

情感天地

职业规划

人才招聘

供需广告

书友会论坛

匠人手记

ARM程序分析与设计

嵌入式网络系统设计

ARM Linux入门与实例

圈圈教你玩USB

裸奔式实时操作系统

感悟设计

专题与交流

论坛站务发展

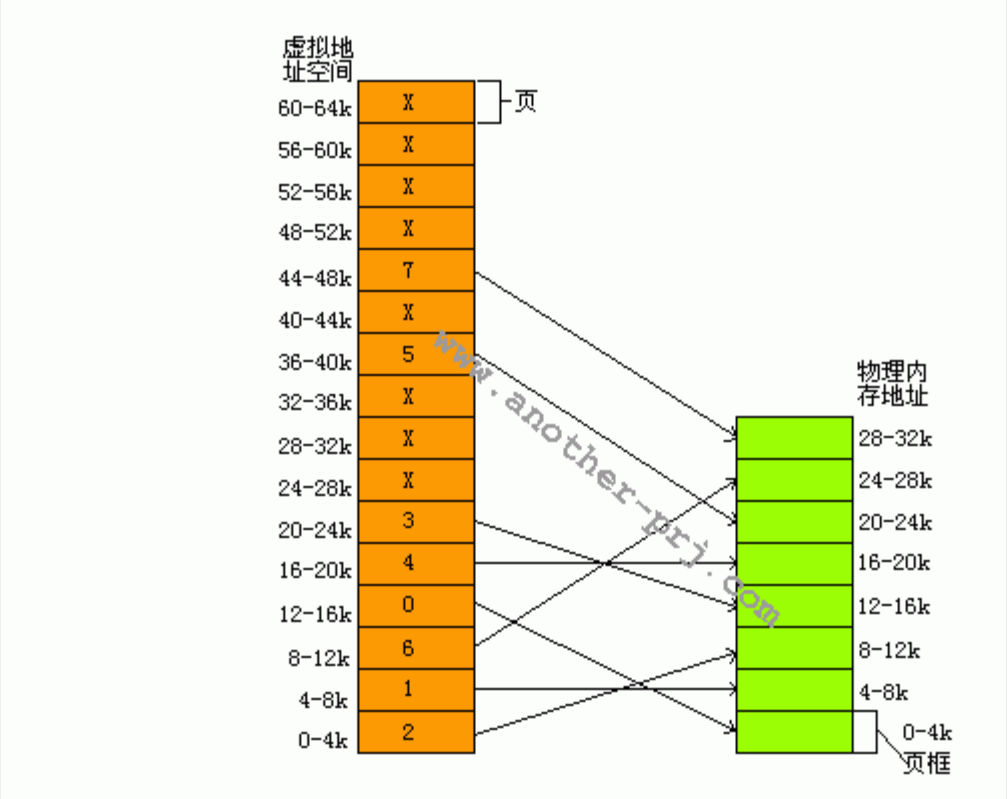
21IC发展大家谈

站长公告

回收站

存地址总线上，而是送到内存管理单元——MMU（主角终于出现了：J）。他由一个或一组芯片组成，一般存在与协处理器中，其功能是把虚拟地址映射为物理地址。

大多数使用虚拟存储器的系统都使用一种称为分页（**paging**）。虚拟地址空间划分成称为页（**page**）的单位,而相应的物理地址空间也被进行划分，单位是页框(**frame**)。页和页框的大小必须相同。接下来配合图片我以一个例子说明页与页框之间在MMU的调度下是如何进行映射的



mmu1.gif (5.3 KB)

2007-3-23 21:19

在这个例子中我们有一台可以生成16位地址的机器，它的虚拟地址范围从0x0000~0xFFFF(64K),而这台机器只有32K的物理地址，因此他可以运行64K的程序，但该程序不能一次性调入内存运行。这台机器必须有一个达到可以存放64K程序的外部存储器（例如磁盘或是FLASH），以保证程序片段在需要时可以被调用。在这个例子中，页的大小为4K,页框大小与页相同（这点是必须保证的，内存和外围存储器之间的传输总是以页为单位的），对应64K的虚拟地址和32K的物理存储器，他们分别包含了16个页和8个页框。

我们先根据上图解释一下分页后要用到的几个术语，在上面我们已经接触了页和页框，上图中绿色部分是物理空间，其中每一格表示一个物理页框。橘本人

本主题由 古道热肠 于 2009-8-9 11:24 加入精华



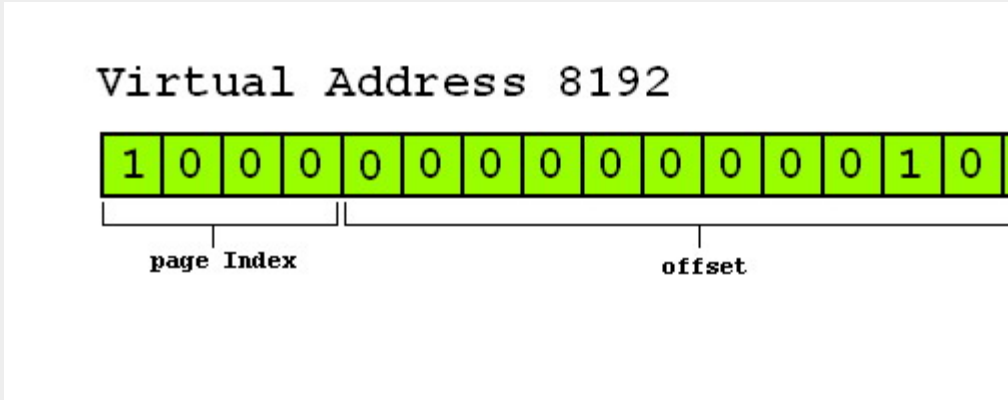
帖子: 762
积分: 2237
🌙⭐

金升阳电源技术交流会，时尚数码礼品等你拿！
Digi-Key、Future、Arrow、Mouser、RS
Components、OnlineComponents实时库存在线查询订购！

itelectron 发表于 2009-8-8 20:56 | 只看该作者 回复 引用 返回版面 TOP

2楼:

