Instruction Processing & Data Flow

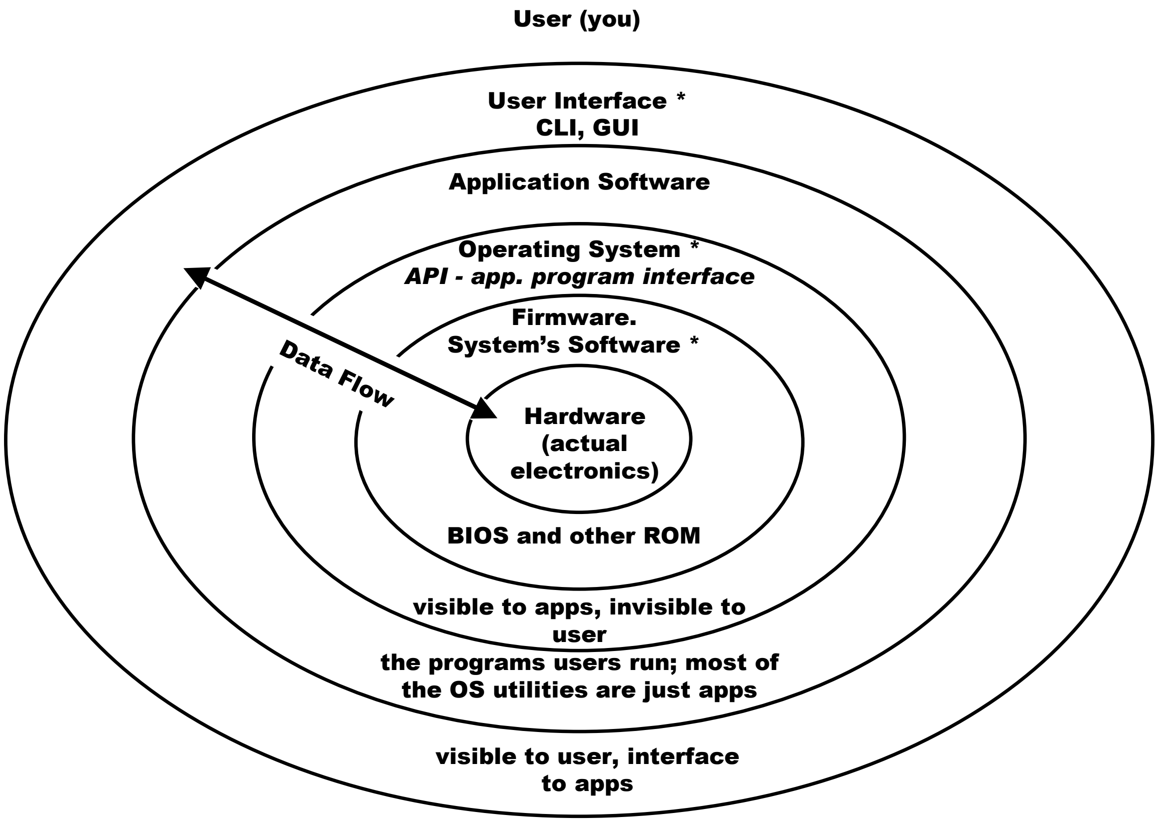
Although an intimidating title, the "Instruction Processing & Data Flow" hierarchy simply diagrams the path data flows through a computer system from *the user* to *physical hardware,* and *physical hardware* back to *the user*.

> **Note**: Everything is *data* to the CPU!
> User input, application software instructions, operating system instructions, and processed BIOS routines are all *data* translated by the CPU to perform operations on other data—everything stored or processed is viewed the CPU as just data with no distinction as to source (BIOS, operating system, or application program).
>
> And the key idea is difficult to grasp. The CPU only ever executes (or "sees") a single program. It really does not see individual "applications," just a single long sequence of instruction and data.

The flow of data within the layers of software (see diagram below) is concentric. This means that any data input to the system must pass through the *user interface, application software, operating system (OS),* and *firmware* before reaching the *hardware*, and any response from the hardware must pass through all layers back to you.

This causes some confusion in understanding, since the user interacts only with hardware and never with software. To keep order in the *user/system* relationship, it is assumed that when discussing software, the user is the last component after the UI. When discussing hardware, the user is the first component before hardware.

It seems quite long and time-consuming to involve so many layers of software, but the multi-levelled communication path is necessary to keep the flow of data timely, consistent, and most importantly, reliable and secure. A similar hierarchy is also defined when discussing network communication (for example, the OSI model).


User (you)
User Interface *
CLI, GUI
Application Software
Operating System *
API - app. program interface
Firmware.
System's Software *
Data Flow
Hardware (actual electronics)
BIOS and other ROM
visible to apps, invisible to user
the programs users run; most of the OS utilities are just apps
visible to user, interface to apps

(* - indicates that the layer is, to some degree, a component of the operating system.)

