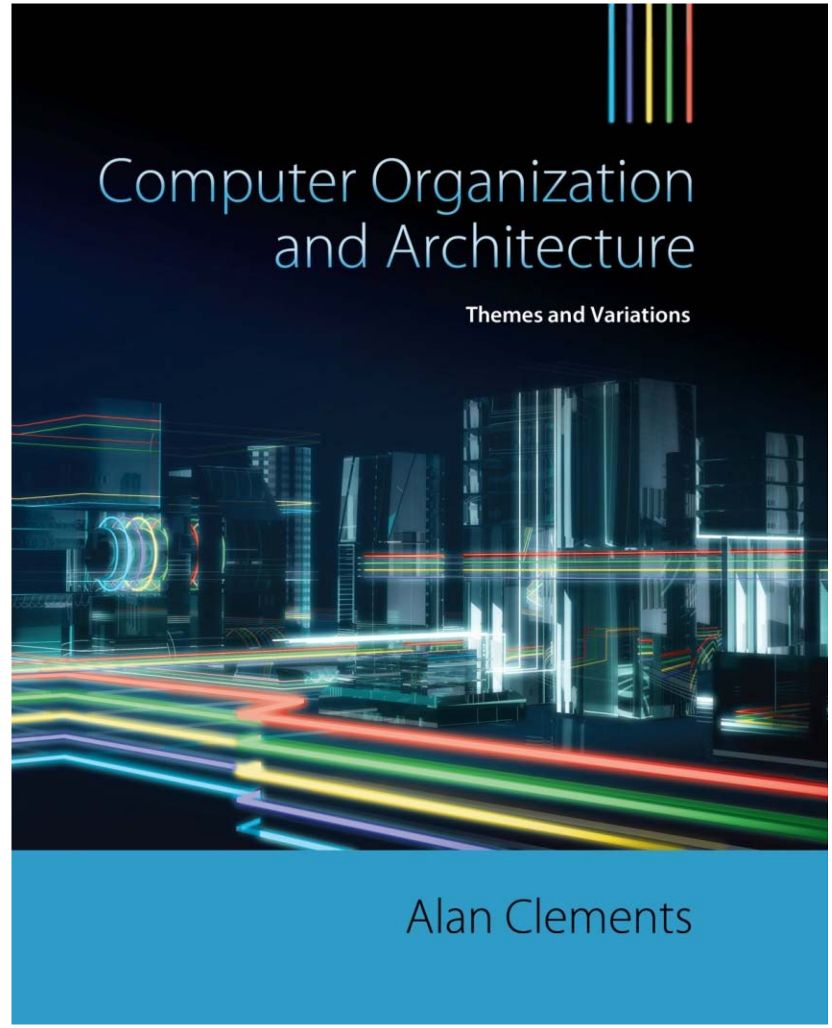
Part 1

CHAPTER 1

Computer Systems Architecture



CENGAGE Learning

Structure of the Book (5 Parts)

Part I ***The Beginning***

introduces the concepts, history and underlying technology of digital computers.

1. ***Computer Systems Architecture***
2. ***Computer Arithmetic and Digital Logic***

Part II ***Instruction Set Architectures (ISAs)***

looks at the ***programming model*** of a computer and introduces the *register model* of a computer, its *instruction types*, and the *addressing modes* of a typical microprocessor.

3. ***Architecture and Organization***
4. ***Instruction Set Architectures - Breadth and Depth***
5. Computer Architecture and Multimedia

Part III ***Organization and Efficiency***

describes how we measure the performance of computers.

6. ***Performance - Meaning and Metrics***
7. Processor Control
8. Beyond RISC: Superscalar, VLIW, and Itanium

Structure of the Book

Part IV ***The System*** X -3350.

covers the other parts of a computer required to ***convert the microprocessor chip into a complete system***; for example, *peripheral subsystems* and the wide range of *memory systems*, *storage devices*, and *buses* available to the computer systems' designer.

9. Cache Memory and Virtual Memory
10. Main Memory
11. Secondary Storage
12. Input/output

Part V ***Processor-Level Parallelism*** X - 4401.

goes beyond the single-processor computer and introduces the notion of ***computers with multiple processors***.

