

TP-LINK®

MIPS BSP研究

SMB Switch

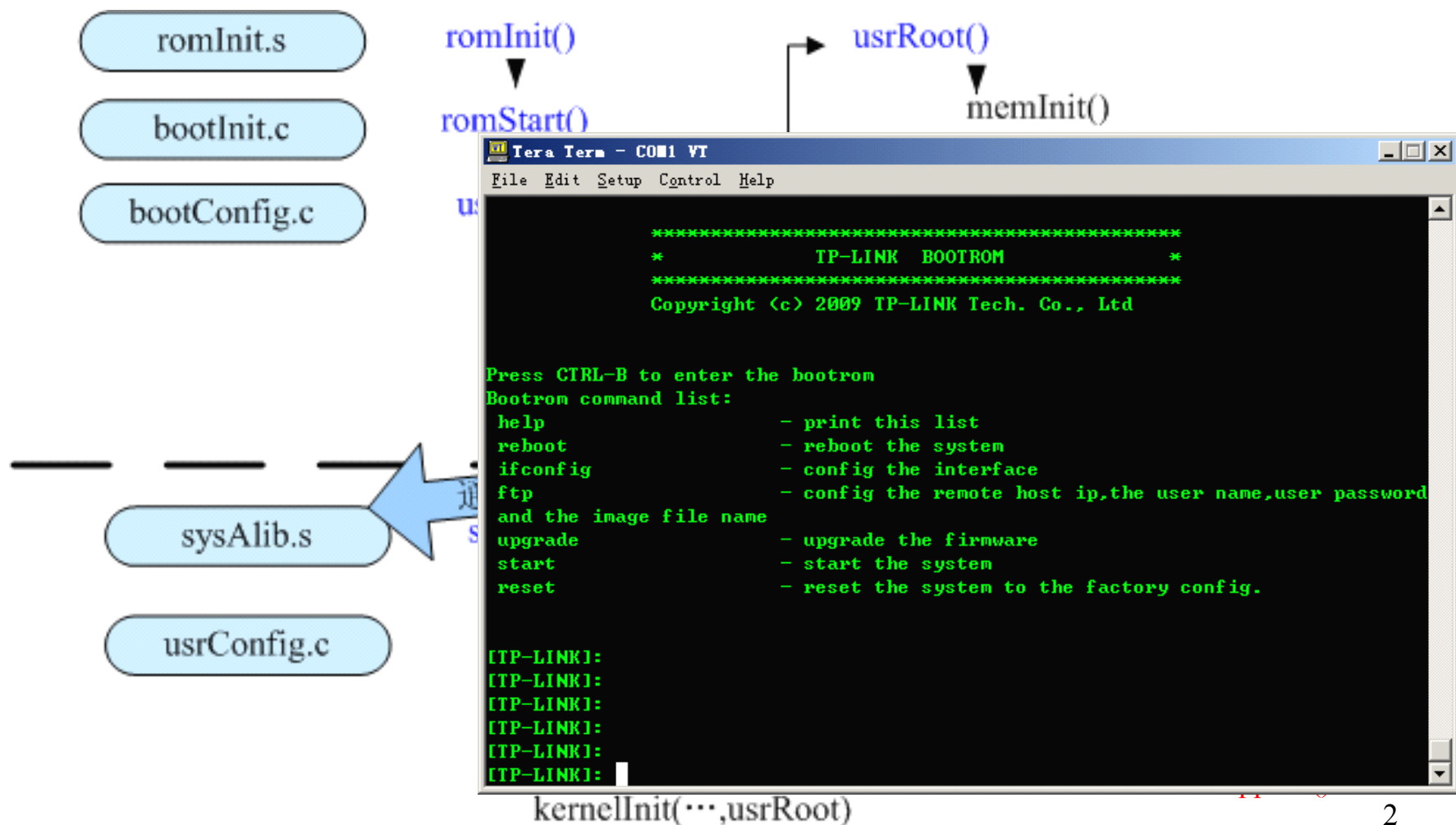
答辩者：聂勇

指导老师：郭次荣

MIPS BSP研究内容（1）

- **BootRom**的启动过程
 - **Makefile**及编译链接
 - 启动的顺序
- **BSP**中的各类驱动
 - **Cache**驱动
 - **Et**驱动
 - 文件系统驱动
 -

MIPS BSP研究内容（2）



答辯内容

启动过程

- Boot Rom启动过程?
- 上电执行的第一段代码romInit()函数的细节?

PIC

- 为什么需要PIC呢?
- Boot Rom中那些代码需要PIC?
- 如何实现PIC的呢?

Cache

- MIPS Cache的概念?
- Cache的初始化及其驱动?

启动过程

- Boot Rom启动过程
romInit(),romStart(),usrInit(),usr
Root()?
- 上电执行的第一段代码romInit()
函数的细节?

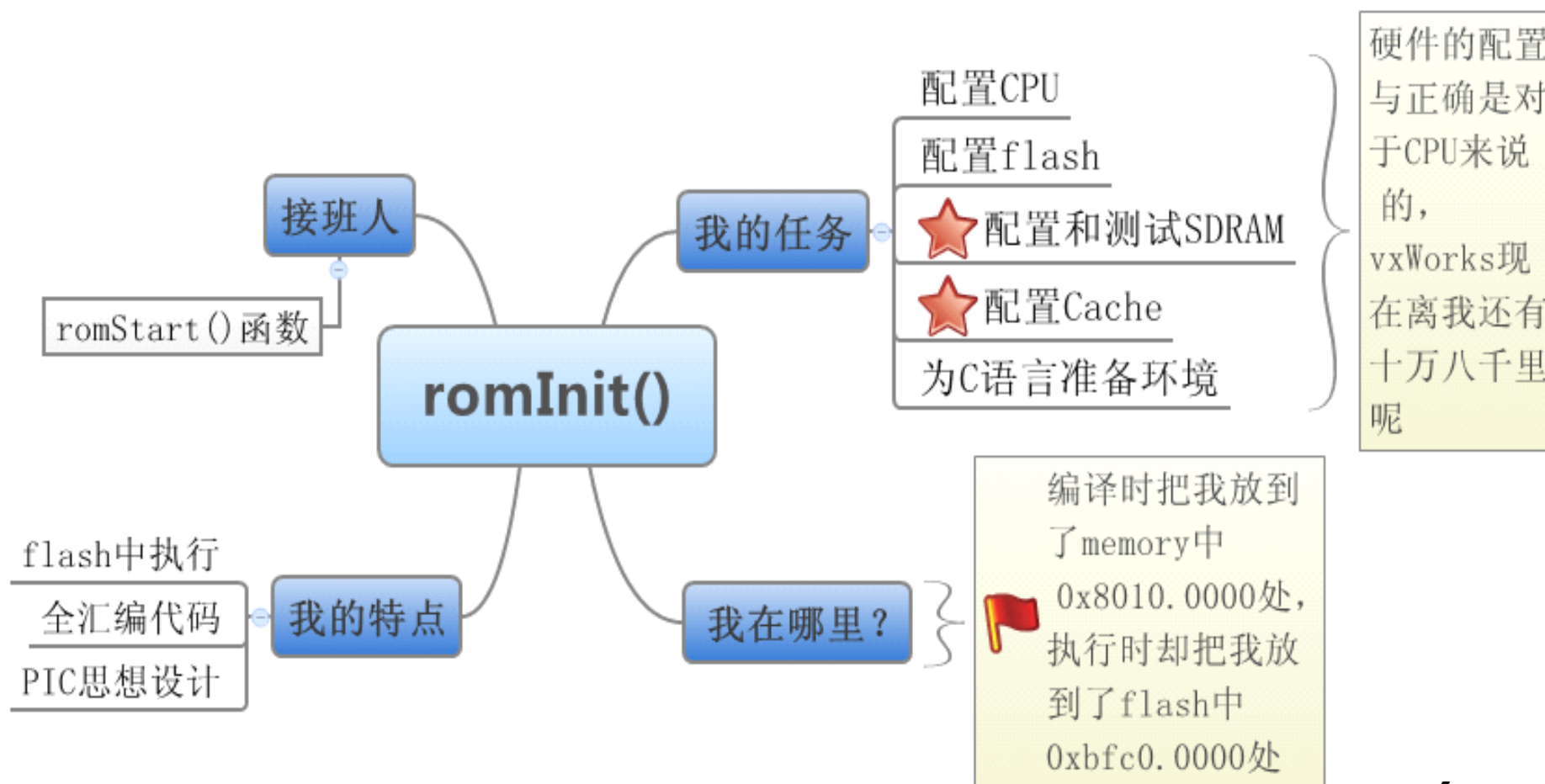
PIC

- 为什么需要PIC呢?
- Boot Rom中那些代码需要PIC?
- 如何实现PIC的呢?

