MEDHA AI

BASICS OF COMPUTERS

Computer basics: number systems, codes, logic, algorithms and spreadsheets

5 topics · 18 practice questions with answers · 2026



Free chapter · the full course at app.medhaai.in



medhaai.in

Notes · Practice · Mock tests · Previous papers

Key terms highlighted

Contents

BASICS OF COMPUTERS

1 Computer basics: number systems, codes, logic, algorithms and spreadsheets — 5 TOPICS

1.1	Algorithms and flowcharts: symbols, properties and examples	3
1.2	Number systems: binary, octal, hexadecimal and conversion to and from decimal	7
1.3	Spreadsheets: cells, ranges, formulas, cell references, functions, editing and printing	10
1.4	Codes (BCD, ASCII, EBCDIC, Unicode), bits, bytes, words and computer memory	15
1.5	Binary arithmetic and complements; logic gates and logical operations	19
	Quick revision	23
	Practice questions and answers	26

Chapter 1 · Basics of computers

Computer basics: number systems, codes, logic, algorithms and spreadsheets

5 TOPICS

1.1 Algorithms and flowcharts: symbols, properties and examples

An algorithm is a finite, step-by-step, unambiguous method for solving a problem; a flowchart draws the same steps with standard symbols joined by arrows.

CORE FACTS

- **Algorithm:** a finite sequence of well-defined steps that turns input into the required output. The word comes from the 9th-century Persian mathematician **al-Khwarizmi**.
- **Five properties** (Knuth):
  - **Finiteness:** it must stop after a finite number of steps.
  - **Definiteness:** every step is precise and unambiguous.
  - **Input: zero or more** inputs.
  - **Output: at least one** output.
  - **Effectiveness:** every step is basic enough to be done exactly, in finite time (with pen and paper).
- **Ways to write an algorithm:** plain-language steps, **pseudocode** (program-like English, no strict syntax) and **flowchart** (diagram).
- **Flowchart symbols:**

Symbol	Shape	Use
Terminal	Oval / rounded rectangle	Start and Stop
Input/Output	Parallelogram	Read, Print
Process	Rectangle	Calculation, assignment
Decision	Diamond (rhombus)	Yes/No or True/False test
Flow line	Arrow	Direction of flow

Symbol	Shape	Use
Connector	Small circle	Joins parts on the same page
Off-page connector	Pentagon (home-plate), textbook convention	Continues on another page
Predefined process	Rectangle with double side lines	Call to a sub-routine

- **Decision box:** one entry, two (or more) exits. All other boxes normally have one exit.
- **Three control structures** (Böhm–Jacopini, 1966): **sequence**, **selection** (if / if-else / case) and **iteration** (loops: while, do-while/repeat-until, for). Any algorithm can be built from these three.
- **Flowchart conventions:** flow from **top to bottom, left to right**; exactly one Start and at least one Stop; flow lines should not cross (use connectors).

Example 1. Which flowchart symbol is used for "Read A, B"? *Solution.* Input and output use the parallelogram. **Answer: parallelogram**

Example 2. An algorithm takes no input and prints "Hello" once. Which property does it still satisfy? *Solution.* Input may be zero; output must be at least one. **Answer: it is a valid algorithm (zero inputs, one output)**

1. Writing an algorithm in steps.

Example 3. Algorithm to find the larger of two numbers.

1. Start.
2. Read A, B.
3. If A > B, then print A; otherwise print B.
4. Stop.

Flowchart: oval *Start* → parallelogram *Read A, B* → diamond *A > B?* → Yes: parallelogram *Print A*; No: parallelogram *Print B* → both arrows meet → oval *Stop*.

2. Largest of three numbers (nested selection).

1. Start. Read A, B, C.
2. If A > B: if A > C, print A, else print C.

3. Otherwise: if B > C, print B, else print C.
4. Stop.

Any path through this flowchart uses exactly **two** decision boxes.

3. Loops: sum and factorial.

Example 4. Algorithm for N! (factorial).

1. Start. Read N.
2. F ← 1, I ← 1.
3. While I ≤ N: F ← F × I; I ← I + 1.
4. Print F. Stop.

Dry run for N = 5: F takes the values 1, 2, 6, 24, 120. **Answer: 120**

Flowchart: the test *I ≤ N?* is a diamond; the "Yes" branch goes to the process box *F ← F × I, I ← I + 1* and an arrow loops back to the diamond; the "No" branch goes to *Print F*.

4. Counting loop iterations.

Example 5. How many times does the body of "for I = 1 to 10 step 3" run? *Solution.* I = 1, 4, 7, 10; next value 13 exceeds 10. **Answer: 4 times**

5. Pre-test and post-test loops.

Loop	Test	Minimum runs
while / for	Before the body (entry-controlled)	0
do-while / repeat-until	After the body (exit-controlled)	1

6. Swapping two values.

Example 6. A = 5, B = 3. Steps: A ← A + B; B ← A − B; A ← A − B. Final values? *Solution.* A = 8; B = 8 − 3 = 5; A = 8 − 5 = 3. **Answer: A = 3, B = 5** (swapped without a third variable). The usual method uses a temporary variable: T ← A; A ← B; B ← T.

7. Dry run (trace table). Executing an algorithm by hand with sample values, writing each variable's value after each step, is called a **dry run** or **desk check**. It finds logical errors before coding.

Deeper

1. Euclid's algorithm for HCF (GCD).

1. Read A, B.
2. While B ≠ 0: R ← A mod B; A ← B; B ← R.
3. Print A.

Example 7. Trace for A = 48, B = 18.

Pass	A	B	R = A mod B
1	48	18	12
2	18	12	6
3	12	6	0

After pass 3, A = 6 and B = 0. **Answer: HCF 6, after 3 passes of the loop**

2. Sum of digits.

Example 8. S ← 0; while N > 0: S ← S + (N mod 10); N ← N div 10. Output for N = 4729? *Solution.* Digits taken 9, 2, 7, 4. S = 9 + 2 + 7 + 4. **Answer: 22**

3. Searching and sorting (the usual degree-level examples).

Method	Needs sorted data?	Worst-case comparisons
Linear search	No	n
Binary search	**Yes**	$\lfloor \log_2 n \rfloor + 1$ (about $\log_2 n$)
Bubble sort	—	$n(n-1)/2$, in $n-1$ passes

Example 9. Maximum comparisons for a binary search in a sorted list of 1,000 items? (a) 9 (b) 10 (c) 100 (d) 500 *Solution.* Each comparison halves the list; worst case = $\lfloor \log_2 1000 \rfloor + 1 = 9 + 1$ (since $2^9 = 512 \le 1{,}000 < 1{,}024 = 2^{10}$). **Answer: (b) 10**

Example 10. Bubble sort on 6 numbers: how many passes and comparisons (basic version)? *Solution.* Passes = 6 − 1 = 5. Comparisons = 6 × 5 ÷ 2. **Answer: 5 passes, 15 comparisons**

4. Merits and limits of flowcharts.

- Merits: clear picture of logic, easy communication, help in debugging and documentation.

