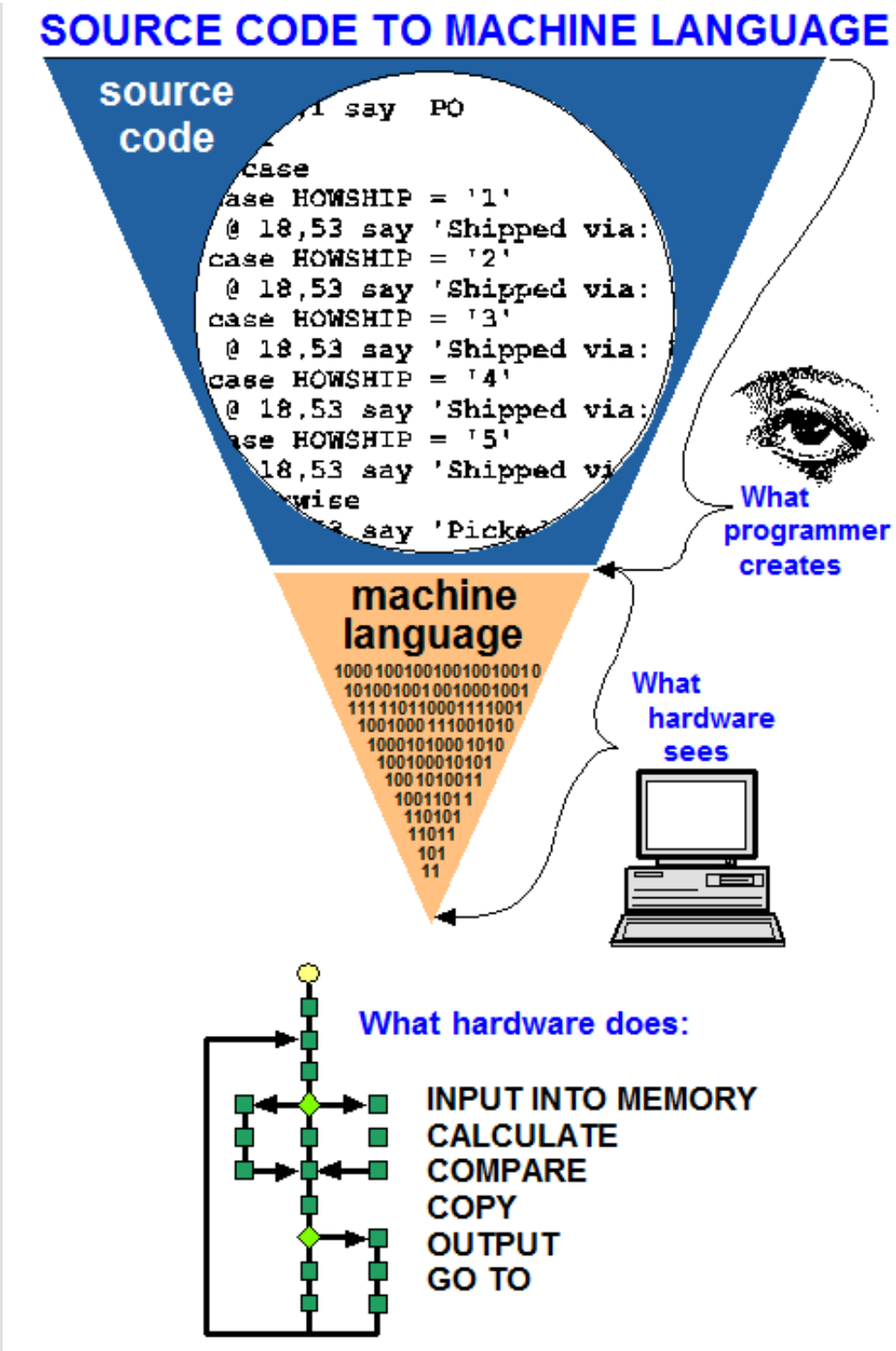
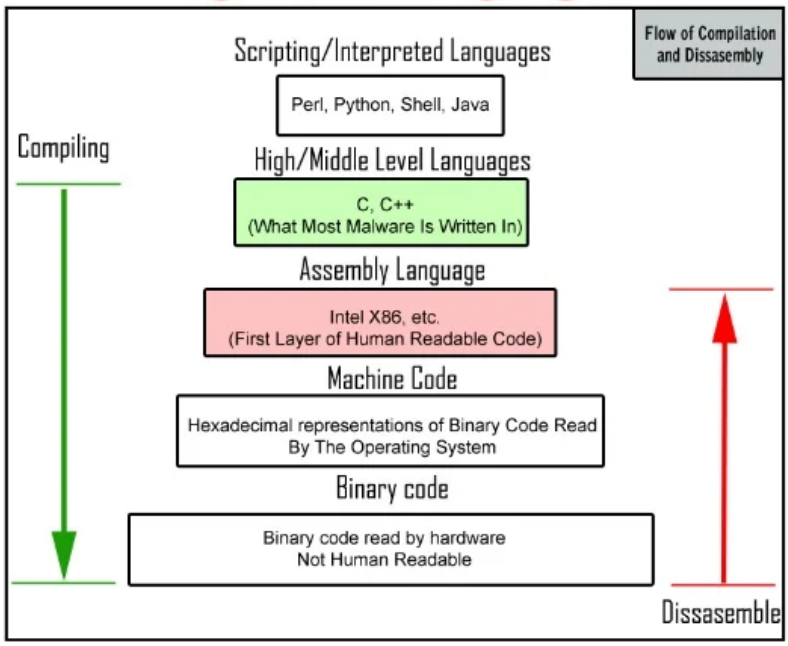
Computer Science

