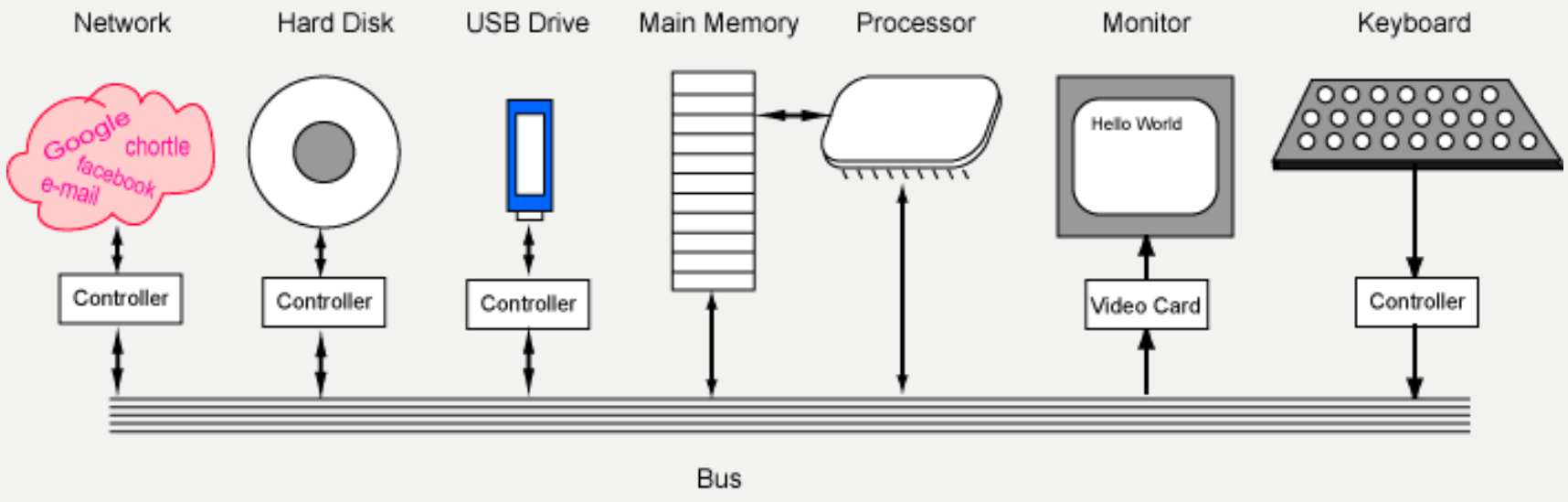
Hardware
&
Software

Hardware components

- Processor
- Main memory
- Secondary memory
- Input devices
- Output devices

Hardware components



Main Components of a Computer System

Memory

- The processor performs all the fundamental computation of the computer system. Other components contribute to the computation by doing such things as **storing data or moving data into and out** of the processor.
- A processor chip has **relatively little memory**. It has only enough memory to hold a few instructions of a program and the data they process.
- Complete programs and data sets are held in memory external to the processor. This memory is of two fundamental types: main memory, and secondary memory.
- Main memory is sometimes called **volatile** because it looses its information when power is removed.
- Secondary memory is usually **non-volatile** because it retains its information when power is removed.

Memory

- **Main memory:**
  - closely connected to the processor.
  - stored data are quickly and easily changed.
  - holds the programs and data that the processor is actively working with.
  - interacts with the processor millions of times per second.
  - needs constant electric power to keep its information.
- **Secondary memory:**
  - connected to main memory through the bus and a controller.
  - stored data are easily changed, but changes are slow compared to main memory.
  - used for long-term storage of programs and data.
  - before data and programs can be used, they must be copied from secondary memory into main memory.
  - does not need electric power to keep its information.

