Princípios da Computação

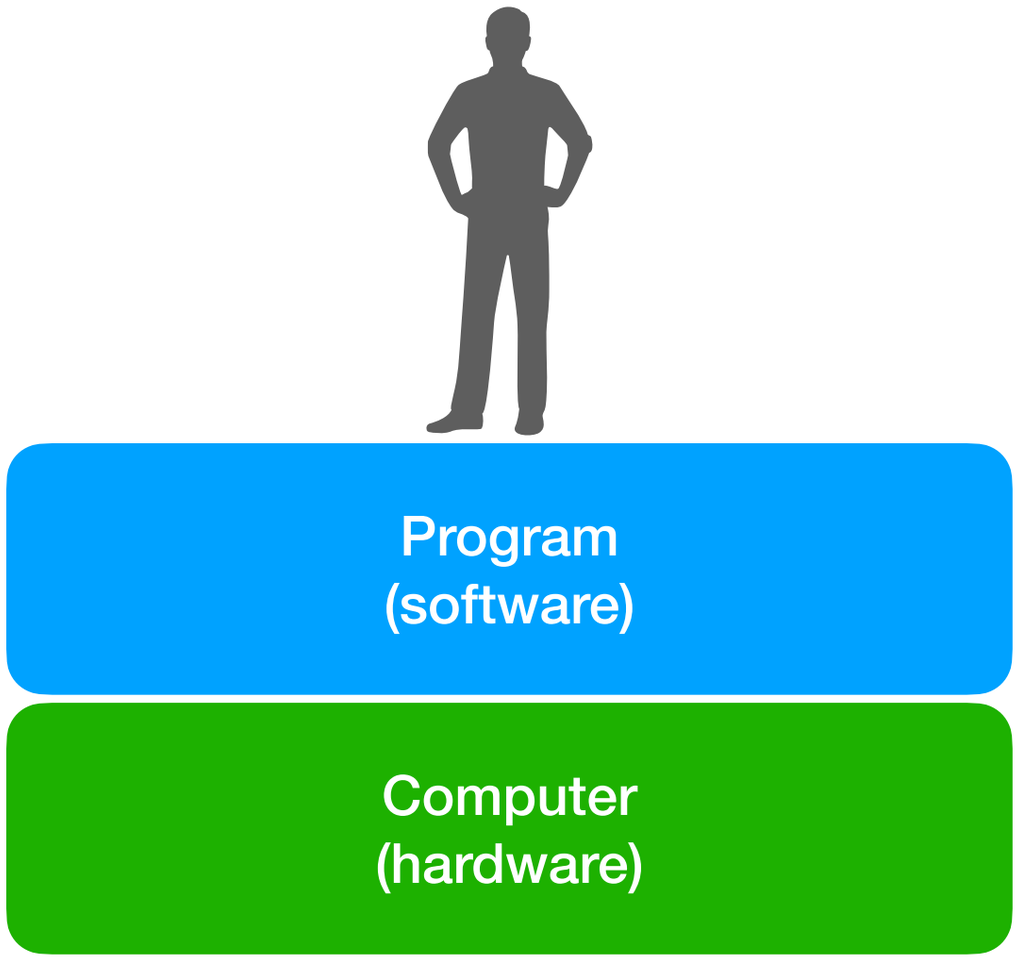
Introduction to operating systems.

Purpose of a computer

- The purpose of a computer is to execute programs that solve user-defined problems as efficiently as possible.

