Module 1

Introduction

Content

- Levels in Architecture
- Virtual Machine
- Evolution of Multi-Level Machines

Computer Architecture & Computer Organization

- Attributes of a system visible to the programmer
- Have a direct impact on the logical execution of a program

Computer Architecture

Architectural attributes include:

- Instruction set, number of bits used to represent various data types, I/O mechanisms, techniques for addressing memory

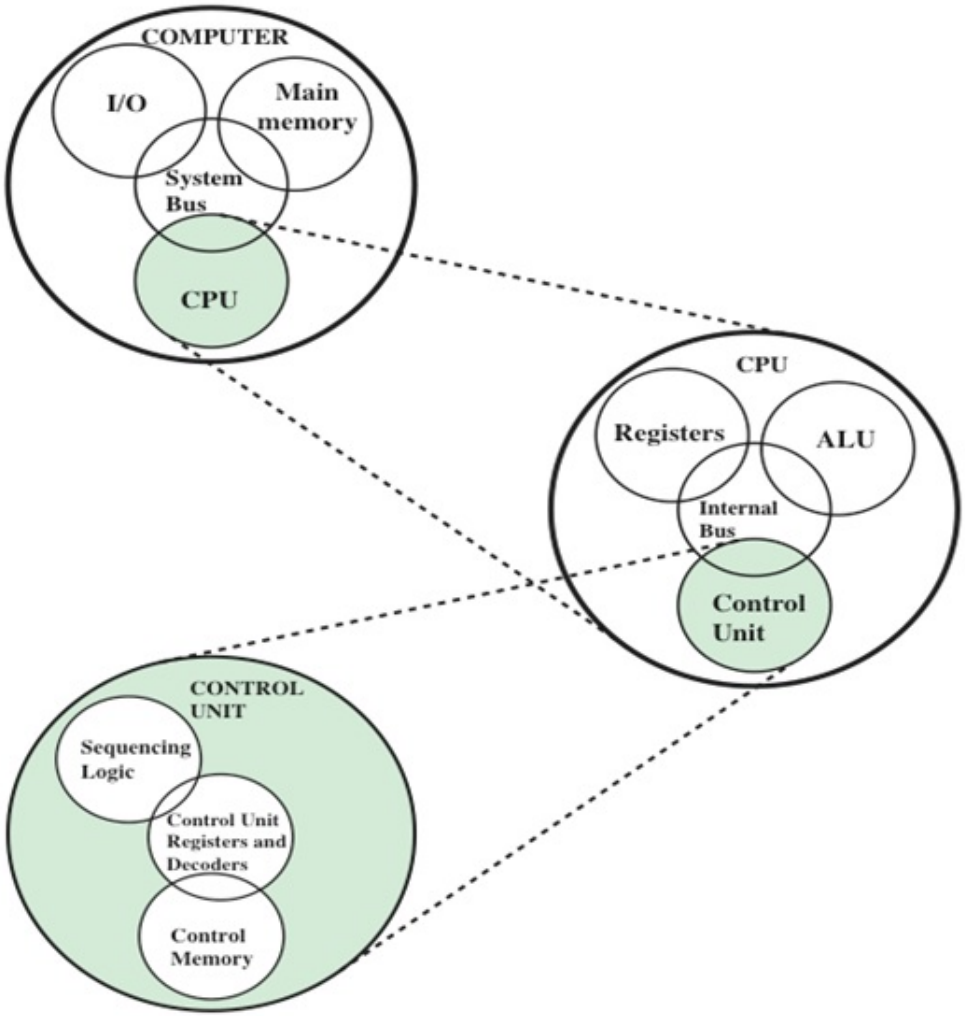
Organizational attributes include:

- Hardware details transparent to the programmer, control signals, interfaces between the computer and peripherals, memory technology used

Computer Organization

- The operational units and their interconnections that realize the architectural specifications

Structure



A Top-Down view of a Computer [1]

The Digital Computer

- Machine to carry out instructions
  - i.e., A program
- Instructions are simple:
  - Add numbers
  - Check if a number is zero
  - Copy data between memory locations
- Primitive instructions in machine language

Structured Computer Organization

- There is a large gap between what is convenient for people and what is convenient for computers
- People want to do X, but computers can only do Y
  - This leads to a problem
- The goal of this course is to explain how this problem can be solved

Levels in architecture & Virtual Machine [2]

- The problem can be resolved in 2 ways, both involve designing a new set of instructions(represented by a Language '$\mathbf{L_1}$') that is more convenient for people to use than the set of built-in machine instructions(Language L_0).
  - The programs written in L_1 must be executed by the computer, which can only execute programs written in its machine language, L_0

- The First Method of executing a program written in L_1 is to replace each instruction in it by an equivalent sequence of instructions in L_0

Method 1 … Contd.

