

# 第 4 章

## ARM 处理器内存管理单元 (MMU)

### 4.1 ARM 处理器中 CP15 协处理器的寄存器

本章和第 5 章的内容都与 ARM 处理器中 CP15 协处理器的寄存器有密切关系，所以我们这里先介绍一下 CP15 寄存器以及访问 CP15 寄存器的汇编指令。

#### 4.1.1 访问 CP15 寄存器的指令

访问 CP15 寄存器指令的编码格式及语法说明如下：

31 28	27 24	23 21	20	19 16	15 12	11 8	7 5	4	3 0
cond	1 1 1 0	opcode_1	L	cr n	rd	1 1 1 1	opcode_2	1	crm

**说明：**

<opcode\_1>：协处理器行为操作码，对于 CP15 来说，<opcode\_1>永远为 0b000，否则结果未知。

<rd>：不能是 r15/pc，否则，结果未知。

<crn>：作为目标寄存器的协处理器寄存器，编号为 C0~C15。

**<crm>**: 附加的目标寄存器或源操作数寄存器, 如果不需要设置附加信息, 将 crm 设置为 c0, 否则结果未知。

**<opcode\_2>**: 提供附加信息比如寄存器的版本号或者访问类型, 用于区分同一个编号的不同物理寄存器, 可以省略**<opcode\_2>**或者将其设置为 0, 否则结果未知。

指令	说 明	语法格式
mcr	将 ARM 处理器的寄存器中的数据写到 CP15 中的寄存器中	<code>mcr{&lt;cond&gt;} p15, &lt;opcode_1&gt;, &lt;rd&gt;, &lt;crn&gt;, &lt;crm&gt;, {&lt;opcode_2&gt;}</code>
mrc	将 CP15 中的寄存器中的数据读到 ARM 处理器的寄存器中	<code>mrc{&lt;cond&gt;} p15, &lt;opcode_1&gt;, &lt;rd&gt;, &lt;crn&gt;, &lt;crm&gt;, {&lt;opcode_2&gt;}</code>

### 4.1.2 CP15 寄存器介绍

CP15 的寄存器列表如表 4-1 所示。

表 4-1 ARM 处理器中 CP15 协处理器的寄存器

寄存器编号	基本作用	在 MMU 中的作用	在 PU 中的作用
0	ID 编码 (只读)	ID 编码和 cache 类型	
1	控制位 (可读写)	各种控制位	
2	存储保护和控制	地址转换表基址	Cachability 的控制位
3	存储保护和控制	域访问控制位	Bufferability 控制位
4	存储保护和控制	保留	保留
5	存储保护和控制	内存失效状态	访问权限控制位
6	存储保护和控制	内存失效状态	保护区域控制
7	高速缓存和写缓存	高速缓存和写缓存控制	
8	存储保护和控制	TLB 控制	保留
9	高速缓存和写缓存	高速缓存锁定	
10	存储保护和控制	TLB 锁定	保留
11	保留		
12	保留		
13	进程标识符	进程标识符	
14	保留		
15	因不同设计而异	因不同设计而异	因不同设计而异

#### • CP15 的寄存器 C0

CP15 中寄存器 C0 对应两个标识符寄存器, 由访问 CP15 中的寄存器指令中的**<opcode\_2>**指定要访问哪个具体物理寄存器, **<opcode\_2>**与两个标识符寄存器的对应关系如下所示:

opcode_2 编码	对应的标识符号寄存器
0b000	主标识符寄存器
0b001	cache 类型标识符寄存器
其他	保留

### 1) 主标识符寄存器

访问主标识符寄存器的指令格式如下所示：

`mrc p15, 0, r0, c0, c0, 0` : 将主标识符寄存器 C0,0 的值读到 r0 中

ARM 不同版本体系处理器中主标识符寄存器的编码格式说明如下。

ARM7 之后处理器的主标识符寄存器编码格式如下所示：

31	24	23	20	19	16	15	4	3	0
由生产商确定	产品子编号		ARM 体系版本号		产品主编号		处理器版本号		

位	说 明
位 [3: 0]	生产商定义的处理器版本号
位 [15: 4]	生产商定义的产品主编号，其中最高 4 位即位 [15:12] 可能的取值为 0~7 但不能是 0 或 7
位 [19: 16]	ARM 体系的版本号，可能的取值如下： 0x1 ARM 体系版本 4 0x2 ARM 体系版本 4T 0x3 ARM 体系版本 5 0x4 ARM 体系版本 5T 0x5 ARM 体系版本 5TE 其他 由 ARM 公司保留将来使用
位 [23: 20]	生产商定义的产品子编号，当产品主编号相同时，使用子编号来区分不同的产品子类，如产品中不同的高速缓存的大小等
位 [31: 24]	生产厂商的编号，现在已经定义的有以下值： 0x41 =A ARM 公司 0x44 =D Digital Equipment 公司 0x69 =I intel 公司

ARM7 处理器的主标识符寄存器编码格式如下所示：

31	24	23	22	16	15	4	3	0
由生产商确定	A	产品子编号		产品主编号		处理器版本号		

位	说 明
位 [3: 0]	生产商定义的处理器版本号
位 [15: 4]	生产商定义的产品主编号，其中最高 4 位即位 [15:12] 的值为 0x7
位 [22: 16]	生产商定义的产品子编号，当产品主编号相同时，使用子编号来区分不同的产品子类，如产品中不同的高速缓存的大小等

续表

位	说 明
位[23]	ARM7 支持下面两种 ARM 体系的版本号: 0x0 ARM 体系版本 3 0x1 ARM 体系版本 4T
位[31: 24]	生产厂商的编号，现在已经定义的有以下值： 0x41 =A ARM 公司 0x44 =D Digital Equipment 公司 0x69 =I Intel 公司

ARM7 之前处理器的主标识符寄存器编码格式如下所示：

31	24	23	22	16	15	4	3	0
由生产商确定	A	产品子编号		产品主编号		处理器版本号		

位	说 明
位[3: 0]	生产商定义的处理器版本号
位[15: 4]	生产商定义的产品主编号，其中最高 4 位即为 [15:12] 的值为 0x7
位[22: 16]	生产商定义的产品子编号，当产品主编号相同时，使用子编号来区分不同的产品子类，如产品中不同的高速缓存的大小等
位[23]	ARM7 支持下面两种 ARM 体系的版本号： 0x0 ARM 体系版本 3 0x1 ARM 体系版本 4T
位[31: 24]	生产厂商的编号，现在已经定义的有以下值： 0x41 =A ARM 公司 0x44 =D Digital Equipment 公司 0x69 =I intel 公司

