试验平台：芯晟 CSM1201

老内核版本：2.6.12.5

第一步，先升级芯晟内核到 Android1.0 的内核版本－2.6.25

第二步，移植 Android

先总结一下第一步的移植过程以及遇到问题的解决办法。

首先是拿到原版 2.6.12.5 内核，与芯晟内核作比较，将芯晟修改和加入的部分，先做简单的拷贝到 2.6.25 内核，这里需要注意有几个目录或目录下下的文件是编译过程中生成的，不要预先拷贝，如 script 下的一些.cmd，arch/asm 目录，include/asm 等。芯晟在 12.5 内核中主要修改了 arch 和 driver 目录，重点关注相应的 Kbuild 和 Makefile。

然后同时开两个 shell 窗口，分别进入 25 内核和芯晟内核做 make menuconfig 操作，对比芯晟内核的配置来作 25 内核的配置，其中有些老内核的配置选项在新内核中已经不再支持或者位置变了，需要灵活处理。另外如果上一步的 Kbuild 修改是正确的，那么芯晟加入的内核配置选项在两版内核的 menuconfig 中应该是基本相同的。

内核配置完后就开始编译。遇到的问题记录如下：

1.entry_armv.S 汇编出错，提示的错误是 bad command 'get_irqnr_preamble r5, lr'，google 之发现只要把这条指令引掉即可；

2.还是 emtry_armv.S,出现错误 internal relocation (type:OFFSET_IMM) not fixed up，google、百度了三天，有的说是编译器问题，有的说是 constant.h 问题，我还换了新版编译器试了仍然不行。就在几乎绝望的时候，最后发现是有个该汇编文件中#include <asm/arch/entry-macro.S>，entr_macro.S 中有个宏 VA_VIC_BASE 没定义，#include <asm/hardware.h>之后解决了。

3.arch/arm/kernel/head.S，错误提示：

```
arch/arm/kernel/head.S:267: Error: missing ')'
arch/arm/kernel/head.S:267: Error: garbage following instruction --
`ldr r4,=(((0xc0000000UL)+0x00008000)-0x4000)'
```

是我的 include/asm-arm/orion2/memory.h 中定义了#define PHYS_OFFSET (0x00000000UL),但是 UL 已经被 include/asm-arm/memory.h 定义成#define UL(x) (x##UL)

因而导致编译器出现错误。将 include/asm-arm/orion2/memory.h 中#define PHYS_OFFSET (0x00000000UL)改为#define PHYS_OFFSET UL(0x00000000)后解决。

4.arch\arm\kernel\entry-common.S,bad command
' arch_ret_to_user r1, lr',原因同 1，引掉即可

接下来就是芯晟加入的驱动在新版内核上的修改了，有一些宏定义、数据结构以及函数已经变化了，做了不少修改。修改了大概两天时间，最后终于编译通过，生成了 Uimage。

以下是内核运行过程中出现的错误及解决办法：

1.在 starting kernel...的位置停下了

解决办法：重新配置内核，在 kernel hacking 中加上 debug 和 lowlevel debug 功能。重新编译 uImage，重新内核启动发现是 machineID 不对，于是修改 arch/arm/tools/mach-types 文件，将芯晟的 machineID 改为 719。

重启 kernel，停在 done, booting the kernel.