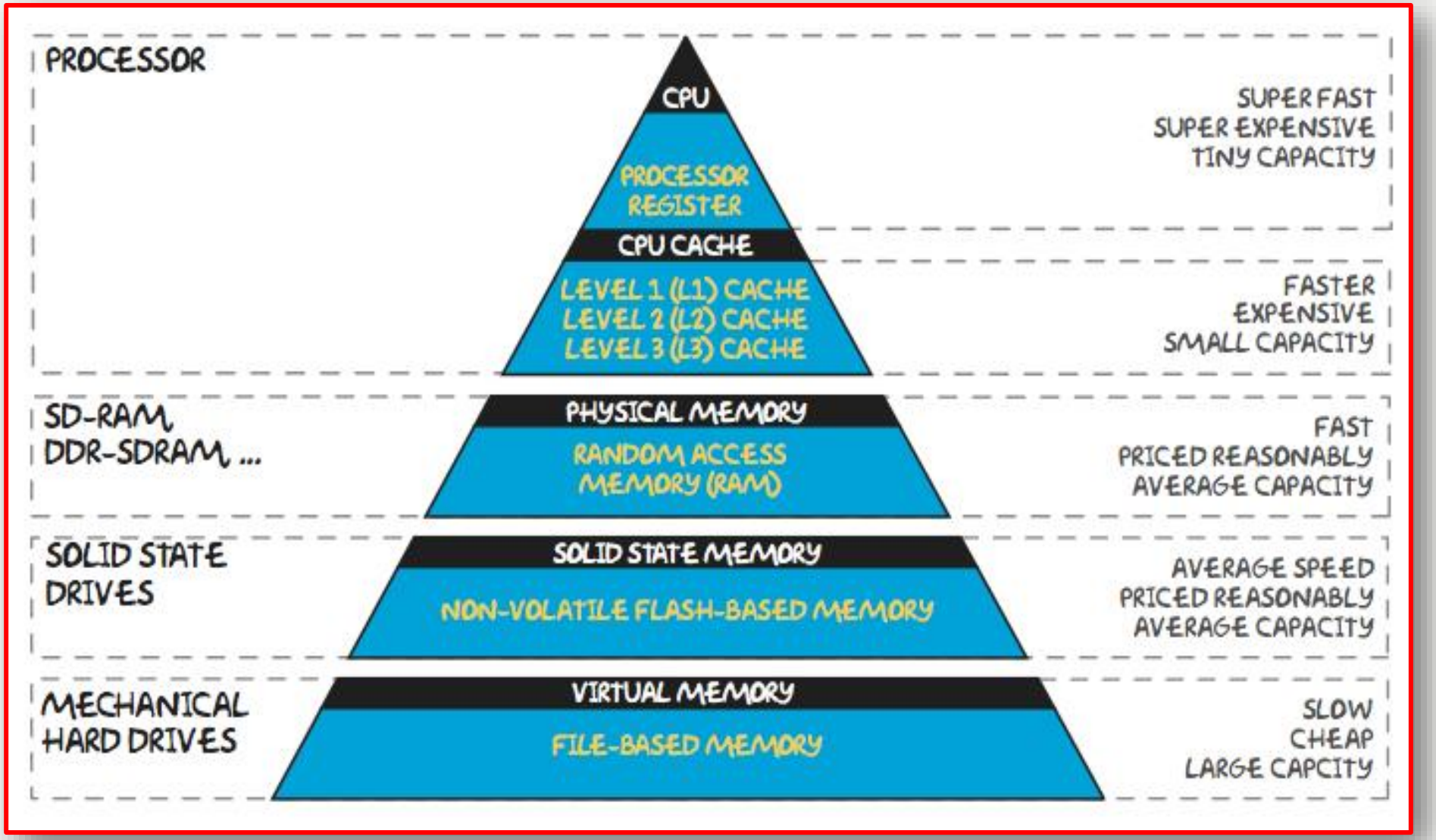
Main Memory

- Main memory is where programs and data are kept when the processor is actively using them. When programs and data become active, they are copied from secondary memory into main memory where the processor can interact with them. A copy remains in secondary memory.
- Main memory is intimately connected to the processor, so moving instructions and data into and out of the processor is very fast.
- Main memory is sometimes called **RAM.** RAM stands for **Random Access Memory**. "Random" means that the memory cells can be accessed in any order. However, properly speaking, "RAM" means the type of silicon chip used to implement main memory.
- Nothing permanent is kept in main memory. Sometimes data are placed in main memory for just a few seconds, only as long as they are needed by the processor

Secondary Memory

- Secondary memory is where programs and data are kept on a long-term basis.
  - The hard disk has enormous storage capacity compared to main memory.
  - The hard disk is used for long-term storage of programs and data.
  - Data and programs on the hard disk are organized into files.
  - A **file** is a collection of data on the disk that has a name.

- Large blocks of data are copied from disk into main memory. This operation is slow, but lots of data is copied. Then, while a program is running, the processor can quickly read and write small sections of that data in main memory. When it is done, a large block of data is written back to disk.
- Often, while the processor is computing with one block of data in main memory, the next block of data from disk is read into another section of main memory and made ready for the processor. One of the jobs of an operating system is to manage main storage and disks this way.


PROCESSOR
CPU
PROCESSOR REGISTER
CPU CACHE
LEVEL 1 (L1) CACHE
LEVEL 2 (L2) CACHE
LEVEL 3 (L3) CACHE
SUPER FAST
SUPER EXPENSIVE
TINY CAPACITY
FASTER
EXPENSIVE
SMALL CAPACITY
SD-RAM
DDR-SDRAM ...
PHYSICAL MEMORY
RANDOM ACCESS MEMORY (RAM)
FAST
PRICED REASONABLY
AVERAGE CAPACITY
SOLID STATE DRIVES
SOLID STATE MEMORY
NON-VOLATILE FLASH-BASED MEMORY
AVERAGE SPEED
PRICED REASONABLY
AVERAGE CAPACITY
MECHANICAL HARD DRIVES
VIRTUAL MEMORY
FILE-BASED MEMORY
SLOW
CHEAP
LARGE CAPCITY

Input and Output Devices

- Input and output devices allow the computer system to interact with the outside world by moving data *into* and *out of* the system. An *input device* is used to bring data into the system. Some input devices are:

Keyboard, Mouse, Microphone, Barcode reader, Graphics tablet

- An *output device* is used to send data out of the system. Some output devices are:

Monitor, Printer, Speaker

- A network interface acts as both input and output. Data flows from the network into the computer, and out of the computer into the network.

I/O

- Input/output devices are usually called **I/O** devices. They are directly connected to an electronic module attached to the motherboard called a **device controller**.
  - For example, the speakers of a multimedia computer system are directly connected to a device controller called an audio card, which in turn is plugged into a bus on the motherboard.
  - With many recent computers, the functions of a device controller are integrated with the motherboard. Some motherboards have audio, graphics, and network controllers built in.
- Often secondary memory devices like the hard disk are called I/O devices (because they move data in and out of main memory). What counts as an I/O device depends on context. To a user, an I/O device is something outside of the computer case. To a programmer, anything outside of the processor and main memory is an I/O device. To an engineer working on the design of a processor everything outside of the processor is an I/O device.

Software

- Computer software consists of both ***programs*** and ***data***. Programs consist of instructions for the processor. Data can be any information that a program needs: character data, numerical data, image data, audio data, and countless other types. The distinction between programs and data is not as clear-cut as you might think, however.
- **Fundamental Idea:** Both programs and data are saved in computer memory in the same way. The electronics of computer memory (both main memory and secondary memory) make no distinction between programs and data.
- The insight that both programs and data can be saved using the same electronic methods is an important concept in computer science. Computer systems use memory for either programs or data, as needed.

Operating Systems

- An operating system is a complex program that keeps the hardware and software components of a computer system coordinated and functioning. It is like the owner of a small shop, who keeps everything in order by attending to customers, accepting deliveries, stocking the shelves, doing the bookkeeping, and so on. The shopkeeper must promptly attend to tasks as they arise. Without the shopkeeper the shop could not function.

- Most computer systems can potentially run any of several operating systems. For example, most Intel-based computers can run either Linux or a Windows operating systems.

Starting a Program

- The user asks to run an application. This is done by clicking on an icon, making a menu choice, or by other means.
- The OS determines the name of the application.
- The OS finds the files on the hard disk where the application and its data are stored.
- The OS finds an unused section of main memory that is large enough for the application.
- The OS makes a copy of the application and its data in that section of main memory.
- The software on the hard disk is unchanged; main memory holds a copy of what is on disk.
- The OS sets up resources for the application.
- Finally, the OS starts the application running.
- As the application runs, the OS is there in the background managing resources, doing input and output for the application, and keeping everything else running.

