Operating Systems Concepts

Lazy Allocation

CS 4375, Fall 2025

Instructor: MD Armanuzzaman (***Arman***)
marmanuzzaman@utep.edu

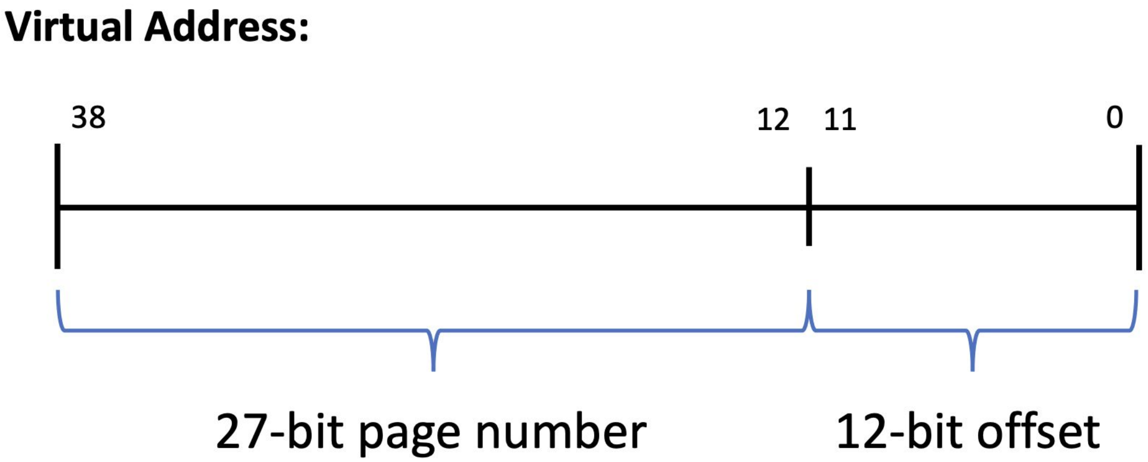
November 03, 2025

Agenda

- xv6 memory mapping
- xv6 address translation
- Current eager allocation of xv6
  - Code walk through
- Lazy allocation
  - Concept overview
  - Important code snippets

Background

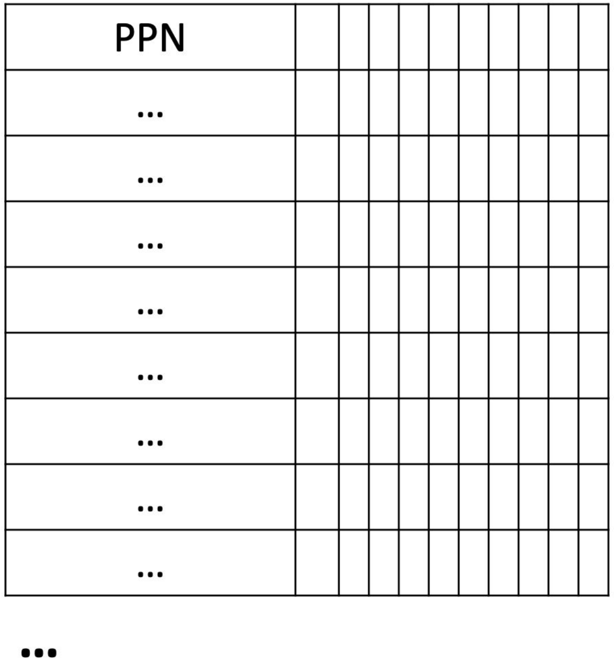
- Virtual addresses are divided into 4-KB “pages”