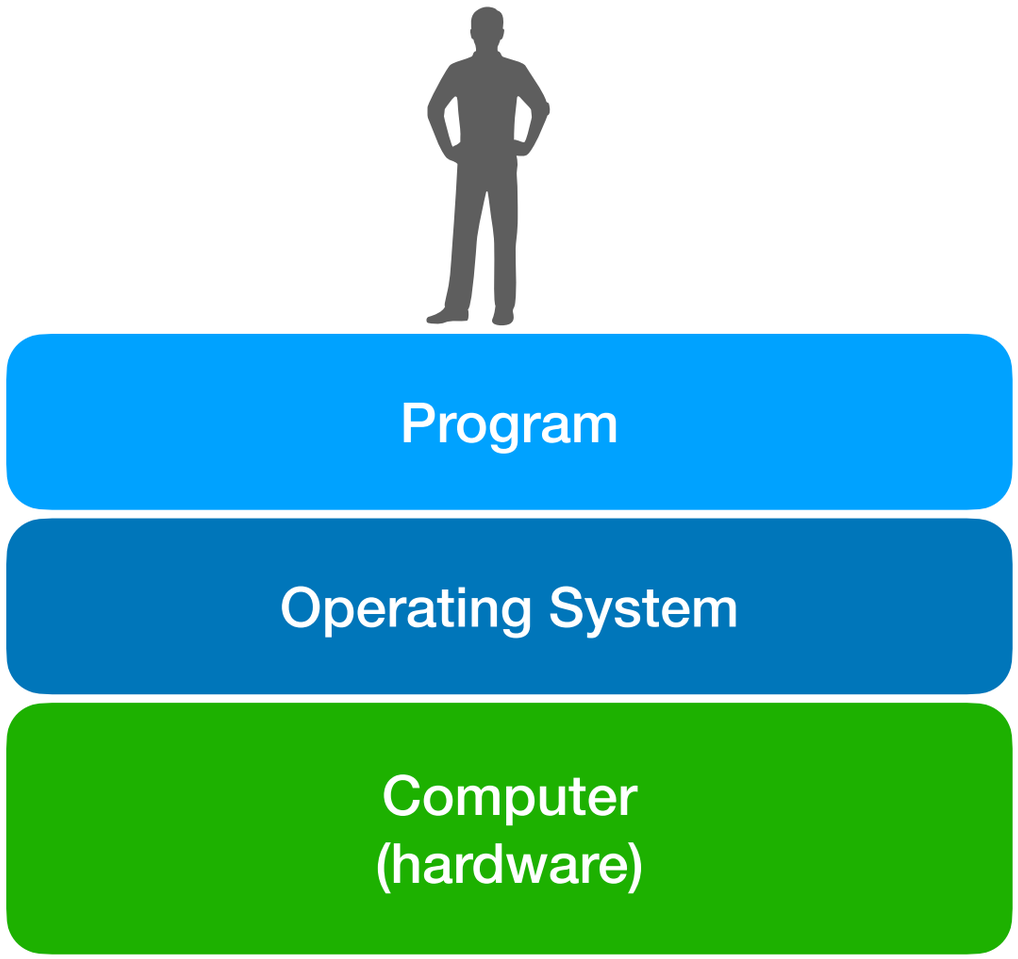
- Running a single program exclusively on a computer is often the most efficient approach. And it works!

- So, why do so many computers rely on an operating system?



The cost of an operating system

- An operating system comes with additional costs:
    - Requires extra memory (*space*) to load.
    - Uses processing power (*time*) to run OS code instead of user program code.
- So, why do we use operating systems?



Serial processing

Serial processing

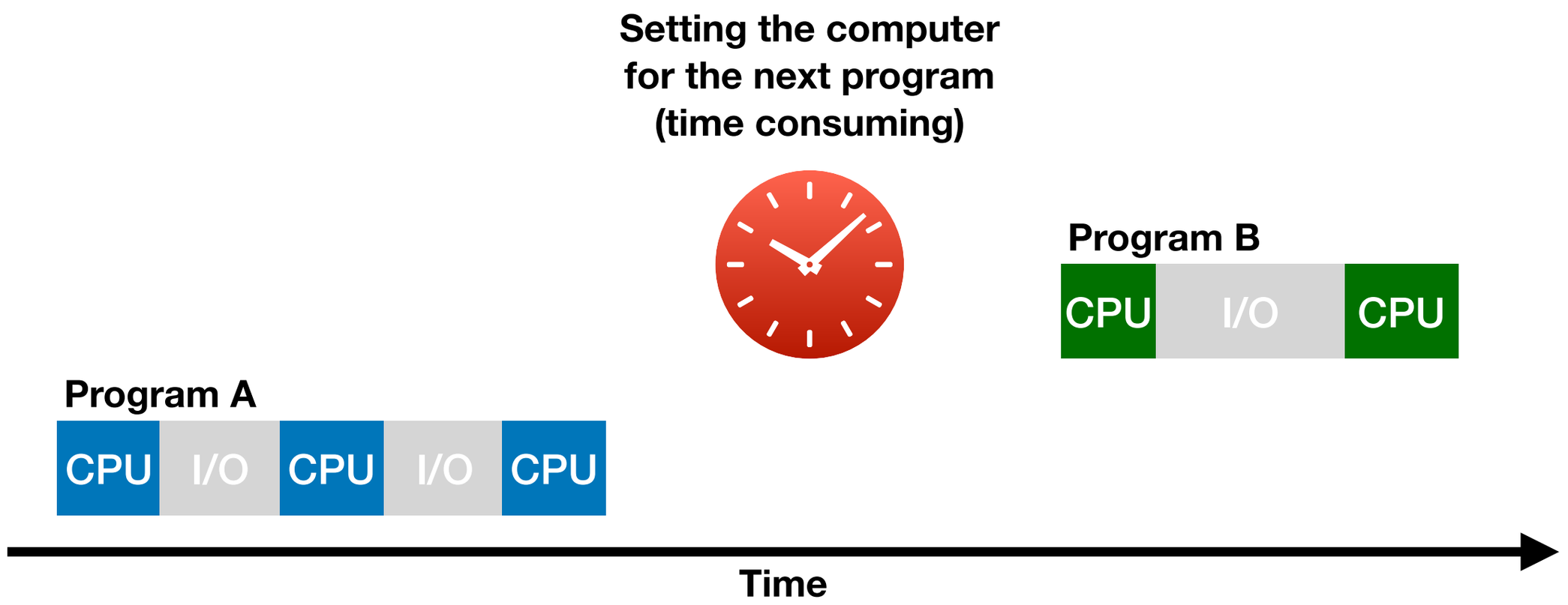
- Early stored-program computers had **no operating system** and **ran one program at a time**.