## 2) cache 类型标识符寄存器

访问 cache 类型标识符寄存器的指令格式如下所示：

mrc p15, 0, r0, c0, c0, 1 ; 将 cache 类型标识符寄存器 C0,1 的值读到 r0 中

ARM 处理器中 cache 类型标识符寄存器的编码格式如下所示：

31	29	28	25	24	23	12	11	0
0 0 0	属性字段	S	数据 cache 相关属性	指令 cache 相关属性				

位	说明
位[28: 25]	指定控制字段位[24: 0]指定的属性之外的 cache 的其他属性，详见表 4-2
位[24]	定义系统中的数据 cache 和指令 cache 是分开的还是统一的： 0 系统的数据 cache 和指令 cache 是统一的； 1 系统的数据 cache 和指令 cache 是分开的
位[23: 12]	定义数据 cache 的相关属性，如果位[24]为 0，本字段定义整个 cache 的属性
位[31: 24]	定义指令 cache 的相关属性，如果位[24]为 0，本字段定义整个 cache 的属性

其中控制字段位[28: 25]的含义说明如下：

表 4-2 cache 类型标识符寄存器的控制字段位[28: 25]

编 码	cache 类型	cache 内容清除方法	cache 内容锁定方法
0b0000	写通类型	不需要内容清除	不支持内容锁定
0b0001	写回类型	数据块读取	不支持内容锁定
0b0010	写回类型	由寄存器 C7 定义	不支持内容锁定
0b0110	写回类型	由寄存器 C7 定义	支持格式 A
0b0111	写回类型	由寄存器 C7 定义	支持格式 B

控制字段位[23: 12]和控制字段位[11: 0]的编码格式相同，含义如下所示：

11	9	8	6	5	3	2	1	0
0	0	0	cache 容量	cache 相联特性	M	块大小		

cache 容量字段 bits[8: 6]的含义如下所示：

编 码	M=0 时含义 (单位 KB)	M=1 时含义 (单位 KB)
0b000	0.5	0.75
0b001	1	1.5
0b010	2	3
0b011	4	6
0b100	8	12
0b101	16	24
0b110	32	48
0b111	64	96

cache 相联特性字段 bits[5: 3]的含义如下所示：

编 码	M=0 时含义	M=1 时含义
0b000	1 路相联 (直接映射)	没有 cache
0b001	2 路相联	3 路相联
0b010	4 路相联	6 路相联
0b011	8 路相联	12 路相联
0b100	16 路相联	24 路相联
0b101	32 路相联	48 路相联
0b110	64 路相联	96 路相联
0b111	128 路相联	192 路相联

cache 块大小字段 bits[1: 0]的含义如下所示：

编 码	cache 块大小
0b00	2 个字 (8 字节)
0b01	4 个字 (16 字节)
0b10	8 个字 (32 字节)
0b11	16 个字 (64 字节)

- CP15 的寄存器 C1

访问主标识符寄存器的指令格式如下所示：

```
mrc p15, 0, r0, c1, c0{, 0} ; 将 CP15 的寄存器 C1 的值读到 r0 中  
mcr p15, 0, r0, c1, c0{, 0} ; 将 r0 的值写到 CP15 的寄存器 C1 中
```

CP15 中的寄存器 C1 的编码格式及含义说明如下：

位	说 明
M	0: 禁止 MMU 或者 PU; 1: 使能 MMU 或者 PU
A	0: 禁止地址对齐检查; 1: 使能地址对齐检查
C	0: 禁止数据/整个 cache; 1: 使能数据/整个 cache
W	0: 禁止写缓冲; 1: 使能写缓冲
P	0: 异常中断处理程序进入 32 位地址模式; 1: 异常中断处理程序进入 26 位地址模式
D	0: 禁止 26 位地址异常检查; 1: 使能 26 位地址异常检查
L	0: 选择早期中止模型; 1: 选择后期中止模型
B	0: little endian; 1: big endian
S	在基于 MMU 的存储系统中, 本位用作系统保护
R	在基于 MMU 的存储系统中, 本位用作 ROM 保护
F	0: 由生产商定义
Z	0: 禁止跳转预测功能; 1: 使能跳转预测指令
I	0: 禁止指令 cache; 1: 使能指令 cache
V	0: 选择低端异常中断向量 0x0~0x1c; 1: 选择高端异常中断向量 0xffff0000~0xffff001c
RR	0: 常规的 cache 淘汰算法, 如随机淘汰; 1: 预测性淘汰算法, 如 round-robin 淘汰算法
L4	0: 保持 ARMv5 以上版本的正常功能; 1: 将 ARMv5 以上版本与以前版本处理器兼容, 不根据跳转地址的 bit[0] 进行 ARM 指令和 Thumb 状态切换: bit[0] 等于 0 表示 ARM 指令, 等于 1 表示 Thumb 指令
附加:	

- CP15 的寄存器 C2

CP15 中的寄存器 C2 保存的是页表的基地址, 即一级映射描述符表的基地址。其编码格如下所示:

31	0
一级映射描述符表的基地址 (物理地址)	

- CP15 的寄存器 C3

CP15 中的寄存器 C3 定义了 ARM 处理器的 16 个域的访问权限。

31	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0
----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----	---

- CP15 的寄存器 C5

CP15 中的寄存器 C5 是失效状态寄存器，编码格式如下所示：

31	9	8	7	4	3	0
UNP/SBZP	0					域标识 状态标识

其中，域标识 bit[7: 4]表示存放引起存储访问失效的存储访问所属的域。

状态标识 bit[3: 0]表示引起存储访问失效的存储访问类型，该字段含义如表 4-3 所示（优先级由上到下递减）。

表 4-3 状态标识字段含义

引起访问失效的原因	状态标识	域标识	C6
终端异常 (Terminal Exception)	0b0010	无效	生产商定义
中断向量访问异常 (Vector Exception)	0b0000	无效	有效
地址对齐	0b00x1	无效	有效
一级页表访问失效	0b1100	无效	有效
二级页表访问失效	0b1110	有效	有效
基于段的地址变换失效	0b0101	无效	有效
基于页的地址变换失效	0b0111	有效	有效
基于段的存储访问中域控制失效	0b1001	有效	有效
基于页的存储访问中域控制失效	0b1101	有效	有效
基于段的存储访问中访问权限控制失效	0b1111	有效	有效
基于页的存储访问中访问权限控制失效	0b0100	有效	有效
基于段的 cache 预取时外部存储系统失效	0b0110	有效	有效
基于页的 cache 预取时外部存储系统失效	0b1000	有效	有效
基于段的非 cache 预取时外部存储系统失效	0b1010	有效	有效

- CP15 中的寄存器 C6

CP15 中的寄存器 C6 是失效地址寄存器，编码格式如下所示：

31	0
失效地址（虚拟地址）	

- CP15 中的寄存器 C7

CP15 的 C7 寄存器用来控制 cache 和写缓存，它是一个只写寄存器，读操作将产生不可预知的后果。

访问 CP15 的 C7 寄存器的指令格式如下所示：

`mcr p15, 0, <rd>, <c7>, crm, <opcode_2>;` <rd>、<crm>和<opcode\_2>的不同取值组合实现不同功能

- CP15 中的寄存器 C8

CP15 的 C8 寄存器用来控制清除 TLB 的内容，是只写寄存器，读操作将产生不可预知的后果。

访问 CP15 的 C8 寄存器的指令格式如下所示：

`mcr p15, 0, <rd>, <c8>, crm, <opcode_2>;` <rd>、<crm>和<opcode\_2>的不同取值组合实现不同功能，见第 4.2 节

