L1 实验报告

221900223 袁唯瀚

实验要求

实现多处理器安全的内存分配和回收，保证分配和回收满足：

1. 原子性
2. 无重叠
3. 对齐
4. 无内存泄漏
5. 错误处理

突破性思路

buddySystem与slabSystem共同使用

在内存分配时，对于大文件，使用buddy_alloc；对于小文件，使用slab_alloc。回收时，则用相应方法回收。buddy与slab共同使用可以高效地使用内存，减少碎片，同时保持快速的分配和释放操作。

对齐处理

对于地址空间，利用

```
#define ALIGN(_A, _B) ((_A + _B - 1)& ~(_B - 1))
```

来对齐地址。
对于内存分配，利用

```
static size_t align_size(size_t size)
{
  if (size <= 8)
    return 8;
  size_t ret = 1;
  while (ret < size)
    ret <<= 1;
  return ret;
}
```

来分配2的幂次的内存。

分配buddy块

对整块内存块，进行逐步二分，得到所需大小的buddy块

block管理

利用addr_to_block与block_to_addr，实现对block的查找与转换。

L2 实验报告

221900223 袁唯瀚

架构设计

使用数组存储 task 列表，链表存储 handler 。在信号量和workload里面，使用随机数随机唤醒或者分配 task。task 由 kmt_create 创建，由 kmt_teardown 回收。task 的分配则是在 schedule 和 context save 里两个 handler 完成

spinlock_t实现

参考xv6代码，利用holding，push_off，pop_off管理中断开关。

在获取cpu_current()时，应该先关中断。

task_t实现

task 的元数据里面有两个变量控制 task 的调度——is_running，block，任意一个值为 true 都不可以被 schedule 调度。

其中 block 由信号量控制，如果没有资源就将其放入sem的task list里面，将block设为true，等待唤醒。如果有资源，随机唤醒一个，将 block设为false并将其移出sem的task list。

对于schedule，每个 cpu 都有一个idle task，防止 cpu 没有合适的task 运行。在获取合适的 task 之后，将 is_running 设为 true。在新的 save tast 覆盖旧的 task 之前， 将 save tast 的 is_running 设为 false，给别的 cpu 运行。

bug

在看2022年速通视频的时候看到将cpu_current()保存为临时变量使用。但是意外将其放在中断外面，出现了问题。。。