- Operators manually loaded each program into memory and started it via console buttons.

- Programs ran without user interaction, loading data from secondary storage (punched cards, tapes) during execution.

- After execution, operators repeated this process for each new program.

Serial processing



- Highly inefficient: most of the time, the CPU is idle.

Serial processing

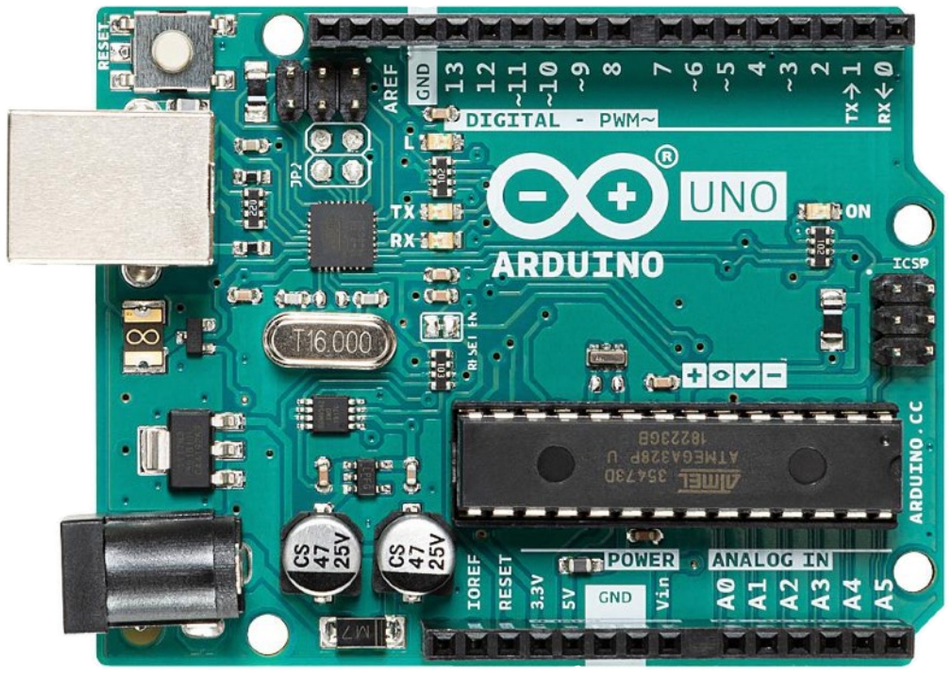
- Setting a computer to execute a program was very time-consuming: all human-performed operations.

- Detecting and correcting an error was even more time-consuming: inspecting machine code, registers and memory!

- The computer was a very expensive machine but, at the end of the day, its utilisation time was low!

Systems with no OS, today

- Single-program, specific-purpose systems.
    - **Focused Execution.** These systems run a single dedicated program throughout their lifetime. No time spent switching between programs.
- **No OS Overhead.** They operate without the additional costs of an operating system.



Systems with no OS, today

- **Embedded Systems.** Microcontrollers and processors in appliances, automotive systems, and consumer electronics typically run a single program without a traditional OS.

- **Dedicated Hardware.** Devices like network routers and switches often use firmware instead of a full OS.

