

CE5045

Embedded System Design

Embedded Software Architecture

<https://github.com/tychen-NCU/EMBS-NCU>

Instructor: Dr. Chen, Tseng-Yi

Computer Science & Information Engineering

Outline

- How to Boot an Embedded System (ES)
 - ✓ Grub for x86 Architecture
 - ✓ U-boot for ARM Architecture
 - ✓ How to Implement a Bootloader in an Embedded System.
- Microkernel for Embedded System
 - ✓ What is Process?
 - ✓ The Process Concept in an Embedded System
 - ✓ The Practical Knowledge of Process Management

Kernel space
(Basic IPC, process
scheduler, etc)

Bootloader
(U-Boot, LILO, etc)

Hardware

Outline

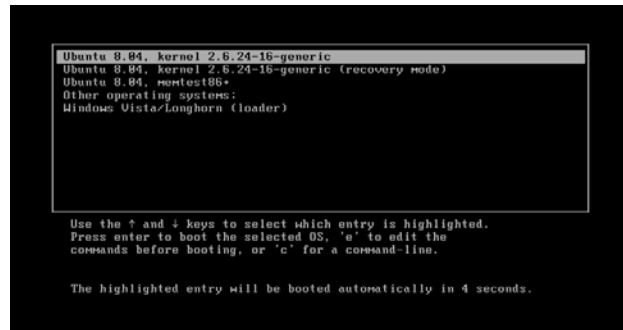
- How to Boot an Embedded System (ES)
 - ✓ Grub for x86 Architecture
 - ✓ U-boot for ARM Architecture
 - ✓ How to Implement a Bootloader in an Embedded System.
- Microkernel for Embedded System
 - ✓ What is Process?
 - ✓ The Process Concept in an Embedded System
 - ✓ The Practical Knowledge of Process Management

What is GRUB?

➤ Before Grub

✓ What is boot sector?

- A boot sector is generally the first sector of the hard drive that is accessed when the computer is turned on.
- Master boot record (MBR) or GUID partition table (GPT)



➤ So Grub ...

- ✓ Is a piece of software that exists in the MBR or GPT
- ✓ Allows a user to opt between multiple Operating systems that are installed on one or more drives existing on the computer

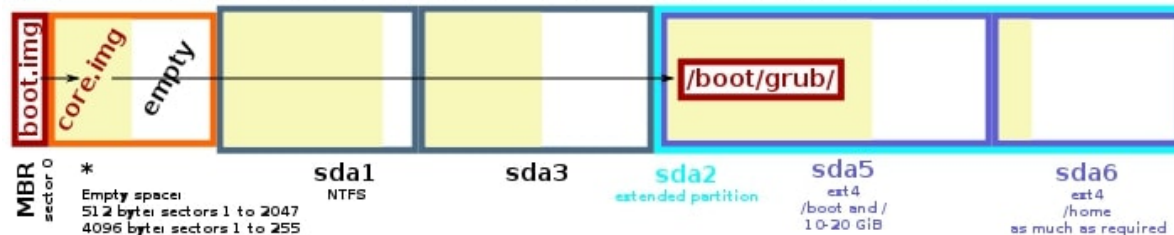
How to Boot with Grub

- Depending upon the boot sector that is either the master boot record or the GUID partition table the physical allocation of the GRUB change.

