

Chapter 4

Computer Organization and Architecture : The CPU, and the OS

Outline

- **Computer Program execution process and the CPU organization**
- **Hierarchical Organization of the Computer System (Architecture)**
- **The Operating System Functions and components**
- **Programming Concepts***
- **About Programming languages ***

Program Execution

- The operating system reads the program into memory
 - adds it to a list of programs that want to execute
- It gives it a time slice of the CPU
 - adds it to a list of programs that are executing
- It saves the current state of the CPU when it gives another program a time slice of the CPU
 - Context switching
- It restores the state of the CPU when a stopped program resumes execution.

When you **double click** on an icon to run a program, here is what happens:

1. The program, which is stored inside the hard disk drive, is transferred to the RAM memory.
2. The CPU, using a circuit called memory controller, loads the program data from the RAM memory as directed by the OS.
3. The data, now inside the CPU, is **processed**. A program is a **sequence of instructions to the CPU**.
4. What happens next will depend on the program. The CPU could **continue to load and execute the program** or could **do something with the processed data**, like **displaying something on the screen**.

The sequence of **CPU steps** can be expressed in pseudocode:

loop

***fetch** the instruction pointed by (the value in) IP*

advance the instruction pointer (IP)

***decode** the instruction*

***execute** the instruction*

*if memory operand needed, **read** value from memory*

*if result is memory operand, **write** result to memory*

else if ...

else ...

continue loop

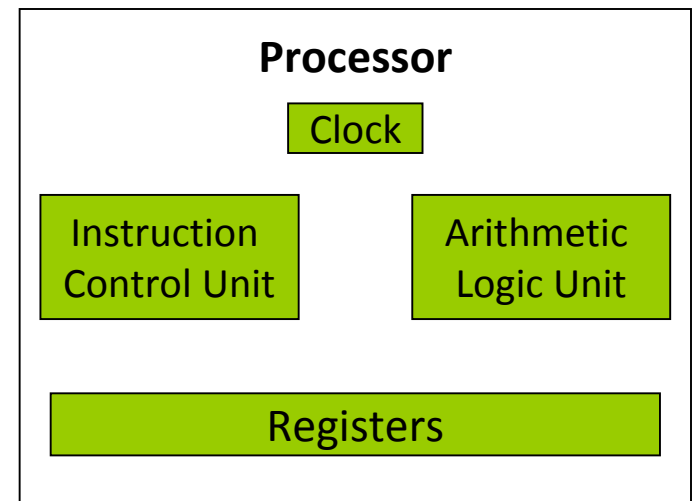


Table 5: List of simple CPU instructions.

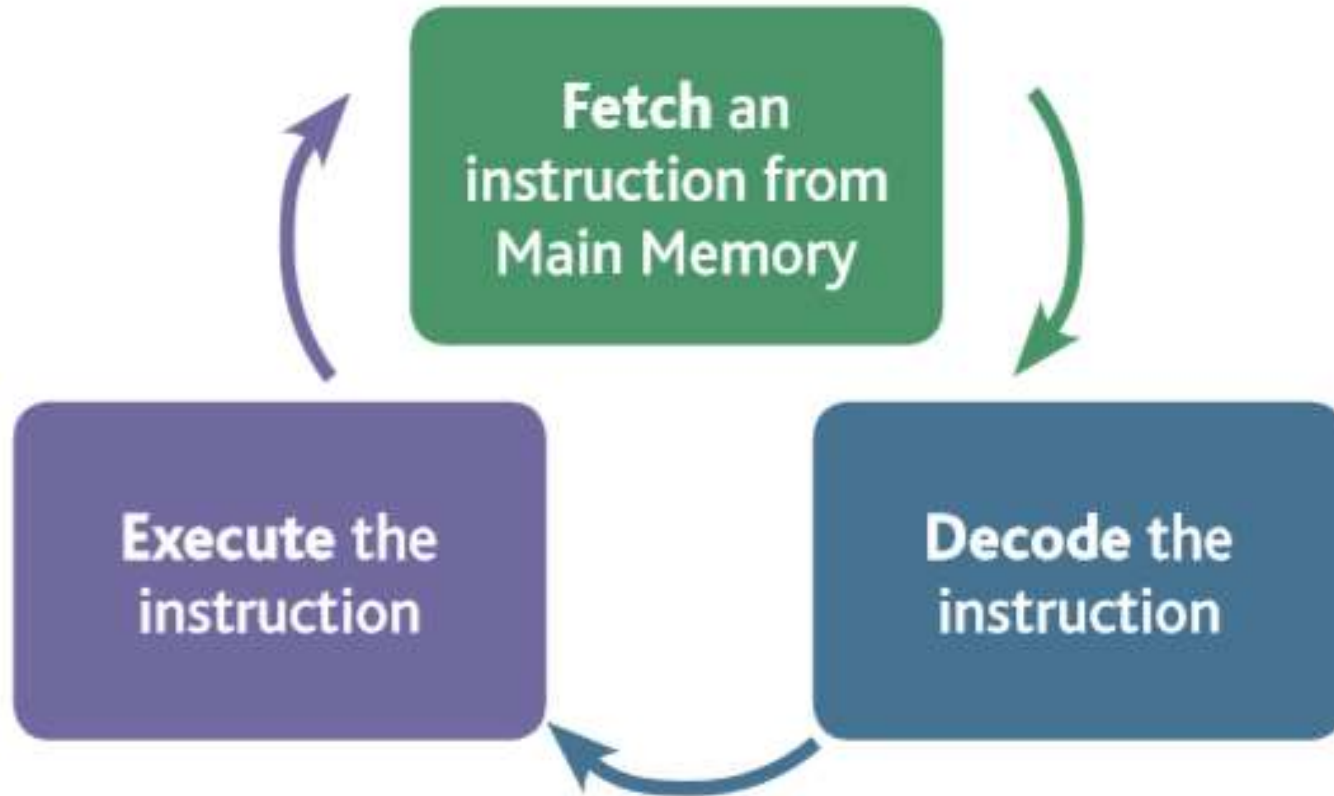
Instruction	Description
NOP	No operation
INPUT	Load Input into A register
OUTPUT	Move A register to Output register
JMP <addr>	JMP to address <addr> in program memory
LOAD A, #<value>	Load <value> into the A register
INC A	Increment the value in register A by 1
MOV B, A	Move register A into register B
ADD A, B	Add register A to register B and store result in register A
HALT	Halt execution

The CPU

- **CPU = Central Processing Unit**
- **Internal clock ticks very fast (e.g., 2.6 GHz = 2.6 billion ticks per second)**
 - activities are synchronized to start on a clock tick
 - some activities take more than one clock tick
- **Instruction execution is automatic**
 - (tick) find memory address of next instruction
 - (tick) retrieve instruction from memory
 - (tick) decode the instruction
 - (tick) fetch argument from memory if necessary
 - (tick) execute instruction
 - (tick) store result in memory if necessary

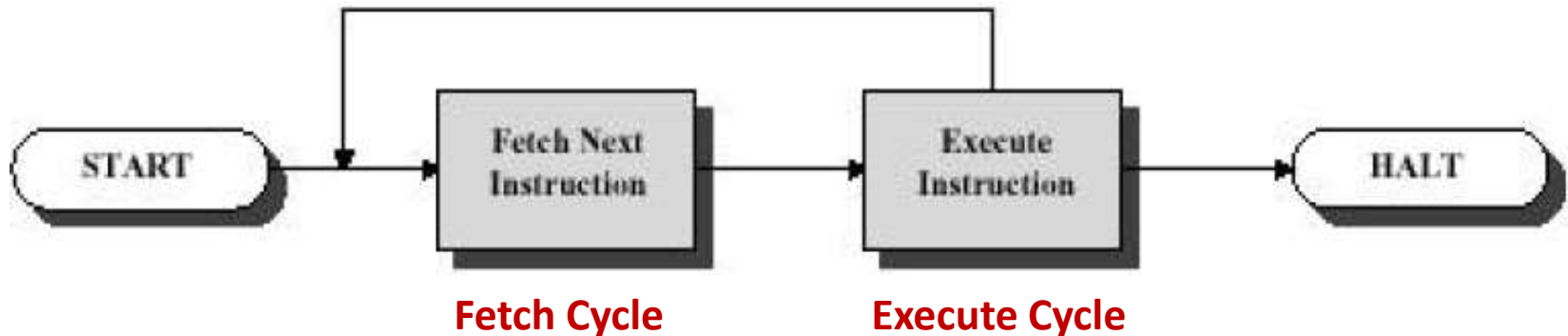
* Tick stands for start of a clock cycle.