打开 printk.c 中的 VIRGO_EARLY_PRINTK 宏，出现打印：

```
<4>[    0.000000] Linux version 2.6.25 (root@ubuntu) (gcc version
4.4.1 (Sourcery G++ Lite 2010q1-202) ) #20 Tue Aug 17
20:29:47 PDT 2010
<4>[    0.000000] start_kernel............13
<4>[    0.000000] CPU: ARM926EJ-S [41069265] revision 5
(ARMv5TEJ), cr=00053177
<4>[    0.000000] Machine: ORION
<4>[    0.000000] Memory policy: ECC disabled, Data cache
writeback
```

```
<7>[    0.000000] On node 0 totalpages: 25600
<7>[    0.000000]   DMA zone: 200 pages used for memmap
<7>[    0.000000]   DMA zone: 0 pages reserved
<7>[    0.000000]   DMA zone: 25400 pages, LIFO batch:7
<7>[    0.000000]   Normal zone: 0 pages used for memmap
<7>[    0.000000]   Movable zone: 0 pages used for memmap
<3>[    0.000000] MM: CPU does not support supersection mapping
for 0x10140000000 at 0xf1140000
<3>[    0.000000] MM: CPU does not support supersection mapping
for 0x101f1000000 at 0xf11f1000
<3>[    0.000000] MM: CPU does not support supersection mapping
for 0x101e0000000 at 0xf11e0000
<3>[    0.000000] MM: CPU does not support supersection mapping
for 0x35000000000 at 0xf8000000
```

问题出在哪呢？

原来是 16 版本内核之后，map_desc 结构发生了变化，

```
struct map_desc {------------------------------2.6.16
unsigned long virtual;---------------虚地址
unsigned long pfn;------------------Page Frame Number 页帧号
unsigned long length;
unsigned int type;
};

struct map_desc {------------------------------2.6.9
unsigned long virtual;
unsigned long physical;------------物理地址
unsigned long length;
unsigned int type;
};
```

根据以上变化，修改后的物理内存—>虚拟地址的映射关系如下：

```
static struct map_desc virgo_io_desc[6] __initdata = {
{
 .virtual = VA_VIC_BASE ,
 .pfn = __phys_to_pfn(PA_VIC_BASE),//PA_VIC_BASE,
 .length = SZ_64K,
 .type = MT_DEVICE
},
{
 .virtual = VA_UARTS_BASE,
 .pfn = __phys_to_pfn(PA_UARTS_BASE),//PA_UARTS_BASE,
 .length = SZ_16K,
 .type = MT_DEVICE
},
{
 .virtual = VA_TIMERS_BASE,
 .pfn = __phys_to_pfn(PA_TIMERS_BASE),//PA_TIMERS_BASE,
 .length = SZ_64K,
 .type = MT_DEVICE
},
{
 .virtual = VA_ETH_BASE,
 .pfn = __phys_to_pfn(PA_ETH_BASE),//PA_ETH_BASE ,
 .length = SZ_64K,
 .type = MT_DEVICE
},
{
 .virtual = VA_ATA_BASE,
 .pfn = __phys_to_pfn(PA_ATA_BASE),//PA_ATA_BASE ,
 .length = SZ_16K,
 .type = MT_DEVICE
},
{
 .virtual = VA_PCMCIA_BASE,
 .pfn = __phys_to_pfn(PA_PCMCIA_BASE),//PA_PCMCIA_BASE,
 .length = SZ_16K,
 .type = MT_DEVICE
}
}; // Data from Virgo MemoryMap
```

修改后停在 calibrate_delay，说明 timer 有问题，jiffs 没好用。

仔细查了一下 25 内核其他 cpu 架构下的 timer 驱动，发现跟 12 内核有很大差别。于是找来芯晟 32 内核中的 timer 部分加以修改，添加到 25 内核中，并修改 arch/arm/Kconfig，

```
config GENERIC_TIME
 bool
 default y

config GENERIC_CLOCKEVENTS
 bool
 default y
```

编译生成 uImage 后重新启动内核，跑到 local_irq_enable 又死了（假死机，因为 timer 中断里还有打印）。

继续调试的时候发现如果把 timer 初始化去掉就可以往下跑，说明还是 timer 中断有问题。仔细对比芯晟 32 内核中 timer 部分的修改发现是我把 PA_TIMER0_BASE 定义错了，也就是定时器 0 的硬件地址不对，改正后 local_irq_enable 过去了。

