① Embedded Linux - Introduction

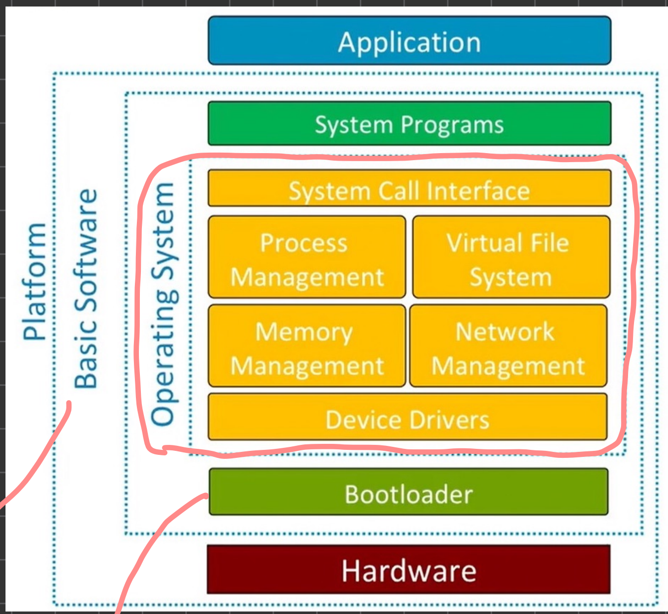
What is Embedded system

↳ special-purpose computer designed for specific app

Embedded System Components

แบ่งเป็น 2 main part

- Application
  - software ที่ให้service ที่จำเป็นต่อ user
- Platform
  - hardware + basic software ที่ทำหน้าที่ service พื้นฐาน



Bootloader: programs ที่ต้อง execute ก่อน เริ่มเพื่อทำ: initialize hardware และ execution ของ OS

Basic software

- เข้าถึง resource ผ่าน system program เช่น ls
- และการเข้าถึง resource โดย OS เป็นคนจัดการ hardware

  เช่น CPU, RTOS, device driver management

OS สำหรับ Embedded system

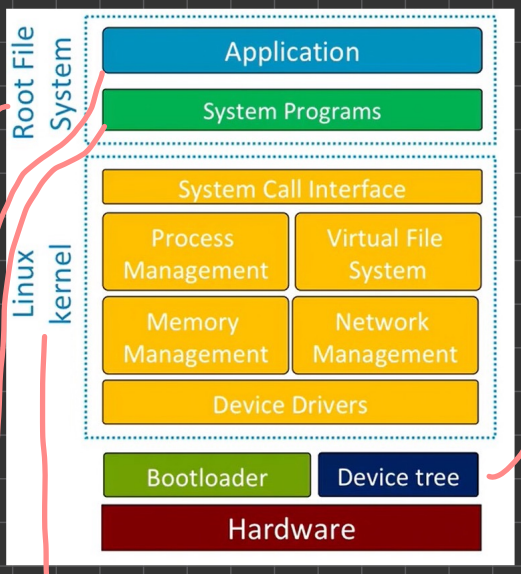
↳ มีหลายแบบขึ้นกับจุดประสงค์

Linux in Embedded system

เช่น mobile phone / In-flight entertainment system
electronics board that display the msg
gas station pump

ทำไม?
- open source
- strong community
- flexibility / adaptability
  - support new hardware เช่น SoC

2 Linux-based Embedded system component stack



Device tree : tree data structure
ที่มี node ที่แทน physical device
ใน hardware ที่ Linux kernel ใช้
ในการ initialize device driver

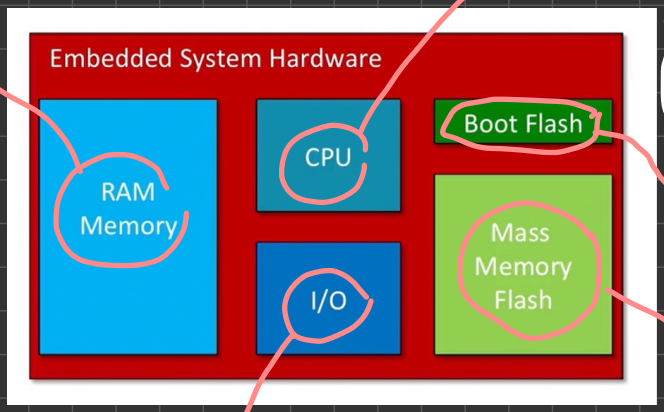
linux kernel main component!
คือ os code ที่มี service ที่จำเป็น
hardware resource

user program
งานของ user ที่ใช้ในการเข้าถึง os service

Application
software ที่ใช้ทำงานตามที่ delivered ให้ user

Root filesystem
ที่เก็บ linux kernel configuration files / the system program / app

