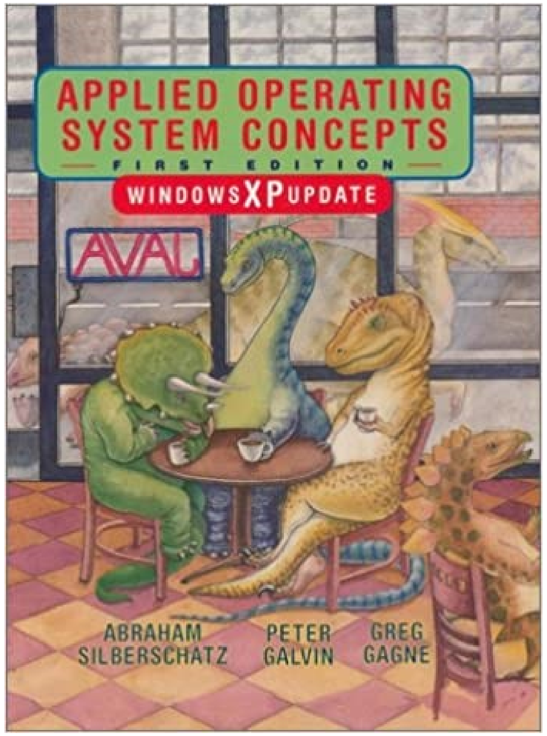
Chapter 1: Introduction

School of Computing, Gachon Univ.
Joon Yoo



Gachon University
School of Computing

Objectives

- To provide the **concept** of operating systems
- To study the basics of **computer system** and what’s **under your program**
- To describe the basic organization of **computer systems**
- To explore several **open-source** operating systems

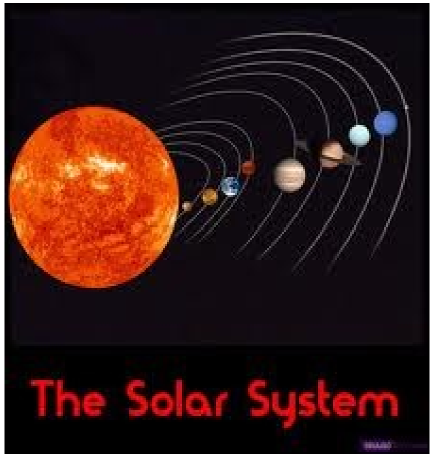
- Note
  - This part includes contents from both Ch.1 and Ch.2 in your textbook.
  - Note that the slide order does not follow your textbook.

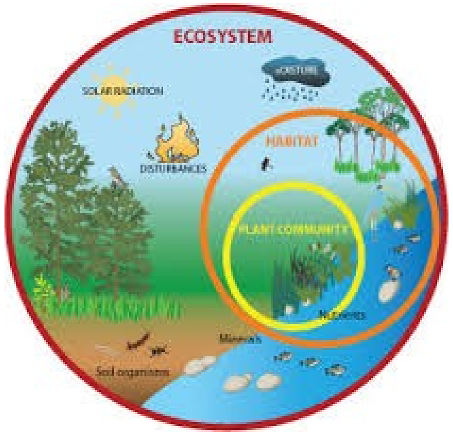
Chapter 1: Introduction

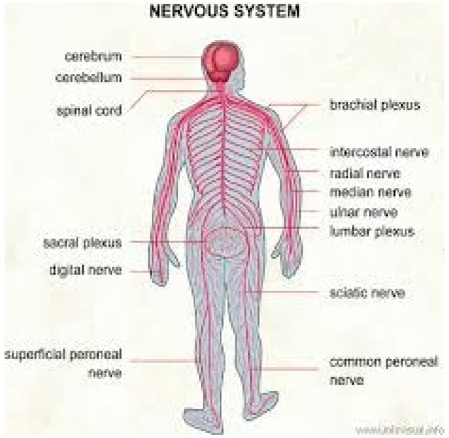
- What is an Operating System?
- What Operating Systems Do
- Computer-System
  - Below your Program
  - Organization, Architecture
- Computing Environments
- Open-Source Operating Systems

Definition of a System (系)

- “A systems is a collection of components linked together and organized in such a way as to be recognizable as a single unit.”
- Examples





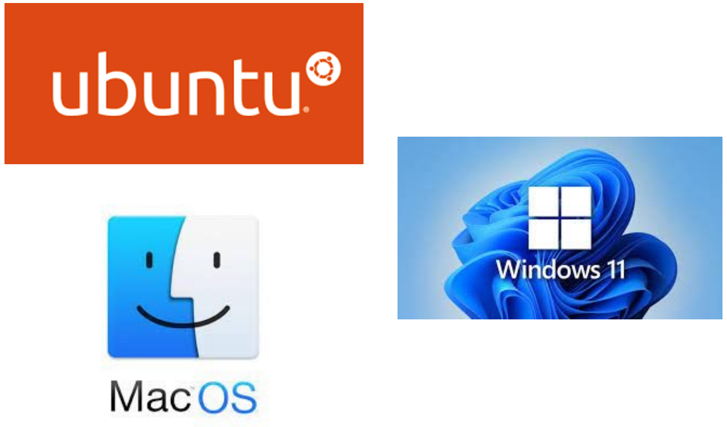


- **Operating System: “A collection of computer programs (=software)”** that **manages computer hardware resources.** Also provides a **basis for application programs** and acts as an **interface between the computer user and the computer hardware**.”

OSs are Everywhere

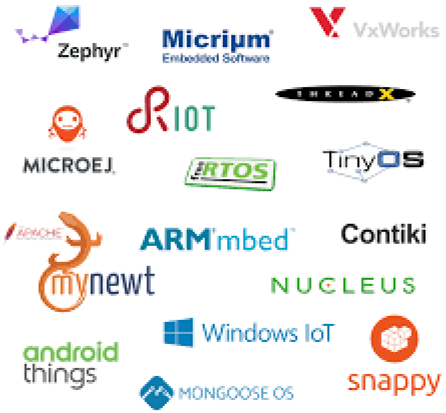
Various OSs

- Servers, PCs
  - UNIX, Linux
  - Microsoft Windows
  - Apple Mac OS



- Mobile
  - Android (Google), iOS (Apple)

- Embedded

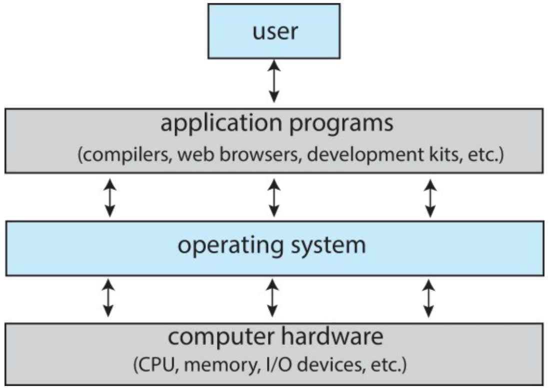


Chapter 1: Introduction

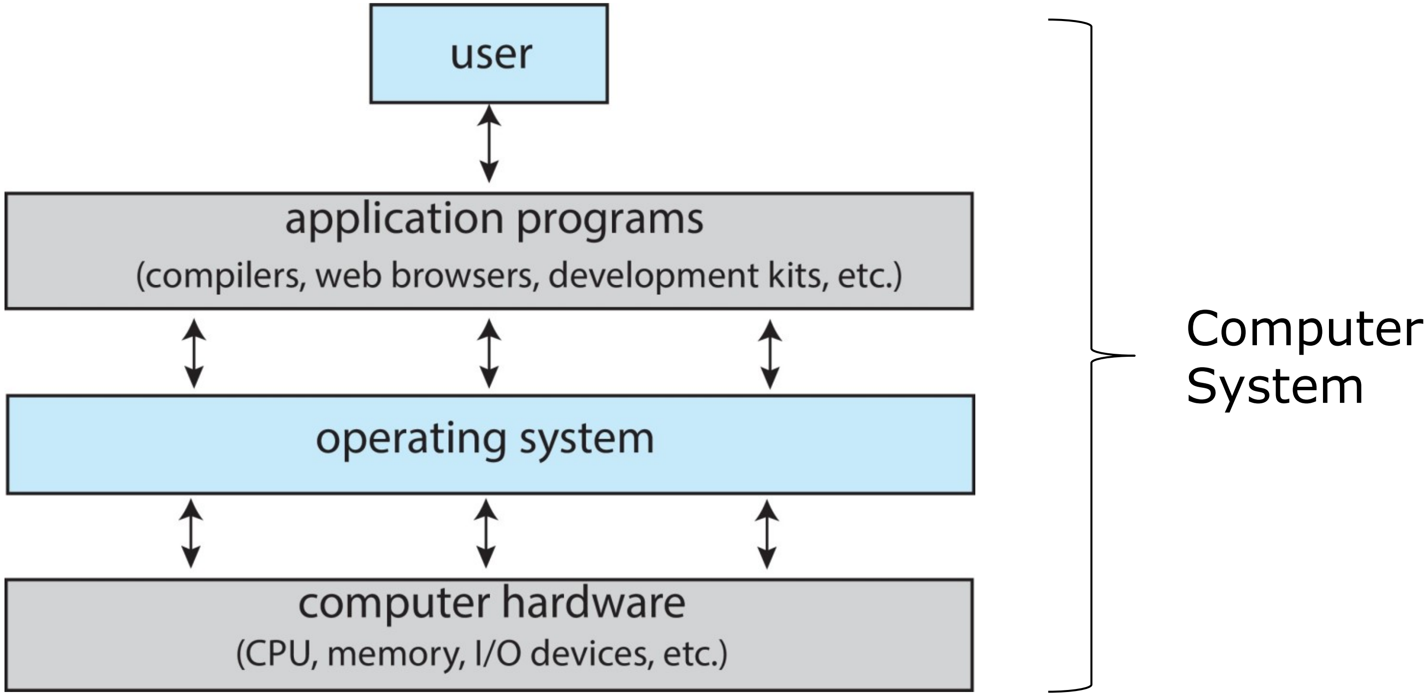
- What is an Operating System?
- What Operating Systems Do
- Computer-System
  - Below your Program
  - Organization, Architecture
- Computing Environments
- Open-Source Operating Systems

Computer System

- Computer system can be divided into four components:
  - **Computer Hardware** – provides basic computing **resources**
    - CPU, memory, I/O devices (Computer Architecture)
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

