CS5460/6460: Operating Systems

Lecture 3: The First Process

Anton Burtsev
January, 2014

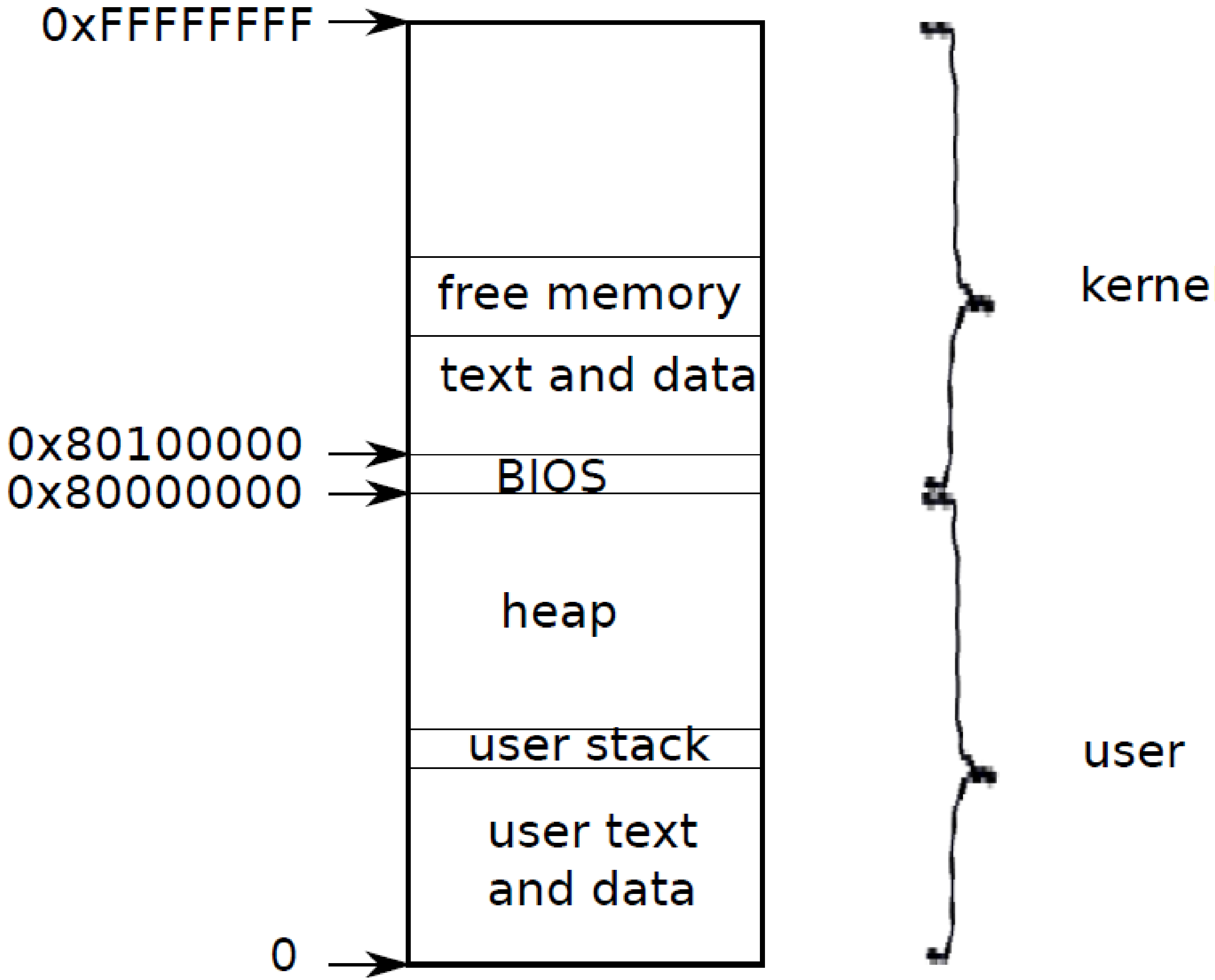
Process

- Illusion of private machine
  - CPU
  - Memory
- CPU is easy
  - Set of registers EAX, EBX, …

