COM/BLM 376 Computer Architecture

Chapter 2 Computer Evolution and Performance

Asst. Prof. Dr. Gazi Erkan BOSTANCI

ebostanci@ankara.edu.tr

Slides are mainly based on

Computer Organization and Architecture: Designing for Performance by William Stallings, 9th Edition, Prentice Hall

Outline

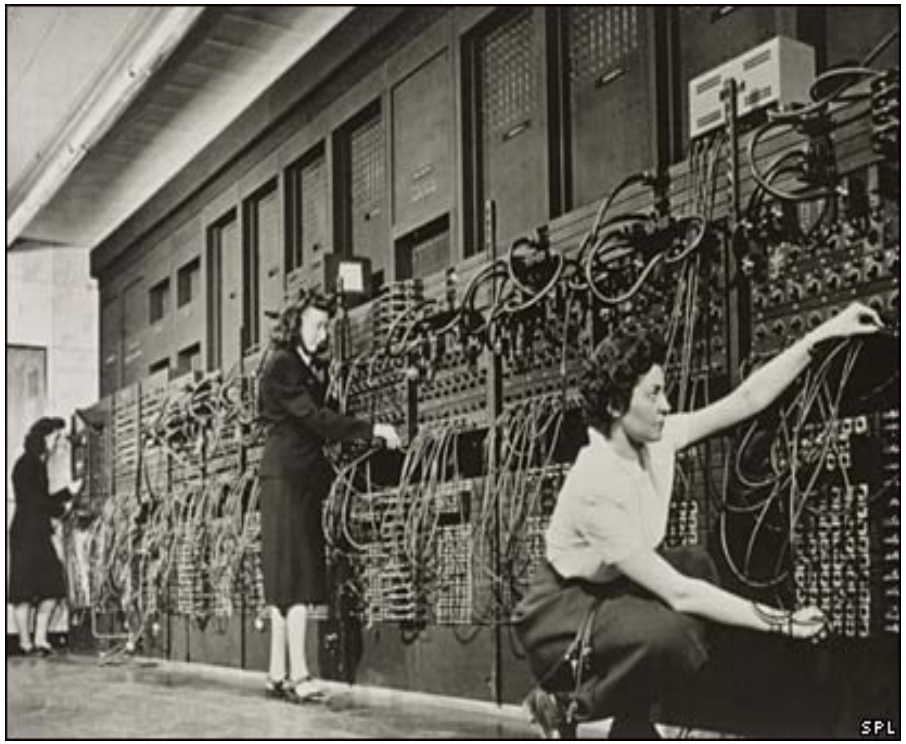
- A Brief History of Computers
- Designing for Performance
- Multicore, MICs, GPGPUs
- Embedded Systems
- Performance Assessment

A Brief History of Computers

The First Generation: Vacuum Tubes

- ***ENIAC*** The ENIAC (Electronic Numerical Integrator And Computer), designed and constructed at the University of Pennsylvania, was the world's first general purpose electronic digital computer.
- The resulting machine was enormous, weighing 30 tons, occupying 1500 square feet of floor space, and containing more than 18,000 vacuum tubes. When operating, it consumed 140 kilowatts of power. It was also substantially faster than any electromechanical computer, capable of 5000 additions per second.

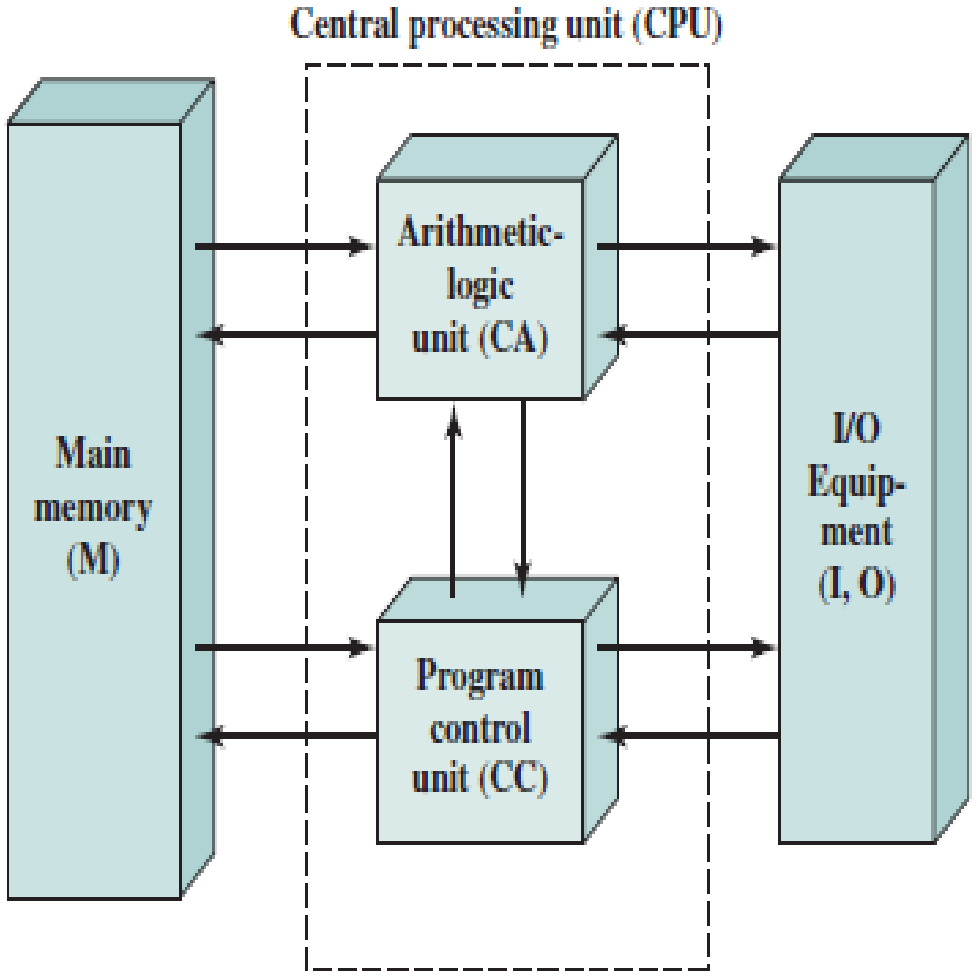
- The ENIAC was a decimal rather than a binary machine. That is, numbers were represented in decimal form, and arithmetic was performed in the decimal system. Its memory consisted of 20 *accumulators*, each capable of holding a 10-digit decimal number. A ring of 10 vacuum tubes represented each digit.
- At any time, only one vacuum tube was in the ON state, representing one of the 10 digits. The major drawback of the ENIAC was that it had to be programmed manually by setting switches and plugging and unplugging cables.