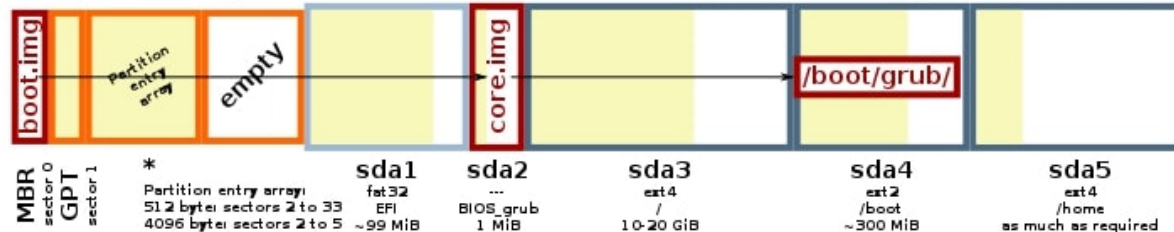
GNU GRUB 2

Locations of *boot.img*, *core.img* and the */boot/grub/* directory

Example 1: An MBR-partitioned hard disk with sector size of 512 or 4096 bytes



Example 2: A GPT-partitioned hard disk with sector size of 512 or 4096 bytes



The Image Files for Grub

➤ boot.img

- ✓ Its size is 446 bytes
- ✓ Is written to the MBR (sector 0)
- ✓ Contains utilities, Operating system, kernel files, diagnostics and various other drivers for the hardware to initiate

➤ core.img

- ✓ Is written to the empty sectors between the MBR and the first partition
- ✓ Default RAW disk image loaded on the hard disk by the manufacturer

➤ File system

- ✓ Takes care of the addressing of the data onto a physical sector. Ex: NTFS, FAT, Ex-FAT, HFS etc.

The Functionality of GRUB (I/II)

➤ Stage 1

- ✓ Find the boot.img in Master Boot Record (MBR) or GUID partition table (GPT)
- ✓ At this stage GRUB directs the next stage using an address of the kernel files of various Operating systems

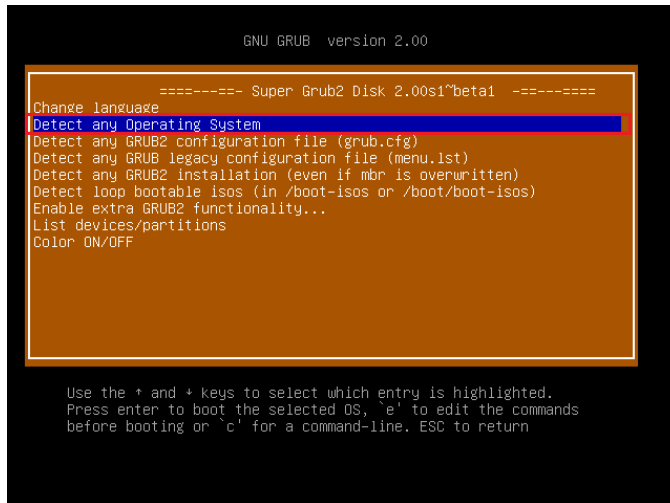
➤ Stage 1.5

- ✓ The core.img will load the file needed for configuration with various other modules like file system drivers acquired from boot.img

The Functionality of GRUB (II/II)

➤ Stage 2

- ✓ In this final stage a Text-based User Interface is displayed which will enable the user to select between the Operating systems.
- ✓ You can specify a default Operating system to load after a user define timeout.



```
GNU GRUB version 2.00

----- Super Grub2 Disk 2.00s1~beta1 -----
Change language
Detect any Operating System
Detect any GRUB2 configuration file (grub.cfg)
Detect any GRUB2 legacy configuration file (menu.lst)
Detect any GRUB2 installation (even if mbr is overwritten)
Detect loop bootable isos (in /boot-isos or /boot/boot-isos)
Enable extra GRUB2 functionality...
List devices/partitions
Color ON/OFF

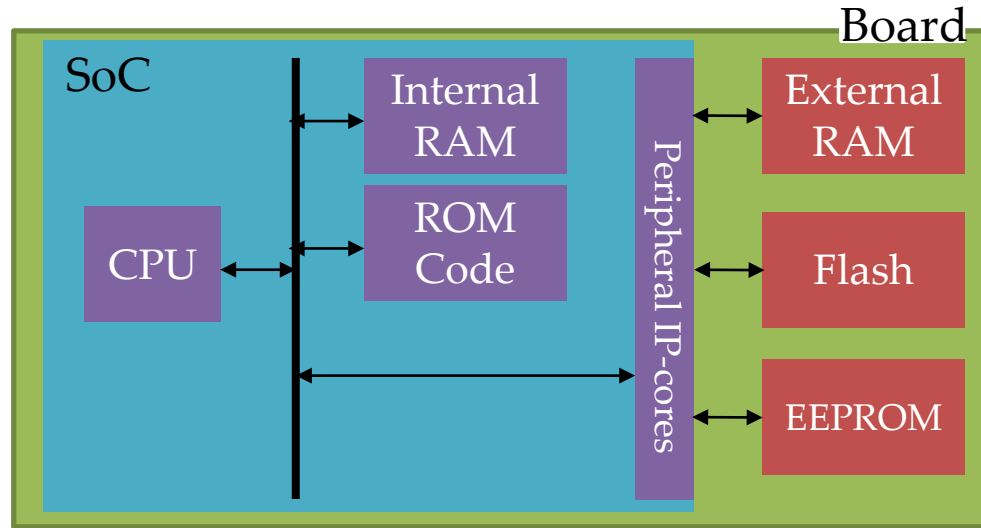
Use the + and - keys to select which entry is highlighted.
Press enter to boot the selected OS, `e` to edit the commands
before booting or `c` for a command-line. ESC to return
```


