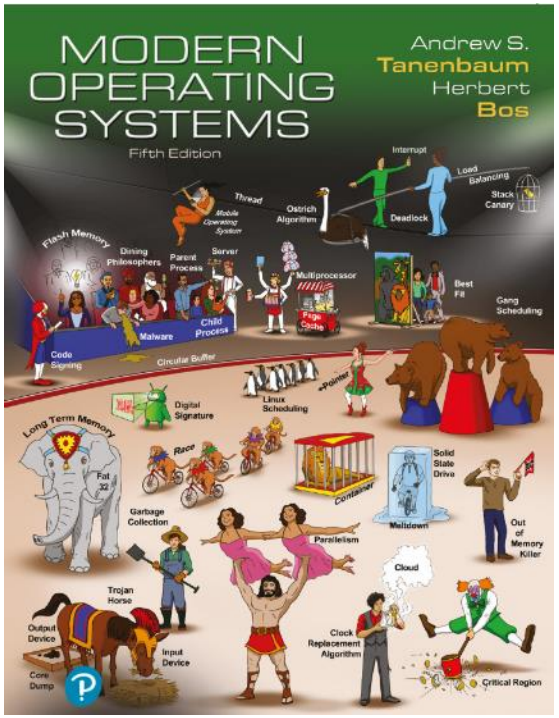
COMP 4736
Operating Systems

Instructor: Rahim Oraji

Name : Rahim
Email : roraji@bcit.ca

Office Hours : Online appointment

Course outline: https://www.bcit.ca/outlines/comp4736



Modern Operating Systems (5th Edition), Andrew S. Tanenbaum & Herbert Bos (Recommended)

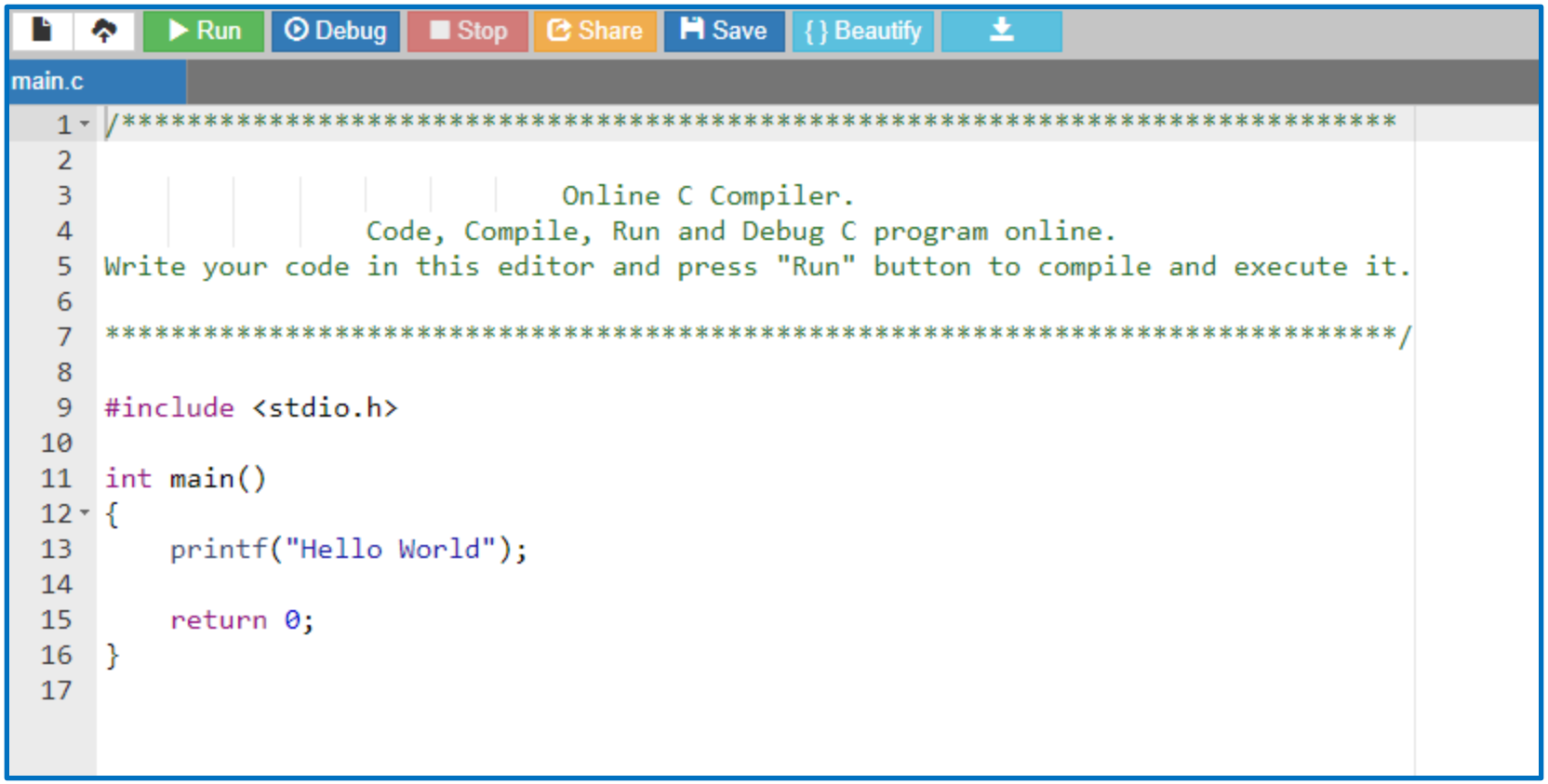
- *Operating Systems: Three Easy Pieces*, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau (University of Wisconsin-Madison).
- *Operating Systems – Internals and Design Principles*, William Stallings, Ninth Edition.

(Optional)

Lab

Programming Language: C

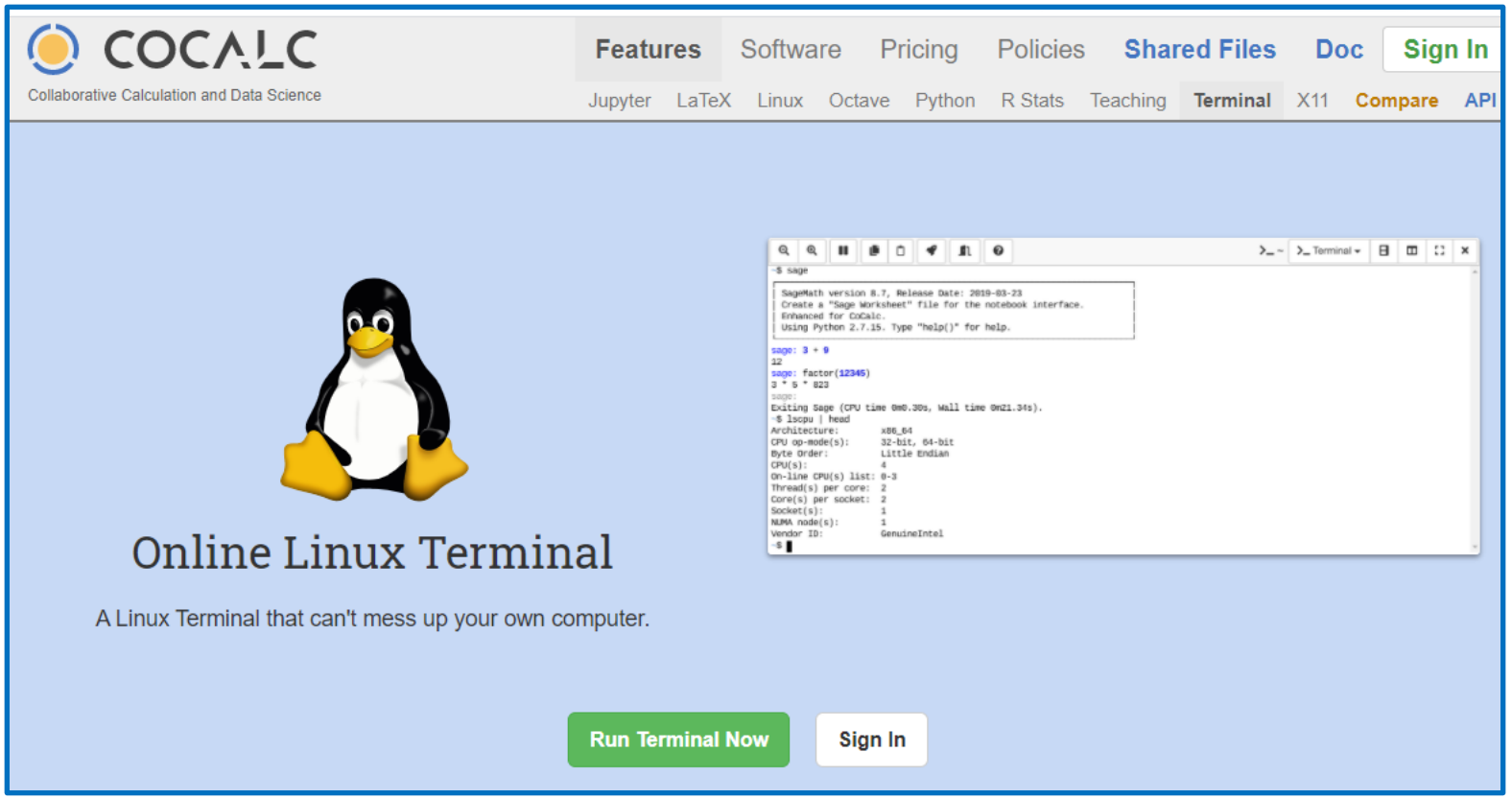
https://www.onlinegdb.com/online_c_compiler



Lab

OS: Linux

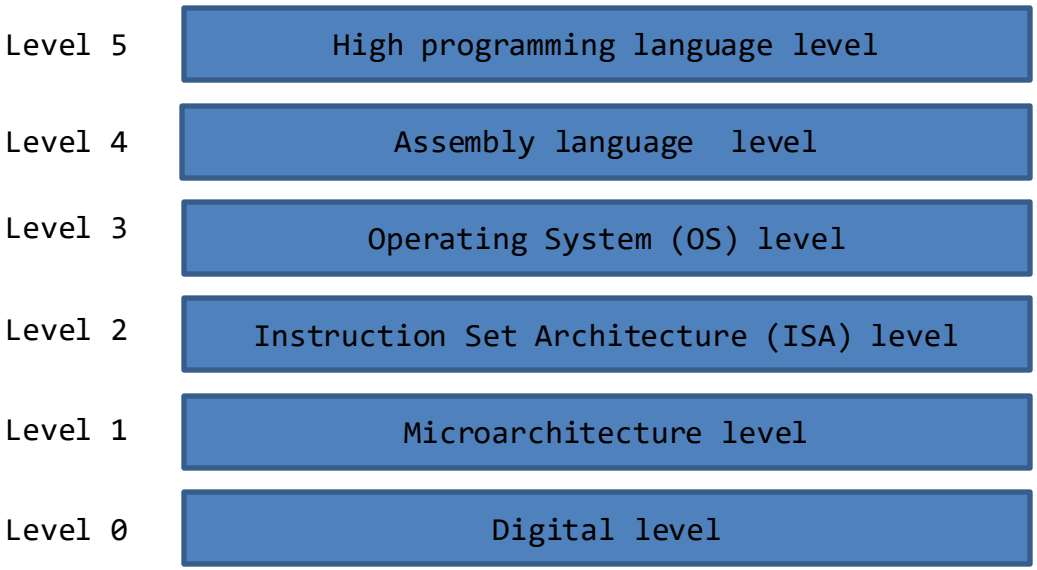
https://cocalc.com/doc/terminal.html

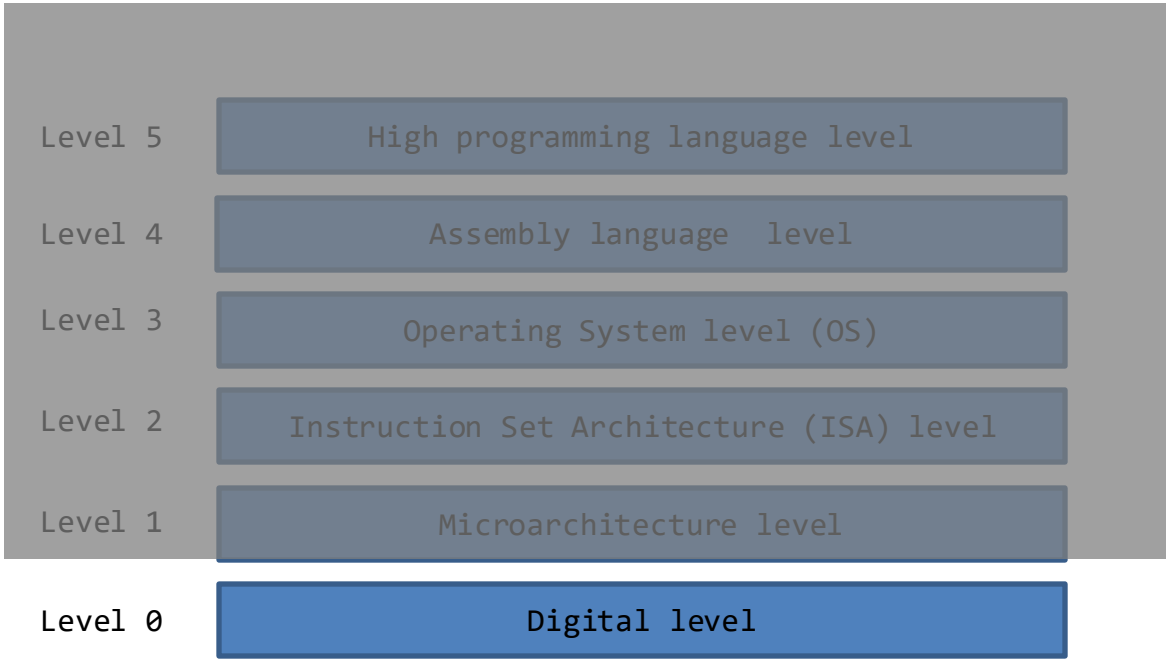


1) Computer Architecture And Operating System Overview
2) Operating System Structures and Clocks
3) Processes
4) Threads
5) System Calls
6) Inter Process Communication
7) Deadlocks
8) Memory Management

Introduction

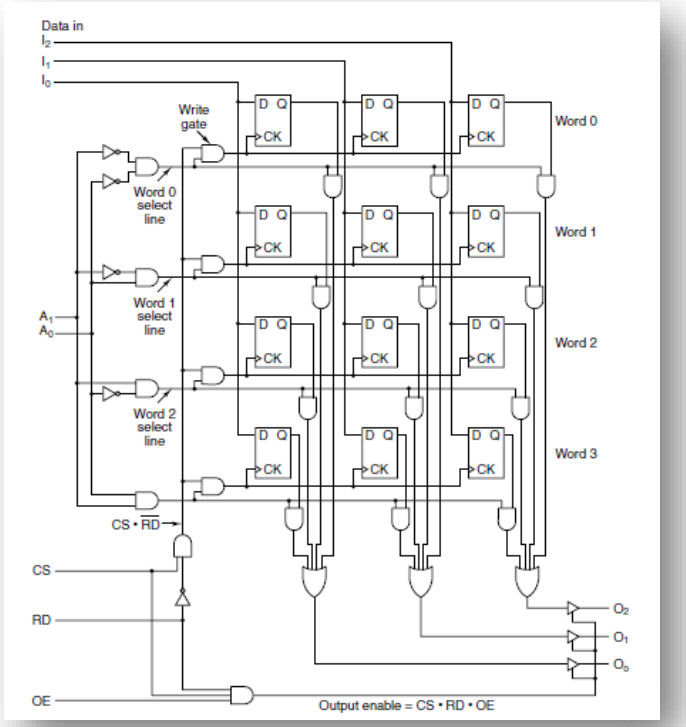
Current Multilevel Machines

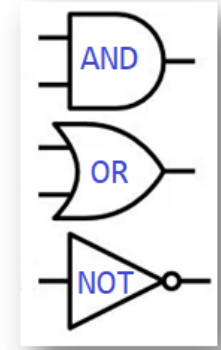


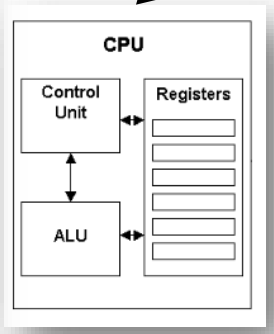


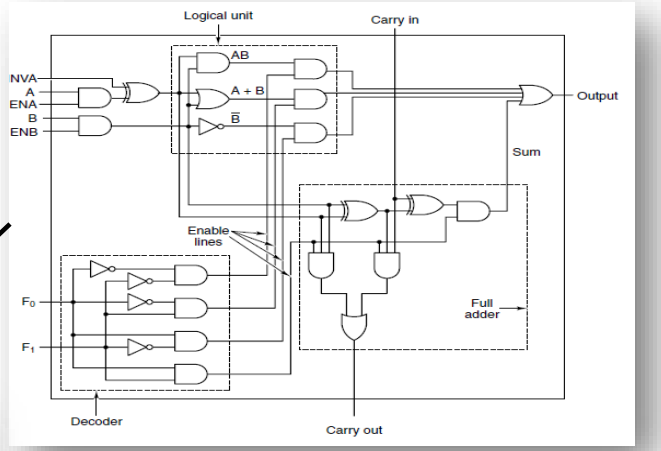
Device level made of transistors and in turn transistors are made of semiconductors. AND, OR, and NOT are the building blocks of any digital circuits

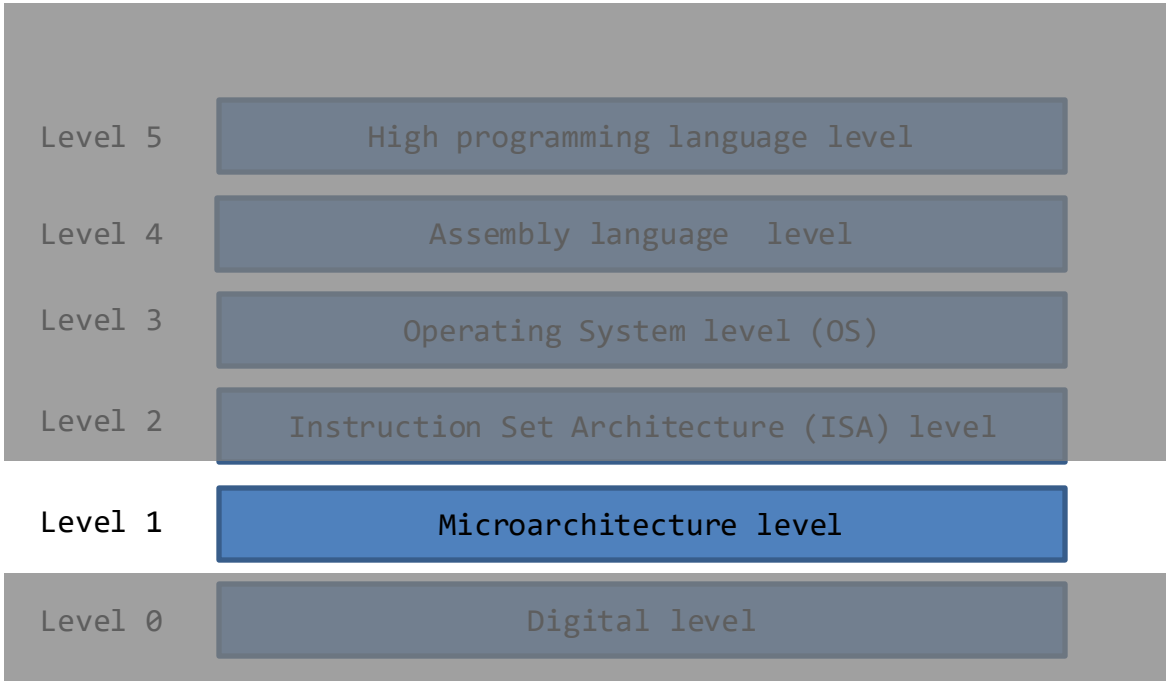
Digital level



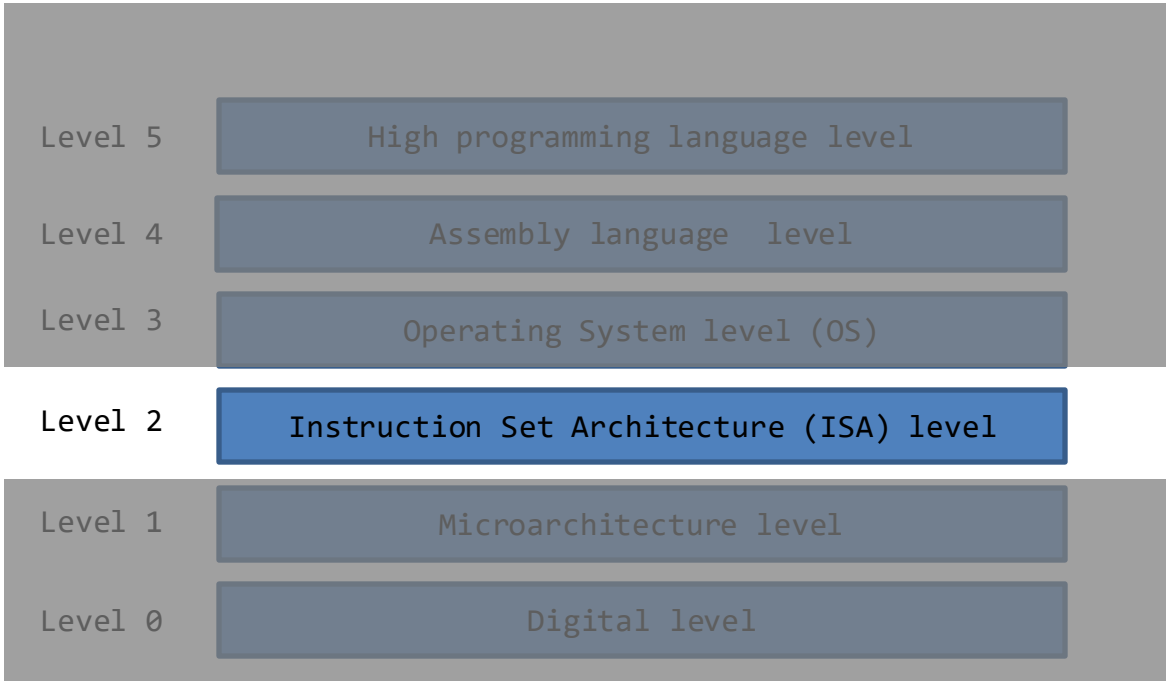








Assume that a program written in a high level language is translated to the low level machine language. A file is created after compilation containing some instructions in machine codes. The instructions are executed in this level employing ALU, Registers, and Microprograms.