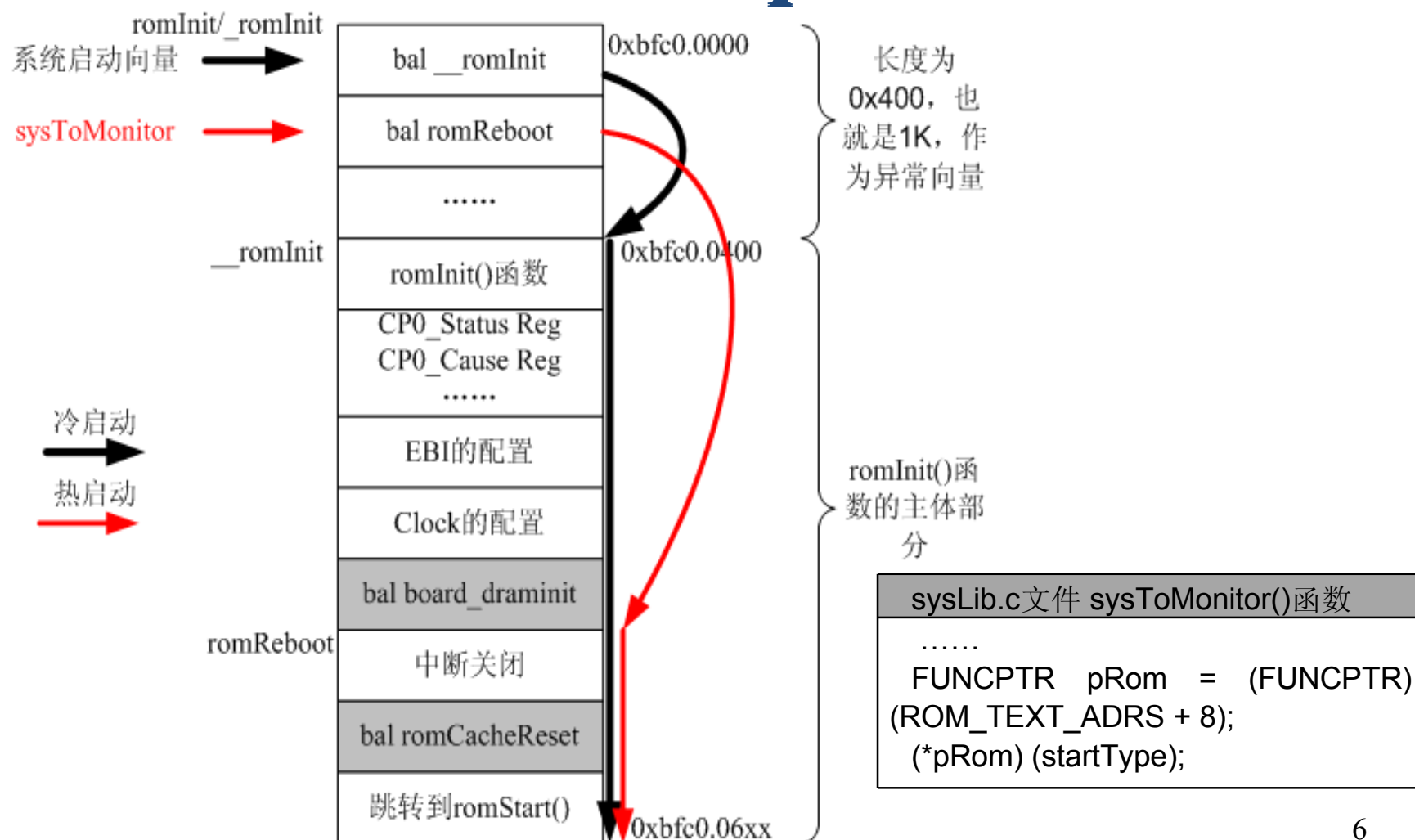
Cache

- MIPS Cache的概念?
- Cache的初始化及其驱动?

