Class 1

CSC 590 ARM Architecture and Security Introduction

Dr. Si Chen (schen@wcupa.edu)

Topics that we focused

- This semester, we will study **AI security, IoT security, and software security** to understand and mitigate the vulnerabilities inherent in these technologies, ensuring a safer and more robust integration of AI and IoT systems, particularly in light of growing threats and the adoption of Legal Lifecycle Management (LLM) processes.
  - **AI Security**
  - **IoT Security.**
  - **Software Security**

REVERSE ENGINEERING ARM

Why learning AI Security

1. **Expanding Attack Surface**: As AI systems are integrated into various aspects of daily life and critical infrastructures, they become appealing targets for cyber attacks.
2. **Data Integrity**: AI systems are highly dependent on data. Manipulating this data (data poisoning, adversarial attacks) can lead to incorrect AI decisions, making the understanding of AI security vital.
3. **Ethical and Legal Concerns**: With the integration of AI in decision-making processes, there's an increasing need to ensure these systems operate in a secure, unbiased, and legally compliant manner.
4. **Automated Threats**: As AI becomes smarter, so do AI-powered cyber-attacks. Understanding AI security helps in anticipating and countering these evolving threats.
5. **Interconnected Risks**: AI systems often interact with other technologies like IoT. A security flaw in an AI system could compromise the broader network it is part of, making AI security key for overall system integrity.

Why learning ARM security

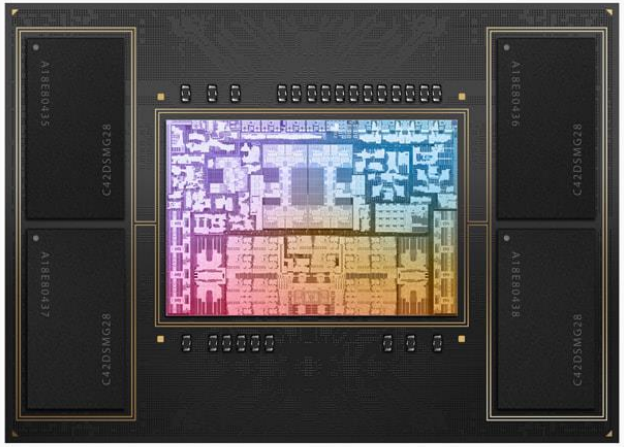
1. **Ubiquity of ARM Architecture**: ARM processors are used in a multitude of devices, ranging from smartphones to IoT devices and even servers. Understanding their internals is key for a comprehensive security perspective.
2. **Firmware Security**: Many vulnerabilities lie at the firmware level, and ARM reverse engineering can uncover such weaknesses, making it crucial for securing embedded systems.
3. **Supply Chain Assurance**: In an age where hardware components may come from various global sources, understanding the ARM architecture through reverse engineering can aid in supply chain security.
4. **Optimization**: Beyond security, reverse engineering ARM can aid in performance optimizations, which is crucial for resource-constrained devices.
5. **Competitive Analysis**: Companies may want to understand how competitors' ARM-based hardware works, both for competitive advantages and for ensuring compliance with intellectual property laws.

ARM Ltd

- Founded in November 1990
  - Spun out of Acorn Computers
- Designs the ARM range of RISC processor cores
- Licenses ARM core designs to semiconductor partners who fabricate and sell to their customers.
  - ARM does not fabricate silicon itself
- Also develop technologies to assist with the design-in of the ARM architecture
  - Software tools, boards, debug hardware, application software, bus architectures, peripherals etc

ARM Partnership Model

The new “mainstream”



Arm processors will become mainstream for Windows in 2024 – Qualcomm CEO

ARM PROCESSOR VS. INTEL PROCESSOR

- There are many differences between Intel and ARM, but the main difference is the instruction set.
  - Intel is a **CISC (Complex Instruction Set Computing)** processor that has a larger and more feature-rich instruction set and allows many complex instructions to access memory.
  - ARM is a **RISC (Reduced instruction set Computing)** processor and therefore has a simplified instruction set (100 instructions or less) and more general purpose registers than CISC.