到一个页中的所有地址),8196的二进制码如下图所示:



mmu2.jpg (15.79 KB)

2007-3-23 21:19

该地址的页号索引为0010（二进制码），既索引的页为页2，第二部分为000000000100（二进制），偏移量为 4。页2中的页框号为6（页2映射在页框6，见上图），我们看到页框6的物理地址是24~28K。于是MMU计算出虚拟地址8196应该被映射成物理地址 24580（页框首地址+偏移量=24576+4=24580）。同样的，若我们对虚拟地址1026进行读取，1026的二进制码为 0000010000000010，page index=0000=0,offset=010000000010=1026。页号为0，该页映射的页框号为2，页框2的物理地址范围是 8192~12287，故MMU将虚拟地址1026映射为物理地址9218（页框首地址+偏移量=8192+1026=9218）以上就是MMU的工作过程。

下面我们针对s3c2410的MMU(注1)进行讲解。

S3c2410总共有4种内存映射方式，分别是：


- 1. Fault (无映射)
- 2. Coarse Page (粗表)
- 3. Section (段)
- 4. Fine Page (细表)

我们以**Section(段)**进行说明。

ARM920T是一个32bit的CPU,它的虚拟地址空间为 $2^{32}=4G$ 。而在**Section**模式，这4G的虚拟空间被分成一个一个称为段（**Section**）的单位(与我们上面讲的页在本质上其实是一致的),每个段的长度是1M (而我们之前所使用的页的长度是4K)。4G的虚拟内存总共可以被分成4096个段（ $1M*4096=4G$ ）,因此我们必须

用4096个描述符来对这组段进行描述，每个描述符占用4个Byte，故这组描述符的大小为16KB (4K*4096)，这4096个描述符构成一个表格，我们称其为**Translation Table**.

Section base address	AP	
----------------------	----	--

 [clip_image002.jpg](#) (8.57 KB)
2007-3-23 21:29

上图是描述符的结构

Section base address: 段基地址（相当于页框号首地址）

AP: 访问控制位Access Permission

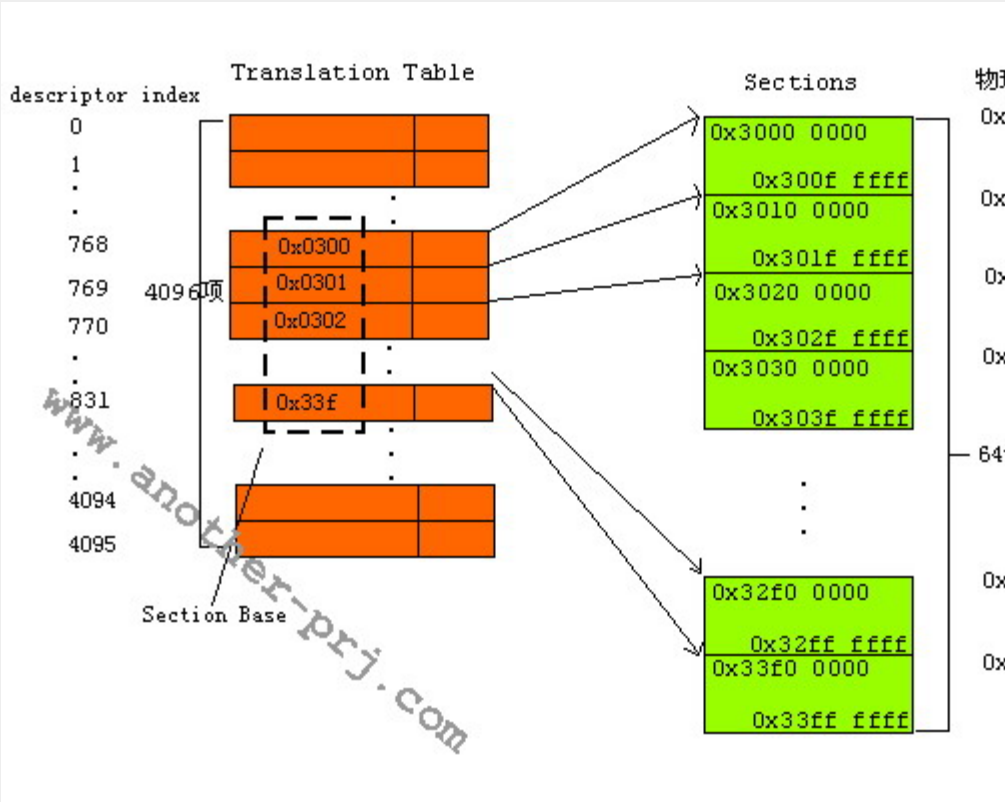
Domain: 访问控制寄存器的索引。Domain与AP配合使用，对访问权限进行检查


C: 当C被置1时为write-through (WT)模式

B: 当B被置1时为write-back (WB)模式

(C,B两个位在同一时刻只能有一个被置1)

下面是s3c2410内存映射后的一个示意图:



 [clip_image001.jpg](#) (55.08 KB)
2007-3-23 21:29

我的s3c2410上配置的SDRSAM大小为64M,该SDRAM的物理地址范围是0x3000 0000~0x33FF FFFF(属于Bank 6)，由于1个Section的大小是1M,所以该物理空间可以被分成64个物理段(页框).

