시스템 아키텍처

Floating Point 11\_04

1. 실수 표현방식
   1. 비율 이진수(Fractional Binary Numbers)
      1. 기존 이진수의 실수표현방식
      2. x/2^k의 형태로 표시되는 숫자만 표시 가능
      3. 컴퓨터로 표현하는 것의 문제 : 소수점을 어떻게 표현할 것이냐. 정수부와 실수부의 구분은?
   2. IEEE 부동 소수점
      1. 소수를 컴퓨터로 표현하는 표준 방식.
      2. 부호 비트 s, 유효숫자 M, 지수 E 로 구성된다.

-1^0(s) \* 11011(M)\* 2^-2(E)

* + 1. 비트 인코딩
       1. MSB : s부호 비트
       2. Exp 필드: 지수 E 의 표현(항상 E와 같지는 않음)
       3. frac필드 : 유효숫자 M의 표현 (항상 M과 같지는 않음)
       4. 컴파일러는 타입에 의한 해석
    2. 정밀도 이슈
       1. 몇 비트 컴퓨터이냐에 따라서 정밀도가 다름
       2. 32비트의 경우 exp 8 bit, frac 23 bit
       3. 64비트의 경우 exp 11 bit, frac 52 bit
       4. 결국 나타낼 수 있는 표현의 경우는 2^비트갯수
    3. 꼼수 : 0~1사이의 실수는 많이 쓰지만 1 이상의 실수가 적게 사용된다는 것을 사용. Exp의 값을 보고 결정한다.
       1. 정규화된 값 : 1보다 큰 실수. Exp != 0 && Exp != 11111…
       2. 비정규화된 값 : 0 ~ 1 사이에 있는 실수 Exp == 0
       3. 특수 값 : 무한대와 NaN 표현 Exp == 11111…
    4. 정규화 값 (Exp != 0 && Exp != 1111…)
       1. E = exp – Bias;
          1. 8비트 값인 exp가 음수도 표현 할 수 있어야 한다.
          2. Exp의 음수표현이 보수형태가 되면 비트만 보면 음수가 양수보다 큰 값처럼 보인다.
          3. 따라서 보수형태로 음수표현 하지 않고 Bias를 빼는 방식으로, 즉 비트 값이 작으면 전체 값도 작도록, 실수의 크기 비교 연산이 비트 수준에서 해결 될 수 있도록 구현한다.
          4. 따라서 Bias = 2^(bit - 1) – 1
       2. M = 1 + f = 1.xxxxxxx…. ( implicit 한 정수부 1이 존재)
          1. 항상 선두에 선 값 1을 지정해줘야 한다. ( 0 제외 )
          2. 항상 들어오므로 생략 가능하다. (비트 하나 생략 개이득)
          3. 따라서 M = 1 + f
    5. 비정규화 값 (Exp == 0)
       1. E = 1 – Bias (exp는 무조건 0 이므로… 그렇다고 0 - Bias아님!)
          1. 결국 표현하는 영역 2^-126 이하 (정규화로 표현할 수 있는 최소값)
       2. M = 0 + f (implicit 0, 0에 가까운 숫자를 쓰기 위해서)
          1. 이때 묵시적 선두 0을 사용해서 자연스럽게 연속된 실수를 표현 가능하다. (2^-126\*0.111111111…. 2^-126\*1.0)
    6. 특수 값 (Exp == 111111….)
       1. Exp == 111….. && frac == 0
          1. 무한대(음수 / 양수)
          2. 연산의 오버 플로우를 표현
       2. Exp == 111….. && frac != 0
          1. Not a Number ( NaN)
          2. Undefined
    7. 부동소수점 값의 분포
       1. Exp의 모든 값마다 같은 개수의 값을 갖는다. 따라서 값이 0 에 가까울 수록 조밀한 분포를 보인다.
       2. 그리고 모든 부동 소수점 값은 비정규 값이나 정규값 모두 자신의 간격만큼 연속되는 속성을 갖는다.
    8. 부동 소수점의 한계
       1. 정밀도의 제약으로 표현하지 못하는 정수가 존재. Frac의 총 비트수가 n이라고 하면 2^n+1 + 1은 표현 불가능. frac으로 그 숫자의 자리 수 전체를 표현할 수 없기 때문이다. 32비트 컴퓨터에서는 2^24 + 1 부터 표현 불가.
    9. 부동 소수점 연산
       1. 보수 개념이 없으므로 양수가 갑자기 음수가 되지 않는다
       2. 오버플로우가 일어나는 경우 근사로 계산할 수 있다. (짝수 근사)
          1. 기본은 반올림
          2. 반올림은 쩜오에 대해 너무 편향적이니까 확률적으로 안정적인 방식 짝수 근사를 적용한다.
          3. 쩜오를 정수부가 짝수가 되도록 올리거나 내린다.
          4. 이진수의 경우 0.5 대신 1/4 가 적용된다.
       3. 덧셈
          1. 지수부가 있으므로 자릿수가 다를 수 있다.
          2. 높은 자릿수에 맞춰서 덧셈 수행한뒤 결과값을 보정한다.
       4. 곱셈
          1. 곱셈이므로 자릿수 맞출 필요 없다.
          2. 유효숫자를 그냥 곱하고 지수부는 더해준다. 그리고 결과값을 보정한다.
    10. 타입 변환
        1. int -> float : 오버플로우 없다. 값 근사
        2. int , float -> double : 값 유지되는 정확한 변환
        3. float, double -> 오버플로우 있다.