Find your way through the x86 firmware maze

Gerd Hoffmann <kraxel@redhat.com>

October 21nd, 2013

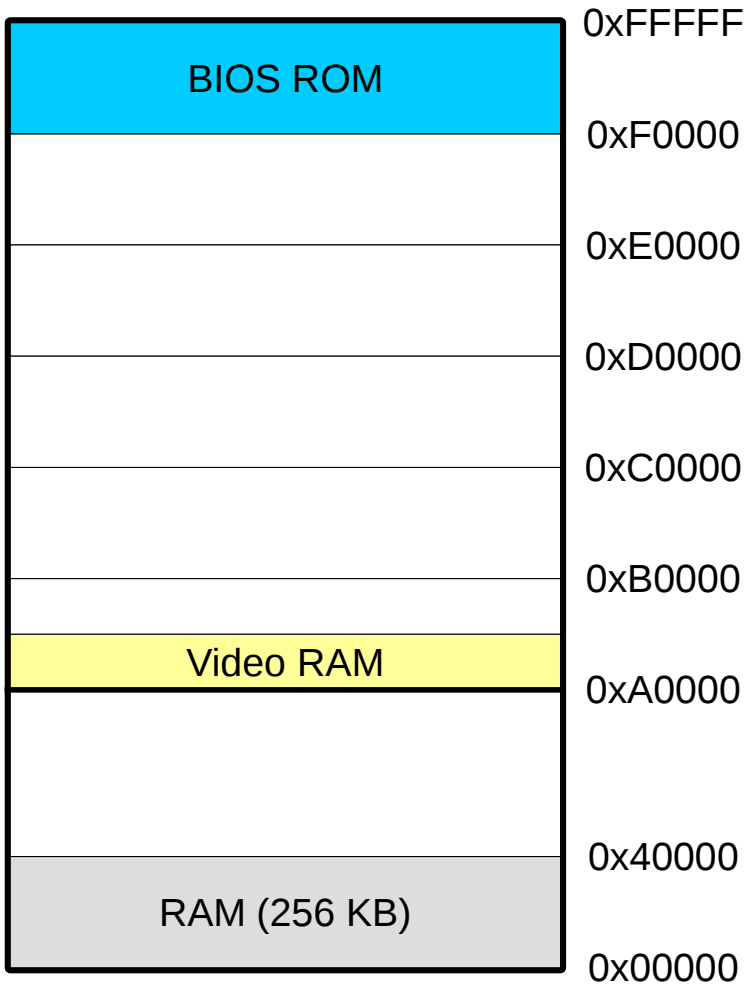
KVM Forum, Edinburgh

Outline

- Some history
- Boot process
  - Focus on seabios
- UEFI
- coreboot

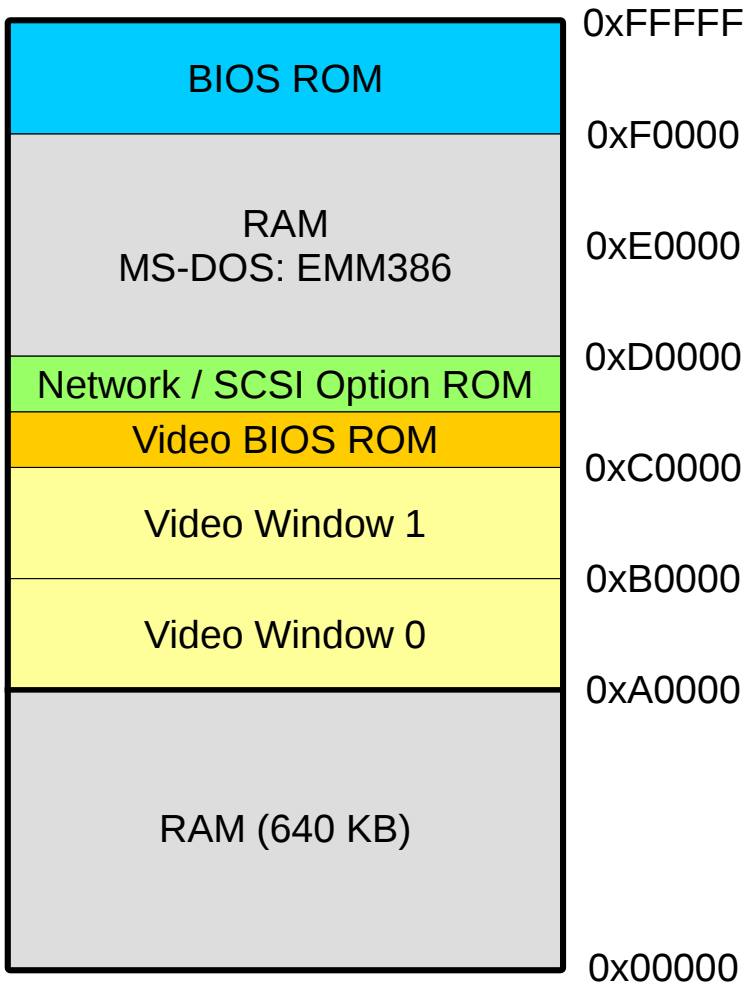
640k is enough for everybody (XT)

