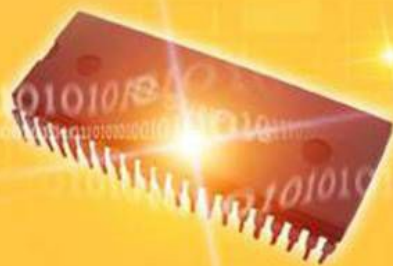


# 嵌入式系统工程师



---

# Bootloader应用分析

---



- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程

- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程

- Bootloader中文解释为：启动引导程序
- Bootloader的种类繁多，归纳如下：

分类标准	说明
针对不同CPU架构	<ul style="list-style-type: none"><li>1、针对X86架构的有LILO、GRUB、ntldr等</li><li>2、针对ARM架构的有vivi、armboot等</li><li>3、针对PPC架构的有ppcboot等</li><li>4、可以支持多种架构的u-boot等</li></ul>
针对不同操作系统	<ul style="list-style-type: none"><li>1、专门用来启动Linux系统的vivi</li><li>2、专门用来启动WinCE系统的eboot</li><li>3、基于eCos系统的引导程序redboot</li><li>4、可以启动多种操作系统的u-boot等</li></ul>

➤ 下面将一些常用Bootloader做简单对比:

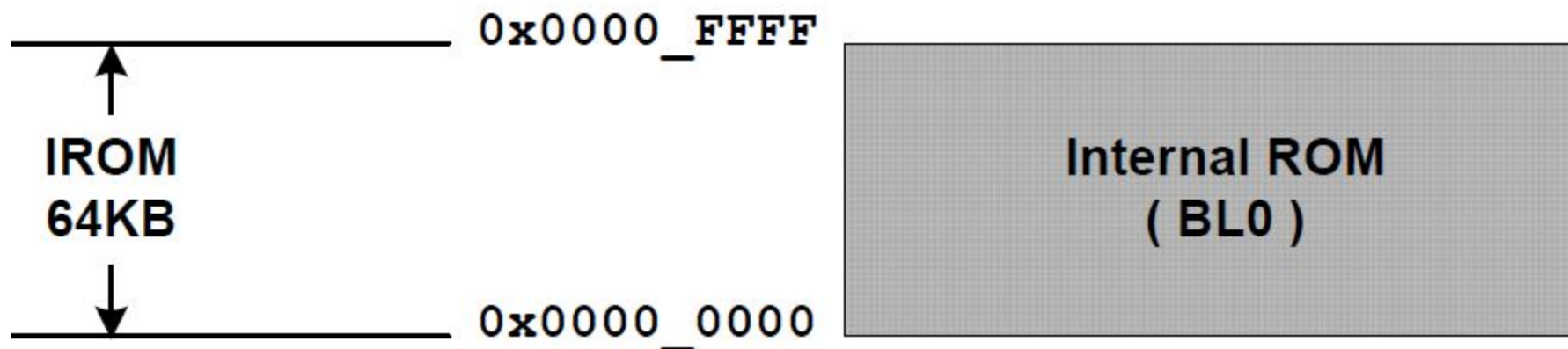
Bootloader	Monitor	描述	X86	ARM
LILO	否	Linux磁盘引导程序	是	否
GRUB	否	GNU的LILO替代程序	是	否
ntldr	否	x86上引导windowsNT系列	是	否
armboot	是	专门为arm架构设计的boot	否	是
ppcboot	是	引导ppc架构操作系统	否	是
vivi	是	韩国Mizi公司针对三星ARM架构CPU设计引导程序	否	是
redboot	是	基于eCos的引导程序	是	是
u-boot	是	通用引导程序, 支持多种CPU架构、多种操作系统	是	是

- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程

- 用户可以通过OM[5:0]进行启动方式选择，详见  
《5PV210\_iROM\_ApplicationNote\_Preliminary\_20091126.pdf》 p22
- 六个拨码开关
  - OM[0]：时钟源选择，选择独立时钟还是USB时钟
  - OM[3:1]：选择启动方式，即选择第一阶段程序来自于哪一种外部存储设备
  - OM[5:4]：选择启动模式
    - OM[5:4]=0x，为正常模式，启动方式将由OM[3:1]决定
    - OM[5:4]=10，为调试模式，调试通道为uart ch2，系统通过串口或USB将程序下载到内存进行引导启动，如果启动失败，依然会根据OM[3:1]进入正常启动过程



- s5pv210支持多种启动方式，无论选择什么样的启动方式，代码都是从iROM开始执行，如下图：

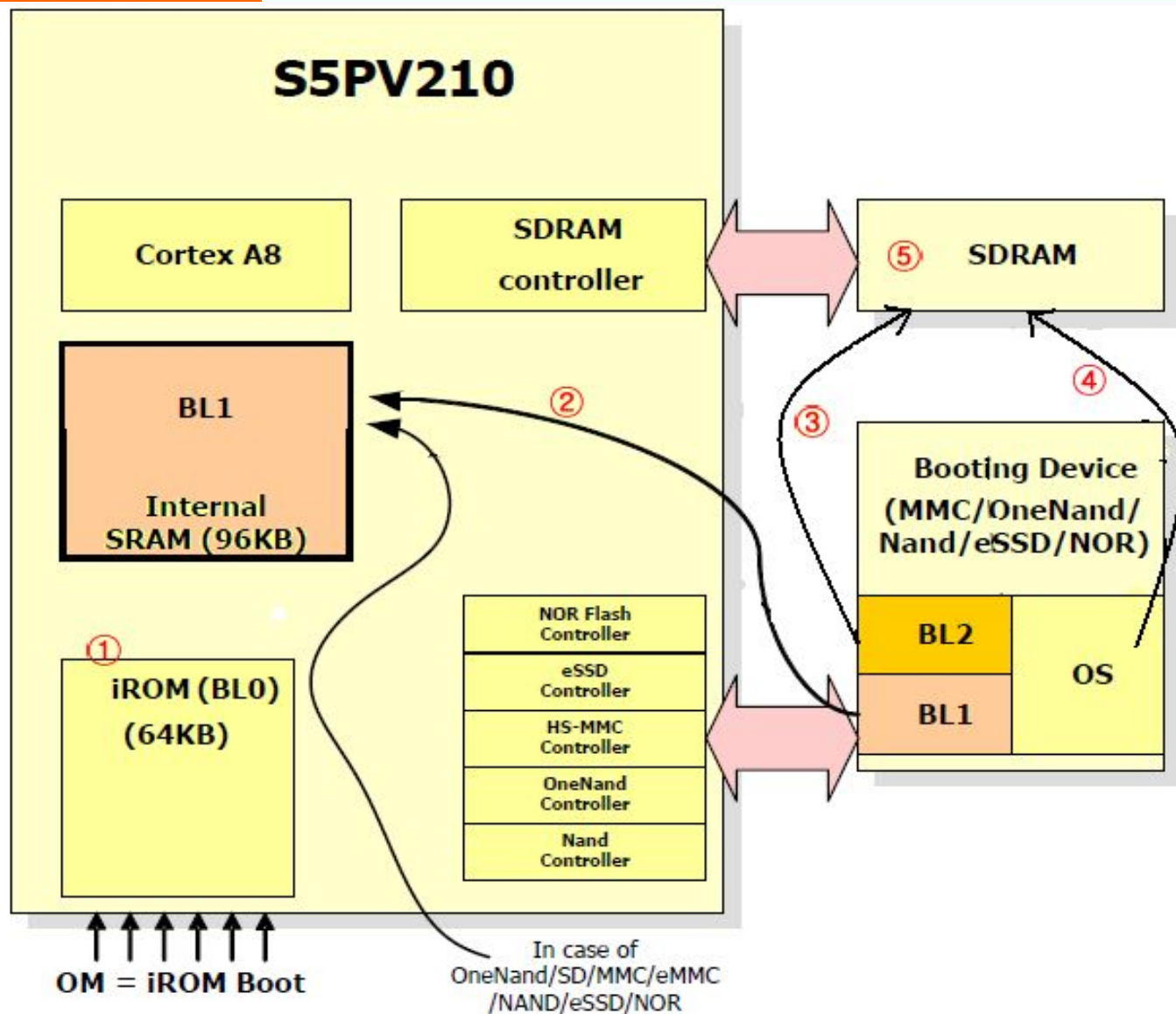


- 嵌入式Bootloader的启动过程可以分为以下两种：  
单阶段(Single-Stage)、多阶段(Multi-Stage)
- 通常多阶段的Bootloader能够提供更复杂的功能，及更好的可读性和移植性。从外部存储设备上启动的Bootloader大多是两阶段的启动过程，其功能如下：

