vxWorks 启动过程研究

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	11Oct2010	BCM 平台下 vxWorks 启动过程研究
V1.1/正式	聂勇	12Nov2010	以函数的依次执行顺序，建立单独文档《romInit()函数.docx》、《romStart()函数.docx》、《usrInit()函数.docx》、《usrRoot()函数.docx》。本文档只作为启动过程的整体说明，相关准备知识介绍，为以上四个步骤的研究做准备。

目　录

1. 说明 3
2. 地址 3
2.1 VIRTUAL ADDRESS 3
2.2 PHYSICAL ADDRESS 5
2.3 VA 和 PA 的映射关系 6
2.4 其它地址 8
3. 代码执行位置 8
3.1 ROM 上执行部分 9
3.2 RAM 上执行部分 9

1. 说明

本文档为 MIPS 芯片下 vxWorks 的启动过程研究结果。本研究课题分为两部分，第一是 Boot ROM 的启动过程，第二是 vxWorks 的启动过程。现在的研究主要是集中在 Boot ROM 的启动过程上。对 vxWorks 的启动研究尚未进行。

研究时使用平台为 bcm5836，开发方案为 broadcom 公司提供的 bcm_harrier 方案。

本文档只是一个索引和准备，具体的启动过程请参考《romInit()函数.docx》、《romStart()函数.docx》、《usrInit()函数.docx》、《usrRoot()函数.docx》等文件。你可以先行阅读本文档中的知识点，这有助于你对具体启动过程的了解；你也可以在阅读过程中遇到问题了，再反过来参考本文的内容。

2. 地址

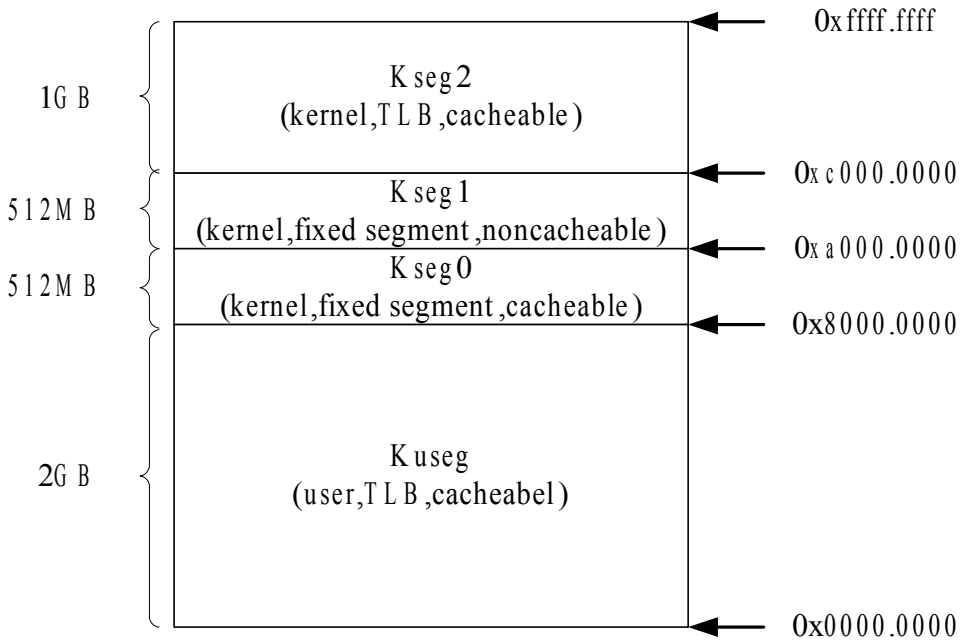
对于底层软件工程师来说，很多时候都在和地址打交道。由于以前学习不够系统，技术资料的不明了，现代 SOC 芯片的模块集成导致的复杂，地址的概念可能比较混淆。下面以 bcm5836 芯片为例，说明其中需要使用到的各类地址的关系。

2.1 Virtual Address

在谈及某一体系架构的 CPU 时，通常会涉及到地址空间（address space）的概念。此时，需要了解的是该处理器核的地址线宽度，例如 16 根地址线，32 根地址线，或者 64 根。此时，可能会给出一个内存映射图（memory map）。

此时，给出的地址，是虚拟地址（virtual address），因为只里的地址，表示的是 CPU 核在处理时使用的寻址的地址。该地址还会被经过一系列的转化（例如，MMU 转化，丢弃最高位 N 根地址线等操作），才会变为实际对其他外围器件（这里的外围是指除 CPU 核之外）进行寻址。