- Real mode
- Start exec: 0xFFFF0
- BIOS Interfaces
  - INT 0x10 -- video
  - INT 0x13 – disk
- Interfaces still in use
  - Boot loader disk access
  - vesafb



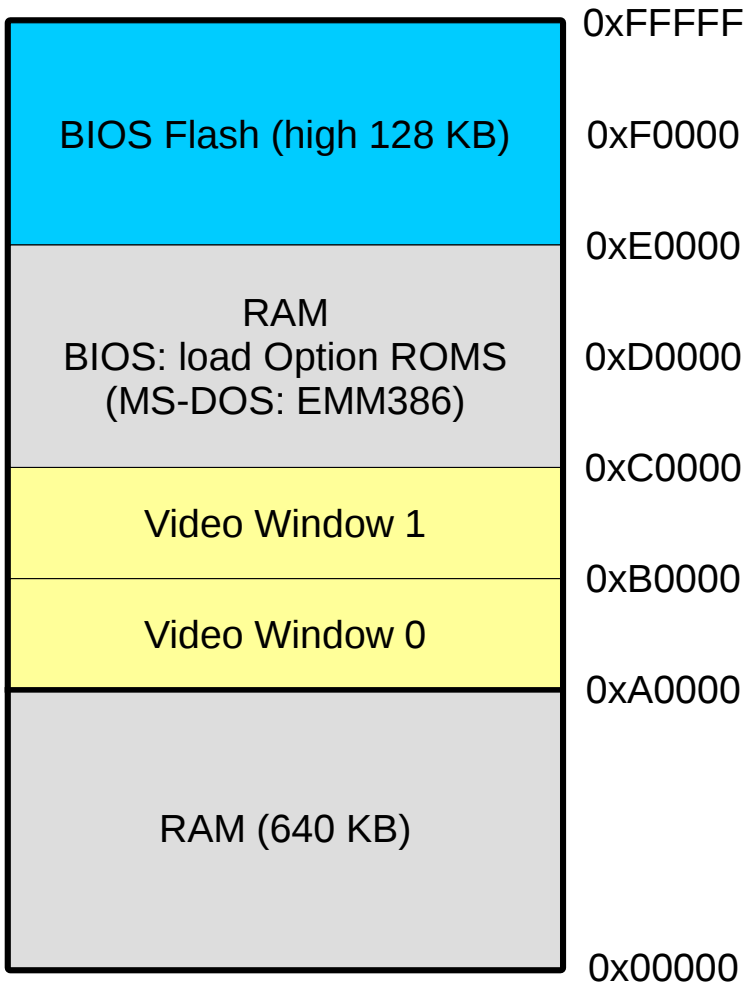
A few years later (i386, still ISA)

- Every real mode address space bit is used.
- ROMs extend BIOS interfaces.
  - VGA: better video.
  - NIC: network boot.
  - HBA: scsi disk boot.
- Qemu up to v0.11 (2009) loaded roms this way.



