Week6-Assignmet-Report

11911609-葛兆宁

Q1

- In CPU hardware, the work it does is to set physical base and bound address in its MMU registers, and accomplish the address translation. Secondly, support special instructions to change the base and bound for OS. Then, it is also responsible for the address-valid check. Finally, it offers and raises exceptions for the illegal memory access and process.
- OS is responsible for creating process and use CPU to find a suitable space for its address space in memory, and cleans up and recycle the space in the memory when the process terminated. Secondly, it is used to manage the context switch. Thirldly, it offers the exception handlers when CPU hardware raises exceptions.

Q2

- size of chunks:
    - segmentation: differnt sizes
    - paging: same size each page
- management of the free spaces:
    - segmentation: easy to have external fragments that needs rearranging
    - paging: get fragmentation problem better but too slow
- status bits:
    - segmentation: not found
    - paging: cpu will use zero flag
- protection bits:
    - segmentation:per segment
    - paging: per page

Q3

- because the page size is 8kB, so physical offset of VA is log(8*1024)=13 bit.
- because the VA is 46bit , so page table offset =46-13=33 bit.
- whatever the x86 PTE, or without those other conditional bit, the size of a page table is 2^x(x is the number of PPN bit), it need x bits for that Page Table offset.

- get x86 as a example,there are 20 bits of PPN, so it needs 2 levels of Page table.((int)33/20+1=2)
- so it needs at least 2 levels of the page table

Q4

- (a)page size=2^12=4kB Page table size(Maximum)= 2^20= 1MB
- (b)
    - first level page bit is 0x30C, offset=770
    - second level page bit is 0x266, offset=1707

Q5

- code:

```
//-----------------------合并空闲块-------------------
list_entry_t *le = &free_list;
le = list_next(le);
while (1)
{
    if (le == &free_list)
    {
        break;
    }
    struct Page *page = le2page(le, page_link);
    list_entry_t *le_next = list_next(le);
    if (le_next == &free_list)
    {
        break;
    }
    struct Page *page_next = le2page(le_next, page_link);
    if (page + page->property == page_next)
    {
        page->property += page_next->property;
        page_next->property = 0;
        // SetPageReserved(page_next);
        // set_page_ref(p, 0);
        // assert(list_next(le) != &free_list);
        // cputs("assert1 success\n");
        le->next = le_next->next;
        (le->next)->prev = le;
        // assert(list_next(le) == &free_list);
        // cputs("assert2 success\n");

        ClearPageProperty(page_next);
        // cprintf("page:%p and page_next:%p\n", page, page_next);
    }
    else
        le = le_next;
}

//-------------------------------------------------
```

(in default_pmm.c)

- result:

```
11911609JohnnyGe@johnny-Ge-WXX9:~/OS/labs/lab6/lab6 (1)$ make qemu
+ cc kern/init/entry.S
+ cc kern/init/init.c
+ cc kern/libs/stdio.c
+ cc kern/debug/panic.c
+ cc kern/debug/kdebug.c
+ cc kern/debug/kmonitor.c
+ cc kern/driver/clock.c
+ cc kern/driver/console.c
+ cc kern/driver/intr.c
+ cc kern/trap/trap.c
+ cc kern/trap/trapentry.S
+ cc kern/mm/pmm.c
+ cc kern/mm/default_pmm.c
+ cc kern/mm/best_fit_pmm.c
+ cc libs/string.c
+ cc libs/printfmt.c
+ cc libs/readline.c
+ cc libs/sbi.c
+ ld bin/kernel
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpenSBI v0.6

   ____                    ____  ____  _____
  / __ \                  / ___||  _ \|_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

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: default_pmm_manager
physcial memory map:
  memory: 0x0000000007e00000, [0x0000000080200000, 0x0000000087ffffff].
check_alloc_page() succeeded!
```

Q6

```
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;
    int min = 1000000;
    while ((le = list_next(le)) != &free_list)
    {
        struct Page *p = le2page(le, page_link);
        if (p->property >= n)
        {
            if (p->property == n)
            {
                page = p;
                break;
            }
            else if (p->property < min)
            {
                page = p;
                min = p->property;
            }
        }
    }
    if (page != NULL)
    {
        list_entry_t *prev = list_prev(&(page->page_link));
        list_del(&(page->page_link));
        // prev->next = (prev->next)->next;
        // (prev->next)->prev = prev;

        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;
}
```

- code: (in best_fi_pmm.c,other balnks is same as default_pmm.c)

- result:

```
11911609JohnnyGe@johnny-Ge-WXX9:~/OS/labs/lab6/lab6 (1)$ make qemu
+ cc kern/init/entry.S
+ cc kern/init/init.c
+ cc kern/libs/stdio.c
+ cc kern/debug/panic.c
+ cc kern/debug/kdebug.c
+ cc kern/debug/kmonitor.c
+ cc kern/driver/clock.c
+ cc kern/driver/console.c
+ cc kern/driver/intr.c
+ cc kern/trap/trap.c
+ cc kern/trap/trapentry.S
+ cc kern/mm/pmm.c
+ cc kern/mm/default_pmm.c
+ cc kern/mm/best_fit_pmm.c
+ cc libs/string.c
+ cc libs/printfmt.c
+ cc libs/readline.c
+ cc libs/sbi.c
+ ld bin/kernel
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpenSBI v0.6

   ____                    ____  ____  _____
  / __ \                  / ___||  _ \|_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

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].
check_alloc_page() succeeded!
```