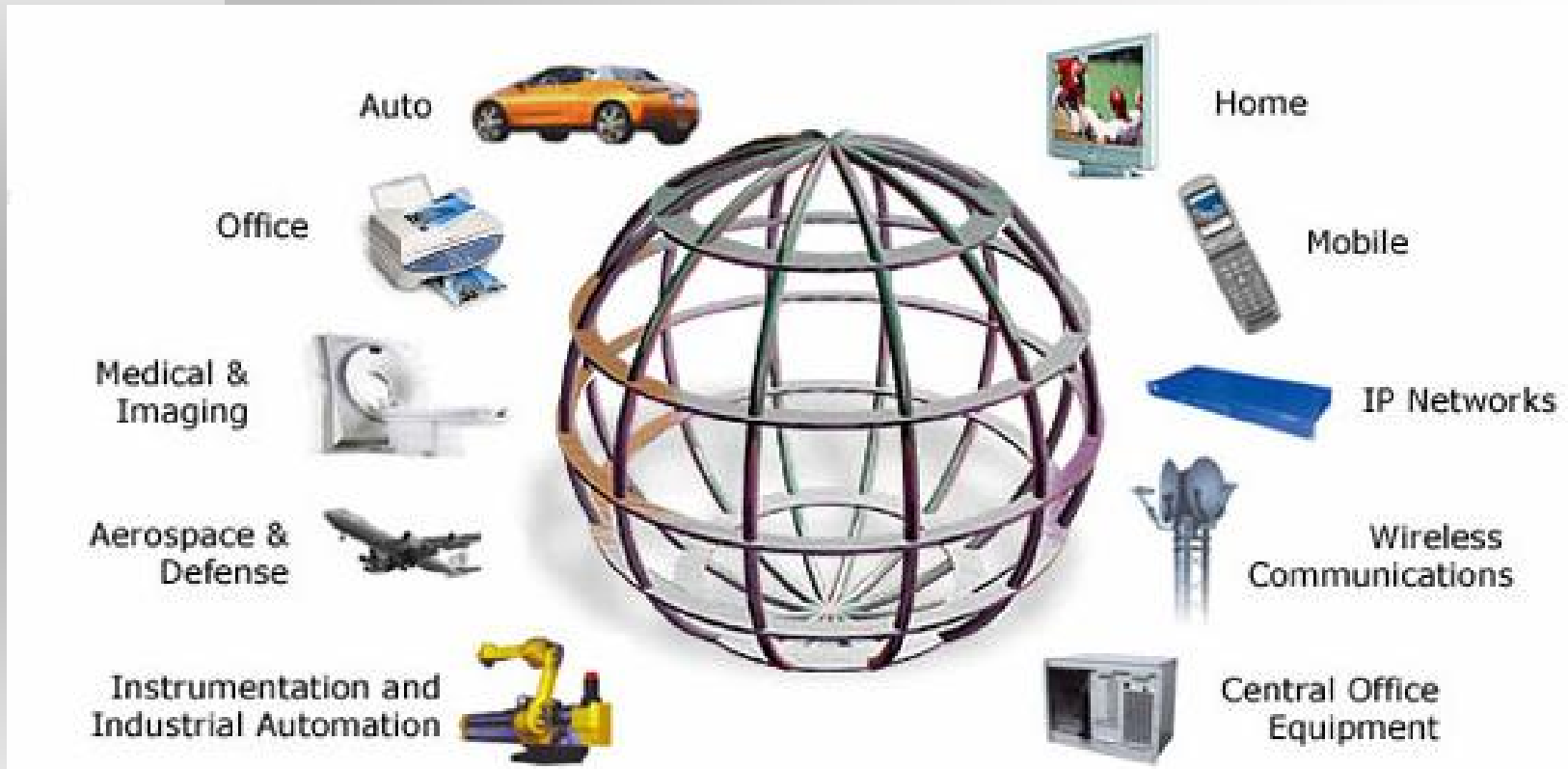
嵌入式系统引导程序开发

www.farsight.com.cn

华清远见

Linux Market



FARSIGHT

今天的内容

- 体系结构开发与引导程序初始化
- 引导程序功能与内核加载
- 引导程序移植与体系结构
  - 以上内容均以arm体系结构、u-boot为例

FARSIGHT

华清远见

嵌入式系统定义

- 嵌入式系统是以应用为中心，以计算机技术为基础，并且软硬件可裁剪，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统

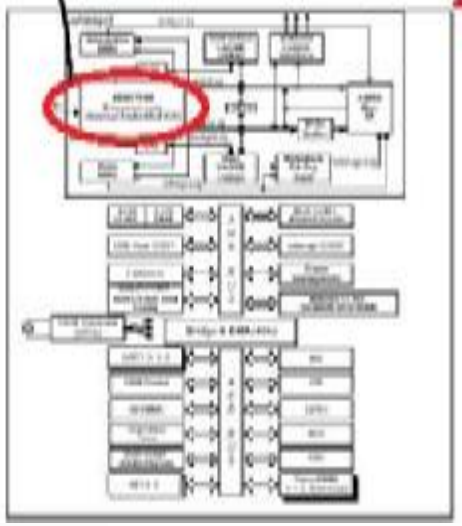
FARSIGHT

嵌入式系统组成（硬件）

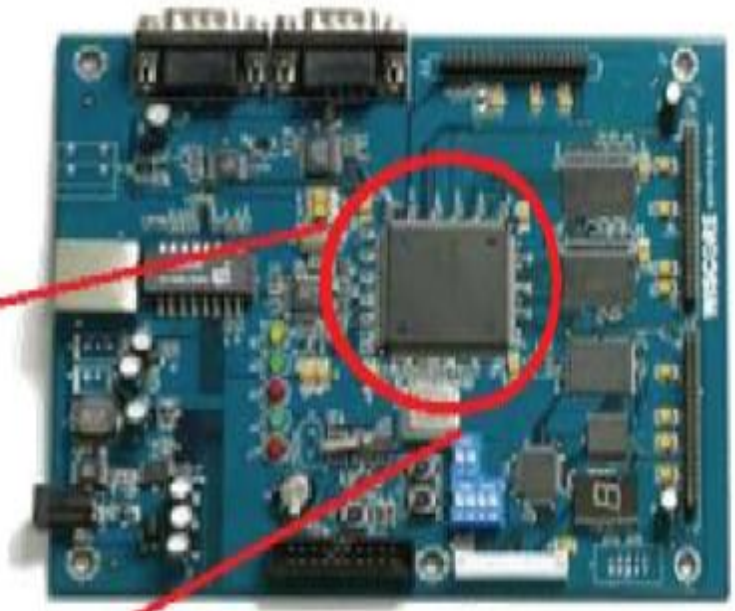
ARM

MIPS TECHNOLOGIES

SH3 INC.

