# 어셈블리프로그램 설계 및 실습


## Lab #6

## Floating-Point

Kwangwoon University
Embedded System Architecture Lab

# 학습 목표

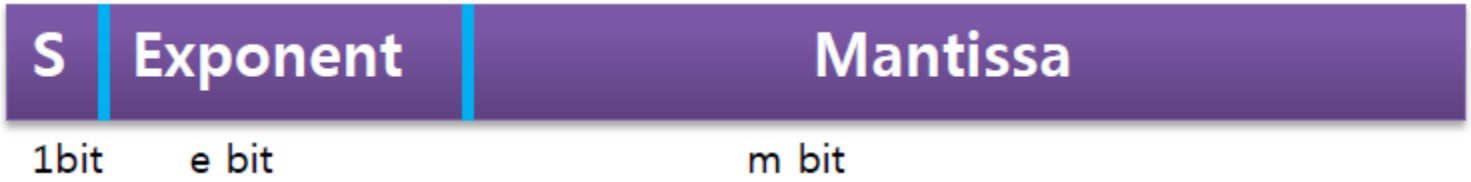
- Arm에는 floating point를 해주는 instruction이 존재하지 않는다. 때문에 floating point가 입력되었을 때 해당 연산을 수행시켜주는 assembly code가 필요하다.
- Floating point의 adder를 구현하여 floating point의 덧셈과 뺄셈에 대하여 알아보자
- Normalize 과정에서 어떤 경우에는 right로, 어떤 경우에는 left로 shift되는지를 고려하며 코드를 작성하여 보자

# Converting to Binary

- Convert the decimal number $41.6875_{10}$ to binary
  - $41_{10} = 101001_2$
  - $0.6875_{10} = 0.1011_2$
- Fixed-point : the binary point is fixed
  - $6.75_{10} = 0110.1100_2$
- Floating-point : the binary point floats to the right of the most significant 1

# Floating-Point

- IEEE 754 Floating-Point Standard




  - Value = $(-1)^S \times (1.\ mantissa) \times 2^{(exponent-127)}$
  - Precisions
    - Single precision (32-bit): e=8, m=23
    - Double precision (64-bit): e=11, m=52

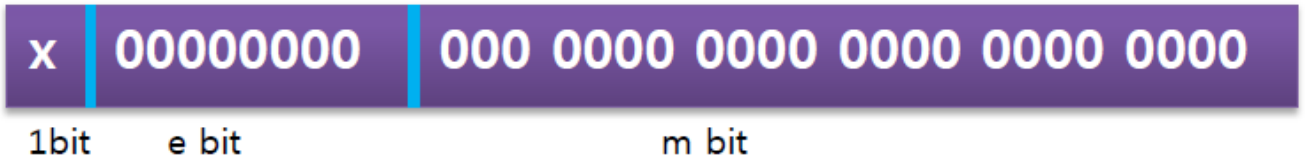
# Example

- Express $-58_{10}$ using the IEEE 754 32-bit floating point standard
  - $58_{10} = 111010_2 = 1.1101_2 \times 2^5$
  - Sign bit is 1
    - Because -58 < 0
  - 8 exponent bits : $127+5 = 132 = 10000100_2$
  - 23 fraction bits : 11010000
    - $1.\mathbf{1101}_2 \times 2^5$