- **Logic Controllers.** Programmable Logic Controllers (PLCs) in industrial settings execute predefined control sequences without an OS.

Batch processing

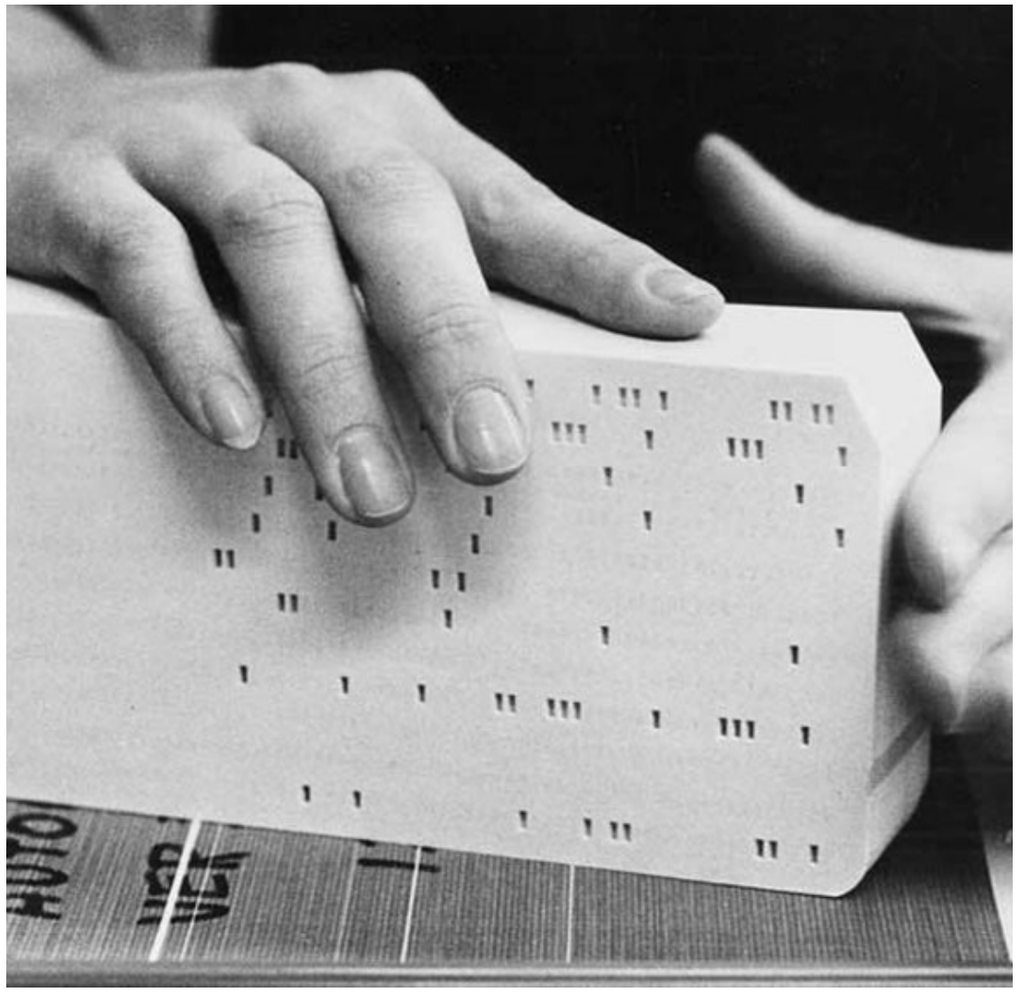
isep Instituto Superior de Engenharia do Porto

Batch processing

- Hardware speed improved over time, but programs remained non-interactive.