阶段	功能特点
第一阶段	通常用汇编实现，完成一些依赖于CPU体系结构的初始化，并调用第二阶段代码
第二阶段	通常用C语言实现，完成体系结构之外的功能

- 以下是s5pv210启动流程：

# s5pv210启动过程



- 内核传参过程：
  - 传递给内核的参数由多个结构体组成
  - 各结构体放在一段连续的内存空间，起始地址为 0x3000\_0100
  - 每一个结构体代表一条信息，并首尾相连
  - 内核引导起来以后，将从指定内存按照同样的数据结构将数据取出
  
- 参数数据结构及存储结构如下：

## ➤ 参数数据结构:

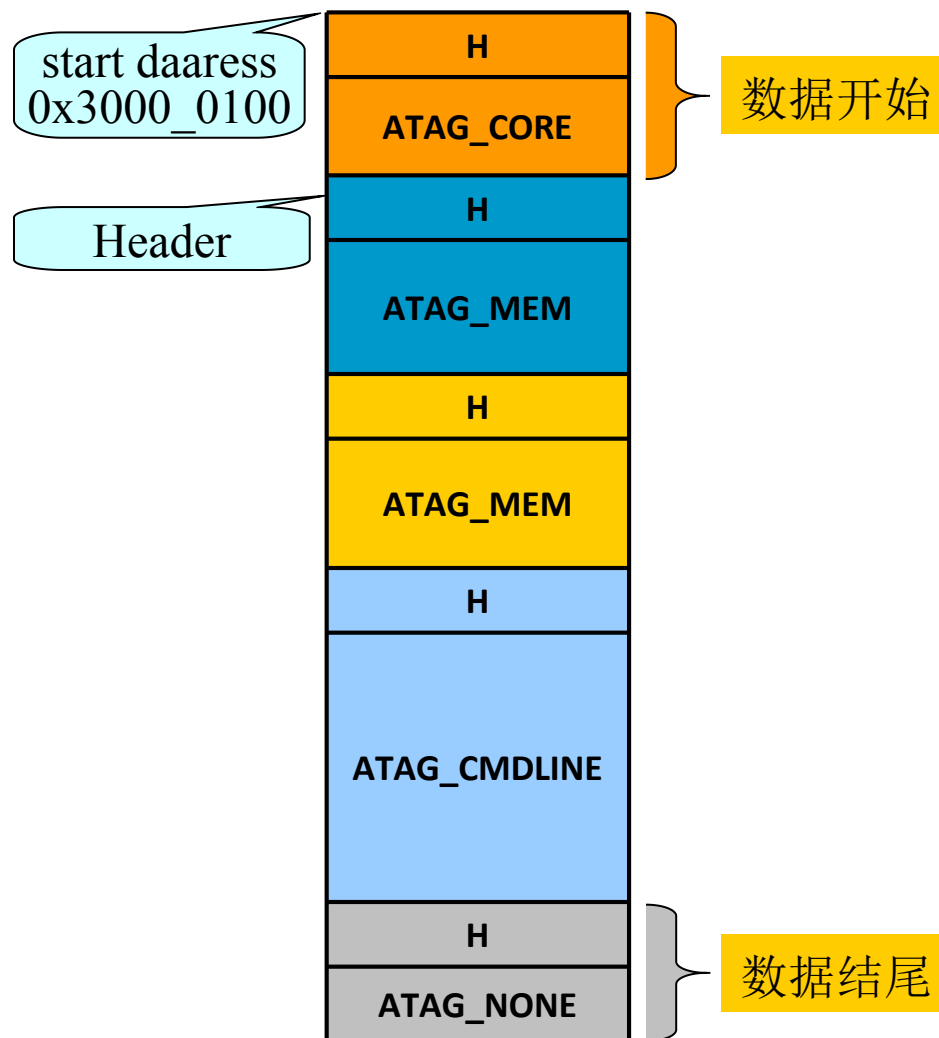
```

struct tag {
    struct tag_header hdr;
    union {
        struct tag_core      core;
        struct tag_mem32     mem;
        struct tag_videotext videotext;
        struct tag_ramdisk   ramdisk;
        struct tag_initrd     initrd;
        struct tag_serialnr   serialnr;
        struct tag_revision   revision;
        struct tag_videolfb   videolfb;
        struct tag_cmdline    cmdline;
        struct tag_acorn       acorn;
        struct tag_memclk     memclk;
        struct tag_mtdpart    mtdpart_info;
    } u;
};

struct tag_header {
    u32      size;
    u32      tag;
};

struct tag_mem32 {
    u32      size;
    u32      start; /*physical
start address*/
};
    
```

## ➤ 参数存储结构:



- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程



## ➤ u-boot

### ➤ 简介:

- u-boot最初是由PPCBoot发展而来的，可以引导多种操作系统、支持多种架构的CPU，它对PowerPC系列处理器的支持最为完善，而操作系统则对Linux系统的支持最好
- 目前已成为Armboot和PPCboot的替代品

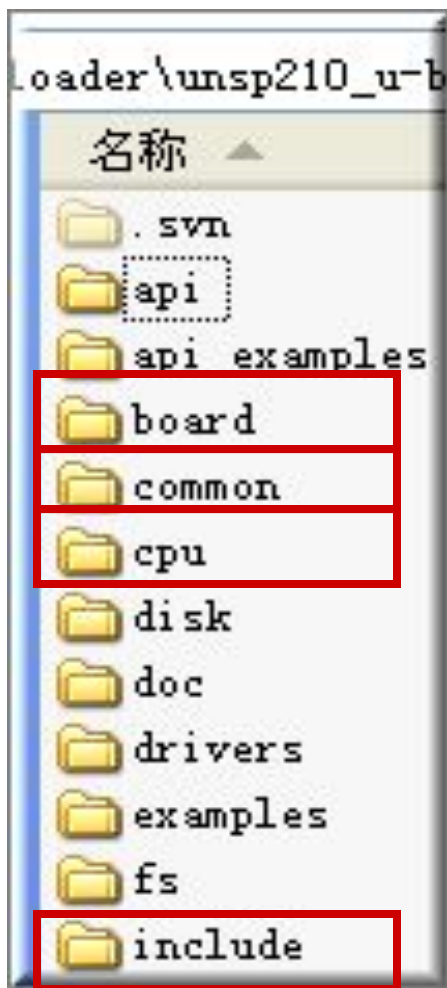
### ➤ 特点:

- 主要支持操作系统：Linux、NetBSD、 VxWorks、QNX、RTEMS、ARTOS、LynxOS等
- 主要支持处理器架构：PowerPC、MIPS、x86、ARM、NIOS、XScale等

### ➤ u-boot 目前最新版本是:

- <http://ftp.denx.de/pub/u-boot/>

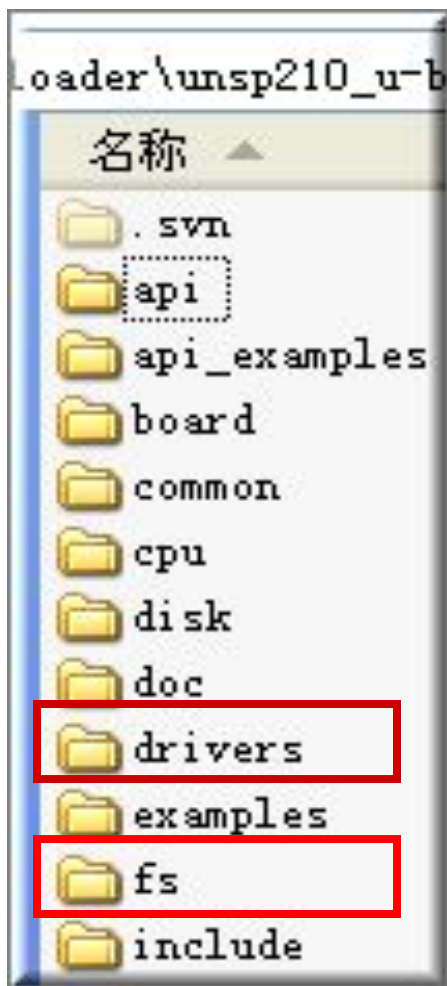
## ➤ 目录结构介绍:



