Embedded Systems (12)

- Will start at 15:10
- PDF of this slide is available via ScombZ

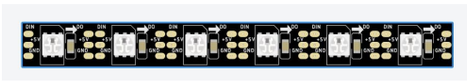
Hiroki Sato <i048219@shibaura-it.ac.jp>

15:10-16:50 on Wednesday

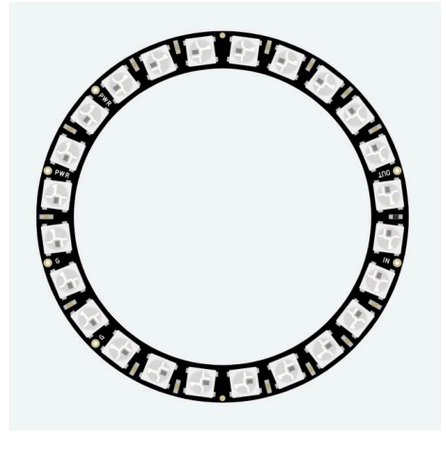
Targets At a Glance

- **What you will learn today**
  - Advanced topics: operating system and systems with multiple processors.
  - Preparation for the end-term project
- **Today's Project**
  - Try a real Arduino Uno development board.

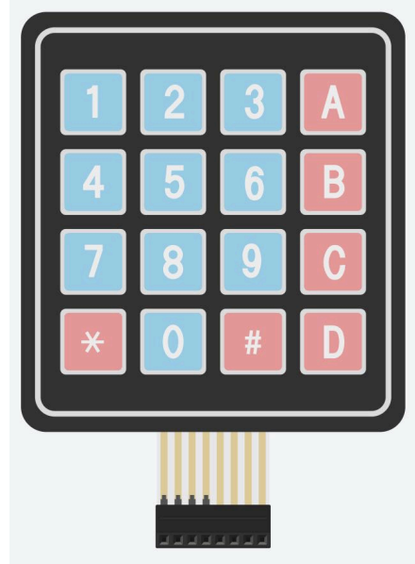
End Term Project



- **Design an embedded system**
  - Your design on TinkerCAD and your report will be evaluated.
- **3 mandatory requirements:**
  - input-process-output model,
  - communication between two or more boards,
  - and model-based development methodology (state machine model)
- **The report must include how you designed it.** (e.g. state transition diagrams should be shown)



LED array



Key pad

End Term Project

- **Schedule**

Dec 13	Dec 20	Jan 10	Jan 17	Jan 20
Day 1	Day 2	Day 3	Day 4	Submission due

Start to plan

- **You must write and submit a report (in PDF) and the design on TinkerCAD by 23:59 on January 20th (JST).**

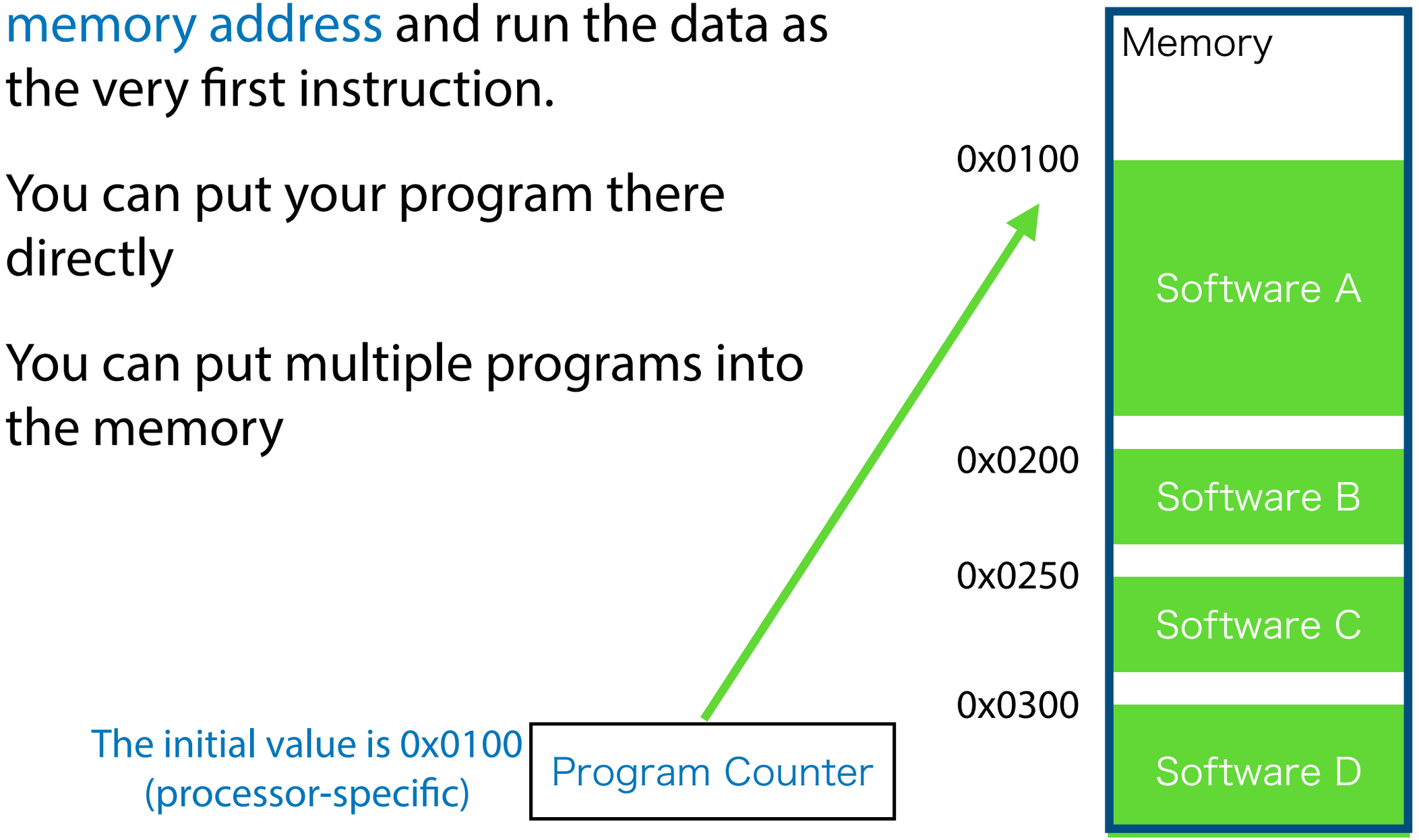
- **You will receive a real development board today. If your design works on it, it will gives extra score.**