The Von Neuman Machine

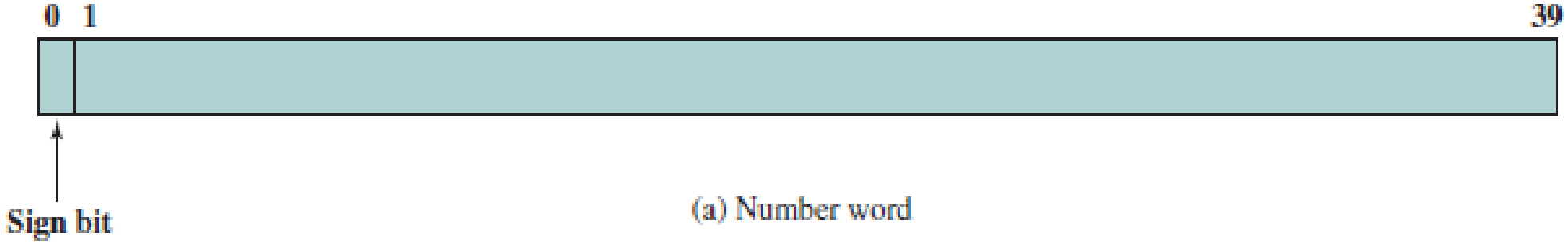
- The task of entering and altering programs for the ENIAC was extremely tedious.
- Stored Program Concept
  - Suppose a program could be represented in a form suitable for storing in memory alongside the data. Then, a computer could get its instructions by reading them from memory, and a program could be set or altered by setting the values of a portion of memory.

- In 1946, von Neumann and his colleagues began the design of a new stored program computer, referred to as the IAS computer, at the Princeton Institute for Advanced Studies.
- It consists of
  - A **main memory**, which stores both data and instructions
  - An **arithmetic and logic unit (ALU)** capable of operating on binary data
  - A **control unit**, which interprets the instructions in memory and causes them to be executed
  - **Input/output (I/O)** equipment operated by the control unit

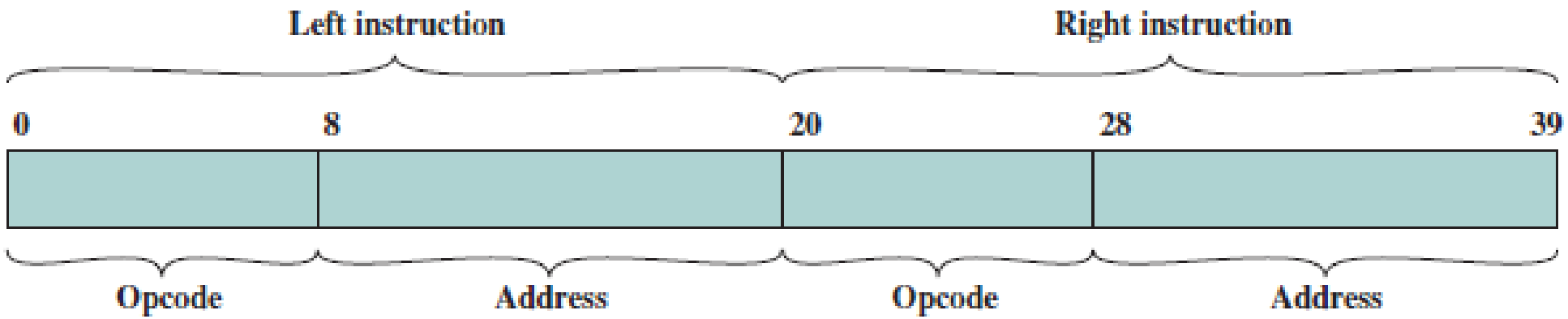


- With rare exceptions, all of today's computers have this same general structure and function and are thus referred to as **von Neumann machines**. Thus, it is worthwhile at this point to describe briefly the operation of the IAS computer.
- The memory of the IAS consists of 1000 storage locations, called **words**, of 40 binary digits (bits) each. Both data and instructions are stored there. Numbers are represented in binary form, and each instruction is a binary code.
  - Note that there is no universal definition for *word* since it depends on the instruction length for a given computer.
- Each number is represented by a sign bit and a 39-bit value. A word may also contain two 20-bit instructions, with each instruction consisting of an 8-bit operation code **(opcode)** specifying the operation to be performed and a 12-bit address designating one of the words in memory numbered from 0 to 999).