Reference Hardware Model



Random access memory

volatile mem เช่น data/mem

central processing unit

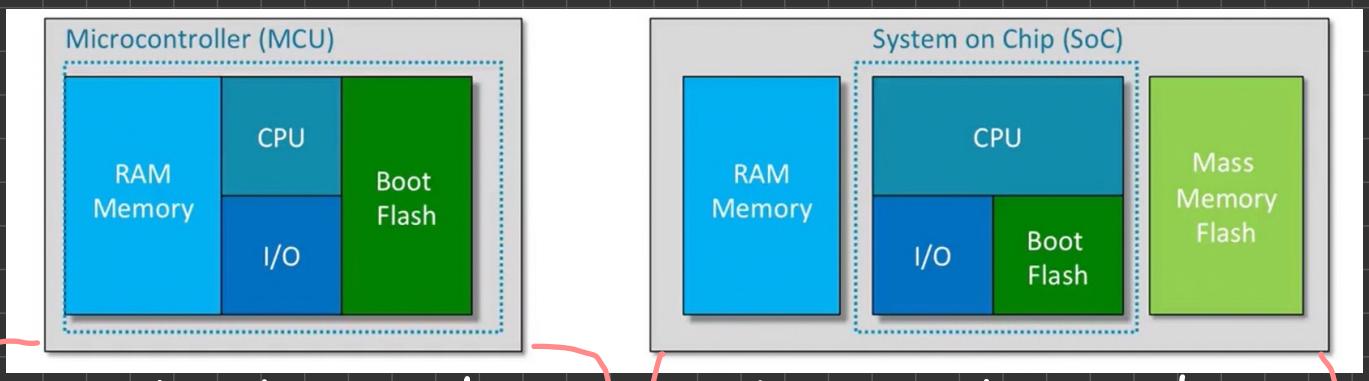
processor ของ software เช่น arithmetic logic

non-volatile mem ขนาดเล็ก

no-volatile mem ขนาดใหญ่

communicate between user and system

implement

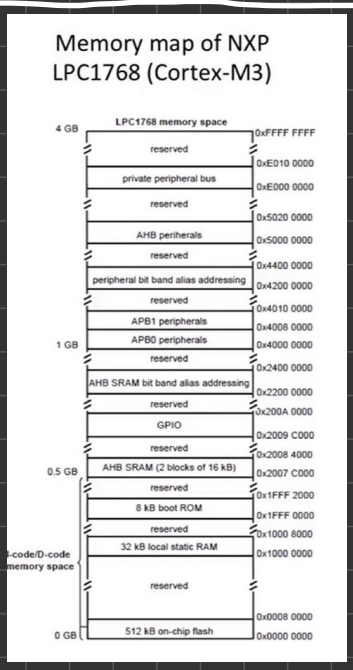


microcontroller-base implement

single device host of the reference model components

System-On-Chip implement

most of reference model components are discrete เช่น มี CPU รวมอยู่กับบางตัวด้วย



- CPU generates 2^N address ที่ต่างกัน ($0 \rightarrow 2^N-1$) โดย N คือ จำนวน bit ของ address bus
- แต่ละ device จะมี address ของมัน

Bootloader

- ทำหน้าที่ boot up machine ในขั้นแรกสุด เพื่อให้ software run
- processor ต้องรู้ว่า → where the software is located
  - → How to get access to the software location
  - → where stack is located

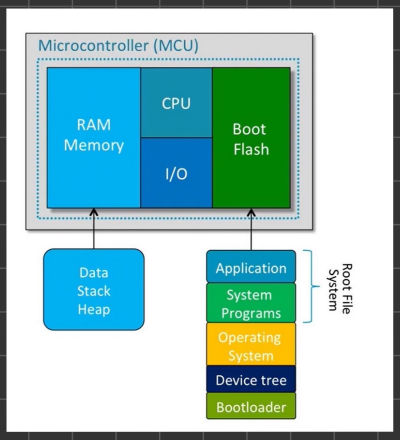
ในขณะที่เปิดโปรแกรม the program counter จะถูก set เป็น default known value เพื่อจะ execute reset vector คือ pointer ที่ชี้ไปที่ address ที่เก็บ instruction แรกที่จะ run ของ firmware

register ที่เก็บ address ของ instruction ต่อไป เพื่อ run

- bootloader คือ สิ่งแรกที่ run เพื่อจะเป็นส่วนที่จัดการในเรื่องข้อมูลต่างๆที่ต้องใช้ processor เช่น startup software location, how to access it, where stack located เพราะอาจมีที่อื่นๆขึ้นอยู่กับ system's architecture

