Computer Basics

TOPICS

- Computer Organization
- Data Representation
- Program Execution
- Computer Languages

Hardware and Software

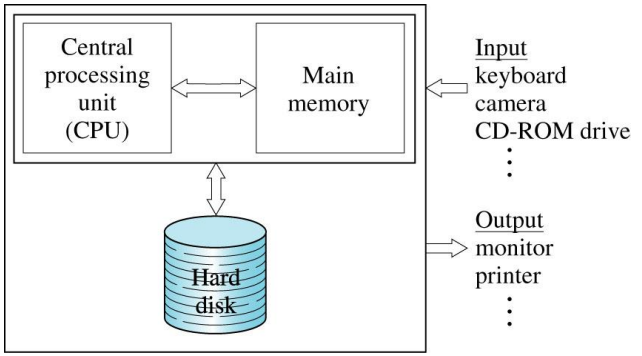
- Computer systems consist of hardware and software.
  - Hardware includes the tangible parts of computer systems.
  - Software includes programs - sets of instructions for the computer to follow.
- Familiarity with hardware basics helps us understand software.

Computer Organization

Hardware

- The majority of modern computers have similar components including:
  - Input devices (keyboard, mouse, etc.)
  - Output devices (display screen, printer, etc.)
  - Central Processing Unit (CPU) or processor
  - Main and auxiliary (secondary) memory

Computer Architecture



Processors

- The processor is also called the CPU (Central Processing Unit)
- Processes a relatively simple set of instructions.
- Programs must be translated into the specific instruction set.
- The power of computing comes from speed and program intricacy.

Main memory

- Working memory used to store
  - set of instructions for current program
  - data the program is using
  - results of intermediate calculations
- Now measured in gigabytes
  - e.g. 8 gigabytes of RAM
  - RAM is short for random access memory
  - A byte is a quantity of memory

Auxiliary Memory

- Also called secondary memory
- Disk drives, optical drives (CD/DVD), flash drives, etc.
- More or less permanent (nonvolatile)
- Usually measured in gigabytes
  - e.g. 512 gigabyte hard drive

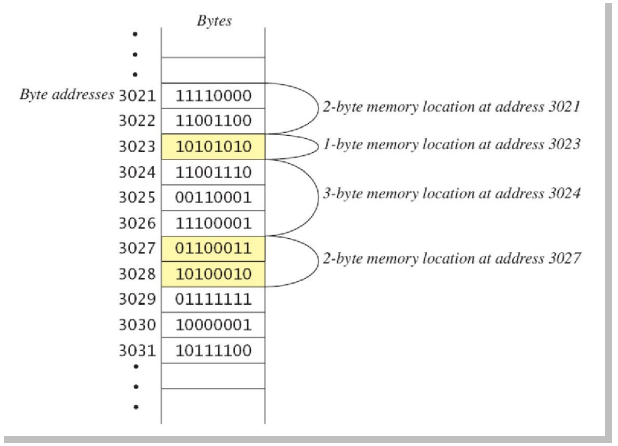
Data Representation

- Computers store data as binary numbers, not decimal!
- Numbers can be used to represent almost any type of data:
  - Characters (e.g. ‘a’) are represented by numbers, strings (e.g. “foo”) are just groups of characters
  - Pictures are represented by dividing them into picture elements known as **pixels**
  - Video images or animations are represented by placing several pictures one after another
  - Sounds are represented by sampling the pressure wave at regular intervals

Bits, Bytes and Words

- **Bit**: 0 or 1
- **Byte**: sequence of eight bits: 00101110
- **Word**: sequence of 2, 4 or 8 bytes
- To computer, everything is a sequence of bits!
- *If we have 4 bits, how many things can we represent?*

Main Memory



Bit Permutations

1 bit	2 bits	3 bits	4 bits	
0	00	000	0000	1000
1	01	001	0001	1001
	10	010	0010	1010
	11	011	0011	1011
		100	0100	1100
		101	0101	1101
		110	0110	1110
		111	0111	1111

Each additional bit doubles the number of possible permutations.

Bit Permutations

- Each permutation can represent a particular item
- There are 2^N permutations of N bits
- N bits are needed to represent 2^N unique items

How many items can be represented by

1 bit ?	$2^1 = 2$ items
2 bits ?	$2^2 = 4$ items
3 bits ?	$2^3 = 8$ items
4 bits ?	$2^4 = 16$ items
5 bits ?	$2^5 = 32$ items

How many items can be represented by 8 bits? 16 bits? 32 bits? 64 bits?

Positional Representation

- Decimal number representation:
  - What does 256 mean?

$2 * 100 + 5 * 10 + 6$

$2 * 10^2 + 5 * 10^1 + 6 * 10^0$

- Binary number representation:
  - What does 10010 mean?

