

*David M. Keil*  
*Framingham State University*

# 4. Computer hardware

1. Introduction and motivation
2. Digital representation of information
3. The binary system of numerals
4. Digital computer architecture
5. I/O and operating systems

*Reading:* Evans et al, pp. 36-47, Chs. 2, 6, 9;  
Handouts (see syllabus)

David Keil Introduction to Information Technology 1/12 1

## Inquiry

- Does a computer work like a brain?
- Why does my phone boot quickly and my laptop boot slowly?

David Keil Introduction to Information Technology 1/12 2

## Objectives

- 4a. Recognize and use the basic terminology of computer hardware
- 4b. Manipulate binary numerals and describe their applications
- 4c. Describe what an assembler or machine-language program does

## 1. Introduction and motivation

*Q:* How does it help a user to study hardware?

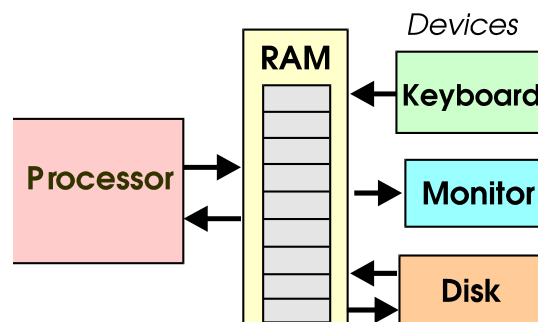
*Some answers:*

- Hardware changes are enabling new communication devices with new features
- Computers process information differently from humans due to different underlying architecture
- To understand representation of data, e.g., “bit,” “byte,” “gigabyte,” “pixel,” “256 Mb”

## IT Hardware categories

- General-purpose computers (PC, Mac, etc.)
- Standalone stored-program I/O and communication devices (cell phones, Blackberries, PDAs)
- Programmable entertainment devices (video game consoles, Ipods)
- I/O devices (printers, scanners, keyboards)
- Embedded devices (modems, fuel injectors, LCD watches)

## Architecture of a computer



- Data flows as shown by arrows, under control of processor
- Processor contains *registers*

## History of computers

- *Mechanical*: Jacquard punch-card loom, Pascal's calculator, Hollerith punch-card census tabulation
- *Vacuum-tube* (1939-1951): Atanasoff-Berry Computer; Harvard Mark I, ENIAC, UNIVAC
- *Transistor* (1945): second generation electronic computers
- *Integrated circuits* (1950): third generation
- *Microprocessors* (1971): fourth generation, VLSI

## History of personal computers

- 1975: Altair
- 1977-1980: Apple, Radio Shack
- 1978: Visicalc spreadsheet, precursor to Lotus 1-2-3, Excel
- 1981: IBM PC; later cloned
- 1984: Apple Macintosh, with GUI, mouse
- 1985-: Windows
- 1993: Web browser Mosaic, explosion of use of Internet

## 2. Digital representation of information

- In computational or IT devices and processes, all data is *digital*, represented by *symbols*, e.g., 0 and 1
- Analog data is on a *continuum* (e.g., sound waves)
- Analog data may be *digitized*, converted to bits, e.g., by sound card
- Digital output may be converted to analog, e.g., by a sound card
- *Information*: the *meaning* of data

## Silicon vs. neurons

- The brain stores information in an analog way
- Neurons respond to the strength of electrical impulses by *firing* (emitting an impulse)
- Memory and processors in computers work with *bits* (binary digits)
- Processors work in *small* steps; most computers have two or four processors
- The brain is massively *parallel* whereas digital computing is *serial* today
- *Emotions* currently play little role in computing systems, play a large role in the brain

### Examples of digital and analog data

- Symbols, such as words and numerals, are digital
- Real-world images and sounds are analog
- A ruler is an analog device
- A calculator is digital
- Output of a DVD reader is a digital signal later converted to analog signal and image
- Output of a digital camera is analog; image is stored digitally
- Output of a computer printing process is a digital bit stream to the printer