Memory

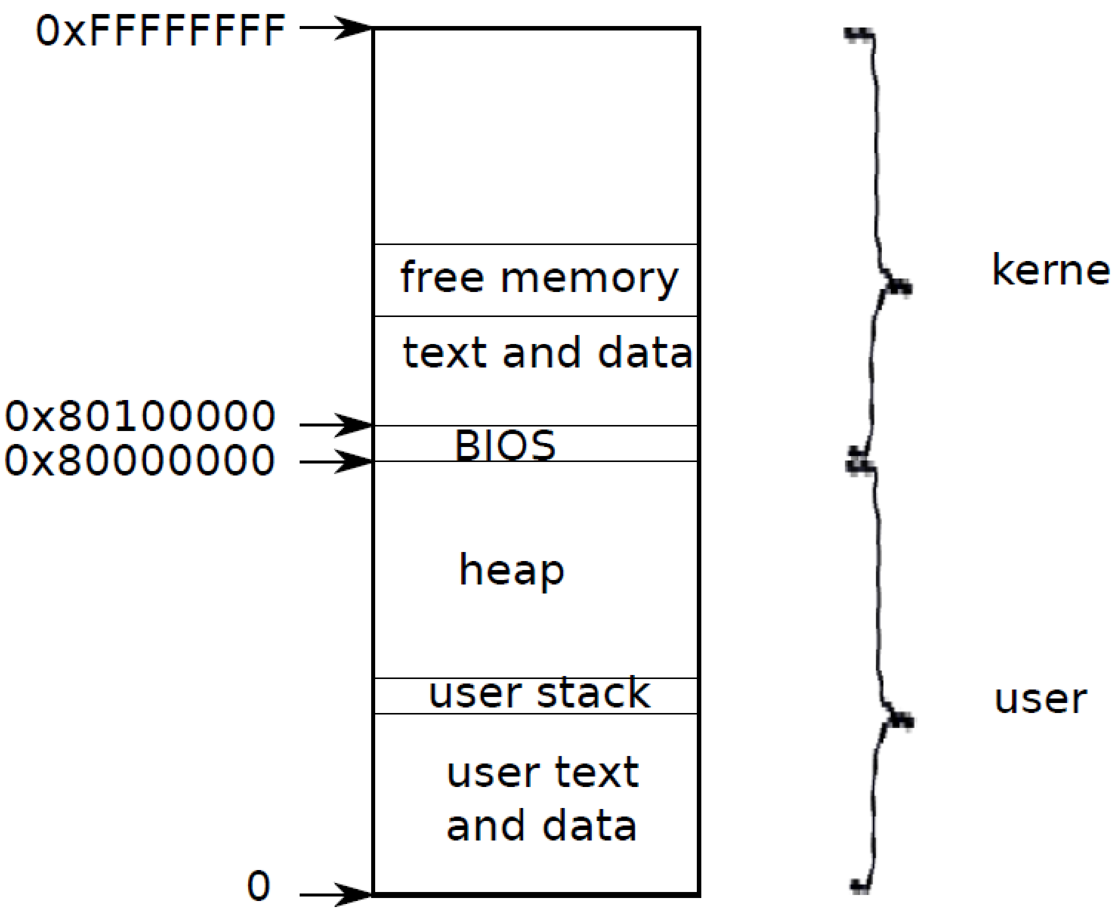
- Private address space
  - Other processes can't read or write
- Implemented with virtual memory
- Each process has a page table
  - They are switched on context switch
  - Page tables “map” which physical pages implement virtual addresses

Address space layout



Address space layout

- Maps both user and kernel memory
- Kernel can easily read/write process memory from a system call
- 2GB for user
- 2GB for kernel



Address space

- Maps both user and kernel memory
    - This way kernel can easily read/write process memory from a system call

Processes

- Kernel maintains information about each process
  - Page table
  - Kernel stack
  - Run state
- Each process has two stacks
  - User
  - Kernel