The CPU Fetch-Execute Cycle



The CPU consists of a **control unit**, **registers**, the **arithmetic and logic unit**, the **instruction execution unit**, and the **interconnections (internal bus)** among these components.

- Processing required for a single instruction is called an **instruction cycle (Fetch-Execute Cycle)**, and can be viewed as shown below: **2 Steps**

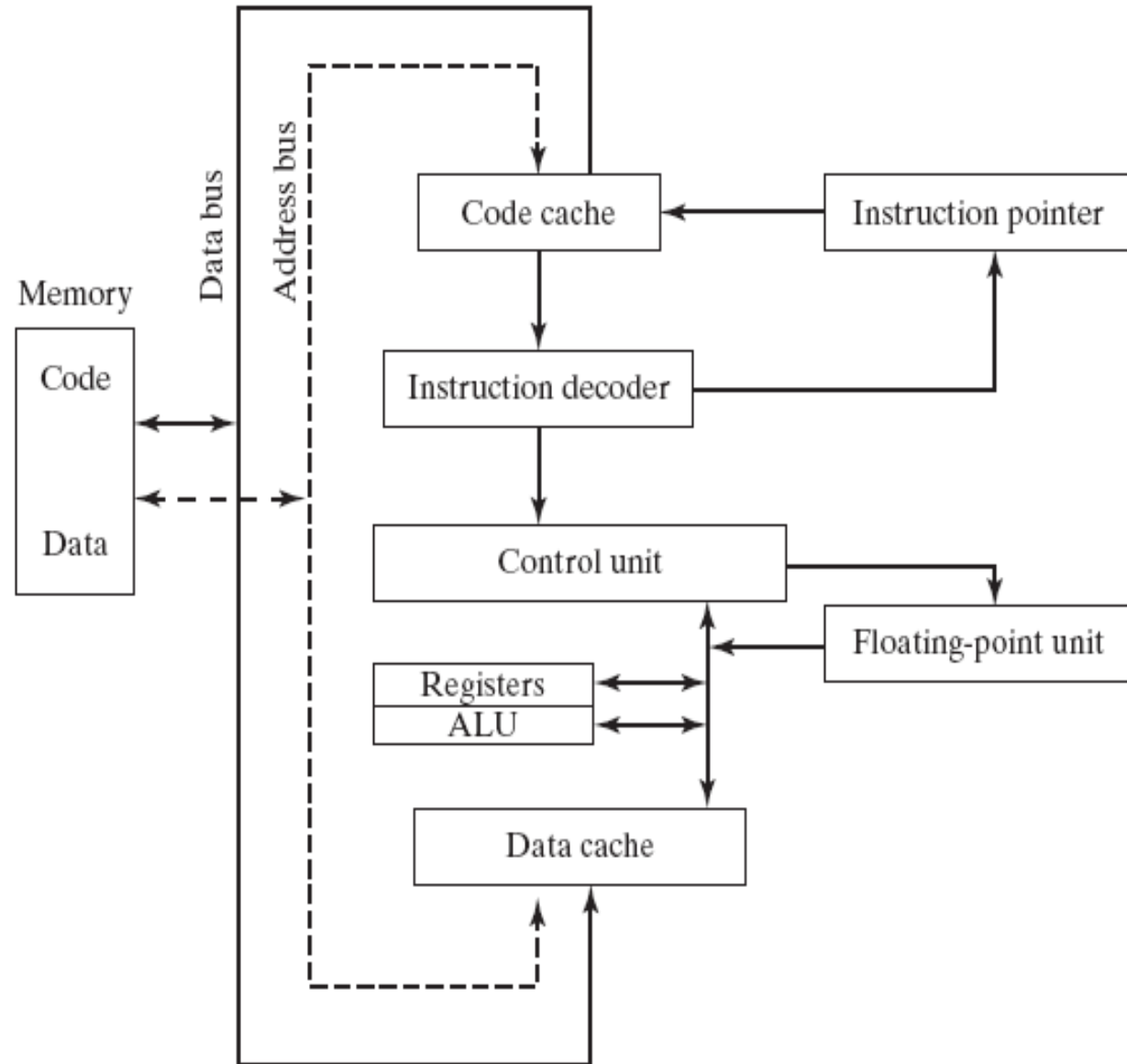
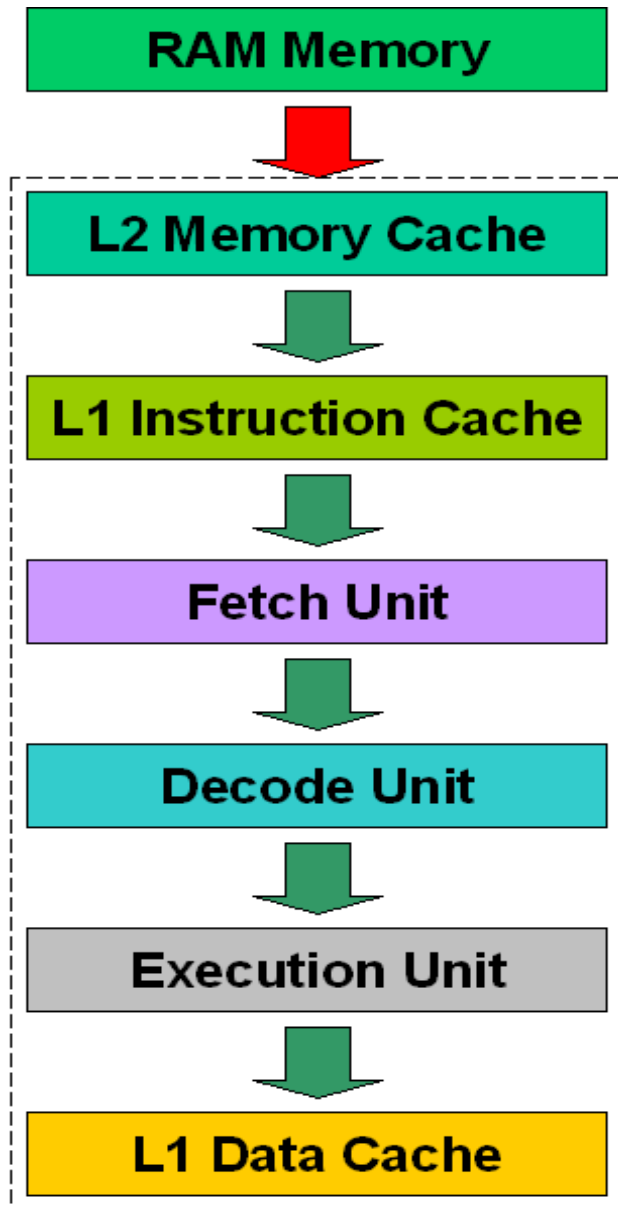


- ❑ **Fetch** – CPU(CU) reads an instruction from a location in memory and decodes the instruction (determine what it means)
 - Program counter (PC/Instruction Pointer) register keeps track of which instruction executes next
 - Fetches instruction is loaded into the instruction register (IR)
 - Normally, CPU increments PC after each fetch

❑ Execute - CPU executes the instruction

- May involve several operations
- May utilize previously changed state of CPU
- General categories:
 - **CPU-Memory**: Data may be transferred from CPU to memory or vice-versa
 - **CPU-IO**: Data may be transferred between CPU and an I/O module
 - **Data Processing**: CPU (**ALU**) may perform some arithmetic or logic operation on the data
 - **Control**: An instruction may specify that the sequence of execution be altered