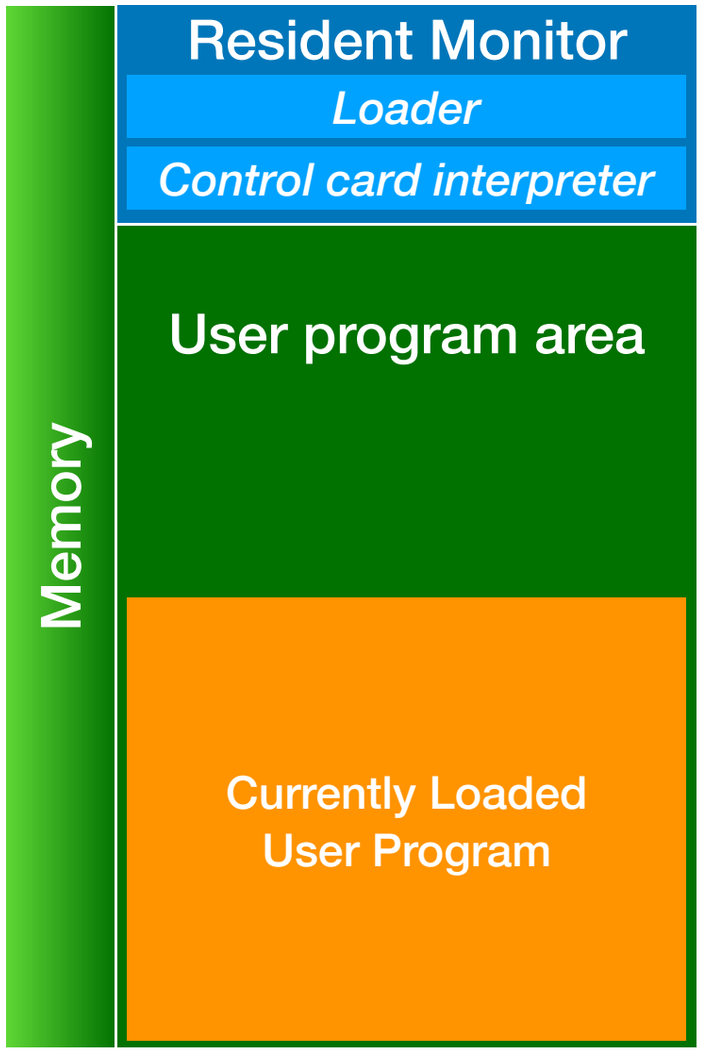
- Higher-level languages and libraries (common functions, device drivers) simplified programming.

- However, each program required manual compilation and loading before execution, adding complexity and reducing efficiency to computer operation.



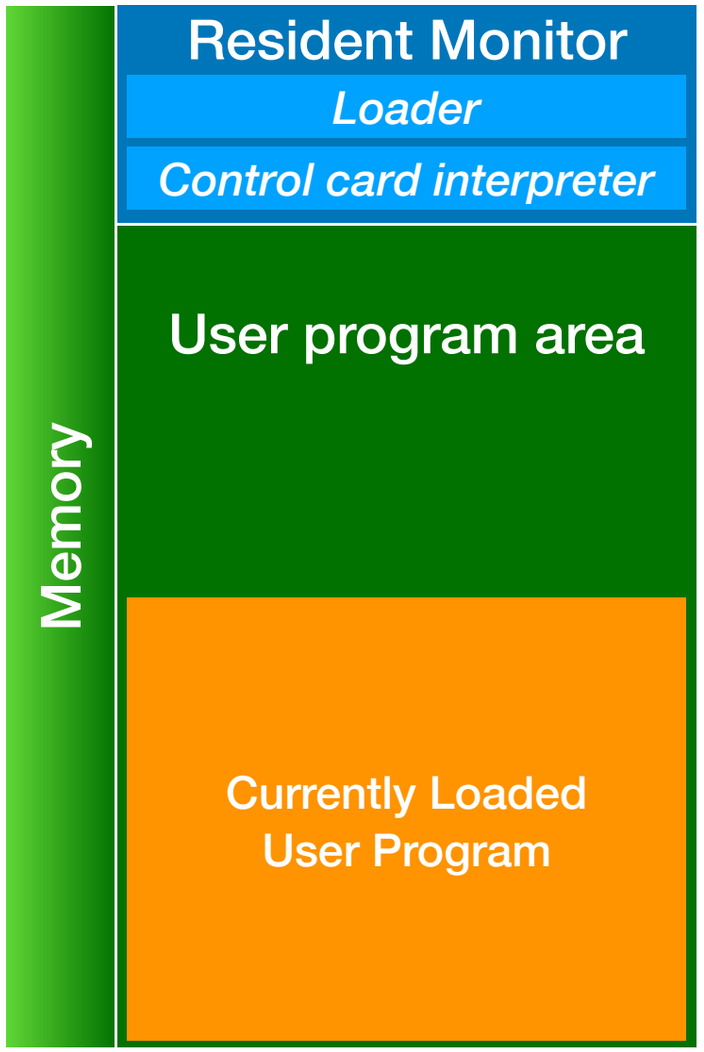
Batch processing

- Solution: automated job sequencing
- The **resident monitor** was the first form of an operating system.
- The **monitor** remained **resident** in memory during the whole operation of the computer.
- The monitor loads and starts a program; upon termination, control returns to the monitor, which then loads and starts the next program.



