

# Operating Systems Concepts

Lazy Allocation

---

CS 4375, Fall 2025

Instructor: MD Armanuzzaman (*Arman*)

[marmanuzzaman@utep.edu](mailto:marmanuzzaman@utep.edu)

November 03, 2025

# Summary

- Process Synchronization
  - Multiprogramming: multiple processes or threads may require consensus to work together
    - How do we achieve that? What issues may occur for shared data?
  - Race Condition
  - Critical-Section Problem
    - Different Solutions

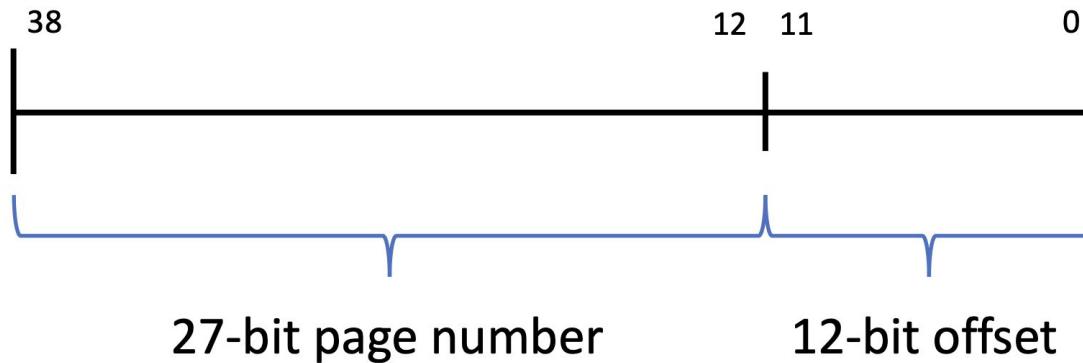
# Agenda

- xv6 memory mapping
- xv6 address translation
- Current eager allocation of xv6
  - Code overview
- Lazy allocation
  - Code overview

# Background

- Virtual addresses are divided into 4-KB “pages”

**Virtual Address:**



# Background

- Large PTE size

`GET_PTE(va) = &ptes[va >> 12]`

3

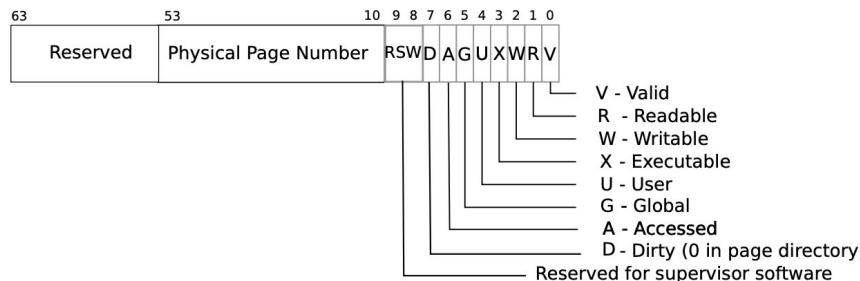
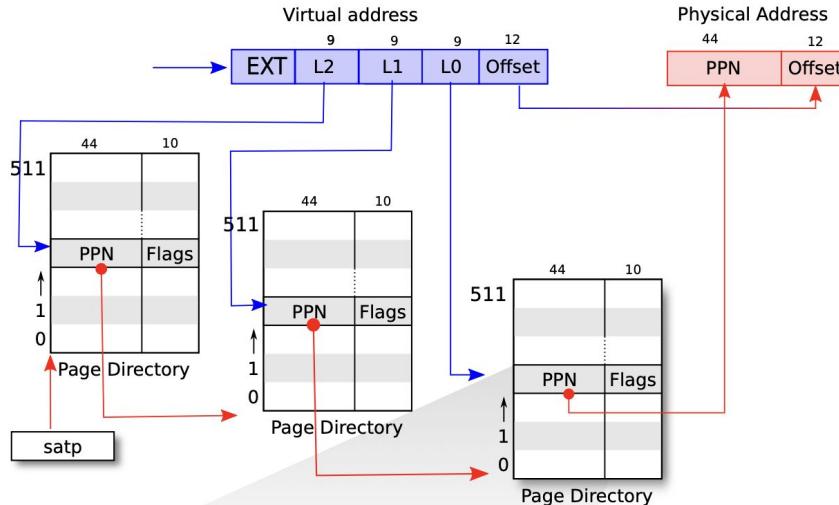
# How large is the array?