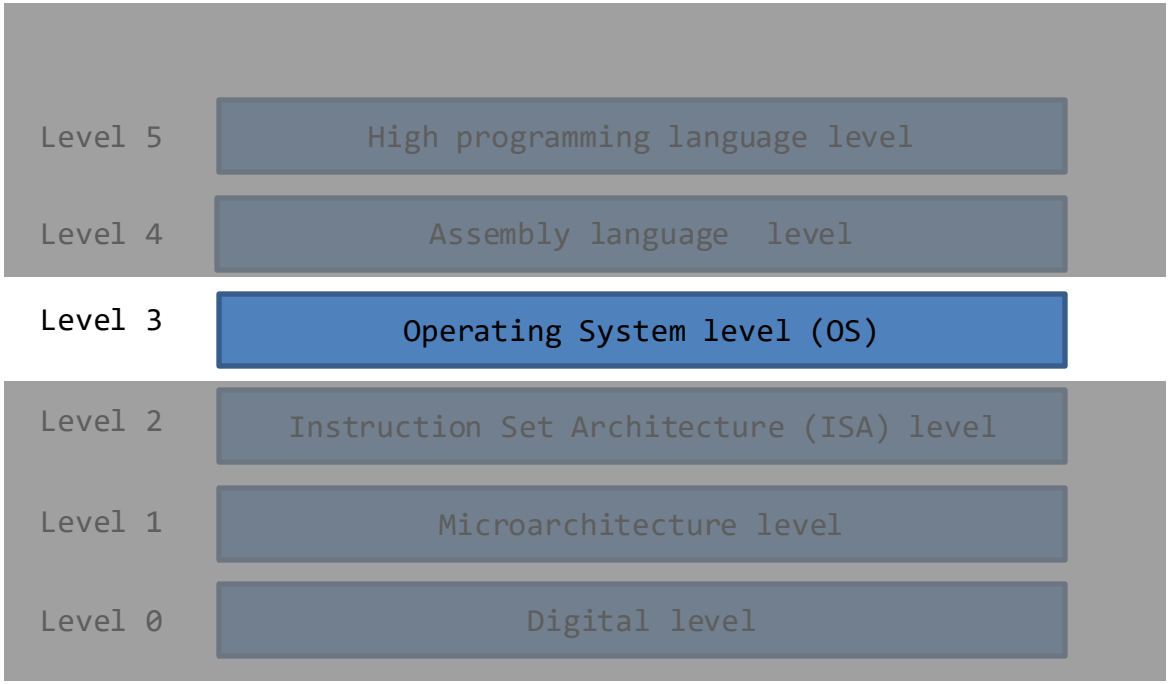


An ISA describes the design of a Computer in terms of the basic operations it must support. The ISA is not concerned with the implementation specific details of a computer. It is only concerned with the set or collection of basic operations the computer must support.

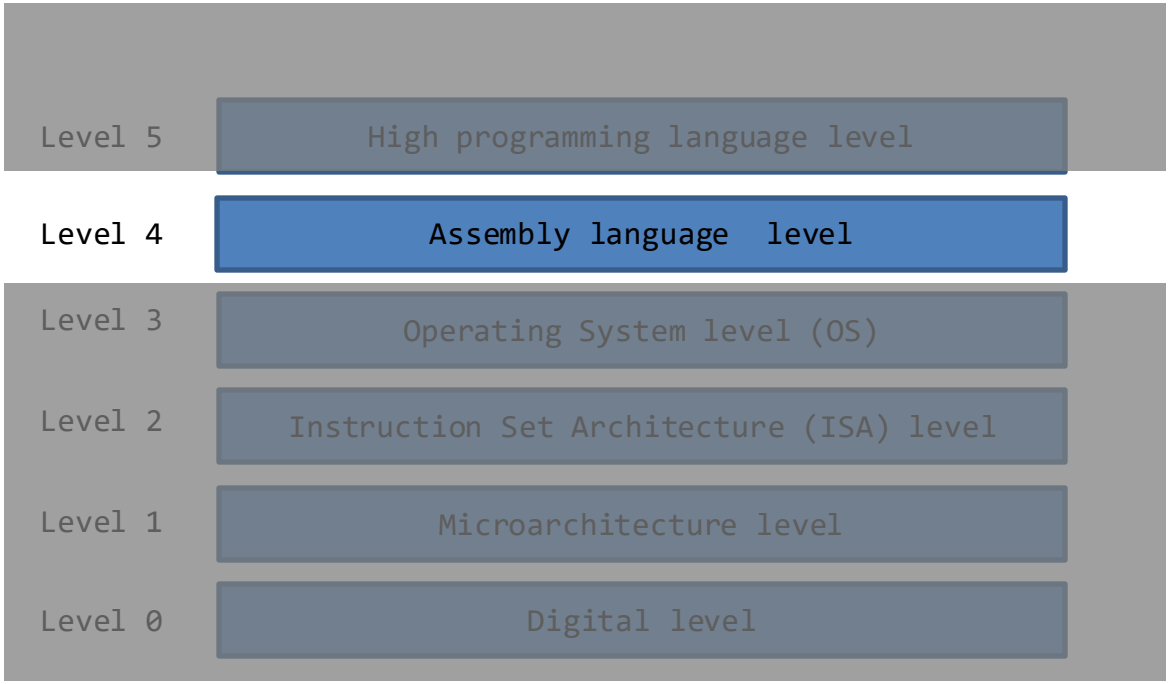
ATmega16 (8-bit AVR microcontroller)

Instruction Set Summary

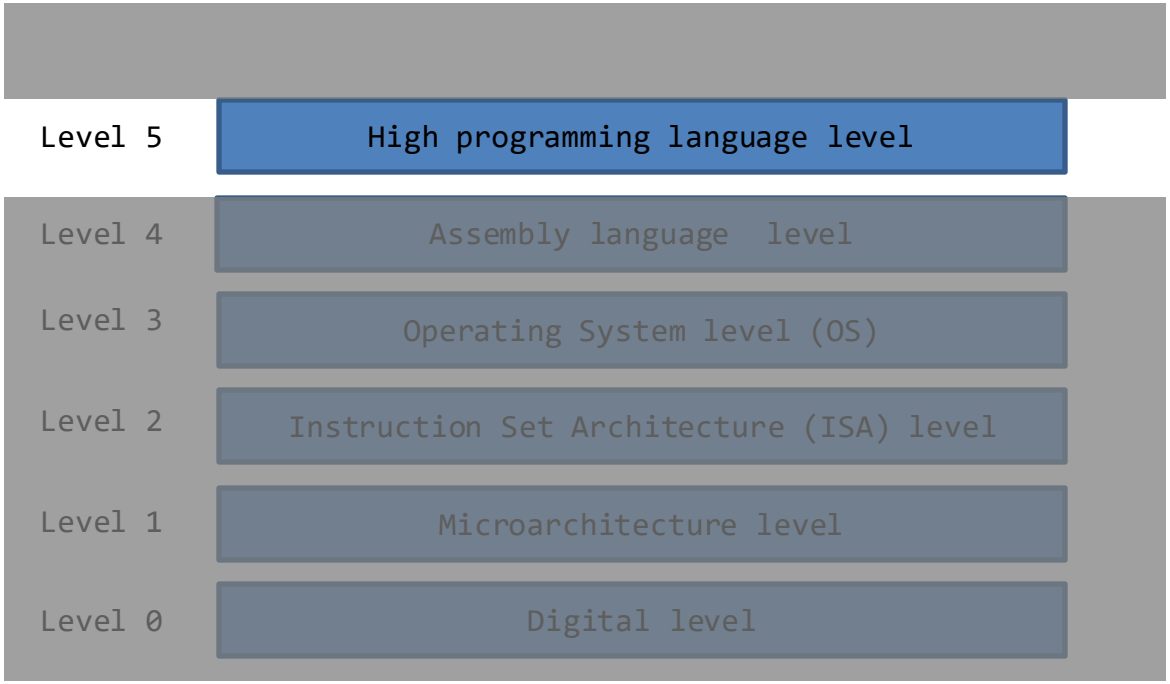
Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1



Main duties : Memory management, processes execution, and system resource protection.

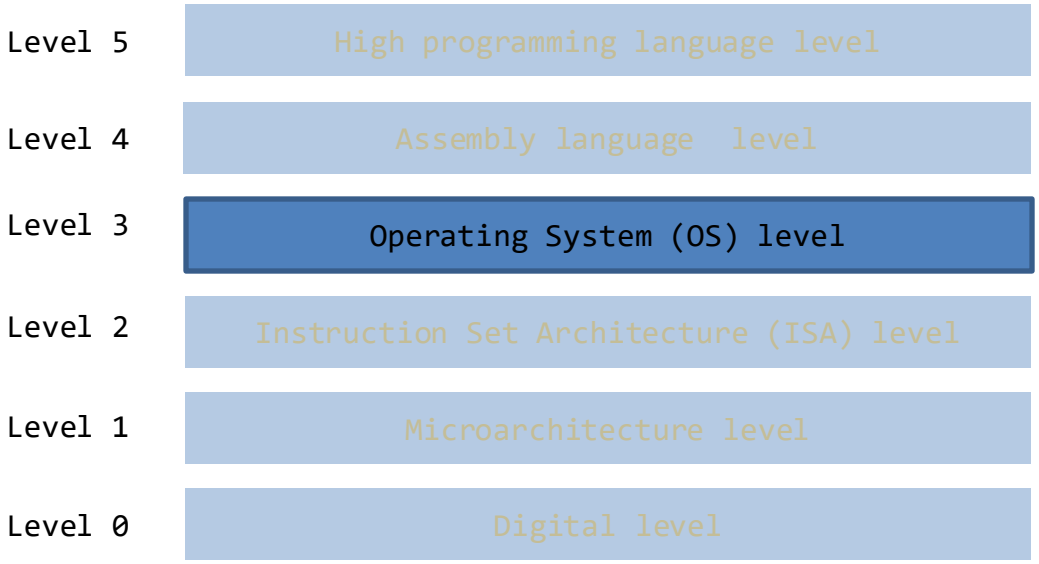


Human readable language, hard to work with, time consuming, and even a simple program contains many lines.



In this level people usually write programs in languages such as Java, Python, and C++.

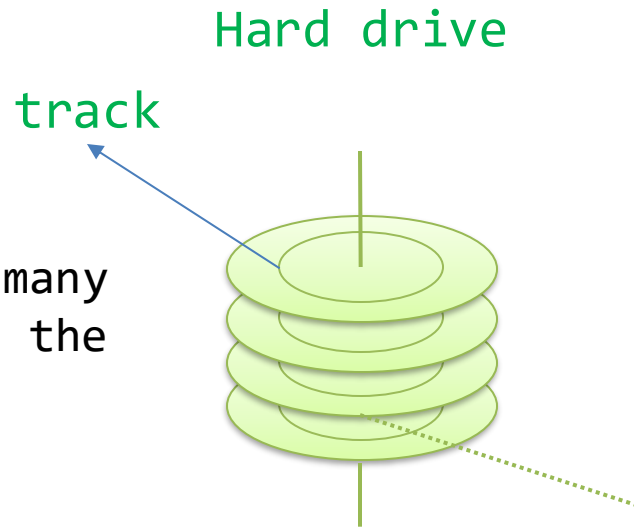
Operating System (OS)



Main duties : Memory management, processes execution, and system resource protection. e.g. Linux, windows, Mac

Operating System

1. A software
2. An extended machine
3. A resource manager that provides many abstraction layers



e.g.
Reading and writing on hard drive require many steps and is not easy task without using the operating system.

<u>OS-abstraction</u>

*ssize_t read(int fd, void *buf, size_t count);*

Kernel and User mode

kernel mode (also called **supervisor mode**)

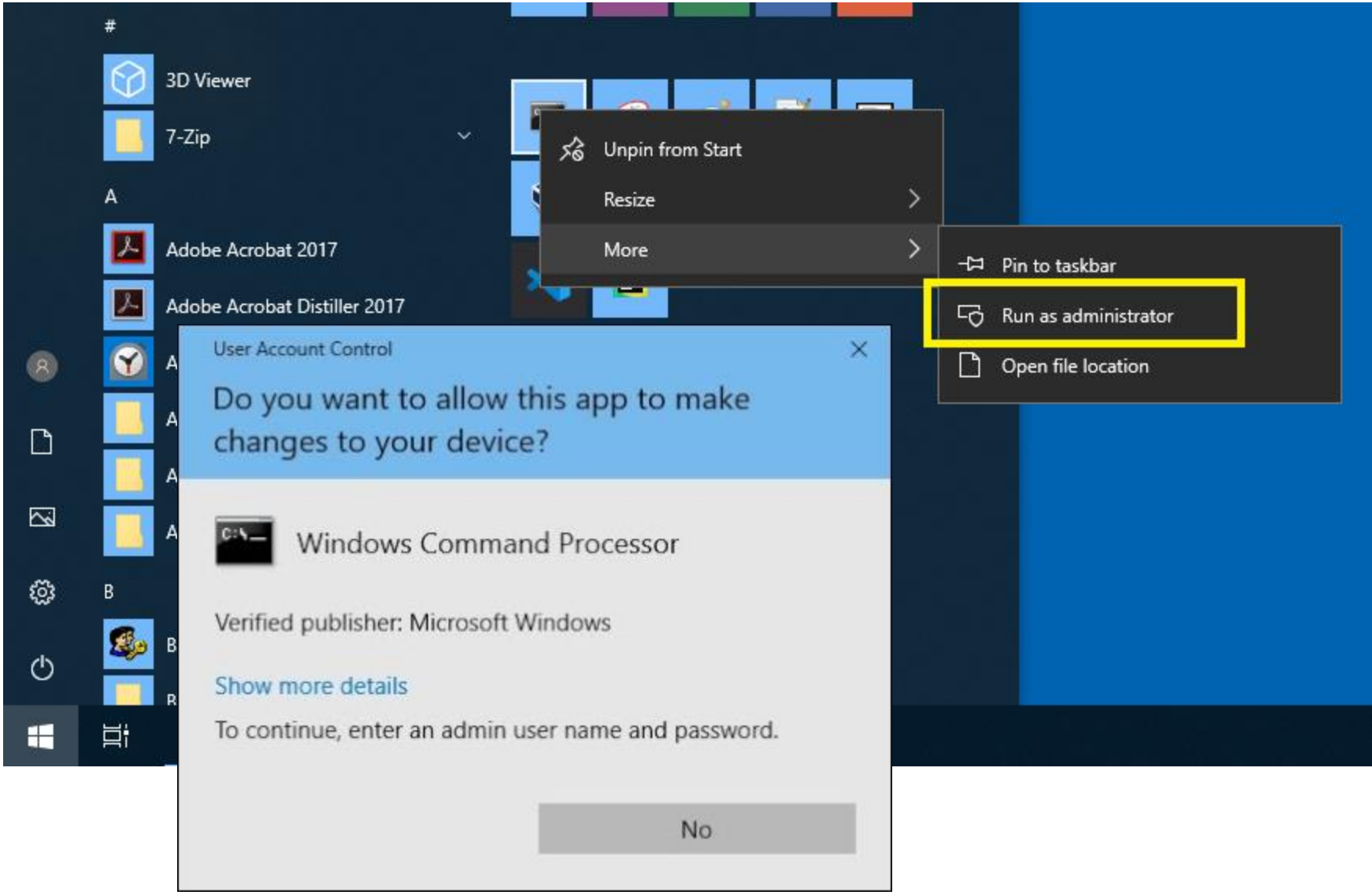
The operating system, the most fundamental piece of software, runs in **kernel mode**. In this mode it has complete access to all the hardware and can execute any instruction the machine is capable of executing.

User mode

The rest of the software runs in **user mode**, in which only a subset of the machine instructions is available.

Instructions that affect control of the machine or do **Input/Output** are forbidden to user-mode programs.

Windows 10


#
3D Viewer
7-Zip
A
Adobe Acrobat 2017
Adobe Acrobat Distiller 2017
B
Unpin from Start
Resize
More
Pin to taskbar
Run as administrator
Open file location
User Account Control
Do you want to allow this app to make changes to your device?
Windows Command Processor
Verified publisher: Microsoft Windows
Show more details
To continue, enter an admin user name and password.
No

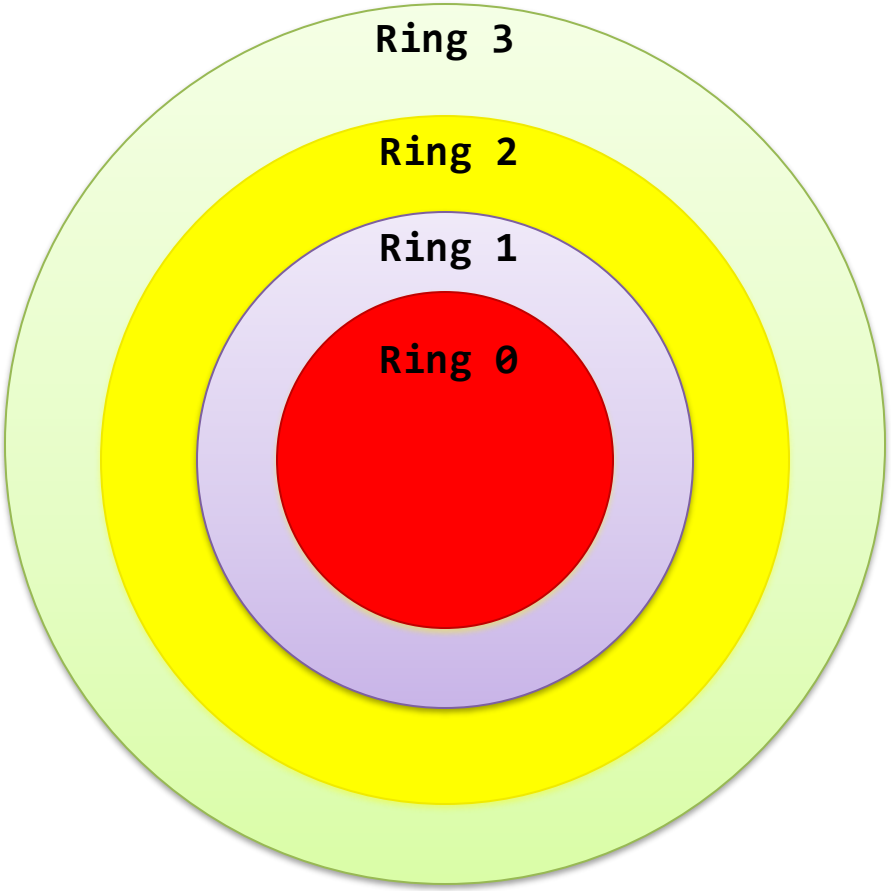
Ubuntu 18.04.4 LTS

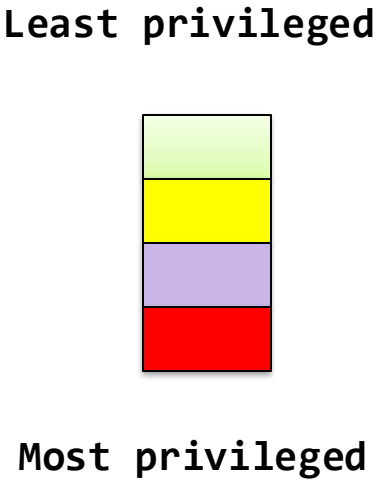
Uncomplicated Firewall (UFW)

Intel x86 architecture

Four privilege levels or security rings

These rings are used for various levels of privilege and access control within the CPU, typically in an operating system context.





Intel x86 architecture

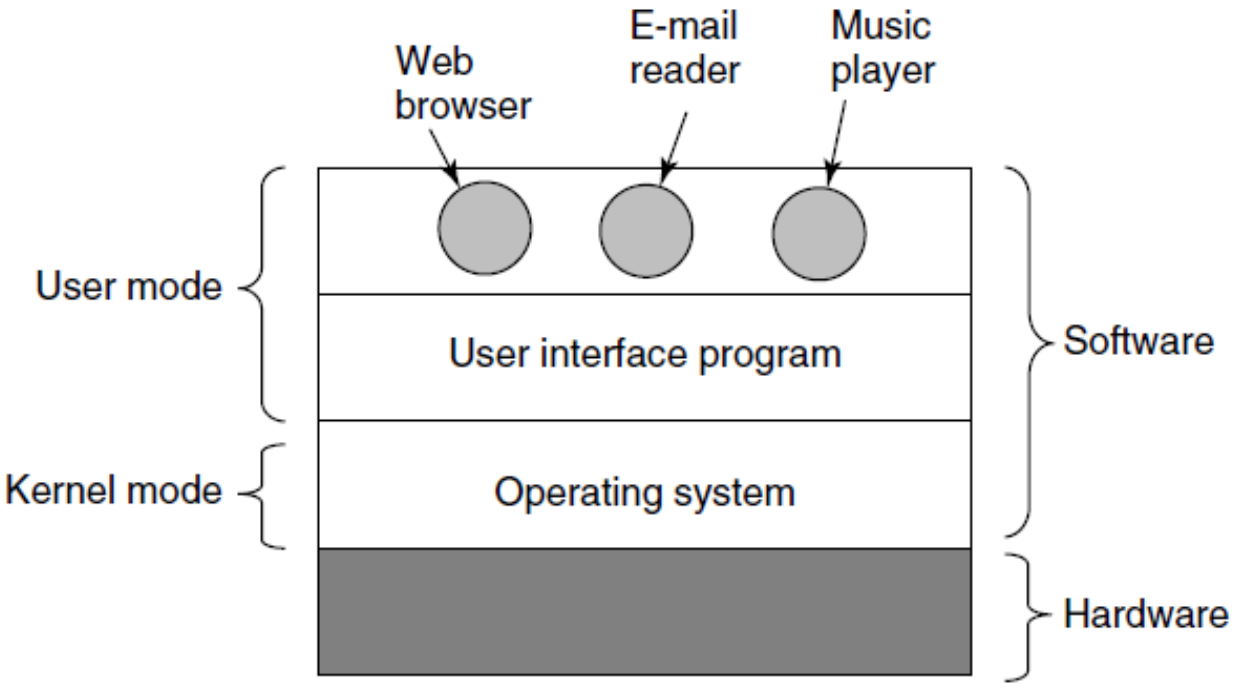
Four privilege levels or security rings

Ring 0 (Kernel Mode): This is the most privileged level, reserved for the operating system kernel. Code running in Ring 0 has unrestricted access to all hardware and system resources. It can execute privileged instructions and manage memory, devices, and processes. Device drivers and the core operating system components run in Ring 0.

Ring 1 and 2 : Operating System Services (Device Drivers, Etc.).

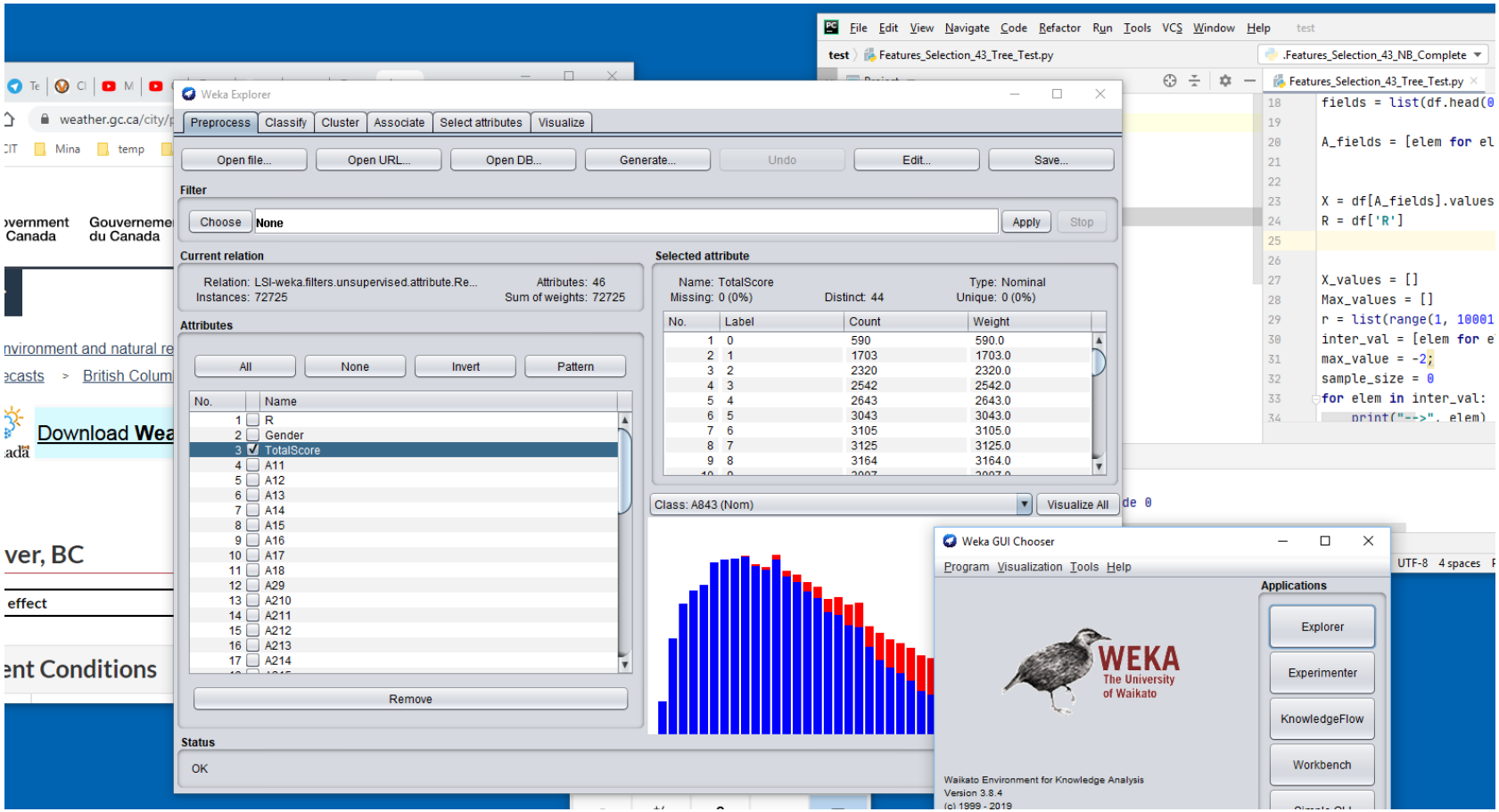
Ring 3 (User Mode): Ring 3 is the least privileged level and is where user applications run. Code running in this ring has limited access to system resources and hardware. It can't execute privileged instructions directly or access hardware resources directly. Instead, it relies on system calls to request services from the operating system.