Binary

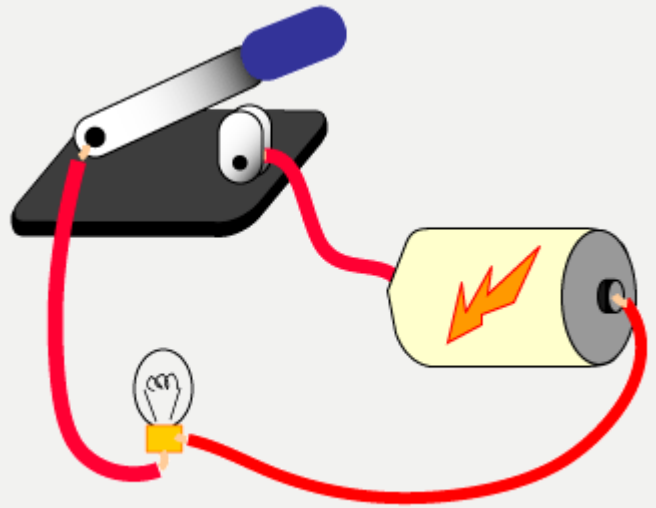
- **Binary** means "two states." The two states are sometimes called "1" and "0", or called "true" and "false", or called "on" and "off", (or other names.) The essential characteristic is that a single binary device can be in just one of two possible states.
- A **bit** is a single on/off value.
  - A good example is a toggle switch, such as a light switch. You can turn it on or off but not (in normal operation) anything else. A light switch holds one bit of information.
  - A light dimmer is not a binary device: it has many positions between off and fully on. If you want a light dimmer to be set to 25%, you must carefully adjust it.

Why Computers use Binary

1. Binary devices are Simple and easy to build.
2. Binary signals are Unambiguous (which gives them noise immunity).
3. Flawless copies can be made of binary data.
4. Anything that can be represented with some sort of pattern can be represented with patterns of bits.

Simple, easy to build

- An on/off switch is simple and easy to build. An on/off switch moves two pieces of metal together or moves them apart. A light dimmer must gradually and smoothly change the current that reaches the light. It has more components that an on/off switch and must be carefully assembled.
- An accurate dimmer (where 25% means *exactly* 25%) is even harder to build.

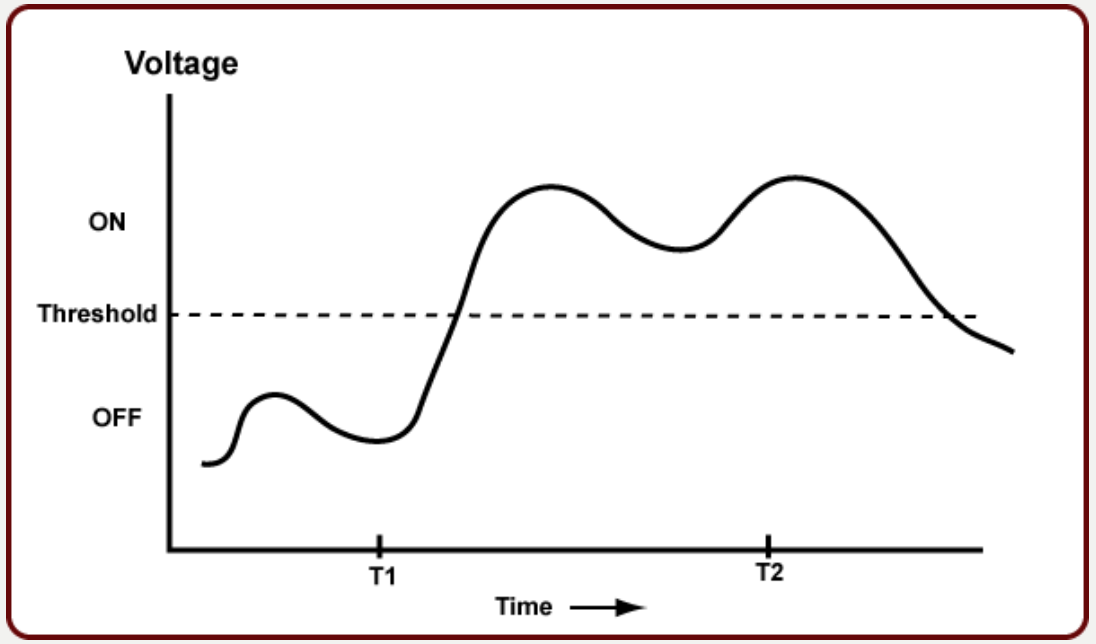


- The same is true for the tiny devices inside of a silicon chip. Silicon on/off switches are relatively easy to fabricate. The devices are cheap, small, and reliable, and millions of them fit into a small area.