Background

- Large PTE size

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



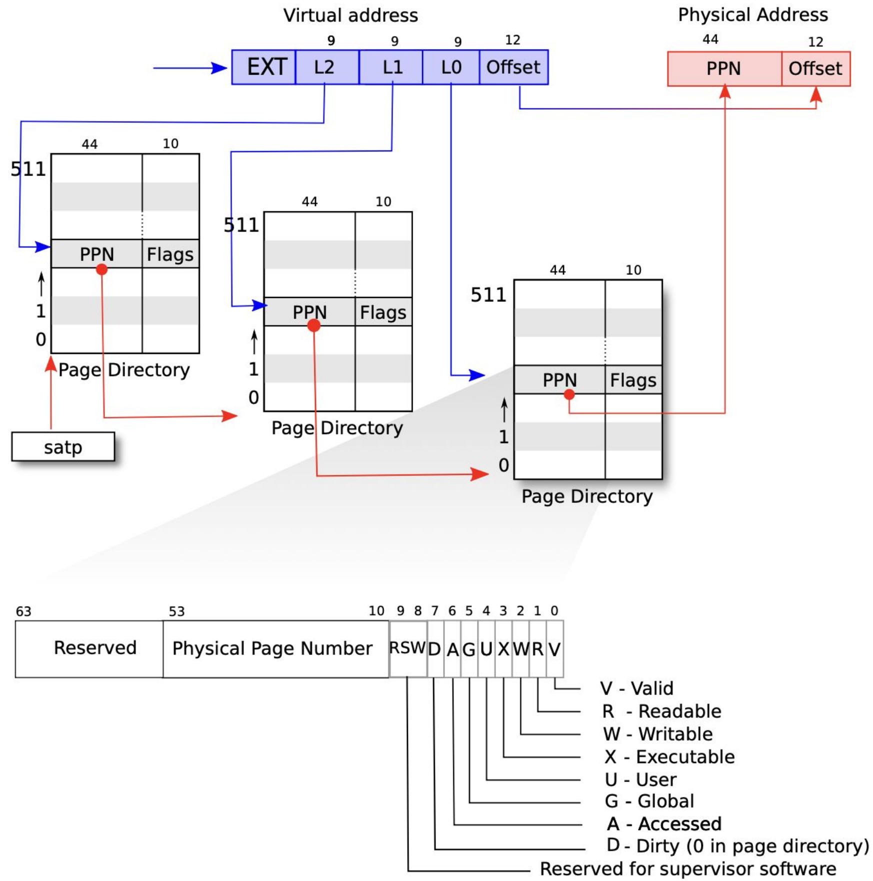
How large is the array?

2^27 * 64 bits

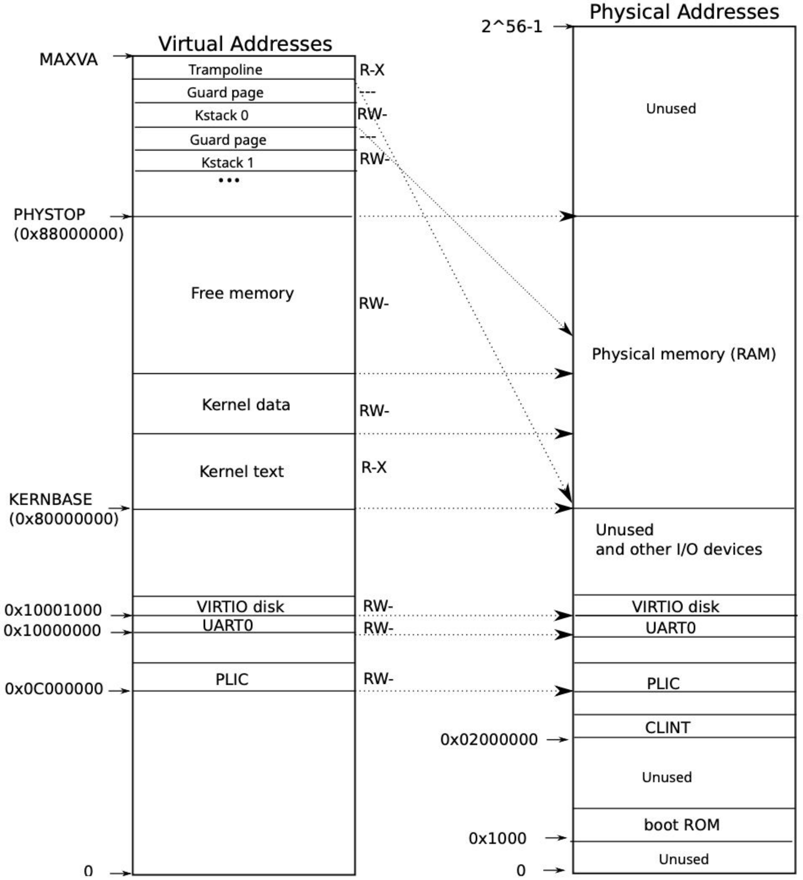
2^27 * 8 bytes

~ 1 Gigabyte!