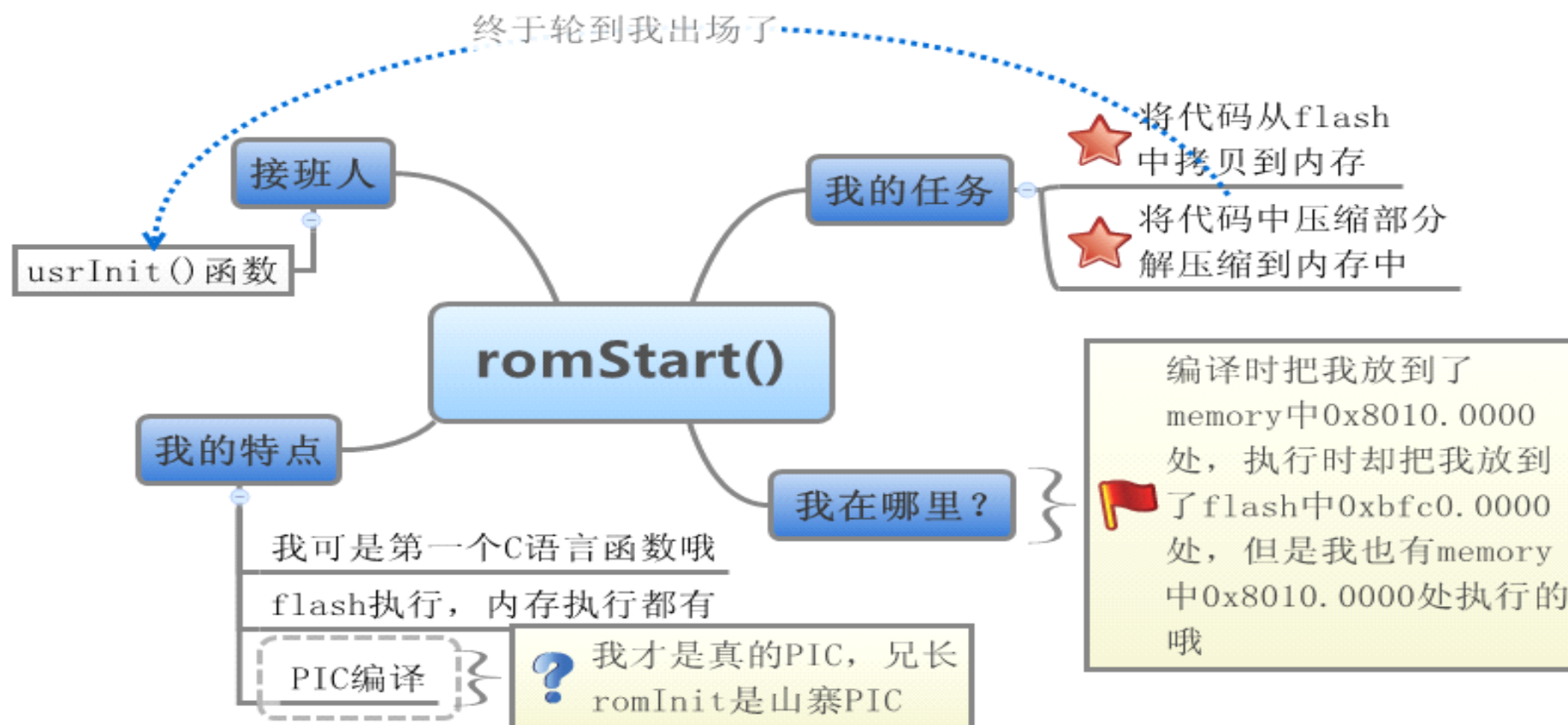
Boot Rom Boot Sequence - romInit



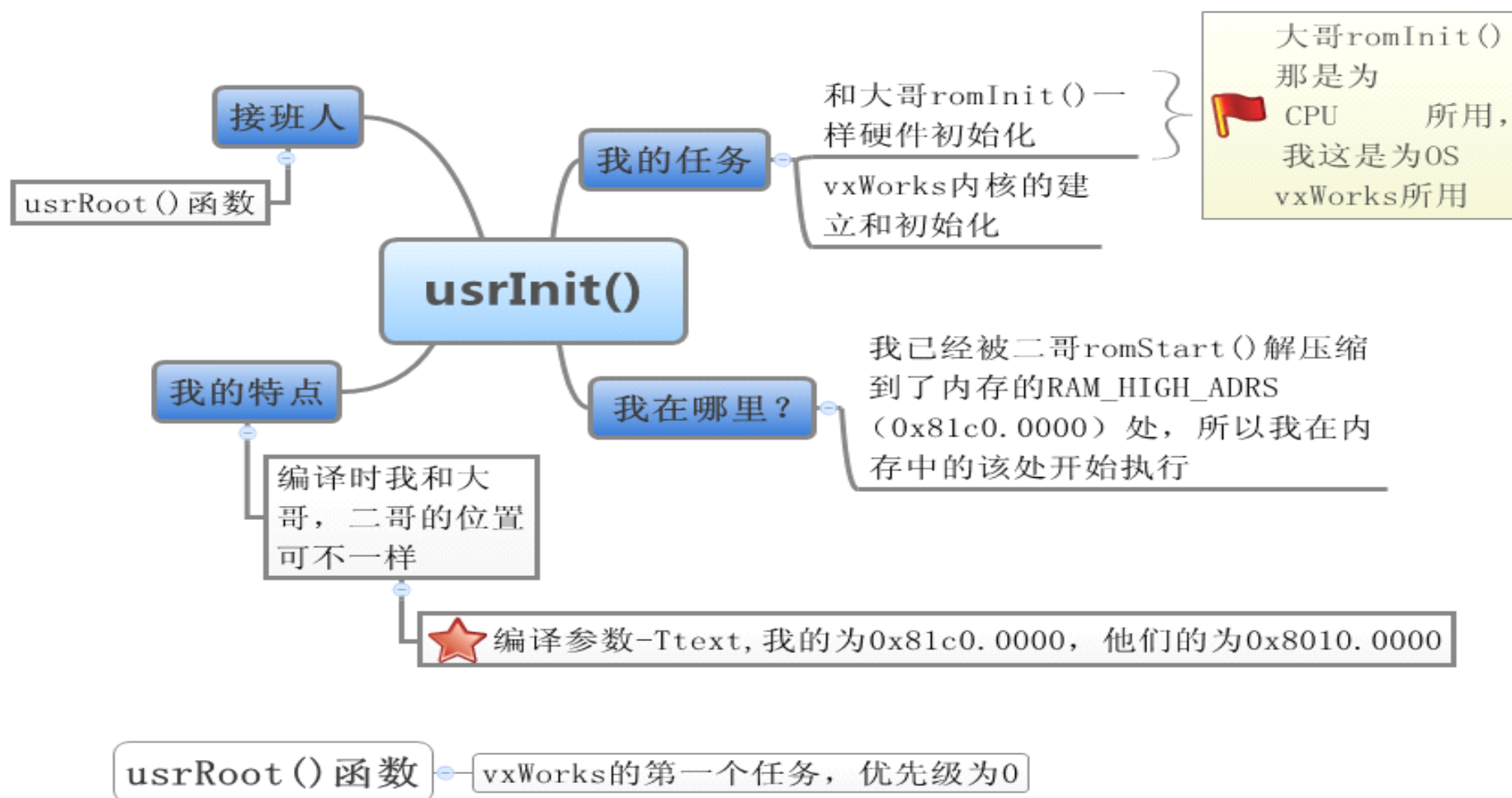
Boot Rom Boot Sequence - romInit



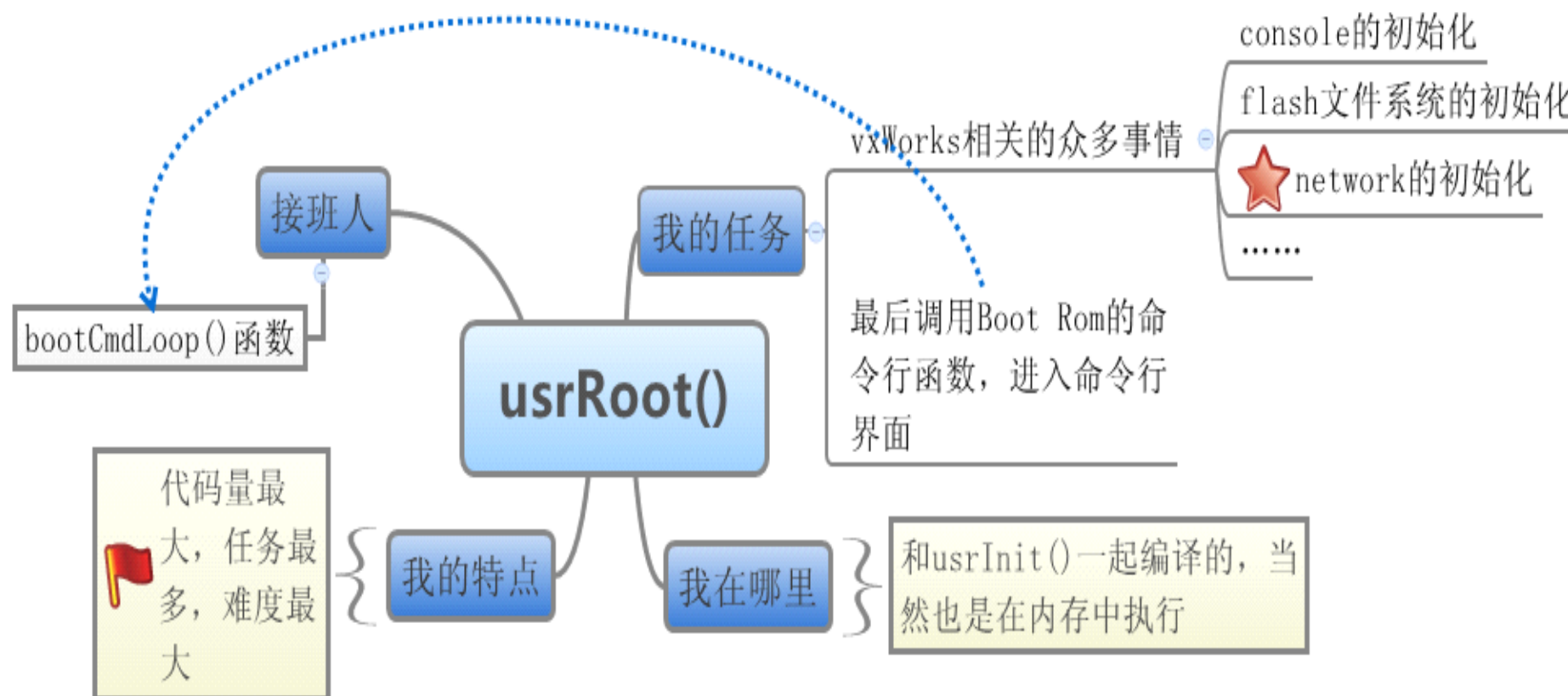
Boot Rom Boot Sequence - romStart



Boot Rom Boot Sequence - usrInit



Boot Rom Boot Sequence - usrRoot



启动过程

- Boot Rom启动过程?
- 上电执行的第一段代码romInit()函数的细节?