- CP15 中的寄存器 C9

CP15 的 C9 寄存器用于控制 cache 内容锁定。

访问 CP15 的 C9 寄存器的指令格式如下所示：

`mcr p15, 0, <rd>, <c9>, c0, <opcode_2>`  
`mrc p15, 0, <rd>, <c9>, c0, <opcode_2>`

如果系统中包含独立的指令 cache 和数据 cache，那么对于数据 cache 和指令 cache 分别有一个独立的 cache 内容锁定寄存器，<opcode\_2>用来选择其中的某个寄存器：

<opcode\_2>=1 选择指令 cache 的内容锁定寄存器；

<opcode\_2>=0 选择数据 cache 的内容锁定寄存器。

CP15 的 C9 寄存器有 A、B 两种编码格式。编码格式 A 如下所示：

31	32-W	31-W	0
cache 组内块序号 index		0	

其中 index 表示当下一次发生 cache 未命中时，将预取的存储块存入 cache 中该块对应的组中序号为 index 的 cache 块中。此时序号为 0~index-1 的 cache 块被锁定，当发生 cache 替换时，从序号为 index 到 ASSOCIATIVITY 的块中选择被替换的块。

编码格式 B 如下所示：

31	30	W	W-1	0
L	0		cache 组内块序号 index	

位	说 明
L=0	当发生 cache 未命中时，将预取的存储块存入 cache 中该块对应的组中序号为 index 的 cache 块中

续表

位	说 明
L=1	如果本次写操作之前 L=0，并且 index 值小于本次写入的 index，本次写操作执行的结果不可预知；否则，这时被锁定的 cache 块包括序号为 0~index-1 的块，当发生 cache 替换时，从序号为 index 到 ASSOCIATIVITY 的块中选择被替换的块

### • CP15 的寄存器 C10

CP15 的 C10 寄存器用于控制 TLB 内容锁定。

访问 CP15 的 C10 寄存器的指令格式如下所示：

```
mcr p15, 0, <rd>, <c10>, c0, <opcode_2>
mrc p15, 0, <rd>, <c10>, c0, <opcode_2>
```

如果系统中包含独立的指令 TLB 和数据 TLB，那么对应于数据 TLB 和指令 TLB 分别有一个独立的 TLB 内容锁定寄存器，<opcode\_2>用来选择其中的某个寄存器：

<opcode\_2>=1 选择指令 TLB 的内容锁定寄存器；

<opcode\_2>=0 选择数据 TLB 的内容锁定寄存器。

C10 寄存器的编码格式如下：

31-30	32-W	31-W	32-2W	31-2W	1	0
可被替换的条目起始地址的 base		下一个将被替换的条目地址 victim		0		P

位	说 明
victim	指定下一次 TLB 没有命中（所需的地址变换条目没有包含在 TLB 中）时，从内存页表中读取所需的地址变换条目，并把该地址变换条目保存在 TLB 中地址 victim 处
base	指定 TLB 替换时，所使用的地址范围，从 (base) 到 (TLB 中条目数-1)；字段 victim 的值应该包含在该范围内
P	1：写入 TLB 的地址变换条目不会受使整个 TLB 无效操作的影响，一直保持有效；0：写入 TLB 的地址变换条目将会受到使整个 TLB 无效操作的影响

### • CP15 的寄存器 C13

C13 寄存器用于快速上下文切换 FCSE。

访问 CP15 的 C13 寄存器的指令格式如下所示：

```
mcr p15, 0, <rd>, <c13>, c0, 0 .
mrc p15, 0, <rd>, <c13>, c0, 0
```

C13 寄存器的编码格式如下所示：

31	25	24	0
PID		0	

其中，PID 表示当前进程的所在的进程空间块的编号，即当前进程的进程标识符，取值为

0~127。

0: MVA (变换后的虚拟地址) = VA (虚拟地址), 禁止 FCSE (快速上下文切换技术),  
系统复位后 PID=0;

非 0: 使能 FCSE。

## 4.2 MMU 简介

MMU 是 Memory Manage Unit 的缩写, 即存储管理单元的意思。MMU 实现以下功能:

- 1) 虚拟存储地址到物理存储地址的映射;
- 2) 控制存储空间访问权限;
- 3) 设置存储空间的缓冲特性。

与 MMU 相关的一些基本概念介绍如下:

### ■ 页表 (Translate Table)

页表又叫翻译表, 用来将虚拟地址翻译成对应的物理地址, 它位于内存中, 是实现 MMU 功能的重要组成部分, 处理器通过查找页表中的描述符来获取虚拟地址对应的物理地址。ARM 处理器是按两级分页来管理内存的, 所以页表包括一级页表和二级页表。一级页表中的每一项 (1 个字, 4 字节) 对应于虚拟存储空间的一段 (section), 一段的大小为 1MB (0x100000 字节), 该项包含了该虚拟存储段对应的物理存储段的基址或者一段存储空间内每页的二级页表描述符组成的表的基址、所属的内存域编号、缓冲特性等, 一级页表的基址保存在 ARM 处理器中 CP15 协处理器的 C2 寄存器中, 该寄存器中保存的是一级页表的物理地址, 而不是虚拟地址。二级页表中的每一项 (1word, 4bytes) 对应于虚拟存储空间的一页, 一页的大小可以是 4KB 或 1KB, 该项包含了该虚拟存储页对应的物理存储页的基址、该页的访问权限和该页的缓冲特性等。我们将页表中的每一项叫做一个地址变换条目 (entry), 也可以叫一个页表项。

### ■ 翻译援助缓冲区 (Translation Lookaside Buffer, TLB)

TLB 在硬件上和 cache 一样, 是处理器内部的一小块高速 SRAM 内存, 用于缓存, 与 cache 不同的是, 它专门缓存存放在内存中的页表, 容量相对比较小, 而 cache 则用于缓存普通内存, 容量相对比较大。TLB 也分为数据 TLB 和指令 TLB, 指令 TLB 用于取指令时的指令地址翻译, 而数据 TLB 用于其他存储访问操作时的地址翻译。有的处理器中数据 TLB 和指令 TLB 是分开的, 有的处理器中这两者是统一的。TLB 是为了提高处理器查询页表的速度而设计的, 所以

TLB 又叫快表。当处理器要查询页表时首先在 TLB 中查找，如果要查找的页表项不在 TLB 中，那么 CPU 从位于内存中的页表中查询，并把相应的结果添加到 TLB 中，这样 CPU 下次查找该页表项时就可以从 TLB 中直接获取。

