National Taiwan University

CSIE3310 - Operating Systems

Machine Problem 2
Demand Paging and Swapping

Department of Computer Science and Information Engineering

TA e-mail: ntuos@googlegroups.com

TA hours: Tue. & Thu. 13:00-14:00 CSIE Building R442 or B04 (Please knock the door)

01 / TA Contact

林祥瑞 / Jerry Lin
- d10922013@ntu.edu.tw
- At room B04

劉昕祐 / Xin-You Liu
- r10944004@csie.ntu.edu.tw
- At room R442

Better to ask your questions on homework discussion page on NTUCOOL instead of emails.

Outline

01

PART ONE

Summary

01 / Summary

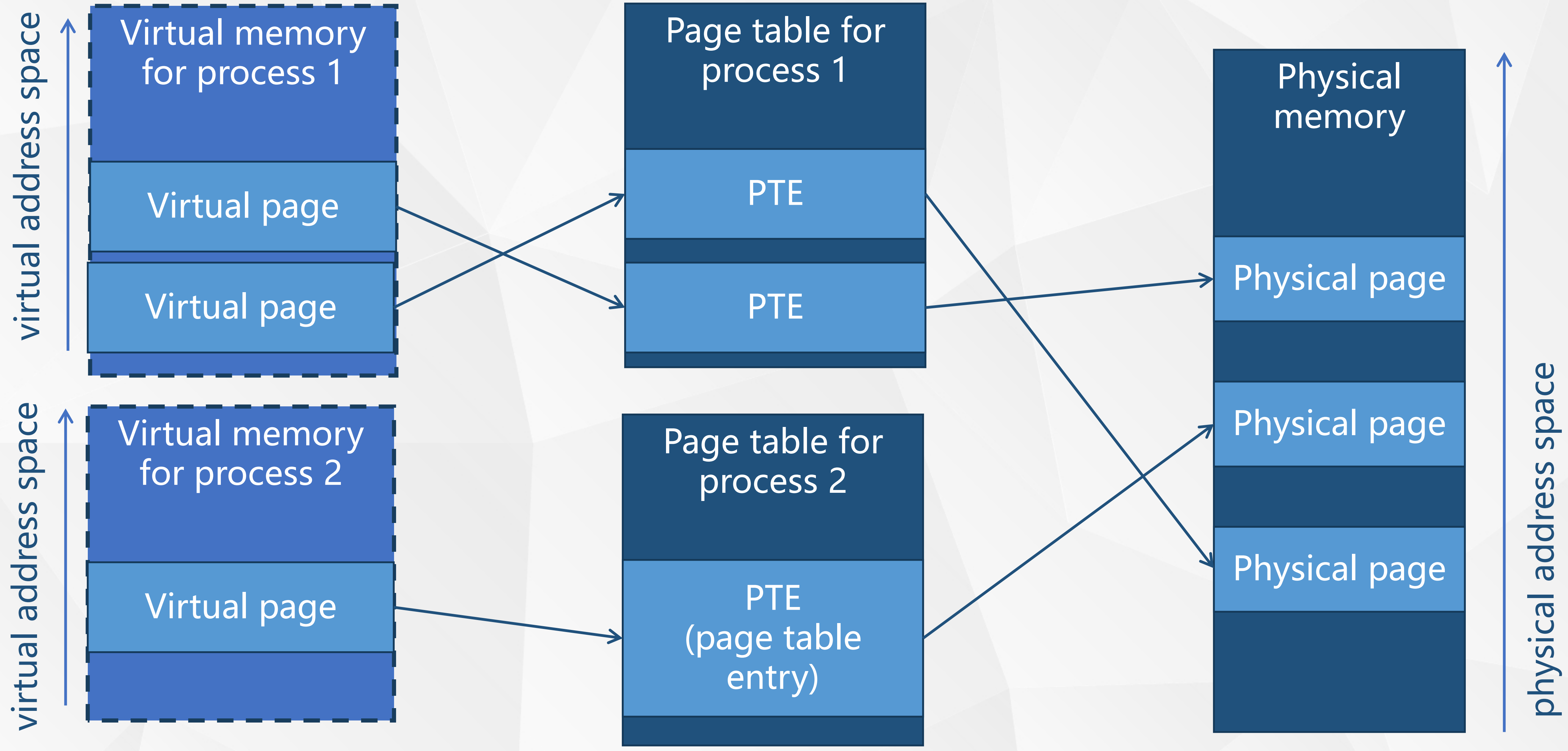
❖ **Virtual Memory:**

- An isolated and abstracted memory space for each process.
- A portion of VM pages are mapped to physical memory through per-process page table.