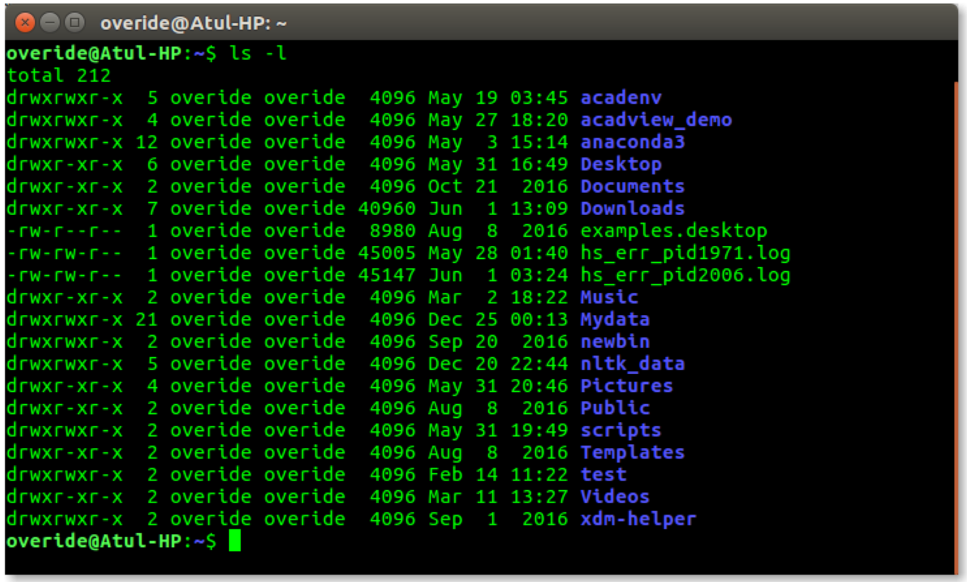

user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

What Operating Systems Do


user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)
Computer
System

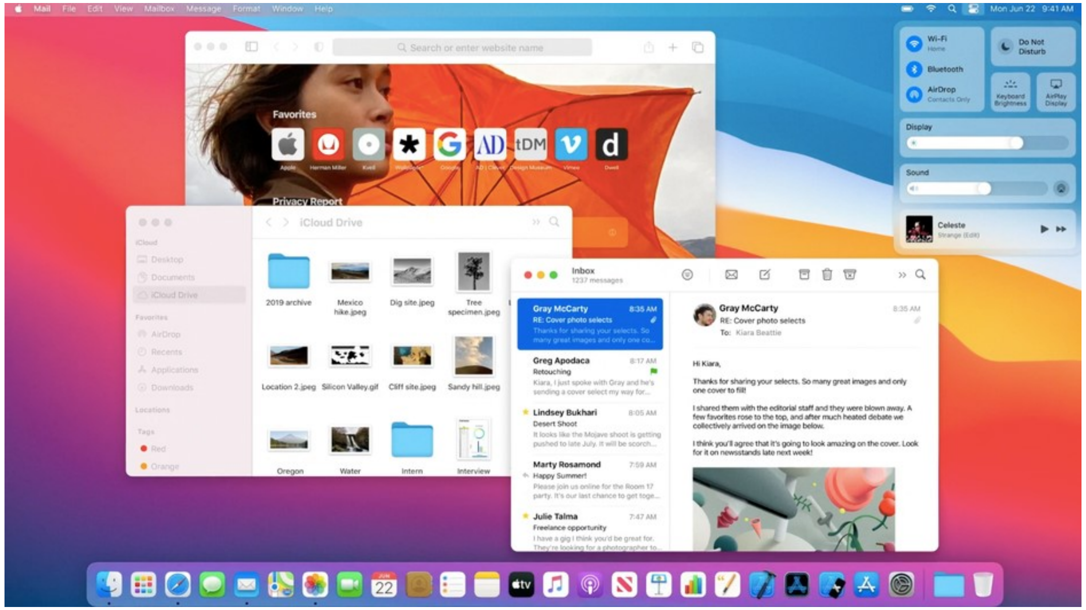
- **Operating System: "A collection of computer programs (=software)"** that manages **computer hardware resources.** Also provides a **basis for application programs** and acts as **an interface between the computer user and the computer hardware.**"

User View: User Interface (Ch. 2.2)



```
overide@Atul-HP:~$ ls -l
total 212
drwxrwxr-x  5 overide overide  4096 May 19 03:45 acadenv
drwxrwxr-x  4 overide overide  4096 May 27 18:20 acadview_demo
drwxrwxr-x 12 overide overide  4096 May  3 15:14 anaconda3
drwxr-xr-x  6 overide overide  4096 May 31 16:49 Desktop
drwxr-xr-x  2 overide overide  4096 Oct 21  2016 Documents
drwxr-xr-x  7 overide overide 40960 Jun  1 13:09 Downloads
-rw-r--r--  1 overide overide  8980 Aug  8  2016 examples.desktop
-rw-rw-r--  1 overide overide 45005 May 28 01:40 hs_err_pid1971.log
-rw-rw-r--  1 overide overide 45147 Jun  1 03:24 hs_err_pid2006.log
drwxr-xr-x  2 overide overide  4096 Mar  2 18:22 Music
drwxrwxr-x 21 overide overide  4096 Dec 25 00:13 Mydata
drwxrwxr-x  2 overide overide  4096 Sep 20  2016 newbin
drwxrwxr-x  5 overide overide  4096 Dec 20 22:44 nltk_data
drwxr-xr-x  4 overide overide  4096 May 31 20:46 Pictures
drwxr-xr-x  2 overide overide  4096 Aug  8  2016 Public
drwxrwxr-x  2 overide overide  4096 May 31 19:49 scripts
drwxr-xr-x  2 overide overide  4096 Aug  8  2016 Templates
drwxrwxr-x  2 overide overide  4096 Feb 14 11:22 test
drwxr-xr-x  2 overide overide  4096 Mar 11 13:27 Videos
drwxrwxr-x  2 overide overide  4096 Sep  1  2016 xdm-helper
overide@Atul-HP:~$
```

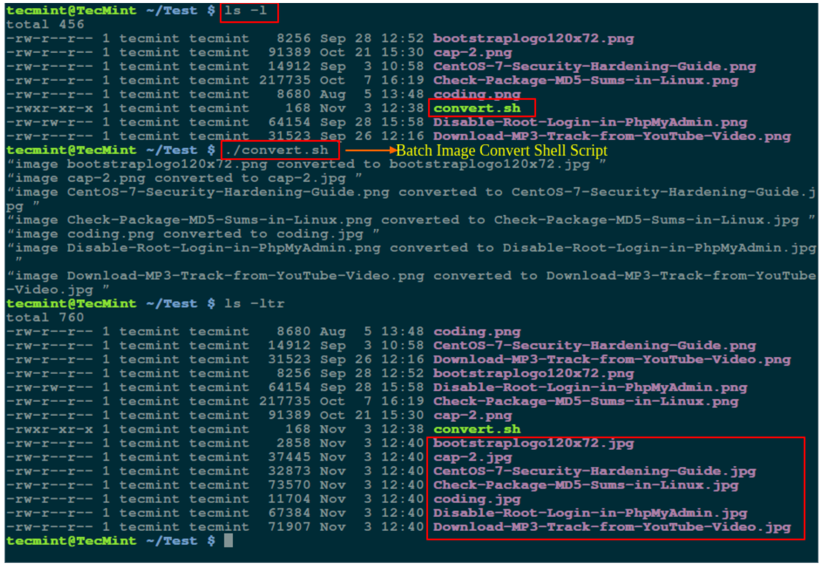
Command Line Interface (CLI) – UNIX Bourne Shell



Graphical User Interface (GUI) – MacOS

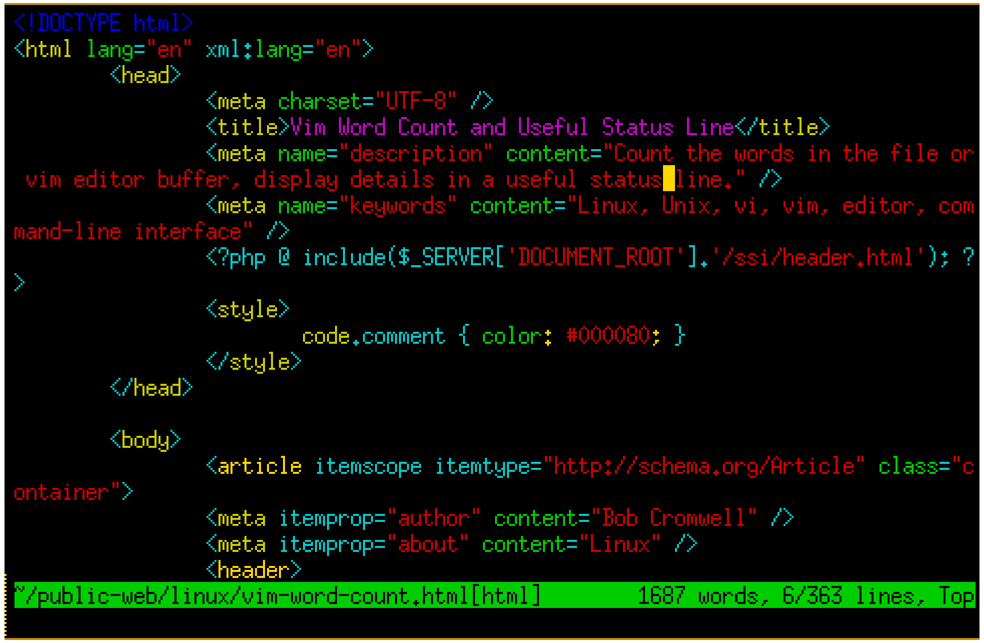
User Operating System Interface

- Why use Command Line Interface (CLI)?
  - On some systems, only a subset of system functions is available via the GUI.
    - So, CLI is usually for power users and programmers (like you)
  - Further, command-line interfaces usually make repetitive tasks easier
    - Linux shell scripts are very common on systems that are command-line oriented.
  - You will learn to use Linux CLI in your first programming assignment.



```
tecmint@TecMint ~/Test $ ls -l
total 456
-rw-r--r-- 1 tecmint tecmint   8256 Sep 28 12:52 bootstraplogo120x72.png
-rw-r--r-- 1 tecmint tecmint  91389 Oct 21 15:30 cap-2.png
-rw-r--r-- 1 tecmint tecmint  14912 Sep  3 10:58 CentOS-7-Security-Hardening-Guide.png
-rw-r--r-- 1 tecmint tecmint 217735 Oct  7 16:19 Check-Package-MD5-Sums-in-Linux.png
-rw-r--r-- 1 tecmint tecmint   8680 Aug  5 13:48 coding.png
-rwxr-xr-x 1 tecmint tecmint    168 Nov  3 12:38 convert.sh
-rw-rw-r-- 1 tecmint tecmint  64154 Sep 28 15:58 Disable-Root-Login-in-PhpMyAdmin.png
-rw-r--r-- 1 tecmint tecmint  31523 Sep 26 12:16 Download-MP3-Track-from-YouTube-Video.png
tecmint@TecMint ~/Test $ ./convert.sh
"image bootstraplogo120x72.png converted to bootstraplogo120x72.jpg "
"image cap-2.png converted to cap-2.jpg "
"image CentOS-7-Security-Hardening-Guide.png converted to CentOS-7-Security-Hardening-Guide.j
pg "
"image Check-Package-MD5-Sums-in-Linux.png converted to Check-Package-MD5-Sums-in-Linux.jpg "
"image coding.png converted to coding.jpg "
"image Disable-Root-Login-in-PhpMyAdmin.png converted to Disable-Root-Login-in-PhpMyAdmin.jpg
 "
"image Download-MP3-Track-from-YouTube-Video.png converted to Download-MP3-Track-from-YouTube
-Video.jpg "
tecmint@TecMint ~/Test $ ls -ltr
total 760
-rw-r--r-- 1 tecmint tecmint   8680 Aug  5 13:48 coding.png
-rw-r--r-- 1 tecmint tecmint  14912 Sep  3 10:58 CentOS-7-Security-Hardening-Guide.png
-rw-r--r-- 1 tecmint tecmint  31523 Sep 26 12:16 Download-MP3-Track-from-YouTube-Video.png
-rw-r--r-- 1 tecmint tecmint   8256 Sep 28 12:52 bootstraplogo120x72.png
-rw-rw-r-- 1 tecmint tecmint  64154 Sep 28 15:58 Disable-Root-Login-in-PhpMyAdmin.png
-rw-r--r-- 1 tecmint tecmint 217735 Oct  7 16:19 Check-Package-MD5-Sums-in-Linux.png
-rw-r--r-- 1 tecmint tecmint  91389 Oct 21 15:30 cap-2.png
-rwxr-xr-x 1 tecmint tecmint    168 Nov  3 12:38 convert.sh
-rw-r--r-- 1 tecmint tecmint   2858 Nov  3 12:40 bootstraplogo120x72.jpg
-rw-r--r-- 1 tecmint tecmint  37445 Nov  3 12:40 cap-2.jpg
-rw-r--r-- 1 tecmint tecmint  32873 Nov  3 12:40 CentOS-7-Security-Hardening-Guide.jpg
-rw-r--r-- 1 tecmint tecmint  73570 Nov  3 12:40 Check-Package-MD5-Sums-in-Linux.jpg
-rw-r--r-- 1 tecmint tecmint  11704 Nov  3 12:40 coding.jpg
-rw-r--r-- 1 tecmint tecmint  67384 Nov  3 12:40 Disable-Root-Login-in-PhpMyAdmin.jpg
-rw-r--r-- 1 tecmint tecmint  71907 Nov  3 12:40 Download-MP3-Track-from-YouTube-Video.jpg
tecmint@TecMint ~/Test $ █
```

