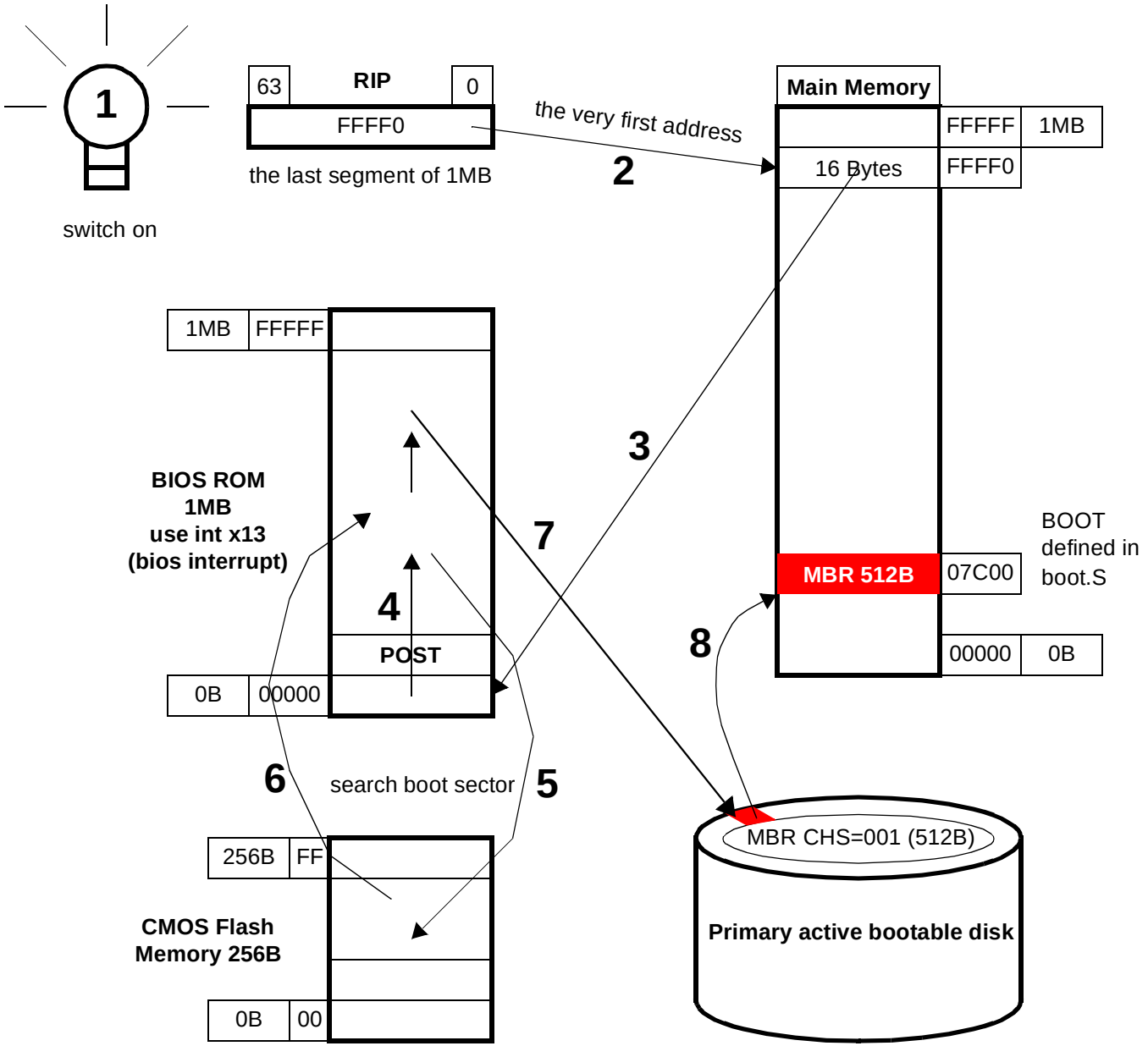
Tue, 1/24/2017 v4.10

Chapter 2 Booting

- Turn the switch on
- RIP = **FFFF0** (last 16B segment), operates in DOS real mode (max 1 MB)
- **BIOS** - basic input output system EEPROM (flash memory)
- CMOS flash memory
- **POST** Power On Self Test - inventory checking
- Search primary active disk for boot record
- BIOS reads *boot.S* MBR master boot record 512 B (HCS001): bootloader (446 B) + partitions (64 B) + signature (2 B)
- **GRUB stage1**: boot.S (**MBR**) -> diskboot.S -> startup.S -> grub_main() in main.c
- **GRUB stage2**: grub_main -> ... -> grub_show_menu -> show_menu -> grub_menu_execute_entry -> **1** grub_parser_execute; **2** grub_command_execute -> grub_linux_boot -> code32_start
- Linux/arch/x86/boot/header.S --> arch/x86/boot/main.c --> arch/x86/boot/pm.c --> arch/x86/boot/pmjump.S
