Proceedings of KFIS Spring Conference 2007 Vol. 17, No. 1.

매우 큰 수치 데이터 저장을 위한 겁수 표현방법

Method of the Kalpa Number Representing for Ultra.Large Numeric Data Store

김귀연[1], 김병기[2], 최준용*
[1 2] 전남대학교 전자컴퓨터학과
E-mail : gykim,bgkim@chonnam.ac.kr
*경북대학교 전자전기컴퓨터학부
E-mail : abyss1225@hanmail.net

요약

수학, 물리학, 전자전기공학에서 취급하는 수치 자료는 매우 크거나 정밀도가 높으므로 계산과정에서 흔히 오버플로우나 언더플로우가 발생할 수 있다. 본 연구는 이런 현상을 최소화하기 위하여 겁수(Kalpa Number) 표현방법을 제안한다. 겁수는 고정소수점수와 부동소수점수를 저장할 수 있는 자료형을 제공하며, 정수형 및 실수형의 지수와 가수부분을 문자형 배열에 저장함으로써 매우 큰 수를 취급할 수 있다. 이를 이용하여 프로그램을 작성하면 베르누이 수나 전자의 움직임과 관련된 계산 프로그램에서 오버플로우나 언더플로우를 최소화할 수 있다.

Key Word : 겁수, Kalpa Number, 큰 수, 부동소수점수

1. 서론

디지털 컴퓨터는 현대 사회에서 매우 중요한 위치를 차지하고 있다. 따라서 컴퓨터 없이는 이룩하지 못할 많은 과학, 산업, 공학 분야 등의 발전을 얻게 되었다. 또한 컴퓨터는 의학치료, 기상예측, 우주과학, 전기전자공학, 수학, 물리학 등의 여러 분야에서 다양한 목적으로 사용되고 있다[8].

컴퓨터에서 이들 각 분야의 프로그램에 사용되는 자료는 이진수로 저장되며, 한정된 크기로 표현된다. C 언어에서 취급하는 정수형의 경우 약 21억까지의 수치를 취급할 수 있으며, 실수형의 경우 소수 이하 유효숫자 15자리와 $10^{\pm308}$ 까지 처리할 수 있다. 따라서 수치가 이보다 크거나 정밀도가 높을 경우 원래 수치의 근사치가 저장되는 오버플로우나 언더플로우로 나타난다.

수학 분야에서 베르누이(Bernoulli) 수나 오일러(Euler) 수를 구하는 문제[6]는 매우 큰 수를 취급하므로 컴퓨터를 이용하여도 정확한 값을 구하기 어렵다. 또한, 전자전기공학이나 물리학 분야의 전자나 양자를 취급하는 프로그램은 매우 정밀도가 높은 수를 취급하므로 이 또한 컴퓨터로 처리하기 쉽지 않다. 전자와 관련된 프로그램을 실행하다 보면 계산 과정에서 $10^{\pm308}$ 범위를 초과하는 값들을 쉽게 발견할 수 있으며, 정밀한 값을 구하기 어려우므로 근사값을 구하여 계산한다[5].

본 연구는 이와 같이 매우 큰 수나 정밀도가 높은 수를 취급하는 분야의 계산에 사용할 수 있는 소프트웨어 설계 및 구현 방안을 제시하고자 한다.

2. 관련연구

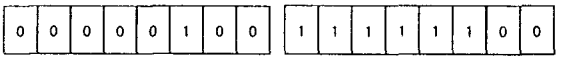
2.1 컴퓨터의 수치 데이터 저장 방법

컴퓨터에서 취급하는 수는 고정소수점 수(fixed point number)와 부동소수점 수(floating point number)가 있다. 또한 10진수는 컴퓨터에서 2진수 형태로 변환되어 처리된다. 본 절은 2진수의 저장방법, 고정소수점 수 및 부동소수점 수의 저장방법, 수치 저장에서 나타나는 문제점, 기존의 연구에 대하여 알아본다.

(1) 2 진수

실생활에서 접하는 모든 10진수는 2진수의 형

태로 변환된다. 또한 10진수는 양수와 음수가 있으므로 2진수에서도 양수와 음수에 대한 표현이 존재한다. 일반적으로 양수인 10진수는 2진수로 변환한 값을 그대로 저장하고, 음수는 2의 보수법(2's complement)으로 저장한다. 예를 들면 10진수 4와 -4는 컴퓨터에서 (그림1)과 같이 저장한다.