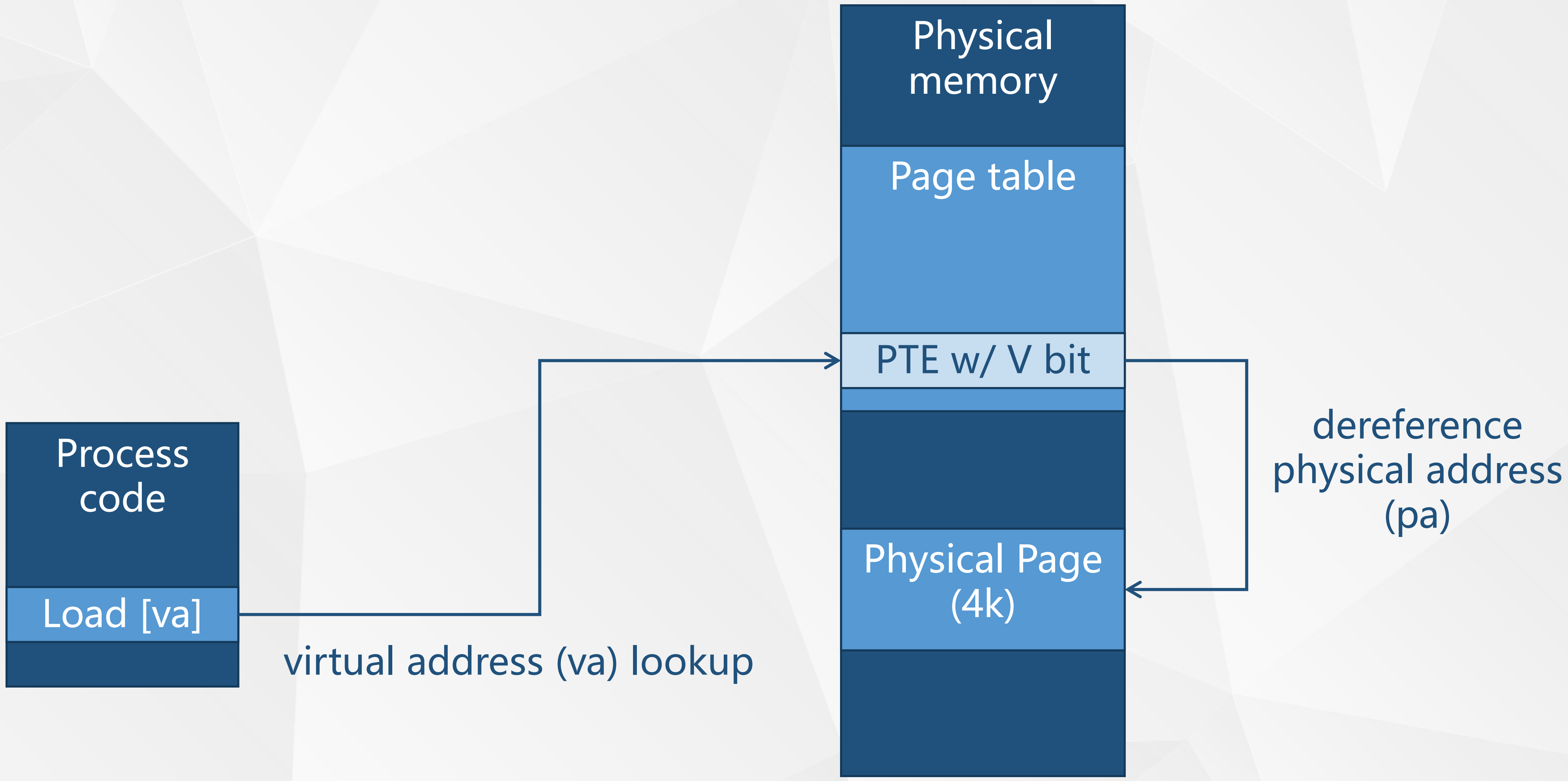
❖ **Swapping :**

- Allow VM pages not only mapped to physical memory pages, can also be mapped to blocks on a disk.
- The OS can swap out “cold” memory pages to disk blocks, or swap in disk blocks to physical memory when needed.