$2^{27} * 64$  bits

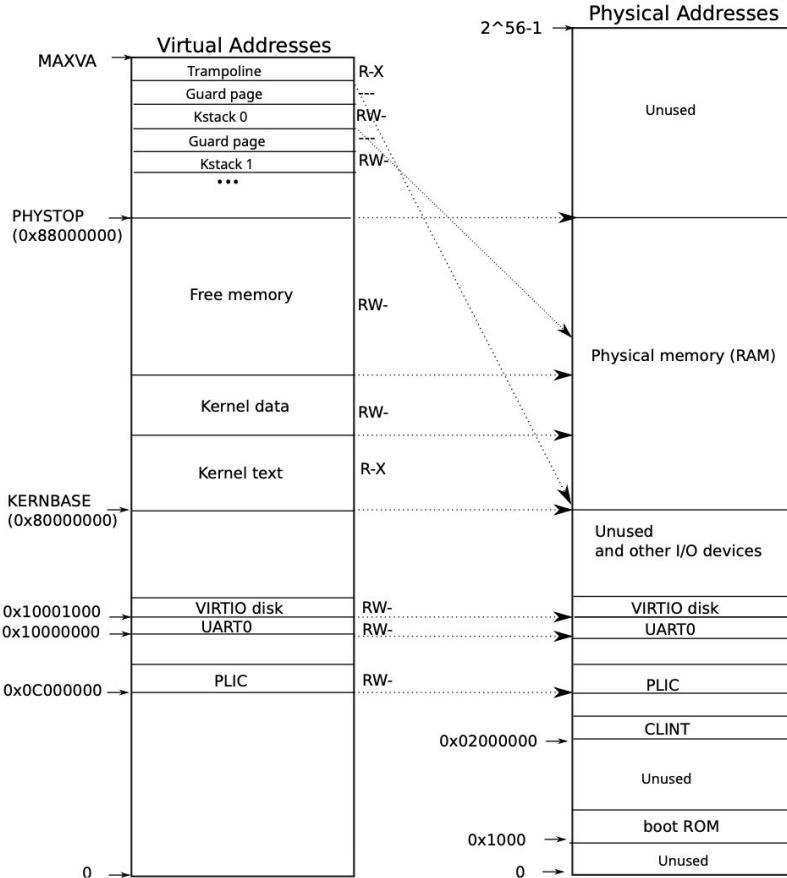
$2^{27} * 8 \text{ bytes}$

**~ 1 Gigabyte!**

# RISC-V address translation

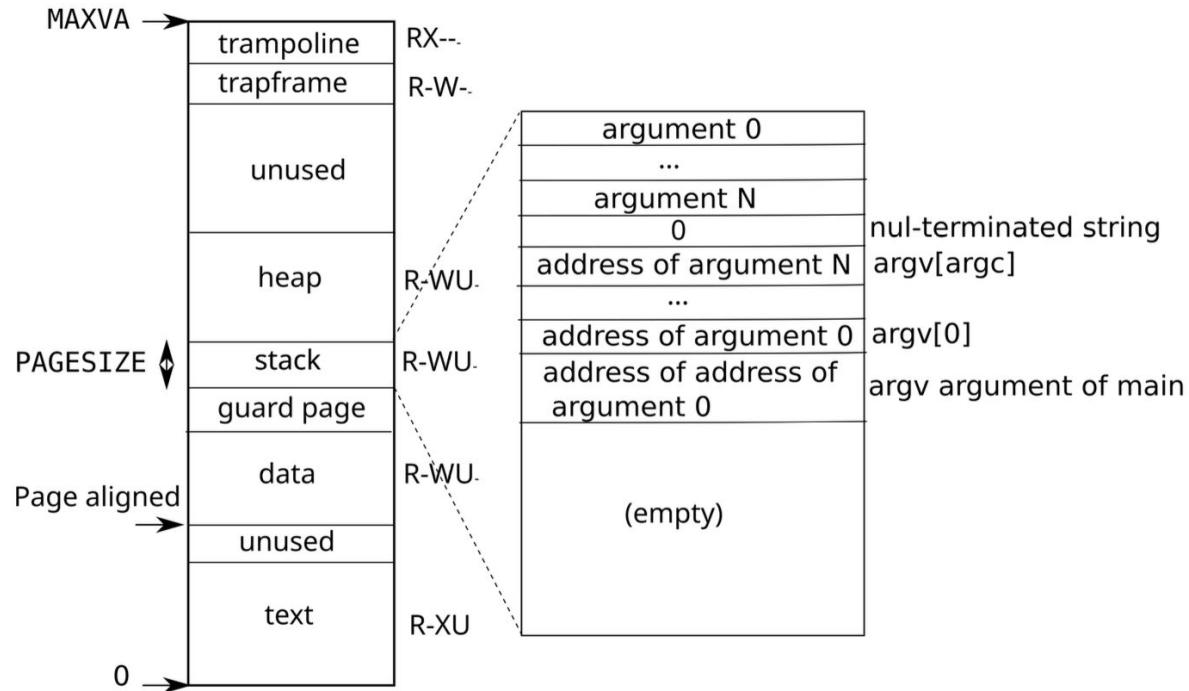


# Xv6 kernel virtual address space

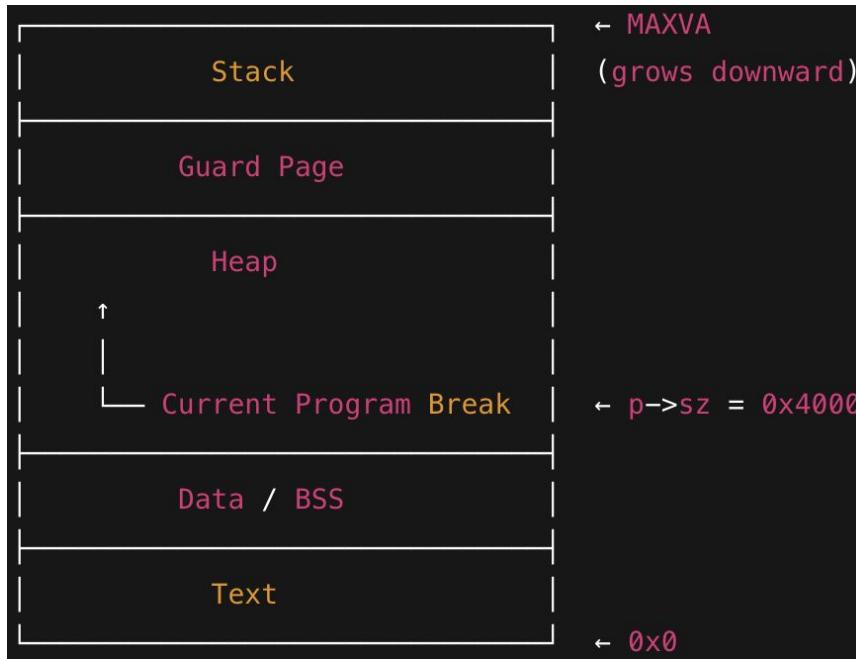


# Xv6 user process map

- **heap** : dynamically allocated memory
  - **malloc()**- to request additional memory; **free()**- to deallocate previously acquired memory



# sbrk system call



# sbrk system call- sbrk(4096)



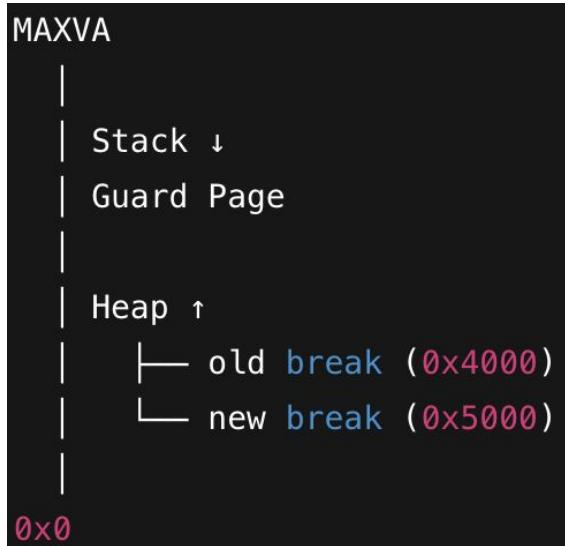
# Eager allocation

- When `sbrk()` increases heap size
  - The kernel immediately allocates physical pages and maps them.
- Pros
  - No page faults when accessing heap
  - Simple implementation
- Cons
  - Wastes memory if the process never touches most of the allocated space
  - Slower `sbrk()` calls (due to `kalloc() + mappages()`)

# Lazy allocation

- `sbrk()` only updates the process size (`p->sz`)
  - No physical memory is allocated yet
  - When the process first accesses a page → page fault occurs
  - The kernel's `usertrap()` allocates and maps the page on demand.
- Pros:
  - Saves physical memory
  - Enables overcommitment (process can reserve more memory than RAM)
- Cons:
  - Page faults add overhead
  - More complex to manage

# Lazy allocation



- Before:
  - Heap End → 0x4000
- After:
  - Heap End → 0x5000
  - New 4 KB region reserved (**no physical allocation yet**)

# RISC-V Exception Codes

- With lazy allocation, memory is reserved by not assigned
- handle in usertrap() - kernel/trap.c

Exception code	Description	Expected behavior
13	Load page fault	No valid mapping for load operation
15	Store/AMO page fault	No valid mapping for store or atomic op

# Test cases

- `memory-user.c`

Test Case	Description	Expected behavior
Case 1	Allocate & free (no touch)	No page faults; free memory unchanged
Case 2	Allocate & touch all page	Page faults per page; free memory decreases
Case 3	Allocate & touch some pages	Partial allocation; free memory drops slightly

# Announcement

- Homework 4
  - Due on November 10th at 11.59PM
- Next Class
  - Quiz 5
  - Topics: lecture 16 -19