PC Boot

BIOS

- Power on → BIOS
  - Stored in a non-volatile memory on the motherboard
  - Prepare hardware
- BIOS → boot loader
  - Stored in the first 512 byte disk sector
  - BIOS loads first sector in memory at 0x7c00
  - Jumps to this address (sets EIP to this address)

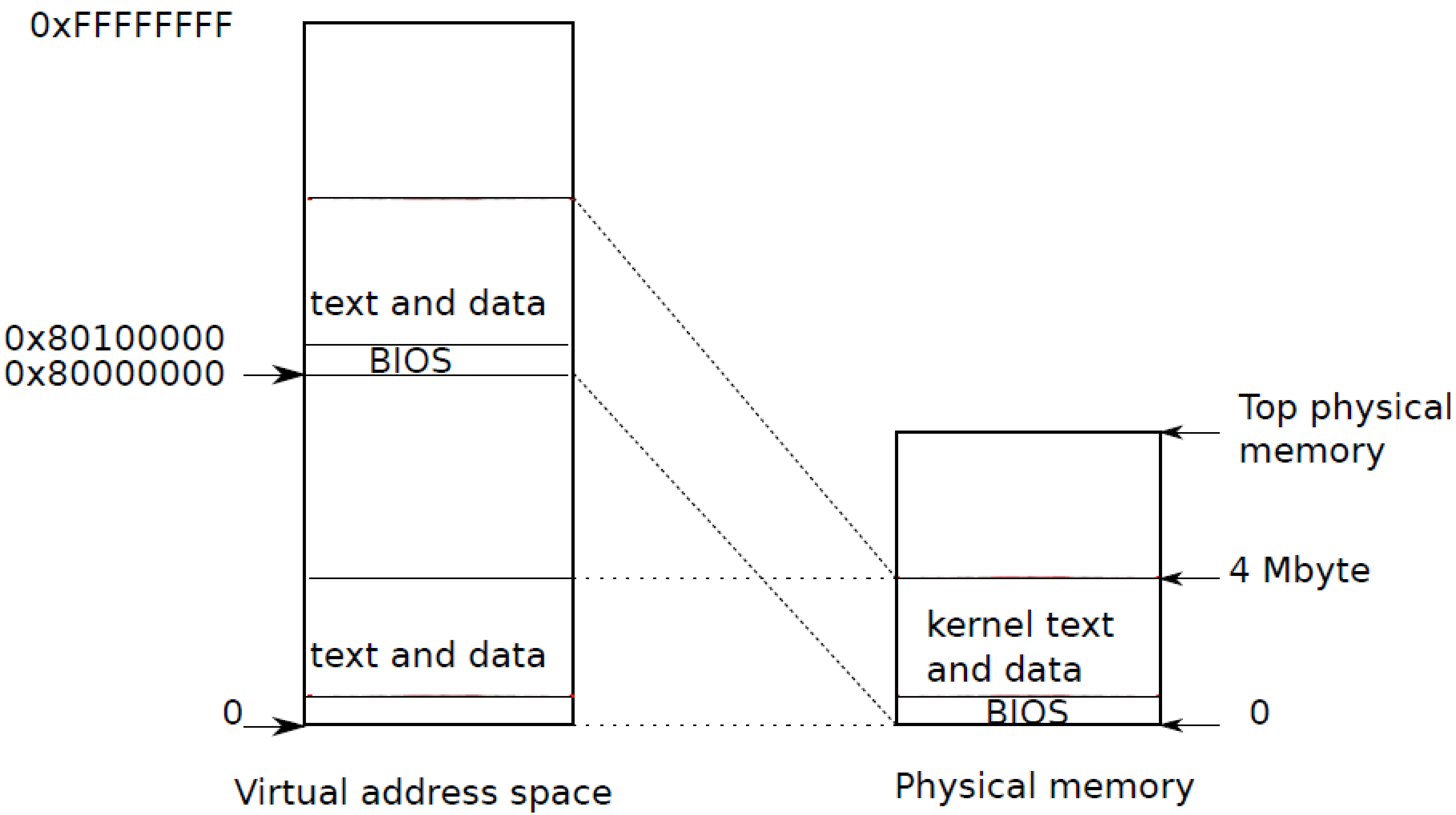
Boot loader

- Part 1: Hand-written ASM
  - Switches CPU from real to protected mode
  - We'll discuss details a bit later
  - Jump to a C function (`bootmain`)
- Part 2: C
  - Expects to find kernel in the second disk sector
  - Kernel is an ELF binary
  - Kernel is copied to physical location 0x100000 (1MB)

