컴퓨터 구조

3장 컴퓨터 산술과 논리 연산4

안형태

anten@kumoh.ac.kr

디지털관 139호

컴퓨터 산술과 논리 연산

7. 부동소수점 산술 연산

덧셈과 뺄셈

- ① 지수 조정 단계
  - 지수들이 일치되도록 조정(alignment) → 더 큰 수가 기준(작은 수를 시프트)
- ② 가수 연산 단계
  - 가수들 간의 연산(더하기 혹은 빼기) 수행
- ③ 정규화 단계
  - 결과를 정규화(normalization)

- **[참고]** 10진수 부동소수점 산술

$(135 \times 10^{-5}) + (246 \times 10^{-3})$ →

$$\begin{array}{r} 1.35 \times 10^{-3} \\ +\ 246 \quad \times 10^{-3} \\ \hline 247.35 \times 10^{-3} \end{array}$$

덧셈과 뺄셈

- **[예제]** 부동소수점 수들 간의 덧셈 $(0.110100 \times 2^3) + (0.111100 \times 2^5)$을 수행하라.

- **[풀이]**

$$\begin{array}{l} 0.110100 \times 2^3 \\ +\ 0.111100 \times 2^5 \\ \hline \end{array}$$

(1) 지수 조정 →

$$\begin{array}{l} 0.001101 \times 2^5 \\ +\ 0.111100 \times 2^5 \\ \hline \end{array}$$

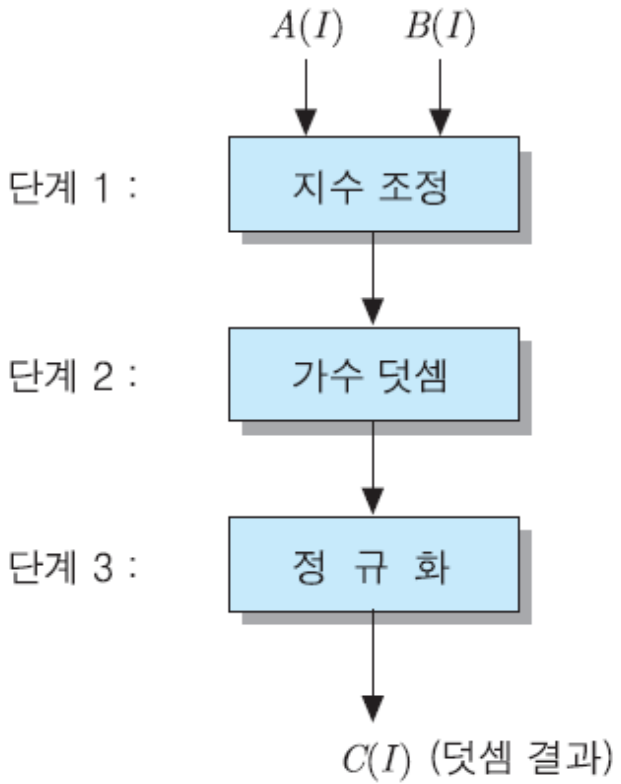
(2) 더하기 → 1.001001×2^5

(3) 정규화 → 0.1001001×2^6 (최종 결과)

부동소수점 산술의 파이프라이닝

- 부동소수점 산술의 파이프라이닝
  - 연산 과정을 독립적 단계들로 분리 가능
  - 단계 수만큼의 속도 향상
  - 대규모의 부동소수점 계산을 처리하는 대부분의 슈퍼 컴퓨터에서 채택
- **[예]** 수의 배열(number array)들 간의 덧셈

$$C(I)=A(I)+B(I)$$



부동소수점 연산의 정밀도 한계

- **[예]** 부동 소수점 연산: **1.2 + 0.3**

$a = 1.2; b = 0.3;$

$a + b == 1.5;$ //true or false?