(a) 10 진수 4 (b) 10 진수 -4
(그림 1) 10 진수 4와 -4의 저장(1 바이트)

(2) 고정소수점 수(fixed point number)

고정 소수점 수는 정수를 의미하며 C 언어의 경우 2바이트(byte) 또는 4바이트에 저장한다. 4바이트에 저장한다면 저장 가능한 수의 범위는 -2^{31} ~ $2^{31}-1$ 이다.

(3) 부동소수점 수(floating point number)

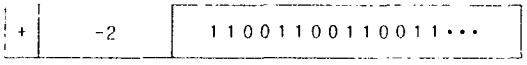
부동소수점 수는 소수점이 있는 수를 의미하며 C 언어의 경우 4바이트 또는 8바이트에 저장한다. 부동소수점 수는 정규화(nomalized)의 과정을 거쳐 지수(exponent)와 가수(mantissa) 부분이 분리되어 저장된다. 8바이트의 경우 소수이하 유효숫자 15자리까지 표현 가능하다. 수의 크기는 $10^{\pm 308}$ 범위이다. 예를 들면 123.45는 $0.12345 * 10^3$ 으로 정규화되어 (그림 2)와 같은 형태로 저장된다.

(그림 2) 123.45의 저장 형태

2.2 문제점

(1) 2진수 변환에서 발생하는 오차

10진수를 2진수로 변환하는 과정에서 오차가 발생한다. 예를 들면 10진수 0.2는 2진수로 변환되어 (그림3)과 같은 형태로 저장된다. 이 과정에서 가수부분의 정해진 영역을 초과한 비트는 무시된다. 따라서 소수가 있는 대부분의 수는 컴퓨터 내부에서 근사값이 저장되므로 실제 계산한 값과 오차가 있을 수 있다.



(그림 3) $(0.001100110011\cdots)_2$의 저장형태

(2) 오버플로우(Overflow)

컴퓨터에서 모든 수는 고정된 영역의 바이트에 저장하므로 표현할 수 있는 범위가 한정되어 있다. 따라서 정해진 범위를 초과하는 매우 큰 수는 일부분만 저장되고 나머지는 무시되므로 우리가 의도한 것과 다른 값이 저장된다. 이를 오버플로우라 한다.

(3) 언더플로우(Underflow)

부동소수점 수에서 소수이하 유효자리 수는 최대 15자리이며, 이보다 정밀한 범위의 수는 무시된다. 따라서 실제 계산결과와 오차가 발생할 수 있다. 이를 언더플로우라 한다.

2.3 수치 데이터 확장에 관한 기존 연구

수학분야에서 π나 자연대수 e 의 값을 구하는 것은 매우 의미있는 일이다. 그러나 이 수는 무리수이므로 정확한 값을 찾을 수 없다. π의 정밀한 값을 찾기 위한 노력으로 HFLOAT 또는 apfloat 과 같은 프로그램이 개발되어 사용되고 있다. 이 프로그램들을 이용하면 π의 수백만 ~ 수십억 자리까지 계산이 가능하다[1][3].

CLN(Class Library for Numbers)은 매우 크거나 정밀한 수를 취급할 수 있도록 개발된 프로그램이다. 이 프로그램이 취급하는 자료의 형(type)은 cl_RA, cl_SF, cl_FF, cl_LF 등과 같이 다양하다. 그 중 가장 큰 수치를 취급할 수 있는 cl_LF 형은 부호 비트 1, 지수 비트 32, 64 이상의 가수(mantissa) 비트를 사용한다, 따라서 지수의 크기가 2^{31} 인 매우 큰 수를 표현할 수 있고, 가수부분은 크기에 제한을 두지 않는다[2].

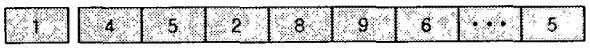
3. 수치 데이터 확장 방안

본 연구는 매우 큰 수를 저장하는 방법과 가수 부분을 최대한 확장하여 정밀도를 높이는 것이 목적이다. 이는 데이터를 저장하는 바이트의 크기를 확장함으로써 해결할 수 있다. 그러나 현재까지 개발된 모든 컴퓨터는 저장역역이 한정되어 있으므로 무한한 크기로 바이트를 확장하는 것은 불가능하다. 따라서 컴퓨터의 허용범위 안에서 최대한의 바이트를 할당하는 방법을 사용할 것이다.

3.1 겁수(Kalpa Number)

정수형과 실수형의 지수 및 가수부분을 확장하는 가장 간단한 방법은 문자형의 배열로 저장하는 것이다. 부호는 별도의 바이트에 양수일 경우 1, 음수일 경우 -1을 저장한다.