在Section模式下，送进MMU的**虚拟地址(注1)**被分为两部分（这点和我们上面举的例子是一样的），这两部分为 **Descriptor Index**(相当于上面例子的Page Index)和 **Offset**, descriptor index长度为12bit($2^{12}=4096$,从这个关系式你能看出什么? :)), Offset长度为20bit ($2^{20}=1M$, 你又能看出什么? :)) .观察一下一个描述符 (Descriptor) 中的Section Base Address部分，它长度为12 bit，里面的值是**该虚拟段（页）映射成的物理段（页框）的物理地址前12bit**，由于每一个物理段的长度都是1M，所以**物理段首地址的后20bit总是为0x000000**(**每个Section都是以1M对齐**)，确定一个物理地址的方法是 **物理页框基地址+虚拟地址中的偏移部分=Section Base Address<<20+Offset** ,呵呵，可能你有点糊涂了，还是举一个实际例子说明吧。假设现在执行指令

MOV REG, 0x30000012

虚拟地址的二进制码为00110000 00000000 00000000 00010010

前12位是Descriptor Index= 00110000 0000=768,故在Translation Table里面找到第768号描述符，该描述的Section Base Address=0x0300,也就是说描述符所描述的虚拟段（页）所映射的物理段（页框）的首地址为0x3000 0000（**物理段（页框）的基地址=Section Base Address左移20bit=0x0300<<20=0x3000 0000**），而Offset=000000 00000000 00010010=0x12,故虚拟地址0x30000012映射成的物理地址=0x3000 0000+0x12=0x3000 0012（**物理页框基地址+虚拟地址中的偏移**）。你可能会问怎么这个虚拟地址和映射后的物理地址一样？这是由我们定义的映射规则所决定的。在这个例子中我们定义的映射规则是把虚拟地址映射成和他相等的物理地址。我们这样书写映射关系的代码：

```
void mem_mapping_linear(void)
{
    unsigned long descriptor_index, section_base, sdram_base,
    sdram_size;
    sdram_base=0x30000000;
    sdram_size=0x 4000000;
    for (section _base= sdram_base,descriptor_index = section
    _base>>20;
        section _base < sdram_base+ sdram_size;
        descriptor_index+=1;section _base +=0x100000)
    {
        *(mmu_tlb_base + (descriptor_index)) = (section _base>>20) |
        MMU_OTHER_SECDISC;
    }
}
```

上面的这段段代码把虚拟空间0x3000 0000~0x33FF FFFF映射到物理空间0x3000 0000~0x33FF FFFF，由于虚拟空间与物理空间空间相吻合，所以虚拟地址与他们各自对应的物理地址在值上是一致的。当初始完Translation Table之后，记得要把**Translation Table**的首地址(第0号描述符的地址)加载进协处理器CP15的Control Register2(2号控制寄存器)中,该控制寄存器的名称叫做Translation table base (TTB) register。

以上讨论的是descriptor中的Section Base Address以及虚拟地址和物理地址的映射关系,然而MMU还有一个重要的功能，那就是访问控制机制(Access

Permission)。

简单说访问控制机制就是CPU通过某种方法判断当前程序对内存的访问是否合法（是否有权对该内存进行访问），如果当前的程序并没有权限对即将访问的内存区域进行操作，则CPU将引发一个异常，s3c2410称该异常为**Permission fault**，x86架构则把这种异常称之为通用保护异常（**General Protection**），什么情况会引起Permission fault呢？比如处于**User级别的程序要**对一个**System级别的内存区域进行写操作**，这种操作是越权的，应该引起一个Permission fault，搞过x86架构的朋友应该听过保护模式（**Protection Mode**），保护模式就是基于这种思想进行工作的，于是我们也可以这么说：**s3c2410的访问控制机制其实就是一种保护机制**。那s3c2410的访问控制机制到底是由什么元素去参与完成的呢？它们间是怎么协调工作的呢？这些元素总共有：

- 1. 协处理器CP15中**Control Register3: DOMAIN ACCESS CONTROL REGISTER**
- 2. 段描述符中的**AP位**和**Domain位**
- 3. 协处理器CP15中**Control Register1(控制寄存器1)**中的**S bit**和**R bit**
- 4. 协处理器CP15中**Control Register5(控制寄存器5)**
- 5. 协处理器CP15中**Control Register6(控制寄存器6)**

DOMAIN ACCESS CONTROL REGISTER 是访问控制寄存器，该寄存器有效位为32，被分成16个区域，每个区域由两个位组成，他们说明了当前内存的访问权限检查的级别，如下图所示：

