15-213 Final Exam Review

Recitation 15: April 28, 2014
Neal Bhasin
Section M
OH: 5:30 – 8PM today

Updates

- Proxy lab due Tuesday
- No late days
- Can still add a partner on autolab
- Don’t rely on given tests

Final Exam

- Monday May 5 → Thursday May 8
  - Signups will work the same way as midterm
- Should take < 3 hours, can have up to 6
- Focus on post-midterm material
- There will be a programming question!
- Can bring handwritten notes sheet
- Read the book, review labs, do old tests

The Programming Question

- Small programming assignment graded with autolab, limited number of submissions
- Should take <30 minutes to complete
- Will have access to man pages and some starter code
- Practice examples:
  - Float←→int conversion
  - Binary search
  - Linked-list cycle detection
  - Detect system endian-ness

Brief Review of Some Topics

- System calls
- Virtual Address Translation
- Concurrency
- Processes
- Signals
- Caching
- Stack
- Network/file IO

System Calls

- Understand failure cases
  - errno
  - return value
  - Example: malloc returning NULL to memset

Some System Calls

- **fork**
  - Called once, returns twice (unless it fails)
    - Returns **0** in the child process
    - Returns the **pid** of the child in the parent process
    - Returns **-1** on failure
  - Makes an exact copy of the entire address space
  - Processes get unique copies of file descriptors, but share open files
  - Execution order of parent and child is arbitrary
- **execve**
  - Called once, doesn't return (unless it fails)
    - Returns **-1** on failure
  - Replaces the currently running process with the specified program

Some System Calls

- **`wait/waitpid`**
  - Reaps one child process
    - By default, blocks until a child process can be reaped
    - **`wait`** will wait for any child
    - **`waitpid`** waits for the specified child process
  - Returns the pid of the child that was reaped, or -1 on error
  - `waitpid` can be passed additional arguments to modify its behavior
    - WNOHANG will prevent waitpid from blocking
    - WUNTRACED will report stopped children
- **`signal`**
  - A simplified (but easier to understand) interface to `sigaction`
  - Installs a signal handler that is run when the specified signal is triggered