Simplified block diagrams of a modern CPU

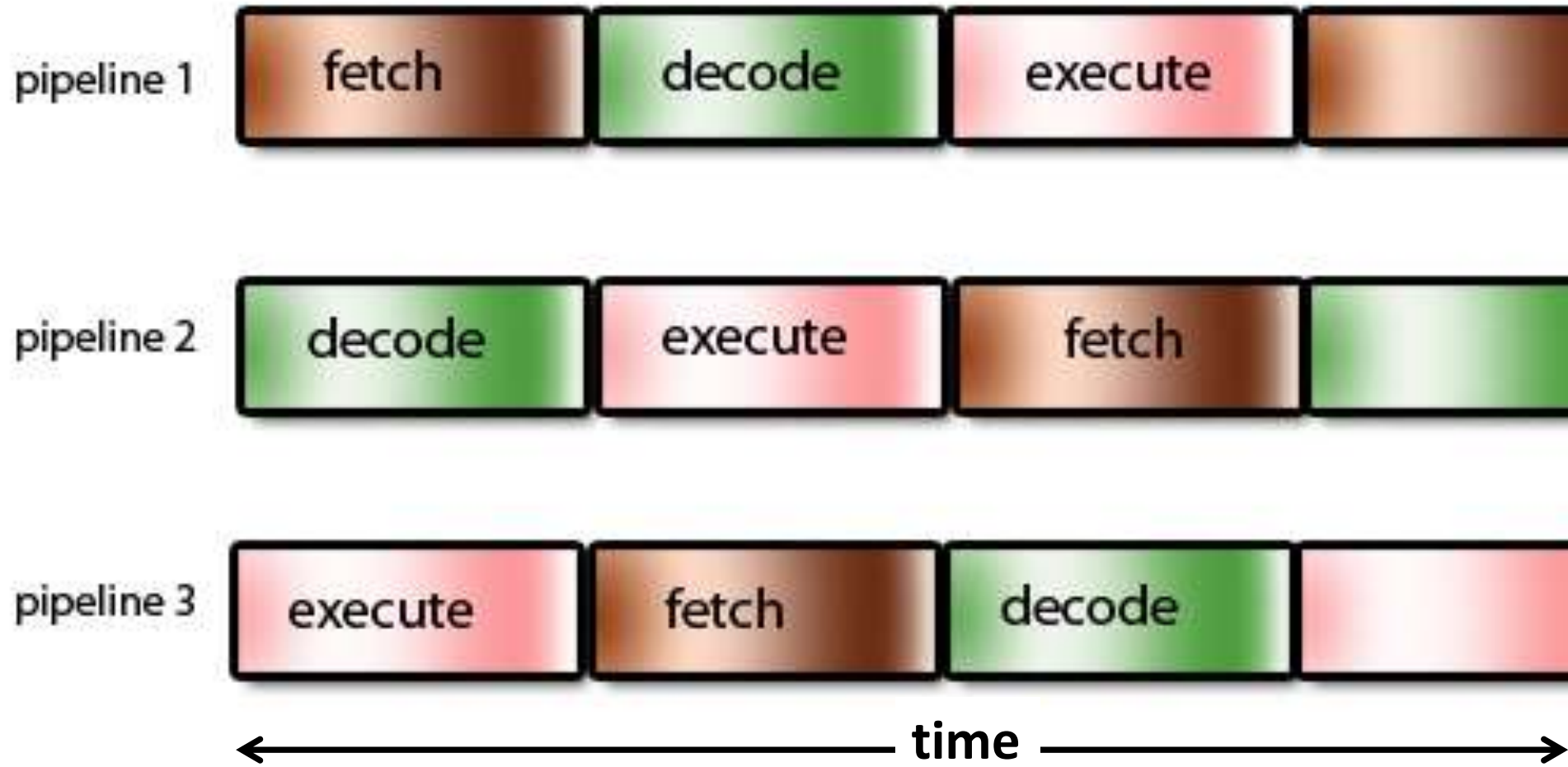


CPU Cont...

- Instruction execution takes place in discrete steps
 - Fetch, decode, load and store, arithmetic or logical
 - Usually require multiple clock cycles per instruction
- Pipelining → simultaneous execution of instructions

CU starts execution of next instruction while other instructions are still being processed in other parts of the CPU (or while waiting for some response).

Parallel processing with pipelines



Each pipeline is a separate part of the CPU

CPU Cont...

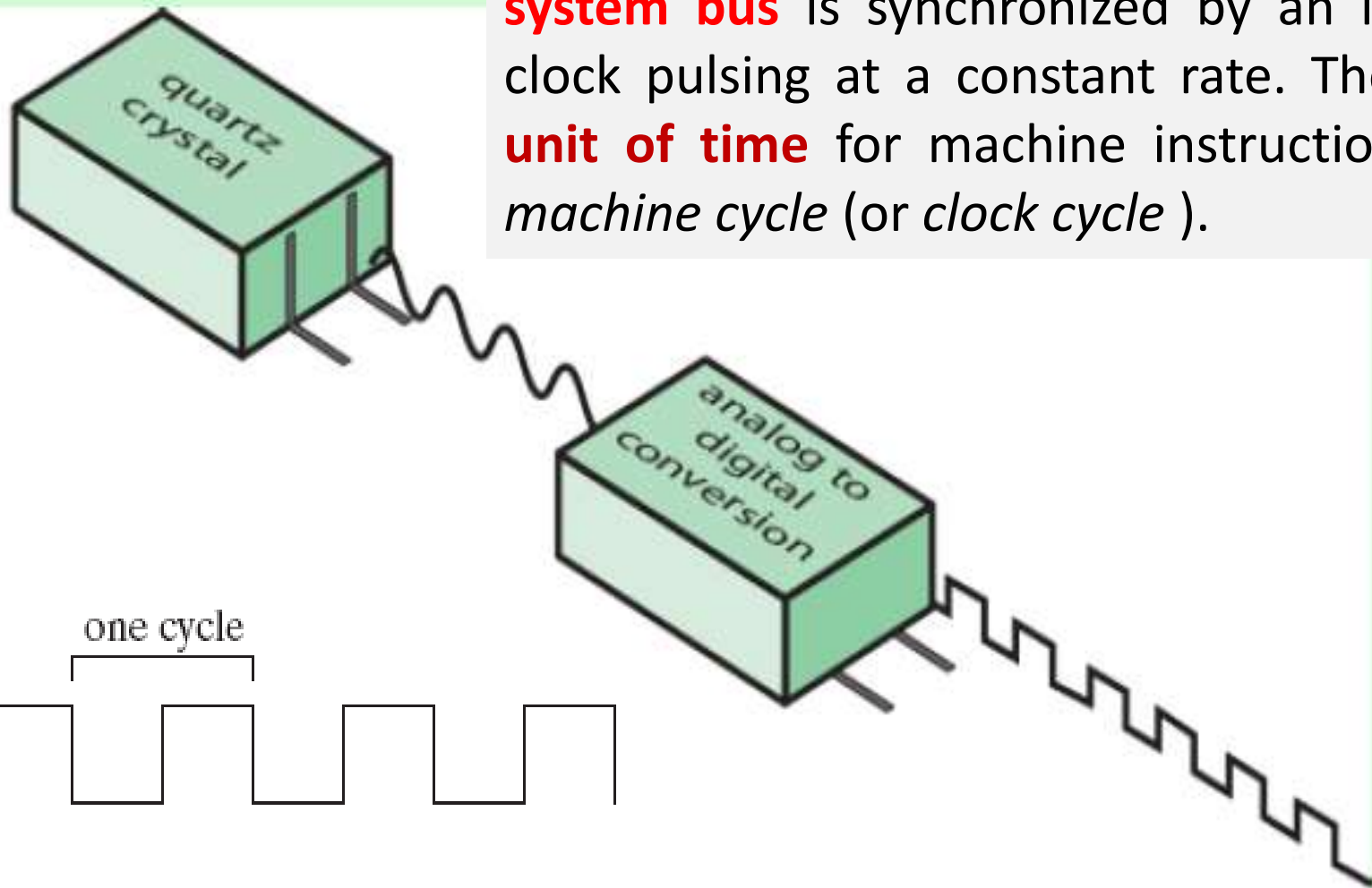
Processor speed depends on:

- Internal Clock Speed
- Type of Instruction Set
- Processor Implementation
- Compiler Design (efficient binary executable)
- Cache and Memory Hierarchy
- etc...

