Lab 6: Page Mapping

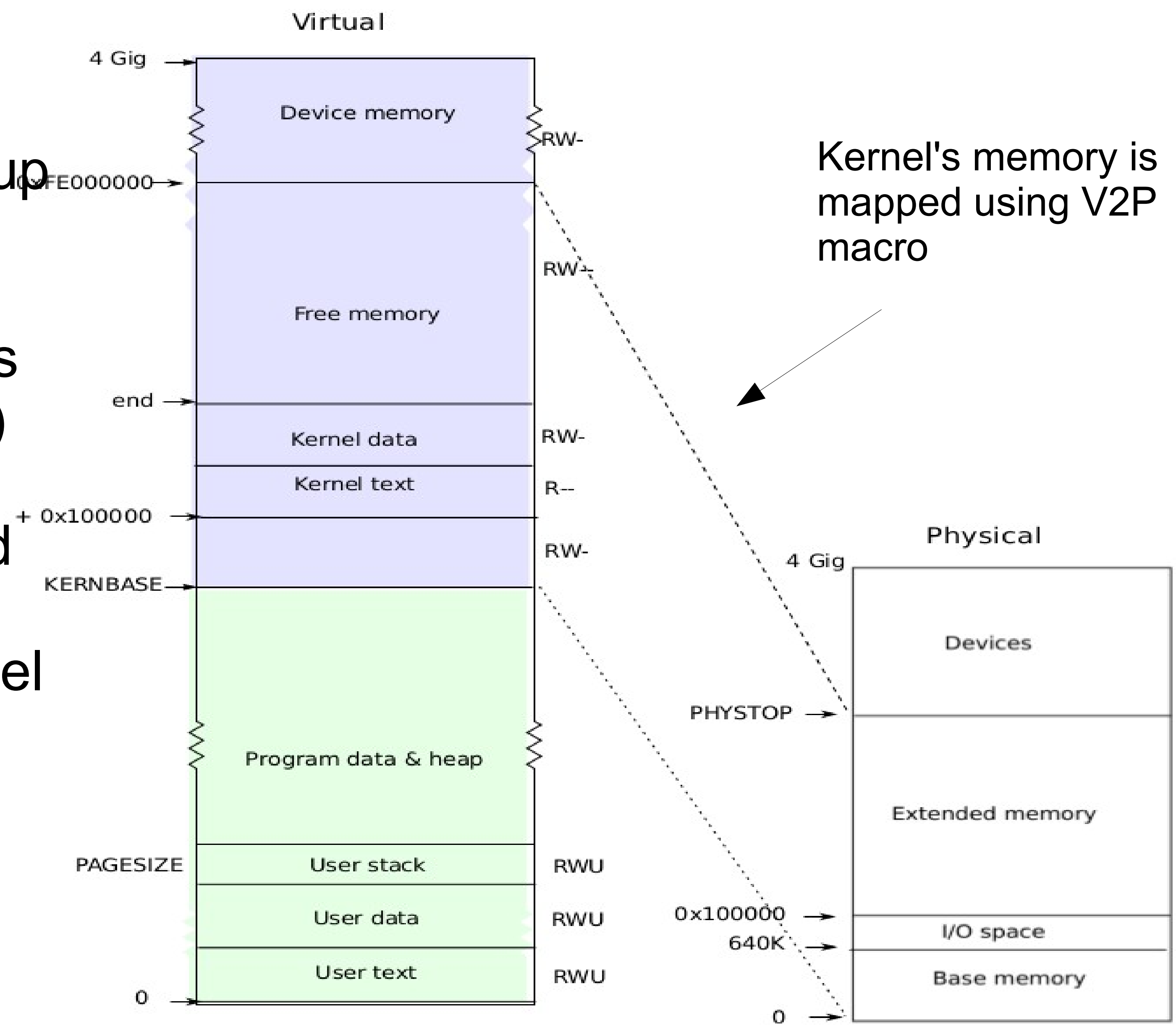
Instructor: Yaoqing Liu
TA: Garegin Grigoryan