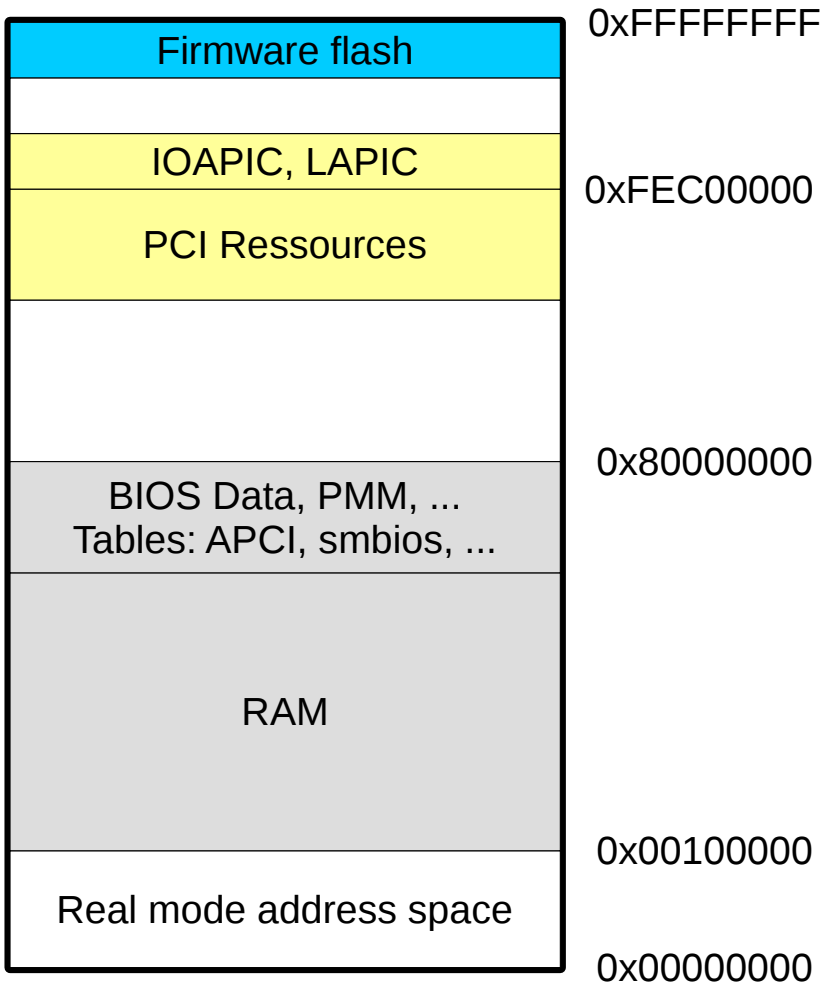
How it looks today (i686 / x86-64, PCI)

- Large BIOS flash
  - high 128k below 1M
- Option ROMs loaded from PCI ROM Bar (or firmware flash).
  - Relocate to high mem, using PMM + big real mode
- Only essential stuff remains below 1M



How it looks today (above 1M)

- Complete flash mapped below 4G
- BIOS Data area at end of RAM
  - private data
  - PMM allocations
  - BIOS tables



seabios initialization sequence (#1)

- Enter 32bit mode.
- Detect RAM, init malloc, relocate into RAM.
- Setup PMM (POST Memory Manager).
- Qemu builds:
  - PCI ressource allocation.
  - Apply chipset & device tweaks.
    - Init pmbase (+ enable pmtimer).
    - Enable legacy IDE ports.
  - Generate / load BIOS tables
    - MPTable, SMBIOS, ACPI

seabios initialization sequence (#2)

- Load vgabios + enable vga console
  - needs 16bit mode transition.
- Detect hardware + init drivers
  - Keyboard, disks+cdroms, usb storage.
- Load non-vga option roms
  - needs 16bit mode transition too.
- Enter 16bit mode, kick boot.
  - Handle BIOS interrupts.

fw_cfg - firmware config interface

- No BIOS Setup, using qemu command line.
  - Bootorder, bootmenu.
- Emulation + real hardware differs.
  - Incomplete emulation.
  - Pimped up i440fx/piix4.
- Load option roms.