Analog Signal

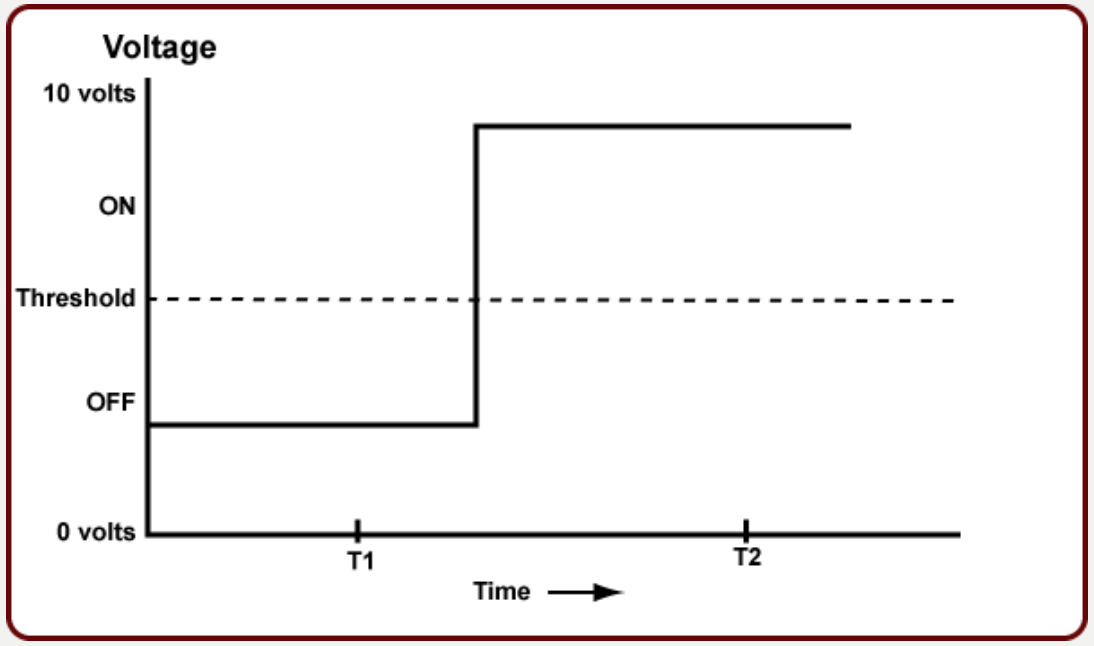
- An analog signal may continuously change in value. Its values can be anything within a range of values, and its exact value at any time is important. The graph represents an audio signal. The exact value at each time is part of the information it contains. For example, the value at time "T2" must be measured exactly.

- Now say that you are observing the voltage of a wire. It has been agreed that any voltage below a threshold will be counted as an "off" signal, and that any value above the threshold will be counted as an "on" signal.



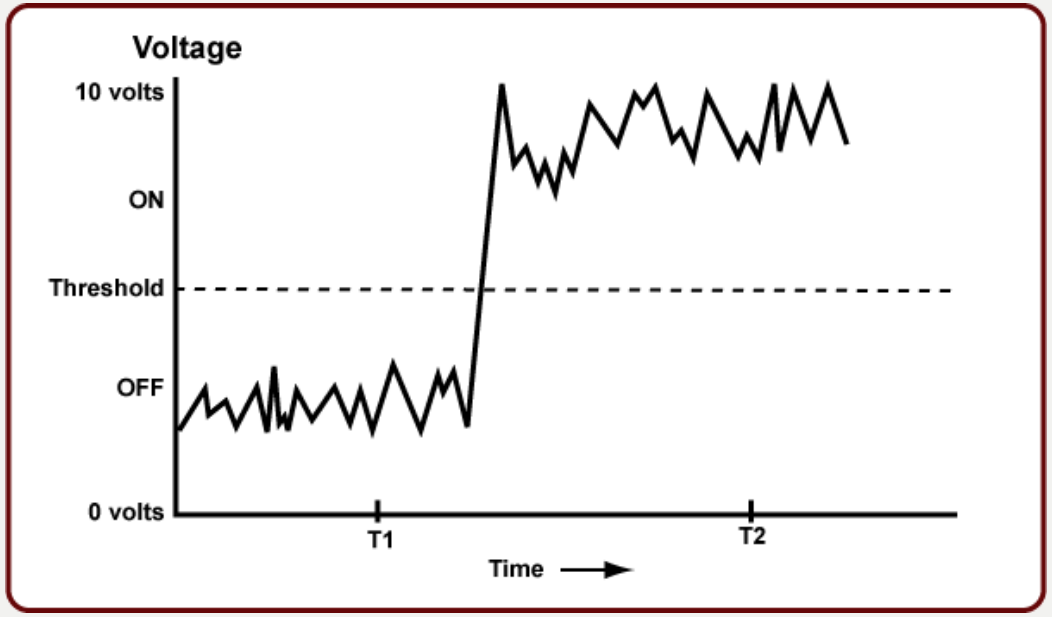
Binary Signal

- Analog signals usually continuously change their value. The information they convey is contained in the exact value at any instant.
- However, by using a threshold, a signal can represent binary data ("on/off" data). Such a signal is called **binary signal**.
- It is easy and fast to determine if a voltage is above or below a threshold. The signal in the figure is "off" at time T1 and then "on" at time T2.



Imperfect Transmission

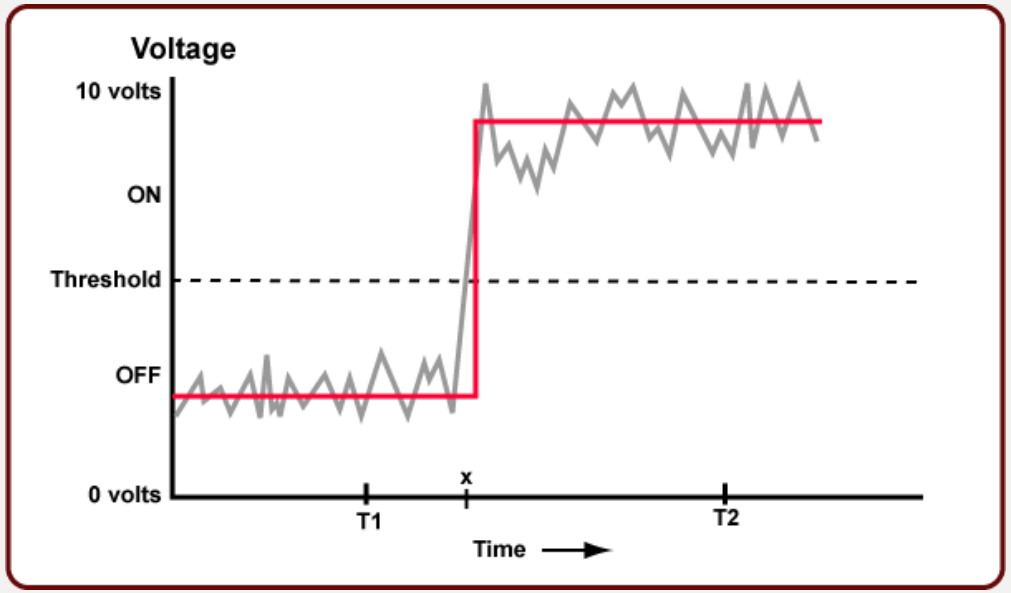
- The "ons" and "offs" of previous signal are clear. But what if the signal is sent down a long wire and someone nearby turns on a vacuum cleaner? The graph shows the signal at the other end of the wire.
- Even though the signal is noisy (at the analog level), the binary values are transmitted perfectly. You (and the electronics) can still tell that at time T1 the signal represents "off" and that at time T2 the signal represents "on" The receiving end just needs to get the binary values.
- Since only the "on" "off" information matters, the analog noise is irrelevant, and the original signal is received perfectly (so far as the binary information goes.)



