ARM的cache和写缓冲器（write buffer）

原创 2013年10月27日 14:02:47 标签：arm / 5748

y695385603

原创 粉丝 喜欢 评论
44 19 2 3

等级： 博客 4 访问量： 7万+
积分： 1220 排名： 4万+

一、cache简介

通常CPU与外部主存之间的访问速度差距很大，因为外部主存的低速率读写而降低了CPU的执行效率，所以引入了高速缓冲存储器cache，cache存储器是一种容量小，速度快的存储器。其实cache说到底就是一块速度非常快的内存。

而cache经常与写缓冲器（write buffer）一起使用，使用writer buffer的目的是，将处理器和cache从较慢的对主存的写操作中脱离出来。

下图是一个计算机系统结构中的存储器分层结构，更好的显示了**cache**和写缓冲器在存储层次结构中所处的位置。

从图中可以看出，凡是从cache中向主存中写入数据时都需要经过写缓冲器。

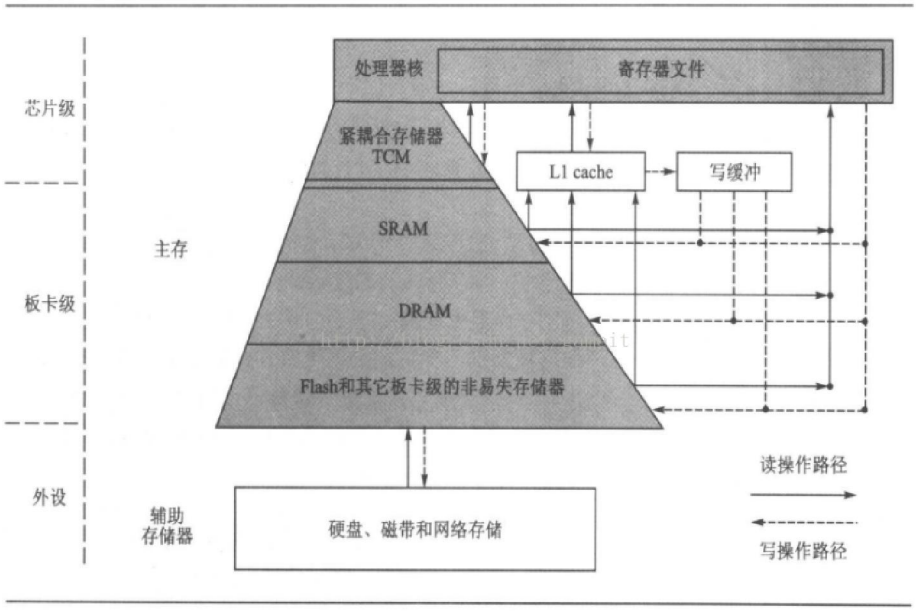


图 12.1 存储器的层次结构

下图显示了有**cache**的系统和没有**cache**的系统之间的差别：

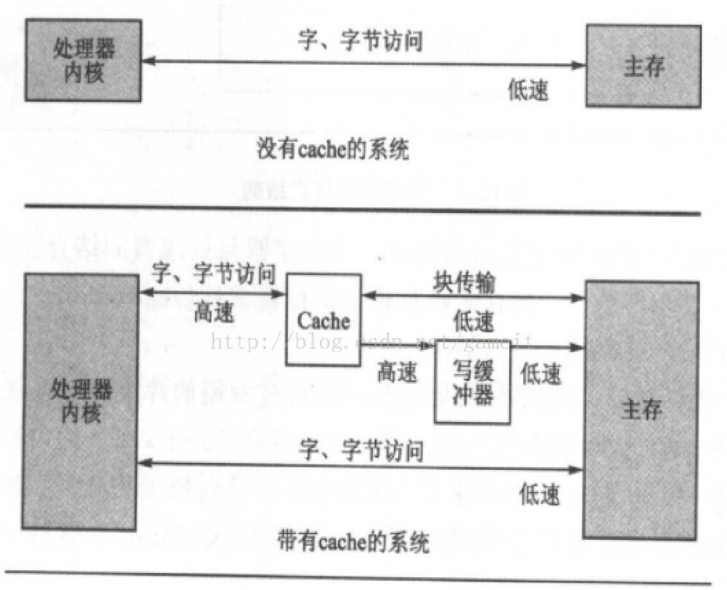


图 12.2 cache、处理器内核及主存之间的关系

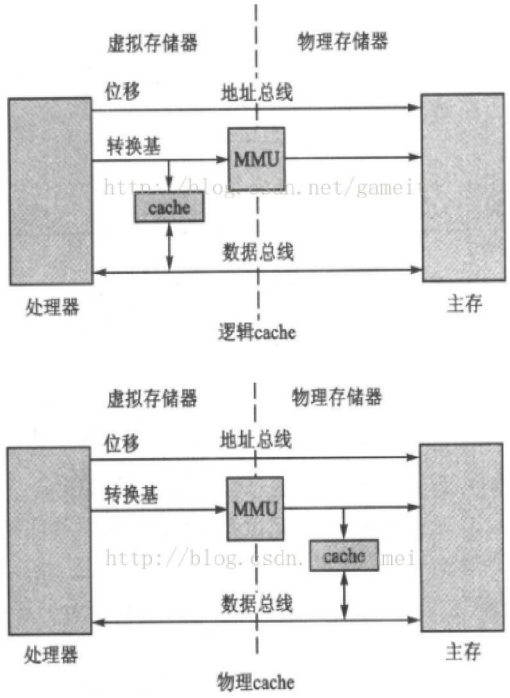
cache和存储管理单元：

cache可以在放在处理器内核与MMU之间，也可以放在MMU与物理存储器之间。

