实用操作系统课程实验报告

实验名称：	实验四 鸿蒙 LiteOS-a 内核移植——内存移植
实验日期：	2023-10-27
实验地点：	文宣楼 B313

学号：	33920212204567
姓名：	任宇
专业年级：	软工 2021 级
学年学期：	2023-2024 学年第一学期

1.实验目的

- 进行鸿蒙 LiteOS-a 内核的内存移植

2.实验内容和步骤

（1） 定义 DDR 内存的基地址 DDR_MEM_ADDR 和大小 DDR _MEM_SIZE：

因为开发板 DDR 容量是 512M，所以修改这个文件 vendor\democop\demochip\board\include\board.h

```
/* physical memory base and size */
#define DDR_MEM_ADDR            0x80000000
#define DDR_MEM_SIZE            0x20000000   /* 448M for os, reseved (512-448) for ramfs and framebuffer */

#define DDR_RAMFS_ADDR (DDR_MEM_ADDR + DDR_MEM_SIZE)
```

（2） 定义设备映射地址范围：

修改 board.h 文件，定义外设的内存映射区域的基地址 PERIPH_PMM_BASE 和大小 PERIPH_PMM_SIZE

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

这样修改后，GIC 的寄存器等硬件资源映射到了这个地址上，软件就可以通过读写这个地址来配置和控制 GIC。

同时，映射地址范围也不能太大，需要做限制

```
/* Peripheral register address base and size */
#define PERIPH_PMM_BASE         0x00a00000
#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)

#if (PERIPH_UNCACHED_BASE >= (0xFFFFFFFFU - PERIPH_UNCACHED_SIZE))
#error "Kernel virtual memory space has overflowed!"
#endif

#define EXC_INTERACT_MEM_SIZE       0x100000
```

（3） 编译内核：

```
clean demochip finish
book@ry-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
*
* Restart config...
*
*
* Compiler
*
LiteOS_Compiler_Type
  1. arm-linux-ohoseabi (COMPILER_HIMIX_32) (NEW)
> 2. clang-llvm (COMPILER_CLANG_LLVM)
choice[1-2?]:
#
# configuration written to .config
#
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
 ^
```

```
lliteipc -lpipes -lc -lsec -lscrew -lc++ -lc++abi -lcppsupport -lz -lposix -lbsd -llinuxkpi -lvfs -lmulti_partition -lbch -lfat -lvirpart -ldisk -lbcache -lra
mfs -lnfs -lproc -ljffs2 -llwip --whole-archive -lhdf -lhdf_config -lhello --no-whole-archive -lhievent -lmem -lmtd_common -lhilog -lshell -ltelnet -lsyscall
-lsecurity --end-group
/home/book/llvm/bin/..//bin/llvm-objcopy -R .bss -O binary /home/book/openharmony/kernel/liteos_a/out/demochip/liteos /home/book/openharmony/kernel/liteos_a/o
ut/demochip/liteos.bin
/home/book/llvm/bin/..//bin/llvm-objdump -t /home/book/openharmony/kernel/liteos_a/out/demochip/liteos |sort >/home/book/openharmony/kernel/liteos_a/out/democ
hip/liteos.sym.sorted
/home/book/llvm/bin/..//bin/llvm-objdump -d /home/book/openharmony/kernel/liteos_a/out/demochip/liteos >/home/book/openharmony/kernel/liteos_a/out/demochip/li
teos.asm
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/apps”
make[2]: 进入目录“/home/book/openharmony/kernel/liteos_a/apps/shell”
make[2]: 离开目录“/home/book/openharmony/kernel/liteos_a/apps/shell”
make[2]: 进入目录“/home/book/openharmony/kernel/liteos_a/apps/init”
make[2]: 离开目录“/home/book/openharmony/kernel/liteos_a/apps/init”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/apps”
book@ry-virtual-machine:~/openharmony/kernel/liteos_a$
```

3.实验总结

在内核移植过程中，内存移植主要是指将操作系统的内核适配到目标硬件平台的内存结构上，主要包括内存分配和内存映射两个部分，前者是根据目标硬件平台的缓存策略，配置内核的内存分配策略，后者则是将操作系统的虚拟地址空间映射到硬件平台的物理地址空间。除了这两部分以外，还有对 MMU 的配置，以支持虚拟内存的使用。

在本次实验中，对于 MMU 中权限管理和地址映射的代码并没有涉及修改，实验过程中只是修改了几个宏。根据移植平台的内存结构，实验过程中定义了 DDR 内存的基地址和大小，同时也定义了外设的内存映射区域的基地址和大小。通过本次实验，我更加深刻理解了操作系统内核的工作原理，以及内存移植的关键技术和方法，提升了自己对操作系统内核移植过程中涉及的技术点的理解和运用能力，为后续的移植奠定坚实的基础。

4.遇到的困难及解决方法

在实验指导视频中老师提到了 MMU 有两大主要功能，一是地址转换，二是权限管理，因为在先前计算机组成原理这门课中也学习到了地址转换相关知识，因此觉得很好理解，但对于权限管理这点并不清楚，因此专门去网上学习了相关知识：

MMU 提供了内存保护机制，防止不同的程序或不同的用户相互干扰。主要原理是以下几点：

1) 访问权限设置：

- 每个内存页或段都有对应的访问权限，如可读、可写、可执行等。
- 当 CPU 尝试访问某个内存地址时，MMU 会首先检查访问权限。

2) 地址隔离：

- 通过虚拟内存，每个进程都有自己独立的地址空间，彼此隔离。

- 不同进程的虚拟地址映射到不同的物理地址，因此它们不能直接访问彼此的内存。

3) 页表保护：

- 页表存储了虚拟地址到物理地址的映射关系，而页表的访问也受到权限的控制，只有特定的进程或操作系统内核才能修改页表。