## 4.3 系统访问存储空间的过程

系统访问存储空间的过程对于使能 MMU 和禁止 MMU 两种情况是不一样的。ARM 处理器中控制 MMU 功能的开关位是 CP15 协处理器的 C1 寄存器的 bit[0]，当该位为 0 时禁止 MMU 功能，当该位为 1 时使能 MMU 功能。

代 码	作 用
mrc p15, 0, r0, c1, 0, 0	使能 MMU
orr r0, #01	
mcr p15, 0, r0, c1, 0, 0	
mrc p15, 0, r0, c1, 0, 0	禁止 MMU
and r0, ~#01	
mcr p15, 0, r0, c1, 0, 0	

### 4.3.1 使能 MMU 时的情况

当 CPU 请求存储访问时，首先在 TLB 中查找虚拟地址，如果该虚拟地址对应的地址变换条目（页表项）不在 TLB 中，CPU 从位于内存中的页表中查询对应于该虚拟地址的页表项，并把相应的结果添加到 TLB 中，这样 CPU 下次再访问该页表项时就可以直接从 TLB 中获取。如果 TLB 已经装满，还应该根据一定的淘汰算法进行替换。

在得到了需要的页表项后，将进行以下操作：

- 1) 得到该虚拟地址对应的物理地址；
- 2) 根据页表项中的缓冲特性控制位 C (Cachable) 和 B (Bufferable) 的值决定是否缓存该存储访问的结果；
- 3) 根据存取权限控制位和所属存储域的权限设置情况确定该存储访问是否被允许；
- 4) 对于不允许缓存 (uncached) 的存储访问，使用上面步骤 1) 中得到的物理地址访问存储空间。对于允许缓存 (cached) 的存储访问，如果 cache 命中，则忽略物理地址；如果 cache 没有命中，则使用上面步骤 1) 中得到的物理地址访问存储空间，并把该数据块读到 cache 中。

### 4.3.2 禁止 MMU 时的情况

如果系统禁止 MMU 功能，那么 CPU 的存储访问规则如下：

- 1) 当禁止 MMU 时, 是否支持 cache 和 Write Buffer 由各具体芯片确定。如果芯片规定当禁止 MMU 的同时也禁止 cache 和 Write Buffer, 则存储访问不考虑缓冲特性控制位 C 和 B 的设置情况; 如果芯片规定当禁止 MMU 时可以使能 cache 和 Write Buffer, 则数据访问时, C=0, B=0, 指令读取时, 如果使用数据和指令分开的 TLB, 则 C=1, 如果使用统一的 TLB, 则 C=0。
- 2) 存储访问不进行权限控制, MMU 也不会产生存储访问中止信号。
- 3) 所有的物理地址和虚拟地址相同, 也就是使用所谓的平板存储模式。

### 4.3.3 使能/禁止 MMU 时应注意的问题

在使能和禁止 MMU 时要注意以下几点:

- 1) 在使能 MMU 前, 要在内存中建立好页表, 同时 CP15 中的各相关寄存器必须完成初始化。
- 2) 如果使用的不是平板存储模式 (物理地址和对应虚拟地址相等), 在禁止/使能 MMU 时, 虚拟地址和物理地址的对应关系会发生改变, 此时应该清除 cache 中的当前地址变换条目 (TLB)。
- 3) 如果完成禁止/使能 MMU 的代码的物理地址和虚拟地址不同, 禁止/使能 MMU 时将造成很大麻烦, 因此强烈建议完成禁止/使能 MMU 的代码的物理地址和虚拟地址相同 (或者与地址无关)。

## 4.4 ARM 处理器地址变换过程

虚拟存储空间到物理存储空间的映射是以内存块为单位进行的, 虚拟存储空间中的一块连续存储空间被映射成物理存储空间中同样大小的一块连续存储空间。每一个地址变换条目 (页表项) 记录了一个虚拟存储空间的存储块的基地址与物理存储空间相应的一个存储块的基地址的对应关系。根据存储块大小不同, 可以有多种地址变换。

ARM 处理器支持的存储块大小有以下几种:

- 1) 段 (section): 大小为 1MB 的存储块。
- 2) 大页 (Large Page): 大小为 64KB 的存储块。
- 3) 小页 (Small Page): 大小为 4KB 的存储块。
- 4) 极小页 (Tiny Page): 大小为 1KB 的存储块。

通过采用适当的访问控制机制, 还可以将大页分成大小为 16KB 的子页, 也可将小页分成大小为 1KB 的子页, 但极小页不能再细分, 只能以 1KB 大小的整页为单位。

ARM 处理器采用两级页表实现地址映射：

1) 一级页表中包含以段为单位的地址变换条目或者指向二级页表的指针，一级页表实现的地址映射粒度较大。

2) 二级页表中包含以大页、小页和极小页为单位的地址变换条目。

当以二级分页管理某段存储空间时，要同时设置一级页表项和二级页表项，一个一级页表项对应一段（1section=1MB）虚拟存储空间的映射关系，一个一级页表项对应一张二级页表，这张二级页表中的所有页表项合在一起对应了前面一级页表项所对应的一段虚拟存储空间的映射关系。ARM 处理器的二级页表分为粗粒度二级页表和细粒度二级页表，一张粗粒度二级页表的最大容量为 1KB，一张细粒度二级页表的最大容量为 4KB，不管是粗粒度还是细粒度页表，每张页表都对应一段虚拟存储空间的映射关系。所以粗粒度二级页表的页表项（页描述符）最小只能描述 4KB 大小（小页）虚拟存储空间的映射关系，如果再小就没有足够的页表空间来存放一段虚拟存储空间的全部页表项（因为粗粒度二级页表的最大容量为 1KB），而细粒度二级页表的页描述符最小可以描述 1KB 大小（极小页）的虚拟存储空间的映射关系。

综上所述，我们可以归纳出以下 6 种 ARM 处理器的地址变换方法：

- 1) 分段地址变换，这种变换只需要一级地址变换，而下面 5 种都需要两级地址变换；
- 2) 粗粒度大页地址变换；
- 3) 粗粒度小页地址变换（Linux 通常使用的地址变换方法）；
- 4) 细粒度大页地址变换；
- 5) 细粒度小页地址变换；
- 6) 细粒度极小页地址变换。

在讲解以上各种地址变换方法之前我们先介绍一下一级映射描述符和二级映射描述符的定义。

#### 4.4.1 MMU 的一级映射描述符

ARM 处理器 MMU 的一级映射描述符编码格式如下所示：

31	20	19	14	13	12	11	10	9	8	5	4	3	2	1	0
粗粒度二级页表基址														00	
段基址	0	TEX	AP	P	Domain	U			01					10	
细粒度二级页表基址														11	