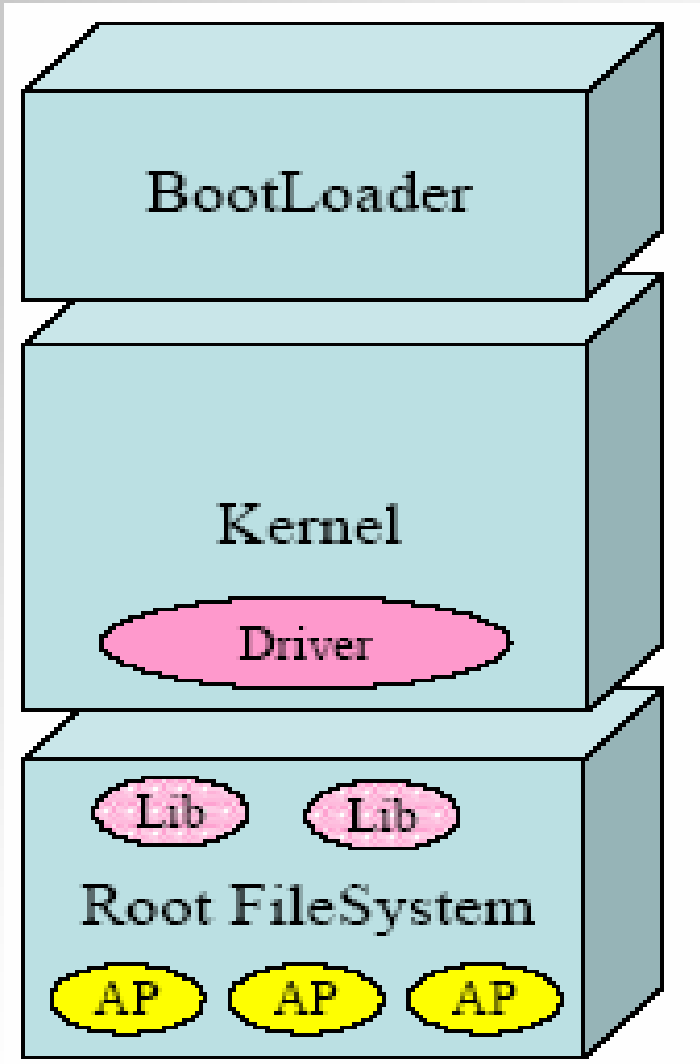


- CPU
- UART
- Memory Interface
- USB Device
- Timer
- DMA
- ……



FARSIGHT

嵌入式系统组成（软件）



开源的Bootloaders

Bootloader	Monitor	Description	x86	ARM	PowerPC
LILO	No	Main disk bootloader for Linux	Yes	No	No
GRUB	No	GNU's successor to LILO	Yes	No	No
Loadlin	No	Loads Linux from DOS	Yes	No	No
BLOB	No	Loader from the LART hardware project	No	Yes	No
U-boot	Yes	Universal loader	Yes	Yes	Yes
RedBoot	Yes	eCos-based loader	Yes	Yes	Yes

FARSIGHT

ARM Bootloaders

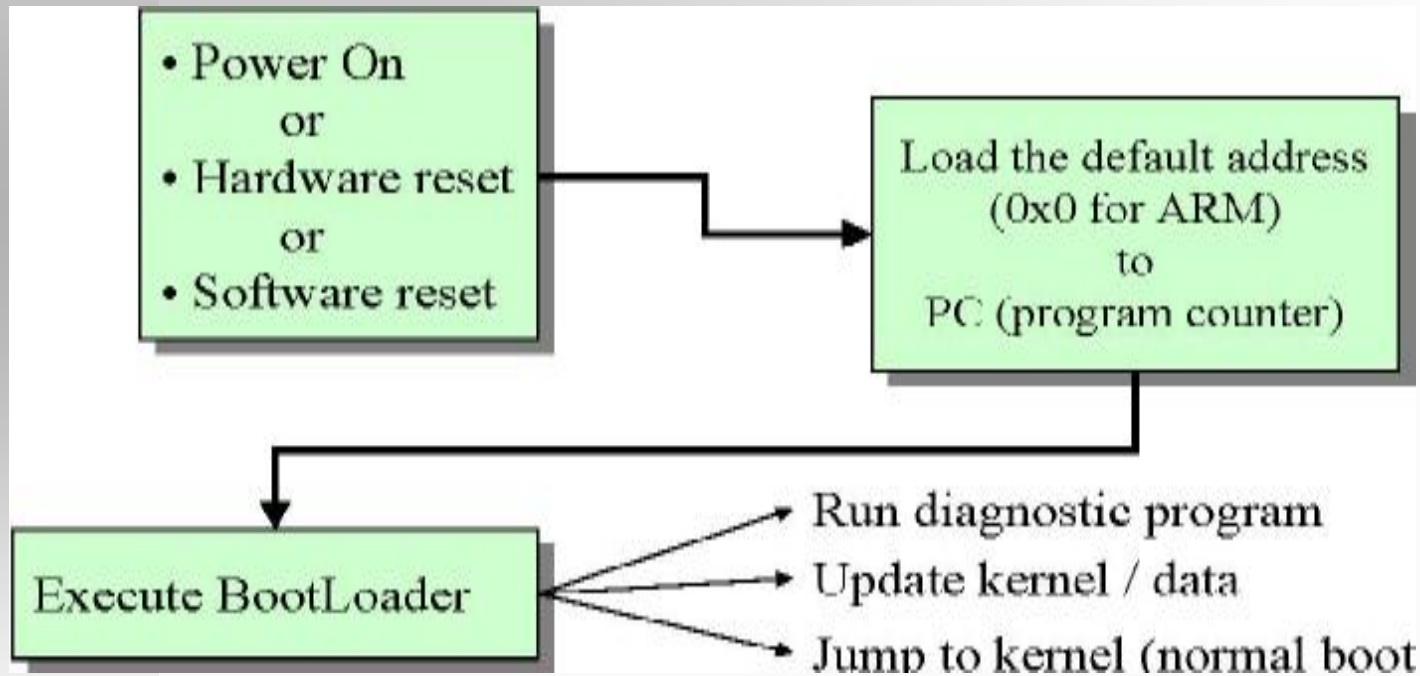
- U-Boot是常用的ARM bootloader
  - Armboot和ppcboot加入到u-boot中
  - 支持arm720, arm920, arm926, sa1100, xscale
  - http://armboot.sourceforge.net/
  - http://sourceforge.net/projects/u-boot
- Blob
  - Blob最早是为 LART项目开发的bootloader
  - Blob被移植到其他许多ARM平台
  - http://www.lart.tudelft.nl/lartware/blob
- Redboot
  - Redboot也被用在许多arm平台的 bootloader

