortex-R y Cortex-A

- Cuando tenemos un procesador ARMv7 Cortex-A o Cortex-R, para habilitar el soporte de Hardware de la Administración de memoria, es necesario habilitar el Coprocesador 15.
- En cada caso la arquitectura de Sistemas provee diferentes registros para configurar el Hardware.
- ARM para el caso del CORTEX-R habla de un sistema de administración de memoria al que denomina **P**rotected **M**emory **S**ystem **A**rchitecture: **PMSA**.

Administración de Memoria en Cortex-R y Cortex-A

- Cuando tenemos un procesador ARMv7 Cortex-A o Cortex-R, para habilitar el soporte de Hardware de la Administración de memoria, es necesario habilitar el Coprocesador 15.
- En cada caso la arquitectura de Sistemas provee diferentes registros para configurar el Hardware.
- ARM para el caso del CORTEX-R habla de un sistema de administración de memoria al que denomina **P**rotected **M**emory **S**ystem **A**rchitecture: **PMSA**.
- Puesto así, pareciera que la otra alternativa no provee Protección. Lo cual obviamente es falso.

Administración de Memoria en Cortex-R y Cortex-A

- Cuando tenemos un procesador ARMv7 Cortex-A o Cortex-R, para habilitar el soporte de Hardware de la Administración de memoria, es necesario habilitar el Coprocesador 15.
- En cada caso la arquitectura de Sistemas provee diferentes registros para configurar el Hardware.
- ARM para el caso del CORTEX-R habla de un sistema de administración de memoria al que denomina **P**rotected **M**emory **S**ystem **A**rchitecture: **PMSA**.
- Puesto así, pareciera que la otra alternativa no provee Protección. Lo cual obviamente es falso.
- En realidad es un esquema de protección muy simplificado que para empezar no requiere ningún tipo de traslación.

Administración de Memoria en Cortex-R y Cortex-A

- Cuando tenemos un procesador ARMv7 Cortex-A o Cortex-R, para habilitar el soporte de Hardware de la Administración de memoria, es necesario habilitar el Coprocesador 15.
- En cada caso la arquitectura de Sistemas provee diferentes registros para configurar el Hardware.
- ARM para el caso del CORTEX-R habla de un sistema de administración de memoria al que denomina **P**rotected **M**emory **S**ystem **A**rchitecture: **PMSA**.
- Puesto así, pareciera que la otra alternativa no provee Protección. Lo cual obviamente es falso.
- En realidad es un esquema de protección muy simplificado que para empezar no requiere ningún tipo de traslación.
- **PMSA** en lugar de utilizar tablas de traslación define Regiones de memoria, de tamaño variable, y con atributos que definen privilegios y derechos de acceso.

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.
  - Por cada región debe definirse, su tamaño, su dirección Base, y sus atributos. El tamaño mínimo es implementación dependiente. La dirección Base debe estar alineada al tamaño de la región.

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.
  - Por cada región debe definirse, su tamaño, su dirección Base, y sus atributos. El tamaño mínimo es implementación dependiente. La dirección Base debe estar alineada al tamaño de la región.
  - Puede definirse hasta 8 subregiones dentro de una región.

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.
  - Por cada región debe definirse, su tamaño, su dirección Base, y sus atributos. El tamaño mínimo es implementación dependiente. La dirección Base debe estar alineada al tamaño de la región.
  - Puede definirse hasta 8 subregiones dentro de una región.
  - Las regiones se pueden solapar. (te suena?).

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.
  - Por cada región debe definirse, su tamaño, su dirección Base, y sus atributos. El tamaño mínimo es implementación dependiente. La dirección Base debe estar alineada al tamaño de la región.
  - Puede definirse hasta 8 subregiones dentro de una región.
  - Las regiones se pueden solapar. (te suena?).
  - Si las regiones se solapan se define un esquema de prioridades tal que, a mayor número de región mayor es su prioridad. La prioridad define la región cuyos atributos se imponen para regular el acceso.

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.
  - Por cada región debe definirse, su tamaño, su dirección Base, y sus atributos. El tamaño mínimo es implementación dependiente. La dirección Base debe estar alineada al tamaño de la región.
  - Puede definirse hasta 8 subregiones dentro de una región.
  - Las regiones se pueden solapar. (te suena?).
  - Si las regiones se solapan se define un esquema de prioridades tal que, a mayor número de región mayor es su prioridad. La prioridad define la región cuyos atributos se imponen para regular el acceso.
  - Un acceso erróneo deriva en un Memory Abort, a menos que se produzca en PL1 en cuyo caso usa un mapa default de memoria.

Administración de Memoria en Cortex R

- ✓ Reglas que gobiernan las Regiones de memoria.
  - Por cada región debe definirse, su tamaño, su dirección Base, y sus atributos. El tamaño mínimo es implementación dependiente. La dirección Base debe estar alineada al tamaño de la región.
  - Puede definirse hasta 8 subregiones dentro de una región.
  - Las regiones se pueden solapar. (te suena?).
  - Si las regiones se solapan se define un esquema de prioridades tal que, a mayor número de región mayor es su prioridad. La prioridad define la región cuyos atributos se imponen para regular el acceso.
  - Un acceso erróneo deriva en un Memory Abort, a menos que se produzca en PL1 en cuyo caso usa un mapa default de memoria.
  - Las direcciones que se definen en las regiones son físicas. No existe ningún tipo de traslación.

Administración de Memoria en Cortex R

- ✓ Region Background.

Administración de Memoria en Cortex R

- ✓ Region Background.
  - Es un mapa que abarca los 4 GiB del espacio de direccionamiento.

Administración de Memoria en Cortex R

- ✓ Region Background.
  - Es un mapa que abarca los 4 GiB del espacio de direccionamiento.
  - Su prioridad es mas baja que la de cualquier otra región.

Administración de Memoria en Cortex R

- ✓ Region Background.
  - Es un mapa que abarca los 4 GiB del espacio de direccionamiento.
  - Su prioridad es mas baja que la de cualquier otra región.
  - Cualquier acceso erróneo a las regiones definidas termina cayendo en esta región y toma sus atributos.

Administración de Memoria en Cortex R

- ✓ Region Background.
  - Es un mapa que abarca los 4 GiB del espacio de direccionamiento.
  - Su prioridad es mas baja que la de cualquier otra región.
  - Cualquier acceso erróneo a las regiones definidas termina cayendo en esta región y toma sus atributos.
  - Si no se define esta región un acceso indebido a memoria genera un Memory Abort.

Administración de Memoria en Cortex R

- ✓ Region Background.
  - Es un mapa que abarca los 4 GiB del espacio de direccionamiento.
  - Su prioridad es mas baja que la de cualquier otra región.
  - Cualquier acceso erróneo a las regiones definidas termina cayendo en esta región y toma sus atributos.
  - Si no se define esta región un acceso indebido a memoria genera un Memory Abort.
  - Se configura mediante una serie de registros de control del Coprocesador 15.

Administración de Memoria en Cortex R

- ✓ Region Background.
  - Es un mapa que abarca los 4 GiB del espacio de direccionamiento.
  - Su prioridad es mas baja que la de cualquier otra región.
  - Cualquier acceso erróneo a las regiones definidas termina cayendo en esta región y toma sus atributos.
  - Si no se define esta región un acceso indebido a memoria genera un Memory Abort.
  - Se configura mediante una serie de registros de control del Coprocesador 15.
  - Siempre tener presente que es rígido, pero muy ágil en su funcionamiento y sobre todo sus tiempos de operación cumplen con el criterio de determinismo que en ocasiones un RTOS requiere de manera muy estricta.

Administración de Memoria en Cortex R

Las regiones pueden ser activas o durmientes. Las primeras contienen código en ejecución o datos en uso. Las segundas contienen código o datos que no están siendo utilizados en este momento. A pesar de ello esas regiones se encuentran protegidas de accesos indebidos por parte de las regiones activas.
De este modo la Unidad de Memoria de un Cortex R (y también la de un Cortex M) si bien no implementa alocación dinámica de memoria ni provee mecanismos de Memoria Virtual para aplicaciones que no lo requieren, implementa eso si, un sistema de protección basado en los atributos que el kernel le asigne a cada región los cuales son controlados por Hardware, que en caso de accesos indebidos genera un Memory Abort (excepción) o habilita una Región Background.

Atributos de memoria con **MPU**

Atributo de Región	Opciones de configuración
Tipo	instrucciones, datos
Dirección Base	Múltiplo de su tamaño
Size	4 KiB a 4 GiB
Permisos de acceso	lectura, escritura, ejecución
Cache	copyback, writethrough
Write buffer	habilitado, deshabilitado

Atributos de memoria con **MPU**

Atributo de Región	Opciones de configuración
Tipo	instrucciones, datos
Dirección Base	Múltiplo de su tamaño
Size	4 KiB a 4 GiB
Permisos de acceso	lectura, escritura, ejecución
Cache	copyback, writethrough
Write buffer	habilitado, deshabilitado

Cada región recibe atributos de esta tabla. Se controla hasta la política de escritura de la región respecto del cache. La principales diferencias con la otra arquitectura que ARM llama **MMU** (¡como si ésta no lo fuera!), pueden resumirse en:

Atributos de memoria con **MPU**

Atributo de Región	Opciones de configuración
Tipo	instrucciones, datos
Dirección Base	Múltiplo de su tamaño
Size	4 KiB a 4 GiB
Permisos de acceso	lectura, escritura, ejecución
Cache	copyback, writethrough
Write buffer	habilitado, deshabilitado

Cada región recibe atributos de esta tabla. Se controla hasta la política de escritura de la región respecto del cache. La principales diferencias con la otra arquitectura que ARM llama **MMU** (¡como si ésta no lo fuera!), pueden resumirse en:
[✓] No implementa mecanismo de Memoria Virtual.

Atributos de memoria con **MPU**

Atributo de Región	Opciones de configuración
Tipo	instrucciones, datos
Dirección Base	Múltiplo de su tamaño
Size	4 KiB a 4 GiB
Permisos de acceso	lectura, escritura, ejecución
Cache	copyback, writethrough
Write buffer	habilitado, deshabilitado

Cada región recibe atributos de esta tabla. Se controla hasta la política de escritura de la región respecto del cache. La principales diferencias con la otra arquitectura que ARM llama **MMU** (¡como si ésta no lo fuera!), pueden resumirse en:
[✓] No implementa mecanismo de Memoria Virtual.
[✓] Las regiones un amplio rango de tamaños posibles.

Atributos de memoria con **MPU**

Atributo de Región	Opciones de configuración
Tipo	instrucciones, datos
Dirección Base	Múltiplo de su tamaño
Size	4 KiB a 4 GiB
Permisos de acceso	lectura, escritura, ejecución
Cache	copyback, writethrough
Write buffer	habilitado, deshabilitado

Cada región recibe atributos de esta tabla. Se controla hasta la política de escritura de la región respecto del cache. La principales diferencias con la otra arquitectura que ARM llama **MMU** (¡como si ésta no lo fuera!), pueden resumirse en:

[✓] No implementa mecanismo de Memoria Virtual.

[✓] Las regiones un amplio rango de tamaños posibles.

[✓] Las regiones se definen en determinados registros memory mapped de la MPU versus tablas en memoria para la MMU.

Temario

VMSA

El perfil Cortex-A ha sido diseñado con lo que ARM denomina **V**irtual **M**emory **S**ystem **A**rchitecture (**VMSA** de ahora en más). Esta arquitectura implementa una **MMU** que se encarga de traducir direcciones virtuales a físicas, controlar los accesos a memoria por parte de la CPU, en base a atributos y permisos que le configura a cada bloque de memoria gestionado.

VMSA

El perfil Cortex-A ha sido diseñado con lo que ARM denomina **V**irtual **M**emory **S**ystem **A**rchitecture (**VMSA** de ahora en más). Esta arquitectura implementa una **MMU** que se encarga de traducir direcciones virtuales a físicas, controlar los accesos a memoria por parte de la CPU, en base a atributos y permisos que le configura a cada bloque de memoria gestionado.

La traslación toma como entrada una dirección correspondiente a un instruction fetch o al acceso a un dato en memoria, *dirección virtual*, y genera una dirección de salida que ARM denomina:

VMSA

El perfil Cortex-A ha sido diseñado con lo que ARM denomina **V**irtual **M**emory **S**ystem **A**rchitecture (**VMSA** de ahora en más). Esta arquitectura implementa una **MMU** que se encarga de traducir direcciones virtuales a físicas, controlar los accesos a memoria por parte de la CPU, en base a atributos y permisos que le configura a cada bloque de memoria gestionado.

La traslación toma como entrada una dirección correspondiente a un instruction fetch o al acceso a un dato en memoria, *dirección virtual*, y genera una dirección de salida que ARM denomina:
[✓] Dirección Física (**PA**) por **P**hysical **A**ddress, si trabaja con traslación en 1 Nivel.

VMSA

El perfil Cortex-A ha sido diseñado con lo que ARM denomina **V**irtual **M**emory **S**ystem **A**rchitecture (**VMSA** de ahora en más). Esta arquitectura implementa una **MMU** que se encarga de traducir direcciones virtuales a físicas, controlar los accesos a memoria por parte de la CPU, en base a atributos y permisos que le configura a cada bloque de memoria gestionado.

La traslación toma como entrada una dirección correspondiente a un instruction fetch o al acceso a un dato en memoria, *dirección virtual*, y genera una dirección de salida que ARM denomina:
[✓] Dirección Física (**PA**) por **P**hysical **A**ddress, si trabaja con traslación en 1 Nivel.
[✓] Dirección Física Intermedia (**IPA**) por **I**ntermediate **P**hysical **A**ddress, si trabaja con traslación en 1 Nivel.

Alcance de nuestro análisis

El perfil Cortex-A tiene diferentes extensiones. Las mas comunes de encontrar a lo largo de la documentación de Arquitectura de Sistemas son las extensiones de Seguridad y Virtualización.

Alcance de nuestro análisis

El perfil Cortex-A tiene diferentes extensiones. Las mas comunes de encontrar a lo largo de la documentación de Arquitectura de Sistemas son las extensiones de Seguridad y Virtualización.

De estas extensiones derivan Modos de trabajo particulares como el Modo Seguro y No-Seguro, o el modo Hypervisor, que a su vez introduce un tercer nivel de privilegio (PL2), mas alto que los dos existentes en los demás modos.

Alcance de nuestro análisis

El perfil Cortex-A tiene diferentes extensiones. Las mas comunes de encontrar a lo largo de la documentación de Arquitectura de Sistemas son las extensiones de Seguridad y Virtualización.

De estas extensiones derivan Modos de trabajo particulares como el Modo Seguro y No-Seguro, o el modo Hypervisor, que a su vez introduce un tercer nivel de privilegio (PL2), mas alto que los dos existentes en los demás modos.

Por supuesto, estas extensiones requieren ser habilitadas, por lo tanto podríamos denominar modo default o base al modo en el que no tenemos estas extensiones habilitadas.

Alcance de nuestro análisis

En caso de habilitarse una cualquiera o ambas extensiones, se tienen diferentes registros para configurar los mismos recursos, o se emplean versiones diferentes de los mismos registros, empleando el habitual truco de Banqueo de registros típico de ARM.

Alcance de nuestro análisis

En caso de habilitarse una cualquiera o ambas extensiones, se tienen diferentes registros para configurar los mismos recursos, o se emplean versiones diferentes de los mismos registros, empleando el habitual truco de Banqueo de registros típico de ARM.

El objetivo de este curso es comprender la administración dinámica de memoria, y memoria virtual. El modo default es mas que suficiente para ello, así que nos limitaremos a éste sin incluir las extensiones de Virtualización ni de seguridad.

Alcance de nuestro análisis

En caso de habilitarse una cualquiera o ambas extensiones, se tienen diferentes registros para configurar los mismos recursos, o se emplean versiones diferentes de los mismos registros, empleando el habitual truco de Banqueo de registros típico de ARM.

El objetivo de este curso es comprender la administración dinámica de memoria, y memoria virtual. El modo default es mas que suficiente para ello, así que nos limitaremos a éste sin incluir las extensiones de Virtualización ni de seguridad.

Considerar las extensiones solo suma complejidad, multiplicando los registros a analizar de acuerdo con el modo de trabajo. No suma un solo concepto adicional al análisis en el modo default.

VMSA: Espacios de direccionamiento

VMSA: Espacios de direccionamiento

- Una dirección virtual es un número de 32 bit, de modo que direcciona desde 0x00000000 a 0xFFFFFFFF.

VMSA: Espacios de direccionamiento

- Una dirección virtual es un número de 32 bit, de modo que direcciona desde 0x00000000 a 0xFFFFFFFF.

- Una **IPA** puede tener un espacio de direccionamiento de hasta 40 bit.

VMSA: Espacios de direccionamiento

- Una dirección virtual es un número de 32 bit, de modo que direcciona desde 0x00000000 a 0xFFFFFFFF.
- Una **IPA** puede tener un espacio de direccionamiento de hasta 40 bit.
- Large Physical Address Extensions define dos formatos de tablas de descriptores

VMSA: Espacios de direccionamiento

- Una dirección virtual es un número de 32 bit, de modo que direcciona desde 0x00000000 a 0xFFFFFFFF.
- Una **IPA** puede tener un espacio de direccionamiento de hasta 40 bit.
- Large Physical Address Extensions define dos formatos de tablas de descriptores

1. **Long-descriptor Format** permite acceder a un espacio de direccionamiento de hasta 40 bit, tanto para **IPA** como para **PA**, con una granularidad de 4 KiB. Es una suerte de equivalente al PAE en la arquitectura x86

VMSA: Espacios de direccionamiento

- Una dirección virtual es un número de 32 bit, de modo que direcciona desde 0x00000000 a 0xFFFFFFFF.
- Una **IPA** puede tener un espacio de direccionamiento de hasta 40 bit.
- Large Physical Address Extensions define dos formatos de tablas de descriptores

1. **Long-descriptor Format** permite acceder a un espacio de direccionamiento de hasta 40 bit, tanto para **IPA** como para **PA**, con una granularidad de 4 KiB. Es una suerte de equivalente al PAE en la arquitectura x86
2. ***Short-descriptor format*** permite acceder a un espacio de 32 bit con una granularidad de 4 KiB, y opcionalmente puede proveer un espacio de hasta 40 bit, aunque con una granularidad de 16 MiB.

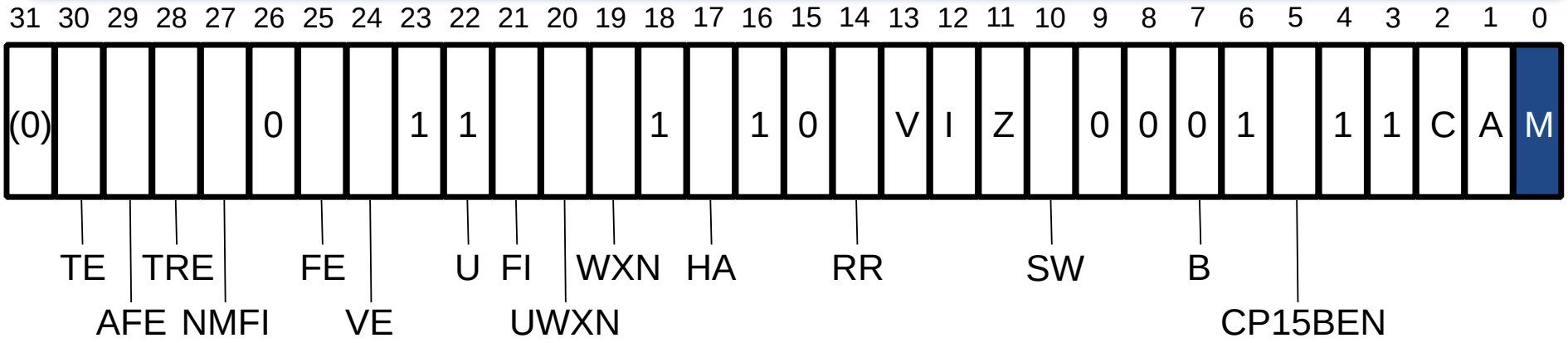
Temario

VMSA: Habilitación / Deshabilitación

El Registro de Control del Sistema (**`SCTRL`**), accesible en PL1, y banqueable, tiene en su bit menos significativo la llave para habilitar la traslación de páginas. **`SCTRL.M`**

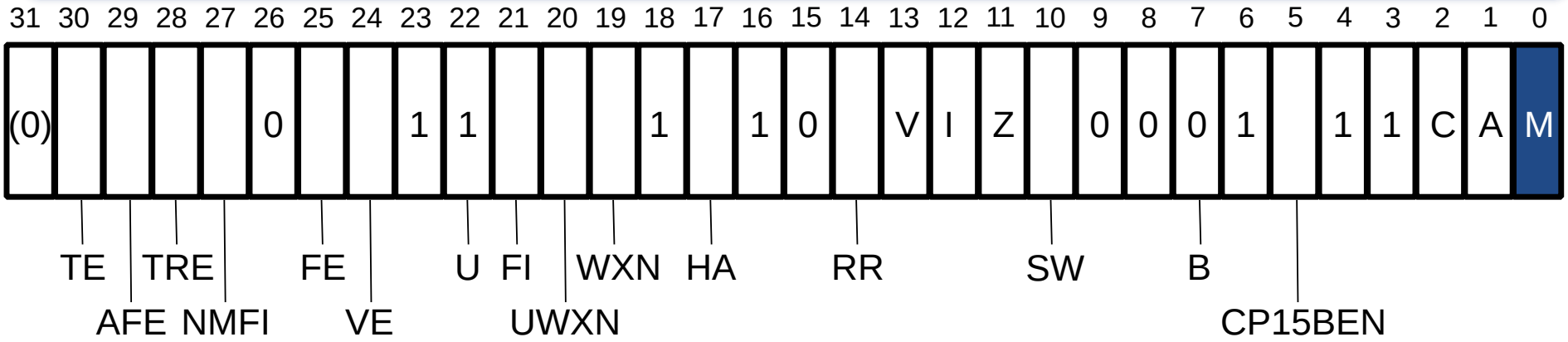
VMSA: Habilitación / Deshabilitación

El Registro de Control del Sistema (**SCTRL**), accesible en PL1, y banqueable, tiene en su bit menos significativo la llave para habilitar la traslación de páginas. **SCTRL.M**



VMSA: Habilitación / Deshabilitación

El Registro de Control del Sistema (**`SCTRL`**), accesible en PL1, y banqueable, tiene en su bit menos significativo la llave para habilitar la traslación de páginas. **`SCTRL.M`**

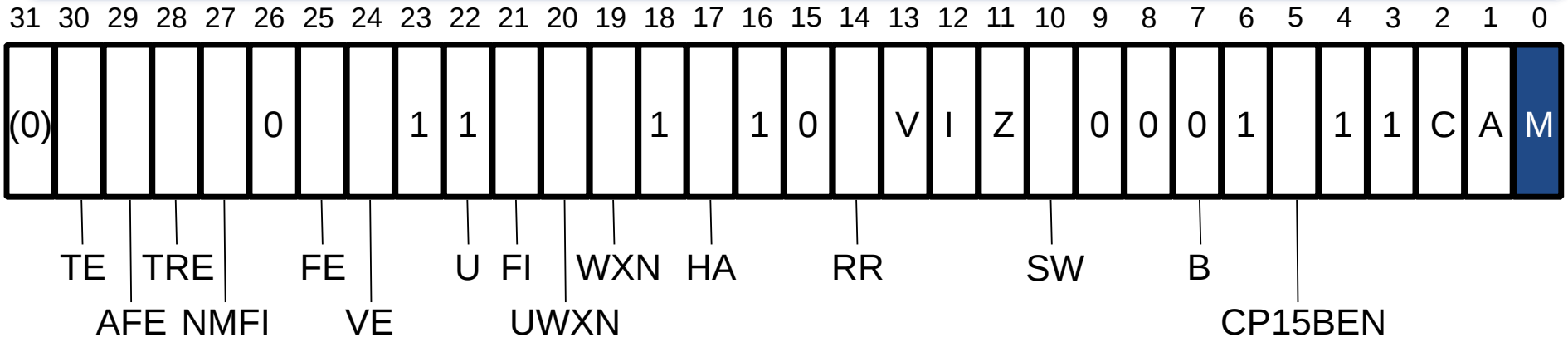


Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c1, c0, 0 ; Read SCTRL into Rt
MCR p15, 0, <Rt>, c1, c0, 0 ; Write Rt to SCTRL
```

VMSA: Habilitación / Deshabilitación

El Registro de Control del Sistema (**SCTRL**), accesible en PL1, y banqueable, tiene en su bit menos significativo la llave para habilitar la traslación de páginas. **SCTRL.M**



Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c1, c0, 0 ; Read SCTRL into Rt
MCR p15, 0, <Rt>, c1, c0, 0 ; Write Rt to SCTRL
```

Habilitación de VMSA

```
1 MRC  p15, 0, r0, c1, c0, 0 /* Lee SCTRL del cp15*/
2 or   r0, 0x1               /* Set SCTRL.M*/
3 MCR  p15, 0, r0, c1, c0, 0 /* Aplica en el registro del cp15*/
```

Temario

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

- Trabaja con traslación en uno o dos niveles.

