perplexity

how 42.75 binary will be rep as sign, eexponent and mantessa, and will 1 alwasy be added, do we weite digitsa after decimal in mantessa and what will we write in exponent, do we always add 1023 and why

Here's a clear, step-by-step explanation of how the number **42.75** is represented in binary floating-point format, focusing on the **sign, exponent, and mantissa (fraction)** fields, normalization, the "hidden 1", and exponent bias, especially for double-precision (64-bit, IEEE 754) numbers:

1. Step 1: Convert 42.75 to Binary

- **Integer part (42):**
  $42_{10} = 101010_2$
- **Fractional part (0.75):**
  $0.75 \times 2 = 1.5 \Rightarrow 1$
  $0.5 \times 2 = 1.0 \Rightarrow 1$
  $\rightarrow .11_2$

So, 42.75_{10} = $\mathbf{101010.11_2}$

2. Step 2: Normalize to the form 1.xxxxx × 2^n

- Move the binary point so there's a single `1` before the point:
  - $101010.11_2 = 1.0101011 \times 2^5$

  (We moved the point 5 places left.)

3. Step 3: IEEE 754 Representation (Double Precision: 64 bits)

A. Sign Bit

- **0** for positive numbers, **1** for negative.
- 42.75 is positive ⇒ **sign = 0**

B. Exponent Field

- In IEEE 754 double-precision (64-bit), the exponent is **11 bits**, and is written as a "biased" value.
- **Bias (for double precision) = 1023**
- **Exponent field = Actual exponent + Bias**
  - Actual exponent (from step 2) = 5
  - So, Exponent = 5 + 1023 = 1028
- **In binary:**
  $1028_{10} = 10000000100_2$
  (This is the value you write in the exponent field.)

Q: Do we always add 1023?

- **Yes, for double-precision.** In single-precision, the bias is 127. The bias allows for both positive and negative exponents.

C. Mantissa (Fraction Field)

- After normalization, only the fractional binary digits (digits **after** the point) are written in the mantissa field.
- The leading `1.` (the "hidden 1") is **not stored** (unless the number is subnormal/denormalized).
  This "implicit 1" saves space.

> **For 1.0101011×2^5:**
> Mantissa bits: **0101011...** (write remaining 52 bits as zeroes for double precision)

- **So:**
  - Mantissa = `010101100000......` (up to 52 bits for double precision)

4. Summary Table

Field	Value for 42.75	How to Get It
Sign	0	Positive number
Exponent	10000000100 (binary)	5 + 1023 = 1028 → binary form
Mantissa	010101100000...(52b)	Fractional part after normalizing (`.0101011...`)

5. Answers to Your Specific Questions

- **Will 1 always be "added"?**
  - Yes, for normal (not denormal) values in IEEE 754, computers assume ("hide") the leading `1`—only the fraction after the point is stored.
- **Do we write digits after the decimal in the mantissa?**

- Yes, all bits after the 1. (“binary decimal”) go in the mantissa field—pad with zeros to the needed length (52 for double, 23 for single).
- **What goes in the exponent?**
  - The actual exponent after normalization (here, 5) **plus the bias** (1023 for double, 127 for single), encoded as an unsigned binary value.
- **Do we always add 1023?**
  - **Yes, for double-precision (64-bit) floating-point.** This is called the “exponent bias” and it lets us represent both large and small exponents (positive and negative).

6. Putting It All Together (for 42.75 in IEEE 754 double)

Field	Bits
Sign	0
Exponent	10000000100 (11 bits, for value 1028)
Mantissa	0101011000000000000000000000000000000000000000000000

In Summary:

- IEEE 754 floating-point splits a number into: `sign | exponent (with bias) | mantissa (fractional part after normalization, no leading 1)`
- The bias (1023 for double-precision) is **always added** to the *actual* exponent.
- The mantissa stores only the digits after the “binary decimal”, right after the normalized leading 1.

Let me know if you want the **hexadecimal** representation, all 64 bits, or a step-by-step for another number!