CS4101 Introduction to Embedded Systems

Design and Implementation

Prof. Chung-Ta King

Department of Computer Science

National Tsing Hua University, Taiwan

Materials from *Computers as Components: Principles of Embedded Computing System Design*, Wayne Wolf, Morgan Kaufman;
An Embedded Software Primer, David E. Simon, Addison Wesley

Recap

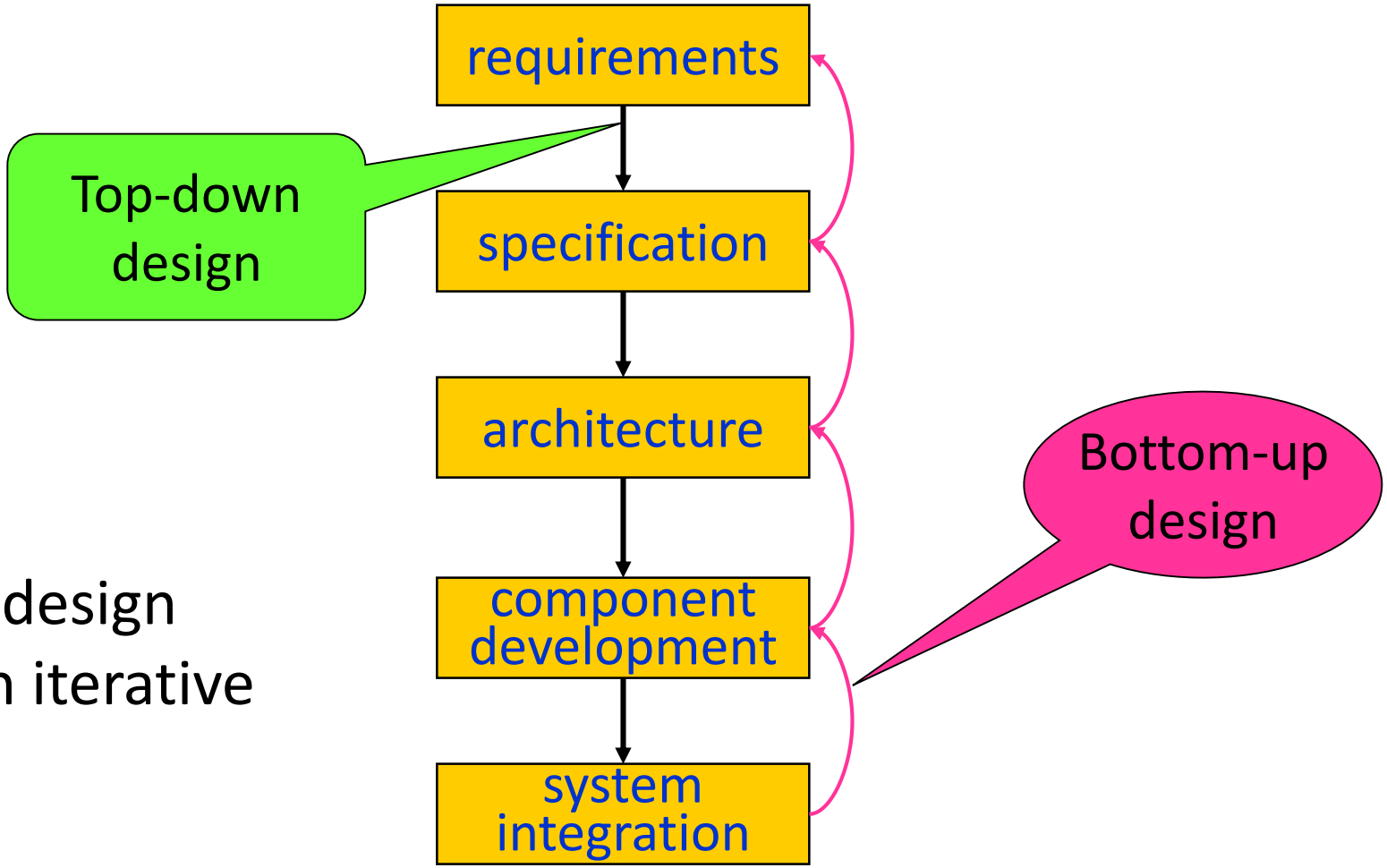
- More and more physical things will be augmented or embedded with computing
  - Things become “smarter”
  - Computing becomes ubiquitous
- An embedded system is a system that is embedded with programmable computers for specific applications of that system
- Why embedded systems?
  - Faster, more flexible development at lower cost
  - More complex functionalities