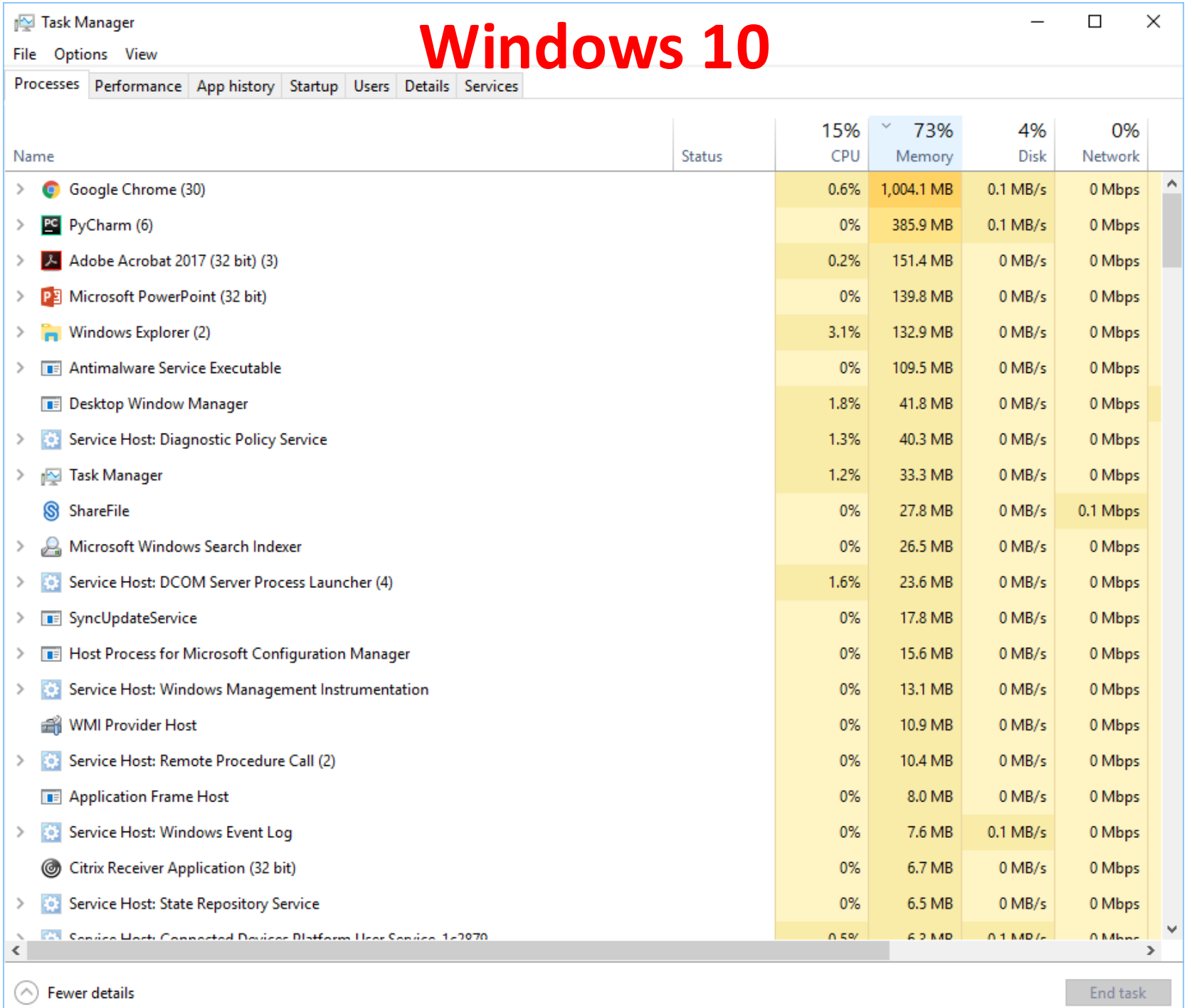
Components of a Modern Computer



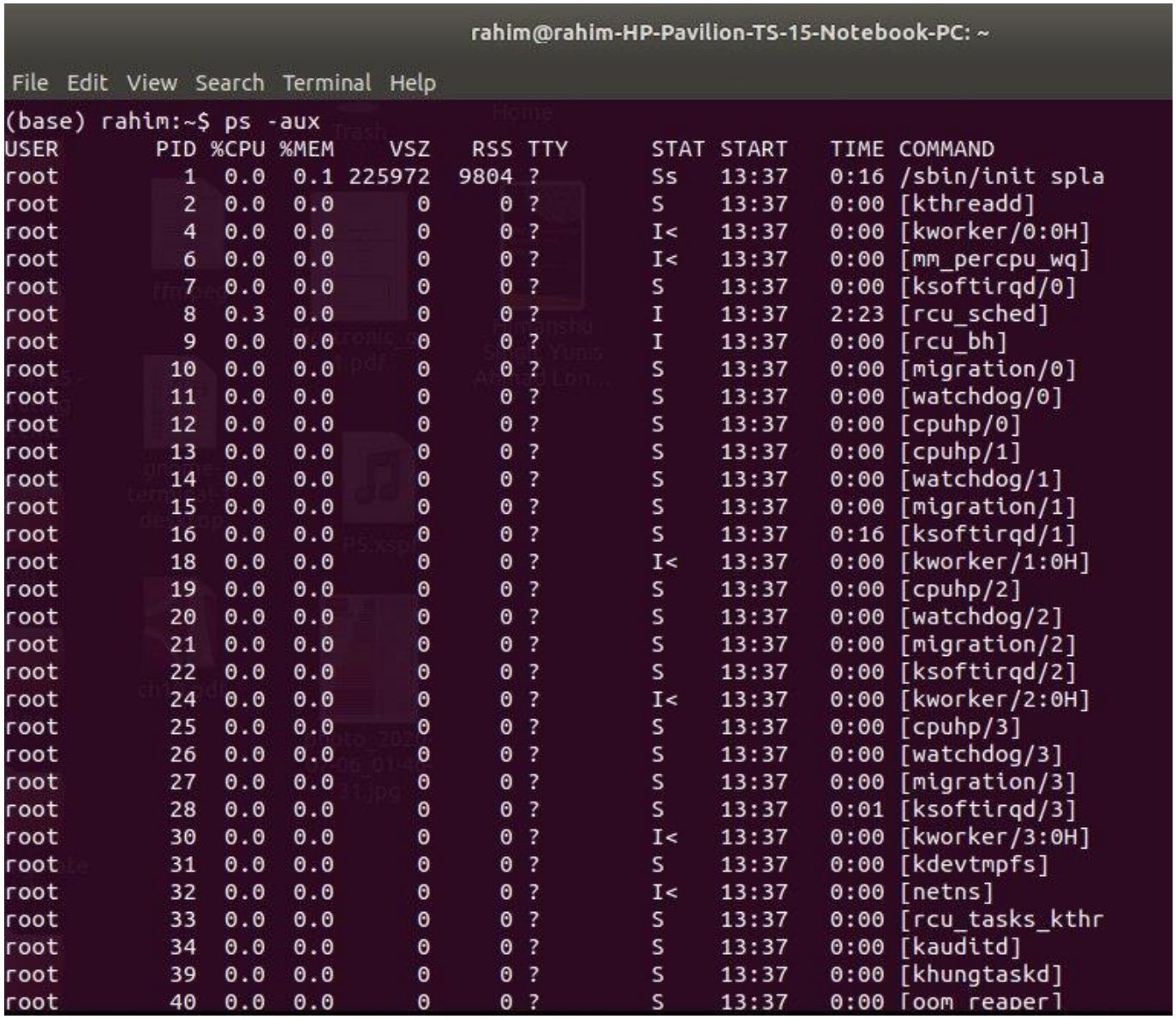
Where the operating system fits in.

Windows 10

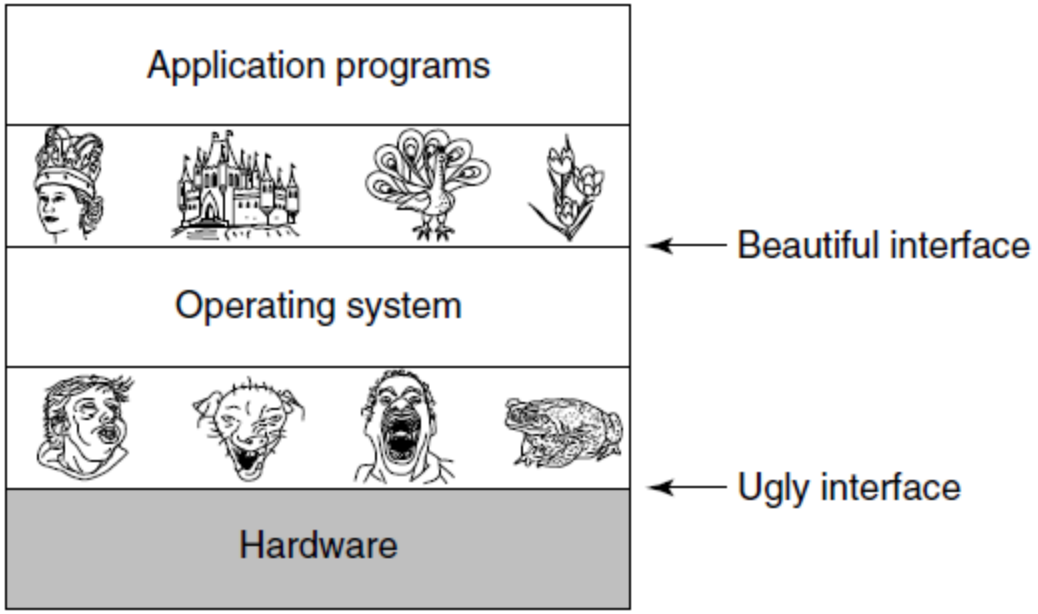

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: LSI-weka.filters.unsupervised.attribute.Re...
Instances: 72725
Attributes: 46
Sum of weights: 72725
Selected attribute
Name: TotalScore
Missing: 0 (0%)
Distinct: 44
Type: Nominal
Unique: 0 (0%)
Attributes
All
None
Invert
Pattern
Remove
Class: A843 (Nom)
Visualize All
Status
OK
Weka GUI Chooser
Program
Visualization
Tools
Help
Applications
Explorer
Experimenter
KnowledgeFlow
Workbench
WEKA
The University of Waikato
Waikato Environment for Knowledge Analysis
Version 3.8.4
(c) 1999 - 2019
File
Edit
View
Navigate
Code
Refactor
Run
Tools
VCS
Window
Help
Features_Selection_43_Tree_Test.py
Features_Selection_43_NB_Complete


Task Manager
Windows 10
File Options View
Processes Performance App history Startup Users Details Services
Name Status 15% CPU 73% Memory 4% Disk 0% Network
Google Chrome (30) 0.6% 1,004.1 MB 0.1 MB/s 0 Mbps
PyCharm (6) 0% 385.9 MB 0.1 MB/s 0 Mbps
Adobe Acrobat 2017 (32 bit) (3) 0.2% 151.4 MB 0 MB/s 0 Mbps
Microsoft PowerPoint (32 bit) 0% 139.8 MB 0 MB/s 0 Mbps
Windows Explorer (2) 3.1% 132.9 MB 0 MB/s 0 Mbps
Antimalware Service Executable 0% 109.5 MB 0 MB/s 0 Mbps
Desktop Window Manager 1.8% 41.8 MB 0 MB/s 0 Mbps
Service Host: Diagnostic Policy Service 1.3% 40.3 MB 0 MB/s 0 Mbps
Task Manager 1.2% 33.3 MB 0 MB/s 0 Mbps
ShareFile 0% 27.8 MB 0 MB/s 0.1 Mbps
Microsoft Windows Search Indexer 0% 26.5 MB 0 MB/s 0 Mbps
Service Host: DCOM Server Process Launcher (4) 1.6% 23.6 MB 0 MB/s 0 Mbps
SyncUpdateService 0% 17.8 MB 0 MB/s 0 Mbps
Host Process for Microsoft Configuration Manager 0% 15.6 MB 0 MB/s 0 Mbps
Service Host: Windows Management Instrumentation 0% 13.1 MB 0 MB/s 0 Mbps
WMI Provider Host 0% 10.9 MB 0 MB/s 0 Mbps
Service Host: Remote Procedure Call (2) 0% 10.4 MB 0 MB/s 0 Mbps
Application Frame Host 0% 8.0 MB 0 MB/s 0 Mbps
Service Host: Windows Event Log 0% 7.6 MB 0.1 MB/s 0 Mbps
Citrix Receiver Application (32 bit) 0% 6.7 MB 0 MB/s 0 Mbps
Service Host: State Repository Service 0% 6.5 MB 0 MB/s 0 Mbps
Fewer details
End task

Ubuntu 18.04.4 LTS



The Operating System as an Extended Machine



Operating systems turn ugly hardware into beautiful abstractions

The Operating System as a Resource Manager

- **Top down view**
  - Provide abstractions to application programs.
- **Bottom up view**
  - Manage pieces of complex system.
- **Alternative view**
  - Provide orderly, controlled allocation of resources.

The Operating System as a Resource Manager

Resource management includes sharing resources in two different ways: in **time** and in **space.**

E.g. Multiprogramming (each program gets a fair amount of time and memory space for execution) and sharing printer.

History of Operating Systems

Multics , Unix, and Linux

Multics (Multiplexed Information and Computing Service)

It was an influential but ultimately unsuccessful early time-sharing operating system developed in the 1960s and 1970s.

Introduction: Developed in the 1960s by MIT, General Electric, and Bell Labs.

Features: Multics was one of the first operating systems to implement many features we take for granted today, such as hierarchical file systems, dynamic linking, and a security model with access controls.

Legacy: Though not widely adopted, Multics significantly influenced the development of future operating systems, particularly Unix. Its design principles and innovations laid the groundwork for modern operating system concepts.

History of Operating Systems
Multics , Unix, and Linux

Unix is one of the most influential operating systems in the history of computing.

Introduction: Developed in the late 1960s and early 1970s at AT&T's Bell Labs by Ken Thompson, Dennis Ritchie, and others. It began as a small, experimental operating system developed on a PDP-7 minicomputer.

Features: Unix introduced the concept of a modular design with a small core set of utilities and a powerful command-line interface. It used a hierarchical file system, and its design emphasized simplicity and reusability of code.

Legacy: Unix became widely used in academia and industry and greatly influenced subsequent operating systems. Its design principles also shaped many modern operating systems.

History of Operating Systems
Multics , Unix, and Linux

Introduction: Created by Linus Torvalds in 1991 as a free and open-source alternative to Unix.

Features: Linux is known for its robustness, flexibility, and modularity. It uses the Linux kernel and supports a wide range of hardware architectures. It has become popular for its use in servers, desktops, and embedded systems.

Legacy: Linux has become a major force in computing, powering everything from smartphones (Android) to web servers and supercomputers. Its open-source nature has fostered a vast ecosystem of distributions and community-driven development.

Wide Adoption: Linux is now used in a wide range of systems, from embedded devices and smartphones (Android is built on a Linux kernel) to servers and cloud computing platforms.

History of Operating Systems

MS-DOS and Windows

DOS(Disk Operating System)

- Origin (1981): MS-DOS was originally developed by Microsoft for IBM's first personal computer, the IBM PC, released in 1981. It was a command-line-based operating system that provided basic file management and hardware control.

- MS-DOS Dominance: MS-DOS quickly became the dominant operating system for IBM-compatible PCs. It was sold separately from the hardware, and its success laid the foundation for Microsoft's future dominance in the software industry.

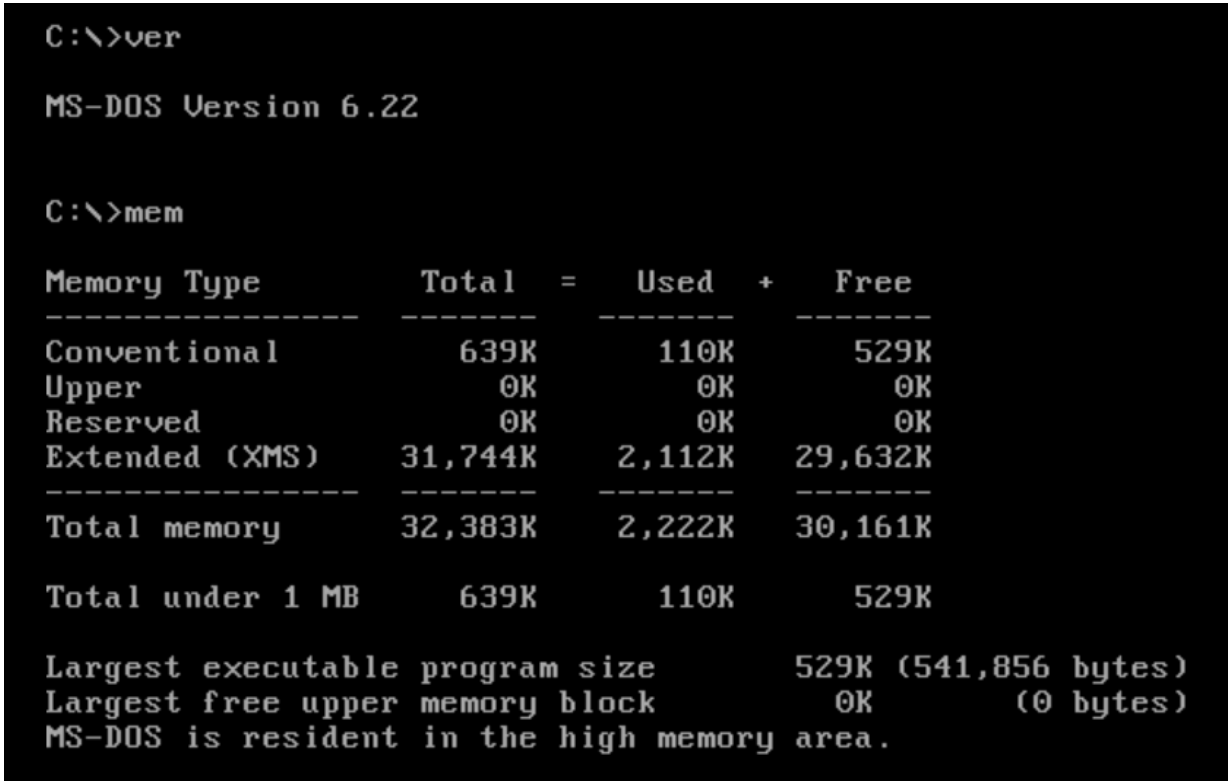
- Versions and Competition: Over the years, MS-DOS went through several versions, with MS-DOS 6.22 being one of the most well-known. Concurrently, other DOS-based operating systems like PC-DOS (IBM's version) and DR-DOS (Digital Research's version) were in the market, but MS-DOS maintained its leadership.

History of Operating Systems

MS-DOS and Windows

DOS(Disk Operating System)

- Graphical User Interfaces: While DOS was text-based, it lacked a graphical user interface (GUI). This led to the development of Windows as an overlay for MS-DOS.



History of Operating Systems
MS-DOS and Windows

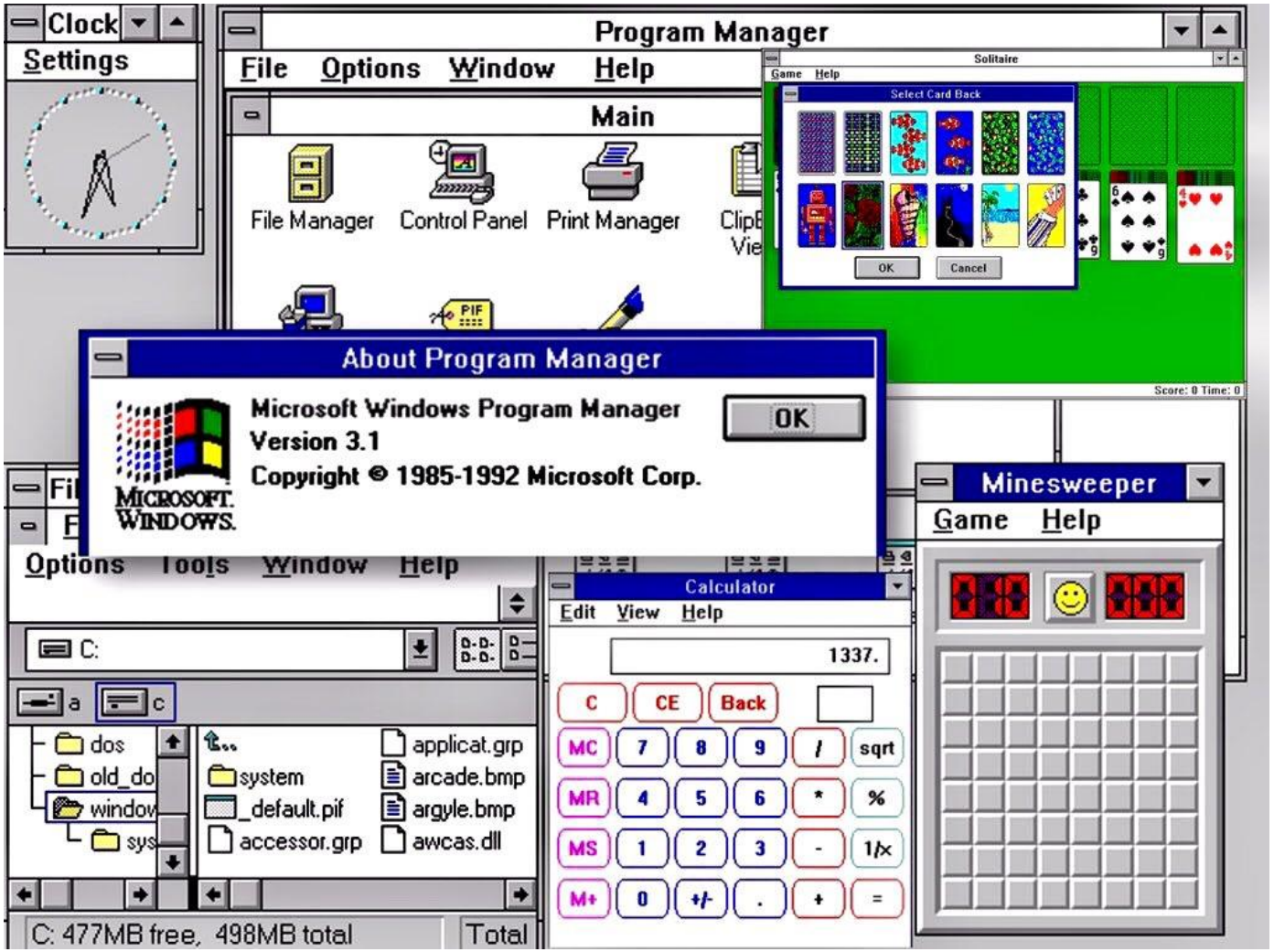
Windows

- Windows 1.0 (1985): Microsoft introduced Windows 1.0 in 1985. It was not an operating system but rather a graphical shell that ran on top of MS-DOS. It provided a limited GUI environment with overlapping windows.

