Computer Architecture

Fall, 2022
Week 8
2021.10.31

組別：__________ 簽名：____
____

Group6

Which of the following statements are true?

(A) 0 10000101 01001100000000000000000 in IEEE 754 represents 83.375 in decimal.

(B) 1 10000010 10010100000000000000000 in IEEE 754 represents -12.625 in decimal.

(C)If some values (nonzero) are divided by zero, MIPS will raise an exception.

(D)In bias 15, 01101 represents -2.

ANS : B、D

(A) 83

(C) MIPS don’t check.

Group14

Please explain/fill in the following according to the IEEE 754 standard:

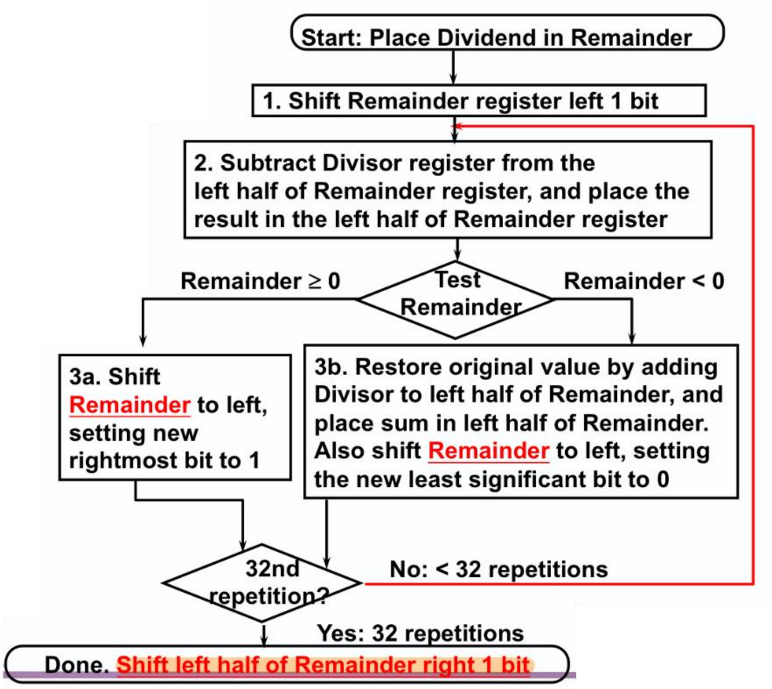
a. Why is there a need for the designation of denormal (subnormal numbers) in the standard?
b. Why does the value of the mantissa (significand) always ignores the digit left to the decimal point?
c. Why is there a need for the biased notation (instead of a 2's complement representation for signed numbers)?
d. How many different NaN values are there in the single-precision floating point number standard? (Answer can be presented in exponent notation)

Ans:

a. To allow a **gradual underflow** from the least significant (i.e. lowest in absolute value, $\pm 1.0\times 2^{-126}$) normal number to zero, instead of jumping straight to zero.
b. Since a binary floating point number, when represented in a scientific notation, can only start with 1 (values smaller than 1 will be represented with a lower exponent), such digit is ignored in the normalized form.
c. It is implemented for the ease of comparing different exponents.
d. There are NaNs in the positive and negative range, and there is a range of 23 bits in the mantissa (significand), hence $2\times(2^{23} - 1) = 2^{24} - 2$.

Group 2

Please fill out the table according to steps of 1011/0110 and the following flow chart. Write down Quotient and Remainder.



Step	Remainder		Divisor	Description
0	0000	1011	0110	Initialization

Quotient: Remainder:

Hint:

To fill the Description, there are some options:

- Shift xxxxxxxxx left/right
- xxxxxxxxx < 0 / > 0
- Restore original value
- Subtract/Add xxxxxxx
- Set the new significant bit to 1/0

……

Ans:

0110->2’s complement->1001+1->1010

Step	Remainder		Divisor	Description
0	0000	1011	0110	Initialization
1.1	0001	0110		Shift Remainder left
1.2	1011	0110		Subtract Divisor -> Remainder < 0
1.3b	0010	1100		Restore original value Shift Remainder left Set the new significant bit to 0
2.2	1100	1100		Subtract Divisor -> Remainder < 0
2.3b	0101	1000		Restore original value Shift Remainder left Set the new significant bit to 0
3.2	1111	1000		Subtract Divisor -> Remainder < 0
3.3b	1011	0000		Restore original value Shift Remainder left Set the new significant bit to 0
4.2	0101	0000		Subtract Divisor -> Remainder > 0
4.3a	1010	0001		Shift Remainder left Set the new significant bit to 1
	0101	0001		Shift left half of Remainder right 1 bit

Quotient: 0001 Remainder: 0101

Group4

Which of the following statements are true?