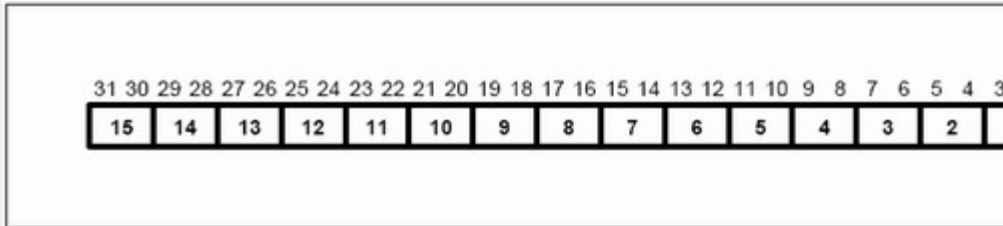


Figure 3-10. Domain Access Control Register Format

mmu5.jpg (16 KB)

2007-3-23 21:19

每区域可以填写的值有4个，分别为00,01,10,11(二进制)，他们的意义如下所示：

Table 3-5. Interpreting Access Control Bits in Domain Access Control Register		
Value	Meaning	Notes
00	No Access	Any access will generate a domain fault.
01	Client	Accesses are checked against the access permission section or page descriptor.
10	Reserved	Reserved. Currently behaves like the no access mode.
11	Manager	Accesses are <i>not</i> checked against the access permission. A permission fault cannot be generated.

mmu6.jpg (32.18 KB)

2007-3-23 21:19

00: 当前级别下, 该内存区域不允许被访问, 任何的访问都会引起一个domain fault

金升阳电源技术交流会, 时尚数码礼品等你拿!
Digi-Key、Future、Arrow、Mouser、RS
Components、OnlineComponents实时库存在线查询订购!



itelectron 发表于 2009-8-8 20:57 | 只看该作者 回复 引用 返回版面 TOP

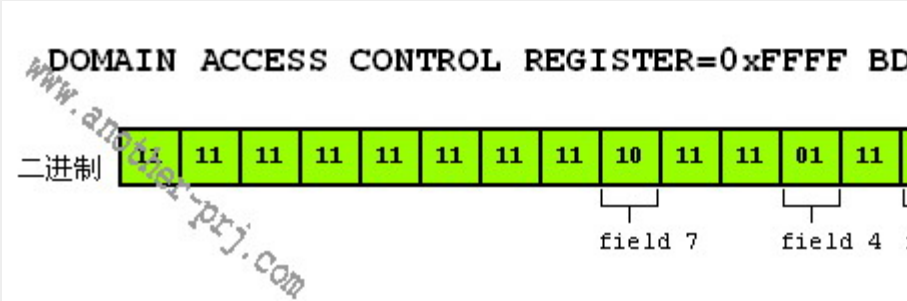
3楼:

帖子: 762
积分: 2237
🌙🌟

00: 当前级别下, 该内存区域不允许被访问, 任何的访问都会引起一个domain fault
01: 当前级别下, 该内存区域的访问必须配合该内存区域的段描述符中AP位进行权检查
10: 保留状态 (我们最好不要填写该值, 以免引起不能确定的问题)
11: 当前级别下, 对该内存区域的访问都不进行权限检查。
我们再来看看discriptor中的Domain区域, 该区域总共有4个bit, 里面的值是对DOMAIN ACCESS CONTROL REGISTER中16个区域的索引. 而AP位配合S bit和A bit对当前描述符描述的内存区域被访问权限的说明, 他们的配合关系如下图所示:

Table 3-6. Interpreting Access Permission (AP) Bits					
AP	S	R	Supervisor Permissions	User Permissions	
00	0	0	No access	No access	Any access permission
00	1	0	Read only	No access	Supervisor permitted
00	0	1	Read only	Read only	Any write permission
00	1	1	Reserved		
01	x	x	Read/write	No access	Access allowed supervisor
10	x	x	Read/write	Read only	Writes in user permission
11	x	x	Read/write	Read/write	All access for both modes
xx	1	1	Reserved		

AP位也是有四个值, 我结合实例对其进行说明.
在下面的例子中, 我们的DOMAIN ACCESS CONTROL REGISTER都被初始化成0xFFFF BDCF, 如下图所示:





mmu7.jpg (23.15 KB)

2007-3-23 21:19

例1:

Discriptor 中的domain=4,AP=10(这种情况下S bit ,A bit 被忽略)

假设现在我要对该描述符描述的内存区域进行访问:

由于domain=4,而DOMAIN ACCESS CONTROL REGISTER中field 4的值是01,系统会对该访问进行访问权限的检查。

假设当前CPU处于Supervisor模式下,则程序可以对该描述符描述的内存区域进行读写操作。

假设当前CPU处于User模式下,则程序可以对该描述符描述的内存进行读访问,若对其进行写操作则引起一个permission fault.

例2:

Discriptor 中的domain=0,AP=10(这种情况下S bit ,A bit 被忽略)

domain=0,而DOMAIN ACCESS CONTROL REGISTER中field 0的值是11,系统对任何内存区域的访问都不进行访问权限的检查。

由于统对任何内存区域的访问都不进行访问权限的检查,所以无论CPU处于合种模式下(Supervisor模式或是User模式),程序对该描述符描述的内存都可以顺利地进行读写操作

例3: Discriptor 中的domain=4,AP=11(这种情况下S bit ,A bit 被忽略)

由于domain=4,而DOMAIN ACCESS CONTROL REGISTER中field 4的值是01,系统会对该访问进行访问权限的检查。

由于AP=11,所以无论CPU处于合种模式下(Supervisor模式或是User模式),程序对该描述符描述的内存都可以顺利地进行读写操作

例4:

Discriptor 中的domain=4,AP=00, S bit=0,A bit=0

由于domain=4,而DOMAIN ACCESS CONTROL REGISTER中field 4的值是01,系统会对该访问进行访问权限的检查。

由于AP=00, S bit=0,A bit=0,所以无论CPU处于合种模式下

(Supervisor模式或是User模式),程序对该描述符描述的内存都只能进行读操作,否则引起permission fault.