### Binary data

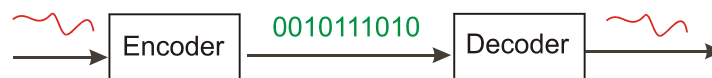
- All data is stored and communicated in IT as binary digits (bits)
- *Bit*: a single truth value or on/off switch (0, 1)
- *Byte*: 8 bits, capable of storing a small number or a character
- *Word*: 32 or 64 bits, the size value a processor can manipulate in one operation
- Access is *sequential* (e.g., I/O from ports) or *random* (e.g., RAM)

## Digitizing information

- *Definition*: using discrete symbols to represent continuous information
- *Discrete*: requires detecting presence/absence (true/false)
- *Bit patterns* are arbitrary abstractions that stand for other things
- *Example*: numeric digits for phone dialing
- Digitizing must preserve *meaning* (information content) as the *form* of data is altered

## Any information is digitally encodable

- *Bias-Free Universal Medium Principle*: “Bits can represent all discrete information; bits have no inherent meaning” (L. Snyder, Ch. 11)
- Analog information, e.g., in wave or sensor-supplied form, can be digitized by sampling or other methods such as OCR
- Meaning of bits depends on interpretation set by an application



- *Example*: TCP/IP packet stores *any* data available over Internet

## Digital encoding of color image data

- *Color*: RGB monitor has 3 color elements per pixel, each of some intensity, e.g., 0 to 255 (0 to  $2^8-1$ ) for 8-bit
- Computing on a representation: examples are tinting, boosting highlights on a photo
- Printers, cathode-ray-tube (CRT) and liquid-crystal diode (LCD) displays represent colors differently

## Digital encoding of sound

- *Digitizing sound* includes sampling waves to convert analog to digital (bit) form
- *Sound card* converts analog to digital (microphone) or digital to analog (speaker)
- File formats: *.wav, mp3, ...*
- *Compression* is a concern: reduce number of bits by using patterns

### 3. The binary system of numerals

- Appropriate for two-state devices
- Is the form in which all information is represented (numeric, text, graphical, sound)
- Uses two digits rather than ten
- Like decimal, uses place values
- Conversion to decimal
- Decimal-to-binary conversion
- Binary addition and subtraction

### Hardware representation of numbers

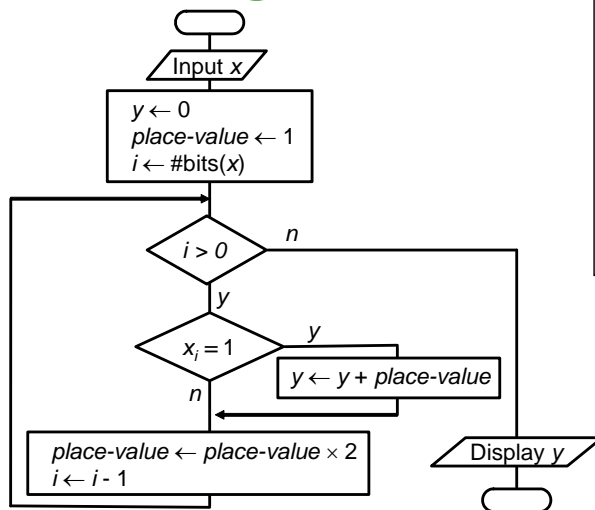
- Values are stored as sequences of *bits*
- One bit can store either of two values: 0, 1
- Two bits can store 4 different values
- Four bits can store  $2^4 = 16$  different values
- One byte is eight bits; a byte can store  $2^8 = 256$  different values, e.g., 0 .. 255
- On a 32-bit computer, a register can store up to  $2^{32} \approx 4$  billion different values
- In general,  $k$  bits of storage store  $2^k$  different values

## Binary-to-decimal conversion

- Add the powers of 2 represented by binary digits of the numeral
- The maximum value of an  $n$ -bit binary numeral is  $2^n - 1$
- An *algorithm* for conversion is pictured:

$$\begin{array}{r}
 11010_2 \\
 \begin{array}{l}
 \rightarrow 0 \times 2^0 = 0 \\
 \rightarrow 1 \times 2^1 = 2 \\
 \rightarrow 0 \times 2^2 = 0 \\
 \rightarrow 1 \times 2^3 = 8 \\
 \rightarrow 1 \times 2^4 = 16 \\
 \hline
 26_{10}
 \end{array}
 \end{array}$$

