实验 4 鸿蒙 LiteOS-a 内核移植——内存移植

22920212204396　　黄子安

一、实验目的

1、给上次移植的 demochip 单板增加内存映射，从而实现内存的移植

二、实验环境

1、物理机 Windows11

2、虚拟机 Ubuntu18.04

三、实验内容

移植内存首先要做的就是知道内存结构，通过阅读 IMX6ULL 官方的手册可以知道内存的物理基地址为 8000_0000，视频中讲到内存的大小为 512MB，不是手册的 2GB，按照视频的内容进行

Table 2-1. System memory map

Start address	End address	Size	Description
8000_0000	FFFF_FFFF	2048 MB	MMDC—x16 DDR Controller.
7000_0000	7FFF_FFFF	256 MB	Reserved
6000_0000	6FFF_FFFF	256 MB	QSPI1 Memory
5800_0000	5FFF_FFFF	128 MB	EIM Aliased
5000_0000	57FF_FFFF	128 MB	EIM (NOR/SRAM)
1000_0000	4FFF_FFFF	1024 MB	Reserved
0E00_0000	0FFF_FFFF	32 MB	Reserved
0C00_0000	0DFF_FFFF	32 MB	QSPI1 Rx Buffer
0900_0000	0BFF_FFFF	48 MB	Reserved
0800_0000	08FF_FFFF	16 MB	Reserved
02C0_0000	07FF_FFFF	84 MB	Reserved
0230_0000	02BF_FFFF	9 MB	Reserved
0220_0000	022F_FFFF	1 MB	Table 2-4 AIPS-3. See IP listing on the separate map.
0210_0000	021F_FFFF	1 MB	Table 2-3 AIPS-2. See the IP listing on the separate map.
0200_0000	020F_FFFF	1 MB	Table 2-2 AIPS-1. See the IP listing on the separate map.

在内核代码中修改物理基地址和内存的大小，对应的内核文件为openharmony/vendor/democom/demochip/board/include/board.h

```
#define __BOARD_CONFIG_H__

#include "menuconfig.h"

#ifdef __cplusplus
#if __cplusplus
extern "C" {
#endif /* __cplusplus */
#endif /* __cplusplus */

/* physical memory base and size */
#define DDR_MEM_ADDR            0x80000000
#define DDR_MEM_SIZE            0x20000000

#define DDR_RAMFS_ADDR (DDR_MEM_ADDR + DDR_MEM_SIZE)
#define DDR_RAMFS_SIZE 0x4000000  /* 60M for ramfs, 4M for lcd */

#define LCD_FB_BASE    (DDR_RAMFS_ADDR + DDR_RAMFS_SIZE)
#define LCD_FB_SIZE    0x400000  /* 4M */

/* Peripheral register address base and size */
#define PERIPH_PMM_BASE         0x40000000
#define PERIPH_PMM_SIZE         0x20000000

/* GIC base and size : 1M-align */
#define GIC_PHY_BASE          0xA0000000
#define GIC_PHY_SIZE          0xA0100000
```

IMX6ULL 芯片上设备地址分布比较零散，从 0 到 0x6FFFFFFF 都有分布，期中有很多保留地址没有被使用，如果全部映射完地址空间会不够，因此需要将各个模块单独映射

Table 2-1. System memory map

Start address	End address	Size	Description
8000_0000	FFFF_FFFF	2048 MB	MMDC—x16 DDR Controller.
7000_0000	7FFF_FFFF	256 MB	Reserved
6000_0000	6FFF_FFFF	256 MB	QSPI1 Memory
5800_0000	5FFF_FFFF	128 MB	EIM Aliased
5000_0000	57FF_FFFF	128 MB	EIM (NOR/SRAM)
1000_0000	4FFF_FFFF	1024 MB	Reserved
0E00_0000	0FFF_FFFF	32 MB	Reserved
0C00_0000	0DFF_FFFF	32 MB	QSPI1 Rx Buffer
0900_0000	0BFF_FFFF	48 MB	Reserved
0800_0000	08FF_FFFF	16 MB	Reserved
02C0_0000	07FF_FFFF	84 MB	Reserved
0230_0000	02BF_FFFF	9 MB	Reserved
0220_0000	022F_FFFF	1 MB	Table 2-4 AIPS-3. See IP listing on the separate map.
0210_0000	021F_FFFF	1 MB	Table 2-3 AIPS-2. See the IP listing on the separate map.
0200_0000	020F_FFFF	1 MB	Table 2-2 AIPS-1. See the IP listing on the separate map.
0181_0000	01FF_FFFF	8128 KB	Reserved
0180_C000	0180_FFFF	16 KB	Reserved

实现最小系统移植需要完成串口和中断控制器的地址映射，他们的地址信息如下

Table 2-1. System memory map (continued)