Batch Image Convert Shell Script



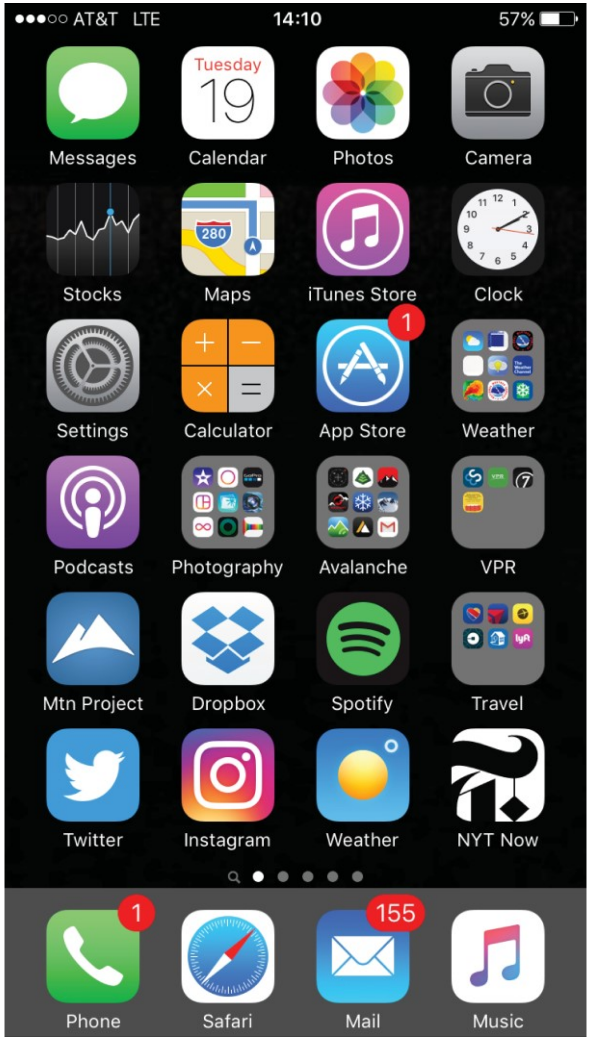
```
<!DOCTYPE html>
<html lang="en" xml:lang="en">
        <head>
                <meta charset="UTF-8" />
                <title>Vim Word Count and Useful Status Line</title>
                <meta name="description" content="Count the words in the file or
 vim editor buffer, display details in a useful status█line." />
                <meta name="keywords" content="Linux, Unix, vi, vim, editor, com
mand-line interface" />
                <?php @ include($_SERVER['DOCUMENT_ROOT'].'/ssi/header.html'); ?
>
                <style>
                        code.comment { color: #000080; }
                </style>
        </head>

        <body>
                <article itemscope itemtype="http://schema.org/Article" class="c
ontainer">
                <meta itemprop="author" content="Bob Cromwell" />
                <meta itemprop="about" content="Linux" />
                <header>
~/public-web/linux/vim-word-count.html[html]        1687 words, 6/363 lines, Top
```

User View

Touchscreen Interfaces

- Touchscreen devices require new interfaces
  - Mouse not possible or not desired
  - Actions and selection based on gestures
  - Virtual keyboard for text entry
- Voice commands (e.g., Amazon Alexa, Google assistant, Siri)



System View

System View of Operating System (=Kernel)

- OS makes **hardware** useful to the programmer
  - A computer system has many resources to solve a problem
    - Resources?: CPU time, memory space, storage, I/O devices
  - OS acts as a manger of these resources – decides how to allocate the resources to the programs
- OS controls **user programs**
  - Manages the execution of user programs to prevent errors and improper use of the computer

Chapter 1: Introduction

- What is an Operating System?
- What Operating Systems Do
- Computer-System
  - Below Your Program
  - Organization, Architecture
- Computing Environments
- Open-Source Operating Systems

Basics: Storage definition and notations

- To actually speak to electronic hardware (e.g., CPU, memory, I/O), you need to send *electronic signals*

- The easiest signal for computers to understand are *on* and *off*

- So, the computer alphabet is just two letters (*on* and *off*, or 0 and 1) called ***bits***

Basics: Storage definition and notations

- Two letters of computer alphabet do not limit what computers can do (just as 26 letters of English, A…Z, do not limit how much can be written)

- The two letters are 0 and 1, and we commonly think computer language is number in ***base 2***, ***binary digit (bit)***.
  - *Example*: 01000011

Basics: bit (b)

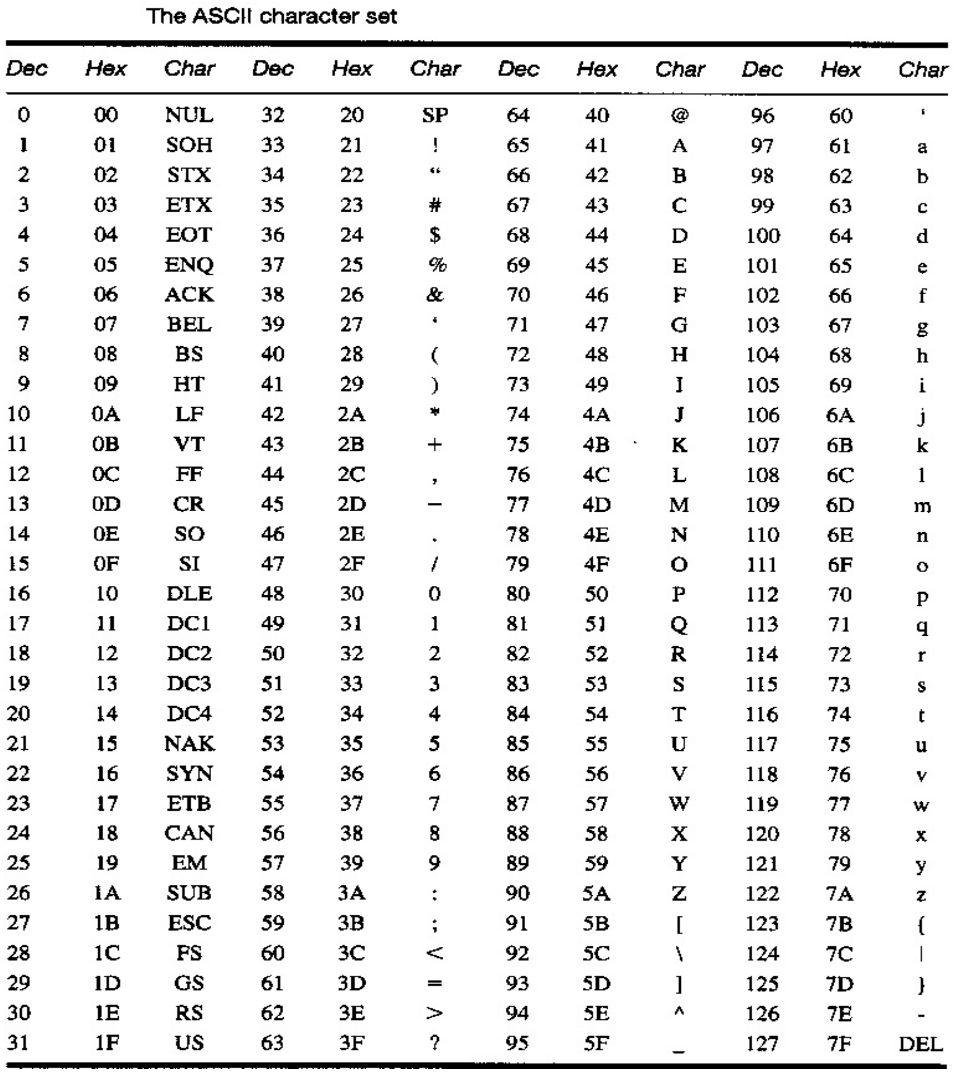
- **bit (0 or 1)**
  - Basic unit of computer storage.
  - 1bit: 0, 1 → count $2^1 = 2$
  - 2bits: 00, 01, 10, 11 → count $2^2 = 4$
  - 3bits: 000, 001, 010, 011, 100, 101, 110, 111 → $2^3 = 8$
  - …
  - n bits: → Can represent 2^n numbers.
  - 8 bits = 1 **byte**
    - How many numbers can one byte represent?

Basics: bit (b)

Using bits, computer can represent numbers, letters, images, movies, sounds, documents, and programs

Alphabet Letters

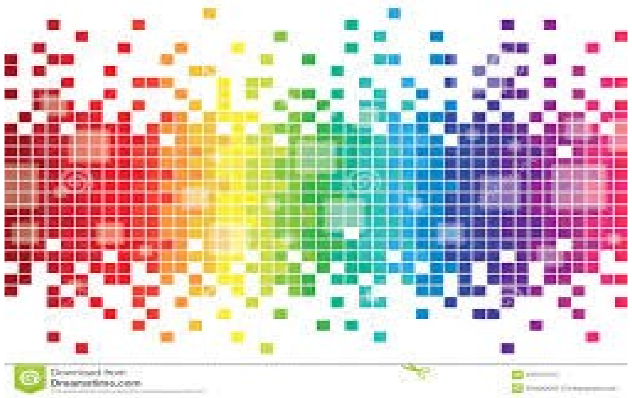
- **ASCII characters** (1byte)



The ASCII character set

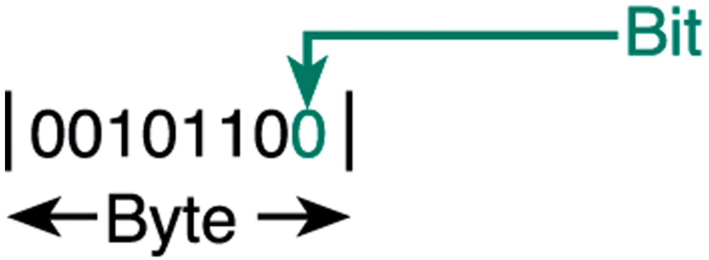
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	32	20	SP	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	–	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	-
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