Flawless copies can be made

- The receiver of the signal is only interested in the binary values. All it has to do it check if the signal is above or below the threshold. This can be done perfectly (as long as the noise is not too great.) For example, the picture shows the noisy signal with the on/off values recovered from it.

- The original signal has been recovered flawlessly. This process can occur as many times as needed with a perfect copy made each time. This is essential in a computer system, where bit patterns (patterns of one and zero, or on and off) are copied back and forth between the processor and memory millions of times a second. The copies have to be perfect



Clocks

- Digital systems are built so that the on/off (binary) value is tested only at certain points in time. This gives the wire (or other device) time to change. This is why computer systems have a **clock**. The clock generates **ticks**, which are points in time when signals may be measured. In the picture, T1 and T2 are ticks.
- The clock keeps all these time points synchronized. Faster clocks mean wires can be tested more times per second, and the whole system runs faster.
- Processor chips are often described in terms of their clock speed. Clock speed is measured in **Hertz**, where one Hertz is one clock tick per second. The symbol **MHz** means **megahertz**, a million (10^6) clock ticks per second. The symbol **GHz** means **gigahertz**, a billion (10^9) clock ticks per second.
- A 2 GHz processor checks binary values two billion times in each second. In between these times values are allowed to change and settle down. The faster a processor chip is, the more times per second values can be tested, and the more decisions per second can be made.

Representing Anything

- Since data of all kinds is stored in computer memory (main and secondary) using the same electronic methods, this means that endless perfect copies can be made of any type of data or program.
- Any system of symbols can be translated into bit patterns. An example is how English characters are represented as eight-bit patterns. The agreement about what patterns represent what characters is called ASCII. The hardware and the software of a computer system (usually) follow this agreement when the data is "text". (You will learn more about this later). Other types of data (non-character data) are represented using other methods.

Representing Anything

- Japanese and Chinese characters have been assigned bit patterns, and computers can manipulate those characters just as easily as ASCII. Unicode is an agreement created by an international committee on how to represent characters using 16 bits. Here is a 16-bit pattern 1111100111111110 and here is the character it represents in Unicode:

- Say that the international committee decides to represent a new Chinese character. How can they do this? Easy: they find a bit pattern not yet used to represent any symbol and assign the new character to that pattern.

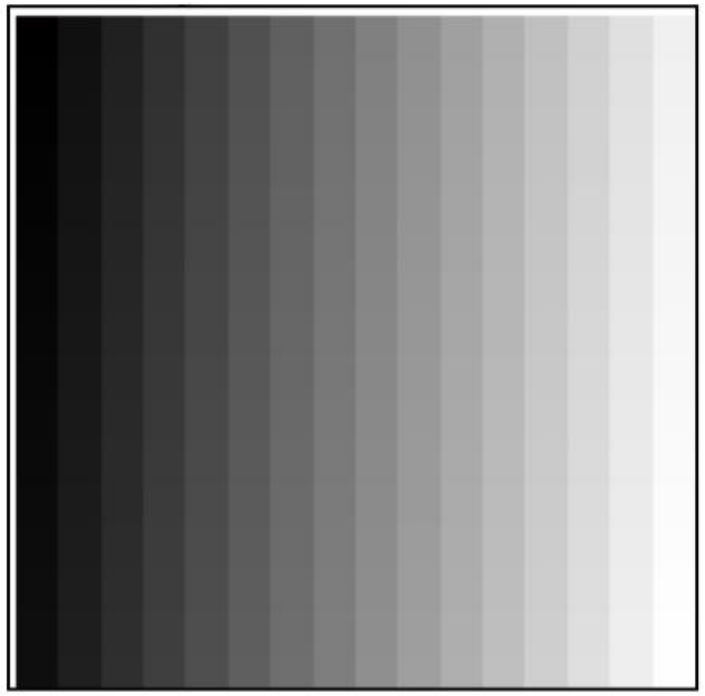
- The correspondence between human language symbols and bit patterns is arbitrary. All you have to do is be sure that you are using enough bits so that all the symbols of the language have a unique bit pattern to represent them.