- **startup_32() #1**: arch/x86/boot/compressed/head_32.S -> extract_kernel()
- **startup_32() #2**: arch/x86/kernel/head_32.S: initial_code -> i386_start_kernel
- head_32.c: i386_start_kernel -> start_kernel
- Linux/init/main.c: **start_kernel()**

Machine booting process: Power-on to start_kernel()

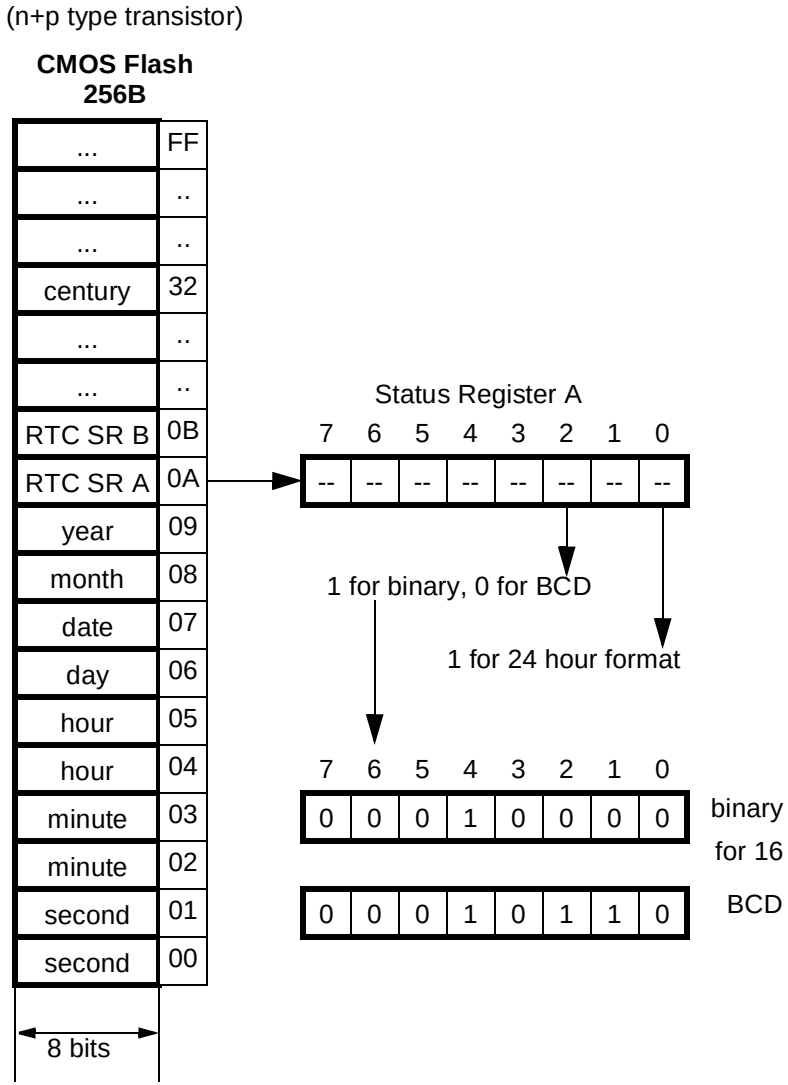


1. Flip the switch
2. RIP - FFFF0 (operate in DOS real mode)
3. BIOS - basic input output system ROM
4. POST power on self test
5. CMOS flash memory
6. Search boot device for OS boot sector
7. Copy MBR to 0x7c00 (defined in boot.S)
8. BIOS loads MBR.
9. Grub stage1: boot.S, diskboot.S, startup.S
10. Grub stage2: show_menu()
11. arch/x86/boot/header.S > main.c ->
12. -> pm.c -> pmjump.S
13. compressed/head_32.S: startup_32() #1
14. kernel/head_32.S: startup_32() #2
15. i386_start_kernel()
16. init/main.c: start_kernel()