Programming Language -> Communicate with Computer



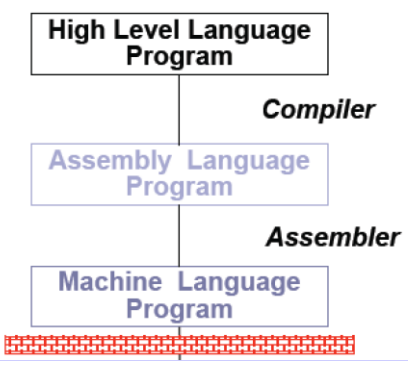


High Vs. Low Level Languages

Human languages	E.g., English, French, Spanish, Chinese, German, Arabic etc.	High level
High level programming language	E.g., Python, Java, C++ for (i = 1; i <= 10; i++)	
Low level programming language	Assembly MOV #10, R0	
Machine language	Binary 10100000 1010 00	
	Computer hardware	Low level

James Tam

Levels of Representation



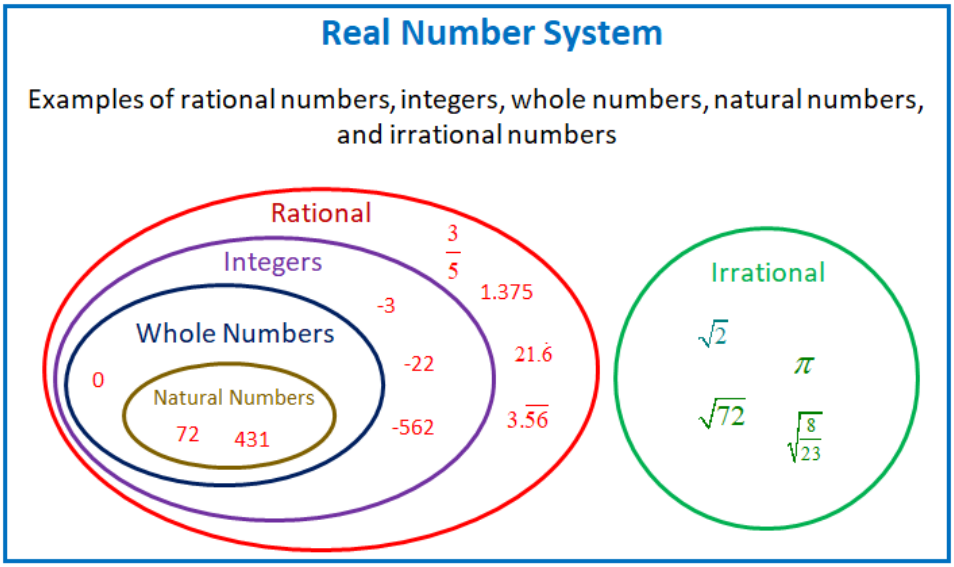

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

- mov BL, 02h
- mov AX,0A56h
- mul BL

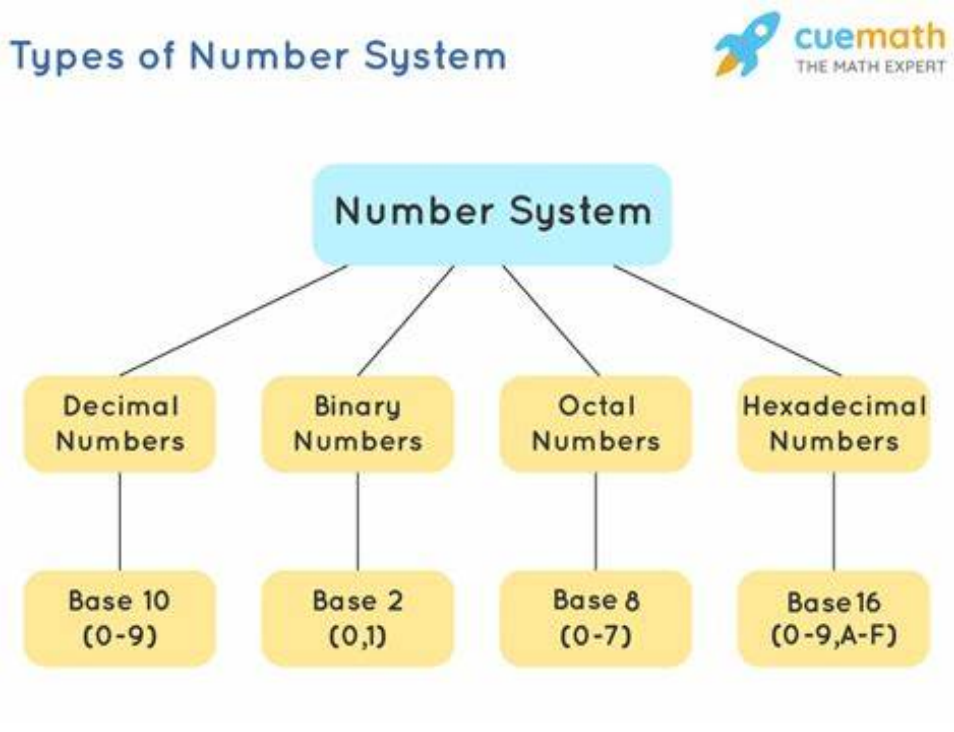
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Number System

