Homework 1

Computer Organization and Architecture

Software Engineering Program,

Department of Computer Engineering,

School of Engineering, KMITL

67011352 Theepakorn Phayonrat

Topic 1: Linux Booting (Raspberry Pi)

Question 1: Draw and explain the full boot sequence of Raspberry Pi (from power-on to login).

Answer:

When powered on, the Raspberry Pi begins execution from the SoC's built-in boot ROM. The sequence is:

1. **First-stage bootloader** (in SoC ROM) runs on the GPU and loads `bootcode.bin` from the SD card.
2. **Second-stage bootloader** `bootcode.bin` initializes SDRAM and loads `start.elf`.
3. **Firmware** `start.elf` reads configuration files (`config.txt`, `cmdline.txt`) and loads the device tree blob.
4. It then loads the Linux kernel image (`kernel.img`) into memory.
5. Control passes to the ARM CPU, which executes the kernel.
6. The kernel initializes hardware, mounts the root filesystem, and starts `systemd`.
7. `systemd` launches necessary services and finally presents the login prompt.

(Source: https://www.raspberrypi.com/documentation/computers/)

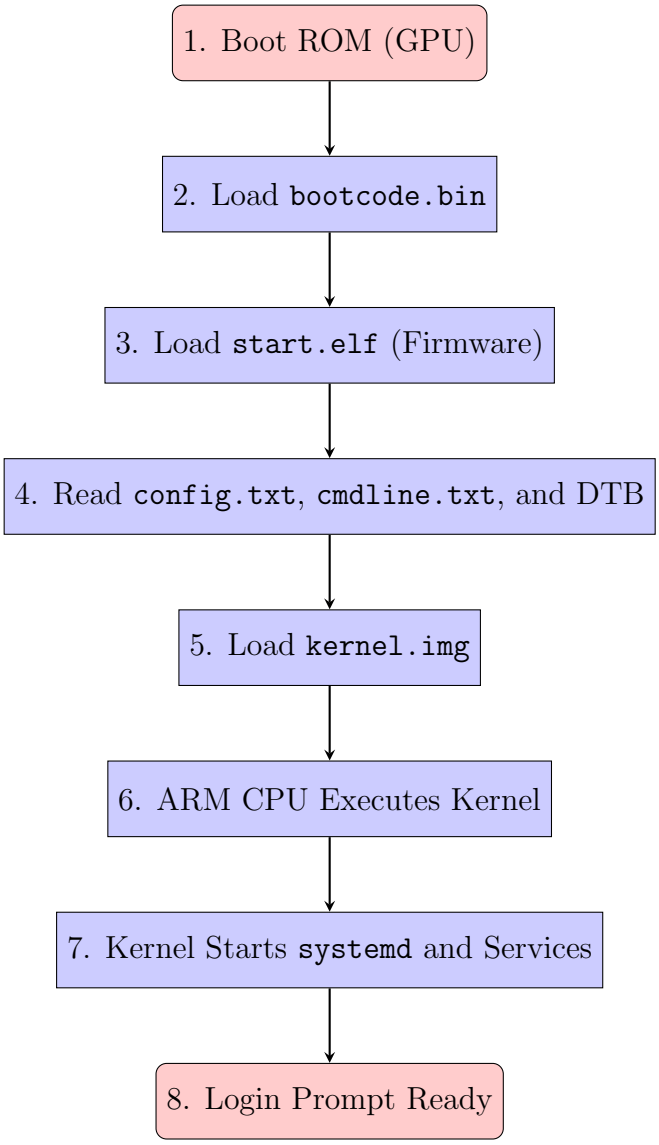
Question 2: If the Raspberry Pi fails to boot after update, what steps and files would you check? Explain your troubleshooting process.

Answer:

If the Raspberry Pi fails to boot after an update, follow these steps:

1. Check the SD card's `/boot` partition and verify the presence of critical files: `bootcode.bin`, `start.elf`, `kernel.img`, `config.txt`, `cmdline.txt`, and device tree files.
2. Inspect `config.txt` and `cmdline.txt` for syntax errors or incorrect parameters.
3. Attach a serial console to view early boot logs if there is no display output.
4. Test the SD card for corruption; reflash the OS if necessary.
5. As a final check, boot with a known good SD card to isolate hardware issues.

(Source: https://forums.raspberrypi.com/viewtopic.php?t=260328)


1. Boot ROM (GPU)
2. Load bootcode.bin
3. Load start.elf (Firmware)
4. Read config.txt, cmdline.txt, and DTB
5. Load kernel.img
6. ARM CPU Executes Kernel
7. Kernel Starts systemd and Services
8. Login Prompt Ready

Topic 2: BIOS vs UEFI

Question 1: Create a comparison table (BIOS vs UEFI) and explain why Raspberry Pi doesn't use them and explain why Raspberry Pi doesn't use them.

Answer:

Comparison Table

Feature	BIOS (Legacy)	UEFI
Firmware Type	Legacy PC-specific code	Modern modular interface
Initialization	Basic hardware init	Rich boot services, network booting
Architecture Support	Primarily x86	Multi-architecture (x86, ARM, etc.)
Boot Media	MBR only	GPT, large disk support
Extensibility	Minimal	Extensible, supports shell and drivers

Why Raspberry Pi Doesn't Use BIOS/UEFI

Raspberry Pi boots directly using proprietary Broadcom firmware built into the SoC. This design avoids a separate firmware chip, simplifies hardware, reduces cost, and ensures a lightweight boot process.
(Source: https://raspberrypi.stackexchange.com/questions/8475/what-bios-does-raspberry-pi-use)

Question 2: If Raspberry Pi adopts UEFI, what would be the pros & cons?

Answer:

Pros:

- Standardized boot process.
- Support for GPT and large drives.
- Easier network booting.

Cons:

- More complex boot process.
- Increased firmware size.
- Higher development and maintenance overhead.

(Source: https://www.eisfunke.com/posts/2023/uefi-boot-on-raspberry-pi-3.html)

Topic 3: Init and Systemd

Question 1: Explain how systemd manages a real service (e.g., SSH, Bluetooth).

Answer:

For example, the SSH service is managed by `systemd` using the unit file `ssh.service`. To control it:

```
sudo systemctl enable ssh     # Enable service on boot
sudo systemctl start ssh      # Start service now
sudo systemctl status ssh     # Check status
sudo journalctl -u ssh        # View logs
```

Systemd handles dependencies and ensures the service runs in the proper order during boot.
(Source: https://opensource.com/article/20/5/systemd-startup)

Question 2: Map runlevel 5 to systemd target. Show how to change default target via CLI and risks.

Answer:

Runlevel 5 (multi-user mode with GUI) maps to `graphical.target`. To change the default target:

```
sudo systemctl set-default graphical.target
```

To switch immediately:

```
sudo systemctl isolate graphical.target
```

Risks: Using `isolate` can stop active services unexpectedly. Changing the default target to `multi-user.target` will disable the GUI until changed back.
(Source: https://askubuntu.com/questions/788323/how-do-i-change-the-runlevel-on-systemd)