---

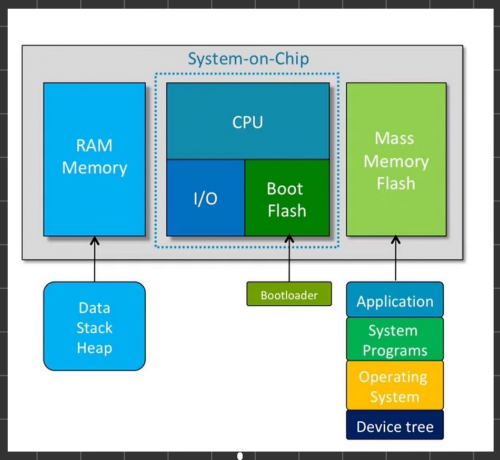
Possible scenarios



scenarios 1

Scenario 1, typical of microcontrollers

- 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.

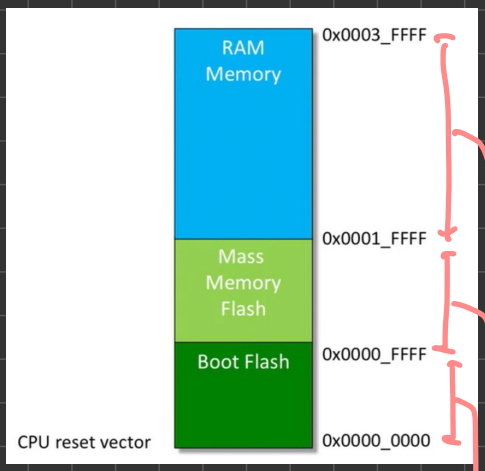


scenarios 2

Scenario 2, typical of System-on-Chip

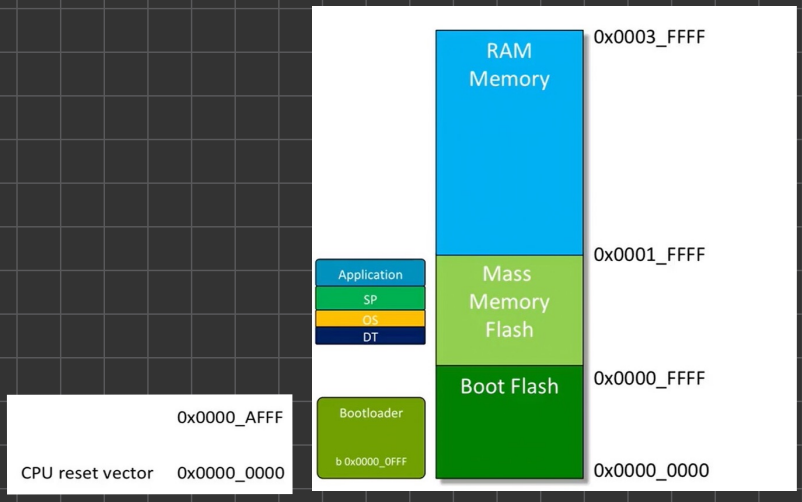
- The bootloader is stored into the boot flash.
- The CPU reset vector is located in the boot flash.
- The root filesystem, operating systems, and device tree are stored in the mass memory flash and loaded in RAM memory by the bootloader.
- The RAM memory is external to the SoC. It will store the operating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

example of bootloader operation



พิจารณา scenario 2 ด้วย SoC ที่มี memory map ตามนี้

- RAM mem 512k byte (128 kword) แต่ละช่องเก็บ 32 byte long (0X0002_0000 - 0X0003_FFFF)
- mass memory flash 256 kbytes (0X0001_0000 - 0X0001_FFFF)
- boot flash 256 kbytes (0X0C0_CCCC - 0X0C00_FFFF)



ตอนก่อน power up

- boot flash จะถูก load bootloader ไว้เพื่อ
  โดย first bootloader instruction คือ 0x0000_0FFFF
  เช่น last bootloader instruction คือ 0x0080_AFFF