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

- Trabaja con traslación en uno o dos niveles.
- En cualquier caso, el tamaño de un descriptor de cualquier entrada de Nivel 1 o Nivel 2 es de 32 bit.

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

- Trabaja con traslación en uno o dos niveles.
- En cualquier caso, el tamaño de un descriptor de cualquier entrada de Nivel 1 o Nivel 2 es de 32 bit.
- Las direcciones virtuales de entrada, son siempre de 32 bit.

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

- Trabaja con traslación en uno o dos niveles.
- En cualquier caso, el tamaño de un descriptor de cualquier entrada de Nivel 1 o Nivel 2 es de 32 bit.
- Las direcciones virtuales de entrada, son siempre de 32 bit.
- Produce direcciones físicas de salida de hasta 40 bit.

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

- Trabaja con traslación en uno o dos niveles.
- En cualquier caso, el tamaño de un descriptor de cualquier entrada de Nivel 1 o Nivel 2 es de 32 bit.
- Las direcciones virtuales de entrada, son siempre de 32 bit.
- Produce direcciones físicas de salida de hasta 40 bit.
- Para obtener direcciones físicas de salida de mas de 32 bits utiliza ***super secciones*** con una granularidad de 16 MiB.

Organización de los bloques de memoria

El formato Short-descriptor es el original definido para 32 bit y es el único formato soportado por implementaciones que no incluyen Large Physical Address Extension.

- Trabaja con traslación en uno o dos niveles.
- En cualquier caso, el tamaño de un descriptor de cualquier entrada de Nivel 1 o Nivel 2 es de 32 bit.
- Las direcciones virtuales de entrada, son siempre de 32 bit.
- Produce direcciones físicas de salida de hasta 40 bit.
- Para obtener direcciones físicas de salida de mas de 32 bits utiliza ***super secciones*** con una granularidad de 16 MiB.
- Maneja el concepto de Dominio: No access, Cliente, y Manager.

Short-Descriptor Formato de Tablas

Las tablas de traslación cuando se trabaja con Short-Descriptor format pueden organizar el mapa de memoria en páginas o secciones, de acuerdo con los contenidos de las tablas de traslación del primer nivel.

Short-Descriptor Formato de Tablas

Las tablas de traslación cuando se trabaja con Short-Descriptor format pueden organizar el mapa de memoria en páginas o secciones, de acuerdo con los contenidos de las tablas de traslación del primer nivel.

Supersections Bloques de memoria de 16 MiB. Su soporte es opcional, salvo si el core tiene Large Physical Address Extension, y soporta mas de 32 bit de memoria física. En este caso solo las *Supersections* le pueden permitir cubrir ese espacio.

Short-Descriptor Formato de Tablas

Las tablas de traslación cuando se trabaja con Short-Descriptor format pueden organizar el mapa de memoria en páginas o secciones, de acuerdo con los contenidos de las tablas de traslación del primer nivel.

Supersections Bloques de memoria de 16 MiB. Su soporte es opcional, salvo si el core tiene Large Physical Address Extension, y soporta mas de 32 bit de memoria física. En este caso solo las *Supersections* le pueden permitir cubrir ese espacio.

Sections Bloques de memoria de 1 MiB.

Short-Descriptor Formato de Tablas

Las tablas de traslación cuando se trabaja con Short-Descriptor format pueden organizar el mapa de memoria en páginas o secciones, de acuerdo con los contenidos de las tablas de traslación del primer nivel.

Supersections Bloques de memoria de 16 MiB. Su soporte es opcional, salvo si el core tiene Large Physical Address Extension, y soporta mas de 32 bit de memoria física. En este caso solo las *Supersections* le pueden permitir cubrir ese espacio.

Sections Bloques de memoria de 1 MiB.

Large pages Bloques de memoria de 64 KiB.

Short-Descriptor Formato de Tablas

Las tablas de traslación cuando se trabaja con Short-Descriptor format pueden organizar el mapa de memoria en páginas o secciones, de acuerdo con los contenidos de las tablas de traslación del primer nivel.

Supersections Bloques de memoria de 16 MiB. Su soporte es opcional, salvo si el core tiene Large Physical Address Extension, y soporta mas de 32 bit de memoria física. En este caso solo las *Supersections* le pueden permitir cubrir ese espacio.

Sections Bloques de memoria de 1 MiB.

Large pages Bloques de memoria de 64 KiB.

Small pages Bloques de memoria de 4 KiB.

Short-Descriptor Formato de Tablas

El Short-Descriptor format trabaja con uno o dos niveles de traslación. Dependiendo de como estén finalmente organizados los espacios de memoria, será lo que podamos ubicar en cada nivel.

Short-Descriptor Formato de Tablas

El Short-Descriptor format trabaja con uno o dos niveles de traslación. Dependiendo de como estén finalmente organizados los espacios de memoria, será lo que podamos ubicar en cada nivel.

Tabla de 1er. Nivel Contiene descriptores de 1er. Nivel, que directamente contienen la dirección base y atributos de *Super-sections* o *Sections*, o los atributos y un puntero a una segunda tabla de traslación en la que estarán finalmente los descriptores de Páginas *Large* o *Small*.

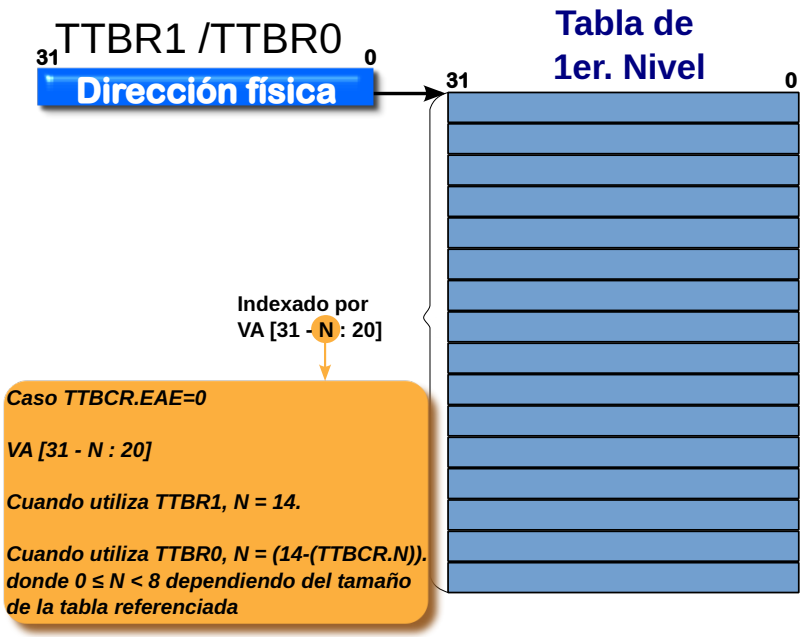
Short-Descriptor Formato de Tablas

El Short-Descriptor format trabaja con uno o dos niveles de traslación. Dependiendo de como estén finalmente organizados los espacios de memoria, será lo que podamos ubicar en cada nivel.

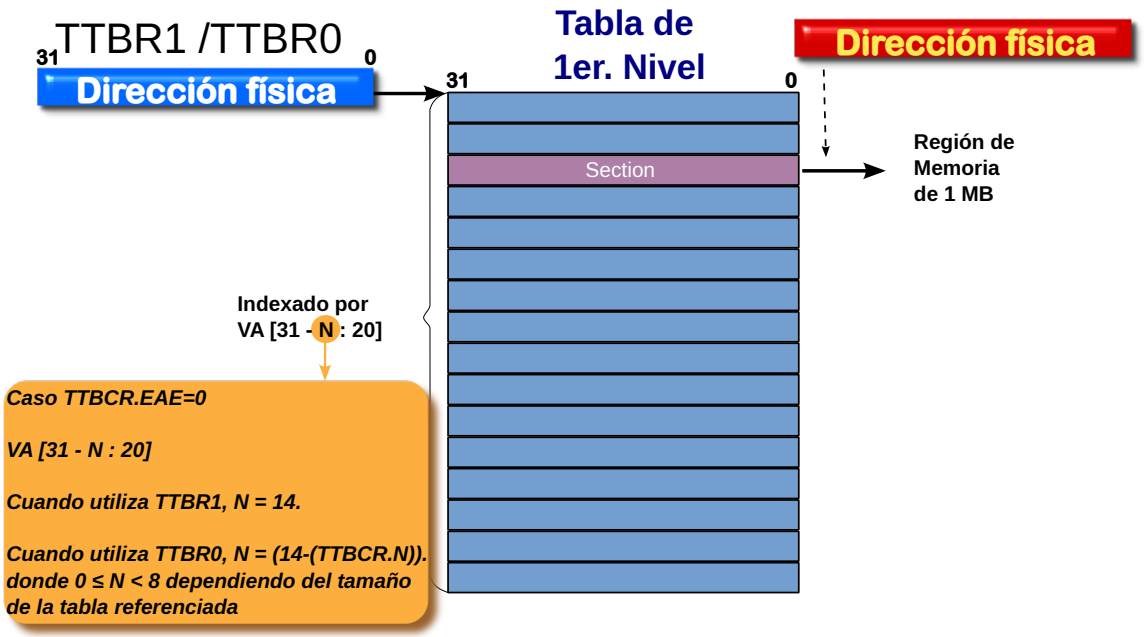
Tabla de 1er. Nivel Contiene descriptores de 1er. Nivel, que directamente contienen la dirección base y atributos de *Super-sections* o *Sections*, o los atributos y un puntero a una segunda tabla de traslación en la que estarán finalmente los descriptores de Páginas *Large* o *Small*.

Tabla de 2do. Nivel Siempre los descriptores de segundo nivel corresponden a *Small Pages* o *Large Pages*. En virtud de este contenido las tablas de segundo nivel suelen denominarse *Tablas de Páginas*

