IB Computer Science SL 2026 — System Fundamentals

IB SL Study Guide

Contents

Systems in Organizations

A **system** is a set of interrelated components that work together to achieve a common goal. In an organizational context, an information system processes data to produce useful information for decision-making.

Planning and Installation

Before any system is installed, organizations follow a structured planning process:

1. **Feasibility study** — assesses whether the new system is technically possible, economically viable, and legally compliant
2. **Requirements analysis** — stakeholders (managers, end users, clients) define what the system must do
3. **System specification** — a formal document listing functional and non-functional requirements
4. **Resource allocation** — hardware, software, personnel, and budget are identified

> **IB TIP**
>
> The IB syllabus specifically mentions that students should be able to identify the relevant stakeholders for a given system change. In Paper 1 questions, always name specific groups (e.g., "warehouse staff", "HR managers") rather than saying "users" generically.

User Focus and Stakeholders

A **stakeholder** is any person or group affected by the system, including:

- **End users** — people who interact directly with the system daily
- **Managers** — need reports and dashboards; define business requirements
- **IT staff** — responsible for installation, maintenance, and security
- **Clients/customers** — external parties whose data may be held
- **Suppliers** — may interface with procurement or inventory systems

User focus means designing and evaluating systems from the end-user perspective, ensuring usability, accessibility, and fit-for-purpose design.

System Backup

Backup strategies protect data against loss due to hardware failure, human error, or disaster.

Strategy	Description	Pros	Cons
Full backup	Complete copy of all data	Simple restoration	Slow, large storage
Incremental backup	Only changed files since last backup	Fast, efficient	Complex restoration (chain of backups needed)
Differential backup	Changed files since last **full** backup	Faster restore than incremental	Grows larger over time

MEMORISE THIS

Backup Rule of 3-2-1: Keep **3** copies of data, on **2** different media types, with **1** copy offsite (e.g., cloud storage).

EXAM ALERT

A common exam trap is confusing incremental and differential backups. Incremental saves only changes since the **last backup** (full or incremental). Differential saves changes since the **last full backup** only. Restoration from incremental requires the full backup plus every subsequent incremental; differential only needs the full backup plus the most recent differential.

Software Deployment

When deploying new software across an organization, administrators must consider:

- **Compatibility** — does the new software work with existing hardware and OS versions?
- **Licensing** — single-user, site, or concurrent licensing models
- **Training** — end-user training and documentation
- **Rollout strategy** — phased, direct, or pilot (see Change Management below)

System Design Basics

This section covers the hardware, software, and operating system components that form a functioning computer system.

Components of a Computer System

A computer system consists of hardware, software, data, and people working together.

Hardware components:

- **Central Processing Unit (CPU)** — executes instructions
- **Primary memory** — RAM and ROM (directly accessible by CPU)
- **Secondary storage** — hard drives, SSDs, optical media (non-volatile, persistent)
- **Input devices** — keyboard, mouse, scanner, microphone
- **Output devices** — monitor, printer, speakers

- **Communication devices** — network interface cards, modems

Software components:

- **System software** — operating system, utilities, device drivers
- **Application software** — word processors, browsers, ERP systems

System Resources

System resources are the components managed by the operating system to ensure efficient operation:

- **CPU time** — allocated to processes via scheduling algorithms
- **Primary memory (RAM)** — allocated to running processes; managed using techniques such as paging and virtual memory
- **Secondary storage** — file system management, read/write access
- **Network bandwidth** — managed by network stack and protocols
- **Peripheral devices** — controlled through device drivers and I/O management

> **IB TIP**
>
> When asked about resource management in exams, link each resource to a specific OS function. For example: "The OS allocates CPU time using a scheduling algorithm such as round-robin" rather than just listing "CPU" as a resource.

Role of the Operating System

The **operating system (OS)** acts as an intermediary between hardware and application software. Its core responsibilities are:

1. **Process management** — creates, schedules, and terminates processes; handles multitasking
2. **Memory management** — allocates RAM to processes; handles virtual memory and paging
3. **File management** — organizes files into a directory structure; controls read/write permissions
4. **Device management** — communicates with peripheral hardware via device drivers
5. **User interface** — provides either a Command-Line Interface (CLI) or Graphical User Interface (GUI)
6. **Security** — manages user authentication, access control, and system logs

> **MEMORISE THIS**
>
> **OS mnemonic — PM-FM-DU-S:** Process management, Memory management, File management, Device management, User interface, Security

Computer Architecture

This section examines the internal structure of the CPU and the fetch-decode-execute cycle by which it processes instructions.