$1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$

$1 * 16 + 0 * 8 + 0 * 4 + 1 * 2 + 0 * 0 = 18$

- Let's count 0 to 15 in binary.
  - *Add 1 each time, carry just like in base 10*

Text Representation

- Remember to a computer everything is stored in a binary format
- Need to convert from characters (what is on a keyboard) to bit representation
  - ASCII: 7 bit mapping in one byte,
    - Each character maps to different value,
    - A decimal digit is also a character and has a mapping, e.g., '0' is 00110000 (48 in decimal).

ASCII

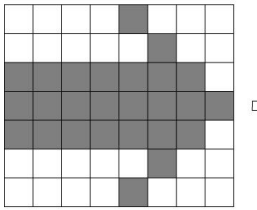
Dec	Char		Dec	Chr	Dec	Chr	Dec	Chr
0	NUL	(null)	32	Space	64	@	96	`
1	SOH	(start of heading)	33	!	65	A	97	a
2	STX	(start of text)	34	"	66	B	98	b
3	ETX	(end of text)	35	#	67	C	99	c
4	EOT	(end of transmission)	36	$	68	D	100	d
5	ENQ	(enquiry)	37	%	69	E	101	e
6	ACK	(acknowledge)	38	&	70	F	102	f
7	BEL	(bell)	39	'	71	G	103	g
8	BS	(backspace)	40	(	72	H	104	h
9	TAB	(horizontal tab)	41	)	73	I	105	i
10	LF	(NL line feed, new line)	42	*	74	J	106	j
11	VT	(vertical tab)	43	+	75	K	107	k
12	FF	(NP form feed, new page)	44	,	76	L	108	l
13	CR	(carriage return)	45	-	77	M	109	m
14	SO	(shift out)	46	.	78	N	110	n
15	SI	(shift in)	47	/	79	O	111	o
16	DLE	(data link escape)	48	0	80	P	112	p
17	DC1	(device control 1)	49	1	81	Q	113	q
18	DC2	(device control 2)	50	2	82	R	114	r
19	DC3	(device control 3)	51	3	83	S	115	s
20	DC4	(device control 4)	52	4	84	T	116	t
21	NAK	(negative acknowledge)	53	5	85	U	117	u
22	SYN	(synchronous idle)	54	6	86	V	118	v
23	ETB	(end of trans. block)	55	7	87	W	119	w
24	CAN	(cancel)	56	8	88	X	120	x
25	EM	(end of medium)	57	9	89	Y	121	y
26	SUB	(substitute)	58	:	90	Z	122	z
27	ESC	(escape)	59	;	91	[	123	{
28	FS	(file separator)	60	<	92	\	124	|
29	GS	(group separator)	61	=	93	]	125	}
30	RS	(record separator)	62	>	94	^	126	~
31	US	(unit separator)	63	?	95	_	127	DEL

Source: www.LookupTables.com

Pixels

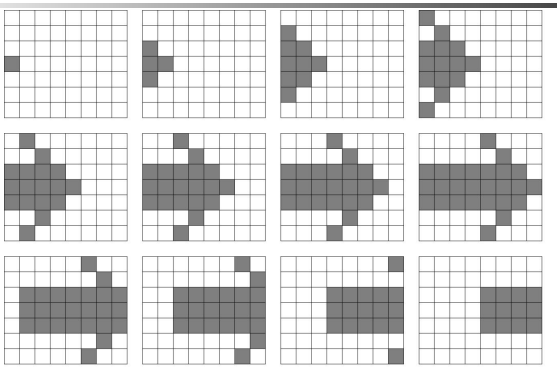
- Everything is stored as 0's and 1's
- Pictures are reduced to rectangles, or pixels

Pixels



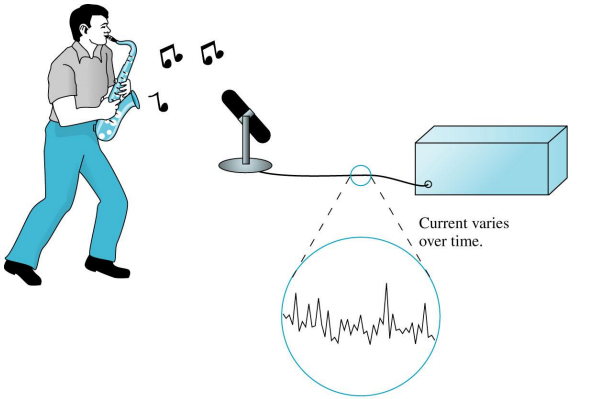
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 = 8, 4, 254, 255, 254, 8, 4
1 1 1 1 1 1 1 0
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0