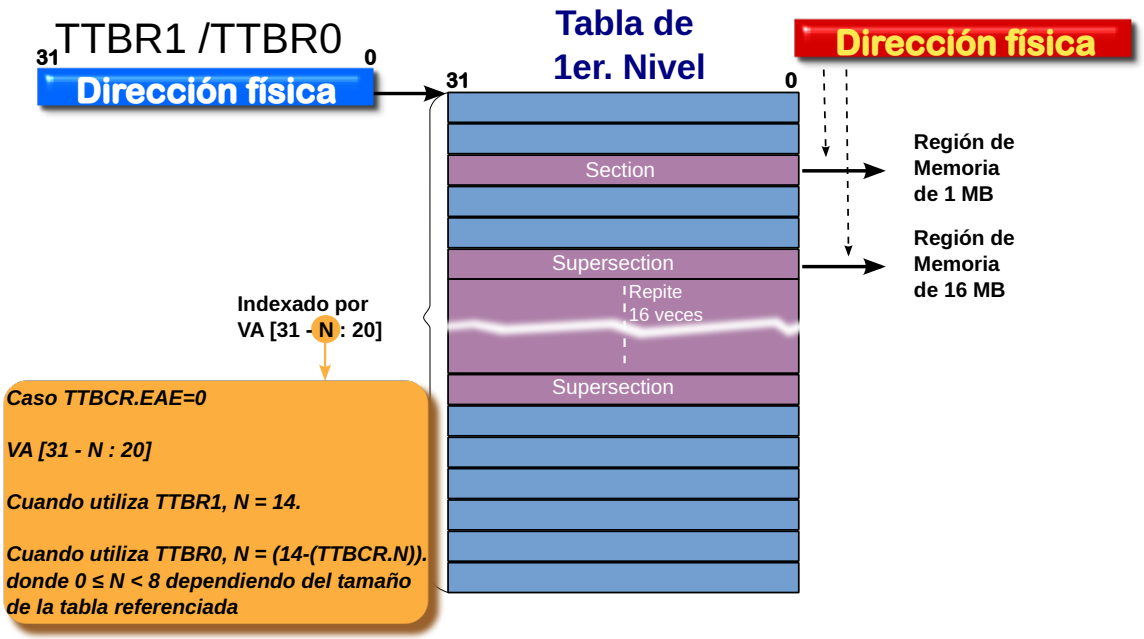
Short-Descriptor Formato de Tablas



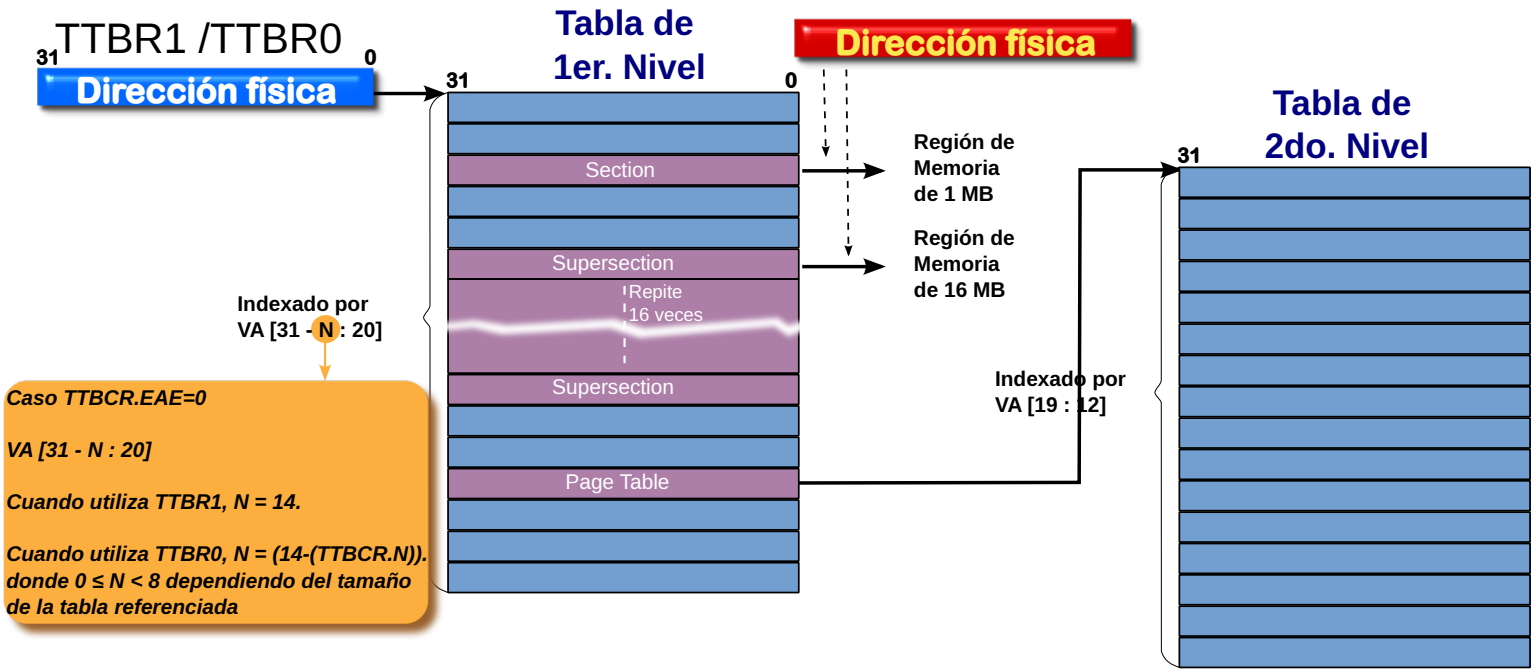
Short-Descriptor Formato de Tablas



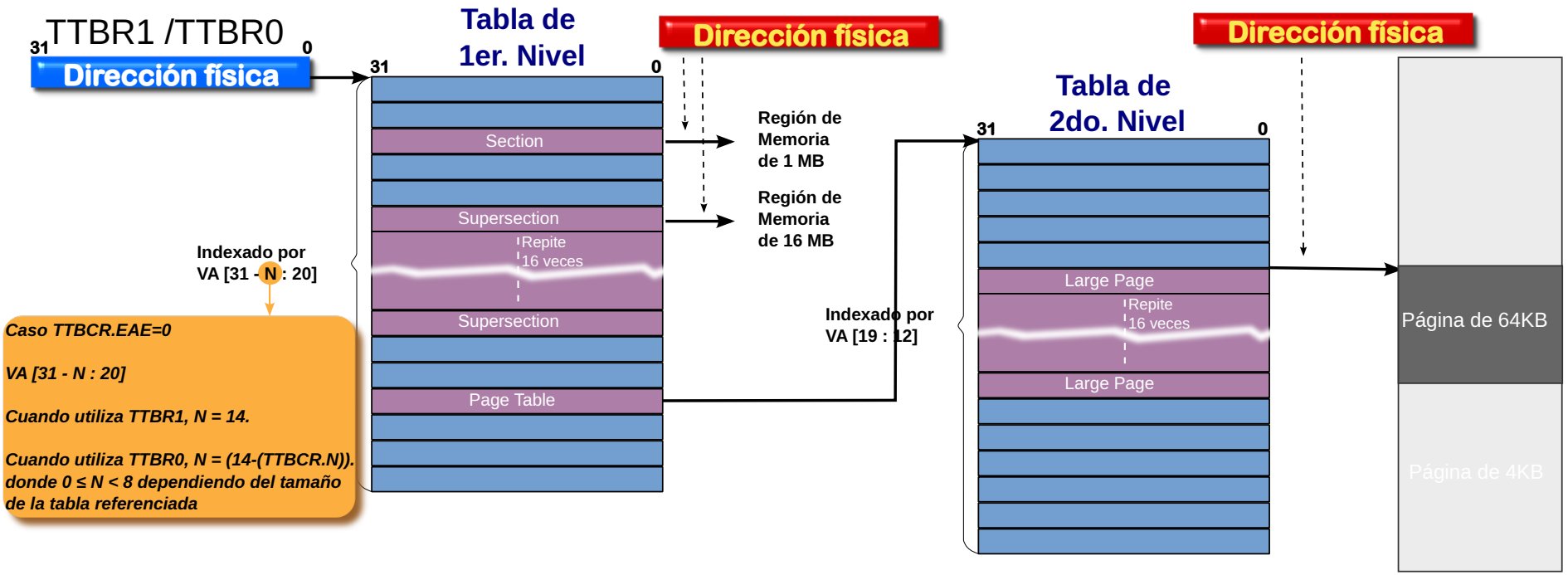
Short-Descriptor Formato de Tablas



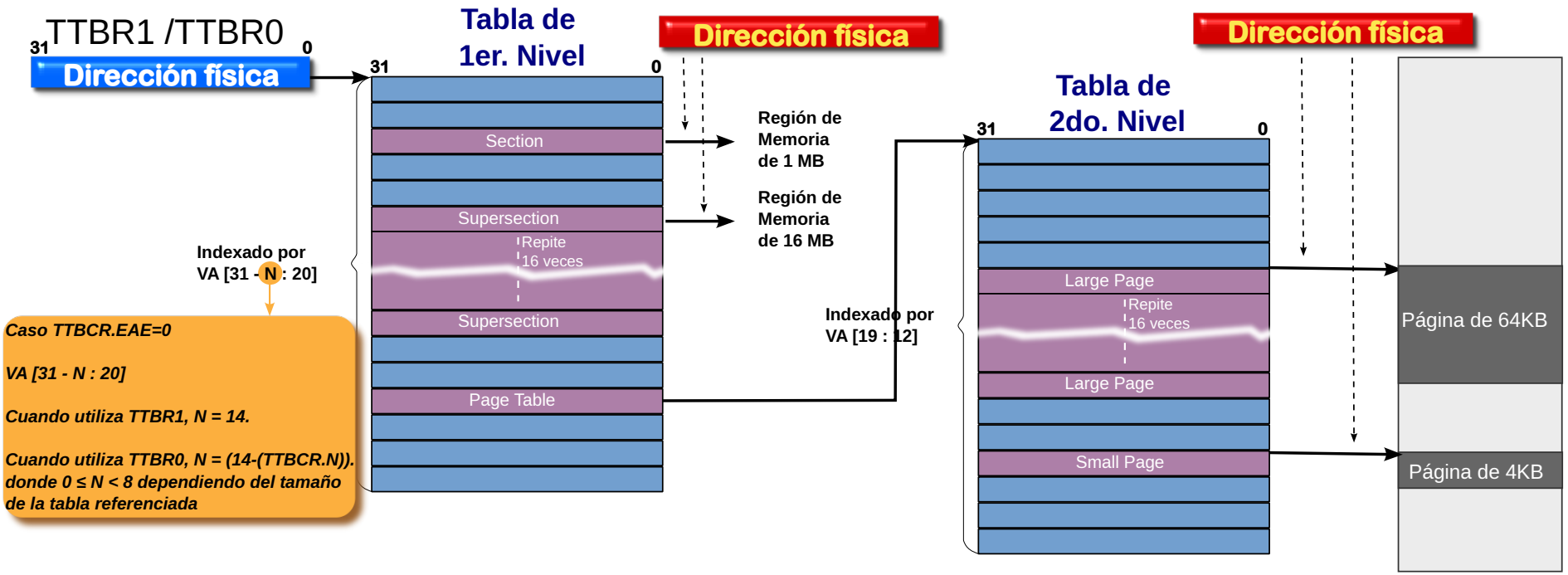
Short-Descriptor Formato de Tablas



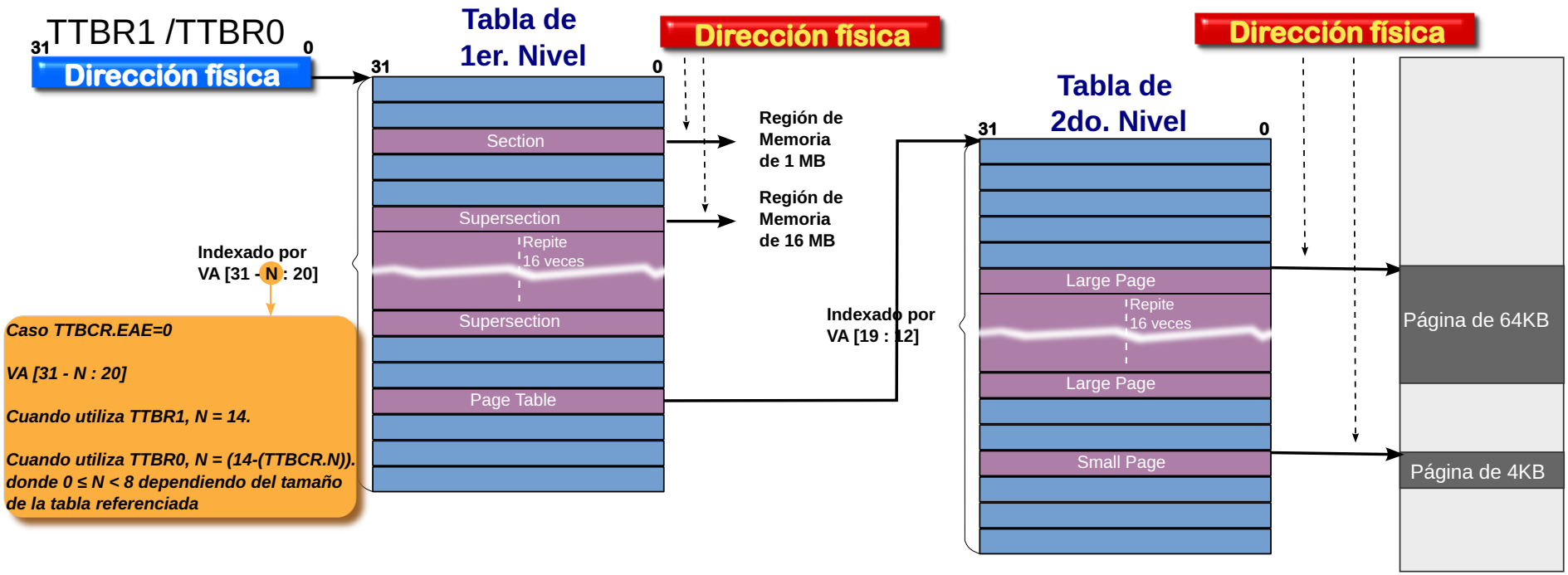
Short-Descriptor Formato de Tablas



Short-Descriptor Formato de Tablas



Short-Descriptor Formato de Tablas



Cada descriptor ubicado en uno u otro de los dos niveles se denomina genéricamente “Entry” por entrada, en referencia a cada entrada de la tabla que contiene un descriptor.

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Invalid o Fault Entry Entrada de formato inválido o *"fault entry"*.

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Invalid o Fault Entry Entrada de formato inválido o *"fault entry"*.

Page Table Entry Apunta a una tabla de traslación de segundo nivel

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Invalid o Fault Entry Entrada de formato inválido o *"fault entry"*.

Page Table Entry Apunta a una tabla de traslación de segundo nivel

Page Entry Define en el segundo nivel de traslación los atributos de una página de memoria física de 4 KiB o de 64 KiB.

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Invalid o Fault Entry Entrada de formato inválido o *"fault entry"*.

Page Table Entry Apunta a una tabla de traslación de segundo nivel

Page Entry Define en el segundo nivel de traslación los atributos de una página de memoria física de 4 KiB o de 64 KiB.

Section Entry Define en el primer nivel de traslación los atributos de una sección de memoria física de 1 MiB de tamaño.

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Invalid o Fault Entry Entrada de formato inválido o *"fault entry"*.

Page Table Entry Apunta a una tabla de traslación de segundo nivel

Page Entry Define en el segundo nivel de traslación los atributos de una página de memoria física de 4 KiB o de 64 KiB.

Section Entry Define en el primer nivel de traslación los atributos de una sección de memoria física de 1 MiB de tamaño.

Supersection Entry Define en el primer nivel de traslación los atributos de una sección de memoria física de 16 MiB de tamaño.

Diferentes clases de entradas

Los diferentes tipos de descriptores que podemos encontrar en las tablas de traslación son:

Invalid o Fault Entry Entrada de formato inválido o *"fault entry"*.

Page Table Entry Apunta a una tabla de traslación de segundo nivel

Page Entry Define en el segundo nivel de traslación los atributos de una página de memoria física de 4 KiB o de 64 KiB.

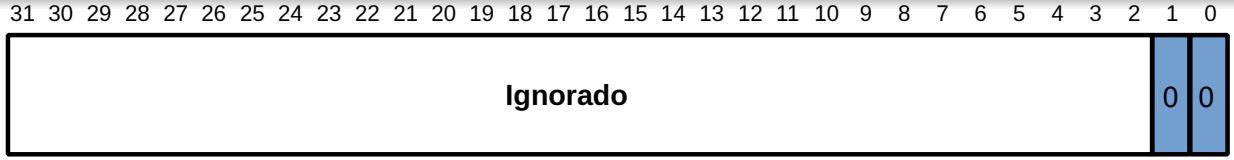
Section Entry Define en el primer nivel de traslación los atributos de una sección de memoria física de 1 MiB de tamaño.

Supersection Entry Define en el primer nivel de traslación los atributos de una sección de memoria física de 16 MiB de tamaño.

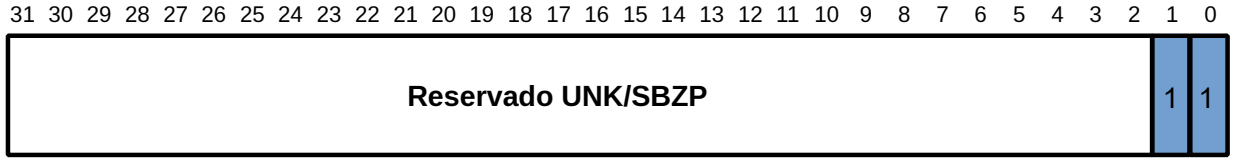
Reserved Format

Formato de las entradas del 1er Nivel de Traslación

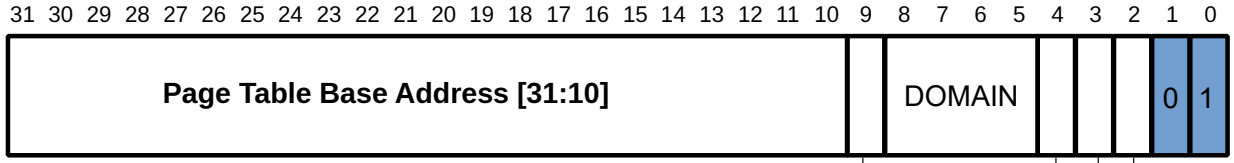
Inválido



Reservado (LPAE no implementado)

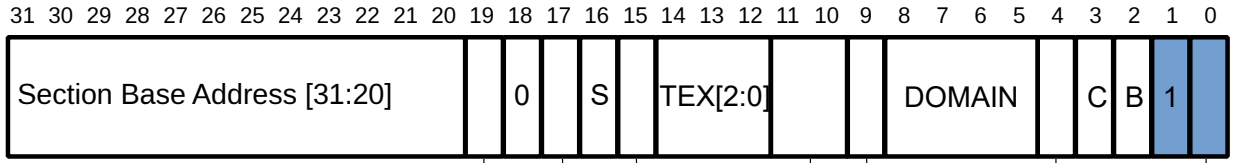


Page Table



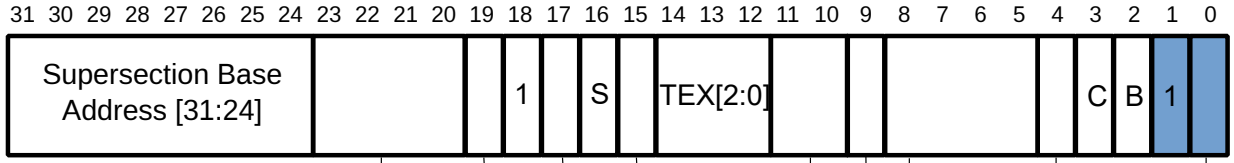
PXN* = SBZ si no soporta PXN

Section



PXN** = 0 si no soporta PXN

Supersection



PXN** = 0 si no soporta PXN

Descriptores de Primer nivel del Traslación

Descriptores de Primer nivel del Traslación

- Cada entrada en el primer nivel de traslación describe el mapeo de un rango de Direcciones virtuales de 1 MiB.

Descriptores de Primer nivel del Traslación

- Cada entrada en el primer nivel de traslación describe el mapeo de un rango de Direcciones virtuales de 1 MiB.
- Los bits [1:0] identifican el tipo de descriptor.

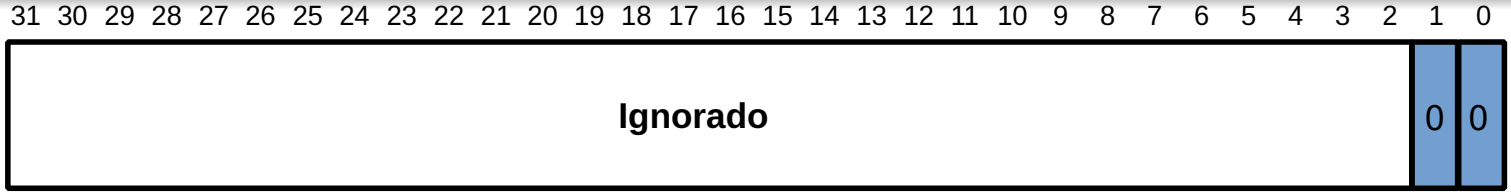
Descriptores de Primer nivel del Traslación

- Cada entrada en el primer nivel de traslación describe el mapeo de un rango de Direcciones virtuales de 1 MiB.
- Los bits [1:0] identifican el tipo de descriptor.

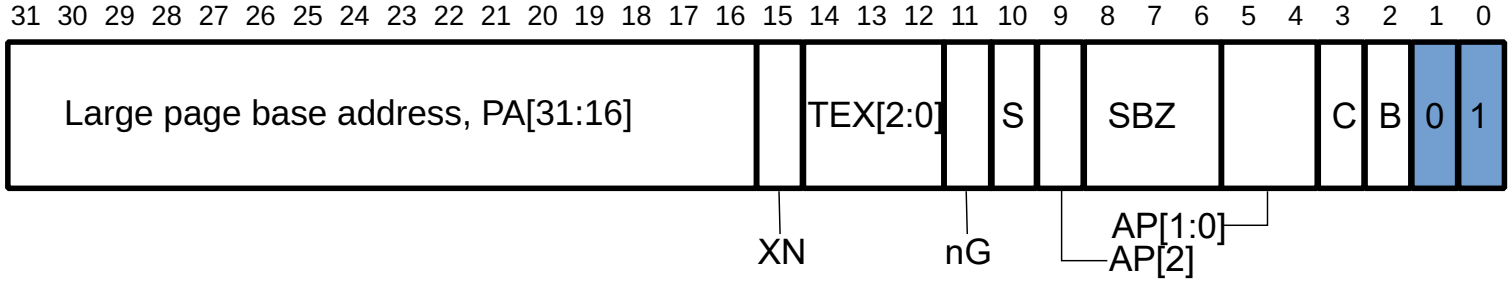
Bits[1:0]	Tipo	Descripción
0b00	Inválido	La VA asociada no está mapeada. Translation Fault. [31:2] ignorados.
0b01	Page Table	El descriptor contiene dirección base y atributos de una tabla de segundo nivel que mapea el rango de VA de 1 MiB
0b10	Sección o Super-sección	El descriptor contiene dirección base y atributos de un bloque de 1 MiB o 16 MiB de acuerdo con el valor del bit [18]. Si soporta ***PXN***, lo define **0**.
0b11	Sección o Super-sección con PXN	Idéntico al anterior pero al soportar ***PXN***, lo define **1**
0b11	Reservado si no soporta PXN	No debe utilizarse si no soporta ***PXN***. Translation Fault

Formato de las entradas del 2do Nivel de Traslación

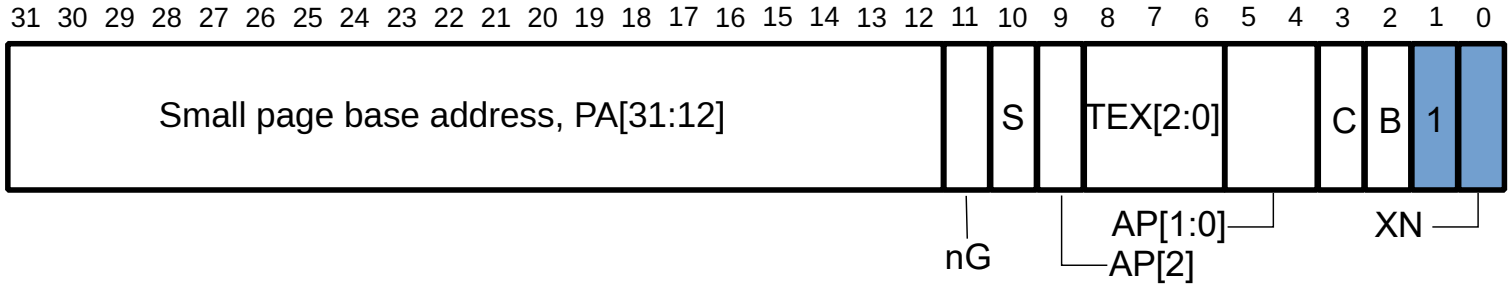
Inválido



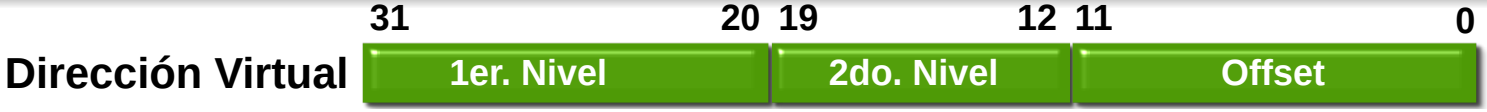
Large Page



Small Page

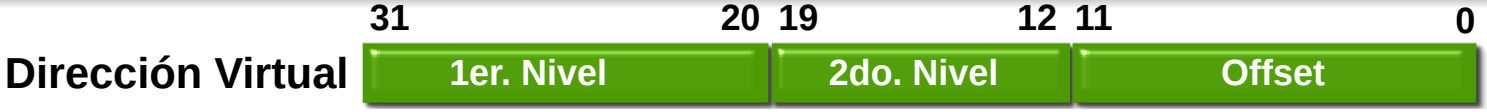


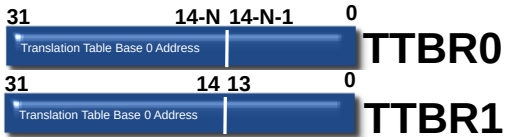
Traslación de Páginas de 4 KiB



Short Descritpion Format
4KiB Page Traslation

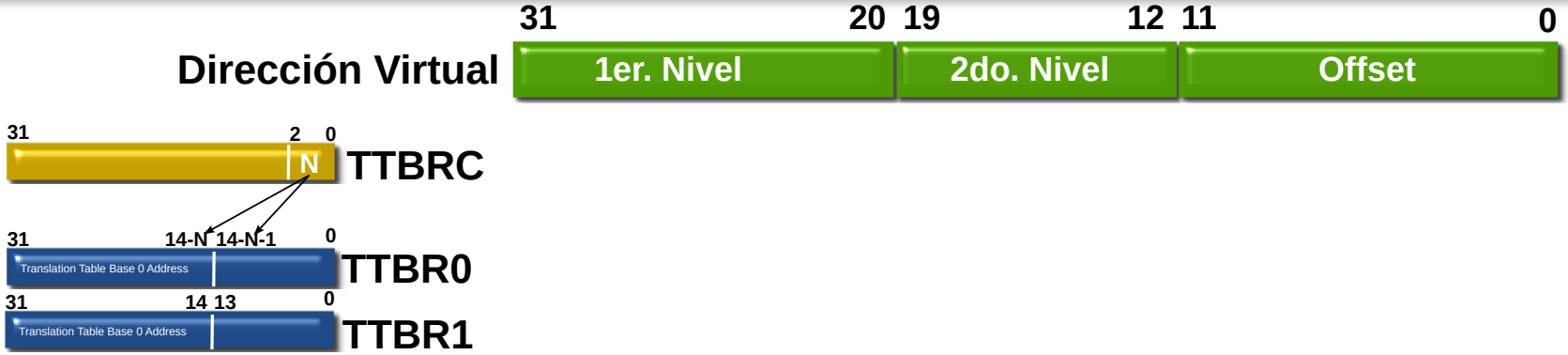
Traslación de Páginas de 4 KiB





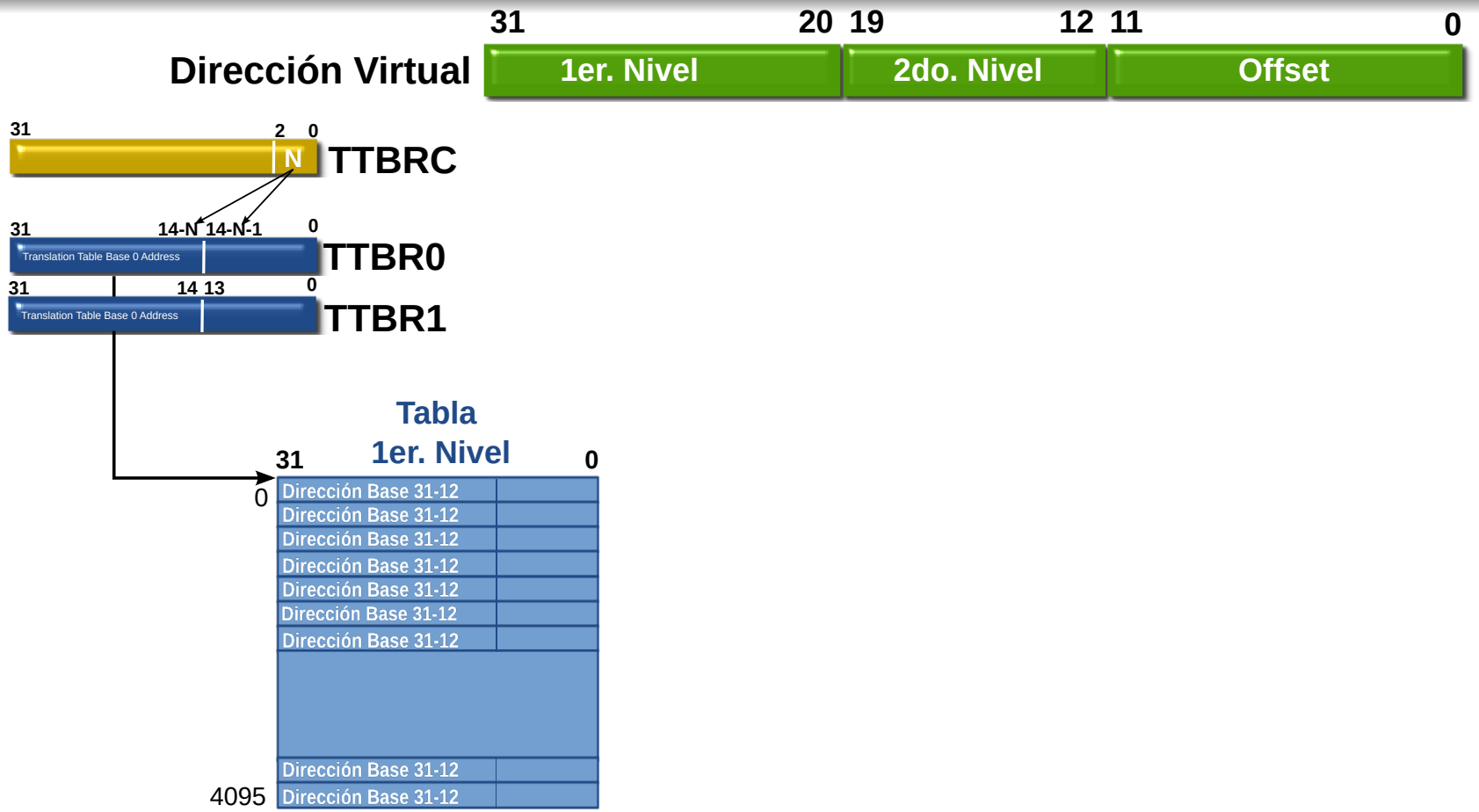
Short Descritpion Format
4KiB Page Traslation

