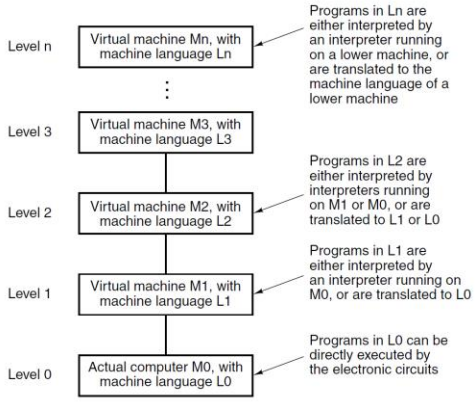
Virtual Machines

Rather than thinking in terms of translation or interpretation, it is often simpler to imagine the existence of a hypothetical computer or **virtual machine** whose machine language is L1. Let us call this virtual machine M1 (and let us call the machine corresponding to L0, M0).

For practical reasons, the languages L0 and L1 must not be "too" different. Often means that L1, will still be far from ideal for most applications

The obvious approach is to invent still another set of instructions that is more people-oriented and less machine-oriented than L1. This third set also forms a language, which we will call L2 (and with virtual machine M2). People can write programs in L2 just as though a virtual machine with L2 as its machine language really existed. Such programs can be either translated to L1 or executed by an interpreter written in L1.

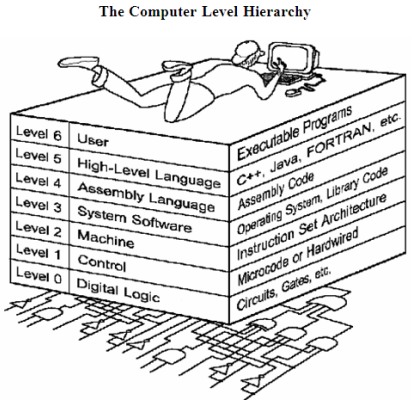


The invention of a whole series of languages, each one more convenient than its predecessors, can go on indefinitely until a suitable one is finally achieved. Each language uses its predecessor as a basis, so we may view a computer using this technique as a series of layers or levels, one on top of another.

23

Virtual Machines

- The machine defined by a certain language may be enormously complicated and expensive to construct directly out of electronic circuits but we can imagine it nevertheless.
- A machine with C or C++ or Java as its machine language would be complex indeed but could be built using today's technology.
- There is a good reason, however, for not building such a computer: it would not be cost effective compared to other techniques.
- Merely being doable is not good enough: a practical design must be cost effective as well.



The terms "level" and "virtual machine" are used in this context interchangeably.

24

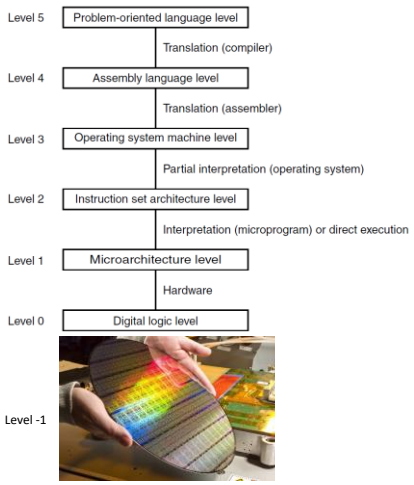
Virtual Machines

- A person who writes programs for the level *n* virtual machine need not be
- aware of the underlying interpreters and translators.
- The machine structure ensures that these programs will somehow be executed. It is of no real interest whether they are carried out step by step by an interpreter which, in turn, is also carried out by another interpreter, or whether they are carried out by the electronic circuits directly. The same result appears in both cases: the programs are executed.
- Most programmers using an *n*-level machine are interested only in the top level, the one least resembling the machine language at the very bottom.
- However, people interested in understanding how a computer really works must study all the levels.
- People who design new computers or new levels must also be familiar with levels other than the top one. The concepts and techniques of constructing machines as a series of levels and the details of the levels themselves is the subject of Computer Organization and Computer Architecture.

25

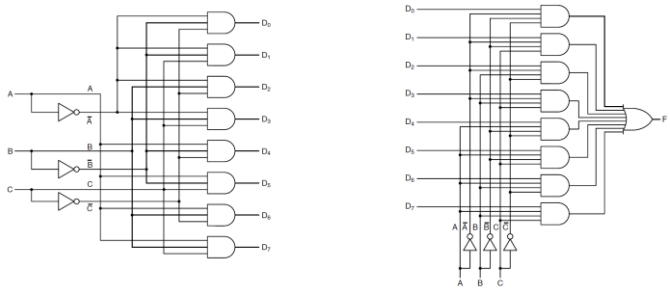
Virtual Machines

- Most modern computers consist of two or more levels.
- Level 0, the **Digital Logic Level**, consists of a number of hardware components, known as **Logic Gates**.