- เช่น mass memory flash จะ load ตัวไหน
  1. Device tree (DT)
  2. Operating system (OS)
  3. System programs (SP)
  4. Application

ตอนแรก power up

- CPU เริ่ม execute software จาก reset vector
  - ↳ มันจะกระโดดไปที่ first instruction ของ bootloader
  - ↳ เพราะมันจะรัน bootloader software จาก booth? flash memory

ตอนระหว่าง bootstrap

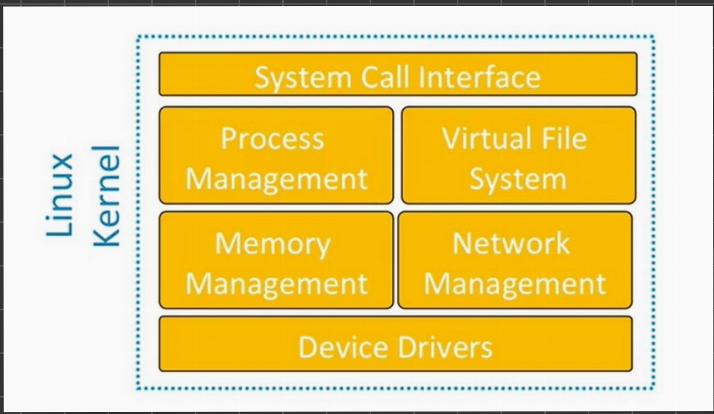
- CPU จะ execute bootloader ที่
  - ↳ initialize CPU RAM memory controller
  - ↳ set up the CPU registers เพื่อการ mapping stack และ heap ไปที่ RAM memory
  - ↳ ก๊อปปี้ device tree, operating system, system programs, applications ไปที่ RAM

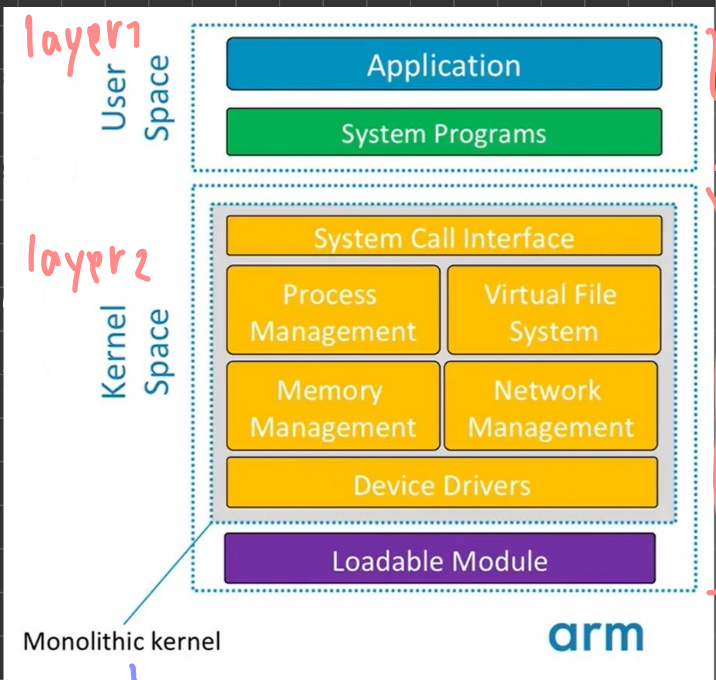
ตอนจบ bootstrap

- bootloader กระโดดไปที่ first instruction ของ OS
- CPU จะ execute OS software ที่เอาไว้ที่
  1. setting-up execution environment ของ app
  2. เริ่ม application execution

Kernel

the linux kernel ที่เป็น software ที่มีหน้าที่ในการ manage embedded system's hardware resources ให้ optimal และก็มีหน้าที่ offer services ต่างๆ





Linux kernel มีหลาย layer ที่เรียกว่า layered operating system architecture

→ high level part เพราะมี code ที่ใช้ได้ run

ที่ๆ kernel execute service for the system

2 layers นี้มี different address space

ถ้าเป็น basic services จะ delivered โดย single executable (monolithic)

แต่ services can be extended at run-time through loadable kernel modules

ข้อดี

- มีการแบ่งแยกระหว่าง app/system programs และ kernel
- bug จาก user space จะไม่ส่งผลต่อ kernel

