Qi - Lightweight Boot Loader Applied in Mobile and Embedded Devices

气

Jim Huang (jserv) &
Matt Hsu
from 0xlab

FreedomHEC 2009, Taipei
June 10, 2009

Rights to copy

Our Background

We are Taiwanese engineers who are always enthusiastic in modern technologies and open source software development.

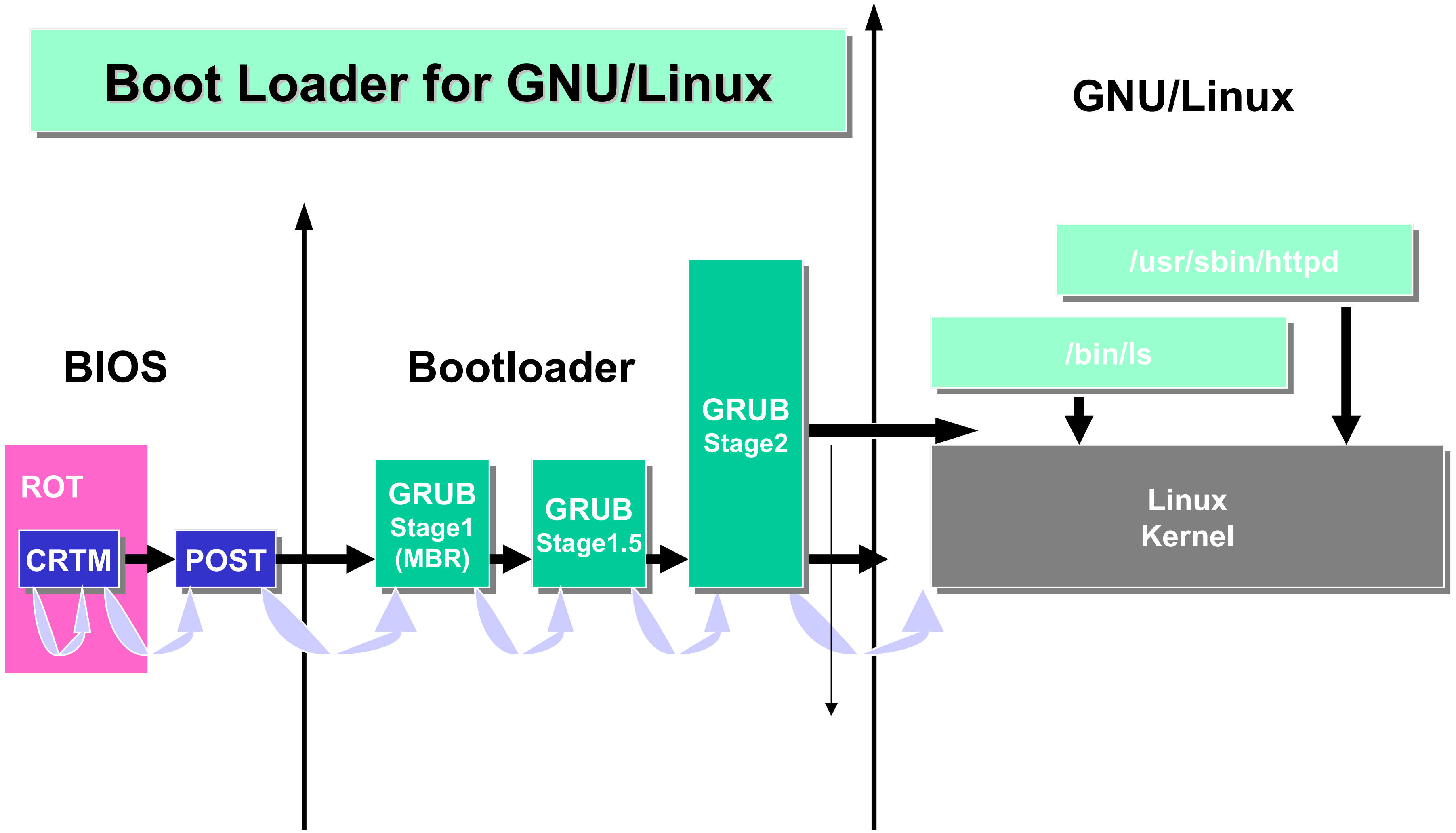
- Jim Huang (jserv) <jserv@0xlab.org>
  - 0xlab co-founder, Openmoko coreteam, LXDE co-founder, Kaffe/Free Java developer
  - Involved in design/implementation for consumer electronics, such as mobile phone, GPS/PND, Digital TV, mobile TV, etc.
- Matt Hsu <matt@0xlab.org>
  - 0xlab kernel maintainer, Openmoko kernel developer
  - {u-boot,kernel}-{hxd8,gta02,gta03/3d7k}
  - Dash Express (the first two-way, Internet-connected GPS navigation system)

Agenda

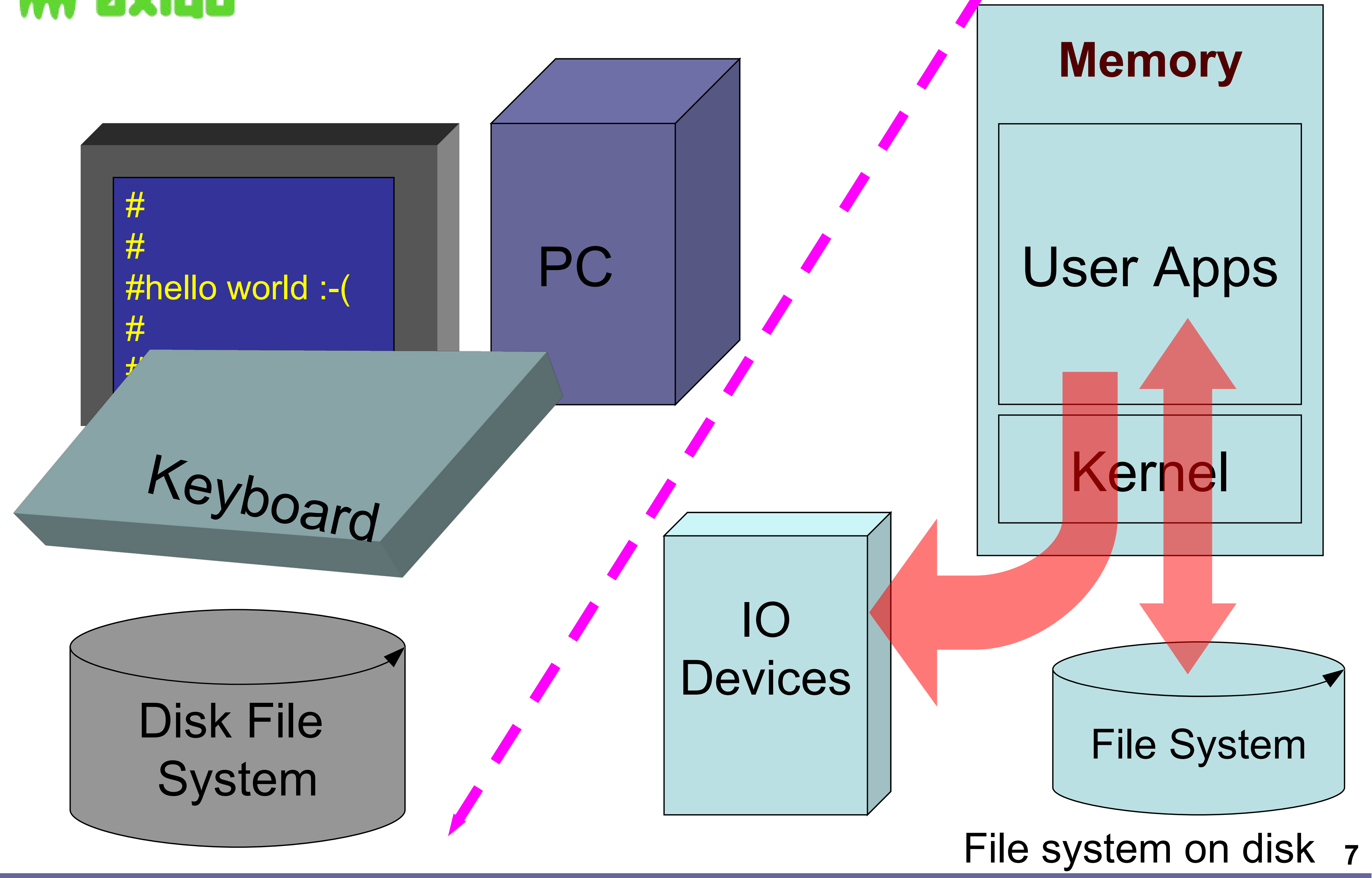
- Evolution of Boot Loader
  - The “function” of boot loaders
    - Our experience from Openmoko project
  - Applied in PC and Embedded Systems
- Overview of Qi
  - Principal: KISS (Keep It Simple and Stupid)
- Practical Qi
  - Real examples following the idea behind Qi
- Future Perspectives

- **Evolution of Boot Loader**
  - **The “function” of boot loaders**
    - **Our experience from Openmoko project**
  - **Applied in PC and Embedded Systems**
- Overview of Qi
  - Principal: KISS (Keep It Simple and Stupid)
- Practical Qi
  - Real examples following the idea behind Qi
- Future Perspectives

