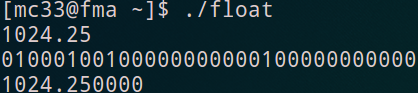
# Multicore Computing HW 1 Report

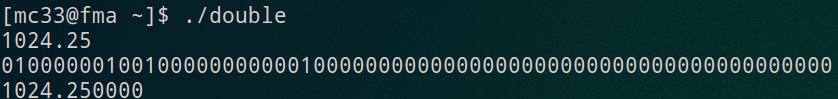
2014-12472 오흥연

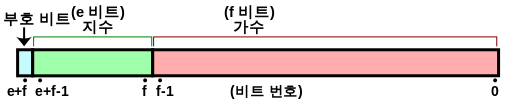
# 1. 부동소수점 표현

과제의 내용은 어떻게 실수가 이진수로 표현되는지 알아보는 것이었다. 구현의 편의를 위해 scanf(“%f”)를 통해 실수를 이진수 값으로 저장하였다. 위에서 저장한 값을 union struct를 이용해 32 bit의 int로 형을 바꾼 뒤 bit-wise and연산을 통해 float의 각 bit의 값을 저장하여 출력하였다. 또 이진수로 받아온 float를 printf(“%f”)를 통해 다시 십진수로 출력하였다. 그 결과는 다음과 같다.



같은 과정을 double과 64 bit int로 이용한 경우다.

위와 같은 변환이 어떻게 일어나는지 살펴보자. 먼저 floating point number의 구성을 보면 다음과 같다



우리의 예제와 비교해보면 첫 비트가 0이므로 양수인 것을 알 수 있다.

그 다음은 가수이다. 먼저 이진수 표현법으로 바꾸면 우리의 값은 이렇게 바뀐다.

10000000000.01

이는 다음과 같이 표현할 수 있다

1.000000000001 \* *29*

즉 이 9가 지수가 되고 0000000000001 이 가수가 된다. Bias에 의해 9는 127이 더해져 10001001이 된다. 그래서 결론적으로 다음과 같은 이진수가 나온다.

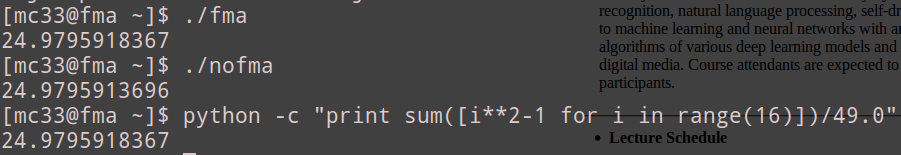
010001010000000000001

# 2. fma

## 2.1 What & Why

fma는 fused multiply accumulator 의 약자로 floating point 숫자의 곱셈과 덧셈을 한번에 하고 rounding도 한번만 일어나는 연산이다. Rounding이 적게 일어나므로 정확도가 더 높으며 성능의 향상을 기대 할 수 있다.

## 2.2 Accuracy

길이 16짜리 두 벡터의 내적을 구하는 코드를 fma를 쓴 것과 안 쓴 결과가 각각 첫 줄과 두 번째 줄이다. 이에 상응하는 연산을 한번의 rounding 만을 이용해 구한 값이 3번째 줄이다. Fma를 쓴 경우 안 쓴 경우보다 rounding이 절반만 일어나 더 정확한 것을 알 수 있다.

## 2.3 Speed

실험은 서버에서 했으며 iteration은 코드 내부에서 반복한 횟수이고 length는 내적하는 두 벡터의 크기이다. fma의 여부는 objdump를 통해 assembly 코드 상에 나타나는 지를 통해 확인하였다.

각 크기와 반복 횟수 별로 101번 반복했으며 그래프에 나타낸 값은 그 평균 값이다. 시간의 측정은 clock\_gettime으로 ns단위로 구했다. Fma.c코드와 test.sh코드를 참고하면 알 수 있다.

실험 결과 fma를 사용할 경우 약 20%의 성능 이득을 얻을 수 있음을 알 수 있다.