The Differences Between PC and ES

- The bootloader of embedded system is more complicated than that of PC
 - ✓ Embedded systems do not have a BIOS to perform the initial system configuration.
- Bootloader in x86 architecture consists of two parts
 - ✓ BIOS (Basic Input/Output System)
 - ✓ OS loader (located in MBR of hard disk)
 - e.g., Linux LOader(LILO) and GRUB

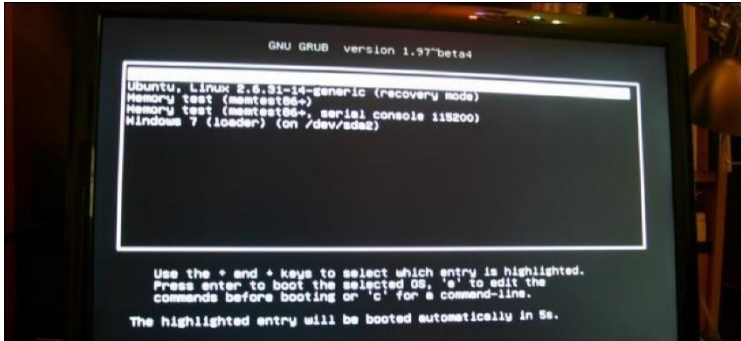


Embedded Board



What is Bootloader?

- Somebody will say ...
 - ✓ The first section of code to be executed after the embedded system is powered on or reset on any platform.
 - ✓ A program that starts whenever a device is powered on to activate the right operating system.



The Purpose of Bootloader

➤ Bootloader in (embedded) computing systems

✓ ROM code has limitations:

- Doesn't know about RAM address
- Doesn't know about board name
- Not flexible enough

ROM Code



Kernel

The Purpose of Bootloader

➤ Bootloader in (embedded) computing systems

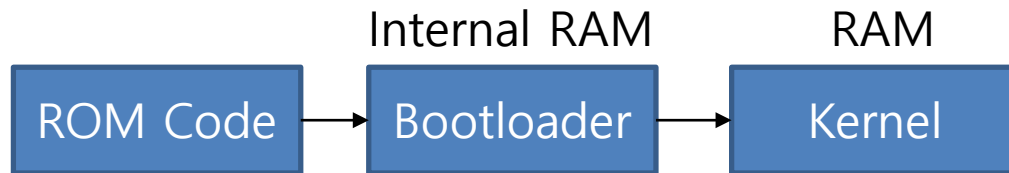
✓ ROM code has limitations:

- Doesn't know about RAM address
- Doesn't know about board name
- Not flexible enough