- Limits: tedious for large programs, hard to modify (redraw), no standard way to show detail; pseudocode is easier to change.

5. Algorithm efficiency. Two correct algorithms may differ in **time** (number of steps) and **space** (memory) used. Big-O notation gives the growth rate: linear search O(n), binary search O(log n), bubble sort $O(n^2)$.

COMMON TRAPS

- ✗ "An algorithm must have at least one input." ✓ **Zero or more** inputs; **at least one** output.
- ✗ "Decisions use a rectangle." ✓ **Diamond**; the rectangle is for **processing**.
- ✗ "Input/output uses a rectangle with wavy base." ✓ **Parallelogram**; wavy base is a **document**.
- ✗ "Start/Stop is a circle." ✓ **Oval (terminal)**; a small circle is a **connector**.
- ✗ "Binary search works on any list." ✓ The list must be **sorted**.
- ✗ "A while loop always runs at least once." ✓ It may run **zero** times; **do-while/repeat-until** runs at least once.
- ✗ "Finiteness means the input is finite." ✓ It means the algorithm **terminates** after finitely many steps.

1.2 Number systems: binary, octal, hexadecimal and conversion to and from decimal

A number system is fixed by its base (radix): computers work in binary (base 2), and octal (base 8) and hexadecimal (base 16) are short ways of writing binary; any base converts to decimal by adding digit × place value, and decimal converts to any base by repeated division (whole part) and repeated multiplication (fraction).

CORE FACTS

- **Base (radix)** = number of distinct digits. Digits run from 0 to base − 1.

System	Base	Digits	Example
Binary	2	0, 1	101101_2
Octal	8	0 to 7	55_8

System	Base	Digits	Example
Decimal	10	0 to 9	45_{10}
Hexadecimal	16	0 to 9, A to F	$2D_{16}$

- **Hex letters:** A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
- **Positional value:** each place is a power of the base. In binary, from the right: 1, 2, 4, 8, 16, 32, 64, 128, ...; after the point: ½, ¼, ⅛, ...
- **Any base → decimal:** multiply each digit by its place value and add.
- **Decimal (whole part) → base b:** divide by b repeatedly; read the **remainders from bottom to top**.
- **Decimal (fraction) → base b:** multiply by b repeatedly; read the **integer parts from top to bottom**.
- **Binary ↔ octal:** groups of **3 bits** ($8 = 2^3$). **Binary ↔ hex:** groups of **4 bits** ($16 = 2^4$). Group from the point outwards; pad with zeros.
- **Octal ↔ hex:** go through binary.
- **MSB** = most significant (leftmost) bit; **LSB** = least significant (rightmost) bit.
- **n bits** give $\mathbf{2^n}$ patterns; the largest unsigned value is $\mathbf{2^n - 1}$ (8 bits: 0 to 255).
- Useful pairs: $2^{10} = 1{,}024$; $255 = 11111111_2 = 377_8 = FF_{16}$.

Example 1. Convert 45_{10} to binary. *Solution.* 45 ÷ 2 = 22 r 1; 22 ÷ 2 = 11 r 0; 11 ÷ 2 = 5 r 1; 5 ÷ 2 = 2 r 1; 2 ÷ 2 = 1 r 0; 1 ÷ 2 = 0 r 1. Read upwards. **Answer: 101101_2**

Example 2. Convert 11010_2 to decimal. *Solution.* 16 + 8 + 0 + 2 + 0. **Answer: 26**

1. Decimal to octal and hexadecimal.

Example 3. Convert 156_{10} to octal and to hexadecimal. *Solution.* Octal: 156 ÷ 8 = 19 r 4; 19 ÷ 8 = 2 r 3; 2 ÷ 8 = 0 r 2 → **234_8**. Hex: 156 ÷ 16 = 9 r 12 (C); 9 ÷ 16 = 0 r 9 → **$9C_{16}$**. Check: 2 × 64 + 3 × 8 + 4 = 156; 9 × 16 + 12 = 156. **Answer: 234_8 and $9C_{16}$**

2. Octal and hexadecimal to decimal.

Example 4. Convert 754_8 and $3F7_{16}$ to decimal. *Solution.* 7 × 64 + 5 × 8 + 4 = 448 + 40 + 4 = 492. 3 × 256 + 15 × 16 + 7 = 768 + 240 + 7 = 1,015. **Answer: 492 and 1,015**

3. Binary to octal and hex by grouping.

Example 5. Write 1101011_2 in octal, hexadecimal and decimal. *Solution.* Groups of 3 from the right: 1 | 101 | 011 → **153_8**. Groups of 4: 110 | 1011 → 6 | B → **$6B_{16}$**. Decimal: 64 + 32 + 8 + 2 + 1 = **107**.

Example 6. Convert $A5_{16}$ to octal. (a) 245 (b) 165 (c) 255 (d) 225 *Solution.* A5 = 1010 0101 → regroup in threes: 10 | 100 | 101 → 2 | 4 | 5. **Answer: (a) 245_8** ($A5_{16} = 245_8 = 165$ in decimal; (b) is the decimal value, a trap).