- **Evaluation**
  - 80% for the end-term project and 20% for the exercises.
- **Questions**
  - Ask the teaching assistant during class hours or email the instructor.

Operating System

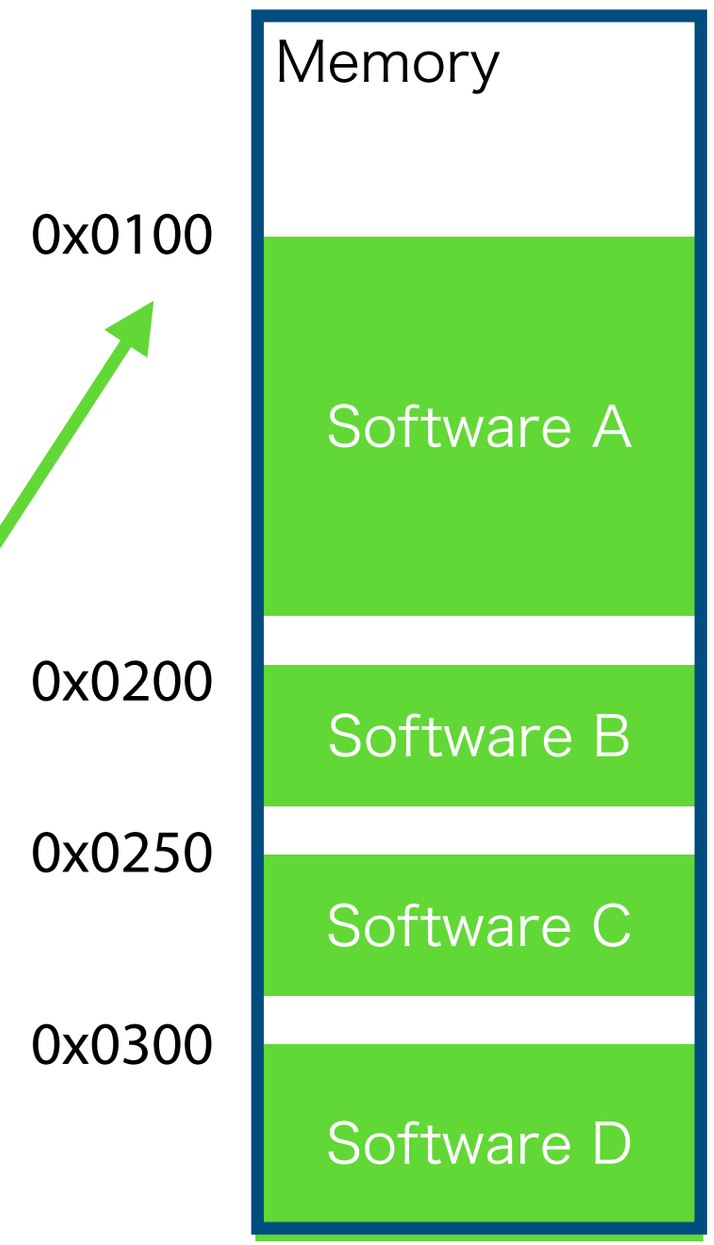
When Turning on the System

- The processor will read the specific memory address and run the data as the very first instruction.
- You can put your program there directly
- You can put multiple programs into the memory