Real Number System



Number System Conversion



Number Systems Conversion Chart

Binary

Place	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Weight	2048	1024	512	256	128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625

Binary, Hex, and Octal Conversions

Binary	Octal	Hexadecimal	Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

Methodology

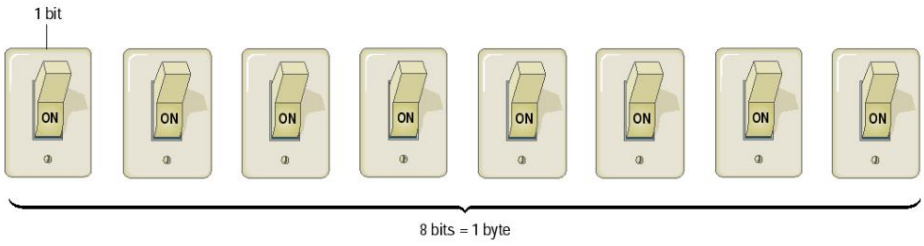
1. Convert from decimal to binary
   Divide the decimal by the largest binary weight it is divisible by and place a "1" in that column. Then select the next largest weight, if it is divisible put a "1" in that column otherwise place a "0" in the column. Continue until all the columns have either a "1" or "0" resulting in a binary expression.
2. Convert from decimal to hex.
   Convert to binary first, then group the binary in groups of 4 beginning on the right working to the left. For each group determine the hex value based on the table to the left.
3. Convert to octal
   Convert to binary first, then group the binary in groups of 3 beginning on the right working to the left. For each group determine the octal value based on the table to the left.

