IPADS INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

课程实验2：内存管理

陈海波 / 夏虞斌 负责

助教：黄政(huangzheng@sjtu.edu.cn)

上海交通大学并行与分布式系统研究所

https://ipads.se.sjtu.edu.cn

版权声明

- 本内容版权归**上海交通大学并行与分布式系统研究所**所有
- 使用者可以将全部或部分本内容免费用于非商业用途
- 使用者在使用全部或部分本内容时请注明来源：
  - 内容来自：上海交通大学并行与分布式系统研究所+材料名字
- 对于不遵守此声明或者其他违法使用本内容者，将依法保留追究权
- 本内容的发布采用 Creative Commons Attribution 4.0 License
  - 完整文本：https://creativecommons.org/licenses/by/4.0/legalcode

实验二简介

实验二

- **发布时间: 2021-03-17**
- **截止时间: 2020-04-02 23:59 (GMT+8)**
- **负责助教: 黄政 (huangzheng@sjtu.edu.cn)**
- **实验目的**
  - 了解Chcore中的物理内存管理机制
  - 了解Chcore中的虚拟内存管理机制
  - 如何为kernel映射内存空间

获取lab2

```
# fetch the remote updates, you are in branch lab1
chcore$ git fetch upstream

# you switch to the branch lab2, whose code is based on the empty code provided by the
TAs
chcore$ git checkout -b lab2 upstream/lab2
Branch 'lab2' set up to track remote branch 'lab2' from 'upstream'.
Switched to a new branch 'lab2'

# update the remote tracking branch to your origin repo instead of the upstream repo
chcore$ git push -u origin
To https://ipads.se.sjtu.edu.cn:2020/[username]/chcore.git
 * [new branch]      lab1 -> lab1
Branch 'lab1' set up to track remote branch 'lab1' from 'origin'.
```

获取lab2

```
chcore$ git merge lab1
Auto-merging kernel/main.c
Auto-merging Makefile
Auto-merging CMakeLists.txt
Merge made by the 'recursive' strategy.
 Merge made by the 'recursive' strategy.
 boot/print/printf.c    | 32 ++++++++++++++++++++++++++++++--
 kernel/common/printk.c | 34 ++++++++++++++++++++++++++++++---
 kernel/monitor.c       | 12 ++++++++++
 3 files changed, 73 insertions(+), 5 deletions(-)
```

三个部分

- Part A: Physical Page Management
- Part B: Virtual Memory
- Part C: Kernel Address Space

Part A: Physical page Management

- **了解Chcore中物理内存管理机制**
  - Chcore中是如何管理物理内存
  - Chcore中使用的数据结构
- **回忆伙伴系统**
  - 如何通过伙伴系统管理内存
  - 如何merge split内存块

Part A: Physical page Management

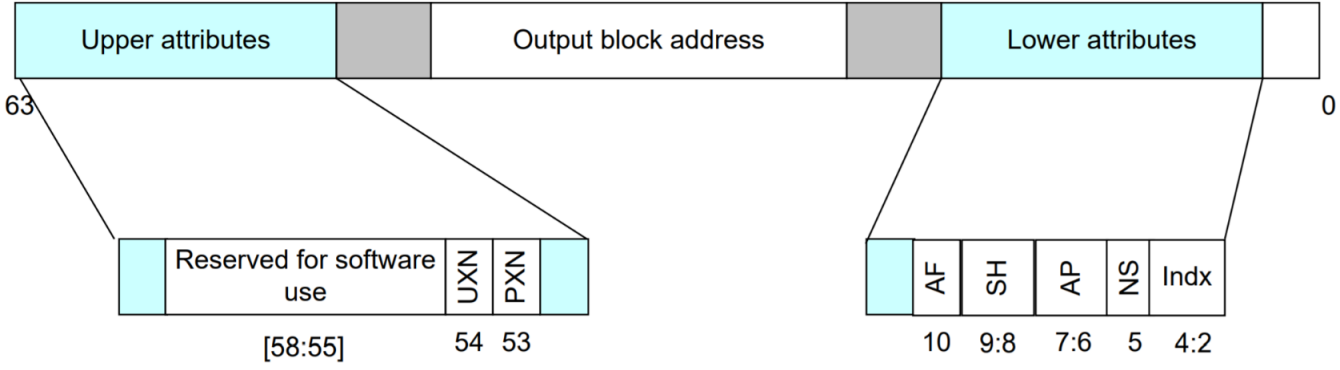
- **熟悉Chcore中物理内存布局**

Part B: Virtual Memory

- **了解arm中地址翻译的机制**
  - Arm中与页表翻译相关的寄存器
  - Arm中页条目中的属性位
- **64位四级页表**
  - Chcore是如何映射遍历页表的
  - 如何区别不同的条目（页，块，表）

Part B: Virtual Memory

- **区别X86和ARM：**
  - Ttbr和cr3寄存器
  - Memory attribution



Part C: Kernel Address space

- **了解kernel的页表映射**
  - 如何获取当前ttbr寄存器
  - 块条目与页条目的区别
- **如何保护kernel地址空间**
  - 页表中的属性位
  - 多个ttbr寄存器

需要做什么

- **Exercise**
  - 完成八个函数
  - Part A 和Part B单元测试
    - minunit测试工具
  - Part C会在runtime的时候检查

Make grade测试所有exercise

```
running chcore: (0.1s)
  buddy: OK
  page table: OK
  kernel space check: OK
Score: 100/100
```

```
1 tests, 640510 assertions, 0 failures

Finished in 0.17951529 seconds (real) 0.09242477 seconds (proc)

root@0699dcf5cc36:/chos/tests/mm/buddy#
```

需要做什么

- **Question**
  - 6个问题，每个问题需要在文档中做简单的回答

- **Challenge**
  - 2个挑战，会有相应的bonus
  - 可以在文档中附上相应的解决思路与测试

Enjoy your lab