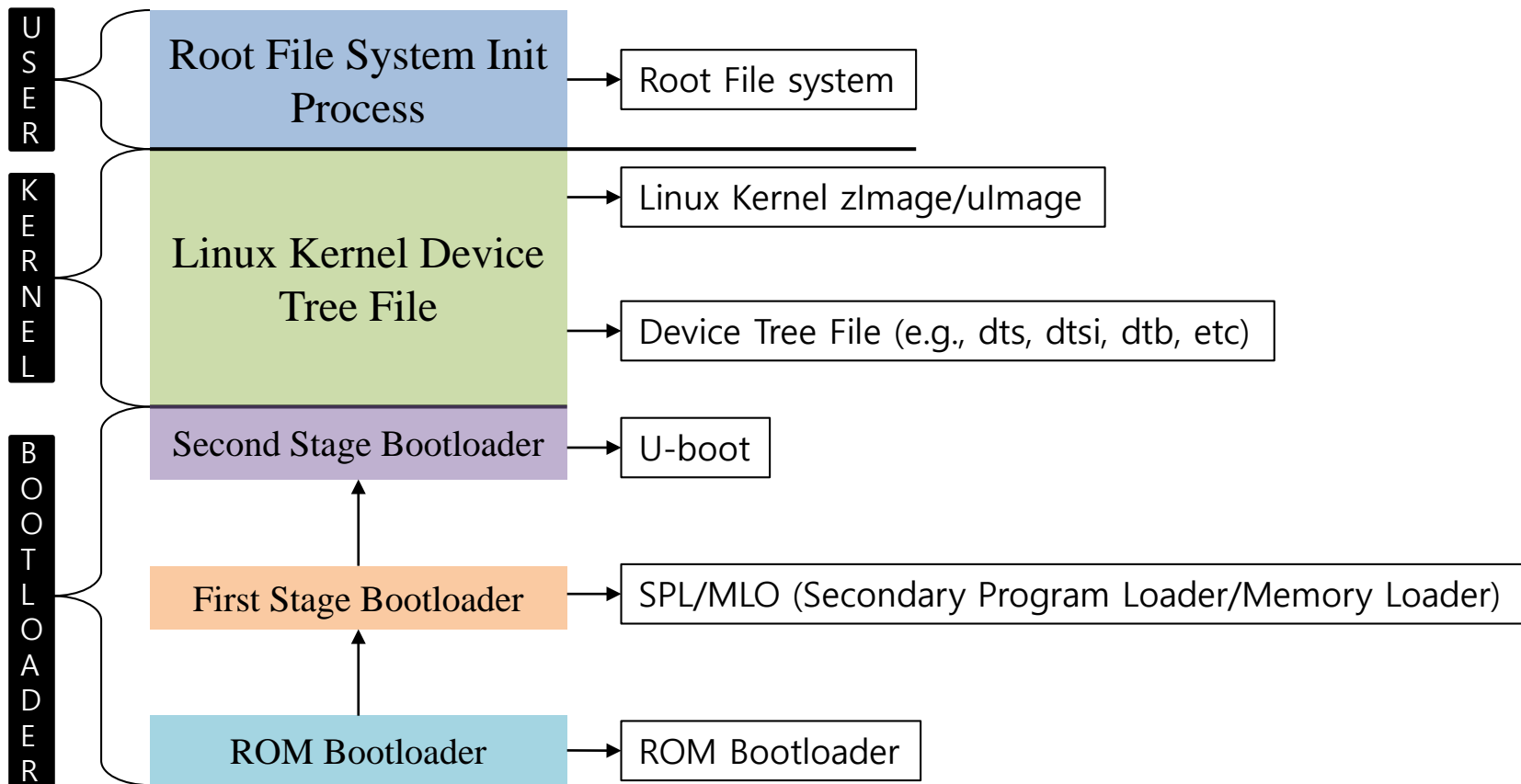


✓ Bootloader to the rescue:

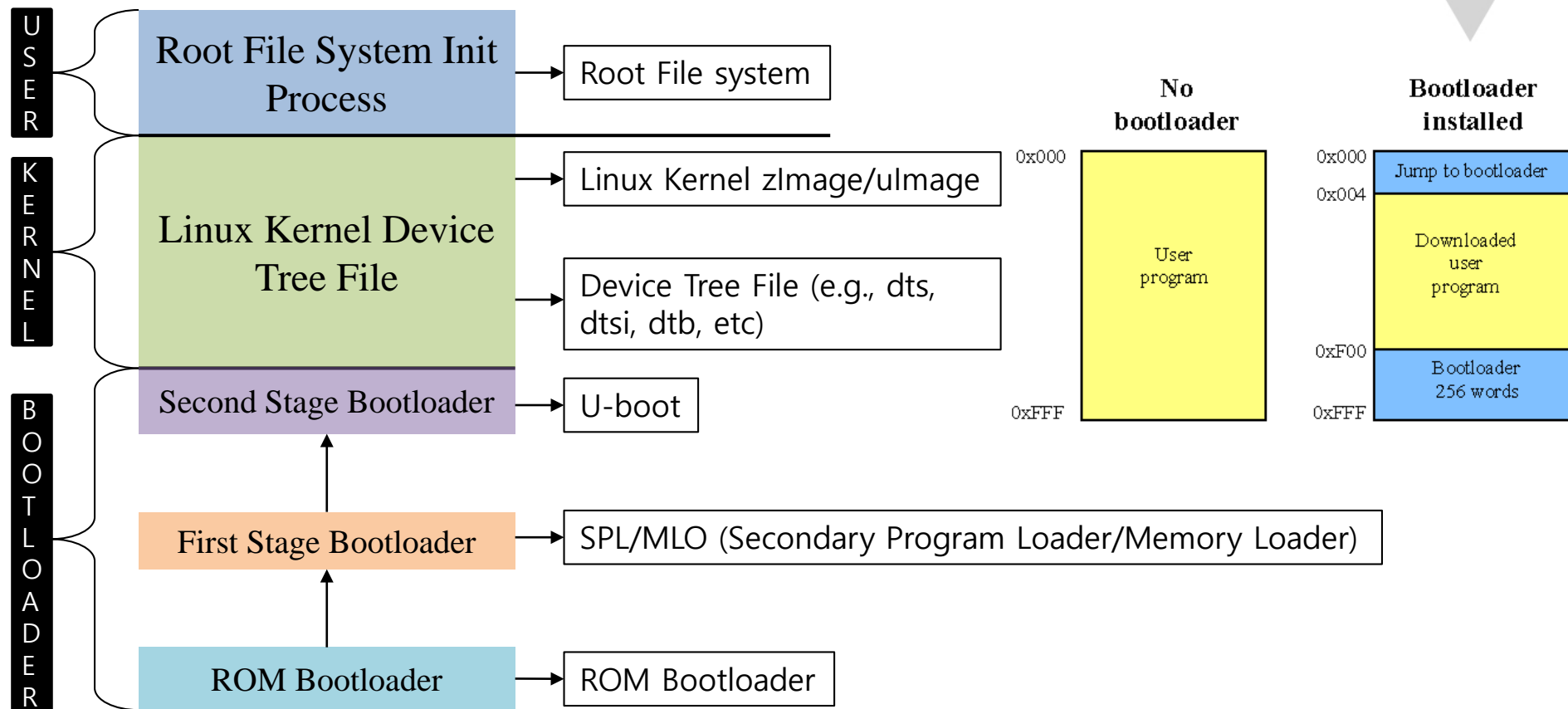
- Reside in flash. But why?
- Able to configure RAM
- Knows boot procedure
- Convenient features



How Does Bootloader Work?



How Does Bootloader Work?



What is U-Boot

➤ Das U-Boot

- ✓ A GPL'ed cross-platform boot loader
- ✓ Created by Wolfgang Denk

➤ U-Boot provides out-of-the-box support for hundreds of embedded boards and a wide variety of CPUs including PowerPC, ARM, XScale, MIPS, Coldfire, NIOS, Microblaze, and x86

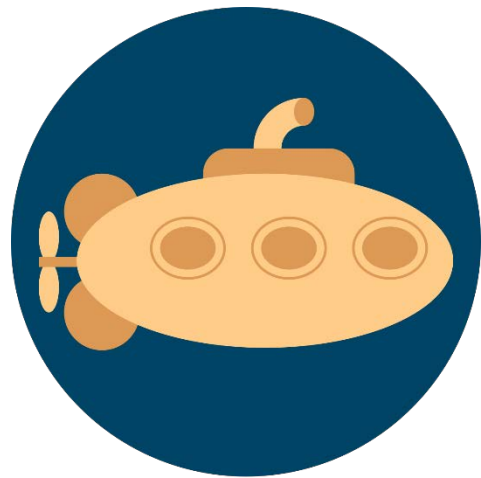
➤ Official website

- ✓ <https://www.denx.de/wiki/U-Boot/WebHome>



Why U-Boot?

