National Yang Ming Chiao Tung University
Computer Architecture & System Lab

RISC-V Paging

IOC5226 Operating System Capstone

Tsung Tai Yeh
Department of Computer Science
National Yang Ming Chiao Tung University

Acknowledgements and Disclaimer

- Slides were developed in the reference with
  - MIT 6.828 Operating system engineering class, 2018
  - MIT 6.004 Operating system, 2018
  - Remzi H. Arpaci-Dusseau etl. , Operating systems: Three easy pieces. WISC

Outline

- RISC-V Paging
  - Sv39 RISC-V
  - Multi-level page table
  - Kernel address space
  - Process address apce
    - `sbrk`
    - `exec`

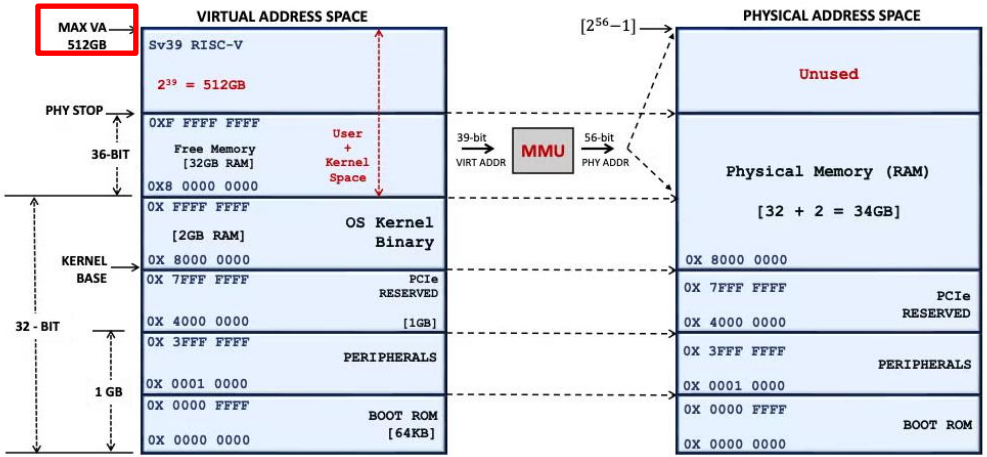
Paging Hardware

- Sv39 RISC-V
  - Only the bottom 39 bits of a 64-bit virtual address are used, 2^{39} = 512 GB, each page is 4KiB
  - A RISC-V page table is logically an array of 2^{27} page table entries
  - Each PTE contains a 44-bit physical page number (PPN) and some flags

Scheme	Virtual address width	Physical address width	Page table levels	Page sizes	Max VA space	Max PA space
Sv32	32 bits	34 – 40 bits	2 levels	4KiB, 2MiB	4GiB	Up to 1 TiB
Sv39	39 bits	up to 56 bits	3 levels	4KiB, 2MiB, 1GiB	512 GiB	Up to 64 PiB
Sv48	48 bits	up to 56 bits	4 levels	4KiB, 2MiB, 1GiB	256 TiB	Up to 64 PiB
Sv57	57 bits	up to 56 bits	5 levels	4KiB, 2MiB, 1GiB	128 PiB	Up to 64 PiB

Paging Hardware

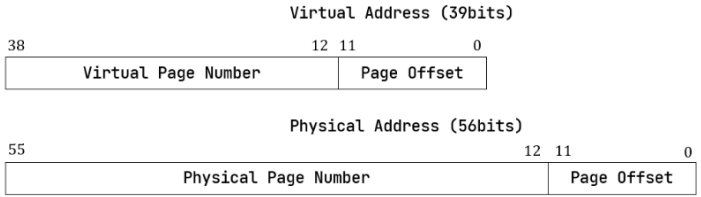
- Sv39 RISC-V
  - A 39-bit VA with 512 GB of VA address space
  - A 56-bit PA with 64 PB of PA address space
  - The firmware, bootloader codes are 32-bit, and executed without MMUs

