03 Little Endian

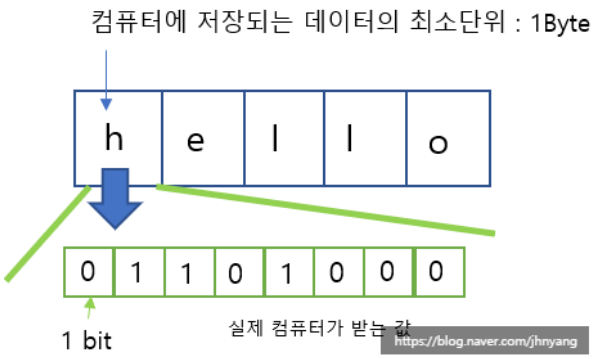
Endianness

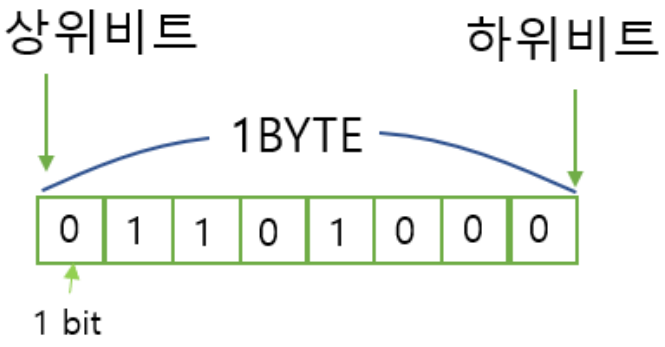
1. 개념

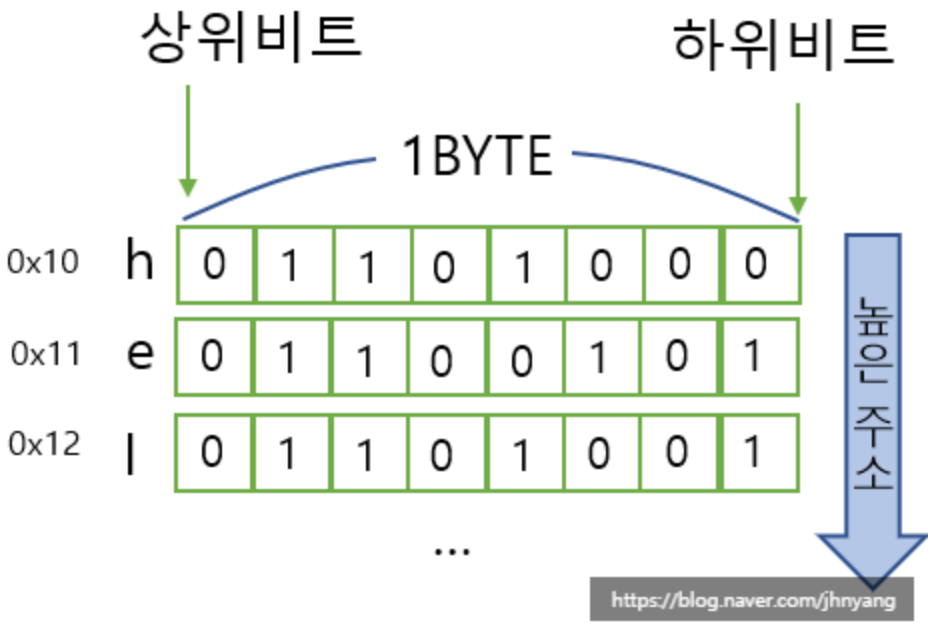
- 컴퓨터의 메모리와 같은 1차원의 공간에 여러 개의 연속된 대상을 배열하는 방법
  - Byte order : 바이트를 배열하는 방법

2. MSB & LSB

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







요렇게 사실 1Byte는 8개의 bit로 구성되어 있어요
근데 우리가 '안녕하세요' 반대로 읽는다 하면 '요세하녕안'으로 읽지,
'안'의 내부 구성단위까지 쪼개서 반대로 읽지는 않잖아요? 반대의 순서대로 읽는다고 했을 때 'ㅇ'+'ㅏ'+'ㄴ' 까지 거꾸로 해서
'ㄴ'+'ㅏ'+'ㅇ'이렇게 읽지는 않아요 ㅎㅎ
왜냐면 '안', 이걸 최소의 단위라고 생각하기 때문이죠!

컴퓨터도 마찬가지로 **1BYTE가 최소의 단위**이기 때문에 bit는 영향을 받지 않습니다.
다시 한번~ 바이트 단위로 끊어서 읽는 방향에 따라 저장 방식이 달라지므로 BYTE ORDER라고 합니다. 왼쪽부터 읽을수도 있고 오른쪽부터 읽을수도 있고~~
그런데 비트 단위에서 순서는 우리가 왼쪽에서 읽는 것처럼 읽고, 바이트 오더처럼 막 컴퓨터별로 상이하지 않습니다. 걍 왼쪽부터 읽어요 ㅎㅎ'안'을 굳이 쪼개서 거꾸로 읽지 않는거처럼~

3. Big / Little endian

1. Big - endian

   큰 단위가 앞에 나오는 빅 엔디안

- 최 상위 바이트(Most Significant Byte; MSB)부터 차례로 저장하는 방식
- 소프트웨어의 디버그를 편하게 해주는 경향
  - 사람이 숫자를 읽고 쓰는 방법과 같다.
- Network Byte Oder : 네트웍 데이터 통신
  - TCP/IP, XNS, SNA 통신 규약은 16bit와 32bit 정수에서 사용된다.
    - ntonl 같은 (Host to Network)를 이용해서 바이트 순서 정렬 필요.
- Unit format

2. Little - endian

> *작은 단위가 앞에 나오는 리틀 엔디안*

- 최 하위 바이트(Least Significant Byte; LSB)부터 차례로 저장하는 방식
- Intel Format
- 메모리에 저장된 값의 하위 바이트들만 사용할 때 별도의 계산이 필요 없다.
  - 32bit 숫자인 0x2A는 리틀 엔디언으로 표현하면 2A 00 00 00이 되는데,
    이 표현에서 앞의 두 바이트 또는 한 바이트만 떼어 내면 하위 16bit or 8bit를 얻을 수 있다.
    - 빅 엔디언 환경에서는 하위 16bit or 8bit값을 얻기 위해선 2byte or 3byte를 더해야 한다.

4. Bi - endian and Middle - endian

1) Bi - endian

- 빅 엔디언과 리틀 엔디언 중 하나를 선택할 수 있도록 설계계된 것을 Bi-endian Architecture

2) Middle - endian

- 32bit 정수가 2바이트 단위로는 빅 엔디언이고 그 안에서 1byte 단위로는 리틀 엔디언인 경우를 Middle - endian Architecture

Big endian	Little endian
Unix의 RISC계열의 프로세서가 사용하는 바이트 오더링	Intel 계열의 프로세서가 사용하는 바이트 오더링
네트워크에서 사용하는 바이트 오더링	
앞에서부터 스택에 Push	뒤에서부터 스택에 Push
비교 연산에서 빠르다.	계산 연산에서 빠르다.