- Windows 2.0 and 3.0 (1987-1990): Windows 2.0 added improved functionality and better graphics. Windows 3.0, released in 1990, marked a significant leap in popularity due to its improved performance and the ability to run multiple applications in separate windows.

- Windows 95 (1995): Windows 95 was a major milestone, featuring a completely redesigned user interface and better integration with the underlying MS-DOS. It introduced the Start Menu and 32-bit multitasking, making it more stable than previous versions.

History of Operating Systems
MS-DOS and Windows



History of Operating Systems
MS-DOS and Windows

Windows

- Windows 98 (1998) and Windows Me (2000): These versions provided incremental improvements but were not as successful as Windows 95.

- Windows NT (1993) and Windows 2000 (2000): These versions were part of the Windows NT family and aimed at the business market. They were known for their stability and security. Windows 2000 was the first to merge the consumer and business lines into a single codebase.

- Windows XP (2001): Windows XP was a major success, offering a stable platform for both consumers and businesses. It featured an updated GUI, enhanced multimedia capabilities, and improved networking.

History of Operating Systems
MS-DOS and Windows

- Windows Vista (2007) and Windows 7 (2009): Vista faced criticism for its performance and compatibility issues. Windows 7, released two years later, addressed many of these concerns and was well-received.

- Windows 8 (2012) and Windows 8.1 (2013): These versions introduced a touch-centric interface and were met with mixed reviews. Windows 8.1 brought back some traditional desktop elements.

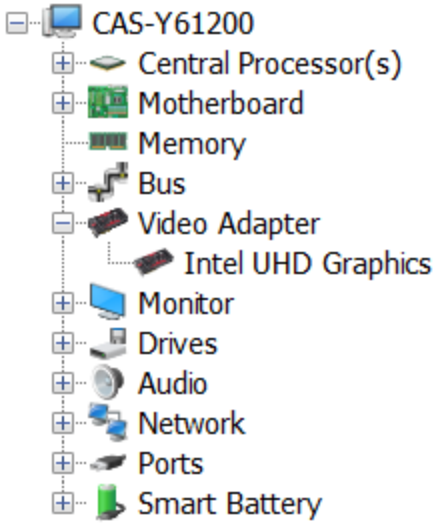
- Windows 10 (2015): released as a free upgrade for Windows 7 and 8 users, aimed to provide a unified platform for PCs, tablets, and smartphones. It has seen regular feature updates.

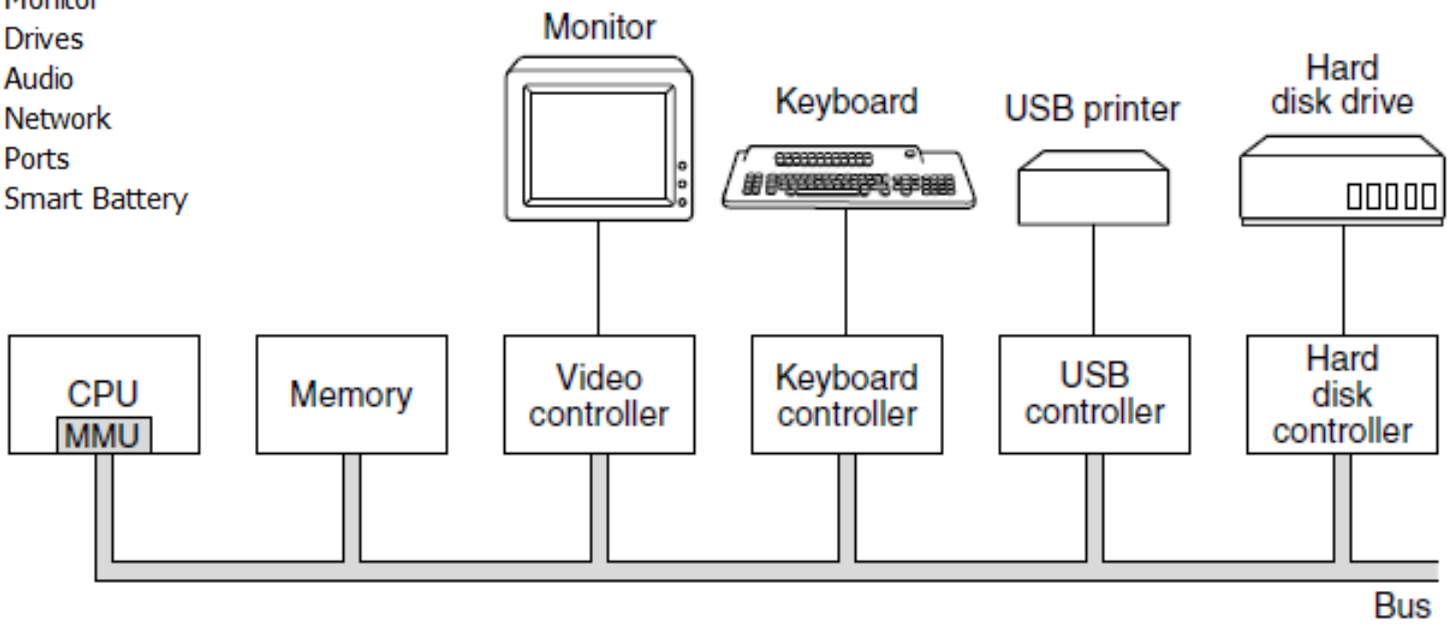
- Windows 11 (2021): Windows 11 is the latest version, featuring a redesigned Start Menu, improved performance, and enhanced gaming capabilities.

MS-DOS and Windows Operating System

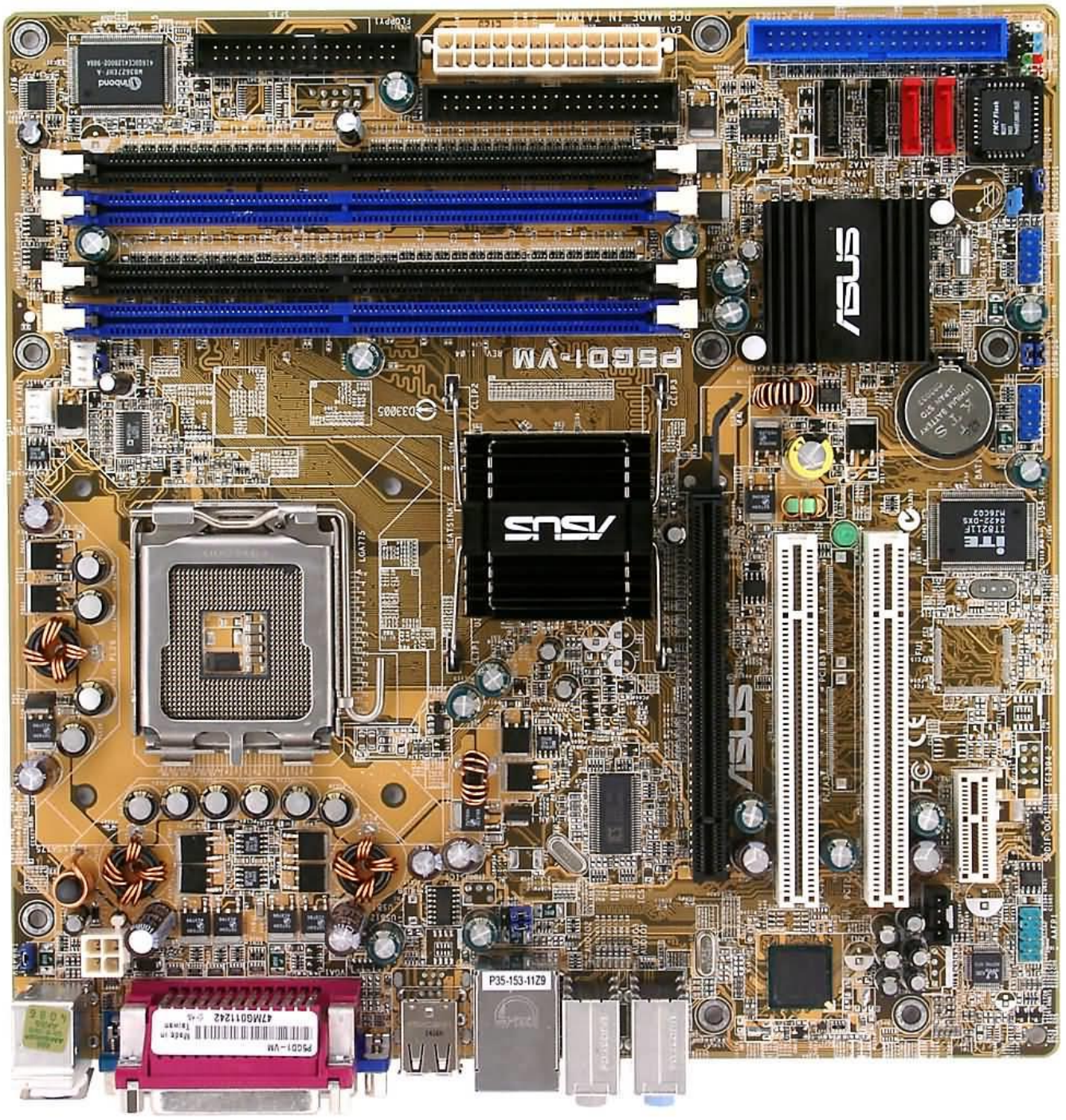
MS-DOS
Windows 1.0 - 2.0
Windows 3.0 – 3.1
Windows 95
Windows 98
Windows ME - Millennium Edition
Windows NT 31. - 4.0
Windows 2000
Windows XP
Windows Vista
Windows 7
Windows 8
Windows 10
Windows 11
Windows Server
Windows Home Server
Windows CE (Embedded Compact)
Windows Mobile
Windows Phone 7-10

Computer hardware review

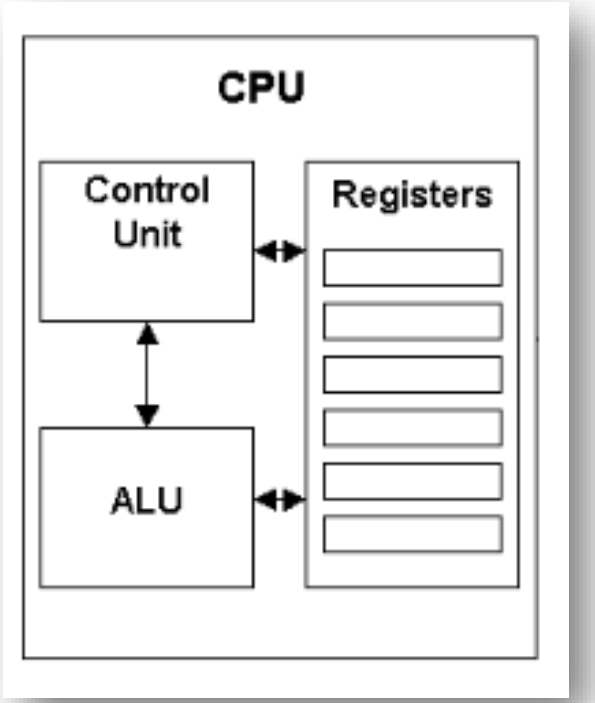




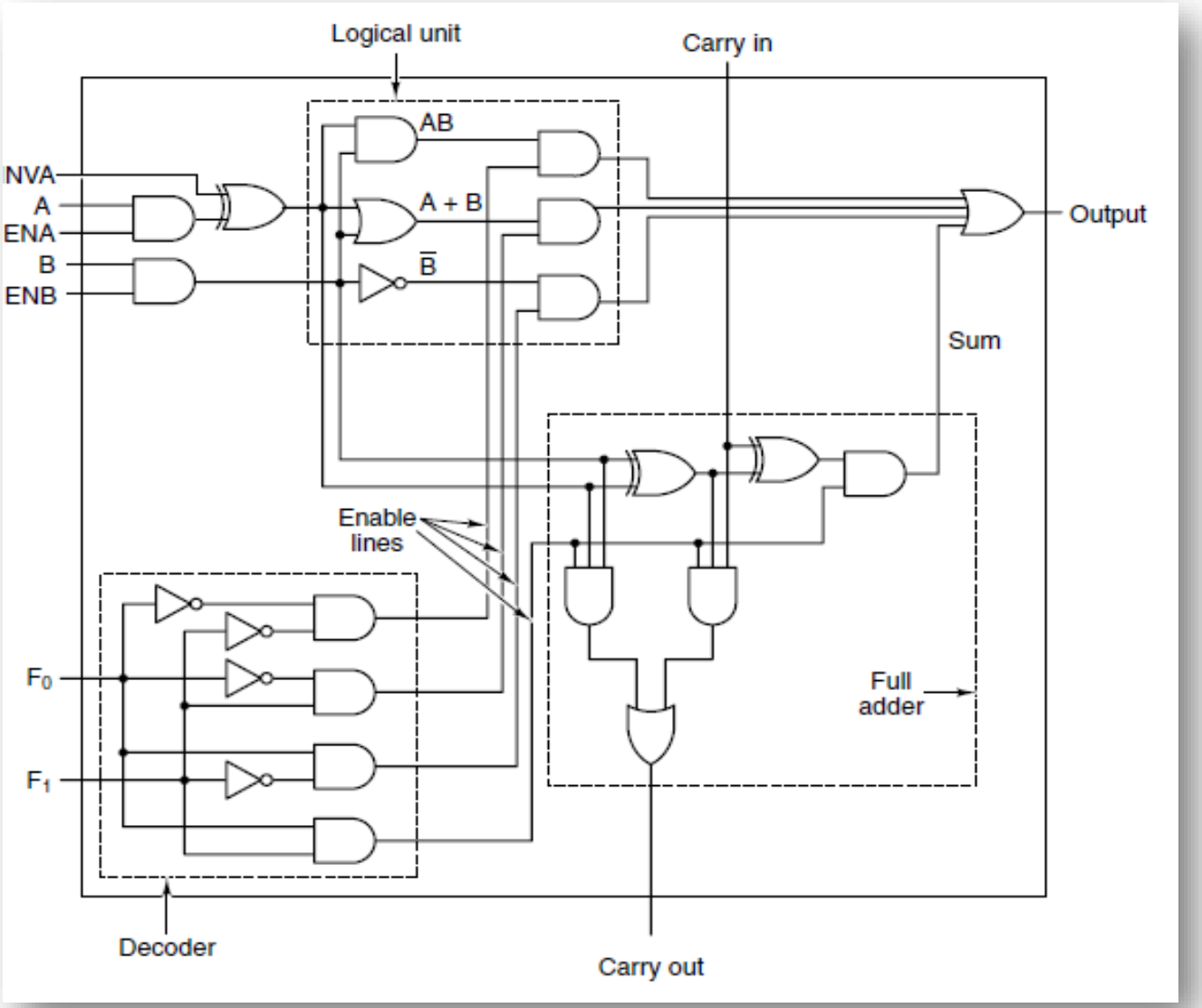
Some of the components of a simple personal computer


P5GD1-VM
ASUS
P35-153-11Z9

CPU



ALU


Logical unit
Carry in
AB
A + B
$\overline{B}$
NVA
A
ENA
B
ENB
Output
Sum
Enable lines
F_0
F_1
Full adder
Decoder
Carry out

ALU

Logic gates

ALU (list of operations)

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	–1

ALU

Example:

Use the ALU operation table to obtain B – A where A = 3 and B = 4 (use 8-bit representation for each number).

	A	B	INVA	F0	F1	ENA	ENB	INC	Res (Out)
1	00000011	00000100							

Example:

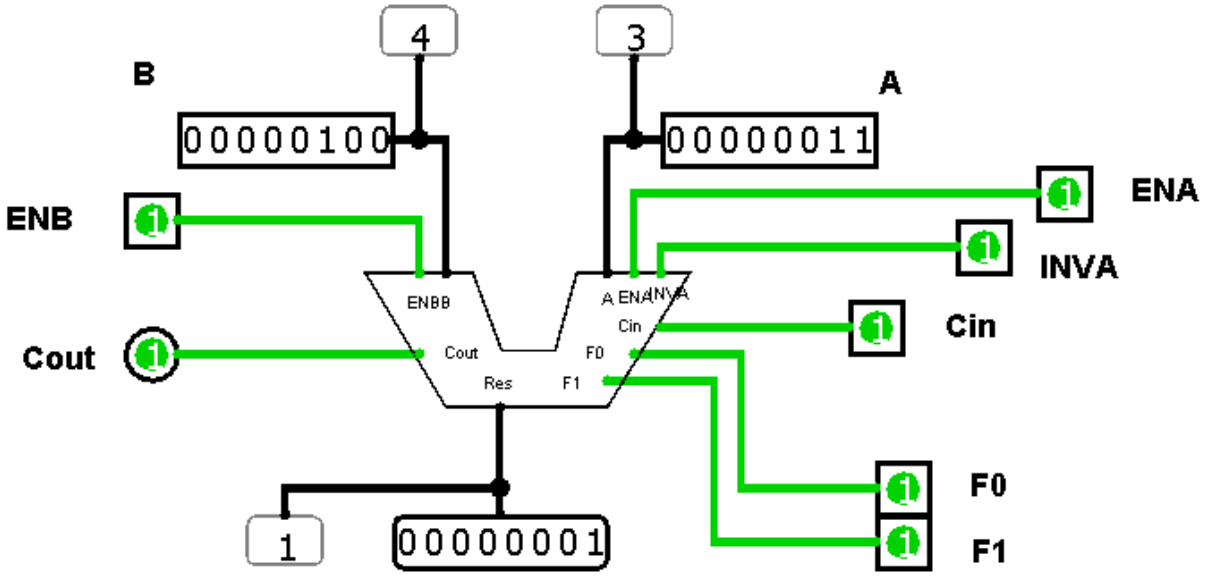
ALU

Use the ALU operation table to obtain B – A where A = 3 and B = 4 (use 8-bit representation for each number).

	A	B	INVA	F0	F1	ENA	ENB	INC	Res (Out)
1	00000011	00000100	1	1	1	1	1	1	100000001

B – A = B + NOT(A) + 1 = 00000100 +NOT(00000011) + 1

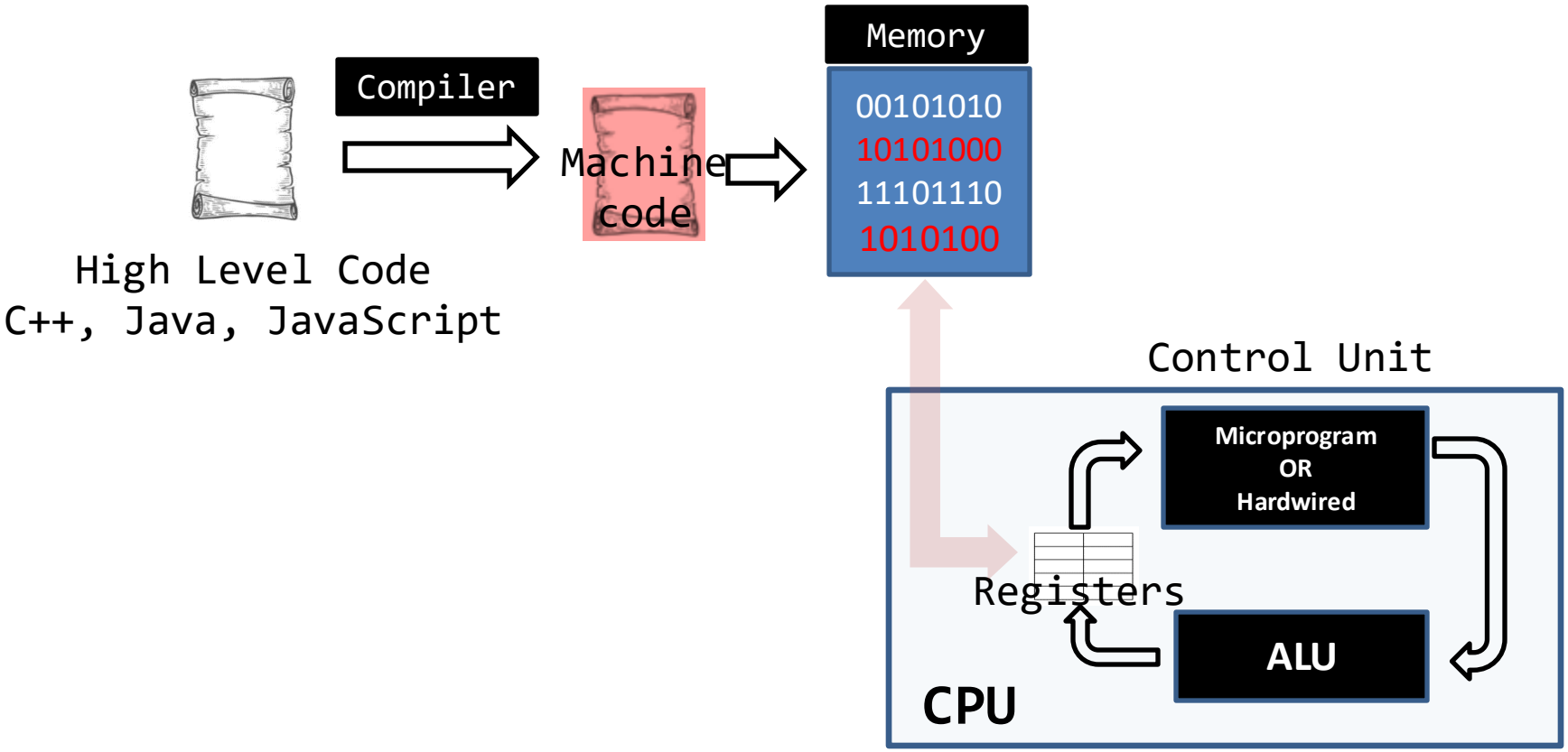
= 00000100 + 11111100 + 1 = 100000000+1 = 100000001



