romStart()函数执行过程研究

SWD 聂勇

版 本 历 史

版本/状态	责任人	起止日期	备注
V1.0/正式	聂勇	26Oct2010	romStart()函数执行过程研究
V1.1/正式	聂勇	12Nov2010	添加解压缩过程的相关分析，主要是被解压缩代码是位于flash 中呢，还是位于 sdram 中问题的分析。

目　录

1. romStart()函数分析

romStart()函数只完成两个事情，第一是将 flash 中的 bootrom 代码拷贝到 RAM 中，第二是解压缩 bootrom 中被压缩的代码段到 RAM 中。至于拷贝，解压缩的起始地址，长度，目的地址等，都是可以灵活配置的，在下面将有进一步的分析。

romStart()函数代码量其实很少，原来提供的代码为了适应不同的开发平台，添加了大量的宏。下面就是将其中的宏去掉之后，得到的最后的 romStart()函数。

bootInit.c

```
……
void romStart ( FAST int startType)        /* start type */
{
    volatile FUNCPTR absEntry;
    volatile FUNCPTR absUncompress;

    ((FUNCPTR)ROM_OFFSET(copyLongs))(ROM_TEXT_ADRS,
    (UINT)K0_TO_K1(romInit),((UINT)edata - (UINT)romInit) / sizeof (long));

    absUncompress = (FUNCPTR) UNCMP_RTN;
    if ((absUncompress) ((UCHAR *)ROM_OFFSET(binArrayStart),
    (UCHAR *)K0_TO_K1(RAM_DST_ADRS),
    (int)((UINT)&binArrayEnd - (UINT)binArrayStart)) != OK)
    {
        return;          /* if we return then ROM's will halt */
    }

    /* jump to VxWorks entry point (after uncompressing) */
    absEntry = (FUNCPTR)RAM_DST_ADRS;          /* compressedEntry () */
    (absEntry) (startType);
}    ……
```

romStat()函数中一共有三个函数调用。

黄色标记：将 flash 中的 bootrom 代码拷贝到 RAM_LOW_ADRS 中，copyLongs()函数为在同一文件 bootInit.c 中实现。

绿色标记：将拷贝到 RAM 中的 bootrom 被压缩的部分解压缩到 RAM_HIGH_ADRS 处。UMCMP_RTN 为 inflate()函数，在文件开头有定义。inflate()函数是 zip 库（或者说是 vxWorks 库）中实现。

青色部分：跳转到 RAM_HIGH_ADRS 处开始执行。RAM_DST_ADRS 就是 RAM_HIGH_ADRS 的地址，在文件开头有定义。

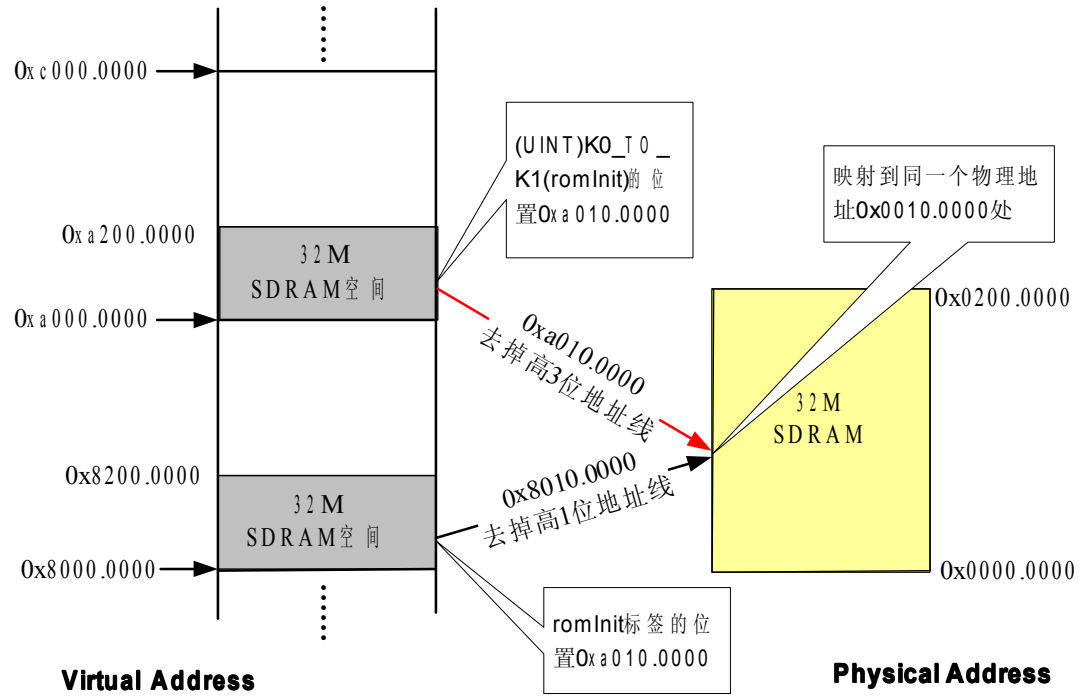
在拷贝函数和解压缩函数的输入参数中，使用到了很多关于地址的宏定义，以及一些编译链接时的特殊符号（参考《编译过程研究_MIPS_NY.docx》附件）。