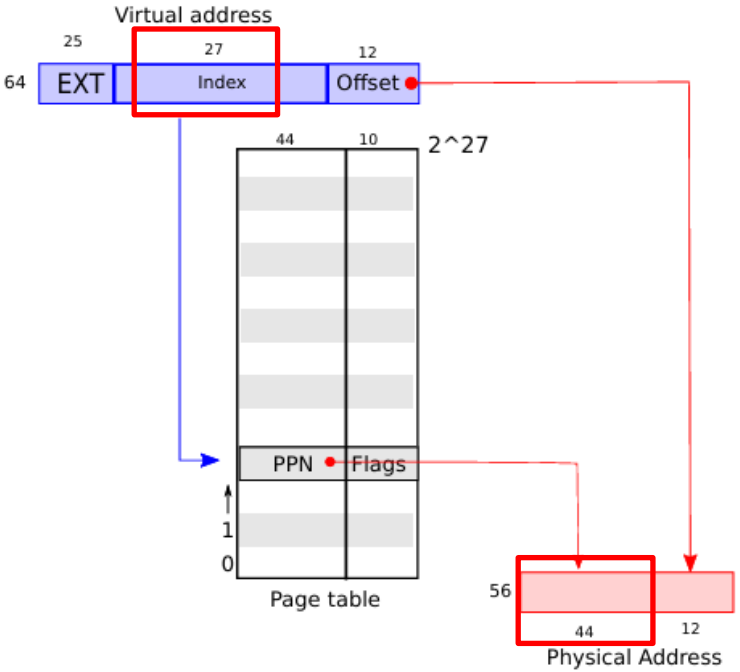


https://riscv.org/blog/design-approaches-and-architectures-of-risc-v-socs/

Paging Hardware

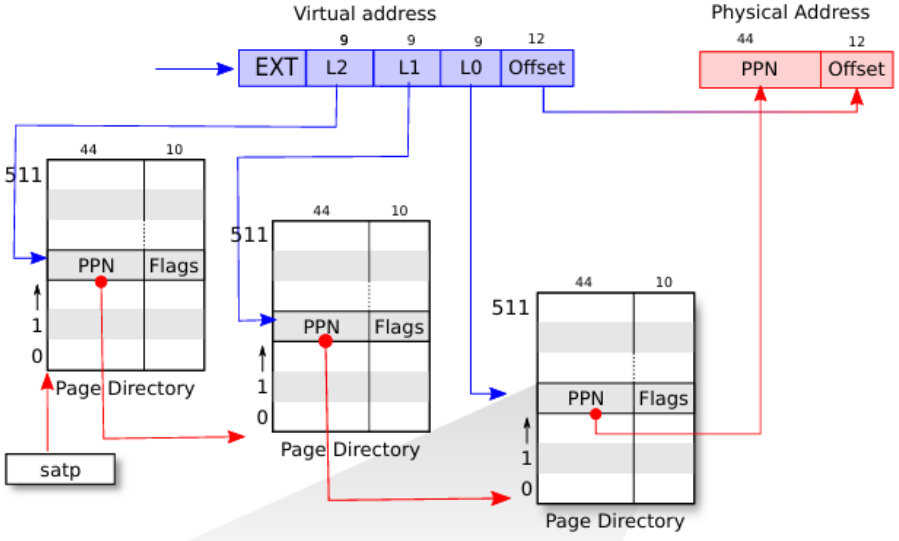
- Sv39 RISC-V
  - The paging hardware translates top 27 bits VAs to index into the page table to find a PTE
  - 56-bit physical address whose top 44 bits come from the PPN in the PTE and bottom 12 bits are from original VA





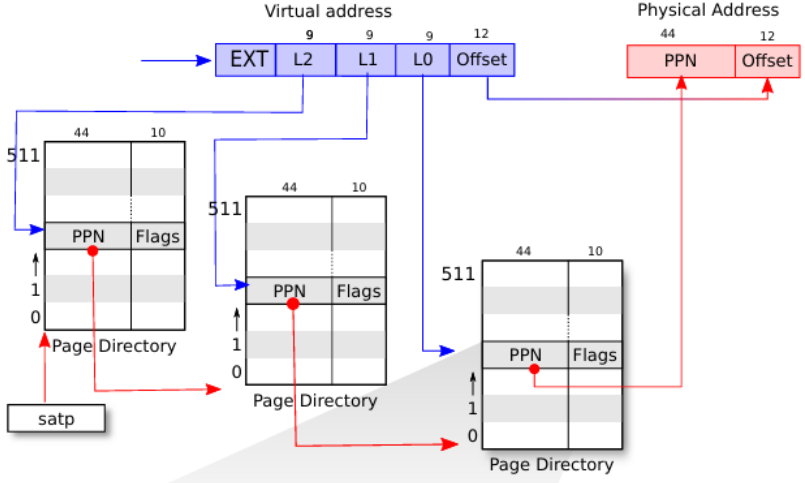
Paging Hardware

- The three-level page table structure
  - Allows a memory-efficient way of recording PTEs
  - Save 511 pages for intermediate page directories
  - 511 x 512 pages for bottom-level page directories