Control unit

Instruction execution

Fetch-Decode-Execute



RISC vs CISC

RISC	CISC
AVR	AMD
ARM	Pentium
MIPS	VAX

Hardwired Approach

Microprogrammed Approach

Processors

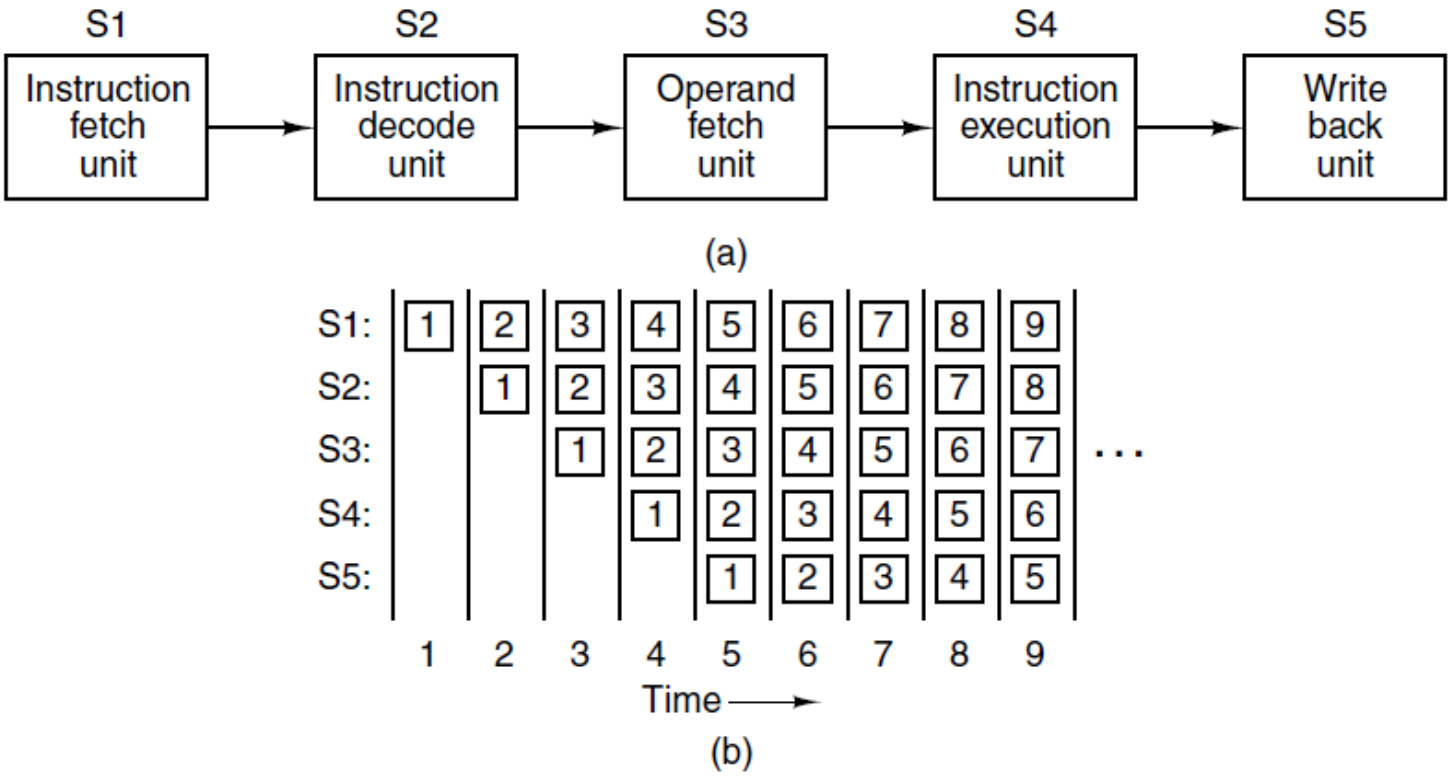
Performance improvement

- Instruction-Level Parallelism (Pipelining)
- Multithreading and Multicore Chips
- Multiprocessors and Multi-computers

Processors

Performance improvement

Instruction-Level Parallelism
Pipelining



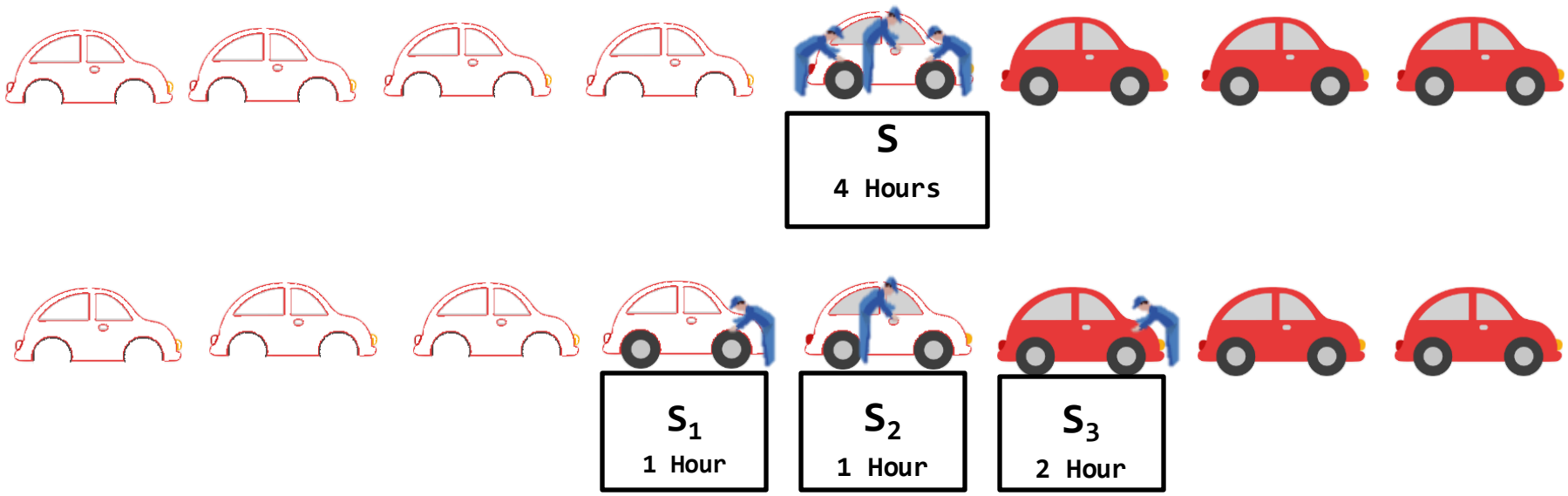
(a) A five-stage pipeline. (b) The state of each stage as a function of time. Nine clock cycles are illustrated.

Pipelining

Example:

Car factory (the process of making a car)

Which assembly line is faster ?

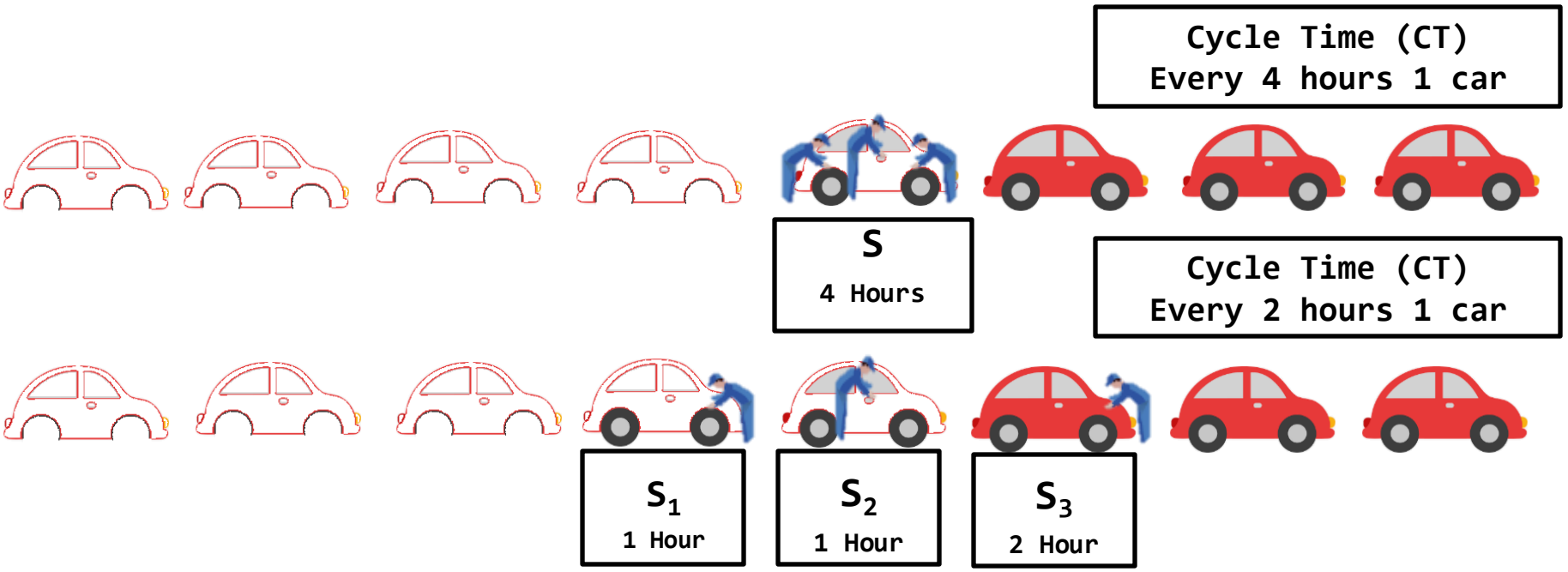


Pipelining

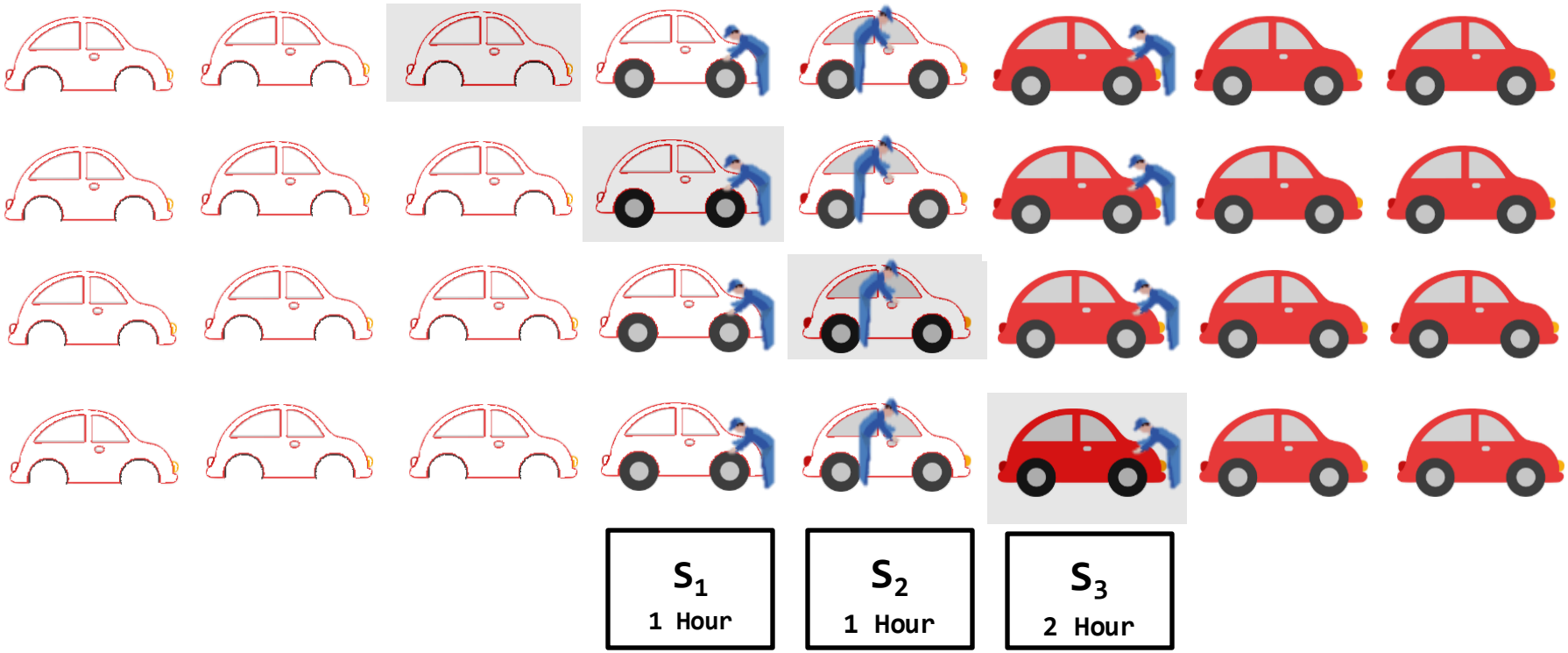
Example:

Car factory (the process of making a car)

Which assembly line is faster ?



Example: Latency for each car

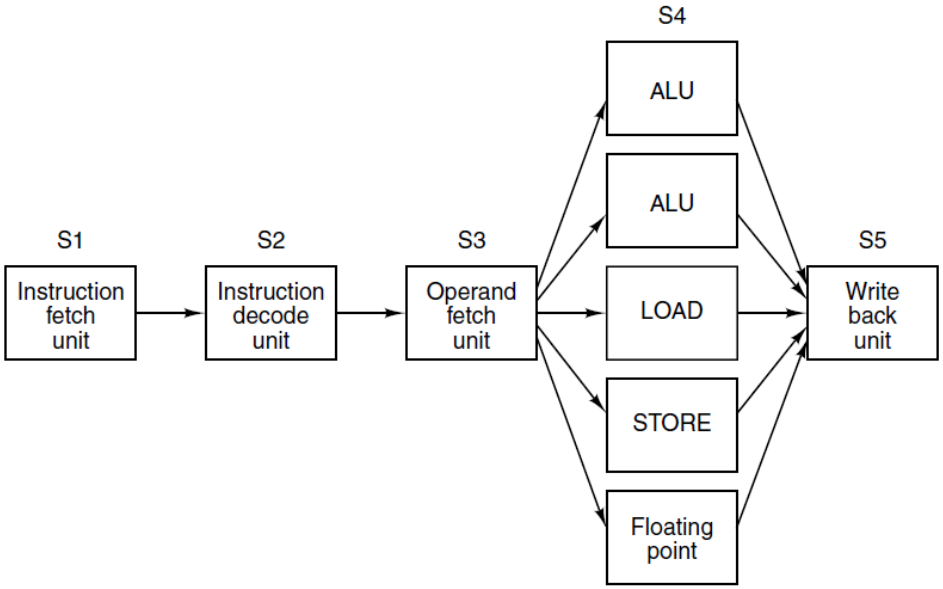


Latency for each car = 2 + 2 + 2 = 6 hours

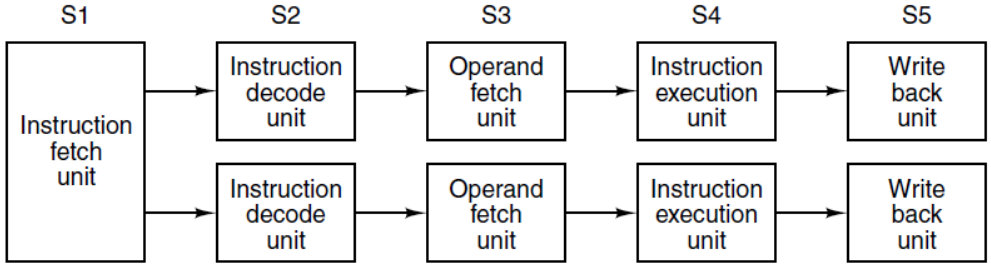
Processors

Performance improvement

Instruction-Level Parallelism



A superscalar processor with five functional units



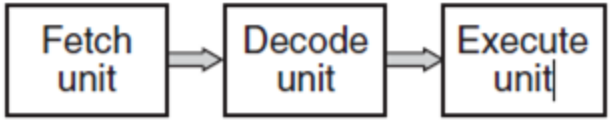
Dual five-stage pipelines with a common instruction fetch unit

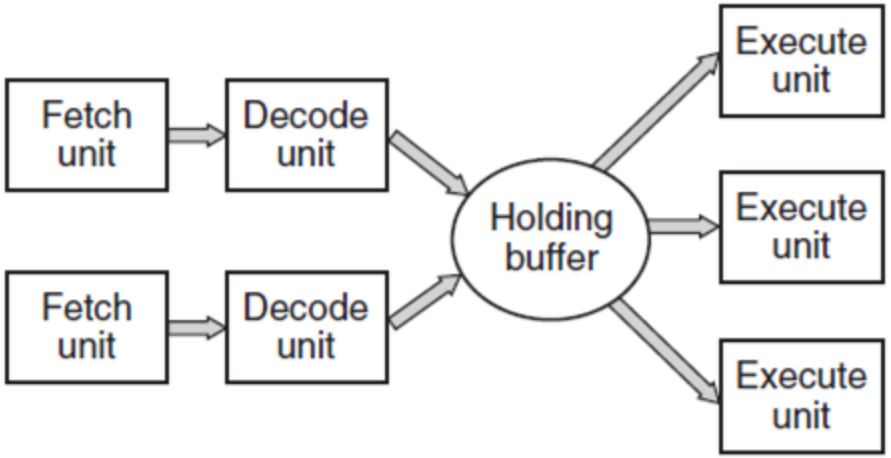
Processors

Performance improvement

Instruction-Level Parallelism

A three-stage pipeline





Integer arithmetic

Floating-point arithmetic

Boolean operations

Superscalar CPU

Processors

Performance improvement

Multithreading and Multicore Chips

- Allows the CPU to hold the state of two different threads and then switch back and forth on a nanosecond time scale e.g. SPARC, the Power5, the Intel Xeon, and the Intel Core family.

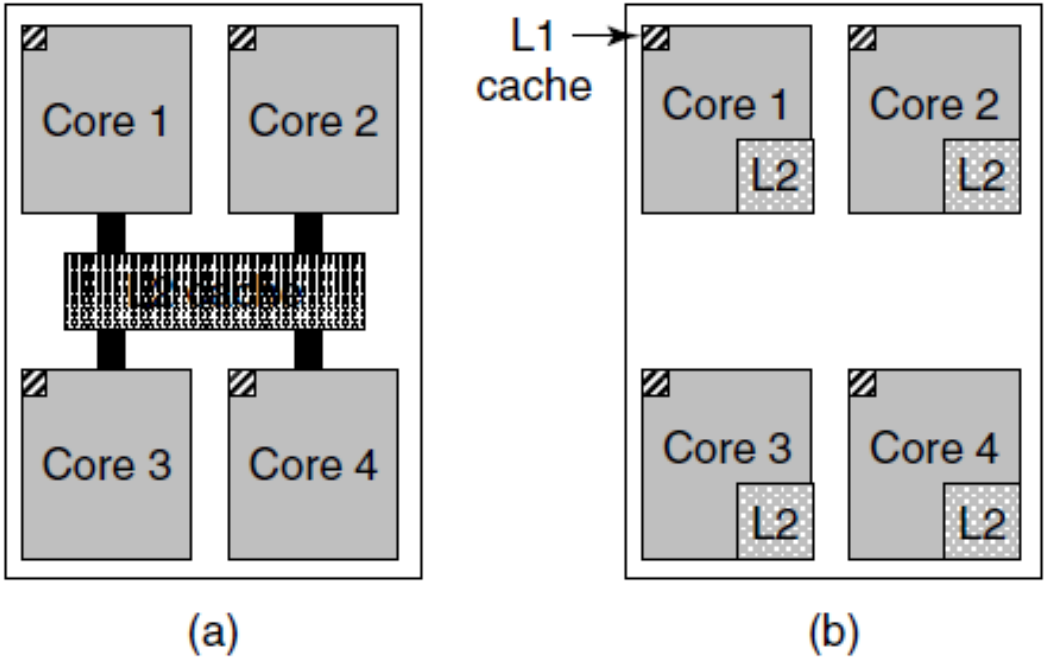
- **Multicore chips** , CPU chips now have four, eight, or more complete processors or cores on them e.g. Intel Xeon Phi and the Tilera TilePro, more than 60 cores on a single chip.

Processors

Performance improvement

Multithreading and Multicore Chips

Intel multicore chips **AMD**



(a) A quad-core chip with a shared L2 cache.
(b) A quad-core chip with separate L2 caches.

Processors

Performance improvement

Multiprocessors and Multi-computers

Processor-Level Parallelism

Multiprocessors

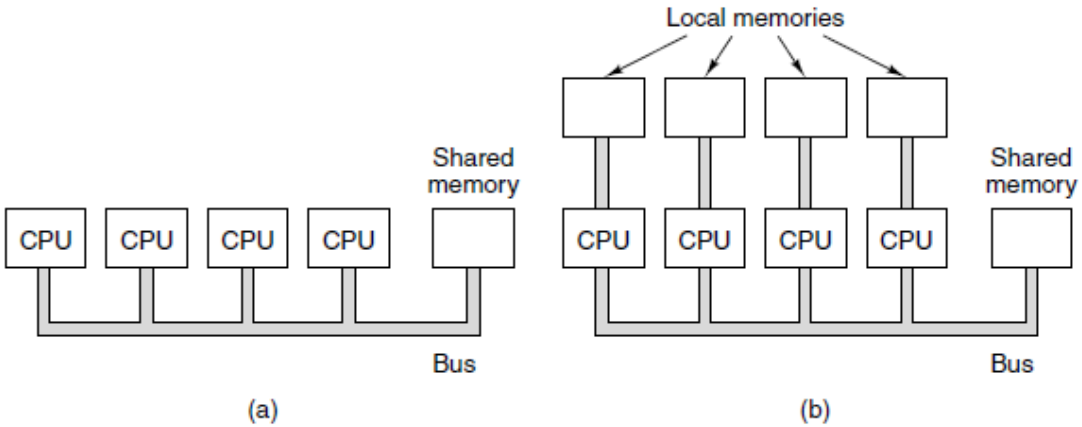
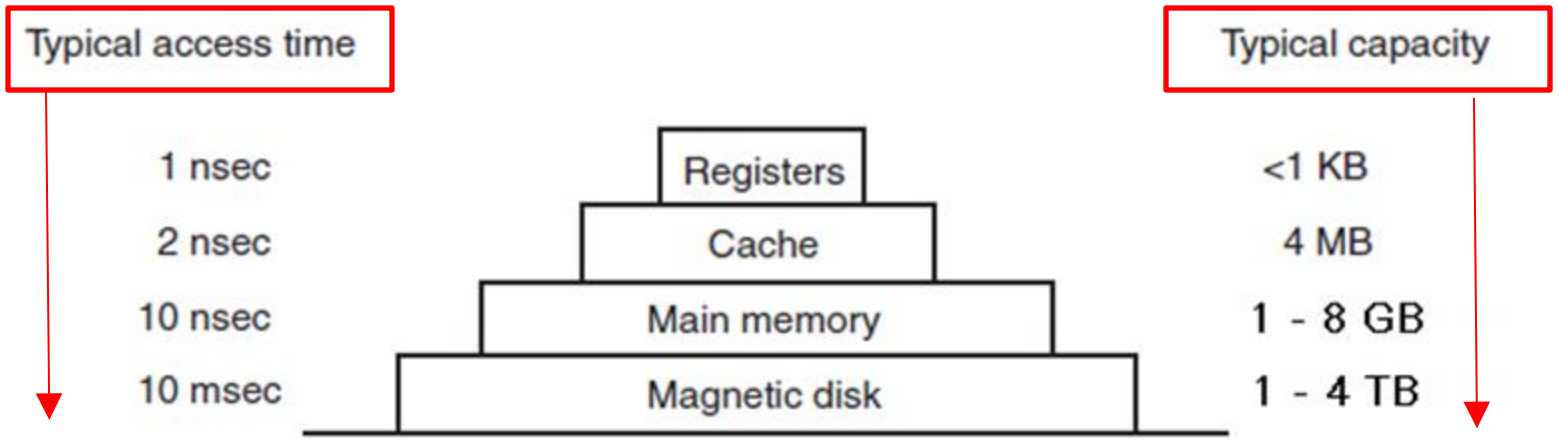


Figure 2-8. (a) A single-bus multiprocessor. (b) A multicomputer with local memories.

Processor-Level Parallelism

Multi-computers

Memory



A typical memory hierarchy. The numbers are very rough approximations.

Registers

- Local fast memories
- All CPUs contain some registers inside to hold key variables and temporary results e.g.