4. Fractions.

Example 7. Convert 0.625_{10} to binary. *Solution.* 0.625 × 2 = 1.25 → 1; 0.25 × 2 = 0.5 → 0; 0.5 × 2 = 1.0 → 1. Read downwards. **Answer: 0.101_2**

Example 8. Convert 25.375_{10} to binary. *Solution.* 25 = 11001; 0.375 × 2 = 0.75 → 0; 0.75 × 2 = 1.5 → 1; 0.5 × 2 = 1.0 → 1, so 0.011. **Answer: 11001.011_2**

Example 9. Convert 1101.11_2 and $1A.8_{16}$ to decimal. *Solution.* 8 + 4 + 1 + ½ + ¼ = 13.75. 1 × 16 + 10 + 8/16 = 26.5. **Answer: 13.75 and 26.5**

Summary of methods

From → to	Method
Base b → decimal	Σ digit × b^position
Decimal → base b	÷ b (whole), × b (fraction)
Binary → octal / hex	groups of 3 / 4 bits
Octal ↔ hex	via binary

Deeper

- **Not every decimal fraction ends in binary.** $0.1_{10} = 0.000110011\ldots_2$ (the block 0011 repeats), which is why 0.1 + 0.2 is not exactly 0.3 in most programs. A fraction ends in binary only if its denominator (in lowest terms) is a power of 2.
- **Bits needed for a decimal number N:** $\lfloor \log_2 N \rfloor + 1$. 100 needs 7 bits (1100100); 1,000 needs 10 bits.
- **Validity of digits:** a numeral is valid in base b only if every digit is less than b. "789" is not an octal number; "1234" needs base 5 or more.
- **Why octal and hex?** They compress long bit strings: one hex digit = 4 bits, two hex digits = one byte. Memory addresses, colour codes (#FF0000) and machine code listings use hex.

- **Shifting:** appending a 0 to a binary number doubles it (1011 → 10110, 11 → 22); dropping the last bit halves it (whole part only).

Example 10. If 121 in base b equals 49 in decimal, find b. (a) 5 (b) 6 (c) 7 (d) 8 *Solution.* $b^2 + 2b + 1 = 49$, so $(b + 1)^2 = 49$ and b = 6. **Answer: (b) 6**

Example 11. Convert $2F_{16}$ to octal. *Solution.* 2F = 0010 1111 = 101 111 → 57_8. Check: 2 × 16 + 15 = 47 = 5 × 8 + 7. **Answer: 57_8**

Example 12. Arrange in ascending order: 110011_2, 62_8, 33_{16}. *Solution.* $110011_2 = 51$; $62_8 = 50$; $33_{16} = 51$. **Answer: $62_8 < 110011_2 = 33_{16}$**

COMMON TRAPS

- ✗ "Read the remainders from top to bottom." ✓ For whole numbers read **bottom to top** (last remainder is the MSB).
- ✗ "Read fractional digits bottom to top." ✓ For fractions read the integer parts **top to bottom**.
- ✗ "Group bits from the left." ✓ Group from the **binary point outwards** (right to left for the whole part), padding with zeros.
- ✗ "Octal uses digits 0 to 8." ✓ Octal uses **0 to 7**; hex uses 0 to 9 and **A to F**.
- ✗ "Hex F = 16." ✓ **F = 15**; $10_{16} = 16$.
- ✗ "8 bits can hold up to 256." ✓ 8 bits give **256 patterns**, so the largest unsigned value is **255**.
- ✗ "Every decimal fraction converts exactly." ✓ 0.1_{10} **recurs** in binary.

1.3 Spreadsheets: cells, ranges, formulas, cell references, functions, editing and printing

A spreadsheet is a grid of rows and columns whose cells hold numbers, text or formulas; formulas recalculate automatically when any value they use changes.

CORE FACTS

- **Spreadsheet programs:** MS Excel, LibreOffice Calc, Google Sheets; the first PC spreadsheet was **VisiCalc (1979)**, followed by **Lotus 1-2-3**.

- **Workbook** = the file; **worksheet** (sheet) = one grid inside it. Excel file types: **.xlsx** (2007 onwards), **.xls** (older), **.xlsm** (with macros); Calc uses **.ods**; **.csv** is plain comma-separated text.
- **Rows** are numbered (1, 2, 3 …); **columns** are lettered (A … Z, AA … XFD).
- **Excel 2007 onwards: 1,048,576 rows × 16,384 columns** per sheet; last column **XFD**. (Excel 2003: 65,536 rows × 256 columns, last column IV.)
- **Cell:** the meeting point of a row and a column; its **address** is column letter + row number, e.g. **C5**.
- **Active cell:** the selected cell with a thick border; its address shows in the **Name Box**; its content shows in the **Formula Bar**.
- **Range:** a rectangular block written with a colon, e.g. **A1:C4** (3 columns × 4 rows = **12 cells**).
- **Formula:** starts with **=** (Lotus used +/@), e.g. =A1+B1. A **function** is a built-in formula, e.g. =SUM(A1:A10).
- **Cell references:**
  - **Relative** A1: changes when copied.
  - **Absolute** A1: never changes.
  - **Mixed** $A1 (column fixed) or A$1 (row fixed).
  - **F4** cycles A1 → A1 → A$1 → $A1 → A1.
- **Default alignment:** numbers and dates **right**, text **left**.
- **Fill handle:** small square at the bottom-right of the selection; drag to copy or **AutoFill** a series (Mon, Tue …; 1, 2, 3 …).
- **Chart, sort, filter, pivot table, conditional formatting** are standard features.

Example 1. How many cells are in the range B2:E6? *Solution.* Columns B to E = 4; rows 2 to 6 = 5. 4 × 5. **Answer: 20**

1. Entering and editing.

- **F2** edits the active cell; **Esc** cancels the entry; **Enter** confirms and moves down; **Tab** moves right.
- **Ctrl+Z** undo, **Ctrl+Y** redo, **Ctrl+C / X / V** copy, cut, paste; **Ctrl+F** find, **Ctrl+H** replace.
- **Paste Special** pastes only values, formats, formulas, or transposes rows and columns.
- **Ctrl+;** inserts today's date; **Ctrl+Home** goes to A1; **Alt+Enter** starts a new line inside a cell.

- **Insert / Delete** rows, columns, cells; **Clear** removes contents but keeps the cell; **Delete** removes the cell and shifts others.
- **Merge & Center** joins cells; **Wrap Text** fits long text in a cell; **Freeze Panes** keeps headings visible while scrolling.
- **####** in a cell: the column is **too narrow** to show the number (widen it).

2. Arithmetic operators and precedence (Excel).

Order	Operator	Meaning
1	: , (space)	Reference operators
2	–	Negation (–5)
3	%	Percent
4	^	Power
5	* /	Multiply, divide
6	+ –	Add, subtract
7	&	Join text
8	= < > <= >= <>	Compare

Brackets override the order. Same level: left to right.