Suppose You Have a Product Idea

- To develop a device that monitors the temperature of the interior of a container
  - Monitor the temperature
  - If the temperature rises above a threshold, sound an alarm and notify the backend server

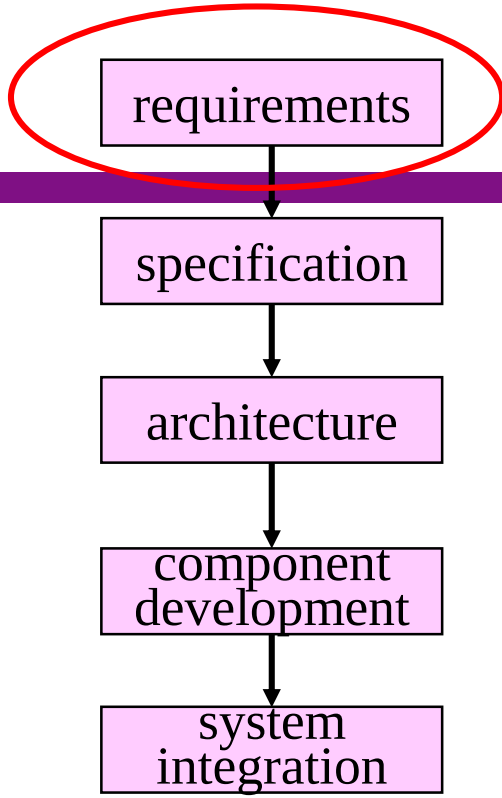
How to start from here?

Typical Design Flow



Real design often iterative

Requirements



- Plain language description of what the user wants and expects to get
  - e.g. *to develop a device that monitors the temperature of containers*
  - Describe how the end product is used by the user
- May be developed in several ways:
  - talking directly to customers
  - talking to marketing representatives
  - providing prototypes to users for comment

Requirements

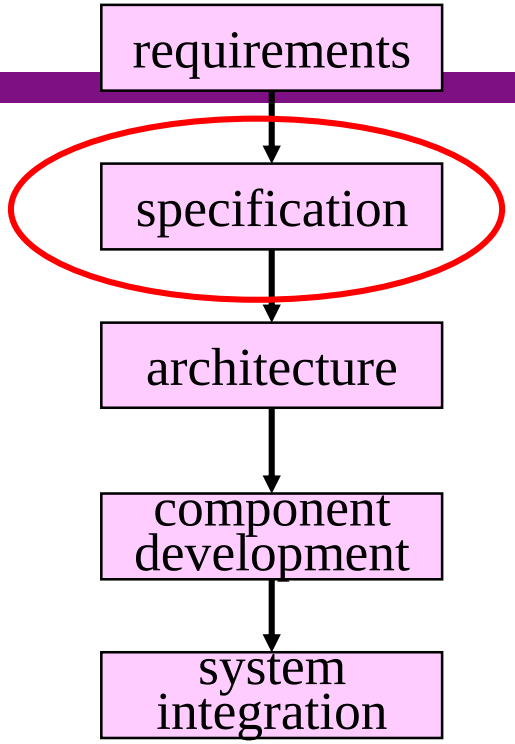
- Functional requirements:
  Internals as a black box and describe only the outputs as a function of input;
  - *Sound an alarm when temperature rises above a threshold*
  - *Notify backend server when temperature rises above a threshold and drops below the threshold*
  - *Buttons to reset the device*
- Non-functional requirements:
  - Performance, reliability, etc.
  - Size, weight, etc.
  - Power consumption
  - Cost
  - …