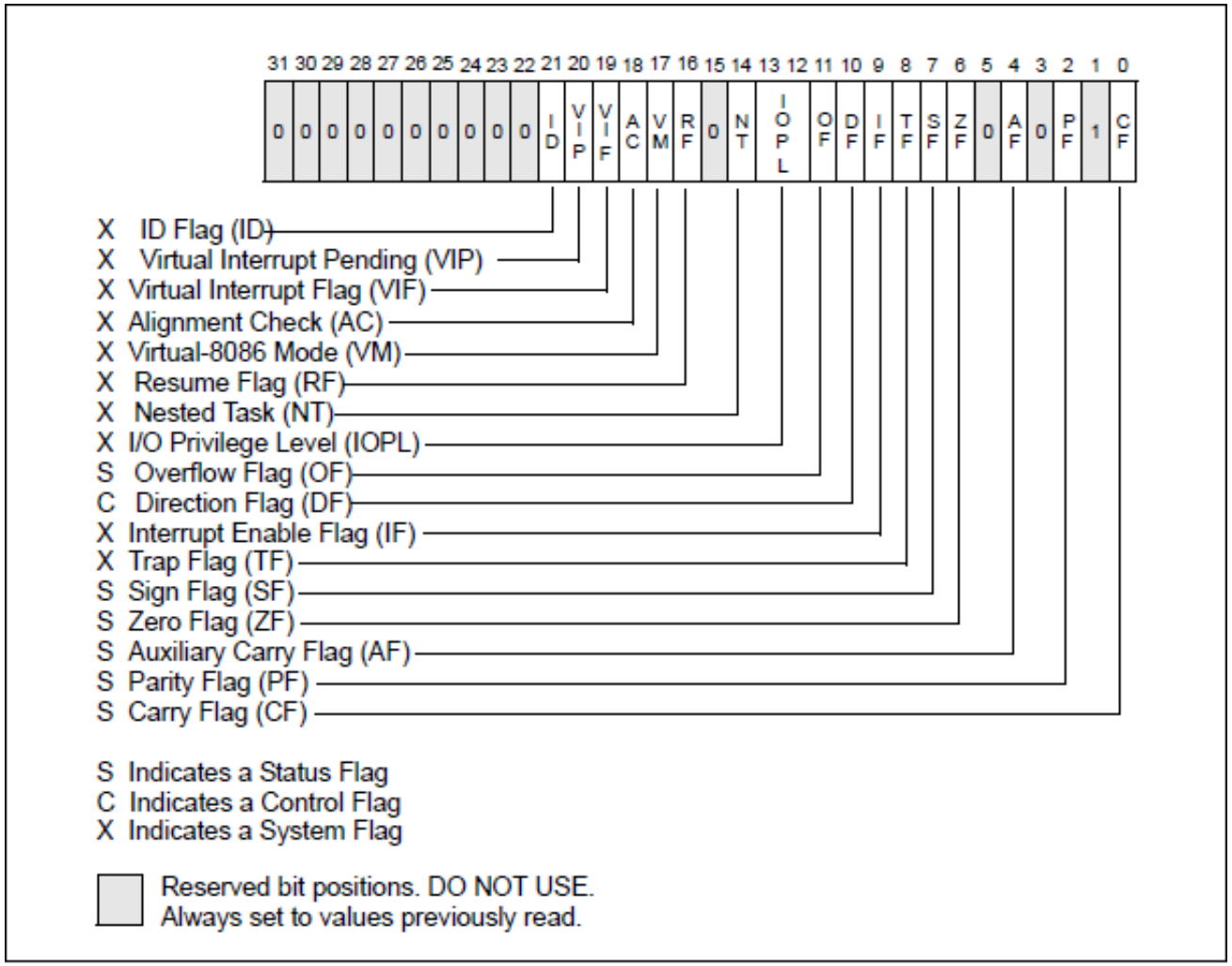
  PC (program counter) which contains the memory address of the next instruction to be fetched.

  Stack pointer, which points to the top of the current stack in memory.

  PSW (Program Status Word) which contains the condition code bits, which are set by comparison instructions, the CPU priority, the mode (user or kernel), and various other control bits. User programs may normally read the entire PSW but typically may write only some of its fields.

  The PSW plays an important role in system calls and I/O.

EFLAGS Register



EFLAGS Register

EFLAGS Register

CF (bit 0)	**Carry flag** — Set if an arithmetic operation generates a carry or a borrow out of the most-significant bit of the result; cleared otherwise. This flag indicates an overflow condition for unsigned-integer arithmetic. It is also used in multiple-precision arithmetic.
PF (bit 2)	**Parity flag** — Set if the least-significant byte of the result contains an even number of 1 bits; cleared otherwise.
AF (bit 4)	**Auxiliary Carry flag** — Set if an arithmetic operation generates a carry or a borrow out of bit 3 of the result; cleared otherwise. This flag is used in binary-coded decimal (BCD) arithmetic.
ZF (bit 6)	**Zero flag** — Set if the result is zero; cleared otherwise.
SF (bit 7)	**Sign flag** — Set equal to the most-significant bit of the result, which is the sign bit of a signed integer. (0 indicates a positive value and 1 indicates a negative value.)
OF (bit 11)	**Overflow flag** — Set if the integer result is too large a positive number or too small a negative number (excluding the sign-bit) to fit in the destination operand; cleared otherwise. This flag indicates an overflow condition for signed-integer (two's complement) arithmetic.

IOPL (bits 12 and 13)

I/O privilege level field — Indicates the I/O privilege level of the currently running program or task. The current privilege level (CPL) of the currently running program or task must be less than or equal to the I/O privilege level to access the I/O address space. The POPF and IRET instructions can modify this field only when operating at a CPL of 0.

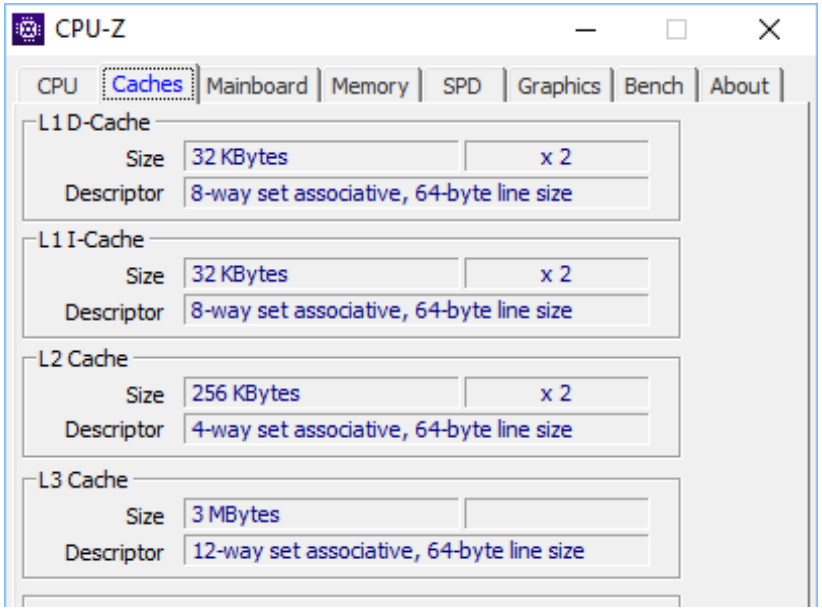
Cache Memory

Imbalance Between CPU and Memory

- Historically, CPU is faster than memories.
- The capacity of memories are increased, not speed.
- The slower the memory, the more cycles the CPU will have to wait.

Practical Solution

- Cache memory (**L1**, **L2**, and **L3**)
- Main memories and caches are divided up into fixed-size blocks (**cache lines**) e.g. typically 64 bytes.

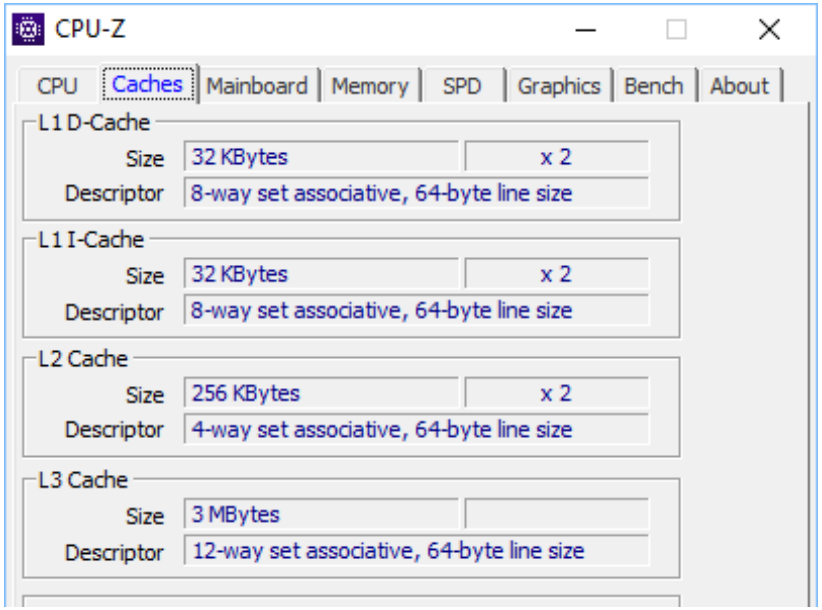


Cache Memory

Example :

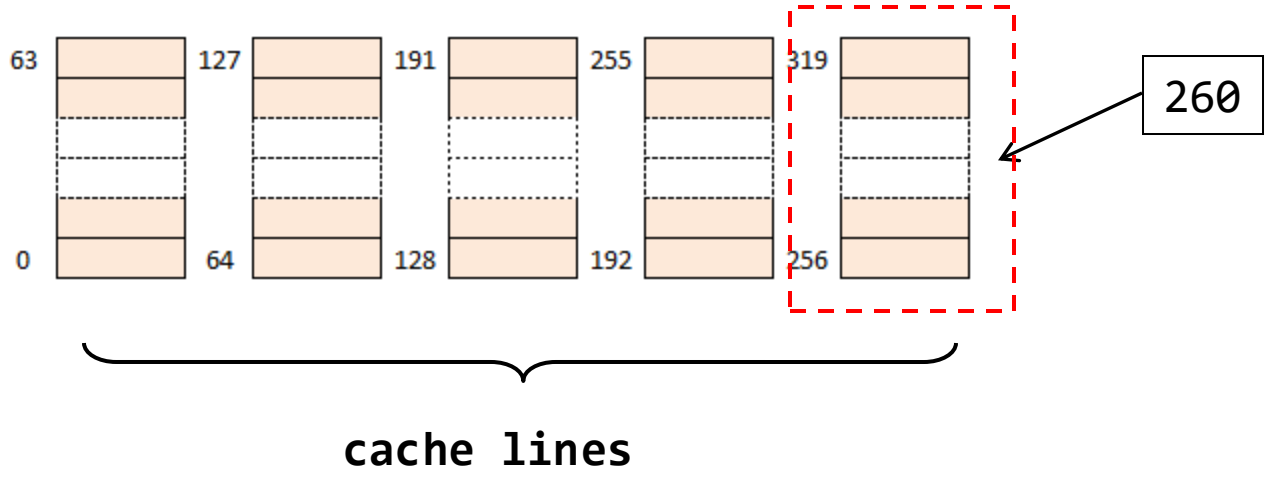
Obtain the number of cache lines and sets for **L1D** Cache?

Total L1D = 32KB x 2 = 64 KB
Number of cache lines = 64 KB/64B = 64x1024/64 = 1024
Number of sets = Number of cache lines/8 = 128



Cache Memory

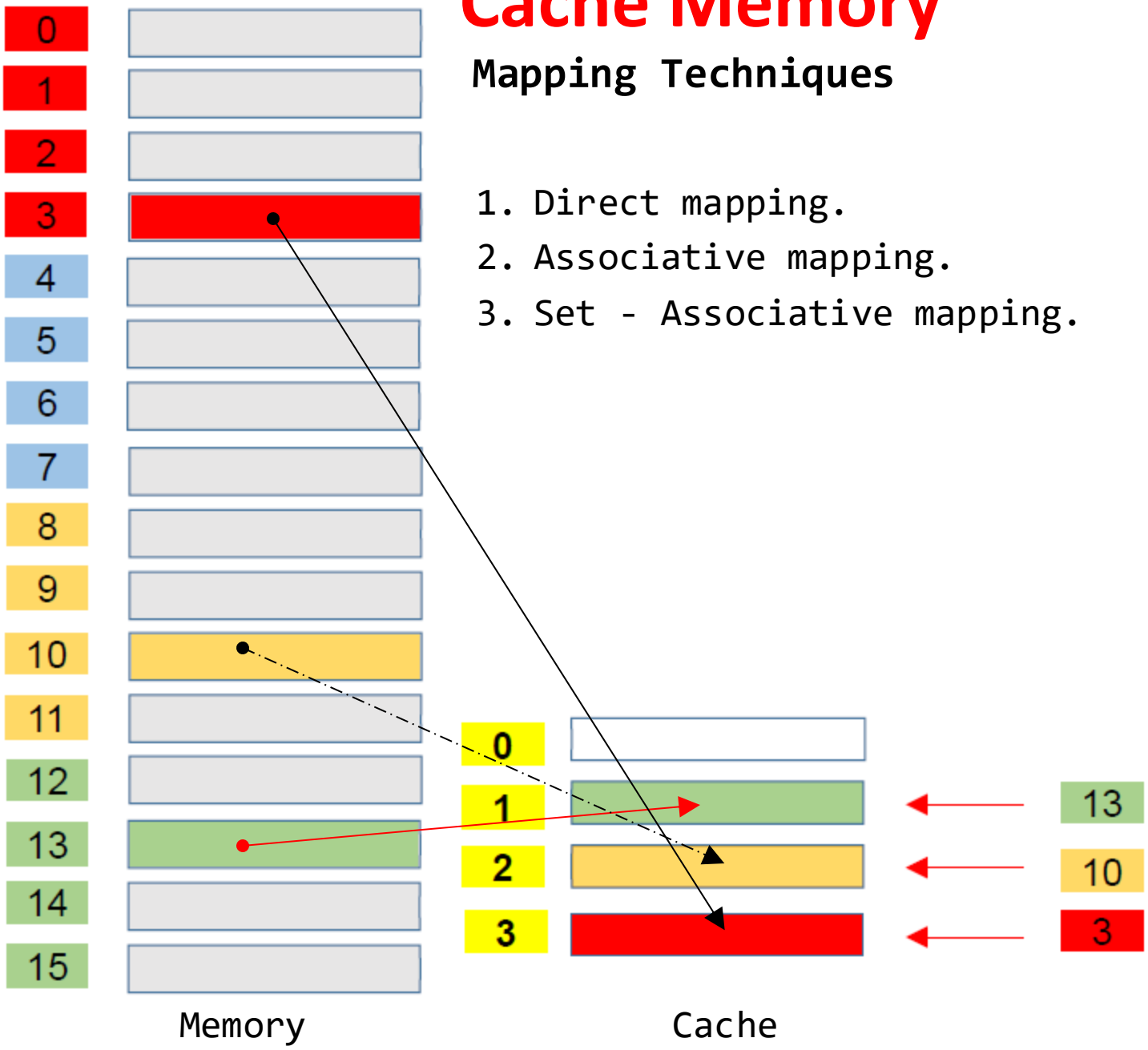
Example : a **64-byte** line size, a reference to memory address 260, what line is pulled into one cash line ?



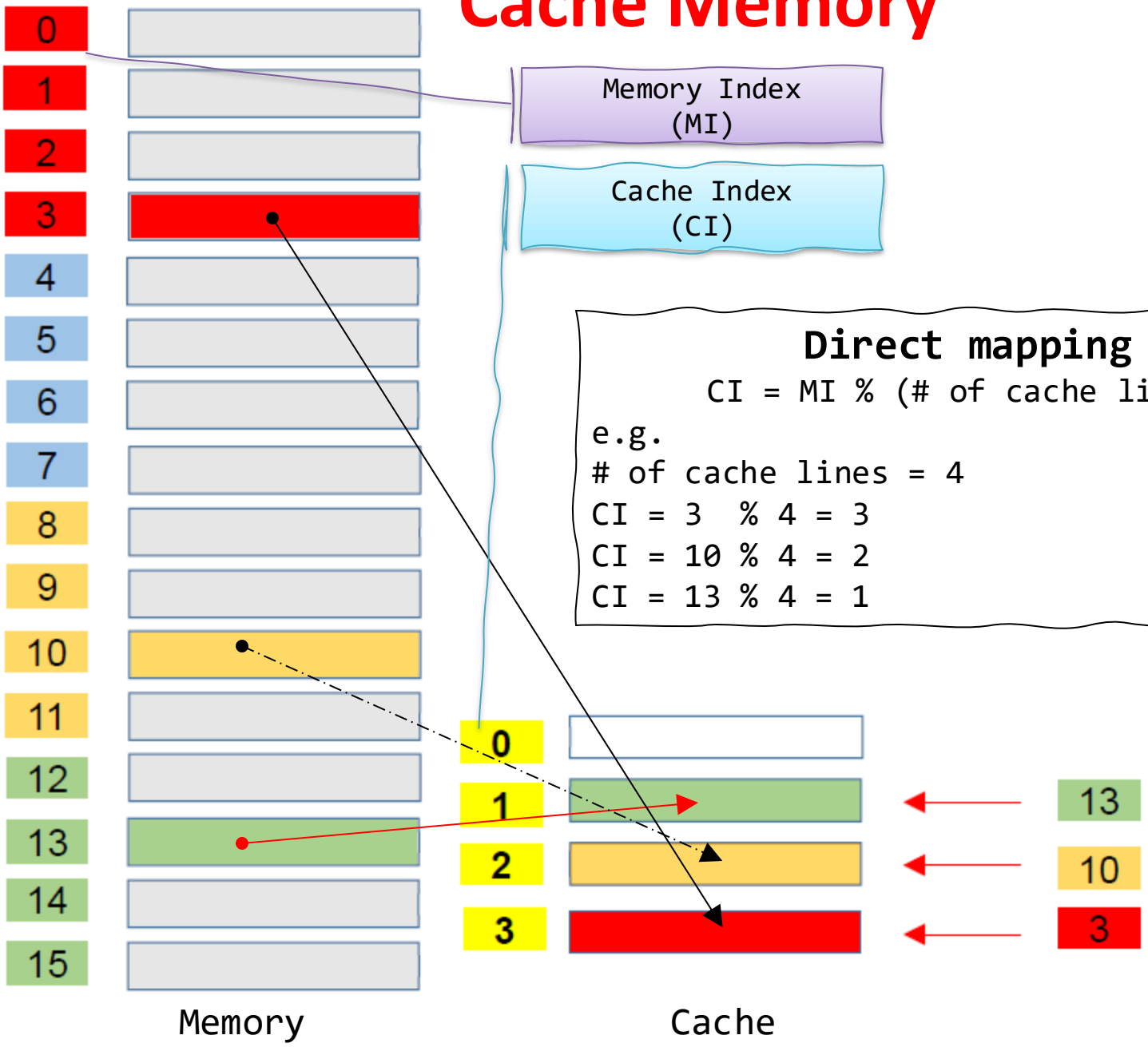
Cache Memory

Mapping Techniques

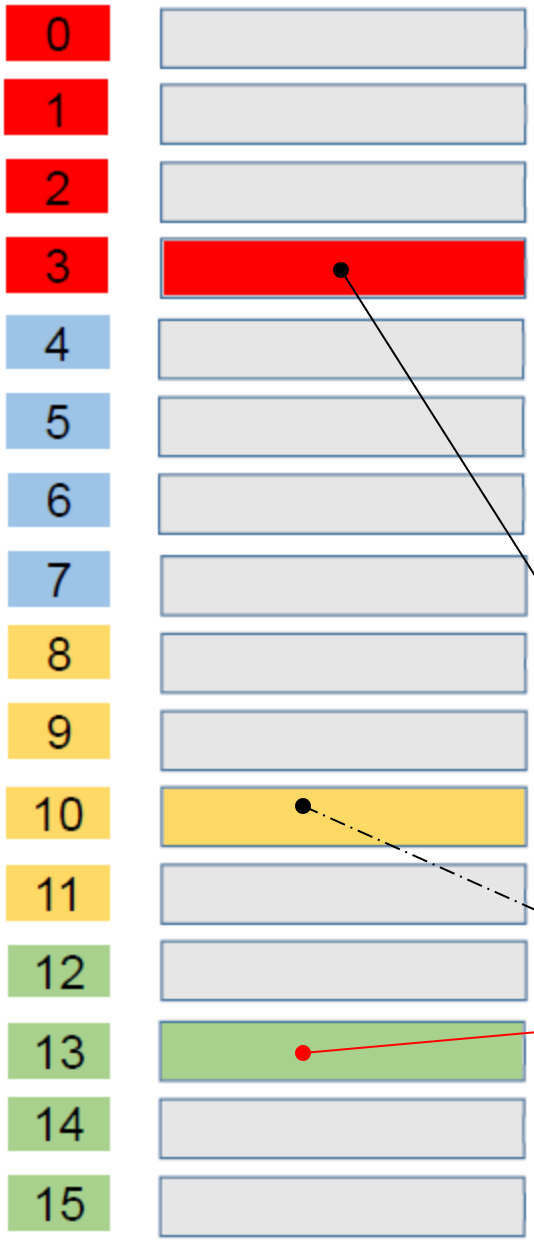
1. Direct mapping.
2. Associative mapping.
3. Set - Associative mapping.



Cache Memory



Cache Memory



Direct mapping

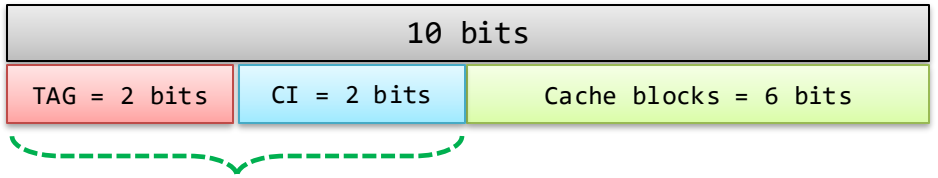
MI = 4 bits

CI = 2 bits

TAG = MI - CI

Cache block size = 64 bytes (6 bits)

Memory address = 4 + 6 = 10 bits



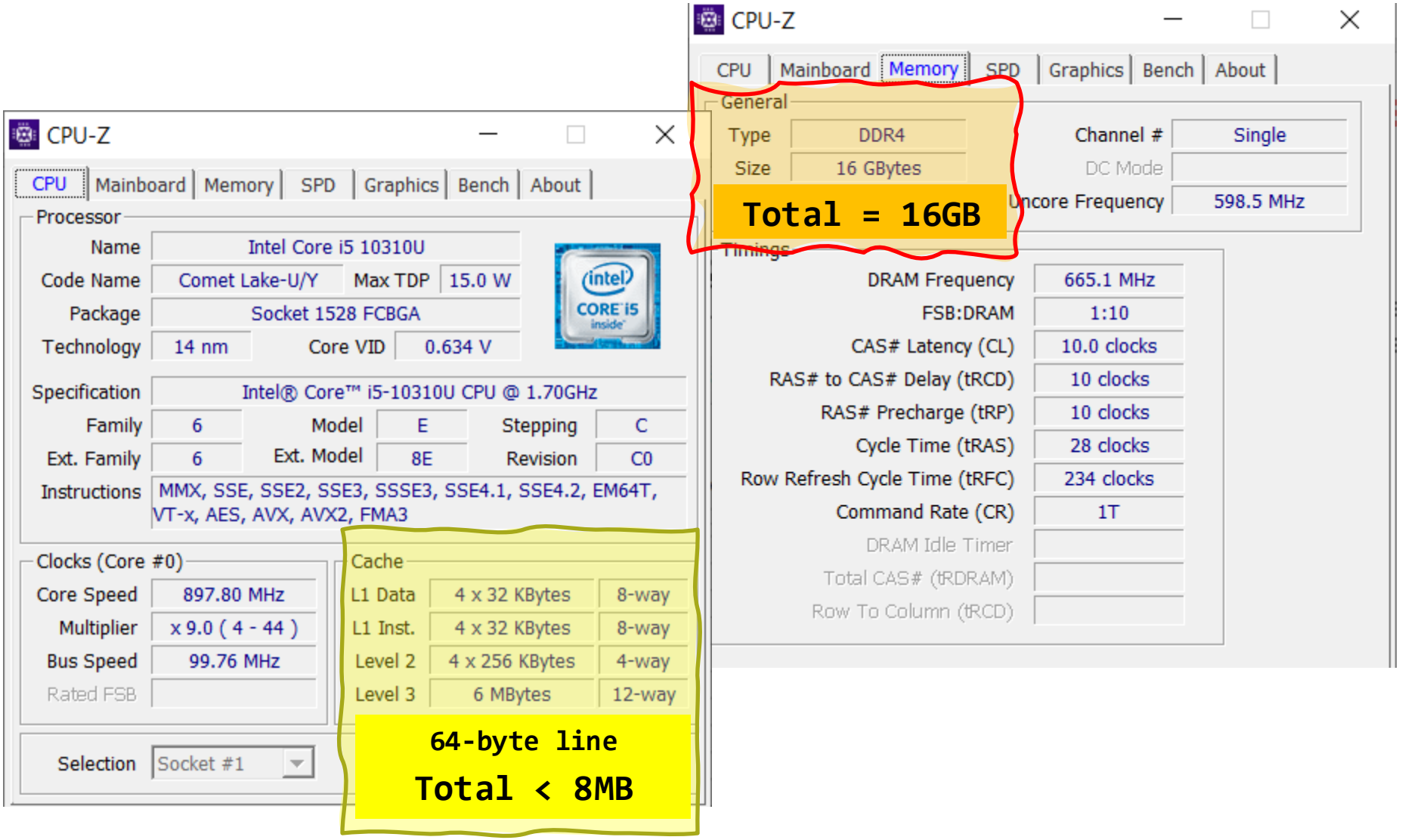
e.g.

