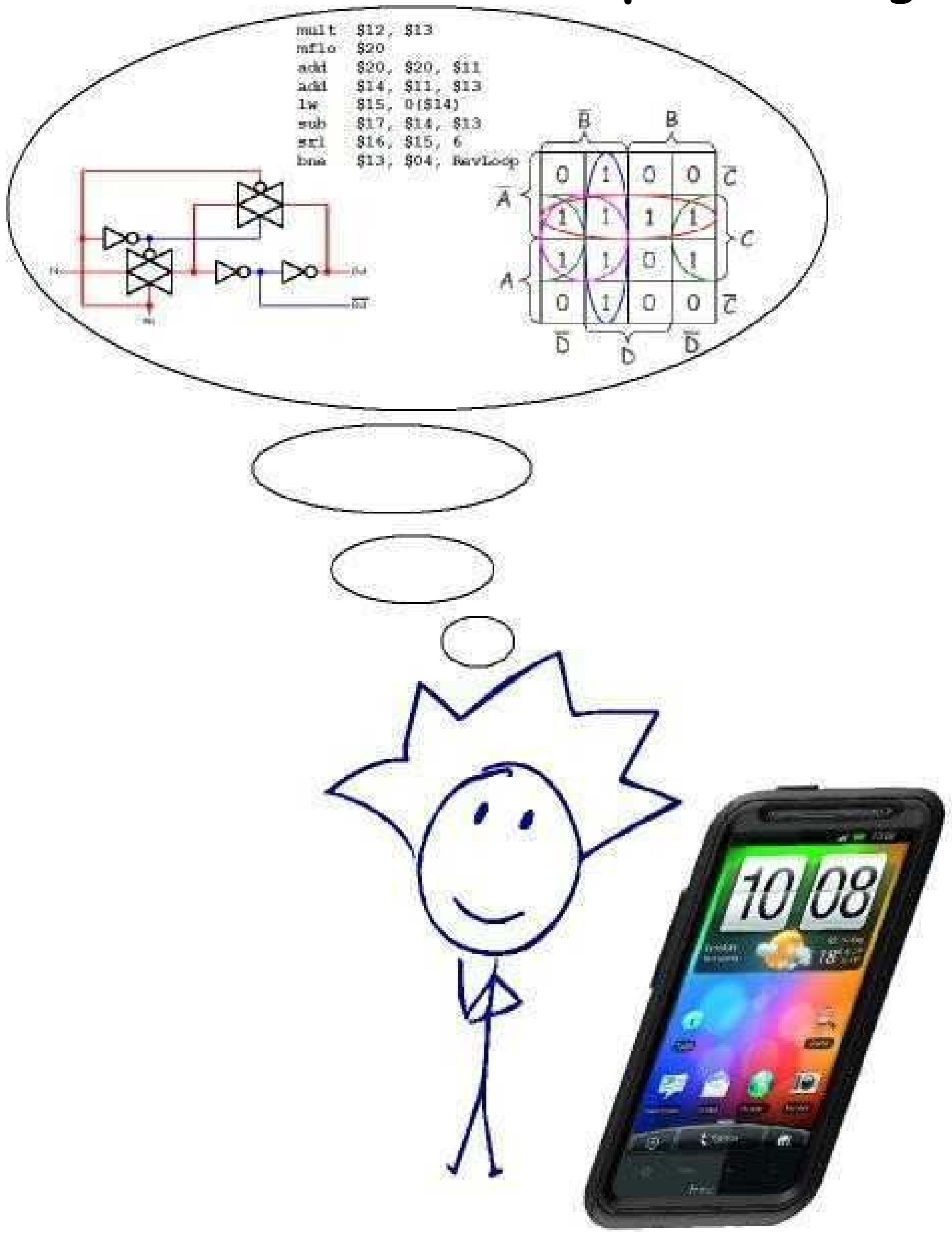
Designing Computer Systems

Introduction to Computer Design



```
mult  $12, $13
mflo  $20
add   $20, $20, $11
add   $14, $11, $13
lw    $15, 0($14)
sub   $17, $14, $13
srl   $16, $15, 6
bne   $13, $04, RevLoop
```

