Introduction to Embedded Systems

Minsoo Ryu

Hanyang University

Outline

1. Definition of embedded systems
2. History and applications
3. Characteristics of embedded systems
   - Purposes and constraints
   - User interfaces
   - Processors for embedded systems
   - Development issues

What is an embedded system?

❐ Definition (from Wikipedia) <-> general purpose system

- An embedded system is a special-purpose system in which the computer is completely encapsulated by the device it controls. Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few pre-defined tasks, usually with very specific requirements.

❐ General view

- Embedded system is a computing system embedded into a larger product

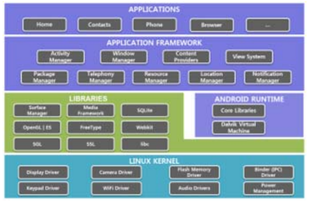
Smartphone as an embedded system

Smartphone

=

Hardware

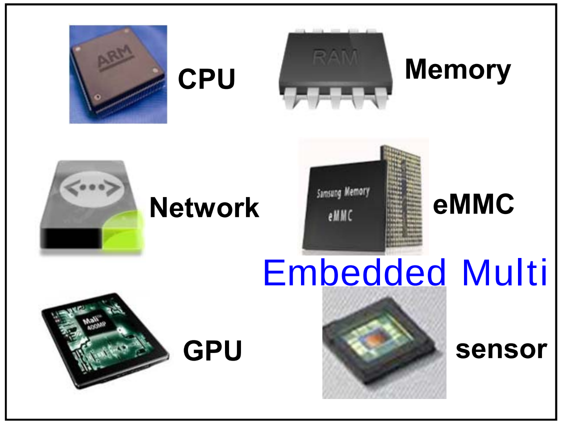
+



Software

SAMSUNG LG

Apple mi 小米 xiaomi.com



CPU

Memory

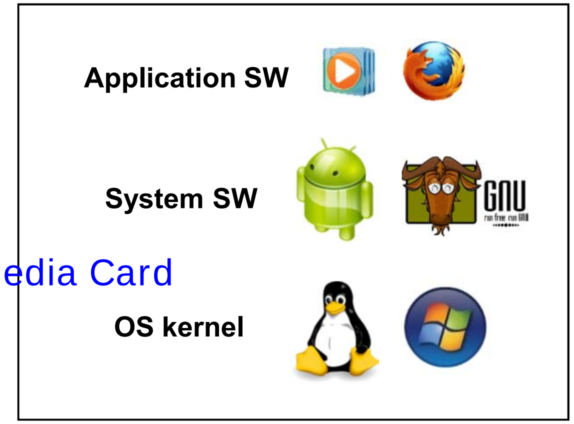
Network

eMMC

Embedded Multi Media Card

GPU

sensor



Application SW

System SW

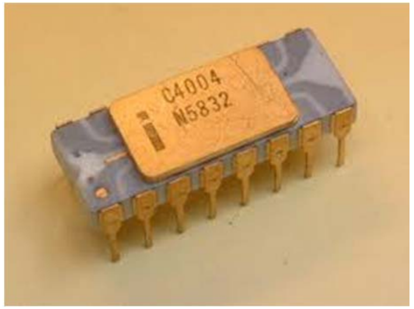
OS kernel

Early history

- One of the very first embedded systems was the Apollo Guidance Computer
  - Developed by Charles Stark Draper at MIT

- An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961

- Since 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality
  - An early microprocessor, the Intel 4004, was designed for calculators and other small systems



Typical applications

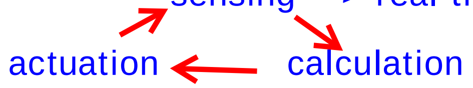
- ❒ Vehicles
  - Ignition Systems, Engine Control, Antilock Braking System,
- ❒ Consumer Electronics
  - TVs, STBs, appliances, toys, automobiles, cell phones …
- ❒ Industrial Control
  - robotics, control systems…
- ❒ Medical devices and systems
  - Infusion Pumps, Dialysis Machines, Prosthetic Devices, Cardiac Monitors, …
- ❒ Networks
  - routers, hubs, gateways, …
- ❒ Office Automation
  - fax machines, photocopiers, printers, monitors, …