Example 2. Evaluate =2+3*4^2/8.* Solution.* 4^2 = 16; 3 × 16 = 48; 48 ÷ 8 = 6; 2 + 6. **Answer: 8**

Example 3. Evaluate =(5+3)*2^2.* Solution.* 8 × 4. **Answer: 32**

3. Copying formulas: relative vs absolute.

Example 4. C1 contains =A1+B1. It is copied to C2 and to D1. What do they contain? *Solution.* Down one row: =A2+B2. Right one column: =B1+C1. **Answer: C2 =A2+B2; D1 =B1+C1**

Example 5. C2 contains =$A2*B$1. It is copied to D3. Result? (a) =$A3*C$1 (b) =$A2*C$1 (c) =$B3*C$2 (d) =$A3*B$1* Solution.* Move: one column right, one row down. $A2: column fixed, row 2 → 3. B$1: column B → C, row fixed. **Answer: (a)**

Use: =B2*E1 multiplies every price in column B by one rate held in E1; copying keeps pointing to E1.

4. Common functions. Data: A1 = 10, A2 = 20, A3 blank, A4 = "abc", A5 = 30, A6 = 0.

Function	Does	Result for A1:A6
SUM(A1:A6)	Adds numbers	60
COUNT(A1:A6)	Counts **numbers**	4
COUNTA(A1:A6)	Counts **non-empty** cells	5
COUNTBLANK(A1:A6)	Counts empty cells	1
AVERAGE(A1:A6)	SUM ÷ COUNT	15
MAX / MIN	Largest / smallest	30 / 0

AVERAGE **ignores blanks and text** but **includes zeros**: 60 ÷ 4 = 15, not 60 ÷ 6.

Example 6. Marks in B2:B6: 35, 62, 48, 90, 27 (pass mark 40).

Formula	Result
=AVERAGE(B2:B6)	52.4
=MEDIAN(B2:B6)	48
=COUNTIF(B2:B6,">=40")	3
=SUMIF(B2:B6,">=40")	200
=IF(B2>=40,"Pass","Fail")	Fail
=RANK(B4,B2:B6)	3

5. Printing.

- **Print Preview** shows pages before printing; **Page Setup** sets **orientation** (portrait/landscape), paper size, **margins**, **scaling** ("Fit to 1 page wide").
- **Print Area** prints only a selected range; **Print Titles** ("Rows to repeat at top") repeats heading rows on every page.
- **Header and footer** carry page numbers, date, file name.
- **Gridlines and row/column headings are not printed by default**; turn them on in Page Setup → Sheet.
- **Page Break Preview** shows and moves page breaks. Shortcut: **Ctrl+P**.

Deeper

1. More functions used in statistics work.

Function	Use	Example → result
ROUND(x, n)	Round to n decimals	ROUND(3.14159, 2) → 3.14
ROUND(x, −2)	Round to hundreds	ROUND(1234.5, −2) → 1200
INT(x)	Round **down** to integer	INT(−3.7) → −4
MOD(a, b)	Remainder, sign of b	MOD(17, 5) → 2; MOD(−7, 3) → 2

Others: **MEDIAN, MODE, STDEV** (sample) / **STDEV.P** (population), **VAR, CORREL, PERCENTILE, QUARTILE, GEOMEAN, HARMEAN, FREQUENCY** (an array function for frequency tables), **SUMPRODUCT** (Σfx in one step), **VLOOKUP / HLOOKUP** (look up a value in the first column / row of a table), **AND, OR, NOT**, **CONCATENATE** or &, **TODAY(), NOW()**.

Example 7. Frequencies in B2:B6 against values in A2:A6. Write the mean in one formula.
Solution. Mean = Σfx ÷ Σf. **Answer: =SUMPRODUCT(A2:A6,B2:B6)/SUM(B2:B6)**

2. Error values.

Error	Cause
#DIV/0!	Division by zero (or by an empty cell)
#VALUE!	Wrong type, e.g. text in arithmetic
#REF!	Reference to a deleted cell
#NAME?	Misspelt function or name
#N/A	Value not available (lookup failed)
#NUM!	Invalid number (e.g. square root of −1)
#NULL!	Two ranges that do not intersect

A formula that refers to its own cell (directly or indirectly) is a **circular reference**.

3. Tricky precedence. In Excel, **negation comes before power**: =-2^2 gives **4**, while =0-2^2 gives **−4** (here − is subtraction). Excel rounds a half **away from zero**: ROUND(2.5, 0) = 3, ROUND(−2.5, 0) = −3.

4. Other cell-reference forms.

- Whole column **A:A**, whole row **5:5**.
- Another sheet: **Sheet2!B4**; another workbook: **[Book2.xlsx]Sheet1!A1**.
- **Union** (comma): SUM(A1:A3, C1:C3); **intersection** (space): SUM(A1:C3 B2:D4) adds only the shared cells B2:C3.
- **Named range:** call B2:B6 "Marks" and write =AVERAGE(Marks).

5. Data tools. Sort (ascending/descending, multi-level), **Filter / AutoFilter** (shows only matching rows; others are hidden, not deleted), **Data Validation** (restrict entries, e.g. 0–100), **Conditional Formatting** (colour cells by rule), **Pivot Table** (cross-tabulation, the spreadsheet's tabulation tool), **Goal Seek** (find the input that gives a target result), **charts** (column, bar, line, pie, scatter/XY, histogram).

COMMON TRAPS

- ✗ "Formulas start with +." ✓ In Excel and Calc they start with **=**.
- ✗ "COUNT counts all non-empty cells." ✓ COUNT counts **numbers**; **COUNTA** counts non-empty cells.
- ✗ "AVERAGE treats blank cells as zero." ✓ Blanks (and text) are **ignored**; a typed 0 **is** counted.
- ✗ "$A1 fixes the row." ✓ **$A1 fixes the column**; **A$1** fixes the row.
- ✗ "#### means a formula error." ✓ The **column is too narrow**.
- ✗ "Gridlines print by default." ✓ They **do not**; switch them on in Page Setup.
- ✗ "A workbook and a worksheet are the same." ✓ A **workbook** (file) contains **worksheets**.
- ✗ "Excel 2007+ has 65,536 rows." ✓ **1,048,576 rows**, 16,384 columns (to XFD); 65,536 was Excel 2003.