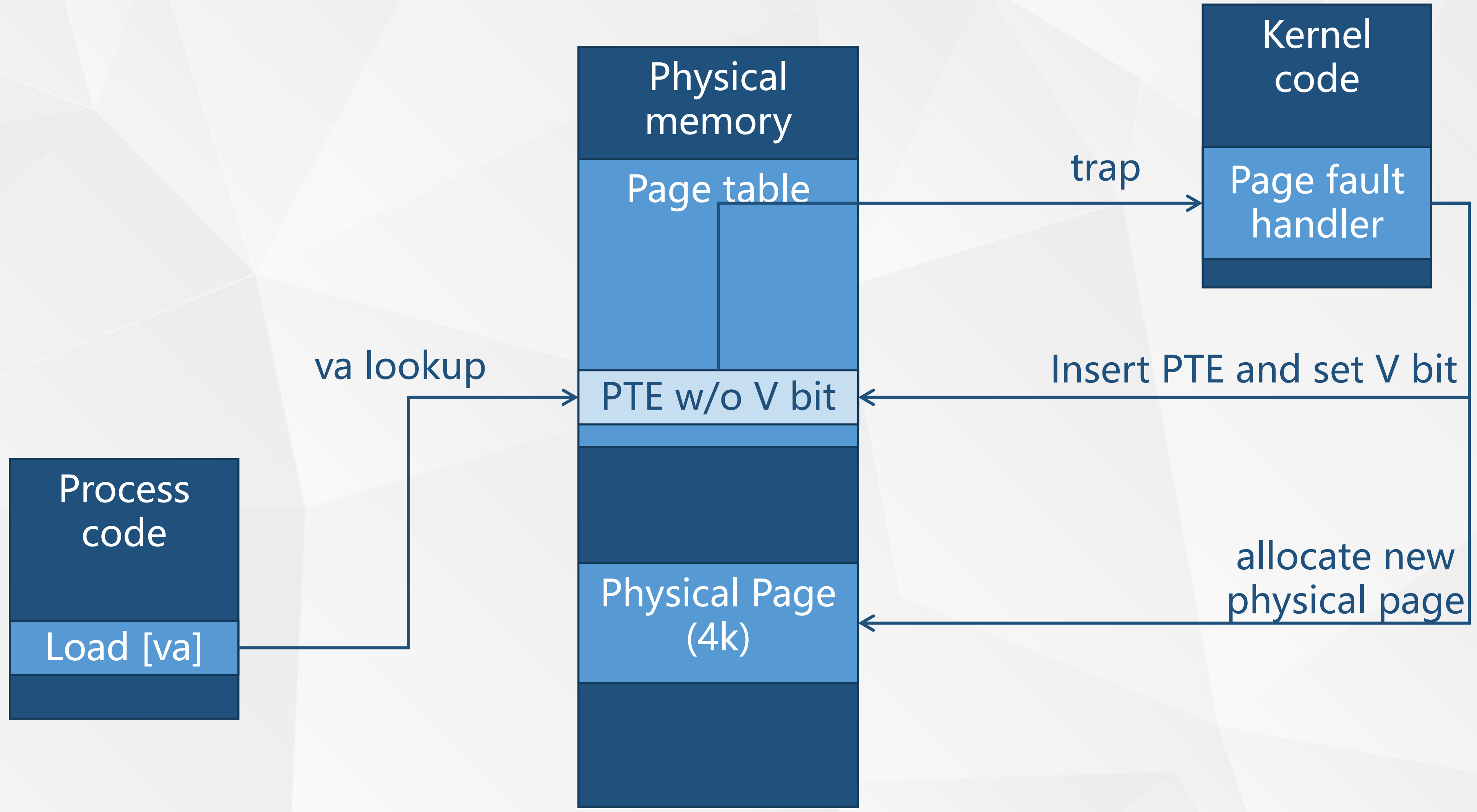
01 / Logical View of Virtual Memory



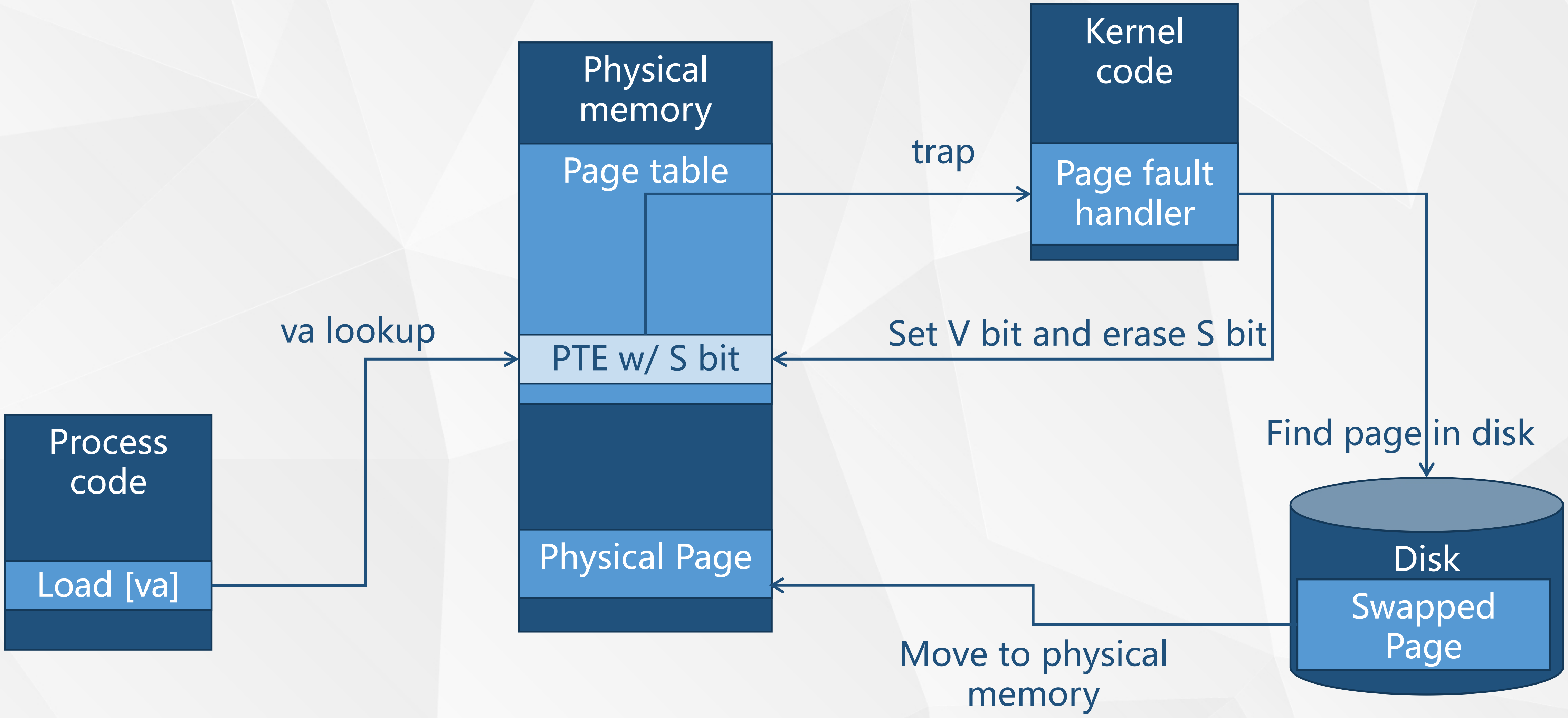
01 / Page Hit Workflow



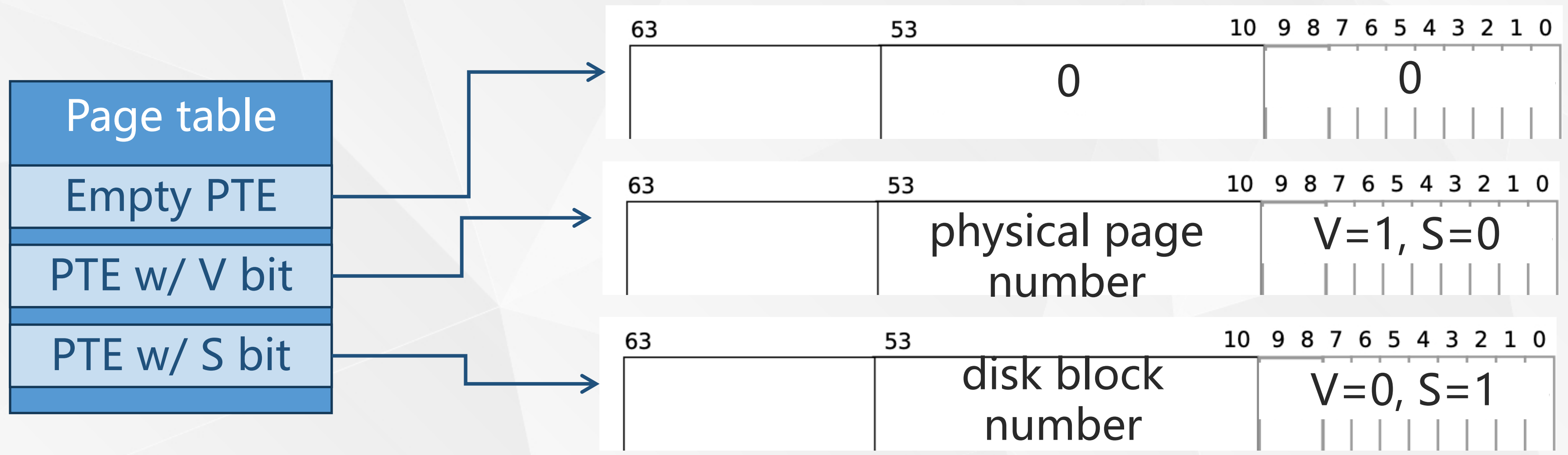
01 / Page Miss Workflow

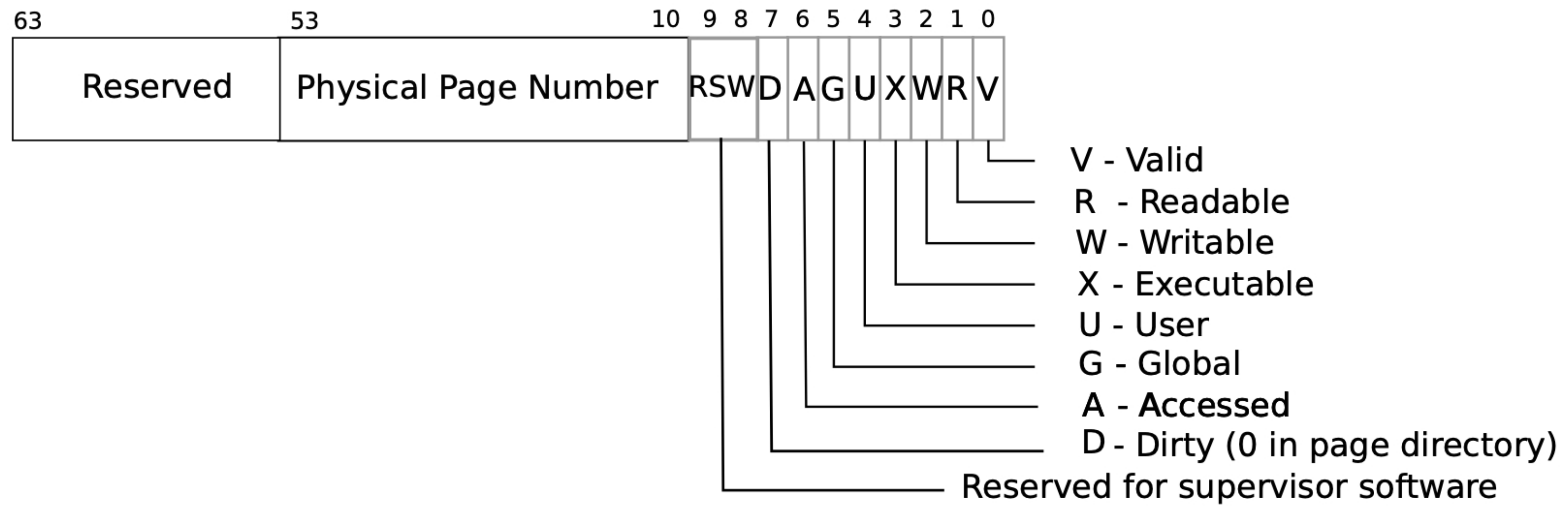


01 / Page Miss w/ Swap



01 / Page Table Entry Format





01 / Summary

- **Machine Problem 2 Goal:**
  - **Demand Paging :**

    The physical memory is allocated on demand only when the virtual pages are accessed.
  - **Swapping :**

    A technique to store memory pages on a disk.

  – With proper memory management, OS can maintain processes with large virtual memory spaces but small physical memory in use.

- MP2 assignment will add “Demand Paging” and “Swapping” to existing page table on xv6.

02

PART TWO

Launching

