리틀 엔디안과 빅 엔디안

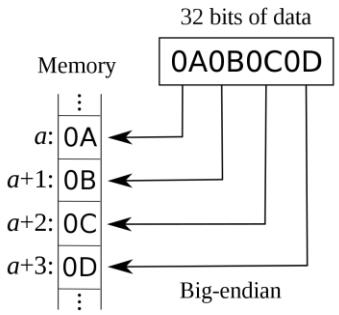
- 엔디언(Endianness)

컴퓨터 메모리와 같은 1차원 공간에 여러 개의 연속된 대상을 배열하는 방법을 뜻함.

바이트를 배열하는 방법을 **바이트 순서(Byte order)**라고 함.

참고 : https://ko.wikipedia.org/wiki/%EC%97%94%EB%94%94%EC%96%B8

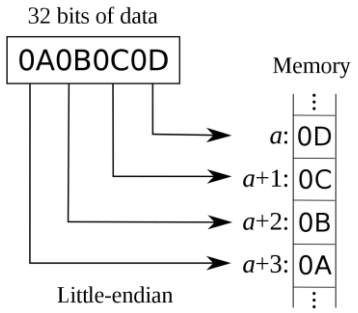
1. 빅 엔디안 : MSB(Most Signifcant Bit)가 가장 앞쪽에 저장되는 방식.



네트워크 주소를 빅엔디언으로 씀.

장점 : 사람이 숫자를 읽고 쓰는 방법과 같음. -> 디버깅이 편함.

2. 리틀 엔디안 : LSB(Least Significant Bit)가 가장 앞쪽에 저장되는 방식.



대부분의 x86 아키텍처를 사용하는 대부분의 데스크톱에서 사용.

장점 : 메모리에 저장된 값의 하위 바이트들만 사용할 때 별도의 계산이 필요 없음.

Ex) 32비트 숫자인 0x2A는 리틀 엔디언으로 표현하면 2A 00 00 00이 되는데, 이 표현에서 앞의 두 바이트 또는 한 바이트만 떼어 내면 하위 16비트 또는 8비트를 바로 얻을 수 있다.

반면 32비트 빅 엔디언 환경에서는 하위 16비트나 8비트 값을 얻기 위해서는 변수 주소에 2바이트 또는 3바이트를 더해야 한다.

3. 각 방식을 사용하는 곳들

Little Endian	Big Endian
Intel x86	IBM
AMD	SPARC
DEC	Motorola

출처 : https://jhnyang.tistory.com/226

ARM 프로세서들의 경우, 성능 향상을 목적으로 빅, 리틀 엔디안을 선택할 수 있도록 되어 있음.

4. 바이 엔디언

빅 엔디언, 리틀 엔디언 둘 중 하나를 선택할 수 있음. 이를 사용하는 아키텍쳐로는 ARM, PowerPC, DEC 알파, MIPS, PA-RISC, IA-64 등이 있음.

5. 미들 엔디언

한 방향으로 순서가 정해져 있는 게 아님.

이를테면 32비트 정수가 2바이트 단위로는 빅 엔디언이고 그 안에서 1바이트 단위로는 리틀 엔디언인 경우가 종종 있는데 이를 미들 엔디언(Middle-endian)이라 한다.

종류	0x1234의 표현	0x12345678의 표현
빅 엔디언	12 34	12 34 56 78
리틀 엔디언	34 12	78 56 34 12
미들 엔디언	-	34 12 78 56 또는 56 78 12 34

6. 게임 개발에서의 빅, 리틀 엔디안

엔디안은 보통 게임 차원에서 추상화 되기 때문에 신경쓸 필요는 없지만, 엔진에서 추상화 되어있지 않으면 수정해야 함.

그러나 C/C++ 멀티 플랫폼 엔진의 로우 레벨 부분에서 작업할때는 처리해야 할 가능성이 있음.

세 메이저 콘솔들이 빅 엔디안 기반의 PowerPC 아키텍처를 사용하는 반면, PC는 리틀 엔디안 기반의 x86 아키텍처를 사용함.

따라서, 데이터 구조(binary serialization, networking 등의)에 넣기 위해 어디로부터 raw byte(원시 바이트)를 읽는 작업을 수행하려면 엔디안 구조를 신경써야 함.

출처 : https://gamedev.stackexchange.com/questions/37712/little-and-big-endianness-in-games