Binary Number System

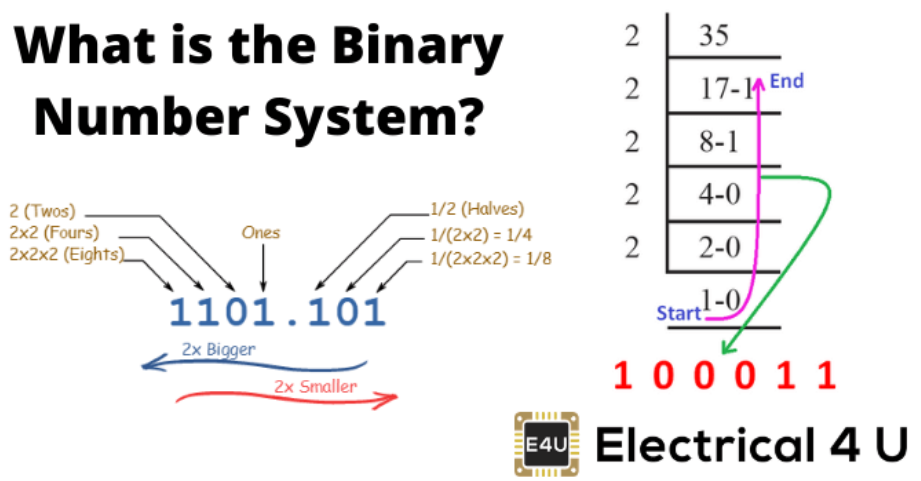
Number System

Bits and Bytes

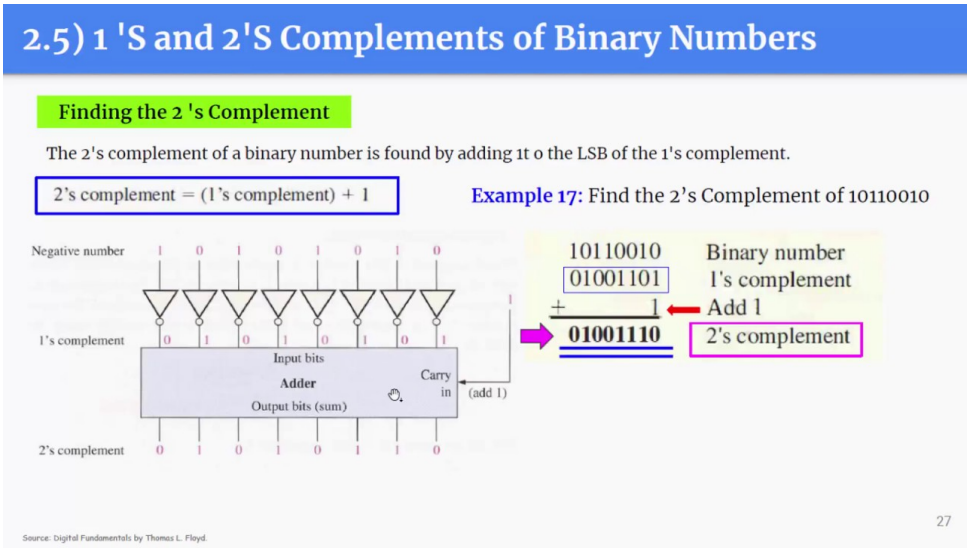
- A single unit of data is called a bit, having a value of 1 or 0.
- Computers work with collections of bits, grouping them to represent larger pieces of data, such as letters of the alphabet.
- Eight bits make up one byte. A byte is the amount of memory needed to store one alphanumeric character.
- With one byte, the computer can represent one of 256 different symbols or characters.

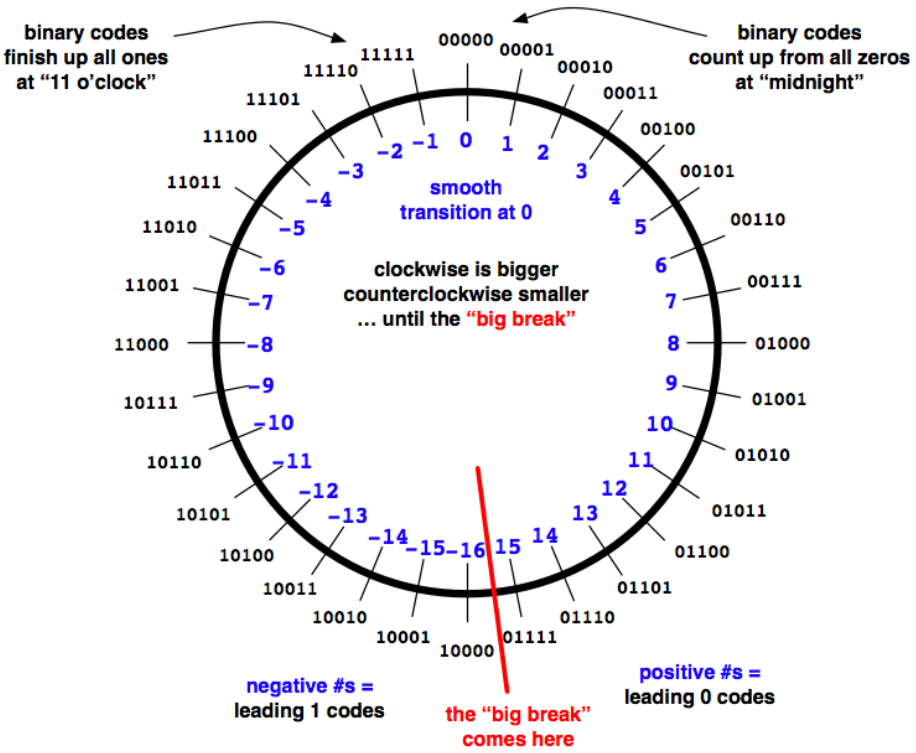


1 1 1 1 1 1 1 1


What is the Binary Number System?
2 (Twos)
2x2 (Fours)
2x2x2 (Eights)
Ones
1/2 (Halves)
1/(2x2) = 1/4
1/(2x2x2) = 1/8
1101.101
2x Bigger
2x Smaller
2 35
2 17-1
2 8-1
2 4-0
2 2-0
2 1-0
End
Start
1 0 0 0 1 1
E4U
Electrical 4 U

Complement


2.5) 1 'S and 2'S Complements of Binary Numbers
Finding the 2 's Complement
The 2's complement of a binary number is found by adding 1t o the LSB of the 1's complement.
2's complement = (1's complement) + 1
Example 17: Find the 2's Complement of 10110010
Negative number
1's complement
Input bits
Adder
Output bits (sum)
Carry in
(add 1)
2's complement
10110010 Binary number
01001101 1's complement
+ 1 Add 1
01001110 2's complement
27
Source: Digital Fundamentals by Thomas L. Floyd


binary codes finish up all ones at "11 o'clock"
binary codes count up from all zeros at "midnight"
smooth transition at 0
clockwise is bigger
counterclockwise smaller
... until the "big break"
negative #s = leading 1 codes
positive #s = leading 0 codes
the "big break" comes here

Complements of numbers

(r-1)'s Complement

•Given a number N in base r having n digits,
•the (r- 1)'s complement of N is defined as
$(r^n - 1) - N$

•For decimal numbers the base or $r = 10$ and $r- 1= 9$,

•so the 9's complement of N is $(10^n-1)-N$

•$99999\ldots\ldots - N$

9	9	9	9	9
Digit n	Digit n-1	Next digit	Next digit	First digit

-

Complements

♣ There are two types of complements for each base-r system: the radix complement and diminished radix complement.