- **board:** 开发板相关, 根据厂商进行分类, 如当前平台board\samsung\unsp210, 包含第一阶段要用到的一些初始化程序: lowlevel\_init.S
- **cpu:** 体系结构相关, 按架构进行分类, 如当前架构cpu\s5pc11x, 第一阶段启动代码start.S就在这里
- **common:** 各种命令的实现, 通常一个命令就是一个C文件
- **include:** 各种头文件和开发板配置文件, 如include\configs\unsp210.h

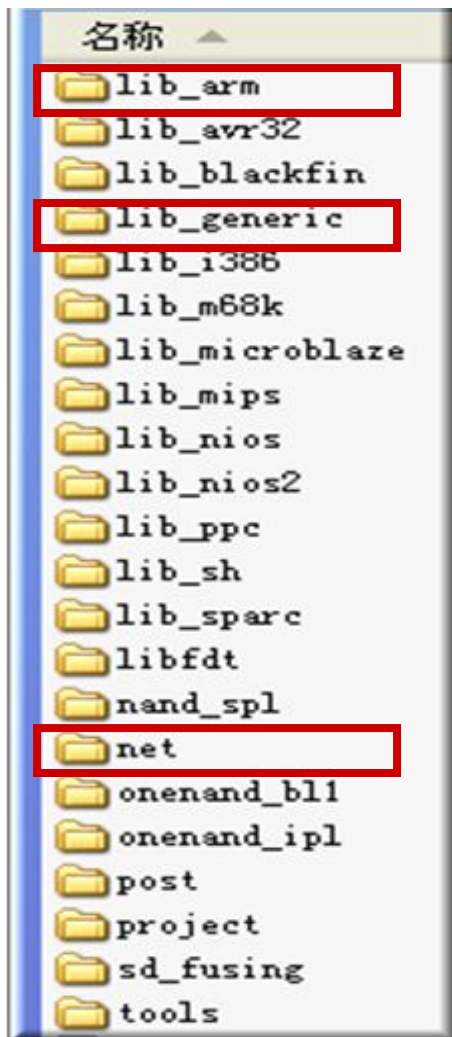


## ➤ 目录结构介绍:



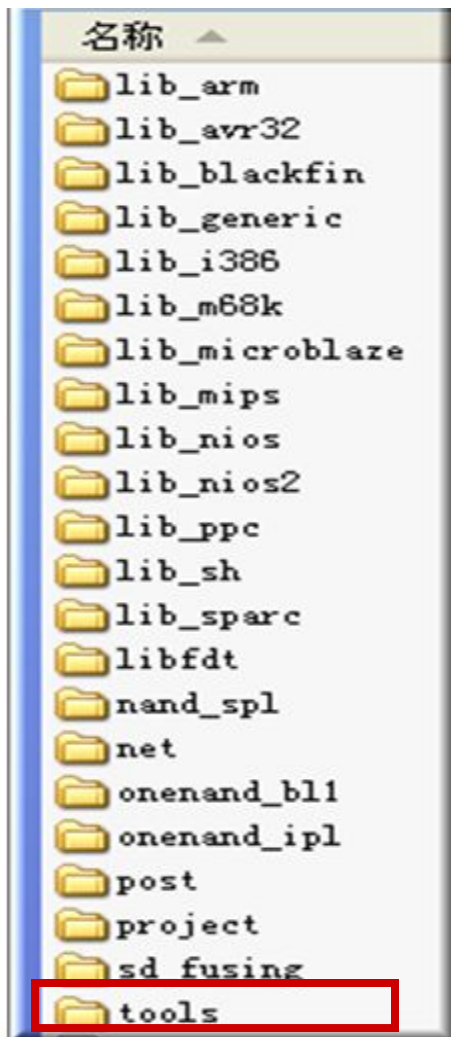
- **api/api\_examples:** 用于演示测试用的代码，通常不参与工程编译
- **disk:** 硬盘接口程序，disk驱动分区处理代码，嵌入式不常用
- **doc:** 开发使用文档，主要介绍不同平台的配置编译方法
- **drivers:** 设备驱动，如:网卡、SD卡、USB等
- **examples:** 一些独立运行的实例，通常不参与工程编译
- **fs:** 所能支持的文件系统，如fat、ubifs等，用于访问带文件系统的存储设备

## ➤ 目录结构介绍:



- **lib\_arm**: 用于存放平台依赖文件，如第二阶段代码入口(start\_armboot())、中断处理、启动相关等，其它以“lib\_”开头的也是类似
- **lib\_generic**: 存放通用且不依赖于平台的库，如一些C库，终端操作接口等
- **nand\_spl**: 一些配置参考文件
- **net**: 独立于网卡驱动的网络协议，如用于下载传输的TFTP协议，网络文件系统中的NFS协议等
- **onenand\_b11**: onenand启动的第一阶段代码
- **onenand\_ipl**: onenand启动时，IPL和SPL过程中的第一个过程，即初始化阶段

## ➤ 目录结构介绍:



- **post**: 上电自检程序，与旧的PPC相关，当前平台下未编译到工程
- **sd\_fusing**: 一些操作SD卡的脚本，主要用于制作SD引导启动相关
- **tools**: 工具软件，如mkimage用于制作内核镜像，scripts用于生成指定的config.mk配置文件，还有支持GDB的调试工具等

## ➤ u-boot的配置编译需要经过以下步骤:

1、在u-boot的根目录下执行:

➤ #make unsp210\_config //对应开发板配置

➤ Makefile 会构建编译结构, 如: 架构、cpu、开发板、厂商、芯片、目录等, 为下一步真正编译链接做准备。

2、修改include/configs/unsp210.h配置文件

3、在根目录下执行: make

➤ 根据以上两步产生编译和连接所需文件的信息

➤ 最终make完成, 在根目录下将生成:

u-boot.bin    u-boot.dis    u-boot.map    .....

## ➤ 配置过程如下:



```
unsp210_config :      unconfig
    @$(MKCONFIG) $(@:_config=) arm s5pc11x unsp210 samsung s5pc110
    @echo "TEXT_BASE = 0xc3e00000" > $(obj)board/samsung/unsp210/config.mk
```

加载地址

目标

ARCH

CPU

BOARD

VENDOR

SOC

上图和右图为顶层目录  
Makefile部分截图：

```
96 SRCTREE      := $(CURDIR)
97 TOPDIR       := $(SRCTREE)
98 LNDIR        := $(OBJTREE)
99 export      TOPDIR SRCTREE OBJTREE
100
101 MKCONFIG     := $(SRCTREE)/mkconfig
102 export MKCONFIG
```

右图为顶层目录  
mkconfig部分截图：

```
22
23 [ "${BOARD_NAME}" ] || BOARD_NAME="$1"
24
25 [ $# -lt 4 ] && exit 1
26 [ $# -gt 6 ] && exit 1
27
28 echo "Configuring for ${BOARD_NAME} board..."
```