Describe "WHAT", not "HOW"

Requirements Form

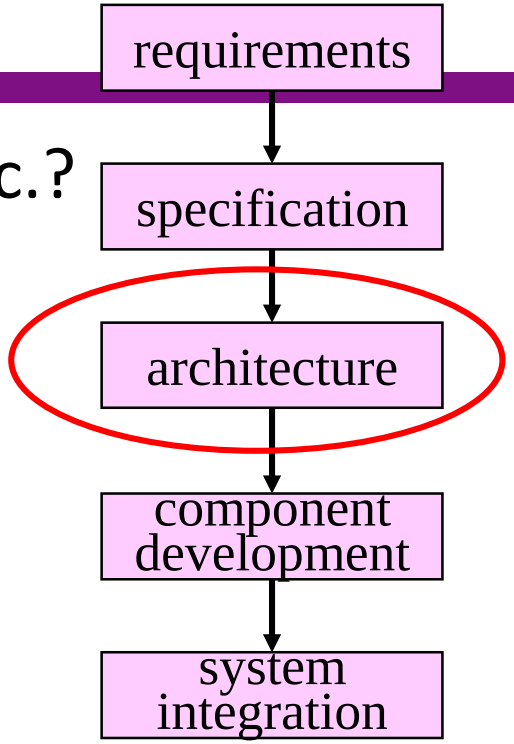
Name	Container thermometer
Purpose	Monitor temperature of containers
Inputs	Reset
Outputs	LED alarm, serial port
Performance	Response time < 1 sec
Manufacturing cost	$20
Power	100 mW
Physical size/weight	< 2” x 2”, 4 g

Specification



- More precise, usually quantitative description of the system:
  - Should not imply a particular architecture
  - List assumptions
  - e.g., *normal temp. monitoring: every 5 min above threshold: every 5 sec*
- May include functional and non-functional elements
- May be executable or may be in mathematical form for proofs
  - e.g. UML (Unified Modeling Language)

Architecture Design



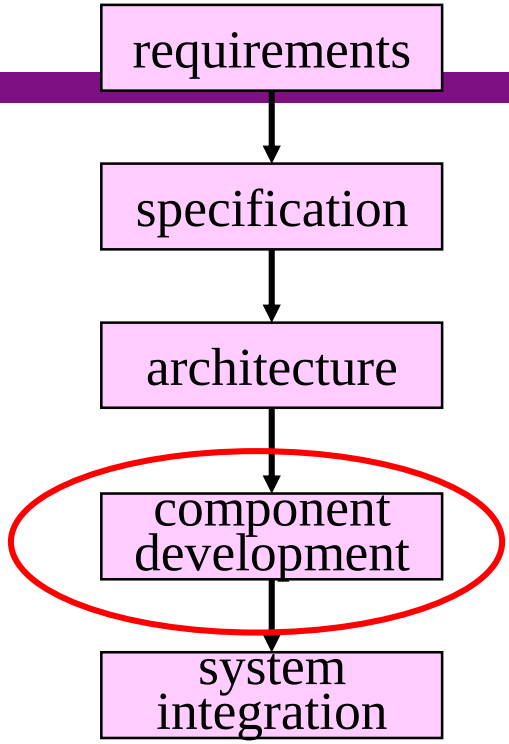
- What major components satisfy the spec.?
  - Need to know what are available
  - Hardware/software partition
- Hardware components:
  - CPUs, peripherals, etc.
  - e.g. *MSP430 CPU, thermometer*
- Software components:
  - Major programs and their operations
  - e.g. *no OS, thermometer driver, PC driver*
- Must take into account functional and non-functional specifications