PIC

- 为什么需要PIC呢?
- Boot Rom中那些代码需要PIC?
- 如何实现PIC的呢?

Cache

- MIPS Cache的概念?
- Cache的初始化及其驱动?

PIC—Position Independent Code

• 为什么需要PIC

代码运行的地址（PC程序计数器）和链接时指定的文本段地址（-Ttext）不相同

• 哪些代码需要PIC

romInit()函数


romStart()函数

你们都是山寨的，只是使用了PIC的思想

```
E:\source\bcm4704_BSP_ny>readelfmips -l bootrom

Elf file type is EXEC (Executable file)  ld-mips -EB -X -N-e romInit
Entry point 0x80100000 -Ttext 80100000 .....
There are 1 program headers, starting at offset 52

Program Header:
  Type           Offset    VirtAddr    PhysAddr    FileSiz MemSiz  Flg Align
  LOAD           0x000080  0x80100000  0x80100000  0x5d470 0x770f0 RWE 0x40
```



PIC-编译

- 在编译时实现**PIC**

romStart()函数（bootInit.c文件）

-fpic参数：指示GCC产生地址无关代码

-fPIC参数对GOT大小没有限制

-fpic参数对硬件平台的依赖性更高

,

编译bootInit.c文件

```
.....  
ccmips -c -G 0 -mno-branch-likely -mips2 -EB  
-ansi -fno-builtin -O0 -Wall -l/h -l. -  
IE:\Tornado2.2.1-mips\target\config\all -  
IE:\Tornado2.2.1-mips\target/h -  
IE:\Tornado2.2.1-mips\target/h/wrn\coreip -  
IE:\Tornado2.2.1-mips\target/src/config -  
IE:\Tornado2.2.1-mips\target/src/drv -  
DCPU=MIPS32 -DTOOL_FAMILY=gnu -  
DTOOL=sfgnu -D_WRS_KERNEL -DMIPSEB -  
DSOFT_FLOAT -DMIPSEB -DSOFT_FLOAT -  
DBC56218 -fpic -msoft-float  
E:\Tornado2.2.1-mips\target\config\all\bootInit.c  
.....
```

PIC—RELOC宏

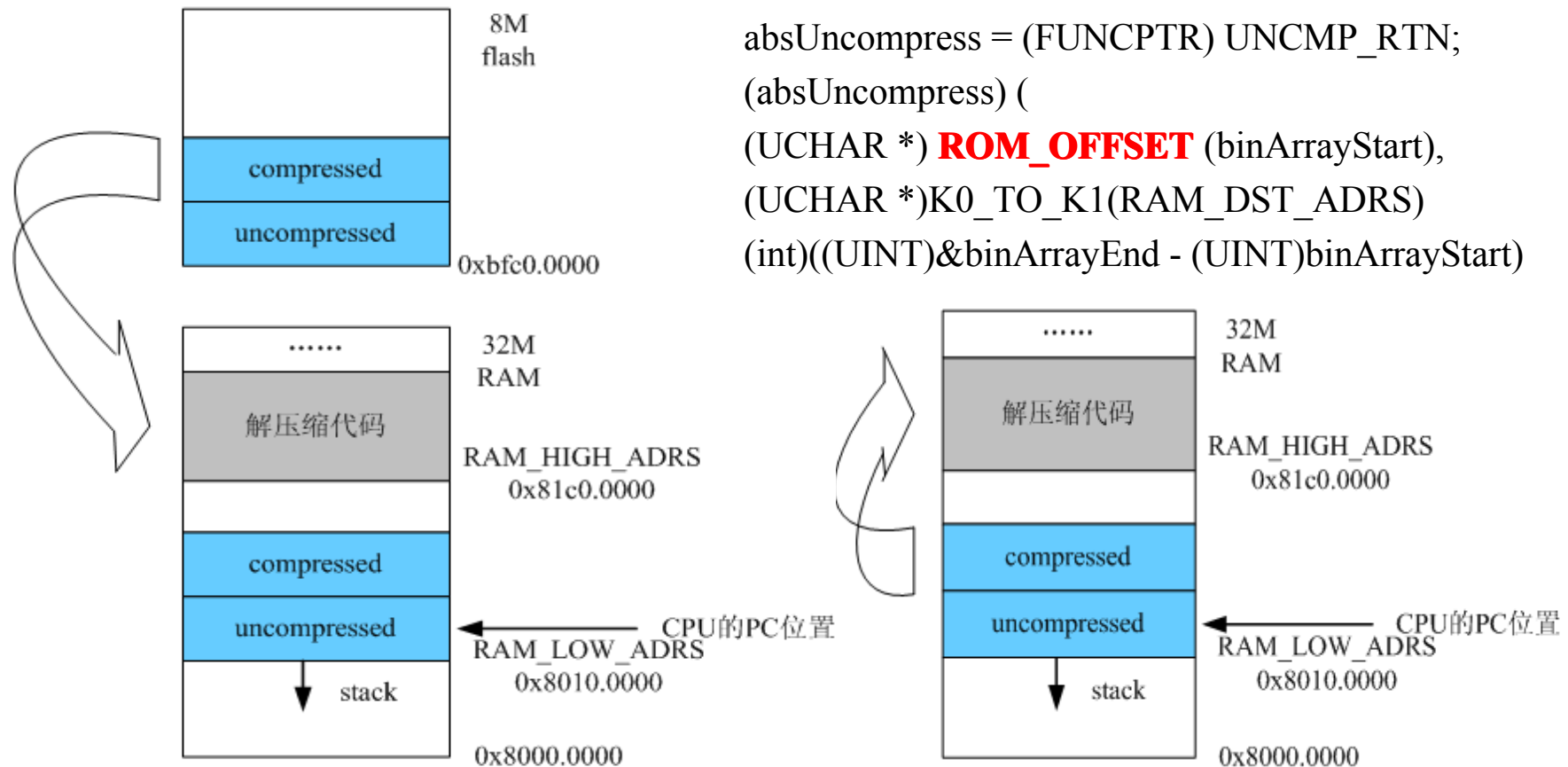
.....	
sync	→	sync
RELOC(t0, romStart)		bal 9f
jal t0		9:
		la t0 , romStart
		addu t0 , ra
		la ra , 9b
		subu t0 , ra
		jal t0

$$t0 = (\text{romStart 标签地址} - 9\text{标签 地址}) + ra$$

链接时指定的文本段地址（-Ttext） 决定

代码运行的地址（PC程序计数器）

PIC—代码解压缩



```
absUncompress = (FUNCPTR) UNCMP_RTN;
(absUncompress) (
    (UCHAR *) ROM_OFFSET (binArrayStart),
    (UCHAR *)K0_TO_K1(RAM_DST_ADRS)
    (int)((UINT)&binArrayEnd - (UINT)binArrayStart)
```

ROM_OFFSET(adr) = (((UINT)adr - (UINT)romInit) + ROM_TEXT_ADRS)

启动过程

- Boot Rom启动过程?
- 上电执行的第一段代码romInit()函数的细节?