FARSIGHT

华清远见

引导程序（bootloader）

- 存储在target的ROM或 flash中（地址为0x0），包含第一条可执行指令



FARSIGHT

Bootloader任务（1/2）

- 初始化处理器以及外设的硬件资源配置，CPU片内和片外设备，例如正确配置SDRAM控制器
- 初始化 I/O 芯片，可能有的设备：
  - 串口，报告Bootloader成功/失败
  - 网络或者Flash接口，引导操作系统
- 执行系统自检，报告检测结果
- 通过用户命令行提供特定应用程序

Bootloader任务（2/2）

- 使用TFTP协议从网口接收(或者xmodem协议从串口接收)操作系统镜像文件到RAM
  - 将镜像烧写到flash中，重启后负责找到该镜像、解压到RAM中，并跳转到解压位置处执行
  - 直接跳转到RAM处执行该镜像

FARSIGHT

U-boot演示

FARSIGHT

体系结构开发与引导程序初始化

FARSIGHT

ARM微处理器简介

- ARM（Advanced RISC Machines）
  - 32位固定指令长度，指令类型少
  - 37个32位寄存器堆用于所有目的
  - Load store结构，专门存储器指令
  - 硬连接指令译码，单周期执行，流水线
  - 所有指令可根据前面执行结果决定是否被执行，从而提高指令执行效率
  - 可用加载/存储指令批量传输数据，以提高数据传输效率
  - 可在一条数据处理指令中同时完成逻辑处理和移位处理
  - 在循环处理中使用地址自动增减来提高运行效率

FARSIGHT

寄存器组织


ARM状态下的通用寄存器与程序计数器
System & User
FIQ
Supervisor
About
IRG
Undefined
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15 (PC)
R8_fiq
R9_fiq
R10_fiq
R11_fiq
R12_fiq
R13_fiq
R14_fiq
R13_svc
R14_svc
R13_abt
R14_abt
R13_irq
R14_irq
R13_und
R14_und
ARM状态下的程序状态寄存器
CPSR
SPSR_fiq
SPSR_svc
SPSR_abt
SPSR_irq
SPSR_und
= 分组寄存器