Standard 7-bit ASCII Table

theascii.com

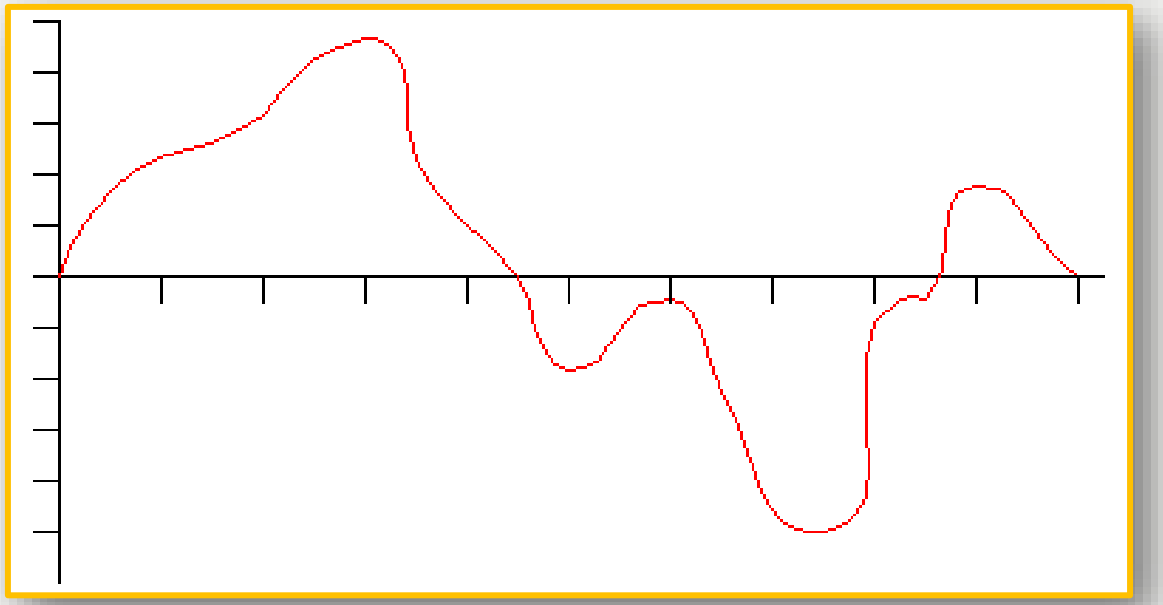
Dec	Hex	Oct	Binary	Char	Description	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
0	0	0	0	NUL	Null character	32	20	40	100000	space	64	40	100	1000000	@	96	60	140	1100000	`
1	1	1	1	SOH	Start of header	33	21	41	100001	!	65	41	101	1000001	A	97	61	141	1100001	a
2	2	2	10	STX	Start of text	34	22	42	100010	"	66	42	102	1000010	B	98	62	142	1100010	b
3	3	3	11	ETX	End of text	35	23	43	100011	#	67	43	103	1000011	C	99	63	143	1100011	c
4	4	4	100	EOT	End of transmission	36	24	44	100100	$	68	44	104	1000100	D	100	64	144	1100100	d
5	5	5	101	ENQ	Enquiry	37	25	45	100101	%	69	45	105	1000101	E	101	65	145	1100101	e
6	6	6	110	ACK	Acknowledge	38	26	46	100110	&	70	46	106	1000110	F	102	66	146	1100110	f
7	7	7	111	BEL	Bell ring	39	27	47	100111	'	71	47	107	1000111	G	103	67	147	1100111	g
8	8	10	1000	BS	Backspace	40	28	50	101000	(	72	48	110	1001000	H	104	68	150	1101000	h
9	9	11	1001	HT	Horizontal tab	41	29	51	101001	)	73	49	111	1001001	I	105	69	151	1101001	i
10	0A	12	1010	LF	Line feed	42	2A	52	101010	*	74	4A	112	1001010	J	106	6A	152	1101010	j
11	0B	13	1011	VT	Vertical tab	43	2B	53	101011	+	75	4B	113	1001011	K	107	6B	153	1101011	k
12	0C	14	1100	FF	Form feed	44	2C	54	101100	,	76	4C	114	1001100	L	108	6C	154	1101100	l
13	0D	15	1101	CR	Carriage return	45	2D	55	101101	–	77	4D	115	1001101	M	109	6D	155	1101101	m
14	0E	16	1110	SO	Shift out	46	2E	56	101110	.	78	4E	116	1001110	N	110	6E	156	1101110	n
15	0F	17	1111	SI	Shift in	47	2F	57	101111	/	79	4F	117	1001111	O	111	6F	157	1101111	o
16	10	20	10000	DLE	Data link escape	48	30	60	110000	0	80	50	120	1010000	P	112	70	160	1110000	p
17	11	21	10001	DC1	Device control 1	49	31	61	110001	1	81	51	121	1010001	Q	113	71	161	1110001	q
18	12	22	10010	DC2	Device control 2	50	32	62	110010	2	82	52	122	1010010	R	114	72	162	1110010	r
19	13	23	10011	DC3	Device control 3	51	33	63	110011	3	83	53	123	1010011	S	115	73	163	1110011	s
20	14	24	10100	DC4	Device control 4	52	34	64	110100	4	84	54	124	1010100	T	116	74	164	1110100	t
21	15	25	10101	NAK	Negative acknowledge	53	35	65	110101	5	85	55	125	1010101	U	117	75	165	1110101	u
22	16	26	10110	SYN	Synchronize	54	36	66	110110	6	86	56	126	1010110	V	118	76	166	1110110	v
23	17	27	10111	ETB	End transmission block	55	37	67	110111	7	87	57	127	1010111	W	119	77	167	1110111	w
24	18	30	11000	CAN	Cancel	56	38	70	111000	8	88	58	130	1011000	X	120	78	170	1111000	x
25	19	31	11001	EM	End of medium	57	39	71	111001	9	89	59	131	1011001	Y	121	79	171	1111001	y
26	1A	32	11010	SUB	Substitute	58	3A	72	111010	:	90	5A	132	1011010	Z	122	7A	172	1111010	z
27	1B	33	11011	ESC	Escape	59	3B	73	111011	;	91	5B	133	1011011	[	123	7B	173	1111011	{
28	1C	34	11100	FS	File separator	60	3C	74	111100	<	92	5C	134	1011100	\	124	7C	174	1111100	|
29	1D	35	11101	GS	Group separator	61	3D	75	111101	=	93	5D	135	1011101	]	125	7D	175	1111101	}
30	1E	36	11110	RS	Record separator	62	3E	76	111110	>	94	5E	136	1011110	^	126	7E	176	1111110	~
31	1F	37	11111	US	Unit separator	63	3F	77	111111	?	95	5F	137	1011111	_	127	7F	177	1111111	DEL