PIC

- 为什么需要PIC呢?
- Boot Rom中那些代码需要PIC?
- 如何实现PIC的呢?

Cache

- MIPS Cache的概念?
- Cache的初始化及其驱动?

Cache

MIPS Cache软件设计

- MIPS Cache的初始化
- MIPS Cache的驱动

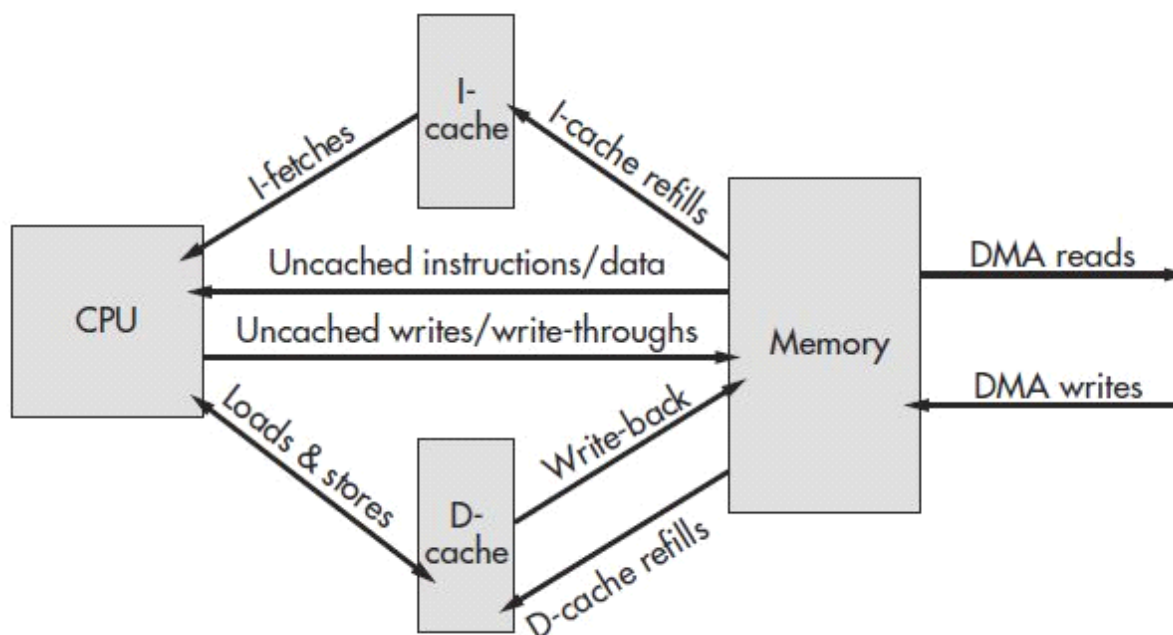
MIPS Cache的软硬件接口

- Cache相关的寄存器
- Cache指令

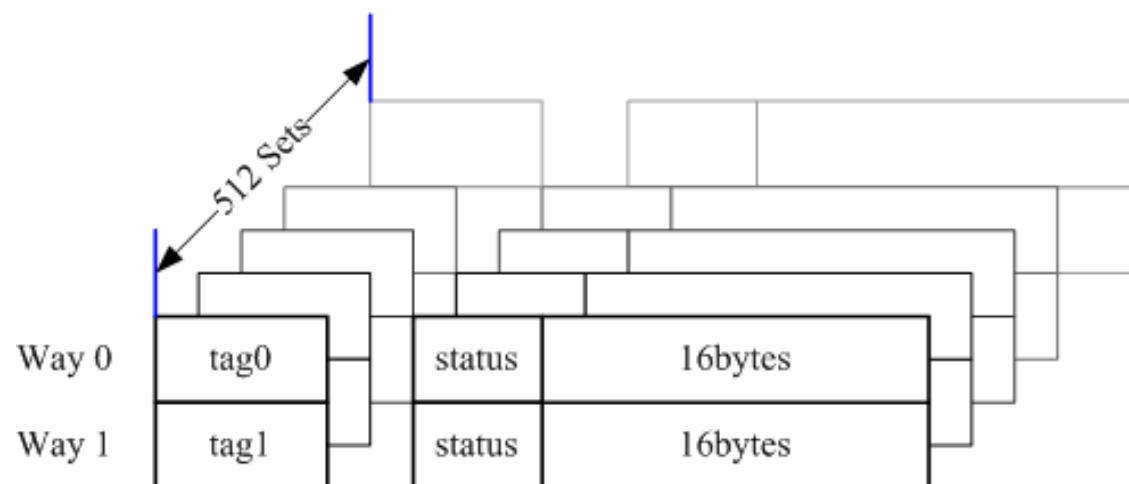
Cache的原理和硬件实现

- Cache的设计原理
- Cache参数

CPU、Cache、Memory、DMA



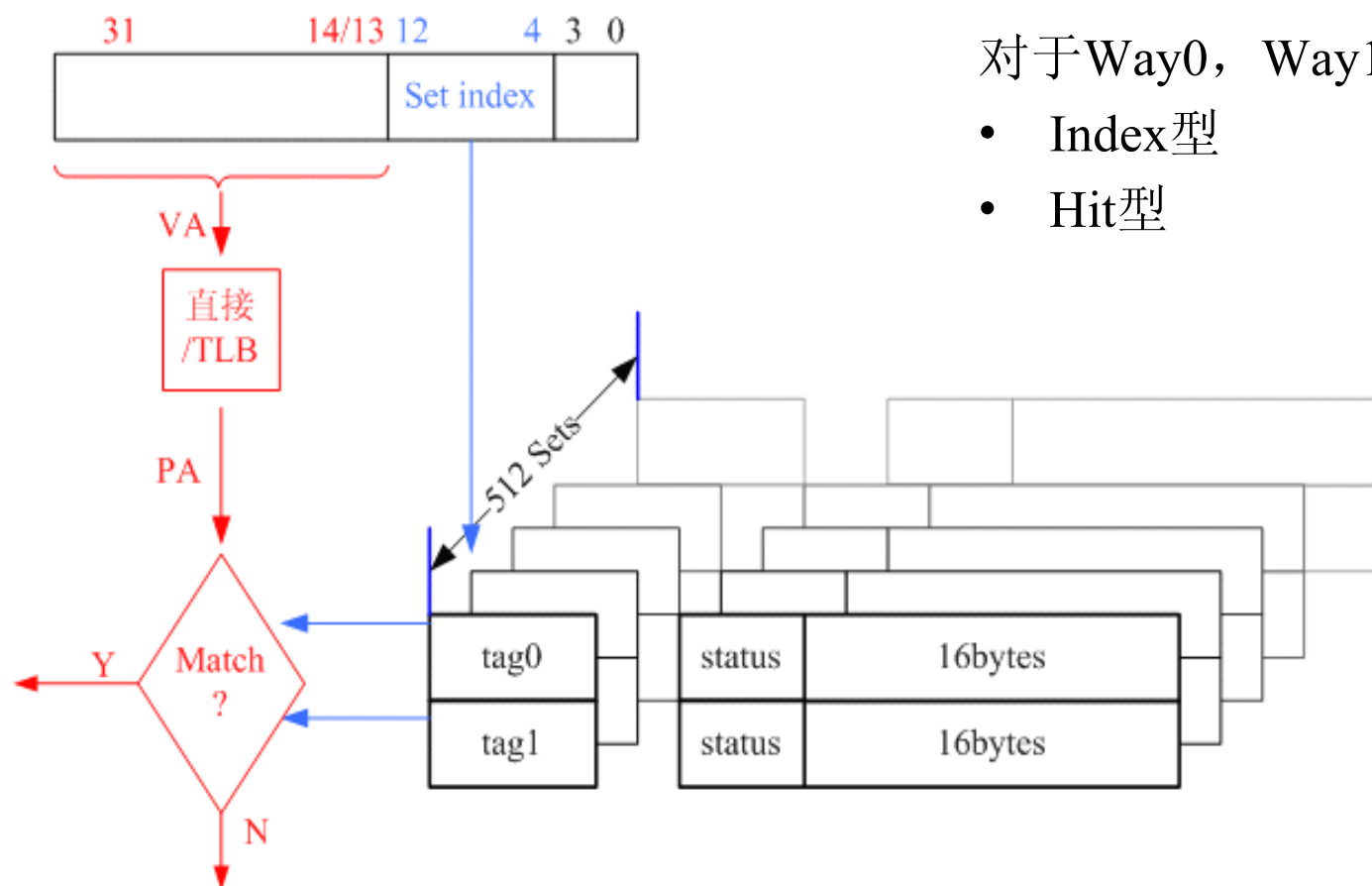
Cache的硬件结构



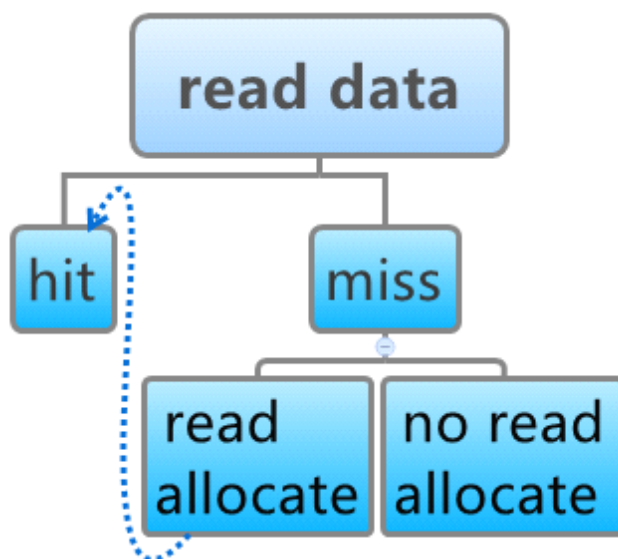
Cache的参数:

参数名称	示例
Cache大小	16KB
K路组相连	2-way associative
Line size	16bytes

Cache工作机制 (1)



Cache工作机制（2）



Read Allocate
No Read Allocate

Write Through
Write Back

Write Allocate
No Write Allocate



MIPS Cache软件接口（1）

- 寄存器

TagHi和TagLo用于暂存Cache行的Tag中的数据，都是32位CP0寄存器

- **Cache指令**

MIPS体系结构只引入了cache指令作为软件控制cache的统一接口

MIPS Cache软件接口（2）

cache ops, addr

ops在指令中占据5位，低2位指定Cache的类型，高3位指定执行的操作

ops低2位	Cache的类型
00	L1 I-Cache
01	L1 D-Cache
10	L2 Cache
11	L3 Cache

Ops高3位	执行的操作
000	Index Invalidate
