# 1.2 Eight great ideas in computer architecture

We now introduce eight great ideas that computer architects have invented in the last 60 years of computer design. These ideas are so powerful they have lasted long after the first computer that used them, with newer architects demonstrating their admiration by imitating their predecessors. These great ideas are themes that we will weave through this and subsequent chapters as examples arise. To point out their influence, in this section we introduce icons and highlighted terms that represent the great ideas and we use them to identify the nearly 100 sections of the book that feature use of the great ideas.

### Design for Moore's Law


The one constant for computer designers is rapid change, which is driven largely by Moore's Law. ***Moore's Law*** states that integrated circuit resources double every 18-24 months. Moore's Law resulted from a 1965 prediction of such growth in IC capacity made by Gordon Moore, one of the founders of Intel. As computer designs can take years, the resources available per chip can easily double or quadruple between the start and finish of the project. Like a skeet shooter, computer architects must anticipate where the technology will be when the design finishes rather than design for where it starts. We use an "up and to the right" Moore's Law graph to represent designing for rapid change.

### Use abstraction to simplify design


Both computer architects and programmers had to invent techniques to make themselves more productive, for otherwise design time would lengthen as dramatically as resources grew by Moore's Law. A major productivity technique for hardware and software is to use ***abstractions*** to characterize the design at different levels of representation; lower-level details are hidden to offer a simpler model at higher levels. We'll use the abstract painting icon to represent this second great idea.

### Make the common case fast


Making the ***common case*** fast will tend to enhance performance better than optimizing the rare case. Ironically, the common case is often simpler than the rare case and hence is usually easier to enhance. This common sense advice implies that you know what the common case is, which is only possible with careful experimentation and measurement (see COD Section 1.6 (Performance)). We use a sports car as the icon for making the common case fast, as the most common trip has one or two passengers, and it's surely easier to make a fast sports car than a fast minivan!

### Performance via parallelism


Since the dawn of computing, computer architects have offered designs that get more performance by computing operations in ***parallel***. We'll see many examples of parallelism in this book. We use multiple jet engines of a plane as our icon for **parallel performance**.

### Performance via pipelining


A particular pattern of parallelism is so prevalent in computer architecture that it merits its own name: ***pipelining***, which moves multiple operations through hardware units that each do a piece of an operation, akin to water flowing through a pipeline. For example, before fire engines, a "bucket brigade" would respond to a fire, which many cowboy movies show in response to a dastardly act by the villain. The townsfolk form a human chain to carry a water source to fire, as they could much more quickly move buckets up the chain instead of individuals running back and forth. Our pipeline icon is a sequence of pipes, with each section representing one stage of the pipeline.

### Performance via prediction


Following the saying that it can be better to ask for forgiveness than to ask for permission, the next great idea is prediction. The idea of ***prediction*** is that, in some cases it can be faster on average to guess and start working rather than wait until you know for sure, assuming that the mechanism to recover from a misprediction is not too expensive and your prediction is relatively accurate. We use the fortune-teller's crystal ball as our prediction icon.

### Hierarchy of memories


Programmers want the memory to be fast, large, and cheap, as memory speed often shapes performance, capacity limits the size of problems that can be solved, and the cost of memory today is often the majority of computer cost. Architects have found that they can address conflicting demands of fast, large, and cheap memory with a ***hierarchy of memories***, with the fastest, smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom. As we shall see in COD Chapter 5 (Large and Fast: Exploiting Memory Hierarchy), caches give the programmer the illusion that main memory is almost as fast as the top of the hierarchy and nearly as big and cheap as the bottom of the hierarchy. We use a layered triangle icon to represent the memory hierarchy. The shape indicates speed, cost, and size: the closer to the top, the faster and more expensive per bit the memory; the wider the base of the layer, the bigger the memory.

### Dependability via redundancy


Computers not only need to be fast; they need to be dependable. Since any physical device can fail, we make systems ***dependable*** by including redundant components that can take over when a failure occurs *and* to help detect failures. We use the tractor-trailer as our icon, since the dual tires on each side of its rear axles allow the truck to continue driving even when one tire fails. (Presumably, the truck driver heads immediately to a repair facility so the flat tire can be fixed, thereby restoring redundancy!)

**PARTICIPATION ACTIVITY** 1.2.1: Eight great ideas in computer architecture.

Match the situation with the closest analog of a great idea in computer architecture.

Hierarchy of Memories | Use Abstraction to Simplify Design

Make the Common Case Fast  Design for Moore's Law

A soccer player runs not to where the ball is, but to where the ball will be.

A customer talks to a phone agent. If there's a problem, he talks to the agent's supervisor.

A house architect first designs a house with 5 rooms, then designs room details like closets, windows, and flooring.

A college student rents an apartment closer to campus than to her favorite weekend beach spot.

Reset

**PARTICIPATION ACTIVITY** 1.2.2: Eight great ideas in computer architecture (continued).

Match the situation with the closest analog of a great idea in computer architecture.

Dependability via Redundancy  Performance via Parallelism

Performance via Pipelining  Performance via Prediction

A sister is hanging clothes to dry. Her brother helps by hanging clothes simultaneously.

A brother is washing and drying dishes. His sister helps by drying each dish immediately after the brother washes each.

A mom expects her son will be hungry after a long airplane flight, so she cooks dinner just in case. If he's not hungry, she'll whip up a dessert instead.

A drummer's stick breaks, but he quickly grabs another one and continues playing the song.

Reset

Provide feedback on this section