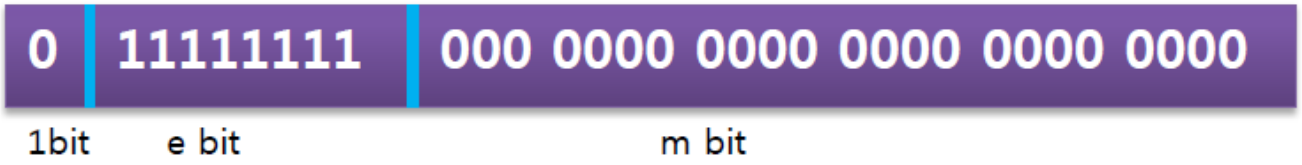

  - Express above number in hexadecimal
    - 0xC2680000

# Floating-Point Number

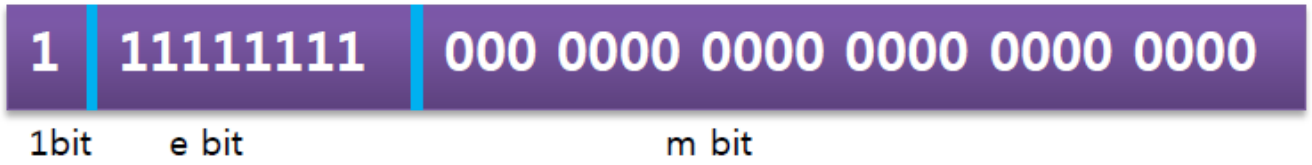
- Special Cases
  - 0



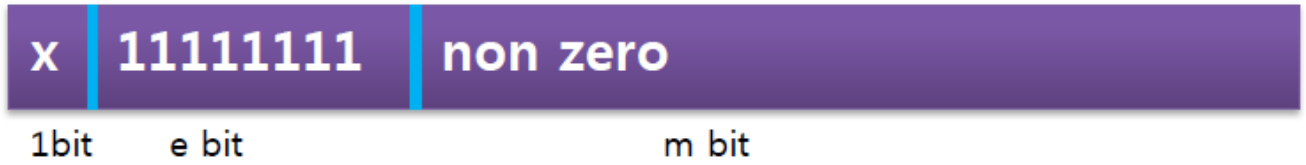

  - ∞




  - -∞




  - Nan



# Floating-Point Addition (1/3)

- 1. 두 floating point의 부호가 같은 경우
  - 아래의 bit data를 각 숫자에 대해서 추출
    - Sign bit
    - Exponent bits
    - Mantissa(Fraction) bits
  - Mantissa형태 앞에 1을 붙임
  - Exponent를 비교
    - 두 floating point의 exponent의 차이 값 구함
    - |Exponent 1 – Exponent 2| = $Shift\ num$
  - Shift num이 0이 아닌 경우,
    - Exponent값이 작은 값의 mantissa값을 shift num만큼 오른쪽으로 shift
    - Mantissa의 앞에 1.이 있음을 주의할 것.
    - Ex) 1.1 >> 1 → 0.11 ($\times 2^1$)

# Floating-Point Addition (2/3)

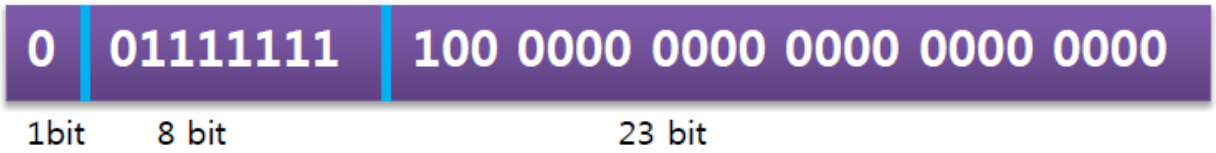
- 각 mantissa값을 더해 줌
- Mantissa값을 Normalize 해줌
  - Exponent값 수정
  - Ex) $10.011 \times 2^1 = 1.0011 \times 2^2$
  - Ex) $0.01101 \times 2^1 = 1.101 \times 2^{-1}$
- 수정이 완료된 mantissa값과 exponent값을 사용하여 결과를 도출
  - S = 두 floating point의 sign bit
  - Exponent = Normalize result exponent + 127
  - Fraction = mantissa

# Floating-Point Addition (3/3)

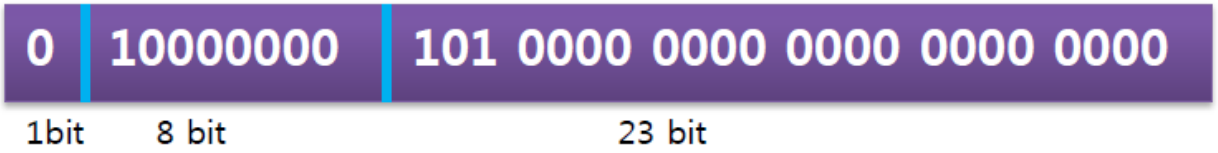
- Add the following floating-point numbers :
  - 0x3FC00000
  - 0x40500000