CPU Structure

The CPU contains three main components:

Component	Function
Arithmetic Logic Unit (ALU)	Performs arithmetic operations (add, subtract) and logical operations (AND, OR, NOT, comparisons)
Control Unit (CU)	Fetches, decodes, and executes instructions; coordinates all CPU activity
Registers	Ultra-fast temporary storage locations inside the CPU

Key Registers

Register	Name	Purpose
PC	Program Counter	Holds the memory address of the **next** instruction to be fetched
IR	Instruction Register	Holds the **current** instruction being decoded/executed
MAR	Memory Address Register	Holds the address in memory to be read from or written to
MDR	Memory Data Register	Holds the data just read from or about to be written to memory
ACC	Accumulator	General-purpose register holding the result of ALU operations

The Fetch-Decode-Execute Cycle

The CPU continuously repeats three steps:

1. **Fetch** — the CU copies the instruction at the address in the PC into the IR; PC is incremented
2. **Decode** — the CU interprets the instruction in the IR
3. **Execute** — the CU activates the appropriate hardware (ALU, memory, I/O) to carry out the instruction

WORKED EXAMPLE

Worked Example — Trace through one cycle:

Suppose PC = 200, and memory address 200 contains the instruction `LOAD 50` (load the value at address 50 into the accumulator).

- **Fetch:** MAR ← 200; MDR ← memory[200] (instruction `LOAD 50`); IR ← MDR; PC ← 201
- **Decode:** CU interprets `LOAD 50` — it is a memory-read instruction
- **Execute:** MAR ← 50; value at address 50 transferred via MDR into ACC

Primary Memory

Type	Stands for	Volatile?	Contents
RAM	Random Access Memory	Yes (lost on power-off)	Currently running programs and data
ROM	Read-Only Memory	No (permanent)	Firmware, bootstrap loader (BIOS/UEFI)
Cache	—	Yes	Frequently accessed instructions/data; faster than RAM

Secondary Storage

Secondary storage is non-volatile and used for long-term data persistence.

Type	Technology	Speed	Capacity
HDD	Magnetic spinning platters	Slower	High (low cost/GB)
SSD	NAND Flash memory	Fast	Medium-high
Optical (CD/DVD/Blu-ray)	Laser	Slow	Low-medium
USB Flash	NAND Flash	Fast	Low-high
Cloud storage	Remote servers over network	Depends on bandwidth	Effectively unlimited

EXAM ALERT

IB Paper 1 frequently asks students to “state one advantage and one disadvantage” of a storage type. Learn at least two distinct advantages and disadvantages for HDD, SSD, and cloud storage. Never repeat the same property as both an advantage and disadvantage.

System Lifecycle

The system lifecycle (also called the Systems Development Life Cycle, SDLC) describes the stages a system passes through from inception to retirement.

SDLC Phases

1. **Analysis** — determine what the current system does and what the new system must do; produce a requirements specification
2. **Design** — plan the architecture, data structures, user interface, and algorithms; no coding yet
3. **Development (Implementation)** — write the code and assemble hardware based on the design
4. **Testing** — verify the system works correctly and meets requirements
5. **Implementation** — install and deploy the system for real users
6. **Maintenance** — fix bugs, add features, and adapt to changing requirements over the system's operational life

> **IB TIP**
>
> The IB syllabus uses the term "implementation" for **both** the coding phase and the deployment/installation phase depending on context. In the lifecycle, "implementation" typically means deployment to users. Clarify which meaning applies based on context.

Stakeholder Involvement Across the Lifecycle

Phase	Primary Stakeholders Involved
Analysis	Managers, end users, clients
Design	IT architects, developers, managers
Development	Developers, database administrators
Testing	Testers, end users (UAT), quality assurance team
Implementation	IT staff, managers, all end users
Maintenance	IT support staff, developers, all users

Types of Testing

Testing ensures the system is correct, reliable, and meets user requirements.