C 언어의 경우 문자형은 char이며, -256~255 범위의 정수를 저장할 수 있으나[7], 본 논문은 0~9 범위의 수를 저장한다. 이런 방법을 사용하면 컴퓨터가 허용할 수 있는 최대 크기의 배열을 사용하였을 때 매우 큰 수를 저장할 수 있다. 예를 들면 (그림 4)와 같은 형태로 저장한다. 가장 왼쪽의 1은 부호(+)를 의미한다.



(그림 4) 수의 저장 방법

(그림 4)는 정수형일 경우 $(+452896\bullet\bullet\bullet5)_{10}$, 지수일 경우 $10^{+452896\bullet\bullet\bullet5}$, 가수일 경우 $(+0.452896\bullet\bullet\bullet5)_{10}$를 의미한다.

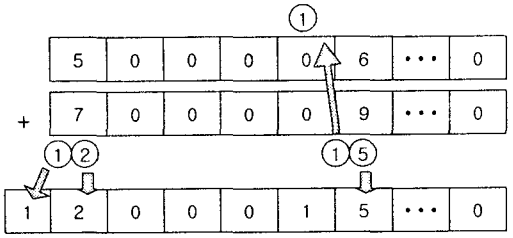
본 논문은 이와 같은 방법으로 수를 표현하는 방법을 겁수(Kalpa Number) 표현법이라 하고, 겁수 표현법으로 저장된 수를 겁수라 정의한다. 겁수는 정수형 겁수와 실수형 겁수 두 가지 형을 제공한다.

3.2 겁수의 4칙 연산

겁수와 겁수간의 연산을 위하여 다양한 연산자를 정의하여야 한다. 본 절은 겁수의 4칙 연산만을 논의하고, 기타 연산자에 대한 연구는 제외한다.

(1) 정수형 겁수의 덧셈과 뺄셈

정수형 겁수의 덧셈과 뺄셈은 종이에 연필로 덧셈, 뺄셈하는 것과 동일한 방법을 사용한다. 예를 들면 양수인 두 정수형 겁수 덧셈의 경우 (그림5)와 같은 방법을 이용한다. (그림5)에서 가장 왼쪽에 발생한 올림수는 배열의 왼쪽 공간에 추가하여 저장한다.



(그림 5) 정수형 겁수의 덧셈

뺄셈의 경우 절대값이 큰 수에서 작은 수를 빼고, 계산결과의 수에 절대값이 큰 쪽의 부호를 저장한다.

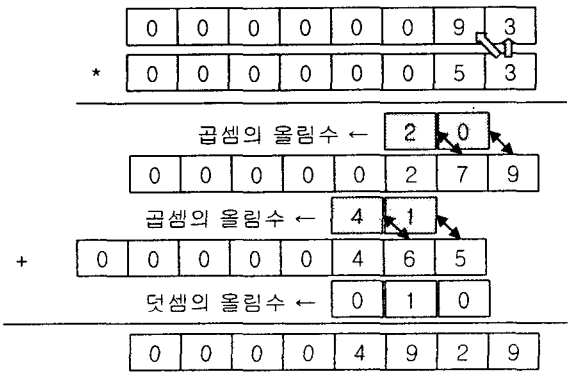
(2) 실수형 겁수의 덧셈과 뺄셈

실수형 겁수의 덧셈과 뺄셈은 우선 지수의 값을 동일하게 조정한다. 지수가 작은 수를 지수가 큰 수와 같도록 소수점을 이동한다. 예를 들면 $0.12 * 10^5$와 $0.34*10^2$을 덧셈할 경우 $0.34*10^2$을 $0.00034*10^5$ 으로 조정한다. 지수의 값이 동일하면, 가수부분은 정수형 겁수의 덧셈 또는 뺄셈과 동일한 방법으로 계산한다.

즉, 가수의 덧셈은 (그림 5)와 같다. 이 때 가장 왼쪽에 발생한 올림수는 배열의 왼쪽 공간에 추가하여 저장하고, 지수를 1 증가한다. 뺄셈도 유사한 방법으로 처리할 수 있다.

(3) 정수형 겁수의 곱셈

정수형 겁수의 곱셈은 종이에 연필을 이용하여 계산하는 방법과 동일하게 처리한다. 예를 들면 93*53은 (그림 6)과 같이 처리한다.