Linux Bootstraping

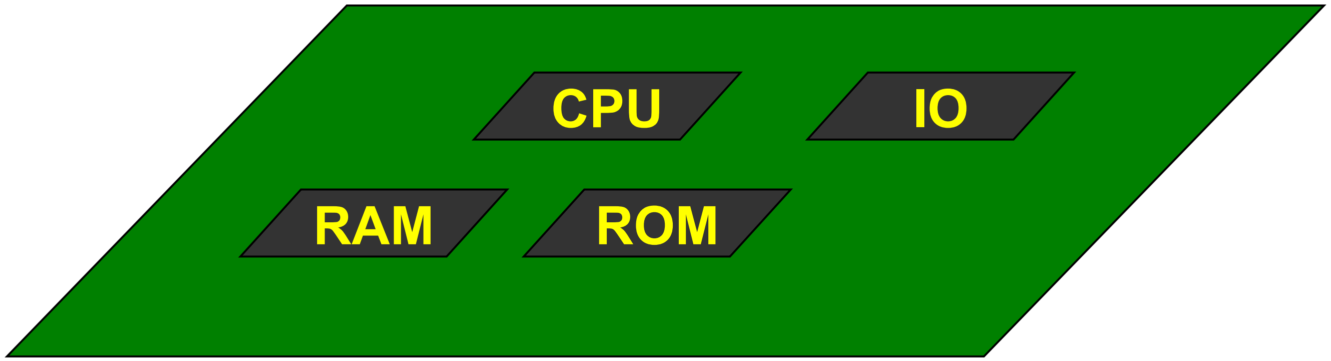


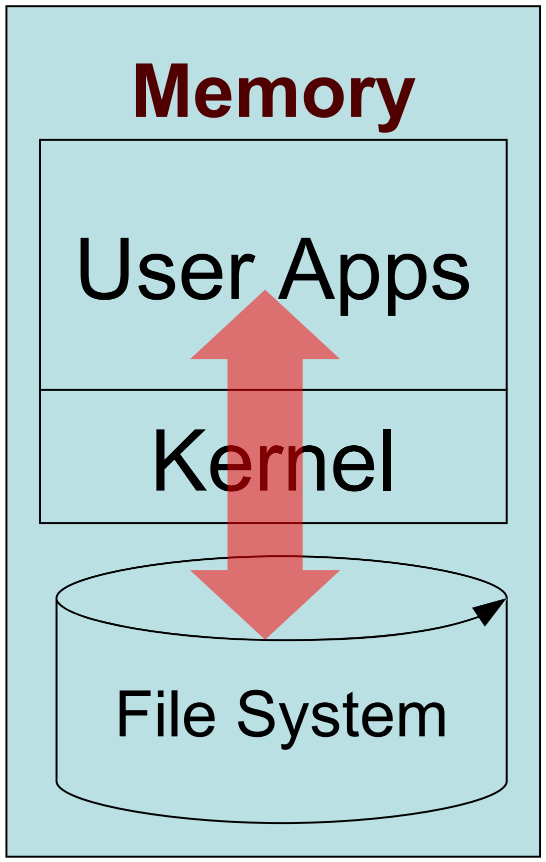
Linux on PC



#
#
#hello world :-(
#
PC
Keyboard
Disk File System
Memory
User Apps
Kernel
IO Devices
File System
File system on disk

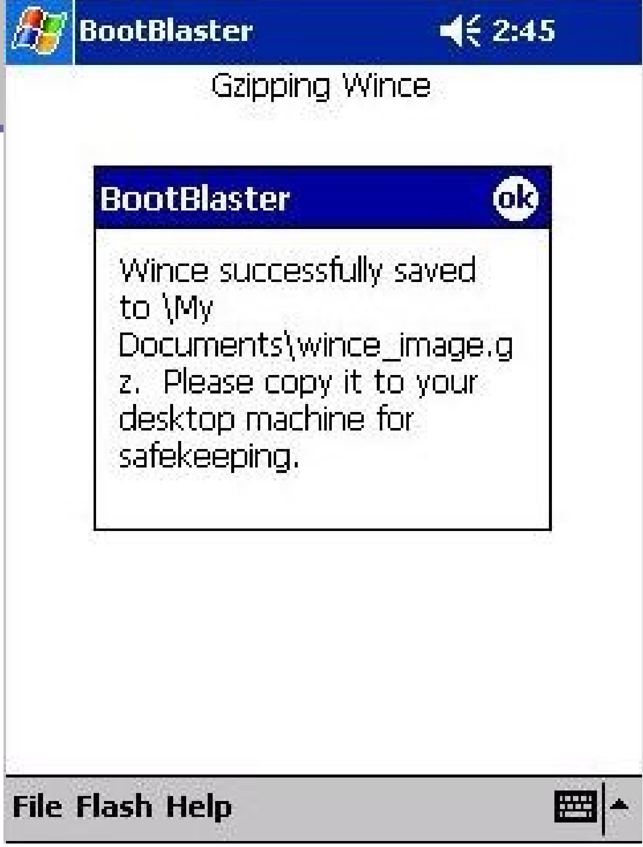
Linux on Embedded





File system on chip

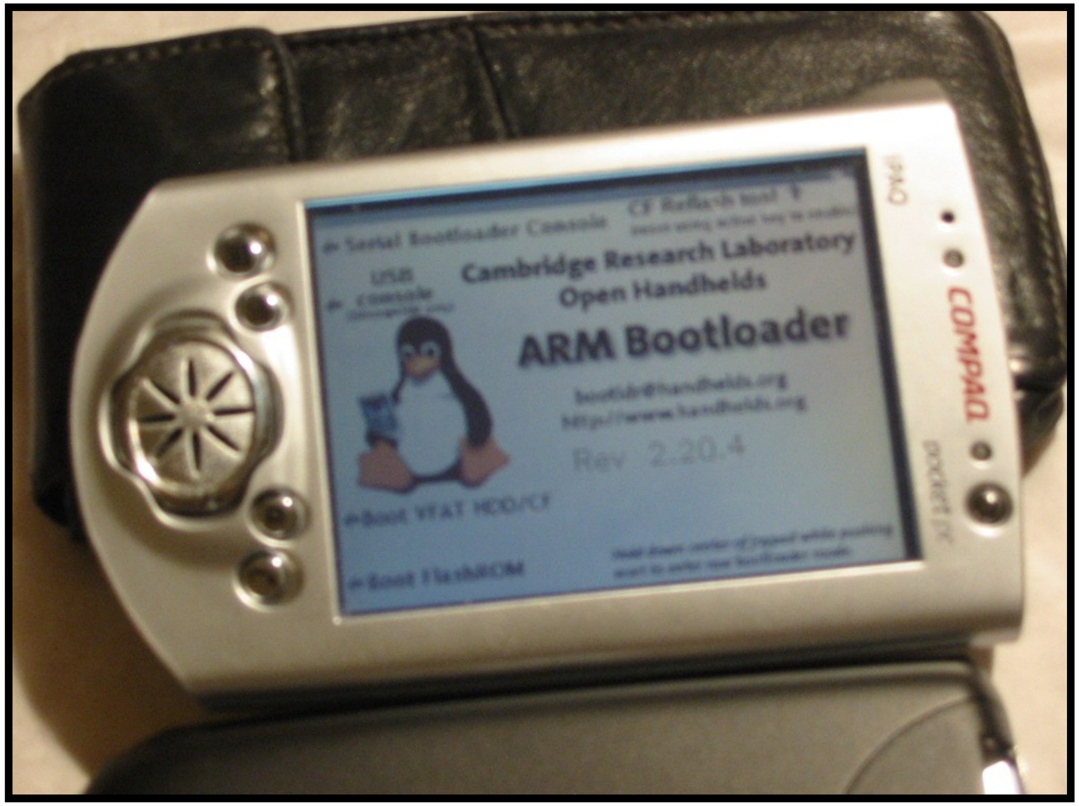
- Originally made by Compaq (now HP)
- iPAQ 3650 launched in 2000
- Runs Windows CE 3.0/PocketPC
- bootldr from handhelds.org started.