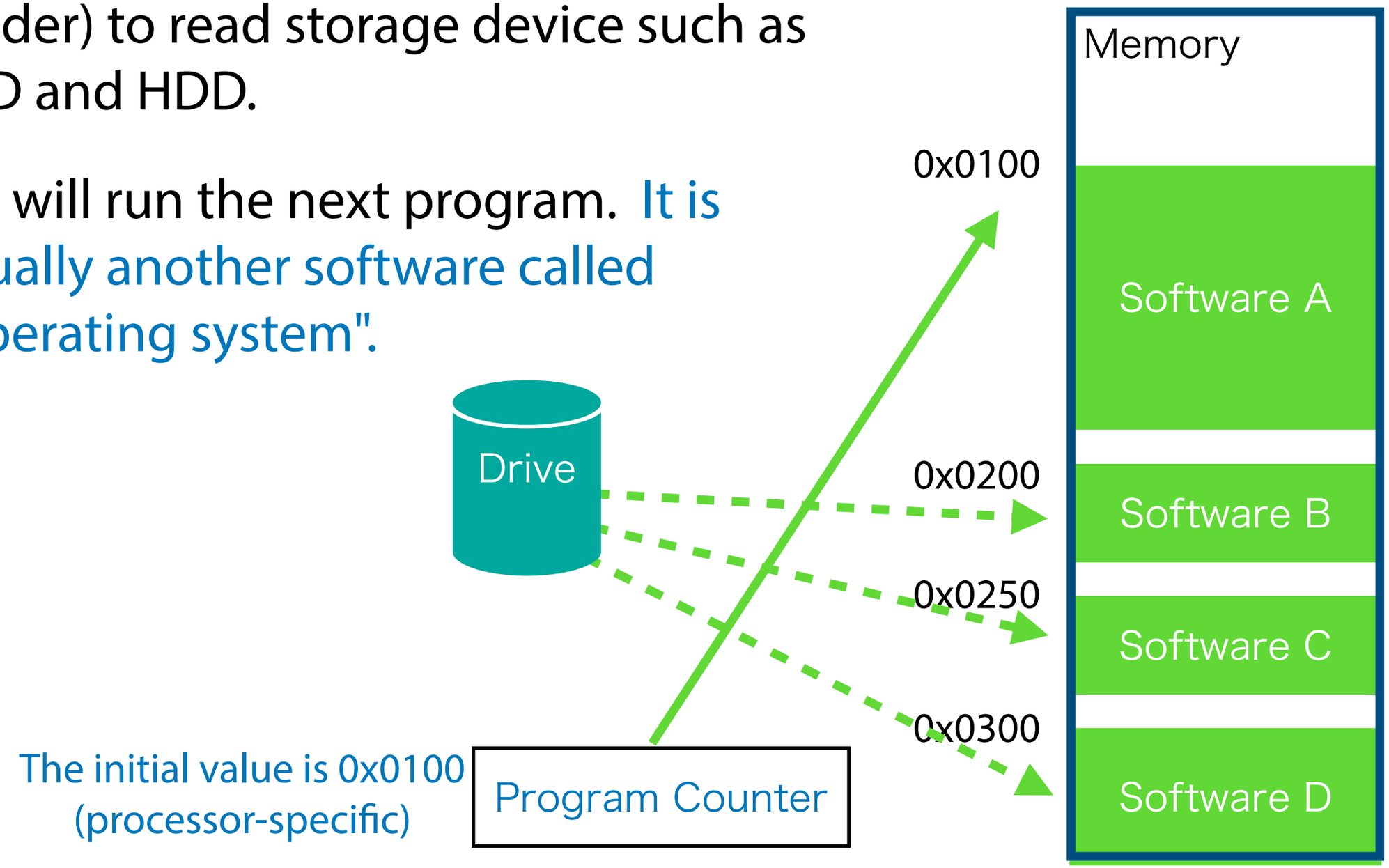
Execution Context Management

- You can have two or more programs at the same time, but you can run only one of them.
- Software A is responsible for switching the context to other software.
- The problem is you need to write a program to do it.
  - Duplicated efforts



