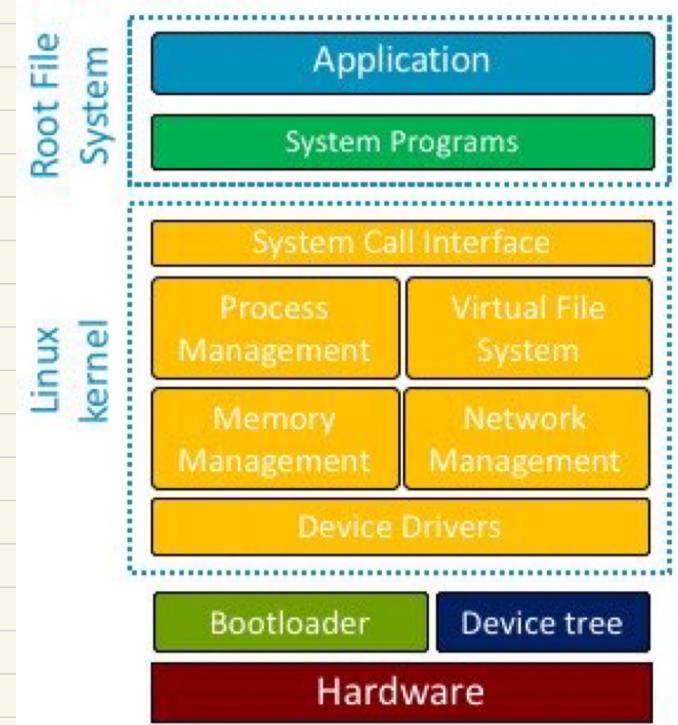


linux based embedded system component

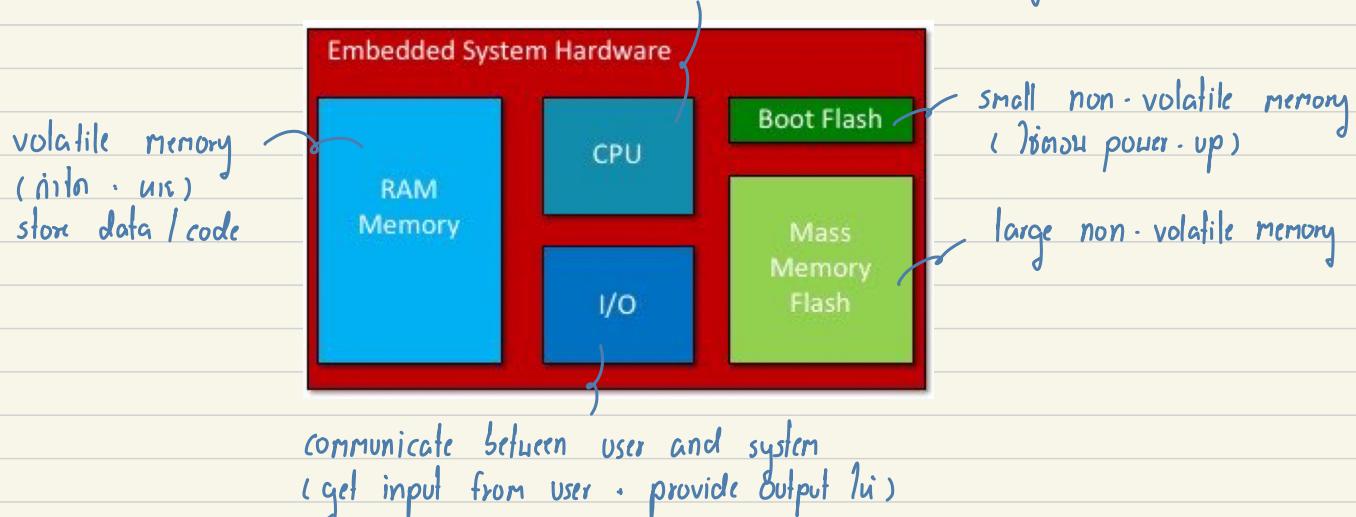
1. Bootloader → software ngn execute
nou power - up nio set - up
hardware lnis run os
2. Device tree - describe physical device
hardware n linux kernel
nios lnis init device drivers
3. Linux kernel → OS code n providing
service lnis manage
hardware resource
4. System program
5. Application → Software n implement
functionalities n embedded
system user