Basic Input Output System (BIOS) ROM

- Flip the switch - power good signal from the power supply
- instruction pointer (RIP/EIP or program counter) is loaded FFFF0 (FFFFF is 1 MB - IBM DOS real mode). FFFF0 is hardwired by microprocessor/mobo.
- FFFF0 has a jump instruction to jump to BIOS ROM.
- Executes routines at the memory mapped to BIOS ROM
- power on self test (POST)
- look for video card, video card's built in BIOS program at xC000, executes.
- look for other device's ROMs, IDE/ATA disk BIOS at xC8000, executes.
- display startup screen
- memory count-up test
- perform a system inventory, search for COM and LPT ports.
- Detect and configure Plug and Play devices
- display a summary screen about system configuration
- search for a drive to boot from: A, B, C, ... using the CMOS boot sequence setting.
- Search for a master boot record at CHS=001.
- Copy MBR to RAM at 0x7C00 (BOOT defined in boot.S) -> **boot.S=grup stage 1**

CMOS Flash memory
Basic Input Output Systems (BIOS EEPROM)

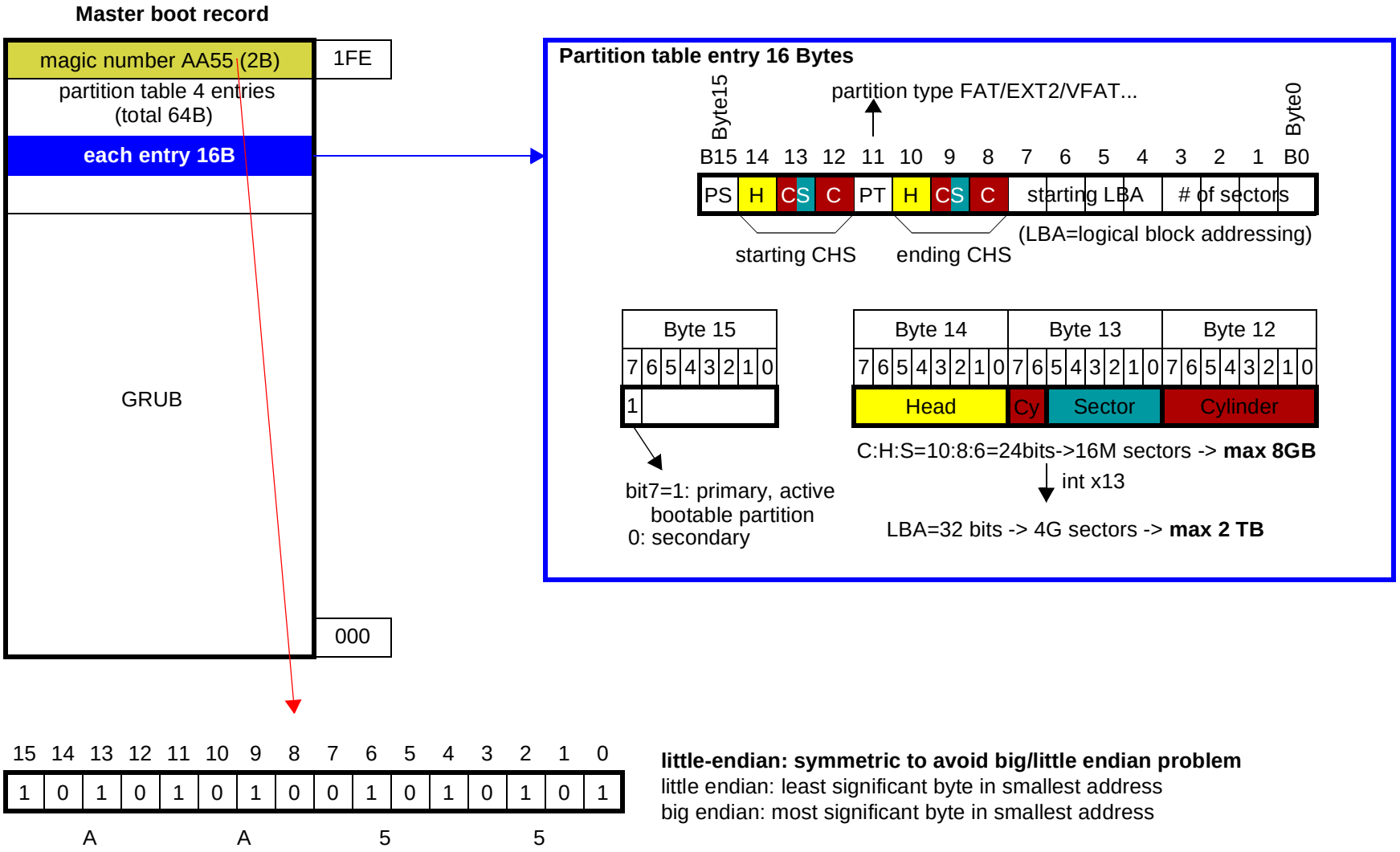


BIOS

- in mega btyes
- Vendors include
- American Megatrends (AMI)
- Phoenix
- Insyde Software
- AwardBIOS
- AMIBIOS
- SeaBIOS