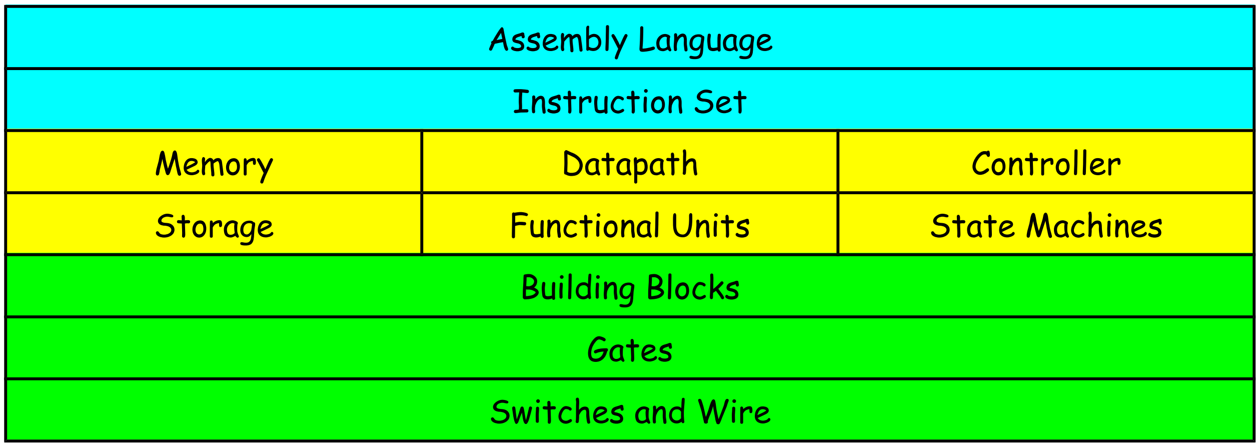
Introduction to Computer Design

Our world is full of computer-based systems: smartphones, laptops, tablets. And new, useful and attractive products are appearing regularly. How are they designed? What are they made of? Today's high tech products contain dozens of subsystems, each composed of many components. Some are specialized, like a color display or a wireless transceiver. But the *computing* in a computing system is general and programmable. Interestingly, while computing provides the system "smarts", it is built using two very simple components: switches and wire.

So how does one create a smart product out of dumb components? First, these components are extremely versatile. Wire-connected switches can be used to build functional elements (e.g., perform arithmetic operations), memory elements (e.g., store values and program instructions), and control elements (e.g., sequence program instructions in a execution datapath). These elements can for a programmable computer that can exhibit a very smart behavior.

Second, technology offers the ability to integrate millions or even billions of wire-connected switches in an extremely compact integrated circuit (IC). This large number of switches supports the construction of a powerful programmable computer and sophisticated real-world interfaces. The low cost of these devices allows them to be incorporated in a wide range of products.

Designing systems with millions of components is a challenge. A successful strategy is a *design hierarchy*. Use several simple elements to produce a more complex component. Then design the next level using these more complex components. The design hierarchy below spans from switches and wire to assembly programming.