- IAS Memory Formats

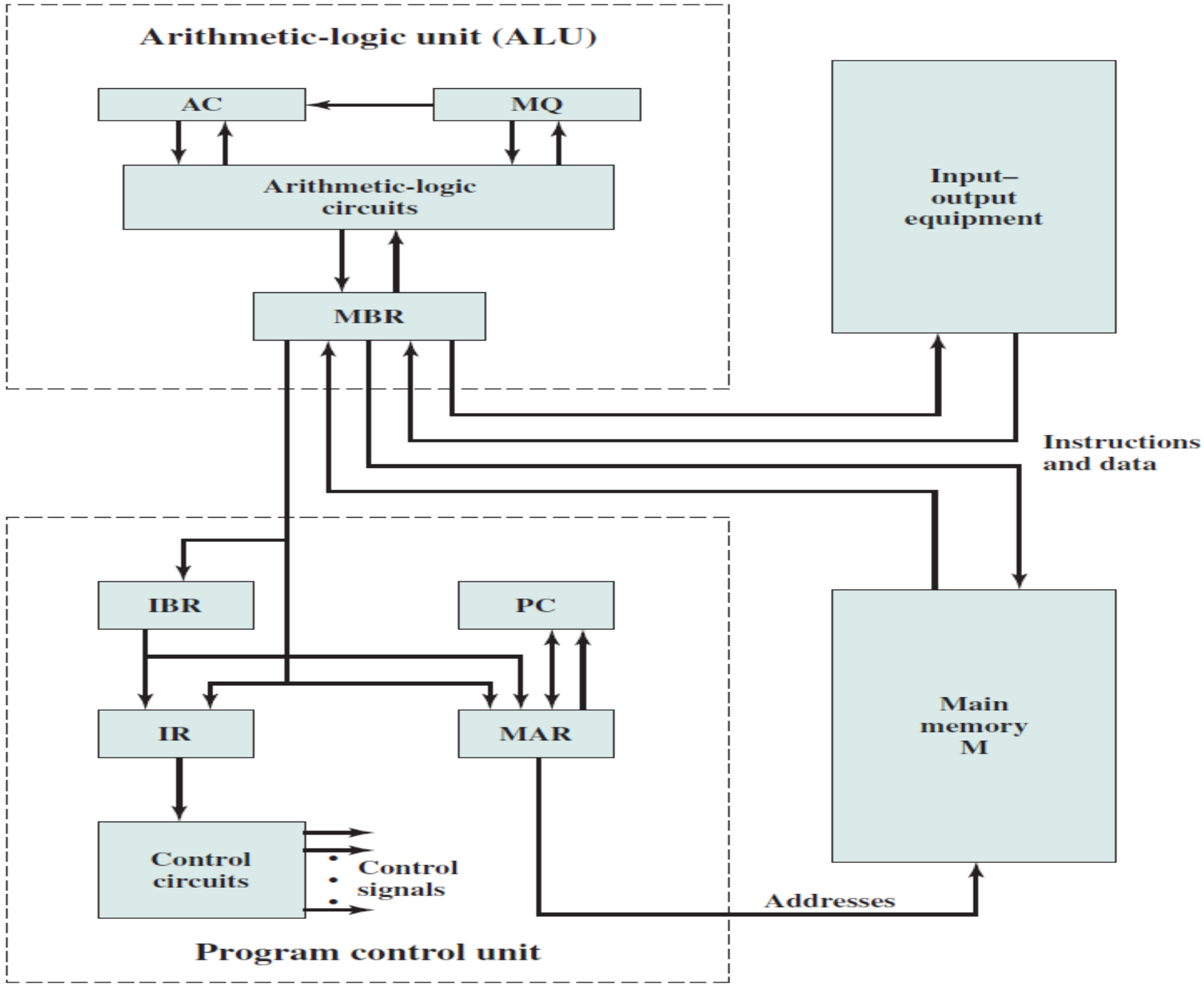


(a) Number word



(b) Instruction word

- The control unit operates the IAS by fetching instructions from memory and executing them one at a time.
- To explain this, a more detailed structure diagram is needed with the following components:
  - **Memory buffer register (MBR):** Contains a word to be stored in memory or sent to the I/O unit, or is used to receive a word from memory or from the I/O unit.
  - **Memory address register (MAR):** Specifies the address in memory of the word to be written from or read into the MBR.
  - **Instruction register (IR):** Contains the 8-bit opcode instruction being executed.
  - **Instruction buffer register (IBR):** Employed to hold temporarily the right hand instruction from a word in memory.
  - **Program counter (PC):** Contains the address of the next instruction pair to be fetched from memory.
  - **Accumulator (AC) and multiplier quotient (MQ):** Employed to hold temporarily operands and results of ALU operations. For example, the result ofmultiplying two 40-bit numbers is an 80-bit number; the most significant 40 bitsare stored in the AC and the least significant in the MQ.


Arithmetic-logic unit (ALU)
AC
MQ
Arithmetic-logic circuits
MBR
Input–output equipment
Instructions and data
IBR
PC
IR
MAR
Main memory M
Control circuits
Control signals
Addresses
Program control unit

- The IAS operates by repetitively performing an **instruction cycle**.
- Each instruction cycle consists of two subcycles. During the **fetch cycle**, the opcode of the next instruction is loaded into the IR and the address portion is loaded into the MAR. This instruction may be taken from the IBR, or it can be obtained from memory by loading a word into the MBR, and then down to the IBR,IR, and MAR.
- Once the opcode is in the IR, the **execute cycle** is performed.
- Control circuitry interprets the opcode and executes the instruction by sending out the appropriate control signals to cause data to be moved or an operation to be performed by the ALU.

- The IAS computer had a total of 21 instructions which can be grouped as follows:
  - **Data transfer:** Move data between memory and ALU registers or between two ALU registers.
  - **Unconditional branch:** Normally, the control unit executes instructions in sequence from memory. This sequence can be changed by a branch instruction, which facilitates repetitive operations.
  - **Conditional branch:** The branch can be made dependent on a condition, thus allowing decision points.
  - **Arithmetic:** Operations performed by the ALU.
  - **Address modify:** Permits addresses to be computed in the ALU and then inserted into instructions stored in memory. This allows a program considerable addressing flexibility.

- Each instruction must conform to the instruction format. The opcode portion(first 8 bits) specifies which of the 21 instructions is to be executed.
- The address portion (remaining 12 bits) specifies which of the 1000 memory locations is to be involved in the execution of the instruction.
- Note that each operation requires several steps. Some of these are quite elaborate.
- The multiplication operation requires 39 suboperations, one for each bit position except that of the sign bit.

- **The Second Generation: Transistors**
- The first major change in the electronic computer came with the replacement of the vacuum tube by the transistor. The transistor is
  - smaller,
  - cheaper, and
  - Dissipates less heat than a vacuum tube but can be used in the same way as a vacuum tube to construct computers.
- Unlike the vacuum tube, which requires wires, metal plates, a glass capsule, and a vacuum, the transistor is a *solid-state device*, made from silicon.

- The use of the transistor defines the *second generation* of computers. It has become widely accepted to classify computers into generations based on the fundamental hardware technology employed. Each new generation is characterized by greater processing performance, larger memory capacity, and smaller size than the previous one.

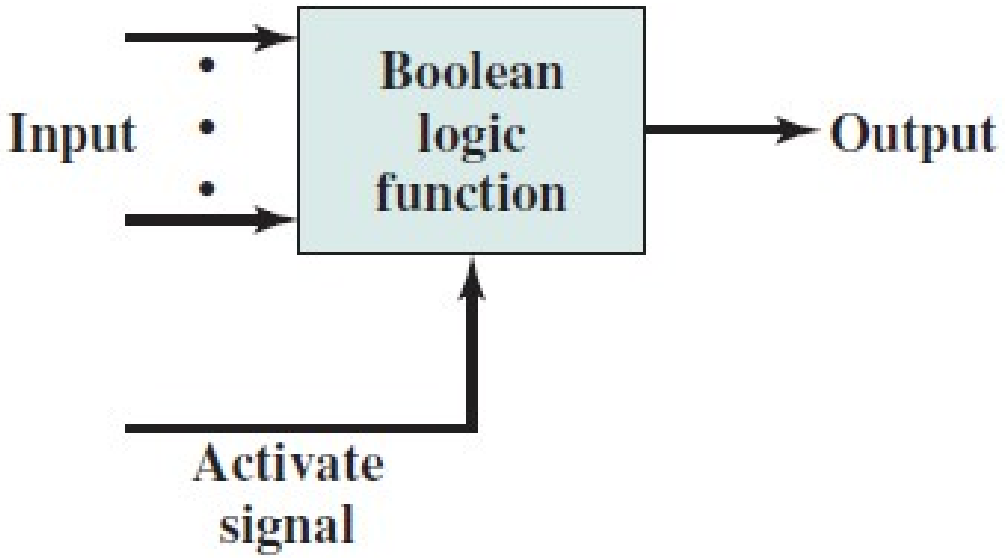
Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1958–1964	Transistor	200,000
3	1965–1971	Small- and medium-scale integration	1,000,000
4	1972–1977	Large-scale integration	10,000,000
5	1978–1991	Very-large-scale integration	100,000,000
6	1991–	Ultra-large-scale integration	1,000,000,000