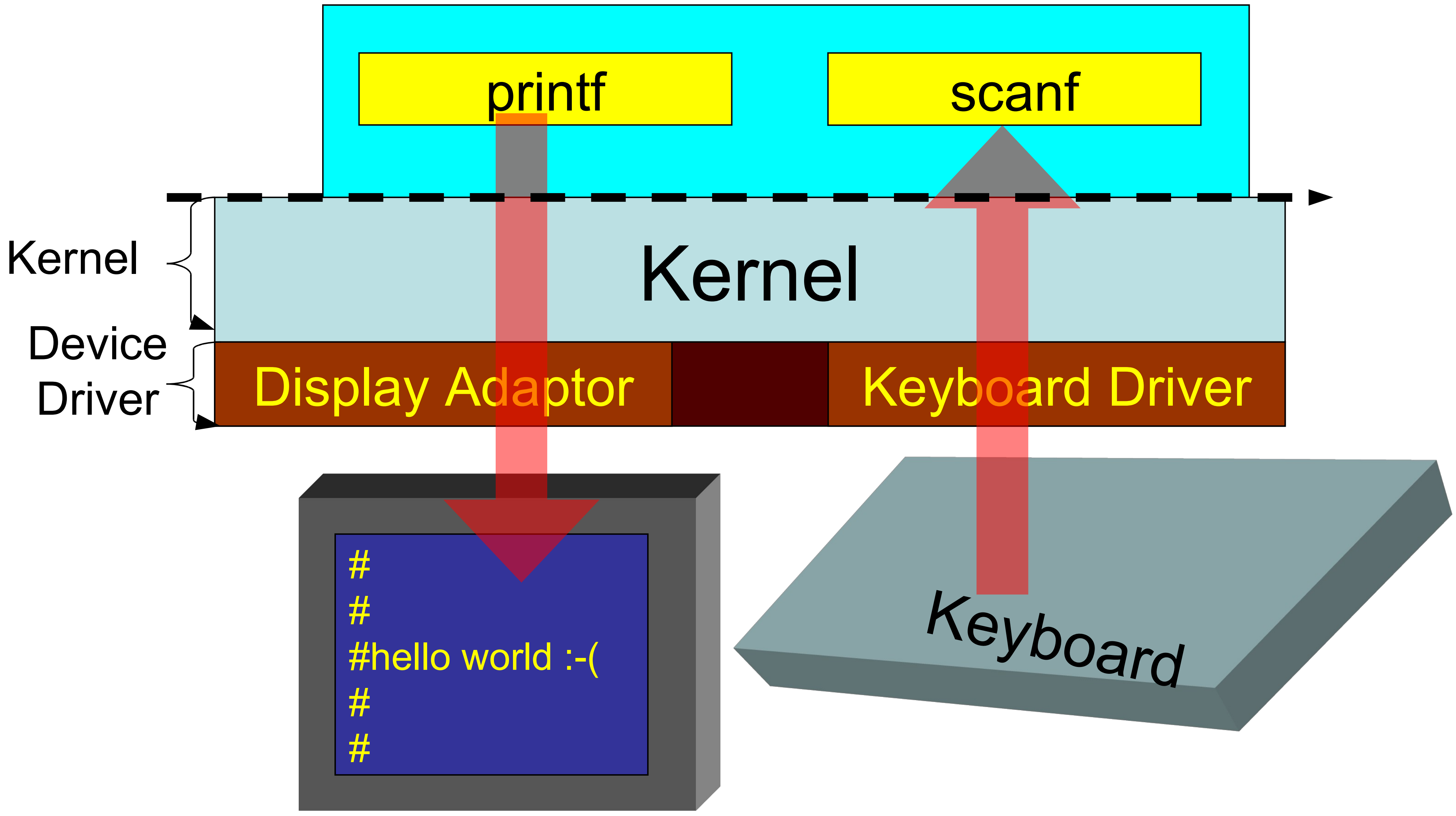
Gzipping Wince

BootBlaster

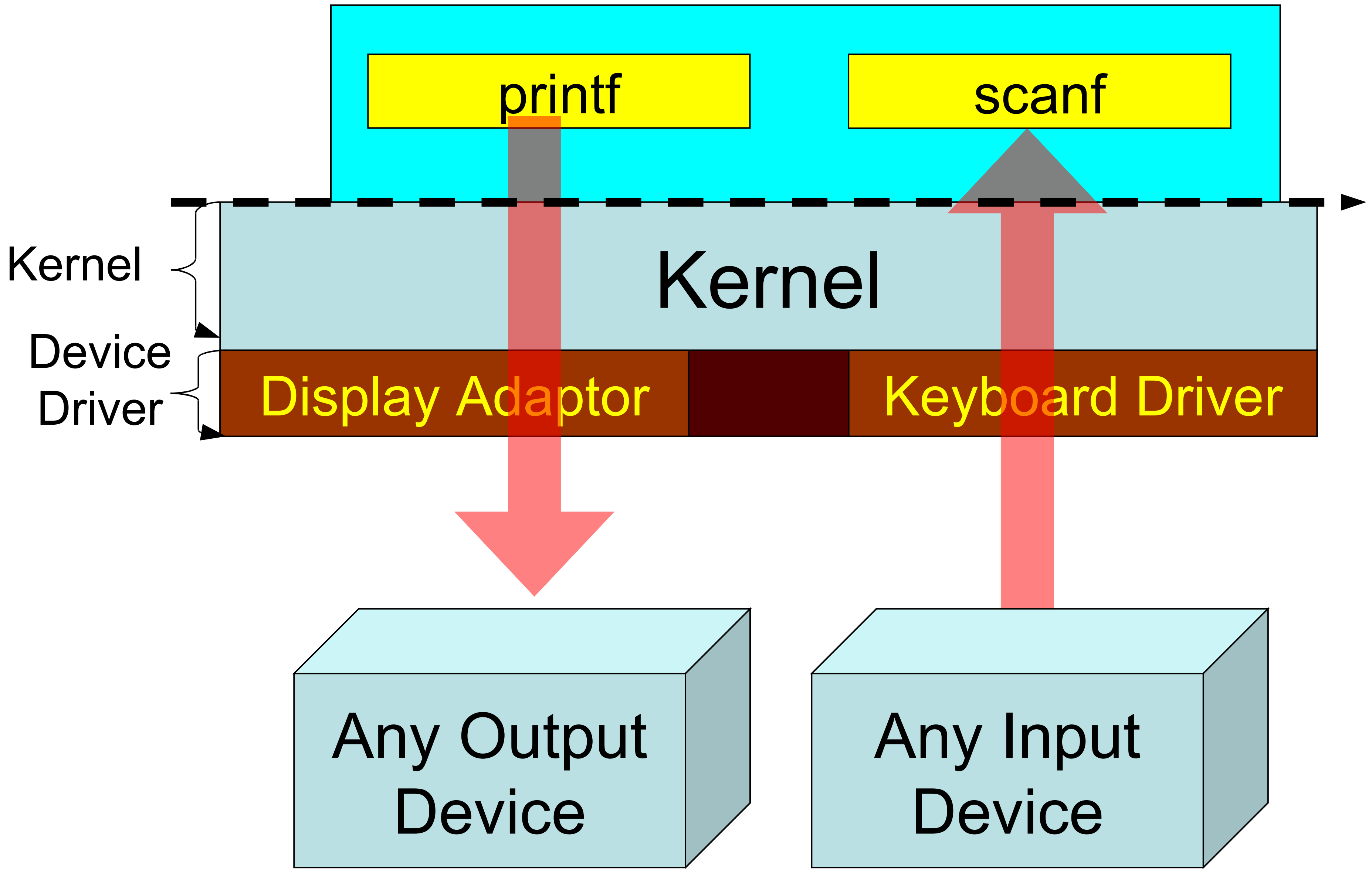
Wince successfully saved to \My Documents\wince_image.gz. Please copy it to your desktop machine for safekeeping.