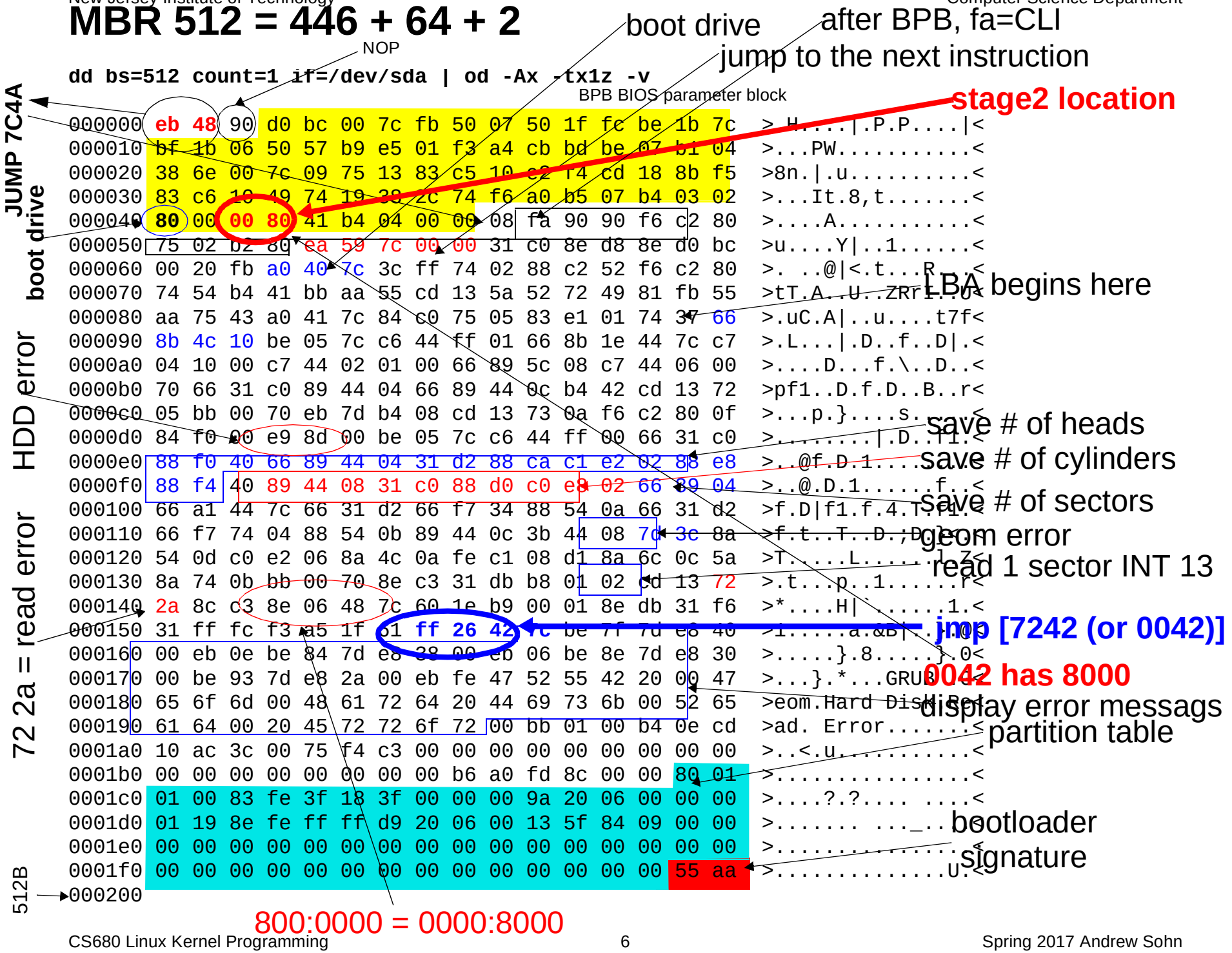
Master Boot Record C:H:S=0:0:1 (512 B)

