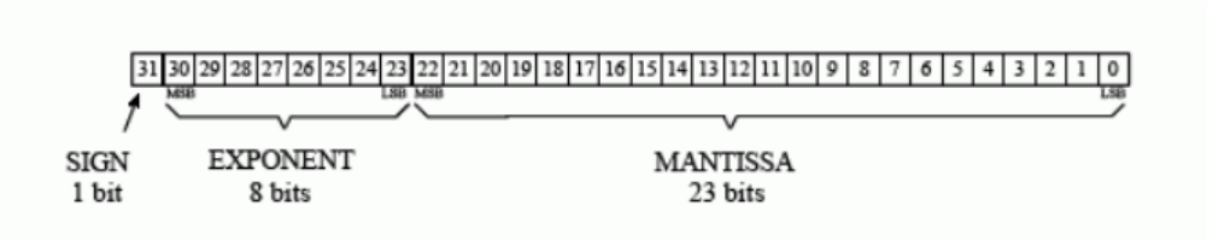
[수치해석] Fixed point

Floating point

C언어에서는 실수 표현 및 연산을 위해 **Floating point**를 사용한다.

32bit 기준



- **Sign, Exponent, Mantissa**로 나뉜다.

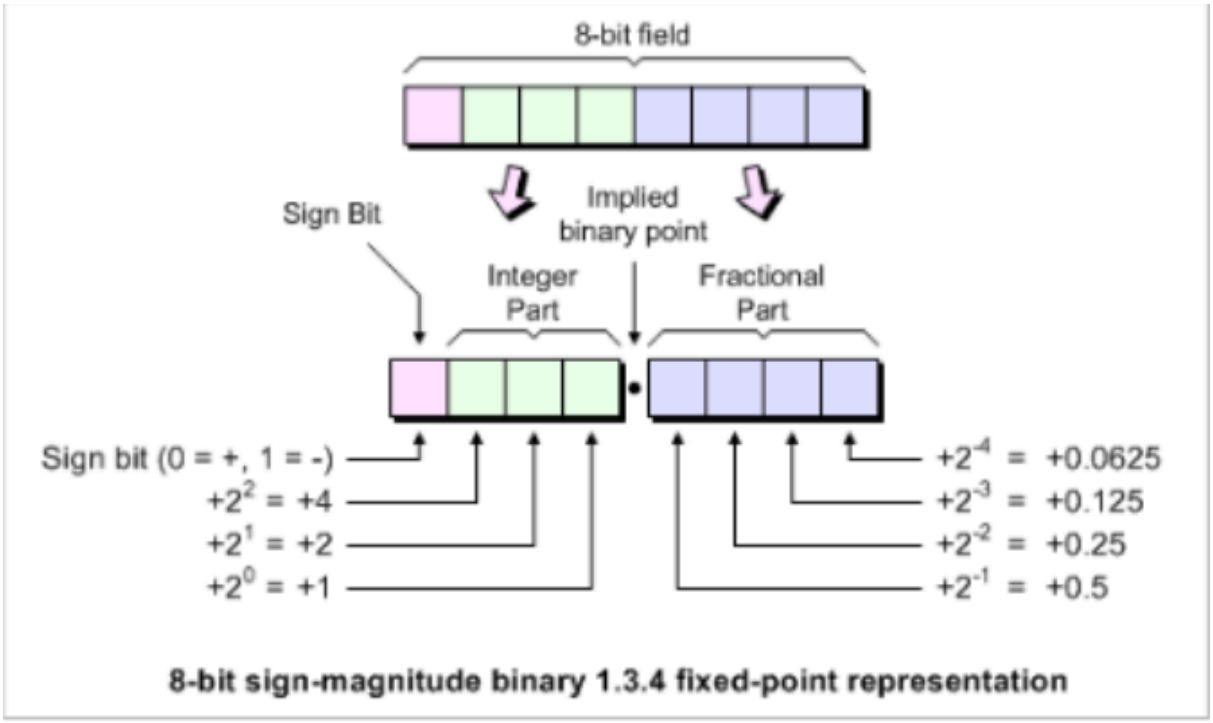
Floating point operation의 특징

- 표현 가능한 수의 범위가 크다.
- 정수 연산에 비해 연산량이 많다.
- 빠른 연산 속도를 위해 별도의 FPU (Floating Point Unit)이 필요하다.

Fixed point

정수부와 소수부의 비트 수를 미리 고정해놓고 실수를 표현하는 방식

8bit 기준 예시



8-bit sign-magnitude binary 1.3.4 fixed-point representation

- 예를 들어 **00111100**은 **3.75**를 의미한다.
- 이때, 소수점의 위치가 고정돼있다는 말은 **소수점의 위치를 특정 자리(여기서는 4번째 비트 뒤)로 고정하여 사용한다는 의미이다.**
- 컴퓨터가 연산을 처리할 때도 **정수의 연산처럼 동작해 연산량이 적고 Hardware cost가 낮다.**
  - 00011000 + 00100100 = 00111100 (1.5 + 2.25 = 3.75)의 경우를 살펴보자.
  - 각 수의 **소수부 자릿수가 4개이므로 각 수에 2^4 = 16배**를 해볼 수 있다.
  - 위 이진수의 연산을 실제 정수 연산처럼 생각해보자 24 + 36 = 60
  - 60을 8bit로 표현하면 00111100이다.
  - 즉, 실제로 **소수의 연산이지만 이진수에서는 정수의 연산과 동일하게 연산하고 해석만 Fixed point에 따라서 달리 하면 된다.**

Fixed point가 사용되는 곳

- **FPU가 없는 경우**에 사용한다.
  - FPU가 없다면 Floating point를 사용하는 것은 연산량 부담이 크다.

- **시스템 (Systems)**
  - **임베디드 시스템 / 마이크로프로세서**: 세탁기, 리모컨, 자동차 제어 장치 등에 들어가는 작은 컴퓨터같은 것. 이런 시스템은 **저비용**과 **저전력**이 매우 중요하기 때문에 **복잡한 FPU를 빼는 경우**가 많습니다.
  - **모바일 GPU**: 스마트폰의 그래픽 처리 장치(GPU)는 배터리 수명을 위해 **전력 효율**이 극도로 중요하다. 부동소수점 연산은 전력을 많이 소모하므로, 비교적 간단한 그래픽 연산에는 **전력 소모가 적은 고정소수점 방식을 사용해 효율**을 높인다.

- **알고리즘 (Algorithms)**
  - **신호 처리 (Digital Signal Processing, DSP)**
    - **영상, 음성**: 오디오나 비디오 데이터는 실시간으로 대량의 데이터를 빠르게 처리해야 한다. 이런 경우, 부동소수점의 넓은 표현 범위보다는 고정소수점의 **빠른 연산 속도**가 훨씬 유리하다.
  - **그래픽스 (Graphics)**:
    - 초기 컴퓨터 그래픽스나 일부 2D 그래픽 처리에서 좌표 계산, 색상 값 처리 등에 고정소수점이 널리 사용되었다. 이 역시 **연산 속도**가 중요하기 때문이다.