13. Processor-Level Parallelism

Computer Architecture

- A computer is characterized by its *instruction set architecture* (***ISA***)
- An ***ISA*** is an *abstract entity* because it does not consider the specific design or implementation of a computer
- An ***ISA*** is concerned with the computer's 1 *register set*, 2 *instruction set*, and 3 *addressing modes*
- An ***ISA*** defines the model of a computer *from the programmer viewpoint*
- The computer's assembly language embodies its ***ISA***

Arthitecture - Concept
Abstract.
Organization - Implement

Computer Organization

- Computer *organization* is concerned with the implementation of an ISA
- Any given ISA can have many different organizations - what you see.
  - Examples
    keep the same? • Architecture - Arms of a watch
    keep changing Organization - Screw, String that make these arms work

Should be "*organization*", not "*architecture*", as in the original slide.

- Computer manufacturers regularly *modify the organization* of a processor while *keeping its ISA essentially constant*

what makes it works.

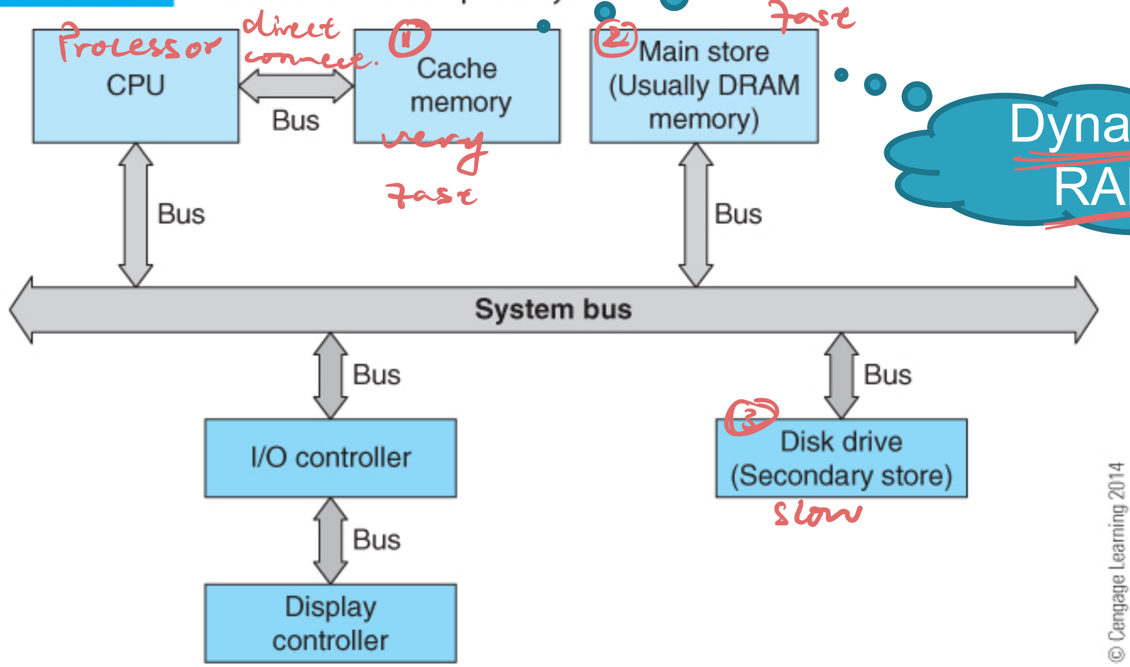
- Today, a computer's organization is often referred to as its *microarchitecture*
- In **theory**, ***architecture*** and ***organization*** are orthogonal; that is, they are entirely independent
- You could say that
  - ***architecture*** tells you *what* a computer does and
  - ***organization*** tells you ***how*** it does it

Computer Structure

- Figure 1.1 describes the structure of a computer.
- The term computer describes the entire system.
- The CPU is the *Central Processing Unit* that reads instructions and executes them.
- The CPU is often synonymous with microprocessor.
- Modern microprocessors usually include cache (high-speed) memory on-chip.
- A key component of computers is the bus (or family of busses) that moves information around the computer between different functional units (data highway).