例如，对于 32 位 MIPS 架构 CPU，它的基本地址空间如下图：



必须强调，对于软件开发，我们使用的地址都是虚拟地址。无论是在汇编代码中，还是在 C 语言代码中。如果涉及到一些物理地址的操作，我们也要将之转化为虚拟地址。例如：

romInit.s Line 1038

```
……
/*
 * Memory segments (32bit kernel mode addresses)
 */
#define KUSEG           0x00000000
#define KSEG0           0x80000000
#define KSEG1           0xa0000000
#define KSEG2           0xc0000000
#define KSEG3           0xe0000000
/*
 * Map an address to a certain kernel segment
 */
#define _KSEG0ADDR(a)       (((a) & 0x1fffffff) | KSEG0)
#define _KSEG1ADDR(a)       (((a) & 0x1fffffff) | KSEG1)
#define _KSEG2ADDR(a)       (((a) & 0x1fffffff) | KSEG2)
#define _KSEG3ADDR(a)       (((a) & 0x1fffffff) | KSEG3)
……
```

“kernel segment”就是我们上面的 MIPS 的虚拟地址空间，上面代码就是将物理地址 a 转化成为对应段的虚拟地址（参考 VA 和 PA 的映射关系一节）。

romInit.s Line 1065

```
……
board_draminit:
    SYSLED(LED_RED)
    /* Save return address */
    move    t6,ra           # change v0 to t6
    /* Scan for an SDRAM controller (a0) */
    li    a0, _KSEG1ADDR(SB_ENUM_BASE)
……
```

在对 RAM 的初始化的时候，就用到了物理地址到虚拟地址的转化。SB_ENUM_BASE 定义在 Sbconfig.h 文件中。

Sbconfig.h Line 35

```
……
#define SB_ENUM_BASE      0x18000000   /* Enumeration space base */
……
```

有关 0x18000000 物理地址的相关知识，参考下一节中的 SOC 的内存映射关系表，可以看到，这其实是 SOC 片上模块的起始地址，在下一节中将会讲到，这就是一个物理地址。

2.2 Physical Address

在很多 SOC 上，都有很多模块的集成，这个时候，就涉及到各个模块的地址。通俗讲，就是 CPU 核是如何找到这些模块的呢，是如何配置这些模块相关寄存器，如何向这些模块存取数据？