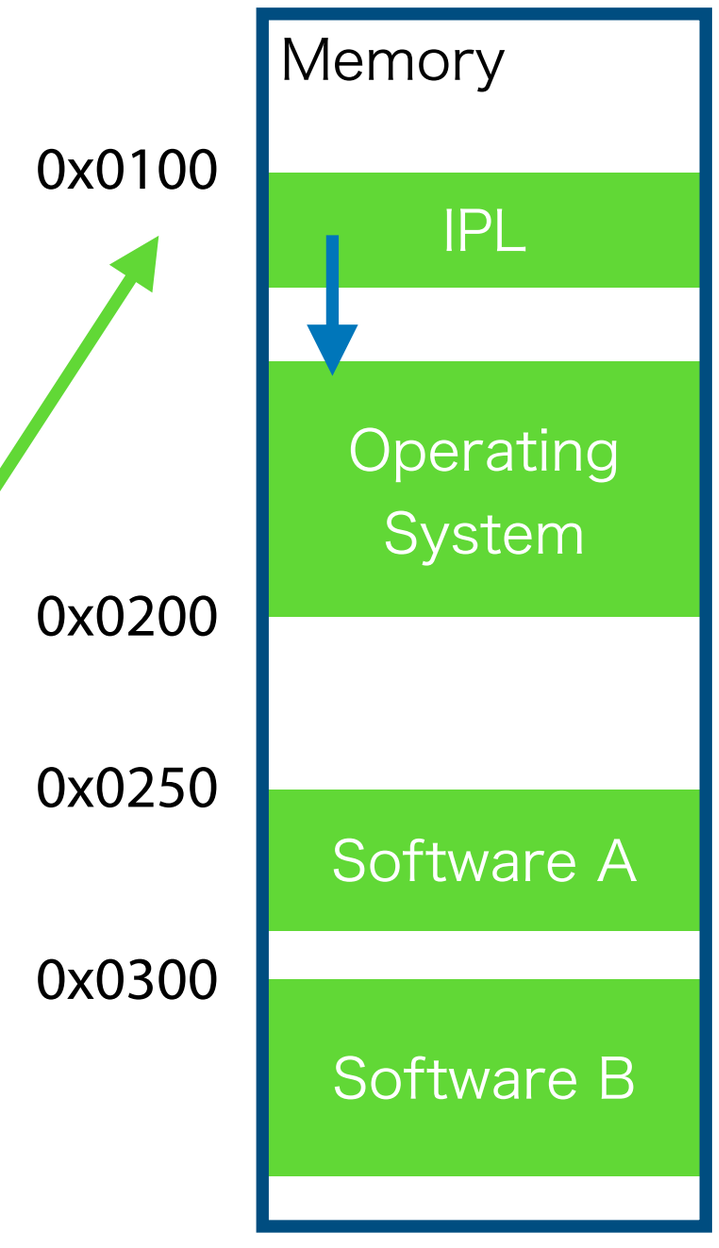
On Your Laptop/Smartphone

- Software A is IPL (initial program loader) to read storage device such as SSD and HDD.
- IPL will run the next program. It is usually another software called "operating system".



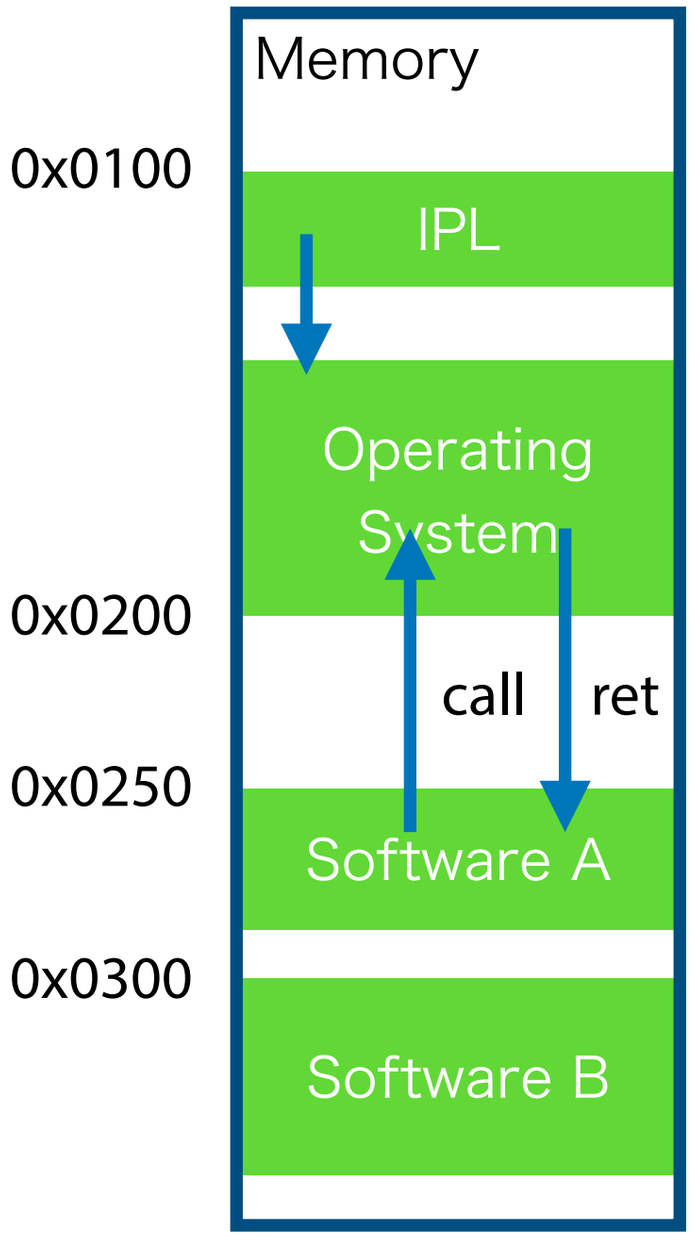
Operating System

- Software to manage other software
  - It provides a way to run Software A showing icons you can click on the screen, for example.
  - A and B are no longer need to take care of the execution context management.