Reading Assignment: CISC & RISC, MIPS & MFLOPS

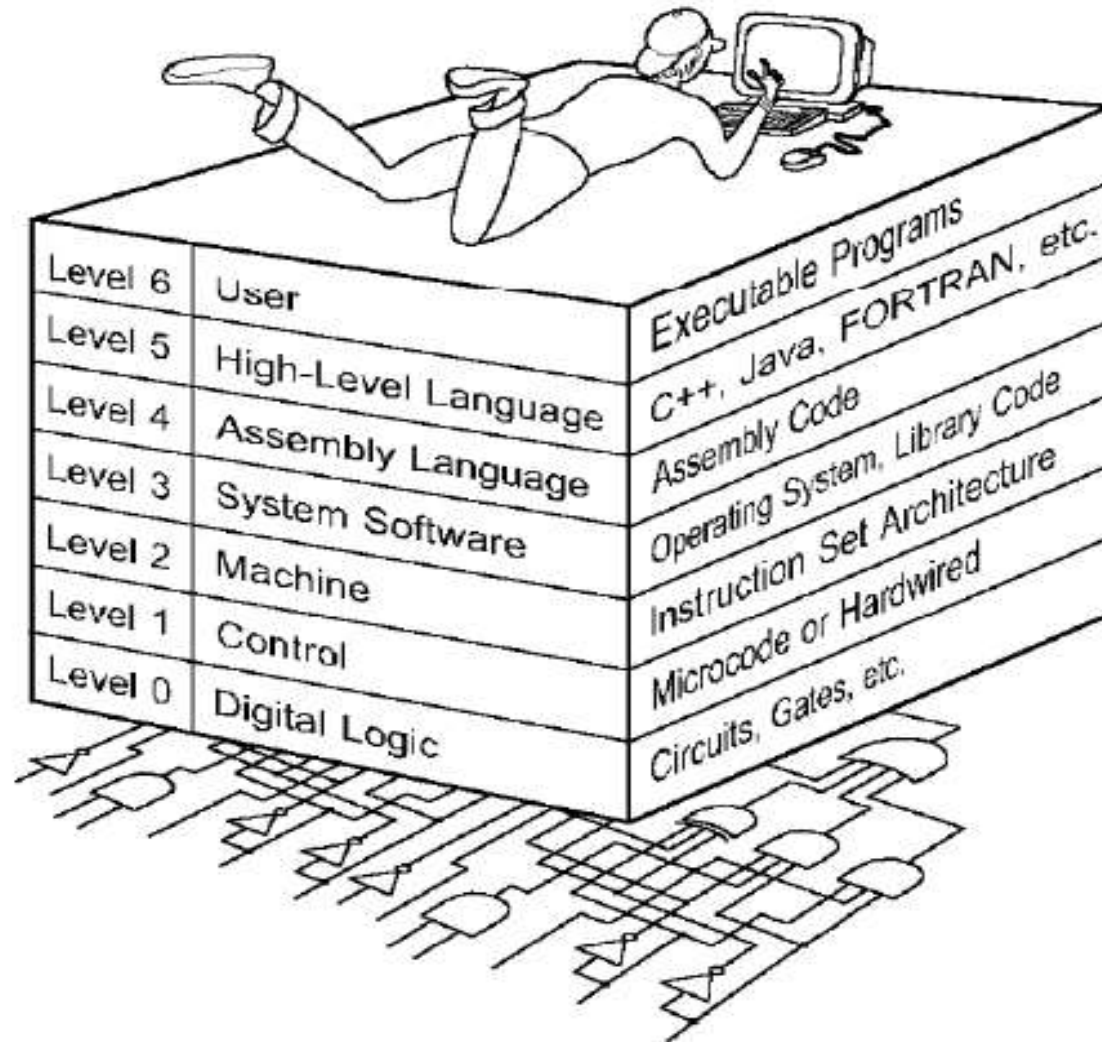
System Clock

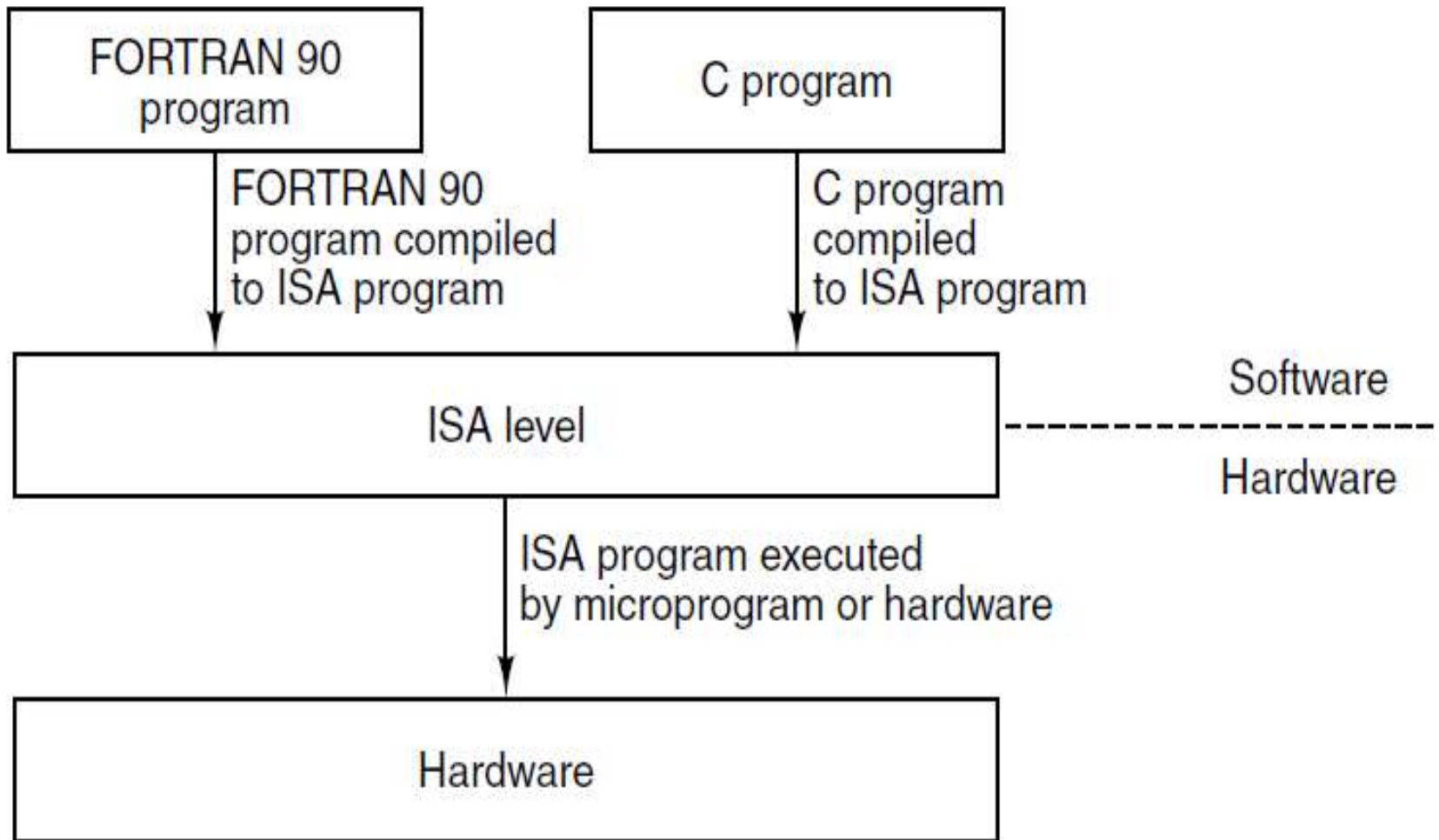
Each operation involving the CPU and the **system bus** is synchronized by an internal clock pulsing at a constant rate. The **basic unit of time** for machine instructions is a *machine cycle* (or *clock cycle*).