The role of the operating system (OS) is to allow communication and interaction between applications and the system's hardware, but the operating system itself *an application program that requires interaction with the hardware*.

For practical purposes, the operating system acts as a "proxy" for other applications, handling requests for input, execution of application instructions (by CPU), storage (RAM, disk), and output. If the CPU could be interviewed, it would seem to be executing <u>only one</u> application running: <u>the operating system</u>.

…well, actually, that's not so true. The operating system is not the lowest application running, since BIOS routines are also involved to exchange instructions and data to/from the RAM and CPU across the internal, system bus. In an interview, the CPU might say that it only sees BIOS routines, and these are pretty advanced and doing some pretty crazy things (since the implication is that the BIOS routines are acting as a "proxy" for the operating system).

For clarity's sake, the term "operating system" refers to both the *BIOS routines* (called "system's software" and firmware) and the *operating system* (which is just an application loaded from disk).

General Aspects of an Operating System (OS)

To completely understand the role and function of an operating system would require a solid understanding of how CPUs, system buses, and disk storage interact, along with a background in low-level program execution; clearly, these topics are beyond the scope of one lecture (and the course, in general).

The overview below decomposes the important concepts of operating systems, and provides a starting point for further reading on each section. (Operating systems are further discussed in COMP263—Introduction to Networking).

A. Resource Management

1. Secondary Storage (disk, tape; removable storage)
   - how disk files are stored (logical & physical: cylinders, sectors, clusters)
   - file indexing scheme; file system, allocation table (FAT) (also, see textbook for ***VFAT***)
   - verification and security of stored files
   - buffering and transfer (read/write) of data to secondary storage
   - co-ordination of *virtual memory* on disk (see Memory Management)

2. Input/Output device channels
   - communication parameters (serial, parallel, peripheral bus; data transfer rate (speed))
   - synchronisation between read and write devices and between communication devices (mouse-input, modem-communication)
   - Interrupt Request handling (IRQ's) to correctly communicate with devices

3. Memory Management
   - memory segmentation (memory blocks/pages; for Windows: *virtual 86/real mode*)
   - protection of secured memory (BIOS, OS)
   - allocation and security of memory for <u>devices</u> (I/O address), <u>direct memory access channels</u> (DMA), <u>ROM addressing</u> (reallocating ROM to secure RAM addresses), <u>OS and application software</u> (protected mode)
   - virtual memory (disk swapping control and memory page addressing)

B. Program Execution *(by CPU; coming from applications, OS, and firmware)*

1. Loading of programs from disk or ROM into RAM (note: programs can only be loaded from RAM)
2. Polling of next instruction to load from,
   - single-task OS (DOS) *polling* between all device I/O and application processing
   - multi-task OS (NT/2000/XP, UNIX) polling between
     - device I/O (IRQs, DMA)
     - all running applications
     - virtual memory, and the required segment to load or swap
3. Capture memory segment used by current routine/program (protected mode)
4. Execution of instruction and storage of necessary data in RAM
   - store result and memory address conflict resolution
   - determine if virtual memory swap required for reading or writing to memory
5. Interpret BIOS calls from application
   - disk access (data read or write)
   - memory reference (determine the correct memory block to access, "is the block available for current application? must it be loaded/swapped from disk? does it belong to the application?")
   - I/O channel request (result of IRQ of device)
   - operating system (DOS, Linux, etc.) software interrupt call, either,
     - an OS-specific software call, such as "hooking" into file access routine for write
     - a low-level BIOS call (to access specific hardware I/O, or CPU)
   - OS API (application program interface) request

C. User Interaction *(interface control and presentation)*

1. Presentation of user interface (UI), either,
   - Command Line Interface (CLI-"kl-eye")
   - Graphical User Interface (GUI-"gooey")
2. Interpretation of user's request through UI,
   - decision on action,
     - performed solely by the OS via an internal routine
     - direction or instruction to an application (running or otherwise)
   - determine if any output is returned to user *(is there anything for the UI to do?)*

D. Network Communication (Network-features of OS) *(*discussed in more detail in COMP263)*

(server-class operating systems: Netware, WinNT/2000/XP, UNIX/Linux, MacOS 10.x)

1. Network group and node identification (individual computer ID and location on LAN, WAN),
   - individual system ID management
   - location of nodes (devices) in LAN, WAN (*to which network does each device belong?)*

2. Protocol processing and data packet management (just above the hardware layer),
   - co-ordination of discrete data sets from various to/from various destination through various protocols
   - separation between network processing and local processing
     - scheduling and prioritising tasks (part of multi-tasking operations)

3. Security,
   - access to network system and local system, for both remote users and remote applications
   - protection of data in primary or secondary storage: visibility and access (read/write)

4. Authorisation of network nodes and relationship model: *peer-to-peer* or *client-server*,
   - which services are being offered (as a server)
   - identification of access level for defined groups (individuals) and under what conditions