(a) The unsigned multiplier of two 32-bit numbers requires a 32-bit register for multiplicand and a 32-bit register for product.

(b) Based on 32-bit IEEE 754 standard's single precision, no other floating point number is greater than 0x7f800000.

(c) Hi and Lo registers are used in both multiplication and division, and Hi would store the quotient in division.

(d) If there were only 16 bits for significand field in floating point representation, it is equivalent to 4 decimal digits of precision.

(e) For 32-bit unsigned division, we only need 32 iterations and shift one register to get the correct result.

(f) By IEEE-754 single precision floating-point representation, the largest positive normalized number is $+(1-2^{-23}) \times 2^{+127}$.

(g) Exponents with all 1's are reserved for $\pm\infty$ and NaN.

Ans: (b)(d)(e)(g)

(a) In version 1, a 32-bit multiplier requires a 64-bit multipland register and a 64-bit product register. In version 2, a 32-bit multiplier requires a 32-bit multipland register and a 64-bit product register.

(b) 0 and 255 are reserved in exponent value. 255 in exponent and 0 in significand stands for +/- infinity. Hence, 0111 1111 1000 0000 0000 0000 0000 0000 means infinity. In hexadecimal representation is 0x7f800000

(c) Hi stores the remainder.

(d) $16 \times log2 \approx 4$ decimal digits of precision.

(f) The largest positive number= $(1+1-2^{-23}) \times 2^{+127} = (1-2^{-24}) \times 2^{128}$

Group12

True or False:

A. when we use mult \$t1, \$t2, we will push most significant 32 bits to lo and least significant 32 bits to hi.

B. In multiply version 2 we will place multiplier to product register's right hand and shift right until the multiply end.

C. Divide version 1 and multiply version 1 have same repetition times.

D. when we use div \$t1, \$t2, we will push remainder to hi and quotient to lo, and we can use mflo \$t3 and mfhi \$t4 to copy the lo and hi value to register t3 and t4.

E. For 32-bit IEEE 754 floating-point standard, the smallest positive single precision denormalized number is: $0.0000\ 0000\ 0000\ 0000\ 0000\ 001_2$ x 2 ^ -126.

F. $0.6875_{10} = 0.0111_2$

G. In the IEEE 754 floating-point representation, the precision of represented numbers is determined by the size of exponent.

H. In the IEEE 754, we use 2's complement in exponent field.

Ans:

A. F, when we use mult \$t1, \$t2, we will push most significant 32 bits to hi and least significant 32 bits to lo.

B. T

C. F, Divide version 1 need to do 33 repetitions, and multiply version1 need to do 32 repetitions.

D. T

E. T

F. F, 0.687510 = 0.10112

G. In the IEEE 754 floating-point representation, the precision of represented numbers is determined by the size of significand.

H. F, In the IEEE 754, we use bias notation in exponent field .

Group1

Below are some steps for performing a basic floating-point multiplication. Please order the steps.

a. Normalize the product and check for overflow/underflow when shifting
b. Add the exponents of operands to get the exponent of the product
c. Round the mantissa and renormalize when necessary
d. Multiply the mantissa of operands
e. Set the sign of the product

Ans: bdace

Explanation: Please refer to the slides on page 112 (titled Floating-Point Multiplication)

Group 7

Half-precision floating-point (FP16) has 1 bit of signed bit, 5 bits of exponent, and 10 bits of mantissa. The exponent uses bias of 15.

S	Exponent					Significand									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

For the following question, calculate the results and represent them in FP16 bit representation:

1) $13_{(10)}$
2) $1.111_{(2)}$ * 2^-14 - $1.000_{(2)}$ * 2^-13
3) $1024_{(10)}$ * $512_{(10)}$

(hint : 512 = 2^9, 1024 = 2^10)

1) 13 = 0b1101 = 1.101 * 2^3
2) 1.111 * 2^-14 - 1.000 * 2^-13 = -0.0001 * 2^-13 = -0.001 * 2^-14 (denormalized)
3) 1024 * 512 = 2^19 → +Infinity (overflow)

Representing them with FP16 bit representation:

#	S	Exponent					Significand									
1)	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0
2)	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3)	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0