Characteristics of embedded systems

- ❒ **Dedicated purposes**
- ❒ **Real-time requirements**
  - **Deadlines and periods**
- ❒ **Mass production**
- ❒ **Harsh operating conditions**
- ❒ **Limited resources**
  - **Limited processing power and memory**
  - **Many systems are battery-powered**
- ❒ **Portability and mobility**

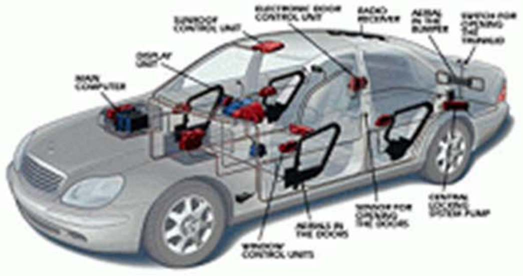
real-time requirements -> 사람의 개입이 적음 ,스스로 동작

sensing -> real time requirement가 요구됨, 스스로 판단해야하기 때문에



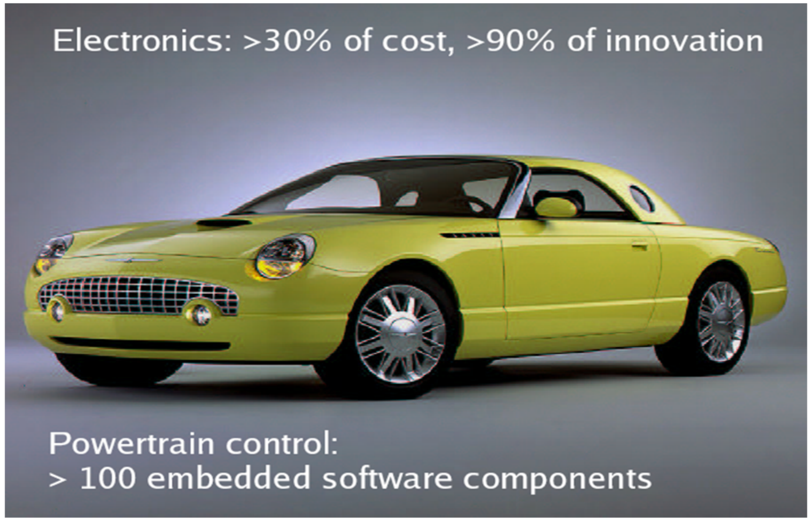
mass production -> cost
harsh operating conditions -> 예를 들어, 외부 온도가 너무 높아 cpu clock이 낮아져서 소프트 웨어적으로 오차가 발생

Examples



99% of CPUs are used in embedded systems

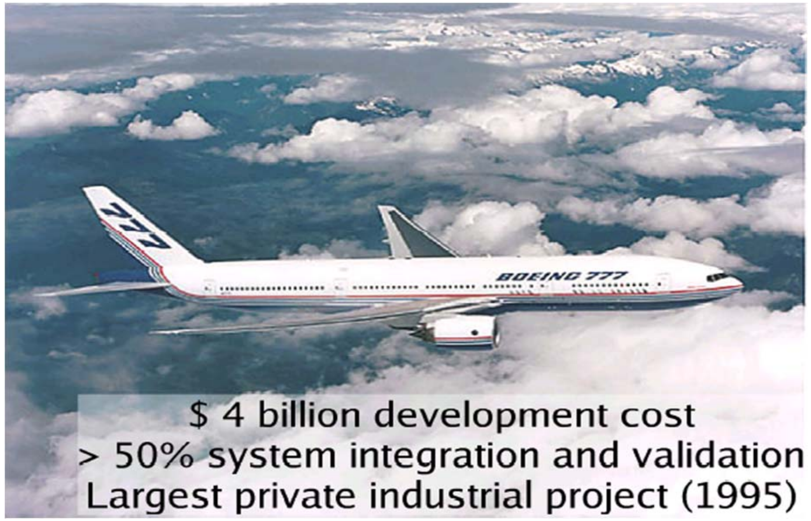
Examples



Electronics: >30% of cost, >90% of innovation

Powertrain control:
> 100 embedded software components

Mars, December 3, 1999
Lander lost due to embedded software design error
$ 184 million development cost