⟹ the r's complement and the second as the $(r - 1)$'s complement.

Diminished Radix Complement

Given a number N in case r having n digits, the $(r - 1)$'s complement of N is defined as $(r^n - 1) - N$. For decimal numbers, $r = 10$ and $r - 1 = 9$, so the 9's complement of N is $(10^n - 1) - N$.

Example:

The 9's complement of 546700 is $999999 - 546700 = 453299$.
The 9's complement of 012398 is $999999 - 012398 = 987601$.

♣ For binary numbers, $r = 2$ and $r - 1 = 1$, so the 1's complement of N is $(2^n - 1) - N$.

Example:

The 1's complement of 1011000 is 0100111
The 1's complement of 0101101 is 1010010

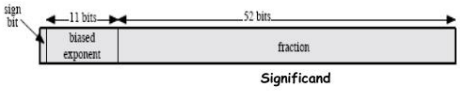
Precision floating-point

Double Precision Floating Point Numbers
IEEE Standard

64 bit Double Precision Floating Point Numbers are stored as:

S EEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

S: Sign - 1 bit
E: Exponent - 11 bits
F: Fraction - 52 bits



The value V:

- If E=2047 and F is nonzero, then V= NaN ("Not a Number")
- If E=2047 and F is zero and S is 1, then V= - Infinity
- If E=2047 and F is zero and S is 0, then V= Infinity
- If 0<E<2047 then V= (-1)**S * 2 ** (E-1023) * (1.F) (exponent range = -1023 to +1024)
- If E=0 and F is nonzero, then V= (-1)**S * 2 ** (-1022) * (0.F) ("unnormalized" values)
- If E=0 and F is zero and S is 1, then V= - 0
- If E=0 and F is zero and S is 0, then V= 0

Note: 2047 decimal = 11111111111 in binary (11 bits)

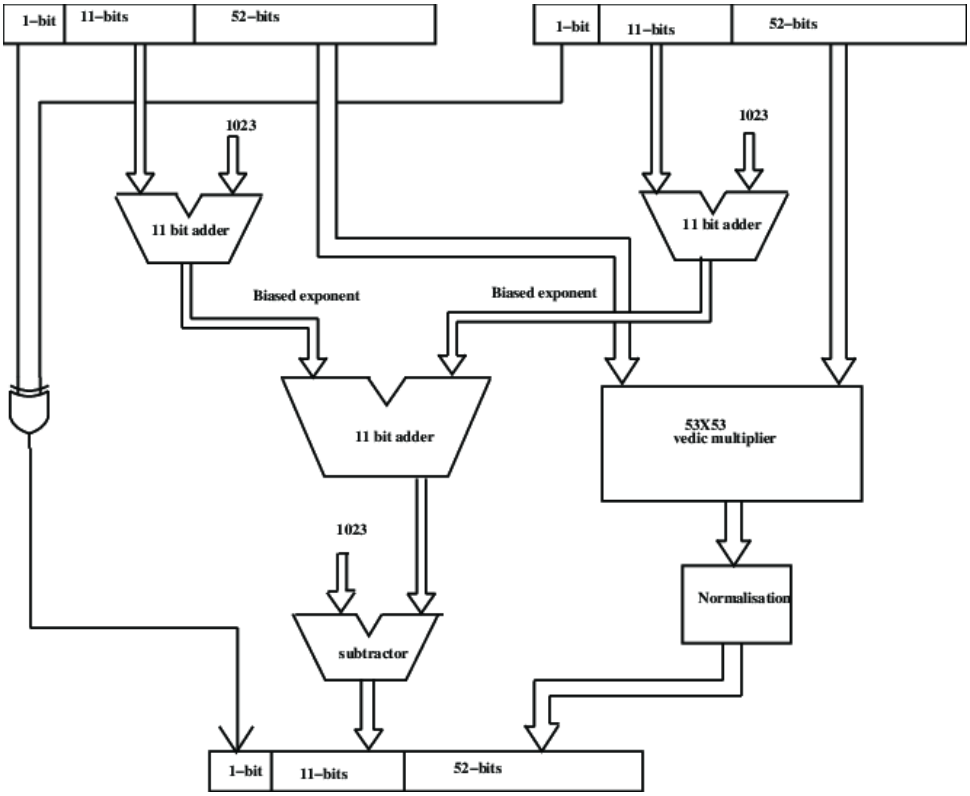
Biased Exponent - Cont'd

- For double precision, exponent field is 11 bits
  - *E* can be in the range 0 to 2047
  - *E* = 0 and *E* = 2047 are reserved for special use
  - *E* = 1 to 2046 are used for normalized floating point numbers
  - Bias = 1023 (half of 2046), val(*E*) = *E* – 1023
  - val(*E*=1) = –1022, val(*E*=1023) = 0, val(*E*=2046) = 1023
- Value of a Normalized Floating Point Number is