RISC-V address translation

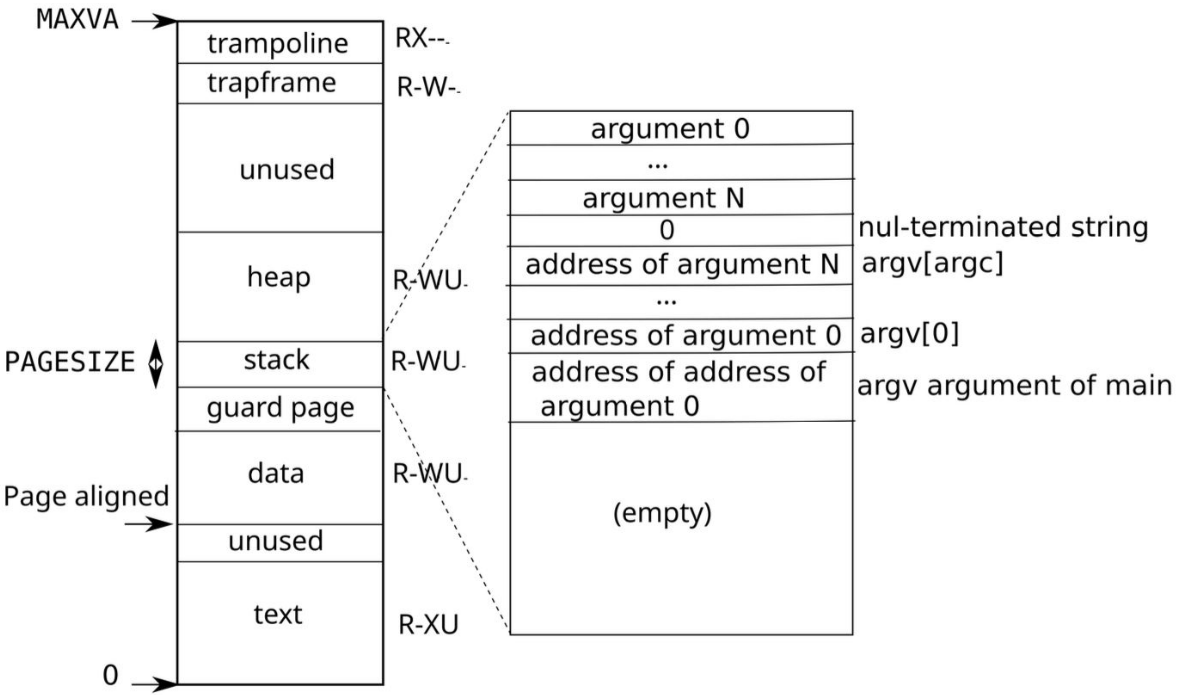


Xv6 kernel virtual address space

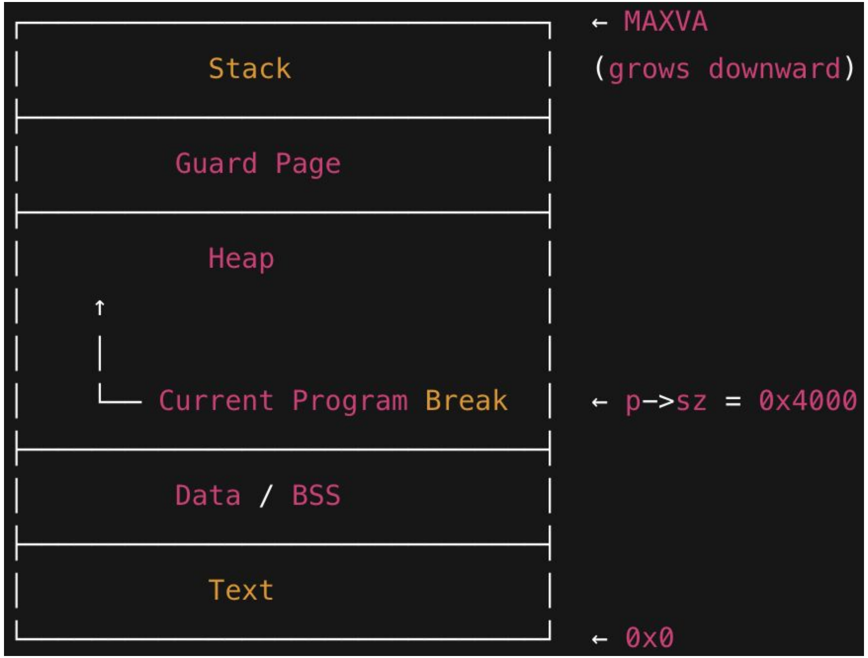


Xv6 user process map

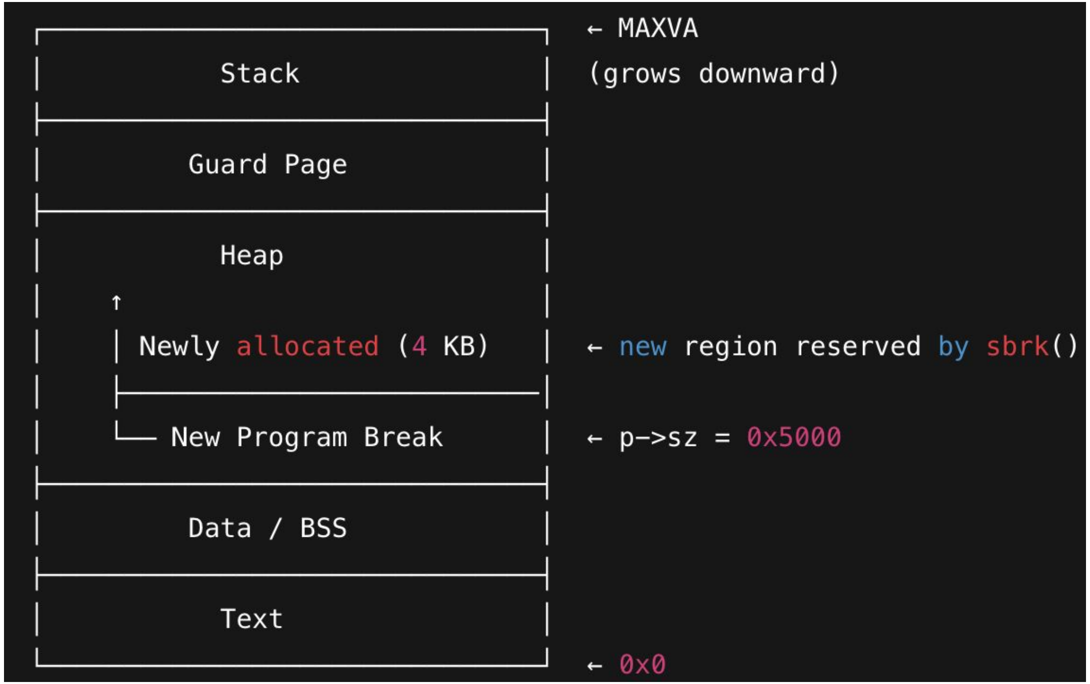
- **heap** : dynamically allocated memory
  - **malloc**()- to request additional memory; **free**()- to deallocate previously acquired memory



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

```
MAXVA
  │
  │ Stack ↓
  │ Guard Page
  │
  │ Heap ↑
  │   ├── old break (0x4000)
  │   └── new break (0x5000)
  │
0x0
```

- 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