start.S F:\codec-pj\source\uis7885-android13\bsp\bootloader\lk\arch\arm64 12641 2024/12/14

.section .text.boot

FUNCTION(\_start)

.globl arm\_reset

arm\_reset:

bl lk\_main //汇编跳转到lk main函数

main.c F:\codec-pj\source\uis7885-android13\bsp\bootloader\lk\top 7544 2024/12/14 133

--->init\_sprdlog(INIT\_LOG\_BUFFER);

--->thread\_init\_early();

--->FTL\_Savepoint\_Private(PHASE\_ARCH\_EARLY\_INIT);

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_EARLIEST, LK\_INIT\_LEVEL\_ARCH\_EARLY - 1);

--->arch\_early\_init();

--->FTL\_Savepoint\_Private(PHASE\_PLATFORM\_EARLY\_INIT);

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_ARCH\_EARLY, LK\_INIT\_LEVEL\_PLATFORM\_EARLY - 1);

--->platform\_early\_init();

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_PLATFORM\_EARLY, LK\_INIT\_LEVEL\_TARGET\_EARLY - 1);

--->target\_early\_init();

--->sprd\_serial\_init();

--->FTL\_Savepoint\_Private(PHASE\_HEAP\_INIT);//// bring up the kernel heap

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_TARGET\_EARLY, LK\_INIT\_LEVEL\_HEAP - 1);

--->heap\_init();

--->call\_constructors(); // deal with any static constructors

--->FTL\_Savepoint\_Private(PHASE\_KERNEL\_INIT); // initialize the kernel

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_HEAP, LK\_INIT\_LEVEL\_KERNEL - 1);

--->kernel\_init();

// create a thread to complete system initialization

--->thread\_t \*t = thread\_create("bootstrap2", &bootstrap2, NULL, DEFAULT\_PRIORITY, DEFAULT\_STACK\_SIZE\_16K);

--->bootstrap2

--->FTL\_Savepoint\_Private(PHASE\_BOOTSTRAP2\_ENTER);

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_THREADING, LK\_INIT\_LEVEL\_ARCH - 1);

--->arch\_init(); // initialize the rest of the platform

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_ARCH, LK\_INIT\_LEVEL\_PLATFORM - 1);

--->platform\_init();

// initialize the target

--->FTL\_Savepoint\_Private(PHASE\_TARGET\_INIT);

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_PLATFORM, LK\_INIT\_LEVEL\_TARGET - 1);

--->target\_init();

// init sprdlog : init\_write\_log after MMC Initialization

--->init\_sprdlog(INIT\_LOG\_WRITER);

--->FTL\_Savepoint\_Private(PHASE\_LOG\_PARTITION\_DONE);

//initializing apps

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_TARGET, LK\_INIT\_LEVEL\_APPS - 1);

---> apps\_init();

--->app->init(app) //初始化\*.apps段中的函数。展锐平台俩个app APP\_START(sprdboot)和APP\_START(shell)

sprdboot.c F:\codec-pj\source\uis7885-android13\bsp\bootloader\lk\app\sprdboot 17714 2024/12/14 400

--->console\_init();

--->console\_register\_commands(block);//注册commands ,通过STATIC\_COMMAND注册，\_\_commands\_start

--->manual\_cmd\_shell();//debug定义，执行手动cmd

--->sprd\_boot();

--->FTL\_Savepoint\_Private(PHASE\_SPRDBOOT\_INIT);

--->autoload\_mode();

--->adjust\_spl\_slot()

--->do\_select\_ab();//android ab系统启动

--->get\_boot\_role()//是否为BOOTLOADER\_MODE\_LOAD

--->ddr\_eng\_mode\_debug();

--->usb\_phy\_enable(0);

--->boot\_pwr\_check();