001	Index Load Tag
010	Index Store Tag
011	
100	Hit Invalidate
101	Fill/Hit Writeback Invalidate
110	Hit Writeback
111	Fetch and Lock

MIPS Cache的初始化（1）

- 开启**Cache**功能

设置寄存器CP0 config 0、CP0 bcrn config 0等，使能I-Cache和D-Cache功能，开启Kseg0的Cache功能

- 初始化**Cache**

将未知状态的Cache行的Tag域置为0

MIPS Cache的初始化（2）

.....

```
/* Calc an address that will correspond to the last cache line */
```

```
addu  a3, a2, a0
```

```
subu  a3, a1
```

```
/* Loop through all lines, invalidating each of them */
```

```
1:
```

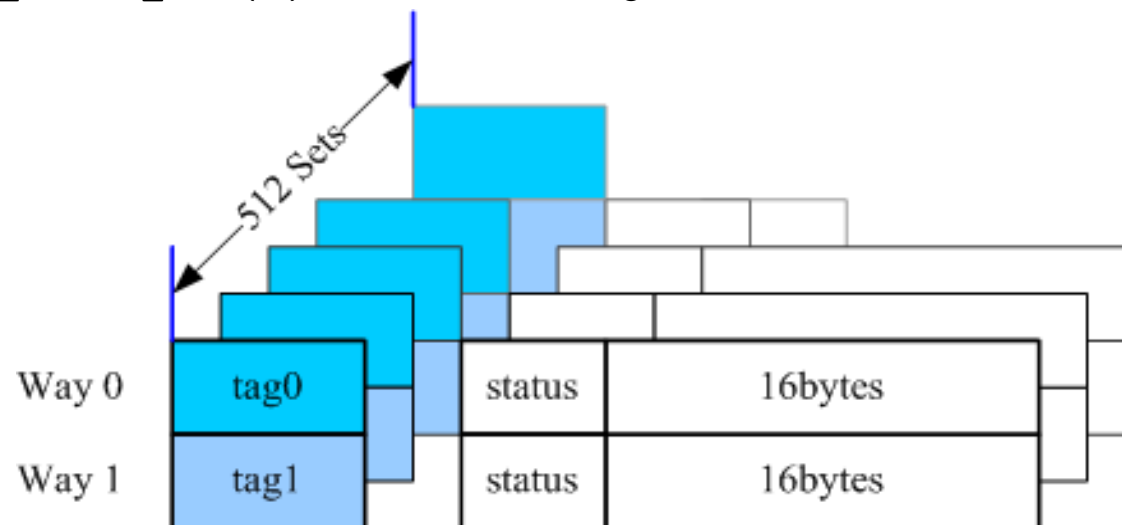
```
cache ICACHE_INDEX_STORE_TA 0(a2)
```

```
/* clear tag */
```

```
bne   a2, a3, 1b
```

```
addu  a2, a1
```

.....



MIPS Cache驱动 (1)

cacheAlib.s

函数	操作
FlushAll_Dcache	Ops(0x0,1)
FlushAll_Icache	Ops(0x0,0)
Invalidate_Icache(xx)	Ops(0x4,0)
Invalidate_Dcache(xx)	Ops(0x5,1)
Clear_Dcache(xx)	Ops(0x5,1)
Flush_Dcache(xx)	Ops(0x6,1)
Reset_caches	Ops(0x2,0) Ops(0x2,1)