Representing Anything

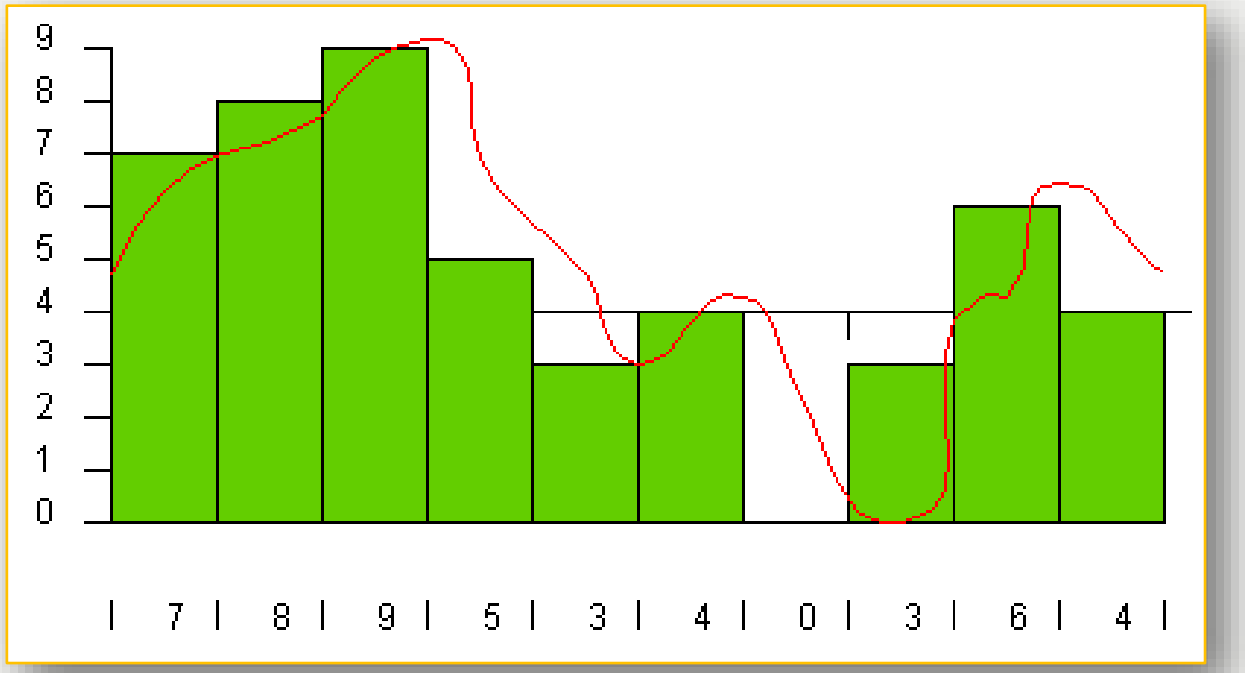


0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

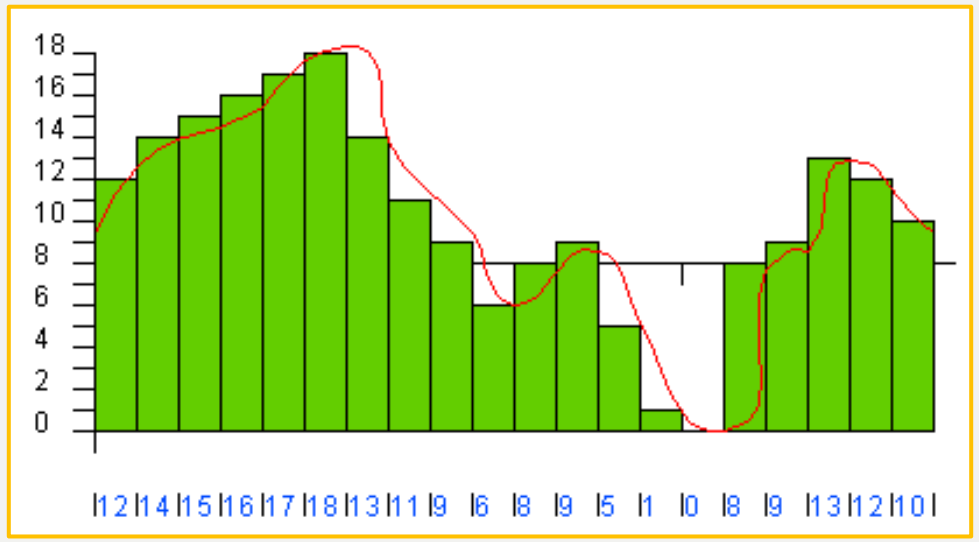
Representing Anything

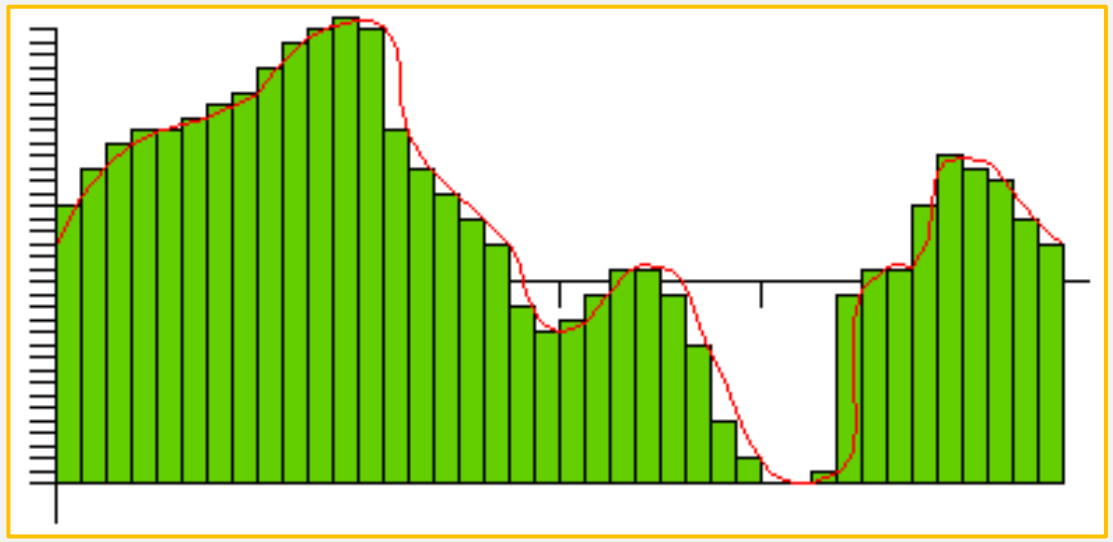


Representing Anything



Representing Anything





Bit

A bit is a single on/off value. Only these two values are possible. There are many ways in which a bit can be implemented. Here are some ways that bits are implemented:

- A mechanical electrical switch (like a light switch.)
- Voltage on a wire.
- A single transistor.
- A tiny part of the surface of a magnetic disk.
- A tiny part of the surface of a magnetic tape.
- A hole punched in a card.
- A tiny part of the light-reflecting surface of a CD.
- Part of a radio signal.
- Part of an audio signal.

The implementation of bits is different in main memory and secondary memory, but logically, both types of memory store information represented as bit patterns.

