1.

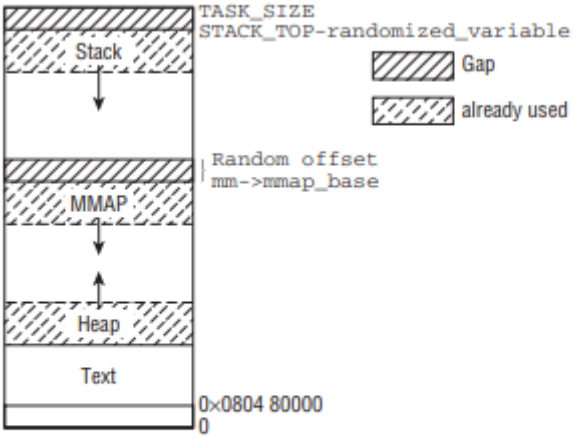

Figure 4-2: Layout of the virtual address space on IA-32 machines when the mmap region is expanded from top to bottom.

2 ,3:

Mainc creates a page table for the kernel's use with a call to kvmalloc, and mpmain causes the x86 paging hardware to start using that page table with a call to vmenable. This page table maps most virtual addresses to the same physical address, so turning on paging with it in place does not disturb execution of the kernel. kvmalloc calls setupkvm and stores a pointer to the resulting page table in kpgdir, since it will be used later. An x86 page table is stored in physical memory, in the form of a 4096-byte "page directory" that contains 1024 PTE-like references to "page table pages." Each page table page is an array of 1024 32-bit PTEs. The paging hardware uses the top 10 bits of a virtual address to select a page directory entry. If the page directory entry is marked PTE_P, the paging hardware uses the next 10 bits of the virtual address to select a PTE from the page table page that the page directory entry refers to. If either of the page directory entry or the PTE has no PTE_P, the paging hardware raises a fault. This two-level structure allows a page table to omit entire page table pages in the common case in which large ranges of virtual addresses have no mappings. setupkvm allocates a page of memory to hold the page directory. It then calls mappages to install translations for ranges of memory that the kernel will use; these translations all map each virtual address to the same physical address. The translations include the kernel's instructions and data, physical memory up to PHYSTOP, and memory ranges which are actually I/O devices. setupkvm does not install any mappings for the process's memory; this will happen later. mappages installs mappings into a page table for a range of virtual addresses to a corresponding range of physical addresses. It does this separately for each virtual address in the range, at page intervals. For each virtual address to be mapped, mappages calls walkpgdir to find the address of the PTE that should the address's translation. It then initializes the PTE to hold the relevant physical page number, the desired permissions (PTE_W and/or PTE_U), and PTE_P to mark the PTE as valid. 4 walkpgdir mimics the actions of the x86 paging hardware as it looks up the PTE for a virtual address. It uses the upper 10 bits of the virtual address to find the page directory entry. If the page directory entry isn't valid, then the required page table page hasn't yet been created; if the create flag is set, walkpgdir goes ahead and creates it. Finally it uses the next 10 bits of the virtual address to find the address of the PTE in the page table page. The code uses the physical addresses in the page directory entries as virtual addresses. This works because the kernel allocates page directory pages and page table pages from an area of physical memory (between the end of the kernel and PHYSTOP) for which the kernel has direct virtual to physical mappings. vmenable loads kpgdir into the x86 %cr3 register, which is where the hardware looks for the physical address of the current page directory. It then sets CR0_PG in %cr0 to enable paging.

4.
The xv6 kernel calls kalloc and kfree to allocate and free physical memory at run-time.

5.
setupkvm allocates a page of memory to hold the page directory. It then calls mappages to install translations for ranges of memory that the kernel will use; these translations all map each virtual address to the same physical address. The translations include the kernel's instructions and data, physical memory up to PHYSTOP, and memory ranges which are actually I/O devices. setupkvm does not install any mappings for the process's memory; this will happen later. mappages installs mappings into a page table for a range of virtual addresses to a corresponding range of physical addresses. It does this separately for each virtual address in the range, at page intervals.

6.
For each virtual address to be mapped, mappages calls walkpgdir to find the address of the PTE that should the address's translation. It then initializes the PTE to hold the relevant physical page number, the desired permissions (PTE_W and/or PTE_U), and PTE_P to mark the PTE as valid.
PTE for a virtual address. It uses the upper 10 bits of the virtual address to find the page directory entry. If the page directory entry isn't valid, then the required page table page hasn't yet been created; if the create flag is set, walkpgdir goes ahead and creates it.
Finally it uses the next 10 bits of the virtual address to find the address of the PTE in the page table page. The code uses the physical addresses in the page directory entries as virtual addresses. This works because the kernel allocates page directory pages and page table pages from an area of physical memory (between the end of the kernel and PHYSTOP) for which the kernel has direct virtual to physical mappings. vmenable loads kpgdir into the x86 %cr3 register, which is where the hardware looks for the physical address of the current page directory. It then sets CR0_PG in %cr0 to enable paging.