- The resulting program consists entirely of L_0 instructions
- The computer then executes the new L_0 program instead of the old L_1 program
- This technique is called Translation
- The Second Method is to write a program in L_0 that takes programs in L_1 as input data and carries them out by examining each instruction in turn and executing the equivalent sequence of L_0 instructions directly.
  - This does not require first generating a new program in L_0
- This is called Interpretation
  - Program that carries it out is called an interpreter.

Translation and Interpretation

- Translation and interpretation are similar
  - In both methods, the computer carries out instructions in L_1 by executing equivalent sequences of instructions in L_0.

- The difference between two methods:
- In translation, the entire L_1 program is first converted to an L_0 program, the L_1 program is thrown away, and then the new L_0 program is loaded into the computer's memory and executed
  - During execution, the newly generated L_0 program is running and in control of the computer.

- In interpretation, after each L_1 instruction is examined and decoded, it is carried out immediately. No translated program is generated.
  - Here, the interpreter is in control of the computer. To it, the L1 program is just data.

- Both methods, or a combination of the two, are widely used.

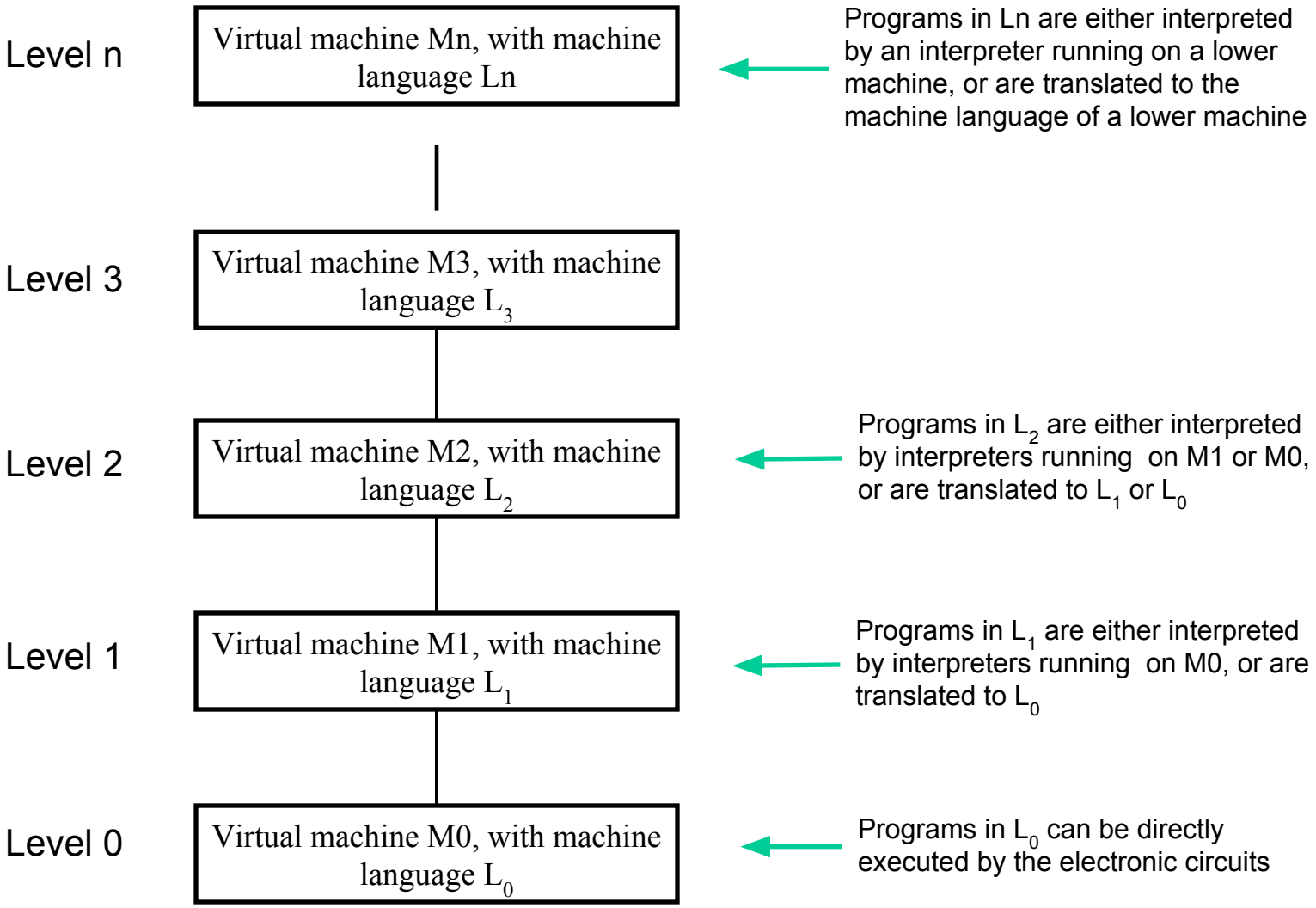
Virtual Machine

- 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 L_1
- Let us call this virtual machine M1
  - and let us call the machine corresponding to L_0 as M0
- To make translation or interpretation more practical, the obvious approach is to invent still another set of instructions that is more people-oriented and less machine-oriented than L_1.
- 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.

Virtual Machine

- 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.
- The bottommost language or level is the simplest and the topmost language or level is the most sophisticated.
- Each Virtual machine has a machine language, consisting of all the instructions that the machine can execute.

A Multi-Level Machine [2]