26

Level 0: The Digital Logic Level



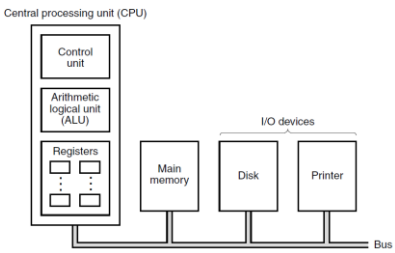
- Built from analog components, such as transistors, gates actas digital devices.
- Each gate has one or more digital inputs (signals representing 0 or 1) and computes as output some simple function of these inputs, such as AND or OR.
- A small number of gates can be combined to form a 1-bit memory, which can store a 0 or a 1.
- The 1-bit memories can be combined in groups of (for example) 16, 32, or 64 to form registers.
- Each **register** can hold a single binary number up to some maximum value.
- Gates are also be combined to form the main computing engine itself.

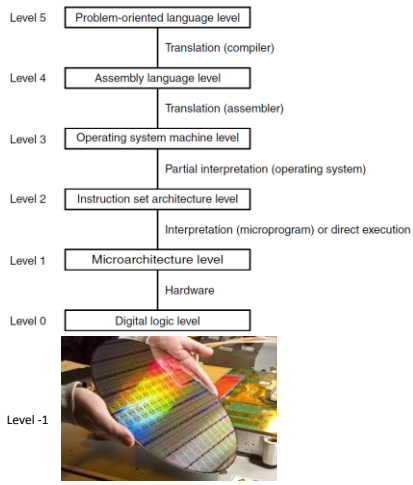
This level is explored in CS1110

27

Virtual Machines

- Level 1, the **microarchitecture level,** a collection of (typically) 8 to 32 **registers** that form a local memory, a **Control Unit** and a circuit called an **ALU** (**Arithmetic Logic Unit**), which is capable of performing simple arithmetic operations. The registers are connected to the ALU to form a **data path**, over which the data flow.

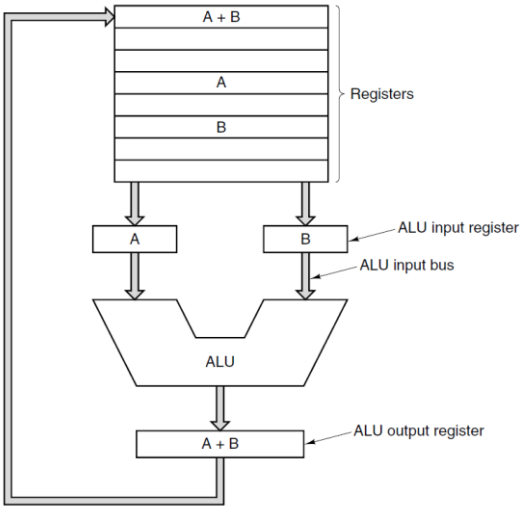




28

Level 1: The Microarchitecture Level

- The basic operation of the data path consists of selecting one or two registers, having the ALU operate on them (for example, adding them together), and storing the result back in some register.
- On some machines the operation of the data path is controlled by a program called a **microprogram**. On other machines the data path is controlled directly by hardware.
  - The microprogram is an interpreter for the instructions at Level 2.

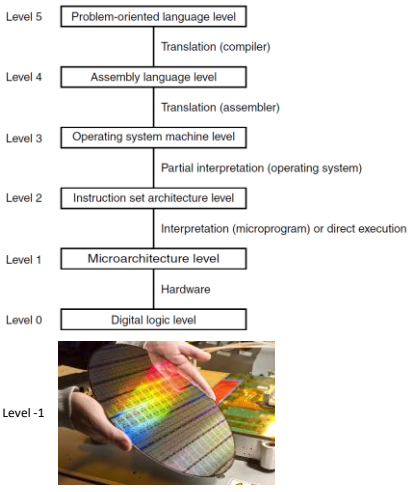


This level is explored in CS1110

29

Virtual Machines

- Level 2, the **Instruction Set Architecture (ISA) Level**, consists of a number of Instructions (defined in the hardware's manufacturer "Machine Reference Manual").
- These are the instructions carried out interpretively by the microprogram or hardware execution circuits.