在处理器和MMU之间的cache称为**逻辑cache**，逻辑cache在虚拟地址空间存储数据，处理器可以直接通过cache访问数据，而无需通过MMU。逻辑cache又称为虚拟cache。

在MMU与物理存储器之间的cache称为**物理cache**，当处理器访问存储器时，MMU必须先把虚拟地址转换成物理地址，cache存储器才可以向处理器内核提供数据。

下图是两中**cache**：



二、cache结构

1、cache结构

图12.4是一个4KB大小的**cache存储器**，这个cache存储器分为256行，每行由三部分组成：**目录存储段，状态信息段和数据项段**。

目录存储段：用来记录每个cache行所对应的主存中的地址，即告诉cache这个行里存放的数据时从主存的什么地方拷贝进来的。这个目录项里的数据被称为**“cache标签”**
状态信息段：用来记录状态信息的状态位，两个常见的**状态信息位**是**有效位(valid bit)**和**脏位(dirty bit)**。有效位用来标记当前的cache行是活动的，即当前的cache行已经存放了从主存中取得的数据，并且可以被处理器所使用。脏位用来标记该cache行中所含的内容是与主存中的内容是否一致，即该cache行中的数据已经被改变，但还没有被写回到主存中，主存中的数据还是旧的数据。
数据项段：从主存中拷贝来的数据放在数据项段

这个cache存储器拥有4KB的大小，被分成256行，而每行所占的大小为4个字（16字节）大小，所以为4KB。注意**cache标签**和**状态信息位**所占的空间并不计算在内。
以上就是一个**cache存储器**的结构，详细看下图**12.4**：