Design Considerations

- Environment which the embedded system is in
- External and internal stimulus sources that interact with the embedded system ← I/O
  - Actions and events caused by stimulus
  - Elements of the embedded system that could be affected by the stimulus
- Desired system responses to the stimulus, which reflects one or more system requirements ← algorithm/workflow
- How the system responses can be measures

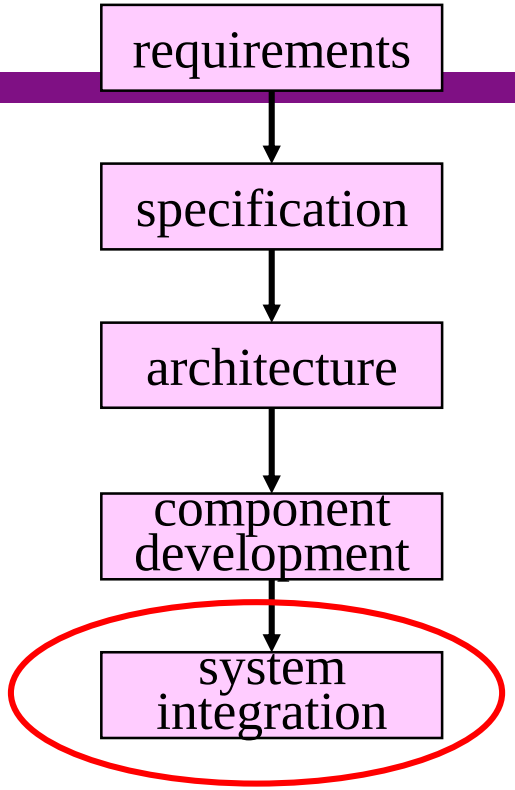
Component Development

- Actual implementation of individual hardware and software components
  - Must spend time architecting the system before you start coding
- Some components are ready-made, some can be modified from existing designs, others are to be designed from scratch
  - e.g. *MSP430 CPU, thermometer*
- Good surveys help



System Integration

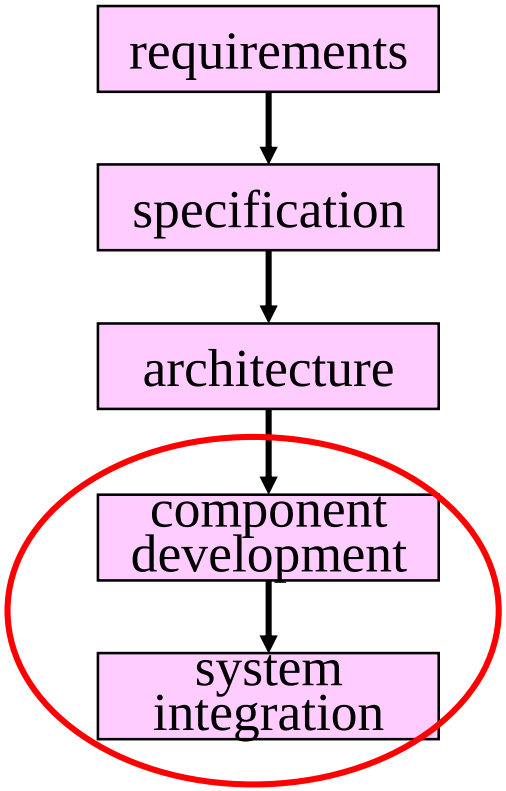
- Put together the components
  - Many bugs appear only at this stage
  - Require good interface definition from the start
- Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible → *test and verification*



