Seminar 1 Notes

Introduction to Computing Fundamentals

Section 2: Introduction to Computing

Evolution of Computing: From Mechanical to AI

Key Eras

- **Mechanical Era (1800s):** Babbage's Analytical Engine introduced the concept of programmable computation. Ada Lovelace wrote the first algorithm.
- **Electronic Era (1940s-1970s):** ENIAC, transistors replaced vacuum tubes, and integrated circuits emerged. Moore's Law observed.
- **Personal Computing (1980s-1990s):** IBM PC, Apple Macintosh, Windows. Computing moved from institutions to homes.
- **Internet & Mobile Era (2000s-2010s):** Cloud computing, smartphones, social media. Computing became ubiquitous.
- **AI Era (2020s+):** Machine learning, neural networks, generative AI. Computing now learns and adapts autonomously.

Classical vs Quantum Computing

Aspect	Classical Computing	Quantum Computing
Basic Unit	Bits (0 or 1)	Qubits (0, 1, or superposition)

Processing	Sequential or parallel operations	Simultaneous via superposition
Best For	General tasks, web, and databases	Cryptography, optimisation, simulation
Current State	Mature, widely deployed	Experimental, limited qubits

Computing Impact Across Industries

- **Healthcare:** AI diagnostics, electronic health records, drug discovery simulations, robotic surgery, predictive patient monitoring.
- **Finance:** Algorithmic trading, fraud detection, risk modelling, blockchain, robo-advisors.
- **Cybersecurity:** Threat detection via ML, encryption, identity management, automated incident response, and penetration testing tools.

Case Study: AI-Driven Stock Trading

High-frequency trading (HFT) algorithms now execute thousands of trades per second, analysing market data in real-time. Key developments:

- Machine learning models predict price movements from historical patterns and sentiment analysis
- Natural language processing scans news and social media for market-moving information
- Automated decision-making reduces human emotional bias
- Raises questions about market fairness, flash crashes, and regulatory oversight

Section 3: Logic, Gates and Boolean Algebra

Basic Logic Gates

Logic gates are the fundamental building blocks of digital circuits. They perform basic logical operations on binary inputs (0 and 1).

Gate	Symbol	Function	Output = 1 When...
AND	$A \cdot B$ or $A \wedge B$	Both inputs are 1	A = 1 AND B = 1
OR	$A + B$ or $A \vee B$	At least one input is 1	A = 1 OR B = 1 (or both)
NOT	$\neg A$ or A'	Inverts the input	A = 0
XOR	$A \oplus B$	Exclusive OR	Inputs are different
NAND	$\neg(A \cdot B)$	NOT AND	NOT (A = 1 AND B = 1)
NOR	$\neg(A + B)$	NOT OR	Both inputs are 0

Truth Tables

Truth tables show all possible input combinations and their resulting outputs. Essential for verifying circuit logic.

Example: AND Gate Truth Table

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	**1**

Boolean Algebra Laws

- **Identity:** A + 0 = A | A · 1 = A
- **Null:** A + 1 = 1 | A · 0 = 0
- **Complement:** A + A' = 1 | A · A' = 0
- **De Morgan's:** (A · B)' = A' + B' | (A + B)' = A' · B'
- **Distributive:** A · (B + C) = (A · B) + (A · C)

How Logic Gates Form Microprocessors

Logic gates combine hierarchically to build complex circuits:

- **Gates → Adders:** Half-adders (XOR + AND) combine to form full adders for arithmetic
- **Gates → Flip-Flops:** Memory circuits that store single bits (basis of RAM and registers)
- **Adders + Registers → ALU:** Arithmetic Logic Unit performs all mathematical and logical operations
- **ALU + Control Unit + Memory → CPU:** The complete processor executing instructions

Section 4: Set Theory in Computing

Core Definitions

- **Set:** An unordered collection of distinct elements. No duplicates, order does not matter. Example: {Python, Java, SQL}
- **Subset (A ⊆ B):** Every element in A is also in B. If A = {1, 2} and B = {1, 2, 3, 4}, then A ⊆ B
- **Union (A ∪ B):** All elements from both sets. {1, 2} ∪ {2, 3} = {1, 2, 3}
- **Intersection (A ∩ B):** Only elements in both sets. {1, 2} ∩ {2, 3} = {2}
- **Difference (A - B):** Elements in A but not in B. {1, 2, 3} - {2} = {1, 3}

Applications in Relational Databases

SQL operations map directly to set operations:

Set Operation	SQL Equivalent	Example Use
Union	UNION	Combine customer lists
Intersection	INNER JOIN	Find common records
Difference	EXCEPT / NOT IN	Find exclusive records

Set Theory in Search Algorithms

Search engines use set operations for indexing:

- Each search term maps to a set of document IDs
- **Boolean AND:** Intersection of document sets (narrows results)
- **Boolean OR:** Union of document sets (broadens results)
- **Boolean NOT:** Set difference (excludes documents)

Set Theory in AI and Big Data

- **Classification:** Grouping data points into sets based on features
- **Clustering:** Creating sets of similar items (K-means, DBSCAN)
- **Feature Selection:** Choosing subsets of relevant attributes
- **Deduplication:** Using set properties (uniqueness) to clean data

Activity Preparation: Venn Diagrams

Practice visualising these queries:

- **Intersection:** "Customers who bought Product A AND Product B" (overlapping region)
- **Union:** "Users in Group X OR Group Y" (entire shaded area)
- **Difference:** "Orders this month but NOT last month" (A minus the overlap)