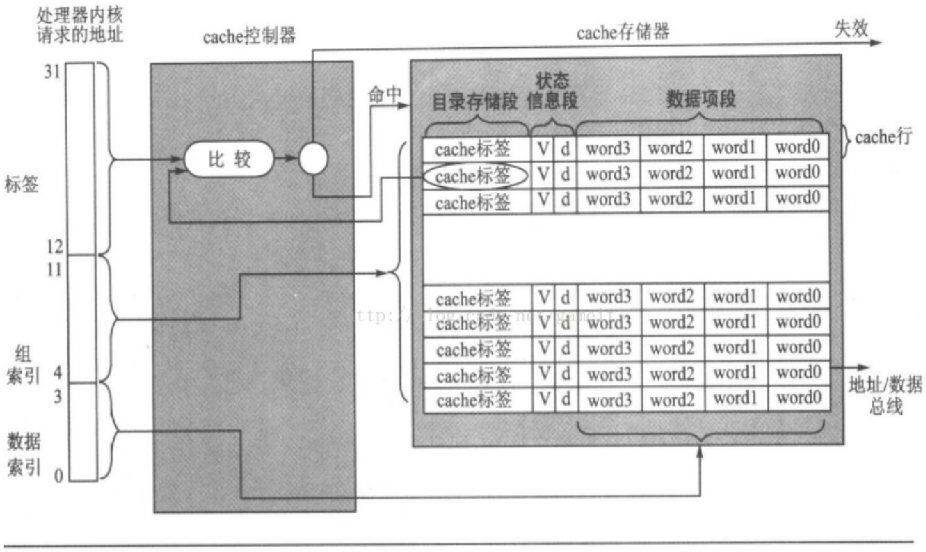


图 12.4 拥有 256 行的 4 KB cache，每行 4 个 32 位字（word）

2、cache控制器的基本操作

cache控制器是用来控制cache的，它将主存中的数据或代码自动拷贝到cache存储器中，搬移工作自动完成。
当CPU访问主存时，在请求被发送到存储控制器之前，主存地址会被cache控制器拦截，然后cache控制器将要访问的地址分为三个部分：**标签域，组索引域**和**数据索引域**。
注：**cache存储器中的一行是对应很多个主存地址的。如图12.5中一行对应了1MB的主存地址空间**

组索引：用来确定某一个cache行。因为一共有256个行，而组索引宽度是8位，所以正好查找一个cache行。
标签： 其实就是用来确定当前要访问的主存地址是否已经被缓存到了所找到的cache行中。如果该标签与cache行中存放的cache标签的值相等，且该cache行是活动的，则cache命中，否则失效。
数据索引：确定要访问该cache行中的哪一个数据。

当组索引确定好某个cache行后，cache控制器便会检查该行的有效位，如果当前cache行是有效的，即处于活动状态，并且标签域的值与cache标签相等，则说明要访问的主存地址中的数据已经被缓存到了该cache行中，cache命中。
否则cache失效，这时，cache控制器从要访问的主存地址中拷贝整个cache行的数据大小到cache存储器中。这种拷贝操作称为**cache行填充**。
在cache命中的情况下，控制器直接从cache存储器中得到数据。使用**数据索引**在当前cache行中选择实际的要获得的数据或指令。