Testing Strategies

Unit testing

- Tests individual modules or functions in isolation
- Carried out by developers
- Uses test data: normal, boundary, and erroneous values

Integration testing

- Tests that modules work correctly together once combined
- Identifies interface errors between components

System testing

- Tests the complete, integrated system against the requirements specification
- Checks functional requirements (does it do what it should?) and non-functional requirements (performance, security, usability)

User Acceptance Testing (UAT)

- Carried out by end users (not developers)
- Confirms the system is fit for purpose in a real-world environment
- The final gate before deployment

Alpha testing

- Internal testing within the development organization
- Often by a dedicated QA team rather than the original developers

Beta testing

- Testing by a selected group of **external** users (not employed by the developer)
- Provides feedback on real-world usage before full public release

MEMORISE THIS

Testing hierarchy: Unit → Integration → System → UAT → (Alpha → Beta for commercial products)

Test Data

When designing test cases, three categories of data must be used:

Category	Description	Example (for a field accepting 1–100)
Normal (valid)	Typical expected input	50
Boundary (extreme valid)	Values at the exact edge of the valid range	1, 100
Erroneous (invalid)	Data that should be rejected	0, 101, “abc”, negative numbers

EXAM ALERT

Boundary data does **not** mean values just outside the boundary. Boundary values are at the exact limit of what is acceptable (e.g., 1 and 100 for a range of 1–100). Values just outside (0 and 101) are erroneous data. The IB mark scheme is strict about this distinction.

WORKED EXAMPLE

Worked Example — Designing a test table:

A system accepts a student's age, which must be between 11 and 18 inclusive.

Test case	Input	Category	Expected output
1	15	Normal	Accepted
2	11	Boundary	Accepted
3	18	Boundary	Accepted
4	10	Erroneous	Rejected — error message
5	19	Erroneous	Rejected — error message
6	"abc"	Erroneous	Rejected — error message

Change Management

When an organization moves from one system to another, it must manage the transition carefully to minimize disruption. There are four main changeover strategies.

Direct Changeover (Big Bang)

The old system is switched off and the new system is switched on simultaneously.

Pros	Cons
Cheap and fast	High risk: if the new system fails, there is no fallback
No resources wasted running two systems	Staff may be unprepared
Clean transition, no synchronization issues	Data loss risk if migration fails

Best for: Small organizations, non-critical systems, or when the old and new systems are incompatible.

Parallel Running

Both the old system and the new system run simultaneously for a period. Outputs are compared to verify the new system is correct.

Pros	Cons
Very low risk: old system is a fallback	Expensive — doubled hardware, software, and staff costs
Results can be cross-checked for accuracy	Staff must enter data twice
Confidence built before switching off old system	Complex to manage

Best for: Critical systems where data integrity is paramount (e.g., banking, payroll).

Phased Implementation

The new system is introduced in stages (by department, location, or functionality) over time.

Pros	Cons
Problems contained to a small area before full rollout	Interfaces between old and new parts of the system must be managed
Organization can learn and adapt between phases	Slower overall transition
Lower immediate training demands	Inconsistency within the organization during the transition

Best for: Large organizations with many departments, or modular systems.

Pilot Testing (Pilot Running)

The new system is introduced fully in one part of the organization (e.g., one branch) before a wider rollout.

Pros	Cons
Real-world testing with limited risk	Selected site may not be representative of all locations
Problems caught before full deployment	Slow if many pilots are run
Staff at pilot site become expert trainers	Potential resentment from non-pilot sites

Best for: Multi-site organizations such as retail chains or school districts.

MEMORISE THIS

Change strategies summary:

- **Direct** — fast and cheap, high risk
- **Parallel** — safe but costly
- **Phased** — gradual by module or department
- **Pilot** — real test at one site first

Data Migration

When moving to a new system, existing data must be transferred. Data migration challenges include:

- **Format incompatibility** — data stored in one format (e.g., .mdb) must be converted to another (e.g., SQL)
- **Data cleaning** — duplicate, incomplete, or outdated records must be identified and corrected

- **Validation** — migrated data must be checked against the original for completeness and accuracy
- **Downtime** — migration may require the system to be offline during transfer

> **IB TIP**
>
> When asked to "describe a problem that may arise during data migration," always provide a specific example. Do not just write "data may be lost" — write "data stored in the old system's proprietary format (e.g., comma-separated values) may not map correctly to the relational database schema of the new system, causing fields to be truncated or misassigned."

Video Resources

Watch: IB CS Topic 1 — System Fundamentals Overview VIDEO

Practice Questions

▶ Q1 — Identify two stakeholders for a school's new online grade reporting system and state one requirement each stakeholder would have. [4 marks]

▶ Q2 — Explain the difference between incremental and differential backup strategies. [4 marks]

▶ Q3 — State the role of the Program Counter (PC) register during the fetch stage of the fetch-decode-execute cycle. [2 marks]

▶ Q4 — A hospital is replacing its patient records system. Justify why parallel running would be a more appropriate changeover strategy than direct changeover. [4 marks]

▶ Q5 — A developer is testing a module that accepts exam scores from 0 to 100 inclusive. Construct a test table with five test cases, stating the input, category, and expected result for each. [5 marks]