Contemporary Multilevel Machines

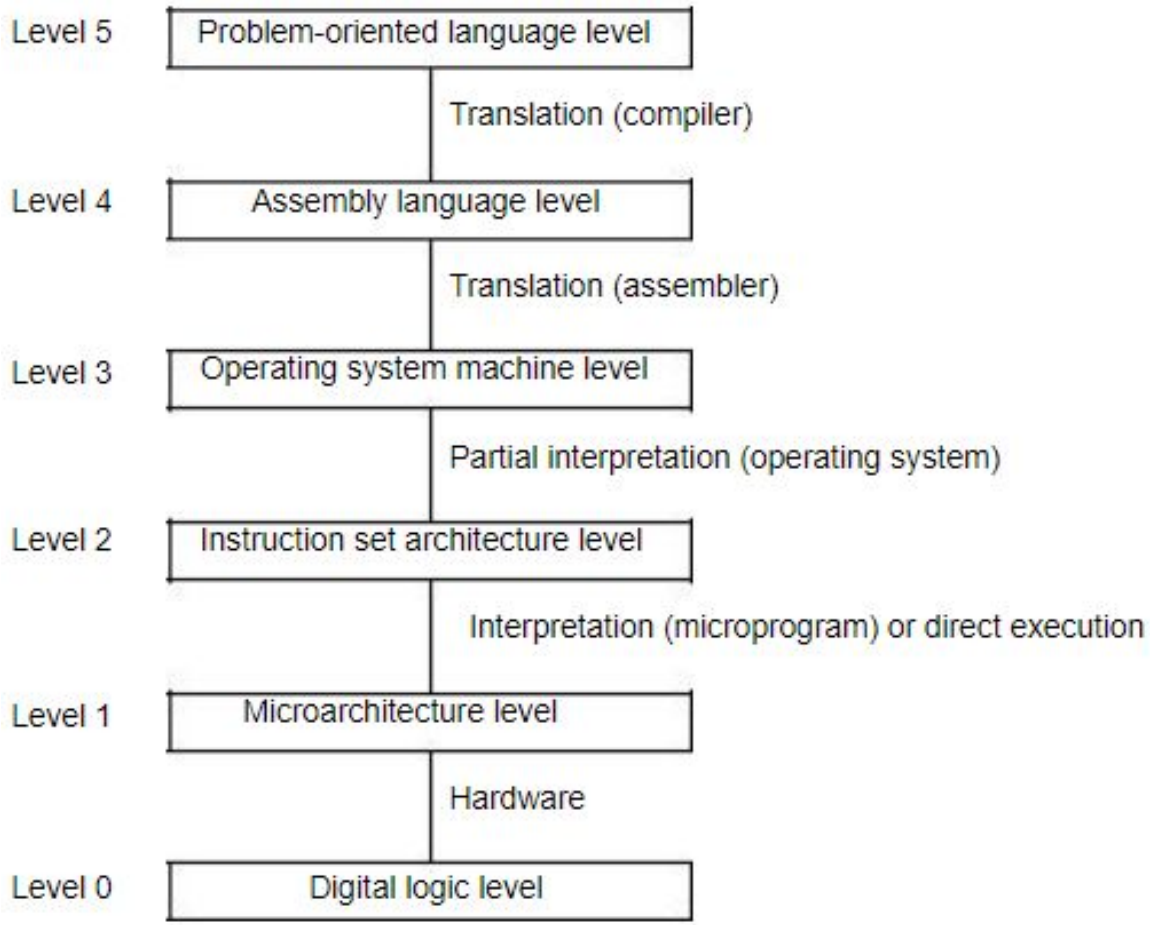


Figure: A six-level computer. The support method for each level is indicated below it (along with the name of the supporting program).

Evolution of Multilevel Machines [2]

- Invention of microprogramming
- Invention of operating system
- Migration of functionality to microcode
- Elimination of microprogramming

Evolution of Multilevel Machines [2]

Invention of microprogramming:

- The first digital computers(in 1940s) had only two levels: the **ISA level**, in which all the programming was done, and the **digital logic** level, which executed these programs. The digital logic level's circuits were complicated, difficult to understand and build, and unreliable.

- In 1951, a three-level computer came into existence, in order to drastically simplify the hardware and thus reduce the number of (unreliable) vacuum tubes needed (**Wilkes, 1951**). This machine was to have a built-in, unchangeable interpreter (the microprogram), whose function was to execute ISA-level programs by interpretation.

- Because the hardware would now only have to execute microprograms, which have a limited instruction set, instead of ISA-level programs, which have a much larger instruction set, fewer electronic circuits would be needed. Because electronic circuits were then made from vacuum tubes, such a simplification promised to reduce tube count and hence enhance reliability (i.e., the number of crashes per day).

Evolution of Multilevel Machines [2]

Invention of operating system:

- In early years, most computers were ‘‘open shop,’’ which meant that the programmer had to operate the machine personally, with a deck of punched cards (an early input medium).
- Around 1960 people tried to reduce the amount of wasted time by automating the operator’s job.
- A program called an **operating system** was kept in the computer at all times. In subsequent years, operating systems became more and more sophisticated.
- New instructions, facilities, and features were added to the ISA level until it began to take on the appearance of a new level. Some of this new level’s instructions were identical to the ISA-level instructions, but others, particularly input/output instructions, were completely different. The new instructions were often known as **operating system macros** or **supervisor calls**. The usual term now used is **system call**.