详细请看下图**12.5**：

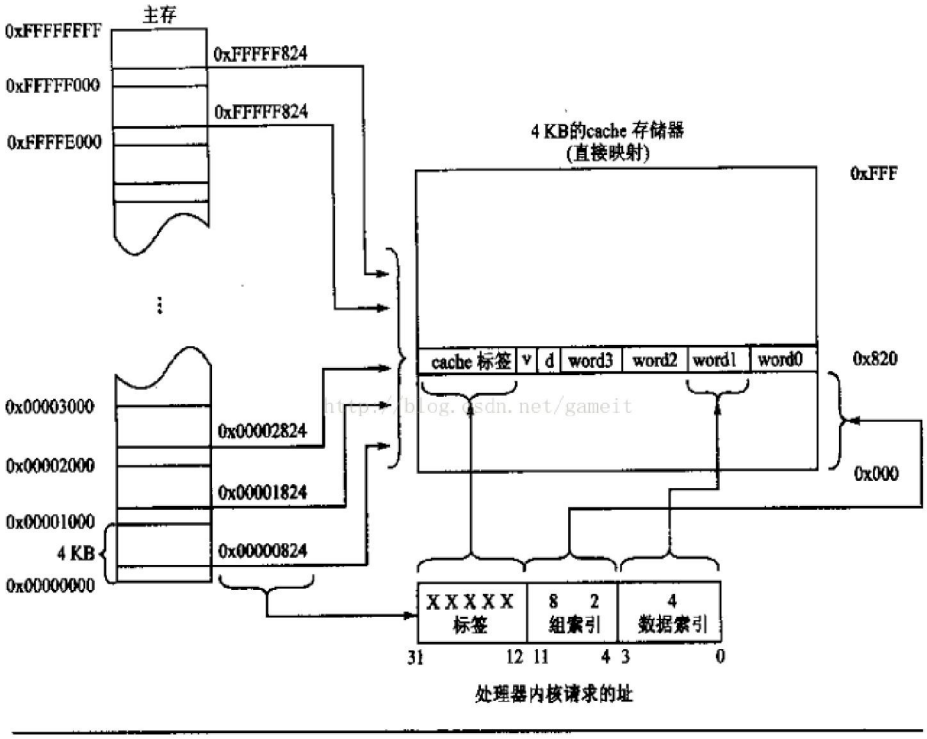


图 12.5 主存与直接映射 cache 的映射

cache的映射方式有三种：直接映射，组相联映射和全相联映射

- 直接映射

上图**12.5**便是一个直接映射的显示，在直接映射cache中，由于主存的容量要远远大于cache的容量，所以为了让大容量的主存能够全部映射到小容量的cache中，往往是主存中的多个地址对应cache中的一行。在上图**12.5**的例子中行，**组索引**的值为**0x82**，于是便找到了cache存储器中对应的一个cache行。

注意，因为这一个cache行对应了多个主存地址，在本例中一个cache行对应了**1M**的主存地址范围，而在任一时刻，该cache行只能对应着**1M**地址范围中的一个主存地址，所以要用**标签**再确定该cache行中具体的一个主存地址。用cache行中的cache标签与标签域中的标签比较，如果相等，则**cache命**中，然后便对该cache行进行读写操作。如果不相等，则**cache失效**，那么这个cache行的整个数据会被删除，并替换为**CPU**要访问的主存地址的内容，同时也将数据送到处理器中，这个过程称为**替换**。

- 组相联映射

直接映射虽然简单方便，但是如果程序同时用到了对应于同一个cache行中的两个主存地址，那么就会发生冲突，结果就是导致这个cache行不停的进行替换操作。所以就有了**组相联映射**。

组相联映射将这cache存储器中的256个行分为了**4路**，每路有**64个cache行**。这时根据组索引找到的 cache行不再是一个，而是同时找到 4个 cache行，所以才被称为**组索引**。**每路**中包含一个**cache行**。如下图**12.7**所示，便是一个**组相联映射**的cache。

因为现在每路中有 64 个 cache 行，所以**组索引**的宽度变成了**6位**，而**标签域**的标签宽度变成了**12位**。

直接映射中组索引一次只能对应一个cache 行，而现在组相联映射中组索引一次能对应**4个cache行**，所以现在主存中的一个地址可以存放到**4个cache行中的任意一行**。 而在直接映射中，主存中的一个地址只能存放到**对应的一个cache行中**。所以一个cache行被**替换**的概率减少为原来的四分之一。