- Image pixels



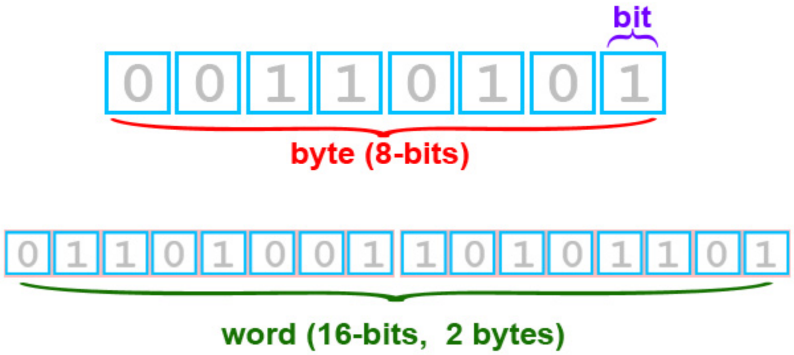
Basics: Byte (B)

- 1 byte = 8 bits
- Smallest chunk of data for most computers
  - Minimum main memory unit is 1-byte
- Why 8-bits? (not 7 not 9?)
  - First used 8-bits for ASCII characters
    - e.g., “A”: 01000001, “z”: 01111010
  - Used 8-bits for early Internet
  - Intel developed 8-bit microprocessors



Basics: Word

- One or more bytes
  - A machine with 32-bit Registers and 32-bit memory address (or 32-bit machine) has 32-bit (or 4-byte) words
    - 1 word = 4 byte = 32 bit
  - In a 64-bit machine 1 word = 8 bytes (same reason as above)



bit

0 0 1 1 0 1 0 1

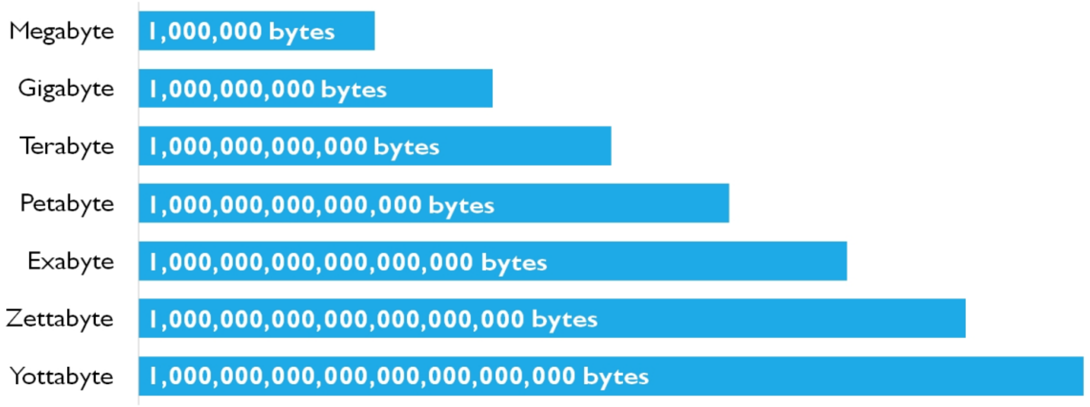
byte (8-bits)

0 1 1 0 1 0 0 1 1 0 1 0 1 1 0 1

word (16-bits, 2 bytes)

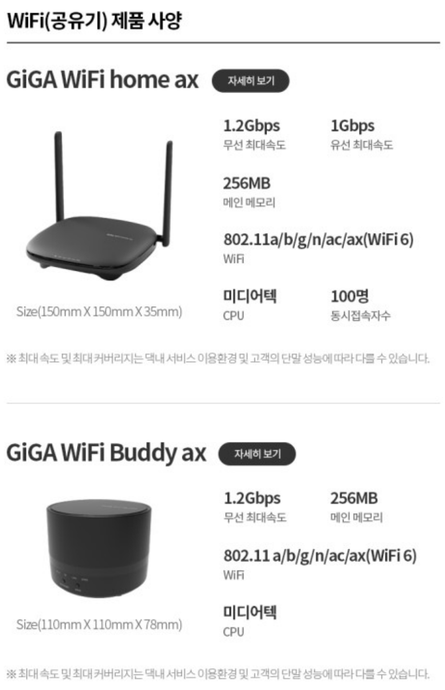
Basics: Collection of bytes (or bits)

- e.g., File 2MB, Memory 4GB, HDD 1TB, …



Megabyte	1,000,000 bytes
Gigabyte	1,000,000,000 bytes
Terabyte	1,000,000,000,000 bytes
Petabyte	1,000,000,000,000,000 bytes
Exabyte	1,000,000,000,000,000,000 bytes
Zettabyte	1,000,000,000,000,000,000,000 bytes
Yottabyte	1,000,000,000,000,000,000,000,000 bytes

- Networking measurements are usually in **bits** (not bytes)
  - 1Gbps = 1Gbit/s = 1,000M bits per second
    = 125MB/s (Bytes per second)



WiFi(공유기) 제품 사양

GiGA WiFi home ax

1.2Gbps
무선 최대속도

1Gbps
유선 최대속도

256MB
메인 메모리

802.11a/b/g/n/ac/ax(WiFi 6)
WiFi

미디어텍
CPU

100명
동시접속자수

Size(150mm X 150mm X 35mm)

※ 최대 속도 및 최대 커버리지는 댁내 서비스 이용환경 및 고객의 단말 성능에 따라 다를 수 있습니다.

GiGA WiFi Buddy ax

1.2Gbps
무선 최대속도

256MB
메인 메모리

802.11 a/b/g/n/ac/ax(WiFi 6)
WiFi

미디어텍
CPU

Size(110mm X 110mm X 78mm)

※ 최대 속도 및 최대 커버리지는 댁내 서비스 이용환경 및 고객의 단말 성능에 따라 다를 수 있습니다.

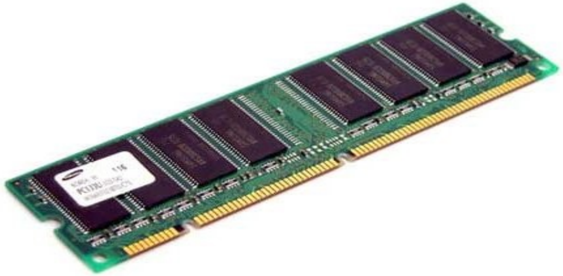
Storage Definitions and Notation Review

The basic unit of computer storage is the **bit** . A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes. A **kilobyte** , or KB , is 1,024 bytes; a **megabyte** , or **MB** , is $1{,}024^2$ bytes; a **gigabyte** , or GB , is $1{,}024^3$ bytes; a **terabyte** , or **TB** , is $1{,}024^4$ bytes; and a **petabyte** , or **PB** , is $1{,}024^5$ bytes. Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Basics: Memory

- Address
  - relative position in memory
- Contents
  - the data stored in a memory cell
- The size of each memory cell is **one BYTE**

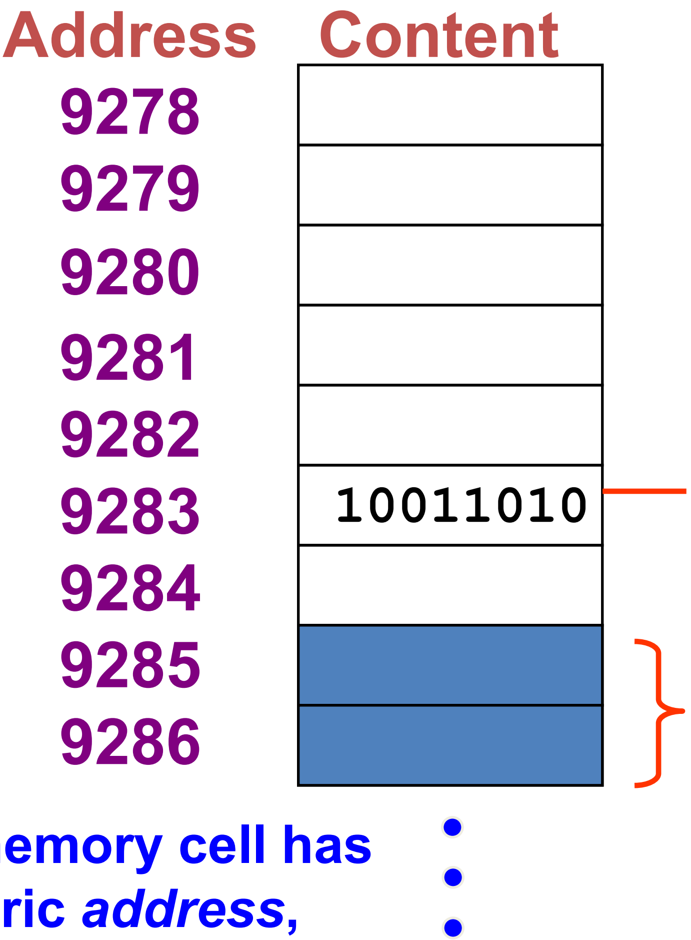


1000 Memory Cells in Main Memory

Memory

Address	Contents
0	−27.2
1	354
2	0.005
3	−26
4	H
. . .	. . .
998	X
999	75.62

Basics: Computer systems use Binary numbers So, conceptually, memory looks like …



Main memory looks like a table

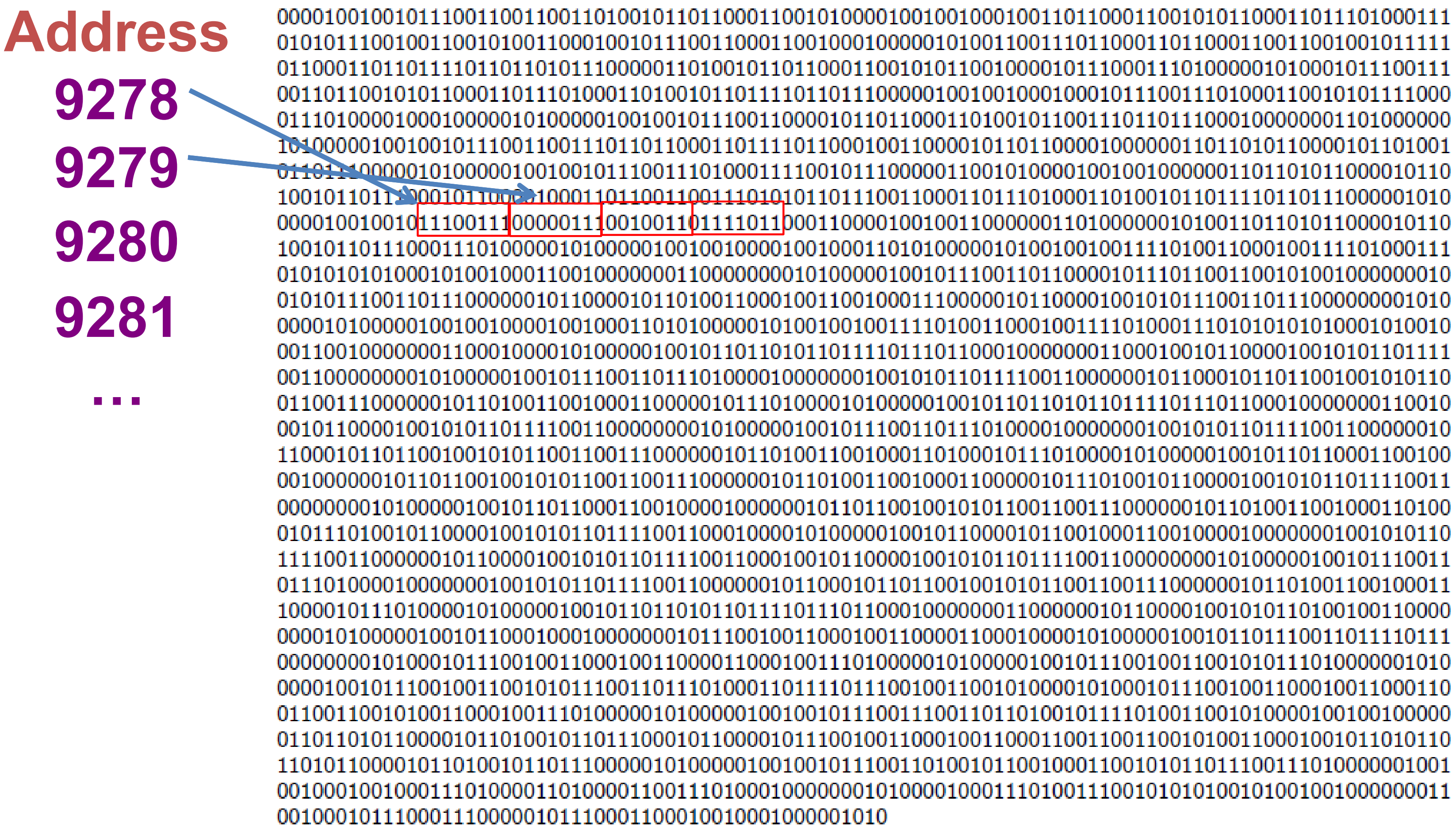
Main memory is divided into many memory locations (or *cells*)