Static RAM (faster and more expensive than DRAM)

FIGURE 1.1 Structure of a computer system



Dynamic RAM

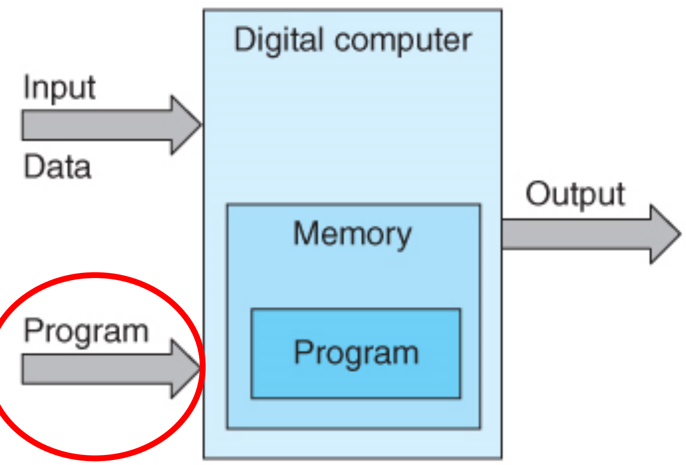
Processor Register

- A processor register is a memory element that holds a single unit data (a word of data). very very fast - located inside CPU any variable.
- A processor register is specified in terms of the number of bits it holds, which is typically, 8, 16, 32, or 64.
  - Currently, most of computers either have 32-bit or 64-bit-wide registers.
- Each processor has a specified number or registers.
- There is no fundamental difference between
  - a register and
  - a word in memory.
- The practical difference is that registers are located within the CPU
  - can be accessed more rapidly than other memories.

Computer Types

- Computers are either dedicated or general-purpose.
  - A dedicated computer solves only one class of problems (e.g., a computer in a calculator, a cruise speed control, or washing machine).
  - A general-purpose computer can be programmed to solve any problem.
- Figure 1.2 describes the structure of a general-purpose computer.
- A key feature of the general-purpose computer is that *the program* and *its data* are held in the same memory.
- Such a computer is called a **von Neumann machine**.

FIGURE 1.2 The general-purpose computer



General-purpose digital machine. By changing the program, this machine can carry out any task capable of being performed by a computer.

© Cengage Learning 2014

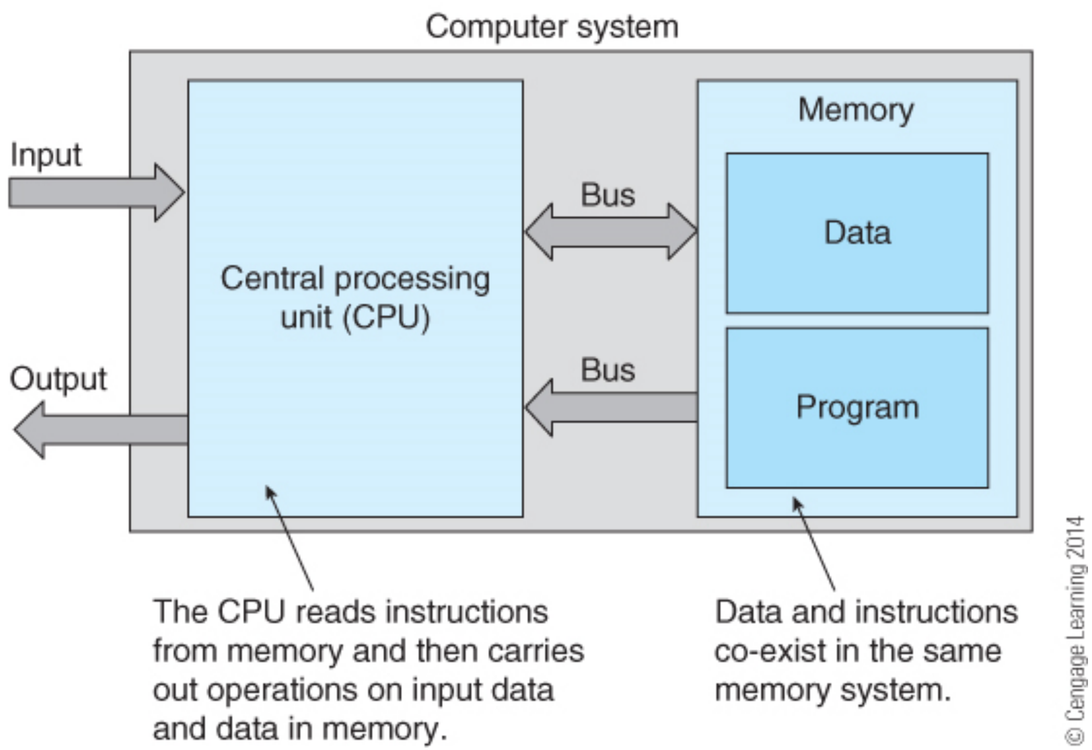
The ability to freely change the program is the main difference between a dedicated and a general-purpose computers.