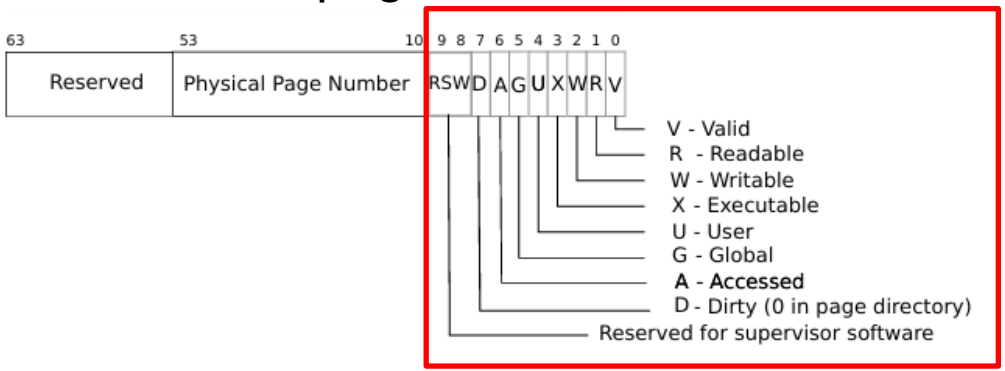
Paging Hardware

- Translation Look-aside Buffer (TLB)
  - CPU must load **three** PTEs from memory to perform the translation of the virtual address to a physical address in sv39
  - To avoid the cost of loading PTEs from physical memory
    - RISC-V CPU caches PTEs in a TLB



Paging Hardware

- Flag bits
  - Each PTE contains flag bits to tell the paging hardware how the associated virtual address is allowed to be used
    - **PTE_V** indicates whether the PTE is present
    - **PTE_U** controls whether instructions in user mode are allowed to access the page



Paging Hardware

- The `satp` register
  - To tell a CPU to use a page table, the OS kernel must write the PA of the root page table page into the `satp` register
  - Each CPU has its own `satp` so that different CPUs can run different processes
    - **MODE**: define the translation mechanism
    - **ASID**: enables tag entries in TLB with a specific address space, reducing the need to flush TLB on context switches
    - **PPN**: holds physical page number of the root page table

Paging Hardware

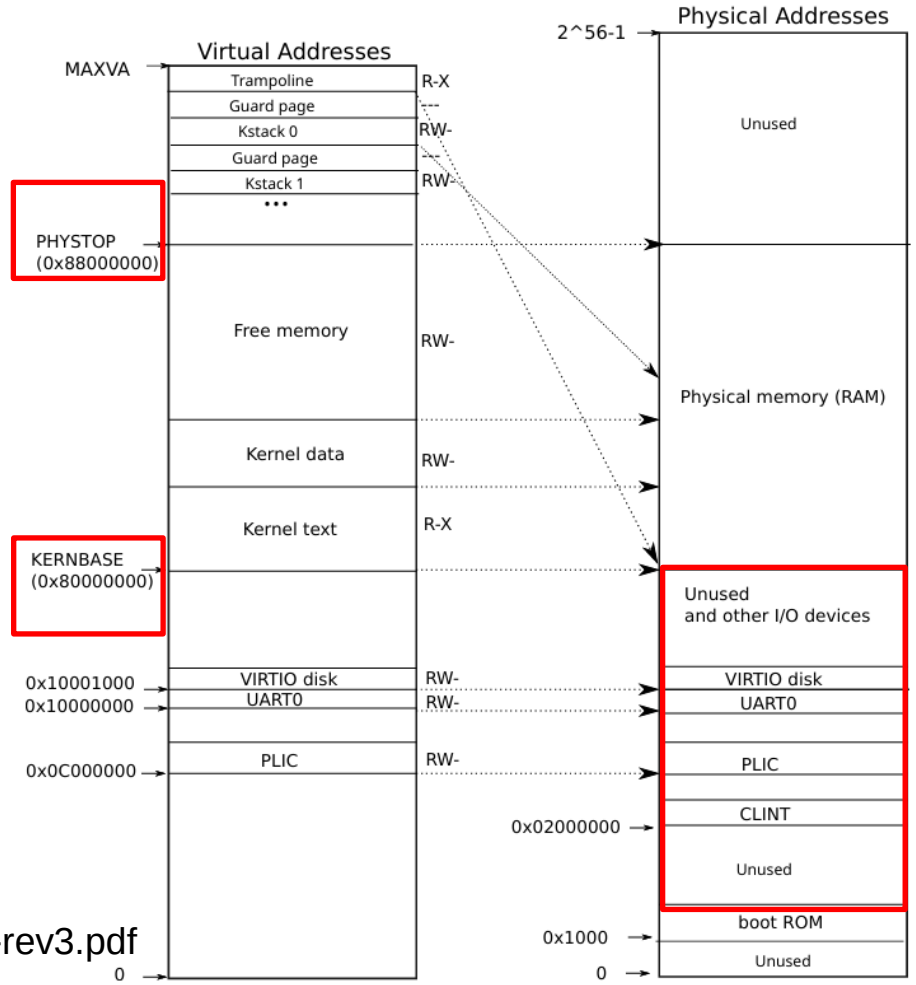
- The `satp` register
  - The MODE fields define the translation mechanism
    - Bare (0): MMU disabled. VA directly maps to PA
    - Sv32 (1)
    - Sv39 (8)
    - Sv48 (9)
    - Sv57 (10)