## Binary-to-decimal algorithm



```

Input x
y ← 0
pv ← 1
i ← #bits(x)
While i > 0
  If x_i = 1
    y ← y + pv
  pv ← pv × 2
  i ← i - 1
Display y
    
```

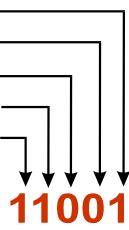
Note  $x_i$  is the  $i$ th bit of string  $x$ .  
 Example: If  $x = '10'$  and  $i = 2$ , then  $x_i = '0'$

## Decimal-to-binary conversion

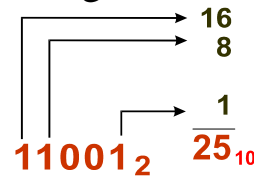
*Two methods:*

(a) Repeatedly divide decimal numeral by 2, jotting down remainder, right to left

$$\begin{array}{l} 25 \div 2 = 12 \text{ R } 1 \\ 12 \div 2 = 6 \text{ R } 0 \\ 6 \div 2 = 3 \text{ R } 0 \\ 3 \div 2 = 1 \text{ R } 1 \\ 1 \div 2 = 0 \text{ R } 1 \end{array}$$






(b) Find the set of powers of 2 that add up to the decimal value; record each as a binary 1, left to right



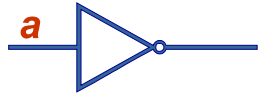
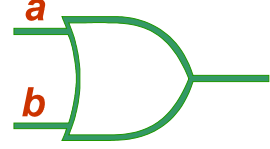
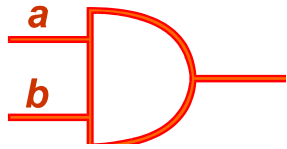
## 4. Digital computer architecture

- *Fundamental units:* Gates, flip-flops
- *Development of technology:* Relays, vacuum tubes, transistors, integrated circuits
- *Processor (CPU):* Consists of control unit and arithmetic logic unit
- *Memory:* ROM (read-only memory), RAM (random-access: addressable, volatile)
- *Peripherals (I/O):* Monitor; keyboard; mouse; disk; flash memory; DVD; sound card
- *Ports (e.g., USB, Ethernet):* channels used to communicate with peripherals and with network

## Logic gates

- Used to manipulate binary data
- 1 or 2 bit input, 1-bit output
- Specified using truth tables
- NOT (negation) 
- AND (conjunction) 
- OR (disjunction) 
- Used as components of more complex circuits: adders, etc.

## Gates and truth tables

<i>Gate</i>	<i>Schematic</i>	<i>Truth table</i>															
<b>NOT</b>		<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>not a</i></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> </tr> </tbody> </table>	<i>a</i>	<i>not a</i>	1	0	0	1									
<i>a</i>	<i>not a</i>																
1	0																
0	1																
<b>OR</b>		<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>a or b</i></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	<i>a</i>	<i>b</i>	<i>a or b</i>	0	0	0	0	1	1	1	0	1	1	1	1
<i>a</i>	<i>b</i>	<i>a or b</i>															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
<b>AND</b>		<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>a and b</i></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	<i>a</i>	<i>b</i>	<i>a and b</i>	0	0	0	0	1	0	1	0	0	1	1	1
<i>a</i>	<i>b</i>	<i>a and b</i>															
0	0	0															
0	1	0															
1	0	0															
1	1	1															

## Data storage

- Registers are within processor; are high-speed
- Memory is
  - *Cache* (high speed within processor)
  - *Read-only* (e.g., programs that help the computer start up) or
  - *Random-access* (RAM, electronic speed) or
  - *Secondary* (e.g., hard disks; a mechanical speed, slow)
- Memory sticks are RAM that appears to the user as a hard disk
- RAM is accessible by numeric *addresses*