1101xxxxxx => 1101xxxxxx MI = 13 CI = 13 % 4 = 1

1101xxxxxx => 1101xxxxxx CI = 01_2 = 1

Cache Memory

Cache size vs Memory size



Cache Memory

Cache size vs Memory size

Caching system issues

A. When to put a new item into the cache.

B. Which cache line to put the new item in.

C. Which item to remove from the cache when a slot is needed.

D. Where to put a newly evicted item in the larger memory.

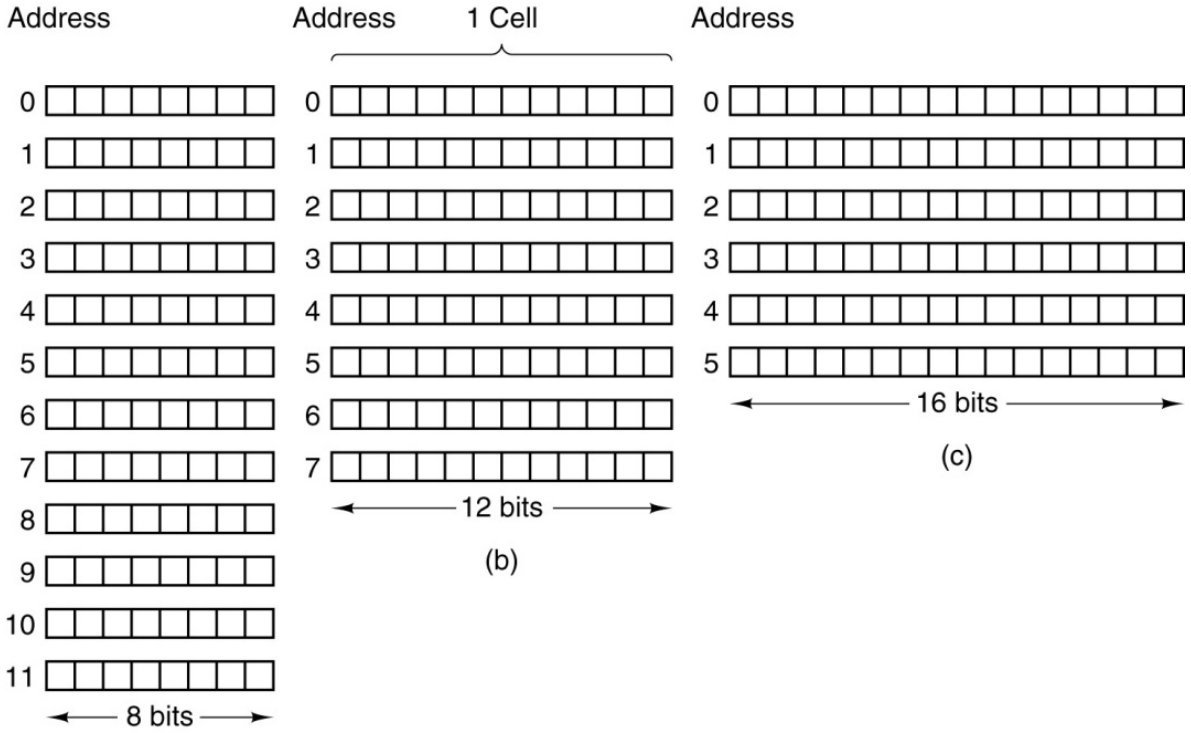
There are some algorithms which are used to replace the existing block in the cache memory with the new fetched data.

Cache Replacement Algorithms

1. Least Recently Used (LRU).
2. First In First Out (FIFO).
3. Least Frequently Used (LFU).
4. Random.
5. Clock algorithm.

RAM

1. Memory is part of computer and contains a number of cells.
2. Programs and data are stored in memory.
3. The basic unit of memory is **bit {0, 1}.**
4. Other Units 1 byte = 8 bits.
   word = 1, 2, .. Byte

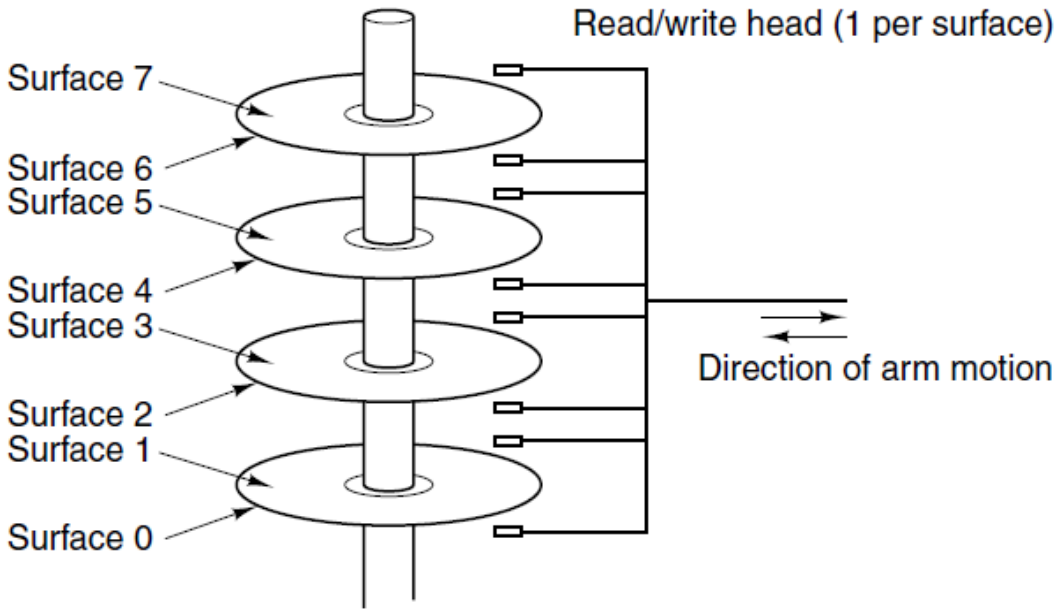


Computer	Bits/cell
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

(a) Three ways of organizing a 96-bit memory

Disks

- A disk consists of one or more metal platters that rotate at 5400, 7200, 10,800 RPM or more.
- A mechanical arm pivots over the platters from the corner.
- The outer cylinders contain more sectors than the inner ones.



Structure of a disk drive

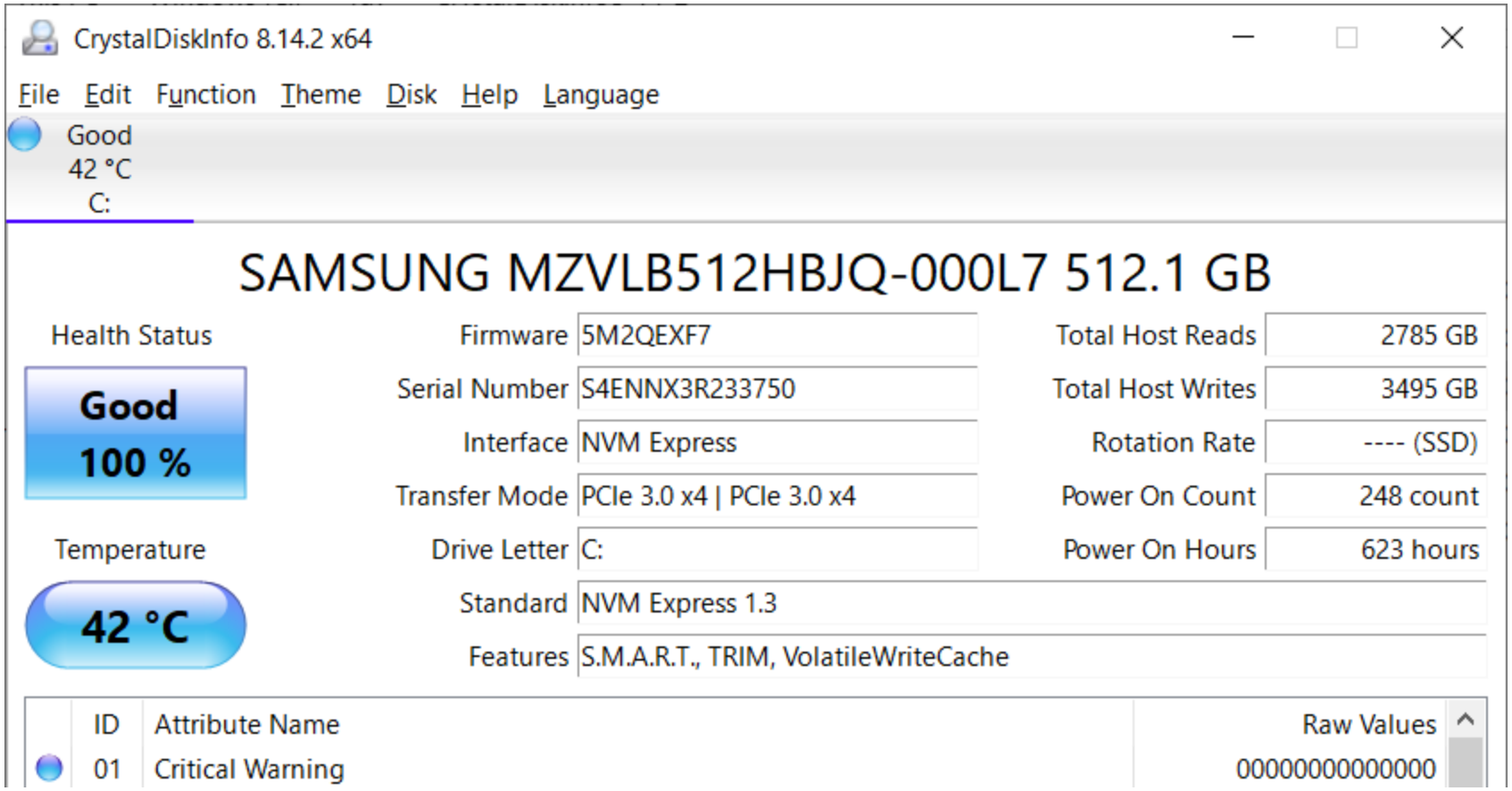
Disks

- Moving the arm from one cylinder to the next takes about 1 msec. Moving it to a random cylinder typically takes 5 to 10 msec, depending on the drive.
- Once the arm is on the correct track, the drive must wait for the needed sector to rotate under the head, an additional delay of 5 msec to 10 msec, depending on the drive's RPM.
- Once the sector is under the head, reading or writing occurs at a rate of 50 MB/sec on low-end disks to 160 MB/sec on faster ones.

Solid State Disks

- SSDs do not have moving parts, do not contain platters in the shape of disks, and store data in (Flash) memory.
- they also store a lot of data which is not lost when the power is off.

Solid State Disks



I/O Devices

I/O units often consist of a mechanical component and an electronic component (device controller or adapter).

The controller is a chip or a set of chips that physically controls the device.

I/O Devices

I/O devices also interact heavily with the operating system.

The software that talks to a controller, giving it commands and accepting responses, is called a **device driver.** the driver has to be put into the operating system so it can run in kernel mode.

Every controller has a small number of registers that are used to communicate with it.

For example, a minimal disk controller might have registers for specifying the disk address, memory address, sector count, and direction (read or write).

To activate the controller, the driver gets a command from the operating system, then translates it into the appropriate values to write into the device registers.

I/O Devices

Input and output can be done in three different ways:

1. **Busy waiting**

User program issues a system call, which the kernel then translates into a procedure call to the appropriate driver.

The driver then starts the I/O and sits in a tight loop continuously polling the device to see if it is done (usually there is some bit that indicates that the device is still busy). When the I/O has completed, the driver puts the data (if any) where they are needed and returns.

The operating system then returns control to the caller. This method is called busy waiting and has the disadvantage of tying up the CPU polling the device until it is finished.

I/O Devices

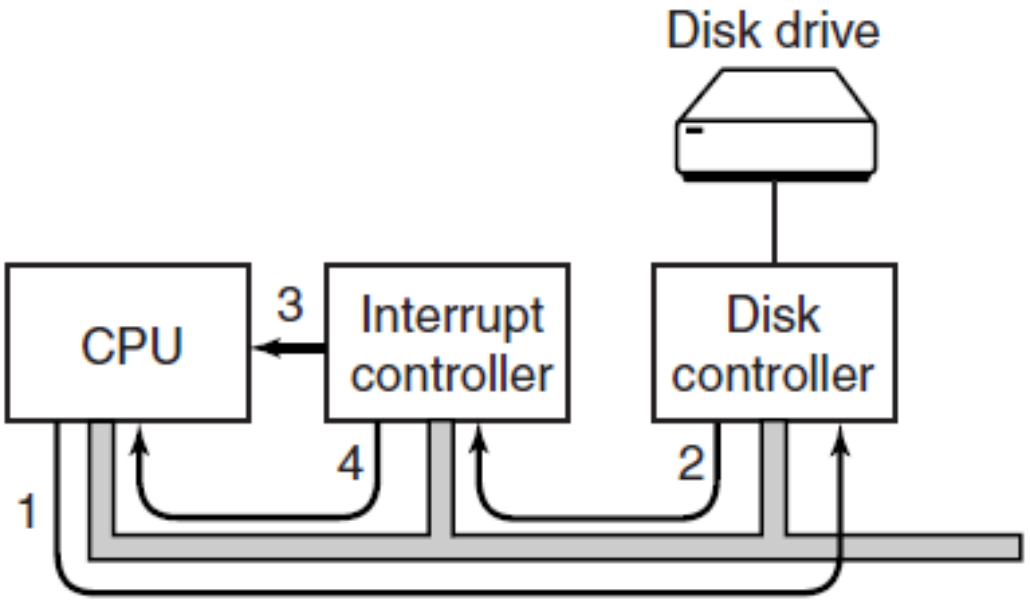
2. **Using interrupt**

The second method is for the driver to start the device and ask it to give an interrupt when it is finished. At that point the driver returns.
The operating system then blocks the caller if need be and looks for other work to do. When the controller detects the end of the transfer, it generates an **interrupt** to signal completion.

3. **Use of special hardware** e.g. DMA (**Direct Memory Access**)

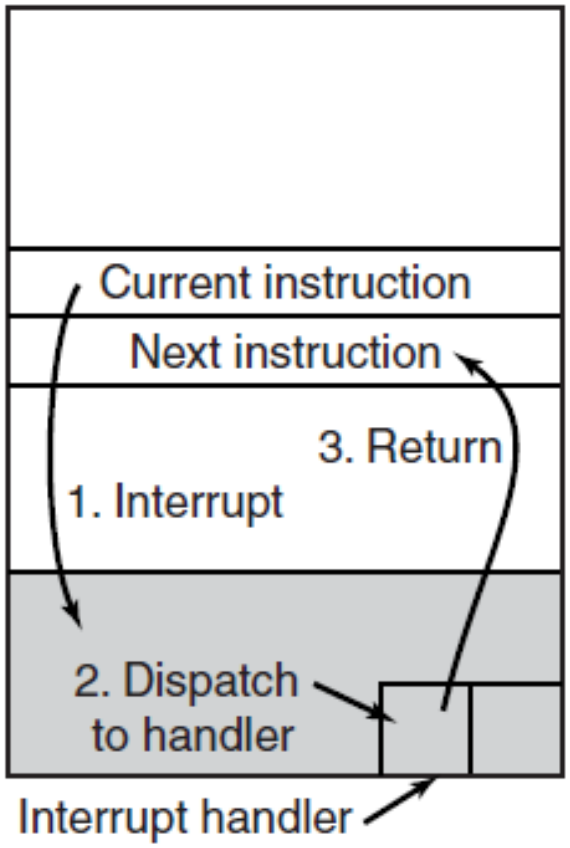
The CPU sets up the DMA chip, telling it how many bytes to transfer, the device and memory addresses involved, and the direction, and lets it go. When the DMA chip is done, it causes an interrupt.

I/O Devices



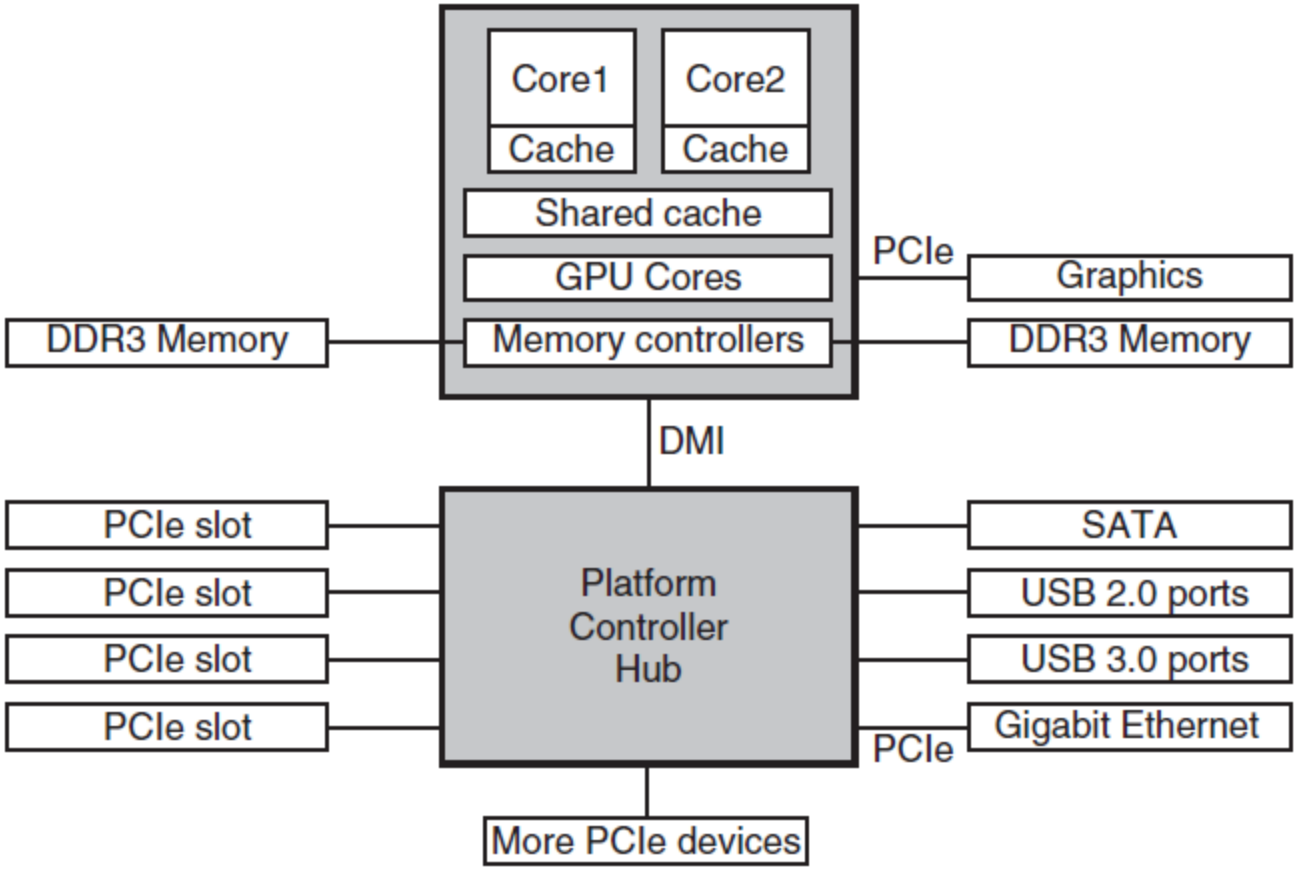
The steps in starting an I/O device and getting an interrupt

I/O Devices



Interrupt processing involves taking the interrupt, running the interrupt handler, and returning to the user program

I/O Devices
Buses



The structure of a large x86 system

I/O Devices
Buses

SCSI (**Small Computer System Interface**)

The SCSI bus is a high-performance bus intended for fast disks, scanners, and other devices needing considerable bandwidth. Nowadays, we find them mostly in servers and workstations. They can run at up to 640 MB/sec.

USB (**Universal Serial Bus**)

Any USB device can be connected to a computer and it will function immediately, without requiring a reboot.

DMI (Direct Media Interface) bus.

PCI bus

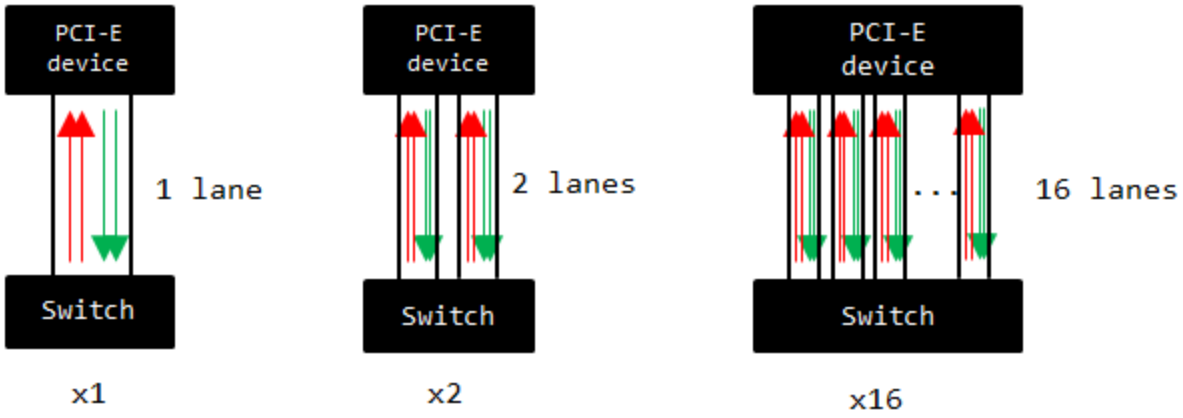
An internal synchronous bus for interconnecting chips, expansion boards, and processor/memory subsystems e.g. used for connecting adapters such as hard disks, sound cards, network cards and graphics cards.