Each memory cell stores a set number of bits (usually 8 bits, or one *byte*)

Large values are stored in consecutive memory locations

Each memory cell has a numeric *address*, which uniquely identifies it

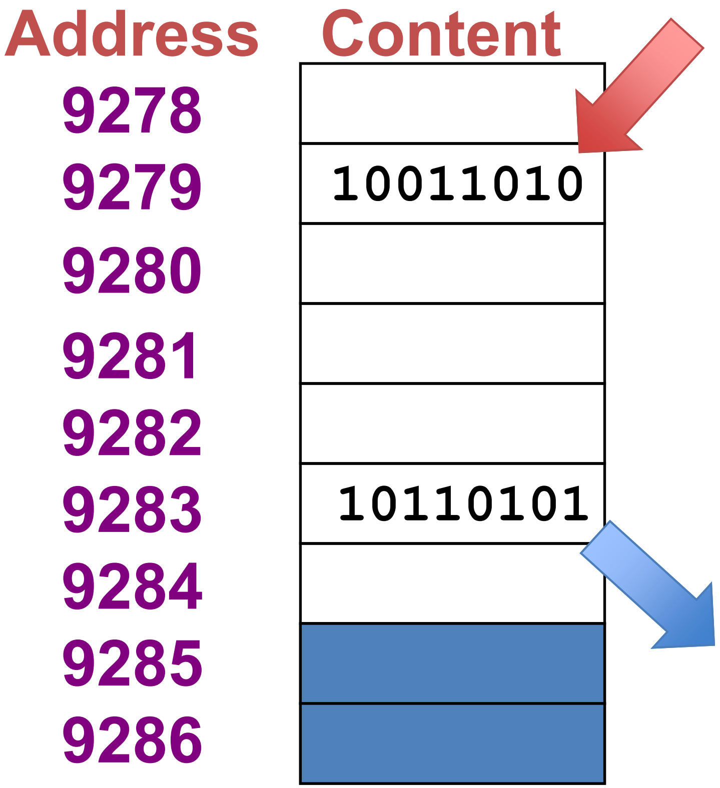
Basics: Actually, memory looks like …



```
00001001001011100110011001101001011011000110010100001001001000100110110001100101011000110111101000111
01010111001001100101001100010010111001100011001000100000101001100111011000110110001100110010010111 11
01100011011011110110110101110000011010010110110001100101011001000010111000111010000010100010111001 11
00110110010101100011011110100011010010110111101101110000010010010001000101110011101000110010101111000
01110100001000100000101000000100100101110011000010110110001101001011001110110111000100000001101000000
10100000010010010111001100111011011000110111101100010011000010110110000100000011011010110000101101001
01101110000001010000001001001011100111010001111001011100000110010100001001001000000110110101100000101 10
10010110111000010111001110001101100110011101010110111001100011011101000110100101101111011011100000101 0
00001001001011100111000001110010011011111011000110000100100110000001101000000101001101101011000010110
10010110111000111010000010100000100100100001001000110101000001010010010011110100110001001111010001 11
01010101010001010010001100100000001100000000101000001001011100110110000101110110011001010010000000 10
01010111001101110000001011000010110100110001001100100011100000101100001001010111001101110000000010 10
00001010000001001001000010010001101010000010100100100111101001100010011110100011101010101010001010010
00110010000000110001000010100000100101101101011011110111011000100000001100010010110000100101011011 11
00110000000001010000010010111001101110100001000000010010101101111001100000010110001011011001001010110
01100111000000101101001100100011000001011101000010100000100101101101011011110111011000100000000110010
00101100001001010110111100110000000010100000100101110011011101000010000000100101011011110011000000 10
11000101101100100101011001100111000000101101001100100011010001011101000010100000100101101100011001 00
00100000001011011001001010110011001110000000101101001100100011000001011101001011000010010101101111 0011
00000000101000001001011011000110010000100000010110110010010101100110011100000010110100110010001101 00
01011101001011000010010101101111001100010000101000001001011000010110010001100100001000000010010101 10
11110011000000101100001001010110111100110001001011000010010101101111001100000000101000001001011100 11
01110100001000000001001010110111100110000000101100010110110010010101100110011100000010110100110010 0011
10000101110100001010000010010110110101101111011101100010000000110000000101100001001010110100100110000
00001010000001001011000100010000000010111001001100010011000011000100001010000010010110111001101111 0111
00000000101000101110010011000100110000110001001110100000101000001001011100100110010101110100000010 10
00001001011100100110010101110011011101000110111101110010011001010000101000101110010011000100110001 10
01100110010100110001001110100000101000001001001011100111001101101001011110100110010100001001001000 00
01101101011000010110100101101110001011000010111001001100010011000110011001100101001100010010110101 10
11010110000101101001011011100000101000001001001011100110100101100100011001010110111001110100000010 01
00100010010001110100001101000011001110100010000000101000010001110100111001010101001010010010000000 11
00100010111000111000000101110001100010010001000001010
```

Basics: Storage and Retrieval of Information in Memory

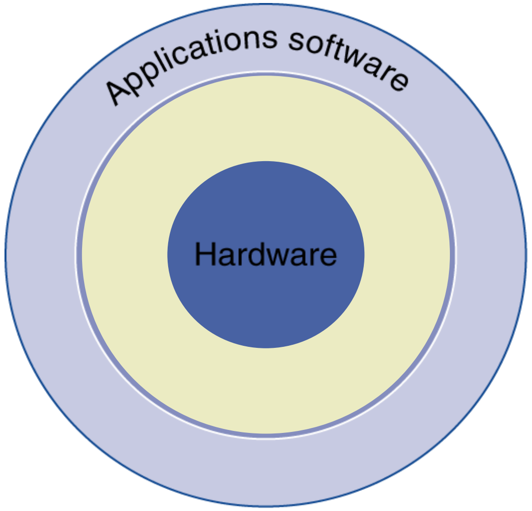
- Data storage (Write)
  - Setting the individual bits of a memory cell to 0 or 1, destroying its previous contents
- Data retrieval (Read)
  - Copying the contents of a particular memory cell to another storage area



Address	Content
9278	
9279	10011010
9280	
9281	
9282	
9283	10110101
9284	
9285	▨
9286	▨

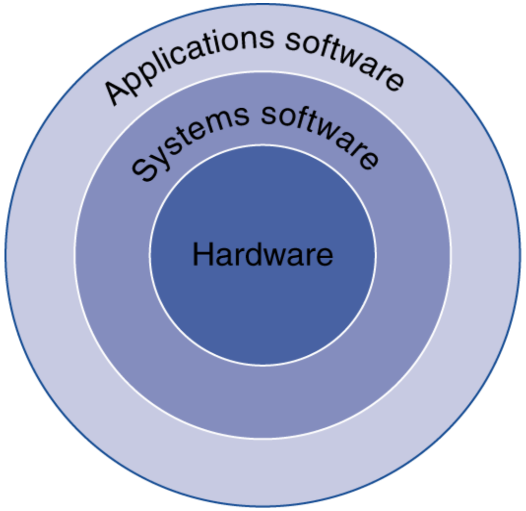
Basics: Below Your Program

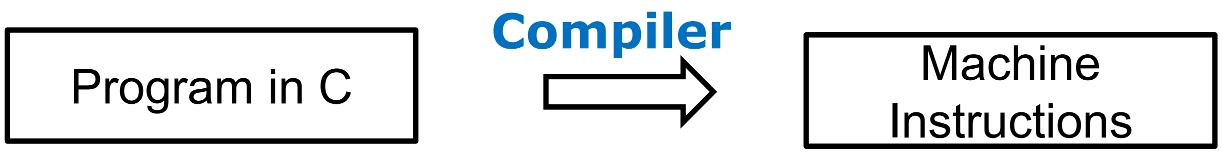
- Application software
  - e.g., Word processor, Media player, Web browser, …
  - Usually millions of lines of code
  - Use sophisticated libraries
  - Written in **high-level language (HLL)** – e.g., C, Java, Python, HTML, ...
- Hardware in a computer
  - CPU, memory, I/O, …
  - Can only execute extremely simple **low-level language** called **machine instructions**
  - Otherwise, CPU hardware will be very complex and expensive!
- Need something in-between…



Below Your Program

- System software
  - Operating System: Interface between S/W and H/W
    - e.g., Windows, MacOS, Linux
    - Handles service from application software
    - Manages hardware resources (e.g, CPU, memory, I/O, …)
  - **Compiler**
    - translates **HLL** code (e.g., C, C++, Java) into **instructions** that H/W can execute
    - Remember compiling your first C program “Hello World”?



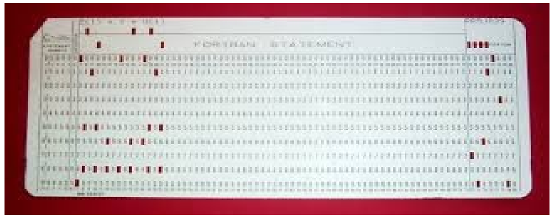

Program in C

Compiler

Machine Instructions

Machine Language (Instructions)

- **Instructions** are collection of ***bits*** that the computer understands and obeys
  - e.g., the instruction 10011001010000 tell one computer to add two numbers.
- The first computer programmers communicated to computers in binary numbers
  - Too tedious, hard to recognize



Punched card from a Fortran program.