Intel proposes the x86-S architecture for a simpler, more efficient CPU instruction set

It's finally the end of the x86-compatible PC as we know it.

By **Alfonso Maruccia** May 22, 2023 at 7:37 AM | 16 comments

Architectures and Profiles

- Dozens of different processors exist across the Arm ecosystem, each with different features, performance, power consumption, and other characteristics.
- To provide consistency across these processors and allow existing compiled applications to run on new processors when they are released, each processor in the Arm ecosystem conforms to an ***architecture*** and a ***profile***.
- **The architecture** specifies the supported instruction sets, the available set of registers, and the different levels of privilege as part of the exception model, programmer's model, and memory model of the system. (e.g., Cortex-A32, Cortex-A35, Cortex-A72, Cortex-A65, Cortex-A78)
- When you reverse engineer an executable from an Arm-based device, there are two things you need to determine before you start digging through the assembly code.
    - *What is the micro-architecture of the processor?*
    - *Which architecture does the processor implement?*

ARM family	ARM architecture
ARM7	ARM v4
ARM9	ARM v5
ARM11	ARM v6
Cortex-A	ARM v7-A
Cortex-R	ARM v7-R
Cortex-M	ARM v7-M

Architectures and Profiles

- Another important distinction is the **profile**. The name of the processor core often already reveals the specific profile, e.g., Cortex-A72 for the A-profile, Cortex-R82 for the R-profile, and so on.
- Within the Arm-v8 architecture, there are three profiles - **A, R, and M**:
  - **A:** This is the “application” profile (Armv8-A). Armv8-A is designed for rich operating systems found in devices such as mobile phones, IoT devices, laptops, and servers.
  - **R:** This is the “real-time” profile (Armv8-R), which also supports the AArch64 and AArch32 execution states. Armv8-R is designed for hard real-time or safety-critical systems, such as medical devices, avionics, and electronic brakes in vehicles. R-profile processors run 32-bit code and support a much more limited memory architecture compared to the A-profile.
  - **M:** This is the “microcontroller” profile (Armv8-M). Armv8-M is designed for use as a microcontroller in low-cost deeply embedded systems, such as industrial devices and some low-cost IoT devices. Armv8-M only runs 32-bit programs using the Thumb instruction set.

Architectures and Profiles

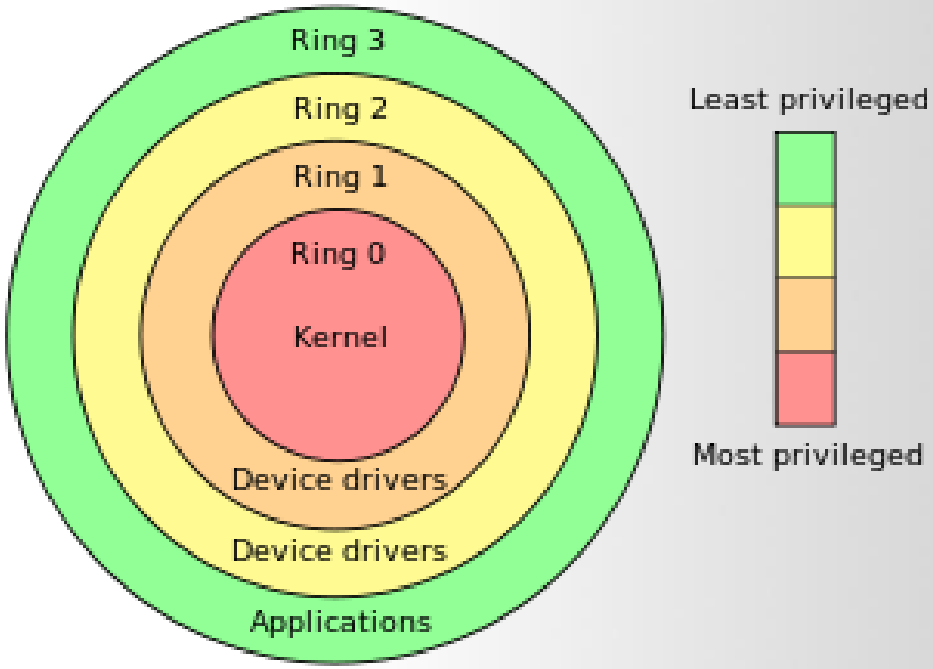
- In this class, we will focus on the Armv8 processors based on the **A-profile**, which is the profile of the main CPU in modern devices running a rich OS like Android.
- But it is still important to remember that **modern smartphones contain all three processor types**.
  - For example, R-profile processors provide cellular connectivity, and M-profile processors are used in camera components, power management, touchscreen and sensor hubs, Bluetooth, GPS, and the Flash controller. For SIM or smart cards, an M-profile processor with additional security features, called ***SecureCore***, is used.

