FREE EDITION · NOTES + 3 SAMPLE MCQS

CUET · COMPUTER SCIENCE · CLASS XI · CODE 308

Encoding Schemes and Number System

CUET unit: Encoding Schemes and Number System

By **UniDrill** · NCERT-grounded study material

WWW.UNIDRILL.IN

UniDrill

📌 Snapshot

- Human-readable text is converted to binary form that computers process, using standard encoding schemes (ASCII, ISCII, UNICODE).
- Four number systems are used in computing — decimal, binary, octal, and hexadecimal — each positional.
- Conversion algorithms exist between every pair of number systems, including numbers with fractional parts.
- CUET tests this area heavily through conversion calculations, encoding scheme properties, and number-system identification questions.
- Hexadecimal applications (memory addressing, colour codes) are a favourite source of applied/case-based MCQs.

📖 Detailed Notes

2.1 Core concepts

- **Encoding — why it exists.** When a key is pressed on a keyboard, the computer internally maps it to a unique decimal code (e.g., 'A' → 65), which is then converted to binary (65 → 1000001) for processing. This mapping mechanism is called encoding. The same code is used across all keyboards because of standardised encoding schemes. (NCERT §2.1, p. 27)
- **ASCII (American Standard Code for Information Interchange).** Developed in the early 1960s to standardise character representation across computers. Originally used 7 bits, allowing 2^7 = 128 unique characters covering the English keyboard. It is still the most commonly used coding scheme. ASCII encodes only the English language character set. (NCERT §2.1.1, p. 28)
- **ISCII (Indian Script Code for Information Interchange).** Developed in India in the mid-1980s to support Indian languages on computers. It is an 8-bit code (2^8 = 256 characters). It retains all 128 ASCII codes and uses the remaining 128 codes for Indian language characters. Additional codes (160–255) are assigned for aksharas of the language. (NCERT §2.1.2, p. 29)
- **UNICODE.** Because ASCII- and ISCII-based schemes could not communicate with each other (each represented characters in its own way), UNICODE was developed as a universal standard. UNICODE provides a unique number for every character, regardless of device, operating system, or application. Commonly used UNICODE

encodings are UTF-8, UTF-16, and UTF-32. UNICODE is a superset of ASCII; values 0–128 have the same characters as ASCII. The Devanagari script Unicode range starts at hexadecimal 0900 and ends near 097F. (NCERT §2.1.3, pp. 29–30)

- **Number system — definition and base/radix.** A number system is a method to represent numbers. Every number system has a set of unique characters or literals; the count of these literals is called the radix or base. Number systems are positional: the value of each digit depends on both the digit itself and its position (position value = base^position number). (NCERT §2.2, p. 30)
- **Decimal number system.** Base-10; uses digits 0–9. Positional value is expressed as powers of 10. The rightmost integer digit has position 0, increasing by 1 to the left; the first fractional digit has position –1, decreasing to the right. (NCERT §2.2.1, p. 31)
- **Binary number system.** Base-2; uses only digits 0 and 1, corresponding to the OFF/ON state of transistors in ICs. Examples: 1001011, 1011.101. A binary number can be mapped to an equivalent decimal number. (NCERT §2.2.2, p. 32)
- **Octal number system.** Base-8; uses digits 0–7. Devised for compact representation of binary numbers. Three binary digits are sufficient to represent any octal digit because 8 = 2^3. Examples: (237.05)_8, (13)_8. (NCERT §2.2.3, pp. 32–33)
- **Hexadecimal number system.** Base-16; uses 16 symbols (0–9 and A–F). Each hexadecimal digit represents a group of 4 binary digits because 2^4 = 16. Decimal values 10–15 are represented by letters A–F. Examples: (23A.05)_16, (1C3)_16. (NCERT §2.2.4, p. 33)
- **Applications of hexadecimal.** (1) Memory addresses: a 16-bit address like 1100000011110001 is compactly written as C0F1 in hexadecimal. (2) Web colour codes: each colour is defined by three 8-bit components (Red, Green, Blue); the 24-bit binary code is written in hexadecimal — e.g., RED = (FF,00,00)_16. (NCERT §2.2.5, p. 34)
- **Decimal to other number systems — conversion method.** Divide the decimal integer repeatedly by the target base; collect remainders from bottom to top. For fractional parts, repeatedly multiply by the base; collect integer parts from top to bottom. (NCERT §2.3.1, pp. 35–37)
- **Other number systems to decimal.** Multiply each digit by its positional value (base^position) and add all products. For fractional digits, use negative position numbers. (NCERT §2.3.2, pp. 37–38)
- **Binary to/from octal and hexadecimal — shortcut.** Group binary digits into sets of 3 (for octal) or 4 (for hexadecimal) from right to left for integer part and left to right for fractional part; replace each group with its equivalent symbol. Reverse the process for octal/hexadecimal to binary. (NCERT §2.3.3, pp. 38–40)
- **Fractional number conversions.** For non-decimal to decimal with fractions, use negative powers of the base. For fractional binary to octal/hexadecimal, group