Launching

Launching Docker

- **Launching Docker Image of MP2**

1. Download the MP2.zip from NTUCOOL, unzip it, and enter it.

```
$ unzip MP2.zip
$ cd mp2
```

2. Pull Docker image from Docker Hub

```
$ docker pull ntuos/mp2
```

3. Use docker run to start the process in a container and allocate a TTY for the container process.

```
$ docker run -it -v $(pwd):/home/mp2/xv6 ntuos/mp2
```

4. Check the environment in the Docker container

```
$ cat /etc/os-release
```

03

PART THREE

Assignment

03 / Scoring

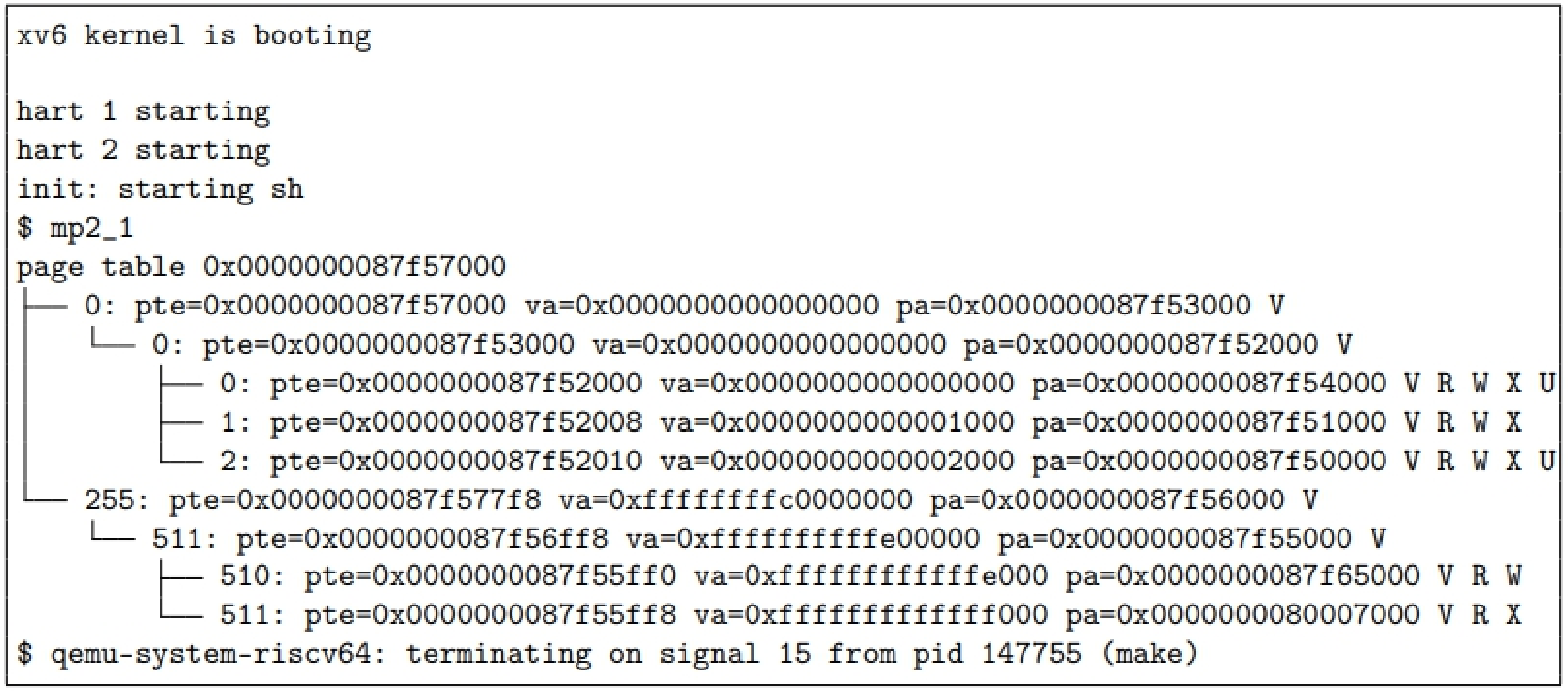
- Public program tests (70%)
- Private program tests (15%)
- Required report (15%)
- Bonus report (10%)

- Public program test code is shipped with MP2.zip.
- Private program tests will be disclosed after the deadline.