1.4 Codes (BCD, ASCII, EBCDIC, Unicode), bits, bytes, words and computer memory

Computers store everything as bits; codes (BCD, ASCII, EBCDIC, Unicode) decide which bit pattern stands for which digit or character, and memory holds these patterns in a hierarchy from fast, small registers and cache to large, slow secondary storage.

CORE FACTS

- **Bit** = binary digit (0 or 1), the smallest unit. **Nibble** = 4 bits. **Byte** = 8 bits (one character in ASCII). **Word** = the number of bits the CPU handles as one unit (16, 32 or 64 bits; machine-dependent).
- **Units (binary convention used in textbooks):**

Unit	Equals	Power of 2
1 KB	1,024 bytes	2^{10}
1 MB	1,024 KB	2^{20}
1 GB	1,024 MB	2^{30}
1 TB	1,024 GB	2^{40}

- Beyond TB: PB (2^{50}), EB (2^{60}), ZB (2^{70}), YB (2^{80}).
- **BCD (8421 code):** each decimal digit in **4 bits**; 1010 to 1111 are unused (invalid).
- **Excess-3:** BCD + 3 (0011) for each digit; **self-complementing**.
- **Gray code:** consecutive values differ in **only one bit** (unit-distance, reflected binary code).
- **ASCII:** American Standard Code for Information Interchange, **7-bit**, **128** characters; extended ASCII uses 8 bits (256).
- **EBCDIC:** Extended Binary Coded Decimal Interchange Code, **8-bit**, developed by **IBM** for mainframes.
- **ISCII:** Indian Script Code for Information Interchange, 8-bit, BIS standard of **1991**.
- **Unicode:** one code for every script in the world; code points **U+0000 to U+10FFFF** (1,114,112). Encodings: **UTF-8** (1 to 4 bytes, ASCII-compatible), **UTF-16** (2 or 4 bytes), **UTF-32** (4 bytes).
- **ASCII values:** 'A' = **65**, 'a' = **97**, '0' = **48**, space = 32.

- **Primary memory:** RAM (volatile, read/write) and ROM (non-volatile, read-only). **Secondary:** hard disk, SSD, pen drive, CD/DVD, tape (non-volatile).
- **Fastest to slowest:** registers → cache → RAM → secondary storage.

1. Codes for numbers.

Example 1. Write 59 in BCD and in pure binary. *Solution.* BCD codes each digit: 5 = 0101, 9 = 1001 → **0101 1001** (8 bits). Pure binary: 59 = **111011** (6 bits). BCD needs more bits but converts easily to and from decimal display.

Example 2. Find the excess-3 code of decimal 5 and the Gray code of binary 1011. *Solution.* 5 + 3 = 8 → **1000**. Gray: keep the first bit (1), then XOR each pair of neighbours: 1⊕0 = 1, 0⊕1 = 1, 1⊕1 = 0 → **1110**.

2. Codes for characters. Upper case 'A' to 'Z' are 65 to 90; lower case 'a' to 'z' are 97 to 122; a lower-case letter is its capital **+ 32**. Digits '0' to '9' are 48 to 57, so the character '7' is not the number 7.

Example 3. The ASCII code of 'D' in decimal and binary is (a) 68, 1000100 (b) 100, 1100100 (c) 67, 1000011 (d) 68, 1000010 *Solution.* 'A' = 65, so 'D' = 68 = 64 + 4. **Answer: (a)**

Code comparison

Code	Bits	Characters	Use
ASCII	7	128	PCs, internet text
EBCDIC	8	256	IBM mainframes
ISCII	8	Indian scripts	older Indian software
Unicode	8/16/32 units	all scripts	modern systems, web

Telugu has its own Unicode block, **U+0C00 to U+0C7F**, which is why Telugu text typed on a phone displays correctly on any computer.

3. Parity bit. An extra bit added so that the number of 1s is even (**even parity**) or odd (**odd parity**); it detects a single-bit error but cannot correct it.

Example 4. Find the even-parity bit for 1011001. *Solution.* It has four 1s, already even. **Answer: 0** (odd parity would need 1)

4. Kinds of memory.

Type	Volatile?	Feature
SRAM	Yes	flip-flops, fast, no refresh; used in cache
DRAM	Yes	capacitors, must be refreshed; main memory
ROM / PROM	No	PROM written once by the user
EPROM / EEPROM	No	erased by UV light / electrically

- **Flash memory** is a kind of EEPROM erased in blocks; it is used in pen drives, memory cards and SSDs.
- **Firmware** (BIOS/UEFI start-up program) is kept in ROM or flash.
- **Cache** sits between CPU and RAM; levels L1 (fastest, smallest), L2, L3.
- **Registers** are inside the CPU (accumulator, program counter, instruction register).
- **Secondary storage:** hard disk (magnetic; tracks, sectors, cylinders), SSD (flash, no moving parts), optical discs (CD about 700 MB, DVD 4.7 GB single layer, Blu-ray 25 GB single layer), magnetic tape (**sequential access**, backups).

Deeper

- **Memory hierarchy:** going down (register → cache → RAM → disk → tape), **speed falls, capacity rises and cost per bit falls**. The hierarchy works because programs reuse nearby data (locality of reference); the fraction of requests found in cache is the **hit ratio**.
- **Access:** RAM is **random access** (equal time for any location); tape is **sequential**; disk is **direct (semi-random) access**. Disk access time = **seek time + rotational latency** (+ transfer time).
- **Virtual memory:** part of the hard disk used as an extension of RAM, so programs larger than RAM can run; pages are swapped in and out.
- **Address lines and capacity:** n address lines select $\mathbf{2^n}$ locations; capacity = 2^n × word size.
- **Units in practice:** disk makers use decimal units (1 GB = 10^9 bytes); the IEC names KiB, MiB, GiB for 2^{10}, 2^{20}, 2^{30}. Exams follow the textbook convention 1 KB = 1,024 bytes unless told otherwise.