通过以上4个例子我们得出两个结论:

1. 对某个内存区域的访问是否需要权限检查是由该内存区域的描述符中的Domain域决定的。

2. 某个内存区域的访问权限是由该内存区域的描述符中的AP位和协处理器CP15中Control Register1(控制寄存器1)中的S bit和R bit所决定的。

关于访问控制机制我们就讲到这里.

注1:对于s3c2410来说,MMU是以Modify Visual Address(MVA)进行寻址的,这个地址是Virtual Address的一个变换,我将在以后谈论到进程切换

的时候向大家介绍MVA

原文地址 <http://lionwq.spaces.eepw.com.cn/articles/article/item/17578>

 金升阳电源技术交流会，时尚数码礼品等你拿！
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购！



帖子: 762
积分: 2237


 [itelectron](#) 发表于 2009-8-8 21:12 | 只看该作者 回复 引用 返回版面 TOP

4楼:

NND以前 看过些资料 一直都比较迷糊 这个讲得太好了

 金升阳电源技术交流会，时尚数码礼品等你拿！
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购！



帖子: 762
积分: 2237


 [itelectron](#) 发表于 2009-8-8 21:26 | 只看该作者 回复 引用 返回版面 TOP

5楼:

以下内容转载自中计报

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。指令地址的分布本来就是连续的，再加上循环程序段和子程序段要重复执行多次。因此，对这些地址的访问就自然地具有时间上集中分布的倾向。

数据分布的这种集中倾向不如指令明显，但对数组的存储和访问以及工作单元的选择都可以使存储器地址相对集中。这种对局部范围的存储器地址频繁访问，而对此范围以外的地址则访问甚少的现象，就称为程序访问的局部性。

根据程序的局部性原理，可以在主存和CPU通用寄存器之间设置一个高速的容量相对较小的存储器，把正在执行的指令地址附近的一部分指令或数据从主存调入这个存储器，供CPU在一段时间内使用。这对提高程序的运行速度有很大的作用。这个介于主存和CPU之间的高速小容量存储器称作高速缓冲存储器(Cache)。

系统正是依据此原理，不断地将与当前指令集相关联的一个不太大的后继指令集从内存读到Cache，然后再与CPU高速传送，从而达到速度匹配。

CPU对存储器进行数据请求时，通常先访问Cache。由于局部性原理不能保证所请求的数据百分之百地在Cache中，这里便存在一个命中率。即CPU在任一时刻从Cache中可靠获取数据的几率。

命中率越高，正确获取数据的可靠性就越大。一般来说，Cache的存储容量比主存的容量小得多，但不能太小，太小会使命中率太低；也没有必要过大，过大不仅会增加成本，而且当容量超过一定值后，命中率随容量的增加将不会有明显地增长。

只要Cache的空间与主存空间在一定范围内保持适当比例的映射关系，Cache的命中率还是相当高的。

一般规定Cache与内存的空间比为4：1000，即128kB Cache可映射32MB内存；256kB Cache可映射64MB内存。在这种情况下，命中率都在90%以上。至于没有命中的数据，CPU只好直接从内存获取。获取的同时，也把它拷进Cache，以备下次访问。

Cache的基本结构

Cache通常由相联存储器实现。相联合存储器的每一个存储块都具有额外的存储信息，称为标签(Tag)。当访问相联存储器时，将地址和每一个标签同时进行比较，从而对标签相同的存储块进行访问。Cache的3种基本结构如下：

全相联Cache

在全相联Cache中，存储的块与块之间，以及存储顺序或保存的存储器地址之间没有直接的关系。程序可以访问很多的子程序、堆栈和段，而它们是位于主存储器的不同部位上。因此，Cache保存着很多互不相关的数据块，Cache必须对每个块和块自身的地址加以存储。当请求数据时，Cache控制器要把请求地址同所有地址加以比较，进行确认。这种Cache结构的主要优点是，它能够在给定的时间内去存储主存器中的不同的块，命中率高；缺点是每一次请求数据同Cache中的地址进行比较需要相当的时间，速度较慢。

直接映像Cache

直接映像Cache不同于全相联Cache，地址仅需比较一次。在直接映像Cache中，由于每个主存储器的块在Cache中仅存在一个位置，因而把地址的比较次数减少为一次。其做法是，为Cache中的每个块位置分配一个索引字段，用Tag字段区分存放在Cache位置上的不同的块。单路直接映像把主存储器分成若干页，主存储器的每一页与Cache存储器的大小相同，匹配的主存储器的偏移量可以直接映像为Cache偏移量。Cache的Tag存储器(偏移量)保存着主存储器的页地址(页号)。

以上可以看出，直接映像Cache优于全相联Cache，能进行快速查找，其缺点是当主存储器的组之间做频繁调用时，Cache控制器必须做多次转换。

组相联Cache

组相联Cache是介于全相联Cache和直接映像Cache之间的一种结构。这种类型的Cache使用了几组直接映像的块，对于某一个给定的索引号，可以允许有几个块位置，因而可以增加命中率和系统效率。

Cache与DRAM存取的一致性

在CPU与主存之间增加了Cache之后，便存在数据在CPU和Cache及主存之间如何存取的问题。读写各有2种方式。

贯穿读出式(Look Through)