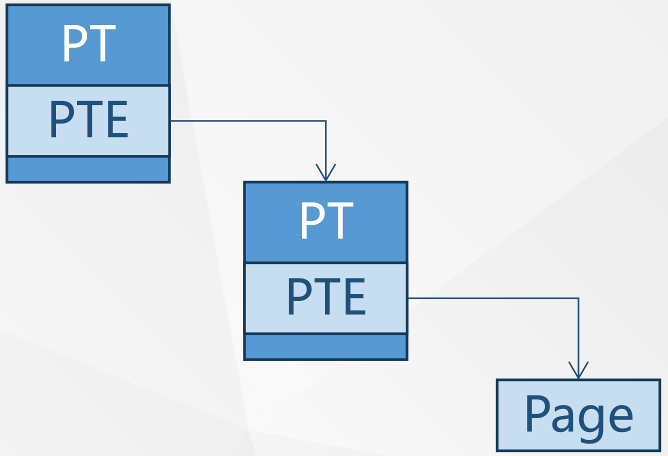
Assignment

1. Print a Page Table (Public Test 5% + Report 5%)

- Traverse the 3-level page table on xv6.
- Print physical addresses and flags, and compute corresponding virtual addresses.



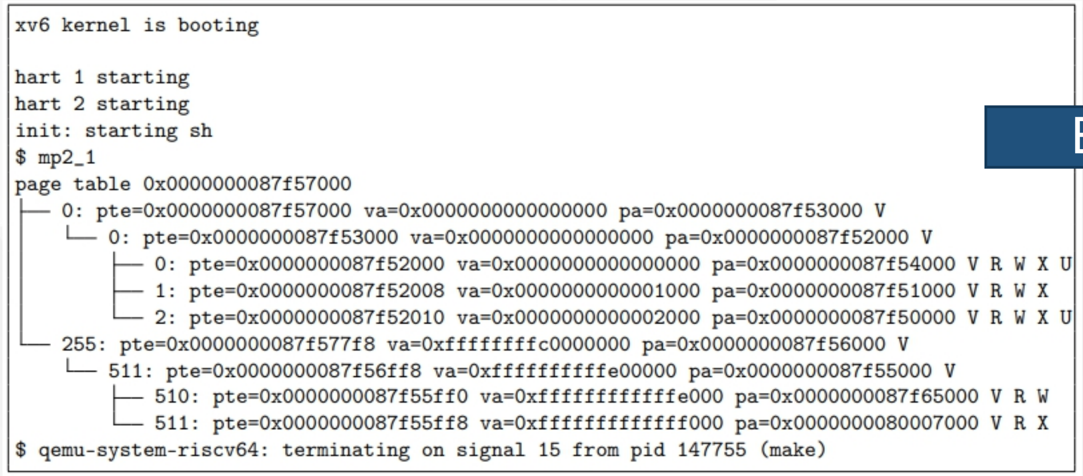
```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ mp2_1
page table 0x0000000087f57000
├── 0: pte=0x0000000087f57000 va=0x0000000000000000 pa=0x0000000087f53000 V
│   └── 0: pte=0x0000000087f53000 va=0x0000000000000000 pa=0x0000000087f52000 V
│       ├── 0: pte=0x0000000087f52000 va=0x0000000000000000 pa=0x0000000087f54000 V R W X U
│       ├── 1: pte=0x0000000087f52008 va=0x0000000000001000 pa=0x0000000087f51000 V R W X
│       └── 2: pte=0x0000000087f52010 va=0x0000000000002000 pa=0x0000000087f50000 V R W X U
└── 255: pte=0x0000000087f577f8 va=0xffffffffc0000000 pa=0x0000000087f56000 V
    └── 511: pte=0x0000000087f56ff8 va=0xffffffffffe00000 pa=0x0000000087f55000 V
        ├── 510: pte=0x0000000087f55ff0 va=0xffffffffffffe000 pa=0x0000000087f65000 V R W
        └── 511: pte=0x0000000087f55ff8 va=0xfffffffffffff000 pa=0x0000000080007000 V R X
$ qemu-system-riscv64: terminating on signal 15 from pid 147755 (make)
```



03 / Assignment

1. Print a Page Table (Public Test 5% + Report 5%)

Report Part: (5%) Please answer the following questions:



```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ mp2_1
page table 0x0000000087f57000
├── 0: pte=0x0000000087f57000 va=0x0000000000000000 pa=0x0000000087f53000 V
│   └── 0: pte=0x0000000087f53000 va=0x0000000000000000 pa=0x0000000087f52000 V
│       ├── 0: pte=0x0000000087f52000 va=0x0000000000000000 pa=0x0000000087f54000 V R W X U
│       ├── 1: pte=0x0000000087f52008 va=0x0000000000001000 pa=0x0000000087f51000 V R W X
│       └── 2: pte=0x0000000087f52010 va=0x0000000000002000 pa=0x0000000087f50000 V R W X U
└── 255: pte=0x0000000087f577f8 va=0xffffffffc0000000 pa=0x0000000087f56000 V
    └── 511: pte=0x0000000087f56ff8 va=0xffffffffffe00000 pa=0x0000000087f55000 V
        ├── 510: pte=0x0000000087f55ff0 va=0xffffffffffffe000 pa=0x0000000087f65000 V R W
        └── 511: pte=0x0000000087f55ff8 va=0xfffffffffffff000 pa=0x0000000080007000 V R X
$ qemu-system-riscv64: terminating on signal 15 from pid 147755 (make)
```