Kernel address space

- One page table per process
  - Describe each process's user address space, plus
  - A single page table that describe the kernel's address space
- The kernel configures the memory layout
  - Gives itself access to physical memory
  - Provides various hardware resources at predictable virtual addresses

Kernel address space

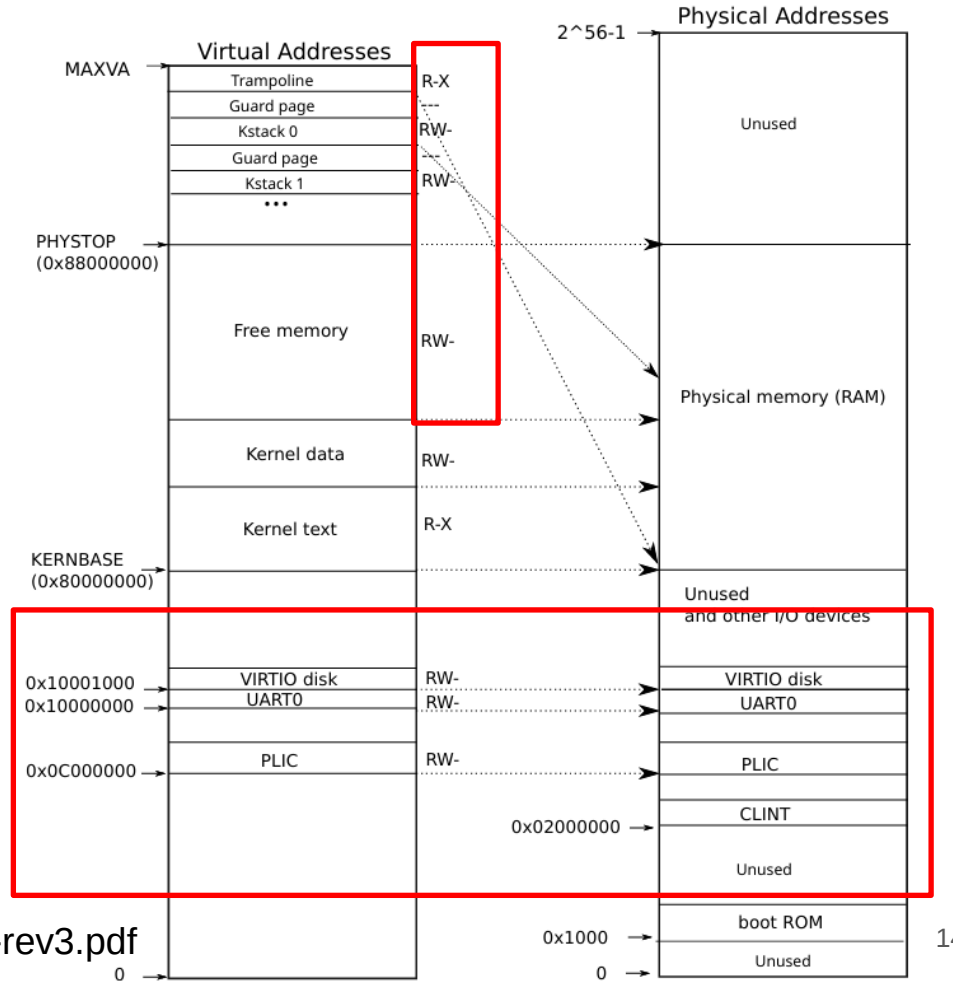
- QEMU memory layout
  - RAM starts at physical address 0x80000000 until 0x88000000
  - Exposes device interfaces to software as
    - Memory-mapped control registers
    - Below 0x80000000