获取第一个参数

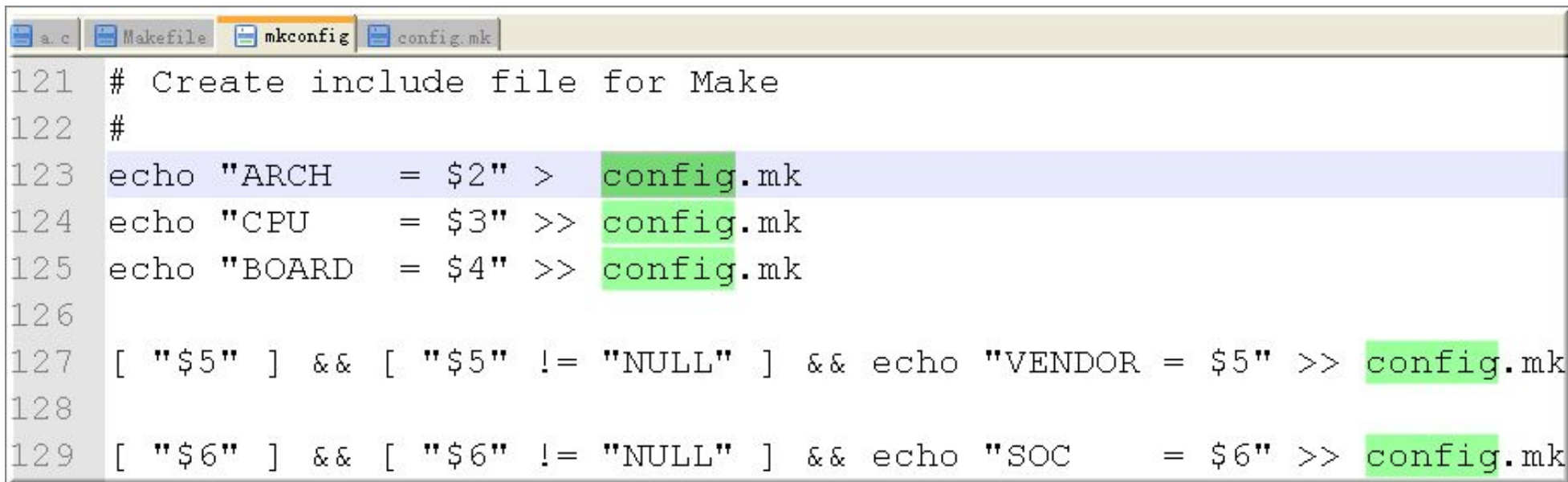
## ➤ 继续执行顶层目录mkconfig

创建新链接

```
33 if [ "$SRCTREE" != "$OBJTREE" ] ; then
34     mkdir -p ${OBJTREE}/include
35     mkdir -p ${OBJTREE}/include2
36     cd ${OBJTREE}/include2
37     rm -f asm
38     ln -s ${SRCTREE}/include/asm-$2 asm
39     LNPREFIX="../../include2/asm/"
40     cd ../include
41     rm -rf asm-$2
42     ? rm -f asm
43     mkdir asm-$2
44     ln -s asm-$2 asm
45 else
46     cd ../include
47     rm -f asm
48     ln -s asm-$2 asm
49 fi
```

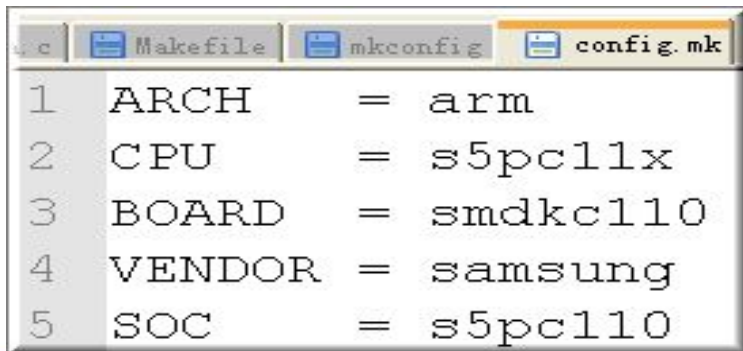
进入include目录

下图为顶层目录mkconfig部分截图：



```
a.c  Makefile  mkconfig  config.mk
121 # Create include file for Make
122 #
123 echo "ARCH    = $2" > config.mk
124 echo "CPU      = $3" >> config.mk
125 echo "BOARD    = $4" >> config.mk
126
127 [ "$5" ] && [ "$5" != "NULL" ] && echo "VENDOR = $5" >> config.mk
128
129 [ "$6" ] && [ "$6" != "NULL" ] && echo "SOC      = $6" >> config.mk
```

下图为include/config.mk截图：



```
a.c  Makefile  mkconfig  config.mk
1  ARCH    = arm
2  CPU      = s5pc11x
3  BOARD    = smdkc110
4  VENDOR   = samsung
5  SOC      = s5pc110
```



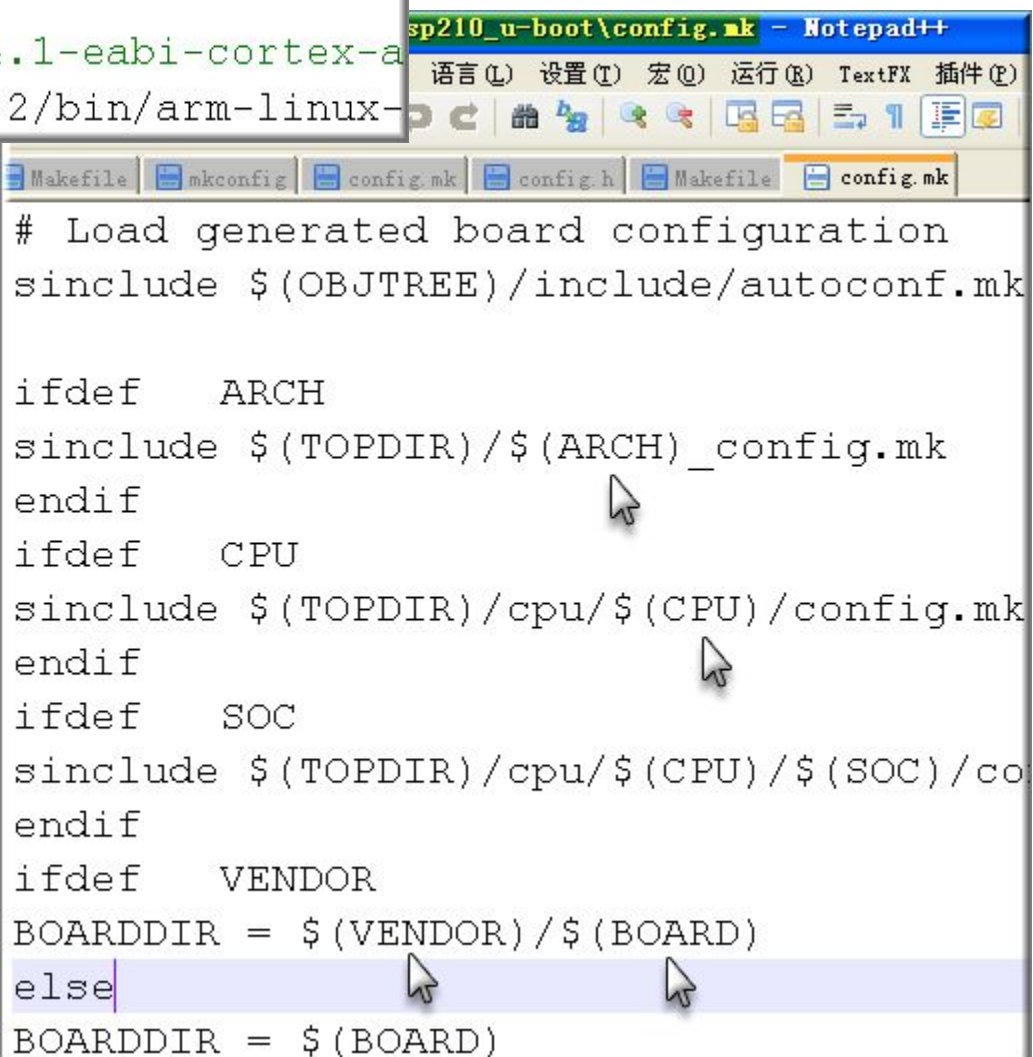
```
X:\work\platform\bootloader\unsp210_u-boot\Makefile - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 语言(L) 设置(T) 宏(O) 运行(R) TextFX 插件(P) 窗口
顶层Makefile包含config.mk
132 # load ARCH, BOARD, and CPU configuration
133 include $(obj)include/config.mk
134 export ARCH CPU BOARD VENDOR SOC
135
```



```
143 ifeq ($(ARCH),arm)
144 #CROSS_COMPILE = arm-linux-
145 #CROSS_COMPILE = /usr/local/arm/4.4.1-eabi-cortex-a
146 CROSS_COMPILE = /usr/local/arm/4.3.2/bin/arm-linux-
```

上图为顶层目录Makefile部分截图:

## ➤ 环境变量的引用:



```
sp210_u-boot\config.mk - Notepad++
语言(L) 设置(S) 宏(O) 运行(R) TextFX 插件(P)

Makefile mkconfig config.mk config.h Makefile config.mk

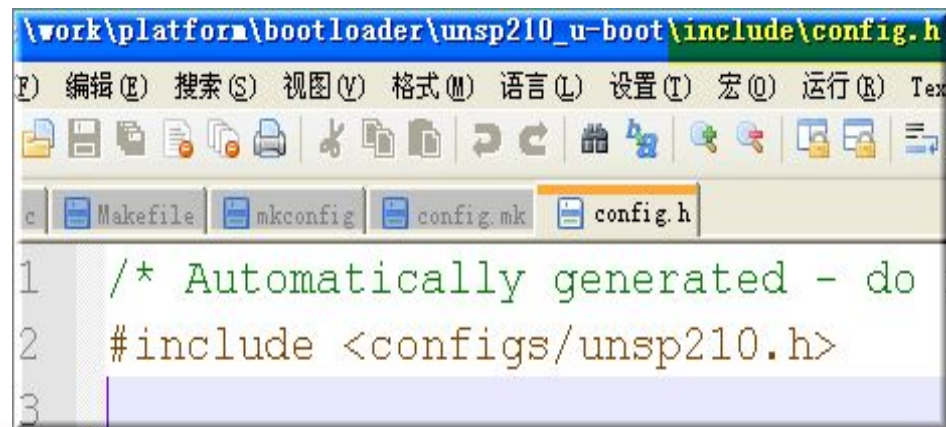
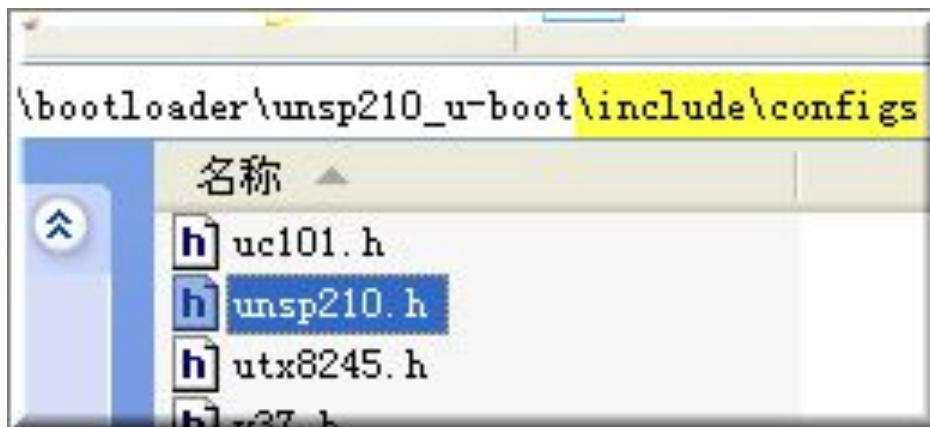
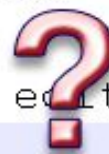
# Load generated board configuration
#include $(OBJTREE)/include/autoconf.mk

ifdef ARCH
#include $(TOPDIR)/$(ARCH)_config.mk
endif
ifdef CPU
#include $(TOPDIR)/cpu/$(CPU)/config.mk
endif
ifdef SOC
#include $(TOPDIR)/cpu/$(CPU)/$(SOC)/co
endif
ifdef VENDOR
BOARDDIR = $(VENDOR)/$(BOARD)
else
BOARDDIR = $(BOARD)
```



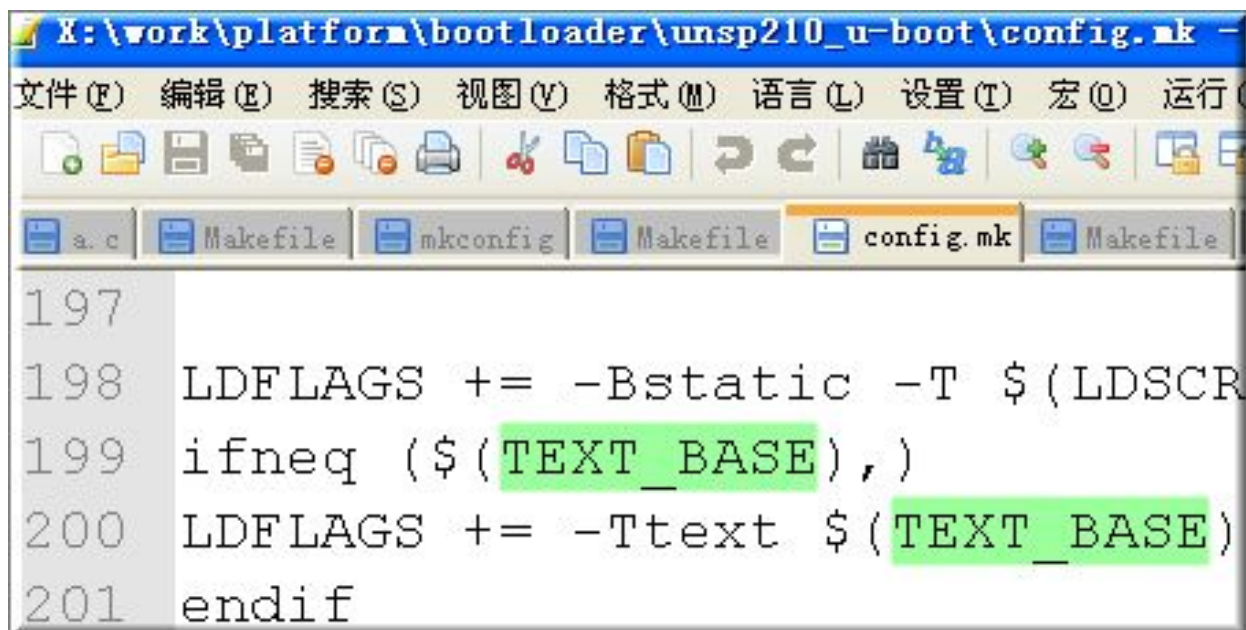
- 添加平台头文件 (unsp210.h)，可以根据当前平台进行修改

```
a.c Makefile mkconfig config.mk
132 # Create board specific header file
133 #
134 if [ "$APPEND" = "yes" ]      # Append to existing config file
135 then
136     echo >> config.h
137 else
138     > config.h      # Create new config file
139 fi
140 echo "/* Automatically generated - do not edit */" >> config.h
141 echo "#include <configs/$1.h>" >> config.h
```



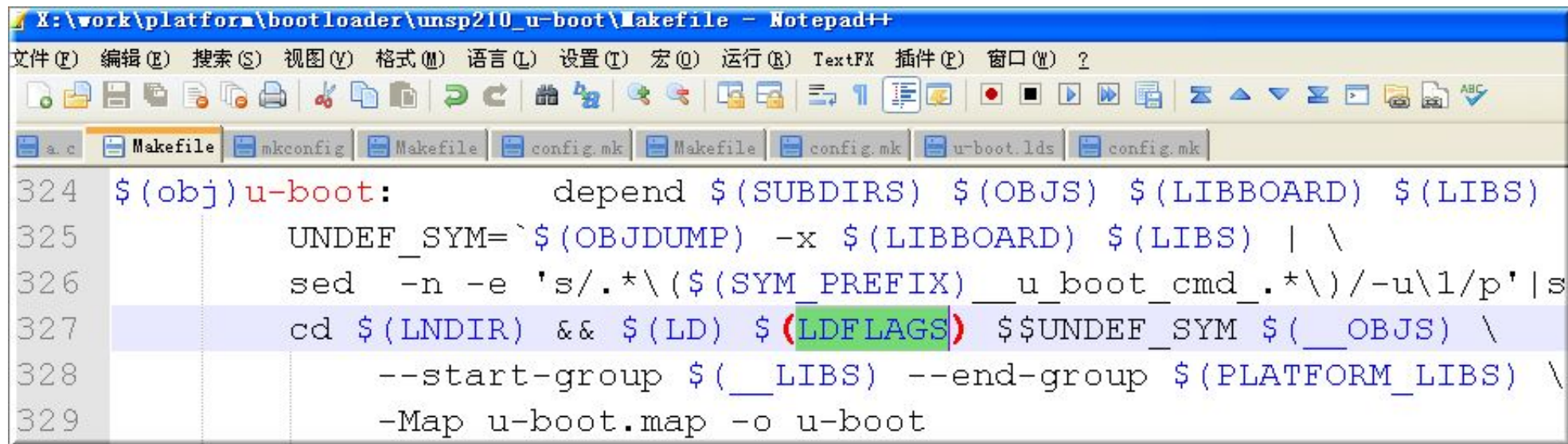
## ➤ 链接过程如下:

- 链接地址定义在board/samsung/unsp210/config.mk
- 链接脚本定义在board/samsung/unsp210/u-boot.lds
- board/samsung/unsp210/config.mk被顶层config.mk包含并设置链接选项LDFLAGS



```
X:\work\platform\bootloader\unsp210_u-boot\config.mk -
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R)
a.c Makefile mkconfig Makefile config.mk Makefile
197
198 LDFLAGS += -Bstatic -T $(LDSCRIPT)
199 ifneq ($(TEXT_BASE),)
200 LDFLAGS += -Ttext $(TEXT_BASE)
201 endif
```