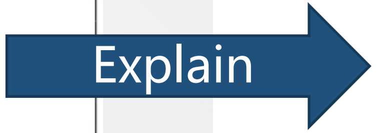
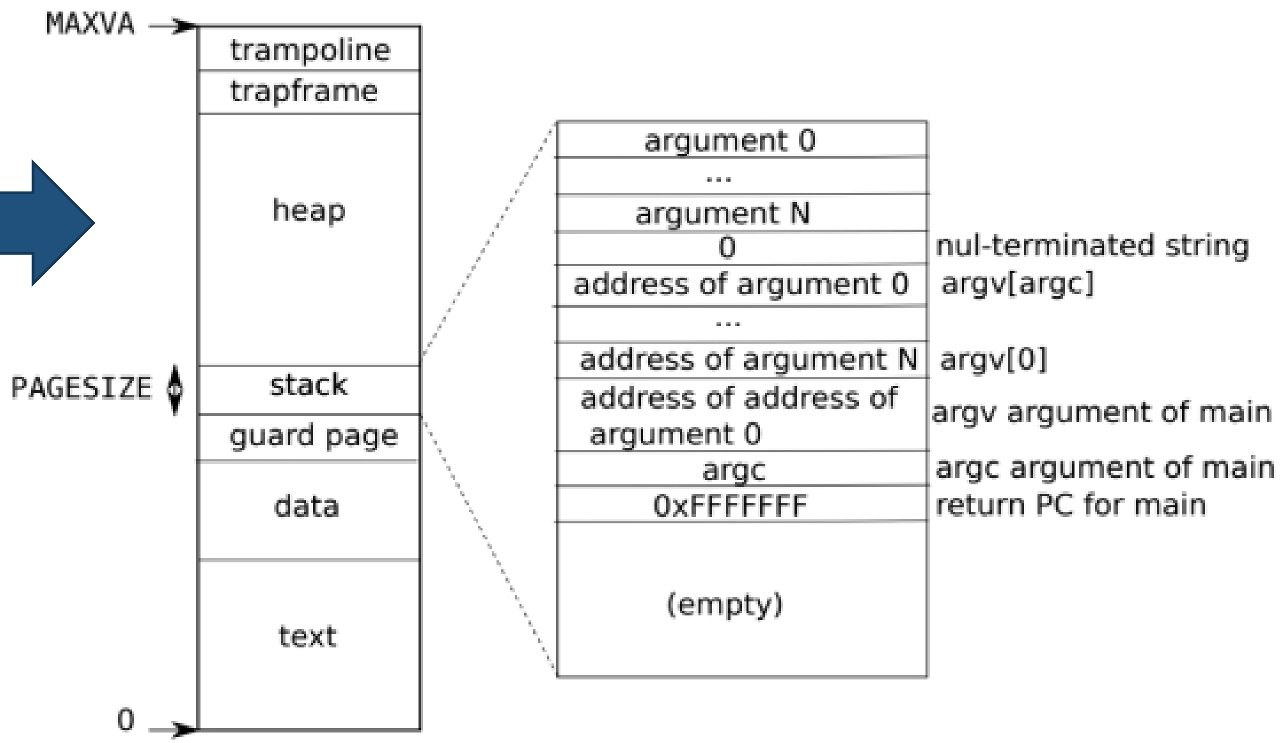


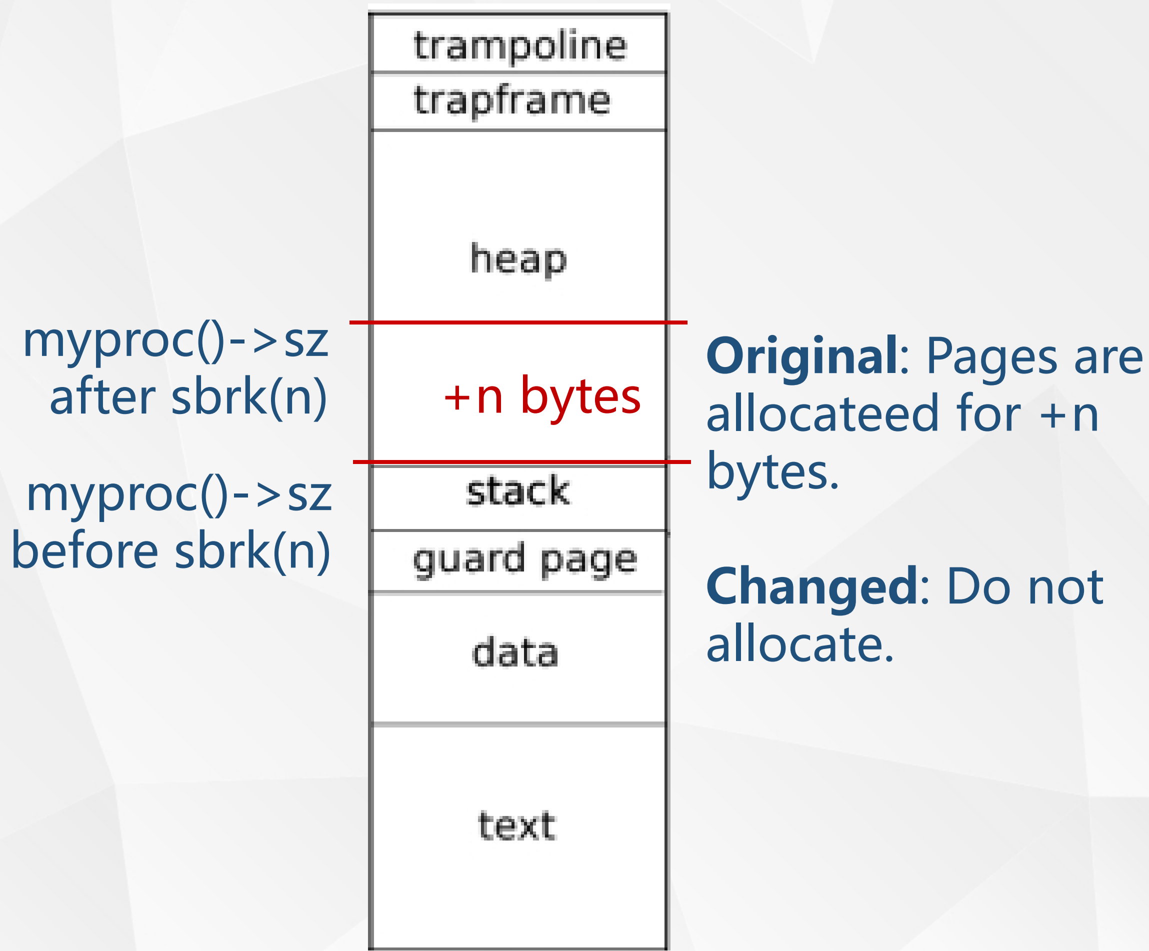
Figure 1: The virtual memory layout



03 / Assignment

2. Generate a Page Fault (Public Test 20%)

- sbrk(n) adds n bytes to process memory if n > 0, or decreases by |n| bytes if n < 0.
- Change sbrk(n) so that it does not allocate.
- Implement page fault handler so that physical pages are allocated only when page fault.


trampoline
trapframe
heap
myproc()->sz after sbrk(n)
+n bytes
myproc()->sz before sbrk(n)
stack
guard page
data
text
Original: Pages are allocateed for +n bytes.
Changed: Do not allocate.