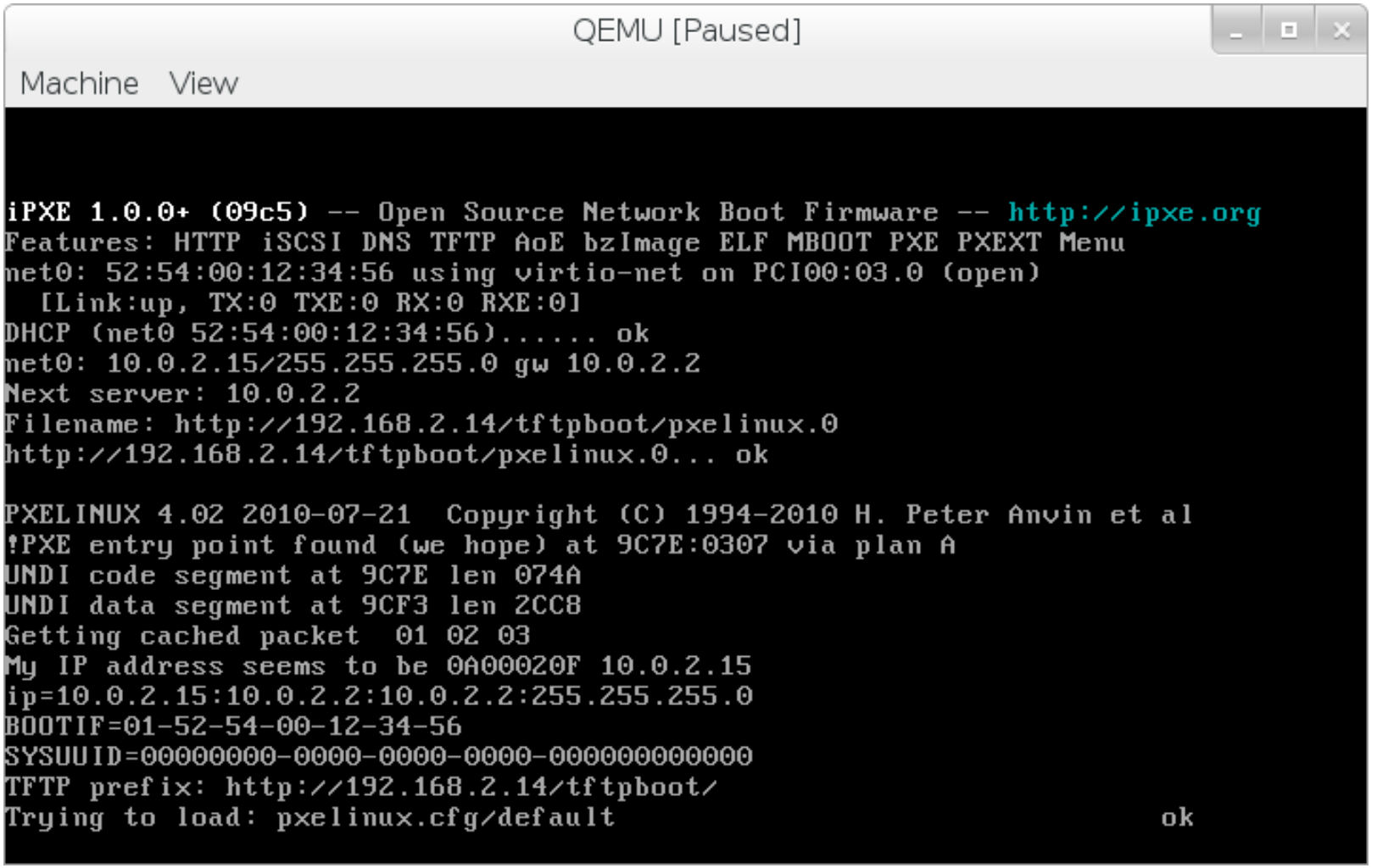
BIOS tables

- Informations qemu/firmware provide for the OS
  - MPTable (MP = Multi Processor).
    - Created by seabios.
    - SMP + APIC.
  - SMBIOS (System Management BIOS)
    - DMIBIOS successor (dmidecode).
    - Created by seabios.
    - Partly configurable via fw_cfg.
  - ACPI (Advanced Configuration and Power Interface)
    - Old: seabios reads info from fw_cfg and generates tables.
    - New: qemu generates acpi tables, seabios loads them from fw_cfg.