## ► LDFLAGS将在顶层Makefile最终链接时发挥作用



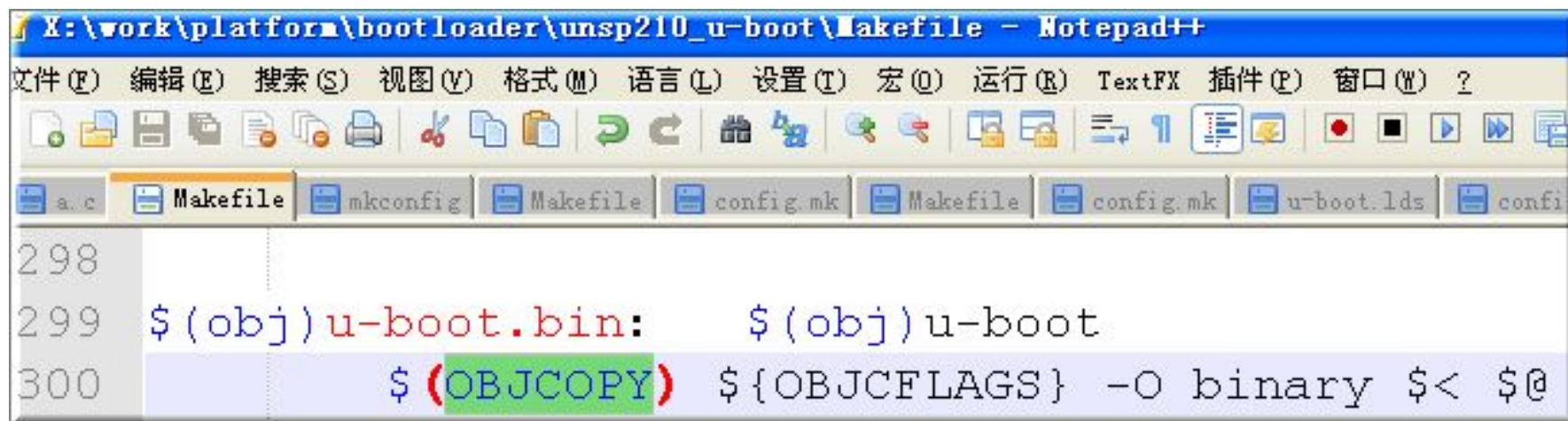
```
X:\work\platform\bootloader\unsp210_u-boot\Makefile - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 语言(L) 设置(T) 宏(O) 运行(R) TextFX 插件(P) 窗口(W) ?
a.c Makefile mkconfig Makefile config.mk Makefile config.mk u-boot.lds config.mk
324 $(obj)u-boot:      depend $(SUBDIRS) $(OBJS) $(LIBBOARD) $(LIBS)
325      UNDEF_SYM=`$(OBJDUMP) -x $(LIBBOARD) $(LIBS) | \
326      sed -n -e 's/.*\($(SYM_PREFIX) __u_boot_cmd_.*\) /-u\1/p' | s
327      cd $(LNDIR) && $(LD) $(LDFLAGS) $$UNDEF_SYM $(__OBJS) \
328      --start-group $(__LIBS) --end-group $(PLATFORM_LIBS) \
329      -Map u-boot.map -o u-boot
```

## ► 编译过程部分截图:

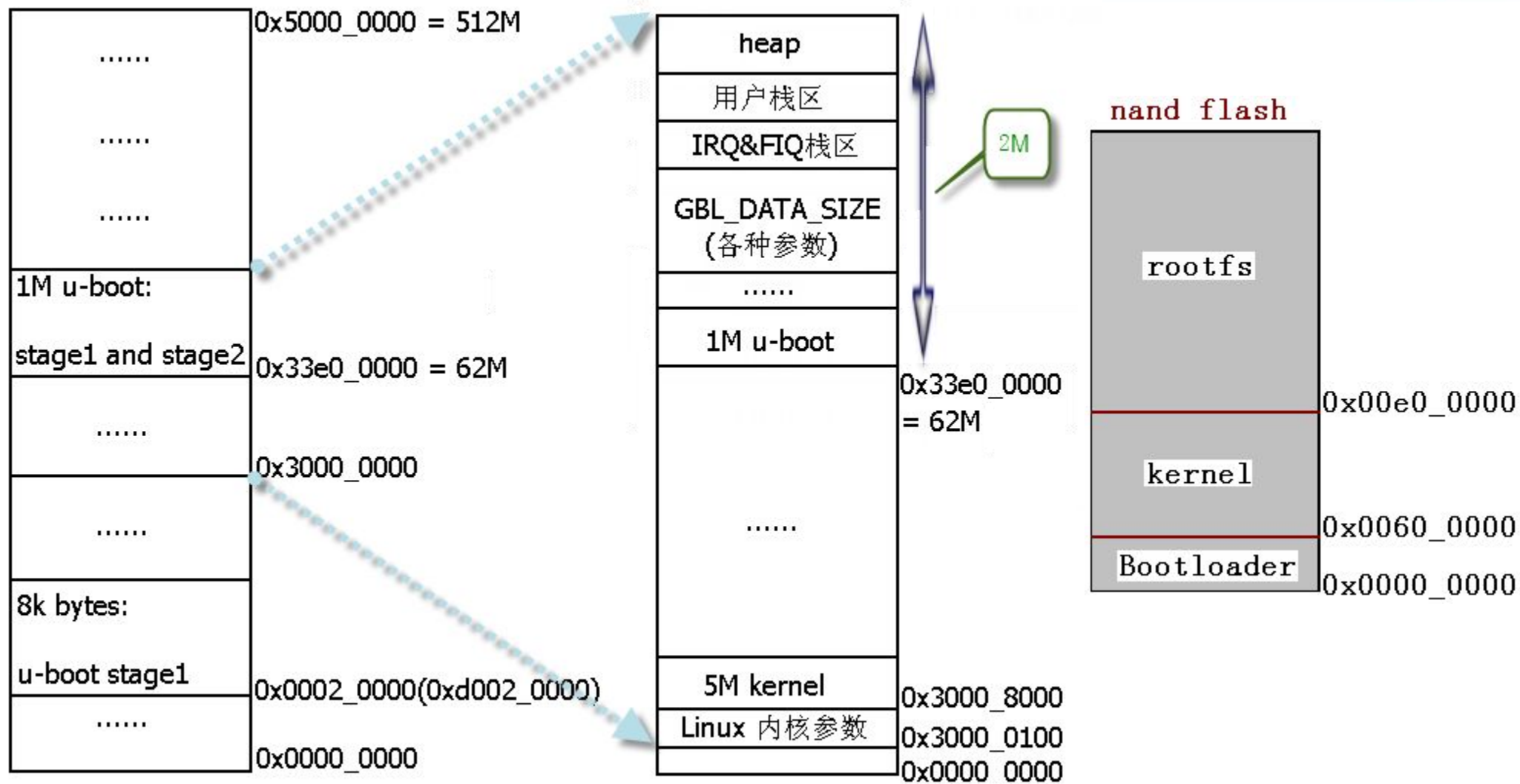
```
cd /home/xiuhai/work/platform/bootloader/unsp210_u
-boot && /usr/local/arm/4.3.2/bin/arm-linux-ld -Bstatic -T /home/x
iuhai/work/platform/bootloader/unsp210_u-boot/board/samsung/unsp21
0/u-boot.lds -Ttext 0xc3e00000 $UNDEF_SYM cpu/s5pc11x/start.o \
```



- 最终通过二进制工具去掉ELF格式信息，得到下载可直接运行的二进制文件：u-boot.bin



```
298
299 $(obj)u-boot.bin: $(obj)u-boot
300 $(OBJCOPY) ${OBJCFLAGS} -O binary $< $@
```



- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程

- 在启动过程中快速按下空格或回车键可进入下载模式，否则会执行事先内置的一条命令引导启动系统。
- 进入下载模式：
  - 输入help，可显示所有支持的命令
  - 详细可参见《常用 U-boot命令详解.mht》
  - 这里我们只介绍最常用的命令
- u-boot的命令非常多，且操作非常的灵活，主要分为以下几类：

- 串口下载类指令
- 网络命令
- NAND FLASH操作类
- 环境变量类指令
- 系统引导指令
- 脚本运行指令
- USB 操作指令
- SD卡 (MMC) 指令
- FAT文件系统指令



## ➤ 环境变量设置命令

- u-boot中采用环境变量来协调各命令工作时所需的参数及数据
- 环境变量可以记录Bootloader在运行过程中用到的可配置参数列表，以及需要传递给内核的参数
- 用户可以通过printenv、setenv、saveenv进行查看、设置、保存这些参数
- 如：

```
#printenv
```

```
#setenv bootdelay 8
```

```
#saveenv
```

```
#reset
```