该方式将Cache隔在CPU与主存之间，CPU对主存的所有数据请求都首先送到Cache，由Cache自行在自身查找。如果命中，则切断CPU对主存的请求，并将数据送出；不命中，则将数据请求传给主存。

该方法的优点是降低了CPU对主存的请求次数，缺点是延迟了CPU对主存的访问时间。

旁路读出式(Look Aside)

在这种方式中，CPU发出数据请求时，并不是单通道地穿过Cache，而是向Cache和主存同时发出请求。由于Cache速度更快，如果命中，则Cache在将数据回送给CPU的同时，还来得及中断CPU对主存的请求；不命中，则Cache不做任何动作，由CPU直接访问主存。它的优点是没有时间延迟，缺点是每次CPU对主存的访问都存在，这样，就占用了一部分总线时间。

写穿式(Write Through)

任一从CPU发出的写信号送到Cache的同时，也写入主存，以保证主存的数据能同步地更

新。

它的优点是操作简单，但由于主存的慢速，降低了系统的写速度并占用了总线的时间。

回写式(Copy Back)

为了克服贯穿式中每次数据写入时都要访问主存，从而导致系统写速度降低并占用总线时间的弊病，尽量减少对主存的访问次数，才有了回写式。

它是这样工作的：数据一般只写到Cache，这样有可能出现Cache中的数据得到更新而主存中的数据不变(数据陈旧)的情况。但此时可在Cache 中设一标志地址及数据陈旧的信息，只有当Cache中的数据被再次更改时，才将原更新的数据写入主存相应的单元中，然后再接受再次更新的数据。这样保证了Cache和主存中的数据不致产生冲突。

...

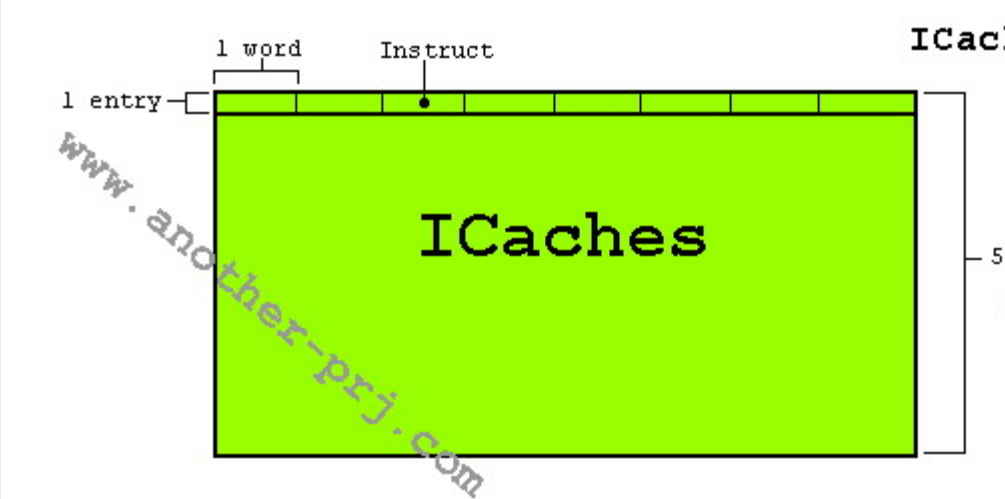
.....

你可以通过<http://www.chinaunix.net/jh/45/180390.html>阅读完全文

s3c2410 内置了指令缓存 (ICaches) ,数据缓存 (DCaches) ,写缓存 (write buffer) , 物理地址标志读写区 (Physical Address TAG RAM),CPU将通过它们来提高内存访问效率。


我们先讨论指令缓存 (ICaches) 。

ICaches使用的是虚拟地址，它的大小是16KB,它被分成512行(entry)，每行8个字 (8 words,32Bits) 。



 **ICaches.jpg** (19.94 KB)

2007-3-26 15:41

 金升阳电源技术交流会，时尚数码礼品等你拿！
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购！

 [itelectron](#) 发表于 2009-8-8 21:27 | 只看该作者 回复 引用 返回版面 TOP

6楼:

当系统上电或重起 (Reset) 的时候，ICaches功能是被关闭的，我们必须往lcr bit置1去开启它，lcr bit在CP15协处理器中控制寄存器1的第12位 (关闭ICaches功能则是往该位置0)。ICaches功能一般是在MMU开启之后被使用的 (为了降低MMU查表带来的开销)，但有一点需要注意，并不是说MMU被开启了ICaches才会被开启，正如本段刚开始讲



帖子: 762
积分: 2237
 

的，ICaches的开启与关闭是由lcr bit所决定的，无论MMU是否被开启，只要lcr bit被置1了，ICaches就会发挥它的作用。

大家是否还记得descriptor（描述符）中有一个C bit我们称之为Ctt,它是指明该描述符描述的内存区域内的内容（可以是指令也可以是数据）是否可以被Cache，若Ctt=1,则允许Cache,否则不允许被Cache。于是CPU读取指令出现了下面这些情况：