- Invented new notations that were closer to the way humans think
  - Translated the new notations into binary codes = **assembly language**

Assembly Language

- Programmer would write,

```
add A, B
```

→ Assembly Language

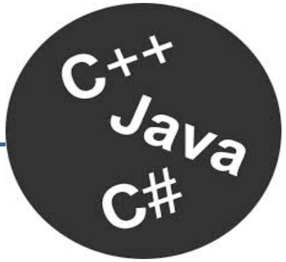
- **Assembler** would translate this notation into

```
10011001010000
```

→ Machine Language

- Assembly language requires the programmer to write one line for every instruction that the computer will follow
  - The programmer has to *think like* the computer - Why is this a problem?
  - Note, machine instructions must be very simple! - otherwise, CPU hardware will be very complex and expensive.
- Program can be written to translate a more powerful (or high-level) language

High-level language (HLL)



- e.g., C, C++, Java
- Allows programmers to think in more natural language
  - Algorithms: set of steps that define how a task is performed
- Improve programmer productivity – less time to develop programs
  - Programming: use a computer to solve a problem
- Programming languages are independent of the computer – HLL can be run in any computer machine

C code

```
while (save[i] == k) i += 1;
```

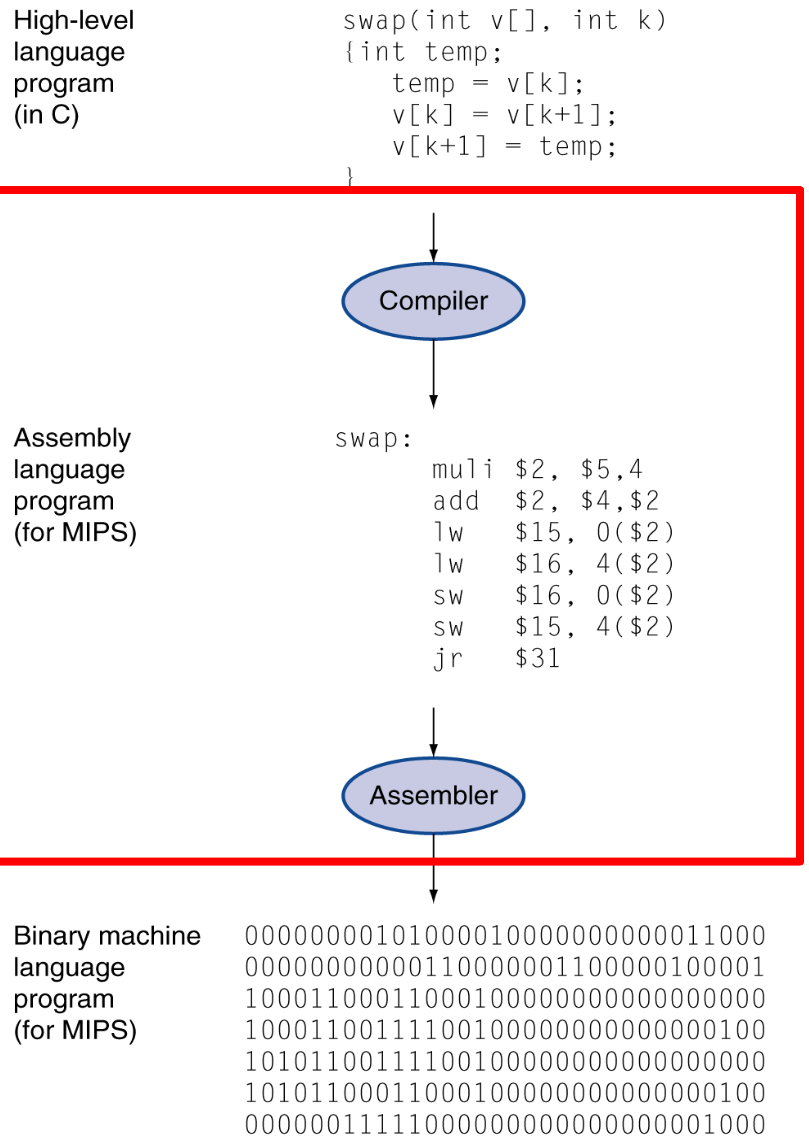
vs.

Assembly code (machine instructions)

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```

Levels of Program Code

- The below two steps can generally just be called “**Compiler**”
  - Compiler: Translate HLL into Assembly language
  - Assembler: Translate Assembly language into Machine language



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Chapter 1: Introduction

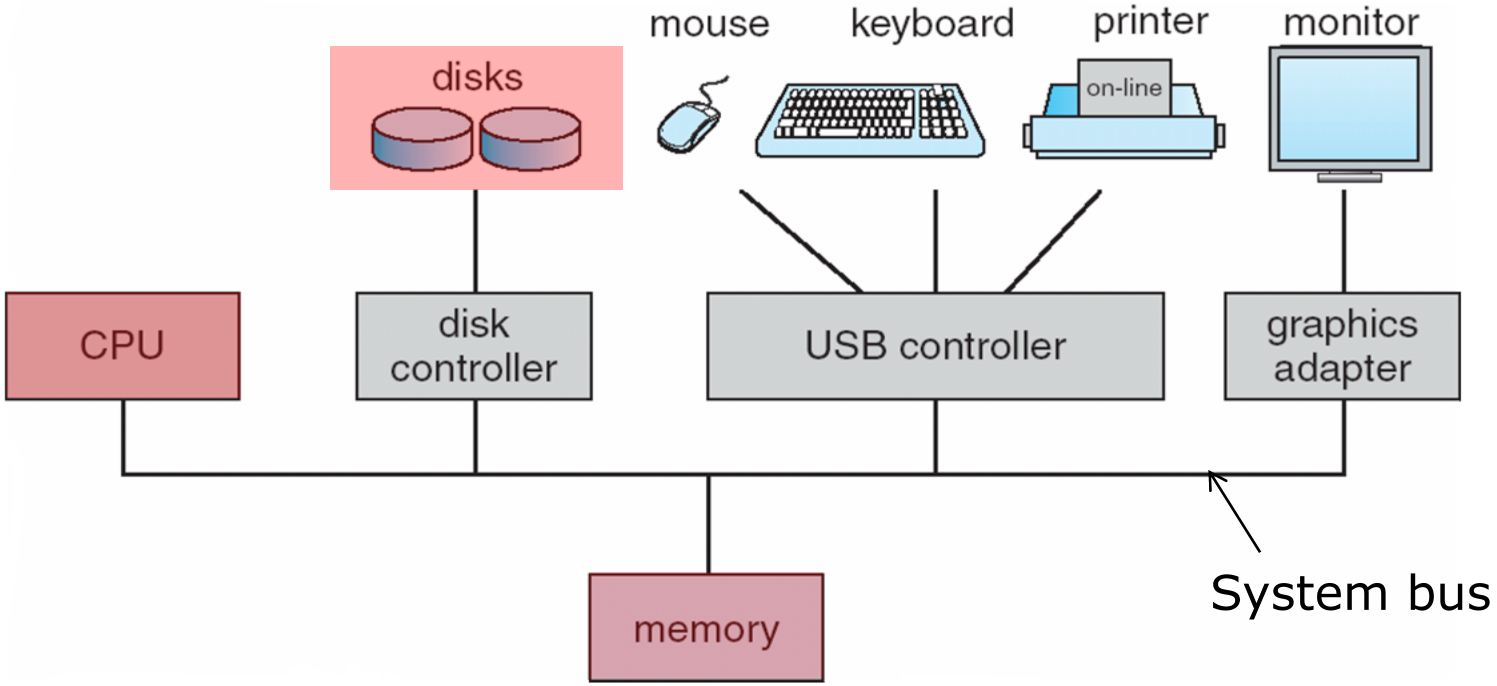
- What is an Operating System?
- What Operating Systems Do
- Computer-System
  - Below your Program
  - Organization, Architecture (Ch. 1.2)
- Computing Environments
- Open-Source Operating Systems

Why Computer Systems?

- It is important to understand the organization and architecture of computer hardware.
- This includes the CPU, memory, and I/O devices, as well as storage.
- A fundamental responsibility of an operating system is to allocate these resources to programs.

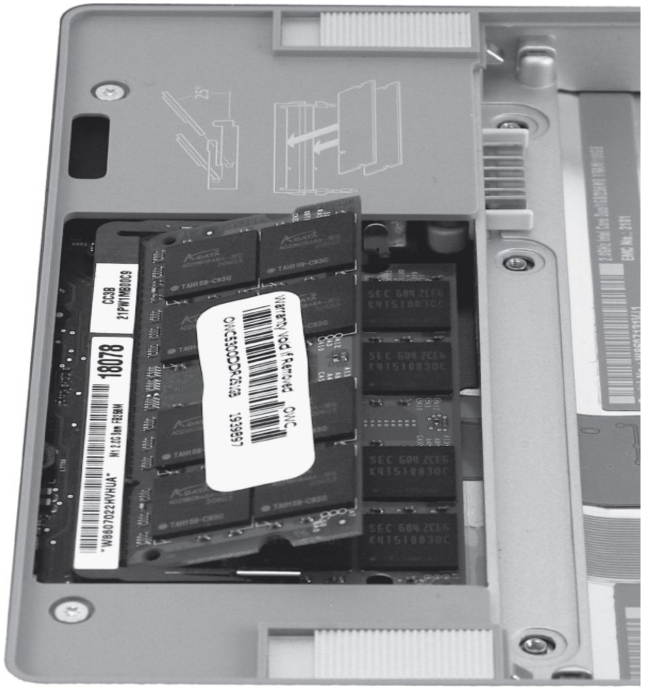
Computer System Organization

- Computer-system operation
  - CPU, I/O device controllers connect through common System bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Basics: Memory (RAM)

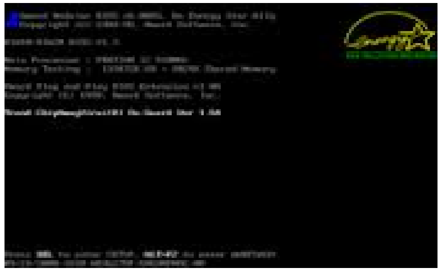
- The memory contains
  - **programs** when they are running (a running program is called a ***process***) – non-running programs are usually stored in ___________
  - **data** needed by running programs
- PC memories are usually built from DRAM (Dynamic Random Access Memory) chips.
  - Random access?
    - memory access takes basically same amount of time no matter what portion is read
    - hard disk is sequential access



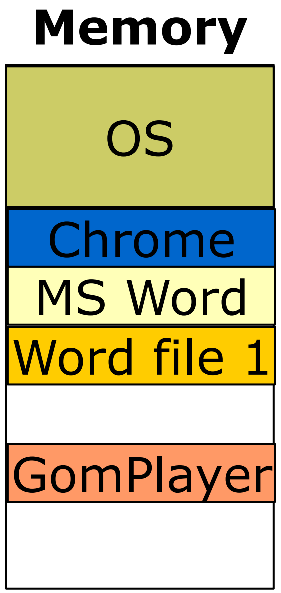
Basics: Memory (RAM)

- How stuff works: [link]
- Your first reading assignment
  - Read this article very carefully (there are **5** pages!!)
  - Will be included in quizzes and discussions
- Learn
  - Why do we need RAM to store data, when we already have hard disk?
  - What happens when you turn your computer on until you shut down?
  - Why are there so many memory systems?
  - What are cache and Registers?