System Development

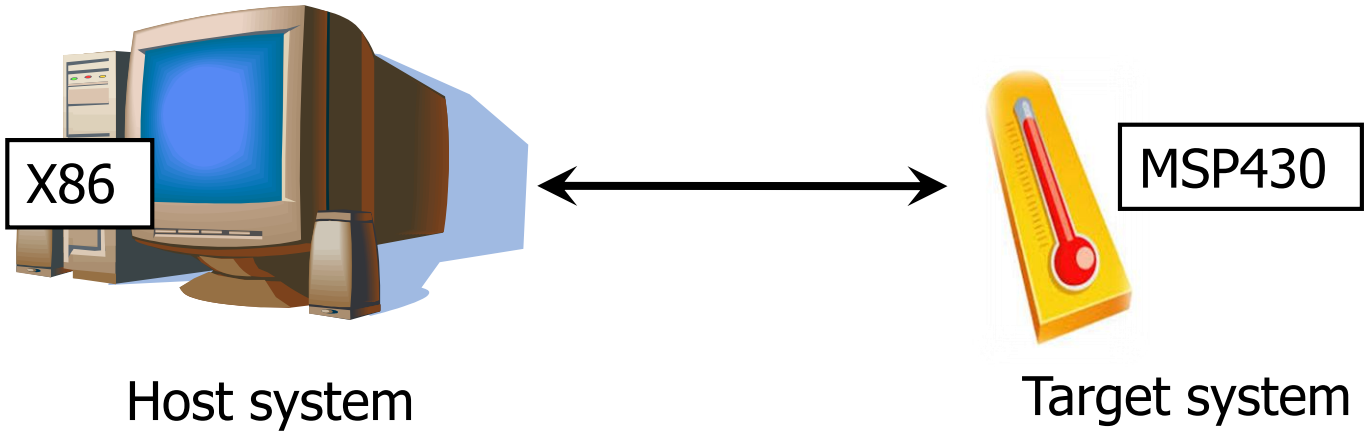
- Now that you have better idea of the requirements, specifications, and architecture of the container thermometer

- How to proceed to develop the components and integrate the system?
  - Real hardware?
    Programming environment?



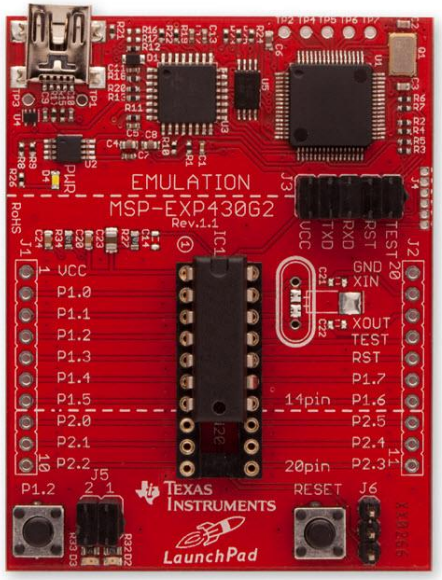
Development Environment

- Host: a computer running programming tools for development of the programs
- Target: the HW on which code will run
- After program is written, compiled, assembled and linked, it is transferred to the target