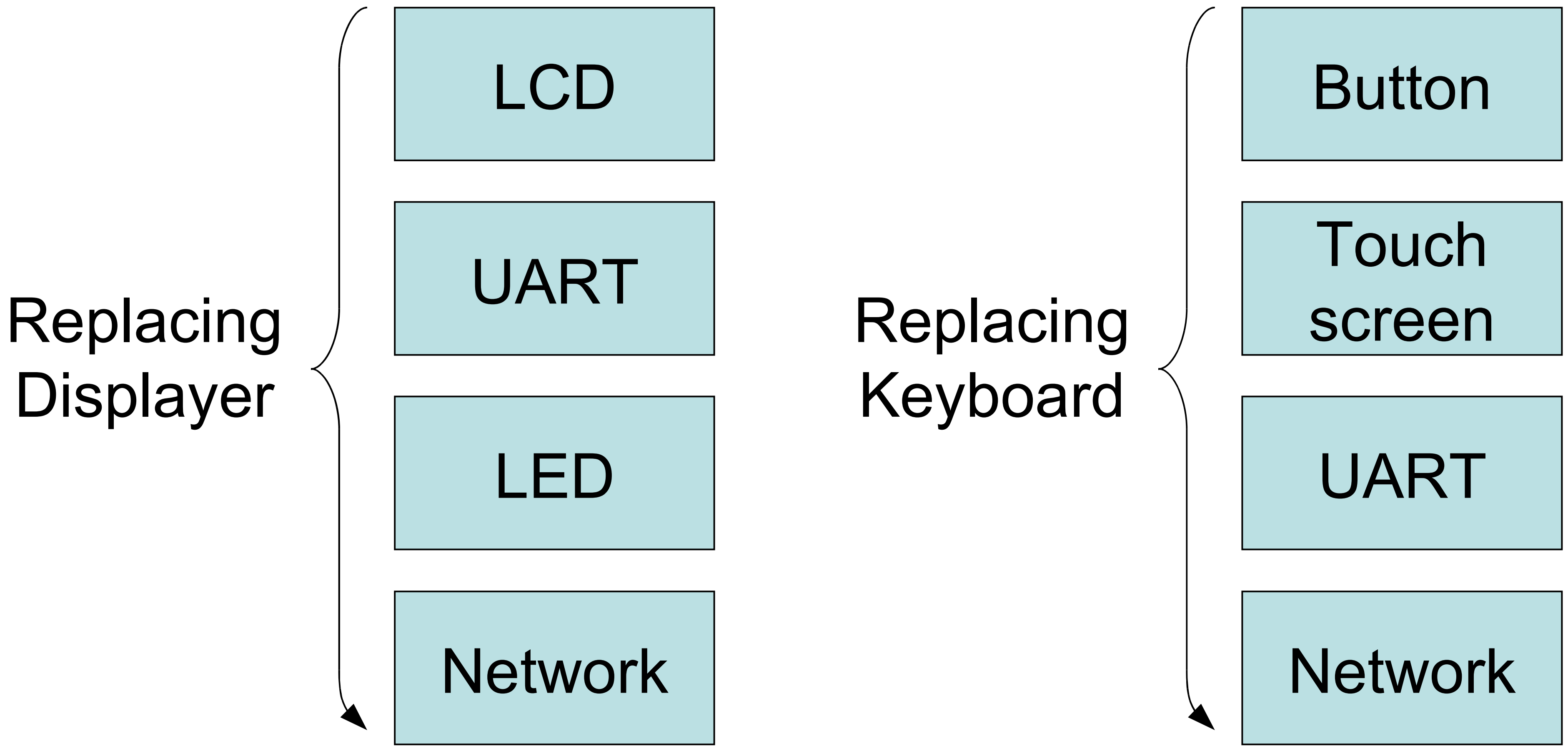
User Interaction on PC



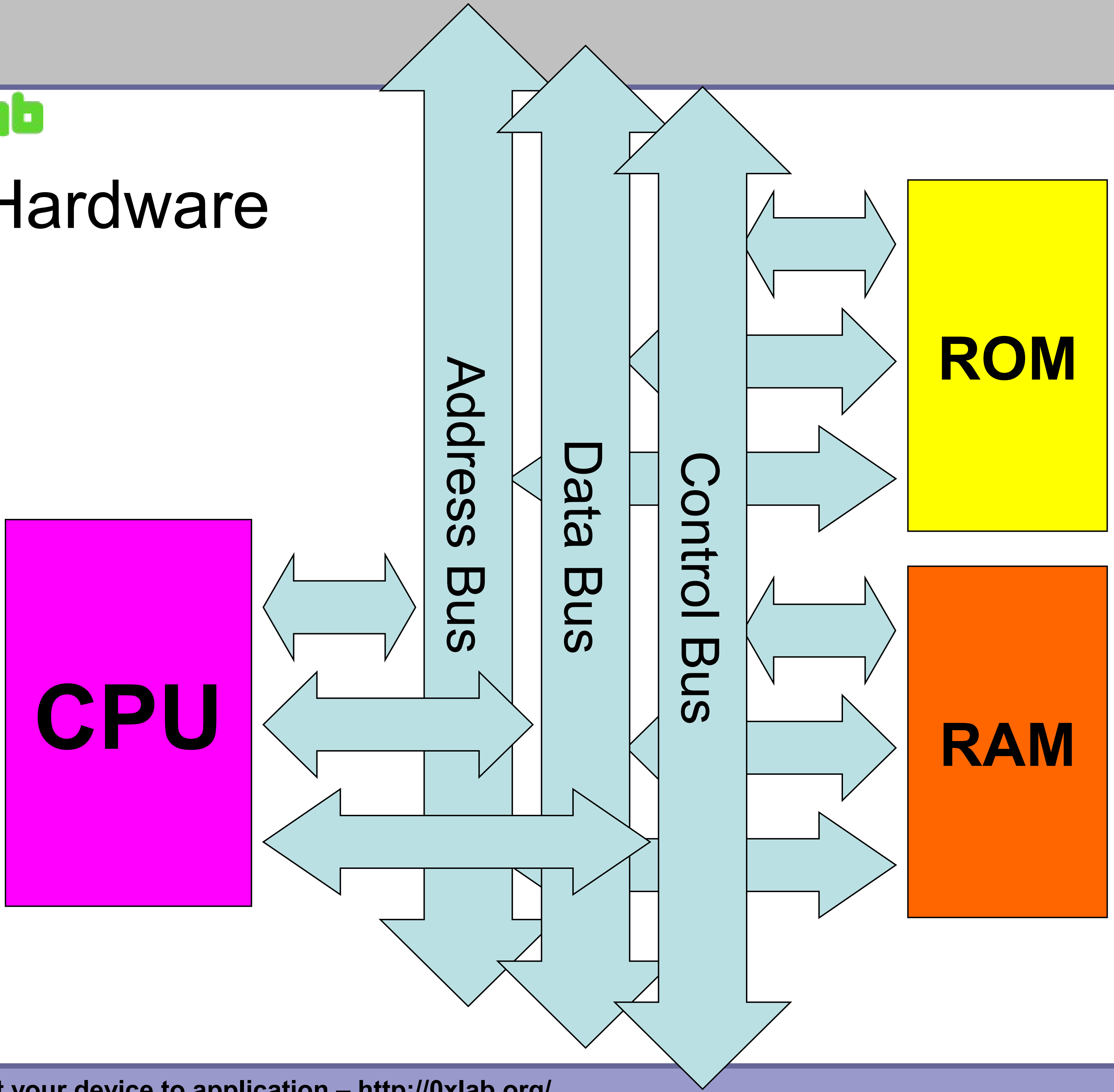
User Interaction on Embedded

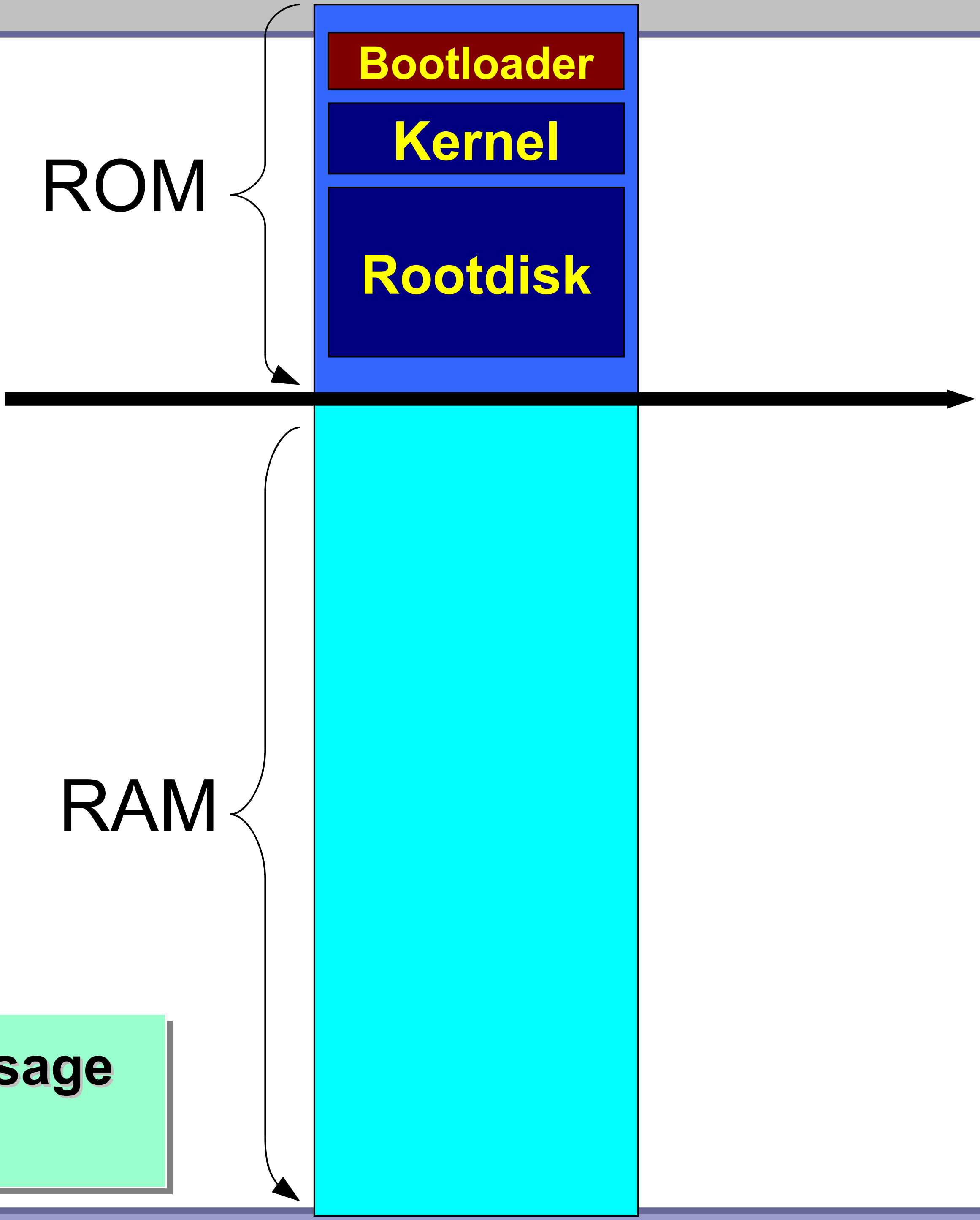


Embedded with Hardware

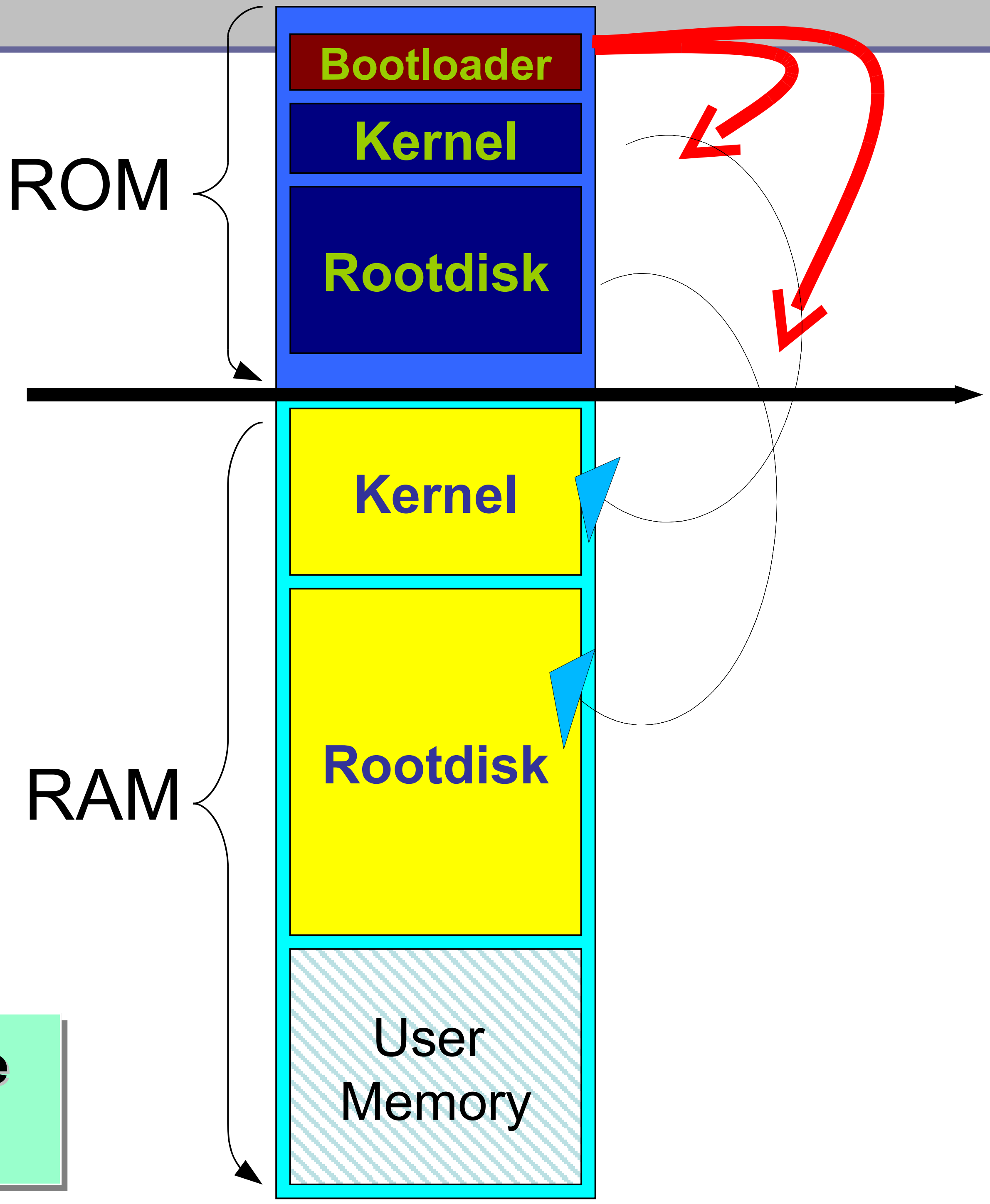


Hardware

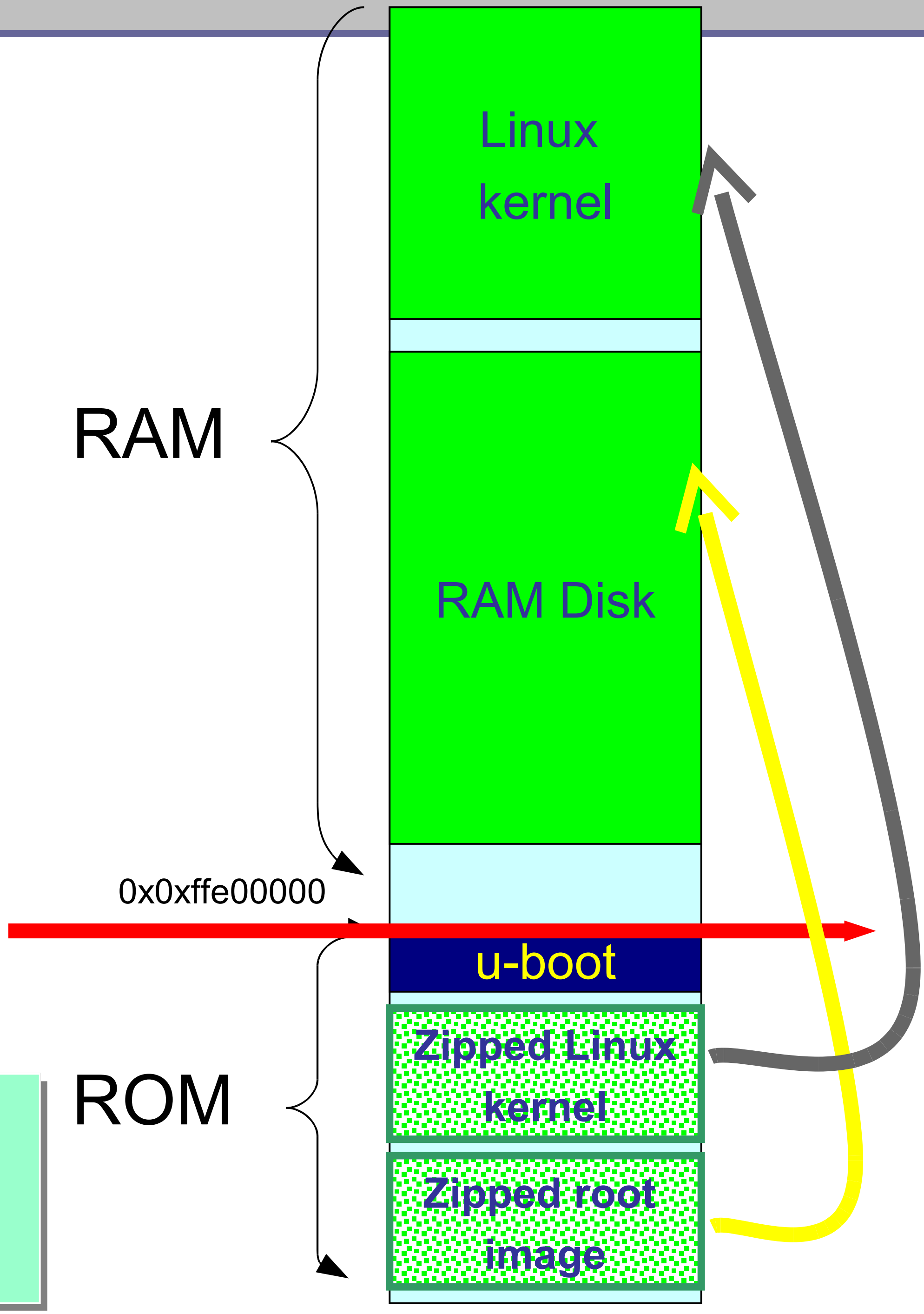




System Memory Usage (static)