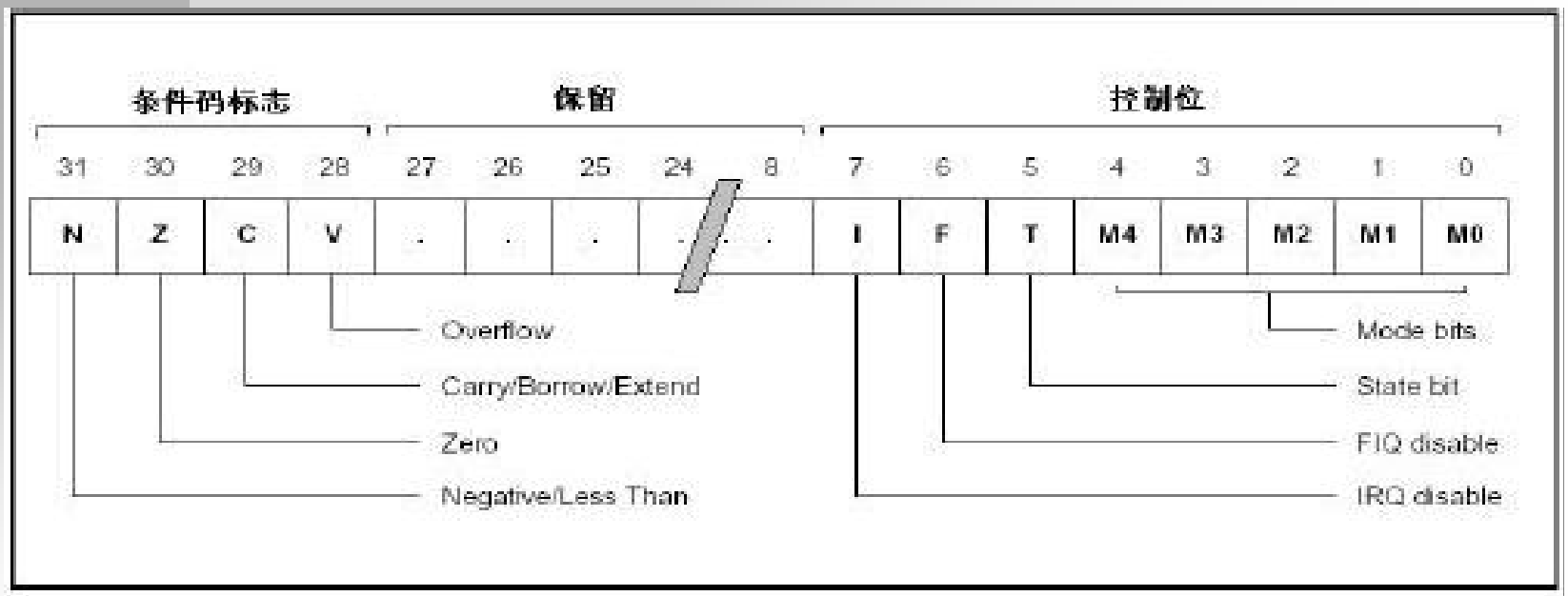
寄存器组织

- 通用寄存器包括R0—R15，可分为三类：
  - 未分组寄存器R0—R7
  - 分组寄存器R8—R14
    - 寄存器R13常用作堆栈指针
    - R14也称作子程序连接寄存器LR
    - BL指令时， R15（PC）拷入R14
  - R15程序计数器 (PC)
  - 综合使用R13,R14,R15：
    - BL Func1
    - 在子程序Func1入口处将R14入栈:
      - STMFD SP！,{<Regs>,LR}
    - 子程序返回:
      - LDMFD SP！,{<Regs>,PC}

寄存器组织

寄存器R16(CPSR)

- CPSR可在任何运行模式下被访问，每一模式又有一个专用物理状态寄存器SPSR，当异常发生时，SPSR用于保存/恢复CPSR



指令系统分类

- **数据处理**
  - 数据处理指令
  - 乘法指令与乘加指令
  - 移位指令（操作）
- **数据传送**
  - 加载/存储指令
  - 批量数据加载/存储指令
  - 数据交换指令
- **控制流指令**
  - 跳转指令
- **其他杂项**
  - 程序状态寄存器访问指令
  - 协处理器指令
  - 异常产生指令
- **配合多种寻址方式：**
  - 立即数寻址、寄存器寻址、寄存器间接寻址、基址变址寻址、相对寻址、多寄存器寻址、堆栈寻址等

FARSIGHT

ARM GAS汇编例子

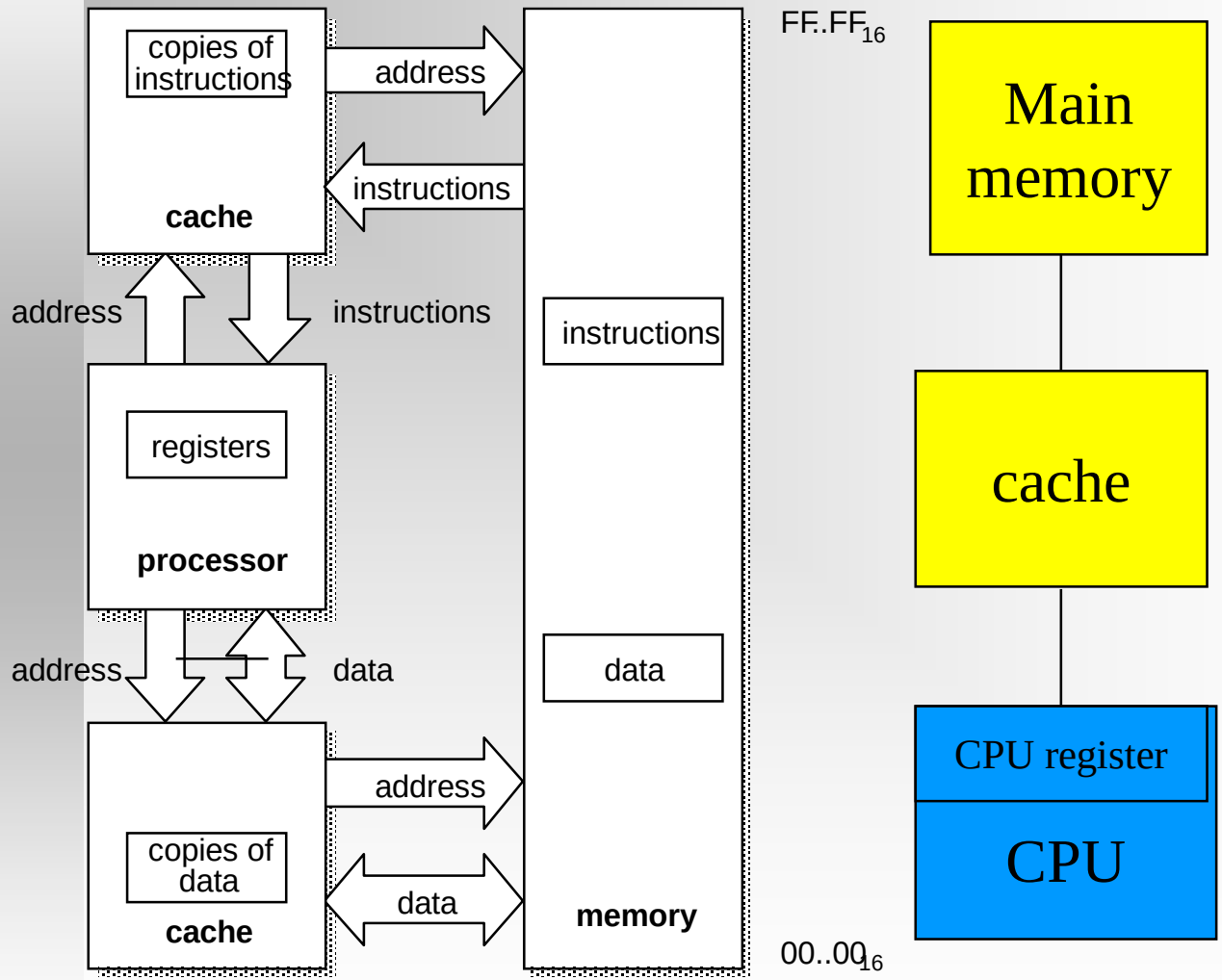
```
.globl _start
_start: b       reset
        ldr     pc, _undefined_instruction
        ldr     pc, _software_interrupt
        ldr     pc, _prefetch_abort
        ldr     pc, _data_abort
        ldr     pc, _not_used
        ldr     pc, _irq
        ldr     pc, _fiq

_undefined_instruction: .word undefined_instruction
_software_interrupt:    .word software_interrupt
_prefetch_abort:        .word prefetch_abort
_data_abort:            .word data_abort
_not_used:              .word not_used
_irq:                   .word irq
_fiq:                   .word fiq
        .balignl 16,0xdeadbeef
reset:
        mrs     r0,cpsr
        bic     r0,r0,#0x1f
        orr     r0,r0,#0x13
        msr     cpsr,r0  ; * set the cpu to SVC32 mode
        bl      cpu_init_crit
```