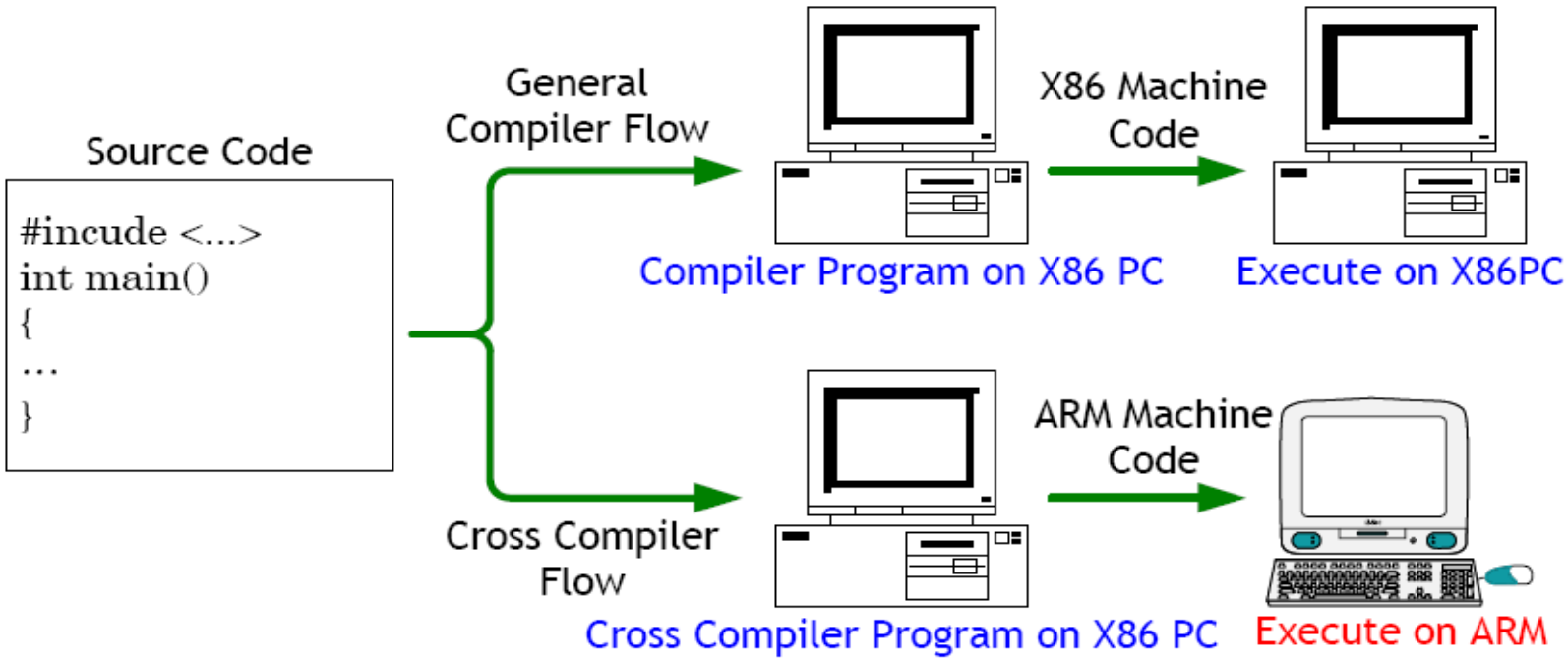
What If Real HW Not Available?

- Development board:
  - Before real hardware is built, software can be developed and tested using development boards
  - Development boards usually have the same CPU as the end product and provide many IO peripherals for the developed software to use as if it were running on the real end product
- Tools for program development
  - *Integrated Development Environment* (IDE): cross compiler, linker, loader, …
  - OS and related libraries and packages