Start address	End address	Size	Description
0180_8000	0180_BFFF	16 KB	BCH
0180_6000	0180_7FFF	8 KB	GPMI
0180_4000	0180_5FFF	32 KB	APBH DMA
0180_0000	0180_3FFF	16 KB	Reserved
0120_0000	017F_FFFF	6 MB	Reserved
0110_0000	011F_FFFF	1 MB	Reserved
0100_0000	010F_FFFF	1 MB	Reserved
00F0_0000	00FF_FFFF	1 MB	Reserved
00E0_0000	00EF_FFFF	1 MB	(per_m) configuration port
00D0_0000	00DF_FFFF	1 MB	(cpu) configuration port
00C0_0000	00CF_FFFF	1 MB	GPV_1 PL301
00B0_0000	00BF_FFFF	1 MB	GPV_0 PL301 configuration port
00A0_8000	00AF_FFFF	992 KB	Reserved
00A0_0000	00A0_7FFF	32 KB	ARM Peripherals: GIC400 Only visible to ARM core(s)
009C_0000	009F_FFFF	256 KB	Reserved

0203_4000	0203_7FFF	AIPS-1 (via SPBA) Glob,Module ENABLE	ASRC	16 KB
0203_0000	0203_3FFF		SAI3	16 KB
0202_C000	0202_FFFF		SAI2	16 KB
0202_8000	0202_BFFF		SAI1	16 KB
0202_4000	0202_7FFF		ESAI	16 KB
0202_0000	0202_3FFF		UART1	16 KB
0201_C000	0201_FFFF		Reserved for SDMA internal registers	16 KB

这里为了一次性移植两个设备，将内存的大小开得稍微大一些，能够覆盖到串口

```
#endif /* __cplusplus */
#endif /* __cplusplus */

/* physical memory base and size */
#define DDR_MEM_ADDR            0x80000000
#define DDR_MEM_SIZE            0x20000000

#define DDR_RAMFS_ADDR (DDR_MEM_ADDR + DDR_MEM_SIZE)
#define DDR_RAMFS_SIZE 0x4000000  /* 60M for ramfs, 4M for lcd */

#define LCD_FB_BASE    (DDR_RAMFS_ADDR + DDR_RAMFS_SIZE)
#define LCD_FB_SIZE    0x400000  /* 4M */

/* Peripheral register address base and size */
#define PERIPH_PMM_BASE         0x00a00000 //GIC base address
#define PERIPH_PMM_SIZE         0x02300000 //

/* GIC base and size : 1M-align */
#define GIC_PHY_BASE         0xA0000000
#define GIC_PHY_SIZE         0xA0100000

#define KERNEL_VADDR_BASE       0x40000000
#define KERNEL_VADDR_SIZE       DDR_MEM_SIZE

#define SYS_MEM_BASE            DDR_MEM_ADDR
#define SYS_MEM_SIZE_DEFAULT    0x2000000
#define SYS_MEM_END             (SYS_MEM_BASE + SYS_MEM_SIZE_DEFAULT)
```

使用 make 命令编译下确保可以编译通过

```
book@hza-virtual-machine:~/openharmony/kernel/liteos_a$ make clean
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/arch/arm/arm”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/arch/arm/arm”
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/platform”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/platform”
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/kernel/common”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/kernel/common”
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/kernel/base”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/kernel/base”
make[1]: 进入目录“/home/book/openharmony/vendor/democom/demochip/board”
make[1]: 离开目录“/home/book/openharmony/vendor/democom/demochip/board”
```

```
book@hza-virtual-machine:~/openharmony/kernel/liteos_a$ make -j 16
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a”
/home/book/openharmony/kernel/liteos_a/tools/menuconfig/conf --silentoldconfig
/home/book/openharmony/kernel/liteos_a/Kconfig
mv -f /home/book/openharmony/kernel/liteos_a/include/generated/autoconf.h /home
/book/openharmony/kernel/liteos_a/platform/include/menuconfig.h
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a”
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/arch/arm/arm”
src/startup/reset_vector_up.S:138:2: warning: deprecated since v7, use 'dsb'
 mcr p15, 0, r0, c7, c10, 4 @ DSB
 ^
src/startup/reset_vector_up.S:139:2: warning: deprecated since v7, use 'isb'
 mcr p15, 0, r0, c7, c5, 4 @ ISB
 ^
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/arch/arm/arm”
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/platform”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/platform”
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/kernel/common”
```

四、实验心得

本次实验移植了 Liteos-a 的内存，主要是为了下次实验铺垫移植串口，大概对 imx6ull 的内存映射结构有初步的了解，为后续学习操作系统对内存管理做了一定铺垫。