位	说 明
Bit [1:0]:	一级描述符类型标识。00: 无效; 01: 粗页; 10: 段; 11: 细页
C、B:	该一级描述符对应的存储空间的 cache 和 Write Buffer 特性控制位, 如表 4-4 所示
U:	由用户定义
Domain:	指明该存储空间所属的域号 0~15; 见前面“MMU 中的域”说明
P:	保护标志位; 只有 ARMv5 或以上版本处理器有定义
AP:	访问权限控制位, 详见前面“ARM 处理器的存储域”说明
TEX:	在 ARMv5 以上版本处理器中有定义, 如果 TEX 等于 1, 表示系统支持写时分配 cache
粗粒度二级页表基址:	如果 bit[1:0]=01, 那么该存储空间是二级分大页管理, 1 页=4KB, 粗粒度二级页表基址表示的是页描述符表的基址, 1KB 对齐, 一张页描述符表占 1KB 空间, 256 个页描述符条目, 描述了 1MB 空间的映射关系以及访问权限和域控制属性
段基址:	如果 bit[1:0]=10, 那么该存储空间是分段管理, 该段描述符号描述了 1MB 存储空间的映射关系以及访问权限和域控制属性, 段基址就是该存储空间的物理地址, 1MB 对齐
细粒度二级页表基址:	如果 bit[1:0]=11, 那么该存储空间是二级分小页管理, 1 页=1KB, 细粒度二级页表基址表示的是页描述符表的基址, 4KB 对齐, 一张页描述符表占 4KB 空间, 1 024 个页描述符条目, 描述了 1MB 空间的映射关系以及访问权限和域控制属性

表 4-4 一级描述符对应的存储空间的 cache 和 write buffer 特性控制位

C	B	写通类型 cache	写回类型 cache	可选择写通属性的写回类型 cache
0	0	Uncached/unbuffered	Uncached/unbuffered	Uncached/unbuffered
0	1	Uncached/buffered	Uncached/buffered	Uncached/buffered
1	0	cached/unbuffered	不可预测	写通 cached/buffered
1	1	Cached/buffered	Cached/buffered	写回 cached/buffered

#### 4.4.2 MMU 的二级映射描述符

ARM 处理器 MMU 的二级映射描述符编码格式如下所示:

31	16	15	12	11	10	9	8	7	6	5	4	3	2	1	0
														00	
大页基址	0	AP3	AP2	AP1	AP0	C	B	01							
小页基址	AP3	AP2	AP1	AP0	C	B	10								
极小页基址	0	AP	C	B	11										

位	说 明
Bit [1:0]:	页描述符类型标识。00: 无效; 01: 大页; 10: 小页; 11: 极小页
C、B:	该二级描述符对应的存储空间的 cache 和 Write Buffer 特性控制位, 如表 4-5 所示
APx:	访问权限控制位, 详见前面“MMU 中的域”说明
TEX:	在 ARMv5 以上版本处理器中有定义, 如果 TEX 等于 1, 表示系统支持写时分配 cache
大页基址:	该描述符对应的大页存储空间的物理地址, 64KB 对齐
小页基址:	该描述符对应的小页存储空间的物理地址, 4KB 对齐
极小页基址:	该描述符对应的极小页存储空间的物理地址, 1KB 对齐

表 4-5 二级描述符对应的存储空间的 cache 和 write buffer 特性控制位

C	B	写通类型 cache	写回类型 cache	可选择写通属性的写回类型 cache
0	0	Uncached/unbuffered	Uncached/unbuffered	Uncached/unbuffered
0	1	Uncached/buffered	Uncached/buffered	Uncached/buffered
1	0	cached/unbuffered	不可预测	写通 cached/buffered
1	1	Cached/buffered	Cached/buffered	写回 cached/buffered