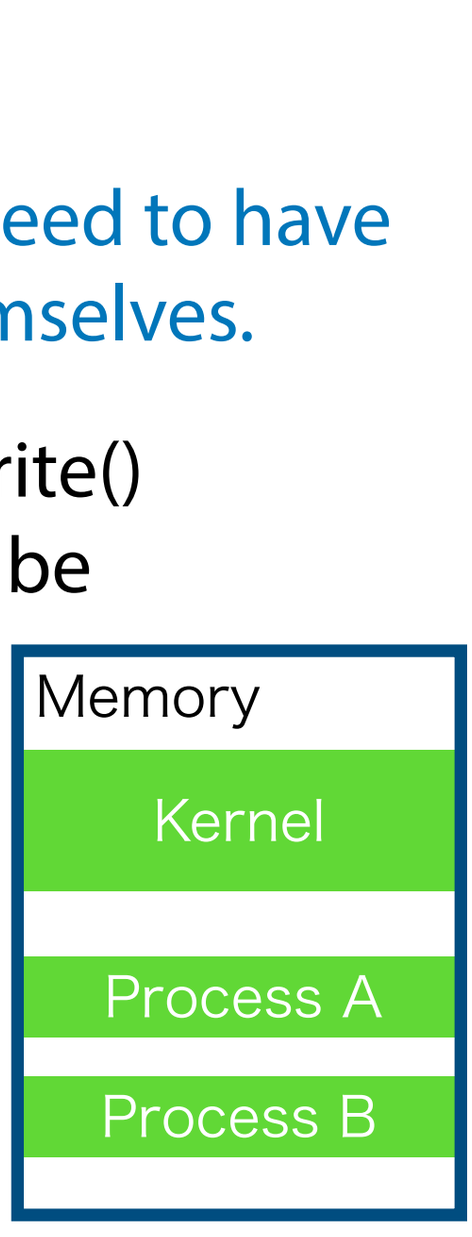
Operating System

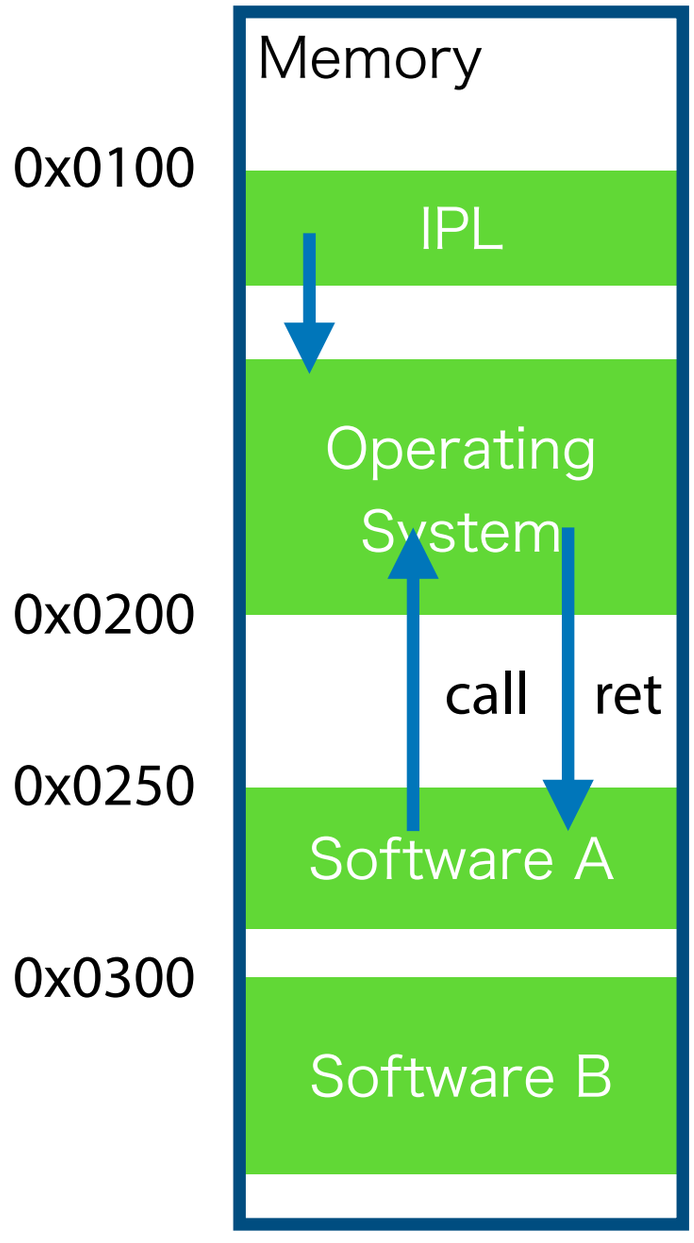
- Another aspect of OS: collection of routines that software can use like a library.
- Software A and B do not need to have them as library inside themselves.
- Duplication of digitalWrite() function in A and B can be eliminated.