How much is the time that takes ten cpu cycles (10 clock ticks) in a 4 ghz processor?

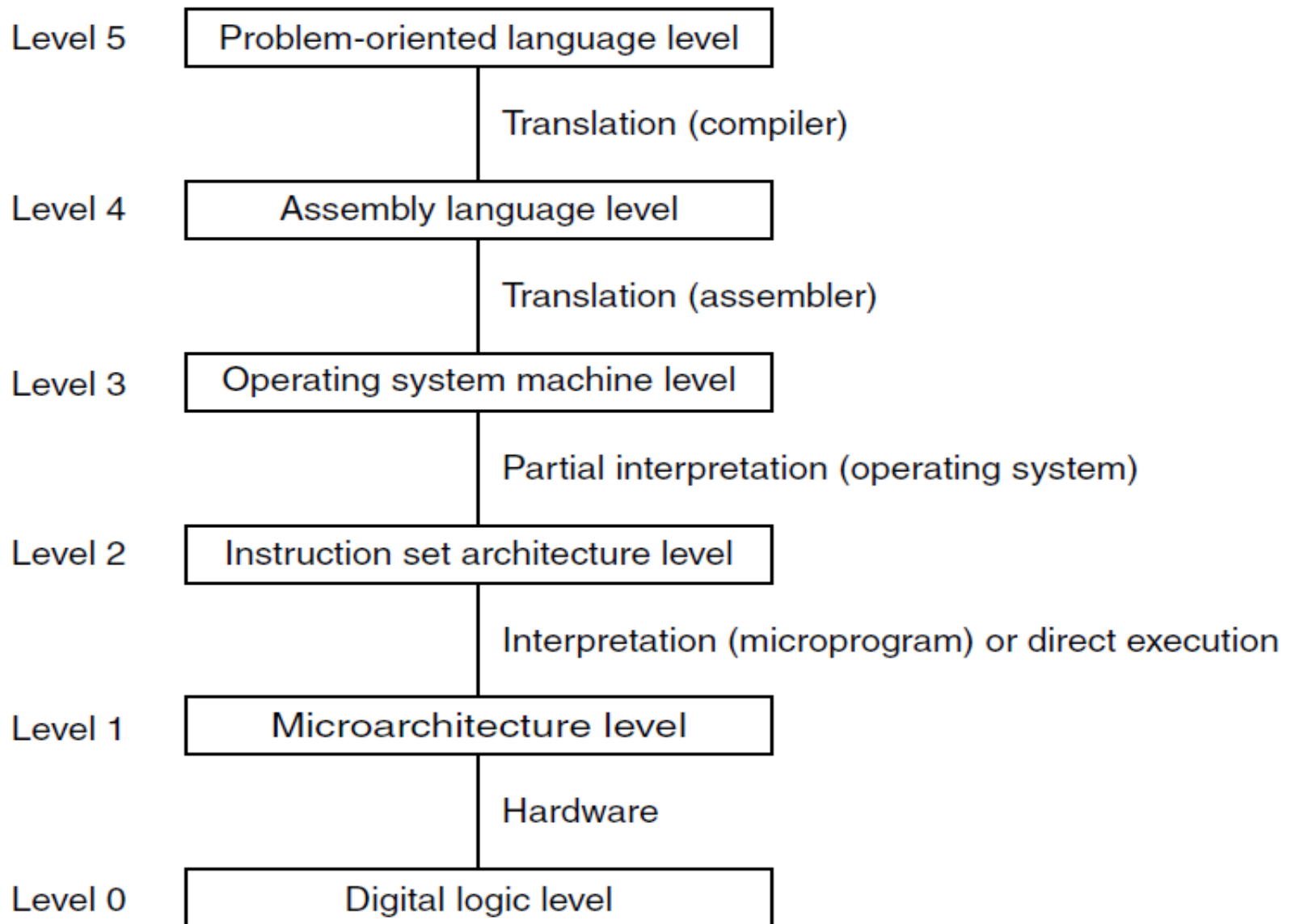
Computer Level Hierarchy





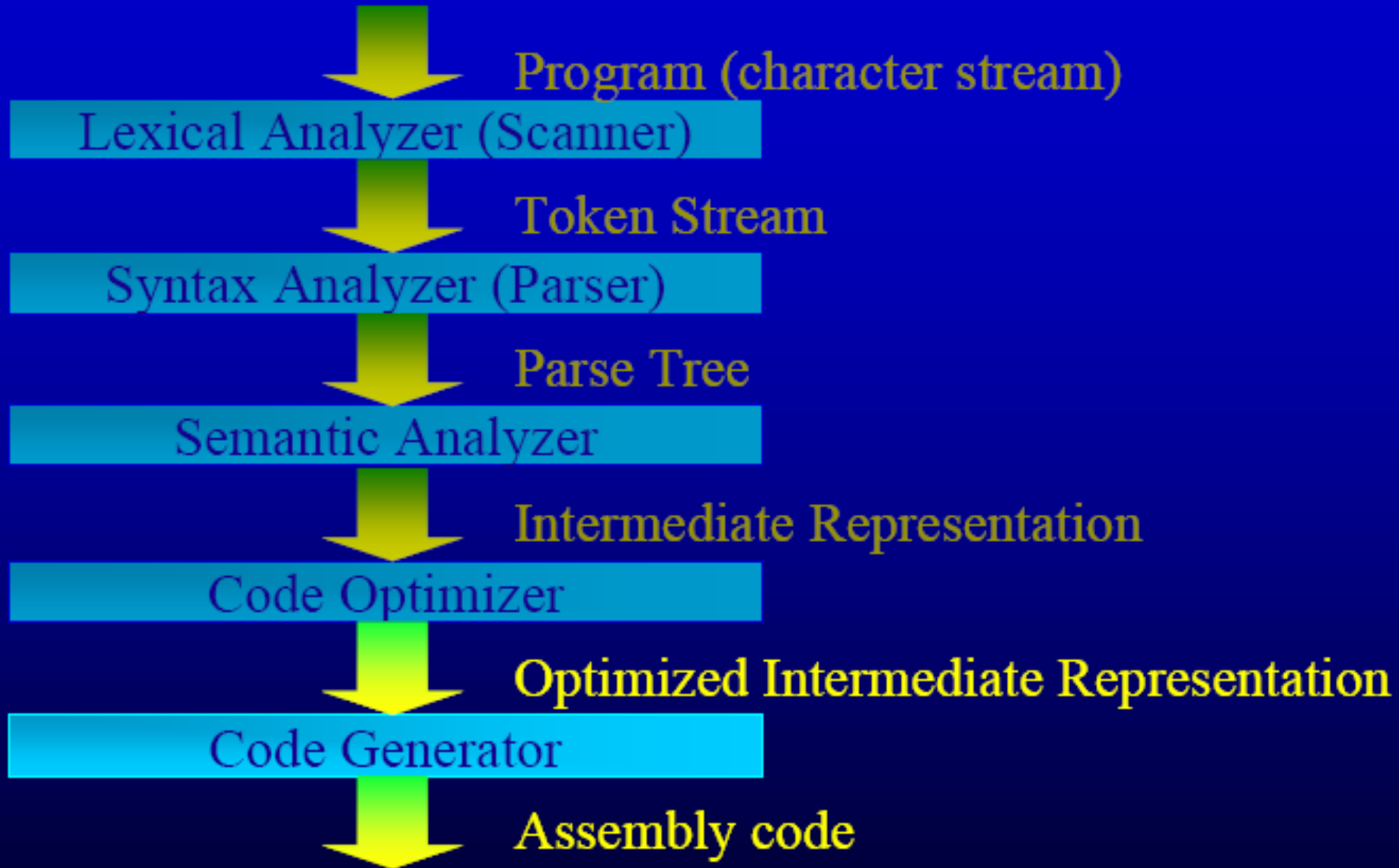
The ISA level is the interface between software and hardware.