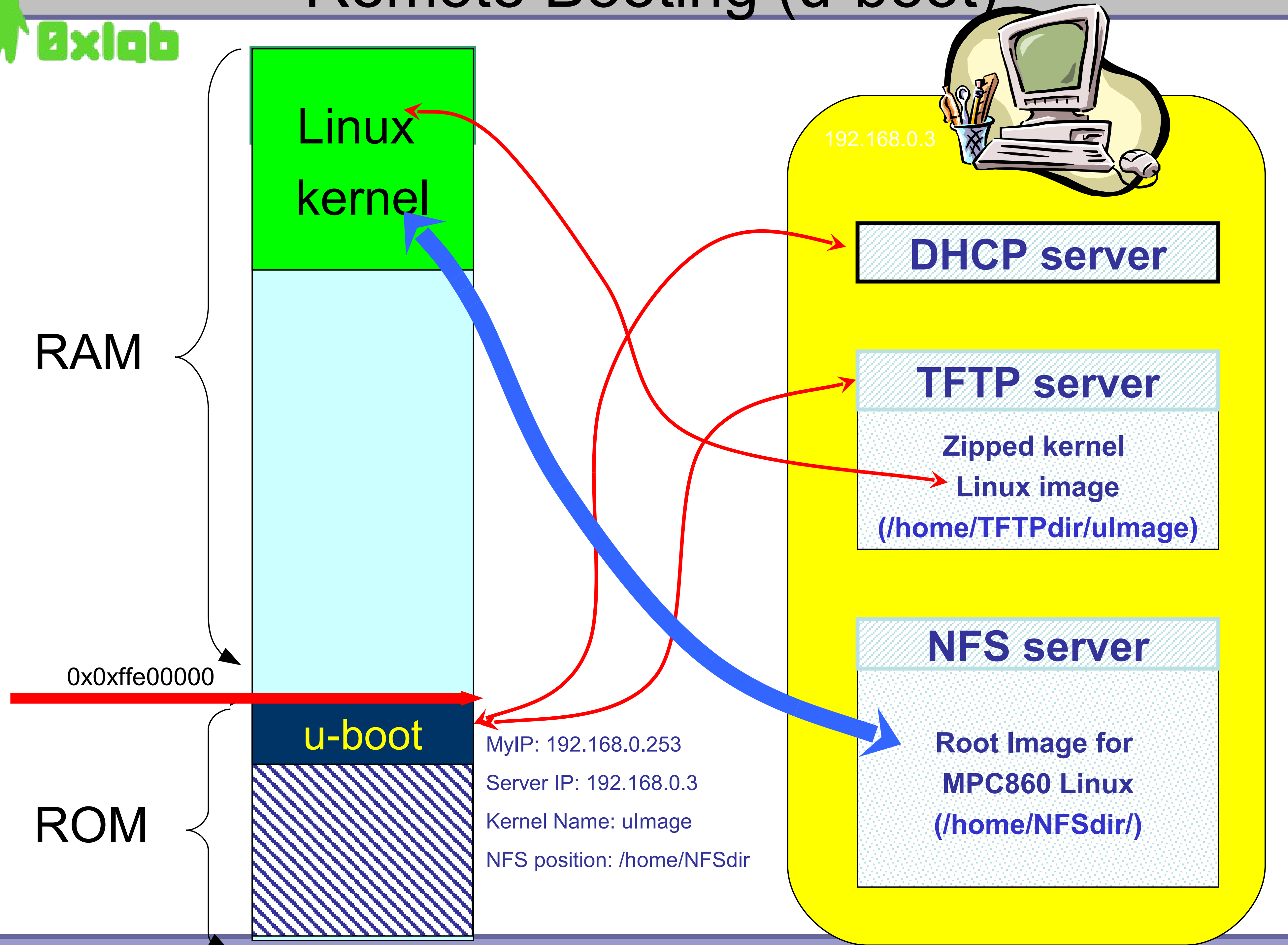


System Memory Usage (running)



Local Booting
(u-boot, famous in embedded world)

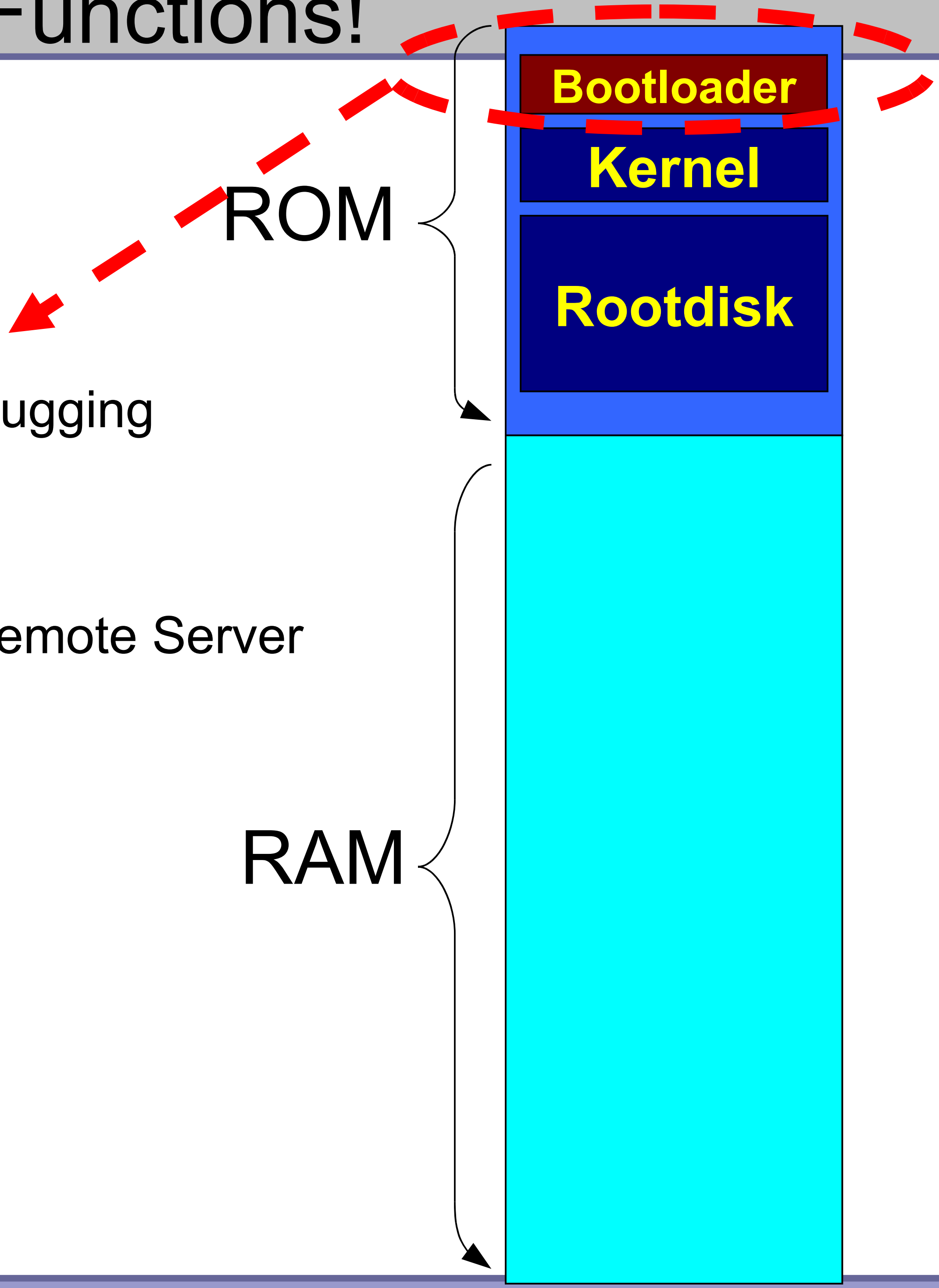
Remote Booting (u-boot)



MyIP: 192.168.0.253
Server IP: 192.168.0.3
Kernel Name: uImage
NFS position: /home/NFSdir

Functions!

- Initialize Board
  - Also enable basic debugging function
- Loading Kernel
  - From ROM or From Remote Server
- Load Root Image
- Starting Kernel

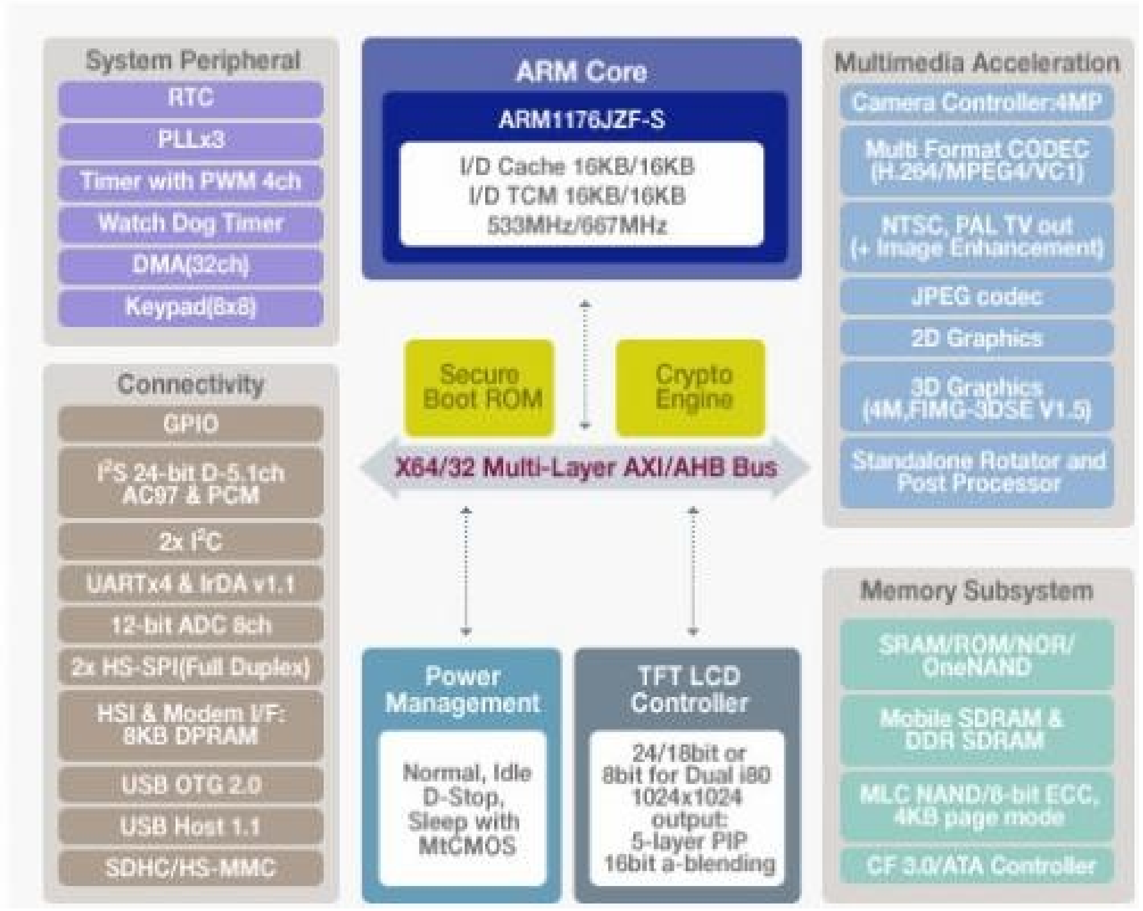


Boot Loader is No More Simple Now

u-boot is proud of rich feature/functions:

- SD/MMC card + FAT file system
- Autoboot
- OS loading commands
- Upgrade itself
- Networking
- Environment variables
- NOR/NAND Flash
- Self-testing
- ...

