# 详解LMA（装载内存地址）与VMA（虚拟内存地址）

工作和技术 crifan 8年前 (2009-10-04) 202浏览 0评论

## 详解LMA（装载内存地址）与VMA（虚拟内存地址）

version: 20091004
author: green-waste@163.com

关于LMA和VMA，这个问题，有点点小复杂，不过，此处，我会把我的理解，尽量通过通俗的方式解释出来，以方便理解。当然，鄙人水平有限，难免有错，希望各位批评指正。

一般提及LMA和VMA，多数情况都是和ld，链接器相关的。
在了解这两个名词的详细含义之前，有些基本知识和前提要说一下：

**［基础知识］**

1。从你写的源代码到执行你的程序，一般经历了这几个过程：
源代码编辑 -> 编译 -> 链接 -> 装载 -> 执行
2。编译，简单说就是用编译工具，将你的源码，变成可以执行的二进制代码，也叫做目标文件，当然只是对应某一种硬件平台，比如此处我用的是Intel的X86系列的CPU，编译出来的，就是针对X86的二进制代码。
3。链接就是，将多个目标文件合并为一个目标文件，称作可执行文件。
4。每个目标文件都包含一连串的section，最常见，最基础的至少有：

.text，代码段，就是CPU要运行的指令代码；
.data，数据段，程序中包含的一些数据，放在这个段里；
.bss，未初始化段，记录了程序里有哪些未初始化的变量，就相当于只记录对应的名字，留着程序运行前去初始化为0，所以，此处并不占用具体空间。打个比方就是，只记录人名，没有人站在这里占地方，而对应的.text和.data段，都是既有人名（函数或者变量名）,又占对应的地方（包含具体空间记录到底是什么指令代码和数据的数值是多少）。
5。section一般可以分为loadable与allocatable.
通俗点说就是：
loadable，可加载，就是，原先目标文件里面包含对应的代码或数据，所以，装载器要把这些内容，load到对应的地址，以便程序可以运行；
而allocatable，可分配的，最简单理解就是上面提到的.bss段，那里记录了人名，到时候，你要给这些人名分配空间给你站的地方，对应着也就是变量所要占用的具体内存空间了。
其他还有既不是loadbale的，也不是allocatable的，比如只存储debug信息的段，此处不多解释。

**[ 前提 ]**

程序已经编译好了，有了一个可执行文件，也叫目标文件，二进制文件，才会有后面的把程序装载，运行的事情。

看完了基础知识和前提，再说我们此处的主题，才能更加清楚是咋回事：
对于目标文件中的loadable或allocatable的section，其都有两个地址：VMA 和 LMA 。
知道了其来由，再看具体解释：

**[ LMA 详解 ]**

LMA的英文原版解释：
LMA（Load Memory Address）： the address at which the section will be loaded.
什么是Load Memory Address，内存装载地址呢? 此处，单单从名字上，我们就可以看出几层意思：

**1。load，装载**

为何要装载呢？ 因为，如果想要使你的程序（即经历过，由你的源码，通过编译器的编译，链接器的链接，形成的那个可执行文件），能在内存里面运行，那么肯定涉及到一点，就是，有人，把你的这个程序，
，从此处常见的存储器硬盘里面，搬到内存里面去了，然后才有可能运行。而这里的装载，就是对应这个意思。就是把程序，从硬盘里面，装载Load，到内存里面去了。
对应地，放到内存哪里去了呢？就是LMA，Load Memory Address，就是把你的程序中的对应的内容，详细点说就是，把其中的.text代码段，.data数据段等内容，搬到，也就是copy拷贝到，内存的LMA地址处了。

**2。Memory，内存**

上面已经解释了，这里再多说几句。
程序运行的本质，就是CPU读取到指令，然后执行。这里就涉及到，如果想要你的程序运行，首先，你应该把对应的指令，放到合适的地方，CPU 才能读到，才能执行。

此处合适的地方，有人想到，直接放到硬盘这里，CPU过来读取，然后执行不就可以了吗，还不用这么麻烦地将（指令）代码搬来搬去的，多省事。但是实际上，系统就是这么“笨”地搬来搬去，原因在于，从硬盘上直接读取指令，速度比直接从内存，一般PC 上是各种类型的RAM，比如DDR，此处统称为Memory/内存，
要慢很多倍，所以，系统才会不嫌弃麻烦，把代码拷贝到内存里面去，然后从内存里面读取指令，然后执行，这样效率会高很多。
所以，此处简单说就是，为了总体效率，对于普通系统，比如PC，程序的执行都是在Memory，内存里面执行的。