The Computer as a multilevel machine

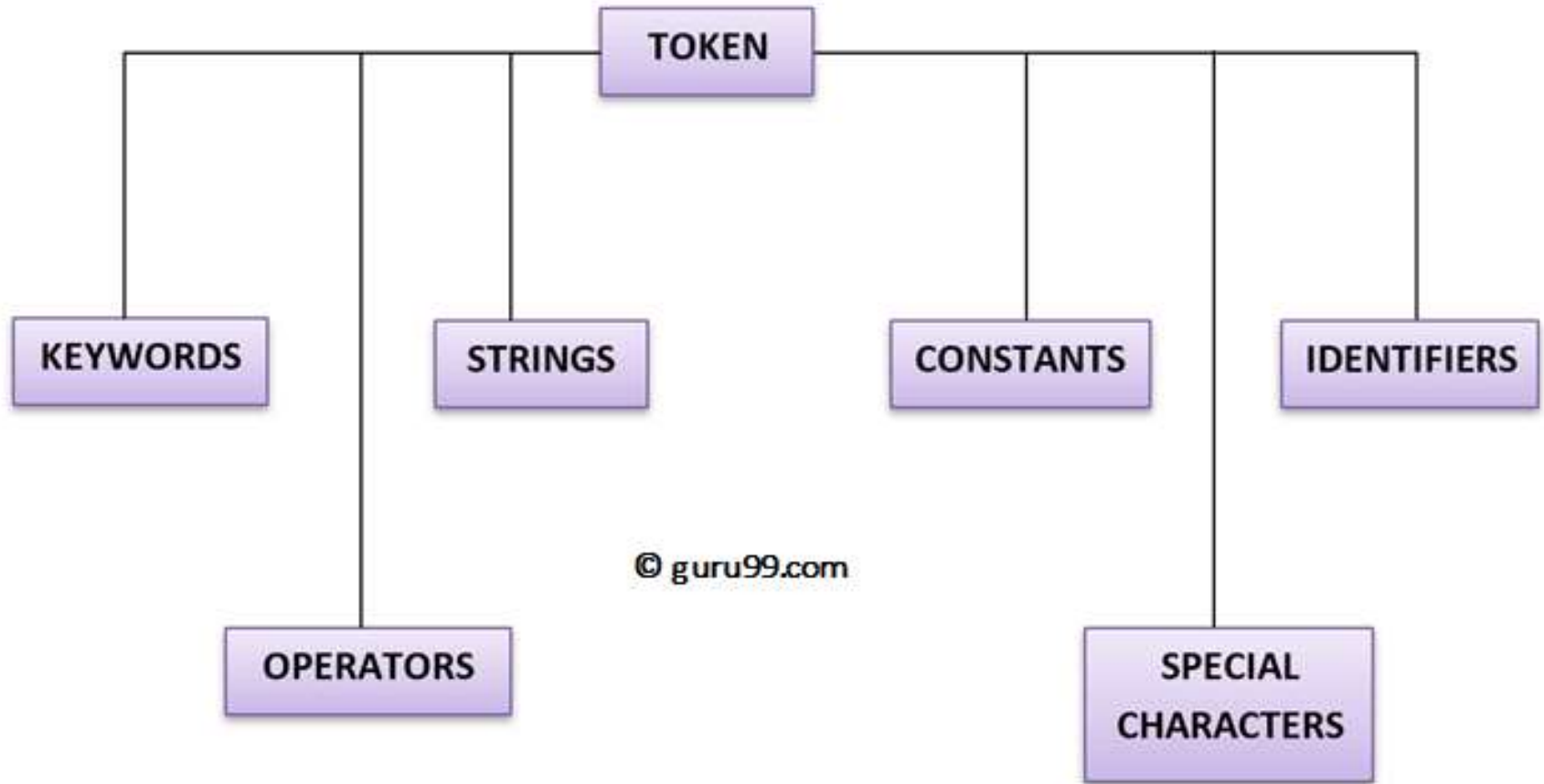


A six-level computer. The support method for each level is indicated below it (along with the name of the supporting program)

Anatomy of a Compiler



**A token is the smallest unit in a computer program.
A token is divided into six different types as follows:**

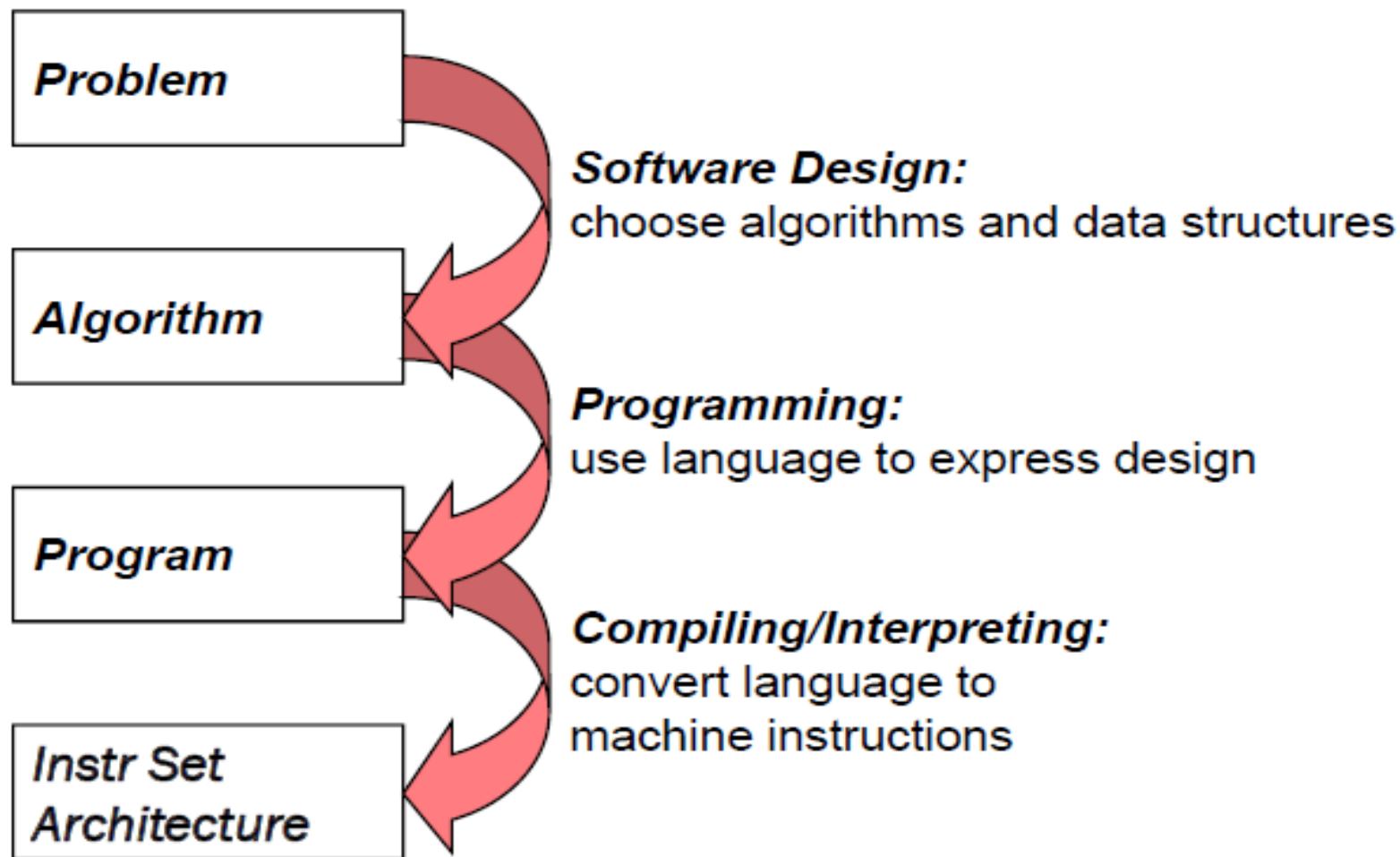


Tokens are the basic building blocks of programs—the shortest strings of characters with individual meaning.