然后跑到挂载根文件系统的时候再次出现 kernel panic，估计是我之前改了一些跟 MTD block driver 和 fs 的东西造成的。因为目前只能通过 NFS 方式挂载 android 文件系统，因此我决定先将 boot argument 改为 nfs 方式挂载。这么改完后仍然出现 kernel panic，PC 指针停在了 cn100_interrupt 的位置，这是 CSM1201 板子上的以太网中断服务程序。仔细检查发现是 cn100_interrupt 中有一个指针搞错了。但是矫正后内核仍然无法启动。因为我之前将 cn100 的驱动针对新版内核做了一些改动，就是这部分导致网络不好用。最后对比芯晟 32 内核中 cn100 驱动进行修改，网络驱动终于好用了。

接下来我还是继续调试 MTD block 驱动部分。

首先配置内核，将 MTD block 驱动的 debug log 放开，打印出：

```
mtd: Giving out device 0 to physmap-flash.0
```

google 后发现可能跟 boot 参数有关。仔细检查 boot 参数：

```
setenv bootargs 'root=/dev/mtdblock/2 rootfstype=jffs2 rw
init=/init mem=100M console=ttyS0,115200
mtdparts=phys_mapped_flash:640k(u-
boot1)ro,1664k(kernel),4480k(jffs2),1088k(app),320k(stbinfo),512
k(u-boot2)ro,5824k(res),320k(cainfo),1024k(bootlogo)'
```

发现 mtdparts=phys_mapped_flash 这句比较可疑，phys_mapped_flash 是从哪来的呢?

在老版内核中搜索“phys_mapped_flash”，发现内核 pphysmap.c 中有:

```
struct map_info physmap_map = {
 .name = "phys_mapped_flash",
 .phys = CONFIG_MTD_PHYSMAP_START,
 .size = CONFIG_MTD_PHYSMAP_LEN,
 .bankwidth = CONFIG_MTD_PHYSMAP_BANKWIDTH,
};
```

于是到新版内核中去找，发现有：

```
static struct platform_device physmap_flash = {
 .name  = "physmap-flash",
 .id  = 0,
 .dev  = {
  .platform_data = &physmap_flash_data,
 },
 .num_resources = 1,
 .resource = &physmap_flash_resource,
};
```

于是将 boot 参数改为 mtdparts=physmap-flash。重启内核后仍然出现

```
mtd: Giving out device 0 to physmap-flash.0
```

