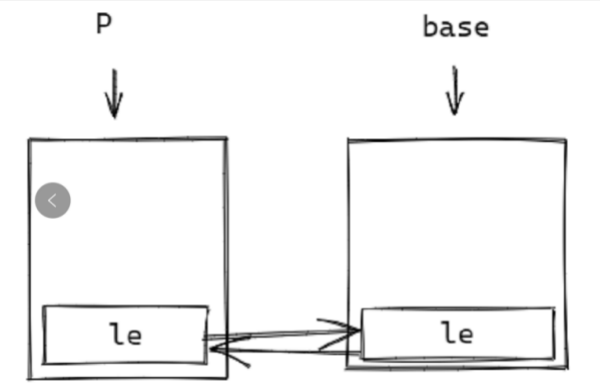
Assignment5

11911839 聂雨荷

Q1[50pts]

Please realize merging free blocks in `default_free_pages()`

In the lab code, there may exists the continuous free pages are divided into several blocks after released. Complete `default_free_pages()` to merge these continuous free pages in to a single node.



There may exists free block(p) before or after the newly released free block(base) . To merge them, you should:

1. Modify P's property
2. clear base's property
3. update the free_list

You can modify the check function in `default_pmm.c` to check your code.

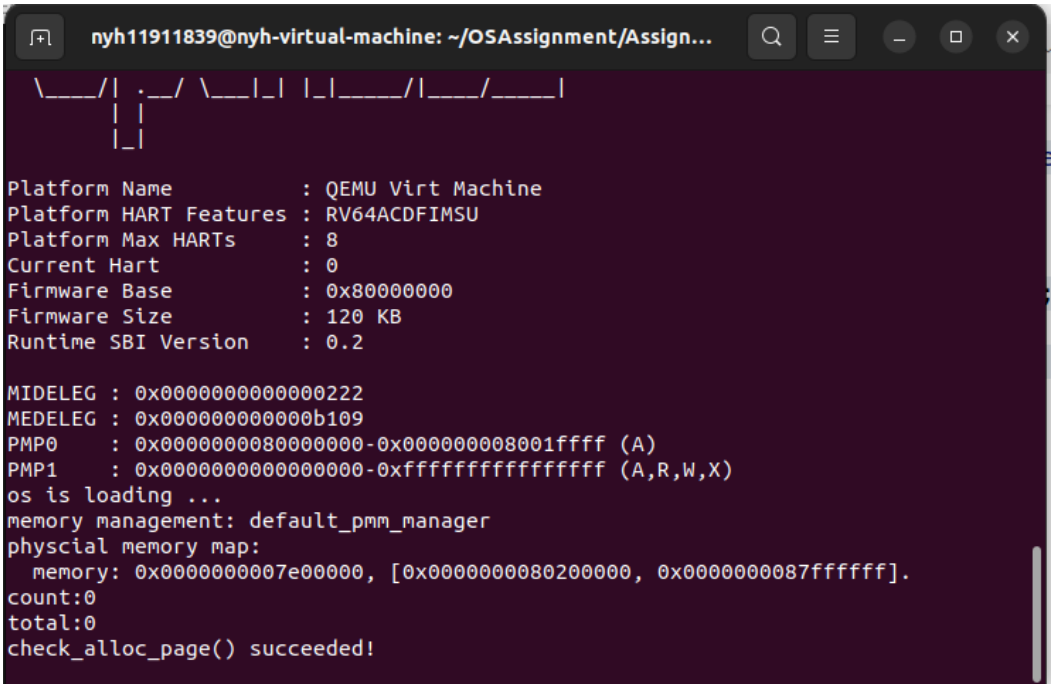
Your report should include the screenshot of your code(with annotations) and the check result(run `make qemu` to see the result).

add the following codes to the `deault_free_pages` inorder to merge free blocks

```
static void
default_free_pages(struct Page *base, size_t n) {
    assert(n > 0);
    struct Page *p = base;
    for (; p != base + n; p ++) {
        // assert(!PageReserved(p) && !PageProperty(p));
        p->flags = 0;
        set_page_ref(p, 0);
    }
    base->property = n;
    SetPageProperty(base);
```

```
    nr_free += n;

    if (list_empty(&free_list)) {
        list_add(&free_list, &(base->page_link));
    } else {
        list_entry_t* le = &free_list;
        while ((le = list_next(le)) != &free_list) {
            struct Page* page = le2page(le, page_link);
            if (base < page) {
                list_add_before(le, &(base->page_link));
                break;
            } else if (list_next(le) == &free_list) {
                list_add(le, &(base->page_link));
                break;
            }
        }
    }

    //-----------------------Merge free blocks-------------
-------
    list_entry_t* le = list_prev(&(base->page_link));//
find the previous block
    if (le != &free_list) { // if there exists such block
        p = le2page(le, page_link);
        if (p + p->property == base) { // check whether the
end address of block p equals to the address of base
            p->property += base->property;  // merge base
block to p block
            ClearPageProperty(base);
            list_del(&(base->page_link)); // delete base
from the free_list
            base = p;
        }
    }

    le = list_next(&(base->page_link));// find the next
block
    if (le != &free_list) {// if there exists such block
        p = le2page(le, page_link);
        if (base + base->property == p) {// check whether
the end address of base block equals to the address of the
next block
```

```
            base->property += p->property; // merge p block
to base block
            ClearPageProperty(p);
            list_del(&(p->page_link)); // delete previous
block from the free_list
        }
    }
    //-----------------------------------------------------


}
```



Q2[50pts]

Realize `bestfit` in `best_fit_pmm.c`.

You can modify `pmm_manager()` in `default_pmm.c` to check your code.

Your report should include the screenshot of your code(with annotations) and the check result(run `make qemu` to see the result).

The most important part is to modify the `best_fit_alloc_pages` function. Other functions remain the same as them in `default_pmm.c`.

```
static struct Page *
best_fit_alloc_pages(size_t n)
{
    assert(n > 0);
    if (n > nr_free) {
        return NULL;
    }
    struct Page *page = NULL;
```

```
    list_entry_t *le = &free_list;

    // find the best fit page
    struct Page *best_p = NULL;
    int best_proerty = __INT32_MAX__; // record the best
property
    while ((le = list_next(le)) != &free_list) {
        struct Page *p = le2page(le, page_link);
        if (p->property >= n) {// if current block's
property >= n
            if(p->property < best_proerty){ // check
whether current block's property is less than the
best_property, if true, current block is better then the
previous one
                best_proerty = p->property;
                best_p = p; // update the bestfit page
            }
        }
    }
    page = best_p;

    if (page != NULL) {
        list_entry_t* prev = list_prev(&(page->page_link));
        list_del(&(page->page_link));
        if (page->property > n) {
            struct Page *p = page + n;
            p->property = page->property - n;
            SetPageProperty(p);
            list_add(prev, &(p->page_link));
        }
        nr_free -= n;
        ClearPageProperty(page);
    }

    return page;
}
```

nyh11911839@nyh-virtual-machine: ~/OSAssignment/Assign...

```
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
os is loading ...
memory management: best_fit_pmm_manager
physcial memory map:
  memory: 0x0000000007e00000, [0x0000000080200000, 0x0000000087ffffff].
count:0
total:0
check_alloc_page() succeeded!
```