Byte

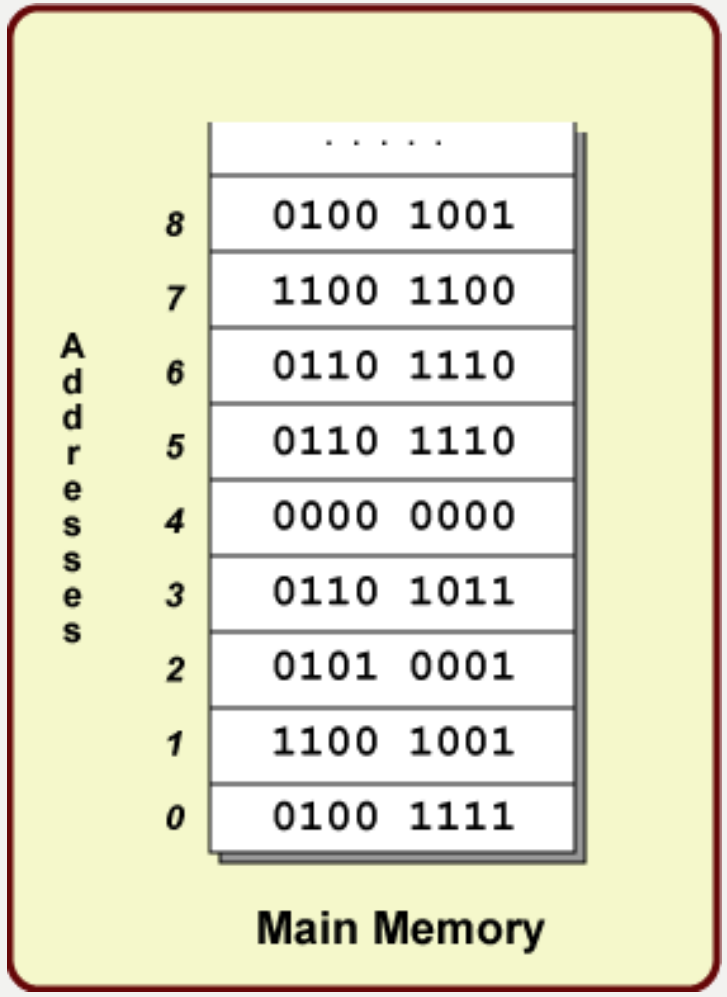
- One bit of information is so little that usually computer memory is organized into groups of eight bits. Each eight bit group is called a **byte**.
- One byte is about enough memory to hold a single character. When more than eight bits are required for some data, several of bytes are used.

Name	Number of Bytes	power of 2
byte	1	2^0
kilobyte	1024	2^{10}
megabyte	1,048,576	2^{20}
gigabyte	1,073,741,824	2^{30}
terabyte	1,099,511,627,776	2^{40}

Contents of Main Memory

- Main memory consists of a very long list of bytes. In most modern computers, each byte has an **address** that is used to locate it. The picture shows a small part of main memory:
- Each box in this picture represents a single byte. Each byte has an address. In this picture the addresses are the integers to the left of the boxes: 0, 1, 2, 3, 4, ... and so on. The addresses for most computer memory start at 0 and go up in sequence until each byte has an address.
- Each byte contains a pattern of eight bits. When the computer's power is on, every byte contains some pattern or other, even when those bytes are not being used for anything. (Remember the nature of binary: when a binary device is working it is either "on" or "off", never in-between.)

Contents of Main Memory



Contents of Main Memory

- Main memory (as all computer memory) stores bit patterns. That is, each memory location consists of eight bits, and each bit is either 0 or 1. For example, the picture shows the first few bytes of memory.
- The *only* thing that can be stored at one memory location is eight bits, each with a value of 0 or 1. The bits at a memory location are called the *contents* of that location.
- The information that a particular pattern represents depends on its context (how a program is using it.) You cannot look at an arbitrary bit pattern (such as those in the picture) and say what it represents.

Files

- Hard disks (and other secondary memory devices) are used for long-term storage of large blocks of information, such as programs and data sets. Usually disk memory is organized into files.
- A file is a collection of information that has been given a name and is stored in secondary memory. The information can be a program or can be data.
- Information in a file is represented the same as with any digital information — it consists of bits, usually grouped into eight bit bytes. Files are frequently large; their size is measured in kilobytes or megabytes.
- One of the jobs of a computer's operating system is to keep track of file names and where they are on its hard disk. For example, in DOS the user can ask to run the program DOOM like this:
- D:\New folder\note.txt
- The operating system now has to find the file somewhere on its hard disk. The program will be copied into main storage and will start running. As the program runs it asks for information stored as additional files on the hard disk, which the operating system has to find and copy into main memory.

Files and the Operating System

- Most collections of data outside of main storage are organized into files. Keeping track of all this information is one of the jobs of the operating system. If the computer is part of a network, keeping track of all the files on all the computers is a big job, and involves all the operating systems on the network.
- Application programs (including programs that you might write) do not directly read, write, create, or delete files. Since the operating system has to keep track of everything, all other programs ask it to do file manipulation tasks. For example, say that a program has just calculated a set of numbers and needs to save them. The following might be how it does this:
  - **Program:** asks the operating system to create a file with a name RESULTS.DAT
  - **Operating System:** gets the request; finds an unused section of the disk and creates an empty file. The program is told when this has been completed.
  - **Program:** asks the operating system to save the numbers in the file.
  - **Operating System:** gets the numbers from the program's main memory, writes them to the file. The program is told when this has been completed.
  - **Program:** continues on with whatever it is doing.

Types of Files

- As far as the hard disk is concerned, all files are the same. At the electronic level, there is no difference between a file containing a program and a file containing data. All files are named collections of bytes. Of course, what the files are used for is different. The operating system can take a program file, copy it into main memory, and start it running. The operating system can take a data file, and supply its information to a running program when it asks.

- Often then last part of a file's name (the extension) shows what the file is expected to be used for. For example, in mydata.txt the .txt means that the file is expected to be used as a collection of text, that is, characters. With doom.exe the .exe means that the file is an "executable," that is, a program that is ready to run. With program1.java the .java means that the file is a source program in the language Java. To the hard disk, each of these files is the same sort of thing: a collection of bytes holding bit patterns.