$ 4 billion development cost
> 50% system integration and validation
Largest private industrial project (1995)

Ariane 5, French Guyana, June 4, 1996
$800 million embedded software failure

User interfaces in embedded systems

- Embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop operating systems
  - Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system
  - More sophisticated devices use a graphical screen with touch sensing or screen-edge buttons

buttons

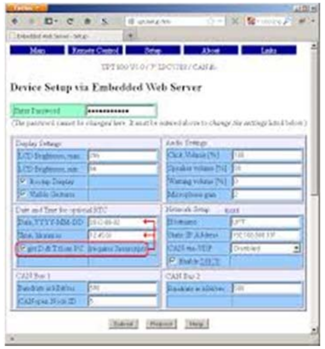
character LCD

LEDs

7-segment LED

User interfaces in embedded systems

- ❐ **Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I²C, etc.) or network (e.g. Ethernet) connection**
  - A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network router
  - The user interface is displayed in a web browser on a PC connected to the device, therefore needing no software to be installed monitor program - interface 담당 program



Device Setup via Embedded Web Server

Enter Password ••••••••••••

Display Settings

Audio Settings

Date and Time for optional RTC

Network Setup

CAN Bus 1

CAN Bus 2

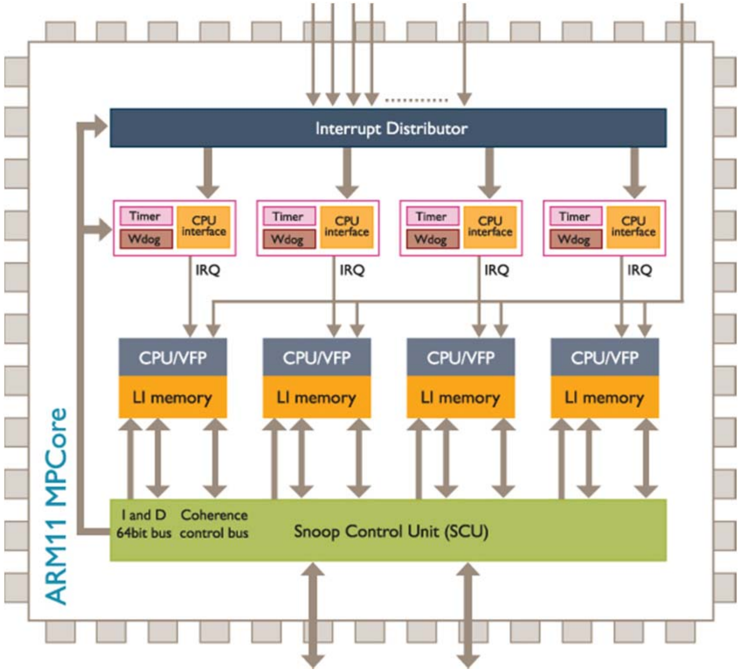
Processors for embedded systems

- ❒ General purpose processors (mostly low power)
  - A microprocessor is a single chip CPU
  - ARM, Intel Atom, Motorola's 680x0

