1. 合并空闲块

```
//-----------------------合并空闲块--------------------
list_entry_t* le;
struct Page* page;

le = list_next(&base->page_link);
page = le2page(le, page_link);
if (base + n == page) {  // free n页之后，base向后n页是否就是下一个空闲块
    base->property += page->property;  // 页数相加
    ClearPageProperty(page);  // 设置为不可分配
    list_del(&page->page_link);  // 从空闲页列表中清除
}

le = list_prev(&base->page_link);
page = le2page(le, page_link);
if (base == page + page->property) {
    page->property += base->property;
    ClearPageProperty(base);
    list_del(&base->page_link);
}
//-------------------------------------------------
```

```
MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
os is loading ...
memory management: default_pmm_manager
physcial memory map:
  memory: 0x0000000007e00000, [0x0000000080200000, 0x0000000087ffffff].
check_alloc_page() succeeded!
QEMU: Terminated
root@Bill:~/codes/CS334-OS/ass05_1#
```

2. best fit

```
list_entry_t *le = &free_list;
struct Page *page = NULL;
// --------------------best fit------------------------
while ((le = list_next(le)) != &free_list) {
    struct Page *p = le2page(le, page_link);
    // 寻找最小的满足条件的空闲页
    if ((page == NULL && p->property >= n) ||
    (p->property >= n && p->property < page->property)) {
        page = p;
    }
}
// --------------------best fit------------------------
```

```
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
os is loading ...
memory management: best_fit_pmm_manager
physcial memory map:
  memory: 0x0000000007e00000, [0x0000000080200000, 0x0000000087ffffff].
check_alloc_page() succeeded!
QEMU: Terminated
root@Bill:~/codes/CS334-OS/ass05_2#
```