$$(-1)^S \times (1.F)_2 \times 2^{E-\text{Bias}}$$

$$(-1)^S \times (1.f_1f_2f_3f_4\ldots)_2 \times 2^{E-\text{Bias}}$$

$$(-1)^S \times (1 + f_1\times 2^{-1} + f_2\times 2^{-2} + f_3\times 2^{-3} + f_4\times 2^{-4}\ldots)_2 \times 2^{E-\text{Bias}}$$


1–bit
11–bits
52–bits
1–bit
11–bits
52–bits
1023
1023
11 bit adder
11 bit adder
Biased exponent
Biased exponent
11 bit adder
53X53
vedic multiplier
1023
Normalisation
subtractor
1–bit
11–bits
52–bits

ASCII

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	|	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Representing Text

ASCII Example

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

National Taiwan University OpenCourseWare 臺大開放式課程

Instructor: Tian-Li Yu Data Storage 17 / 1

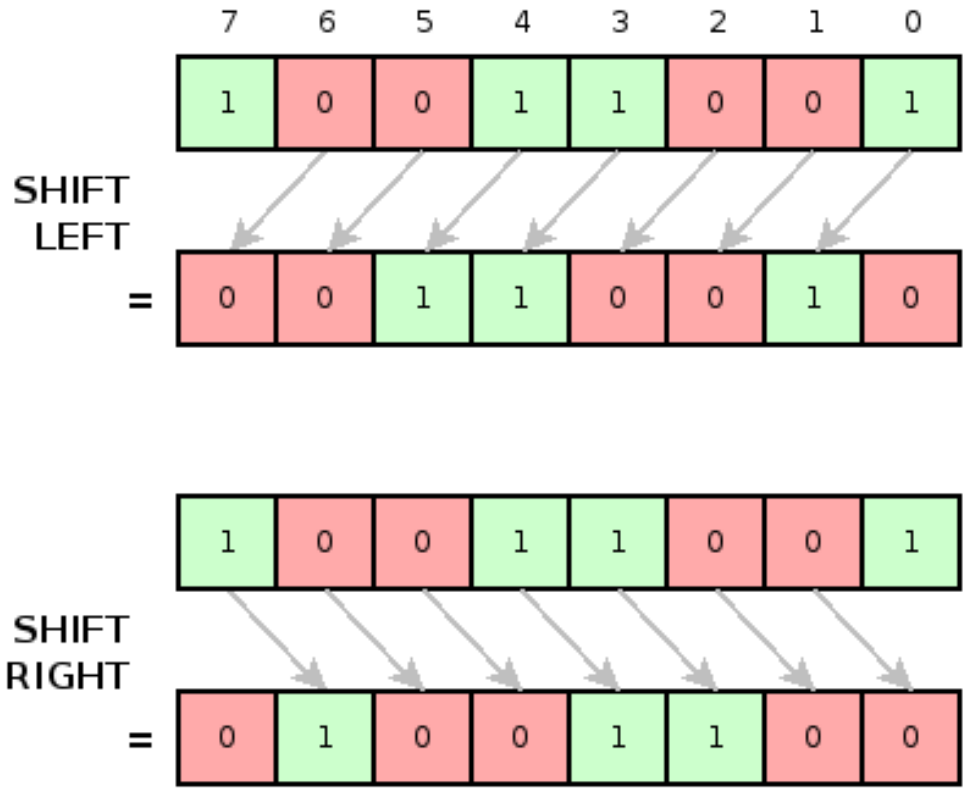
Bitwise Operation

Bitwise Operators

int a = 10, b = 2 for all examples below

Operator	Meaning	Example	Result
~	Bitwise unary NOT	~a	-11
&	Bitwise AND	a&b	2
|	Bitwise OR	a|b	10
^	Bitwise Ex-OR	a^b	8
>>	Shift right	a>>1	5
>>>	Shift right zero fill	a>>>1	5
<<	Shift left	a<<1	20
&=	Bitwise AND assignment	a &= b	2
|=	Bitwise OR assignment	a |= b	10
^=	Bitwise Ex-OR assignment	a ^= b	8
>>=	Shift right assignment	a >>= 1	5
>>>=	Shift right zero fill assignment	a >>>=1	5
<<=	Shift left assignment	a <<= 1	20

Startertutorials.com



Machine Language

Machine Language

- Instructions, like registers and words of data, are also 32 bits long
  - Example: add $t0, $s1, $s2
  - registers have numbers, $t0=8, $s1=17, $s2=18
- Instruction Format:

000000	10001	10010	01000	00000	100000
op	rs	rt	rd	shamt	funct

- **Can you guess what the field names stand for?**

op = Basic operation of the instruction: opcode
rs = The first register source operand
rt = The second register source operand
rd = The register destination operand
shamt = shift amount
funct = function code

25

Assembly Language

Assembly Language

- Tied to the specifics of the underlying machine
- Commands and names to make the code readable and writeable by humans
- Hand-coded assembly code may be more efficient
- E.g., IA32 from Intel

```
      movl   $0, %ecx
.loop:
      cmpl   $1, %edx
      jle    .endloop
      addl   $1, %ecx
      movl   %edx, %eax
      andl   $1, %eax
      je     .else
      movl   %edx, %eax
      addl   %eax, %edx
      addl   %eax, %edx
      addl   $1, %edx
      jmp    .endif
.else:
      sarl   $1, %edx
.endif:
      jmp    .loop
.endloop:
```

Reading IA32 Assembly Language

- Assembler directives: starting with a period (“.”)
  - E.g., “.section .text” to start the text section of memory
  - E.g., “.loop” for the address of an instruction
- Referring to a register: percent size (“%”)
  - E.g., “%ecx” or “%eip”
- Referring to a constant: dollar sign (“$”)
  - E.g., “$1” for the number 1
- Storing result: typically in the second argument
  - E.g. “addl $1, %ecx” increments register ECX
  - E.g., “movl %edx, %eax” moves EDX to EAX
- Comment: pound sign (“#”)
  - E.g., “# Purpose: Convert lower to upper case”

