System Software Laboratory

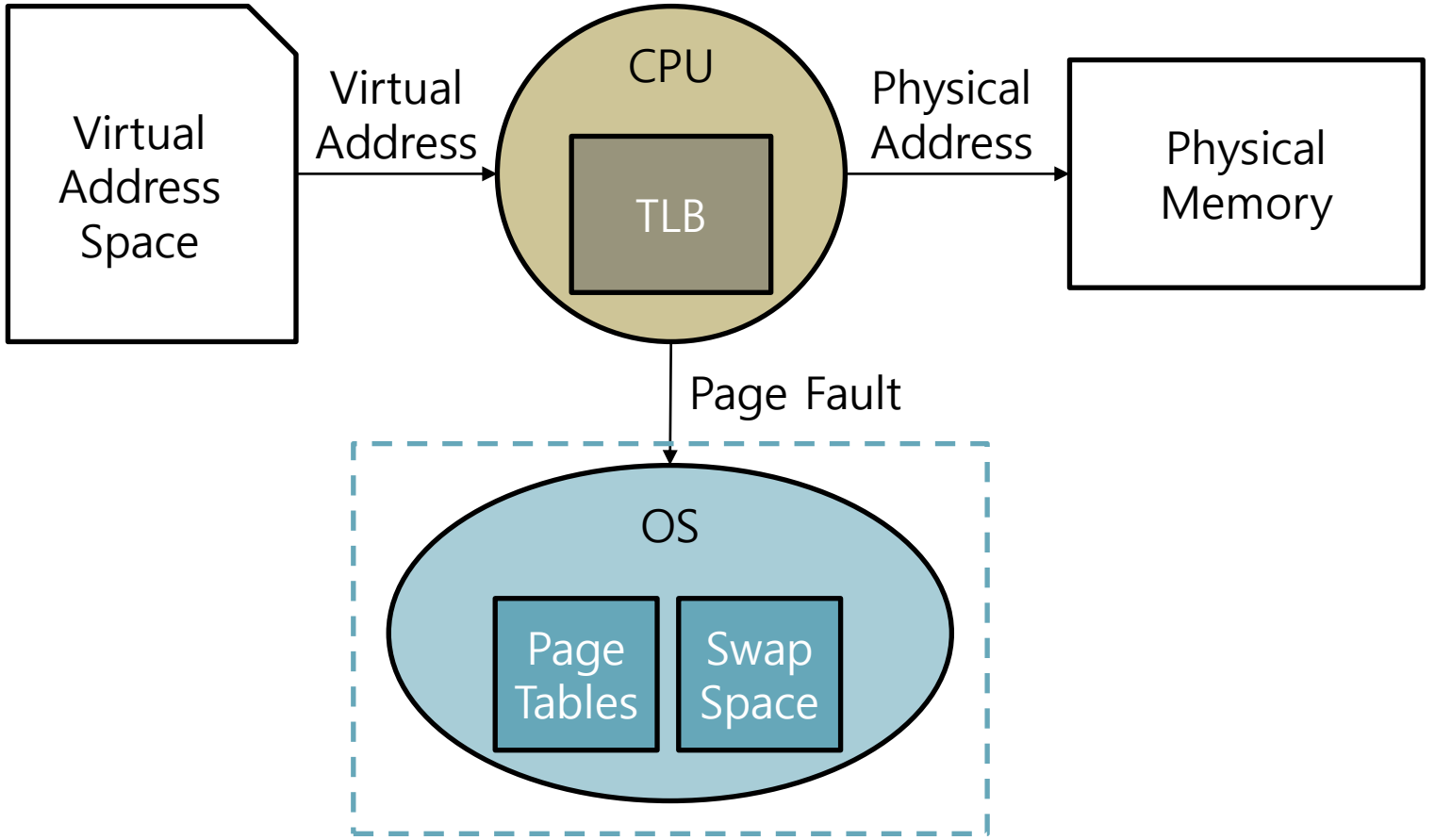
KU KONKUK UNIVERSITY

Operating Systems

Assignment #2: KU_MMU

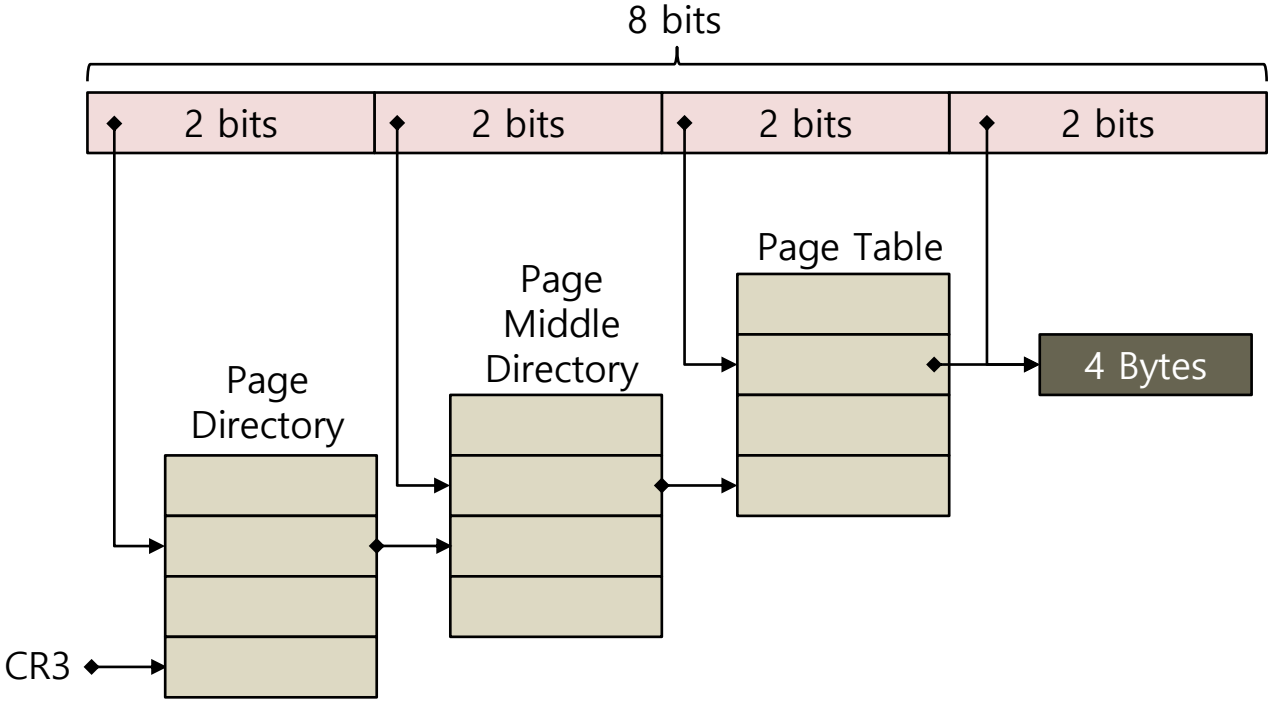
Hyun-Wook Jin
System Software Laboratory
Dept. of Computer Science and Engineering
Konkuk University
jinh@konkuk.ac.kr

KU_MMU


Virtual Address Space
Virtual Address
CPU
TLB
Physical Address
Physical Memory
Page Fault
OS
Page Tables
Swap Space