## General-purpose computers

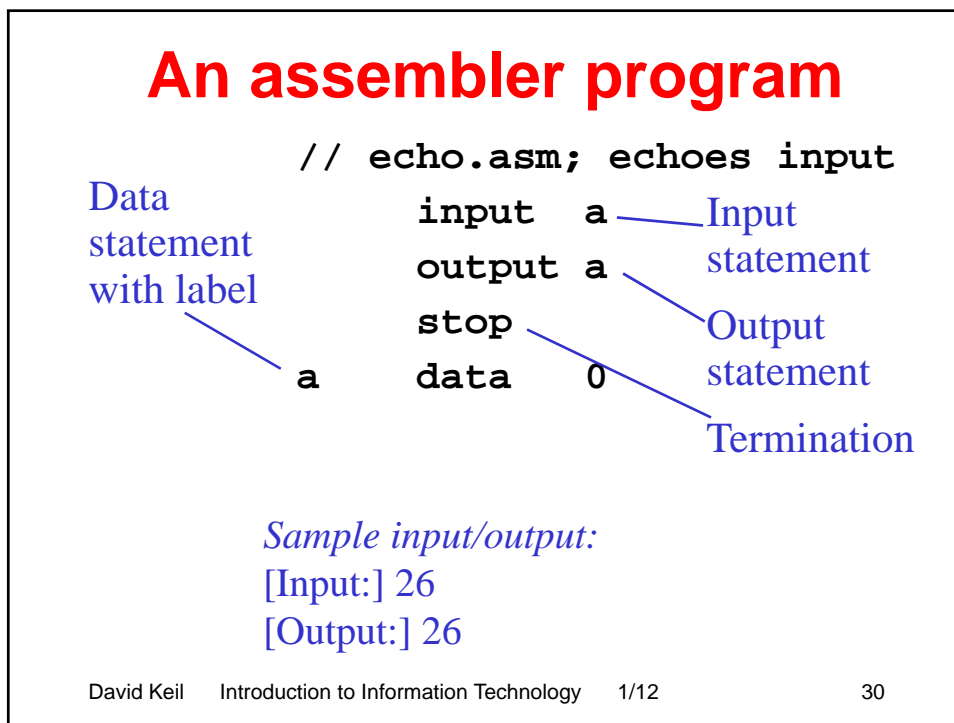
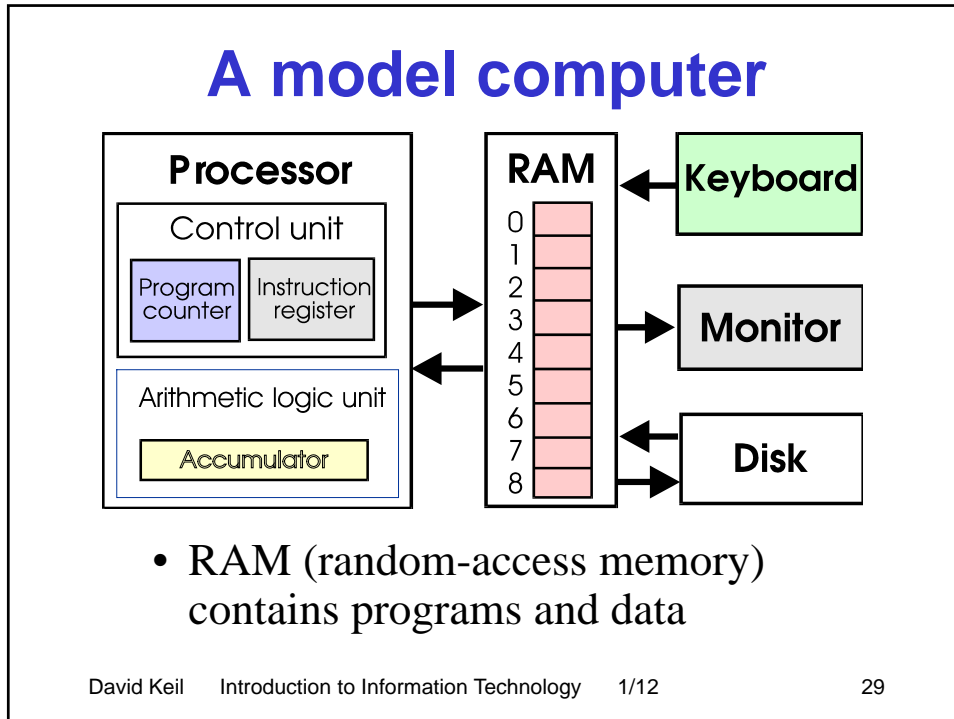
- In this course, we will use “computer” to refer to *general-purpose*, or *stored-program*, devices.
- iPods and cell phones may be *programmable*, but not in the general-purpose sense
- *Program*: A series of instructions to enable a computer to solve a problem.
- Kinds of computers: desktops, notebooks, handhelds, tablets, mainframes, supercomputers
- Computers may be *serial*, or *parallel* (multiprocessor)