#### 4.4.3 基于段的地址变换过程（见图 4-1）



图 4-1 分段地址变换过程

#### 4.4.4 粗粒度大页地址变换过程（见图 4-2）

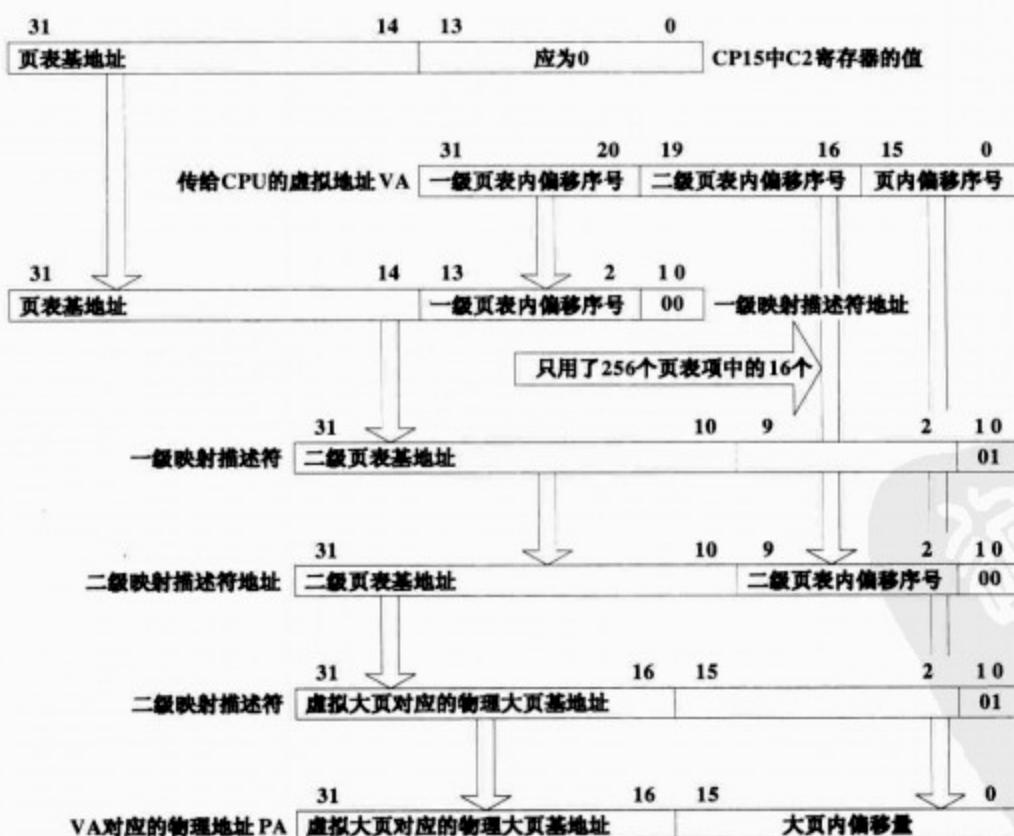


图 4-2 粗粒度大页地址变换过程

#### 4.4.5 粗粒度小页地址变换过程（见图 4-3）

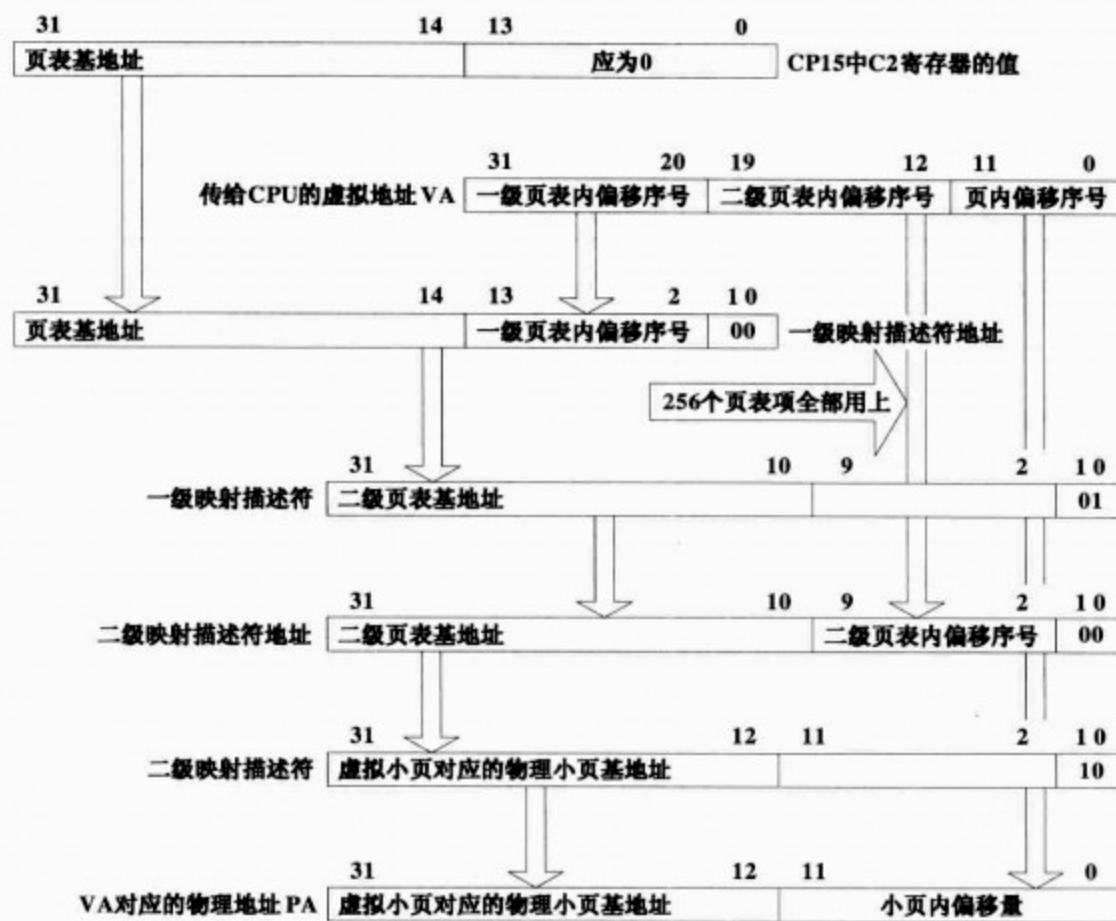


图 4-3 粗粒度小页地址变换过程

#### 4.4.6 细粒度大页地址变换过程（见图 4-4）

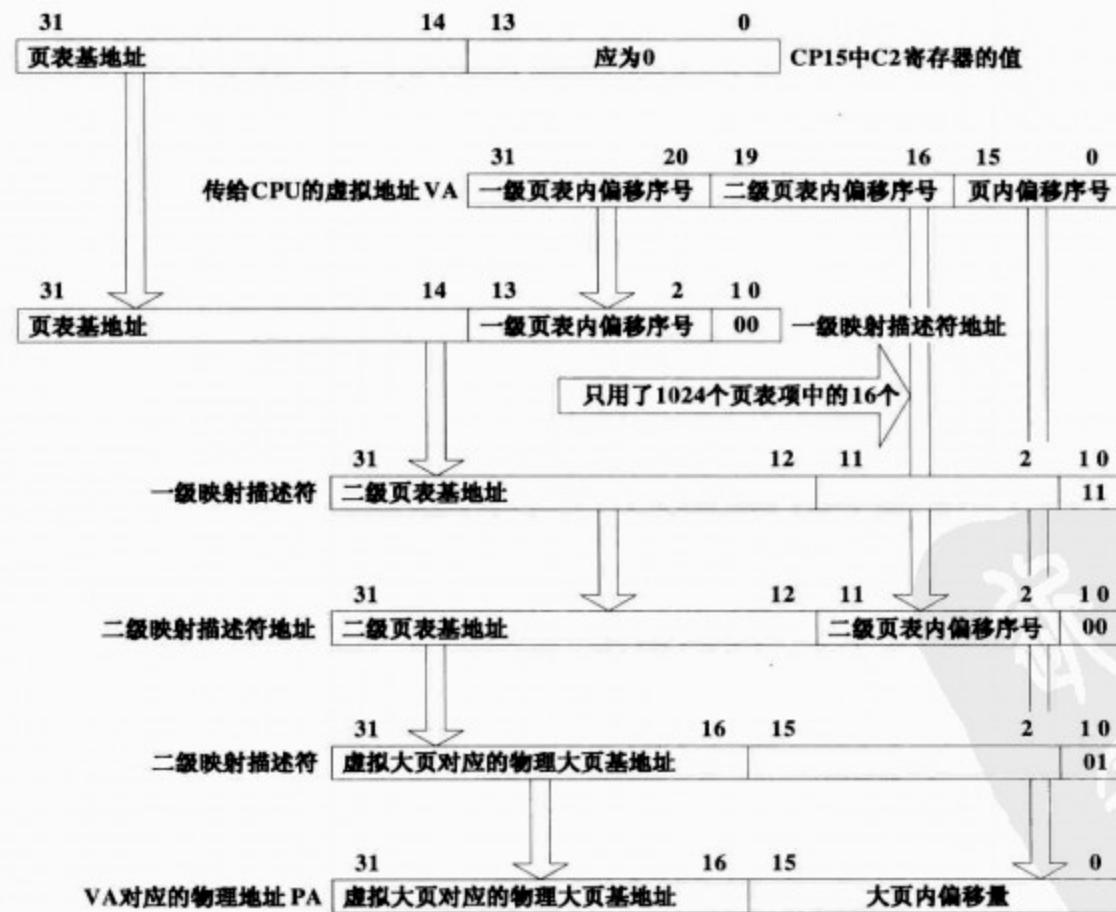


图 4-4 细粒度大页地址变换过程

#### 4.4.7 细粒度小页地址变换过程（见图 4-5）

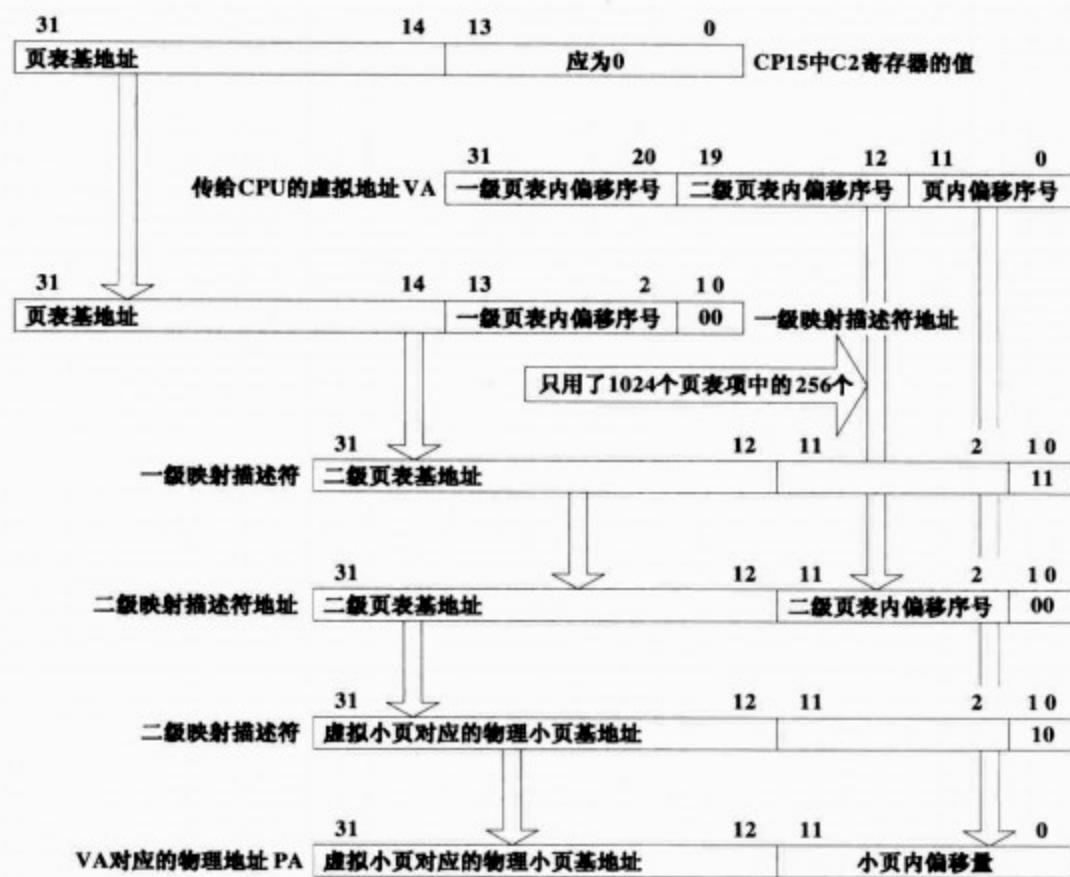


图 4-5 细粒度小页地址变换过程

#### 4.4.8 细粒度极小页地址变换过程（见图 4-6）

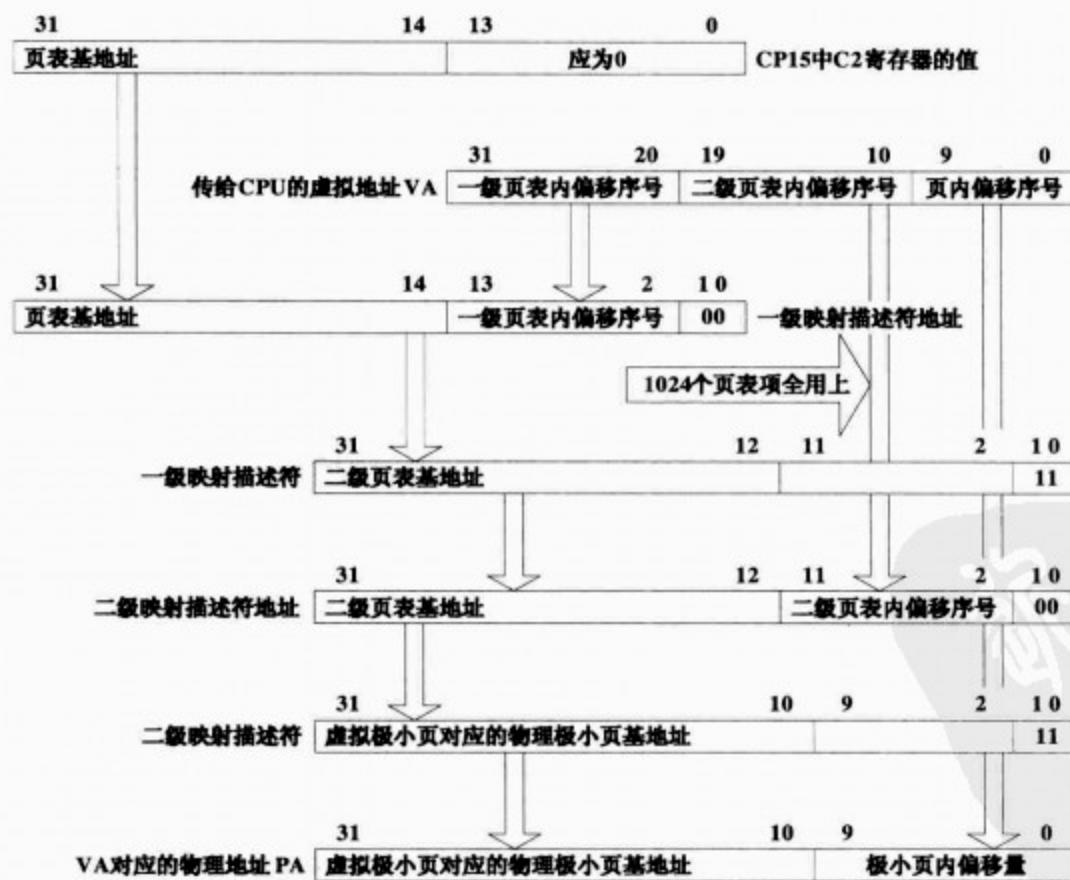


图 4-6 细粒度极小页地址变换过程

## 4.5 ARM 存储空间访问权限控制

在 ARM 处理器中，MMU 将整个存储空间分成最多 16 个域，记作 D0~D15，每个域对应一定的存储区域，该区域具有相同的访问控制属性。

在 ARM 处理器中，MMU 中的每个域的访问权限分别由 CP15 的 C3 寄存器中的两位来设定，C3 寄存器刚好可以设置 16 个域的访问权限。C3 寄存器的域定义如表 4-6 所示。

表 4-6 CP15 的 C3 寄存器的域定义

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

C3 寄存器的 D0~D15 各占两位，它们分别控制 D0~D15 共 16 域的访问类型，具体说明如表 4-7 所示。

表 4-7 域的访问控制字段编码及含义

控制位编码	访问类型	含    义
0b00	没有访问权限	这时访问该域将产生访问失效
0b01	客户 (client)	根据 CP15 的 C1 控制寄存器中的 R 和 S 位以及页表中地址变换条目中的访问权限控制位 AP 来确定是否允许各种系统工作模式的存储访问
0b10	保留	使用该值会产生不可预知的结果
0b11	管理者 (manager)	不考虑 CP15 的 C1 控制寄存器中的 R 和 S 位以及页表中地址变换条目中的访问权限控制位 AP，在这种情况下不管系统工作在特权模式还是用户模式都不会产生访问失效