Operating System

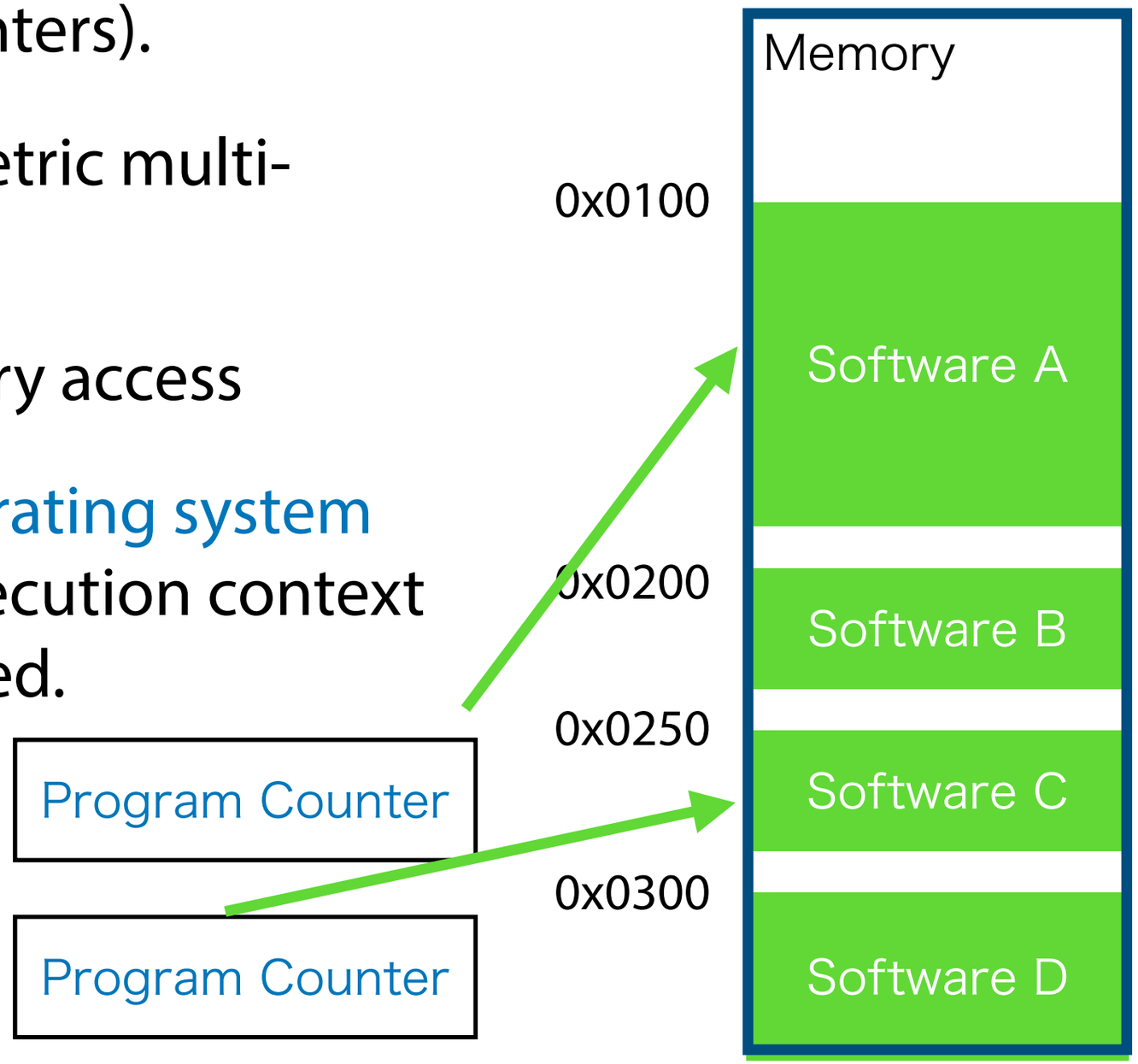
- Another aspect of OS: collection of routines that software can use like a library.
- Software A and B do not need to have them as library inside themselves.
  - Duplication of digitalWrite() function in A and B can be eliminated.
- Kernel-Process model





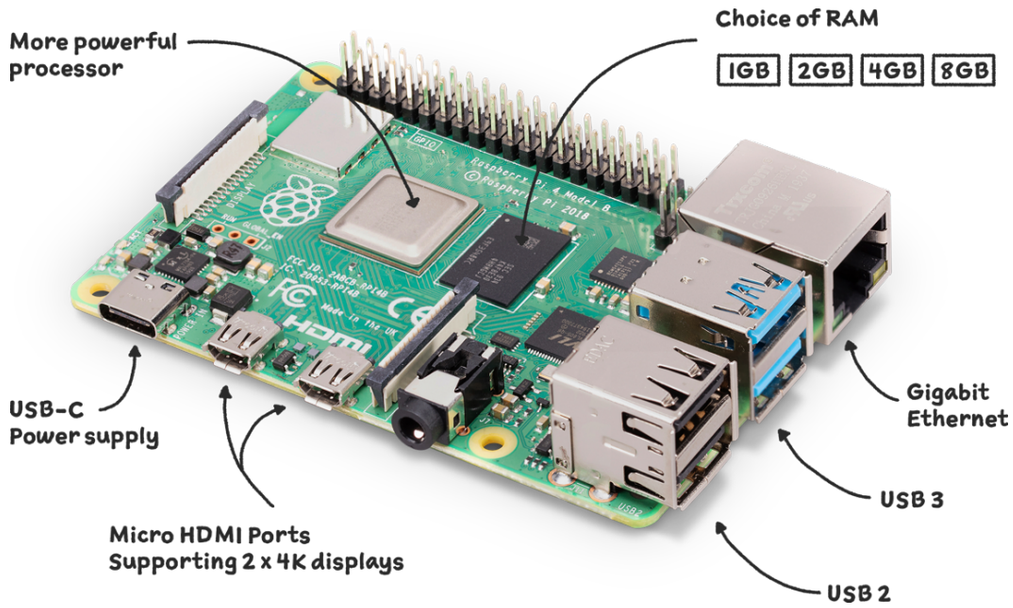
Multiprocessor

- A system with multiple processors (i.e. multiple program counters).
- Symmetric v.s. asymmetric multi-processing (SMP/AMP)
  - Difference in memory access
- Usually require an operating system because a complex execution context management is required.



