# Discussion 6

Harishankar Vishwanathan

# Overview

- The boot process
  - BIOS, bootloader, and the kernel
- HW3 overview
  - Setup & outline
  - First steps
  - Paging
  - Tools that will help

# The boot process

BIOS → Bootloader (real mode, 16-bit) → kernel (protected mode, 32-bit)

# BIOS

- Firmware/software that is in the motherboard’s ROM.
- Initialize controllers, network interfaces, etc.
- Ends by loading the bootloader, in xv6 at 0x7c00

# Bootloader

- The BIOS loads this code from the first sector of the hard disk into memory at physical address 0x7c00 and starts executing in real mode.
- Bootloader
  - switch to 32−bit protected mode, jump into C.
  - xv6/bootasm.S

# Kernel

- Load the kernel by reading the kernel ELF file from disk
  - xv6/bootmain.c
- Once in kernel main()
  - Set up page tables.
  - xv6/main.c
  -

# HW3: first steps

- Writing a minimal “Hello World” kernel and booting into it.
- Steps:
  - Use GRUB as our bootloader
    - Needs a header file that uses follows the multiboot specification: multiboot_header.asm
  - Write a boot.asm file that
  - Link them together into kernel.bin
  - Make an ISO using grub2-mkrescue
  - Run our kernel in QEMU

# Demo

- Downloading source files
- Writing our minimal “Hello World” kernel
- Compiling, linking, and booting.

# At this point ...

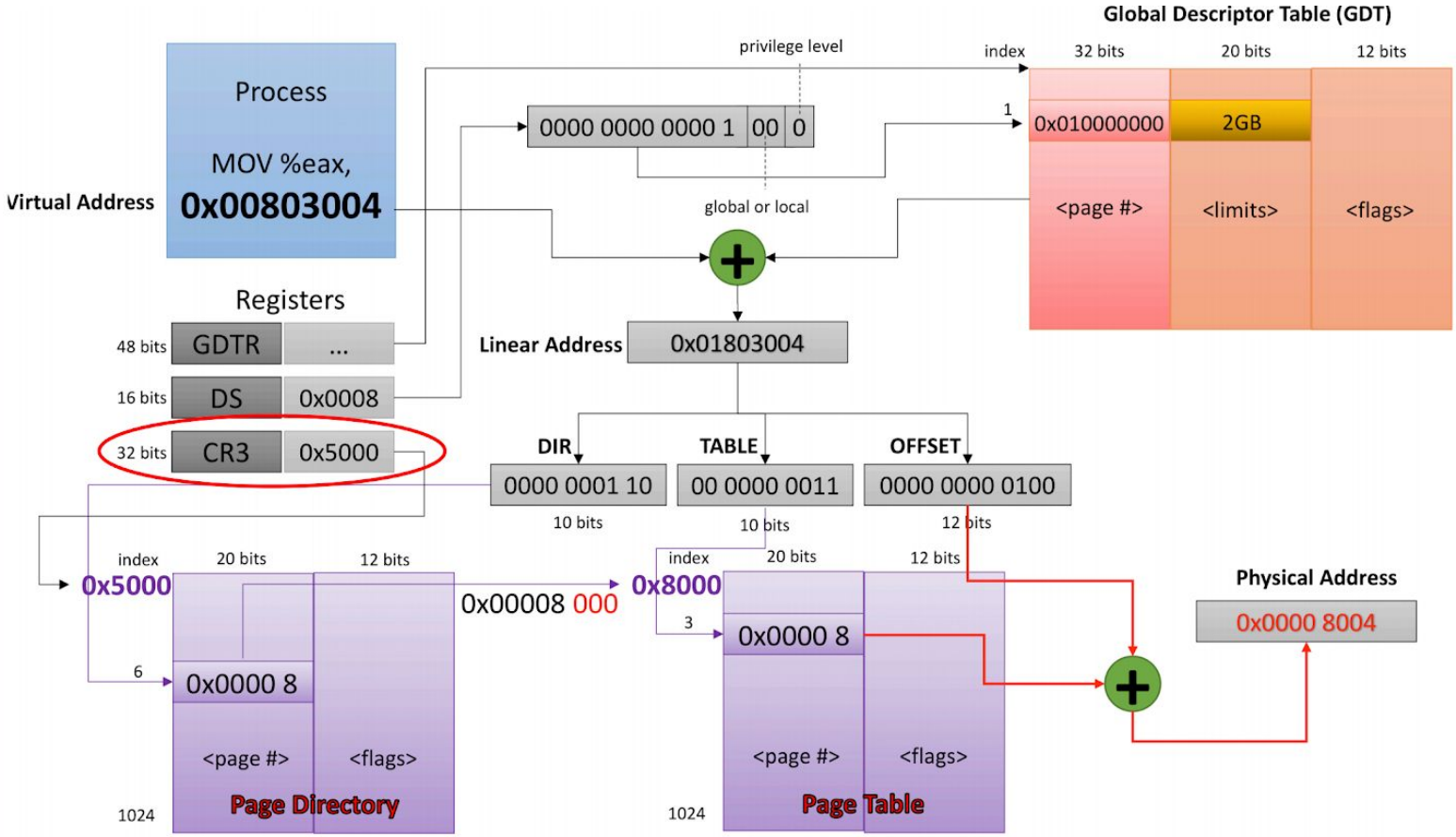
- `make qemu`
  - Boots into our minimal kernel and prints: `Hello World`
- Makefile errors
  - `Recipe:ingredients` syntax
  - Follow the recipes and see what gave you an error
  - List of files (demo)
- Qemu errors
  - Try to switch to a different server (eg. circinus-14 to circinus-15)
  - Or open another terminal and type `killall qemu-system-i386`
  - To quit QEMU virtual monitor, `Alt+2`, followed by `q` or `quit`