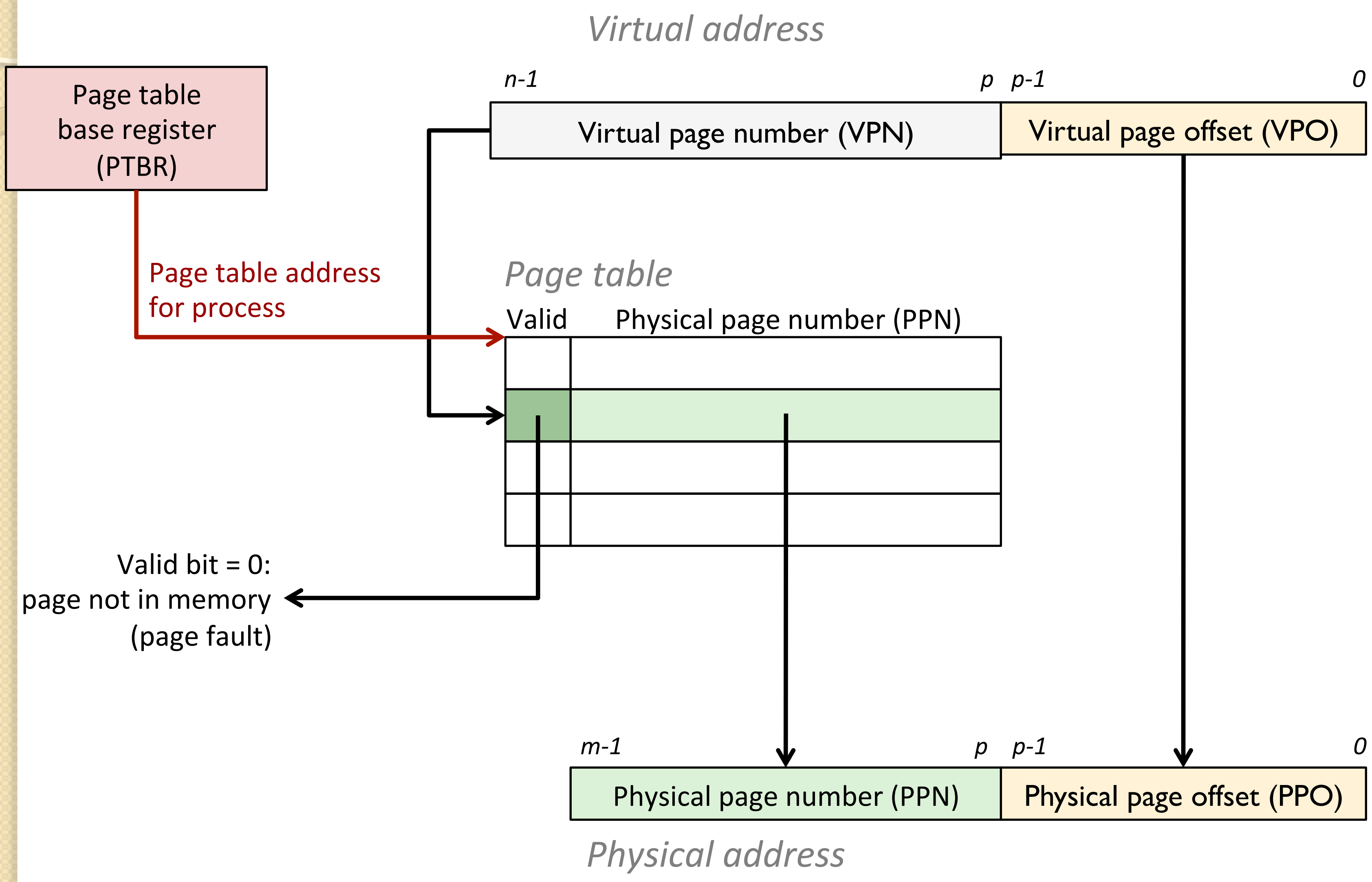
Some System Calls

- **`sigprocmask`**
  - Can block signals, unblock signals, or set the signal mask
    - SIG_BLOCK adds the given signals to the set of blocked signals
    - SIG_UNBLOCK removes the given signals
    - SIG_SETMASK replaces the blocked signals with the given signals
- **`sigsuspend`**
  - Replaces the signal mask with the specified mask
  - Blocks until one signal that isn't masked is handled
  - After the one signal is handled, the signal mask is restored

Virtual Address Translation

- Basic Parameters
  - **N = 2^n** : Number of addresses in virtual address space
  - **M = 2^m** : Number of addresses in physical address space
  - **P = 2^p** : Page size (bytes)
- Components of the virtual address (VA)
  - **VPO**: Virtual page offset
  - **VPN**: Virtual page number
  - **TLBI**: TLB index
  - **TLBT**: TLB tag
- Components of the physical address (PA)
  - **PPO**: Physical page offset (same as VPO)
  - **PPN:** Physical page number