The Third Generation: Integrated Circuits

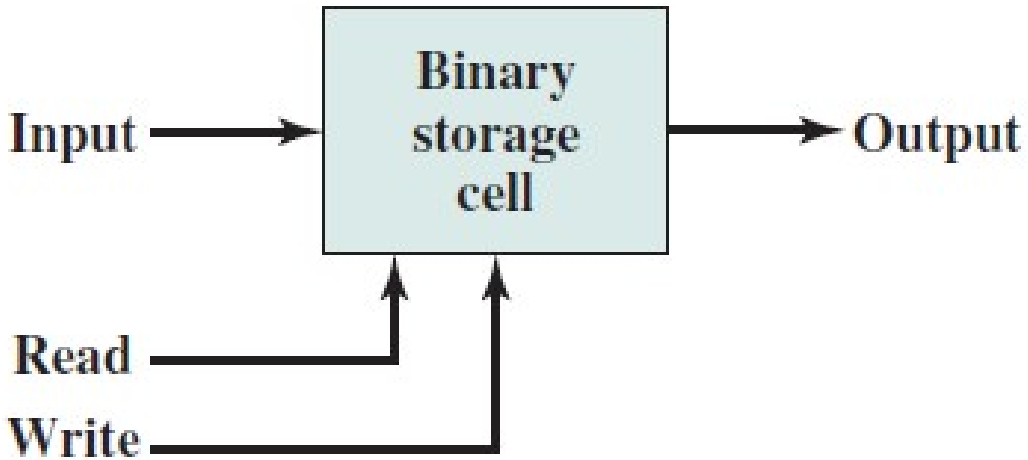
- A single, self-contained transistor is called a *discrete component*.
- Throughout the 1950s and early 1960s, electronic equipment was composed largely of discrete components—transistors, resistors, capacitors, and so on.
- Discrete components were manufactured separately, packaged in their own containers, and soldered or wired together onto circuit boards, which were then installed in computers and other electronic equipment. Whenever an electronic device called for a transistor, a little tube of metal containing a pinhead-sized piece of silicon had to be soldered to a circuit board.
  - The entire manufacturing process, from transistor to circuit board, was expensive and cumbersome.
- In 1958 came the achievement that revolutionized electronics and started the era of microelectronics: the invention of the integrated circuit.

Microelectronics

- Literally, “small electronics.” Since the beginnings of digital electronics and the computer industry, there has been a persistent and consistent trend toward the reduction in size of digital electronic circuits.
- The basic elements of a digital computer, as we know, must perform storage, movement, processing, and control functions. Only two fundamental types of components are required: gates and memory cells.
- A gate is a device that implements a simple Boolean or logical function, such as IF *A* AND *B* ARE TRUETHEN *C* IS TRUE (AND gate). Such devices are called gates because they control data flow in much the same way that canal gates control the flow of water.