# Floating-Point Addition (3/3)

- Add the following floating-point numbers :
  - 0x3FC00000
  - 0x40500000



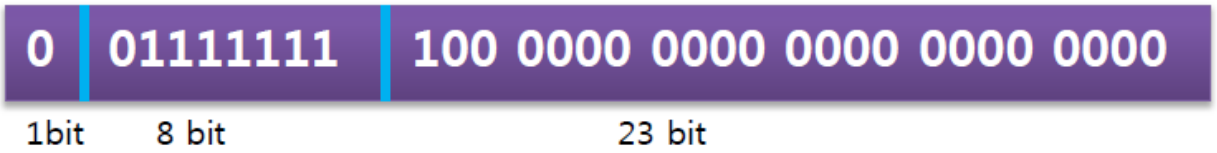

- $S_1 = 0, E_1 = 127, F_1 = 1.1$



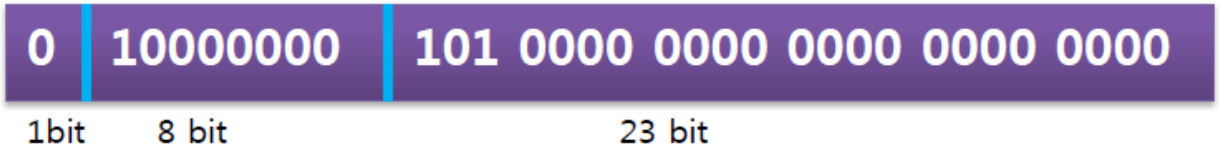

- $S_2 = 0, E_2=128, F_2 = 1.101$

# Floating-Point Addition (3/3)

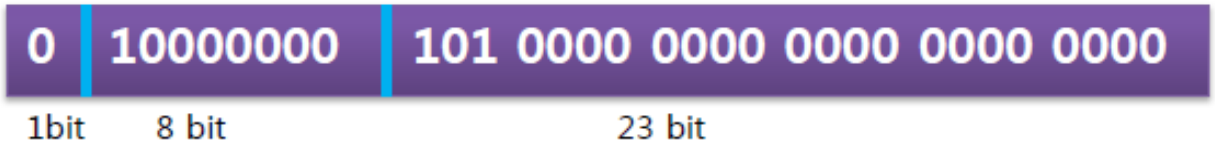
- Add the following floating-point numbers :




- $S_1 = 0, E_1 = 127, F_1 = 1.1$




- $S_2 = 0, E_2=128, F_2 = 1.101$
- Shift num = $E_2 - E_1 = 1$
- $F_1 = F_1>>1 = 0.11$ , $F_1+F_2 = 10.011= 1.0011 X2$
- Result : $S_R=0, E_R=129, F_R= 1.0011$




=>0x40980000

# Floating-Point Subtraction (1/3)

- 2. 두 floating point의 부호가 다른 경우
  - 아래의 bit data를 각 숫자에 대해서 추출
    - Sign bit
    - Exponent bits
    - Mantissa(Fraction) bits
  - Mantissa형태 앞에 1을 붙임
  - Exponent를 비교
    - 두 floating point의 exponent의 차이 값 구함
    - |Exponent 1 – Exponent 2| = $Shift\ num$
  - Shift num이 0이 아닌 경우,
    - Exponent값이 작은 값의 mantissa값을 shift num만큼 오른쪽으로 shift
    - Mantissa의 앞에 1.이 있음을 주의할 것.
    - Ex) 1.1 >> 1 → 0.11 ($\times 2^1$)

# Floating-Point Subtraction (2/3)

- Mantissa값을 큰 값에서 작은값을 빼 줌
- Mantissa값을 Normalize 해줌
  - Exponent값 수정
  - Ex) $10.011 \times 2^1 = 1.0011 \times 2^2$
  - Ex) $0.01101 \times 2^1 = 1.101 \times 2^{-1}$
- 수정이 완료된 mantissa값과 exponent값을 사용하여 결과를 도출
  - S = 두 floating point 중 절대값이 큰 값의 sign bit
  - Exponent = Normalize result exponent + 127
  - Fraction = mantissa

# Problem

- Problem
  - Floating point값 2개를 메모리로 부터 load하여 덧셈 연산을 수행
  - 메모리에서 불러오는 값은 **실수**
    - 양수, 음수 및 0값이 가능
  - 결과값은 다음 메모리 주소에 저장
    - 0x40000000

# Homework

- 제출기한
  - Soft copy
    - 2019.10.25(금) 16시 29분 59초까지 u-campus에 제출
    - 압축 파일은 각 과제 파일을 모아서 제출(ini파일 반드시 포함)
    - 압축파일 형식
      - Assignment_(번호)_(학번)_(이름).zip
        - ex) Assignment_4_2012722069_홍길동.zip
    - 하드카피 제출 X

# Q&A

Thank you