ข้อเสีย

- bug ใน kernel component เพียงเล็กน้อย จะส่งผล system ทั้งหมด

Device Tree

I/O device, memory, etc

- ในตอนที่ kernel จะ manage <u>hardware</u> resources

kernel จะต้องรู้ว่า resources ไหนบ้างที่อยู่ใน embedded system

มี 2 วิธีในการทำให้ kernel รู้ information เหล่านี้คือ

① hardcode! ลงไปใน kernel binary code
แต่ทุกครั้งที่มีการแก้ไข hardware definition จะต้อง compile source code ใหม่

② provide it to the kernel ในตอนที่ bootloader ใช้ binary file (device tree blob)

device tree blob (DTB) file มาจาก device tree source (DTS)

↳ hardware definition จะถูกแก้ไขได้ง่ายมาก โดยใช้แค่ DTS recompilation

↳ ไม่ต้อง kernel recompilation ตอนที่ hardware definition มีการเปลี่ยนแปลง

System programs

คือ executable ของ command line โดยที่มันจะเป็นส่วนของ environment ให้ program development และ execution

They can be divided into:

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

เช่น ls, cat, cd

Application

คือ software ที่ใช้เพื่อตอบสนอง 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

ในตอน startup linux kernel ต้องใช้ file system (root filesystem) ซึ่งส่วนนี้จะมี configuration file ที่ใช้ในการ จัดเตรียม execution environment สำหรับ application เพราะก็คือ first user-level (init) process

The root filesystem can be:

- ส่วนหนึ่งของ RAM ที่เป็นเหมือน filesystem (initial RAM disk) (initrd)
  ถ้า embedded ไม่ได้จำเป็นต้องใช้ ในการเก็บข้อมูลการระหว่าง operation นั้นๆ
- A persistent (ถาวร) storage ใน embedded system
- A persistent storage accessed over the network

Typical Layout of the Root Filesystem

```
/                               # 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
                        rcS     # The script run at sysinit
        /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
```

32 © 2017 Arm Limited

arm

Quiz

Quiz 2

You have unlimited attempts.

Lecture 2:

1. "The bootloader is a piece of software responsible for..."

- ○ Providing all the services to manage the hardware resources
- ✔ Setting-up the hardware to run the operating system
- ○ Implementing the functionalities to be delivered to the embedded system user
- ○ Storing the Linux Kernel Configuration files, the system programs, and the application

Your answer is correct.

2. "In the 'Reference Hardware Model', which component is responsible for volatile memory?"

- ○ Boot Flash
- ○ Mass Memory Flash
- ✔ RAM
- ○ CPU

Your answer is correct.

3. "The Linux kernel is split into..."

- ✔ Two layers: User space and kernel space
- ○ Two layers: Developer space and user space
- ○ Three layers: OS, hardware, and software

Your answer is correct.

4. "Which of the following are methods for informing the kernel of which resources are available in the embedded system?"

- ☐ Get the kernel to look it up online
- ✔ Using a device tree blob
- ☐ Look it up in a memory map
- ✔ Hardcode it into the kernel binary code

Your answer is correct.

5. "What produces a device tree blob (DTB) file?"

- ○ Device Tree Origin
- ✔ Device Tree Source
- ○ Device Tree Generation
- ○ Device Tree

Your answer is correct.

6. "In regard to the bootloader – at power-up, the program counter is set to a default value, known as the..."

- ○ Default vector
- ○ Initial vector
- ○ Primary vector
- ✔ Reset vector

Your answer is correct.

7. "A bootloader operation carried out at the power-up stage is..."