(a) Gate

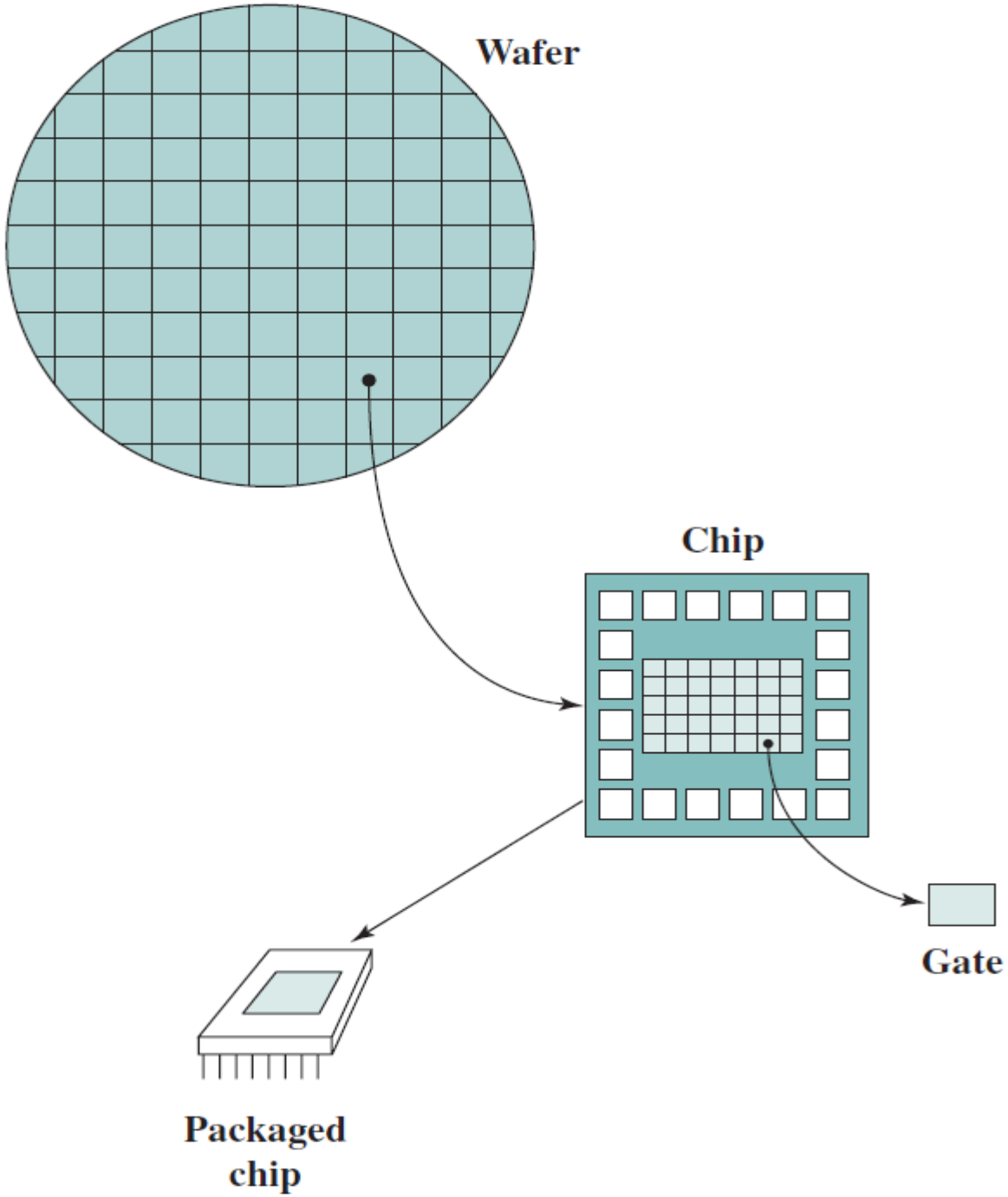


(b) Memory cell

- By interconnecting large numbers of these fundamental devices, we can construct a computer. We can relate this to our four basic functions as follows:
  - **Data storage:** Provided by memory cells.
  - **Data processing:** Provided by gates.
  - **Data movement:** The paths among components are used to move data from memory to memory and from memory through gates to memory.
  - **Control:** The paths among components can carry control signals. For example, a gate will have one or two data inputs plus a control signal input that activates the gate. When the control signal is ON, the gate performs its function on the data inputs and produces a data output. Similarly, the memory cell will store the bit that is on its input lead when the WRITE control signal is ON and will place the bit that is in the cell on its output lead when the READ control signal is ON.

- A thin **wafer** of silicon is divided into a matrix of small areas, each a few millimeters square. The identical circuit pattern is fabricated in each area, and the wafer is broken up into **chips**. Each chip consists of many gates and/or memory cells plus a number of input and output attachment points.

- This chip is then packaged in housing that protects it and provides pins for attachment to devices beyond the chip. A number of these packages can then be interconnected on a printed circuit board to produce larger and more complex circuits.


Wafer
Chip
Gate
Packaged
chip

- Gordon Moore (cofounder of Intel) observed that the number of transistors that could be put on a single chip was doubling every year and correctly predicted that this pace would continue into the near future. To the surprise of many, including Moore, the pace continued year after year and decade after decade. The pace slowed to a doubling every 18 months in the 1970s but has sustained that rate ever since.

The consequences of Moore's law are profound:

1. The cost of a chip has remained virtually unchanged during this period of rapid growth in density. This means that the cost of computer logic and memory circuitry has fallen at a dramatic rate.
2. Because logic and memory elements are placed closer together on more densely packed chips, the electrical path length is shortened, increasing operating speed.
3. The computer becomes smaller, making it more convenient to place in a variety of environments.
4. There is a reduction in power and cooling requirements.
5. The interconnections on the integrated circuit are much more reliable than solder connections. With more circuitry on each chip, there are fewer inter-chip connections.

MICROPROCESSORS Just as the density of elements on memory chips has continued to rise, so has the density of elements on processor chips. As time went on, more and more elements were placed on each chip, so that fewer and fewer chips were needed to construct a single computer processor.

- •A breakthrough was achieved in 1971, when Intel developed its 4004. The4004 was the first chip to contain *all* of the components of a CPU on a single chip: The microprocessor was born.
- •The 4004 can add two 4-bit numbers and can multiply only by repeated addition. By today's standards, the 4004 is hopelessly primitive, but it marked the beginning of a continuing evolution of microprocessor capability and power.

- The next major step in the evolution of the microprocessor was the introduction of the Intel 8008 in 1972. This was the first 8-bit microprocessor and was almost twice as complex as the 4004.
- Neither of these steps was to have the impact of the next major event: the introduction of the Intel 8080 in 1974. This was the first general-purpose microprocessor.

- About the same time, 16-bit microprocessors began to be developed. However, it was not until the end of the 1970s that powerful, general-purpose 16-bit microprocessors appeared. One of these was the 8086.

(a) 1970s Processors

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2300	3500	6000	29,000	29,000
Feature size (μm)	10		6	3	6
Addressable memory	640 Bytes	16 kB	64 kB	1 MB	1 MB

(b) 1980s Processors

	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz–12.5 MHz	16 MHz–33 MHz	16 MHz–33 MHz	25 MHz–50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8–1
Addressable memory	16 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB

(c) 1990s Processors

	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz–33 MHz	60 MHz–166 MHz,	150 MHz–200 MHz	200 MHz–300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

(d) Recent Processors

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 990
Introduced	1999	2000	2006	2011
Clock speeds	450–660 MHz	1.3–1.8 GHz	1.06–1.2 GHz	3.5 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1170 million
Feature size (nm)	250	180	65	32
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1.5 MB L2/12 MB L3

Designing for Performance

- Year by year, the cost of computer systems continues to drop dramatically, while the performance and capacity of those systems continue to rise equally dramatically.
- Today's laptops have the computing power of an IBM mainframe from 10 or 15 years ago. Thus, we have virtually "free" computer power. Processors are so inexpensive that we now have microprocessors we throw away.
- The digital pregnancy test as an example (used once and then thrown away).

- Desktop applications that require the great power of today's microprocessor-based systems include
  - Image processing
  - Speech recognition
  - Videoconferencing
  - Multimedia authoring
  - Voice and video annotation of files
  - Simulation modeling
- Workstation systems now support highly sophisticated engineering and scientific applications, as well as simulation systems, and have the ability to support image and video applications. In addition, businesses are relying on increasingly powerful servers to handle transaction and database processing and to support massive client/server networks that have replaced the huge mainframe computer centers of previous years.

Microprocessor Speed

- What gives Intel x86 processors or IBM mainframe computers such mind-boggling power is the relentless pursuit of speed by processor chip manufacturers.
- In microprocessors, the addition of new circuits, and the speed boost that comes from reducing the distances between them, has improved performance four-or fivefold every three years or so since Intel launched its x86 family in 1978.

- But the raw speed of the microprocessor will not achieve its potential unless it is fed a constant stream of work to do in the form of computer instructions.
  - Anything that gets in the way of that smooth flow undermines the power of the processor.

- Accordingly, while the chipmakers have been busy learning how to fabricate chips of greater and greater density, the processor designers must come up with ever more elaborate techniques for feeding the monster.

- Among the techniques built into contemporary processors are the following:
  - **Pipelining:** With pipelining, a processor can simultaneously work on multiple instructions. The processor overlaps operations by moving data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously.
  - For example, while one instruction is being executed, the computer is decoding the next instruction.
  - **Branch prediction:** The processor looks ahead in the instruction code fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next. If the processor guesses right most of the time, it can *prefetch* the correct instructions and buffer them so that the processor is kept busy.
  - The more sophisticated examples of this strategy predict not just the next branch but multiple branches ahead. Thus, branch prediction increases the amount of work available for the processor to execute.