- u-boot提供的环境变量主要有：
  - bootdelay 执行自动启动的等候秒数
  - baudrate 串口控制台的波特率
  - netmask 以太网接口的掩码
  - ethaddr 以太网卡的网卡物理地址
  - bootfile 缺省的下载文件
  - bootargs 传递给内核的启动参数
  - bootcmd 自动启动时执行的命令
  - serverip 服务器端的ip地址
  - ipaddr 本地ip地址

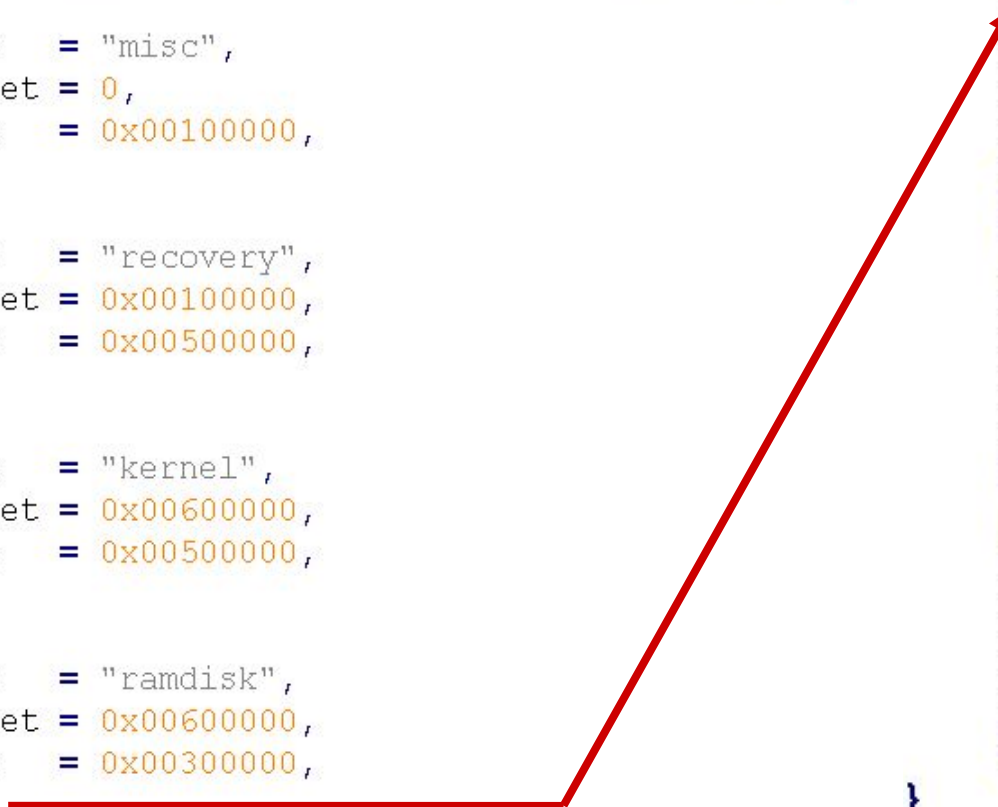
## ➤ 下载命令

- 下载地址：下载地址是为了方便系统更新时，明确镜像放在存储设备的什么位置
- 用户可以通过各种下载协议完成系统镜像资源到磁盘指定位置的更新
- u-boot需要用户手动指定分区地址，分区信息可以参考内核启动时的打印信息或内核源代码：

```
Creating 7 MTD partitions on "s5pv210-nand":
0x00000000c0000-0x0000000100000 : "misc"
0x0000000100000-0x0000000600000 : "recovery"
0x0000000600000-0x0000000b00000 : "kernel"
0x0000000b00000-0x0000000e00000 : "ramdisk"
0x0000000e00000-0x00000020e0000 : "system"
0x00000020e0000-0x0000002220000 : "cache"
0x0000002220000-0x0000004000000 : "userdata"
```

- 为了保证Bootloader将控制权交给内核以后，内核能成功加载应用程序，在Linux内核驱动中也有一张分区表，且必须和Bootloader下载镜像的地址相同，如图：

```
static struct mtd_partition smdk_default_nand_part[] = {  
    [0] = {  
        .name      = "misc",  
        .offset    = 0,  
        .size      = 0x00100000,  
    },  
    [1] = {  
        .name      = "recovery",  
        .offset    = 0x00100000,  
        .size      = 0x00500000,  
    },  
    [2] = {  
        .name      = "kernel",  
        .offset    = 0x00600000,  
        .size      = 0x00500000,  
    },  
    [3] = {  
        .name      = "ramdisk",  
        .offset    = 0x00600000,  
        .size      = 0x00300000,  
    },  
    [4] = {  
        .name      = "system",  
        .offset    = 0x00e00000,  
        .size      = 0x06e00000,  
    },  
    [5] = {  
        .name      = "cache",  
        .offset    = 0x07c00000,  
        .size      = 0x01400000,  
    },  
    [6] = {  
        .name      = "userdata",  
        .offset    = 0x09000000,  
        .size      = 0x06fffffff,  
    },  
}
```



- Loady/loadb命令通过y-modem、kermit协议将文件通过串口下载到内存中

```
#loady 0x40000000
```

```
#loadb 0x40000000
```

- 使用USB命令下载（借助PC端fastboot.exe工具）

```
#fastboot
```

- 使用网络命令下载（使用交叉线连接电脑或接入同一局域网）先下载到内存再写到nand，下载时需要借助一个PC端TFTP工具(TFTP+DHCP\_Server/tftpd32.exe)

```
#ping 192.168.220.x(电脑主机)
```



➤ 下面以一个完整的嵌入式系统更新流程为例，讲解如何使用官方u-boot自带网络协议命令更新整个系统：

➤ 更新u-boot

```
#tftp 0x40000000 192.168.220.x:u-boot.bin
```

```
#nand erase 0x0 0x100000
```

```
#nand write 0x40000000 0x0 0x100000
```

➤ 更新内核zImage:

```
#tftp 0x40000000 192.168.220.x:zImage
```

```
#nand erase 0x600000 0x500000
```

```
#nand write 0x40000000 0x600000 0x500000
```

## ➤ 更新cramfs根文件系统

```
#tftp 0x40000000 192.168.220.x:rootfs.cramfs
```

```
#nand erase 0xE00000 0x6E00000
```

```
#nand write 0x40000000 0xE00000 0x6E00000
```

## ➤ 更新yaffs格式根文件系统

```
#tftp 0x40000000 192.168.220.x:rootfs.yaffs
```

```
#nand erase 0xE00000 0x6E00000
```

```
#nand write.yaffs 0x40000000 0xE00000 0xXXX
```

(0xXXX是rootfs.yaffs实际大小)

- 简易网络下载命令:
  - 更新u-boot (u-boot.bin)  
#update boot
  - 更新kernel (zImage)  
#update kernel
  - 更新yaffs (rootfs.yaffs)  
#update yaffs
  - 所有镜像统一更新  
#update image



## ➤ 系统启动方式

### ➤ 方式一:

```
#boot
```

### ➤ 方式二:

```
#nand read 0x40000000 0x600000 0x500000
```

```
#bootm 0x40000000
```

### ➤ 方式三:

#### ➤ 通过环境变量自定义脚本:

```
#setenv bk nand read 0x40000000 0x600000  
0x500000\;bootm 0x40000000
```

```
#run bk
```

- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程

- u-boot的每一个命令都是通过U\_BOOT\_CMD宏定义来实现的, 这个宏在include/command.h头文件中定义。

```
object - Source Insight - [Command.h (x:\work\platform\bootloader\uns  
#define U_BOOT_CMD(name, maxargs, rep, cmd, usage, help) \  
cmd_tbl_t __u_boot_cmd_ ## name Struct_Section = \  
{ #name, maxargs, rep, cmd, usage, help }
```

- 每一个命令定义了一个cmd\_tbl\_t结构体, 结构体包含的成员变量有:
  - 命令名称、最大参数个数、重复次数、命令执行函数、用法、帮助。

- 结构体包含命令的所有属性，如下：

```
struct cmd_tbl_s {  
    char    *name;           /* Command Name */  
    int     maxargs;         /* maximum number of arguments */  
    int     repeatable;      /* autorepeat allowed? */  
                                /* Implementation function */  
    int     (*cmd)(struct cmd_tbl_s *, int, int, char *[]);  
    char    *usage;          /* Usage message (short) */  
#ifdef CFG_LONGHELP  
    char    *help;           /* Help message (long) */  
#endif  
#ifdef CONFIG_AUTO_COMPLETE  
    /* do auto completion on the arguments */  
    int     (*complete)(int argc, char *argv[], char last_char, int maxv, char *cmdv[]);  
#endif  
};
```