0x4 I不需要回写

0x5 D需要回写

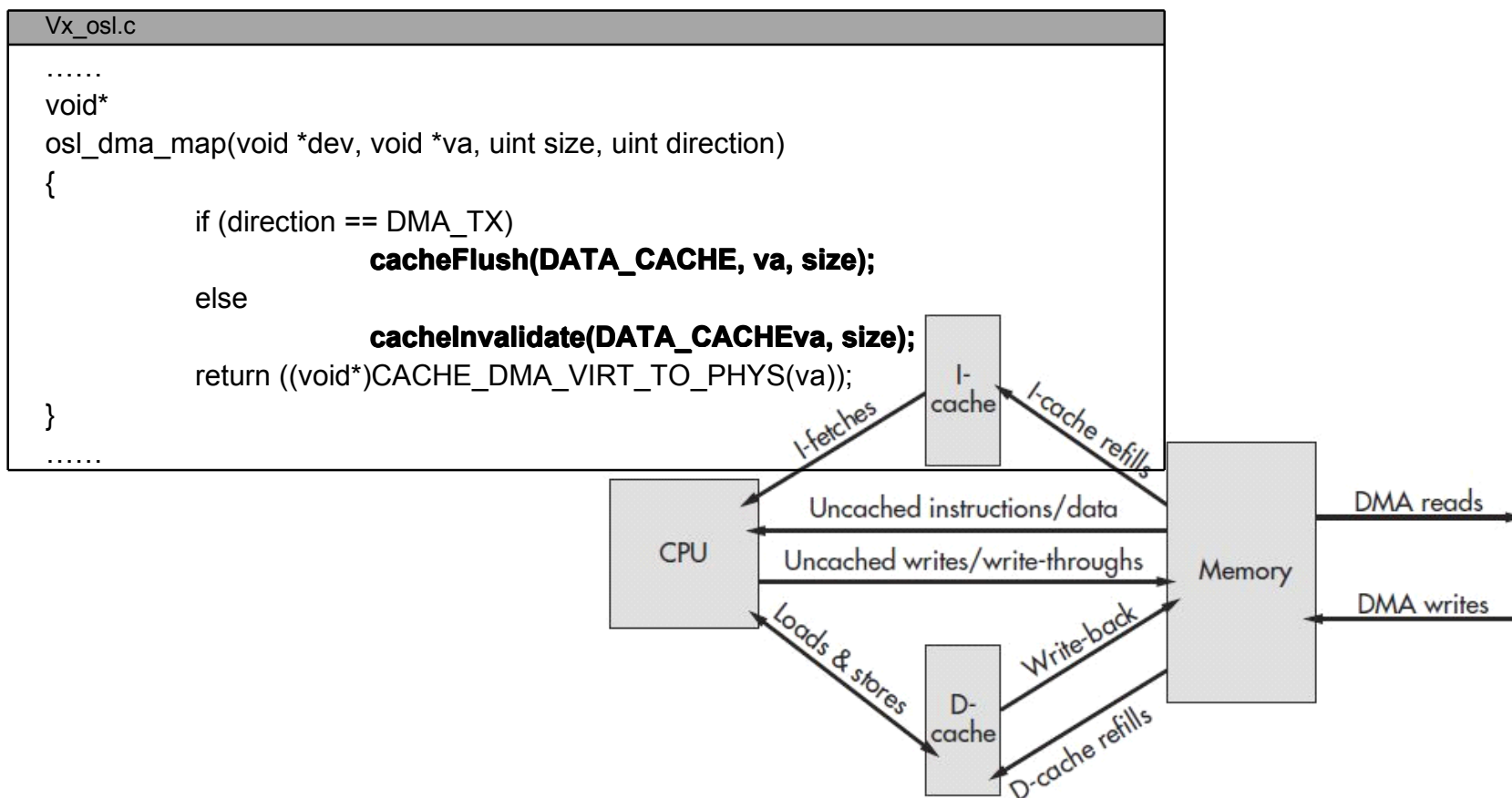
0x5: 回写后该行无效

0x6: 回写后改行有效

cachelib.c

函数	操作
cacheBcm47xxLibInit()	对cacheAlib.s的重新封装
cacheBcm47xxEnable	同上
.....	

MIPS Cache驱动 (2)