Animation



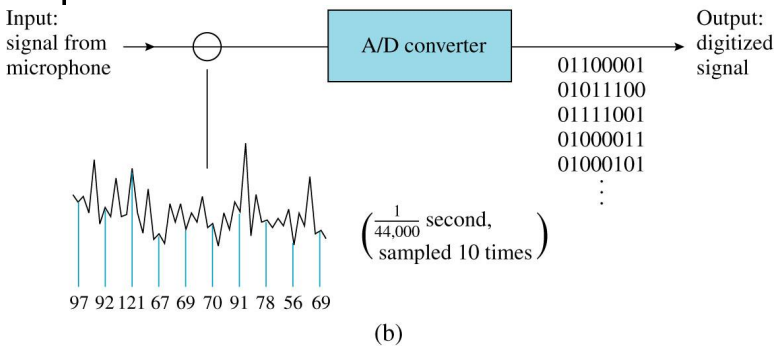
Animation is represented as a sequence of pictures. Each picture is a sequence of numbers. First picture: 0, 0, 0, 128, 0, 0, 0; Second picture: 0, 0, 128, 192, 128, 0, 0; Third picture: 0, 128, 192, 224, 192, 128, 0. Place two numbers at beginning, giving height and width of each picture: 7, 8, 0, 0, 0, 128, 0, 0, 0, 0, 0, 128, 192, 128, 0, 0, . . .

Sound/Audio



(a)

Analog to Digital Conversion



Files

- Large groups of bytes in auxiliary memory are called files.
- Files have names and extensions, managed by operating system.
- Files are organized into groups called directories or folders.
- Java programs are stored in files, and are copied to memory before running.

The Operating System

- The operating system is a supervisory program that:
  - oversees the operation of the computer
  - controls resources such as disk drives
  - retrieves and starts program for you
- Well-known operating systems:
  - Microsoft Windows, Apple Mac OS, Linux, and UNIX.

Computer Languages

- Low-Level Languages
  - Machine Code
  - Assembly Code
- High-Level Languages
  - Fortran
  - COBOL, BASIC
  - Pascal, C,
  - C++, Java
  - Perl, Python
  - R, Matlab
- Interpretation versus Compilation
- Visual Languages

Programming Languages

- High-level languages are relatively easy to use for the programmer:
  - Java, C#, C++, Python, Ruby, etc.
- Low-level languages are very complex and error prone, but computers don't understand high-level languages!
  - High-level language programs must be translated into low-level languages.

Compilers

- A compiler translates a program from a high-level language to a low-level language that the computer can run.
- You compile a program by running the compiler on the source code of the high-level program.
- Compilers produce machine or assembly-language programs called object programs.

Java Byte-Code

- The Java compiler doesn't translate a Java program into assembly or machine language for a particular computer.
- Instead, it translates a Java program into byte-code.
  - Byte-code is the machine language for a hypothetical computer (or interpreter) called the Java Virtual Machine.

Java Byte-Code

- A byte-code program is easy to translate into machine language for any particular computer, this can be done 'on-the-fly'.
- A program called an interpreter translates each byte-code instruction, executing the resulting machine-language.

Compiling, Interpreting, Running

- Use the compiler to translate the Java program into byte-code (done using the javac command).
- Use the Java virtual machine for your computer to translate each byte-code instruction into machine language.
- Eclipse tool makes all this very easy!

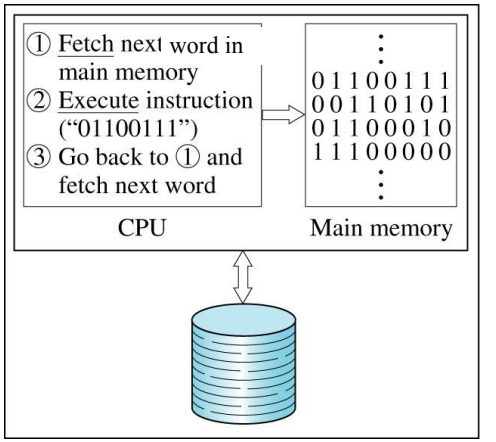
Portability

- After compiling a Java program into byte-code, the byte-code can be used on any computer with a byte-code interpreter without recompiling.
- Byte-code can be sent over the Internet and used anywhere in the world, this makes Java highly portable and thus suitable for Internet applications.

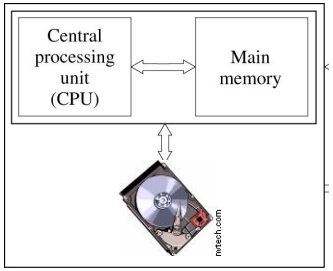
Running a Program

1. First the program and all its data are copied from the hard disk into main memory.
2. The CPU goes to the location of the program instruction and reads that word.
3. The CPU determines what action is requested by decoding its bit pattern representation.
4. The CPU performs the action, usually a math operation or memory read/write.
5. The CPU moves to the location of the next program instruction in memory and repeats the process.

Running a program another view



Instruction Execution Model



- **CPU**
  - Instruction cycle
    - Fetch instruction from main memory
    - Decode instruction
    - Fetch operands
    - Execute instruction
    - Store operands
    - Determine the next instruction to execute

Instructions

- Format: `<operation, operand, operand>`
- Operations
  - Arithmetic & logical
  - Data movement
  - Control
- Operands: locations of data and instructions
- Examples:
  - **`Add reg1 reg2`**
  - **`Move reg1 010010100101001010010100`**
  - **`Goto 110101110101110101110101`**