2. 拷贝过程详解

2.1 输入参数解释

ROM_TEXT_ADRS：被拷贝数据源的地址，也就是要拷贝的数据在 flash 中的地址，在 config.h 中定义，为 0xbfc0.0000。

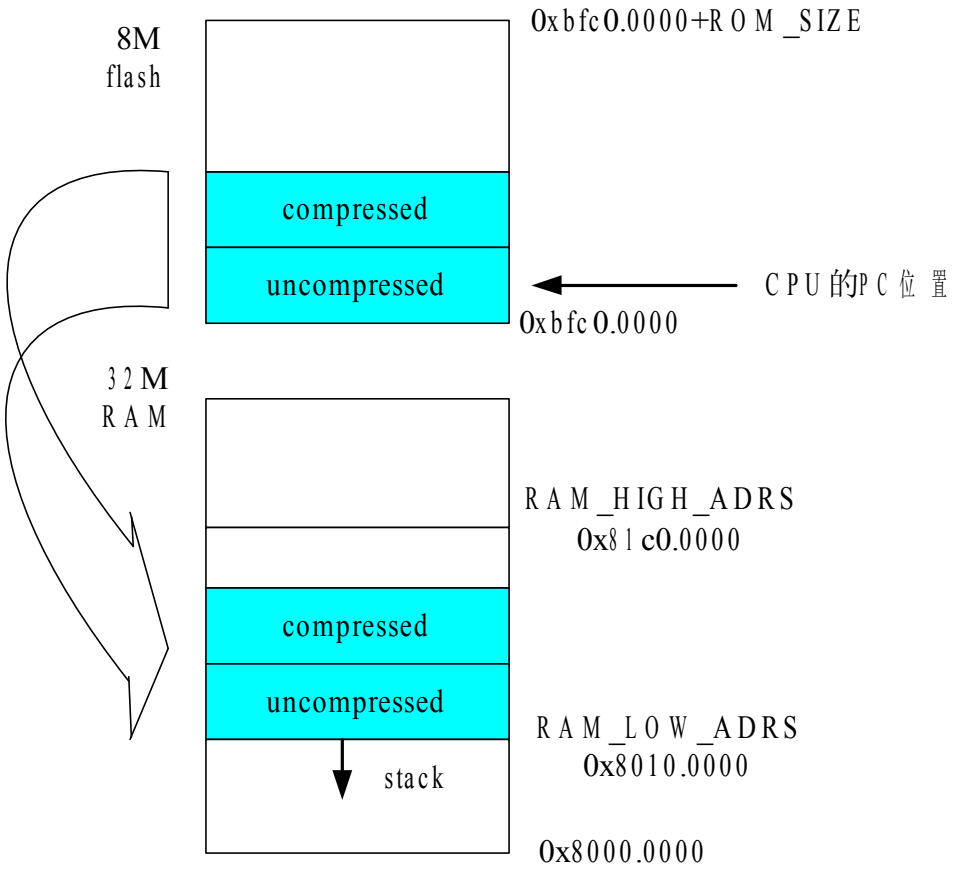
(UINT)K0_TO_K1(romInit)：被拷贝数据目的地址，也就是被拷贝的数据将要被放置的地址。在《》文档中分析过，romInit 这个符号的地址为 0x8010.0000。这应该也就是代码要放置到 RAM 中的位置，那么这里为什么要从 kseg0 映射到 kseg1，当然，这两个对应的物理地址都是指向我们的 RAM 中。这里的原因是，我们的 cache 还没有初始化，这里并不能够使用需要初始化的 kseg0 的地址。只好将 kseg0 的 0x8010.0000 映射到 kseg1 中的 0xa010.0000 处。当然，对于我们物理的 RAM 来说，地址并没有变化。



((UINT)edata - (UINT)romInit) / sizeof (long)：被拷贝的数据的长度。同上，romInit 这个符号的地址为 0x8010.0000，而 edata 这个符号是在链接的时候由编译器确定的符号。代表的意思是可执行文件 bootrom 代码部分（专业术语为文本段+数据段）结束的位置。参考《编译过程研究_MIPS_NY.docx》的特殊符号一节。

2.2 拷贝过程图解

整个拷贝过程具体是如何呢？下面使用图形象的说明了这个过程。



对上图，有几个需要解释的地方。

第一， 上面的地址都是虚拟地址，都是实际 flash 和 ram 所在的物理地址转化成的虚拟地址。

第二， 对于虚拟地址，8M 的 flash 的地址是位于 kseg1 中，32M 的 ram 的地址是位于 kseg0 中。其实，此时使用到的 ram 的地址，应该还是在 kseg1 中的，也就是说，应该写成从 0xa000.0000 开始更合理。这也是(UINT)K0_TO_K1(romInit)的原因。

第三， 因为 flash 空间的重叠块，8M 的 flash 空间应该考虑从 0xbc00.0000 开始。否则，0xbfc0.0000+ROM_SIZE 可定超过 kseg1 的上界 0xc000.0000。

第四， 此时拷贝函数还是 flash 执行的，所以 CPU 的 PC 还是指向的 flash 空间中。
但是因为是一个 C 函数，所以在 ram 空间的最下面 1M 空间设置了堆栈。

3. 解压缩过程详解

解压缩过程还存在相关疑惑：

解压缩函数为何能够在 kseg0 段运行呢？此时 cache 不是还没有初始化吗？

为什么解压缩段开始的地址还是在 flash 中呢？这样不很是影响速度吗？

3.1 输入参数解释

3.2 解压缩过程图解