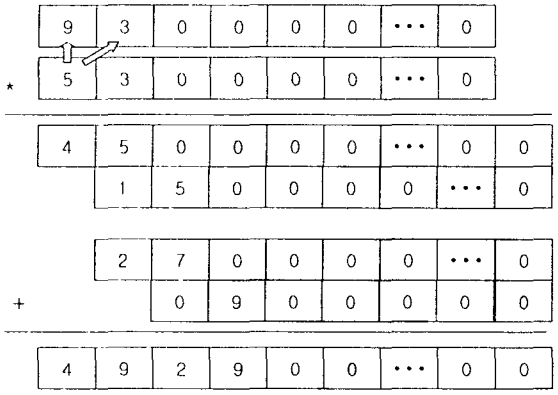


(그림 6) 정수형 겁수의 곱셈 방법

(4) 실수형 겁수의 곱셈

실수형 겁수 곱셈은 지수와 가수를 나누어 연산한다. 지수는 정수형 겁수의 덧셈으로 처리하

고, 지수는 이와 비슷한 방법으로 계산한다. 예를 들면 0.93*0.53의 계산방법은 (그림7)과 같다.

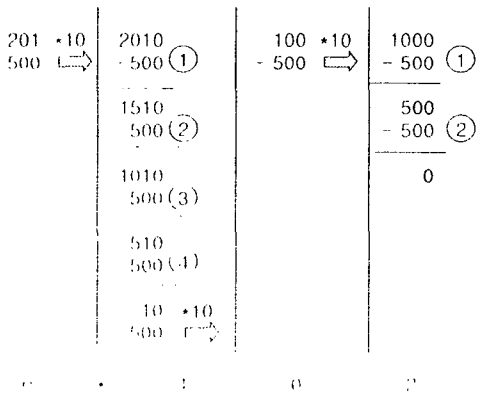


(그림 7) 지수의 곱셈 방법

곱셈 계산은 가장 왼쪽부터 실행한다. 이때 배열의 지정된 크기를 초과하는 오른쪽 끝 부분의 곱셈 결과는 무시한다. 따라서 검수에서도 언더플로우는 발생할 수 있다. (그림7)의 가장 오른쪽 바이트는 가장 왼쪽 바이트의 값이 0일 경우 소수점을 오른쪽으로 한번 이동하기 위한 임시 저장공간이다. 이 경우 지수가 1 감소한다.

(5) 검수의 나눗셈

정수형 검수의 나눗셈은 분자에서 분모를 계속적으로 뺄셈한다. 또한 실수형 검수의 곱셈은 지수와 가수를 나누어 처리한다. 지수 부분은 분모의 지수에 -1을 곱하여 분자의 지수에 더한 후, 가수의 나눗셈을 수행한다. 정수형 검수의 나눗셈과 실수형 검수의 가수 나눗셈은 동일하므로 여기서는 가수의 나눗셈을 예로 든다. 0.201/0.500의 경우 (그림 8)과 같다.



(그림 8) 가수의 나눗셈

(그림8)에서 소수점 왼쪽의 수가 0이 아니면 소수점을 왼쪽으로 한번 이동하고, 지수를 1 증가한다.

4. 결론

수학, 물리학, 전자전기공학 등의 분야에서 취급하는 프로그램은 계산 과정에서 흔히 오버플로우나 언더플로우가 나타날 수 있다. 본 연구는 이런 현상을 최소화하고 좀 더 정확한 값을 얻기 위한 것이다. 이를 위하여 수를 문자형 배열 형태로 저장하는 검수(Kalpa Number) 표현법을 제안하였다.

정수형 검수는 수의 부호와 크기를 분리하여 부호는 하나의 문자형 변수에 저장하고, 수의 크기는 문자형 배열에 저장한다. 실수형 검수는 각각 지수와 가수로 분리하여 정수형 검수와 유사한 방법으로 문자형 배열에 저장한다.

검수의 연산을 위하여 사칙연산 방법을 제시하였다. 사칙연산은 종이에 연필을 사용하여 계산하는 것과 유사한 방법으로 처리한다. 논리연산, 삼각함수, 지수함수 등에 필요한 연산은 추후 연구할 것이다.

[참고문헌]

[1] http://www.bernoulli.org/
[2] http://www.ginac.de/CLN/
[3] http://www.jjj.de/hfloat/
[4] http://www.apfloat.org/apfloat/
[5] Joung Young Sug, at al. "Quantum transition processes in · deformation potential interacing systems using the equilibrium density projection technique", PHYSICAL REVIEW B vol. 64, 2001.
[6] 강동진, 김대열, 박달원, 서종진, 임석훈, 장이채, "연속된 정수의 멱의 합의 변천사에 대한 고찰", 한국수학사학회지 제19권, 제1호, pp.1~16, 2. 2006.
[7] 강환수, 신용현, Perfect C, pp.114~701, INFINITYBOOKS, 2007.
[8] 권영빈, 김준년, 컴퓨터공학, pp.4~25, 대영사, 1988.