For example, the programmer would write

add A,B

and the assembler would translate this notation into

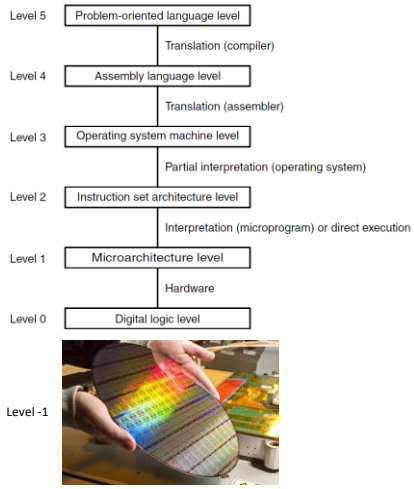
1000110010100000

This instruction tells the computer to add the two numbers A and B. Th e name coined for this symbolic language, still used today, is **assembly language**. In contrast, the binary language that the machine understands is the **machine language**.

30

Virtual Machines

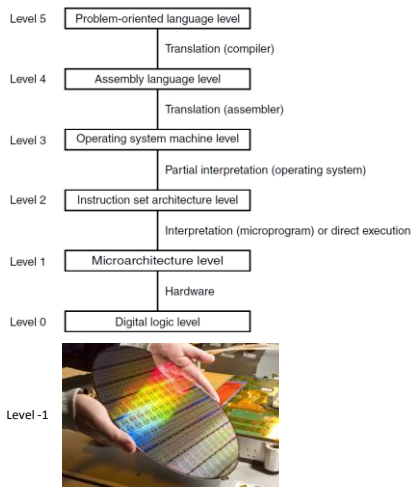
- Level 3, the **Operating System Machine Level**, is a hybrid level.
  - Most of the instructions in its language are also in the ISA level.
  - In addition, there is a set of new instructions, a different memory organization, the ability to run two or more programs concurrently, and various other features.
- The new facilities added at Level 3 are carried out by an interpreter running at Level 2, which, historically, has been called an **operating system**.
- There is a fundamental break between Levels 3 and 4.
  - The lowest three levels are not designed for use by application programmers.
  - They are intended primarily for running the interpreters and translators needed to support the higher levels. These interpreters and translators are written by **systems programmers** who specialize in designing and implementing new virtual machines.
  - Levels 4 and above are intended for the applications programmer.



31

Virtual Machines

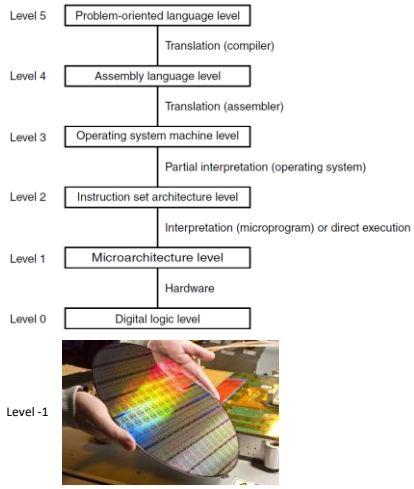
- Level 4, the Assembly Language Level, is a symbolic form for one of the underlying languages.
- This level provides a method for people to write programs for Levels 1, 2, and 3 in a more human-friendly form than languages at those levels.
- Programs in assembly language are first translated to Level 1, 2, or 3 language and then interpreted by the appropriate virtual or actual machine.
- The program that performs the translation is called an **assembler**.
- The machine languages of Levels 1, 2, and 3 are numeric.
  - Programs in them consist of long series of numbers, which are fine for machines but bad for people.
- Starting at Level 4, the languages contain words and abbreviations meaningful to people.
  - add al, bl
  - jmp loop
  - out 04
  - cmp cl, FC



32

Virtual Machines

- Level 5, the problem-oriented language level, usually consists of languages designed to be used by applications programmers.
- Such languages are often called **high-level languages**. Literally hundreds exist. A few of the better-known ones are C, C++, Java, Perl, Python, and PHP.
- Programs written in these languages are generally translated to Level 3 or Level 4 by translators known as **compilers.**
  - although occasionally they are interpreted instead. Programs in Java, for example, are usually first translated to a an ISA-like language called Java byte code, which is then interpreted.
- In some cases, Level 5 consists of an interpreter for a specific application domain, such as symbolic mathematics. It provides data and operations for solving problems in this domain in terms that people knowledgeable in the domain can understand easily.



33

In Summary

- Computers are designed as a series of levels, each one built on its predecessors. Each level represents a distinct abstraction, with different objects and operations present.
- The set of data types, operations, and features of each level is called its **architecture**.
- The architecture deals with those aspects that are visible to the user of that level. Features that the programmer sees, such as how much memory is available, are part of the architecture.
  - Implementation aspects, such as what kind of technology is used to implement the memory, are not part of the architecture.
- The study of how to design those parts of a computer system that are visible to the programmers is called **computer architecture**.
- In common terminology computer architecture and computer organization mean essentially the same thing.

34