Multiprocessor

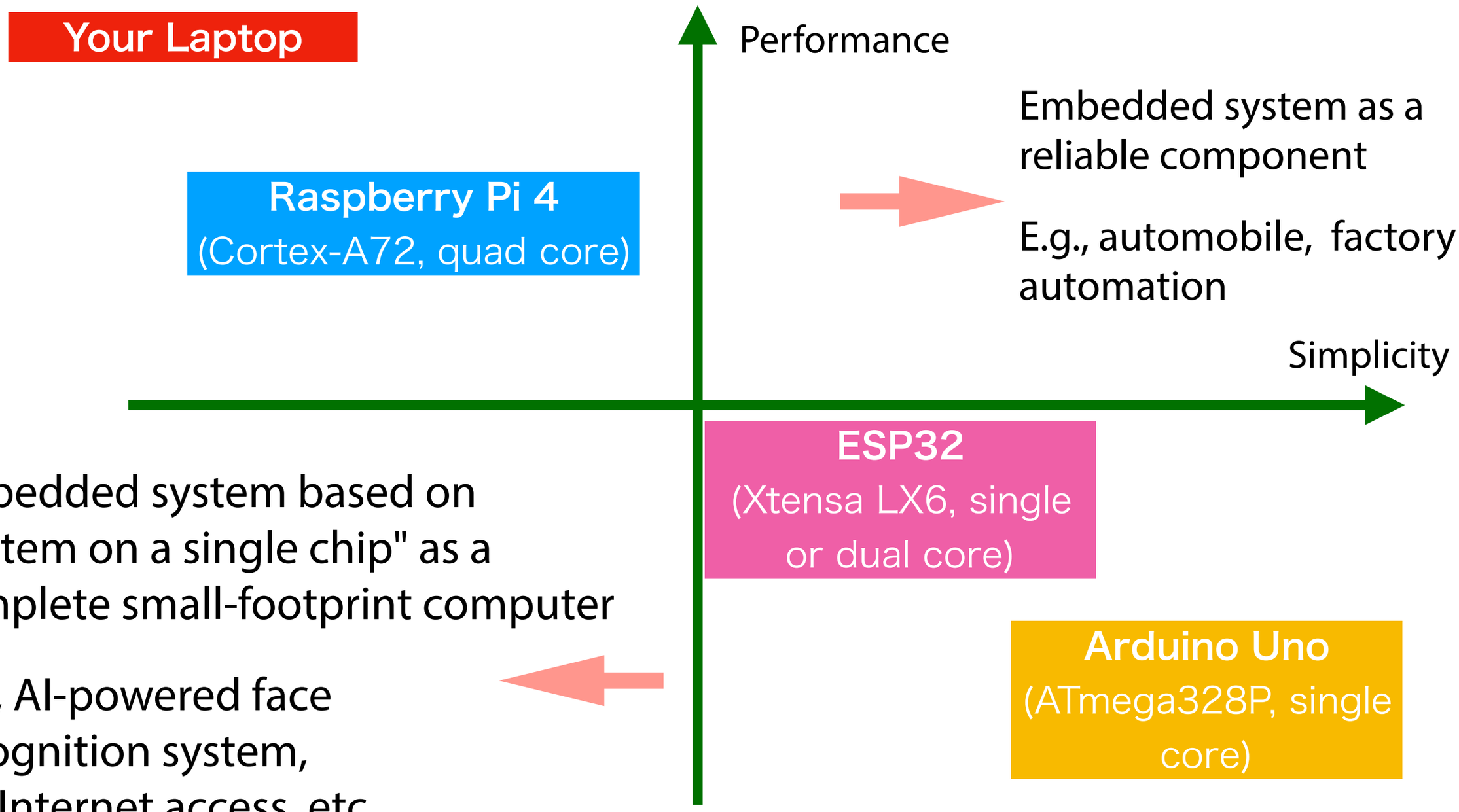
- Few embedded systems use multiprocessing because:
  - **Single processor is sufficient for the purpose.** Difficult to get a good reliability and realtime-ness in a multiprocessor system.
  - **Two or more processors can be used separately.** Remember exercise of communication using two boards. It might be a simpler and controllable system.
  - **Extra complexities are required**: you need an operating system or need to write an equivalent.
- What is the advantages?
  - Better performance!



Raspberry Pi 4
(Cortex-A72, quad core)

- Quad Core Cortex-A72 ARMv8 64-bit processor, 1.8GHz
- 8GB SDRAM, SDcard
- USB, GbE, Wifi, HDMI
- GPIO 40pin

Two Different Directions



Operating Systems

FreeRTOS is one of the most popular OSes in embedded systems. Freely available. Amazon owns it now.

Linux is a POSIX-compliant OS for PC (general-purpose computer). It can be used on powerful systems like RPi. Feature-rich and good at Internet access. Freely available.

FreeBSD is another POSIX-compliant OS originally developed for large-scale computers and ported to GP computers. One of the foundations of Apple macOS/iOS and the professor has been working on this: https://freebsdfoundation.org/about-us/board-of-directors/

Try Real Hardware

Preparation

- Break up into groups of 2.
  - Not for a group work but due to limitation of the number of development boards.
- Go to the teaching assistant and receive a component box, a keypad, and additional development board today.
  - Return them on January 17th in this room.