因此，用一句话总结就是：
代码被装载到内存的某个地方，那个地方的地址，就是LMA 。

**［VMA 详解］**

英文解释：
VMA（Virtual Memory Address）：the address the section will have when the output file is run;
那啥是虚拟内存地址呢？简单说就是，你程序运行时候的所对应的地址。
此处所谓的虚拟，一般来说，指的是启用了MMU之后，才有了虚拟地址和实地址。
此处，我们可以简单的理解为，就是内存的实际地址即可。
程序运行前，要把程序的内容，拷贝到对应的内存地址处，然后才能运行的。
因此，一句话总结就是：
代码要运行的时候，此时对应的地址，就是VMA。

**［理解此句：在多数情况下，LMA和VMA是相等的］**

这句话，说白了，可以（武断地）这么理解：
如果是普通PC电脑，也就是上面说的，大多数情况下，那么LMA和VMA是一样的，也就是，程序被加载到内存的什么地方，也就在什么地方运行。
如果是嵌入式系统，也就是相对的“少数情况”，LMA和VMA不一样。而其中最常见的一种情况就是，
程序被放到ROM中，比如设置为只读的Nor Flash中，也就是LMA的地址是Nor Flash的地址，此如随便举例为0x10000000，而程序要运行时候的地址是内存地址，比如0x30000000,也就是VMA 是0x30000000，这时候，就要我们自己保证，在程序运行之前，把自己的程序，从LMA = 0x10000000拷贝到VMA = 0x3000000处，然后程序才可以正常运行。

有人会问，反正对于ROM来说，CPU 也是可以直接从ROM里面读取代码，然后运行的。为何还要前面提到的，弄个LMA 和VMA不同，搬来搬去的呢？因为ROM，顾名思义，是只读的，只能读取，不能写入的。
而程序中的代码段，由于只是被读取，不涉及到修改写入，是没有问题的。但是对于数据段和s位初始化段来说，里面的所有的程序的变量，多数都是在运行的时候，不仅要读取，而且要

修改成新的值，然后写入新的值的，所以，如果还是放到ROM里面，就没法修改写入了。
而且，另一个原因是，CPU从ROM，比如常见的Nor Flash中读取代码的速度，要远远小于从RAM，比如常见的SDRAM，中读取的速度，所以，才会牵扯到将代码烧写到ROM里面，然后代码的最开始，将此部分程序reaload，重载，也就是从此处的ROM的地址，即LMA，重新拷贝到SDRAM中去，也就是VMA的地方，然后从那里运行。

## [ 后记 ]

**关于LMA 和 VMA：**

Linker，链接器的作用：

1。将LMA写到（可执行的）二进制文件里面去

2。解析符号。即，把不同的符号，根据符号表中的信息，转换成对应的地址。此处只涉及VMA，即程序运行时候的地址。

**Loader，装载器的作用：**

1。从二进制文件中读出对应的段的信息，比如text，data，bss等段的信息，
将内容拷贝到对应的LMA的地址处。此谓，装载（对应内容）到装载地址（LMA）。

2。如果发现VMA!=LMA, 即 程序运行时候的地址，和刚刚把程序内容拷贝到的地址LMA，两者不一样，
那么就要把对应的内容，此处主要是data，数据段的内容，从刚刚装载到的位置，LMA处，拷贝到VMA处，
这样，程序运行的时候，才能够在执行的时候，找到对应的VMA处的变量，才能找到对应的值，程序才能正常运行。

## [ 引用文章 ]

1。谁能解释下VMA和LMA及其在链接时的作用
http://bbs.sjtu.edu.cn/bbscon,board,C,file,M.1235442042.A.html

2。VMA & LMA
http://hi.baidu.com/woaimuxiaoyizhong/blog/item/083c54dd84a862e776c638a8.html

3。VMA vs LMA?
http://www.embeddedrelated.com/usenet/embedded/show/77071-1.php

4。BOOT阅读笔记
http://blog.chinaunix.net/u2/63543/showart_500643.html

5。Output section LMA
http://www.delorie.com/gnu/docs/binutils/ld_33.html

上一篇 swap()函数的4种实现

不开心的进来看！ 笑死你！ 下一篇

## 与本文相关的文章

【已解决】什么是短号码Short Code

【调研】国外消息通知推送服务

【调研】国外 美国 发送短信 服务

[ 已解决 ] lhs rhs是啥意思

[ 整理 ] 企业移动协同办公，saas，slack

【整理】RS232 RTS/CTS的流控制的具体过程/机制

【整理】好的素材网站

[ 整理 ] EABI和OABI

【整理】TCP/IP vs PDP

【整理】E-PLAN==EPlan

走别人没走过的路，让别人有路可走 –探讨国内IT领域内知识体系与传播 – Crifan Li in TechCamp

【整理】界面原型设计工具

## 发表我的评论

写点什么...

表情

提交评论

## 网友最新评论 (2)

写得很不错，赞一个

ghostvip 8年前 (2010-04-28) 回复