## Program execution

- Processor runs a *fetch-execute* cycle
  - repeat to obtain instruction from consecutive locations in RAM
  - then carry out instruction, updating counter
- Instruction format: *opcode/mnemonic; operand*
- *Registers* (storage locations on processor):
  - Program Counter
  - Instruction Register
  - data registers (e.g., Accumulator)
- Instructions may *transform* or *copy* data

## Input/output

- I/O is under control of the processor
- I/O *peripherals* interface with RAM
- In our model, keyboard input and screen output are implemented by instructions
- In actual computers
  - I/O instructions are much more complex
  - *Device drivers* are device-specific software interfaces
  - *External storage* (HD, CD, flash): low-speed, nonvolatile



## The fetch-execute cycle

```
PC ← 0
Repeat
  IR ← instruction at MEM(PC)
  PC ← PC + 1
  Execute operation in IR
until IR = STOP
```

- *PC* = *Program Counter* register
- *IR* = *Instruction Register*
- $\leftarrow$  = “gets” (variable assignment)

## Assembler and machine languages

- A *machine language*, expressed in binary, is the set of instructions that a processor responds to
- An *assembler language* is expressed as *mnemonics* (abbreviations for instructions) and *labels* for data and lines of instruction code
- Machine and assembler are not human-friendly
- Each processor (Pentium, G3, etc.) is designed with its own machine language

## Model processor's language

- *input*: waits for user to key a number; stores that number at location specified in operand
- *output*: displays on screen the number stored at the operand address
- *load*: copies to ACC the value stored at the operand address
- *store*: copies a value from ACC to a RAM address
- *add (sub)*: adds (subtracts) value at operand address to value in ACC, storing result in ACC

## Variables

- A *variable* is a named memory location that stores a value
- Variables in programs must be declared:
  - `sum data 0` (assembler)
  - `int sum;` (JavaScript)
- Variables are assigned values
  - `load 2` (assembler)
  - `store sum` (sum ← 2)

## A program to add

```
// ADD.ASM: Displays sum of 2 inputs
    input    input1
    input    input2
    load     input1
    add      input2
    store    sum
    print    sum
    stop

input1 data 0
input2 data 0
sum     data 0
```

} *sum ← input1 + input2*

- *input* copies data from keyboard to RAM
- *load* and *add* alter contents of ACC
- *store* copies from ACC to memory

## Processor-simulator software

- See D. Keil web site or 63.120 Blackboard site for installer, *asm\_setup.exe*
- This software enables the user to create and edit programs in a 10-instruction *assembler language*
- ...and to step through them visually on a simulated processor
- Purpose: to help understand the fetch-execute cycle, processing of data in the CPU, and motion of data between CPU and RAM

## Microprocessor features

- *Word size*: number of bits processed in a register (in 2006, usually 32 or 64)
- *Cache*: high-speed memory in the processor mirroring some addresses in RAM  
**Registers ↔ Cache ↔ RAM ↔ Hard disk**
- *Pipelining*: Overlapping the steps to execute two or more instructions (like conveyor belt)
- *Parallel* processing (e.g., dual core) executes two or more instructions simultaneously on different CPUs
- *Clock*: synchronizes system; speed is measured in MHz (millions of cycles/sec)

## 5. I/O and operating systems

- The OS is *system software* that runs whenever the computer power is on
- The OS
  - Provides services, such as file management, to other programs and to the user
  - Loads and runs application programs
  - Manages device I/O via *driver* software
  - Manages sharing of memory

## Expansion ports

- Support communication between RAM and peripherals under processor control
- Kinds of connection (most are *ports*, on back of PC):
  - Keyboard
  - Mouse
  - USB (general-purpose)
  - Speaker, microphone
  - Video
  - Ethernet network connector
  - Modem (phone jack)
  - Wireless Ethernet (not a port)

## Buses

- A *bus* is a connection that transfers data between the CPU, RAM, and peripherals
- *Local or front-side bus* connects CPU and RAM
- *Expansion bus* enables adding of cards
- *Bus clock speed*: now about 1 GHz (1 billion cycles/sec)
- *Bus width*: number of bits transferred, in parallel