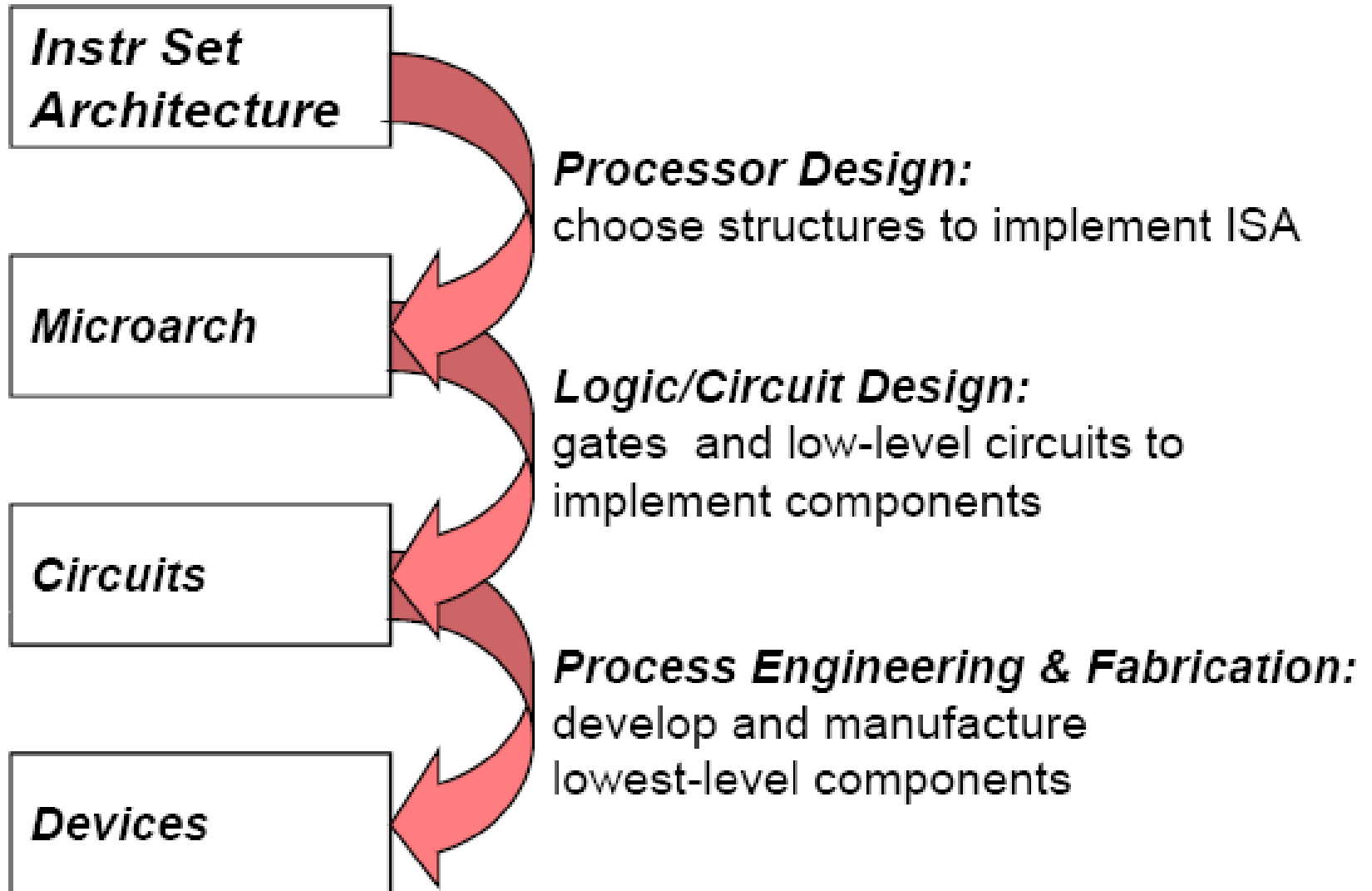
Transformations Between Layers

How do we solve a problem using a computer?

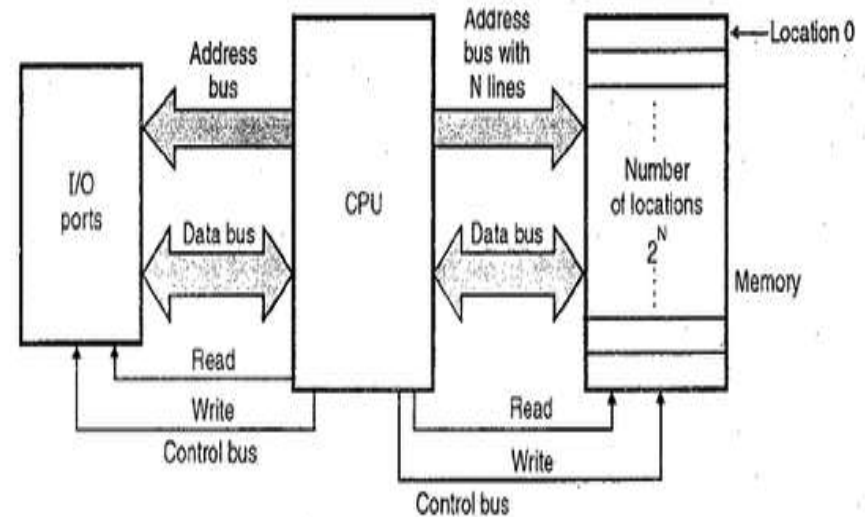
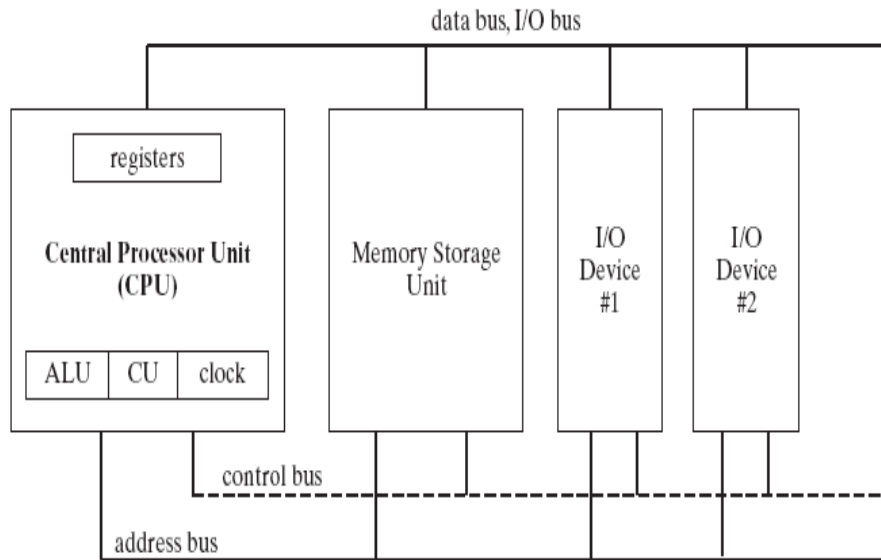
A systematic sequence of transformations between layers of abstraction.



Deeper and Deeper...