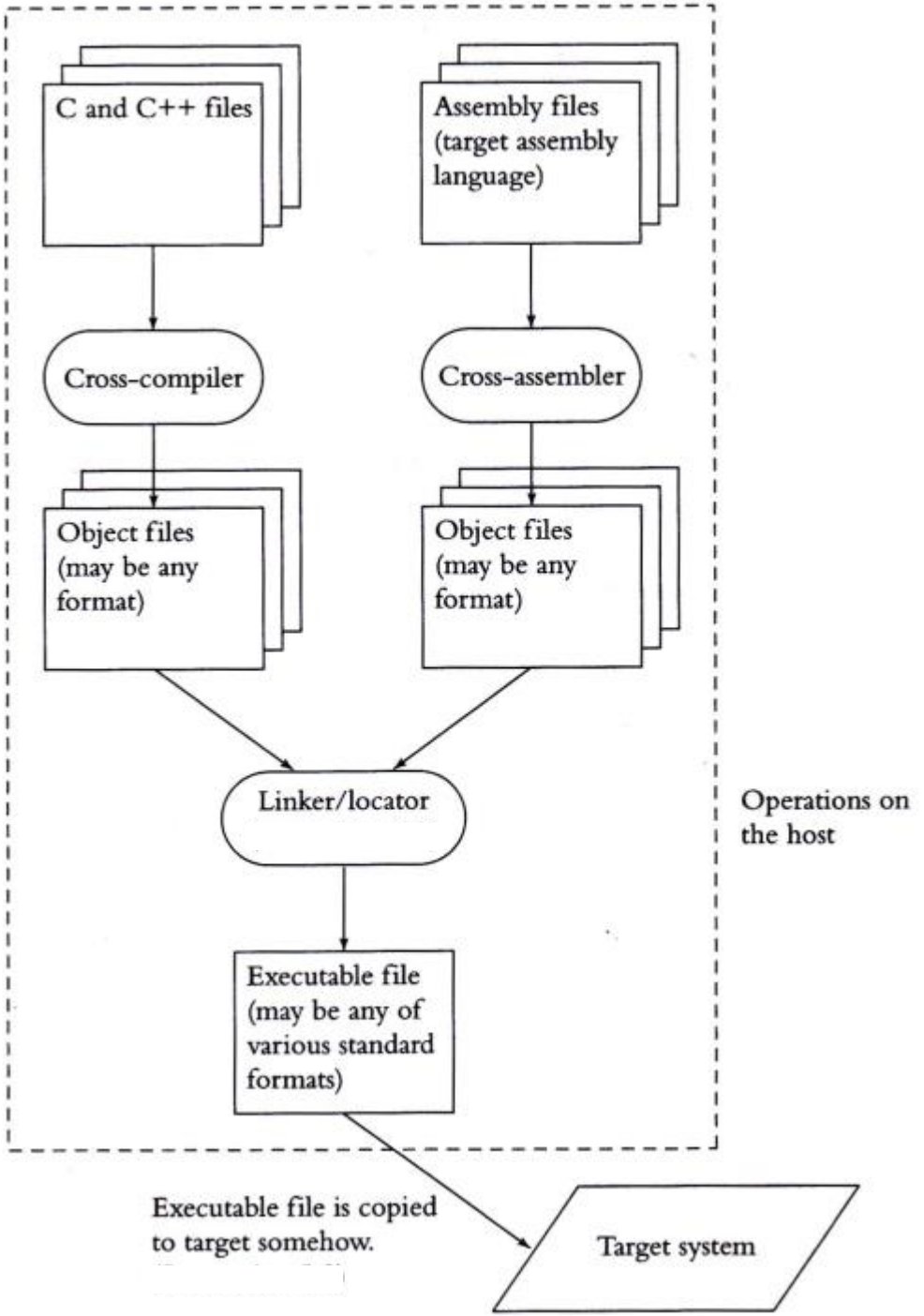
Cross Compiler

- Runs on host but generates code for target
  - Target usually have different architecture from host. Hence compiler on host has to produce binary instructions that will be understood by target



Development Process

- Process for creating instructions that are built on host but meant for the target
- Tools are compatible with each other
  → a *toolchain*
  - Binutils: as, ld
  - Glibc：C runtime Lib
  - GCC：C/C++ compiler



Linker/Locators

- For computers:
  - *Linker*: creates an image file to be run on host
  - *Loader*: loads image file into memory during run-time
- For embedded systems:
  - Locater: creates a file, containing binary image or other format, that will be copied onto target, which run on its own (not through loader)
  - It needs exact addresses beforehand
- Certain parts of program need in ROM and some in RAM
  - Normally done by dividing program in segments
  - Locator needs to be told where in memory to place segments

Summary

- Development of a system usually involves:
  - Requirement, specification, architecture design, component development, system integration, test and validation
- Development environment of an embedded system often includes
  - Development host with toolchain: cross compiler, linker/loader, library, emulator
  - Development board