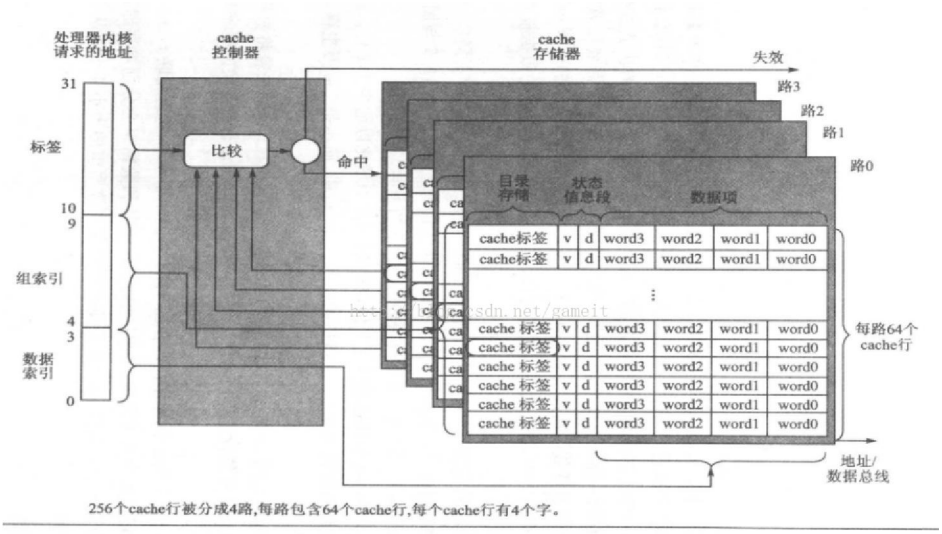
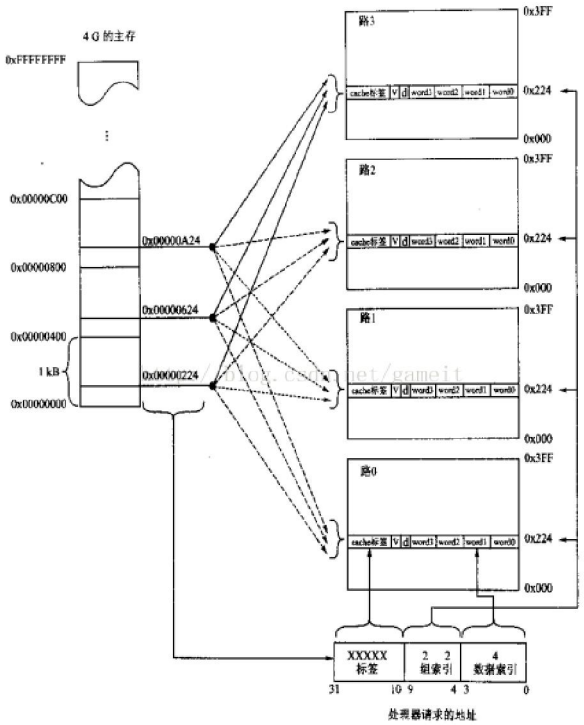


图 12.7 一个 4 KB 四路组相联 cache

一个组相联映射的实例：



- 全相联映射

随着**cache**的相联度提高，冲突的可能性小了。理想的目标是，尽量提高组相联程度，使主存中的一个地址能够映射到**cache**存储器中的任意的**cache**行，这样的映射关系被称为全相联映射。

简单来说直接映射可以看成是一个只有1路的组相联映射，每路有256个cache行。
上面的组相联映射有4路，每路有64个cache行。
而全相联映射有很多路，每路中只有一个cache行。
下图是一个**64**路的组相联映射：

