Assignment 4

Please complete a report in English in English in English in English and upload the corresponding codes.

The files should be uploaded to **blackboard** directly without compression without compression without compression without compression

The files to be submitted for this assignment are:

1. report.pdf
2. default_pmm.c
3. best_fit_pmm.c

---

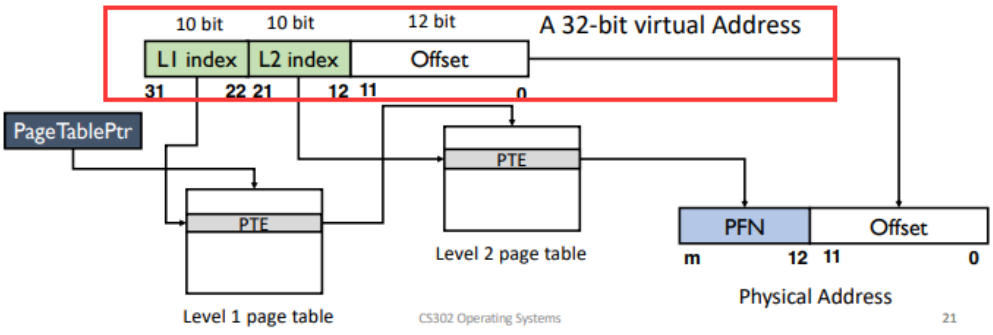
1. [15 pts] Read Chapter 15 of "Three Easy Pieces" (https://pages.cs.wisc.edu/~remzi/OSTEP/vm-mechanism.pdf) and explain how do the CPU hardware and the operating system cooperate in the procedure of address translation.

2. [15 pts] Read Chapter 16 "(https://pages.cs.wisc.edu/~remzi/OSTEP/vm-segmentation.pdf) and chapter 18 (https://pages.cs.wisc.edu/~remzi/OSTEP/vm-paging.pdf) of "Three Easy Pieces" and compare segmentation and paging. Your answer should cover all aspects (e.g., size of chunks, management of free space, context switch overhead, fragmentation, status bits and protection bits, etc.) and compare them side-by-side.

3. [15 pts] Consider a system with the following specifications:
   - 46-bit virtual address space
   - Page size of 8 KBytes
   - Page table entry size of 4 Bytes
   - Every page table is required to fit into a single page

   How many levels of page tables would be required to map the entire virtual address space? Please document the format of a virtual address under this translation scheme. Briefly explain your rationale.

   > hint: Here is the example of the format of a 32-bit virtual address in lecture.



4. [15 pts] Consider a system with following specifications:

   Both virtual address space and physical address are 32bits.

Page table entry size of 4Bytes

(a) Suppose it uses 1-level page table, the format of the translation scheme is

20 bit page	12 bit offset

What is the page size? What is the maximum page table size?

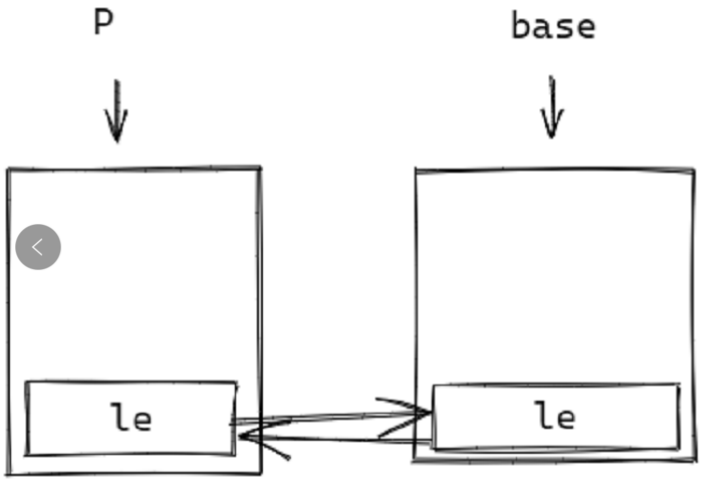
(b) Suppose it uses 2-level page table, the format of the translation scheme is

10 bit page	10 bit page	12 bit offset

- Please write down the 1-st level page number and its offset in decimal(base 10) of virtual address **0xC302C302**(base 16).
- Please write down the 2-nd level page number and its offset in decimal(base 10) of virtual address **0xEC6666AB** (base16)

5. [20 pts] Please realize merging free blocks in `default_free_pages()`

   In the A4 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:

a) Modify P's property

b) clear base's property

c) update the free_list

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

```
const struct pmm_manager default_pmm_manager = {
    .name = "default_pmm_manager",
    .init = default_init,
    .init_memmap = default_init_memmap,
    .alloc_pages = default_alloc_pages,
    .free_pages = default_free_pages,
    .nr_free_pages = default_nr_free_pages,
    .check = default_check,
    // 合并空闲块之后，请将上面的check注释，下面的check解除注释，进行测试
    //.check = firstfit_check_final,
};
```

to

```
const struct pmm_manager default_pmm_manager = {
    .name = "default_pmm_manager",
    .init = default_init,
    .init_memmap = default_init_memmap,
    .alloc_pages = default_alloc_pages,
    .free_pages = default_free_pages,
    .nr_free_pages = default_nr_free_pages,
    //.check = default_check,
    // 合并空闲块之后，请将上面的check注释，下面的check解除注释，进行测试
    .check = firstfit_check_final,
};
```

Your report should include the screenshot of your code(with annotations) and the check result(run `make qemu` to see the result). After finish your code, submit **default_pmm.c** to blackboard not gitlab.

sample result:

```
os is loading ...
memory management: default_pmm_manager
physcial memory map:
  memory: 0x0000000007e00000, [0x0000000080200000, 0x0000000087ffffff].
Checking continuous free pages merging...
check_alloc_page() succeeded!
```

6. [20 pts] Realize `bestfit` in `best_fit_pmm.c`.

   You can modify `init_pmm_manager()` in `pmm.c` to check your code.

```
static void init_pmm_manager(void) {
    pmm_manager = &default_pmm_manager;
    //pmm_manager = &best_fit_pmm_manager;
    cprintf("memory management: %s\n", pmm_manager->name);
    pmm_manager->init();
}
```

   to

```
static void init_pmm_manager(void) {
    // pmm_manager = &default_pmm_manager;
    pmm_manager = &best_fit_pmm_manager;
    cprintf("memory management: %s\n", pmm_manager->name);
    pmm_manager->init();
}
```

   Your report should include the screenshot of your code(with annotations) and the check result(run `make qemu` to see the result). After finish your code, submit **best_fit_pmm.c** to blackboard not gitlab.

   sample result:

```
os is loading ...
memory management: best_fit_pmm_manager
physcial memory map:
  memory: 0x0000000007e00000, [0x0000000080200000, 0x0000000087ffffff].
Checking bestfit...
check_alloc_page() succeeded!
```