Traslación de Páginas de 4 KiB



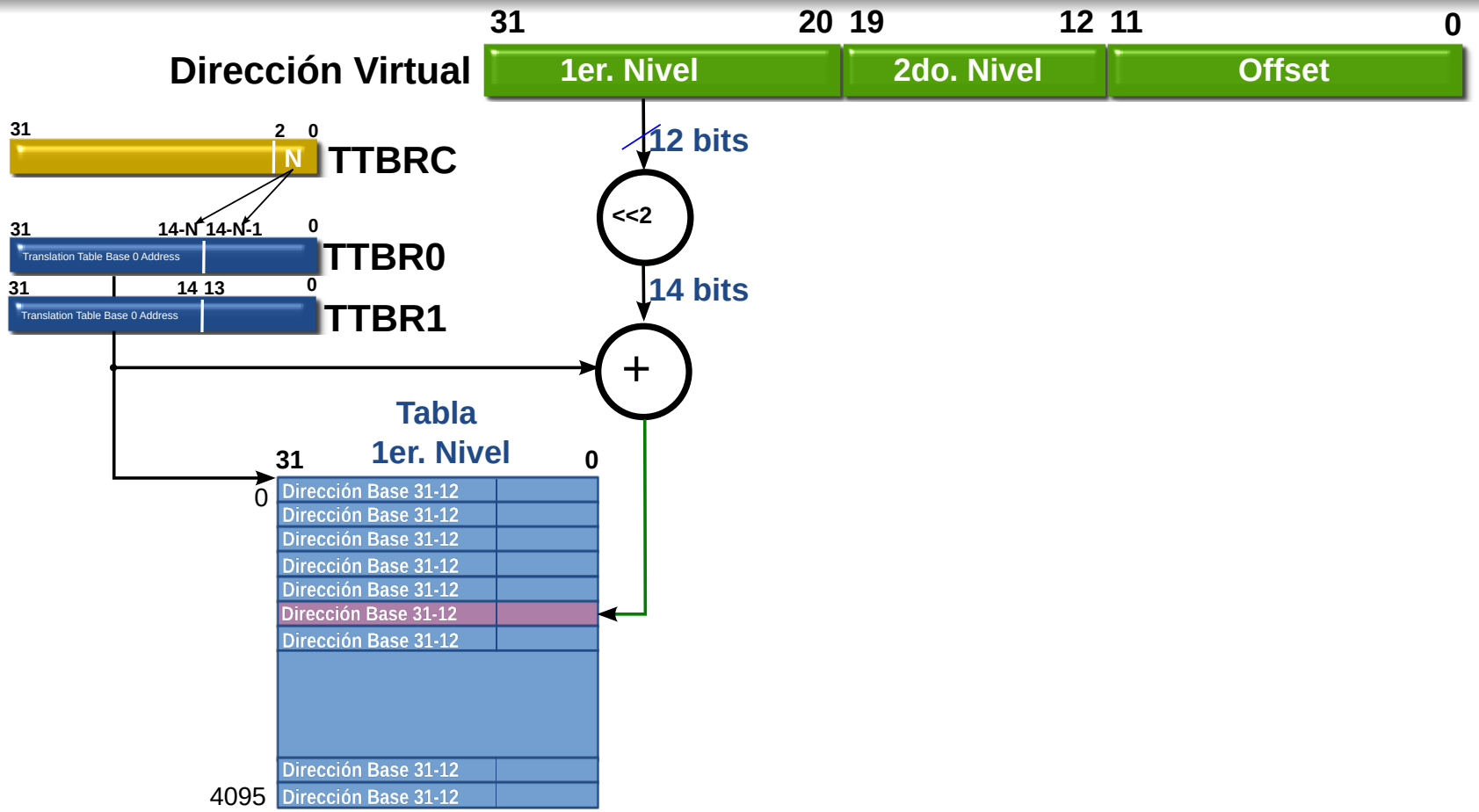
Short Descritpion Format
4KiB Page Traslation

Traslación de Páginas de 4 KiB



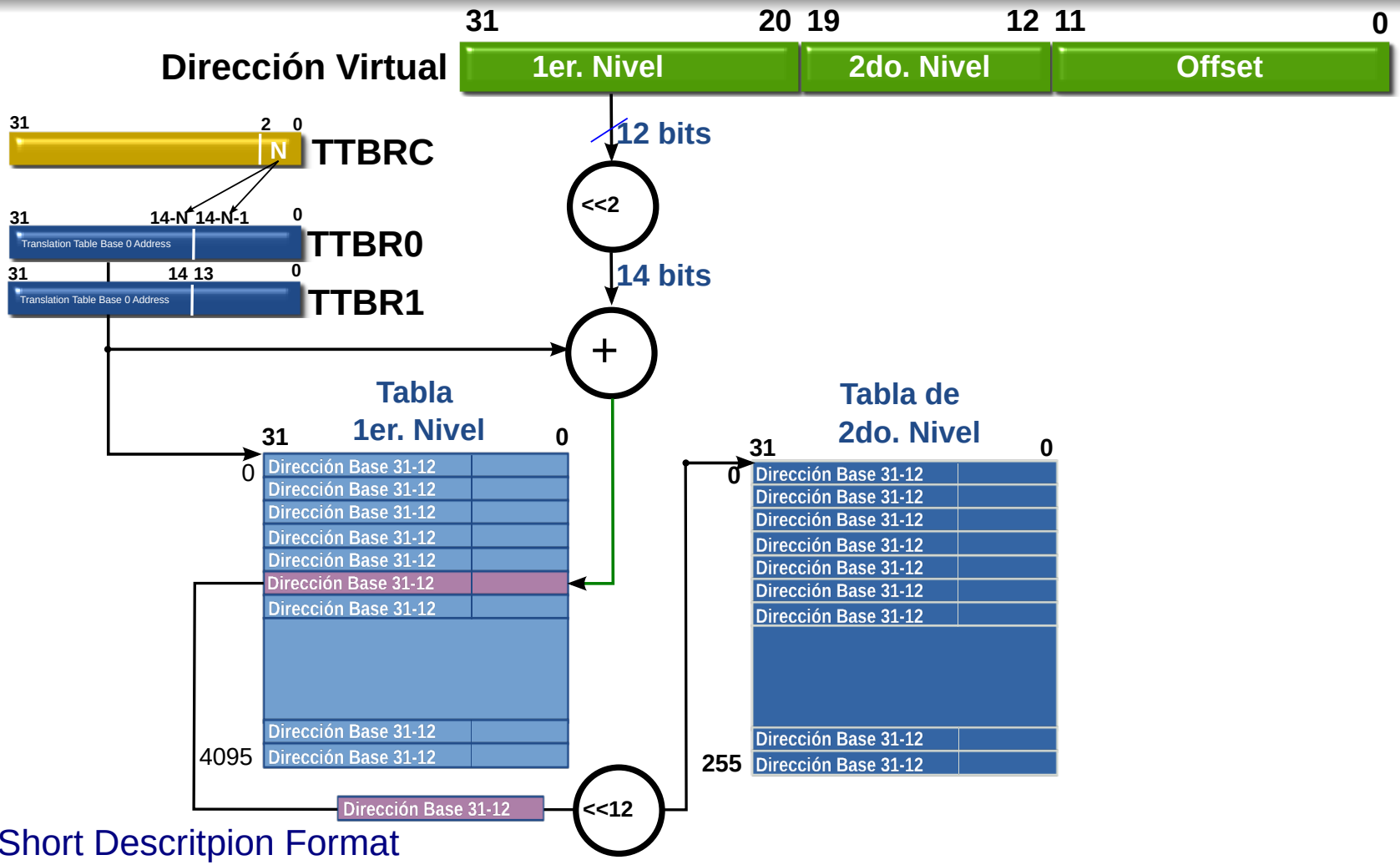
Short Descritpion Format
4KiB Page Traslation

Traslación de Páginas de 4 KiB



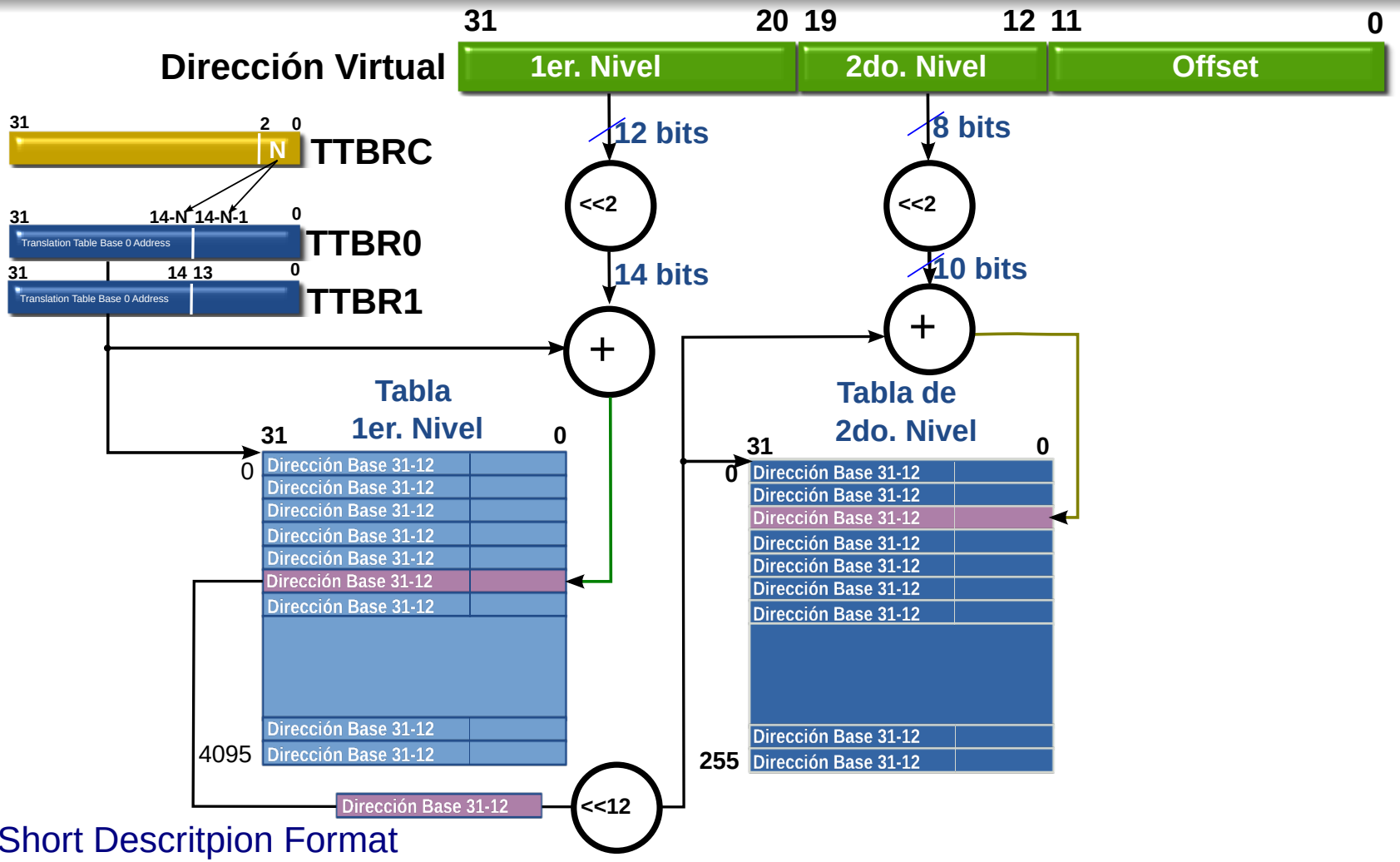
Short Descritpion Format
4KiB Page Traslation

Traslación de Páginas de 4 KiB



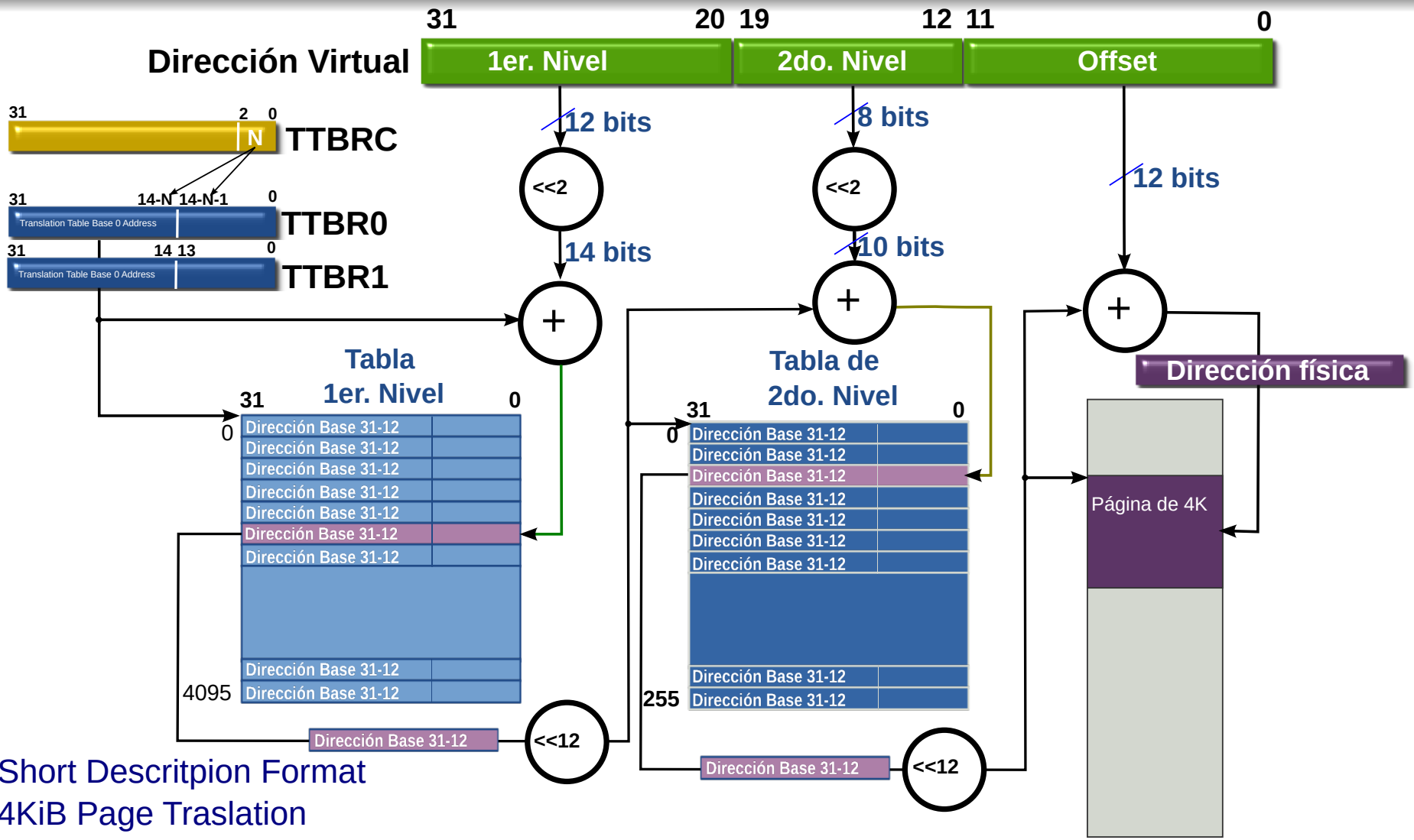
Short Descritpion Format
4KiB Page Traslation

Traslación de Páginas de 4 KiB



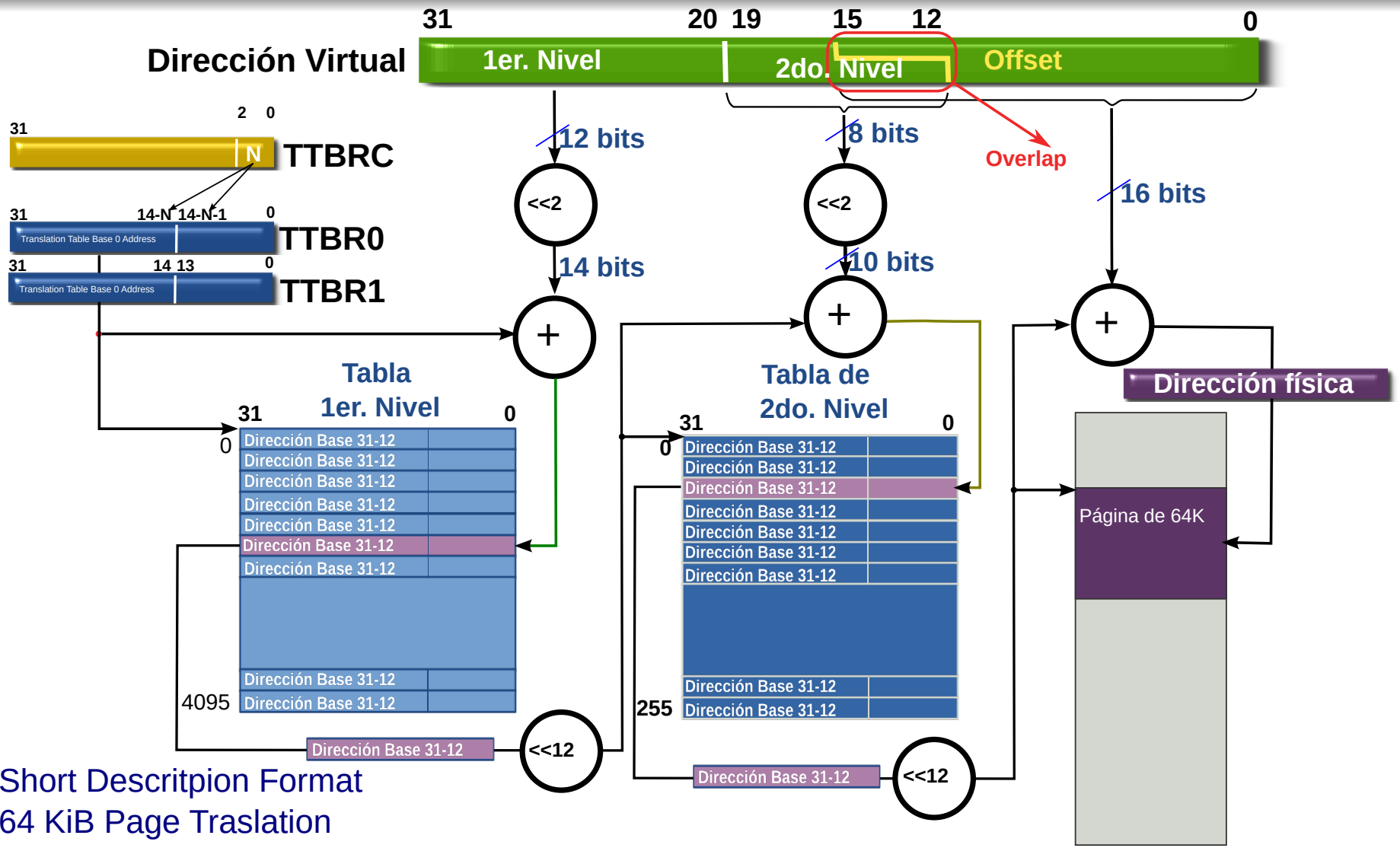
Short Descritpion Format
4KiB Page Traslation