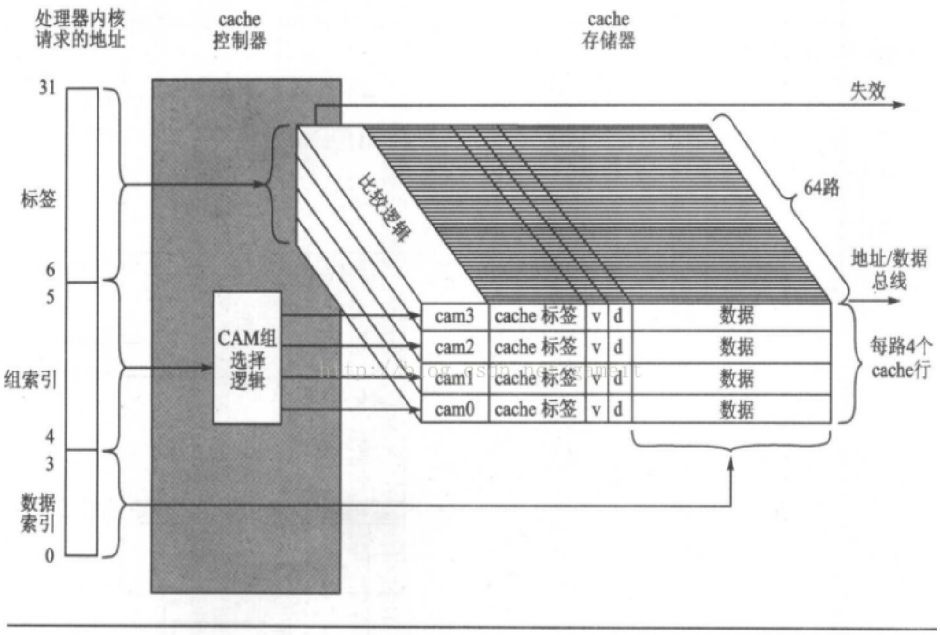


图 12.9 ARM940T——使用 CAM 的 4 KB 64 路组相联 D-cache

三、写缓冲器

写缓冲器是一个非常小的告高速存储缓冲器，用来临时存放处理器将要写入到主存中的数据，在没有写缓冲器的系统中，处理器直接写数据到主存中。在带有写缓冲器的系统中，cache直接将数据先写到写缓冲器中，然后写缓冲器再以低速写入主存中。这就将高速的CPU和cache从对主存的低速读写中脱离了出来。

写缓冲器同时还改善了cache的性能，这体现在cache行被替换时。当cache控制器要替换出一个脏的cache行时，它只将该cache行中的数据放入写缓冲器中，而不写入主存。这样就可以快速填充新的cache行数据，处理器就可以继续从cache存储器中读 / 写数据。

写缓冲器中的数据在没有被写入主存之前，是不能被读取的。同样，被替换的cache行在写缓冲器中时也不能进行读操作。这也是为什么写缓冲器的深度通常比较小的原因之一，一般只有几个cache行的深度。

四、cache策略

有3种可以决定cache操作的策略：**写策略，替换策略和分配策略**，写策略决定了处理器执行写操作时数据存放的位置。替换策略在cache失效的情况下，决定选择替换出哪一路的cache行。分配策略决定cache控制器在何时将要分配cache行。

- 写策略

直写法：

如果cache使用直写策略，那么CPU在写cache命中时，将同时修改cache和主存中的内容，以确保cache和主存中数据一致性。在这种策略下，CPU在每次写cache时也要写相应主存单元。由于要访问主存，直写法的速度比回写法要慢一些。

回写法：

如果cache使用回写策略，那么CPU在写cache命中时，只向cache存储器中写数据，而不立即写入主存。这样，主存地址与相应的cache行数据有可能不一致。cache中的数据时最新的，而主存中的数据可能是较早的，还没有被更新过。

配置成回写法的cahche要使用到cache行的状态信息段中的一个或多个脏位（dirty bit）。当向cache存储器中某一行写入数据时，他会将脏位设置为1。如果此后访问该cache行，那么通过脏位就可以知道cache中含有主存中没有的数据。如果一个脏位被置位的cache行要被替换，那么该cache行中的数据会被自动的先写入到相应的主存单元中，以防止主存中的信息丢失。

- 替换策略

替换策略用于选择要替换cache行中的数据，应该替换哪个cache行。总的来说，组索引在各个way（路）中选择可用的一组cache行，而替换策略决定在该组中哪一路中的cache行会被新的数据替换。