What's the architecture and profile of Apple M1?

AArch64 and AArch32

- The ARMv8 instruction set architecture includes two execution states: a 32-bit execution state (AArch32) and a 64-bit execution state (AArch64).
- These two execution states can run 32-bit and 64-bit applications, respectively.
- The 64-bit execution state is known as AArch64. In the context of many application developments, AArch64 is also referred to as the instruction set architecture, meaning the 64-bit ARM instruction set architecture.
- We'll learn both of them.

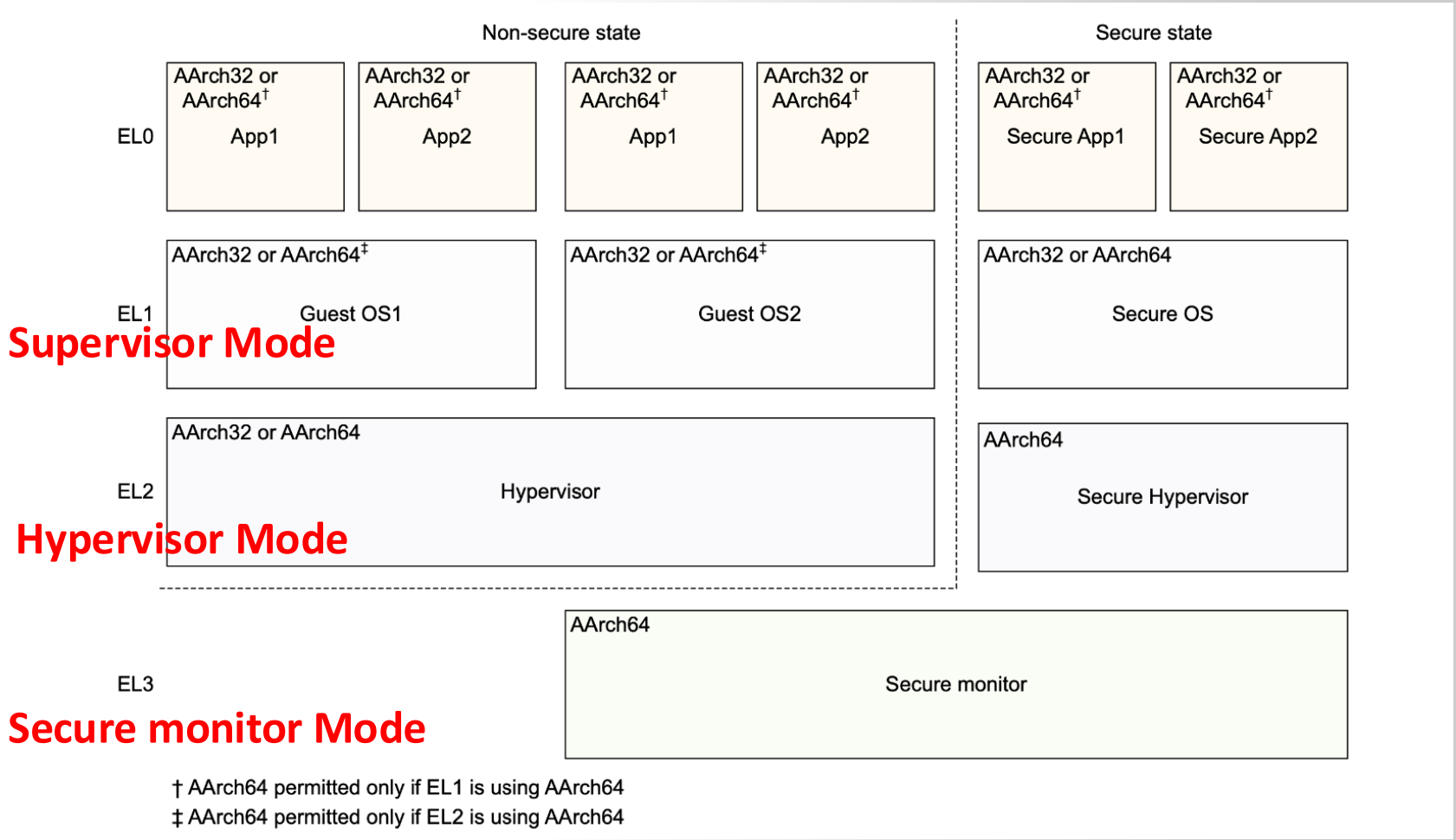
Privilege level (x86)