Basics: Memory (RAM) – Turn computer on

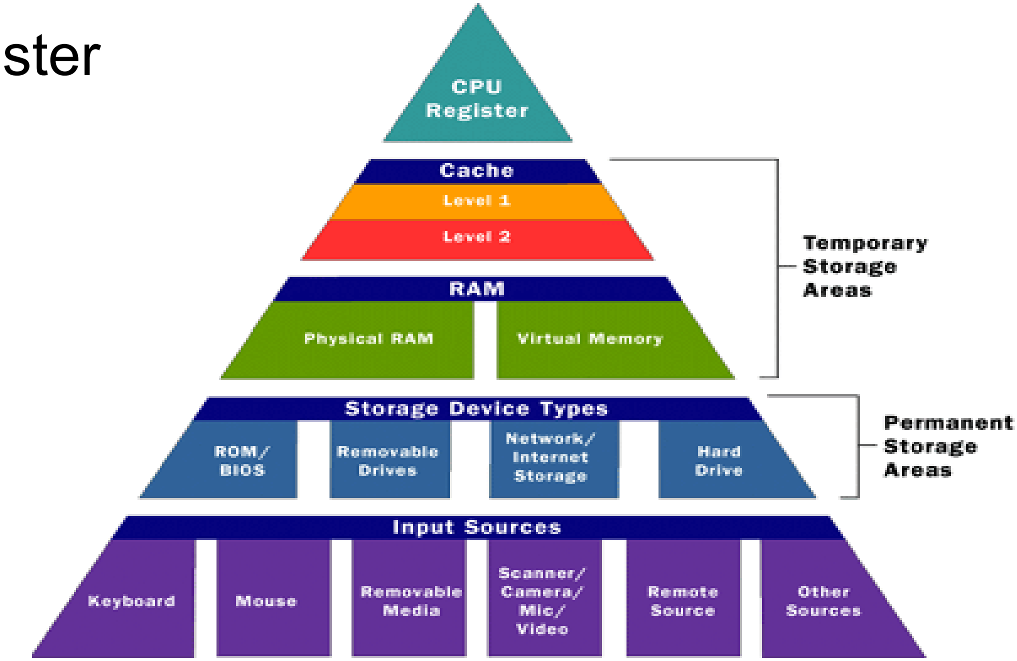


- CPU – hard drive vs. CPU – memory
  - large storage vs. high speed
- Turn computer on
  - **bootstrap program**
    - Initializes the system: test memory and other hardware
    - load OS to memory (from __________)
  - Run application – loaded to _______ (from disk)
  - Open file – loaded to RAM
  - Save/close program/file – written to storage (e.g., hard disk) and removed from RAM

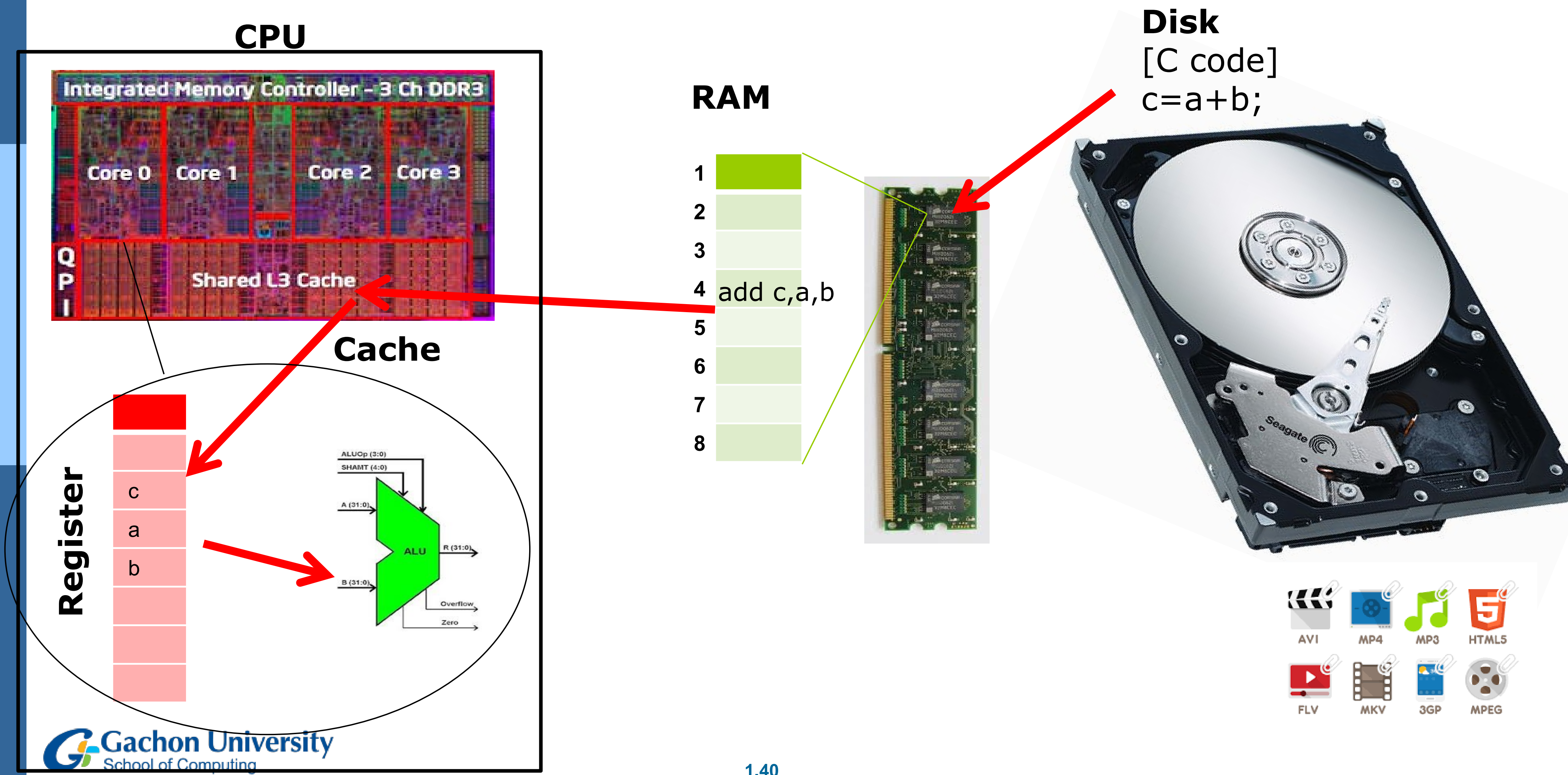


Basics: Types of Memory

- Register, level 1, 2, 3 cache, RAM, (SSD), hard disk, …
- Why so many?
  - CPU is very fast – 5GHz roughly means 5x10^9 CPU cycles per second.
  - CPU needs quick and <u>large memories</u>
  - Hard disk is cheap but slow, RAM is faster but expensive. → Tradeoff!
- Memory Speed
  - Register > Cache > RAM > Hard disk
- Price (per bit)
  - Same as above



Basics: From Disk to Memory to CPU



CPU

Cache

Register

RAM

Disk

[C code]

c=a+b;

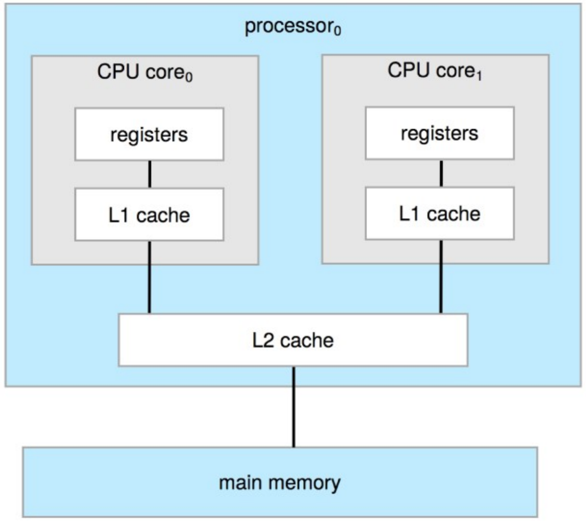
1	🟩
2	
3	
4	add c,a,b
5	
6	
7	
8	

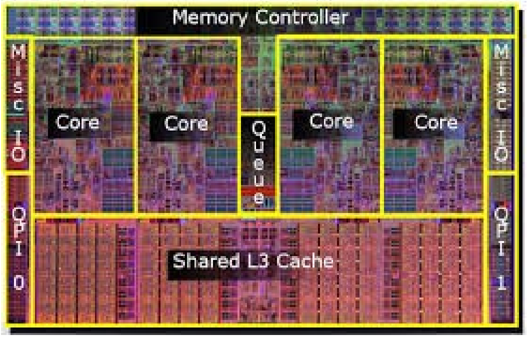
Single-core processor

- Many years ago, most computer systems used single-core processors
  - **Core**: executes instructions and registers for local data store
  - **Single-core**: 1 main CPU with 1 core that executes **general-purpose** instructions (from processes)
    - Most systems use a general-purpose processor (smartphones through mainframe servers)

Multicore processor

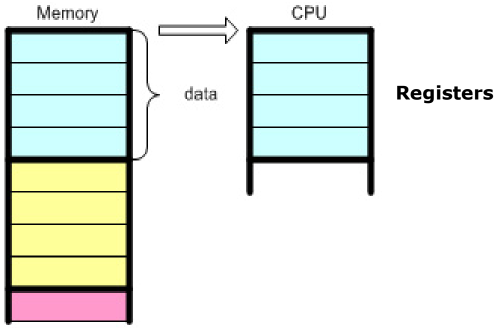
- Modern computers use **Multicore processors** (CPUs)
- **Multicore processor** systems growing in use and importance
  - More than one *core* per chip
  - Advantages include: More work in less time (tasks/s)
    - When multiple cores cooperate on task, a certain amount of **overhead** is incurred in keeping all the parts working correctly





Basics: CPU operation (More in computer architecture)

- To run an **instruction**; load instructions from memory to CPU register
  - C program code:
    - `a=0; b=c=1;`
    - `a = b + c;`
  - Compiled **assembly code** (or **machine code**)
    - `add $s1, $s2, $s3`
    - `(10011111100110010110101011011)`
  - Basic steps to run instruction
    - Step 1: Load machine code instruction from Memory to CPU Registers
    - Step 2: Run machine code instruction at CPU
    - Go to Step 1 and run next instruction

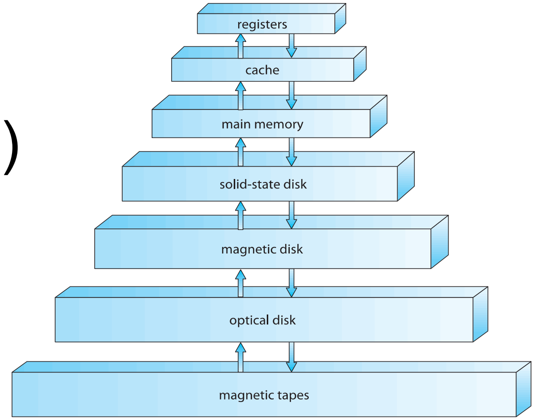


Basics: Storage Structure

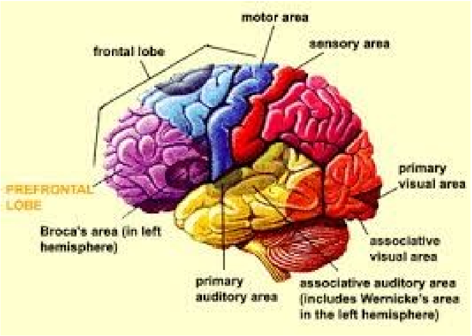
- **Secondary storage**
  - extension of main memory that provides large **nonvolatile** storage capacity
  - **(Magnetic or Hard) disks** – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
    - The **disk controller** determines the logical interaction between the device and the computer
  - **Solid-state disks (SSD)** – faster than magnetic disks, nonvolatile
    - Mostly flash technology
    - Becoming more popular