- 所有命令都通过U\_BOOT\_CMD宏进行命令结构体的定义和初始化

- 为了便于管理命令数据结构，命令结构体被统一链接到一个数据段中：

```
#define Struct_Section __attribute__((unused,section (".u_boot_cmd")))
```

- 从控制台输入的命令都被送到common/command.c中的find\_cmd()函数解释执行，根据匹配输入的命令，从列表中找到对应的命令结构体，并调用其回调处理函数完成命令处理
- 命令响应的过程，就是命令的查找与回调函数处理的过程， find\_cmd()部分代码如下：



```
for (cmdtp = &__u_boot_cmd_start;  
    cmdtp != &__u_boot_cmd_end;  
    cmdtp++) {  
    if (strncmp (cmd, cmdtp->name, len) == 0) {  
        if (len == strlen (cmdtp->name))  
            return cmdtp; /* full match */  
  
        cmdtp_temp = cmdtp; /* abbreviated command ? */  
        n_found++;  
    }  
}
```

## ➤ 执行命令代码如下：

sunsp210\_u-boot\common\hush.c - Notepad++

```
1707             /* OK - call function to do the command */  
1708  
1709             rcode = (cmdtp->cmd)  
1710             (cmdtp, flag, child->argc-i, &child->argv[i]);  
1711             if ( !cmdtp->repeatable )  
1712                 flag_repeat = 0;
```

## ➤ 接下来，三步完成u-boot命令的添加：



1、在include/configs/unsp210.h中增加一项宏定义：

```
#define CONFIG_CMD_HELLOWORLD 1
```

2、并在common/文件夹下建立cmd\_helloworld.c文件

```
#include <common.h>
#include <command.h>
#ifdef CONFIG_CMD_HELLOWORLD
void helloworld (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
{
    int i=0;
    for(i=0;i<argc;i++)
    {
        printf("-----Hello World!\n");
        printf("argv[%d]=%s\n",i,argv[i]);
    }
}
U_BOOT_CMD(hello,3,2,helloworld,
            "hello command","sunplusedu add u-boot command!\n");
#endif
```

## 3、在common/Makefile中增加一项

```
COBJS-y += cmd_movi.o  
COBJS-y += cmd_android.o  
COBJS-y += cmd_updateimage.o  
COBJS-y += cmd_helloworld.o
```

增加一项

- make
- 重新下载u-boot.bin
- 在命令行输入help和hello命令查看结果

- Bootloader介绍
- s5pv210启动过程
- u-boot介绍
  - u-boot介绍及配置编译
  - 控制命令
  - 命令实现
  - 启动过程

- u-boot为标准的两阶段启动bootloader
  - 第一阶段为汇编代码，主要为初始化cpu硬件体系结构
  - 第二阶段为c程序代码，主要提供多种复杂命令并引导操作系统
  
- u-boot两阶段入口代码位置为：
  - 第一阶段位于：cpu/s5pc11x/start.S
  - 第二阶段位于：lib\_arm/board.c

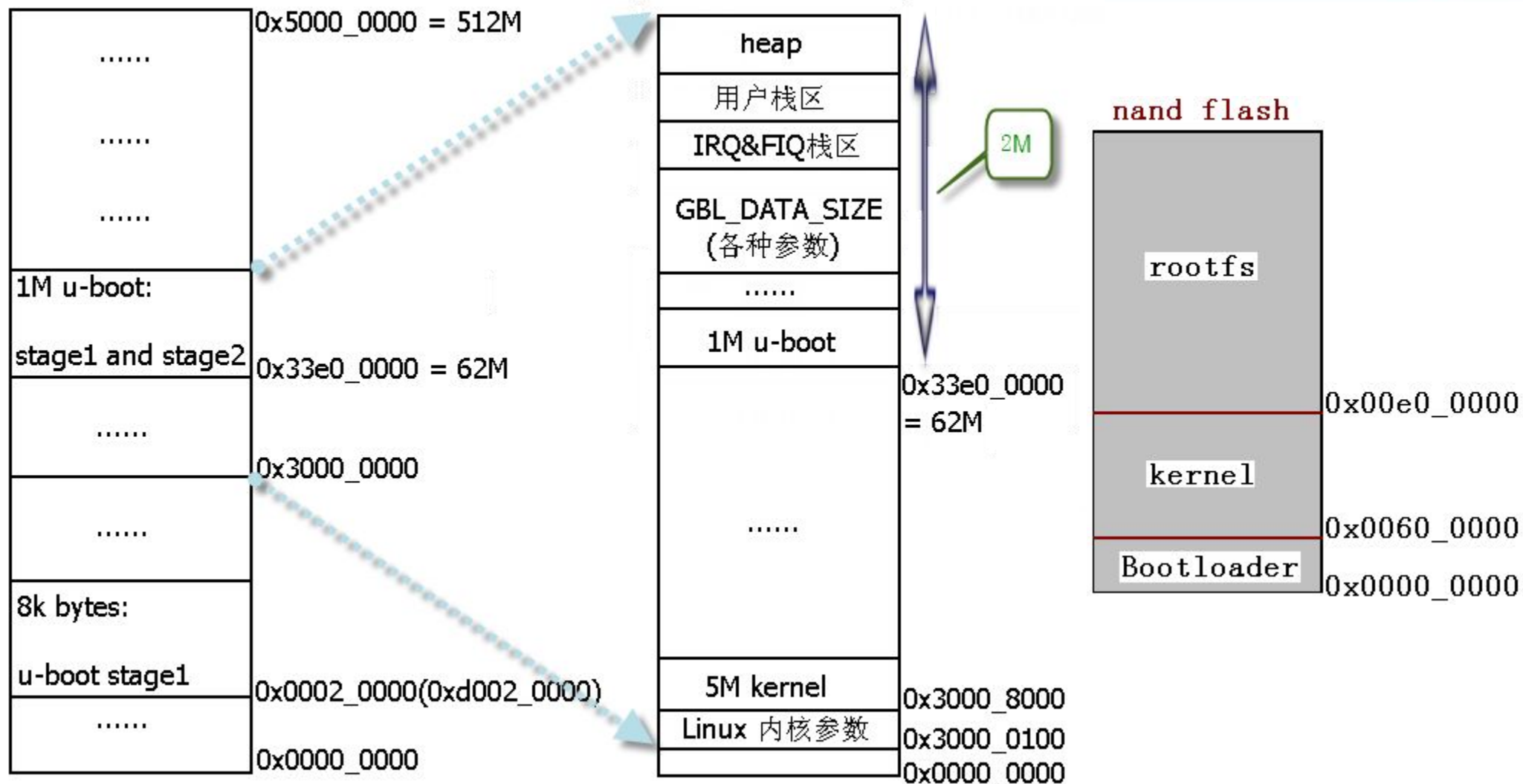
## ➤ u-boot第一阶段完成任务:

- 1、禁用看门狗、初始化系统时钟
- 2、设置异常向量表(用到中断的情况下设置)
- 3、动态内存控制器初始化配置
- 4、初始化调试指示灯(可选)
- 5、初始化UART, 用于开发调试(可选)
- 6、从NAND、NOR或SD卡中复制代码到SDRAM
- 7、跳转到start\_armboot, 进入Bootloader第二阶段



## ➤ u-boot 第二阶段完成任务:

- 1、初始化GPIO
- 2、初始化flash等存储设备
- 3、MMU初始化
- 4、堆初始化
- 5、MTD设备初始化
- 6、各类通信设备相关驱动初始化
- 7、环境变量和参数的加载及初始化
- 8、倒计时监听串口(进入命令模式或启动内核)
- 9、启动内核(拷贝内核镜像并跳转到内核入口)



u-boot 运行第一阶段内存分配

u-boot 运行第二阶段内存分配

## ➤ 任务:

- 根据前面提到的方法，添一个自定义命令
- 加入简易下载菜单，能进入和退出menu



凌阳教育官方微信：Sunplusedu

Tel: 400-705-9680, BBS: [www.51develop.net](http://www.51develop.net), QQ群: 241275518