摘自u-boot start.s

FARSIGHT

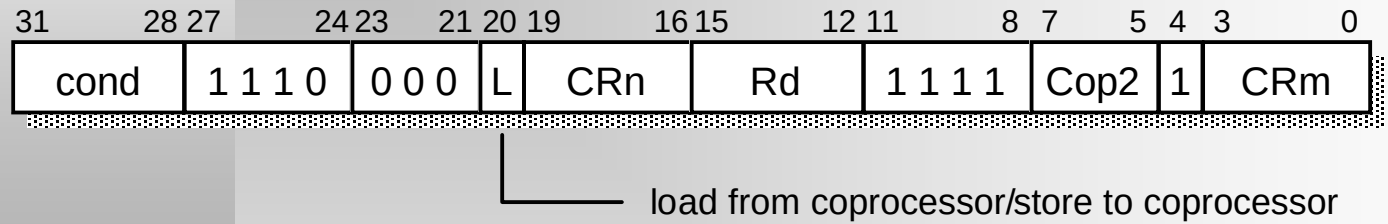
存储类型



FARSIGHT

MMU管理

- CP15协处理器操作指令



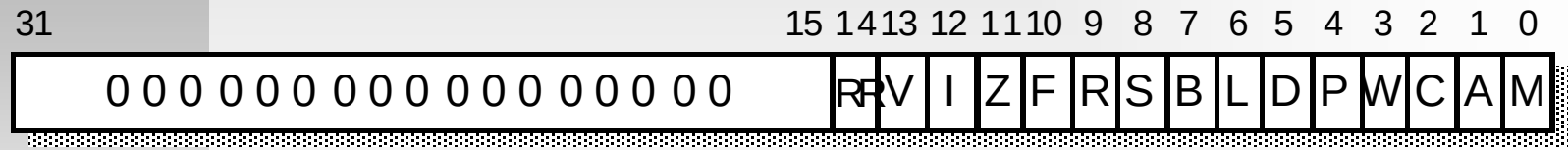
- CP15 MMU寄存器

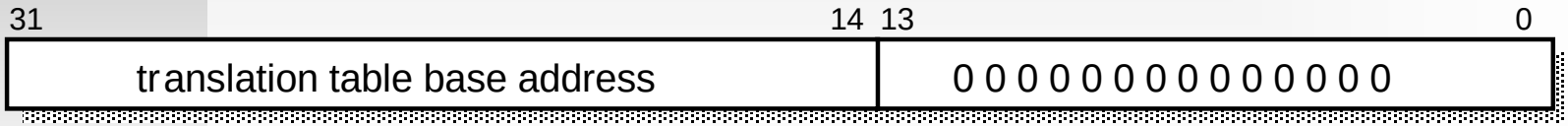
Register	Purpose
0	ID Register
1	Control
2	Translation Table Base
3	Domain Access Control
5	Fault Status
6	Fault Address
7	Cache Operations
8	TLB Operations
9	Read Buffer Operations
10	TLB lockdown
13	Process ID Mapping
14	Debug Support
15	Test & Clock Control
4, 11-12	UNUSED

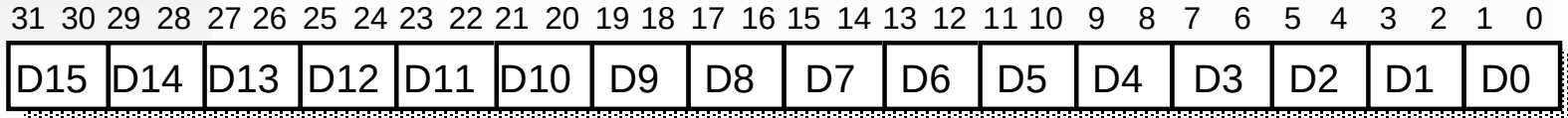
FARSIGHT

CP15与MMU操作相关寄存器

寄存器	作　用
寄存器 C1 中某些位	用于配置 MMU 中　些操作
寄存器 C2	保存内存中页表的基地址
寄存器 C3	设置域(domain)的访问控制属性
寄存器 C4	保留
寄存器 C5	内存访问失效状态指示
寄存器 C6	内存访问失效时失效的地址
寄存器 C8	控制与清除 TLB 内容相关的操作
寄存器 C10	控制与锁定 TLB 内容相关的操作