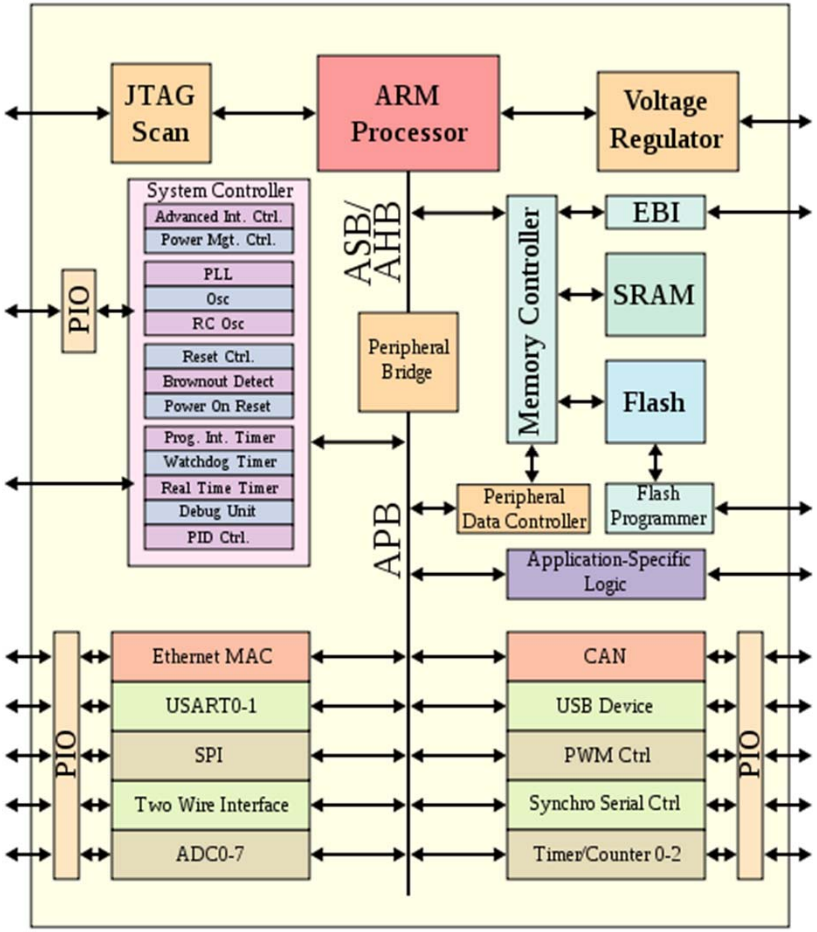
- ❒ Microcontrollers (µC)
  - Include on-chip peripherals (ROM, RAM and I/O ports) as well as CPU, thus reducing power consumption, size and cost
  - Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC 16X

- ❒ SoC (System-on-Chip)
  - Integrates all components of a computer or other electronic system into a single chip
  - It may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single chip substrate
  - Used for very high volume products
  - SoCs can be implemented as an application-specific integrated circuit (ASIC) or using a field-programmable gate array (FPGA)

