CS143A
Principles on Operating Systems
Discussion 07:

Instructor: Prof. Anton Burtsev

TA: Saehanseul Yi (Hans)

Nov 15, 2019 3 PM
Agenda

• Requirements: xv6 installed (i.e. able to execute qemu-system-x86_64)
• HW3 Overview
Booting...

- **BIOS** → Bootloader (real mode) → kernel (virtual address)
- **BIOS**: check connected devices (Storage, USB, ...)
- **Bootloader**: load OS to the memory! security check, etc...
- **kernel**: when it starts, setting page table, etc.
Booting...

- **BIOS** → Bootloader (real mode) → kernel (virtual address)
- **BIOS**: check connected devices (Storage, USB, ...)
- **Bootloader**: load OS to the memory! security check, etc...
- **kernel**: when it starts, setting page table, etc.

![BIOS Setup Utility](image)
Booting...

- BIOS → Bootloader (real mode) → kernel (virtual address)
- **BIOS**: check connected devices (Storage, USB, ...)
- **Bootloader**: load OS to the memory! security check, etc...
- **kernel**: when it starts, setting page table, etc. `main.c`, `boot.asm`, ...
  we will create new page tables and replace the ones created in `boot.asm`
Page Table Recap

• Two-level structure in x86
  • four or more levels in x64—larger memory space to cover
  • “Page Directory”-”Page Table”-”Physical Address”

• CR3 contains the base address of the page directory
Downloading the source files

• `wget --recursive --no-parent <URL>`
  or equivalently, `wget -r -np <URL>`
Makefile

• Don’t read from the start—Follow the targets (recipes) of interest

• $(patsubst %.asm, build/%.o, $(assembly_source_files))
  this converts (compiles) .asm to .o

• $(patsubst %.c, build/%.o, $(c_source_files))
  Likewise, this converts .c to .o
Boot.asm for VGA Hello World..

• Follow the instructions in HW3 description..

If you don’t want to bother with all that, here’s the final code:

```assembly
global start

section .text
bits 32
start:
mov word [0xb8000], 0x0248 ; H
mov word [0xb8002], 0x0265 ; e
mov word [0xb8004], 0x026c ; l
mov word [0xb8006], 0x026c ; l
mov word [0xb8008], 0x026f ; o
mov word [0xb890a], 0x022c ; ,
mov word [0xb800c], 0x0220 ;
mov word [0xb890e], 0x0277 ; w
mov word [0xb010], 0x026f ; o
mov word [0xb8012], 0x0272 ; r
mov word [0xb8014], 0x026c ; l
mov word [0xb8016], 0x0264 ; d
mov word [0xb018], 0x0221 ; !
hit
```
Checkpoint #1: sanity check boot.boot.asm

• When ‘make qemu’, you should be able to see “Hello World”
• if you get qemu error, try to switch the machine(e.g. Circinus-30 -> Circinus-31)
• Or open another terminal, and type killall qemu-system-i386
• To quit QEMU VGA screen(virtual monitor), ESC+2 and then type quit
Boot.asm for paging...

• Follow the instructions in HW3 description..
Checkpoint #2: sanity check main.c

• Print “Hello World” after call main
  • Only if main() works fine, it will print Hello World to VGA
  • comment out hlt()
  • uncomment hlt() before submitting to Gradescope

saehansy@uci.edu
What is 4 MB, 4KB

- We are using **4KB-size page** table which can map **4MB of physical memory**
  
  - $4KB = 4 \times 2^{10} = 4 \times 1024 \approx 4 \times 1000$
  
  - $4MB = 4 \times 2^{20} = 4 \times 2^{10} \times 2^{10} = 4KB \times 1024$
  
- Each entry of page table is 4 bytes
- There are 1024 entries in a page table $\rightarrow$ 4KB !!

- Each entry can map 4KB of physical memory since it only stores first 20 bits out of 32 bit address
What is 4 MB, 4KB

- We are using **4KB-size page** table which can map **4MB of physical memory**

  - $4KB = 4 \times 2^{10} = 4 \times 1024 \approx 4 \times 1000$

  - $4MB = 4 \times 2^{20} = 4 \times 2^{10} \times 2^{10} = 4KB \times 1024$

- Each entry of page table is 4 bytes

- There are 1024 entries in a page table $\rightarrow$ 4KB !!

- Each entry can map 4KB of physical memory since it only stores first 20bits out of 32bit address

- 4KB * 1024 entries = 4MB
What is 4 MB, 4KB

- We are using **4KB-size page table** which maps **4MB of physical memory**
- $4KB = 4 \times 2^{10} = 4 \times 1024 \approx 4 \times 1000$
- $4MB = 4 \times 2^{20} = 4 \times 2^{10} \times 2^{10} = 4KB \times 1024$
- Each entry of page table is 4 bytes
- There are 1024 entries in a page table $\rightarrow$ 4KB!!

- Each entry can map 4KB of physical memory since it only stores first 20 bits out of 32 bit address
- $4KB \times 1024$ entries $= 4MB$
- 8MB of physical memory? $\rightarrow$ two page tables
- 256MB of physical memory? $\rightarrow$ 64 page tables
Why align 4096 (4KB)

- $4096 = 0x1000 = 0b 100000000000$ (12bits)
- $4096 \times 2 = 0x2000 = 0b 100000000000$
- $4096 \times 135 = 0x87000 = 0b 1000 0111 00000000000$
- Since last 12 bits are always zero, we don’t need them (instead, we use ‘offset’ from linear address)
- Offset field is 12 bits which can represent 4KB space
Target assembly

• Now, the remaining job is implementing page tables in main.c
• You don’t need to touch anything else

(2) Map the physical memory to each page table

```
section .bss
align 4096
pt_table:
    resb 4096
ptd_table:
    resb 4096

; map first PTD entry to pt table
mov eax, pt_table
or eax, 0b11 ; present + writable
mov [ptd_table], eax

; move page table address to cr3
mov eax, ptd_table
mov cr3, eax
```

(1) declare page tables and page directory aligned to 4KB
look for __attribute__((aligned (xxx)));;

(3) Let ptd_table entries point to the start address of each page table

```
inc ecx ; increase counter
cmp ecx, 1024 ; if counter == 1024, the whole P1 table is mapped
jne .map_pt_table ; else map the next entry
```

(4) Finally, set the start address of ptd_table to cr3
Why triple fault?

• Many reasons..
• Here, the most common case: when it executes lcr3()
• main.c is using virtual address
• main.c is in `kernel.bin` as well as printk() and other stuff
• kernel is loaded at the 1MB mark as specified in linker.ld
• The current kernel.bin size is 17KB which means main() is accessing the address range of 0x10000 ~ 0x170000
• (for fun) As long as you map this region in page table, the program will run fine (without the last for loop that touches every page, also note that the autograder will check the entire mapping)
Check the result of mapping

• make qemu-gdb-nox
• (qemu) info tlb
• If you can’t scroll, save the output to a file
  make qemu-gdb-nox | tee log.txt
Tips for extra credit (VGA)

- Print on console && VGA screen
- if printk(“Hello”) was called, then inside printk(),
- str[0] = ‘H’ = 0x48 = 72
- Our monitor, the VGA device, is expecting 0x248 i.e. some flags + character to print
- How can you make 0x48 to 0x248?
- Note that 0x48 = 0x0000 0048 and 0x248=0x0000 0248
- So we need to modify the third hex digit..
- Remember we modified the last two bits when we map physical memory to page tables?