Kernel

- Boot loader → kernel (0x1000c)
- Page tables are not enabled
- Kernel must map itself to the high location 0x80100000

Kernel maps itself twice



First process: `userinit()`

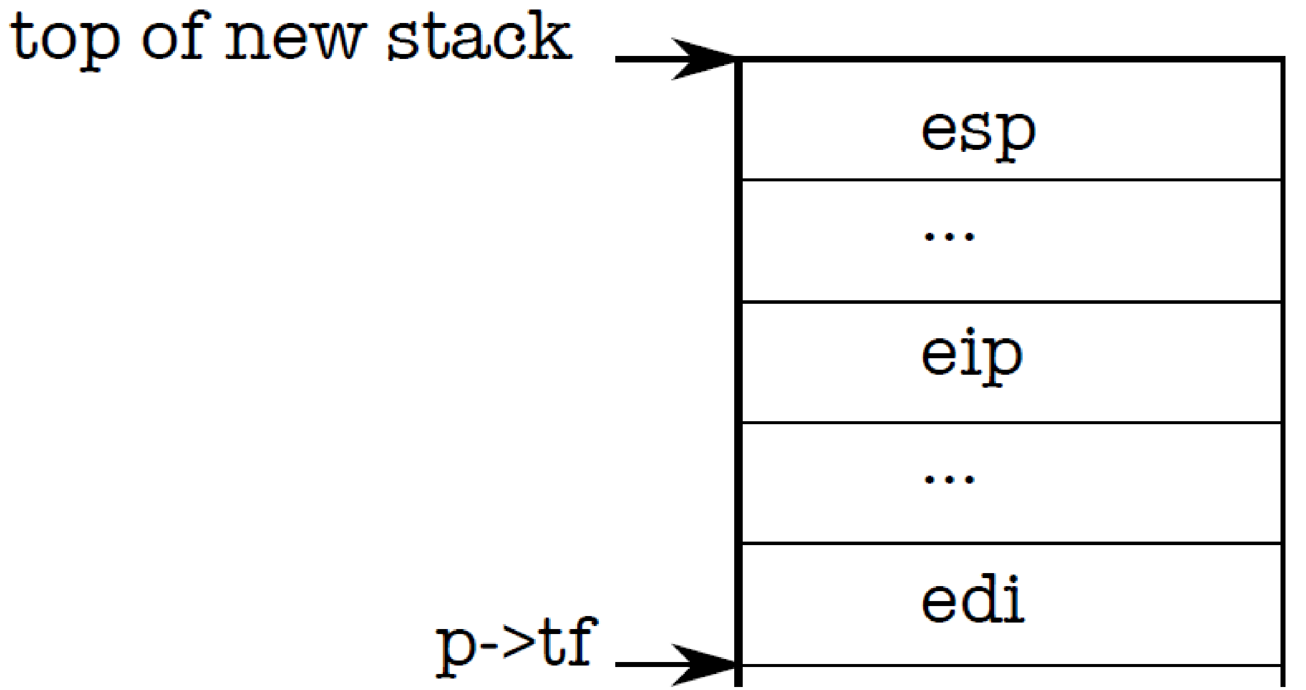
- Allocate the proc data structure
- Allocate process kernel stack
- High level plan
  1) Pretend inside `fork()`
  2) Return from
  3) Return from kernel to user level

How do processes get into kernel?

- fork() is implemented as interrupt
  - Saves user registers on top of the kernel stack

How do processes get into kernel?

- fork() is implemented as interrupt
  - Saves user registers on top of the kernel stack



Normally kernel returns with trapret