Assignment #2

Addressing

- 8-bit addressing
  - Address space: 256 Bytes
  - Page size: 4 Bytes
  - PDE/PTE: 1 Byte



PDE/PTE

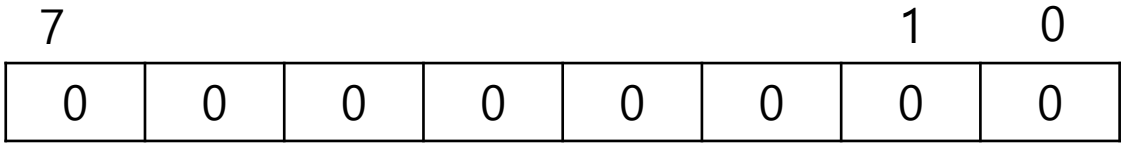
- PDE/PTE

  - PDE and PTE have the same format
  - Unmapped PTE is filled with zeros

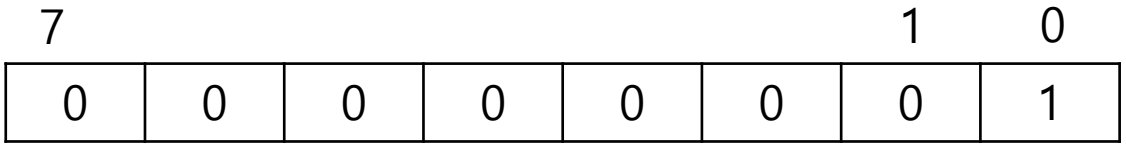
  - Swap space: 512 Bytes (=2^7 * 4 Bytes)
  - Offset starts from 1
    - 0^{th} page in swap space is not used
  - Present bit is 0