boot 参数改成 mtdparts=physmap-flash.0 后，有了一些进展，打印如下：

```
[    8.880000] 9 cmdlinepart partitions found on MTD device
physmap-flash.0
[    8.890032] Creating 9 MTD partitions on "physmap-flash.0":
[    8.900034] 0x00000000-0x000a0000 : "u-boot1"
[    8.910029] mtd: Giving out device 0 to u-boot1
[    8.981516] 0x000a0000-0x00240000 : "kernel"
[    8.990092] mtd: Giving out device 1 to kernel
[    9.060259] 0x00240000-0x006a0000 : "jffs2"
[    9.070113] mtd: Giving out device 2 to jffs2
[    9.141523] 0x006a0000-0x007b0000 : "app"
[    9.160037] mtd: Giving out device 3 to app
[    9.230289] 0x007b0000-0x00800000 : "stbinfo"
[    9.250105] mtd: Giving out device 4 to stbinfo
[    9.310282] 0x00800000-0x00880000 : "u-boot2"
[    9.330098] mtd: Giving out device 5 to u-boot2
[    9.390244] 0x00880000-0x00e30000 : "res"
[    9.400112] mtd: Giving out device 6 to res
[    9.470158] 0x00e30000-0x00e80000 : "cainfo"
[    9.480109] mtd: Giving out device 7 to cainfo
[    9.540383] 0x00e80000-0x00f80000 : "bootlogo"
[    9.550113] mtd: Giving out device 8 to bootlogo
[    9.670462] input: ORION Remote Controller as
/devices/virtual/input/input0
[    9.770478] Orion Watchdog Timer: timer margin 40 sec
[    9.890413] TCP cubic registered
[    9.930079] NET: Registered protocol family 1
[    9.940207] NET: Registered protocol family 17
[   10.000428] RPC: Registered udp transport module.
[   10.010062] RPC: Registered tcp transport module.
[   10.070386] Root-NFS: No NFS server available, giving up.
[   10.080150] VFS: Unable to mount root fs via NFS, trying floppy.
[   10.110265] MTDSB: dev_name "/dev/root"
[   10.120065] MTDSB: path_lookup() returned 0, inode c5c00c08
[   10.130051] List of all partitions:
<4>[   10.140046] 1f00        640 mtdblock0[   10.140046]
1f00        640 mtdblock0 (driver?)
 (driver?)
<4>[   10.160035] 1f01       1664 mtdblock1[   10.160035]
1f01       1664 mtdblock1 (driver?)
 (driver?)
<4>[   10.180031] 1f02       4480 mtdblock2[   10.180031]
1f02       4480 mtdblock2 (driver?)
 (driver?)
<4>[   10.200032] 1f03       1088 mtdblock3[   10.200032]
1f03       1088 mtdblock3 (driver?)
 (driver?)
<4>[   10.220032] 1f04        320 mtdblock4[   10.220032]
1f04        320 mtdblock4 (driver?)
```

```
 (driver?)
<4>[   10.240033] 1f05         512 mtdblock5[   10.240033]
1f05         512 mtdblock5 (driver?)
 (driver?)
<4>[   10.260032] 1f06        5824 mtdblock6[   10.260032]
1f06        5824 mtdblock6 (driver?)
 (driver?)
<4>[   10.280033] 1f07         320 mtdblock7[   10.280033]
1f07         320 mtdblock7 (driver?)
 (driver?)
<4>[   10.300032] 1f08        1024 mtdblock8[   10.300032]
1f08        1024 mtdblock8 (driver?)
 (driver?)
<4>[   10.320022] No filesystem could mount root, tried:
[   10.320022] No filesystem could mount root, tried:  jffs2 jffs2

<0>[   10.350042] Kernel panic - not syncing: VFS: Unable to
mount root fs on unknown-block(2,0)
[   10.350042] Kernel panic - not syncing: VFS: Unable to mount
root fs on unknown-block(2,0)
```

然后 google"mtdblock0 (driver?)"，发现有人遇到同样的问题，原因是新版内核不支持 devfs 造成的，应该将 boot 参数中的 root=/dev/mtdblock/2 改为 root=31:2
试了一下，确实好用。

至此内核终于跑起来了！

最后的 boot 参数如下：

setenv bootargs 'root=31:2 rootfstype=jffs2 rw init=/init mem=100M console=ttyS0,115200 mtdparts=physmap-flash.0:640k(u-boot1)ro,1664k(kernel),4480k(jffs2),1088k(app),320k(stbinfo),512k(u-boot2)ro,5824k(res),320k(cainfo),1024k(bootlogo)'

接下来就是移植 Android 内核到第一步升级后的芯晟内核。通过比较发现 Android 对内核的修改和芯晟对内核的修改几乎没有重合的地方，因此移植相对容易。这里参考了网上一位外国朋友 Peter McDermott 的文章—《**Porting Android to a new device**》：

What did Google change in the kernel?

We checked the differences between the Android kernel and the standard Linux kernel and found that Google had changed 75 files and added an additional 88. We have prepared an annotated list of changed files at the end of this document, and a brief summary here.

- **Goldfish -- 44 Files**--

 The Android emulator runs a virtual CPU that Google calls Goldfish. Goldfish executes ARM 926T instructions and has hooks for input and output --
 such as reading key presses from or displaying video output in the emulator.
These interfaces are implemented in files specific to the Goldfish emulator and will not be compiled into a kernel that runs on real devices. So we safely ignored these files in our work.