Block Diagram

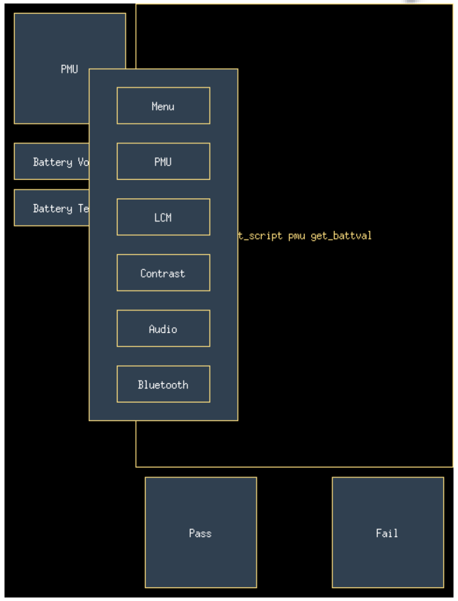


We even hacked more in u-boot

Openmoko hackers improved u-boot as the following:

http://wiki.openmoko.org/wiki/U-boot

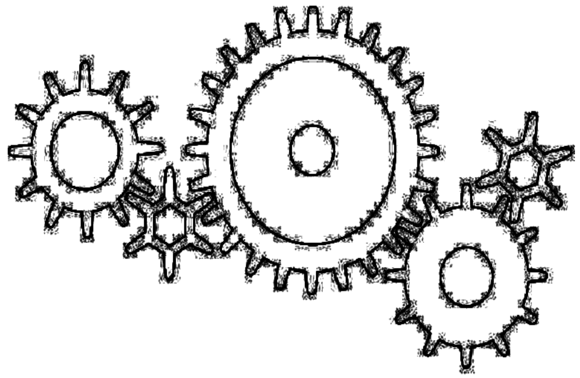
- Graphical menu for multiple boot (Yes, UI is usable.)
- usbtty (Yes, you can make phone call via boot loader.)
- Changeable boot screen (Yes, it comes with personal style.)
- Control GSM modem directly (Yes, you can even digg firmware.)
- Integrate tiny window system (Yes, we were crazy.)
- … (too many) ...



Hacking is Fun, but...

Both grub and u-boot are relatively complex and hard to improve/debug.

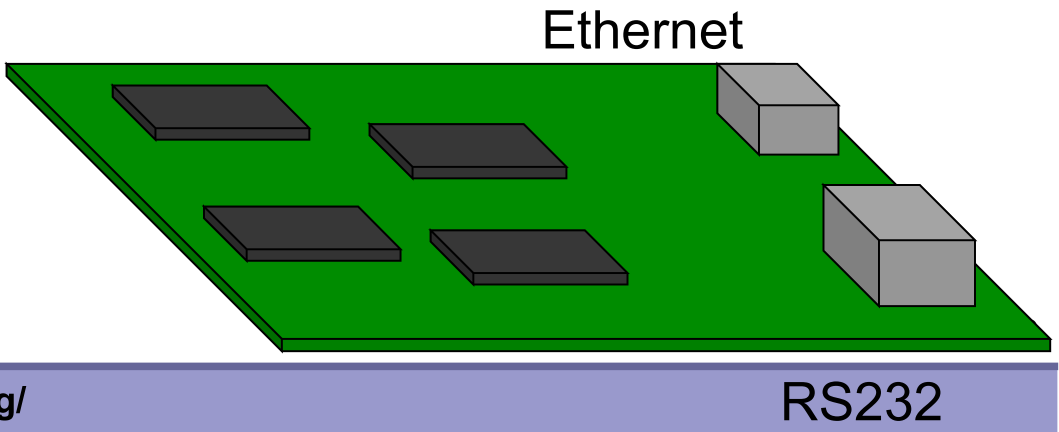
- Grub runs in 80386 protected mode in order to perform multiboot. (We hate software!)
- Grub can access several file systems, which implies huge code implementation.
- u-boot “stole” some device drivers (mainly NIC) from Linux kernel, but the performance is poor and somehow buggy.
- u-boot is single threaded (no tasks), and you have to handle interrupts very carefully.
- Painful programming always.



Practical Reasons to Modify boot loader

For consumer electronics, we modify boot loaders in embedded devices because...

- Make sure Linux can be safely booted under the proper DC battery voltage.
- Couple with charger before Linux is able to work.
- Rescue mode, Engineering mode, Field trial mode, …
- Firmware upgrade (including boot loader itself, Linux kernel, ramdisk, root file system, data, ...)
- Protect the firmware (encrypt system software information)
- Flexible remote booting



People tried to eliminate the efforts.

Concepts: Linux itself can be used as Boot Loader.

- (2003) Joshua Wise, LAB (Linux As Bootloader) a.k.a. “bootldr-ng” for iPAQ.
- Linux has many commonly used features that would be beneficial to a bootloader, however difficult to port properly:
  - USB, MMC/SD, Filesystem, ...
- (2005) Werner Almesberger, "kboot - A Boot Loader Based on Kexec", Linux-Kongress
- (2007) “HTTPFUSE PS3Linux: Internet boot framework using HTTP”

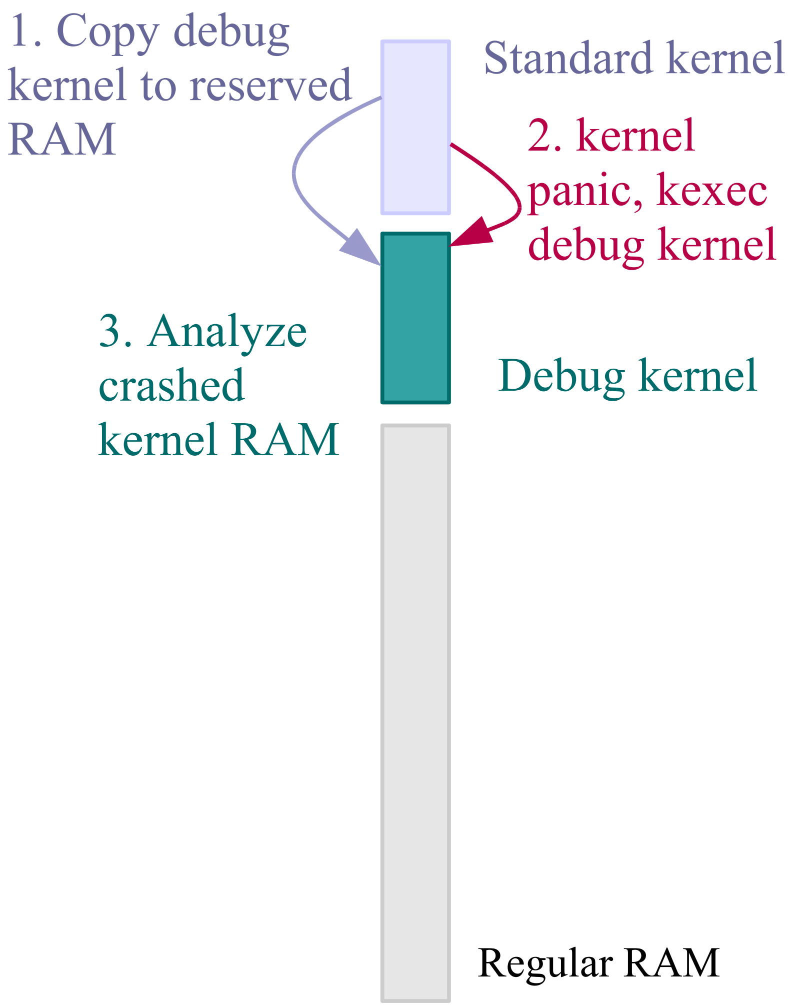
Qi leverages the above ideas to build up totally new one.

- Evolution of Boot Loader
  - The “function” of boot loaders
    - Our experience from Openmoko project
  - Applied in PC and Embedded Systems
- **Overview of Qi**
  - **Principal: KISS (Keep It Simple and Stupid)**
- Practical Qi
  - Real examples following the idea behind Qi
- Future Perspectives

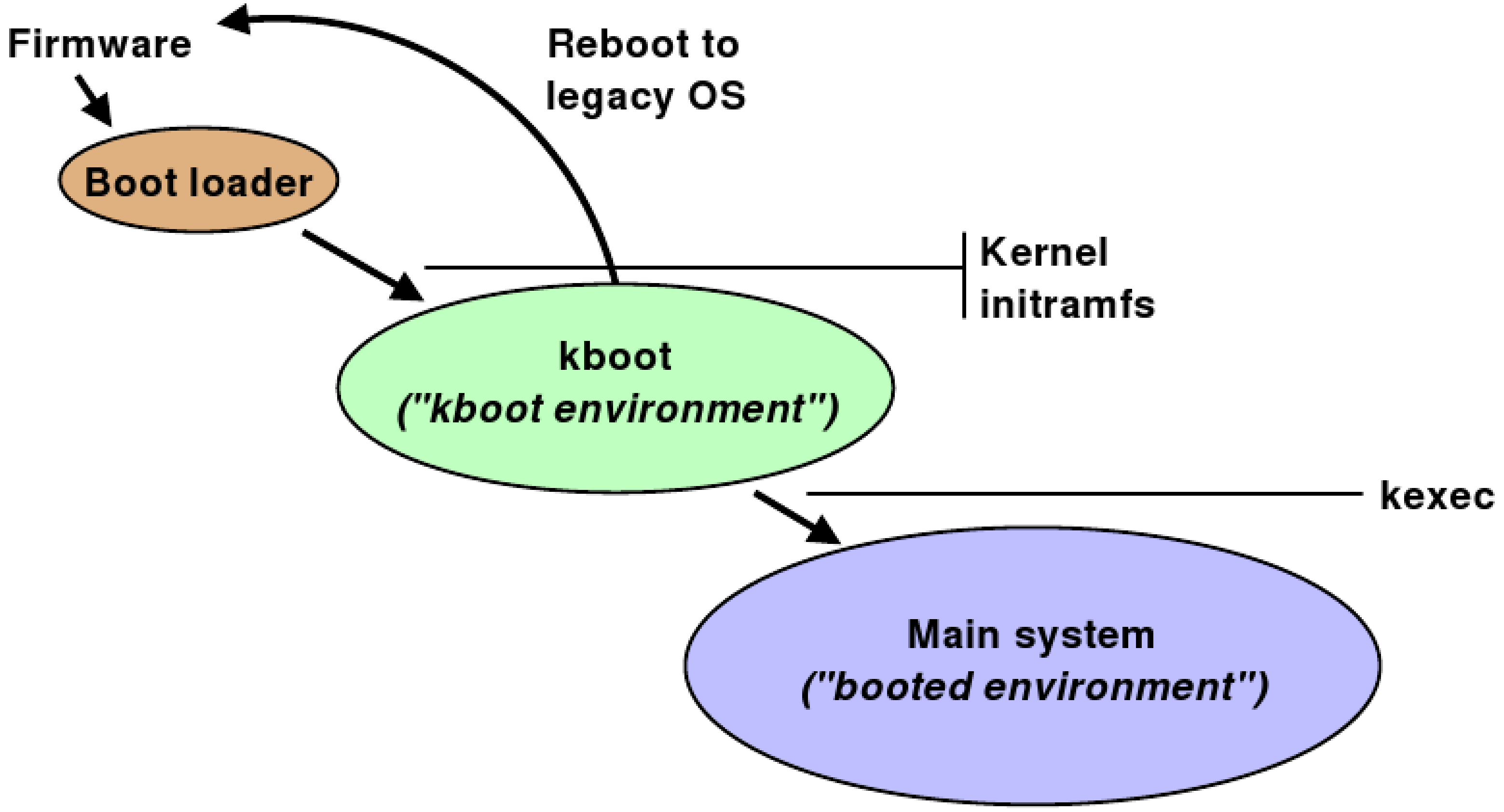
Prerequisite

Before introducing Qi, let us touch the background knowledge.