03 / Assignment

3. Demand Paging and Swapping (Public Test 45% + Private Test 15% + Report 10%)

Add the `madvise()` system call with this signature:

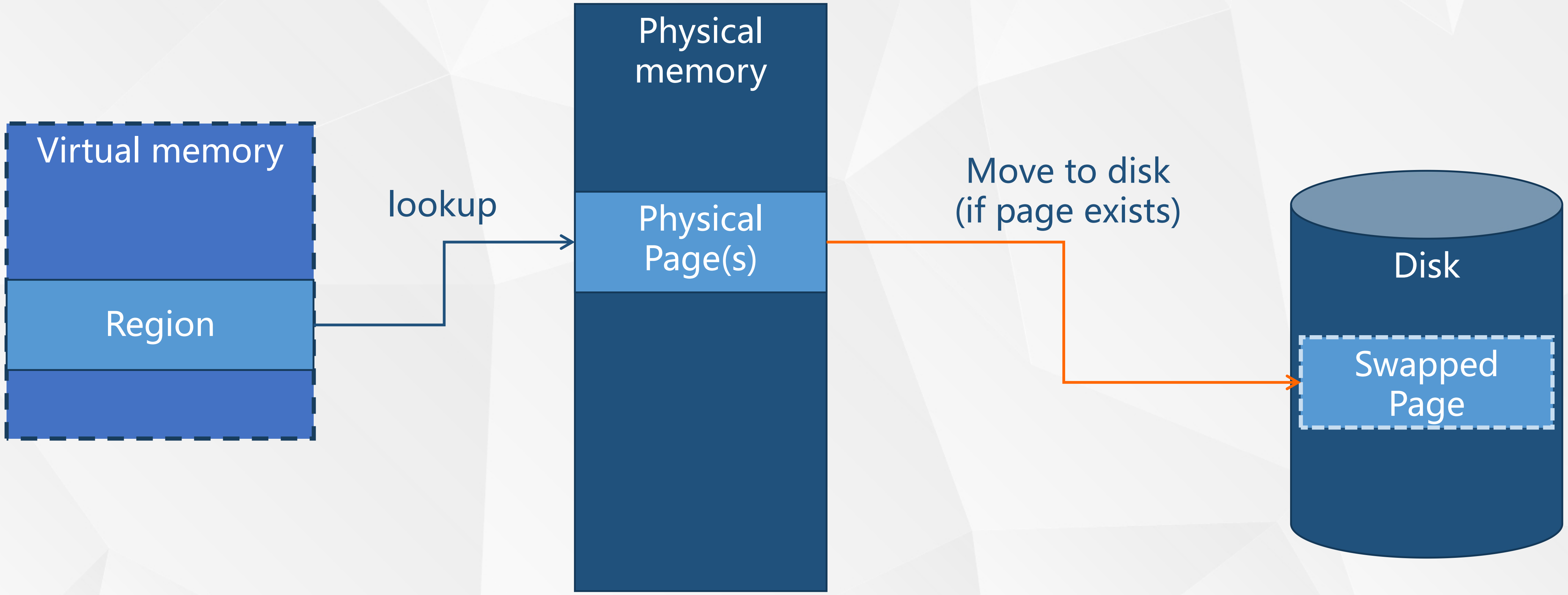
```
int madvise(char *base, int length, int option)
```

```
option :=       MADV_NORMAL,
                MADV_WILLNEED,
                MADV_DONTNEED
```

base and length describes the [base, base + length) virtual memory region in bytes.
Return 0 if success, otherwise -1. Invalid range should return -1.