Goals

- Understand virtual memory address space
- Learn how to initialize a virtual page from physical memory through kernel
- Learn how to map a virtual page to a physical page
- Learn how to verify if the page mapping is correctly conducted through a user program

Virtual address space

- User address space starts at 0 up to KERNBASE
- (0x80000000)
- If a process needs more memory: (a) xv6 finds a free physical page and allocates it using **kalloc(..)** via kernel (b) Maps virtual page to physical page (**mappages(..)**)

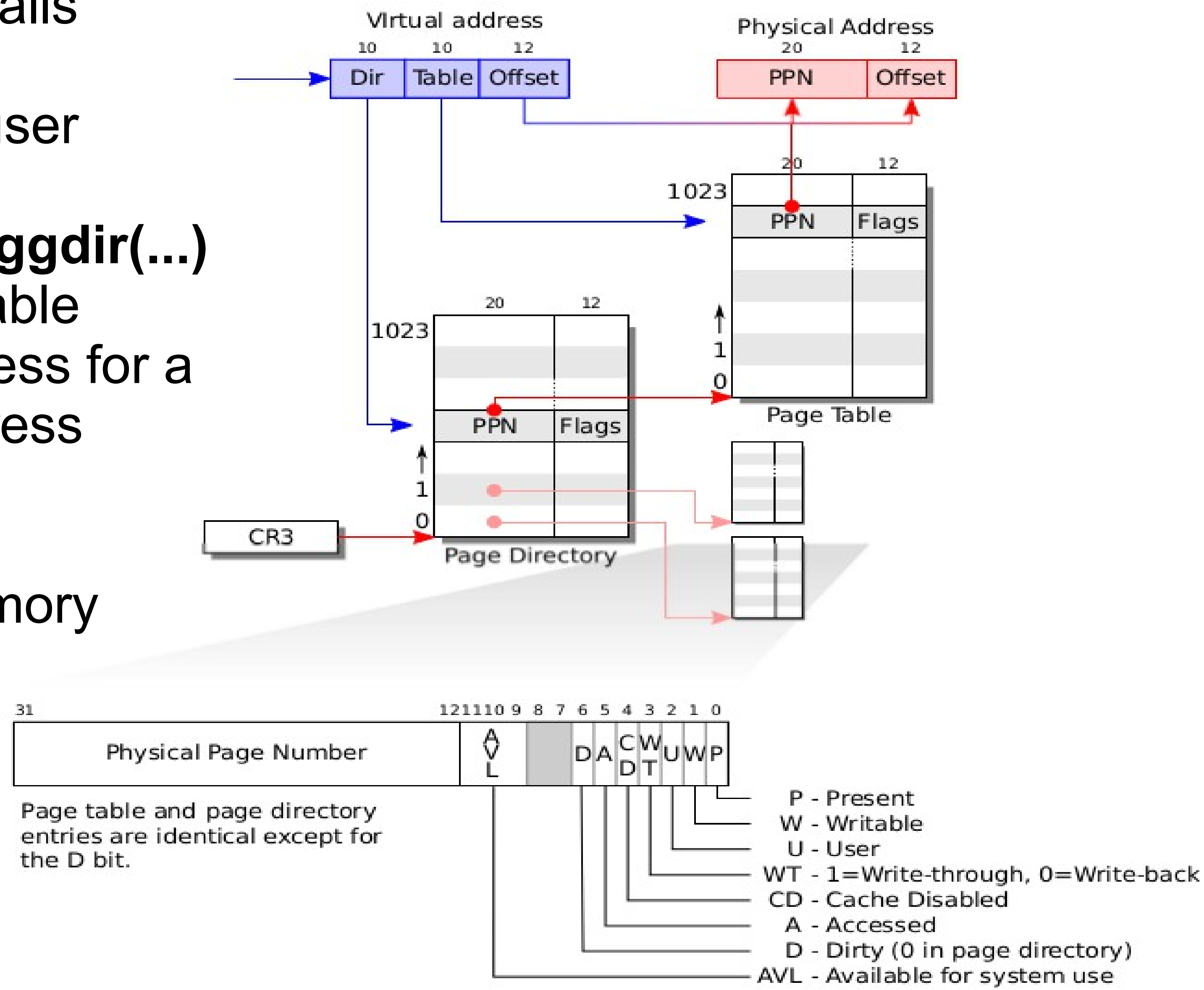


Page Mapping in xv6 (1)

- For the kernel's address space, mapping can be done using 2 macros:
  - V2P(a) – virtual to physical
  - P2V(a) – physical to virtual
- For the user's address space, the mapping is done by Page Table Entry (PTE)
- First 10 bits of the virtual address are used to find page directory entry
- Next 10 bits are used to find page table entry
- High 20 bits of the virtual address are replaced by the physical page number in the PTE

Page Mapping in xv6 (2)

- **mappages(...)** installs translations into the page table of a user process
  - calls **pte * walkpggdir(...)** to get the page table entry (PTE) address for a given virtual address
- Only after that memory may be accessed by a user program (depends on the flags in the PTE)



Your Tasks

- When system boots up, it should allocate a virtual page via kernel and initialize the first 4 bytes (integer type) of the page with your student ID
- Add a system call to map a virtual page in user space to the physical page that was allocated by the kernel
- The system call returns the starting address of the virtual page in user space
- Create a new user program to call the system call and verify that you're able to access to the returned address and read your student ID out

Step 1: Retrieve a new xv6

- Login to odin

$ssh YourName@odin.cslabs.clarkson.edu

$cd ~/cs444-s18/Lab6