Batch processing

- The operator sets a deck of control cards to define the program sequence for the batch.
- The **control card interpreter** reads and executes the instructions on the cards.
- The **loader** loads programs into main memory.
- This sequence repeats automatically until the batch is complete.

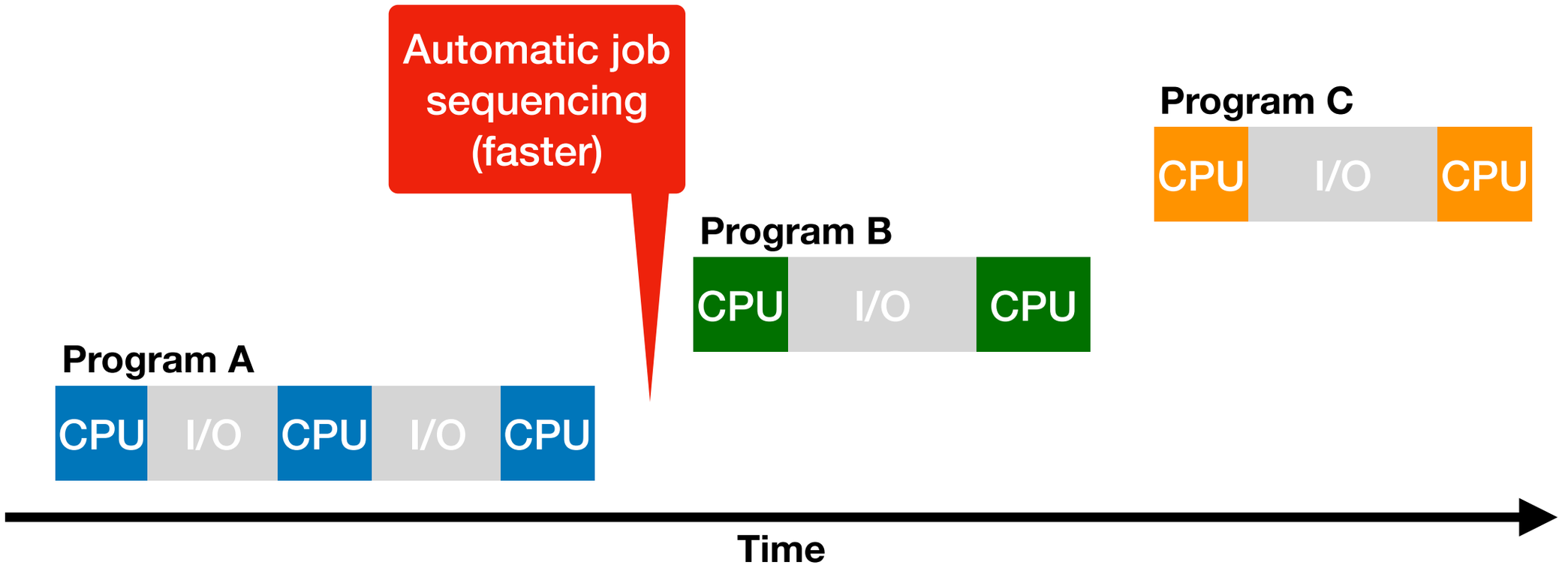


Batch processing

```
//COMPILE   JOB 'FORTRAN JOB',CLASS=A,MSGCLASS=A
//STEP1     EXEC PGM=IFORT
//INPUT     DD DSN=FORTRAN.PROGRAM,DISP=OLD,UNIT=CARD, LABEL=(1,SL)
//SYSLIST   DD SYSOUT=*   // List of compiler messages
//SYSOUT    DD SYSOUT=*   // Standard output

//ASSEMBLE   EXEC PGM=ASSEM
//OBJFILE   DD DSN=OBJECT.OBJ,DISP=(NEW,CATLG),UNIT=TAPE,
//             SPACE=(TRK,(1,1)),LABEL=(1,SL)
//SYSLIST   DD SYSOUT=*   // List of assembly messages
//SYSOUT    DD SYSOUT=*   // Standard output

//RUN        EXEC PGM=LOAD
//OBJECT    DD DSN=OBJECT.OBJ,DISP=SHR,UNIT=TAPE
//SYSOUT    DD SYSOUT=*   // Output from the executed program
//SYSPRINT  DD SYSOUT=*   // Print messages from the program

//ENDJOB    END
```

Illustrative JCL code generated by ChatGPT: a single job that compiles, assembles, and executes a FORTRAN program.