- **YAFFS2 -- 35 Files**--

 Unlike PCs, which store files on disks, mobile phones store files in sold-state flash memory chips. The HTC G1 uses NAND flash, a type of flash memory that is becoming more popular due to its combination of high density and low cost.
YAFFS2 is an acronym for "Yet Another Flash File System, 2nd edition." It provides a high-performance interface between the Linux kernel and NAND flash devices. YAFFS2 was already freely available for Linux. However, it is not part of the standard 2.6.25 Linux kernel, and so Google added it to Android.

- **Bluetooth -- 10 files**--

 Google made changes to 10 files in the Bluetooth communications stack. These changes fix apparent bugs related to Bluetooth headsets, and add Bluetooth debugging and access control functions.

- **Scheduler -- 5 files**--

 The Android kernel also contains slight changes to the CPU process scheduler and time-keeping algorithms. We don't know the history of these changes, and the impact was not evident based on a cursory examination.

- **New Android Functionality -- 28 files**--

 In addition to bug fixes and other small changes, Android contains a number of new subsystems that are worth mentioning here, including the following:

o

o **IPC Binder**-- The IPC Binder is an Inter-
Process Communication (IPC) mechanism. It allows processes to provide services to other processes via a set of higher-
level APIs than are available in standard Linux. An Internet search indicated that the Binde r concept originated at Be, Inc., and then made its way into Palm's software, before Googl e wrote a new Binder for Android.

o **Low Memory Killer**-- Android adds a low-
memory killer that, each time it's called, scans the list of running Linux processes, and kills one. It was not clear in our cursory examination why Android adds a low-
memory killer on top of the already existing one in the standard Linux kernel.

o **Ashmem**--
Ashmem is an Anonymous SHared MEMory system that adds interfaces so processes can s hare named blocks of memory. As an example, the system could use Ashmem to store ico ns, which multiple processes could then access when drawing their UI. The advantage of A shmem over traditional Linux shared memory is that it provides a means for the kernel to r eclaim these shared memory blocks if they are not currently in use. If a process then tries to access a shared memory block the kernel has freed, it will receive an error, and will the n need to reallocate the block and reload the data.

o **RAM Console and Log Device**--
To aid in debugging, Android adds the ability to store kernel log messages to a RAM buffer . Additionally, Android adds a separate logging module so that user processes can read an d write user log messages.

o **Android Debug Bridge**--
Debugging embedded devices can best be described as challenging. To make debugging e asier, Google created the Android Debug Bridge (ADB), which is a protocol that runs over a USB link between a hardware device running Android and a developer writing applications on a desktop PC.

Android also adds a new real-
time clock, switch support, and timed GPIO support. We list the impacted files for these ne w modules at the end of this document.

•

- **Power Management -- 5 files**--

Power management is one of the most difficult pieces to get right in mobile devices, so we split it out into a group separate from the other pieces. It's interesting to note that Google added a new power management system to Linux, rather than reuse what already existed . We list the impacted files at the end of this document.

- **Miscellaneous Changes -- 36 files**--

In addition to the above, we found a number of changes that could best be described as, 'Miscellaneous.' Among other things, these changes include additional debugging support, keypad light controls, and management of TCP networking.

- **NetFilter -- 0 files**--

Finally, our change list showed Netfilter as having 22 changed files. However, examination showed the only difference was the capitalization of the filenames (xt_DSCP.c vs. xc_dscp.c). The contents of the files were all identical. So we ignored these files in our port.

因为我的移植是针对机顶盒的，因此 bluetooth 和 power management 都是不需要的，我的试验平台是 NOR FLASH，因此也不需要 YAFFS2 文件系统。剩下的改动只需做简单的拷贝即可。然后重新做 menuconfig，把 Android 新增的功能选上。编译很容易就过了。 但是内核启动的时候提示：

Kernel panic - not syncing: Attempted to kill init!

搜了一下发现必须用 Android 自己的根文件系统。