X86 OS

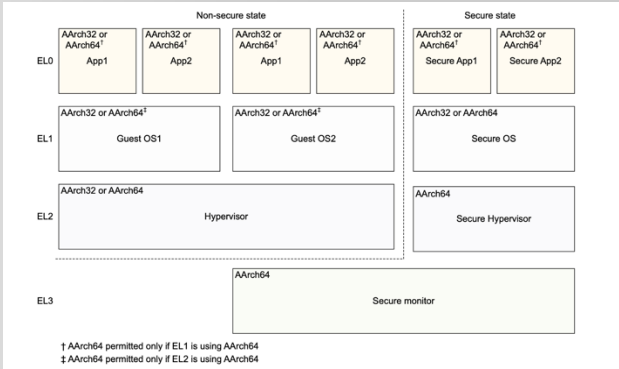
Privilege level (ARM)

- Exception Levels

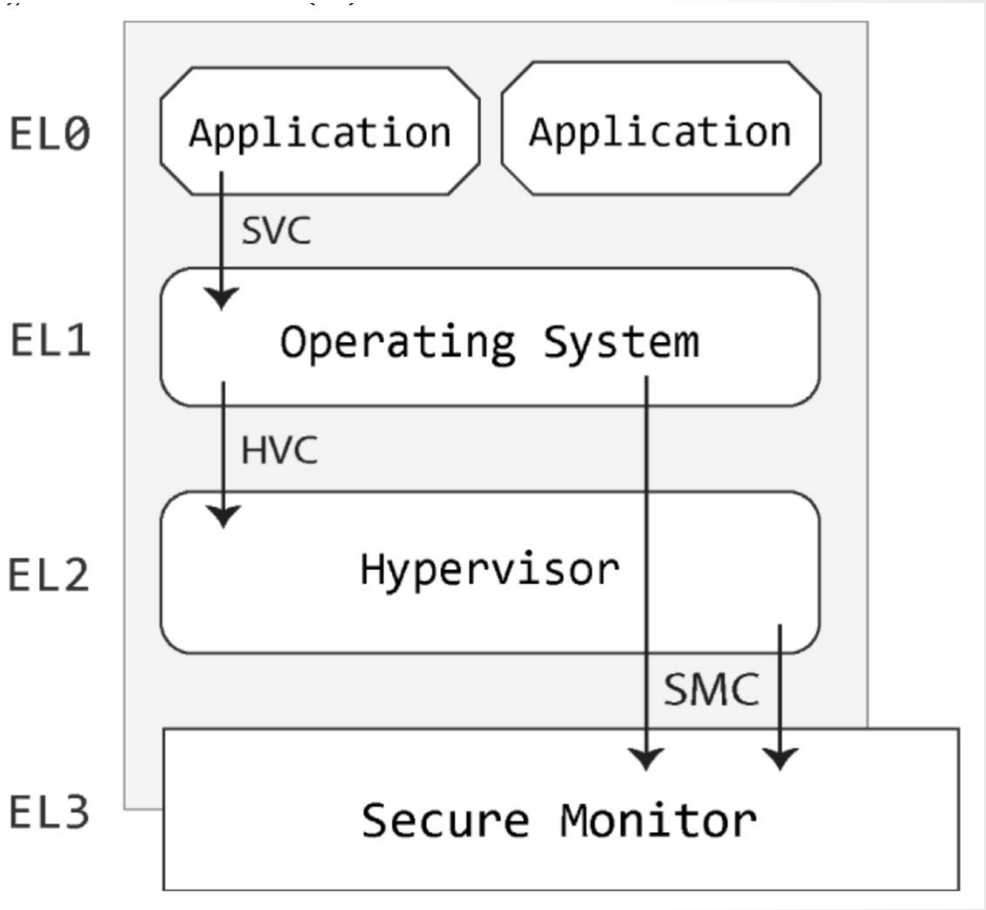


Exception Levels

- EL0 is the least privileged execution mode for a program and is used by ordinary user-mode applications. A program operating at EL0 can perform basic computation and access its own memory address space, but it cannot directly interact with device peripherals or system memory unless explicitly authorized to do so by software running in a higher exception level.
- EL1 is typically used by operating system kernels and device drivers, such as the Linux kernel.
- EL2 is typically used by hypervisors that manage one or more guest operating systems, such as KVM.
- EL3 is used by processors that make use of the TrustZone extension and supports switching between two security states through the Secure Monitor.



Exception Level Changes



Programs running at a given execution level run continuously at that level until the processor encounters an "exception." Exception types can be categorized into two types: **synchronous exceptions and asynchronous exceptions**.

Apple Silicon Hardware Secrets: SPRR and Guarded Exception Levels (GXF):https://blog.svenpeter.dev/posts/m1_sprr_gxf/

Byte Order in ARM

```c
#include <stdio.h>

int main() {
    long a = 0x0123456789abcdef;
    printf("Storing 0x0123456789abcdef into memory is: ");
    char *s = (char *)&a;
    for (int i = 0; i <sizeof(a); i++) {
        printf("%.2x ", s[i] & 0xff);
    }
    printf("\n");

    return 0;
}
```

Little endian

- IA-32 processors use "little endian" as their byte order. This means that the bytes of a word are numbered starting from the least significant byte and that the least significant bit starts of a word starts in the least significant byte.