Traslación de Páginas de 4 KiB



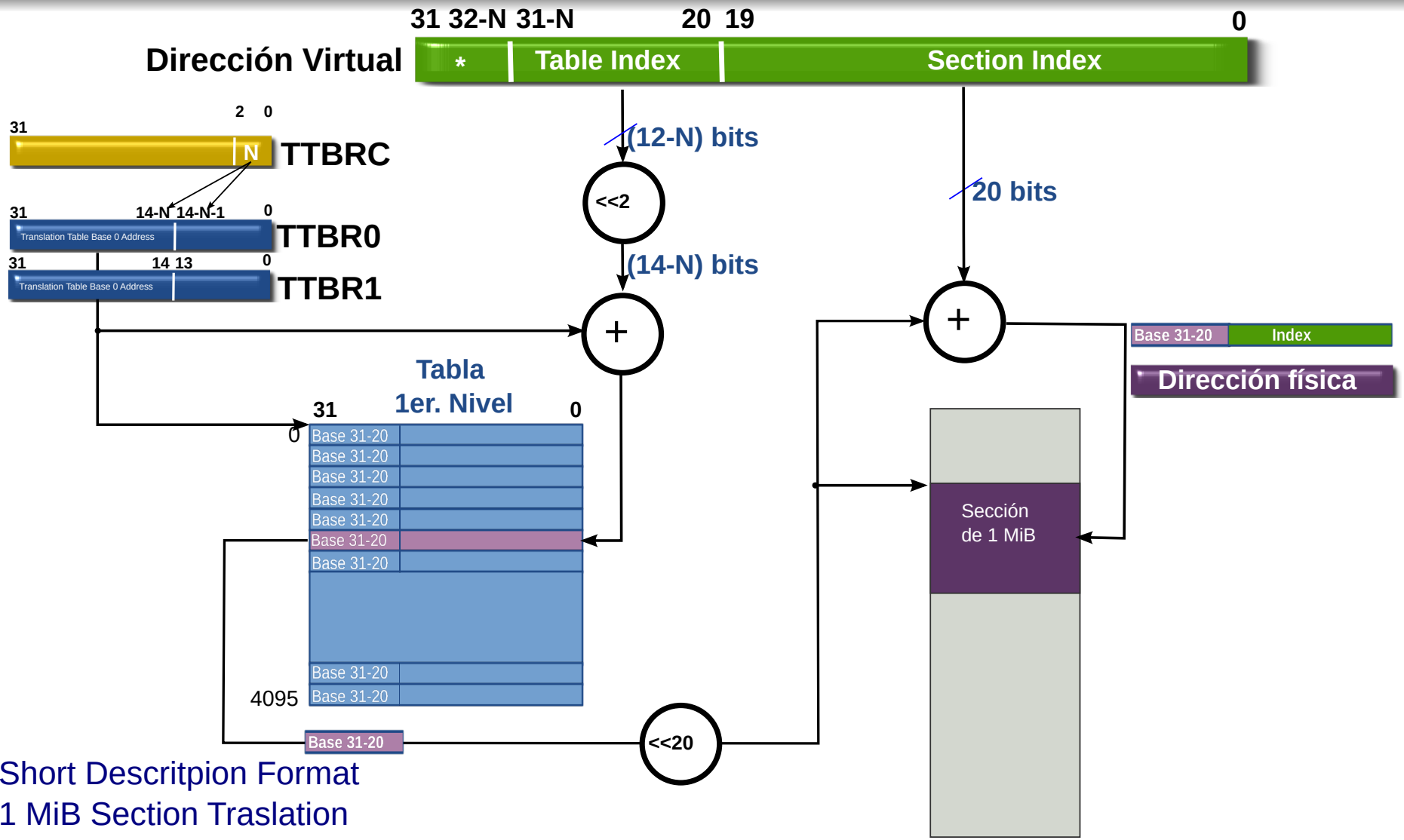
Short Descritpion Format
4KiB Page Traslation

traslación de Páginas de 64 KiB



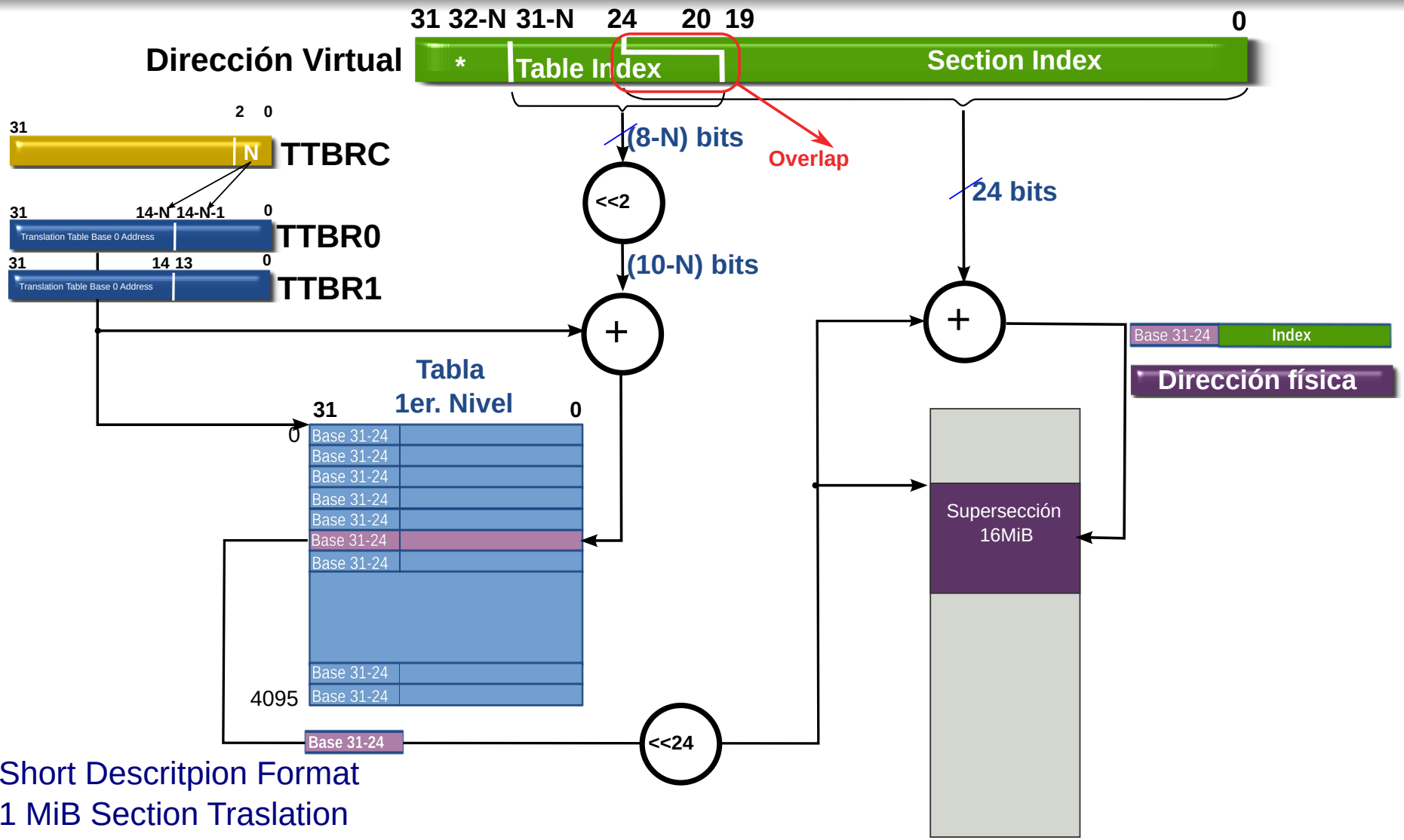
Short Descritpion Format
64 KiB Page Traslation

Traslación de Secciones de 1 MiB



Short Descritpion Format
1 MiB Section Traslation

Traslación de Super Secciones de 16 MiB



Short Descritpion Format
1 MiB Section Traslation

Temario

Acceso a la Tabla de páginas

Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.

Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.
- ARM denomina a esta búsqueda *Translation Table Walk*.

Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.
- ARM denomina a esta búsqueda *Translation Table Walk*.
- El core intentará leer el primer nivel de traslación.

Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.
- ARM denomina a esta búsqueda *Translation Table Walk*.
- El core intentará leer el primer nivel de traslación.
- Como resultado se obtiene una dirección y un set de propiedades, ya sea de la página de salida, o para la siguiente búsqueda.

Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.
- ARM denomina a esta búsqueda *Translation Table Walk*.
- El core intentará leer el primer nivel de traslación.
- Como resultado se obtiene una dirección y un set de propiedades, ya sea de la página de salida, o para la siguiente búsqueda.
- Para iniciar una búsqueda en las tablas de páginas se necesita la dirección de memoria en la que inicia la jerarquía de tablas.

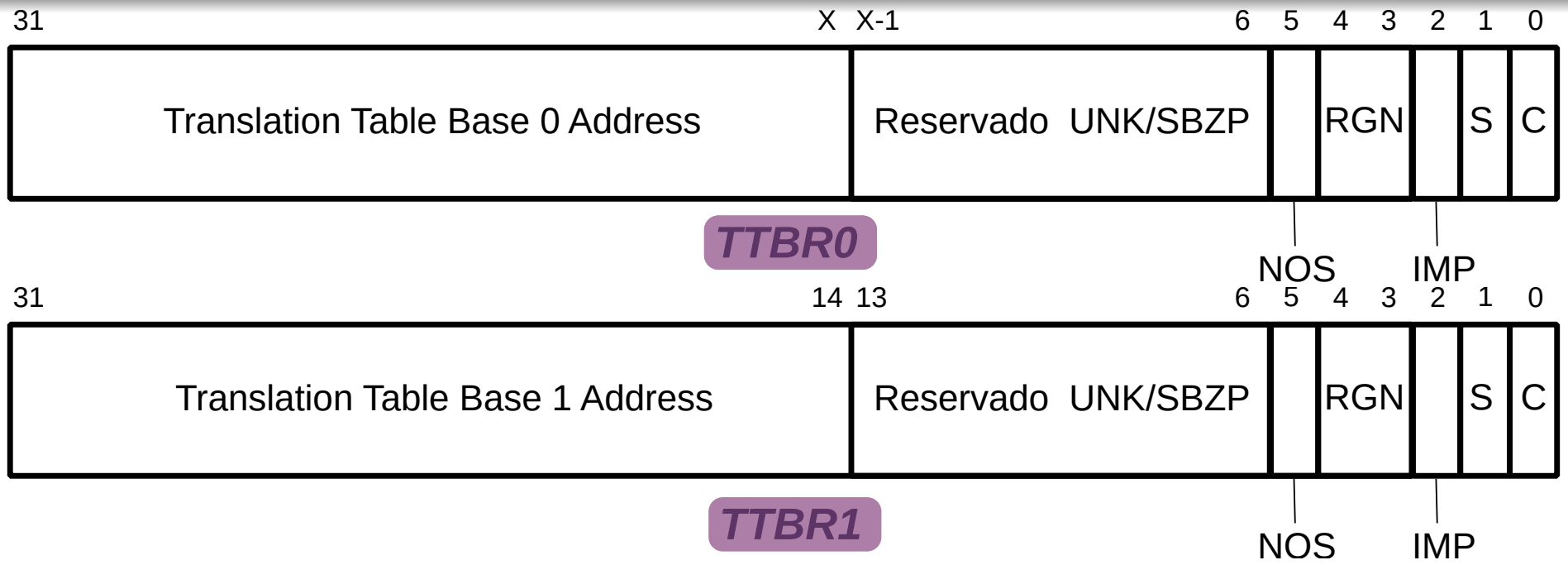
Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.
- ARM denomina a esta búsqueda *Translation Table Walk*.
- El core intentará leer el primer nivel de traslación.
- Como resultado se obtiene una dirección y un set de propiedades, ya sea de la página de salida, o para la siguiente búsqueda.
- Para iniciar una búsqueda en las tablas de páginas se necesita la dirección de memoria en la que inicia la jerarquía de tablas.
- El Coprocesador 15 provee la referencia: **TTBR**'s, por **T**ranslation **T**able **B**ase **R**egiters.

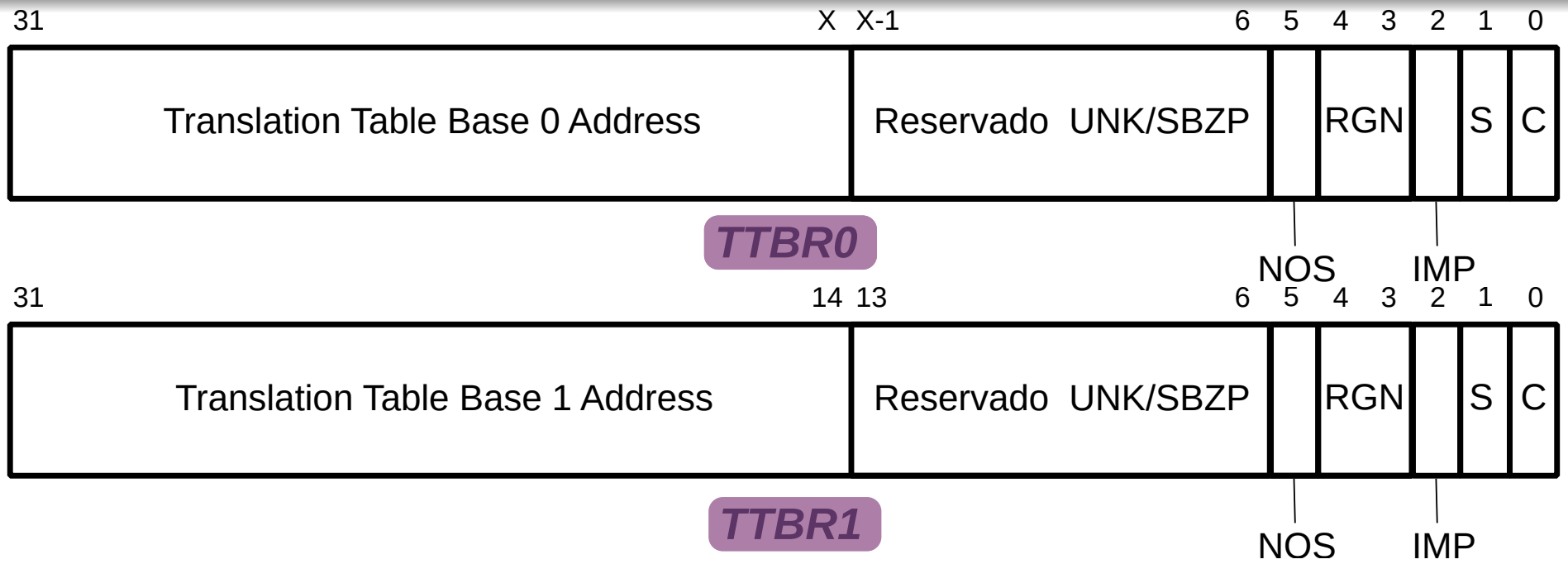
Acceso a la Tabla de páginas

- Si se requiere un descriptor de Página correspondiente a una dirección virtual, y no se lo encuentra en la TLB, hay un Miss en la TLB. Hay que buscarlo en las Tablas de Páginas.
- ARM denomina a esta búsqueda *Translation Table Walk*.
- El core intentará leer el primer nivel de traslación.
- Como resultado se obtiene una dirección y un set de propiedades, ya sea de la página de salida, o para la siguiente búsqueda.
- Para iniciar una búsqueda en las tablas de páginas se necesita la dirección de memoria en la que inicia la jerarquía de tablas.
- El Coprocesador 15 provee la referencia: **TTBR**'s, por **T**ranslation **T**able **B**ase **R**egiters.
- **TTBR0** y **TTBR1**, se utilizan para el Nivel de privilegio 0 y 1 respectivamente y se controlan mediante el registro **TTBCR**

VMSA. Registros de cp15: TTBR0 y TTBR1

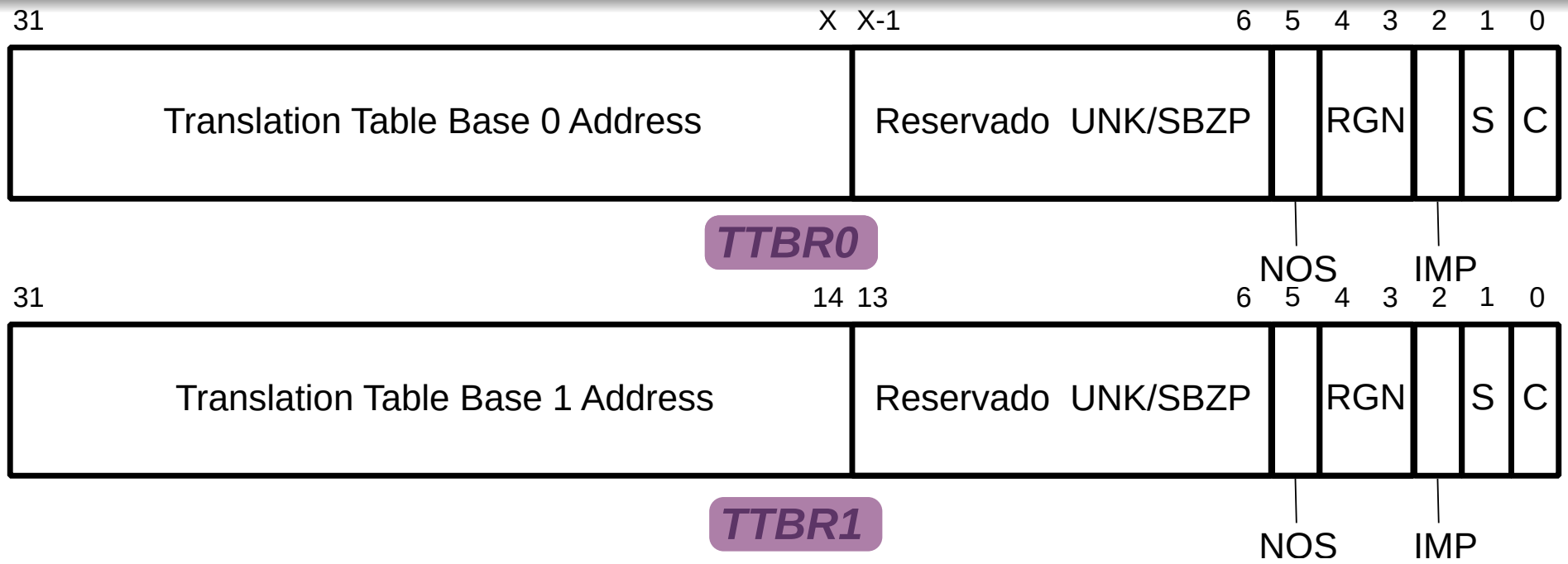


VMSA. Registros de cp15: TTBR0 y TTBR1



- **TTBR0**: Establece la dirección Base de la jerarquía de tablas de páginas proceso-específica. Cambia con cada context switch.

VMSA. Registros de cp15: TTBR0 y TTBR1



- **TTBR0**: Establece la dirección Base de la jerarquía de tablas de páginas proceso-específica. Cambia con cada context switch.
- **TTBR1**: Establece la dirección Base de la jerarquía de Tablas de Páginas para el kernel. No cambia con un context switch. Asegura que todo proceso accede a la traslación de direcciones del kernel cuando ejecuta SVC, o cada vez que debe vectorizarse un handler de excepción o de una interrupción de hardware.

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

Bits[x-1:7] : Reservado (**UNK**nown)/**SBZP**)

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

Bits[x-1:7] : Reservado (**UNK**nown)/**SBZP**)

NOS : **N**o **O**uter **S**hareable. Se ignora si **`TTRB0.S`** == 0. Si **`TTRB0.S`** == 1, **`NOS`** = 0 indica que no es Outer Shareable, y **`NOS`** = 1, Inner Shareable

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

Bits[x-1:7] : Reservado (**UNK**nown)/**SBZP**)

NOS : **N**o **O**uter **S**hareable. Se ignora si **`TTRB0.S`** == 0. Si **`TTRB0.S`** == 1, **`NOS`** = 0 indica que no es Outer Shareable, y **`NOS`** = 1, Inner Shareable

IMP : Implementation Defined

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

Bits[x-1:7] : Reservado (**UNK**nown)/**SBZP**)

NOS : **N**o **O**uter **S**hareable. Se ignora si **`TTRB0.S`** == 0. Si **`TTRB0.S`** == 1, **`NOS`** = 0 indica que no es Outer Shareable, y **`NOS`** = 1, Inner Shareable

IMP : Implementation Defined

S : Shareable

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

Bits[x-1:7] : Reservado (**UNK**nown)/**SBZP**)

NOS : **N**o **O**uter **S**hareable. Se ignora si **`TTRB0.S`** == 0. Si **`TTRB0.S`** == 1, **`NOS`** = 0 indica que no es Outer Shareable, y **`NOS`** = 1, Inner Shareable

IMP : Implementation Defined

S : Shareable

C : Indica el nivel de cache en el que se mantendrán las líneas de esta región de memoria. **`C`**=0 Inner Non Cacheable, **`C`**=1 Inner Cacheable. En principio inner es L1.

VMSA. Registros de cp15: TTBR0 y TTBR1

Bits[31:x] : Dirección Base de la Tabla de traslación de Nivel 1.

Bits[x-1:7] : Reservado (**UNK**nown)/**SBZP**)

NOS : **N**o **O**uter **S**hareable. Se ignora si **`TTRB0.S`** == 0. Si **`TTRB0.S`** == 1, **`NOS`** = 0 indica que no es Outer Shareable, y **`NOS`** = 1, Inner Shareable

IMP : Implementation Defined

S : Shareable

C : Indica el nivel de cache en el que se mantendrán las líneas de esta región de memoria. **`C`**=0 Inner Non Cacheable, **`C`**=1 Inner Cacheable. En principio inner es L1.

Acceso:

```
MRC p15, 0, <Rt>, c2, c0, 0 /* Read 32-bit TTBR0 into Rt */
MCR p15, 0, <Rt>, c2, c0, 0 /* Write Rt to 32-bit TTBR0 */
MRC p15, 0, <Rt>, c2, c0, 1 /* Read 32-bit TTBR1 into Rt */
MCR p15, 0, <Rt>, c2, c0, 1 /* Write Rt to 32-bit TTBR1 */
```

VMSA. Registros de cp15: TTBCR

VMSA. Registros de cp15: TTBCR

- **EAE**: **E**nable **A**ddress **E**xtensions. ’1’ habilita. Existe solo si el core soporta Large Physical Address Extensions. De otro modo se suma a los bit 3 a 30, es decir **UNK**nown / **S**hould **B**e **Z**ero or **P**reserved).

VMSA. Registros de cp15: TTBCR

- **EAE**: **E**nable **A**ddress **E**xtensions. '1' habilita. Existe solo si el core soporta Large Physical Address Extensions. De otro modo se suma a los bit 3 a 30, es decir **UNK**nown / **S**hould **B**e **Z**ero or **P**reserved).
- **N**: Es un valor de 000b a 111b, que regula el ancho en bits de la dirección base mantenida en el registro **TTBR0**, Este valor va desde el bit 31 al bit 14-N. Cuando N=000b, la dirección base de la tabla de traslación es compatible con ARMv5 y ARMv6.

