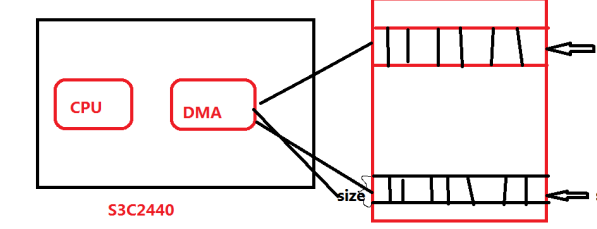
DMA的引入：

以下时2440的一个大致的原理图：右边是内存



假如有如下的代码片段从内存的一个地址复制到另外一个地址:

char \*src = AAA;

char \*dest = BBB;

int i;

for(i=0;i<size;i++) dest[i] = src[i];

我们知道，cpu同一时间只能做一件事情，如果cpu执行右边的代码，那岂不是不能做其他事？

如上图，DMA是一个独立的模块存在于处理器，DMA是不通过CPU而是直接访问内存,使用DMA，可以释放CPU的"压力"，使得它不会一直在做一件事,使用了DMA也能达到直接使用CPU的效果。

要将源，目的， 大小，这个三要素告诉ＤＭＡ,然后启动。数据拷贝完成后会产生中断。

同样要设置DMA参数：

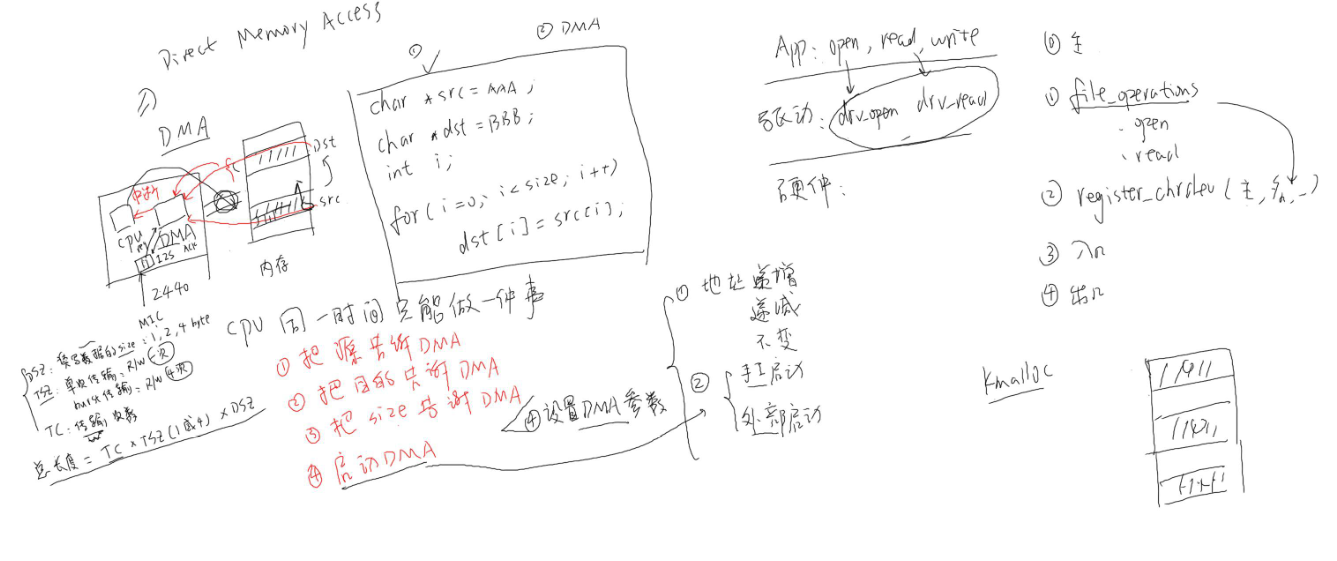
1.拷贝的时候是地址递加或者递减

2.DMA的启动分内部启动和外部启动：

内部启动：设置某个寄存器。来启动

外部启动：例如音频接口，麦克风收到数据后放入IIS固件的缓冲区fifo,完成后，发出一个DMA请求，然后将数据从IIS拷贝到内存中。

编写DMA驱动程序：



具体可以参考我们的代码。

要注意：

1. 不能用kmalloc因为它虽然虚拟地址是连续的但是对应的物理地址不连续，有可能是分开的一块的物理内存这样的话，DMA无法处理。
2. 我们采用手工启动ＤＭＡ

我们DMA支持四个通道。它的源有分在系统总线，或者物理总线模块。

也就是说可以将外设的数据传输到内存，或者在内存之间传输。

DMA　REQUEST　SOURCE　就是他的触发源

我们有０－６个触发源。

３．我们有两种模式：

Ｄｅｍａｎｄ　ｍｏｄｅ　：

请求Ｄｍａ后，（低电平）如果是低电平一直不停的传输数据

Ｈａａｎｄｓｈａｋｅ　ｍｏｄｅ：

请求Ｄｍａ后（低电平），传输数据一次，然后将请求拉高，再拉低.

驱动代码和测试代码写完成后：测试

加载驱动：insmod …

后台执行测试程序：./s3c\_dma\_test nodma &

会一直答打印： 即使我们输入ls 命令不能立马执行因为此刻拷贝在占用cpu资源

./s3c\_dma\_test dma &

输入命令后会执行。因为DMA不会占用ｃｐｕ资源资源。

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

# MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

# MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

MEM\_CPY\_DMA OK

可以尝试修改 寄存器的值来观察现象。

内核已经提供了DMA操作的处理函数：

我们以声卡的驱动程序分析：s3c2410\_uda1341.c

从init 分析：

其实跟我们自己操作差不多。