fractional bits left to right in groups of 3 or 4, padding with 0s on the right as needed. (NCERT §2.3.4, pp. 40–42)

- **Encoding versus encryption — note.** The NCERT text uses the word cipher for encoded data and notes that "encoded data is also called encrypted data" (p. 28). However, in modern usage encoding (a public reversible mapping, like ASCII) differs from encryption (key-based confidentiality). For CUET, treat NCERT's terminology as authoritative: encoded data → cipher → can be reverted via decoding using the same standard table. (NCERT §2.1, p. 27-28)
- **Why three different number systems coexist.** Binary is dictated by hardware (transistor ON/OFF); decimal is dictated by human convention; octal and hexadecimal are practical compromises that compact long binary strings — every 3 binary bits ↔ 1 octal digit, every 4 binary bits ↔ 1 hexadecimal digit. This is why a 16-bit address that takes 16 binary characters becomes just 4 hexadecimal characters. (NCERT §2.2.5, p. 34)
- **Worked decimal → binary example.** To convert (65)_10, divide repeatedly: 65/2 = 32 r 1; 32/2 = 16 r 0; 16/2 = 8 r 0; 8/2 = 4 r 0; 4/2 = 2 r 0; 2/2 = 1 r 0; 1/2 = 0 r 1. Reading remainders bottom-up gives 1000001 — the ASCII code of 'A' in binary form. (NCERT Figure 2.5, p. 35)

2.2 Definitions to memorise

Term	Definition	Page
Encoding	Mechanism of converting data into an equivalent cipher using a specific code	27
Cipher	Data converted to coded form to hide/conceal it; also called encryption	28
Decoding	Reverse process of encoding: converting a coded representation back into the original data	27
ASCII	American Standard Code for Information Interchange; 7-bit code for 128 characters of the English keyboard	28
ISCII	Indian Script Code for Information Interchange; 8-bit code (256 characters) for Indian languages, developed in India in the mid-1980s	29
UNICODE	Universal character encoding standard providing a unique number for every character of every written language; superset of ASCII	29–30
UTF-8	Variable-length UNICODE encoding using 1–4 bytes; ASCII characters fit in 1 byte	30
UTF-16	UNICODE encoding using 16-bit code units; characters outside the BMP use a 32-bit pair	30
UTF-32	UNICODE encoding using fixed 32 bits per character	30

Term	Definition	Page
Aksharas	Indian language characters assigned codes 160–255 in the ISCII scheme	29
Number system	A method to represent numbers, defined by a set of unique literals (the base/radix)	30
Radix / Base	Count of unique literals in a number system (e.g., base 2 for binary, base 16 for hexadecimal)	30
Positional number system	System where each digit's value depends on its position within the number	31
Positional value	Value computed as base^position_number for a digit at a given position	31
Most Significant Bit (MSB)	The leftmost bit of a binary number — has the highest positional value	31–32
Least Significant Bit (LSB)	The rightmost bit of a binary number — positional value 2^0	31–32
Decimal number system	Base-10 positional system using digits 0–9; everyday human numerals	31
Binary number system	Base-2 number system using digits 0 and 1; represents ON/OFF states of transistors	32
Octal number system	Base-8 number system using digits 0–7; used for compact binary representation	32
Hexadecimal number system	Base-16 number system using symbols 0–9 and A–F; each symbol = 4 binary bits	33
Repeated division method	Algorithm to convert a decimal integer to another base: divide by base, collect remainders bottom-up	35
Repeated multiplication method	Algorithm to convert a decimal fraction to another base: multiply by base, collect integer parts top-down	36–37
Web colour code	24-bit RGB code expressed in 6-hex-digit form (e.g., FF0000 = red)	34
Memory address	A binary number identifying a storage location; commonly written compactly in hexadecimal	34
Devanagari Unicode range	UNICODE block starting at hexadecimal 0900 ending near 097F for Devanagari script	30

2.3 Diagrams / processes to remember

- **Figure 2.1 (p. 27):** Shows the chain: keyboard key 'A' → code value 65 (decimal) → 0100 0001 (binary). Illustrates the complete encoding pathway visually.

