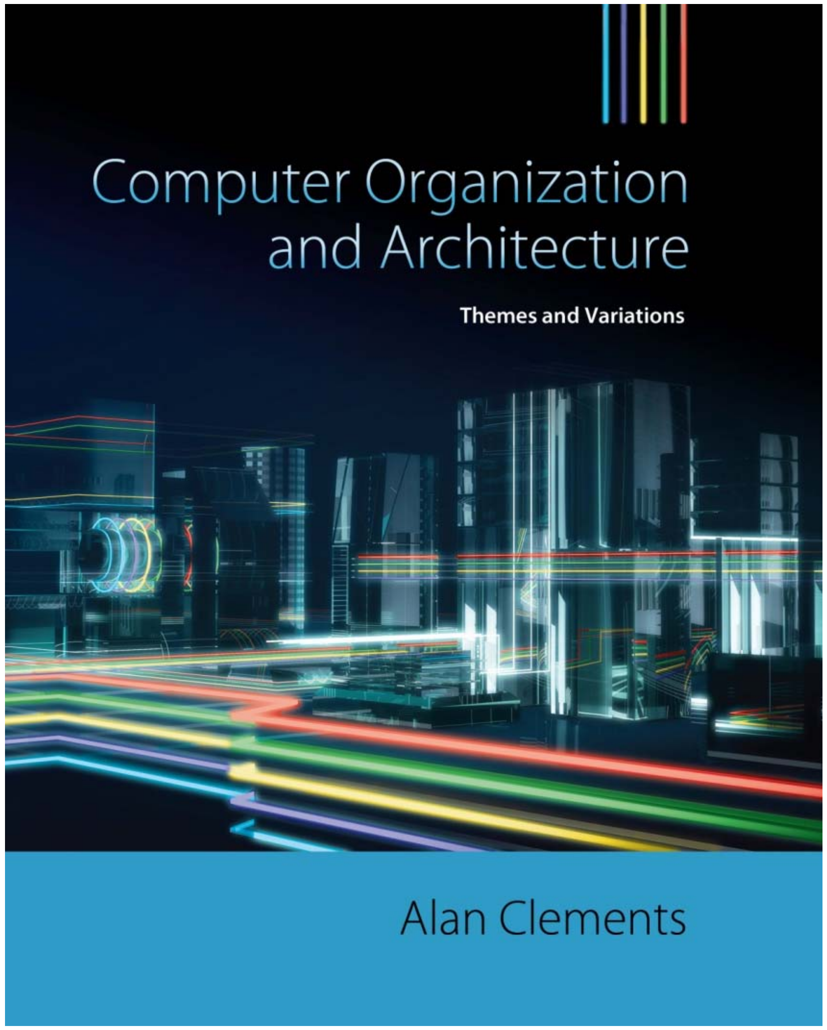
Chapter 3

Architecture and Organization



CENGAGE Learning™

The Instruction Set Architecture

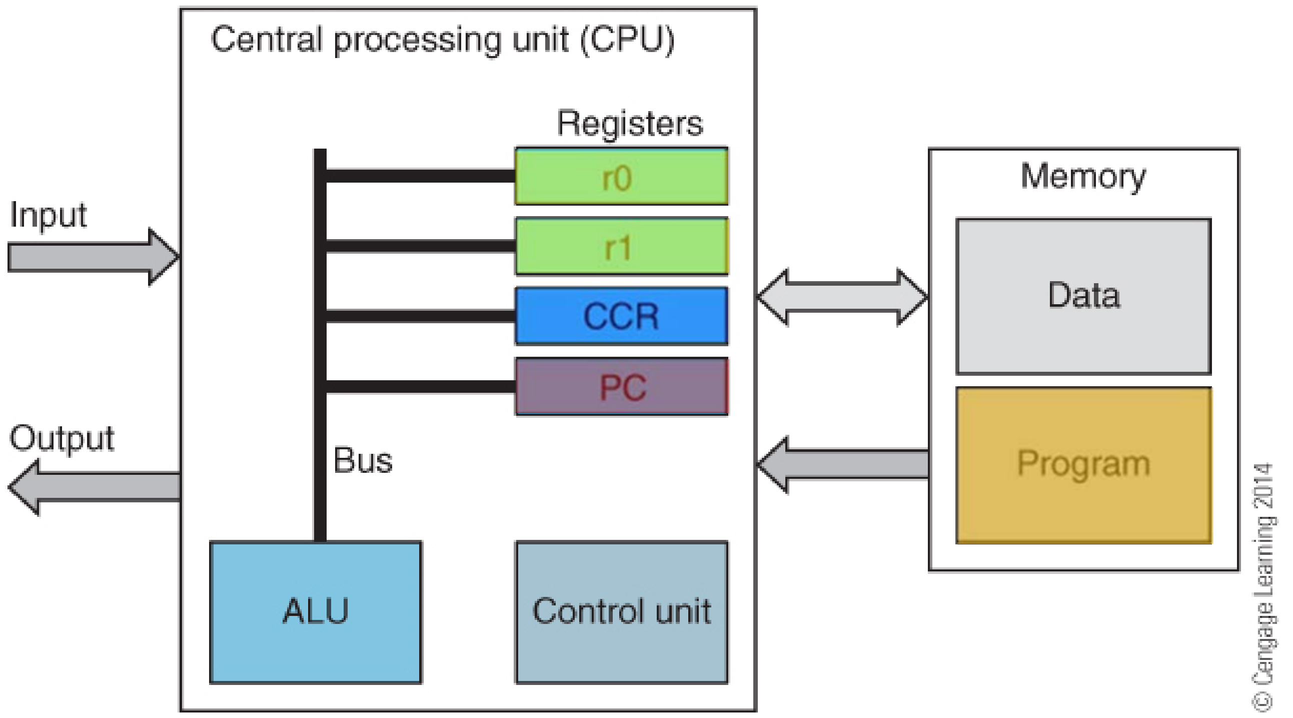
In this chapter, we:

- ❑ **Examine** the *stored program machine* and **show** how an instruction is executed
- ❑ **Introduce** instruction formats for
  - o *memory-to-memory*,
  - o *register-to-memory*, and
  - o *register-to-register* operations
- ❑ **Demonstrate** how a processor implements *conditional behavior*
- ❑ **Describe** a set of computer instructions
- ❑ **Show** how computers access data (*addressing modes*)
- ❑ **Introduce** an ARM's *Integrated Development Environment* (IDE) and
  show how ARM programs are written

The Instruction Set Architecture

- Figure 3.1 illustrate the structure of a simple *hypothetical* *stored program computer*.
- The *CPU reads* instructions from memory and *executes* them.
- *Temporary data* is stored in *registers* such as *r0* and *r1*.
- The ***PC***, *program counter*, is the register that *points at (i.e., contains the address of) the next instruction to be executed*.
- The ***CCR***, *Condition Code Register*, is a collection of flag bits for a processor.

FIGURE 3.1 Fundamental structure of a computer



Instruction Formats

- A computer executes instructions from 8 bits wide to multi-bytes wide.
- The instruction format defines the *anatomy of an instruction*
  - the number of bits devoted to defining the operation,
  - the number of operands, and
  - the format of each operand.
- Below are several examples of actual instructions:

LDR **registerdestination**,memorysource
STR registersource,**memorydestination**
Operation **registerdestination**,registersource1,registersource2

```
LDR   r1,1234
STR   r3,2000
ADD   r1,r2,r3
SUB   r3,r3,r1
```

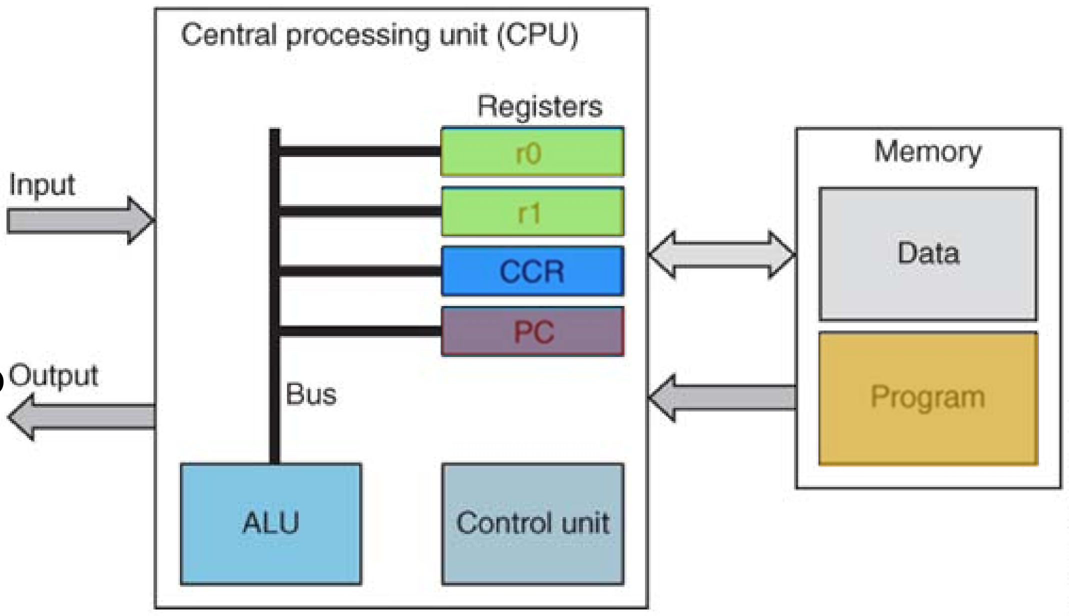
These are just *hypothetical* examples.

Features

- A stored program machine is a computer that has a program in digital form in its main memory.
- The program and data are stored in the same memory.
- The program counter (PC) points to the next instruction to be executed and is incremented after each instruction has been executed.

Fundamental structure of a computer

FIGURE 3.1



- A stored program operates in a *fetch*/execute two-phase mode.
  - In the *fetch phase* the next instruction is read from memory and decoded.
  - In the *execute phase* the instruction is interpreted or executed by the CPU's logic.
- Modern computers are ***pipelined***, where fetch and execute operations ***overlap***.

Features

A stored program computer has several registers.

r0 - r*i*	The register file is a *set of general-purpose registers*, e.g., r0, r1, r2, …, r*i* that store temporary (working) data, for example, the intermediate results of calculations, where *i* is typically 8, 16, or 32. A computer requires at *least one* general-purpose register.
PC	The *program counter* contains the *address* of the next instruction to be executed. Thus, the PC *points* to the location in memory that holds the next instruction.
IR	The *instruction register* stores the instruction most recently read from main memory. This is the instruction currently being executed.
MAR	The *memory address register* stores the *address* of the location in main memory that is currently being accessed by a read or write operation.
MBR	The *memory buffer register* stores data that has just been read from main memory, or data to be immediately *written* to main memory.