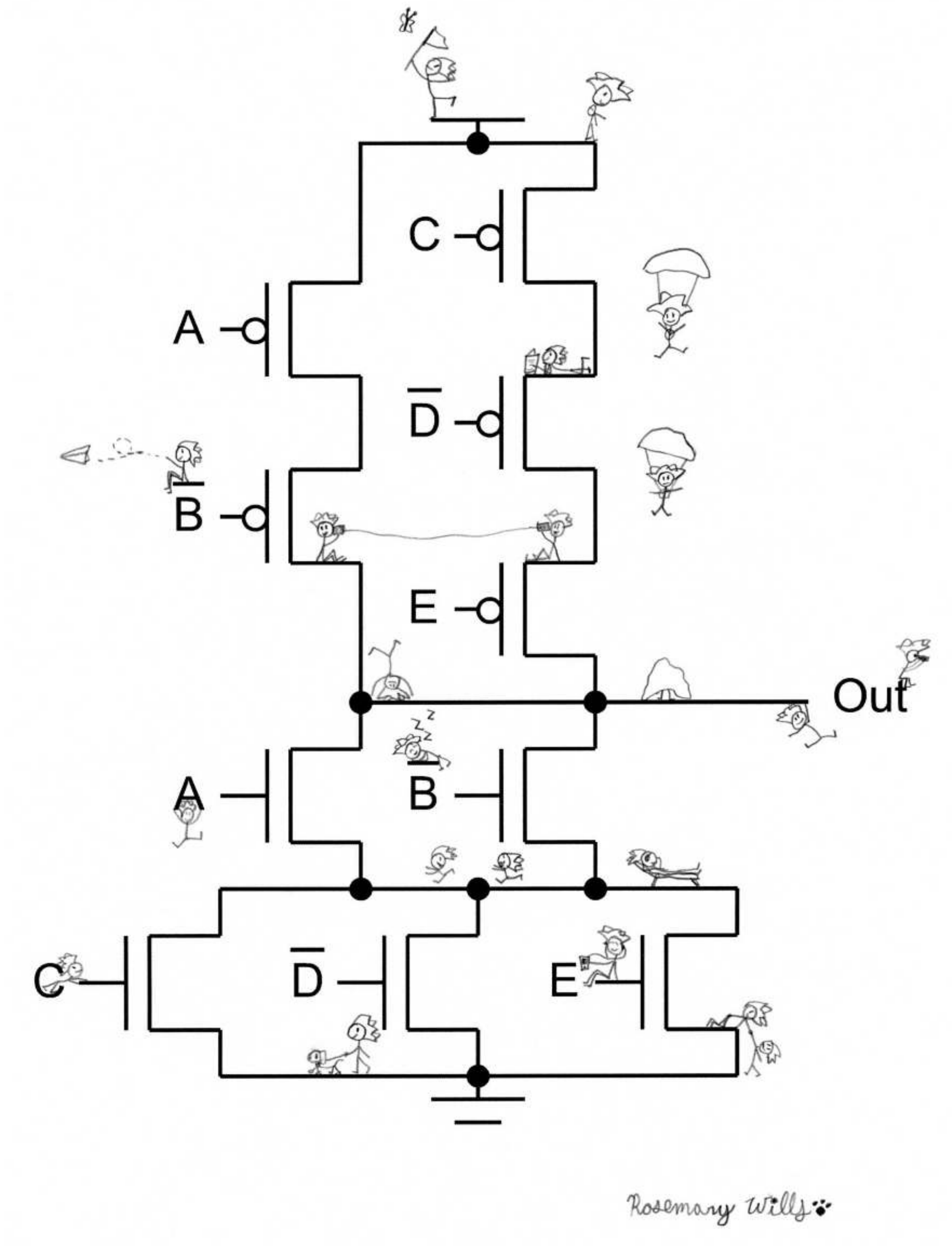
While there are other ways to describe the organization of programmable computers, this hierarchy illustrates the design path presented in the following chapters. Some chapters address specific levels of the hierarchy (designing with switches). Others focus on design techniques (simplification) and mathematics (Boolean algebra). All chapters emphasize design; how do you build a computer system.

Chapters

CompuCanvas: There is a free, open-source tool that can help you try out many of the ideas and techniques describe in these chapters. It is written in the Python programming language (available at www.python.org); so it runs on almost everything. Find out more about it at the CompuCanvas website www.compucanvas.org.

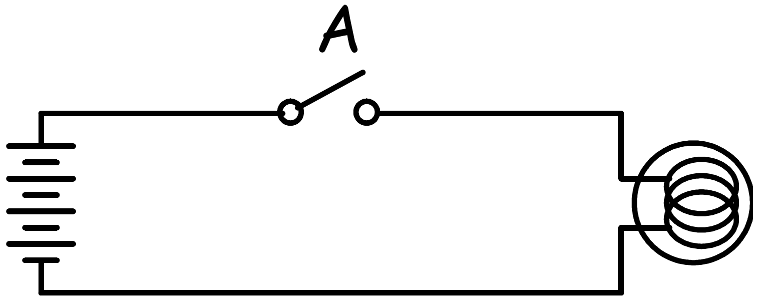
Designing Computer Systems

Switches and Wire



Switches and Wire

Despite their apparent complexity, digital computers are built from simple elements, namely *switches* and *wire*. To see how switches and wire can perform computations, consider the circuit below. The battery on the left is connected to the bulb on the right through the switch labeled **A**.



The battery will light the bulb if there is a complete path for current to flow from one side of the battery to the other. If the switch is open, no current can flow so the light is off. If the switch is closed, current flows and the light is on. The behavior of this simple circuit can be expressed using a table.

switch	light
open	off
closed	on

This type of table has been given the lofty name *Truth Table*. A more meaningful name would be *behavior table* since it describes the behavior of the circuit. Truth tables list all possible inputs to a system on the left and resulting outputs on the right. A truth table specifies how a system should behave. It does not specify how it should be implemented; this can be done in many ways.

Sometimes an icon is used to show connected nodes without drawing a wire. In the circuit below, the triangular symbols below the battery and bulb represent *ground*. We can imagine that all points attached to ground icons are connected together. So this circuit behaves identically to the circuit above.