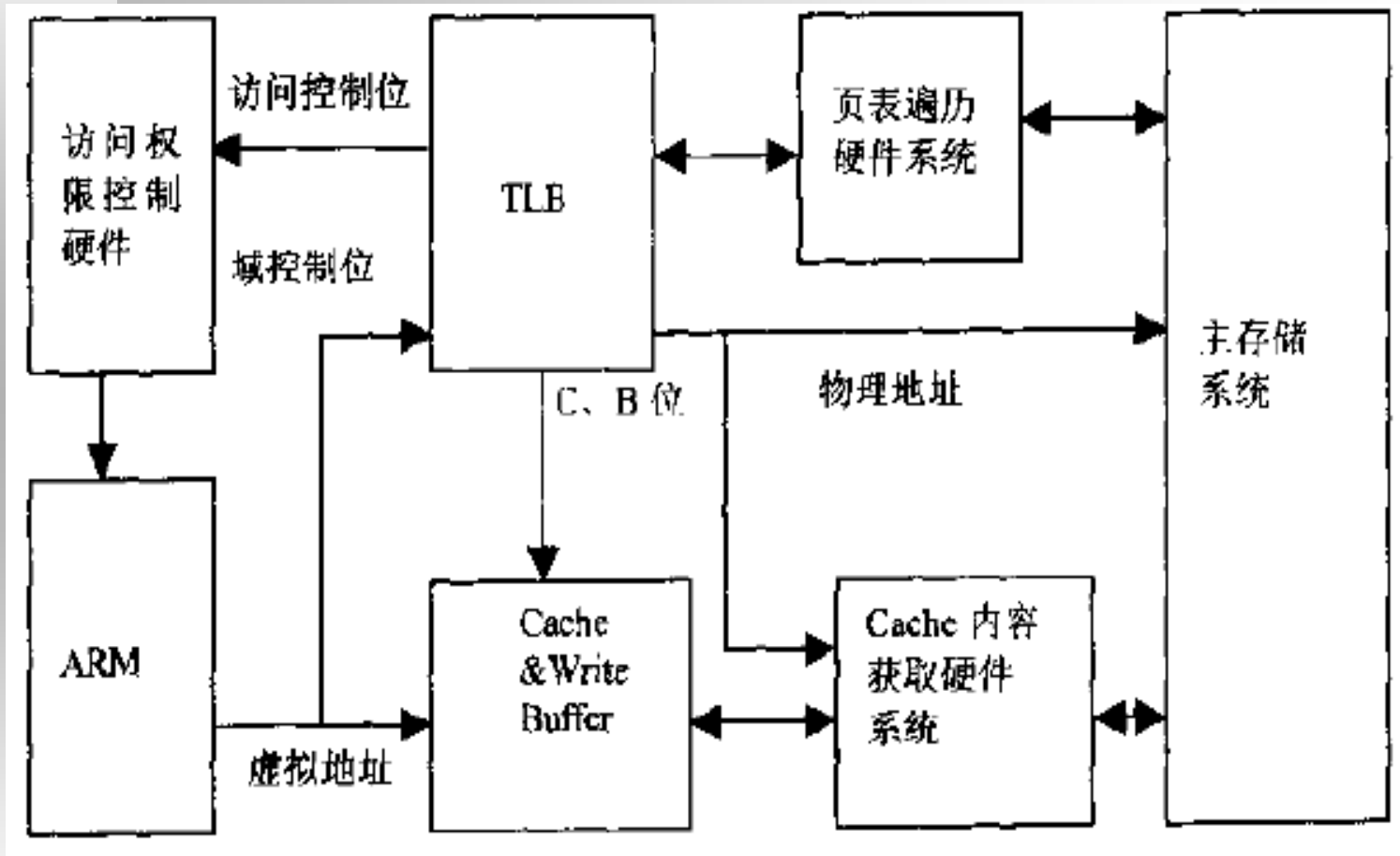




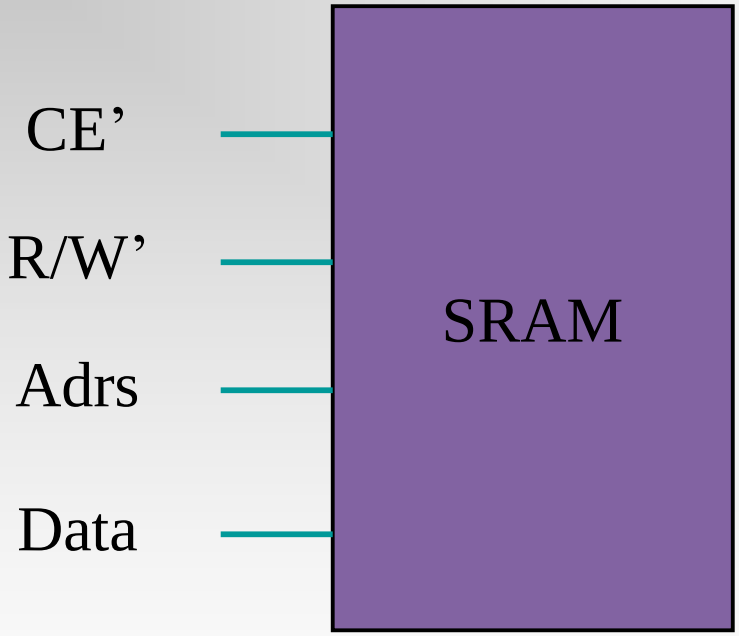


GHT

存储访问


访问权限控制硬件
访问控制位
域控制位
TLB
页表遍历硬件系统
主存储系统
物理地址
C、B 位
ARM
虚拟地址
Cache &Write Buffer
Cache 内容获取硬件系统