  - 결과: 일반적으로 **false**(프로그래밍 언어에 따라 다름)
  - IEEE 754 표준에서 유한한 비트 수로 정밀도에 한계가 있어 1.2와 0.3는 완벽하게 표현되지 않으므로, 1.2 + 0.3의 결과는 정확한 1.5가 아님
    - 1.2의 이진표현: $1.00110011001100110011001_2 = 1.1999998092651367_{10}$(근사값)
    - 0.3의 이진표현: $0.01001100110011001100110011_2 = 0.2999997138977051_{10}$(근사값)
    - 1.2 + 0.3의 이진표현: $1.01111111111111111111111_2 = 1.4999995231628418_{10}$(근사값)
    - 1.5의 이진표현: 1.1_2
  - 오차 허용 범위(**[예]** epsilon)을 설정하여 (a + b – 1.5) < epsilon 활용

부동소수점 곱셈 / 나눗셈

2진수 부동소수점 곱셈 과정

- ① 가수들을 곱함: 정수 곱셈 활용
- ② 지수들을 더함
  - 바이어스된 지수 간의 덧셈은 바이어스를 하나를 제거
- ③ 결과값을 정규화

2진수 부동소수점 나눗셈 과정

- ① 가수들을 나눔: 정수 나눗셈 활용
- ② 피제수의 지수에서 제수의 지수를 뺌
  - 바이어스된 지수 간의 뺄셈은 바이어스를 하나를 추가
- ③ 결과값을 정규화

부동소수점 곱셈 / 나눗셈

- **[예제]** 두 부동소수점 수들의 곱셈 $(0.1011\times 2^{3})\times(0.1001\times 2^{5})$을 수행하라. (부동소수점 표현 형식과 바이어스는 고려 안함)

- **[풀이]**
  - ① 가수 곱하기: $1011\times1001=01100011$
  - ② 지수 더하기: $3+5=8$
  - ③ 정규화: $0.01100011 \times 2^{8} = 0.1100011 \times 2^{7}$

부동소수점 연산 과정에서 발생 가능한 문제들

- **지수 오버플로우(exponent overflow)**
  - 양의 지수 값이 최대 지수 값을 초과 → 수가 너무 커서 표현될 수 없는 상태이므로, +∞ 또는 -∞로 세트
- **지수 언더플로우(exponent underflow)**
  - 음의 지수 값이 최대 지수 값을 초과 → 수가 너무 작아서 표현될 수 없는 상태이므로, 0으로 세트
- **가수 언더플로우(mantissa underflow)**
  - 가수의 소수점 위치 조정 과정에서 비트들이 가수의 우측 편으로 넘치는 상태 → 반올림(rounding) 또는 절삭(truncation) 적용
- **가수 오버플로우(mantissa overflow)**
  - 같은 부호를 가진 두 가수들을 덧셈하였을 때 올림수가 발생 → 가수를 우측 시프트하고 지수를 증가시키는 재조정(realignment) 과정을 통하여 정규화

[TMI] 2진수 반올림 규칙

- 2진수에서 반올림 할 위치 다음 비트가 0이면 버림
  - **[예]** 0.101_2(소수 첫 번째 위치에서 반올림) → 0.1_2
- 2진수에서 반올림 할 위치 다음 비트가 1이고, 뒤의 비트들 중 하나라도 1이면 올림
  - **[예]** 0.11001_2(소수 첫 번째 위치에서 반올림) → 1.0_2
- 2진수에서 반올림할 위치 다음 비트가 1이고, 뒤를 있는 모든 비트가 0이면, 짝수(0)가 되도록 버리거나 올림(ties-to-even)
  - **[예]** 0.01000_2(소수 첫 번째 위치에서 반올림) → 반올림할 자리의 숫자가 0(짝수)이므로 버림: 0.0_2
  - **[예]** 0.11000_2(소수 첫 번째 위치에서 반올림) → 반올림할 자리의 숫자가 1(홀수)이므로 올림: 1.0_2

Q & A

End!