PDE/PTE

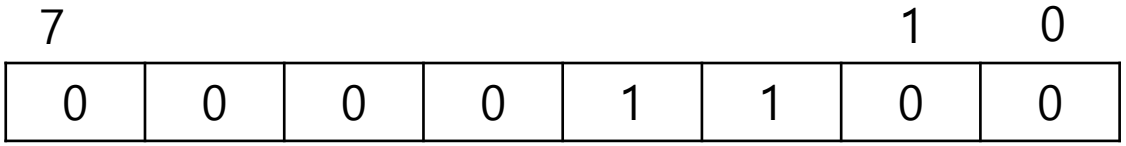
- Examples



- Virtual page is neither mapped nor swapped out



- Virtual page is mapped to Page frame 0 (occupied by OS)



- Virtual page is swapped out to 6th page in swap space

Page Fault Handler

- int ku_page_fault (char pid,
  char va)
  - Handling a page fault caused by demand paging or swapping
    - Page replacement policy: FIFO
    - Pages for page directories, page middle directories, and page tables are not swapped out
  - pid: process id
  - va: virtual address
  - Return value
    - 0: success
    - -1: fail

Miscellaneous Functions

- void *ku_mmu_init (unsigned int mem_size, unsigned int swap_size)
  - Resource initialization function
    - Will be called only once at the initialization phase
  - mem_size: physical memory size in bytes
    - You need to allocate a memory space and manage a free list
      - Assume that page frame 0 is occupied by OS
    - Do consider the memory space consumed by page directories and tables
  - swap_size: swap disk size in bytes
    - Allocate a memory space instead of real disk space
  - Return value
    - Pointer (i.e., address) to the allocated memory area that simulates the physical memory
    - 0: fail

Miscellaneous Functions

- int ku_run_proc (char pid,
  struct ku_pte **ku_cr3)
  - Performs context switch
    - If pid is new, the function creates a process and its page directory
  - pid: pid of the next process
  - ku_cr3: stores the base address of the page directory for the next process
    - Points an 8-bit PDE
    - Its value should be changed appropriately by this function
  - Return value
    - 0: success
    - -1: fail

Provided Files

- ku_cpu.c
  - An example of test code
- ku_trav.o
  - Object file for `ku_traverse()`

ku_cpu.c

```
int main(int argc, char *argv[])
{
        FILE *fd=NULL;
        char fpid, pid=0, va, pa;
        unsigned int pmem_size, swap_size;
        void *ku_cr3, *pmem=NULL;

        if(argc != 4){
                printf("ku_cpu: Wrong number of arguments\n");
                return 1;
        }

        fd = fopen(argv[1], "r");
        if(!fd){
                printf("ku_cpu: Fail to open the input file\n");
                return 1;
        }
```

ku_cpu.c

```
pmem_size = strtol(argv[2], NULL, 10);
swap_size = strtol(argv[3], NULL, 10);
pmem = ku_mmu_init(pmem_size, swap_size);
if(!pmem){
    printf("ku_cpu: Fail to allocate the physical mem\n");
    ku_mmu_fin(fd, pmem);
    return 1;
}

while(fscanf(fd, "%hhd %hhd", &fpid, &va) != EOF){

    if(pid != fpid){
        if(ku_run_proc(fpid, &ku_cr3) == 0)
                pid = fpid; /* context switch */
        else{
                printf("ku_cpu: Context switch is failed\n");
                ku_mmu_fin(fd, pmem);
                return 1;
        }
    }
```

ku_cpu.c

```
        pa = ku_traverse(ku_cr3, va, pmem);
        if(pa == 0){
                if(ku_page_fault(pid, va) != 0){
                        printf("ku_cpu: Fault handler is failed\n");
                        ku_mmu_fin(fd, pmem);
                        return 1;
                }
                printf("[%d] VA: %hhd -> Page Fault\n", pid, va);

                /* Retry after page fault */
                pa = ku_traverse(ku_cr3, va, pmem);
                if(pa == 0){
                        printf("ku_cpu: Addr tanslation is failed\n");
                        ku_mmu_fin(fd, pmem);
                        return 1;
                }
        }
                printf("[%d] VA: %hhd -> PA: %hhd\n", pid, va, pa);
} /* end of while */
```

Submission

- Source codes and documents
  - Source files
    - ku_mmu.h
    - Use the 'ku_mmu_' prefix for static variables if needed
    - Will be compiled and tested on a Linux machine
      - Don't use a special library
  - Document
    - Basic design
    - Description for important functions

Function Name	Functionality	
	Parameters	
	Return Value	

Submission

- Submit your homework through eCampus
  - Deadline: 5/17 Sunday Midnight (11:59 pm)
- Cheating, plagiarism, and other anti-intellectual behavior will be dealt with severely