这个时候，就涉及到 SOC 片上资源地址空间的划分问题。例如，对于芯片 bcm5836 来说，地址映射关系如下表：

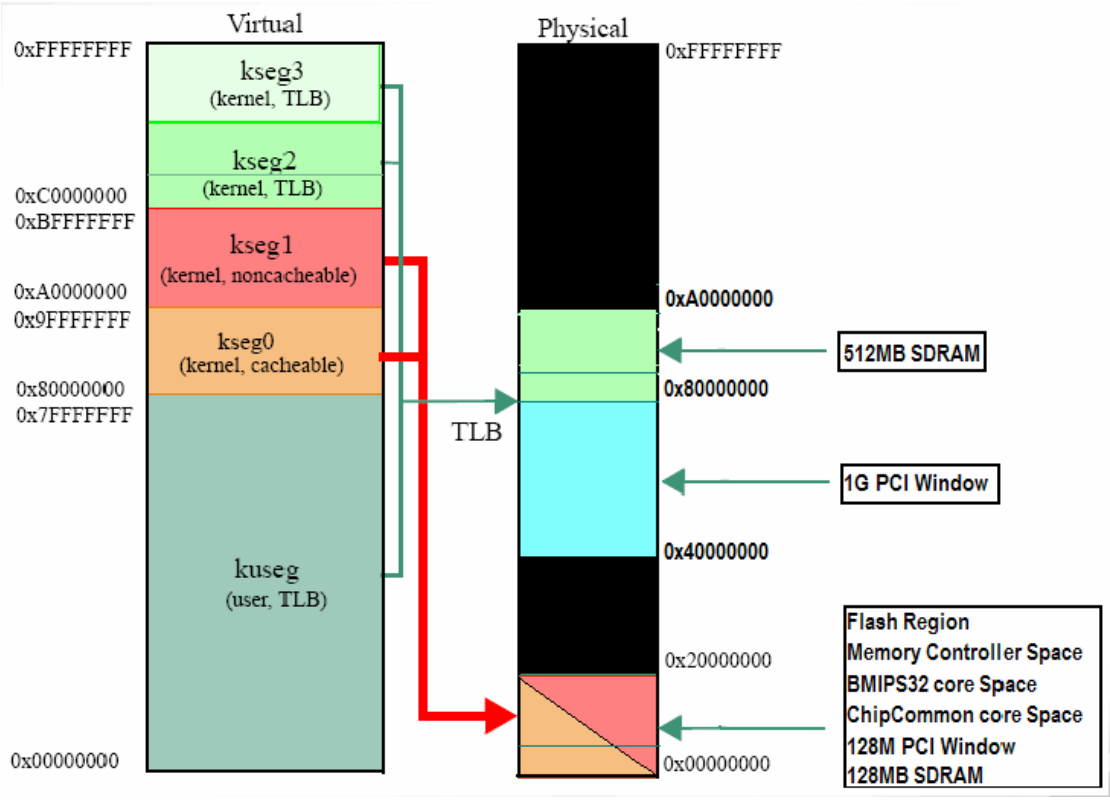
Address	Description
0x00000000 ~ 0x07FFFFFF	128 MB SDRAM, non-swapped region 1
0x08000000 ~ 0x0FFFFFFF	128 MB PCI window
0x10000000 ~ 0x17FFFFFF	128 MB SDRAM, swapped
0x18000000 ~ 0x18000FFF	Chipcommon core
0x18001000 ~ 0x18001FFF	Ethernet MAC0 core
0x18002000 ~ 0x18002FFF	Ethernet MAC1 core
0x18003000 ~ 0x18003FFF	USB core
0x18004000 ~ 0x18004FFF	PCI core
0x18005000 ~ 0x18005FFF	MIPS processor core
0x18006000 ~ 0x18006FFF	Codec core
0x18007000 ~ 0x18007FFF	IPSec Core
0x18008000 ~ 0x18008FFF	Memory controller core
0x18009000 ~ 0x18009FFF	Reserved
0x1A000000 ~ 0x1BFFFFFF	External interface address map (in Chipcommon)
0x1C000000 ~ 0x1DFFFFFF	Flash region 2, totally 32 MB
0x1FC00000 ~ 0x1FFFFFFF	Flash region 1, subset of flash region 2
0x40000000 ~ 0x7FFFFFFF	1 GB PCI window, client mode PCI memory space
0x80000000 ~ 0x9FFFFFFF	512 MB SDRAM, non-swapped region 2, non-swapped region 1 shadowed onto this