ARM-based Processors



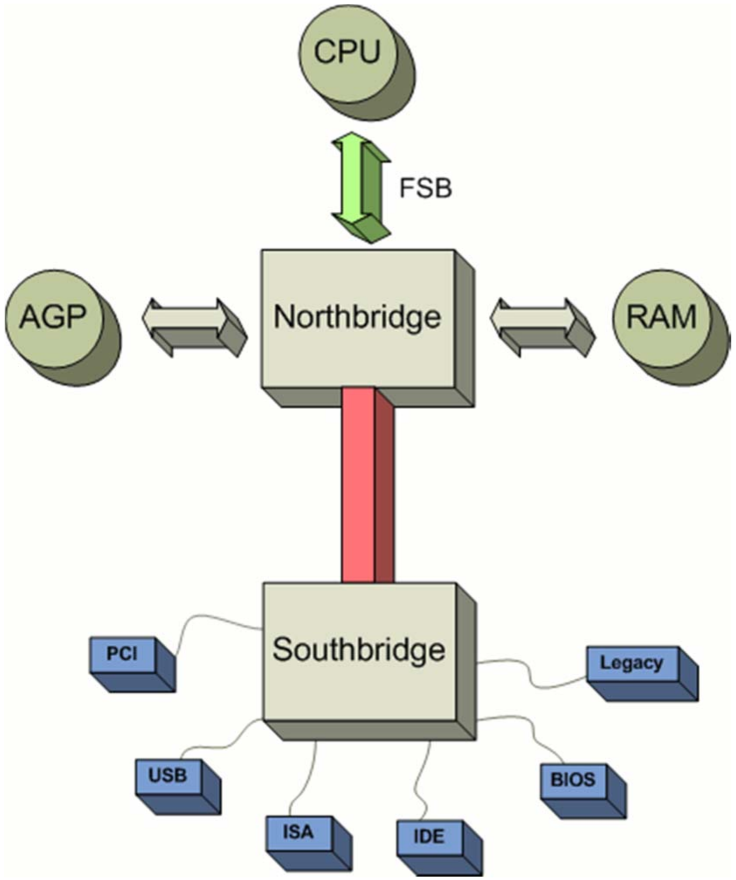
ARM11 MPCore



ARM-based SoC (Mobile AP)

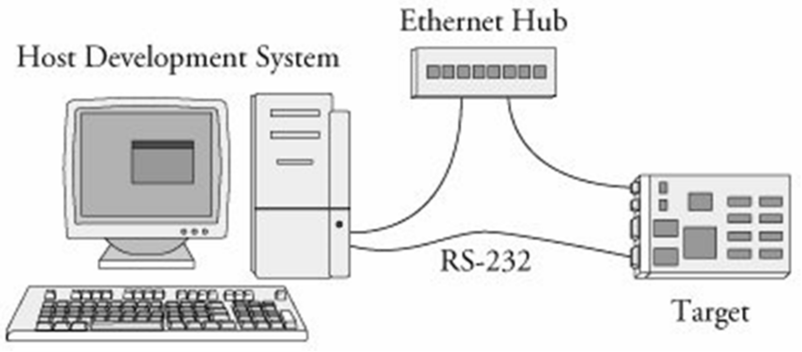
North/Southbridge Layout in Intel Chipsets

- The northbridge is used to manage data communications between a CPU and a motherboard within Intel chipsets
- The southbridge typically implements the slower capabilities of the motherboard
- Increasingly the northbridge functions have migrated to the CPU chip itself, beginning with memory and graphics controllers
  - For Intel Sandy Bridge and AMD Accelerated Processing Unit processors introduced in 2011, all of the functions of the northbridge reside on the CPU