6 Root filesystem → it's container nis Linux kernel configuration file , system program , application

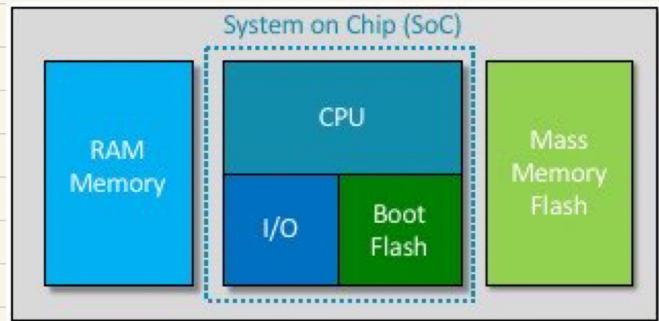
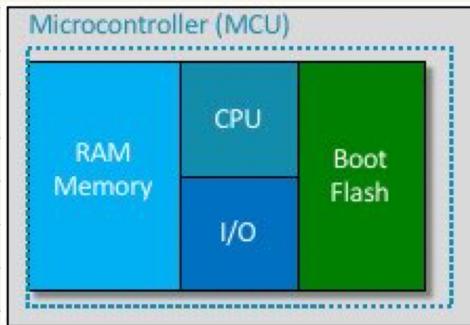
Reference Hardware model

processor n software + n arithmetic logic



Reference Hardware Model Implementation

1. single device hosts most of reference model component
2. reference model component is usually discrete CPU runs some of them (ex. I/O, file)



CPU memory map

- must generate 2^N different address ; N : number of bits on address bus
- must: device is address point of (memory and I/O require memory space)
- Memory map is describe organization of device

The role of the Bootloader

thing need to know

- running software not
- is access software location
- stack running

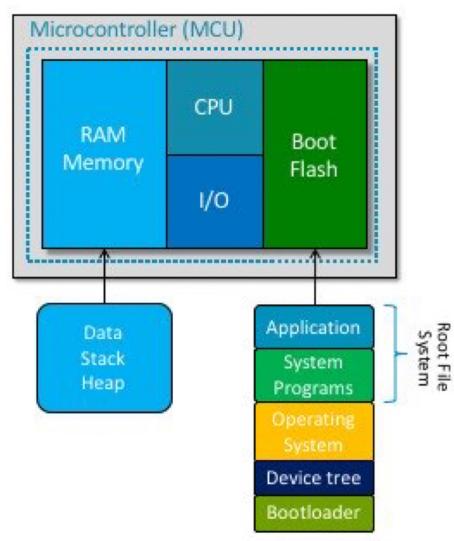
At power-up → program counter is set default known value is execute reset vector