1. 如果CPU从Caches中读取到所要的一条指令（cache hit）且这条指令所在的内存区域是Cacheble的（该区域

所属描述符中Ctt=1），则CPU执行这条指令并从Caches中返回（不需要从内存中读取）。

2. 若CPU从Caches中读取不到所要的指令（cache miss）而这条指令所在的内存区域是Cacheble的（同第1点），则CPU将从内存中

读取这条指令，同时，一个称为“8-word linefill”的动作将发生，这个动作是把该指令所处区域的8个word写进

ICaches的某个entry中，这个entry必须是没有被锁定的（对锁定这个操作感兴趣的朋友可以找相关的资料进行了解）

3. 若CPU从Caches中读取不到所要的指令（cache miss）而这条指令所在的内存区域是UnCacheble的（该区域所属描

述符中Ctt=0），则CPU将从内存读取这条指令并执行后返回（不发生linefill）

通过以上的说明，我们可以了解到CPU是怎么通过ICaches执行指令的。你可能会会有这个疑问，ICaches总共只有512个条目（entry），当512个条目都被填充完之后，CPU要把新读取近来的指令放到哪个条目上呢？答案是CPU会把新读取近来的8个word从512个条目中选择一个对其进行写入，那CPU是怎么选出一个条目来的呢？这就关系到ICaches的替换法则（replacemnet algorithm）了。ICaches的replacemnet algorithm有两种，一种是Random模式另一种Round-Robin模式，我们可以通过CP15协处理器中寄存器1的RR bit对其进行指定（0 = Random replacement 1 = Round robin replacement），如果有需要你还可以进行指令锁定（INSTRUCTION CACHE LOCKDOWN）。

关于替换法则和指令锁定我就不做详细的讲解，感兴趣的朋友可以找相关的资料进行了解。

接下来我们谈数据缓存（DCaches）和 写缓存（write buffer）

DCaches使用的是虚拟地址，它的大小是16KB,它被分成512行（entry），每行8个字（8 words,32Bits）。每行有两个修改标志位（dirty bits），第一个标志位标识前4个字，第二个标志位标识后4个字，同时每行中还有一个TAG 地址（标签地址）和一个valid bit。

与ICaches一样，系统上电或重起（Reset）的时候，DCaches功能是被关闭的，我们必须往Ccr bit置1去开启它，Ccr bit在CP15协处理器中控制寄存器1的第2位（关闭DCaches功能则是往该位置0）。与ICaches不同，DCaches功能是必须在MMU开启之后才能被使用的。

我们现在讨论的都是DCaches,你可能会问那Write Buffer呢？他和DCaches区别是什么呢？其实DCaches和Write Buffer两者间的操作有着非常紧密的联系，很抱歉，到目前为止我无法说出他们之间有什么根本上的区别（-_-!!!），但我能告诉你什么时候使用的是DCaches,什么时候使用的是Write Buffer.系统可以通过Ccr bit对Dcaches的功能进行

开启与关闭的设置，但是在s3c2410中却没有确定的某个bit可以用来开启或关闭Write Buffer...你可能有点懵...我们还是来看一张表吧，这张表说明了DCaches,Write Buffer和CCr,Ctt (descriptor中的C bit),Btt(descriptor中的B bit)之间的关系，其中“Ctt and Ccr”一项里面的值是 Ctt与Ccr进行逻辑与之后的值（Ctt&&Ccr）。

Table 4-1. Data Cache and Write Buffer Configuration		
Ctt and Ccr	Btt	Data cache, write buffer and memory access be
0 (1)	0	Non-cached, non-buffered (NCNB) Reads and writes are not cached and always perform accesses on the ASB. Writes are not buffered. The CPU halts until the write is complete. Cache hits should never occur. (2)
0	1	Non-cached buffered (NCB) Reads and writes are not cached, and always perform accesses on the ASB. Cache hits should never occur. Writes are placed in the write buffer and will appear on the ASB as soon as the write is placed in the write buffer. Reads may be externally aborted. Writes can not be externally aborted.
1	0	Cached, write-through mode (WT) Reads which hit in the cache will read the data from the cache and do not perform an access on the ASB. Reads which miss in the cache cause a linefill. All writes are placed in the write buffer and will appear on the ASB as soon as the write is placed in the write buffer. Writes which hit in the cache update the cache. Writes cannot be externally aborted.
1	1	Cached, write-back mode (WB) Reads which hit in the cache will read the data from the cache and do not perform an ASB access. Reads which miss in the cache cause a linefill. Writes which miss in the cache are placed in the write buffer and will appear on the ASB. The CPU continues execution as soon as the write is placed in the write buffer. Writes which hit in the cache update the cache and mark the cache line as dirty, and do not cause an ASB access. Cache write-backs are buffered. Writes (Cache write-misses and cache write-backs) cannot be externally aborted.

 [DCaches.jpg](#) (114.94 KB)

2007-3-26 15:41

从上面的表格中我们可以清楚的知道系统什么时候使用的是DCaches,什么时候使用的是Write Buffer，我们也可以看到DCaches的写回方式是怎么决定的（write-back or write-through）。

在这里我要对Ctt and Ccr=0进行说明，能够使Ctt and Ccr=0的共有三种情况，分别是
Ctt =0， Ccr=0
Ctt =1， Ccr=0
Ctt =0， Ccr=1
我们分别对其进行说明。

情况1（Ctt =0， Ccr=0）：这种情况下CPU的DCaches功能是关闭的（Ccr=0），所以CPU存取数据的时候不会从DCaches里进行数据地查询，CPU直接去内存存取数据。
情况2（Ctt =1， Ccr=0）：与情况1相同。

情况3 (Ctt =0, Ccr=1)：这种情况下DCaches功能是开启的，CPU读取数据的时候会先从DCaches里进行数据地查询，若DCaches中没有合适的数 据，则CPU会去内存进行读取，但此时由于Ctt =0 (Ctt 是descriptor中的C bit,该bit决定该descriptor所描述的内存区域是否可以被Cache)，所以CPU不会把读取到的数据Cache到DCaches(不发生linefill)。