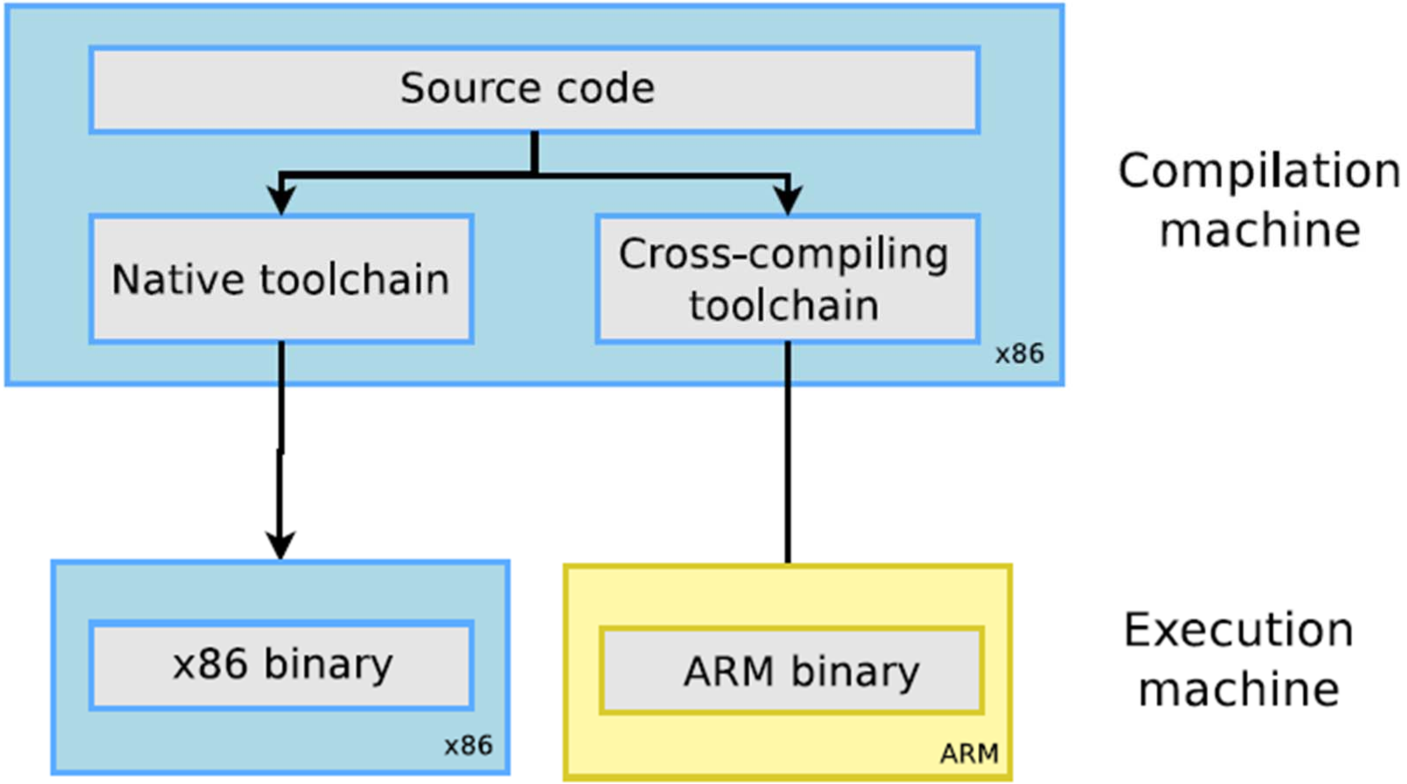
Cross Development Environment

- When doing embedded development, there is always a split between the host, the development workstation, which is typically a powerful PC and the target, which is the embedded system under development
    - They are connected by various means: almost always a serial line for debugging purposes, frequently an Ethernet connection, sometimes a JTAG interface for low-level debugging