Batch processing



- Despite automation, CPU utilisation remains low.
- Program execution time is primarily determined by I/O operations.

Batch processing today

- Though resident monitors are mostly relics of the past, batch processing remains highly relevant.

- Batch processing enables computers to manage high-volume, repetitive data tasks efficiently.

- Compute-intensive jobs like backups, filtering, sorting, and big data mining are often inefficient as individual transactions.

- Supercomputers also process tasks one at a time, executing programs in batches.

Computers with Operating Systems

isep Instituto Superior de Engenharia do Porto

Advances in technology

- **Autonomous I/O Operations.** I/O devices with controllers could operate independently, signalling the processor upon completion, allowing it to perform other tasks concurrently.

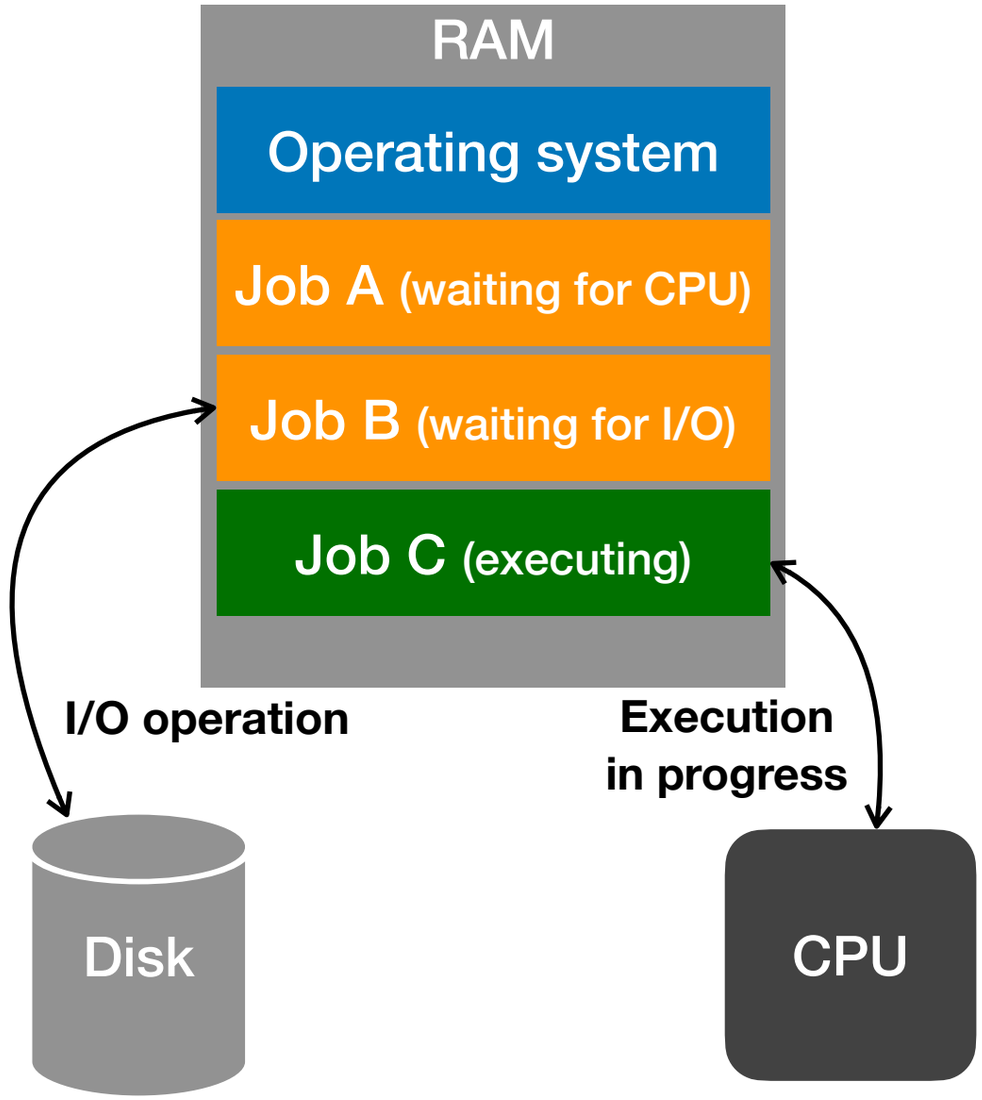
- **Increased Memory Capacity.** Higher memory density and lower costs led to computers with greater RAM capacity.