到此为止我们用两句话总结一下DCaches与Write Buffer的开启和使用：

- 1. DCaches与Write Buffer的开启由Ccr决定。
- 2. DCaches与Write Buffer的使用规则由Ctt和Btt决定。

DCaches与ICaches有一个最大的不同，ICaches存放的是指令，DCaches存放的是数据。程序在运行期间指令的内容是不会改变的，所以ICaches中指令所对应的内存空间中的内容不需要更新。但是数据是随着程序的运行而改变的，所以DCaches中数据必须被及时的更新到内存（这也是为什么ICaches没有写回操作而DCaches提供了写回操作的原因）。提到写回操作，就不得不提到PA TAG 地址（物理标签地址）这个固件，它也是整个Caches模块的重要组成部分。

简单说PA TAG 地址（物理标签地址）的功能是指明了写回操作必须把DCaches中待写回内容写到物理内存的哪个位置。不知道你还记不记得，DCaches中每个entry中都有一个PA TAG 地址（物理标签地址），当一个linefill发生时，被Cache的内容被写进了DCaches,同时被Cache的物理地址则被写入了PA TAG 地址（物理标签地址）。除了TAG 地址（标签地址），还有两个称为dirty bit（修改标志位）的位出现在DCaches的每一个entry中，它们指明了当前entry中的数据是否已经发生了改变（发生改变简称为变“脏”，所以叫dirty bit,老外取名称可真有意思 _-!!!）。如果某个entry中的dirty bit置位了，说明该entry已经变脏，于是一个写回操作将被执行，写回操作的目的地址则是由PA TAG 地址（物理标签地址）索引到的物理地址。

关于Caches，Write Buffer更详细的内容请大家阅读s3c2410的操作手册：]

金升阳电源技术交流会，时尚数码礼品等你拿！
Digi-Key、Future、Arrow、Mouser、RS
Components、OnlineComponents实时库存在线查询订购！



帖子: 896
积分: 3234
🌙⭐

当来211C 发表于 2009-8-9 10:06 | 只看该作者 回复 引用 返回版面 TOP

7楼:

太酷了...

金升阳电源技术交流会，时尚数码礼品等你拿！
Digi-Key、Future、Arrow、Mouser、RS
Components、OnlineComponents实时库存在线查询订购！



古道热肠 发表于 2009-8-9 11:23 | 只看该作者 回复 引用 返回版面 TOP

8楼:

帖子: 4477
积分: 14248
😊🌙⭐

写得很不错，通俗易懂！

SIGNATURE -----

以VS1003B和山景SOC芯片为背景，倾心研制数字化语音录放产品。

排忧解难:xg_2004_sy@126.com

 金升阳电源技术交流会，时尚数码礼品等你拿！
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购！



帖子: 2804
积分: 9941
😊🌙⭐

 [netjob](#) 发表于 2009-8-9 16:42 | 只看该作者 回复 引用 返回版面 TOP

9楼:

学习了~

 金升阳电源技术交流会，时尚数码礼品等你拿！
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购！



帖子: 896
积分: 3234
🌙⭐

 [甞来21IC](#) 发表于 2009-8-9 17:26 | 只看该作者 回复 引用 返回版面 TOP

10楼:

不过楼主，没有贴完，要看全还得去

<http://lionwq.spaces.eepw.com.cn/articles/article/item/17578>
<http://www.chinaunix.net/jh/45/180390.html>

SIGNATURE -----

~~~~~抄代码累了，练练垒IC~~~~~

 金升阳电源技术交流会，时尚数码礼品等你拿！  
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购！




帖子: 20  
积分: 59  
🌙

 [myitlover](#) 发表于 2009-8-10 09:48 | 只看该作者 回复 引用 返回版面 TOP

11楼:

而对于一个64位的CPU，它的地址范围为0~0xFFFFFFFFFFFFFFFF (64T)。

这个不对吧，1T=1024G

 金升阳电源技术交流会，时尚数码礼品等你拿！  
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS](#)

Components、OnlineConpunents实时库存在线查询订购!



帖子: 11147

积分: 35307



[huangqi412](#) 发表于 2009-8-10 16:38 | 只看该作者 回复 引用 返回版面 TOP

12楼:

MARK 慢满看

金升阳电源技术交流会，时尚数码礼品等你拿!  
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购!



帖子: 113

积分: 339



[arm\\_fan168](#) 发表于 2009-8-14 14:32 | 只看该作者 回复 引用 返回版面 TOP

13楼:

up

金升阳电源技术交流会，时尚数码礼品等你拿!  
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购!



帖子: 39

积分: 84



[new1988](#) 发表于 2009-8-19 19:14 | 只看该作者 回复 引用 返回版面 TOP

14楼:

MARK

金升阳电源技术交流会，时尚数码礼品等你拿!  
[Digi-Key](#)、[Future](#)、[Arrow](#)、[Mouser](#)、[RS Components](#)、[OnlineConpunents](#)实时库存在线查询订购!

[返回列表](#)



客户服务热线: **010-59705655** 21ic不良信息举报(24小时): **013681498700** 举报邮箱: [info@21ic.com](mailto:info@21ic.com)

21IC中国电子网 2000-2009 爱奇艺 (北京) 信息科技有限公司版权所有







