Part 1: Foundations of Binary and Hexadecimal

1.1 Why Binary?

Computers use transistors which act as switches. These switches have two states: ON (1) and OFF (0). This base-2 system is the language of hardware.

1.2 Base Conversions (Decimal to Binary)

Problem: Convert (173)_{10} to Binary.

Solution:

173 \div 2 = 86 remainder 1

86 \div 2 = 43 remainder 0

43 \div 2 = 21 remainder 1

21 \div 2 = 10 remainder 1

10 \div 2 = 5 remainder 0

5 \div 2 = 2 remainder 1

2 \div 2 = 1 remainder 0

1 \div 2 = 0 remainder 1

Result (Reading bottom to top): 10101101

1.3 Hexadecimal: The Programmer's Shortcut

Hexadecimal (Base-16) is used because 1 Hex digit perfectly represents 4 Binary bits (a "nibble").

Problem: Convert (10101101)_2 to Hex.

Solution:

Split into groups of 4: 1010 and 1101.

1010 = 10 \rightarrow A

1101 = 13 \rightarrow D

Result: 0xAD

📄 DATA BLOCK 2: Negative Numbers & Floating Point

(Copy everything below this line)

Part 2: Advanced Storage Logic

2.1 2's Complement (Signed Integers)

To store a negative number, computers use 2's complement so that addition and subtraction use the same circuitry.

Problem: Store -42 in an 8-bit system.

Positive 42 in binary: 00101010

1's Complement (Flip bits): 11010101

2's Complement (Add 1): 11010101 + 1 = 11010110

Final Answer: -42 is stored as 11010110.

2.2 IEEE 754 (Decimals/Floating Point)

Standard 32-bit floats are stored in three parts:

Sign (1 bit): 0 for +, 1 for -

Exponent (8 bits): Power of 2 (plus a bias of 127)

Mantissa (23 bits): The actual digits

Step-by-Step Problem: 5.75 to Float

Integer: 5 = 101

Fraction: 0.75 = .11

Combine: 101.11 \rightarrow Normalize to 1.0111 \times 2^2

Exponent: 2 + 127 = 129 (10000001)

Mantissa: 0111000...

Result: 0 10000001 01110000000000000000000

📄 **DATA BLOCK 3: Memory Addressing & Endianness**

(Copy everything below this line)

Part 3: CPU and RAM Architecture

3.1 Memory Addressing

Every byte in RAM has a Hex address. If a system is 64-bit, it means the "pointers" (addresses) are 64 bits long.

3.2 Endianness (Byte Order)

If you have a 4-byte number 0x12345678:

Big Endian: Stores as 12 34 56 78 (Most significant first)

Little Endian: Stores as 78 56 34 12 (Least significant first - used by Intel/AMD)

3.3 Pointer Arithmetic Practice

Problem: You have an array of integers (4 bytes each). The first element is at address 0x1000. What is the address of the 4th element?

Solution:

Index of 4th element is 3 (0, 1, 2, 3).

3 \text{ elements} \times 4 \text{ bytes} = 12 \text{ bytes offset}.

Convert 12 to Hex: 12 = \text{C}.

Final Address: 0x100C.

Helpful Addition: Generating the 20 Pages

To ensure your PDF is over 20 pages, I recommend adding these three things to your Word document:

Table of Contents: This will take 1 full page.

Glossary: Define terms like Bit, Byte, Word, Nibble, Overflow, Underflow, RAM, Register. (Approx. 2 pages).

Full Binary-Hex-Decimal Table: Create a table for numbers 1 to 100. (Approx. 4-5 pages).