典型SRAM访问

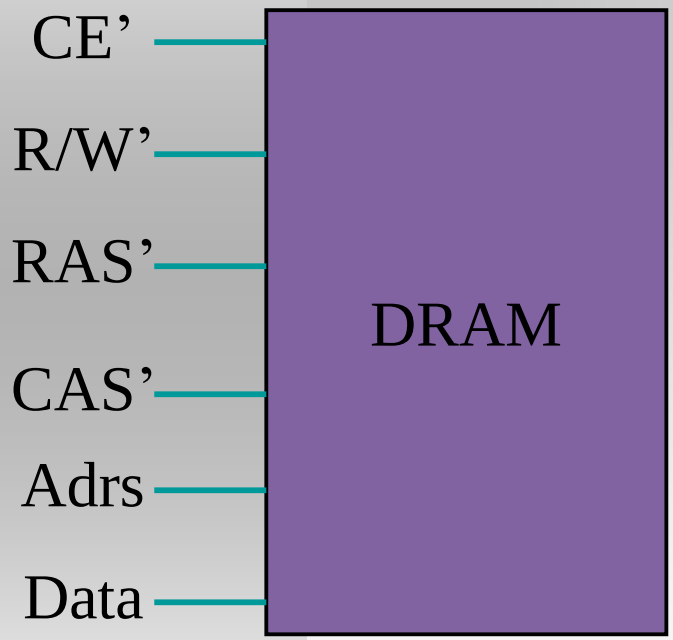


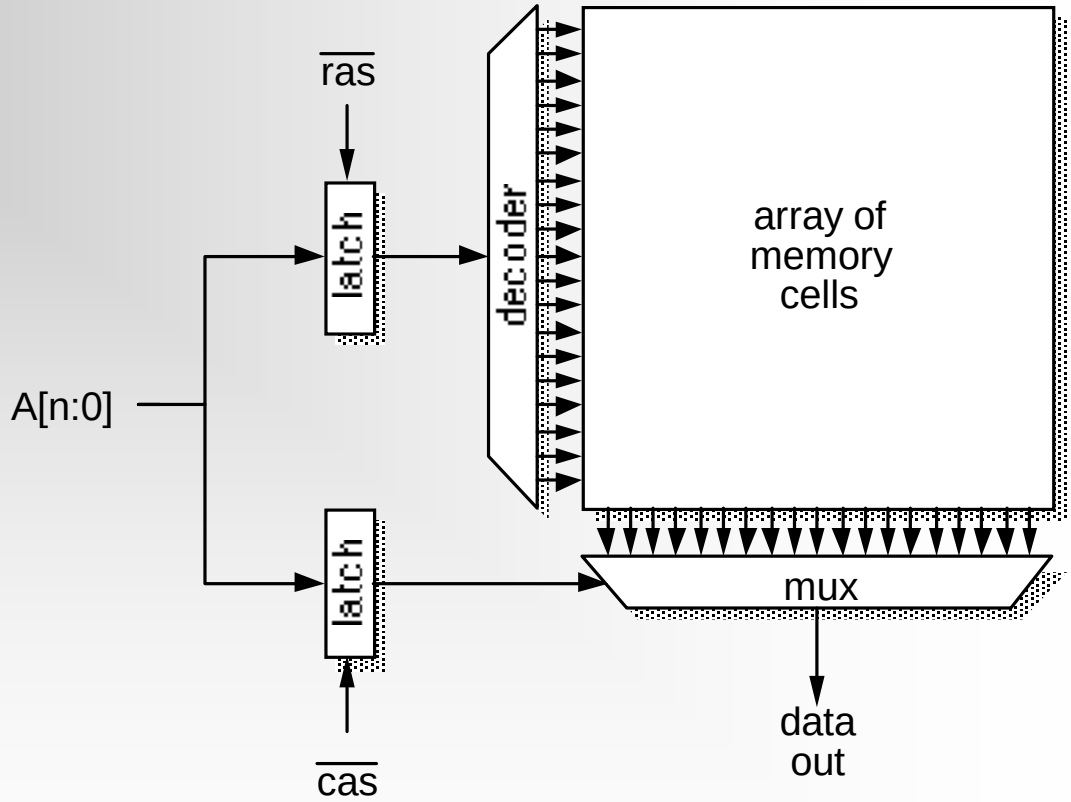
Flash存储访问

- Flash可以在系统电压下擦写
- 擦写时间比读取时间长
- 必须以块方式擦写

FARSIGHT

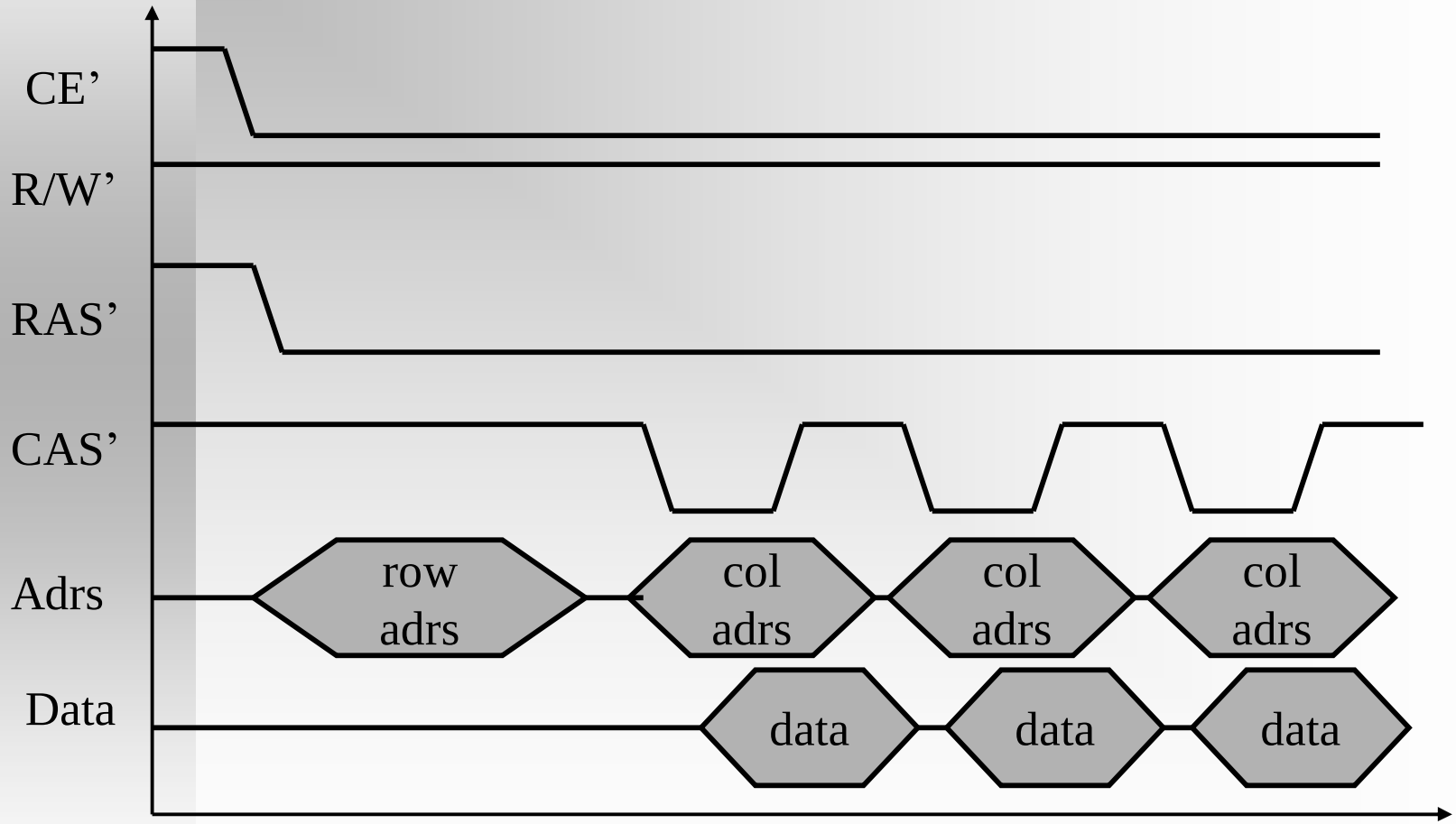
典型DRAM设备





FARSIGHT

SDRAM页模式访问时序



引导过程分析

FARSIGHT

u-boot启动过程

- Stage1:
  - 基本的硬件初始化
  - 为加载stage2准备RAM空间
  - 拷贝stage2到RAM中
  - 设置堆栈指针sp
  - 跳到stage2的入口点