ROM: vgabios

- lgpl vgabios
  - Using bcc (real mode compiler)
  - Still used for blobs in qemu.git
- seavgabios
  - Using .code16gcc support in binutils, like seabios.
  - Lots of 32bit prefixes in the code.
    - Old x86emu versions barf on it.
    - Old Xorg vesa driver breaks (-vga std, x86_64).
- support VBE 2.0 (VESA BIOS Extensions).

ROM: ipxe (network boot).



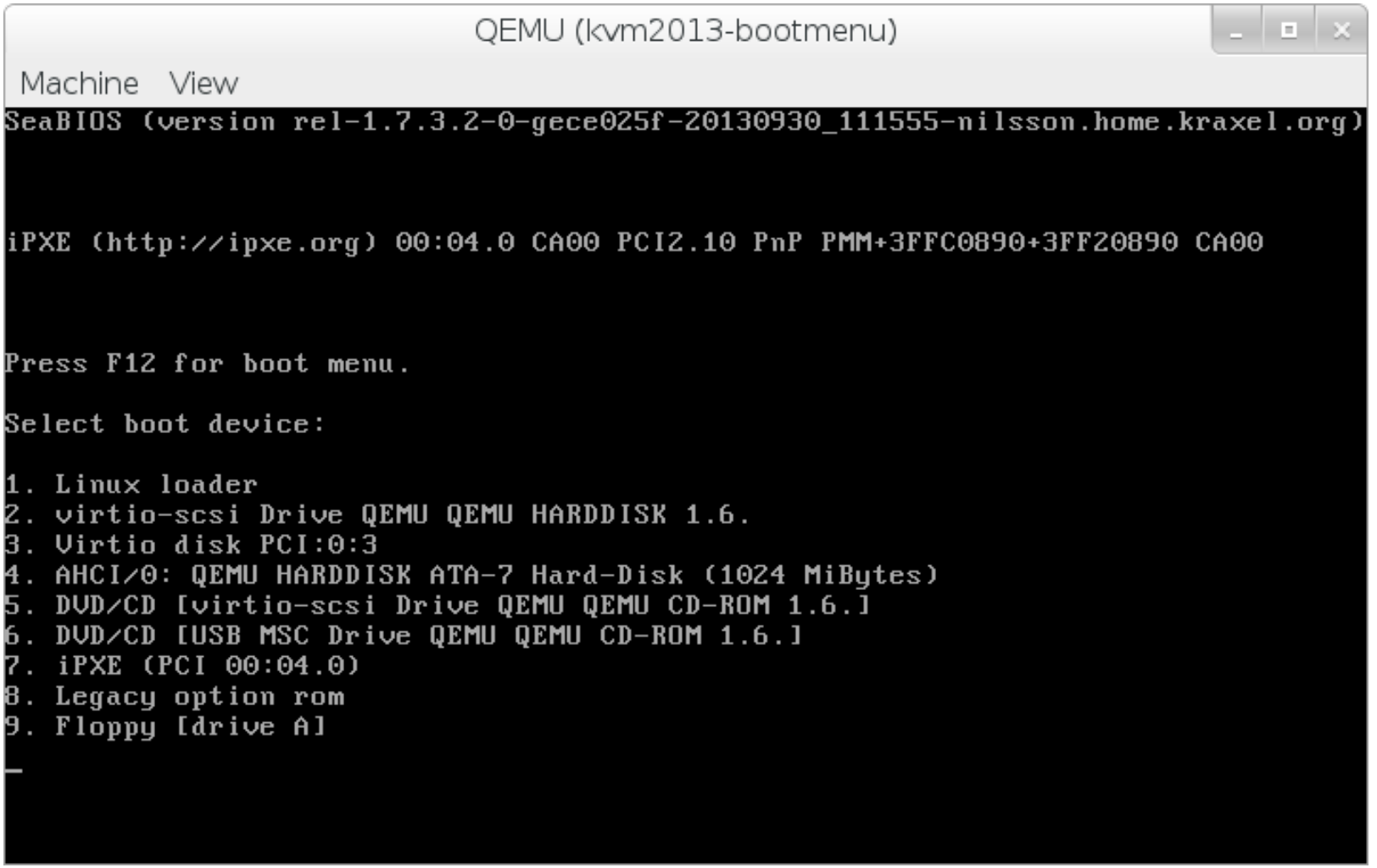
non-ide disk/cdrom boot.