# HW4: paging

- Virtual to physical address translation is hardware-assisted by the MMU
- The MMU uses page tables to do this.
  - Page table contain the virtual - physical mapping
- Once paging enabled, every address that the CPU encounters goes through the translation mechanism.
- We set the page table data structures, and enable paging.

# Page Table Recap

- One level with 4MB pages
  - Page Directory -> Physical address
- Two level page tables with 4k pages
  - Page Directory -> Page Table -> physical address
- CR3 contains the base address of the page directory
  - After constructing page table, you need to put base address of page directory into CR3


Global Descriptor Table (GDT)
Process
MOV %eax,
0x00803004
Virtual Address
privilege level
index
32 bits
20 bits
12 bits
0000 0000 0000 1 00 0
1
0x010000000
2GB
global or local
<page #>
<limits>
<flags>
Registers
48 bits
GDTR
...
16 bits
DS
0x0008
32 bits
CR3
0x5000
Linear Address
0x01803004
DIR
TABLE
OFFSET
0000 0001 10
00 0000 0011
0000 0000 0100
10 bits
10 bits
12 bits
index
20 bits
12 bits
0x5000
6
0x0000 8
<page #>
<flags>
1024
Page Directory
0x00008 000
index
20 bits
12 bits
0x8000
3
0x0000 8
<page #>
<flags>
1024
Page Table
Physical Address
0x0000 8004

# Demo 2

- Page tables
  - Follow instructions
  - Enable paging
  - Boot into main
  - Check serial.log

# Page size discussion

- We use 4KB pages
- A Page Table is 4KB in size:
  - Contains 1024 Page Table Entries.
  - Each PTE is 4 bytes long
  - Each PTE points to a 4KB region of physical memory
  - So 1 Page Table maps 4MB of physical memory
- For 8MB physical memory?
  - How many page tables?  (2 page tables)
- For 256 MB physical memory?
  - 64 page tables.

# Implementing page tables in main.c

Similar to what we did in assembly

- Declare regions of memory to hold the Page Table Directory and Page Tables.
- Load the Page Table Entries in Page Table(s) with the correct physical addresses and flags.
- Make the Page Table Directory entries point to the start of each page table
- Load CR3 with the start address of Page Table Directory

# Notes

- Page alignment
  - Page Table Directory address needs to be page aligned.
  - Page Tables addresses need to be page aligned.
- qemu-gdb
  - https://zoo.cs.yale.edu/classes/cs422/2011/lec/l2-hw.shtml (Section: Remote Debugging xv6 under QEMU)
- Tmux (a terminal multiplexer)
  - Just run: `tmux`
  - `Ctrl + B` is the default bind key. `Ctrl +B` followed by `%` should create two multiplexed terminal sessions.
  - Use `Ctrl + B` followed by arrow keys to navigate between the two sessions.
  - https://linuxize.com/post/getting-started-with-tmux/