Cylinder 0: Head 0: Sector 1



little-endian: symmetric to avoid big/little endian problem
little endian: least significant byte in smallest address
big endian: most significant byte in smallest address

MBR 512 = 446 + 64 + 2



GRUB Stage 1

BIOS copies MBR boot.S to 0x7c00

- jmp to boot.S 0x7c00

grub-core/boot/i386/pc

boot.S at 0x7c00

- copy diskboot.S image to *buffer*
  GRUB_BOOT_MACHINE_BUFFER_SEG 0x7000
- copy from *buffer* to *kernel_address*
  GRUB_BOOT_MACHINE_KERNEL_ADDR 0x8000
- jmp to kernel_address 0x8000

grub-core/kern/i386

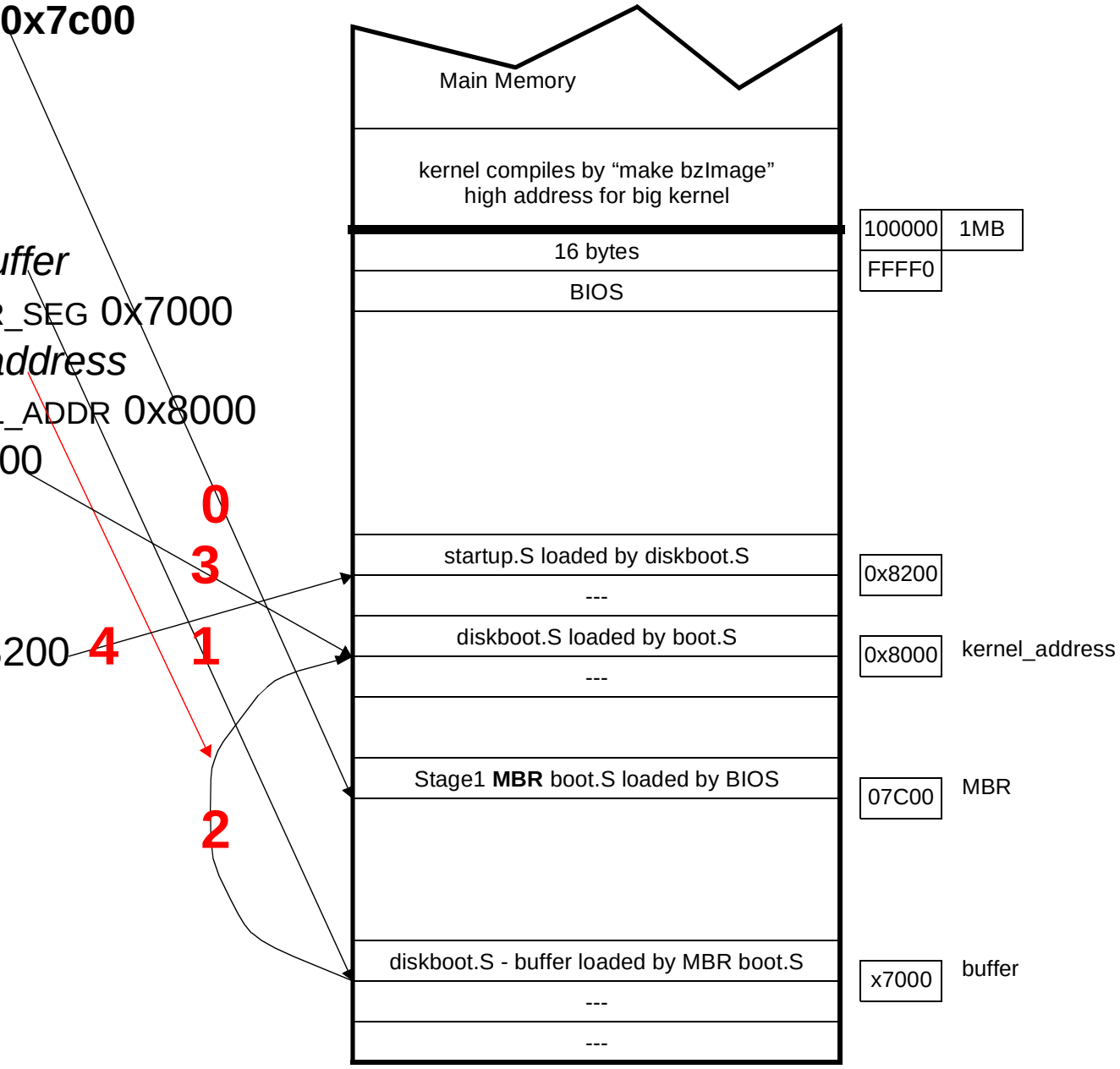
diskboot.S

- copy startup.S image to 0x8200
- jump to 0x8200 startup.S

grub-core/kern/i386

startup.S

- call EXT_C(grub_main)