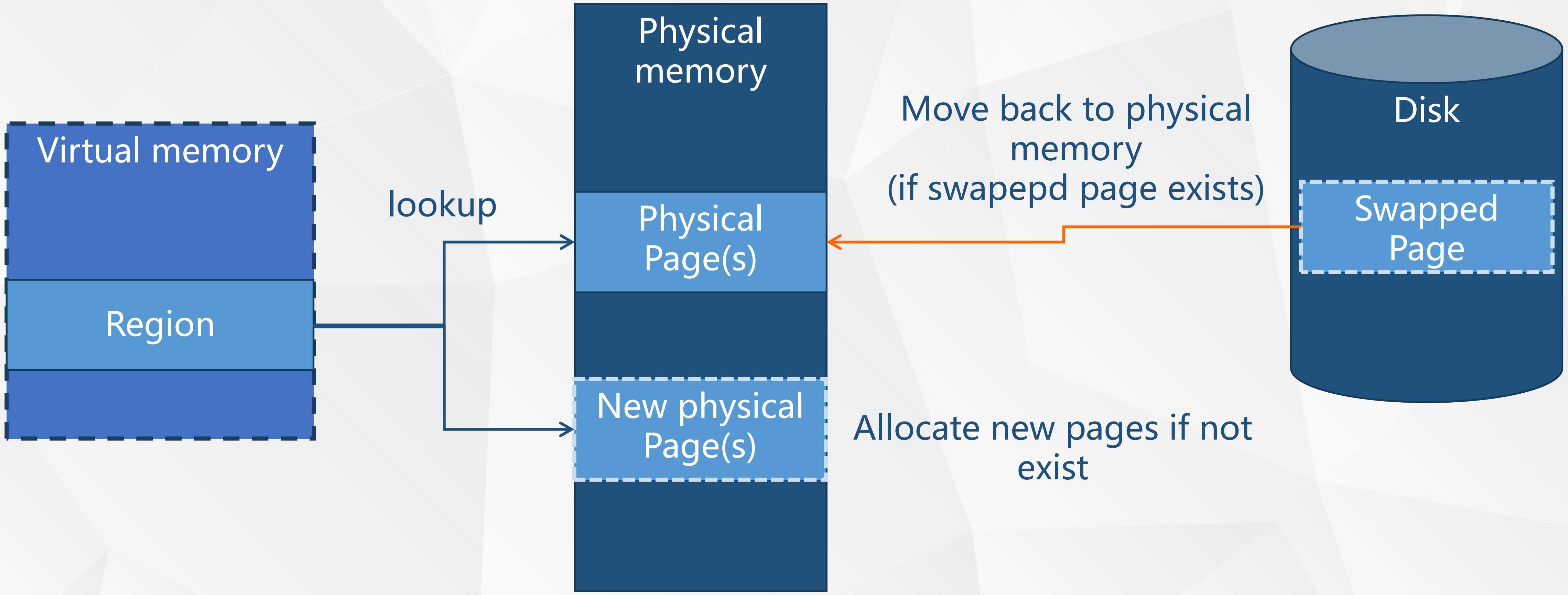
03 / Assignment

MADV_DONTNEED: Move affected pages to the disk.



Assignment

MADV_WILLNEED: Move affected pages from the disk to physical memory.



03 / Assignment

3. Demand Paging and Swapping
(Public Test 45% + Private Test 15% + Report 10%)

Program Part I (Public Test 17%)
Test MADV_NORMAL and ragne checks in madvise() + MADV_DONTNEED test

Program Part II (Public Test 28%)
MADV_DONTNEED + MADV_WILLNEED test

Program Part III (Private Test 15%)
MADV_DONTNEED + MADV_WILLNEED + page fault on swapped pages

Report (10%)

03 / Assignment

4. Bonus Reports (10%)

Pros and Cons of Demand Paging (Bonus + 5%):

Effective Memory Access Time Analysis (Bonus + 5%)

04

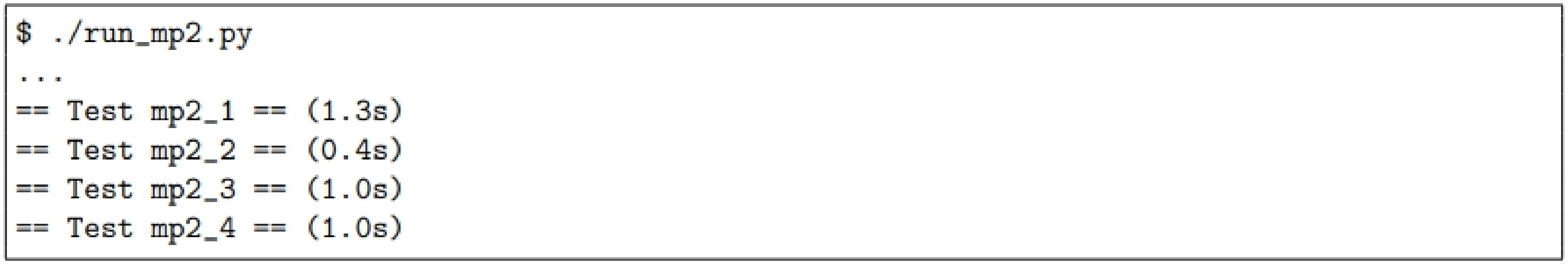
PART FORE

Test Programs

04 / Test Programs

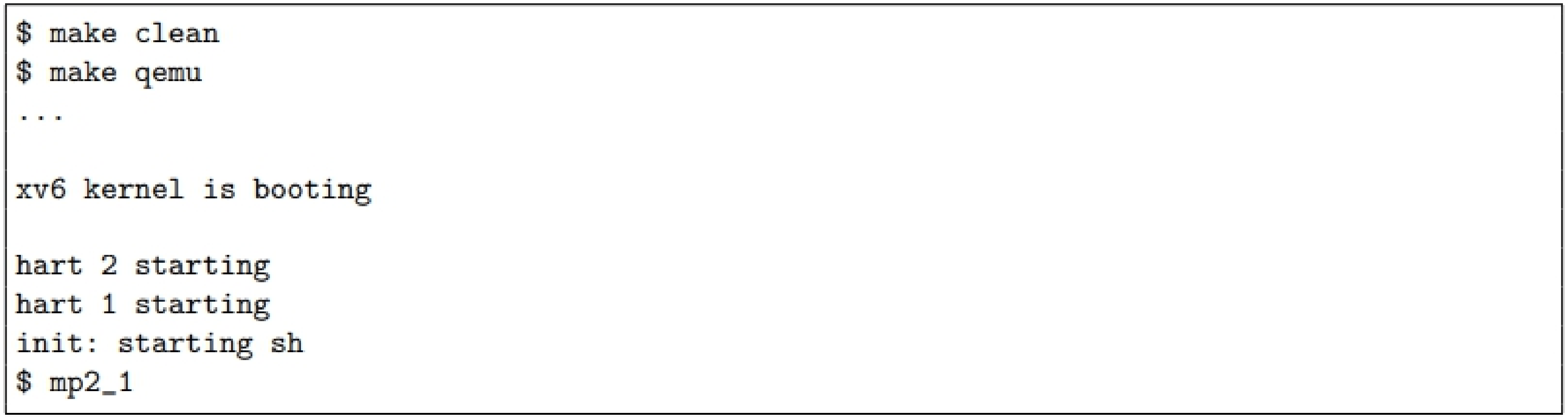
- **Four public test programs, respectively named mp2_N with N = 1,… , 4 , source code at /user/mp2_N.c**

1. Run the command to launch all tests at once, and the output will be saved to mp2_N.out files.



```
$ ./run_mp2.py
...
== Test mp2_1 == (1.3s)
== Test mp2_2 == (0.4s)
== Test mp2_3 == (1.0s)
== Test mp2_4 == (1.0s)
```

2. To run an individual test program instead, run “make qemu” to enter the xv6 shell and “run mp2_N”.



```
$ make clean
$ make qemu
...

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ mp2_1
```

05

PART FIVE

Submission

05 / Submission

MP2 assignment deadline : April 4, 23:59:00

1. **Source code**:
   Submit your d08922025.zip to “Machine Problem 2”

2. **Report**:
   Submit one PDF file to "Report" named d08922025_mp2_report.pdf, for example.

3. **Bonus report (optional)**:
   Submit one PDF file to "Bonus Report" named d08922025_mp2_bonus.pdf, for example.