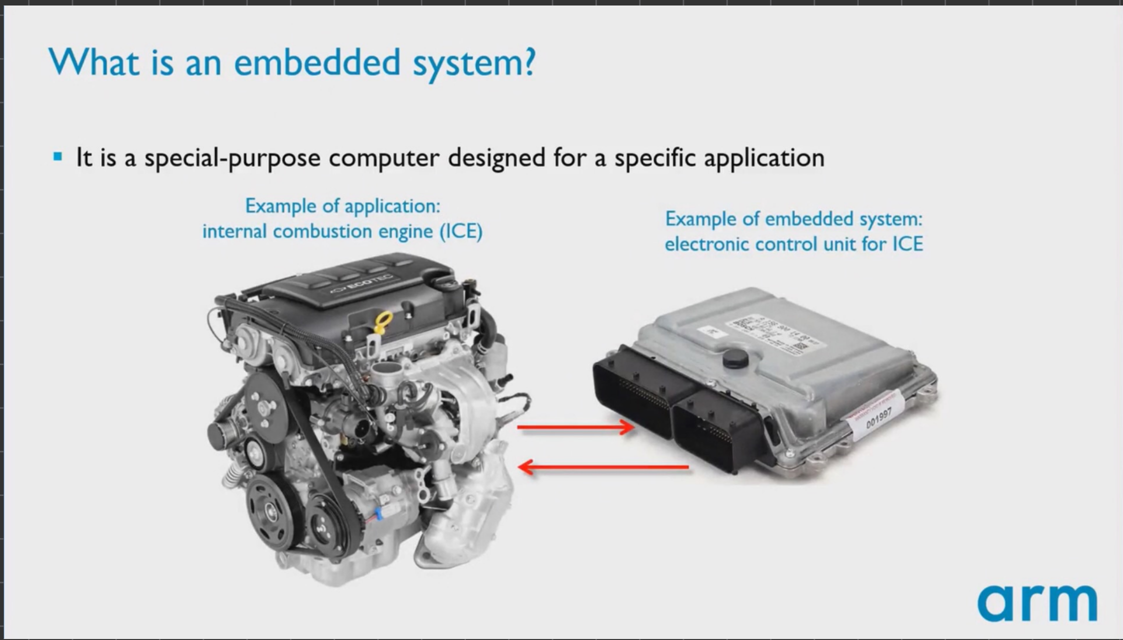
- ✔ Begins executing software from the reset vector
- ○ Preloading the boot flash with the bootloader
- ○ Jumping to the first instruction of the operating system
- ○ Preloading the mass memory flash with the device tree

Your answer is correct.

Yocto Project → open source custom linux-based distribution development tool

Poky → reference distribution of the Yocto Project

Raspberry Pi → single board computer used to host the custom linux-based distribution



Linux Based Embeded System

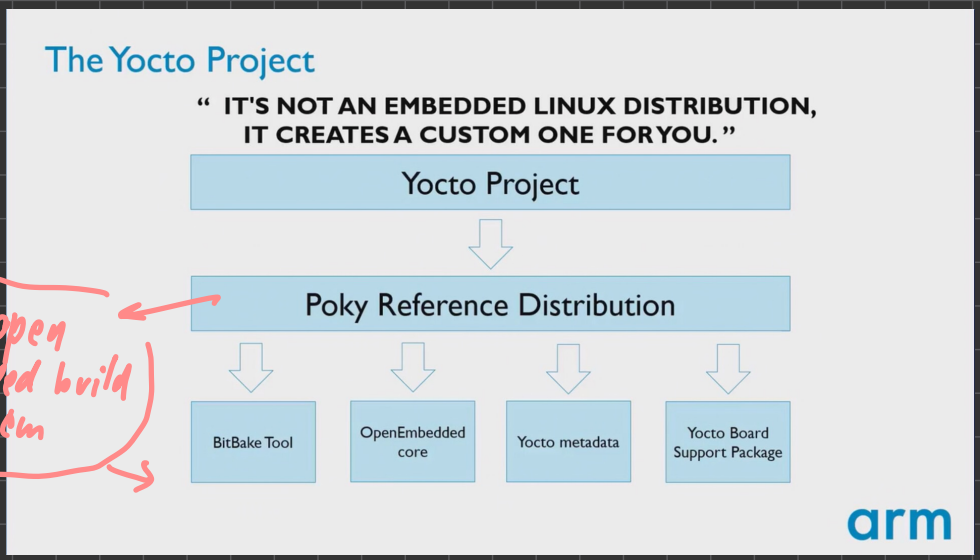
มี 6 ส่วนหลักๆ คือ

1. Bootloader → เก็บใน ROM ควบคุมสิ่งที่เกิดขึ้นใน system ตอนที่เปิดขึ้น
2. Device tree → script ของ physical hardware ที่ available ใน system
3. Linux kernel → software ที่ manager เชื่อมกับ computer และ hardware resource
4. System programs → set ของ utilities ที่ให้ app access resource ของ system
5. Application → software ที่ implement งานของ system ให้ user ใช้
6. Root filesystem → ที่ๆ ใช้เก็บ (โปรแกรมทั้งหมด/application configuration file

LAB

In lab we part we linux system with bootloader
we flash to micro SD card

The Yocto Project provide platform to create
custom Linux space system
for any embedded linux hardware



of open embedded build system