Example 5. A memory chip has 16 address lines and an 8-bit word. What is its capacity? (a) 16 KB (b) 32 KB (c) 64 KB (d) 128 KB *Solution.* 2^{16} = 65,536 locations × 1 byte = 65,536 bytes = 64 KB. **Answer: (c) 64 KB**

Example 6. How much memory can a CPU with a 32-bit byte address address? *Solution.* 2^{32} bytes = $2^2 \times 2^{30}$ bytes. **Answer: 4 GB**

Example 7. How many bits are there in 2 KB? *Solution.* 2 × 1,024 × 8. **Answer: 16,384 bits**

Example 8. Which is non-volatile? (a) SRAM (b) DRAM (c) Cache (d) EEPROM *Solution.* RAM and cache lose data when power is off. **Answer: (d) EEPROM**

COMMON TRAPS

- ✗ "ASCII is an 8-bit code of 256 characters." ✓ Standard ASCII is **7-bit, 128**; 8-bit is extended ASCII.
- ✗ "EBCDIC was developed by ANSI." ✓ **IBM**; ASCII came from the American standards body.
- ✗ "BCD of 59 = 111011." ✓ That is pure binary; BCD is **0101 1001**.
- ✗ "1 KB = 1,000 bytes" (in a computer-basics paper). ✓ **1,024 bytes**.
- ✗ "A word is always 16 bits." ✓ Word length **depends on the processor**.
- ✗ "DRAM is used for cache." ✓ Cache uses **SRAM**; DRAM needs refreshing and is main memory.
- ✗ "EPROM is erased electrically." ✓ EPROM is erased by **ultraviolet light**; EEPROM electrically.
- ✗ "'a' = 65." ✓ **'A' = 65, 'a' = 97**.

1.5 Binary arithmetic and complements; logic gates and logical operations

The ALU does arithmetic on binary numbers (adding, and subtracting by adding the 2's complement) and logical operations (AND, OR, NOT, XOR) bit by bit; these are built from logic gates, and NAND or NOR alone can build every other gate.

CORE FACTS

- **Binary addition:** 0 + 0 = 0; 0 + 1 = 1; 1 + 1 = **10** (0, carry 1); 1 + 1 + 1 = **11** (1, carry 1).
- **Binary subtraction:** 0 − 0 = 0; 1 − 0 = 1; 1 − 1 = 0; 0 − 1 = **1, borrow 1**.
- **Multiplication:** 0 × 0 = 0 × 1 = 0, 1 × 1 = 1 (shift and add). **Division:** repeated shift and subtract.

- **1's complement:** flip every bit (0 ↔ 1). **2's complement** = 1's complement **+ 1**.
- **Decimal analogues:** 9's complement (each digit from 9) and 10's complement (9's + 1).
- **Computers subtract by adding** the 2's complement of the number to be subtracted.
- **n-bit 2's complement range: -2^{n-1} to $2^{n-1} - 1$**; 8 bits: **−128 to +127**. The MSB is the sign bit (0 = +, 1 = −).
- **Basic gates:** AND, OR, NOT. **Universal gates: NAND, NOR**. Others: XOR, XNOR.

Gate	Expression	Output is 1 when
AND	A·B	all inputs are 1
OR	A + B	at least one input is 1
NOT	A′	input is 0
NAND	(A·B)′	not all inputs are 1
NOR	(A + B)′	all inputs are 0
XOR	A ⊕ B	inputs differ
XNOR	(A ⊕ B)′	inputs are equal

- **Truth table** for n inputs has **2^n** rows.
- **De Morgan:** (A + B)′ = A′·B′ and (A·B)′ = A′ + B′.

Example 1. Add 10110_2 and 01101_2. *Solution.* 22 + 13 = 35. Column by column with carries gives 100011. **Answer: 100011_2**

1. The four operations.

Example 2. Find $10110_2 - 01101_2$. *Solution.* 22 − 13 = 9. **Answer: 1001_2**

Example 3. Multiply 1011_2 by 101_2. *Solution.* 1011 × 1 = 1011; next bit 0 → 00000; next bit 1 → 101100. Sum 1011 + 101100 = 110111. Check: 11 × 5 = 55. **Answer: 110111_2**

Example 4. Divide 110110_2 by 110_2. *Solution.* 54 ÷ 6 = 9. **Answer: 1001_2**

2. Complements.

Example 5. Find the 1's and 2's complements of 01011010. *Solution.* Flip: 10100101. Add 1: 10100110. **Answer: 10100101 and 10100110** Shortcut for the 2's complement: copy bits from the right up to and including the first 1, then flip the rest.

Example 6. Write −25 in 8-bit 2's complement. (a) 10011001 (b) 11100110 (c) 11100111 (d) 00011001 *Solution.* +25 = 00011001; flip → 11100110; + 1 → 11100111. **Answer: (c)** ((a) is sign-magnitude, (b) is 1's complement)

3. Subtraction by complements (4 bits).

Example 7. 9 − 5 using 2's complement. *Solution.* 5 = 0101 → 2's complement 1011. 1001 + 1011 = 1 0100. A carry out means the result is positive: **discard it**. **Answer: 0100 = +4**

Example 8. 5 − 9 using 2's complement. *Solution.* 9 = 1001 → 2's complement 0111. 0101 + 0111 = 1100, **no carry**, so the answer is negative: take the 2's complement of 1100 = 0100. **Answer: −4**

Example 9. 9 − 5 using 1's complement. *Solution.* 1's complement of 0101 = 1010. 1001 + 1010 = 1 0011. Add the carry back (**end-around carry**): 0011 + 1 = 0100. **Answer: +4**

Example 10. 8754 − 4567 using 10's complement. *Solution.* 10's complement of 4567 = 5433. 8754 + 5433 = 14187; drop the carry 1. **Answer: 4187**

4. Logical (bitwise) operations. The ALU applies a gate to each pair of bits.

With A = 1100 and B = 1010:

Operation	Result
A AND B	1000
A OR B	1110
A XOR B	0110
NOT A	0011

Uses: AND **clears** chosen bits (masking), OR **sets** bits, XOR **toggles** bits; A XOR A = 0. Spreadsheets use the same logic in the functions AND(), OR() and NOT(), which return TRUE or FALSE.