当域访问权限控制位设置为上面的 0b01 即客户类型权限时，用户模式以及特权模式的访问权限则由 CP15 的 C1 控制寄存器中的 R 和 S 位以及页表中地址变换条目中的访问权限控制位 AP 两位来确定，具体说明如表 4-8 所示。

表 4-8 MMU 中存储访问权限控制

AP[1:0]	S	R	特权 (privileged) 模式访问权限	用户 (user) 模式访问权限
00	0	0	不能访问	不能访问
00	1	0	只读	不能访问
00	0	1	只读	只读
00	1	1	访问导致不可预知的结果	访问导致不可预知的结果
01	x	x	读/写	不能访问
10	x	x	读/写	只读
11	x	x	读/写	读/写

## 4.6 TLB 操作

对 TLB 的操作是通过访问 CP15 的 C8 和 C10 寄存器来完成的，写 CP15 的 C8/C10 寄存器的指令格式如下所示：

```
mcr p15, 0, <rd>, <c8>, crm, <opcode_2>  
mcr p15, 0, <rd>, <c10>, crm, <opcode_2>
```

### 4.6.1 使 TLB 内容无效

使 TLB 中的内容无效是通过写 CP15 的寄存器 C8 来实现的，指令如表 4-9 所示。

表 4-9 使 TLB 内容无效操作指令

指 令	<opcode_2>	<crm>	<rd>	含 义
mcr p15, 0, rd, c8, c7, 0	0b000	0b0111	0	使整个统一 cache 或整个数据和指令 cache 无效
mcr p15, 0, rd, c8, c7, 1	0b001	0b0111	虚拟地址	使统一 cache 中的单个地址变换条目无效
mcr p15, 0, rd, c8, c5, 0	0b000	0b0101	0	使整个指令 cache 无效
mcr p15, 0, rd, c8, c5, 1	0b001	0b0101	虚拟地址	使指令 cache 中的单个地址变换条目无效
mcr p15, 0, rd, c8, c6, 0	0b000	0b0110	0	使整个数据 cache 无效
mcr p15, 0, rd, c8, c6, 1	0b001	0b0110	虚拟地址	使数据 cache 中的单个地址变换条目无效

### 4.6.2 锁定 TLB 内容

锁定 TLB 是通过写 CP15 的寄存器 C10 来实现的。

锁定 TLB 中 N 条地址变换条目的操作步骤说明如下：

- 1) 确保在整个锁定过程中不会产生异常中断，可以通过禁止中断等方法实现；
- 2) 如果锁定的是指令 TLB 或者统一的 TLB，将 base=N、victim=N、P=0 写入 C10；
- 3) 使整个将要锁定的 TLB 无效；
- 4) 如果想要锁定的是指令 TLB，确保与锁定过程所涉及的指令相关的地址变换条目已经加载到指令 TLB 中；如果想要锁定的是数据 TLB，确保与锁定过程所涉及的数据相关的地址变换条目已经加载到数据 TLB 中；如果系统使用的是统一的数据 TLB 和指令 TLB，上述两条都要保证；
- 5) 对于 I=0 到 N-1，重复执行下面操作：

将 base=i、victim=i、P=1 写入寄存器 C10，将每一条想要锁定到快表中的地址变换条目读

取到快表中。对于数据 TLB 和统一 TLB 可以使用 ldr 指令读取一个涉及该地址变换条目的数据，将该地址变换条目读取到 TLB 中。对于指令 TLB，通过操作寄存器 C7，将相应的地址变换条目读取到指令 TLB 中；

- 6) 将 base=N、victim=N、P=0 写入寄存器 C10。

### 4.6.3 解除 TLB 中被锁定的地址变换条目

解除 TLB 中被锁定的地址变换条目，可以使用以下操作步骤：

- 1) 通过操作寄存器 C8，使无效 TLB 中被锁定的地址变换条目；
- 2) 将 base=0、victim=0、P=0 写入 C10 寄存器。

## 4.7 存储访问失效

ARM 处理器通过以下两种机制来检测存储访问失效，进而中止 CPU 的运行：

- 1) MMU 硬件模块检测与内存管理相关的存储访问失效，一旦 MMU 检测到存储访问失效，它向 CPU 发出通知，并将存储访问失效的相关信息保存到寄存器中，具体说就是将失效状态保存在 CP15 的 C5 寄存器中，将导致失效的地址保存在 CP15 的 C6 寄存器中，详见前面 CP15 的 C5、C6 寄存器说明。这种存储访问失效叫做 MMU 失效（MMU Fault）。
- 2) 外部存储系统向 CPU 报告存储访问失效，这种机制叫做外部存储访问中止（External Abort）。

以上两种访问失效统称为存储访问中止（Abort）。如果存储访问中止发生在数据访问周期，CPU 将产生数据访问中止异常中断，即 Data Abort；如果存储访问中止发生在指令预取周期，那么 CPU 产生指令预取中止异常中断，即 Prefetch Abort。

### 4.7.1 MMU 失效（MMU Fault）

MMU 可以产生以下 4 大类存储访问失效：

- 1) 地址对齐失效：地址对齐失效指发生在数据访问周期，比如访问字单元（4 字节）时地址的 bit[1:0]不全是 0 或者访问半字（双字节）单元时地址的 bit[0]不等于 0；访问字节单元不会产生地址对齐失效，指令预取周期也不会产生地址对齐失效。
- 2) 地址变换失效：当一级地址变换条目的 bit[1:0]为 0b00 时，表示产生了基于段的地址变换失效；当二级地址变换条目的 bit[1:0]为 0b00 时，表示产生了基于页的地址变换失效。

3) 域控制失效：如果所访问的存储地址对应的一级地址变换条目中 bit[8:5]的值所指的存储域的两位控制位被设置为 0，就说明产生了域控制失效，域控制失效可能是基于段的也可能是基于页的。

4) 访问权限控制失效：参见第 4.5 节“ARM 存储空间访问权限控制”，如果出现了访问权限冲突就产生访问权限控制失效。

以上 4 大类存储访问失效又可细分为 15 种类型，见前面 CP15 的 C5 寄存器说明。

当发生存储访问失效时，存储系统可以中止 3 种存储访问：cache 内容预取、非缓冲的存储器访问和页表访问。

#### 4.7.2 外部存储访问失效（External Abort）

由外部存储系统向 CPU 报告存储访问失效。

外部存储访问失效通过一个外部存储访问失效引脚实现，下面的存储访问操作可以通过这种机制中止和重启动：

- 1) 读操作；
- 2) 非缓冲的写操作；
- 3) 一级描述符的获取；
- 4) 二级描述符的获取；
- 5) 非缓冲的存储区域中的信号量操作。

**注意：**在系统中标记为可外部终止的存储区域不要进行可缓存的写操作。