https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf

Kernel address space

- QEMU memory layout
  - The kernel interacts with devices through RW these memory-mapped control registers (direct mapping VA == PA)
  - RWX refer to PTE read write, and execute permission



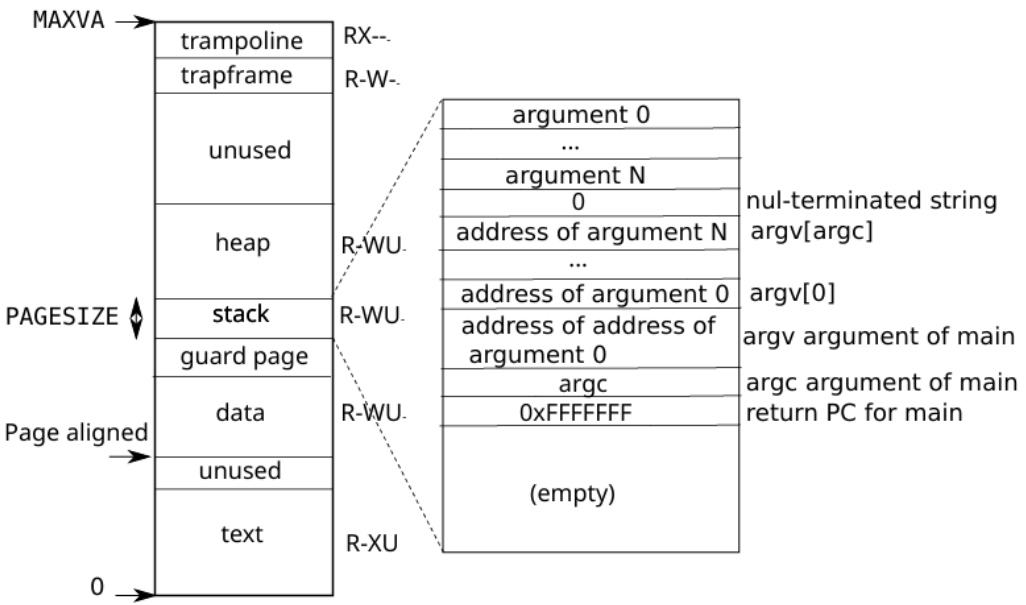
https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf

RISC-V TLB

- How to clean PTE stored on TLB?
  - RISC-V CPU uses the `sfence.vma` command to ensure the data clean on the TLB to be complete
    - Ensure the operations of the old page table complete and start the use of new page table
    - Also use in the change of the `satp` register
  -