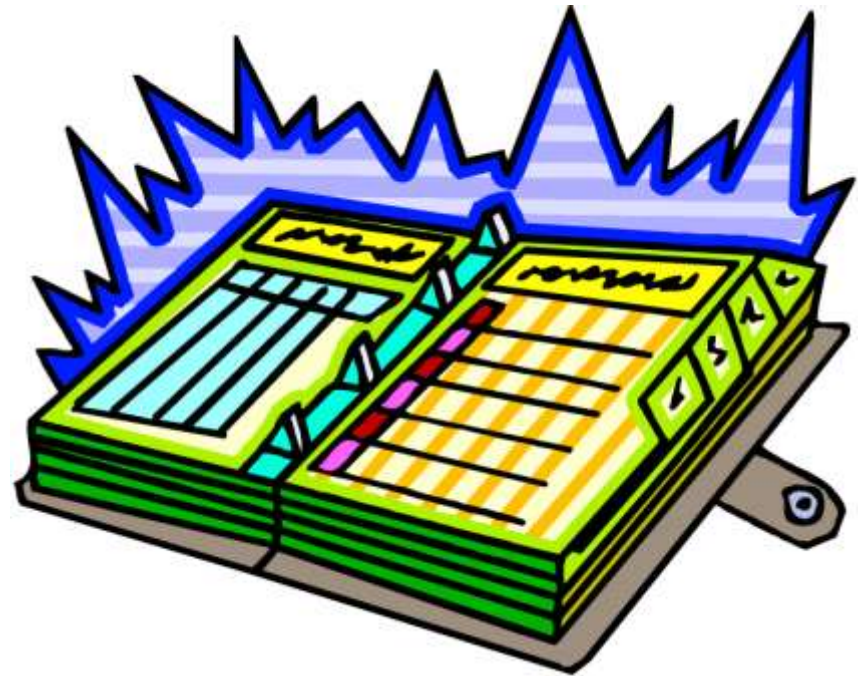
Block Diagram of a Microcomputer



- Address bus width limits the amount of memory that can be installed in the computer

Operating System - Organizer

- Keep track of executing programs
 - Give them time with the CPU
 - A program gets a slice of time with the CPU
- Keep track of memory
 - Decide when to move some data to disk (virtual memory)
- Keep track of disk space
 - Decide where to store stuff
- Interface with the user
 - Accept input via keyboard and mouse
- Keep track of devices
 - USB drives, cameras, etc
- Provides networking capabilities



The Operating System and the Kernel

kernel: The operating system kernel is the part of the operating system that responds to system calls, interrupts and exceptions.

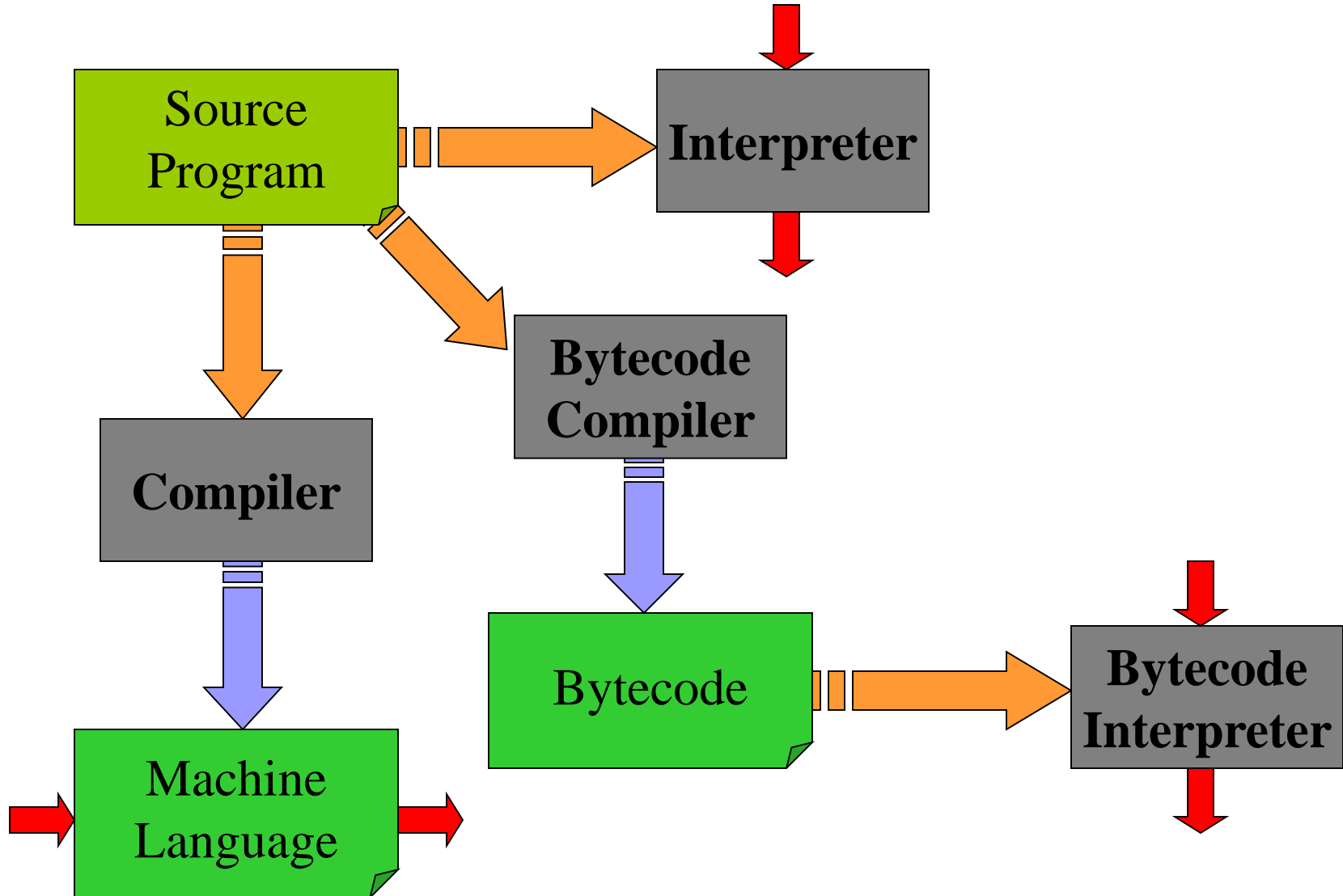
Ex. `system("cls");` from C++ source, in `<stdlib.h>`

operating system: The operating system as a whole includes the kernel, and may include other related programs that provide services for applications.

This may include things like:

- utility programs
- command interpreters
- programming libraries
- device drivers, etc.

Programs can be executed in different ways.



Classification of programming languages

Imperative

- Procedural: C, Ada, **Pascal**, **Algol**, **FORTRAN**, . . .
- Object oriented: **Scala**, C#, Java, **Smalltalk**, **SIMULA**, . . .
- Scripting: Perl, Python, PHP, javascript, . . .

Declarative

- Functional: Haskell, SML, Lisp, Scheme, . . .
- Logic: Prolog
- Dataflow: Id, Val
- Constraint-based: spreadsheets, SQL
- Template-based: XSLT

Why are there so many languages?

- Evolution.
- Special purposes.
- Personal preference.

Assignment:

Make investigations and write an essay about types and functions of operating systems, and CPU scheduling algorithms. Give more emphasis to CPU scheduling algorithms.

**Do the assignment in groups of two students.
Submit your work in printed copy.**

Loader

- To run an executable file, the loader must copy all the instructions into memory and direct the CPU to begin execution with the first instruction.
- As the program executes, it takes input data from source(s) and sends results to output devices.

Device driver

- A computer program that can **establish communication** because it contains information about the characteristics of your computer and of the device.
- Each peripheral device requires a device driver.
- Helps the computer **communicate** with that particular device.
- When we add a device to an existing computer, part of its installation includes adding its device driver to the configuration.

Virtual Memory

- Some data is not used for a long time
 - Why keep it all in memory?
- Copy a unit of data (called a “page”) to hard disk and use memory for other data
- Copy pages back from hard disk to main memory as they're needed
- Process (and its programmer) not aware that main memory is too small (the big memory illusion)
 - It asks for a main memory location (Page # , offset on page)
 - OS has to get that page into main memory if not already there