--->bootmode = s\_boot\_func\_array[i]() //获取启动模式

--->get\_mode\_from\_chipram\_env ///\* 0 get mode from chipram\_env\*/

--->get\_mode\_from\_bat\_low ///\* 1 get mode from bat low\*/

--->write\_sysdump\_before\_boot\_extend ///\* 2 get mode from sysdump\*/

--->get\_mode\_from\_miscdata\_boot\_flag ///\* 3 get mode from miscdata flag\*/

--->get\_mode\_from\_file\_extend ///\* 4 get mode from file\*/

--->get\_mode\_from\_watchdog /\* 5 get mode from watch dog\*/

--->get\_mode\_from\_alarm\_register /\* 6 get mode from alarm register\*/

--->get\_mode\_from\_pctool /\* 7 get mode from calibration detect\*/

--->get\_mode\_from\_smpl /\* 8 get mode from smpl\*/

--->get\_mode\_from\_charger /\* 9 get mode from charger\*/

--->get\_mode\_from\_keypad /\* 10 get mode from keypad\*/

--->get\_mode\_from\_gpio\_extend ///\* 11 get mode from gpio\*/

--->board\_boot\_mode\_regist(boot\_mode\_array);//注册启动模式

--->boot\_mode\_array[bootmode]();//执行启动模式

get boot mode in boot func array[10] 按键检测启动

enter boot mode g\_mode\_str[2]:NORMAL\_MODE 正常启动

--->get\_mode\_from\_keypad

--->key\_mode = check\_key\_boot(key\_code);//检测3种情况的按键，fastboot ,recovery，和normal正常模式，无按键则进入mormal模式。

--->bootcause\_cmdline="Pbint triggered";

--->return CMD\_NORMAL\_MODE;

--->MODE\_REGIST(CMD\_NORMAL\_MODE, normal\_mode);

--->boot\_mode\_array[bootmode:CMD\_NORMAL\_MODE=2]()

--->normal\_mode//正常启动模式

--->vibrator\_hw\_init();//马达初始化

--->set\_vibrator(1);//马达震动

boot\_os\_mp.c F:\codec-pj\source\uis7885-android13\bsp\bootloader\lk\app\sprdboot 11783 2024/12/14 240

--->vlx\_boot(BOOT\_PART, BACKLIGHT\_ON, LCD\_ON);//BOOT\_PART启动boot分区

--->write\_log();

--->secondary\_cpu\_poweron();//启动第二个cpu

--->psci\_call(PSCI\_CPU\_ON\_AARCH64, cpuid << 8, (unsigned long)smp\_boot, 0);

--->secondary\_task\_create(kernel\_pname);

创建线程:sprd\_preload sprd\_lcd secure\_sign\_verify kernel\_dtb\_parse

--->thread\_create("sprd\_preload", &sprd\_preload, NULL, HIGH\_PRIORITY, DEFAULT\_STACK\_SIZE);//镜像加载

--->thread\_create("sprd\_lcd", &sprd\_lcd, NULL, HIGH\_PRIORITY, DEFAULT\_STACK\_SIZE);//sprd\_lcd lcd显示

--->sprd\_lcd

--->sprd\_set\_preload(lcd.status, lcd.blacklight);

--->preload->power()

--->preload->display(LOGO\_NORMAL\_POWER, brightness, lcd\_on);

--->.display = logo\_display

--->drv\_lcd\_init();

--->lcd\_splash(LOGO\_PART);

--->lcd\_enable()

--->set\_backlight(backlight\_value);

--->lt9211d\_main(); //mipi to lvds

--->

--->preload->vibrator(0);//关闭振铃

--->thread\_create("secure\_sign\_verify", &secure\_sign\_verify, (void\*)kernel\_pname, HIGH\_PRIORITY, DEFAULT\_STACK\_SIZE);//签名验证

--->thread\_create("kernel\_dtb\_parse", &kernel\_dtb\_parse, (void\*)kernel\_pname, HIGH\_PRIORITY, DEFAULT\_STACK\_SIZE);//设备树parse

--->thread\_detach(sprd\_threads[i]);

--->thread\_resume(sprd\_threads[0]);

--->i2c\_dvfs\_hwchn\_init();

--->vlx\_entry((uchar \*)DT\_ADR);

--->start\_linux(dt\_addr);//启动linux

--->theKernel = (void (\*)(void \*, int, int, int))KERNEL\_ADR;

--->FTL\_Savepoint\_Private(PHASE\_PRE\_KERNEL);

--->lk\_mp\_final\_fixup(dt\_addr, g\_ramdisk\_addr);

--->charge\_mode\_bootsp();

--->modem\_entry();

--->write\_log\_before\_entry\_kernel();

--->theKernel(dt\_addr, 0, 0, 0);//进入kernel

--->start\_app(app);

--->thread\_create(app->name, &app\_thread\_entry, (void \*)app, DEFAULT\_PRIORITY, stack\_size);

--->app\_thread\_entry

--->app->entry(app, NULL)//sprdboot.c启动

--->lk\_primary\_cpu\_init\_level(LK\_INIT\_LEVEL\_APPS, LK\_INIT\_LEVEL\_LAST);

--->thread\_set\_pinned\_cpu(t, 0);

--->thread\_detach(t);

--->thread\_resume(t);