VMSA. Registros de cp15: TTBCR

- **EAE**: **E**nable **A**ddress **E**xtensions. '1' habilita. Existe solo si el core soporta Large Physical Address Extensions. De otro modo se suma a los bit 3 a 30, es decir **UNK**nown / **S**hould **B**e **Z**ero or **P**reserved).
- **N**: Es un valor de 000b a 111b, que regula el ancho en bits de la dirección base mantenida en el registro **TTBR0**, Este valor va desde el bit 31 al bit 14-N. Cuando N=000b, la dirección base de la tabla de traslación es compatible con ARMv5 y ARMv6.

Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c2, c0, 2 ; Read TTBCR into Rt
MCR p15, 0, <Rt>, c2, c0, 2 ; Write RT to TTBCR
```

Temario

Control de la Memoria

Control de la Memoria

- Una entrada de la tabla de traslación que se refiere a una región de memoria, además de una dirección de salida, incluye campos que definen las propiedades de dicha región.

Control de la Memoria

- Una entrada de la tabla de traslación que se refiere a una región de memoria, además de una dirección de salida, incluye campos que definen las propiedades de dicha región.
- La información devuelta por una búsqueda de tabla de traslación describe la clasificación de esos campos como control de mapa de direcciones, control de acceso y campos de atributos de memoria.

Control de la Memoria

- Una entrada de la tabla de traslación que se refiere a una región de memoria, además de una dirección de salida, incluye campos que definen las propiedades de dicha región.
- La información devuelta por una búsqueda de tabla de traslación describe la clasificación de esos campos como control de mapa de direcciones, control de acceso y campos de atributos de memoria.
- Los campos de control de acceso, determinan si el procesador, en su estado actual, puede realizar el acceso requerido a la dirección de salida dada en el descriptor de la tabla de traslación.

Control de la Memoria

- Una entrada de la tabla de traslación que se refiere a una región de memoria, además de una dirección de salida, incluye campos que definen las propiedades de dicha región.
- La información devuelta por una búsqueda de tabla de traslación describe la clasificación de esos campos como control de mapa de direcciones, control de acceso y campos de atributos de memoria.
- Los campos de control de acceso, determinan si el procesador, en su estado actual, puede realizar el acceso requerido a la dirección de salida dada en el descriptor de la tabla de traslación.
- Si una etapa de traslación no permite el acceso, se genera un fallo para esa etapa de traslación y no se realiza ningún acceso a la memoria.

Control de la Memoria

El control de la memoria se lleva a cabo mediante

Control de la Memoria

El control de la memoria se lleva a cabo mediante

- Permisos de Acceso

Control de la Memoria

El control de la memoria se lleva a cabo mediante

- Permisos de Acceso
- Restricciones sobre el fetch de instrucciones (Execute-Never).

Control de la Memoria

El control de la memoria se lleva a cabo mediante

- Permisos de Acceso
- Restricciones sobre el fetch de instrucciones (Execute-Never).
- Dominios (Solo en el formato Short-Descriptor)

Control de la Memoria

El control de la memoria se lleva a cabo mediante

- Permisos de Acceso
- Restricciones sobre el fetch de instrucciones (Execute-Never).
- Dominios (Solo en el formato Short-Descriptor)
- Los Flags de acceso.

Control de la Memoria

El control de la memoria se lleva a cabo mediante

- Permisos de Acceso
- Restricciones sobre el fetch de instrucciones (Execute-Never).
- Dominios (Solo en el formato Short-Descriptor)
- Los Flags de acceso.
- Aunque las extensiones de virtualización no están dentro de nuestro alcance, si se activan, la **VMSA** chequea el **PL2** en el estado No Seguro.

Permisos de acceso

Controlan el acceso a las regiones de memoria definidas por el descriptor. En el formato Short-Descriptor hay dos posibilidades con los bits **`AP[2:0]`**:

Permisos de acceso

Controlan el acceso a las regiones de memoria definidas por el descriptor. En el formato Short-Descriptor hay dos posibilidades con los bits **`AP[2:0]`**:

Tres Bits: AP [2:0] Los tres Bits se utilizan para definir los permisos.

Permisos de acceso

Controlan el acceso a las regiones de memoria definidas por el descriptor. En el formato Short-Descriptor hay dos posibilidades con los bits **`AP[2:0]`**:

Tres Bits: AP [2:0] Los tres Bits se utilizan para definir los permisos.

Dos Bits: AP [2:1] Se utilizan los dos mas significativos para definir los permisos, y AP[0] se utiliza como bit de acceso.

Permisos de acceso

Controlan el acceso a las regiones de memoria definidas por el descriptor. En el formato Short-Descriptor hay dos posibilidades con los bits **`AP[2:0]`**:

Tres Bits: AP [2:0] Los tres Bits se utilizan para definir los permisos.

Dos Bits: AP [2:1] Se utilizan los dos mas significativos para definir los permisos, y AP[0] se utiliza como bit de acceso.

La configuración del comportamiento de AP[0] se define mediante el bit **`SCTLR.AFE`**.

Permisos de acceso

Controlan el acceso a las regiones de memoria definidas por el descriptor. En el formato Short-Descriptor hay dos posibilidades con los bits **`AP[2:0]`**:

Tres Bits: AP [2:0] Los tres Bits se utilizan para definir los permisos.

Dos Bits: AP [2:1] Se utilizan los dos mas significativos para definir los permisos, y AP[0] se utiliza como bit de acceso.

La configuración del comportamiento de AP[0] se define mediante el bit **`SCTLR.AFE`**.
`SCTLR.AFE` = 0: Participa por Bit de permisos.

Permisos de acceso

Controlan el acceso a las regiones de memoria definidas por el descriptor. En el formato Short-Descriptor hay dos posibilidades con los bits **AP[2:0]**:

Tres Bits: AP [2:0] Los tres Bits se utilizan para definir los permisos.

Dos Bits: AP [2:1] Se utilizan los dos mas significativos para definir los permisos, y AP[0] se utiliza como bit de acceso.

La configuración del comportamiento de AP[0] se define mediante el bit **SCTLR.AFE**.
SCTLR.AFE = 0: Participa por Bit de permisos.
SCTLR.AFE = 1: Participa como bit de acceso.

Permisos de acceso AP[2:1]

AP[2] Write	AP[1] PL	Acceso
0	0	Lectura / Escritura solo en PL1.
0	1	Lectura Escritura en cualquier PL
1	0	Solo lectura solo en PL1.
1	1	Solo lectura en cualquier PL

Permisos de acceso AP[2:1]

AP[2] Write	AP[1] PL	Acceso
0	0	Lectura / Escritura solo en PL1.
0	1	Lectura Escritura en cualquier PL
1	0	Solo lectura solo en PL1.
1	1	Solo lectura en cualquier PL

AP[0]: Bit de Acceso. Si se lo inicializa en 0 sirve para señalar el primer acceso a la página (2do. Nivel) o a la región (1er. Nivel). Setear **`SCTLR.HA`**.

Permisos de acceso AP[2:0]

AP[2]	AP[1:0]	PL1 Access	Unp.Access	Descripción
0	00	No access	No access	Cualquier acceso genera Falta.
	01	R/W	No access	Acceso solo en PL1.
	10	R/W	RdOnly	Write en PL0 genera Falta.
	11	R/W	R/W	Acceso full.
1	00	-	-	Reservado.
	01	RdOnly	No access	Read Only solo en PL1.
	10	RdOnly	RdOnly	Read Only en cualquier PL (obsoleto).
	11	RdOnly	RdOnly	Read Only en cualquier PL.

Atributos de Región de Memoria (traslación Nivel 1)

Atributos de Región de Memoria (traslación Nivel 1)

- Una posibilidad es expresarlos directamente en los bits de los descriptores de tabla.

Atributos de Región de Memoria (traslación Nivel 1)

- Una posibilidad es expresarlos directamente en los bits de los descriptores de tabla.
- La otra posibilidad es definirlos indirectamente mediante un conjunto de registros referenciados por bits de Re mapeo de memoria.

Atributos de Región de Memoria (traslación Nivel 1)

- Una posibilidad es expresarlos directamente en los bits de los descriptores de tabla.
- La otra posibilidad es definirlos indirectamente mediante un conjunto de registros referenciados por bits de Re mapeo de memoria.
- Para seleccionar uno de estos dos abordajes en el formato Short-Descriptor, se utiliza el bit **`SCTLR.TRE`**.

Atributos de Región de Memoria (traslación Nivel 1)

- Una posibilidad es expresarlos directamente en los bits de los descriptores de tabla.
- La otra posibilidad es definirlos indirectamente mediante un conjunto de registros referenciados por bits de Re mapeo de memoria.
- Para seleccionar uno de estos dos abordajes en el formato Short-Descriptor, se utiliza el bit **`SCTLR.TRE`**.
  - Si **`SCTLR.TRE`** = 0, trabaja en forma directa y los atributos se expresan en **`TEX[2:0]`**, **`C`**, y **`B`** son los bits que se utilizan para este propósito.

Atributos de Región de Memoria (traslación Nivel 1)

- Una posibilidad es expresarlos directamente en los bits de los descriptores de tabla.
- La otra posibilidad es definirlos indirectamente mediante un conjunto de registros referenciados por bits de Re mapeo de memoria.
- Para seleccionar uno de estos dos abordajes en el formato Short-Descriptor, se utiliza el bit **SCTLR.TRE**.
  - Si **SCTLR.TRE** = 0, trabaja en forma directa y los atributos se expresan en **TEX[2:0]**, **C**, y **B** son los bits que se utilizan para este propósito.
  - Si **SCTLR.TRE** = 1, se activa el remapeo y los bits **TEX[2:0]**, **C**, y **B** del descriptor operan como índices para dos conjuntos de registros del cp15: **PRRR**, Primary Region Remap Register, y **NMRR**, Normal Memory Remap Regiter.

Atributos de Región de Memoria sin Tex Remap

TEX[2:0]	C	B	Descripción	Tipo de Mem.	Pág. Shareable
000	0	0	Strongly Orderer	Strongly Orderer	Shareable.
		1	Shareable Device	Device	Shareable.
	1	0	Outer and Inner Write-Through, no Write-Allocate	Normal	bit S.
		1	Outer and Inner Write-Back, no Write-Allocate	Normal	bit S.
001	0	0	Outer and Inner Non-cacheable	Normal	bit S.
		1	Reservado	-	-
	1	0	Implementation Defined	Implementation Defined	Implementation Defined
		1	Outer and Inner Write-Back, Write-Allocate	Normal	bit S.
010	0	0	Non-shareable Device	Device	Non-shareable
		1	Reserved	-	-
	1	x	Reserved	-	-
011	x	x	Reserved	-	-
1BB	A	A	Cacheable Memory AA=Inner Attribute, BB=Outer Attribute	Normal	bit S

Atributos de Región de Memoria con Tex Remap

TEX[0]	C	B	Mem.Type	Inner Cache.	Outer Cache	Outer Shareable
0	0	0	PRRR[1:0]	NMRR[1:0]	NMRR[17:16]	NOT(PRRR[24])
		1	PRRR[3:2]	NMRR[3:2]	NMRR[19:18]	NOT(PRRR[25])
	1	0	PRRR[5:4]	NMRR[5:4]	NMRR[21:20]	NOT(PRRR[26])
		1	PRRR[7:6]	NMRR[7:6]	NMRR[23:22]	NOT(PRRR[27])
1	0	0	PRRR[9:8]	NMRR[9:8]	NMRR[25:24]	NOT(PRRR[28])
		1	PRRR[11:10]	NMRR[11:10]	NMRR[27:26]	NOT(PRRR[29])
	1	0	Imp.Def.	Imp.Def.	Imp.Def.	Imp.Def.
		1	PRRR[15:14]	NMRR[15:14]	NMRR[13:30]	NOT(PRRR[31])

Atributos de Región de Memoria con Tex Remap

TEX[0]	C	B	Mem.Type	Inner Cache.	Outer Cache	Outer Shareable
0	0	0	PRRR[1:0]	NMRR[1:0]	NMRR[17:16]	NOT(PRRR[24])
		1	PRRR[3:2]	NMRR[3:2]	NMRR[19:18]	NOT(PRRR[25])
	1	0	PRRR[5:4]	NMRR[5:4]	NMRR[21:20]	NOT(PRRR[26])
		1	PRRR[7:6]	NMRR[7:6]	NMRR[23:22]	NOT(PRRR[27])
1	0	0	PRRR[9:8]	NMRR[9:8]	NMRR[25:24]	NOT(PRRR[28])
		1	PRRR[11:10]	NMRR[11:10]	NMRR[27:26]	NOT(PRRR[29])
	1	0	Imp.Def.	Imp.Def.	Imp.Def.	Imp.Def.
		1	PRRR[15:14]	NMRR[15:14]	NMRR[13:30]	NOT(PRRR[31])

- Cada combinación {TEX[0],C,B} del descriptor no define atributos en forma directa sino que mapea éstos en los bits o campos de bits de los registros PRRR, y/o NMRR indicados en cada caso en la tabla.

Atributos de Región de Memoria con Tex Remap

TEX[0]	C	B	Mem.Type	Inner Cache.	Outer Cache	Outer Shareable
0	0	0	PRRR[1:0]	NMRR[1:0]	NMRR[17:16]	NOT(PRRR[24])
		1	PRRR[3:2]	NMRR[3:2]	NMRR[19:18]	NOT(PRRR[25])
	1	0	PRRR[5:4]	NMRR[5:4]	NMRR[21:20]	NOT(PRRR[26])
		1	PRRR[7:6]	NMRR[7:6]	NMRR[23:22]	NOT(PRRR[27])
1	0	0	PRRR[9:8]	NMRR[9:8]	NMRR[25:24]	NOT(PRRR[28])
		1	PRRR[11:10]	NMRR[11:10]	NMRR[27:26]	NOT(PRRR[29])
	1	0	Imp.Def.	Imp.Def.	Imp.Def.	Imp.Def.
		1	PRRR[15:14]	NMRR[15:14]	NMRR[13:30]	NOT(PRRR[31])

- Cada combinación {TEX[0],C,B} del descriptor no define atributos en forma directa sino que mapea éstos en los bits o campos de bits de los registros PRRR, y/o NMRR indicados en cada caso en la tabla.
- Imp.Def. Implementation Defined. Cada bit o campo de bits en los registros PRRR, y/o NMRR mapeado con el valor 6 de la combinación {TEX[0],C,B}, será también Implementation Defined.

Concepto de región “Shareable”

Concepto de región “Shareable”

Los procesadores modernos de arquitecturas avanzadas disponen de recursos para Multiprocesamiento, donde cada core es normalmente Superescalar (o multi issue), y ejecutan fuera de orden, e implementan ejecución especulativa. En este contexto tecnológico una secuencia de escrituras en memoria debe realizarse en un orden específico bajo determinadas condiciones.

Concepto de región “Shareable”

Los procesadores modernos de arquitecturas avanzadas disponen de recursos para Multiprocesamiento, donde cada core es normalmente Superescalar (o multi issue), y ejecutan fuera de orden, e implementan ejecución especulativa. En este contexto tecnológico una secuencia de escrituras en memoria debe realizarse en un orden específico bajo determinadas condiciones.

ARM define el concepto de Dominio de Compartibilidad (***Shareability Domain***), como “zonas” dentro de la topología del bus dentro de las cuales los accesos a memoria deben mantenerse consistentes (esto es, deben tener lugar de manera predecible) y potencialmente coherentes (con soporte de hardware). Fuera de este dominio, es posible que los observadores no vean el mismo orden de accesos a la memoria que dentro de él.

Concepto de región “Shareable”

NSH Non-Shareable. Se trata de dominios de memoria que consisten de solo un agente. Los accesos no necesitan ser coordinados con otros cores, procesadores, o dispositivos. No se emplea en sistemas SMP.

Concepto de región “Shareable”

NSH Non-Shareable. Se trata de dominios de memoria que consisten de solo un agente. Los accesos no necesitan ser coordinados con otros cores, procesadores, o dispositivos. No se emplea en sistemas SMP.

ISH Inner-Shareable. Se trata de dominios potencialmente compartidos con otros agentes pero usualmente no con todos los agentes del sistema. Un sistema puede tener múltiples dominios shareables. Estos son independientes. Si una operación afecta a uno de los dominios ISH, el resto no se verá afectado.

Concepto de región “Shareable”

OSH Outer-Shareable. Consisten de varios dominios Iner-Shareable. Una operación que afecta un Outer-Shareable, afectará a todos los inner que esté contenidos en aquel. No así una operación sobre un Inner-Shareable. En procesadores como el A15 MPCore, por ejemplo, que implementa LPAE, todos los dispositivos de Memoria DRAM son Outer-Shareable. En otros cores se puede configurar en Shareable o Non-Shareable. Para soportar OSH, un dispositivo debe tener un complejo soporte de Hardware, que soporte Memory Management y Coherencia de Cache.

Concepto de región “Shareable”

OSH Outer-Shareable. Consisten de varios dominios Iner-Shareable. Una operación que afecta un Outer-Shareable, afectará a todos los inner que esté contenidos en aquel. No así una operación sobre un Inner-Shareable. En procesadores como el A15 MPCore, por ejemplo, que implementa LPAE, todos los dispositivos de Memoria DRAM son Outer-Shareable. En otros cores se puede configurar en Shareable o Non-Shareable. Para soportar OSH, un dispositivo debe tener un complejo soporte de Hardware, que soporte Memory Management y Coherencia de Cache.

SY Full System. Una operación en Full System afecta a todos los agentes del sistema; es decir a todas las regiones Non-Shareable, todas las regiones Inner-Shareable, y todas las regiones Outer-Shareable. Normalmente no es necesario tener en un dominio de compartibilidad restringida, periféricos simples como UART, u otros varios más complejos.

Restricciones en instruction fetching: Execute-never

Restricciones en instruction fetching: Execute-never

XN e**X**cecute **N**ever bit. Impide (cuando es '1') que se ejecute código en esa página o sección siempre que la página tenga **`Domain`** Client. Un opcode fetch en una página con **`XN`** = 1 genera una Excepción *Permision Fault*. No existe en la Page Table Entry (1er. Nivel de traslación).

Restricciones en instruction fetching: Execute-never

XN eXcecute Never bit. Impide (cuando es '1') que se ejecute código en esa página o sección siempre que la página tenga **`Domain`** Client. Un opcode fetch en una página con **`XN`** = 1 genera una Excepción *Permision Fault*. No existe en la Page Table Entry (1er. Nivel de traslación).

PXN Priviledge eXcecute Never bit. Cuando está soportado por la versión del procesador, se comporta como **`XN`**, pero si el opcode fecth se realiza desde PL1.

Atributos de los descriptores (1er y 2do Nivel)

Atributos de los descriptores (1er y 2do Nivel)

NS **N**on **S**ecure bit. Si el procesador tiene activas las Extensiones de Seguridad, este bit indica en los accesos a memoria hechas desde el Estado Seguro, si la región de memoria accedida pertenece o no a un espacio de memoria definido como Seguro. Solo existe en el 1er nivel de traslación y afecta a todas las páginas del 2do. nivel derivadas de éste descriptor.

Atributos de los descriptores (1er y 2do Nivel)

NS **N**on **S**ecure bit. Si el procesador tiene activas las Extensiones de Seguridad, este bit indica en los accesos a memoria hechas desde el Estado Seguro, si la región de memoria accedida pertenece o no a un espacio de memoria definido como Seguro. Solo existe en el 1er nivel de traslación y afecta a todas las páginas del 2do. nivel derivadas de éste descriptor.

nG non **G**lobal bit. No existe en Descriptores de Tabla de Página. Indica como es tratada la traslación en la TLB de la dirección virtual a la física descripta por esta entrada. Si nG = 0, significa que la página es compartida por todos los procesos.

Domain

Es un conjunto de hasta 16 regiones de memoria. Es un campo de 4 bits presente en Descriptores de Sección o de Tablas de Página. Cada dominio otorga una condición a toda el área de memoria que lo compone. Las Supersections pertenecen al Dominio 0, y las tablas de Página de 2do Nivel de traslación heredan su Dominio de la entrada de Nivel 1.

Domain

Es un conjunto de hasta 16 regiones de memoria. Es un campo de 4 bits presente en Descriptores de Sección o de Tablas de Página. Cada dominio otorga una condición a toda el área de memoria que lo compone. Las Supersections pertenecen al Dominio 0, y las tablas de Página de 2do Nivel de traslación heredan su Dominio de la entrada de Nivel 1.

No Access Cualquier acceso que utilice una tabla de traslación con este atributo generará Domain Fault.

Domain

Es un conjunto de hasta 16 regiones de memoria. Es un campo de 4 bits presente en Descriptores de Sección o de Tablas de Página. Cada dominio otorga una condición a toda el área de memoria que lo compone. Las Supersections pertenecen al Dominio 0, y las tablas de Página de 2do Nivel de traslación heredan su Dominio de la entrada de Nivel 1.

No Access Cualquier acceso que utilice una tabla de traslación con este atributo generará Domain Fault.

Client Cuando se accede a memoria utilizando un descriptor de tabla con este atributo de Dominio, se chequean los permisos, y eventualmente generará un Domain Fault

Domain

Es un conjunto de hasta 16 regiones de memoria. Es un campo de 4 bits presente en Descriptores de Sección o de Tablas de Página. Cada dominio otorga una condición a toda el área de memoria que lo compone. Las Supersections pertenecen al Dominio 0, y las tablas de Página de 2do Nivel de traslación heredan su Dominio de la entrada de Nivel 1.

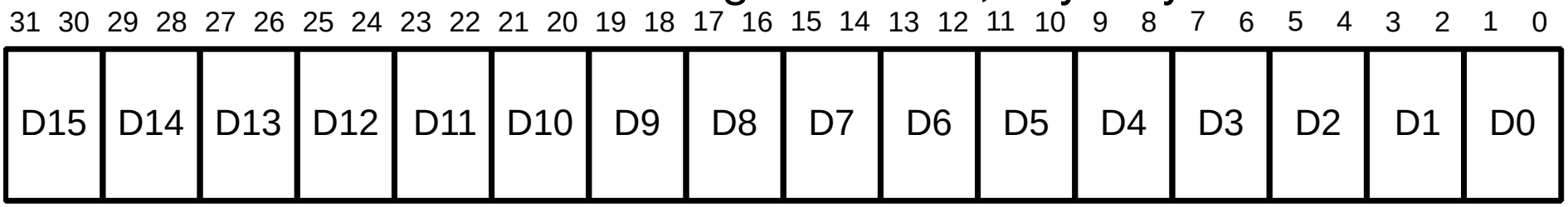
No Access Cualquier acceso que utilice una tabla de traslación con este atributo generará Domain Fault.

Client Cuando se accede a memoria utilizando un descriptor de tabla con este atributo de Dominio, se chequean los permisos, y eventualmente generará un Domain Fault

Manager Cuando se accede a memoria utilizando un descriptor de tabla con este atributo de Dominio, no se chequean los permisos, y no hay posibilidad de un Domian Fault.

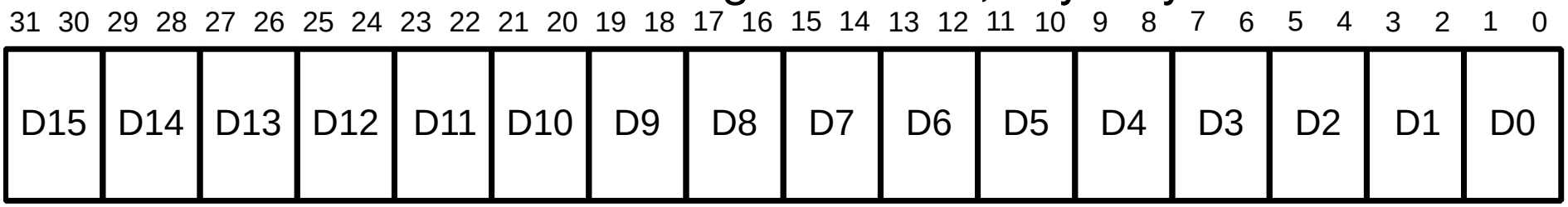
Domain

Los dominios se definen en el registro **DACR**, cuyo layout es:



Domain

Los dominios se definen en el registro **DACR**, cuyo layout es:



DR_n Define los permisos de acceso. El valor *n* es 0 a 15.

- **00** No Access. Su acceso generará Domain Fault.
- **01** Client*.
- **10** Reservado. Efecto impredecible.
- **11** Manager*.

Domain

Los dominios se definen en el registro **DACR**, cuyo layout es:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15		D14		D13		D12		D11		D10		D9		D8		D7		D6		D5		D4		D3		D2		D1		D0	

DR_n Define los permisos de acceso. El valor *n* es 0 a 15.

00 No Access. Su acceso generará Domain Fault.

01 Client*.

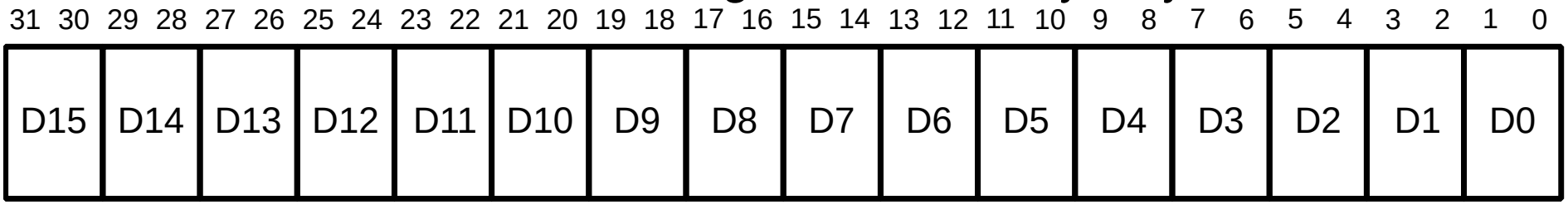
10 Reservado. Efecto impredecible.

11 Manager*.

- **Nota** *: En cada acceso estos permisos se chequean contra los permisos establecidos en las tablas de traslación.

Domain

Los dominios se definen en el registro **DACR**, cuyo layout es:



DR_n Define los permisos de acceso. El valor n es 0 a 15.

00 No Access. Su acceso generará Domain Fault.

01 Client*.

10 Reservado. Efecto impredecible.

11 Manager*.

- **Nota** *: En cada acceso estos permisos se chequean contra los permisos establecidos en las tablas de traslación.
- Es decir que un proceso puede ser Manager de ciertas áreas de memoria de su dominio, cliente de otras y no tener acceso a otras.

Domain

Los dominios se definen en el registro **DACR**, cuyo layout es:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15		D14		D13		D12		D11		D10		D9		D8		D7		D6		D5		D4		D3		D2		D1		D0	

DR_n Define los permisos de acceso. El valor *n* es 0 a 15.

00 No Access. Su acceso generará Domain Fault.

01 Client*.

10 Reservado. Efecto impredecible.

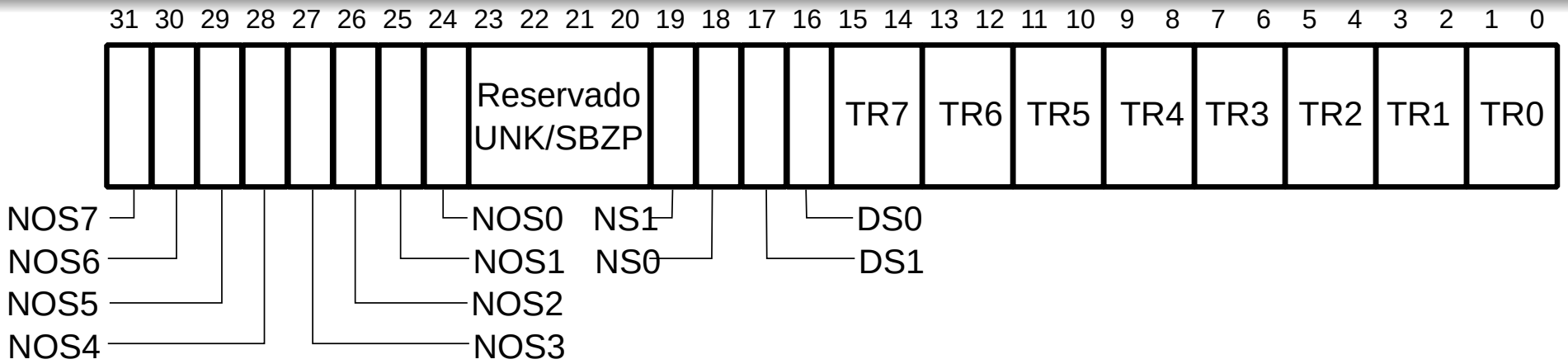
11 Manager*.

- **Nota** *: En cada acceso estos permisos se chequean contra los permisos establecidos en las tablas de traslación.
- Es decir que un proceso puede ser Manager de ciertas áreas de memoria de su dominio, cliente de otras y no tener acceso a otras.

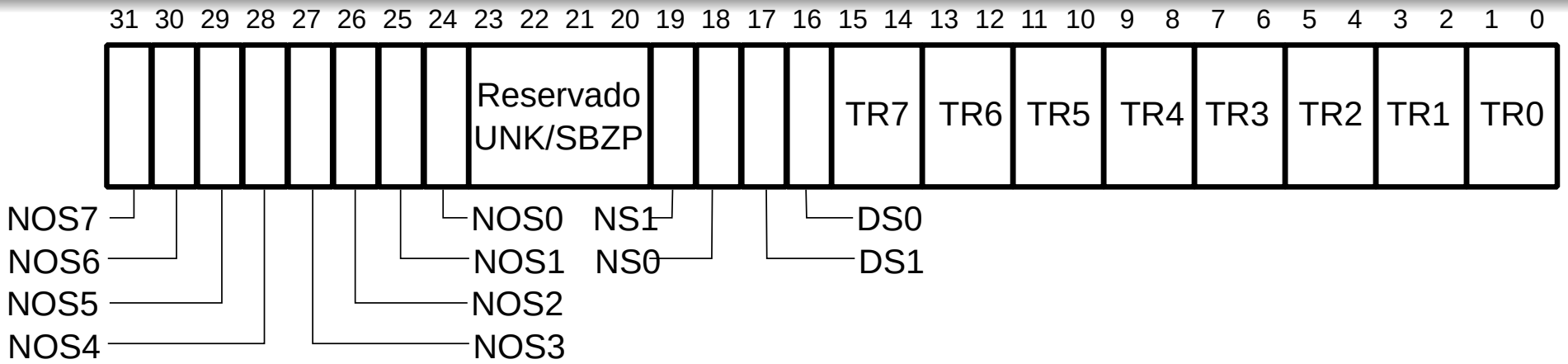
Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c3, c0, 0; Read DACR into Rt
MCR p15, 0, <Rt>, c3, c0, 0; Write RT to DACR
```