x7000	8KB

x8000	?KB

GRUB Stage 2

main.c

- grub_machine_init()
- grub_load_modules();
  -> grub_dl_load_core();
  -> grub_dl_call_init()
  -> mod->init(mod)
- grub_register_core_commands()
  such as ls, insmod, set, unset, env vars
- grub_load_config()
  -> grub_parser_execute
- grub_load_normal_mode()
  -> grub_dl_load("normal")
  -> grub_command_execute("normal")
  -> cmd->func(cmd,argc,argv)

  => grub_cmd_normal()

- grub_enter_normal_mode()
  -> grub_normal_execute(config)
  -> **1 read_config_file(config)**
  -> **2 grub_show_menu()**

commands/boot.c

- GRUB_MOD_INIT(boot)
  -> grub_register_command("boot",..);
  -> grub_register_command_prio()
  -> **grub_command_list** <- *global var*

  => grub_cmd_boot()

- GRUB_MOD_FINI(boot)

- GRUB_MOD_INIT(linux)

  => grub_cmd_linux()

- grub_cmd_linux
  -> grub_loader_set(grub_linux_boot,...)
  -> grub_loader_boot_func = g_l_b
  -> grub_loader_loaded = 1
  -> loaded = 1

- GRUB_MOD_FINI(boot)

GRUB Stage 2

menu.c

- **show_menu()**
  -> run_menu()
  -> **e** = grub_menu_get_entry()
  -> grub_menu_execute_entry(**e**)
    -> **a** grub_script_execute_sorucecode
    -> **b** grub_cmd_execute("boot",...)

- **a grub_script_execute_sourcecode**
  (entry->sourcecode)
  -> parsed_script = grub_script_parse(line)
  -> grub_script_execute(parsed_script)
    -> grub_script_execute_cmd(script->cmd)
      -> cmd->exec(cmd)

- **b grub_cmd_execute("boot",...)**
  -> grub_cmd_boot()
  -> grub_loader_boot()
    -> grub_loader_boot_func()
  -> grub_linux_boot()
    -> state.eip = params->**code32_start**
    -> grub_relocator32_boot()

commands/boot.c

- GRUB_MOD_INIT(boot)
  -> grub_cmd_boot()

- grub_cmd_boot
  -> grub_loader_boot(grub_linux_boot,...)
    -> grub_loader_boot_func()
- GRUB_MOD_INIT(linux)
  -> grub_cmd_linux()
- grub_cmd_linux
  -> grub_loader_set(**grub_linux_boot**,...)
    -> grub_loader_boot_func = g_l_b
    -> grub_loader_loaded = 1
  -> loaded = 1

-> grub_relocator_prepare_relocs()
  -> grub_cpu_relocator_jumper()
  -> *relstart = rels0;
-> asm volatile ("cli")
-> ((void (*) (void)) relst) ()
--> code32_start --> 0x100000

Linux/arch/x86/boot

header.S:
- code32_start: .long 0x100000 in header.S
- calll main # Jump to C code (should not return)

main.c: main(void)
- copy_boot_params();
- ...
- detect_memory();
- keyboard_init();
- go_to_protected_mode();

pm.c: go_to_protected_mode(void)
- setup_idt()
- setup_gdt()
- protected_mode_jump(code32_start)

pmjump.S:
protected_mode_jump:
- .byte 0x66, 0xea # ljmpl opcode
- jmpl *%eax
  # Jump to the 32-bit entrypoint

pmjump.S:
protected_mode_jump:
- .byte 0x66, 0xea # ljmpl opcode
- jmpl *%eax
  # Jump to the 32-bit entrypoint

compressed/head_32.S:
ENTRY(startup_32):
- call extract_kernel
- jmp *%eax

compressed/misc.c: *extract_kerne(..)
- __decompress(...)

x86/kernel/head_32.S:
ENTRY(startup_32):
- jmp *(initial_code)
- ENTRY(initial_code)
- .long i386_start_kernel

x86/kernel/head_32.c:
- asmlinkage __visible void __init i386_start_kernel(void)
  -> **start_kernel();**