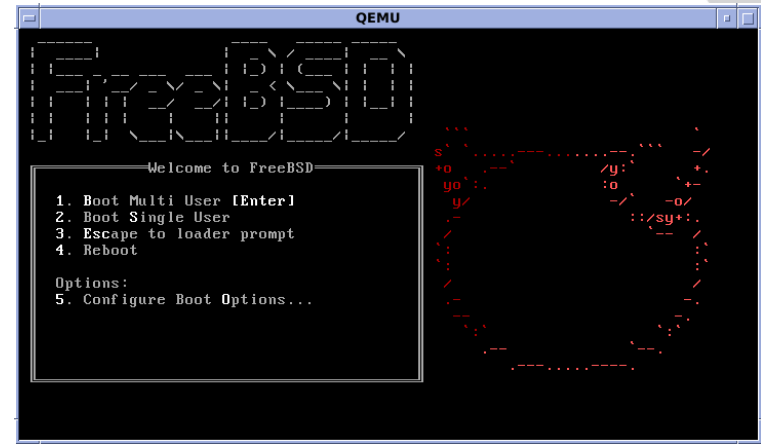
- Bootloader for embedded boards
 - ✓ Popular for Android device
 - ✓ Adoption in automotive
- 13 architectures (including ARM, x86, MIPS, etc.)
- ~300 boards
- Device drivers and lib routines
- Resembles Linux kernel a lot
- Scripting, extensive command set



U-Boot

U-Boot Features

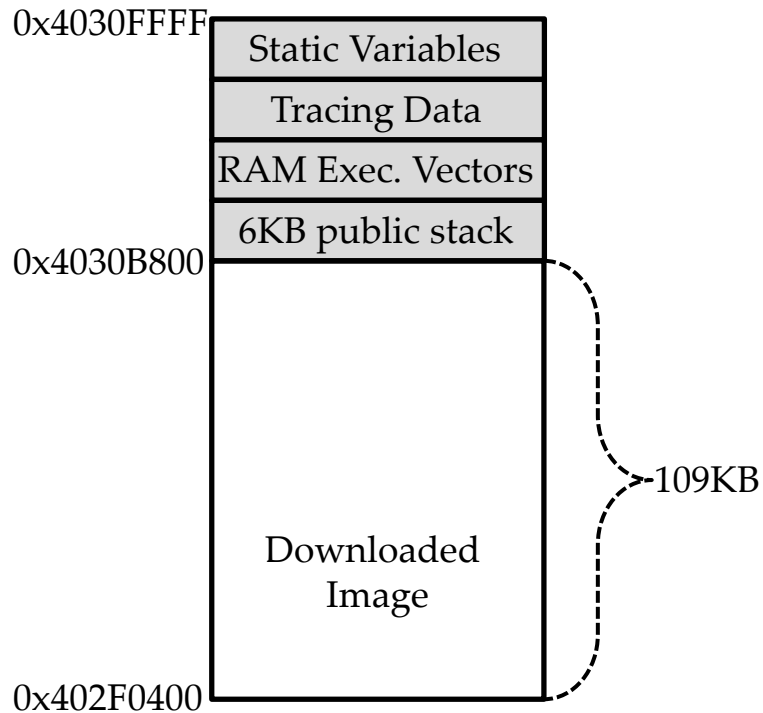
- Boots from various source
 - ✓ Network source
 - ✓ External storage
- Boots various O.S.s
 - ✓ OpenBSD, NetBSD, FreeBSD, 4.4BSD, Linux, SVR4, Esix, Solaris, Irix, SCO, Dell, NCR, VxWorks, LynxOS, pSOS, QNX, RTEMS, ARTOS Device drivers and lib routines
- 2 Stage boot (SPL + U-Boot)
- Falcon mode (SPL only)