- **Data flow analysis:** The processor analyzes which instructions are dependent on each other’s results, or data, to create an optimized schedule of instructions. In fact, instructions are scheduled to be executed when ready, independent of the original program order. This prevents unnecessary delay.
- **Speculative execution:** Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution, holding the results in temporary locations. This enables the processor to keep its execution engines as busy as possible by executing instructions that are likely to be needed.

Multicore, MICS and GPGPUs

- The use of multiple processors on the same chip, also referred to as multiple cores, or **multicore**, provides the potential to increase performance without increasing the clock rate.
- Studies indicate that, within a processor, the increase in performance is roughly proportional to the square root of the increase in complexity.
- But if the software can support the effective use of multiple processors, then doubling the number of processors almost doubles performance. Thus, the strategy is to use two simpler processors on the chip rather than one more complex processor.

- Chip manufacturers are now in the process of making a huge leap forward in the number of cores per chip, with more than 50 cores per chip. The leap in performance as well as the challenges in developing software to exploit such a large number of cores have led to the introduction of a new term: **many integrated core(MIC)**.
- The multicore and MIC strategy involves a homogeneous collection of general-purpose processors on a single chip. At the same time, chip manufacturers are pursuing another design option: a chip with multiple general-purpose processors plus **graphics processing units (GPUs)** and specialized cores for video processing and other tasks.

- Since GPUs perform parallel operations on multiple sets of data, they are increasingly being used as vector processors for a variety of applications that require repetitive computations. This blurs the line between the GPU and the CPU. When a broad range of applications are supported by such a processor, the term **general-purpose computing on GPUs (GPGPU)**is used.

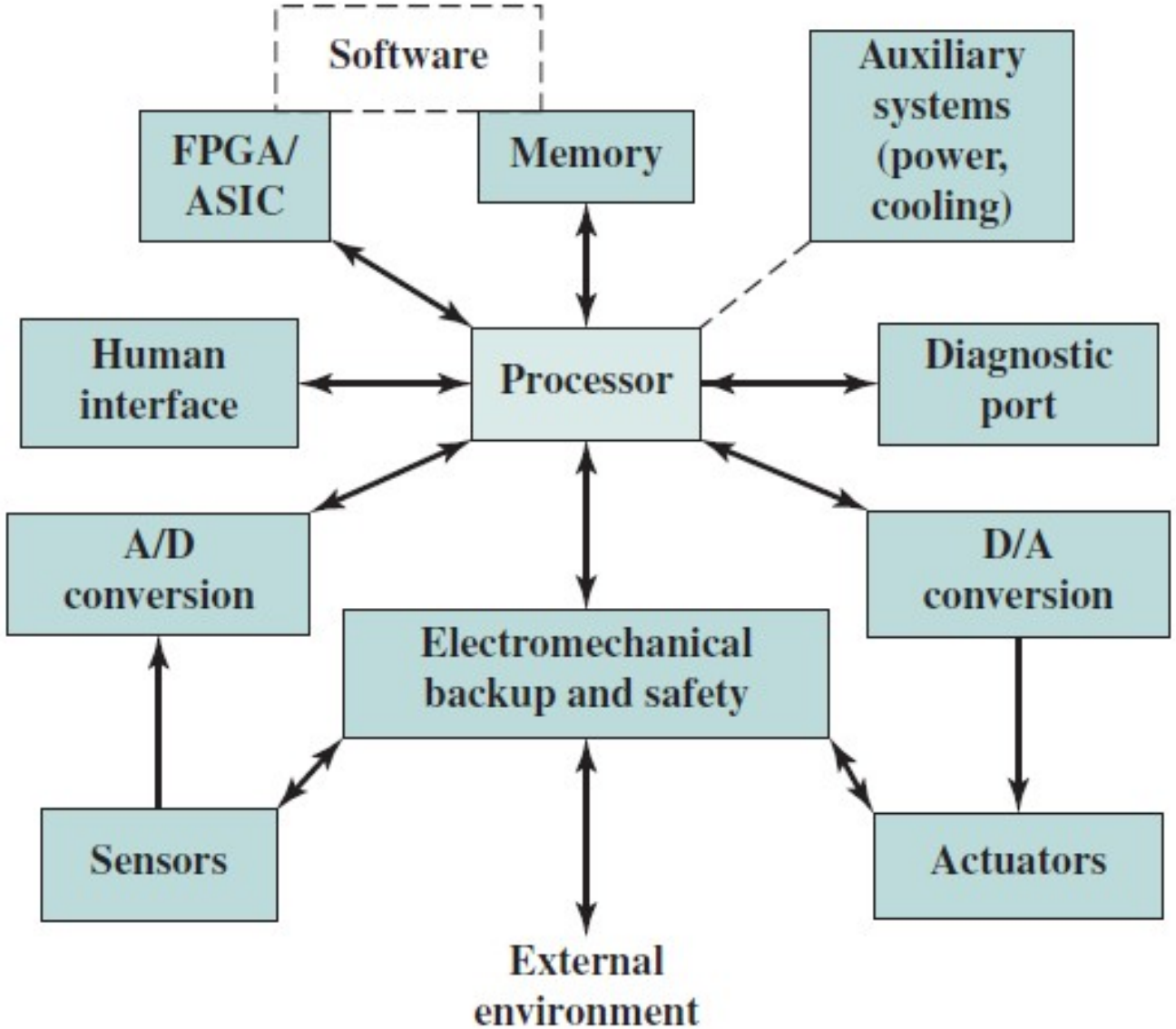
Embedded Systems

- A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function. In many cases, embedded systems are part of a larger system or product, as in the case of an antilock braking system in a car.
- Embedded systems far outnumber general-purpose computer systems, encompassing a broad range of applications

Market	Embedded Device
Automotive	Ignition system Engine control Brake system
Consumer electronics	Digital and analog televisions Set-top boxes (DVDs, VCRs, Cable boxes) Personal digital assistants (PDAs) Kitchen appliances (refrigerators, toasters, microwave ovens) Automobiles Toys/games Telephones/cell phones/pagers Cameras Global positioning systems
Industrial control	Robotics and controls systems for manufacturing Sensors
Medical	Infusion pumps Dialysis machines Prosthetic devices Cardiac monitors
Office automation	Fax machine Photocopier Printers Monitors Scanners

These systems have widely varying requirements and constraints, such as the following:

- Small to large systems, implying very different cost constraints, thus different needs for optimization and reuse
- Relaxed to very strict requirements and combinations of different quality requirements, for example, with respect to safety, reliability, real-time, and flexibility
- Short to long life times
- Different environmental conditions in terms of, for example, radiation, vibrations, and humidity