dedicated computer only can input data, no program

Stored Program Computer

- Figure 1.3 emphasizes the nature of the stored program computer.
  - The CPU reads instructions from memory and then
  - carries out operations on input data and data in memory
  - Data and instructions co-exist in the same memory system

FIGURE 1.3 Structure of the stored program computer

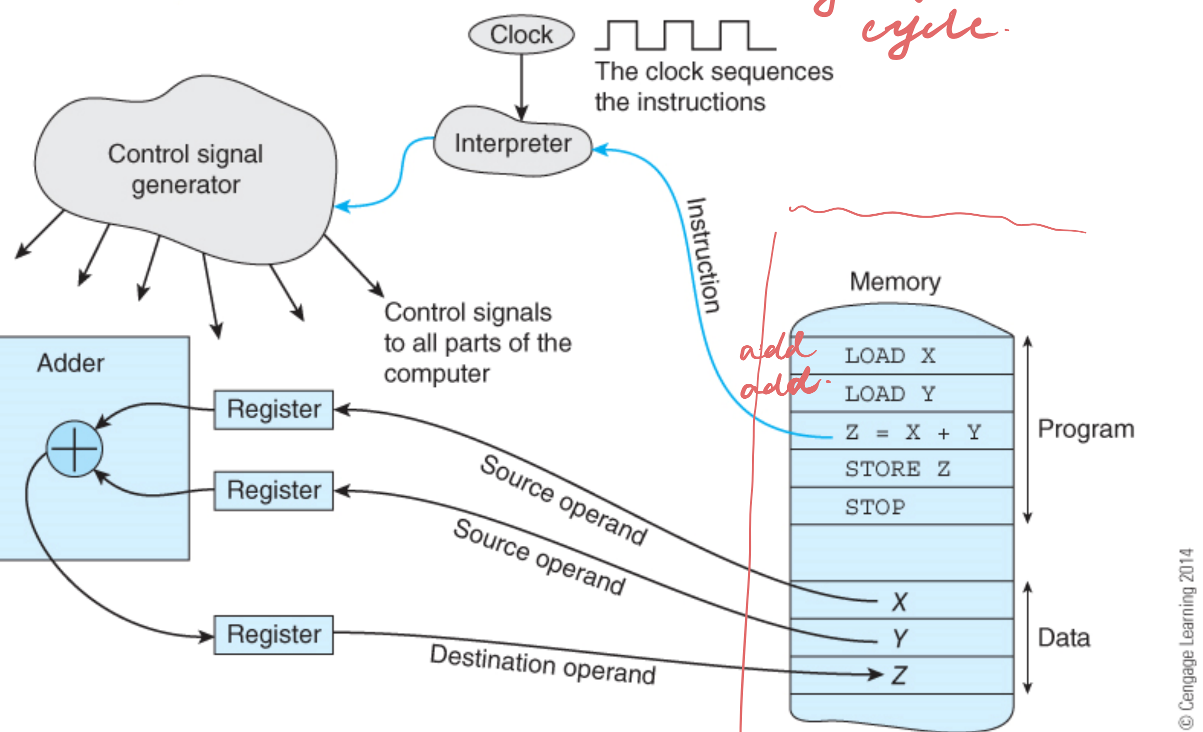


Stored Program Computer

- Figure 1.4 illustrates the operation of a stored program, where the operation Z = X + Y is read from memory, interpreted and used to add X and Y to create Z.
- A clock (a stream of pulses) sequences all operations in a computer.
- All events in a computer are triggered by clock pulses.

must start at the beginning of a clock cycle.