5. Universal gates. Each gate can be built from NAND gates only (or NOR gates only):

Gate wanted	NAND gates	NOR gates
NOT	1	1
AND	2	3
OR	3	2

Gate wanted	NAND gates	NOR gates
XOR	4	5

Deeper

- **Signed representations of −5 in 8 bits:** sign-magnitude **10000101**, 1's complement **11111010**, 2's complement **11111011**. Sign-magnitude and 1's complement have **two zeros** (+0 and −0) and range −127 to +127; 2's complement has **one zero** and range −128 to +127, which is why computers use it.
- **Overflow:** happens when adding two numbers of the **same sign** gives a result of the opposite sign. In 8 bits, 100 + 50 = 150 > 127 gives 10010110, which reads as **−106**. Adding numbers of opposite signs never overflows.
- **Boolean laws:**
  - Identity A + 0 = A, A·1 = A; null A + 1 = 1, A·0 = 0.
  - Idempotent A + A = A, A·A = A; complement A + A′ = 1, A·A′ = 0; involution (A′)′ = A.
  - Absorption A + AB = A, A(A + B) = A; also A + A′B = A + B.
  - Distributive A + BC = (A + B)(A + C) holds in Boolean algebra (not in ordinary algebra).
  - **Duality:** swap + with · and 0 with 1 to get another valid law.
- **Adders:** a **half adder** adds 2 bits: sum = A ⊕ B, carry = A·B. A **full adder** adds 3 bits (with carry in C): sum = A ⊕ B ⊕ C, carry = AB + BC + CA. A half subtractor gives difference A ⊕ B and borrow A′B.
- **Multi-input XOR** gives 1 when the number of 1 inputs is **odd** (this is how a parity bit is generated).
- **Shifts:** shifting left by one place multiplies by 2; shifting right divides by 2 (unsigned whole numbers).
- **Number of Boolean functions** of n variables = 2^(2ⁿ); for 2 variables, 16.

Example 11. Simplify AB + AB′. *Solution.* A(B + B′) = A·1. **Answer: A**

Example 12. Which gate gives output 1 only when both inputs are 0? (a) NAND (b) NOR (c) XOR (d) XNOR *Solution.* NOR is 1 only for 0, 0. XNOR is also 1 for 1, 1. **Answer: (b) NOR**

Example 13. Add $3A_{16} + 2F_{16}$ and $57_8 + 26_8$. *Solution.* Hex: A + F = 10 + 15 = 25 = 16 + 9 → write 9, carry 1; 3 + 2 + 1 = 6 → 69_{16} (58 + 47 = 105). Octal: 7 + 6 = 13 = 8 + 5 → write 5, carry 1; 5 + 2 + 1 = 8 = 10_8 → 105_8 (47 + 22 = 69). **Answer: 69_{16} and 105_8**

COMMON TRAPS

- ✗ "1 + 1 = 2 in binary." ✓ 1 + 1 = **10** (write 0, carry 1).
- ✗ "2's complement = flip the bits." ✓ That is the **1's** complement; 2's complement adds **1**.
- ✗ "In 2's complement subtraction, add the end carry back." ✓ **Discard** it; end-around carry is for **1's** complement.
- ✗ "No carry means the answer is positive." ✓ No carry means the result is **negative**, in complement form.
- ✗ "8-bit signed range is −127 to +128." ✓ **−128 to +127**.
- ✗ "AND, OR and NOT are the universal gates." ✓ The universal gates are **NAND and NOR**.
- ✗ "XOR is 1 when both inputs are 1." ✓ XOR is 1 when inputs **differ**; for 1, 1 it is 0.
- ✗ "(A + B)′ = A′ + B′." ✓ De Morgan: **(A + B)′ = A′·B′**.

Quick revision

READ THIS THE DAY BEFORE THE EXAM

1.1 Algorithms and flowcharts: symbols, properties and examples

- Algorithm: finite, definite, effective; 0+ inputs, 1+ outputs.
- Oval = start/stop; parallelogram = input/output; rectangle = process.
- Diamond = decision (two exits); circle = on-page connector; pentagon = off-page connector.
- Flow top to bottom, left to right; arrows show direction.
- Three structures: sequence, selection, iteration.
- while: test first (0+ runs); do-while: test last (1+ runs).
- Dry run = tracing by hand with a trace table.
- Binary search needs sorted data; about $\log_2 n$ comparisons.
- Bubble sort: $n - 1$ passes, $n(n - 1)/2$ comparisons.
- Euclid: HCF(48, 18) = 6.

1.2 Number systems: binary, octal, hexadecimal and conversion to and from decimal

- Binary 2, octal 8, decimal 10, hex 16; digits 0 to base − 1.
- A to F = 10 to 15.
- To decimal: Σ digit × place value.
- Whole part: ÷ base, remainders bottom up.
- Fraction: × base, integer parts top down.
- 3 bits = 1 octal digit; 4 bits = 1 hex digit.
- n bits → 2^n patterns, max $2^n - 1$.
- $45 = 101101_2 = 55_8 = 2D_{16}$.
- $255 = 11111111_2 = 377_8 = FF_{16}$.
- $0.625 = 0.101_2$; 0.1 recurs in binary.

1.3 Spreadsheets: cells, ranges, formulas, cell references, functions, editing and printing

- Workbook = file; worksheet = grid; cell = column letter + row number.
- Range A1:C4 = 12 cells; Name Box shows the active cell.
- Formulas begin with =; functions like SUM, AVERAGE, COUNT, IF.
- Relative A1; absolute A1; mixed $A1 / A$1; F4 toggles.
- Precedence: negation, %, ^, * /, + –, &, comparisons.
- COUNT = numbers; COUNTA = non-empty; AVERAGE ignores blanks, counts zeros.
- #DIV/0!, #VALUE!, #REF!, #NAME?, #N/A; #### = narrow column.
- Excel 2007+: 1,048,576 rows × 16,384 columns (XFD).
- Printing: Print Area, Print Titles, Page Setup, Print Preview, Ctrl+P.
- Gridlines not printed by default.

1.4 Codes (BCD, ASCII, EBCDIC, Unicode), bits, bytes, words and computer memory

- Nibble 4 bits, byte 8 bits, word = CPU unit (machine-dependent).
- KB 2^{10}, MB 2^{20}, GB 2^{30}, TB 2^{40} bytes.
- BCD: 4 bits per decimal digit; excess-3 = BCD + 3.
- Gray code: one bit changes at a time.
- ASCII 7-bit (128); EBCDIC 8-bit (IBM); ISCII 1991; Unicode all scripts.
- 'A' 65, 'a' 97, '0' 48.
- Parity bit detects a single-bit error.
- SRAM cache, DRAM main memory; EPROM UV, EEPROM electric.
- Registers → cache → RAM → disk: slower, bigger, cheaper.
- n address lines → 2^n locations.