- Download xv6.tar.gz file to your work directory Lab3

$wget http://people.clarkson.edu/~liu/CS444/Spring18/xv6.tar.gz

- Unzip it

$tar –xzvf xv6.tar.gz

Step 2: Allocating a Physical Page

- In vm.c, declare a global variable **int * ka_yourname**.
- In vm.c, write a function called **mem_init** that allocates a physical page to the **ka_yourname. Note:** you may need to cast void* pointer, returned by kalloc, into int*.
- Write your student ID to the kernel address
- In main.c, call that function from *main()*, before the processor's setup mpmain().

Step 3: Implementing Mapping

- Implement a system call **mem_yourname()** in vm.c file. The next slide describes the routine steps for implementing a system call.
- Use mappages(..) to add a PTE to the user process's page table.
  - Initialize a variable **va_yourname =** KERNBASE – 0x1000 as the virtual address in the user space
  - Convert **ka_yourname** to a physical address and use it as one of the arguments.
- The system call shall return the value of **va_yourname**.

Help: adding a new system call <u>sc</u>

1. **userspace/usys.S**
   - SYSCALL(*sc*)
2. **syscall.h**
   - #define SYS_*sc* <newID>
3. **sysproc.c**
   - int sys_*sc*(void) { return *sc*(); }
4. **userspace/user.h**
   - int *sc*(void);
5. **syscall.c**
   - extern int sys_sc(void);
   - [SYS_*sc*] sys_*sc*,
6. **defs.h**
   - int *sc*(void);
7. **Implement** int *sc*(void) {….your code...} in **vm.c**

Step 4: Verification

- Write a user program **mem_test_yourname**, that:
  - Calls **mem_yourname()**
  - Reads the integer value from the returned address
  - Prints both the address and the content stored in the address
- Do not forget to change your userspace/Makefile to run your testing program
- The printed content of the address should be your student ID

Submission

- Take screenshots for the source code, compiling process and the results
- Combine them into a PDF file
- Submit the file to moodle before Mar 5 (Monday), 11:55pm