FIGURE 1.4 Executing a program



CPU is every where in this picture

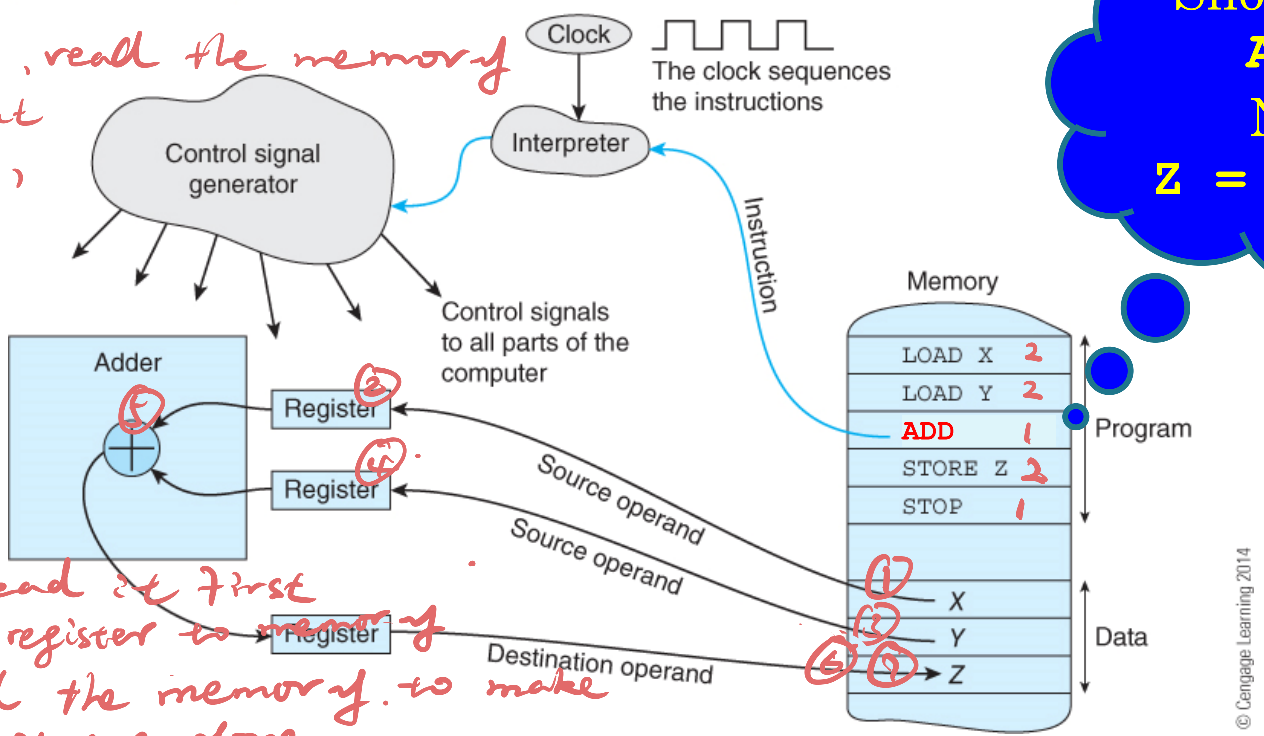
Stored Program Computer

- LOAD moves data from memory to a register and
- STORE moves data from a register to memory.
- Z = X + Y performs a simple operation on data (addition).
- Memory is a bottleneck because
  - instructions have to flow from it.
  - Data has to flow from it to take part in operations and
  - Data has to flow back to store the result.

How many memory access do we need to execute this program? 8

Should be **ADD** Not **Z = X + Y**

FIGURE 1.4 Executing a program



1. load x to CPU, read the memory
2. Interpreter sent control signal, x is saved to register
3.
4.
5. read from memory to add
6. To save z, read it first
7. Store z from register to memory
8. To stop, read the memory to make sure all steps are done.

The Clock

- Most digital electronic circuits have a clock that generates a continuous stream of regularly spaced electrical pulses.
- It's called a clock because the pulses are used to time or sequence all events within the computer; for example, a processor might start executing a new instruction each time a clock pulse arrives.
- A clock is defined in terms of its *repetition rate* i.e., *frequency*.
- Typical clock frequencies in computers range from 1 MHz to about 4.5 GHz.
- Clocks are also defined in terms of the width of a clock pulse, which is the reciprocal of its frequency; that is $\boxed{f = 1/T}$;
  for example a ***1 MHz*** clock has a duration of ***1 μs (i.e., 10^{-6}s)***, $\frac{1}{10^6}$
  and a ***1 GHz*** clock has a duration of ***1 ns (i.e., 10^{-9}s)***.
- A ***5 GHz*** clock has a period of ***0.2 ns*** or simply ***200 ps*** (picoseconds)—***1 ps = 10^{-12} s***
- To feel how a ***ps*** is small, light travels approximately ***6 cm*** in ***200 ps***.

Synchronous vs Asynchronous

- Digital circuits whose events are triggered by a clock are called *synchronous*.
- Some events are *asynchronous* because they can happen at any time.
- If you move the mouse, it sends a signal to the computer. That is an *asynchronous* event.
- The computer may check the status of a device at each clock pulse. That is a *synchronous* event.

Synchronous - trigger by clock.
asynchronous - trigger by event.