- **Figure 2.2 (p. 31):** Diagram of the four number systems used in computers — Binary (base 2, digits 0,1), Decimal (base 10, digits 0–9), Octal (base 8, digits 0–7), Hexadecimal (base 16, digits 0–9 and A–F).
- **Figure 2.3 (p. 31):** Positional value computation for decimal number 123.45 — shows how position numbers and positional values (powers of 10) are assigned to each digit.
- **Figure 2.4 (p. 32):** Positional value representation of 237.25 in decimal, distinguishing integer part (positive powers of 10) and fractional part (negative powers of 10).
- **Figure 2.5 (p. 35):** Step-by-step repeated-division method for converting (65)_10 to binary, showing (65)_10 = (1000001)_2.
- **Figure 2.6 (p. 36):** Repeated-division method for (65)_10 to octal, giving (101)_8.
- **Figure 2.7 (p. 37):** Repeated-division method converting (65)_10 to hexadecimal, giving (41)_16. The step-by-step division produces remainder 1 then 4; read the remainders bottom to top to get (41)_16. Students must always read the remainders bottom to top.
- **Table 2.3 (p. 30):** Unicode table for Devanagari script — characters with their hexadecimal values in the range 0900–097F.
- **Table 2.7 (p. 34):** Colour codes in decimal, binary, and hexadecimal — Black (00,00,00), White (FF,FF,FF), Yellow (FF,FF,00), Grey (80,80,80).

2.4 Common confusions / NTA trap points

- **Reading remainders in wrong order (NCERT §2.3.1, p. 35).** When converting decimal to binary/octal/hexadecimal by repeated division, remainders must be read from bottom to top (last remainder is the most significant digit). Students often read top to bottom and get the mirror-image wrong answer.
- **ASCII vs ISCII bit-width confusion (NCERT §2.1.1-2.1.2, pp. 28-29).** ASCII is 7-bit (128 characters); ISCII is 8-bit (256 characters). NTA distractors often swap these values or claim ASCII is 8-bit.
- **UNICODE encoding names (NCERT §2.1.3, p. 30).** UTF-8, UTF-16, and UTF-32 are formats of UNICODE, not separate standards. A common trap question claims they are independent of UNICODE or that UTF-8 is the only valid encoding.
- **Octal grouping direction (NCERT §2.3.3, pp. 38-39).** For integer parts, group binary digits 3-at-a-time from right to left; for fractional parts, group left to right (adding trailing 0s if needed). Mixing up the direction gives wrong answers — a frequent NTA-style trap.
- **Hexadecimal letters as digits (NCERT §2.2.4, p. 33).** A = 10, B = 11, C = 12, D = 13, E = 14, F = 15. Students forget to substitute the correct decimal value when converting (e.g., (3A5)_16 to decimal: A must be treated as 10, not as the letter A or as 1).

- **Fractional repeated-multiplication direction (NCERT §2.3.1(D), p. 36-37).** Integer parts produced by repeated multiplication for the fractional half are collected top-to-bottom — opposite to the repeated-division direction for the integer half. Reversing this is the second-most-common arithmetic error.
- **ASCII ≠ ISCII subset relationship (NCERT §2.1.2, p. 29).** It is the other way: ISCII retains all 128 ASCII codes in its first half and adds 128 Indian characters in the second half. UNICODE is a superset of ASCII (NCERT §2.1.3, p. 30); ISCII is NOT a superset of UNICODE.
- **Decimal '10' vs hex 'A' (NCERT §2.2.4, p. 33).** In hexadecimal, the literal digit at value-10 is 'A' (a single symbol), not the two-character string "10". Writing (10)_16 means decimal 16, not decimal 10. NTA traps exploit this notational slip.
- **Padding fractional binary groups with the WRONG side (NCERT §2.3.3(B), p. 40).** For binary fractions converted to octal/hex, add padding 0s on the RIGHT (after the last fractional bit), because the place values continue rightwards. Padding on the left changes the magnitude.
- **ENIAC vs Pascaline (NCERT §1.2 link).** Pascaline (1642) is mechanical, not binary programmable. NTA may swap the two when asking about the origin of binary computing.
- **Word colour code direction (NCERT §2.2.5, p. 34).** RGB hex is written R-G-B in that order. (FF, 00, 00) is red, not blue. Students sometimes assume the trailing pair is red.
- **(0)_2 vs (0)_16 both equal decimal 0.** A trap question may ask "how many distinct decimal values can 8 bits represent" — the answer is 256 (0 to 255), not 255. Students often forget to count the value 0.