register running address
to instruction pointer is run

pointer not in address
running instruction is run
is run firmware

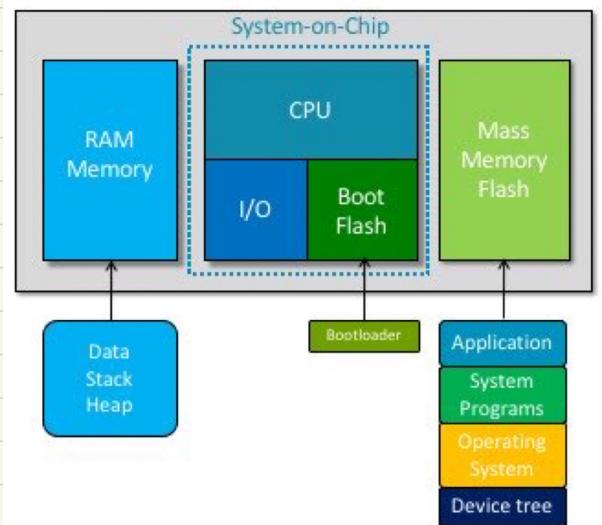
Possible Scenarios

1.) Microcontroller



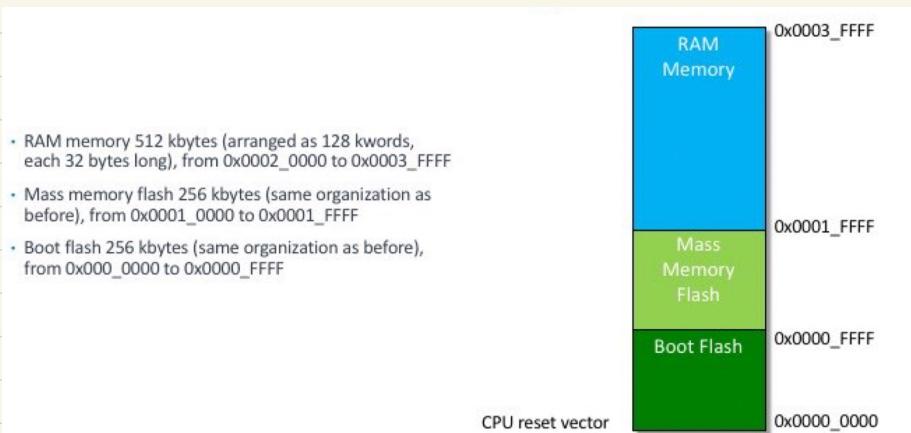
- All the sw (bootloader + device tree + operating system + root filesystem) is stored in persistent storage (boot flash) embedded in the microcontroller.
- All the sw is executed from the persistent storage.
- The CPU reset vector is located in the boot flash.
- The RAM Memory is embedded in the microcontroller and is used for data, stack and heap only.

2.) System on chip



- The bootloader is stored into the boot flash. { *also MCU*
- The CPU reset vector is located in the boot flash.
- The root filesystem, operating systems, and device tree are stored in the mass memory flash and loaded in RAM memory by the bootloader.
- The RAM memory is external to the SoC. It will store the operating system + application software, root filesystem (if configured as RAM disk), data, stack, and heap.

Example of bootloader Operations (SoC)



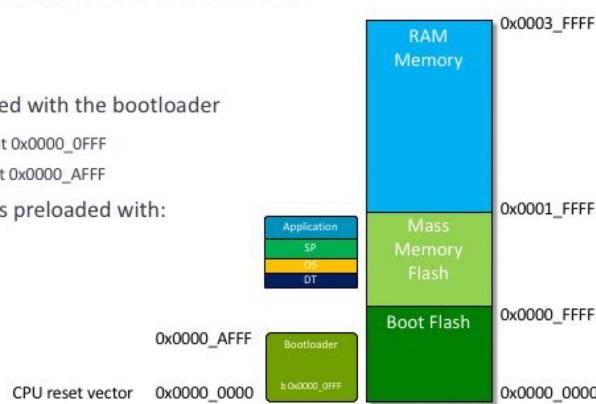
Time = **before power up**

The boot flash is preloaded with the bootloader

- First bootloader instruction at 0x0000_0FFF
- Last bootloader instruction at 0x0000_AFFF

The mass memory flash is preloaded with:

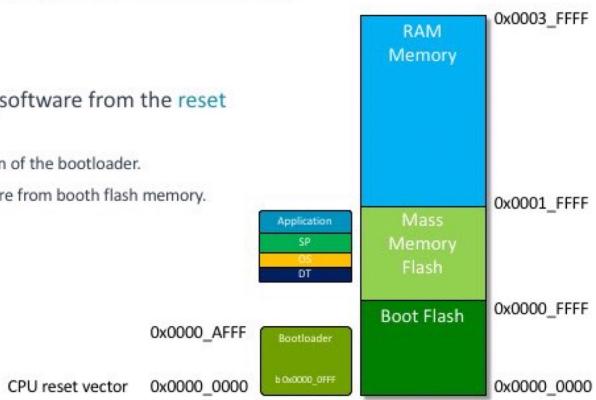
- Device tree (DT)
- Operating systems (OS)
- System programs (SP)
- Application



Time = **power up**

The CPU starts executing software from the **reset vector**:

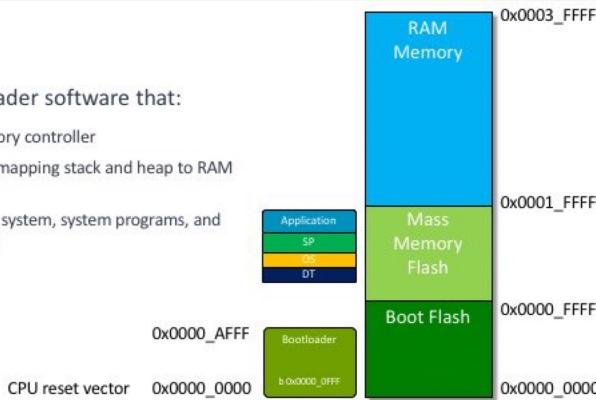
- It jumps to the first instruction of the bootloader.
- It runs the bootloader software from boot flash memory.



Time = **during bootstrap**

The CPU executes bootloader software that:

- Initializes the CPU RAM memory controller
- Sets up the CPU registers for mapping stack and heap to RAM memory
- Copies device tree, operating system, system programs, and applications to RAM memory

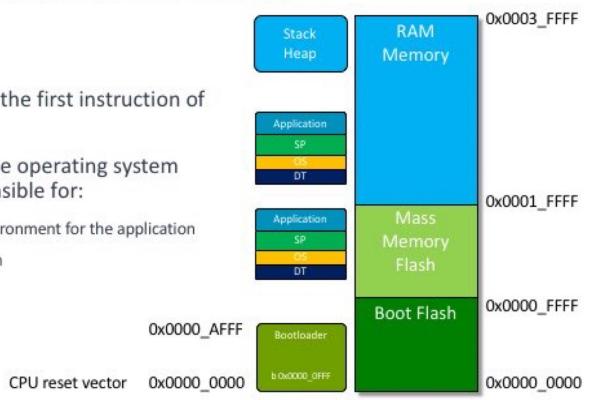


Time = **end of bootstrap**

The bootloader jumps to the first instruction of the operating system.

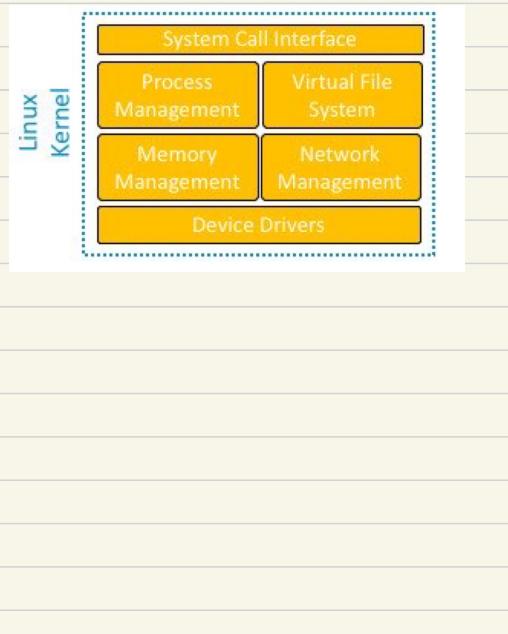
The CPU now executes the operating system software, which is responsible for:

- Setting-up the execution environment for the application
- Starting application execution