VMSA. Registros de cp15: PRRR



VMSA. Registros de cp15: PRRR



NOS_n Si la región está mapeada como Shareable, estos bits indican si es OSH (NOS_n=0), o ISH (NOS_n=1).
El valor *n* sale de la combinación {TEX[0],C,B} del descriptor.
Se ignoran si la región es Non-Shareable.
NS_6 es implementation defined (ya que la combinación {TEX[0] = 1, C = 1, B = 0}, es implementation defined).
Si la implementación no distingue entre ISH y OSH, estos bits están reservados, **RAZ/WI** (Read-As-Zero, Writes Ignored).

VMSA. Registros de cp15: PRRR

VMSA. Registros de cp15: PRRR

NS1 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 1. NS1 = 0, región Non-Shareable, NS1 = 1, región Shareable.

VMSA. Registros de cp15: PRRR

NS1 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 1. NS1 = 0, región Non-Shareable, NS1 = 1, región Shareable.

NS0 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 0. NS0 = 0, región Non-Shareable, NS0 = 1, región Shareable.

VMSA. Registros de cp15: PRRR

NS1 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 1. NS1 = 0, región Non-Shareable, NS1 = 1, región Shareable.

NS0 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 0. NS0 = 0, región Non-Shareable, NS0 = 1, región Shareable.

DS1 Atributo Shareable para una región definida como Device Memory, y que tiene en el descriptor el bit S = 1. DS1 = 0, región Non-Shareable, DS1 = 1, región Shareable.

VMSA. Registros de cp15: PRRR

NS1 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 1. NS1 = 0, región Non-Shareable, NS1 = 1, región Shareable.

NS0 Atributo Shareable para una región definida como Normal Memory, y que tiene en el descriptor el bit S = 0. NS0 = 0, región Non-Shareable, NS0 = 1, región Shareable.

DS1 Atributo Shareable para una región definida como Device Memory, y que tiene en el descriptor el bit S = 1. DS1 = 0, región Non-Shareable, DS1 = 1, región Shareable.

DS0 Atributo Shareable para una región definida como Device Memory, y que tiene en el descriptor el bit S = 1. DS0 = 0, región Non-Shareable, DS0 = 1, región Shareable.

VMSA. Registros de cp15: PRRR

VMSA. Registros de cp15: PRRR

TR_n Nuevamente el valor *n* sale de la combinación {TEX[0],C,B} del descriptor.
Cada campo de dos bits determina el tipo de región de Memoria. Los valores posibles son:

00	Strongly Ordered
01	Device
10	Normal Memory
11	Reservado (su efecto es impredecible)

NS_6 es implementation defined (ya que la combinación {TEX[0] = 1, C = 1, B = 0}, es implementation defined).

VMSA. Registros de cp15: PRRR

TR_n Nuevamente el valor *n* sale de la combinación {TEX[0],C,B} del descriptor.
Cada campo de dos bits determina el tipo de región de Memoria. Los valores posibles son:

00	Strongly Ordered
01	Device
10	Normal Memory
11	Reservado (su efecto es impredecible)

NS_6 es implementation defined (ya que la combinación {TEX[0] = 1, C = 1, B = 0}, es implementation defined).

Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c10, c2, 0; Read PRRR into Rt
MCR p15, 0, <Rt>, c10, c2, 0; Write RT to PRRR
```

VMSA. Registros de cp15: NMRR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR7		OR6		OR5		OR4		OR3		OR2		OR1		OR0		IR7		IR6		IR5		IR4		IR3		IR2		IR1		IR0	

VMSA. Registros de cp15: NMRR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR7		OR6		OR5		OR4		OR3		OR2		OR1		OR0		IR7		IR6		IR5		IR4		IR3		IR2		IR1		IR0	

OR_n Propiedad Outer Cacheable (L2). Tiene validez si la región está mapeada como Normal Memory en **PRRR.TR$_6$** El valor *n* sale de la combinación {TEX[0],C,B} del descriptor.
Cada campo de dos bits determina el tipo de región de Memoria. Los valores posibles son:

00 La Región es Non-Cacheable.
01 La Región es Write-Back, Write Allocate.
10 La región es Write-Through, no Write-Allocate.
11 La Región es Write-Back, no Write Allocate.

NS_6 es implementation defined (ya que la combinación {TEX[0] = 1, C = 1, B = 0}, es implementation defined).

VMSA. Registros de cp15: NMRR

VMSA. Registros de cp15: NMRR

IR_n Propiedad Inner Cacheable (L2). Tiene validez si la región está mapeada como Normal Memory en **PRRR.TR$_6$** El valor *n* sale de la combinación {TEX[0],C,B} del descriptor.
Cada campo de dos bits determina el tipo de región de Memoria. Los valores posibles son:

00 La Región es Non-Cacheable.
01 La Región es Write-Back, Write Allocate.
10 La región es Write-Through, no Write-Allocate.
11 La Región es Write-Back, no Write Allocate.

NS_6 es implementation defined (ya que la combinación {TEX[0] = 1, C = 1, B = 0}, es implementation defined).

VMSA. Registros de cp15: NMRR

IR_n Propiedad Inner Cacheable (L2). Tiene validez si la región está mapeada como Normal Memory en **PRRR.TR$_6$** El valor *n* sale de la combinación {TEX[0],C,B} del descriptor.
Cada campo de dos bits determina el tipo de región de Memoria. Los valores posibles son:

00 La Región es Non-Cacheable.
01 La Región es Write-Back, Write Allocate.
10 La región es Write-Through, no Write-Allocate.
11 La Región es Write-Back, no Write Allocate.

NS_6 es implementation defined (ya que la combinación {TEX[0] = 1, C = 1, B = 0}, es implementation defined).

Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c10, c2, 1; Read NMRR into Rt
MCR p15, 0, <Rt>, c10, c2, 1; Write RT to NMRR
```

Temario

Acerca del reporte de excepciones

Acerca del reporte de excepciones

- Las excepciones se toman en el mas alto nivel de privilegio. Siempre!.

Acerca del reporte de excepciones

- Las excepciones se toman en el mas alto nivel de privilegio. Siempre!.
- Si el core tiene las extensiones de virtualización, y éstas están habilitadas, se atienden en PL2.

Acerca del reporte de excepciones

- Las excepciones se toman en el mas alto nivel de privilegio. Siempre!.
- Si el core tiene las extensiones de virtualización, y éstas están habilitadas, se atienden en PL2.
- En nuestro análisis no abarcaremos virtualización de modo que a los efectos de este curso, nuestras excepciones por VMSA se atenderán en PL1.

Acerca del reporte de excepciones

- Las excepciones se toman en el mas alto nivel de privilegio. Siempre!.
- Si el core tiene las extensiones de virtualización, y éstas están habilitadas, se atienden en PL2.
- En nuestro análisis no abarcaremos virtualización de modo que a los efectos de este curso, nuestras excepciones por VMSA se atenderán en PL1.
- La información de la excepción está en los registros **`FSR`** (Fault Status Register).

Acerca del reporte de excepciones

- Las excepciones se toman en el mas alto nivel de privilegio. Siempre!.
- Si el core tiene las extensiones de virtualización, y éstas están habilitadas, se atienden en PL2.
- En nuestro análisis no abarcaremos virtualización de modo que a los efectos de este curso, nuestras excepciones por VMSA se atenderán en PL1.
- La información de la excepción está en los registros **`FSR`** (Fault Status Register).
- Adicionalmente para excepciones sincrónicas que retornan una o varias direcciones, estas se encuentran en los **`FAR`**'s (Fault Address Registers).

Acerca del reporte de excepciones

- Las excepciones se toman en el mas alto nivel de privilegio. Siempre!.
- Si el core tiene las extensiones de virtualización, y éstas están habilitadas, se atienden en PL2.
- En nuestro análisis no abarcaremos virtualización de modo que a los efectos de este curso, nuestras excepciones por VMSA se atenderán en PL1.
- La información de la excepción está en los registros **`FSR`** (Fault Status Register).
- Adicionalmente para excepciones sincrónicas que retornan una o varias direcciones, estas se encuentran en los **`FAR`**'s (Fault Address Registers).
- Las Faltas del Sistema de Memoria generan Memory Abort Exception, o Prefetch Abort Exception.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

DFAR Contiene direcciones de falta de utilidad para algunos Data Abort Exception sincrónico.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

DFAR Contiene direcciones de falta de utilidad para algunos Data Abort Exception sincrónico.

IFSR Contiene la información de los Prefetch Abort Exception.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

DFAR Contiene direcciones de falta de utilidad para algunos Data Abort Exception sincrónico.

IFSR Contiene la información de los Prefetch Abort Exception.

IFAR Contiene la dirección de falta de un Prefetch Abort Exception.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

DFAR Contiene direcciones de falta de utilidad para algunos Data Abort Exception sincrónico.

IFSR Contiene la información de los Prefetch Abort Exception.

IFAR Contiene la dirección de falta de un Prefetch Abort Exception.

DBGWFAR Puede mantener información útil para un Watchdog Debug Exception.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

DFAR Contiene direcciones de falta de utilidad para algunos Data Abort Exception sincrónico.

IFSR Contiene la información de los Prefetch Abort Exception.

IFAR Contiene la dirección de falta de un Prefetch Abort Exception.

DBGWFAR Puede mantener información útil para un Watchdog Debug Exception.

ADFSR Contiene información auxiliar para el **DFSR**.

Registros de Reporte de excepciones

La Arquitectura define los siguientes registros para excepciones que se toman en PL1

DFSR Contiene información de utilidad para atender un Data Abort Exception.

DFAR Contiene direcciones de falta de utilidad para algunos Data Abort Exception sincrónico.

IFSR Contiene la información de los Prefetch Abort Exception.

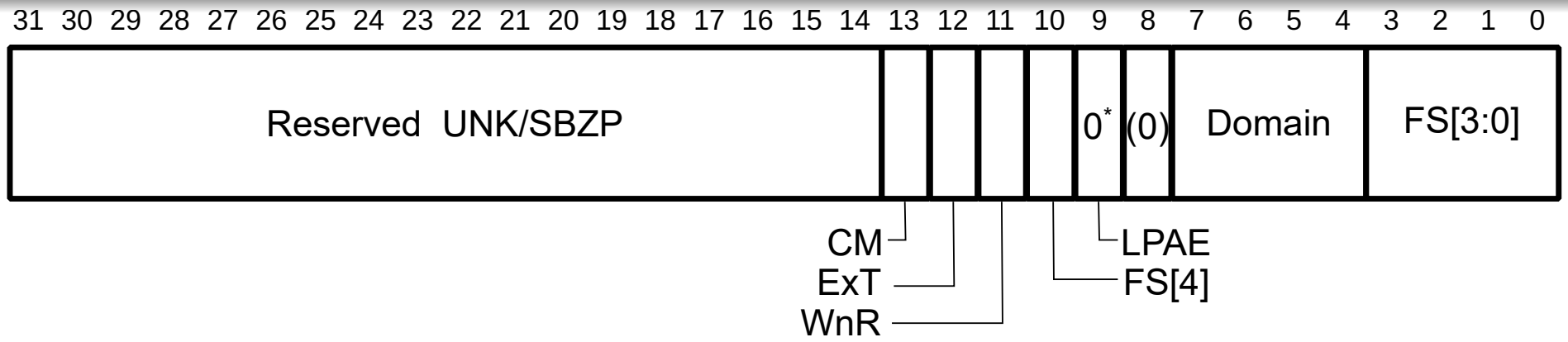
IFAR Contiene la dirección de falta de un Prefetch Abort Exception.

DBGWFAR Puede mantener información útil para un Watchdog Debug Exception.

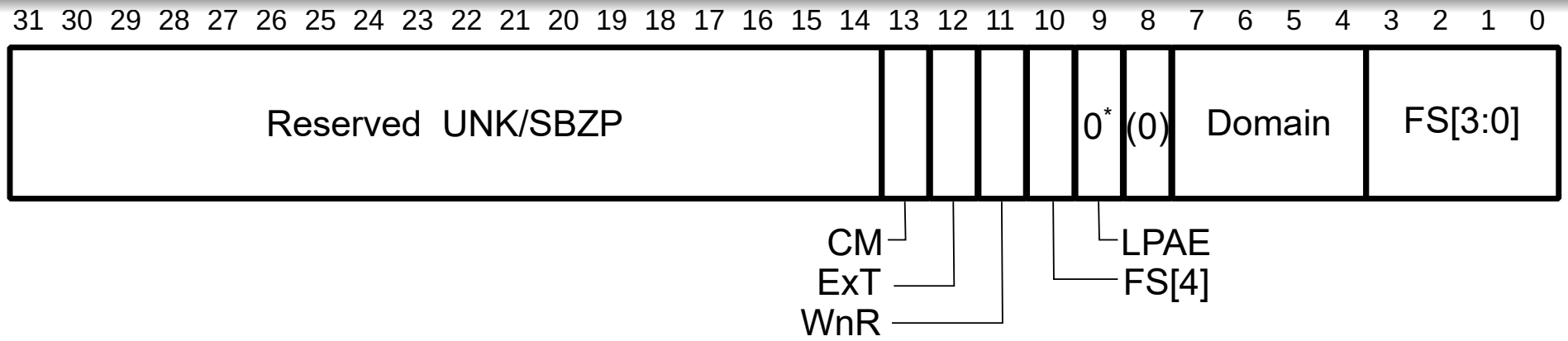
ADFSR Contiene información auxiliar para el **DFSR**.