Software
FPGA/
ASIC
Memory
Auxiliary
systems
(power,
cooling)
Human
interface
Processor
Diagnostic
port
A/D
conversion
D/A
conversion
Electromechanical
backup and safety
Sensors
Actuators
External
environment

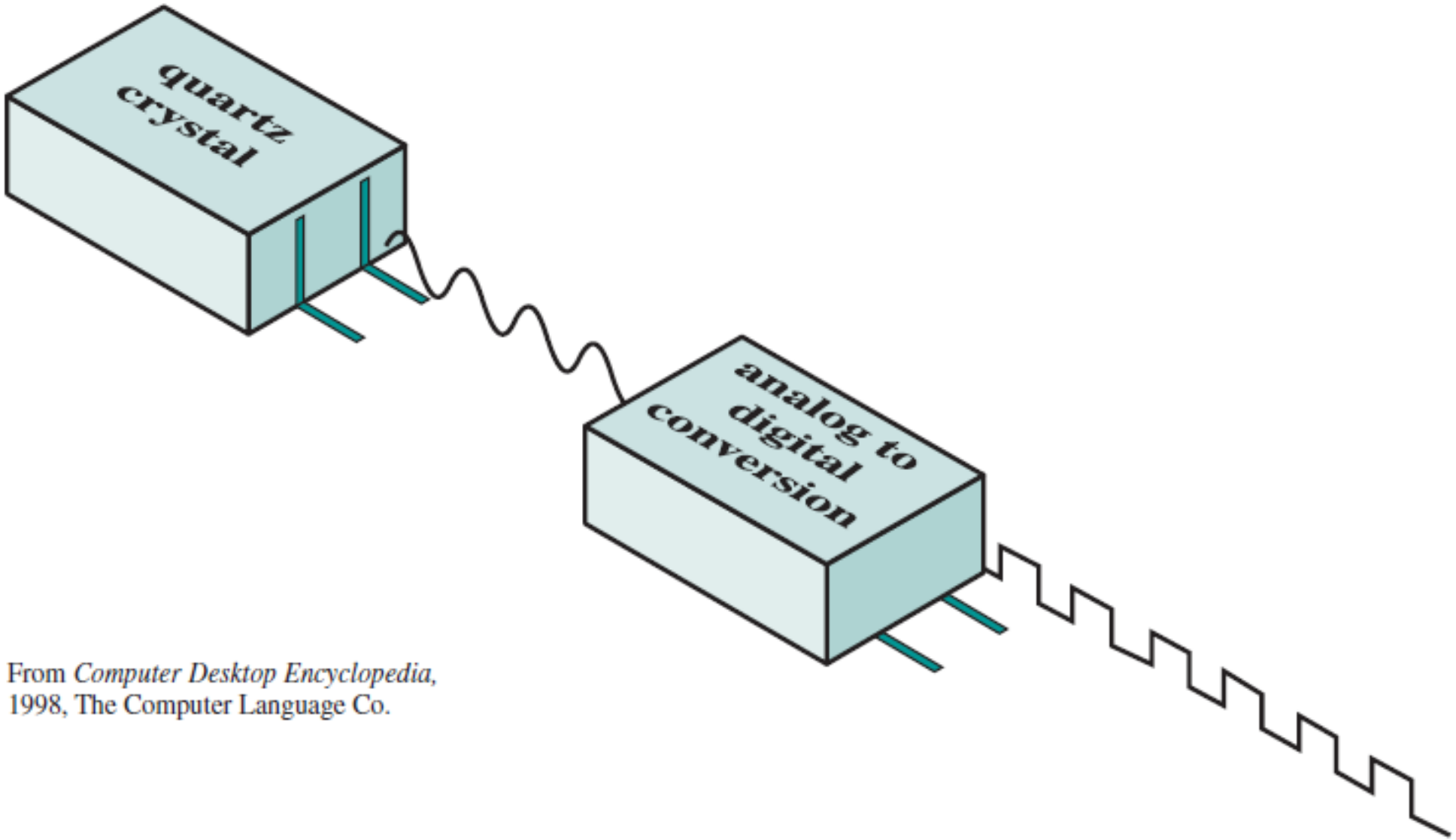
Performance Assessment

- In evaluating processor hardware and setting requirements for new systems, performance is one of the key parameters to consider, along with cost, size, security, reliability, and, in some cases, power consumption.
- It is difficult to make meaningful performance comparisons among different processors, even among processors in the same family. Raw speed is far less important than how a processor performs when executing a given application.
- Unfortunately, application performance depends not just on the raw speed of the processor but also on the instruction set, choice of implementation language, efficiency of the compiler, and skill of the programming done to implement the application.

Clock Speed and Instructions per Second

The System Clock

- Operations performed by a processor, such as fetching an instruction, decoding the instruction, performing an arithmetic operation, and soon, are governed by a system clock.
- All operations begin with the pulse of the clock.
- The speed of a processor is dictated by the pulse frequency produced by the clock, measured in cycles per second, or Hertz (Hz).
- Typically, clock signals are generated by a quartz crystal, which generates a constant signal wave while power is applied. This wave is converted into a digital voltage pulse stream that is provided in a constant flow to the processor circuitry.



From *Computer Desktop Encyclopedia*, 1998, The Computer Language Co.

- For example, a 1-GHz processor receives 1 billion pulses per second.
- The rate of pulses is known as the **clock rate**, or **clock speed**. One increment, or pulse, of the clock is referred to as a **clock cycle**, or a **clock tick**. The time between pulses is the **cycle time**.
- **The clock rate is not arbitrary**, but must be appropriate for the physical layout of the processor. Actions in the processor require signals to be sent from one processor element to another. When a signal is placed on a line inside the processor, it takes some finite amount of time for the voltage levels to settle down so that an accurate value (1 or 0) is available.
- Furthermore, depending on the physical layout of the processor circuits, some signals may change more rapidly than others. Thus, operations must be synchronized and paced so that the proper electrical signal(voltage) values are available for each operation.

- The execution of an instruction involves a number of discrete steps, such as fetching the instruction from memory, decoding the various portions of the instruction, loading and storing data, and performing arithmetic and logical operations.
- Thus, most instructions on most processors require multiple clock cycles to complete. Some instructions may take only a few cycles, while others require dozens.
- In addition, when pipelining is used, multiple instructions are being executed simultaneously.
- Thus, a straight comparison of clock speeds on different processors does not tell the whole story about performance.

Instruction Execution Rate

- A processor is driven by a clock with a constant frequency *f* or, equivalently, a constant cycle time t, where t = 1/*f*.
- Define the instruction count, *Ic*, for a program as the number of machine instructions executed for that program until it runs to completion or for some defined time interval.
- Note that this is the number of instruction executions, not the number of instructions in the object code of the program.
  - Think about loops.