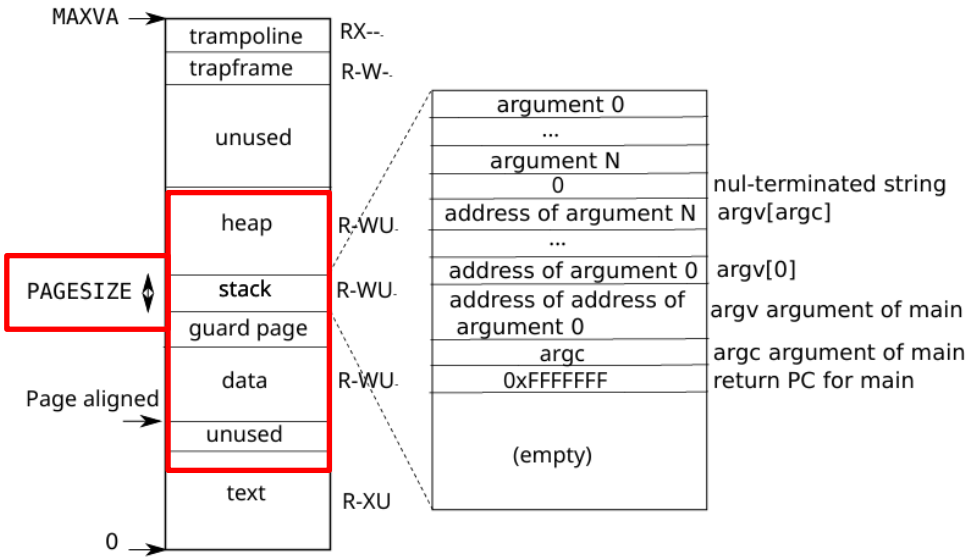
Process Address Space

- Each process has a separate page table
  - The OS kernel changes the page table when switching between processes
  - A process's user memory starts at VA zero and can grow up to MAXVA



Process Address Space

- A process's address space consists of pages
  - The text, pre-initialized data, stack, heap of the program
  - The stack is a single page, and is shown with the initial contents as created by exec
  - When user stack overflows
    - Generate a page-fault exception
    - Guard page is inaccessible in user mode



Code: sbrk

- The system call: `sbrk`
  - Shrink or grow process's memory
  - Implemented by `uvmalloc` or `uvmdealloc`
  - `uvmalloc`
    - allocates physical memory with `kalloc`
    - Adds PTEs to the user page table
  - `uvmdealloc`
    - Finds PTEs and uses `kfree` to free the physical memory

Code: exec

- The system call: `exec`
  - Replace a process's user address space with data read from a file, called a binary or executable file
  - A binary is the output of the compiler and linker and contains the machine instructions and program data
  - The first step: read ELF file
    - ELF header
    - Program section header
      - Describes a section of the application that must be loaded into memory

Code: exec

- The system call: `exec`
  - The program section header for /init, the first user program created with exec
  - The text segment should be loaded at VA 0x0
  - The data should be loaded at address 0x1000. a page boundary

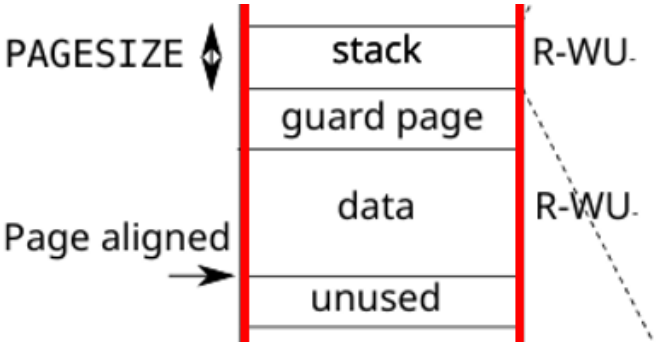
```
# objdump -p user/_init

user/_init:     file format elf64-little

Program Header:
0x70000003 off    0x0000000000006bb0 vaddr 0x0000000000000000
                                     paddr 0x0000000000000000 align 2**0
         filesz 0x000000000000004a memsz 0x0000000000000000 flags r--
    LOAD off    0x0000000000001000 vaddr 0x0000000000000000
                                     paddr 0x0000000000000000 align 2**12
         filesz 0x0000000000001000 memsz 0x0000000000001000 flags r-x
    LOAD off    0x0000000000002000 vaddr 0x0000000000001000
                                     paddr 0x0000000000001000 align 2**12
         filesz 0x0000000000000010 memsz 0x0000000000000030 flags rw-
   STACK off    0x0000000000000000 vaddr 0x0000000000000000
                                     paddr 0x0000000000000000 align 2**4
         filesz 0x0000000000000000 memsz 0x0000000000000000 flags rw-
```

Code: exec

- The system call: `exec`
  - Allocates and initializes the user stack
  - Allocates just one stack page
  - Copies the argument strings to the top of the stack one at a time
  - Places an inaccessible page just below the stack page



Takeaway Questions

- What are correct descriptions about RISC-V Sv39?
  - (A) The Max VA size is 4 GB
  - (B) 2^{27} PTEs
  - (C) Consists of three-level page table
- When can we use the `satp` register?
  - (A) Write the physical address of the root page table page
  - (B) Create a page table
  - (C) Manage the kernel process address

Takeaway Questions

- How to access the device interfaces?
  - (A) page table
  - (B) memory-mapped register
  - (C) `satp` register
- What items are shown in a process address space?
  - (A) Stack
  - (B) Heap
  - (C) Guard page