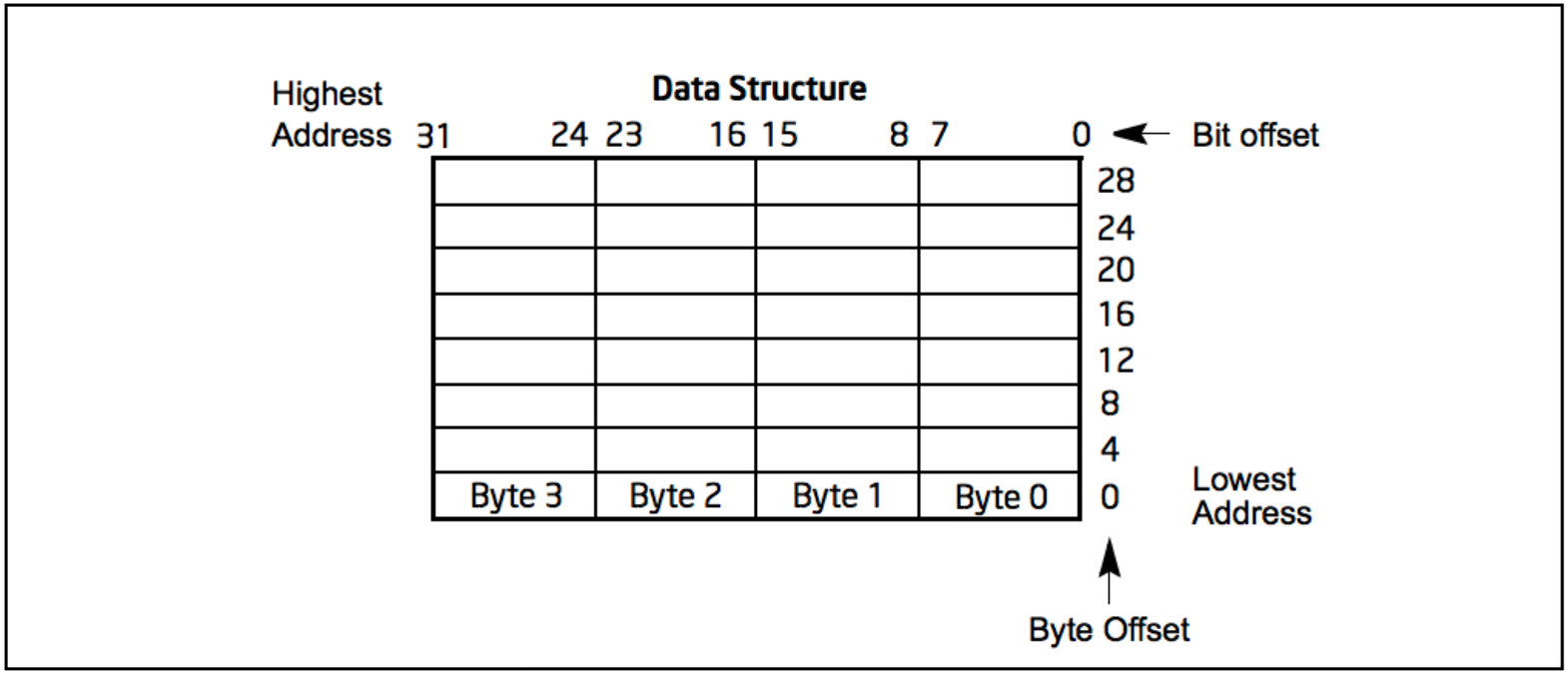
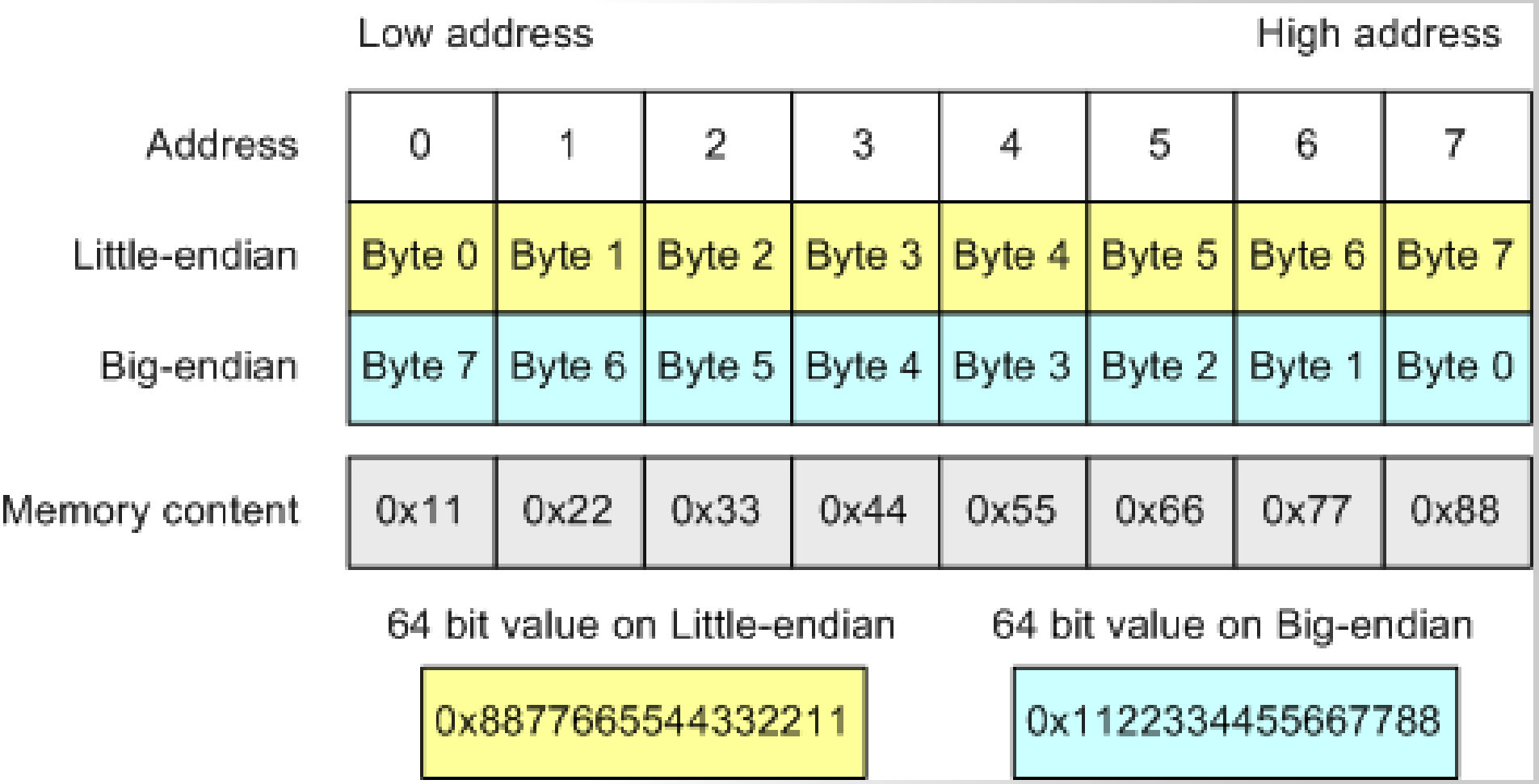


Figure 1-1. Bit and Byte Order

Byte Order


Low address
High address
Address
0
1
2
3
4
5
6
7
Little-endian
Byte 0
Byte 1
Byte 2
Byte 3
Byte 4
Byte 5
Byte 6
Byte 7
Big-endian
Byte 7
Byte 6
Byte 5
Byte 4
Byte 3
Byte 2
Byte 1
Byte 0
Memory content
0x11
0x22
0x33
0x44
0x55
0x66
0x77
0x88
64 bit value on Little-endian
0x887766554433221 1
64 bit value on Big-endian
0x1122334455667788

Byte Order

- Both Apple silicon and Intel-based Mac computers use the little-endian format for data, so you don’t need to make endian conversions in your code.
- In general, the ARM architecture was **little-endian before version 3**. Since then ARM processors became **BI-endian** and feature a setting which **allows for switchable endianness**.

Porting Your macOS Apps to Apple Silicon:
https://developer.apple.com/documentation/apple-silicon/porting-your-macos-apps-to-apple-silicon

Q & A