

組別: _____ 簽名: _____

[group 1]

1. In MIPS, after executing the instruction `div $t1, $t2`, where are the quotient and remainder stored? Which instructions can be used to access them?

Ans:

Quotient: Stored in the LO register and accessed using `mflo`.

Remainder: Stored in the HI register and accessed using `mfhi`.

[group 14]

2. Why does IEEE 754 allow subnormal (denormalized) numbers instead of rounding every value smaller than the smallest normal number directly to zero?

Ans:

Subnormals fill the gap between zero and the smallest normal number. They let a calculation's result become gradually less precise as it approaches zero (gradual underflow), rather than abruptly becoming zero. For example, single precision cannot store 2^{-140} as a normal number, but can store it as a subnormal number.

[group 5]

3. Why does IEEE 754 store the exponent in biased notation instead of two's complement? Use $1/2$ and 2 as an example.

Ans:

Biased notation makes the unsigned order of stored exponents match their numerical order, simplifying comparisons of positive floating-point numbers.

With an 8-bit two's complement exponent, the exponent of $1/2$ (-1) would be stored as `1111 1111`, and the exponent of 2 ($+1$) as `0000 0001`. An unsigned integer comparison of these bit patterns would incorrectly place $1/2$ above 2 .

With a bias of 127 , the stored exponents are 126 (`0111 1110`) and 128 (`1000 0000`), respectively. Their unsigned integer order therefore matches the real order: $1/2 < 2$.

[group 3]

4. Using the IEEE 754 floating point format, write down the bit pattern that would represent $-1/4$. Can you represent $-1/4$ exactly?

Ans:

$$-1/4 = -0.01_2 = -1.0_2 \times 2^{-2}$$

Sign = 1

$$\text{Exponent} = -2 + 127 = 125 = 01111101_2$$

Fraction = 0 (23 zeros)

1 01111101 000000000000000000000000

= `0xBE800000`

(Double precision: 1 0111111101 000...0 = `0xBFD0000000000000`)

Exact? Yes. $-1/4 = -2^{-2}$ is a power of 2, so the fraction is all zeros and there is no rounding error.

[group 4]

5. Convert the decimal number +6.5 into the IEEE 754 single-precision (32-bit) floating-point representation. Find the: Sign bit Exponent (8 bits) Fraction (23 bits)

Ans:

Step 1. Convert to binary, $6.5_{10} = 110.1_2$

Step 2. Normalize (把小數點往左移兩位), $110.1_2 = 1.101_2 \times 2^2$

Step 3. Determine each field:

Sign = 0 (positive)

Exponent = $2 + 127 = 129 = 10000001$

Fraction = 10100000000000000000000

[group 5]

6. A computer uses the shift-and-add multiplication method to calculate 6×5 using 4-bit unsigned binary.

(a) Convert 6 and 5 into 4-bit binary.

(b) Use the shift-and-add method to show how the computer gets the final answer. Explain when the multiplicand is added and why shifting is needed.

(c) When multiplying two 4-bit unsigned numbers, how many bits may be needed to store the result? Explain why.

Ans:

(a) $6 = 0110$; $5 = 0101$

(b) Since $0101 = 2^2 + 2^0$:

$$6 \times 5 = (6 \times 2^2) + (6 \times 2^0) = 24 + 6 = 30$$

Binary process:

00000110 $\leftarrow$ Add 6 because bit 0 = 1

00000000 $\leftarrow$ Do not add because bit 1 = 0

+ 00011000 $\leftarrow$ Add 6 shifted left twice because bit 2 = 1

00011110

00011110 = 30. Bit 3 is 0, so it contributes nothing. The multiplier bits decide whether the multiplicand should be added. Shifting left is used because each next bit represents a higher power of 2.

(c) Up to 8 bits may be needed. The largest 4-bit unsigned number is $1111 = 15$. Since $15 \times 15 = 225$, and $225 = 11100001$ in binary, the result needs 8 bits.

[group 14]

7. Represent -3 and 5 as 4-bit two's-complement numbers. Use shifted partial products to find their full 8-bit product.

Ans:

-3 = 1101, 5 = 0101; sign-extend -3 to eight bits

Since 0101 has 1s in positions 0 and 2:

11111101 -3

+ 11110100 -3 shifted left 2 = -12

11110001 -15

Product: 11110001

[group 11]

8. For IEEE 754 floating-point representation, what are the maximum and minimum positive representable values for single-precision (32-bit) floating-point numbers, considering both normalized and denormalized (subnormal) numbers?
Please provide the answers in scientific notation and explain how the exponent and fraction fields determine these values.

