Imperial College London

Computer Architecture: Year 2

0: Computer abstraction

Xin Wang

September 26, 2020

Contents

1 Introduction **2**
1.1 Eight great ideas of computer architecture 2

2 Computer structure organisation **3**

1 Introduction

Programmers are always concerned about the performance of programs because getting results to the user quickly is critical. In the past where primary constraint on computer performance was the size of the computer's memory, minimising the memory space was the way to make programs fast. In the present, advances in computer design and memory technology have reduced the importance of memory.

Programmers now need to understand the issues such as the parallel nature of processors, the hierarchical nature of memories and the energy efficiency of their programs running locally or in the Cloud.

The questions that are of concern to be covered are:

- Understanding the gap between hardware and software i.e. how programs written in a high-level language such as C or Java are translated into the hardware language and how does the hardware execute the resulting program.
- Bridging the gap between hardware and software i.e. the interface between the software and the hardware, how software instruct the hardware to perform needed functions.
- Factors determining the performance of a program and techniques used to improve it.
- The reasons for and the consequences of the recent switch from sequential processing to parallel processing.

1.1 Eight great ideas of computer architecture

1. **Design for Moore's Law**: The one constant for computer designers is rapid change, driven largely by Moore's Law stating that integrated circuit resources double every 18–24 months.
2. **Abstraction to simplify design**: A productivity technique for hardware and software by using abstractions to represent the design at different levels of representation e.g. lower-level details are hidden to offer a simpler model at higher levels.
3. **Making the common case fast**: Making the common case fast will tend to enhance performance better than optimising the rare case.
4. **Performance via parallelism**: Designs that get more performance by performing operations in parallel.
5. **Performance via pipelining**: A particular pattern of parallelism prevalent in computer architecture.
6. **Performance via prediction**: Sometimes it is faster on average to guess

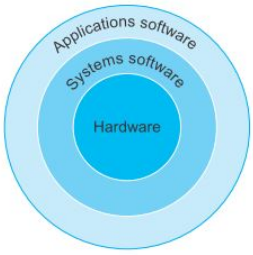
an action and start processing assuming that the mechanism to recover from a mis-prediction is not too expensive and the prediction is relatively accurate.

7. **Hierarchy of memories**: The goal is for memory to be fast, large, and cheap since as memory speed shapes performance, capacity limits size of problems, and cost of memory determines the cost of computers for the consumer. Commonly addressed with a hierarchy of memories, with the fastest, smallest, and most expensive memory per bit and the slowest, largest, and cheapest per bit.

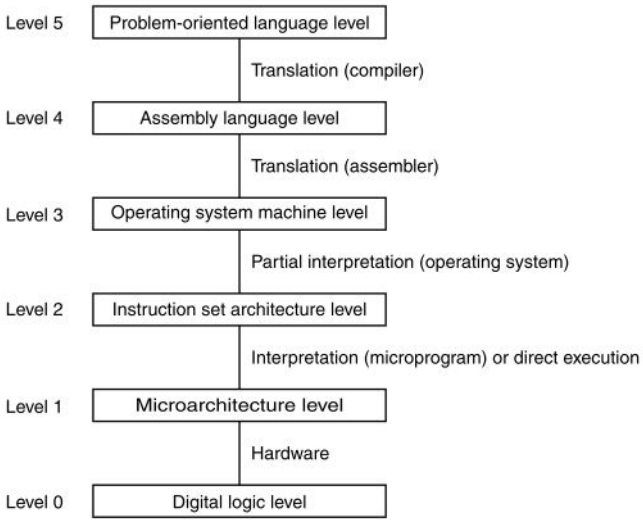
8. **Dependability via redundancy**: Computers need to be fast and dependable. Systems are made dependable by including redundant components that can take over when a failure occurs and to help detect failures.

2 Computer structure organisation

The computer structure can be abstracted into three levels: Applications, Operating system and Hardware.



The modern multilevel view spilts it into six layers:



- **Level 5**: Consists of languages designed to be used by applications programmers to solve problem e.g. Python and C++. Programs written are generally translated to level 4 by **compilers**, although occasionally they are interpreted instead.

  > **Compiler**: A program that translates high-level language statements into assembly language statements.

- **Level 4**: Intended primarily for running the interpreters and assemblers needed to support the higher levels.

  > **Assembler**: A program that translates a symbolic version of instructions into the binary version.

- **Level 3**: The operating system level.

- **Level 2**: The Instruction Set Architecture level. Every computer manufacturer publishes a manual describing the instructions carried out interpretively by the microprogram or hardware execution circuits.

- **Level 1**: At this level a collection of 8 to 32 registers form local memory and the ALU which is capable of performing simple arithmetic operations. The registers are connected to the ALU to form a data path.

- **Level 0**: Logic gates built from transistors to form a register.