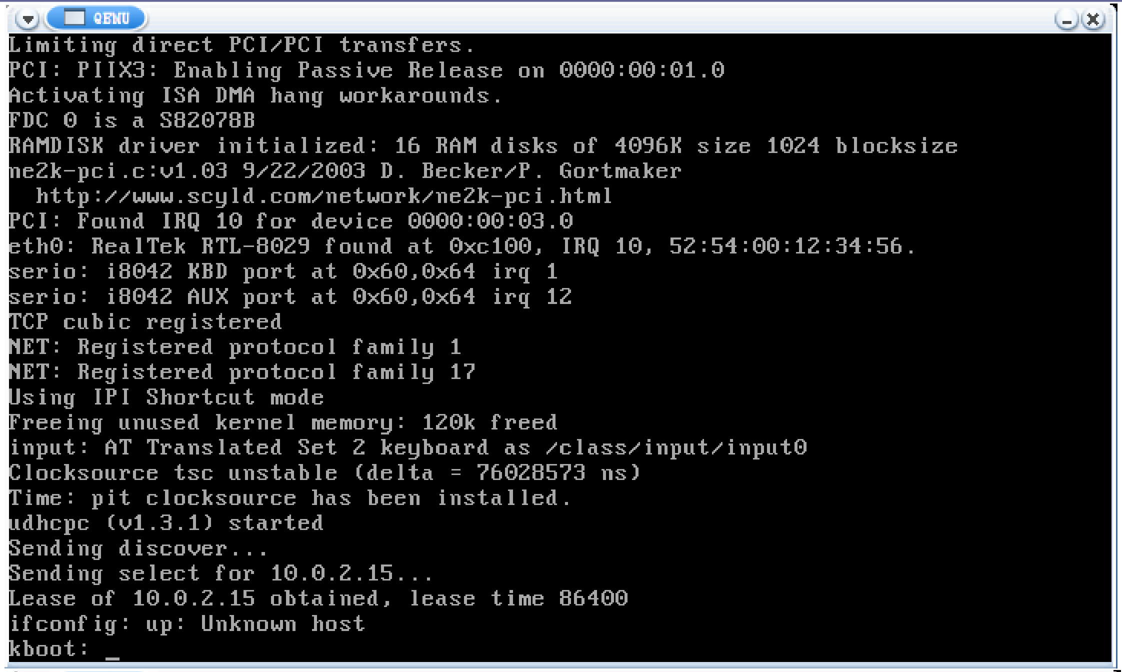
- kexec system call in Linux Kernel 2.6
  - Linux kernel warm-restart
  - kexec makes it possible to call a new kernel, without rebooting and going through the BIOS / firmware.
- kboot
  - a boot loader based on kexec
  - Basic idea: Without working up a sweat
  - Extended project: PS3-Linux


1. Copy debug kernel to reserved RAM
Standard kernel
2. kernel panic, kexec debug kernel
3. Analyze crashed kernel RAM
Debug kernel
Regular RAM

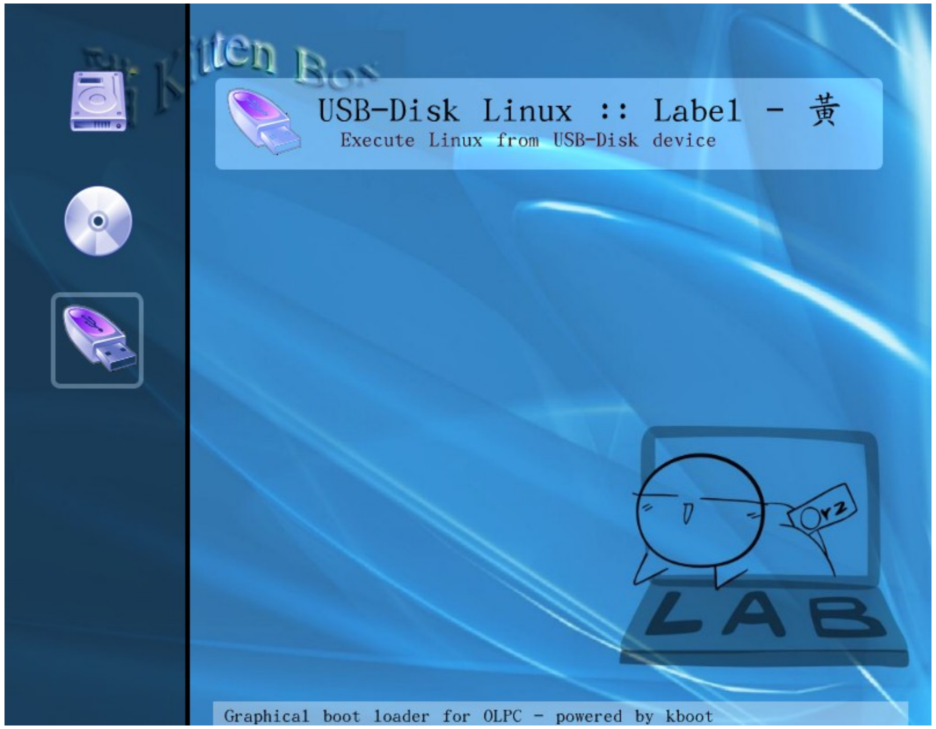
kboot in Action

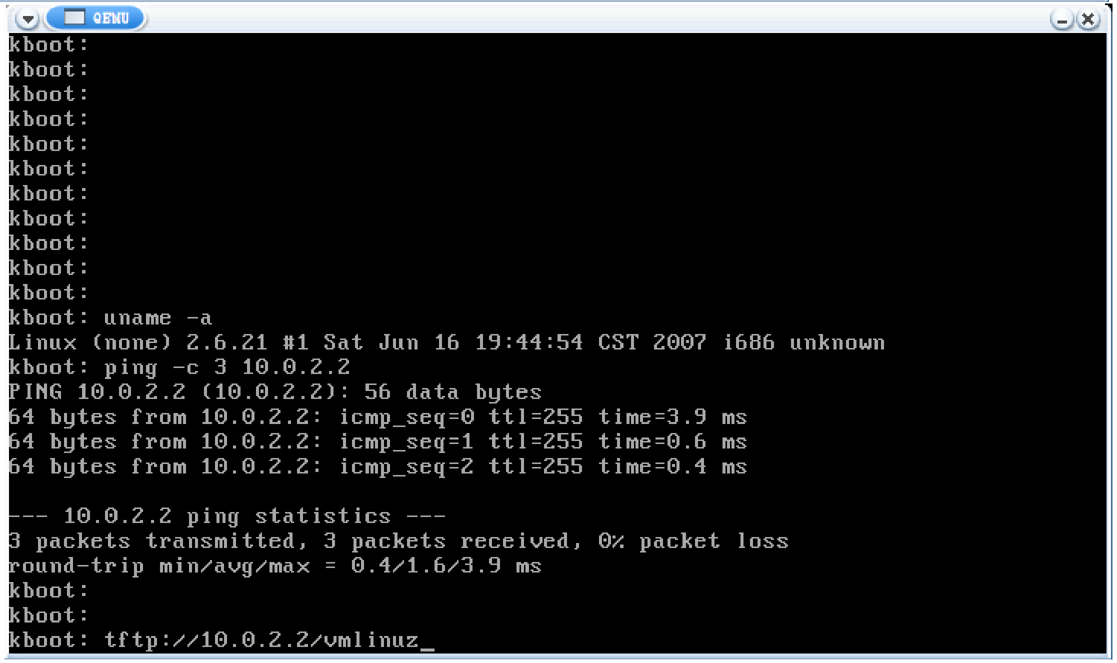


With kboot, it s fairly simple to construct Graphical and Fancy Boot Loader.



```
Limiting direct PCI/PCI transfers.
PCI: PIIX3: Enabling Passive Release on 0000:00:01.0
Activating ISA DMA hang workarounds.
FDC 0 is a S82078B
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
ne2k-pci.c:v1.03 9/22/2003 D. Becker/P. Gortmaker
  http://www.scyld.com/network/ne2k-pci.html
PCI: Found IRQ 10 for device 0000:00:03.0
eth0: RealTek RTL-8029 found at 0xc100, IRQ 10, 52:54:00:12:34:56.
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX port at 0x60,0x64 irq 12
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI Shortcut mode
Freeing unused kernel memory: 120k freed
input: AT Translated Set 2 keyboard as /class/input/input0
Clocksource tsc unstable (delta = 76028573 ns)
Time: pit clocksource has been installed.
udhcpc (v1.3.1) started
Sending discover...
Sending select for 10.0.2.15...
Lease of 10.0.2.15 obtained, lease time 86400
ifconfig: up: Unknown host
kboot: _
```


USB-Disk Linux :: Label - 黃
Execute Linux from USB-Disk device
Graphical boot loader for OLPC - powered by kboot
LAB



```
kboot:
kboot:
kboot:
kboot:
kboot:
kboot:
kboot:
kboot:
kboot:
kboot:
kboot:
kboot: uname -a
Linux (none) 2.6.21 #1 Sat Jun 16 19:44:54 CST 2007 i686 unknown
kboot: ping -c 3 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: icmp_seq=0 ttl=255 time=3.9 ms
64 bytes from 10.0.2.2: icmp_seq=1 ttl=255 time=0.6 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=255 time=0.4 ms

--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.4/1.6/3.9 ms
kboot:
kboot:
kboot: tftp://10.0.2.2/vmlinuz_
```

Qi bootloader

- Openmoko planned to implement kboot and adopt it on GTA0x series. (Hint: Openmoko system architect, Werner Almesberger is the author of lilo and kboot as well.)
- But kboot is a second stage bootloader, we still need a legacy bootloader.
- Why it called “Qi”?

  Umm "Qi" - not really a flower but if I remember my Chinese mythology its the breath that brings things to life ?