Ans:

According to the IEEE 754 standard, floating-point numbers are represented using three fields: the sign bit, the exponent field, and the fraction field.

1. Single-Precision (32-bit)

Sign bit: 1 bit

Exponent: 8 bits, bias = 127

Fraction: 23 bits

A. Normalized Numbers

For normalized numbers, the exponent field ranges from 1 to 254, and the significand has an implicit leading 1.

The value is calculated as:

$$(-1)^s * (1.f)_2 * 2^{E-127}$$

Maximum positive normalized value:

$$(2-2^{-23}) * 2^{127}$$

Minimum positive normalized value:

$$1.0 * 2^{1-127} = 2^{-126}$$

B. Denormalized (Subnormal) Numbers

For subnormal numbers, the exponent field is zero, and there is no implicit leading 1. The effective exponent is $(1-127)=-126$.

The value is calculated as:

$$(-1)^s * (0.f)_2 * 2^{-126}$$

Maximum positive subnormal value:

$$(1-2^{-23}) * 2^{-126}$$

Minimum positive subnormal value:

$$2^{-23} * 2^{-126} = 2^{-149}$$

[group 3]

9. Interpret the following double precision bit pattern as a decimal number:

(a) 0x 402A 0000 0000 0000

(b) 0x 800D 0000 0000 0000

(answer in form: $\pm \text{significand} * 2^{\text{exponent}}$, where significand is in decimal)

Ans:

(a) sign (1 bit): 0 $\Rightarrow$ positive

exponent (11 bit): $10000000010_2 = 2^{10} + 2^1 = 1026$ $\text{exp} = 1026 - 1023 = 3$

significand (52 bit): $1.10100... = 2^0 + 2^{-1} + 2^{-3} = 1.625$

$$+1.625 * 2^3$$

(b) sign (1 bit): 1 $\Rightarrow$ negative

exponent (11 bit): $00000000000_2 \Rightarrow$ (subnormal) $\text{exp} = -1022$

significand (52 bit): $0.11010... = 2^{-1} + 2^{-2} + 2^{-4} = 0.8125$

$$-0.8125 * 2^{-1022}$$

[group 12]

10. Let $\$t1 = -29$ and $\$t2 = 5$.

(i) `div $t1, $t2` is performed. What are LO and HI? Where is the answer currently stored?

(ii) `divu $t1, $t2` is now performed. What are LO and HI now? Compare with (i) and explain if they're different.

(iii) A student then writes the following MIPS code to copy HI into $\$t3$ and LO into $\$t4$. Would this work? If not, give the correct code.

```
add $t3, HI, $zero
```

```
sw LO, 0($t0)
```

```
lw $t4, 0($t0)
```

Ans:

(i)

LO = -5 (0xFFFFFFF5) HI = -4 (0xFFFFFFF4)

The answer is stored in the special HI/LO registers.

(ii)

LO = 858,993,453 (0x3333332D) HI = 2 (0x00000002)

they are very different since `div` reads the top bit as a sign bit, so the value is -29 . `divu` has no sign bit, so it would read -29 (0xFFFFF5) as 4,294,967,267 instead.

(iii)

It will not work as both `add $t3, HI, $zero` and `sw LO, 0($t0)` is invalid as `add` and `sw` takes only general-purpose registers ($\$0-\$31$) as operands, and HI and LO are not one of them.

```
mfhi $t3
```

```
mflo $t4
```

[group 2]

11. Given two floating-point numbers:

$$A = 1.010_2 * 2^2$$

$$B = -1.100_2 * 2^{-1}$$

Calculate $A * B$. Show the following steps:

1. Calculate the resulting exponent.

2. Multiply the mantissas.

3. Determine the sign of the result.

4. Write the final answer in normalized binary floating-point form

Ans:

1. Calculate the resulting exponent:

$$2 + (-1) = 1$$

2. Multiply the mantissas:

$$1.010_2 * 1.100_2 = 1.111_2$$

3. Determine the sign:

Positive * Negative = Negative

4. Final answer in normalized binary floating-point form:

$$A * B = -1.111_2 * 2^1 = -3.75_{10}$$