ORACLE®
SOLARIS



Why Two-stage Boot



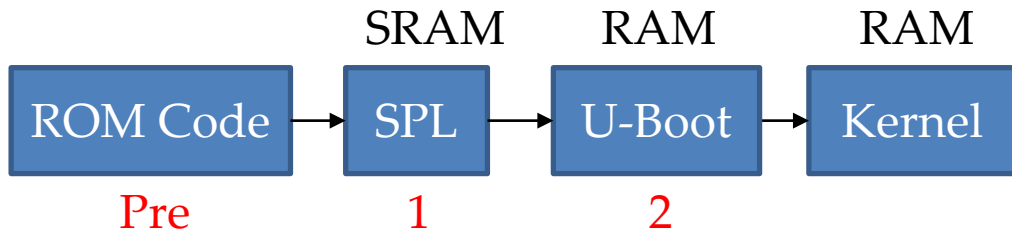
SRAM layout (From AM335x TRM)

But bootloader is so big!

For BeagleBone Black, u-boot.img is 391KB

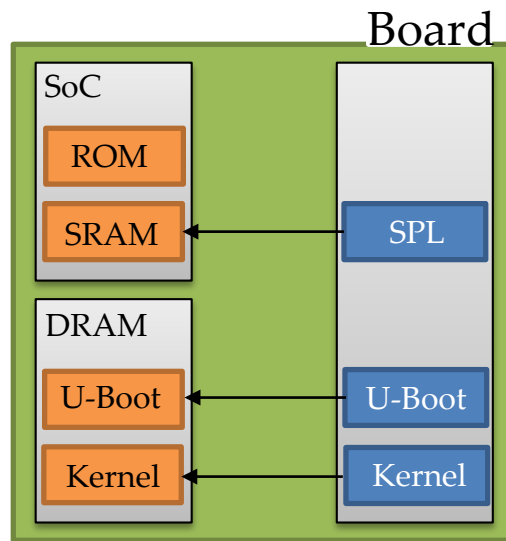
What is Two-stage Boot

An intermediate stage (SPL):



Example: BeagleBone Black

Stage	Size
SPL	75 KB
U-Boot	391 KB



Start-up an Embedded System (I/II)

- Step 1: Load the first instruction from a base address (ROM Boot).
 - ✓ 0x00000000 for ARM
 - ✓ 0xBFC00000 for MIPS
 - ✓ Two main functions
 - Configuration of the device and initialization of primary peripherals
 - Ready device for next bootloader

- Step 2: Secondary program loader (SPL) also referred as to MLO
 - ✓ The first stage of U-boot
 - ✓ To set-up the boot process for the next bootloader stage

Start-up an Embedded System (II/II)