## Display technologies

- *Kinds:*
  - *LCD* (liquid crystal diode): flat screen, more expensive
  - *Plasma*: manipulates miniature fluorescent lights; expensive
  - *LED* (light emitting diode)
- *Screen resolution*: Number of pixels horizontal by vertical; e.g., 1024 × 768

## Hard disk drives

- Hard drives store bits as magnetic orientations of iron oxide particles
- Drive has multiple spinning disks, each accessible by a *read/write head*
- Disk is divided into *tracks* (concentric circles of data locations), *sectors* (pie slices)
- Access time is determined by *seek* (time for head to reach track) and *latency* (time to wait for sector to spin into place)

## Device drivers

- Peripherals' interactions with CPU and RAM are managed by *device driver* software
- Device drivers come with hardware, are available online from manufacturer, or are standard part of operating system (*Plug and Play*)

## Kinds of data storage

- *Registers* in processor enable operations on data
- *Cache memory* data is easily moved to and from registers
- *RAM* (random-access memory) is volatile
- *ROM* (read-only memory) is non-volatile, stores operating system's BIOS (basic I/O system)
- *CMOS* requires battery power, stores long-term but updatable specs
- *Hard disk* enables long-term storage
- *Flash memory* sticks are portable and non-volatile
- *CDROM and DVD* use optical storage technology

## Power-saving methods

- *Sleep* (standby): documents being edited stay in RAM; hard disk and monitor turn off
- *Hibernate*: documents are saved to disk while power is down
- To wake up PC, press a key or move mouse
- PC may be set to power-off monitor and hard disk after a fixed time of inactivity
- *Windows*: see Control Panel, *Power Options*

## Concurrent processing

- *Pipelined execution* enables execution of one instruction in parallel to fetch of next
- CPU may be augmented by *graphical processing unit* (GPU) to handle screen graphics
- *Dual- or multi-core processing* may allocate different processes or threads (e.g., OS, app, and virus scan) to multiple processing units on one or more chips

## Multitasking

- An OS enables multiple programs to execute concurrently
- Processor devotes time slices to each program or process in round robin fashion
- Bar at bottom of Windows screen shows some programs running
- Windows Task Manager shows all processes running at a particular time
- Programs may run in background

## Parallel and distributed processing

- *Parallel processing* with many or thousands of processors is used for weather modeling and other problems that lend themselves to parallel program execution
- *Distributed computing* coordinates networked computers on a single problem

## Cell-phone technology

- Cell phones are IT, have processor, run apps, but are not general-purpose computers
- A *cell* is a geographic area with a tower and with transmit/receive hardware
- Voice sound (analog data) is converted to digital form by a chip
- Digital signal processor compresses bits to be transmitted in packets via radio waves, decompresses at receiving end for conversion to sound

## Portable media players

- Play music in MP3 and other compressed formats, video in AAC and other formats
- Geographical Positioning System (GPS) devices help navigate
- Most storage is flash memory; some PMPs subscribe to and download *podcasts* (media files from same source)

## Hardware concepts

address	digitize	machine	processor
arithmetic logic unit	discrete value	instruction	program counter
assembler language	DVD	machine	RAM
binary numeral	embedded device	language	random access
bit	external storage	memory	register
bitmap	fetch-execute cycle	microprocessor	representation of data
Boolean	file	mnemonic	ROM
byte	flash memory	monitor	screen
cache	gate	operand	sequential access
CDROM	general-purpose computer	output	storage
central processing unit (CPU)	giga-, kilo-, mega-	parallel computation	storage metaphor
computer	hard disk	path name	stored-program device
control unit	input	peripheral	word size
crash	integrated circuit	pipelining	
CRT	LCD	pixel	
		port	

## References

- A. Evans, K. Martin, M. A. Poatsy. *Go! Technology in Action*, 6<sup>th</sup> ed. Prentice Hall, 2010.
- R. Johnson and D. Keil. *Introduction to Computer Programming Using Turbo Pascal*. Thompson, 1995.
- J. Parsons, D. Oja. *Computer Concepts*, 9<sup>th</sup> ed. Thomson, 2007.
- L. Snyder. *Fluency with Information Technology: Skills, Concepts, and Capabilities*. Addison Wesley, 2006.