轮转法：
简单地将当前分配的cache行的下一行作为被替换的行。

伪随机替换法：
从特定的位置上随机地选出一行替换出去。

大多数ARM核两种策略都支持，相比之下，轮转法有更好的可预测性。然而，轮转法在存储器访问发生很小的变化时，有可能照成cache性能有较大的变化。

- 分配策略

在cache失效发生时，ARM的cache可以采取两种策略来分配cache行：读操作分配策略，读写操作分配策略。

读操作分配策略：
在进行存储器写操作时，如果**cache**未命中，只是简单的将数据写入到主存中，而不写到**cache**中，如果**cache**命中，才会将数据写到**cache**中。但是在进行存储器读操作时，如果没有命中，则会替换**cache**行中的内容。

读写操作分配策略：
当进行存储器写操作时，如果**cache**未命中，**cache**控制器会将要访问的主存地址中的数据读到**cache**行中，并执行写操作，将数据写入到**cache**中。

五、清理和清除cache

ARM使用**清理**和**清除**表示对cache的两种基本操作。

清理就是把脏的（即被改写过的）cache行强制写到主存，并把cache行中的脏位清零。清理cache可以重建cache和主存之间的一致性，它只是用在回写策略的D-cache上。
清除就是指清除cache中存储的全部数据，其实就是将cache使无效，将cache行中的有效位清零，让所有访问这个cache行的操作都不命中。

对cache进项清理和清除的操作是通过协处理器CP15中的寄存器C7实现的，具体操作指令看CP15协处理器寄存器详解。

Cache&WriteBuffer学习总结

关于Cache与WriteBuffer的原理可以查阅相关博客或书籍。这里我只想讨论哪种情况下不能使用Cache与WriteBuffer，以及使用Cache与WriteBuffer时给存储系统带来的一致...


llf021421 2011年04月21日 01:28 3088

理解write buffer与cache的好文章--摘自Arm system developer's guide chapter 12

A cache is a small, fast array of memory placed between the processor core and mainmemory that store...


better_yin 2010年09月09日 16:51 1260

cache和write buffer

Cache是一种容量小、速度快的存储器阵列，它位于主存和处理器内核之间，保存着最近一段时间处理器涉及到的主存块内容，主要是为了缓解慢速存储器和处理器之间的速度不匹配造成的访问瓶颈问题。write bu...


yubingjunj 2009年12月23日 15:56 3383

ARM 的FIFO机制，数据缓冲区

看来许多人还没有真正理解FIFO的作用和优点，仍然停留在每收发一个字符就要中断处理一次的老思路上。UART收发FIFO主要是为了解决收发中断过于频繁而导致的CPU效率不高的问题。 FIFO的必要性。...


G1036583997 2013年10月14日 21:17 3587

ARM内存管理（MMU）详解

嵌入式系统中，存储系统差别很大，可包含多种类型的存储器件，如 FLASH ， SRAM ， SDRAM ， ROM 等，这些不同类型的存储器件速度和宽度等各不相同；在访问存储单元时，可能采取平板式的地...


yimu13 2010年11月28日 10:46 11097

ARM外设寄存器Cache一致性问题

ARM外设寄存器Cache一致性问题 2012-05-22 09:44:14 分类： 嵌入式 /* bank4 DM9000 ,关闭cache和写缓冲,否则出...


dcx1205 2014年11月02日 21:31 960

三种Cache写入方式原理简介

三种Cache写入方式原理简介 在386以上档次的微机中，为了提高系统效率，普遍采用Cache（高速缓冲存储器），现在的系统甚至可以拥有多级Cache。Cache实际上是位于CPU与DRAM主...


szu030606 2012年06月29日 20:06 3943

Arm cache 研究

Cache的工作原理 Cache的工作原理是基于程序访问的局部性。 对大量典型程序运行情况的分析结果表明，在一个较短的时间间隔内，由程序产生的地址往往集中在存储器逻辑地址空间的很小范围内...


bournechen 2009年12月02日 17:00 7776

ARM处理器的Cache之cortex a8

原文地址：http://blog.chinaunix.net/uid-28458801-id-3494289.html Cache 是位于 CPU与主存储器DRAM（ Dynamic ...

kangear 2013年06月14日 16:30 2801