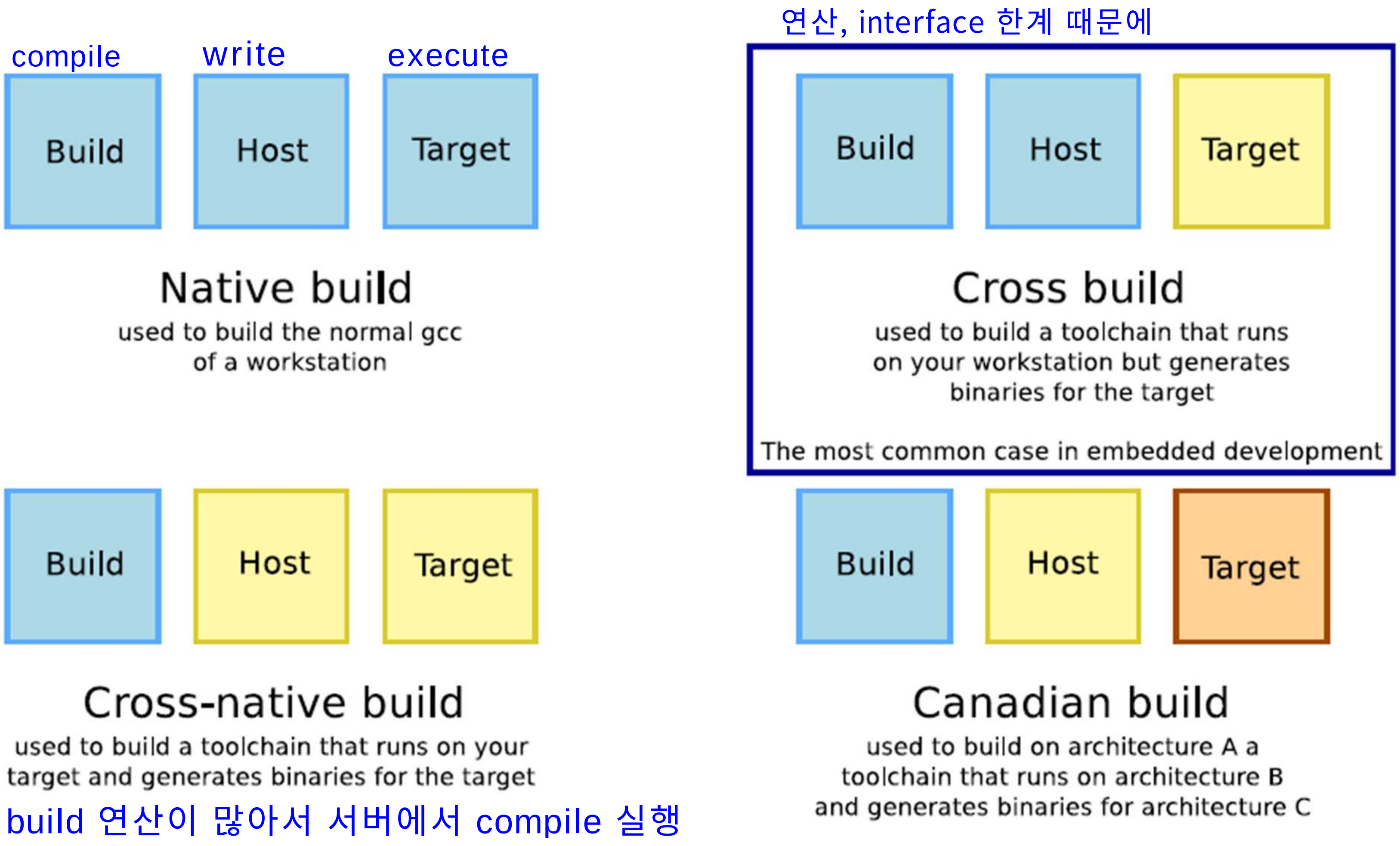


Cross Development Environment

❐ Cross compilation



Cross Development Environment



Native build

used to build the normal gcc of a workstation

연산, interface 한계 때문에

Cross build

used to build a toolchain that runs on your workstation but generates binaries for the target

The most common case in embedded development

Cross-native build

used to build a toolchain that runs on your target and generates binaries for the target

build 연산이 많아서 서버에서 compile 실행

Canadian build

used to build on architecture A a toolchain that runs on architecture B and generates binaries for architecture C

Debugging Embedded Systems

❒ Embedded debugging may be performed at different levels, depending on the facilities available

❒ Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)

❒ External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multicore systems

Debugging Embedded Systems

I/O 작업이 많이 차지하기 때문에, hardware level 디버깅이 필요하다 (register) JTAG 디버깅!!
trace32 디버거

bus signal & hardware 디버깅은 불가능 !

- ❒ An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface

  chip이 만들어질때부터 구현됨
  break point 설정, register 디버깅 가능

  - This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor

- ❒ An in-circuit emulator (ICE) replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor

  chip 생산자들이 emulator chip을 따로 생상하여 디버깅 가능한 chip (JTAG 존재한)을 제공

  - A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified, and allowing debugging on a normal PC
  - The downsides are expense and slow operation, in some cases up to 100X slower than the final system

Debugging Embedded Systems

- ❒ For SoC designs, the typical approach is to verify and debug the design on an FPGA prototype board
  - Tools such as Certus are used to insert probes in the FPGA RTL that make signals available for observation
  - This is used to debug hardware, firmware and software interactions across multiple FPGA with capabilities similar to a logic analyzer
- ❒ Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary
  - For instance, debugging a software- (and microprocessor-) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor)
  - An increasing number of embedded systems today use more than one single processor core
  - A common problem with multi-core development is the proper synchronization of software execution
  - In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance

Debugging Embedded Systems

❒ Real-time operating systems (RTOS) often supports tracing of operating system events

- A graphical view is presented by a host PC tool, based on a recording of the system behavior. The trace recording can be performed in software, by the RTOS, or by special tracing hardware
- RTOS tracing allows developers to understand timing and performance issues of the software system and gives a good understanding of the high-level system behavior
- Commercial tools like RTXC Quadros or IAR Systems exist