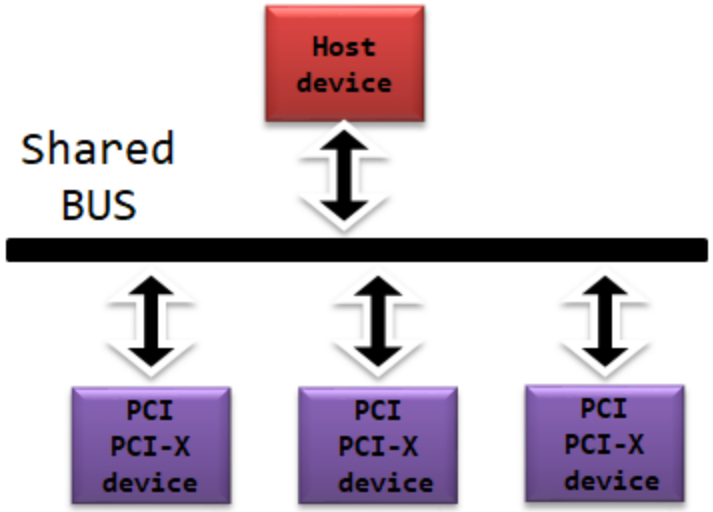
I/O Devices
Buses

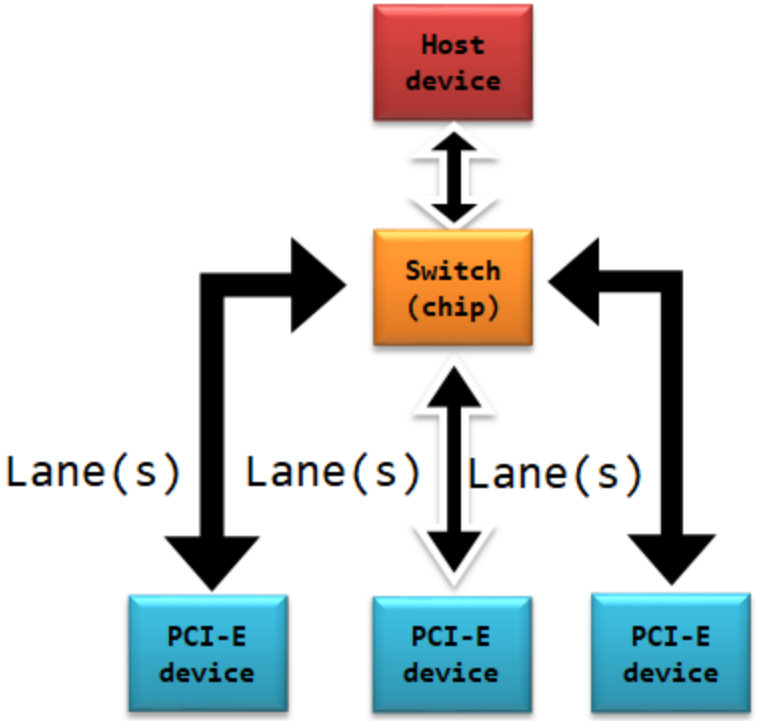
PCI-Express

Ideal for video and graphics applications. Not using a shared bus configuration and has a devoted path, called a lane, to a single chip known as a switch.

Each lane (1x) has conductors for sending and receiving data (red and green colored arrows).







The Operating System Zoo

1. **Mainframe Operating Systems**

   (*OS/390 and Linux*)

Room-sized computers

A mainframe with 1000 disks and millions of gigabytes of data is not unusual.

They typically offer three kinds of services: batch, transaction processing, and timesharing.

2. **Server Operating Systems**

   (Solaris, FreeBSD, Linux and Windows Server 201x)

They run on servers, which are either very large personal computers, workstations, or even mainframes.

The Operating System Zoo

3. **Multiprocessor Operating Systems**
   (Windows and Linux)

4. **Personal Computer Operating Systems**
   (Linux, FreeBSD, Windows 7, Windows 8, and Apple's OS X)

5. **Handheld Computer Operating Systems**
   (Android and Apple's iOS)

6. **Embedded Operating Systems**
   (Embedded Linux, QNX and VxWorks)

Embedded systems run on the computers that control devices that are not generally thought of as computers and which do not accept user-installed software. Typical examples are microwave ovens, TV sets, cars, DVD recorders, traditional phones, and MP3 players.

The Operating System Zoo

7. **Sensor Node Operating Systems**

(TinyOS)

Networks of tiny sensor nodes are being deployed for numerous purposes. These nodes are tiny computers that communicate with each other and with a base station using wireless communication. Sensor networks are used to protect the perimeters of buildings, guard national borders, detect fires in forests, measure temperature and precipitation for weather forecasting, glean information about enemy movements on battlefields, and much more.

8. **Real-Time Operating Systems**

(eCos)

These systems are characterized by having time as a key parameter For example, in industrial process-control systems, real-time computers have to collect data about the production process and use it to control machines in the factory.

The Operating System Zoo

9. **Smart Card Operating Systems**

(JVM)

Some smart cards are Java oriented. This means that the ROM on the smart card holds an interpreter for the Java Virtual Machine (JVM). Java applets (small programs) are downloaded to the card and are interpreted by the JVM interpreter.

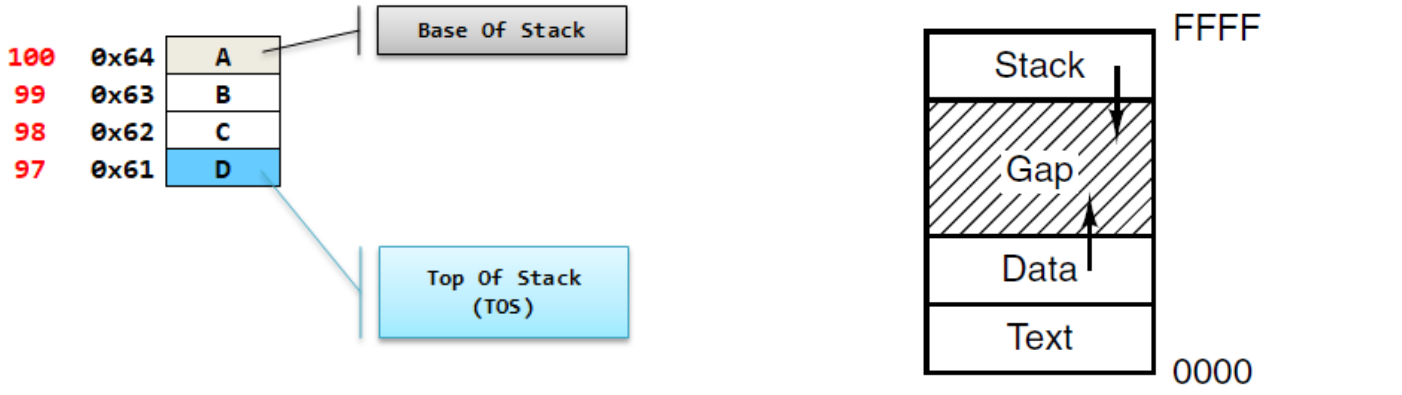
Processes

- Key concept in all operating systems.
- program in execution.
- Process is associated with an address space.
- Each process has some set of addresses it can use, typically running from 0 up to some maximum (address space).
- Also associated with set of resources.
- Process can be thought of as a container.
  - Holds all information needed to run program

In many operating systems, all the information about each process, other than the contents of its own address space, is stored in an operating system table called the **process table**, which is an array of structures, one for each process currently in existence.

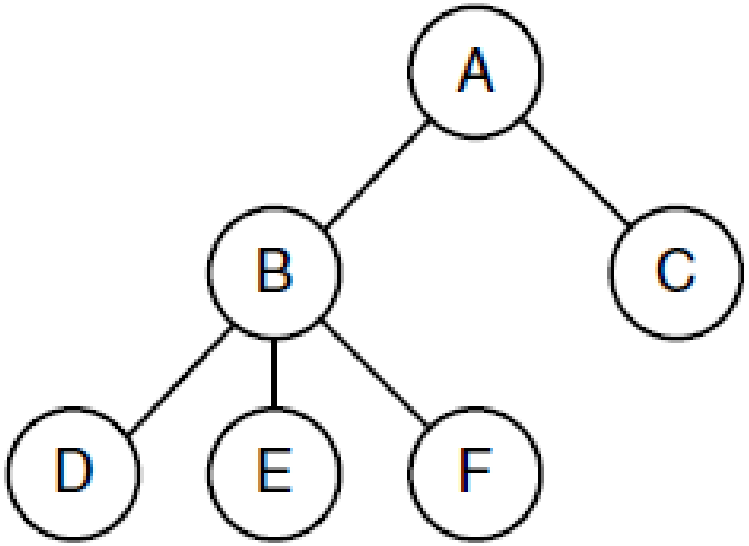
Processes

Processes in UNIX have their memory divided up into three segments: the text segment (i.e., the program code), the data segment (i.e., the variables), and the stack segment. The data segment grows upward and the stack grows downward.



Processes have three segments: text, data, and stack.

Processes



A process tree. Process A created two child processes, B and C. Process B created three child processes, D, E, and F.

```
OS:>>
?(1)──mintty(345)──bash(346)──child_fork(598)─┬─child_fork(601)
                                              └─child_fork(599)──child_fork(600)
```

Files

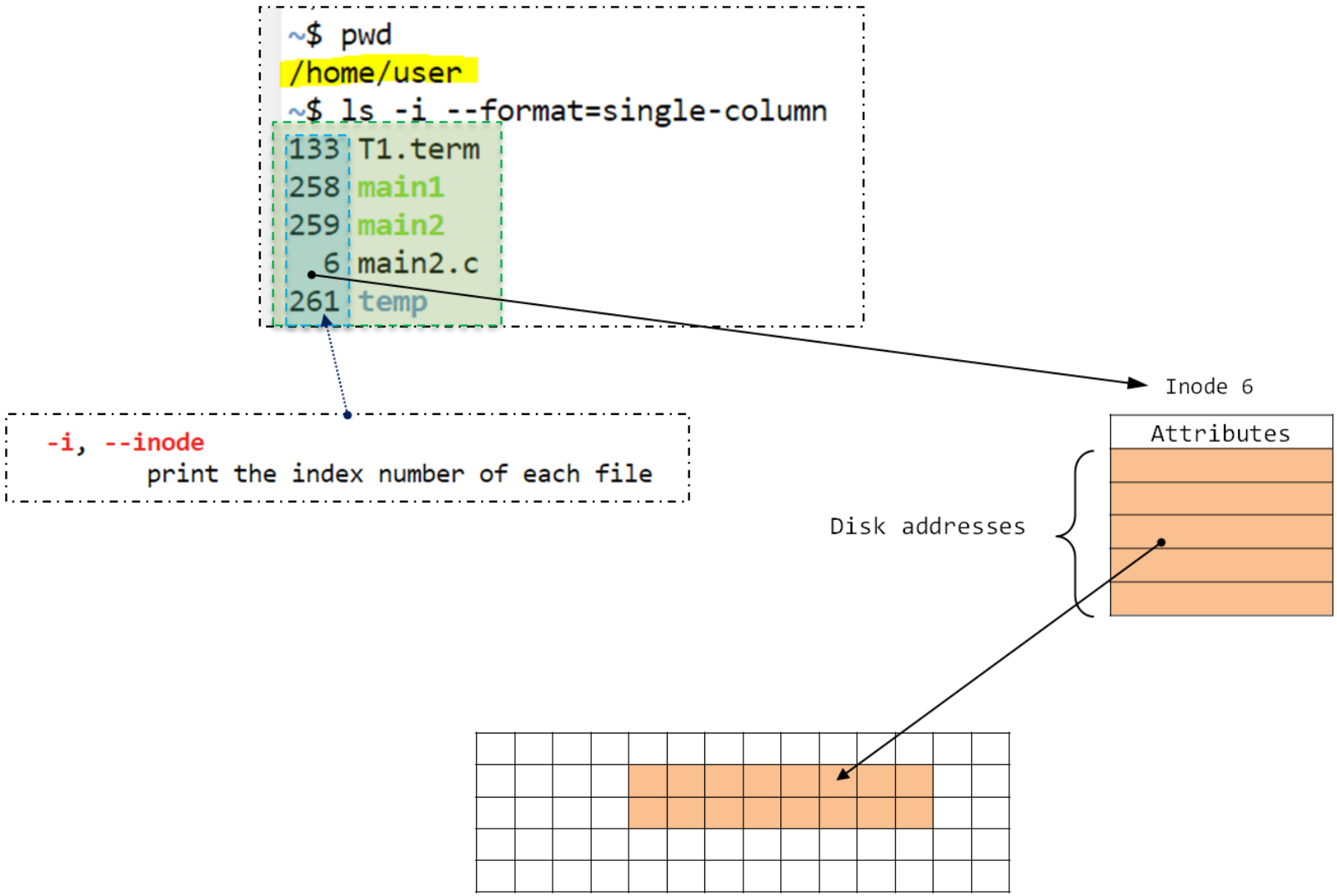
a major function of the operating system is to hide the peculiarities of the disks and other I/O devices and present the programmer with a nice, clean abstract model of device-independent files.

Files are logical units of information created by processes.

Directory, a way of grouping files together. File hierarchies are organized as trees.

Every file in UNIX has a unique number, its i-number, that identifies it. This i-number is an index into a table of i-nodes, one per file, telling who owns the file, where its disk blocks are, and so on.

A directory is simply a file containing a set of (i-number, ASCII name) pairs.


~$ pwd
/home/user
~$ ls -i --format=single-column
133 T1.term
258 main1
259 main2
6 main2.c
261 temp
-i, --inode
print the index number of each file
Inode 6
Attributes
Disk addresses

Files

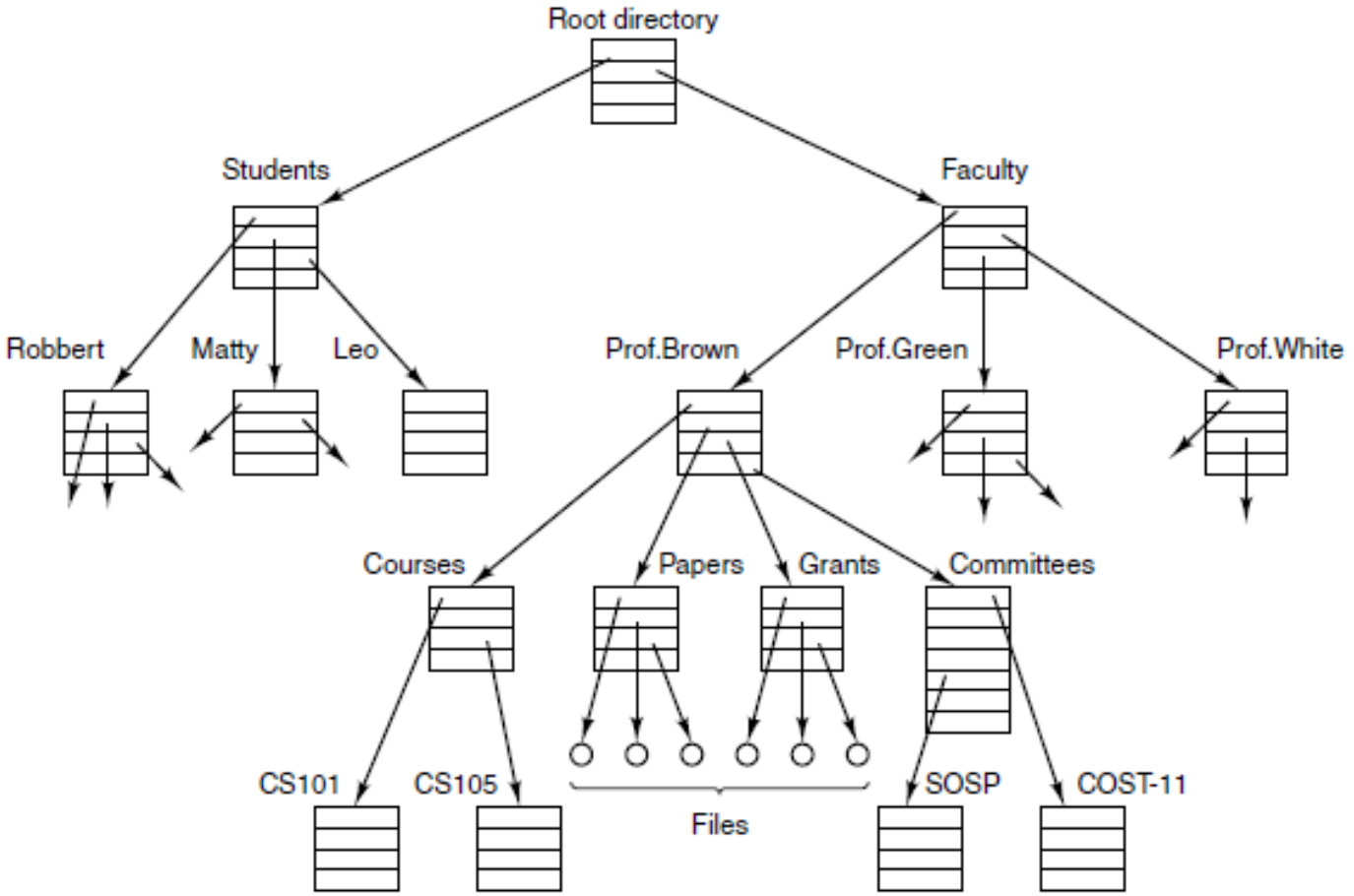
Every file within the directory hierarchy can be specified by giving its path name from the root directory. The path for file CS101 is

/Faculty/Prof.Brown/Courses/CS101 (Unix)

\Faculty\Prof.Brown\Courses\CS101 (MS_DOS)

Before a file is read or written, it must be opened by checking the permissions. If the access is permitted, the system returns a small integer called a file descriptor to use in subsequent operations. If the access is prohibited, an error code is returned.

Files

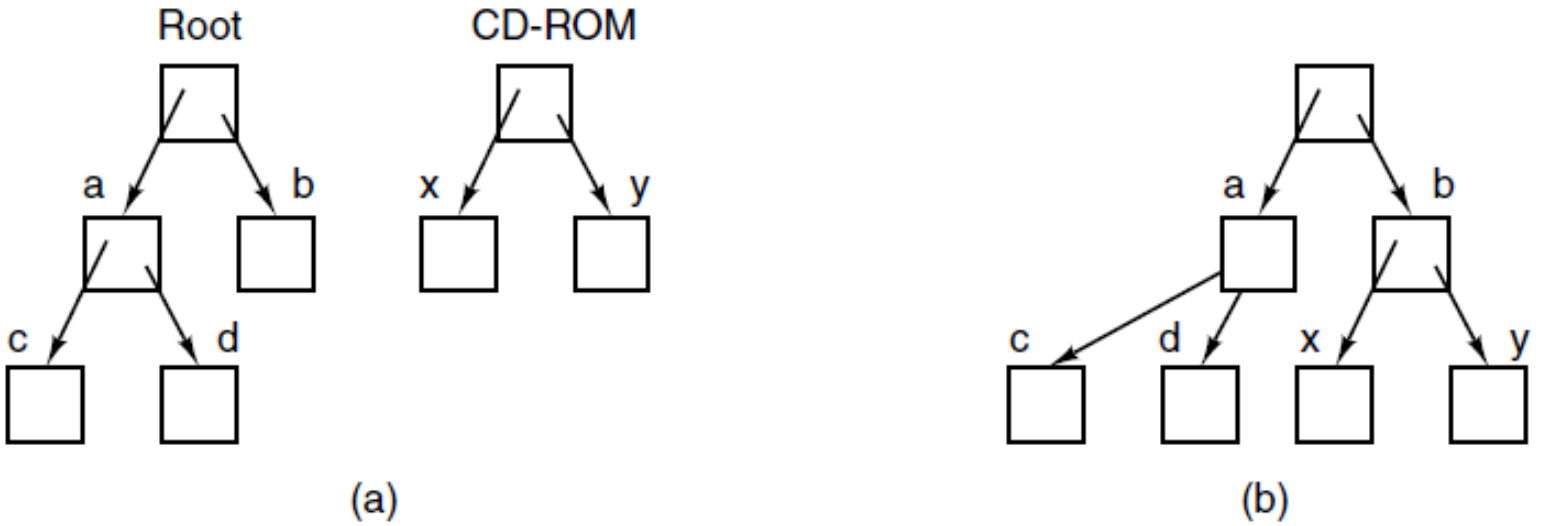


A file system for a university department.

Files

The mount system call allows the file system on the CD-ROM to be attached to the root file system wherever the program wants it to be.

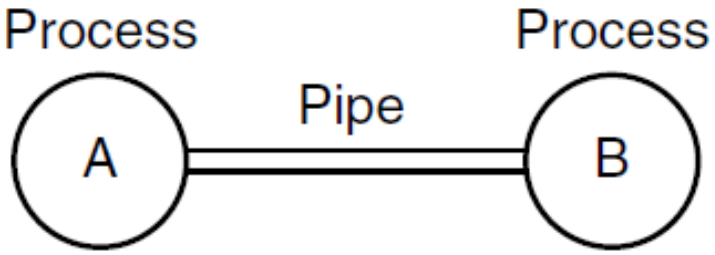
File systems are nearly always mounted on empty directories.



(a) Before mounting, the files on the CD-ROM are not accessible. (b) After mounting, they are part of the file hierarchy.

Files

•A **pipe** is a sort of pseudofile that can be used to connect two processes. One process generates some output that another process uses as input.



Two processes connected by a pipe.

```
OS:>> cat test.txt | tr '[A-Z]' '[a-z]' | tr -C '[a-z]' '\n'|sort| uniq -c|sort -rn
    283
    115 the
     52 and
     31 to
     22 of
     21 was
     20 cow
     19 they
     18 in
     17 that
     16 as
     15 her
     14 their
     14 a
     13 out
     12 she
     11 bell
     11 at
     10 so
      9 with
```

Protection

Computers contain large amounts of information that users often want to protect and keep confidential. This information may include email, business plans, tax returns, and much more. It is up to the operating system to manage the system security so that files, for example, are accessible only to authorized users.

Files in UNIX are protected by assigning each one a 9-bit binary protection code. The protection code consists of three 3-bit fields, one for the owner, one for other members of the owner's group (users are divided into groups by the system administrator), and one for everyone else. Each field has a bit for read access, a bit for write access, and a bit for execute access. These 3 bits are known as the **rwx** bits.

```
-rw-r--r-- 1 A01234567 AD+Group(513)  6340 Jul 18 21:30 res.txt
-rwxr-xr-x 1 A01234567 AD+Group(513)   191 Jul 18 21:50 run.txt
```

Protection
Linux file types

- : regular file
d : directory
c : character device file
b : block device file
s : local socket file
p : named pipe
l : symbolic link

Protection bits

```
asp@asp-VirtualBox:~/Desktop/test$ ls -al
total 12
drwxrwxr-x 3 asp  asp  4096 Jun 17 08:33 .
drwxr-xr-x 4 asp  asp  4096 Jun 17 08:23 ..
drwxrwxr-x 2 asp  asp  4096 Jun 17 08:25 libs
lrwxrwxrwx 1 asp  asp     8 Jun 17 08:25 l_thread -> thread.c
brw-r--r-- 1 root root 1, 2 Jun 17 08:28 my_block_dev
crw-r--r-- 1 root root 4, 3 Jun 17 08:29 my_char_dev
prw-r--r-- 1 root root    0 Jun 17 08:30 my_npipe_dev
-rw-rw-r-- 1 asp  asp     0 Jun 17 08:23 thread.c
asp@asp-VirtualBox:~/Desktop/test$
```