FARSIGHT

u-boot启动过程

- Stage2:
  - 初始化本阶段要使用到的硬件设备
  - 检测系统的内存映射
  - 加载内核映像和文件系统映像
  - 设置内核的启动参数
  - 调用内核

FARSIGHT

u-boot启动顺序

```
_start:                          --cpu/arm920t/start.S
reset:
    cpu_init_crit:
    relocate:
    stack_setup:
    start_armboot()              --lib_arm/board.c
        init_sequence[] = {cpu_init, board_init, …}
        flash_init()             --board/smdk2410/flash.c
      env_relocate()
      devices_init()
      console_init_r ()
      main_loop ()               --common/main.c
```

FARSIGHT

“go”命令

```
int do_go (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
{
        ulong   addr, rc;
        int     rcode = 0;
        if (argc < 2) {
                printf ("Usage:\n%s\n", cmdtp->usage);
                return 1;
        }
        addr = simple_strtoul(argv[1], NULL, 16);
        printf ("## Starting application at 0x%08lX ...\n", addr);
        rc = ((ulong (*)(int, char *[]))addr) (--argc, &argv[1]);
        if (rc != 0)   rcode = 1;
        printf ("## Application terminated, rc = 0x%lX\n", rc);
        return rcode;
}
```

FARSIGHT

“bootm”命令

```
int do_bootm (cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
{
        printf ("   Uncompressing %s ... ", name);
        gunzip ((void *)ntohl(hdr->ih_load), unc_len, (uchar *)data, &len)
        do_bootm_linux()                                    --lib_arm/armlinux.c
        {
                theKernel = (void (*)(int, int, uint))ntohl(hdr->ih_ep);
                char *commandline = getenv ("bootargs");
                setup_commandline_tag (bd, commandline);
                cleanup_before_linux ();
                theKernel (0, bd->bi_arch_number, bd->bi_boot_params);
        }
}
```

FARSIGHT

u-boot详细分析

- 请参考试验手册

FARSIGHT

U-boot代码分析(1)

board	Board dependent files, RPXlite(mpc8xx), smdk2410(arm920t), sc520_cdp(x86) …
cpu	CPU specific files, mpc8xx, ppc4xx, arm720t, arm920t, xscale, i386
lib_ppc	Files generic to PowerPC architecture
lib_arm	Files generic to ARM architecture
lib_i386	Files generic to X86 architecture
include	Header Files and board configs

FARSIGHT

U-boot代码分析(2)

common	Misc functions
lib_generic	Generic library functions
net	Networking code
fs	File System Code
post	Power On Self Test
drivers	Common used device drivers
disk	Hard disk interface code
rtc	Real Time Clock drivers
dtt	Digital Thermometer and Thermostat drivers

FARSIGHT

U-boot代码分析(3)

examples	Example code for standalone applications, etc.
tools	Tools to build S-Record or U-Boot images, etc.
doc	Documentation (don't expect too much)

FARSIGHT

U-boot移植

- 需要根据目标板修改Board和CPU特定的代码
  - 选择一个u-boot已支持的类似板子作为起点
  - 主要完成CPU、内存、FLASH、串口、以太网接口的初始化
  - 配置u-boot、编译、测试、固化
- U-boot的移植
  - Board目录
  - Makefile
  - Config.in
  - 宏定义和参数

关键程序

- 串口初始化
  - /cpu/arm920t/s3c24x0/serial.c
- SDRAM初始化
  - include/.../memory.h
- FLASH初始化
  - board/smdk2410/flash.c
- 以太网初始化
  - net/cs8900a.c

FARSIGHT

U-boot编译

- 源码使用 Makefile管理
- u-boot配置
  - make <BOARD_NAME>_config
    make smdk2410_config
- 交叉环境编译
  - make CROSS_COMPILE = arm-linux-
- 编译后产生文件
  - *System.map* The symbol map
  - *u-boot* U-Boot in ELF binary format
  - *u-boot.bin* U-Boot raw binary image
  - *u-boot.srec* U-Boot image in Motorola's S-Record format

FARSIGHT

U-boot调试

- 添加调试信息
  - SHOW_BOOT_PROGRESS(arg)
  - 用来打印启动运行过程中的信息
- 修改u-boot在RAM中TEXT_BASE基址
  - 在board/xxx/config.mk中定义
- 处理reset_vector
  - board/<board_name>/u-boot.lds
- 往编译参数DBGFLAGS添加-g
  - config.mk中定义
- 重新编译u-boot
- 使用硬件调试器，调试u-boot
  - 用可运行的u-boot调试新的u-boot

启动命令

- 使用 “tftp”和go命令

  =>tftp 30100000 zImage

  =>go 30100000

- 启动uImage

  =>tftp 30100000 uImage

  =>bootm 30100000

- 带ramdisk的uImage启动方式， 00040000、00200000分别是uImage和ramdisk.img所在位置

  =>bootm 00040000 00200000

华清远见

我们提供完善的培训

- 嵌入式Linux预科班（免费）
- 嵌入式Linux系统开发班
- 嵌入式Linux驱动开发班
- 嵌入式Linux应用开发班
- 嵌入式Linux就业培训班
- 企业内训课程

Any questions?
DISCUSS PLZ!

ACCESS ME:
huangxin@farsight.com.cn

FARSIGHT