1.5 Binary arithmetic and complements; logic gates and logical operations

- 1 + 1 = 10; 1 + 1 + 1 = 11; 0 − 1 = 1 borrow 1.
- 1's complement: flip; 2's complement: flip + 1.
- 2's complement subtraction: carry → discard, positive; no carry → negative, re-complement.
- 1's complement: end-around carry.
- 8-bit signed: −128 to +127; −25 = 11100111.
- Overflow: same-sign inputs, opposite-sign result.
- Universal gates NAND, NOR; XOR = 4 NANDs.
- De Morgan: $(A + B)' = A'B'$; $(AB)' = A' + B'$.
- Half adder: sum $A \oplus B$, carry AB.
- Truth table: 2^n rows.

Practice questions

18 QUESTIONS · ANSWERS AT THE END

1. Consider the following statements:

1. Each octal digit can be written as a group of exactly three binary digits.

2. Each hexadecimal digit can be written as a group of exactly four binary digits.

Which of the statements given above is/are correct?

a. 1 only
b. 2 only
c. Both 1 and 2
d. Neither 1 nor 2

2. Consider the following statements about Unicode:

1. The first 128 Unicode code points are the same as the ASCII characters.

2. UTF-8 always uses exactly 2 bytes for every character.

Which of the statements given above is/are correct?

a. 1 only
b. 2 only
c. Both 1 and 2
d. Neither 1 nor 2

3. Consider the following statements about the XNOR gate:

1. Its output is 1 when both inputs are equal.

2. It is the complement of the XOR gate.

3. For inputs A and B its output is A·B + A'·B'.

Which of the statements given above are correct?

a. 1 and 2 only
b. 2 and 3 only
c. 1 and 3 only
d. 1, 2 and 3

4. Match List I (chart type) with List II (best use):

List I:

A. Pie chart

B. Line chart

C. Column chart

D. Scatter (XY) chart

List II:

1. Trend of a value over time

2. Relationship between two numeric variables

3. Share of each part in a whole

4. Comparison of values across categories

Select the correct answer using the code given below.

a. A-3, B-4, C-1, D-2

b. A-3, B-1, C-4, D-2

c. A-4, B-1, C-3, D-2

d. A-3, B-1, C-2, D-4

5. What is the octal equivalent of the hexadecimal number $(3A7)_{16}$?

a. 1637

b. 935

c. 1747

d. 1647

6. Consider the following statements:

1. The 4-bit groups 1010 to 1111 are invalid in BCD (8421) code.

2. BCD usually needs more bits than pure binary to represent the same number.

3. Gray code is a weighted code.

Which of the statements given above are correct?

a. 1 and 3 only

b. 1 and 2 only

c. 2 and 3 only

d. 1, 2 and 3

7. Which of the following pairs of flowchart symbol and its use is NOT correctly matched?

a. Rectangle with double vertical side lines : predefined process (a sub-routine defined elsewhere)

b. Arrow : direction of flow

c. Small circle : start or stop of the flowchart

d. Rectangle with a wavy bottom edge : printed document or report

8. Columns in a worksheet are labelled A, B, ..., Z, AA, AB, ... What is the column number of column AB?

a. 27 b. 28

c. 29 d. 54

9. Consider the following statements:

1. In a four-digit hexadecimal integer, the place value of the leftmost position is 16^3.
2. The hexadecimal letters A to F stand for the decimal values 10 to 15.
3. 8 is a valid digit in the octal system.
4. The binary system uses only two symbols, 0 and 1.

Which of the statements given above are correct?

a. 1 and 2 only b. 2, 3 and 4 only

c. 1, 2 and 4 only d. 1, 2, 3 and 4

10. The Gray code 1101 is converted to binary. What is the result?

a. 1011 b. 1001

c. 1101 d. 0110

11. Consider the following statements:

1. In a rotate operation no bit is lost; the bit moved out at one end comes back at the other end.
2. Shifting an unsigned number left by one place doubles it, provided no 1 bit is moved out of the leftmost position.

Which of the statements given above is/are correct?

a. 1 only b. 2 only

c. Both 1 and 2 d. Neither 1 nor 2

12. In MS-Excel, cell C2 contains the formula =$A2*B$1. This cell is copied and pasted into cell E5. Which formula will E5 contain?

a. =$A2*D$4 b. =$C5*D$1

c. =$A5*B$1 d. =$A5*D$1

13. Match List I (number) with List II (decimal value):

List I:

A. $(1110)_2$

B. $(21)_8$

C. $(1C)_{16}$

D. $(101011)_2$

List II:

1. 17

2. 43

3. 14

4. 28

Select the correct answer using the code given below.

a. A-3, B-4, C-1, D-2

b. A-1, B-3, C-4, D-2

c. A-3, B-1, C-2, D-4

d. A-3, B-1, C-4, D-2

14. Consider the following statements:

1. BCD (8421) is a weighted code.

2. Excess-3 is a weighted code.

Which of the statements given above is/are correct?

a. 1 only

b. 2 only

c. Both 1 and 2

d. Neither 1 nor 2

15. Using 8-bit 2's complement arithmetic, 00111100 (60) is subtracted from 00101101 (45) by adding the 2's complement of 60. What is the 8-bit result?

a. 00001111

b. 11110000

c. 10001111

d. 11110001

16. Match List I (number system) with List II (base):

List I:

A. Binary

B. Octal

C. Decimal

D. Hexadecimal

List II:

1. 16

2. 2

3. 10

4. 8

Select the correct answer using the code given below.

a. A-2, B-4, C-3, D-1

b. A-2, B-1, C-3, D-4

c. A-4, B-2, C-3, D-1

d. A-2, B-4, C-1, D-3

17. Using the binary convention (1 KB = 1024 bytes), which of the following pairs is NOT correctly matched?

a. 1 nibble : 4 bits

b. 1 byte : 8 bits

c. 1 MB : 1024 bytes

d. 1 GB : 1024 MB

18. A memory chip is organised as 32K × 8 bits (32K locations of one byte each). How many address lines does one chip need, and how many such chips are needed to build a 256 KB byte-addressable memory?

a. 8 address lines; 8 chips

b. 15 address lines; 8 chips

c. 15 address lines; 64 chips

d. 32 address lines; 8 chips

Answers: `1-(c) 2-(a) 3-(d) 4-(b) 5-(d) 6-(b) 7-(c) 8-(b) 9-(c) 10-(b) 11-(c) 12-(d) 13-(d) 14-(a) 15-(d) 16-(a) 17-(c) 18-(b)`