- ROMS are usable.
  - extboot (obsolete, disk only).
  - lsi53c895a: vendor rom.
- SCSI support is in seabios.
  - generic code is there anyway (usb).
  - only need code to detect HBA + send SCSI requests.
  - most qemu HBAs have seabios drivers these days.
- AHCI / virtio-blk drivers are in seabios too.

ROM: others

- Direct kernel boot
  - linuxboot.bin, multiboot.bin
- lapic accel for WinXP
  - kvmvapic.bin
- These ROMs are loaded from fw_cfg.

seabios boot menu



Where to go from here?

- BIOS doesn't fit any more.
  - Real mode limits even boot loaders these days.
- UEFI.
- coreboot.

UEFI

- Multiarch: ia64, ia32, x64, arm.
- Modern feature set.
  - Network support.
  - Filesystem support.
  - Boot menu can select OS.
- Must-have for Windows support.

UEFI with qemu

- Tianocore / EDK2 (EFI Developer Kit)
  - OVMF (Open Virtual Machine Firmware)
- CSM (Compatibility Support Module):
  - Allows Firmware to support both EFI and non-EFI OSes.
  - Optional, brings back (some) real mode mess.
  - seabios can be compiled as csm for ovmf.

OVMF qemu vga support

- without CSM
  - EFI driver: QemuVideoDxe
  - supports stdvga, cirrus, qxl
- with CSM:
  - using vgabios ...

OVMF qemu storage support

- EDK2 has IDE drivers.
- OVMF has virtio drivers.
  - VirtioBlkDxe.
  - VirtioScsiDxe.
- fw_cfg bootorder is supported.

UEFI network boot roms

bash

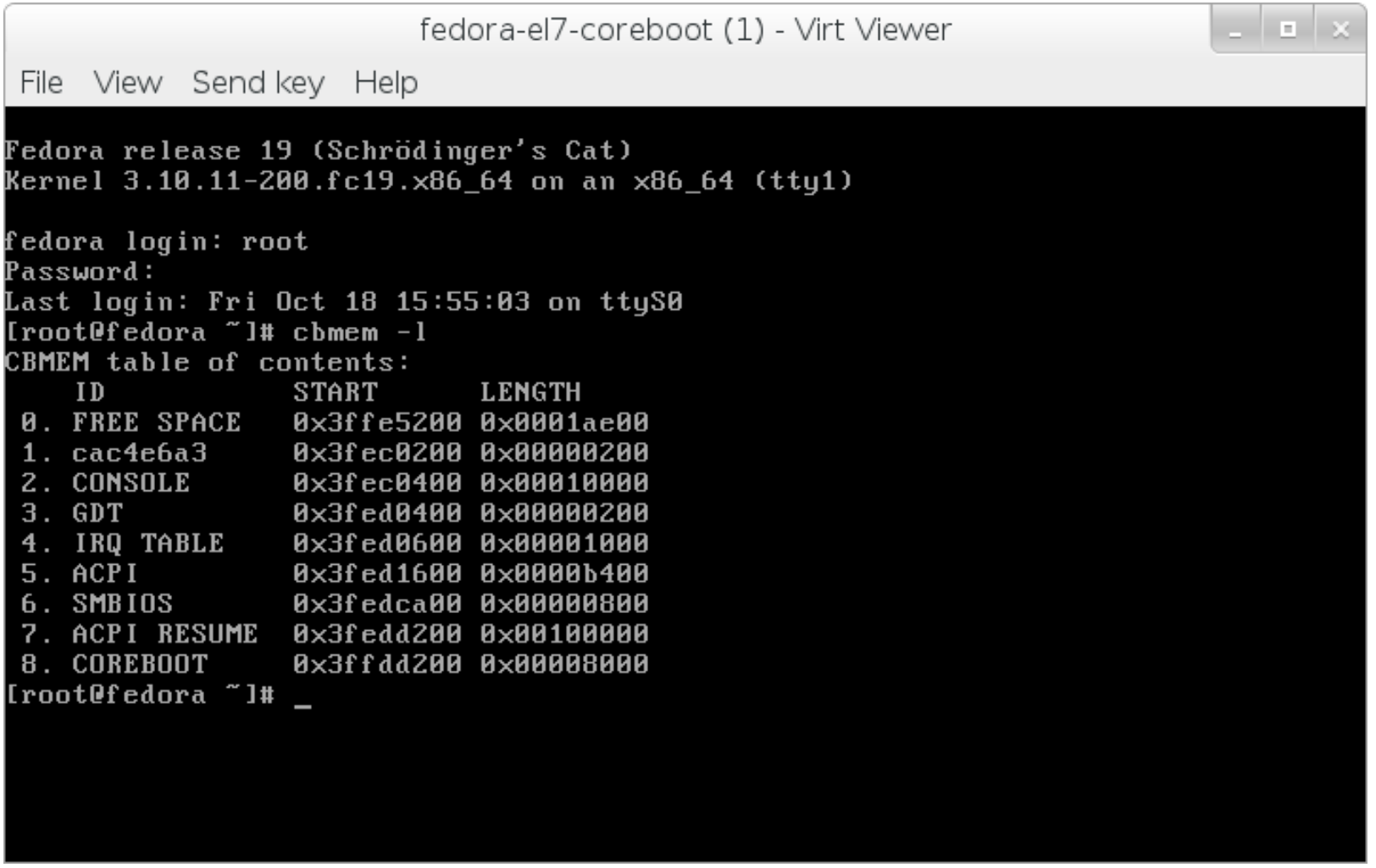
```
$ EfiRom --dump 1af41000.rom
Image 1 -- Offset 0x0
  ROM header contents
    Signature               0xAA55
    PCIR offset             0x001C
    Signature                PCIR
    Vendor ID                0x1AF4
    Device ID                0x1000
    [ ... ]
    Code type                0x00
Image 2 -- Offset 0x10600
  ROM header contents
    [ ... ]
    Code type                0x03   (EFI image)
  EFI ROM header contents
    EFI Signature           0x0EF1
    Compression Type        0x0001 (compressed)
    Machine type            0x014C (IA32)
    Subsystem               0x000B (EFI boot service driver)
    EFI image offset        0x0034 (@0x10634)
Image 3 -- Offset 0x1D800
    [ ... ]
    Machine type            0x8664 (unknown)
    [ ... ]
$
```