  – Alan Cox
- Similar idea as kboot
  - Not reinvent the wheel anymore, avoiding cutting and pasting drivers into u-boot

Qi in a nutshell

- Licensed under GNU GPL.
- Current status
  - Openmoko does not maintain this project anymore
  - Andy Green (ex-Openmoko kernel maintainer) continues contributing to Qi.
    - http://git.warmcat.com/cgi-bin/cgit/qi/log/?h=txtr
- Qi supports few platforms at present.
  - Samsung S3C24xx/S3C6410, Freescale iMX31
- Very small footprint
  - ~28 kb size
- Fast boot time
  - Comparing to u-boot, it's reduced about **28%** boot time (time-to-desktop)

Qi in a nutshell

Test report from SHR distribution running on FreeRunner/GTA02

Booting SHR image with uBoot:

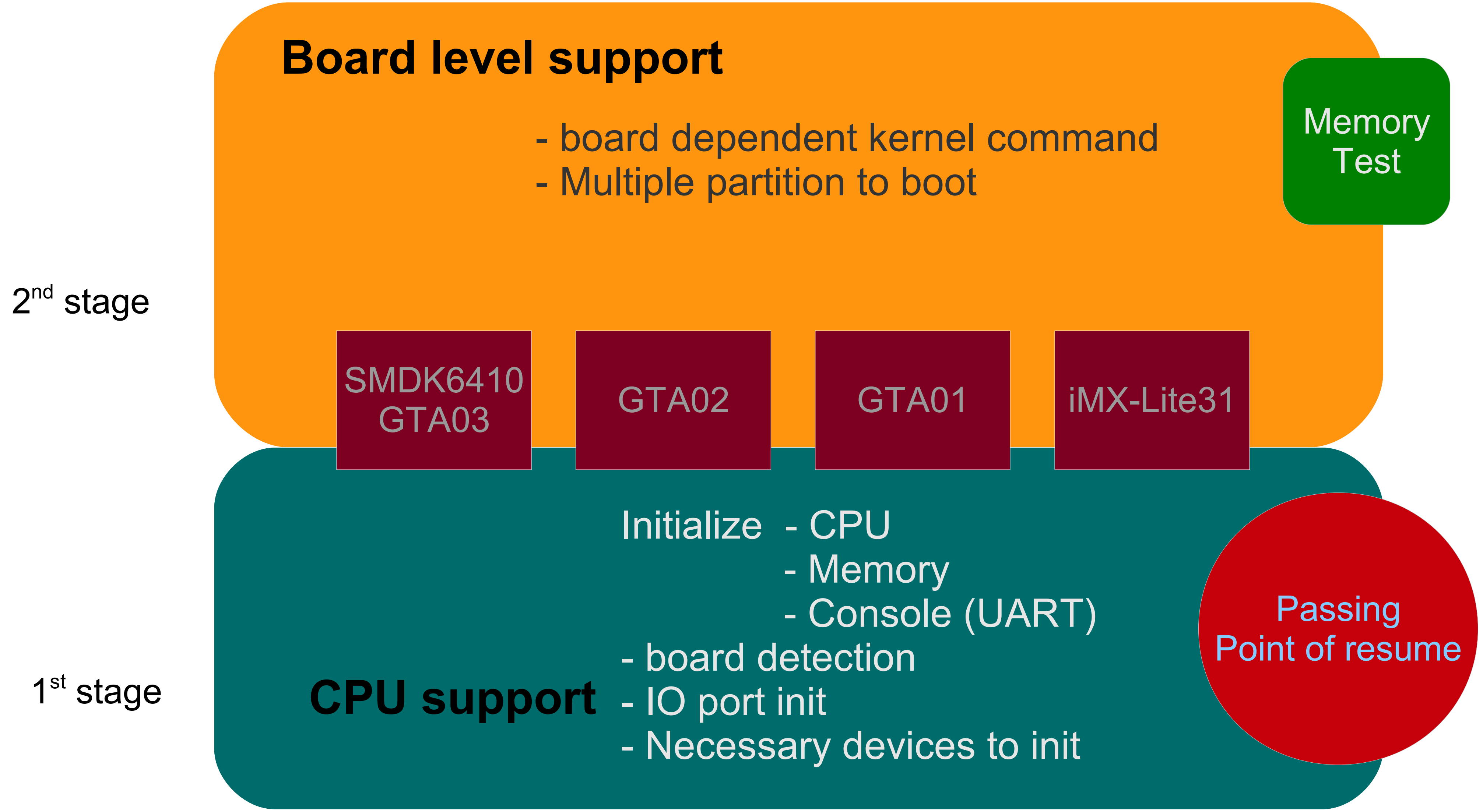
- 0:00 power button held down
- 0:07 splash screen appears
- 0:15 drops to console showing kernel messages scrolling by for ~1 minute
- 1:18 Openmoko 'please wait' splash
- 1:31 desktop animated splash
- 2:38 finished booting

Booting identical setup with **Qi** flashed over uBoot:

- 0:00 power button held down
- 0:06 backlit black
- 0:13 please wait booting... (only this text on console for next 38 seconds)
- 0:51 Angstrom console message (at the end of kernel output with uBoot, but ONLY text display to appear throughout this stage with Qi)
- 0:54 Openmoko 'please wait' splash
- 1:05 desktop animated splash
- 1:54 finished booting

Faster

Qi Architecture



KISS: Keep It Simple and Stupid

```
struct board_api {
    ........
    const struct board_variant
    const * (*get_board_variant)(void);
    int (*is_this_board)(void);
    void (*early_port_init)(void);
    void (*port_init)(void);
    void (*post_serial_init)(void);
    char * (*append_device_specific_cmdline)
                (char *);
    void (*putc)(char);
    ...
    struct kernel_source kernel_source[4];
};
```

```
const struct board_api *boards[] = {
    &board_api_om_3d7k,
    &board_api_smdk6410,
    NULL /* always last */
};
```

```
    this_board = boards[board];
     while (!flag && this_board)
         /* check if it is the right board... */
         if (this_board->is_this_board())
             flag = 1;
         else
             this_board = boards[board++];
```

Misc:
straightforward boot config

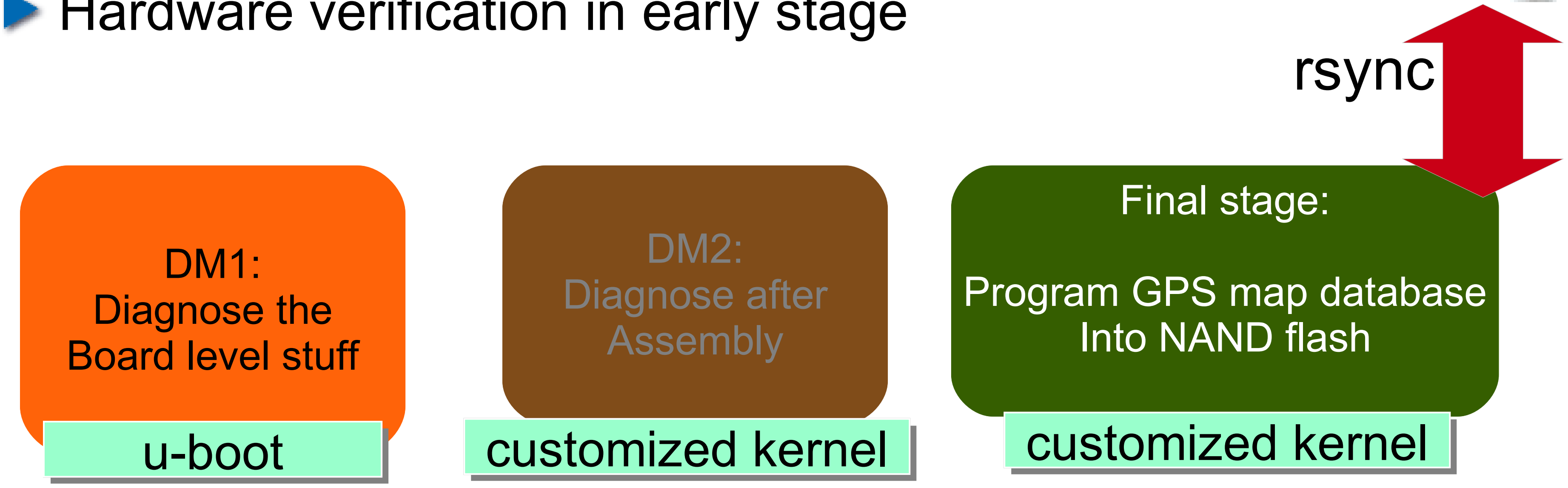
Limitations of Qi

- Required an approach to update boot loader if device could not support booting from SD
  - NOR boot, JTAG
- No boot menu support
- FAT partitions are ignored

- Evolution of Boot Loader
  - The “function” of boot loaders
    - Our experience from Openmoko project
  - Applied in PC and Embedded Systems
- Overview of Qi
  - Principal: KISS (Keep It Simple and Stupid)
- **Practical Qi**
  - **Real examples following the idea behind Qi**
- Future Perspectives

Case Study

- Mass production for HXD8 (hardware name for Dash Express) project at Openmoko.
  - There are two systems to diagnose hardware.
    - u-boot
    - customized kernel for production
  - Duplicated test commands
  - Maintenance is a nightmare
- Hardware verification in early stage


rsync
DM1: Diagnose the Board level stuff
u-boot
DM2: Diagnose after Assembly
customized kernel
Final stage: Program GPS map database Into NAND flash
customized kernel

Future Perspectives

- For mobile and embedded devices, boot sequence is changing.
- Qi is not only a boot loader project but a new approach to comply with the diverse requirements of various consumer electric devices/products.
- What consumers expect
  - Faster boot time
  - Personalization
  - Secure data (client security solution)
- What manufacturers expect
  - Firmware protection
  - Robust system software

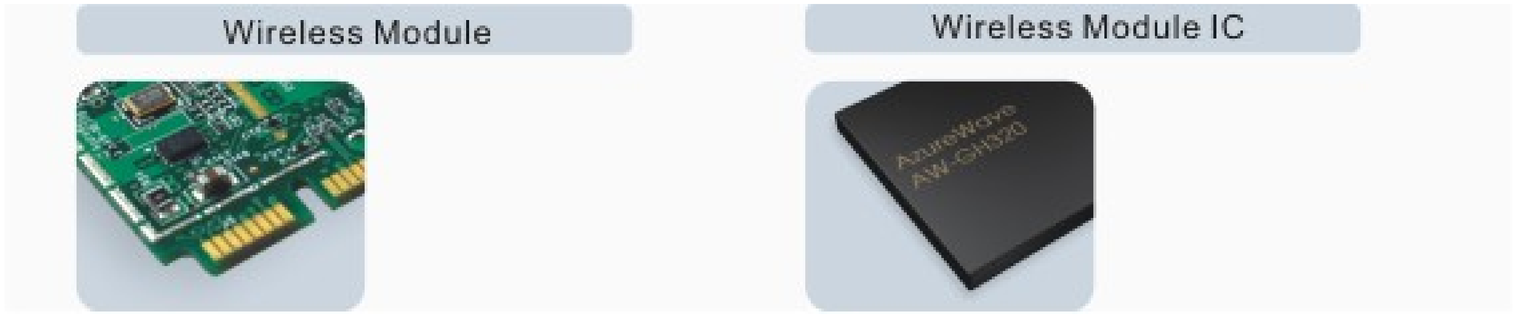
Reference

- U-boot – the universal boot loader
  - http://www.denx.de/wiki/U-Boot
- Qi boot loader wiki
  - http://wiki.openmoko.org/wiki/Qi
- GNU GRUB – multiboot boot loader
  - http://www.gnu.org/software/grub/
- Samsung S3C6410 Mobile Processor
  - http://www.samsung.com/global/business/semiconductor/support/brochures/downloads/systemlsi/s3c6410_datasheet_200804.pdf
- Openmoko wiki – http://wiki.openmoko.org/
- Dash Express – http://dash.net/

Thanks!

We appreciate the great sponsorship from AzureWave Technologies, Inc. (http://www.azurewave.com/)

- Providing WiFi module
- Providing ARM development board



About 0xlab

http://0xlab.org

0xlab

Our focus is to strengthen the connection between hardware device manufacturers and open source software communities and to become the integrated solution provider bringing more devices powered by open source for daily use.

0xlab is founded by a group of engineers enthusiastic in modern technologies and open source software development.

0xlab is an open organization. We appreciate anyone's participation in our projects and planning to contribute to the free software community.

We support open source.

- We open up our development process and results as much as possible.
- We will always be an open organization. Everyone can join for discussion, testing and development.