AIFSR Contiene información auxiliar para el **IFSR**.

Excepciones **VMSA**. Registros de cp15: DFSR

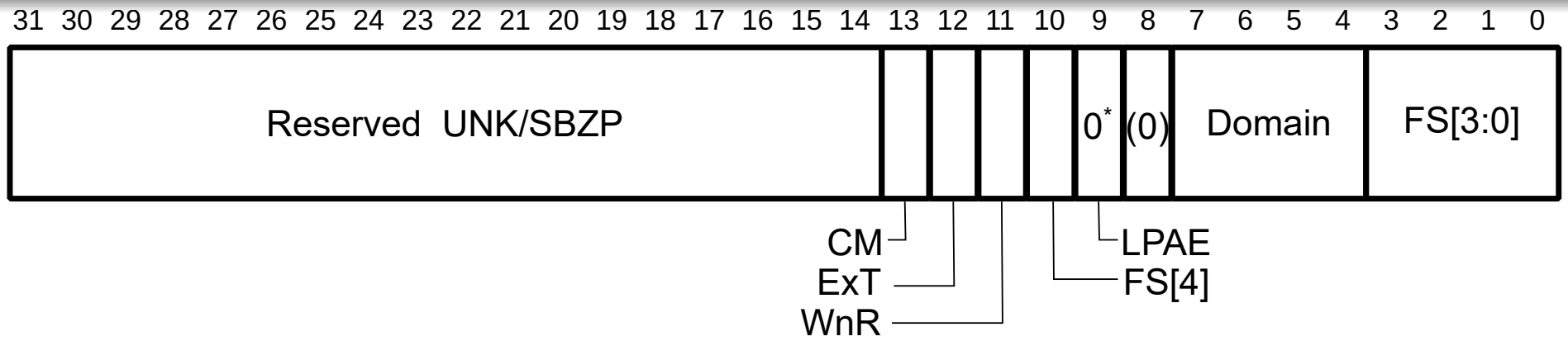


Excepciones **VMSA**. Registros de cp15: DFSR



CM+ Cache Mantaninance Fault. Si la excepción es del tipo Data Abort Sincrónica. Si CM = 0, la falta no fue debida a Cache Mantainance. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP).

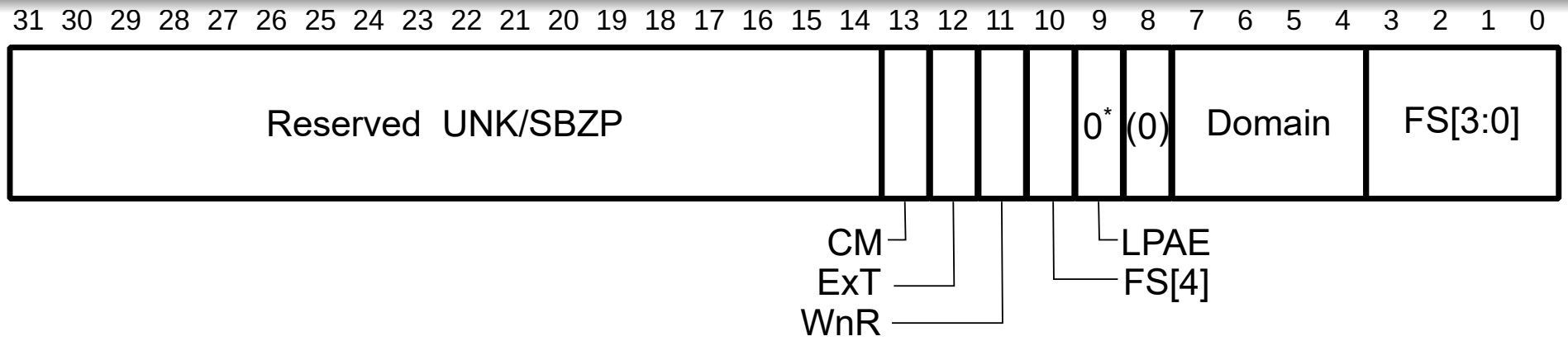
Excepciones **VMSA**. Registros de cp15: DFSR



CM+ Cache Mantaninance Fault. Si la excepción es del tipo Data Abort Sincrónica. Si CM = 0, la falta no fue debida a Cache Mantainance. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP).

ExT External Abort Type. Es implementation defined. Si el core no soporta External Abort, bit[12] es Reservado (UNK/SBZP).

Excepciones **VMSA**. Registros de cp15: DFSR



CM+ Cache Mantaninance Fault. Si la excepción es del tipo Data Abort Sincrónica. Si CM = 0, la falta no fue debida a Cache Mantainance. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP).

ExT External Abort Type. Es implementation defined. Si el core no soporta External Abort, bit[12] es Reservado (UNK/SBZP).

WnR Write no Read. Si es 1 el Data Abort Sincrónico se debe a una escritura. Si es 0 a una Lectura. Es UNKNOWN si la Data Abort Exception fue asincrónica, o debida a un Debug exception.

Excepciones **VMSA**. Registros de cp15: DFSR

Excepciones **VMSA**. Registros de cp15: DFSR

FS,bits[10,3:0] Bits de estado de Falta. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP). Detalle en el slide siguiente.

Excepciones **VMSA**. Registros de cp15: DFSR

FS,bits[10,3:0] Bits de estado de Falta. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP). Detalle en el slide siguiente.

LPAE Si se implementa Large Physical Address Extensions, si este bit es cero indica que se utiliza el formato Short-Descriptor. Se establece su estado por software sin afectarla operación. Si no se implementa LPAE bit[9] es Reservado (UNK/SBZP).

Excepciones **VMSA**. Registros de cp15: DFSR

FS,bits[10,3:0] Bits de estado de Falta. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP). Detalle en el slide siguiente.

LPAE Si se implementa Large Physical Address Extensions, si este bit es cero indica que se utiliza el formato Short-Descriptor. Se establece su estado por software sin afectarla operación. Si no se implementa LPAE bit[9] es Reservado (UNK/SBZP).

Domain Dominio de la dirección de la Falta. ARM lo dejó obsoleto.

Excepciones **VMSA**. Registros de cp15: DFSR

FS,bits[10,3:0] Bits de estado de Falta. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP). Detalle en el slide siguiente.

LPAE Si se implementa Large Physical Address Extensions, si este bit es cero indica que se utiliza el formato Short-Descriptor. Se establece su estado por software sin afectarla operación. Si no se implementa LPAE bit[9] es Reservado (UNK/SBZP).

Domain Dominio de la dirección de la Falta. ARM lo dejó obsoleto.

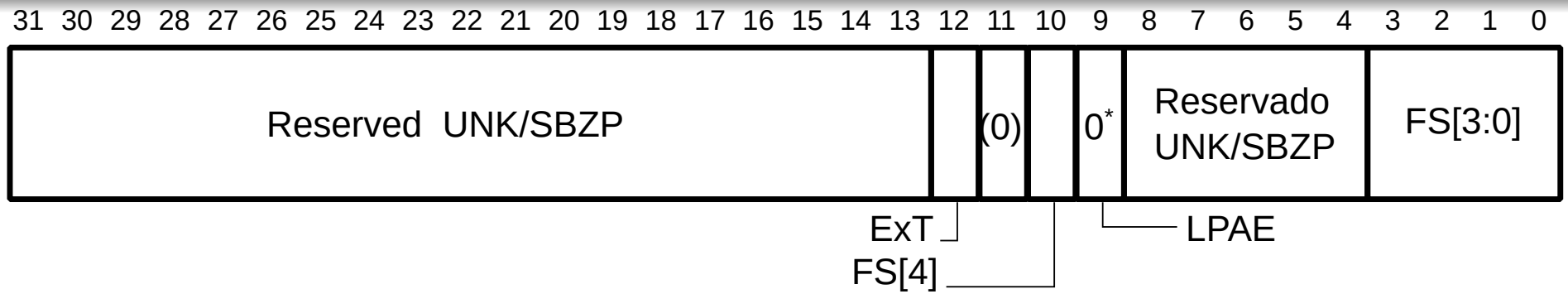
Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c5, c0, 0; Read DSFR into Rt
MCR p15, 0, <Rt>, c5, c0, 0; Write RT to DSFR
```

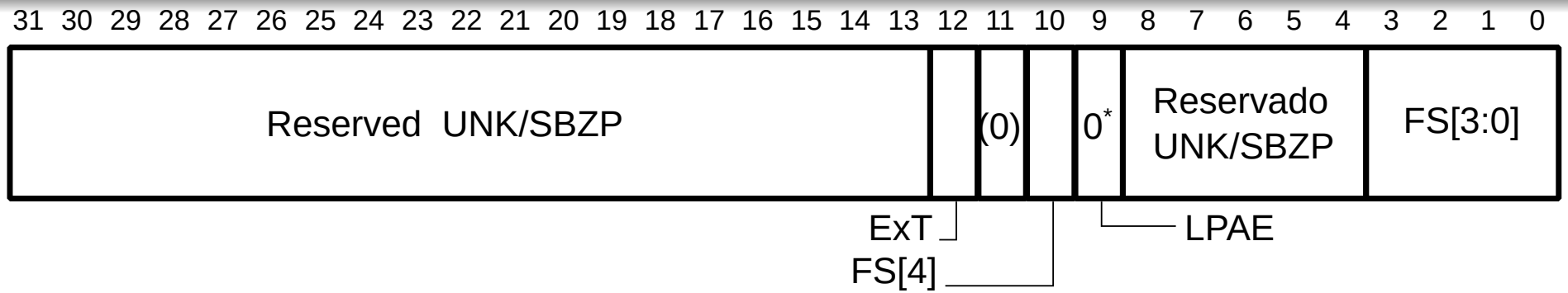
Códigos FSR para formato Short-Descriptor.

FS	Source		Notas
00000	Aligment Fault		Solo DFSR.
00001	Debug Event		Acceso solo en PL1.
00011	Access flag Fault	1er. Nivel	MMU Fault.
00010		2do. Nivel	MMU Fault.
00100	Fault on instruction cache maintenance		Solo DFSR.
00101	Translation fault	1er. Nivel	MMU Fault.
00111		2do. Nivel	MMU Fault.
01100	Synchronous external abort on translation table walk	1er. Nivel	-
01110		2do. Nivel	-
01101	Permission fault	1er. Nivel	MMU Fault
01111		2do. Nivel	MMU Fault
10000	TLB conflict abort		
10100	Implementation Defined		Lockdown
10110	Asynchronous external abort		Solo DFSR.
11000	Asynchronous parity error on memory access		Solo DFSR.
11001	Synchronous parity error on memory access	-	
11010	Implementation Defined		Coprocessor abort
11100	Synchronous parity error on translation table walk	1er. Nivel	-
11110		2do. Nivel	-

Excepciones **VMSA**. Registros de cp15: IFSR

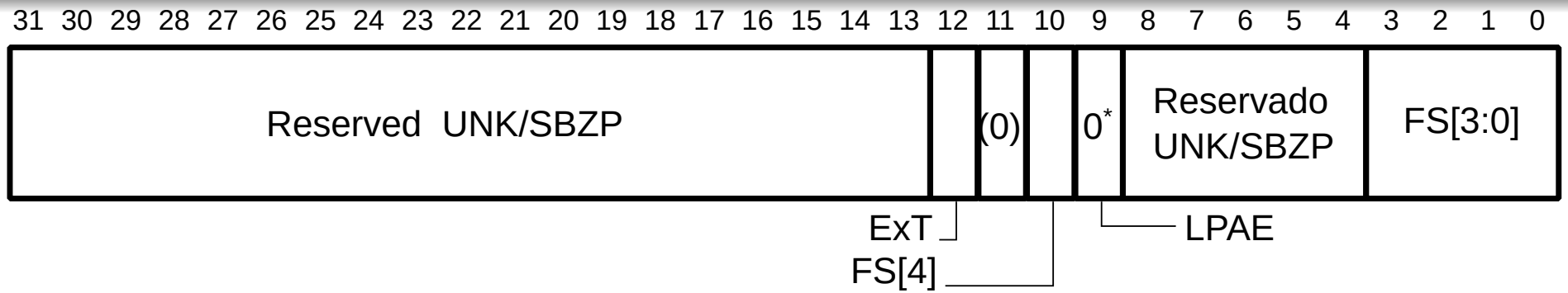


Excepciones **VMSA**. Registros de cp15: IFSR



ExT External Abort Type. Es implementation defined. Si el core no soporta External Abort, bit[12] es Reservado (UNK/SBZP).

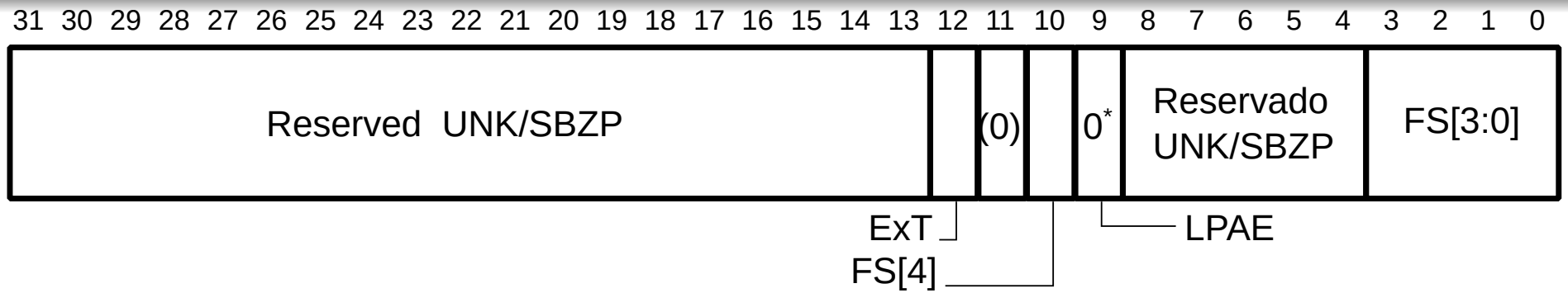
Excepciones **VMSA**. Registros de cp15: IFSR



ExT External Abort Type. Es implementation defined. Si el core no soporta External Abort, bit[12] es Reservado (UNK/SBZP).

FS,bits[10,3:0] Bits de estado de Falta. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP). Detalle en el slide anterior.

Excepciones **VMSA**. Registros de cp15: IFSR



ExT External Abort Type. Es implementation defined. Si el core no soporta External Abort, bit[12] es Reservado (UNK/SBZP).

FS,bits[10,3:0] Bits de estado de Falta. Si es 1, fue por un cache Mantainance. Si el core no soporta LPAE, bit[13] pasa a ser Reservado (UNK/SBZP). Detalle en el slide anterior.

LPAE Si se implementa Large Physical Address Extensions, si este bit es cero indica que se utiliza el formato Short-Descriptor. Se establece su estado por software sin afectarla operación. Si no se implementa LPAE bit[9] es Reservado (UNK/SBZP).

Excepciones **VMSA**. Registros de cp15: IFSR

Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c5, c0, 1; Read IFSR into Rt
MCR p15, 0, <Rt>, c5, c0, 1; Write RT to IFSR
```

Excepciones **VMSA**. Registros de cp15: DFAR IFAR

DFAR

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Dirección Virtual que causó la Excepción Data Abort Sincrónica

Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c6, c0, 0; Read DFAR into Rt
MCR p15, 0, <Rt>, c6, c0, 0; Write RT to DFAR
```

IFAR

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

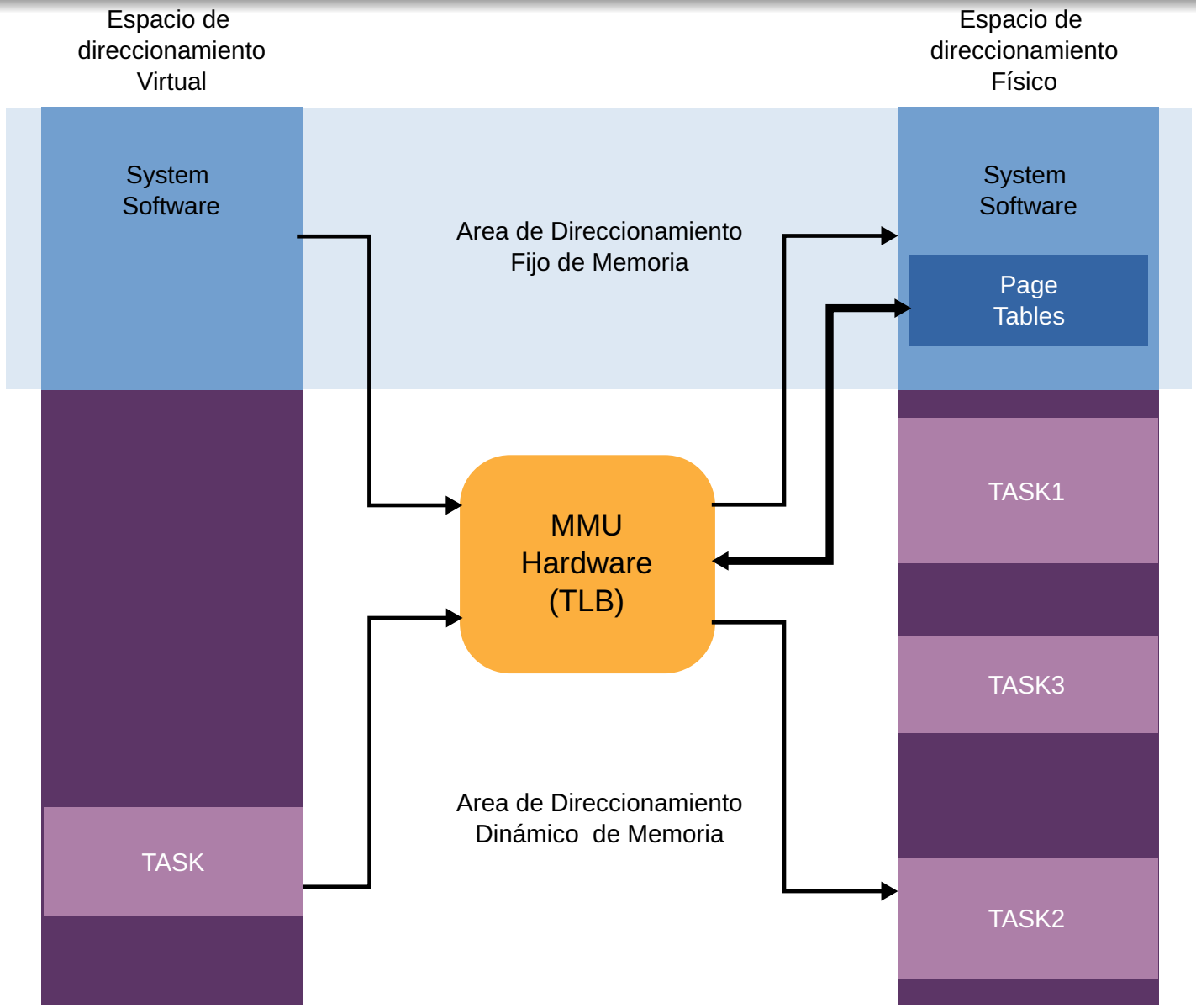
Dirección Virtual que causó la Excepción de Prefetch Abort Sincrónica

Instrucciones de Acceso

```
MRC p15, 0, <Rt>, c6, c0, 2; Read IFAR into Rt
MCR p15, 0, <Rt>, c6, c0, 2; Write RT to IFAR
```

Temario

Definiendo el mapa de memoria



Definiendo el mapa de memoria

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.
   e. Ubicar las tablas de página para las tareas iniciales

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.
   e. Ubicar las tablas de página para las tareas iniciales
   f. Considerar un espacio disponible para albergar tabas de páginas de otras tareas que se sumen en forma dinámICA.

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.
   e. Ubicar las tablas de página para las tareas iniciales
   f. Considerar un espacio disponible para albergar tabas de páginas de otras tareas que se sumen en forma dinámICA.
2. Definir el espacio virtual y físico en el que correrá cada tarea

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.
   e. Ubicar las tablas de página para las tareas iniciales
   f. Considerar un espacio disponible para albergar tabas de páginas de otras tareas que se sumen en forma dinámICA.
2. Definir el espacio virtual y físico en el que correrá cada tarea
3. Inicializar las tablas de página para cada tarea, teniendo en cuenta las páginas del Kernel y las de cada tarea

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.
   e. Ubicar las tablas de página para las tareas iniciales
   f. Considerar un espacio disponible para albergar tabas de páginas de otras tareas que se sumen en forma dinámICA.
2. Definir el espacio virtual y físico en el que correrá cada tarea
3. Inicializar las tablas de página para cada tarea, teniendo en cuenta las páginas del Kernel y las de cada tarea
4. Definir las estructuras que se requieran para controlar para cada tarea sus tablas de página y demás atributos de interés de la tarea

Definiendo el mapa de memoria

1. Definir el espacio físico en el que se alojará el Kernel
   a. Donde ejecuta el programa de inicialización
   b. Donde ejecutan los controladores de E/S
   c. Ubicar la tabla de vectores de interrupción
   d. Establecer el área de memoria en donde se ubicarán los handlers de interrupción.
   e. Ubicar las tablas de página para las tareas iniciales
   f. Considerar un espacio disponible para albergar tabas de páginas de otras tareas que se sumen en forma dinámICA.
2. Definir el espacio virtual y físico en el que correrá cada tarea
3. Inicializar las tablas de página para cada tarea, teniendo en cuenta las páginas del Kernel y las de cada tarea
4. Definir las estructuras que se requieran para controlar para cada tarea sus tablas de página y demás atributos de interés de la tarea
5. Inicializar MMU, caches, y write Buffers