谢谢

答辩者：聂勇
指导老师：郭次荣

TP-LINK®

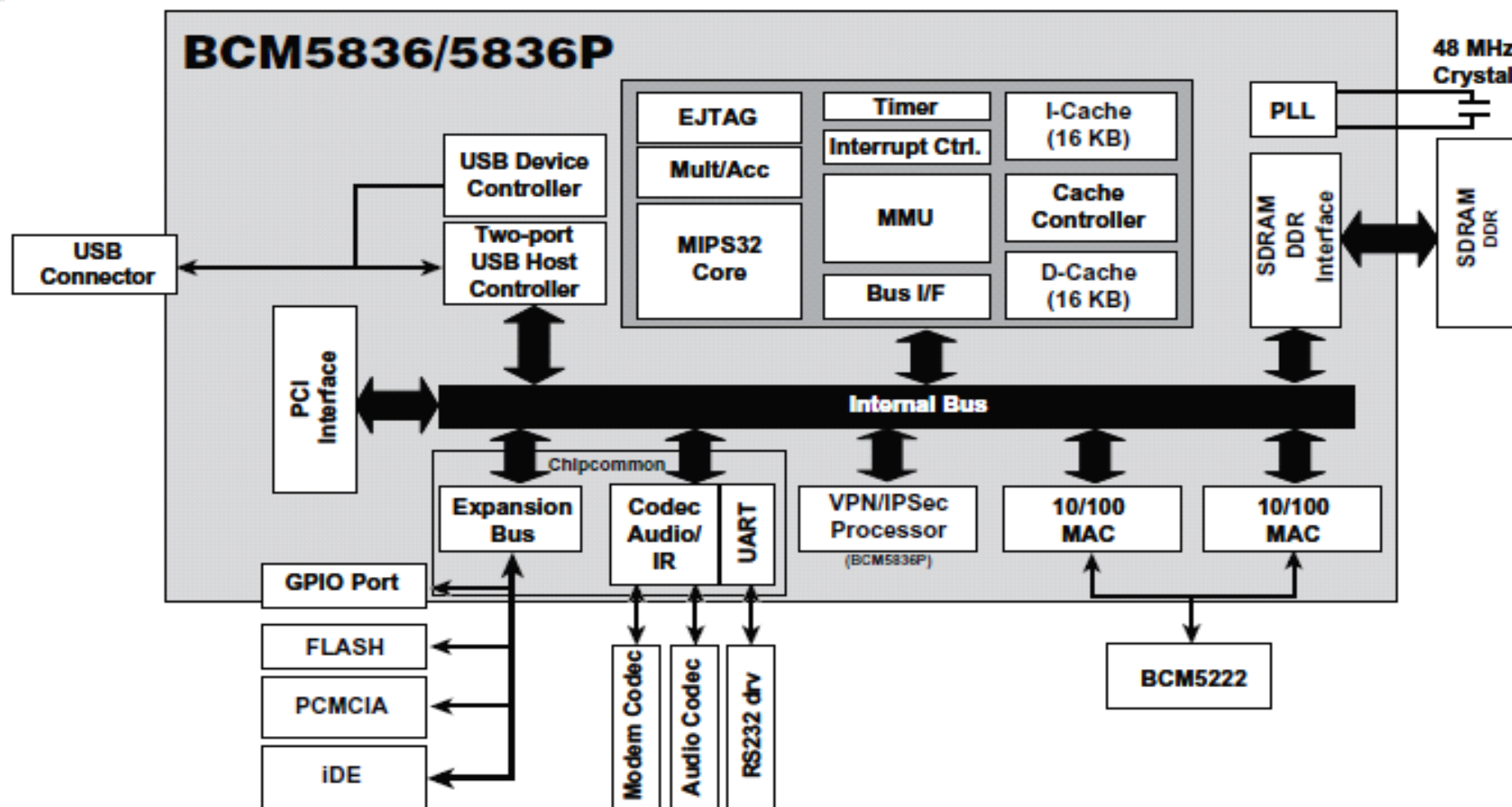
MIPS BSP研究-辅助

SMB Switch

答辩者：聂勇

指导老师：郭次荣

BCM5836 Block Diagram



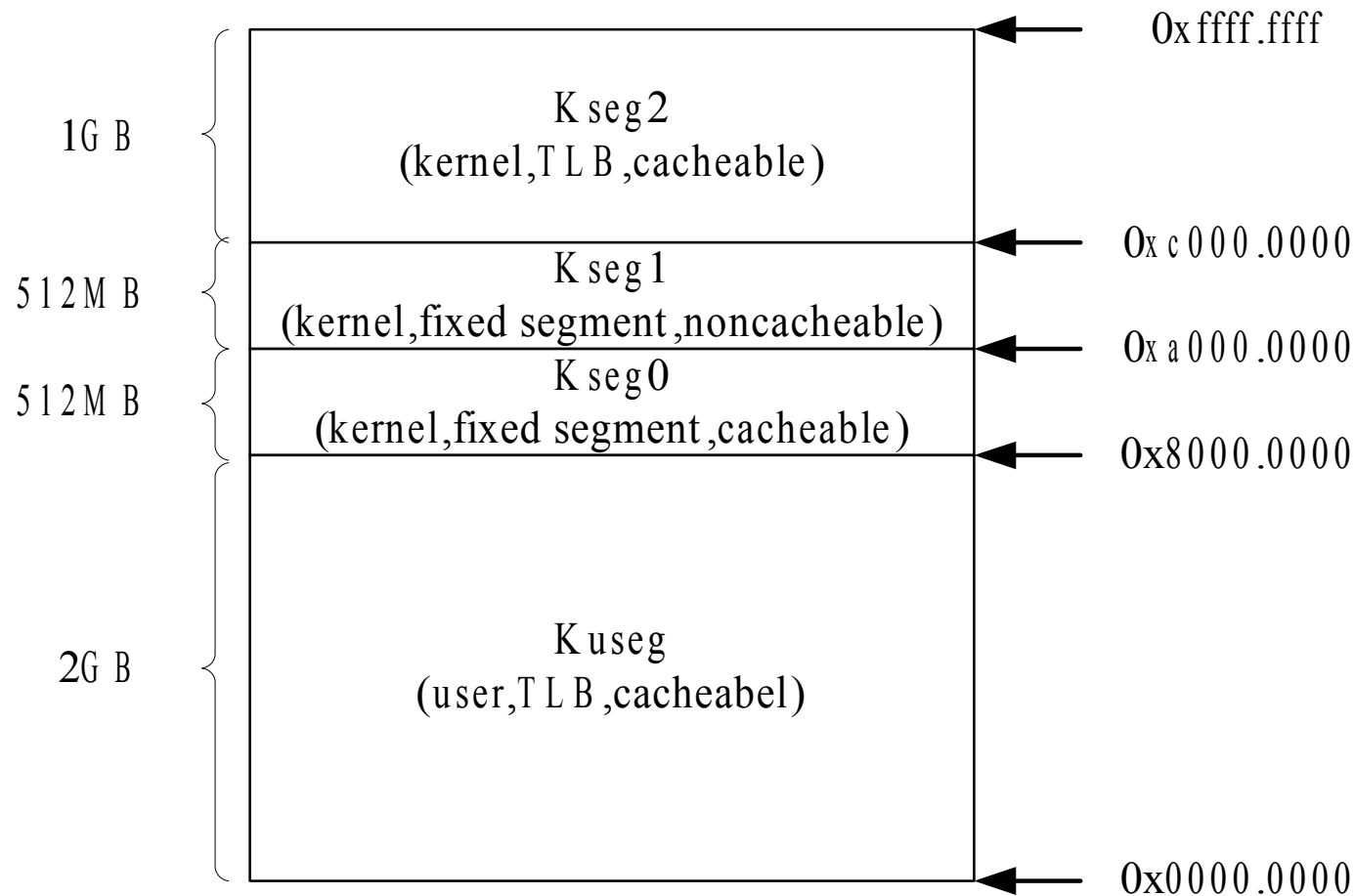
BCM5836 Address Space Map

Address	Description
0x00000000 ~ 0x07FFFFFFF	128 MB SDRAM, non-swapped region 1
0x08000000 ~ 0x0FFFFFFF	128 MB PCI window
0x10000000 ~ 0x17FFFFFFF	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

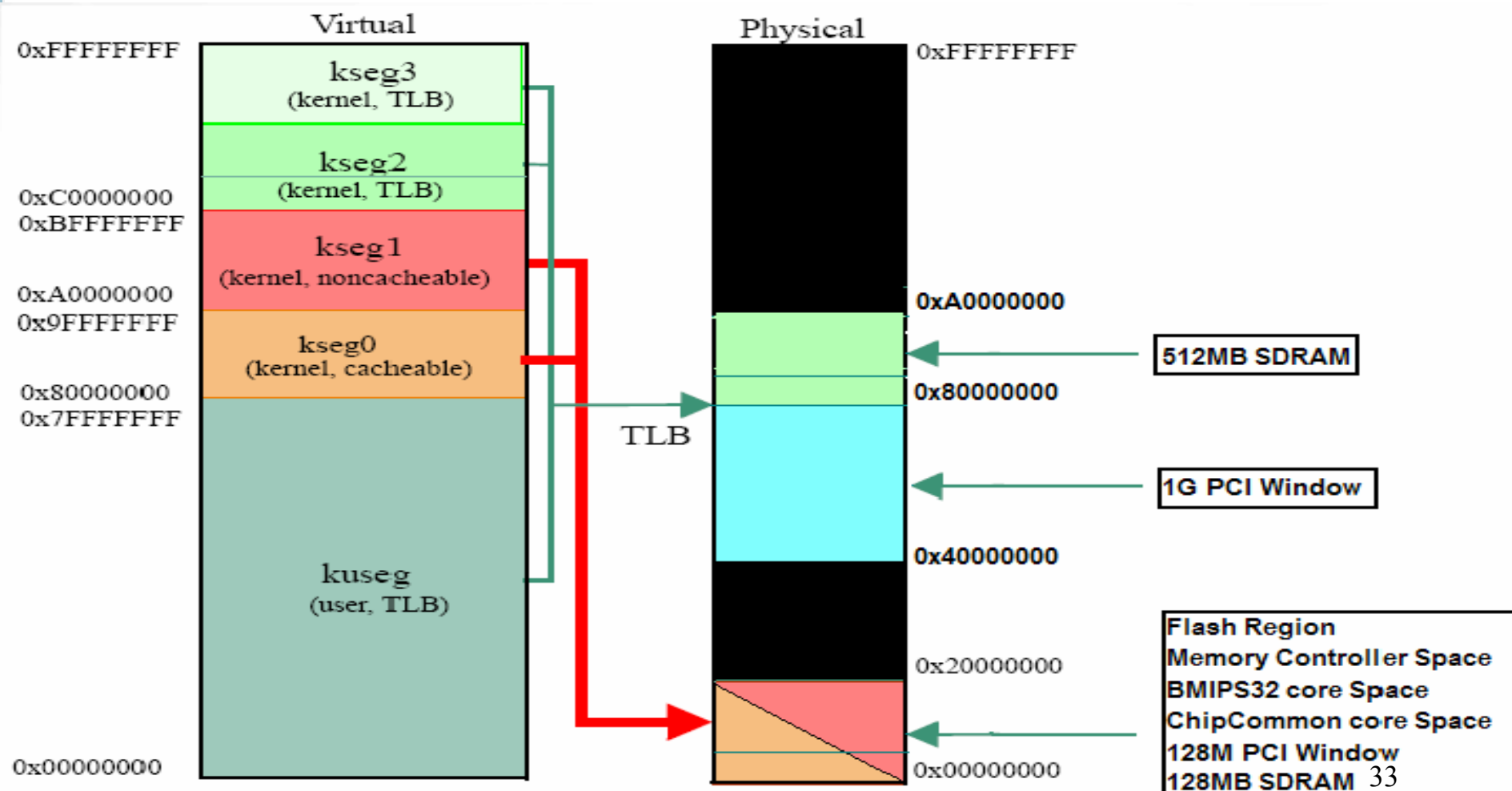
BCM5836 Address Space Map

0x18006000 ~ 0x18006FFF	Codec core
0x18007000 ~ 0x18007FFF	IPSec Core
0x18008000 ~ 0x18008FFF	Memory controller core
0x18009000 ~ 0x18009FFF	Reserved
0x1A000000 ~ 0x1BFFFFFFF	External interface address map (in Chipcommon)
0x1C000000 ~ 0x1DFFFFFFF	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

MIPS32 Virtual Address Space



BCM5836 VA&PA



BCM5836 flash&sdram