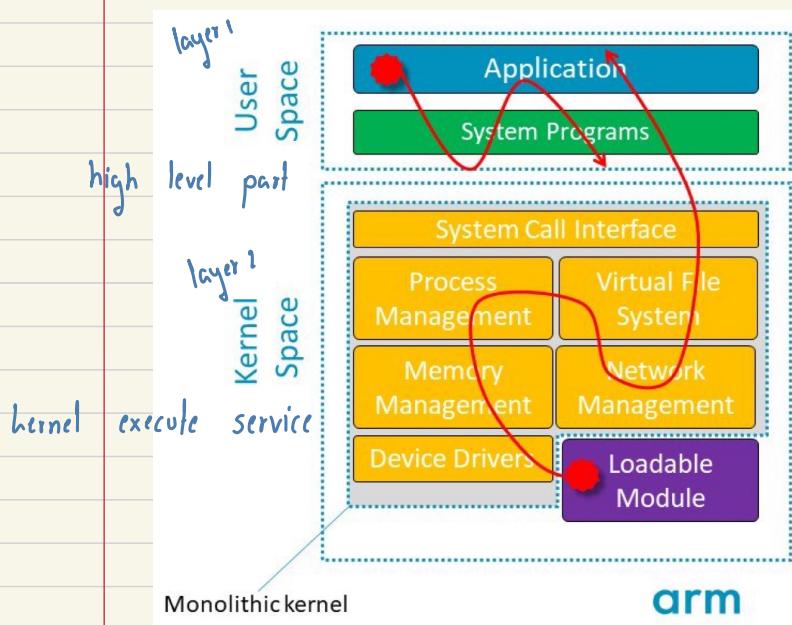
Linux Kernel

It software ដែលគ្រប់ hardware resource នៃ embedded system



It offers services such as:

- Process management → task scheduling
- Process scheduling
- Inter-process communication → share data
- Memory management → share memory ឬ block តូចតាម
- I/O management (device drivers)
- File system
- Networking
- And more



នៅ layer 1 នៃ address space នៅក្នុង

- basic service → delivered by single executable
- service status extended in run-time via loadable kernel modules

advantage

- សម្រាប់ក្រុមហ៊ុន: application, system program, kernel
- Bugs នៃ user space ត្រូវបានកិច្ចការណ៍

disadvantage

- bugs នៃកំណត់ component នឹង
- kernel ត្រូវបានពិនិត្យឡាយ

Device tree

Handles manage hardware resource, kernel manages resource in each embedded system

1.) hardcode into kernel binary code (not suitable for hardware definition) or compile source code (use)

2.) provide it to kernel from bootloader to binary file (device tree blob)

DTB

using device tree source (DTS)

- A hardware definition can be changed more easily as only DTS recompilation is needed.
- Kernel recompilation is not needed upon changes to the hardware definition. This is a big time saver.

System program

an executable on command line (ex. ls, cat)

local environment

program development vs execution

They can be divided into:

- Status information
- File modification
- Programming language support
- Program loading and execution
- Communications
- Application programs

Application

↳ software running user service

Examples can be found in many different products:

- Network Attached Storage (NAS)
- Network router
- In-vehicle infotainment
- Specialized lab equipment
- And more

Root filesystem

now startup linux, kernel uses filesystem (root filesystem) which contains configuration file to initialize execution environment and application uses first user-level process (init)

The root filesystem can be:

- A portion of the RAM treated as a file system known as Initial RAM Disk (initrd), if the embedded system does not need to store data persistently during its operations
- A persistent storage in the embedded system, if the embedded system has to store data persistently during its operations
- A persistent storage accessed over the network, if developing a Linux-based embedded system

```
/          # Disk root
/bin       # Repository for binary files
/lib       # Repository for library files
/dev       # Repository for device files
console c 5 1   # Console device file
null c 1 3    # Null device file
zero c 1 5    # All-zero device file
tty c 5 0     # Serial console device file
tty0 c 4 0    # Serial terminal device file
tty1 c 4 1    #
tty2 c 4 2    #
tty3 c 4 3    #
tty4 c 4 4    #
tty5 c 4 5    #
/etc        # Repository for config files
inittab     # The inittab
/init.d     # Repository for init config files
            # The script run at sysinit
rcS         #
/proc       # The /proc file system
/sbin       # Repository for accessory binary files
/tmp        # Repository for temporary files
/var        # Repository for optional config files
/usr        # Repository for user files
/sys        # Repository for system service files
/media      # Mount point for removable storage
```

LAB | - Yocto Project

- 12.12.2020 Ubuntu 20.04 - 16, 18, 20.04
 - Git clone in branch hroughth -> ciowith branch sumo
 - In rpi-build/conf/bblayers.conf ciowith \$ {BSPDIR}/sources/meta-openembedded/meta-python \
- * In rpi-build/bitbake rpi-basic-image