Caching Concept

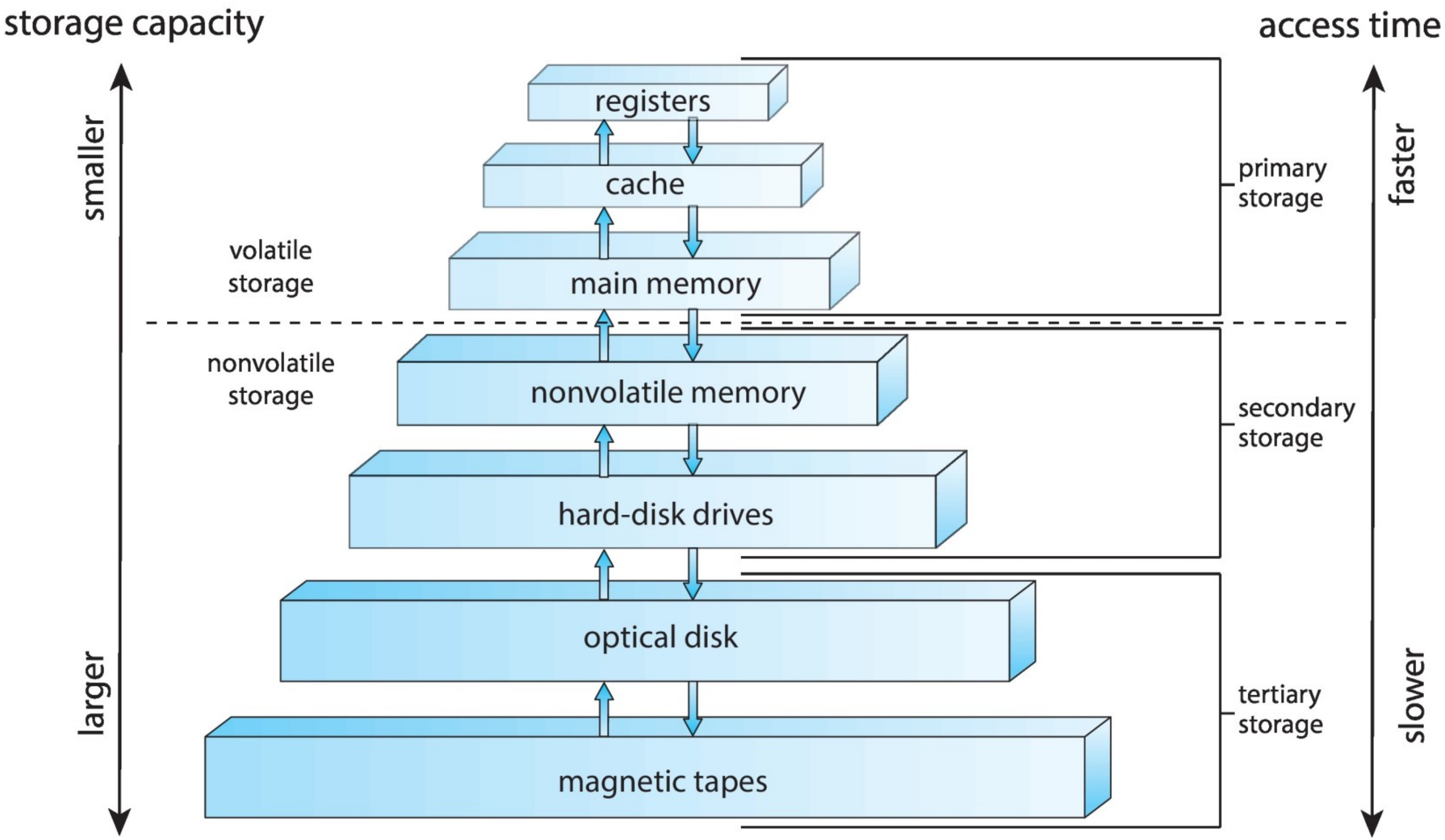
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there



Storage Hierarchy



http://thebrain.mcgill.ca



Performance of Various Levels of Storage

Volatile Non-volatile

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Faster

Cheaper (per byte)

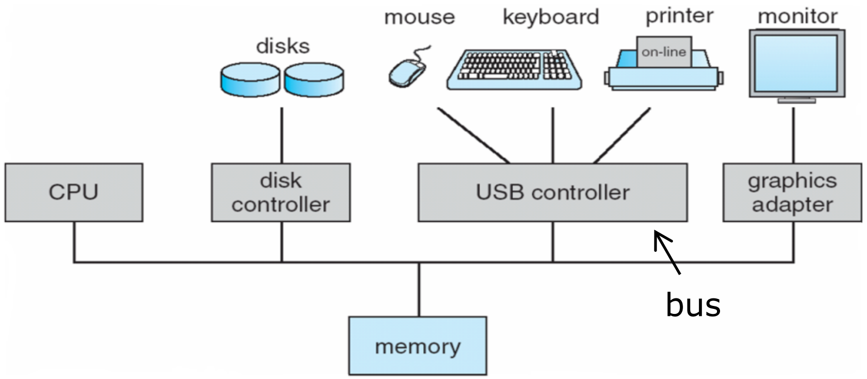
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

- **"Caching concept"** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

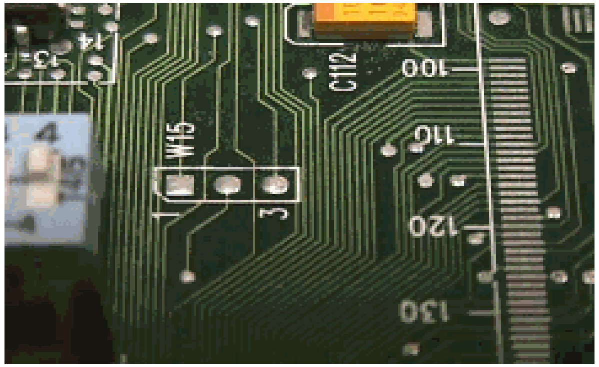
Connecting Processors, Memory, and I/O

- Processor and memory have to communicate
  - Run a program (Internet Explorer) – processor loads the programs to memory and executes.
- Processors and I/O devices; hard disks, DVD, memory, network cards
  - In the above example, program is loaded from hard disk to memory
  - Open “google.com” in Internet Explorer – processor commands network card to fetch google.com webpage



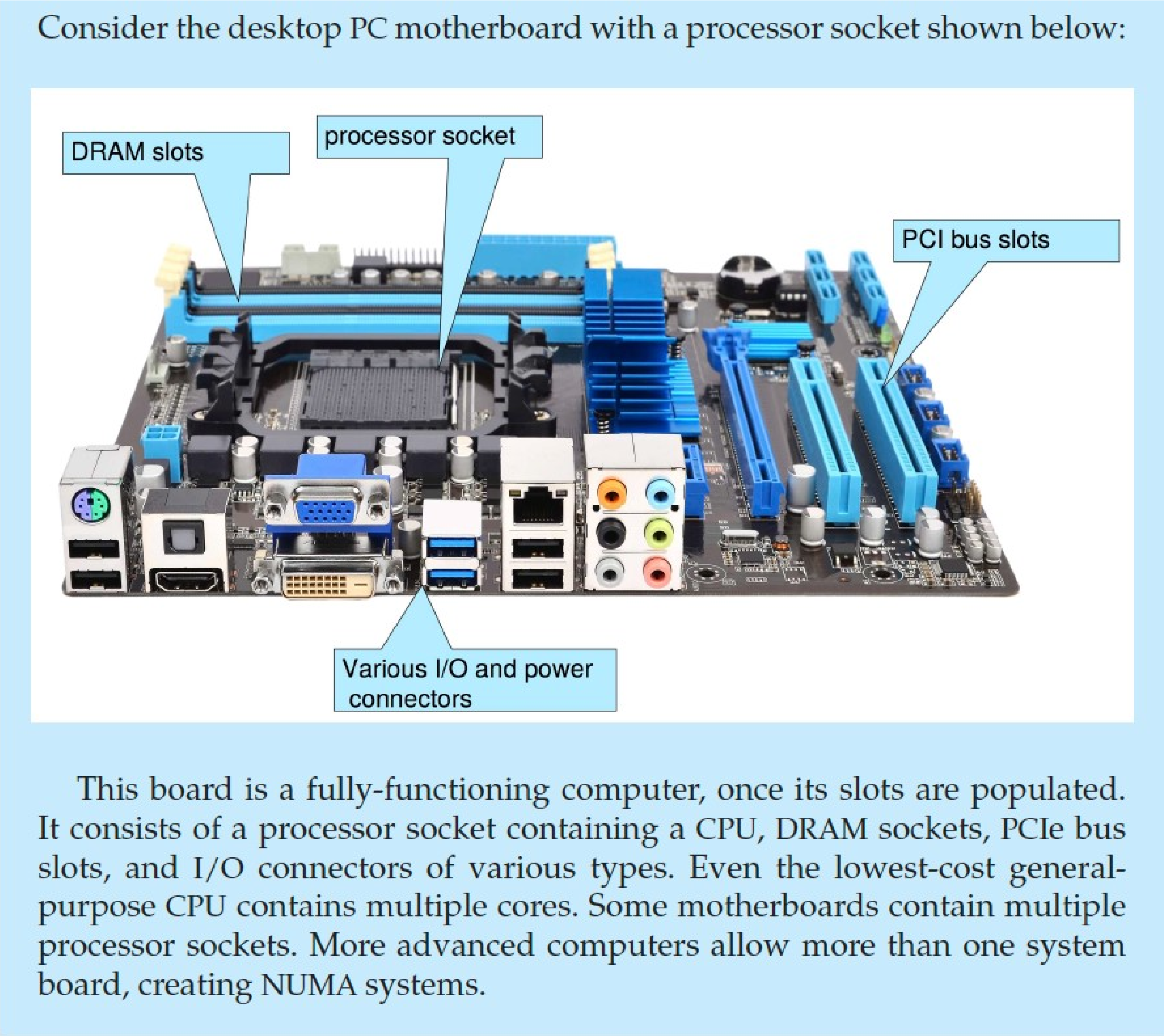
Connecting Processors, Memory, and I/O

- Traditional method: use ***bus***
  - from Latin word *omnibus* meaning 'all'
  - shared communication link for devices
  - supported by the motherboard
  - processor-memory bus, backplane bus



Microcomputer Bus:
http://dl.uncw.edu

PC Motherboard



Consider the desktop PC motherboard with a processor socket shown below:

This board is a fully-functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

Chapter 1: Introduction

- What is an Operating System?
- What Operating Systems Do
- Computer-System
  - Below your Program
  - Organization, Architecture
- Computing Environments
- Open-Source Operating Systems

Free & Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
  - Closed source: MS Windows, MAC OS X and iOS
- Open-source benefits
  - Programmers can contribute to the code by help debug, analyze, provide support.
  - Security?
- Examples include **GNU/Linux** and **BSD UNIX**, and many more

GNU/Linux

- Linus Torvalds (1991), a student in Finland, released an UNIX-like kernel and invited contributions worldwide.
    - Anyone can download the source code via Internet, modify it and submit the changes to Torvalds.

- There are now hundreds of unique distributions (or custom builds) of the Linux system.
    - RedHat, SUSE, Fedora, Debian, Slackware, and Ubuntu.

- Can use VMM like VMware Player (Free on Windows), VirtualBox (open source and free on many platforms - http://www.virtualbox.com)
    - Use to run guest operating systems for exploration