2.5 Code examples

The NCERT chapter is conceptual but the conversion algorithms map cleanly to Python. The snippets below are written in the spirit of NCERT's Python 3 usage and use only standard library features.

```
# 1) ASCII code of a character (NCERT §2.1, p. 27)
ch = 'A'
print(ord(ch))      # 65
print(bin(ord(ch))) # 0b1000001  (the 7-bit ASCII code)
print(chr(97))      # 'a' — reverse mapping

# 2) Decimal to binary via repeated division (NCERT §2.3.1(A), p. 35)
def dec_to_bin(n):
    if n == 0:
        return "0"
    bits = []
    while n > 0:
```

```
        bits.append(str(n % 2))
        n = n // 2
    return "".join(reversed(bits))

print(dec_to_bin(65))   # '1000001'
print(dec_to_bin(255))  # '11111111'

 # 3) Decimal to hexadecimal (NCERT §2.3.1(C), p. 35-36)
def dec_to_hex(n):
    symbols = "0123456789ABCDEF"
    if n == 0:
        return "0"
    digits = []
    while n > 0:
        digits.append(symbols[n % 16])
        n = n // 16
    return "".join(reversed(digits))

print(dec_to_hex(255))   # 'FF'
print(dec_to_hex(49393)) # 'C0F1'  — same as the NCERT 16-bit address example

 # 4) Binary to decimal using positional values (NCERT §2.3.2(A), p. 37)
def bin_to_dec(b):
    total = 0
    for i, bit in enumerate(reversed(b)):
        total += int(bit) * (2 ** i)
    return total

print(bin_to_dec("1101"))   # 13
print(bin_to_dec("1000001"))# 65

 # 5) RGB hex colour code from R, G, B integers (NCERT §2.2.5, p. 34)
def rgb_hex(r, g, b):
    return "{:02X}{:02X}{:02X}".format(r, g, b)

print(rgb_hex(255, 0, 0))    # 'FF0000' — red
print(rgb_hex(255, 255, 0))  # 'FFFF00' — yellow
print(rgb_hex(128, 128, 128))# '808080' — grey
```

🎯 Practice MCQs

Q1. When the key 'A' is pressed on a computer keyboard, it is internally mapped to which decimal code value before being converted to binary?

A. 61

B. 64

C. 65

D. 97

Q2. Which of the following statements about ASCII and ISCII is/are correct? (i) ASCII uses 7 bits and can represent 128 characters. (ii) ISCII uses 8 bits and can represent 256 characters. (iii) ISCII was developed in the early 1960s in the USA. (iv) ISCII retains all 128 ASCII codes.

A. (i) and (ii) only

B. (i), (ii) and (iv) only

C. (ii) and (iii) only

D. (i), (ii), (iii) and (iv)

Q3. A programmer needs to store a 16-bit memory address in compact form. The binary address is 1100000011110001. What is its hexadecimal equivalent?

A. C0E1

B. C0F1

C. B0F1

D. CF01

🔒 **12 more MCQs + answer key**
Get UniDrill Pro · ₹199/year · unidrill.in/pricing

PYQ Alignment

Number systems and encoding are a consistent source of questions in CUET Computer Science, typically contributing 5–7 MCQs per year, drawing on encoding scheme properties (ASCII/ISCII/UNICODE bit-widths and character counts), number system identification (base and digit set), decimal-to-binary/octal/hexadecimal conversion calculations, and hexadecimal applications in memory and colour codes. Conversion questions — especially those involving the repeated-division method or grouping of binary digits — appear almost every year. Use the consolidated PYQ archive for Computer Science for additional practice.