Evolution …Contd.

Migration of functionality to microcode:

Once microprogramming had become common (by 1970), designers realized that they could add new instructions by just extending the microprogram. This revelation led to a virtual explosion in machine instruction sets, such as:

- Instructions for integer multiplication and division,
- Floating-point arithmetic instructions,
- Instructions for calling and returning from procedures etc.
- Furthermore, once machine designers saw how easy it was to add new instructions, they began looking around for other features to add to their microprograms. A few examples of these additions included:
- Features to speed up computations involving arrays (indexing and indirect addressing),
- Features to permit programs to be moved in memory after they have started running (relocation facilities),
- I/O interrupt signals, process switching, instructions for processing audio, image, and multimedia files etc. Numerous other features and facilities have been added over the years, usually for speeding up some particular activity.

.

Evolution …Contd.

Elimination of microprogramming:

- Microprograms grew enormously during the golden years of microprogramming (1960s and 1970s). They also tended to get slower and slower as they acquired more bulk.

- Finally, some researchers realized that by eliminating the microprogram, vastly reducing the instruction set, and having the remaining instructions be directly executed (i.e., **hardware control** of the ALU), machines could be speeded up. In a certain sense, computer design had come full circle, back to the way it was before Wilkes invented microprogramming in the first place.But the wheel is still turning. Modern processors still rely on microprogramming to translate complex instructions to internal microcode that can be executed directly on streamlined hardware.

References

- [1]. Computer Organization and Architecture”, 10/e, by William Stallings, Pearson Education
- [2]. Tanenbaum & Austin, Structured Computer Organization, 6th Edition, Pearson Education

Next Topic

- Performance Evaluation