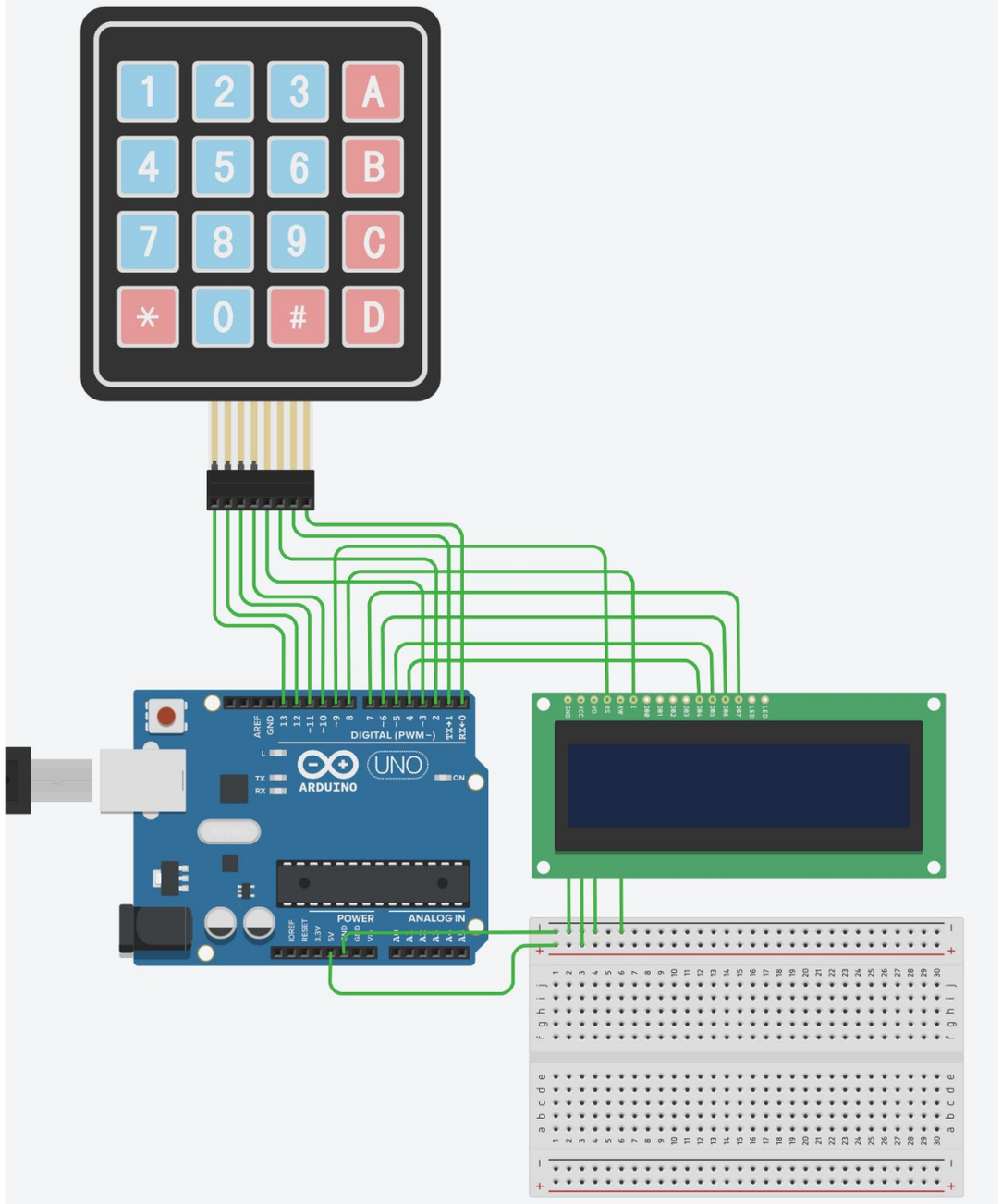
Preparation

- Connect the board to your laptop. If you have a question about this preparation work, contact the teaching assistant.
- Install Arduino IDE
  - https://www.arduino.cc/en/software
  - Windows / Linux / macOS
- Read the tutorial. The user interface is similar to TinkerCAD.
  - https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2
- Try a LED blinker (the board has LED at GPIO pin13).

Preparation

- Discuss how to use the components with your partner
  - Send an email if you have a question
- Try more complex programs using extra components and/or exercises in the last week

Time for Your Project



- Try to implement a system that shows pressed keys as characters on LCD, **a) with a state machine and then b) without a state machine.**
- Note that a) is already shown in the previous pages. Try to understand it.

Time for Your Project

- If you finish a) and b), try implementing a c)keypad+LCD+blinker that has a blinking "*" on the first line in addition to the original keypad+LCD function. The blinker can be added using the timer interrupts.
- if you finish keypad+LCD+blinker, try to implement a d)calculator based on it:
  - 'B' for addition, 'C' for subtraction, 'D' to get the result, 'A' to start over.
  - Design the state transition diagram first, and then implement it.
- Try a vending machine using multiple state machines.