OVMF boot menu

Coreboot

- Project started as linuxbios.
- Supports x86, arm.
- Boot process
  - Initialize hardware.
  - Deploy cbtables.
  - Hand off control to payload.
    - Included in the flash image.
    - Payload needs no hardware-specific code.
- No nic/disk/... drivers.

Coreboot cbtables



Coreboot payloads

- seabios
- tianocore (wip)
- grub2
- linux kernel
- memtest

Coreboot on QEMU

- Supports both i440fx and q35.
  - Not in a single ROM though.
- Limited fw_cfg support.
  - Doesn't read smbios entries.
  - SMP uses NR_CPUS only.
- ACPI support has some bugs.
  - Loading tables from qemu will fix that.

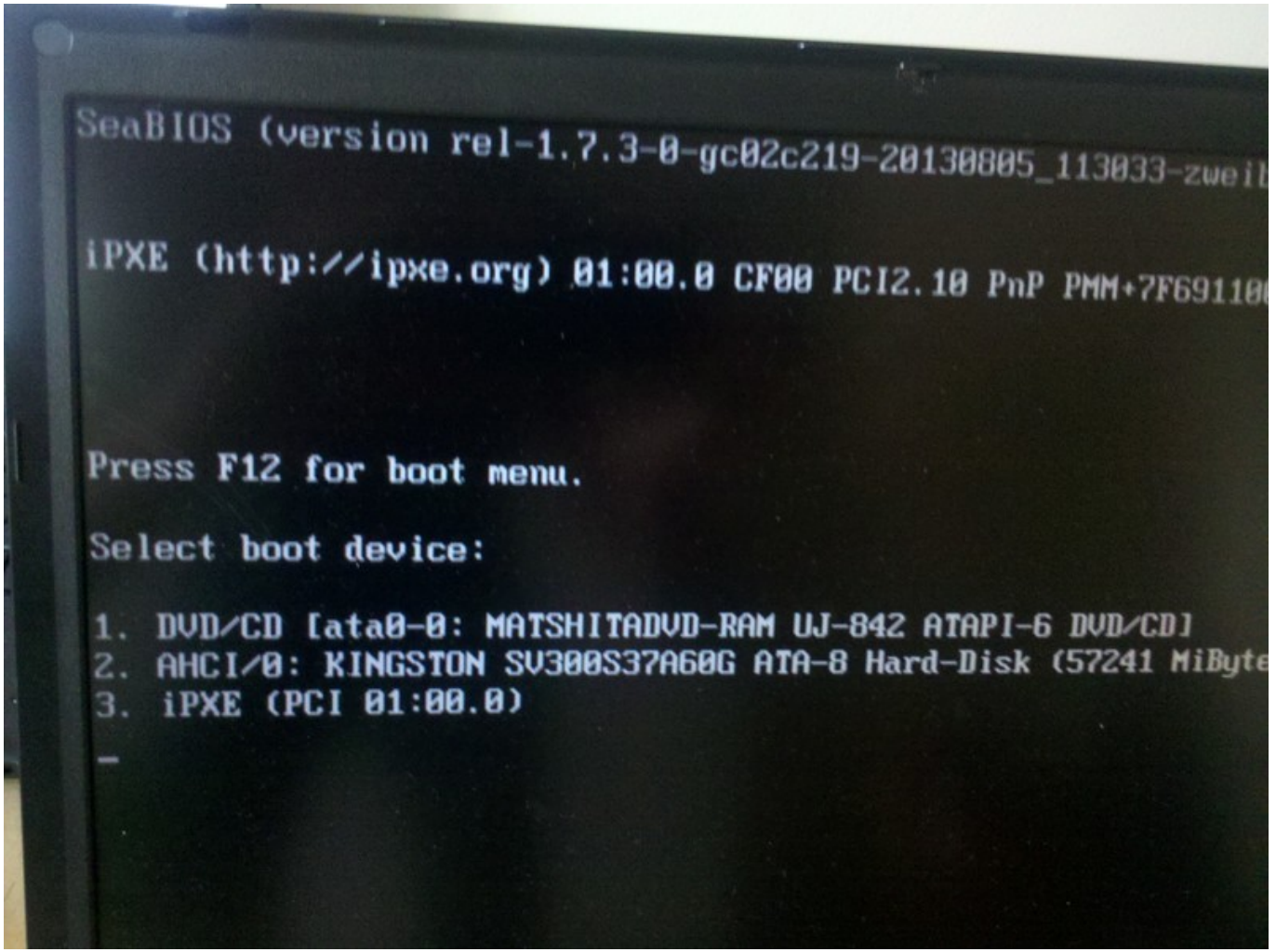
Coreboot flash structure

kraxel@zweiblum:~/tmp

```
$ cbfstool coreboot.rom print
coreboot.rom: 2048 kB, bootblocksize 2328, romsize 2097152, offset 0x0
alignment: 64 bytes

Name                           Offset     Type         Size
cmos_layout.bin                0x0        cmos_layout  1556
pci8086,27a2.rom               0x640      optionrom    65536
fallback/romstage              0x10680    stage        45439
fallback/coreboot_ram          0x1b840    stage        122295
fallback/payload               0x39640    payload      51626
pci8086,109a.rom               0x46040    raw          66560
etc/ps2-keyboard-spinup        0x56480    raw          8
etc/boot-menu-wait             0x564c0    raw          8
(empty)                        0x56500    null         1741208
$
```


Coreboot on Lenovo T60



Hands on: firmware playground

- Firmware repository
  - http://www.kraxel.org/repos/
  - Jenkins building from upstream git master
    - seabios, ipxe, edk2, coreboot
  - Firmware images
    - Ready for use via “-bios $file” or “-L $path”
  - Tools
    - EfiRom: edk2.git-tools
    - cbmem, cbfstool: coreboot.git-tools

Hands on: troubleshooting

- ioport 0x402 has established for firmware debug logging
  - qemu -chardev stdio,id=fw \
    -device isa-debugcon,iobase=0x402,chardev=fw
- coreboot logs can be inspected after boot using 'cbmem -c'