- **Disk Storage Innovations.** Disk systems enabled random access, serving as secondary memory where suspended programs could be temporarily stored and reloaded as needed.

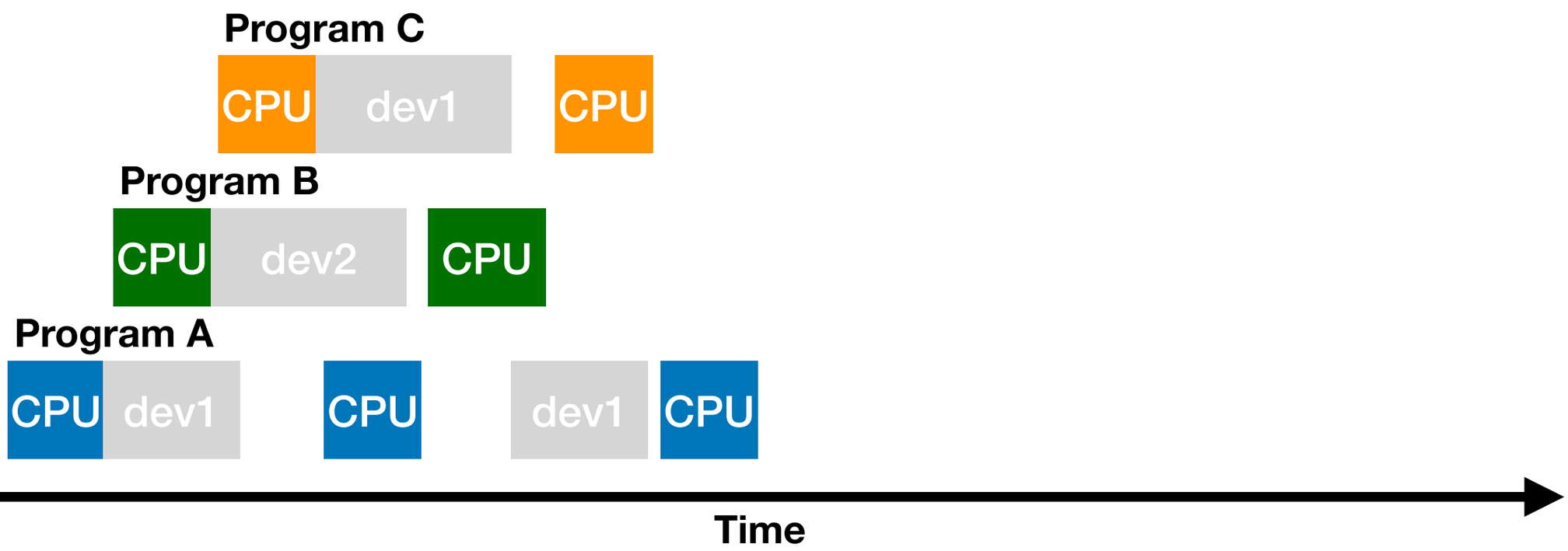
- **Enhanced User Interaction.** Programs could be loaded from disk, and users gained the ability to save files directly to disk.

Multiprogramming

- **Concurrent Program Loading.** Multiple programs are loaded into primary memory.
- **Interleaved Execution.** The CPU interleaves the execution of programs in RAM.
- **Efficient Resource Use.** When a program requests an I/O operation, it yields the processor to another program through a context switch.
- **Optimised CPU Utilisation.** The CPU stays active as long as there are tasks to process!



Multiprogramming



- **Efficient CPU Usage.** The CPU idles only when no program is ready to execute, maximising overall utilisation.

Time sharing

- Multiprogramming settled the ground for:
    - **interactive programs:** a program can wait for user input (I/O), while the CPU is potentially executing other programs;
    - **time-sharing:** multiple users share the computing resources, making a more efficient use of the computer.

Time sharing

- The user interacts with the computer using a **terminal.** The terminal is connected to the computer by a communications line or a network.

- The user **types on the keyboard** the next program to execute.

- Process switching is sufficiently fast to allow interaction between the users and their processes.

Where do we stand?

To use an OS or not?

- Dedicated systems running a single program throughout their lifetime don’t require an operating system.

- Systems running multiple concurrent programs, however, need additional support to manage resources.

- An operating system is specialized software that efficiently manages programs and resources according to system requirements and objectives.