➤ Step 3: U-Boot

- ✓ Powerful command-based control over the kernel boot environment via a serial terminal
- ✓ Environment variables in the uEnv.txt file
- ✓ These environment variables can be viewed, modified, and saved using the `printenv`, `setenv`, and `saveenv` commands, respectively.

```
CCCCCCCC
U-Boot SPL 2011.09 (Jul 26 2012 - 17:18:20)
Texas Instruments Revision detection unimplemented
Found a daughter card connected
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09 (Jul 26 2012 - 17:13:38)

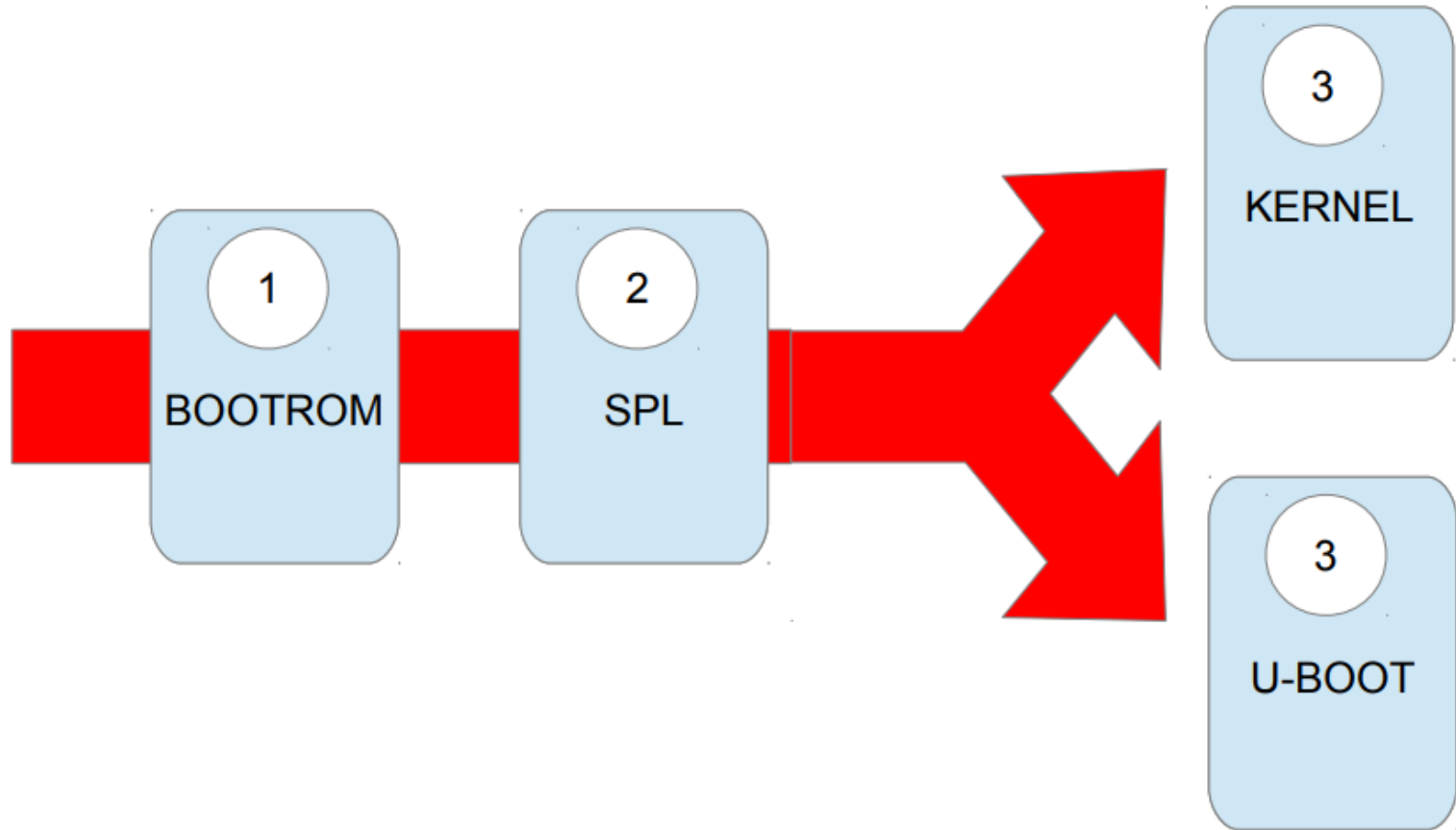
I2C:   ready
DRAM:  256 MiB
WARNING: Caches not enabled
Found a daughter card connected
NAND:  HW ECC Hamming Code selected
256 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - bad CRC, using default environment

Net:   cpsw
Hit any key to stop autoboot:  0
U-Boot#
```

➤ Step 4: Kernel Image

- ✓ uImage is the kernel image wrapped with header info that describes the kernel.

Falcon Boot



Why Falcon

- Saves time to load U-BOOT
- Saves U-BOOT execution time
- Save time to prepare Boot Parameter Area (legacy kernel)

Supported Boards

- A3m071 (PowerPC MPC 5200)
- Lwmon5 (PowerPC 440 EPX)
- Ipam390 (TI davinci)
- TI OMAP5 boards (dra7xx, uevm) NAND only
- Twister, devkit8000 (TI AM3517)
- Am335_evm (TI AM335x)

Supported Storage