- An important parameter is the average cycles per instruction (*CPI*) for a program. If all instructions required the same number of clock cycles, then *CPI* would be a constant value for a processor.

- However, on any give processor, the number of clock cycles required varies for different types of instructions, such as load, store, branch, and so on. Let CPI_i be the number of cycles required for instruction type i and I_i be the number of executed instructions of type i for a given program.

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

- The processor time *T* needed to execute a given program can be expressed as

$$T = I_c \times CPI \times \tau$$

- A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS),referred to as the **MIPS rate**. We can express the MIPS rate in terms of the clock rate and *CPI* as follows:

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

- For example, consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the CPI for each instruction type are given below based on the result of a program trace experiment:

Instruction Type	CPI	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

- The average CPI when the program is executed on a uniprocessor with the above trace results is *CPI* = 0.6 + (2x 0.18) + (4 x0.12) + (8 x0.1) = 2.24.
- The corresponding MIPS rate is $(400 \times 10^6)/(2.24 \times 10^6) \approx 178$.

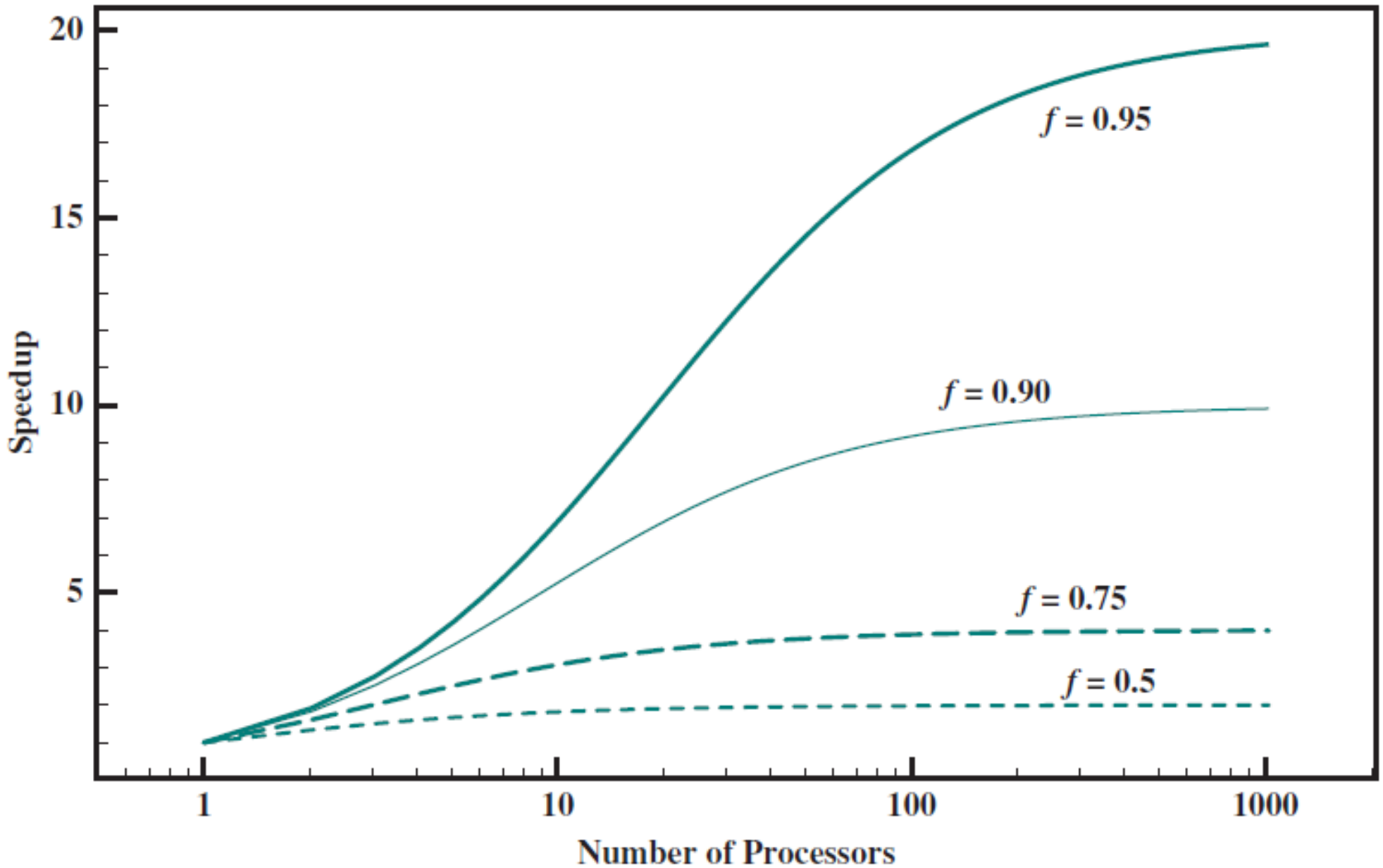
Amdahl’s Law

- When considering system performance, computer system designers look for ways to improve performance by improvement in technology or change in design.
- Examples include the use of parallel processors, the use of a memory cache hierarchy, and speedup in memory access time and I/O transfer rate due to technology improvements.
- In all of these cases, it is important to note that a speedup in one aspect of the technology or design does not result in a corresponding improvement in performance. This limitation is succinctly expressed by Amdahl’s law.

- Consider a program running on a single processor such that a fraction(1 –*f*) of the execution time involves code that is inherently serial and a fraction *f* that involves code that is infinitely parallelizable with no scheduling overhead. Let *T* be the total execution time of the program using a single processor. Then the speedup using a parallel processor with *N* processors that fully exploits the parallel portion of the program is as follows:

$$\text{Speedup} = \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on } N \text{ parallel processors}}$$

$$= \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$


20
15
10
5
Speedup
f = 0.95
f = 0.90
f = 0.75
f = 0.5
1
10
100
1000
Number of Processors

Two important conclusions can be drawn:

1. When *f* is small, the use of parallel processors has little effect.
2. As *N* approaches infinity, speedup is bound by 1/(1 −*f*), so that there are diminishing returns for using more processors.

- Amdahl’s law can be generalized to evaluate any design or technical improvement in a computer system. Consider any enhancement to a feature of a system that results in a speedup. The speedup can be expressed as

$$\text{Speedup} = \frac{\text{Performance after enhancement}}{\text{Performance before enhancement}} = \frac{\text{Execution time before enhancement}}{\text{Execution time after enhancement}}$$