每一个地址范围对应什么资源都是由上表确定的。在这里的地址，就是物理地址。

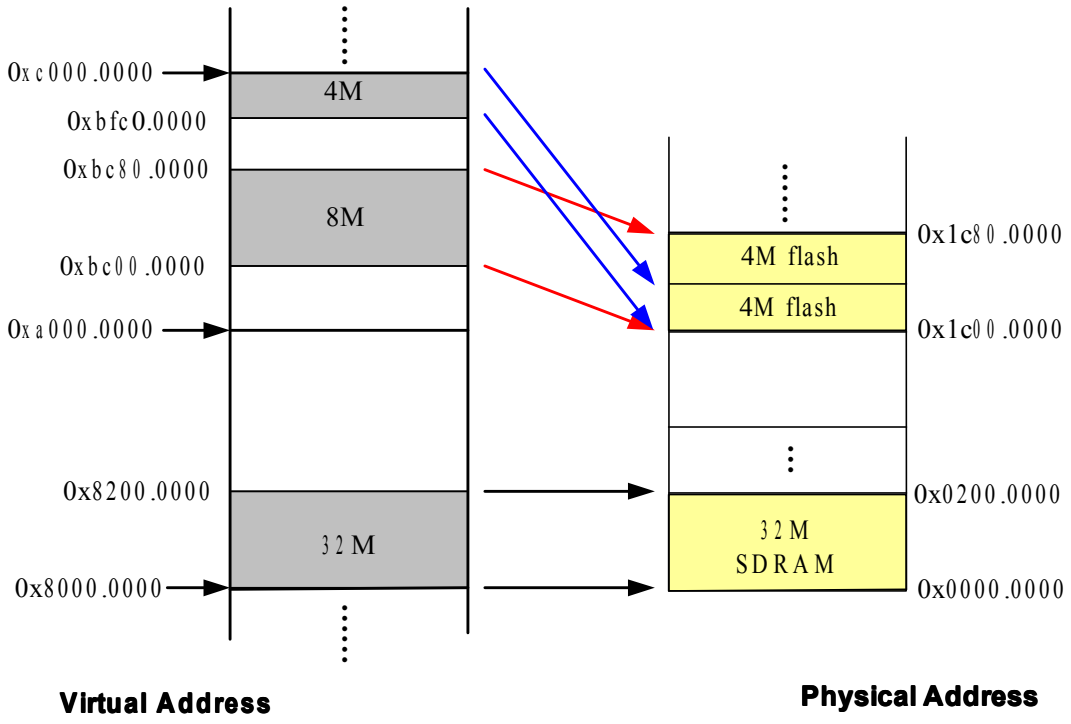
2.3 VA 和 PA 的映射关系

首先，我们看下图所示。这是 bcm5836 这片 SOC 芯片上，虚拟地址到物理地址的映射关系图。

左边的虚拟地址分布，就是 MIPS CPU 架构的基本地址空间的情况，对于某一固定的 CPU 架构，一般是不会变化的。参考《See MIPS Run》P47。右边的物理地址分布，主要是由 SOC 芯片来决定的。对于采用同一 MIPS CPU 架构的不同 SOC，可能不会相同。具体的分布，需要参考的是 SOC 的设计说明书。



下面这个图，介绍的是 harrier 方案中，有关 VA 和 PA 的映射关系图。需要对照上面两节的图表来分析。



上图中，需要说明的一点就是 FLASH 空间的映射关系。可以看到，从 0xbfc0.0000 到 0xc000.0000 的 4M 空间和从 0xbc00.0000 到 0xbc80,0000 的 8M 空间是由重合的。其实，我们可以翻过去看到，在 2.2 节表中，有这么一段：

0x1C000000 ~ 0x1DFFFFFF	Flash region 2, totally 32 MB
0x1FC00000 ~ 0x1FFFFFFF	Flash region 1, subset of flash region 2

那上面这两行使什么意思呢？其实，就是说，物理地址从 0x1fc0.0000 到 0x1fff.ffff 的 4M 空间和从 0x1c00.0000 开始的 32M 空间，被 flash 控制模块映射到实际的 flash 器件的时候，是属于同一位置。其实，这里是 MIPS CPU 的虚拟地址转化成物理地址（在这里就是 kseg1 段去掉高位 3 根地址线）之后，flash 控制模块又将物理地址映射到实际器件的时候，将某一部分物理地址进行了重合。

在开发过程中，常说，CPU 上电之后的开始运行的地址是 0xbfc0.0000（又叫做系统启动向量），所以我们 bootrom 的第一行代码就放在虚拟地址 0xbfc0.0000 所对应的物理地址处（对于软件开发，可能无需了解该地址。因为芯片设计的原因，可能有多个虚拟地址对应此处的同一个物理地址）。

如上所收，可能有多个虚拟地址对应此处的同一个物理地址。们在使用 windriver bench 进行 bootrom 烧写 flash 的时候，可以看到需要选择的 flash 的基地址为 0xbc00.0000，结束地址为 0xbc7f.ffff（一共 8M）。其实，这个虚拟地址 0xbc00.0000 也是对应 flash 器件的起始地址。

2.4 其它地址

在开发过程中，还会遇到其它的一些关于地址的概念，下面对此一一做出解释

- logical address
- running address
- compiling address

3. 代码执行位置

Boot ROM 是系统上电之后首先运行的代码段，是整个嵌入式系统的 boot loader。按照代码执行的位置，分成 ROM 上执行部分和 RAM 上执行部分。

3.1 ROM 上执行部分

ROM 上执行也就是 flash 执行。所谓的 flash 执行，就是说 CPU 执行的每一条指令，都是从 flash 中获取。最明显的特征就是 CPU 的程序计数器(PC)是指向的 flash 的地址空间。

根据执行的顺序和功能，ROM 上执行可以换分成两部分：romInit()函数和 romStart()函数。

注意：其实 romStart()函数有解压缩部分代码并不是在 flash 中执行，而是在 ram 中执行，然后再跳转到 flash 中执行完 romStart()的一部分，直到结束。

具体参考：《romInit()函数》《romStart()函数》。

3.2 RAM 上执行部分

RAM 上执行也就是执行的代码位于内存中，在我们这里，就是位于 sdram 中。CPU 是从内存中取指令执行，那么一定有一个操作，需要将可执行代码加载到内存中。那么，是从哪儿被加载到内存中去的呢？

在 bootrom 启动过程的拷贝、解压缩过程，就是将代码从 flash 加载到内存中。

在 bootrom 的命令行加载 vxWorks 的过程，就是将代码从 host 机器通过网络加载到内存中。

总之，在充分理解了地址，编译链接的相关知识之后，你就可以随心所欲的指定代码的运行位置，起始位置等等。