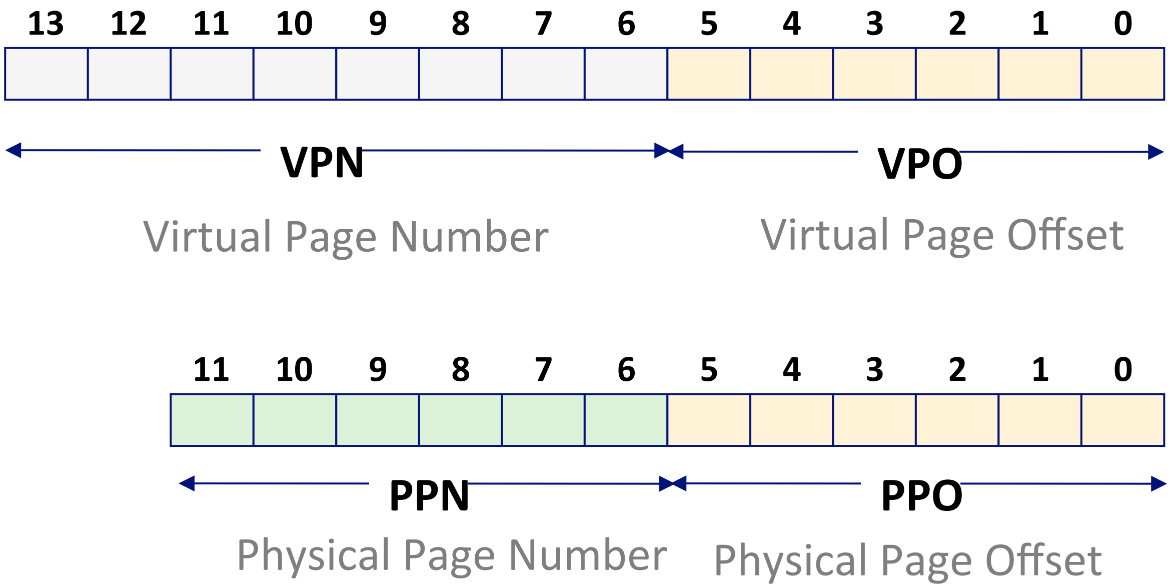
Virtual Address Translation


Virtual address
n-1
p
p-1
0
Page table base register (PTBR)
Virtual page number (VPN)
Virtual page offset (VPO)
Page table address for process
Page table
Valid
Physical page number (PPN)
Valid bit = 0: page not in memory (page fault)
m-1
p
p-1
0
Physical page number (PPN)
Physical page offset (PPO)
Physical address

Virtual Address Translation

- Simple Example

- Addressing
  - 14-bit virtual addresses
  - 12-bit physical address
  - Page size = 64 bytes



13 12 11 10 9 8 7 6 5 4 3 2 1 0

VPN

Virtual Page Number

VPO

Virtual Page Offset

11 10 9 8 7 6 5 4 3 2 1 0

PPN

Physical Page Number

PPO

Physical Page Offset

Concurrency Example

- Dining Philoso-bros (Philosophers) Problem
  - You and a friend share an order of sesame chicken and rice
  - But it comes with only one pair of chopsticks!
  - How do we share the chopsticks?

Eating functions

- `scoop(chopstick one, chopstick two)`
  - Scoop some rice. Requires two chopsticks.
- `stab(chopstick one)`
  - Stab some chicken with one chopstick.

Resources

- Two chopsticks
  - `chopstick1`, `chopstick2`
- Two chopstick mutexes (1 per chopstick)
  - `chopstick1_m`, `chopstick2_m`

The eating algorithm

```
scoop(chopstick1, chopstick2);

stab(chopstick2);

scoop(chopstick1, chopstick2);
```

The eating algorithm (with locking)

```
P(&chopstick1_m);
P(&chopstick2_m);
scoop(chopstick1, chopstick2);
V(&chopstick1_m);
stab(chopstick2);
P(&chopstick1_m);
scoop(chopstick1, chopstick2);
V(&chopstick1_m);
V(&chopstick2_m);
```

The eating algorithm (with locking)

```
P(&chopstick1_m);
P(&chopstick2_m);
scoop(chopstick1, chopstick2);
V(&chopstick1_m);
stab(chopstick2);
P(&chopstick1_m);
scoop(chopstick1, chopstick2);
V(&chopstick1_m);
V(&chopstick2_m);
```

The eating algorithm (with locking)

```
P(&chopstick1_m);
P(&chopstick2_m);
scoop(chopstick1, chopstick2);
stab(chopstick2);
scoop(chopstick1, chopstick2);
V(&chopstick1_m);
V(&chopstick2_m);
```

Questions?

- Thank you, and good luck on the final!