28

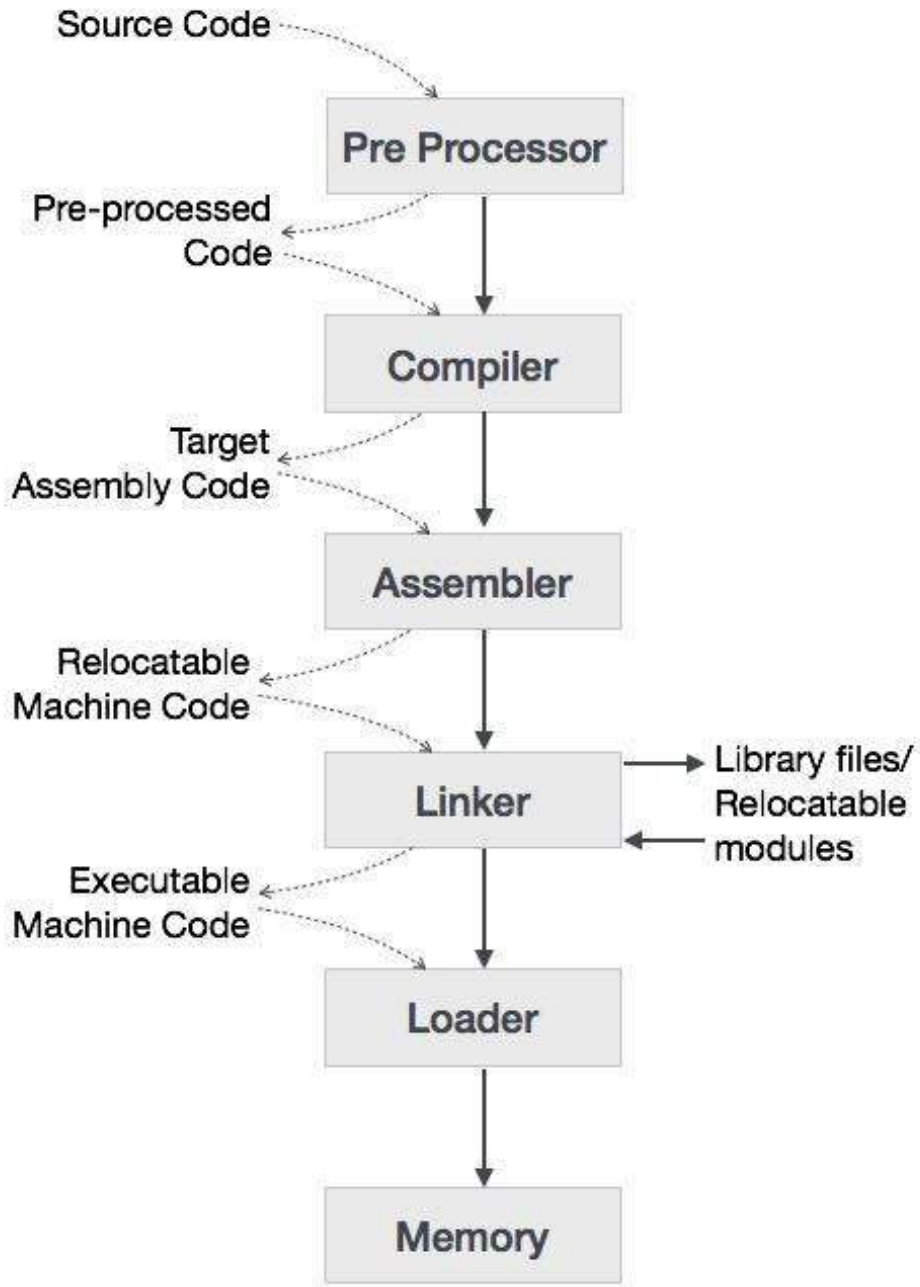
High level language

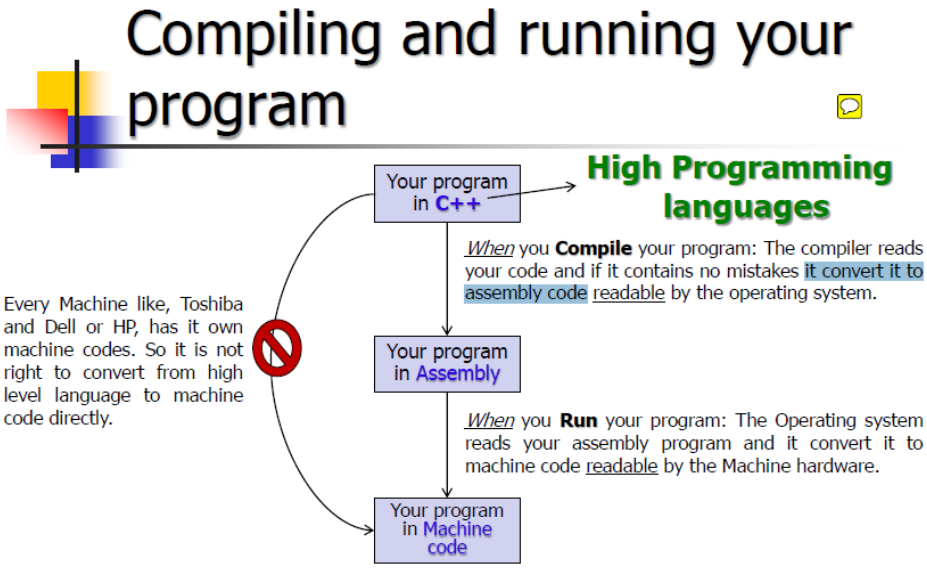
- As stated earlier, a program written in any programming language is a set of logically related instructions. These instructions have two parts, as shown in the following figure:
- The two parts of a programming language instruction are:
  - ,*Operation code (opcode): This part instructs a computer about the operation to be performed.*
  - ,*Operand: This part instructs the computer about the location of the data on which the operation specified by the opcode is to be performed.*

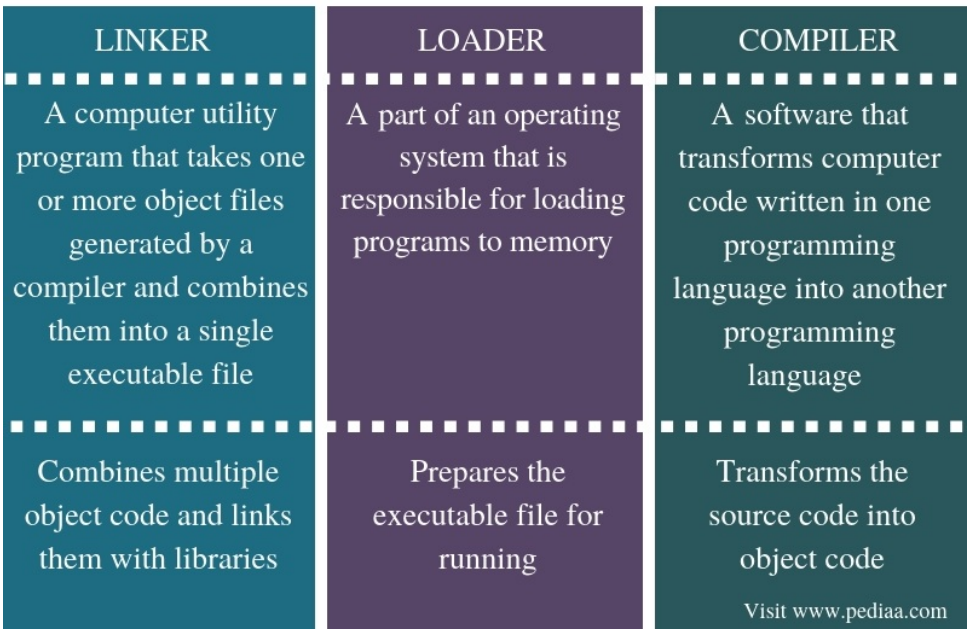
- For example, in the instruction Add A and B, Add is the opcode and A and B are operands

NIIT Hai Phong Software Park 11

Compiler and Assember

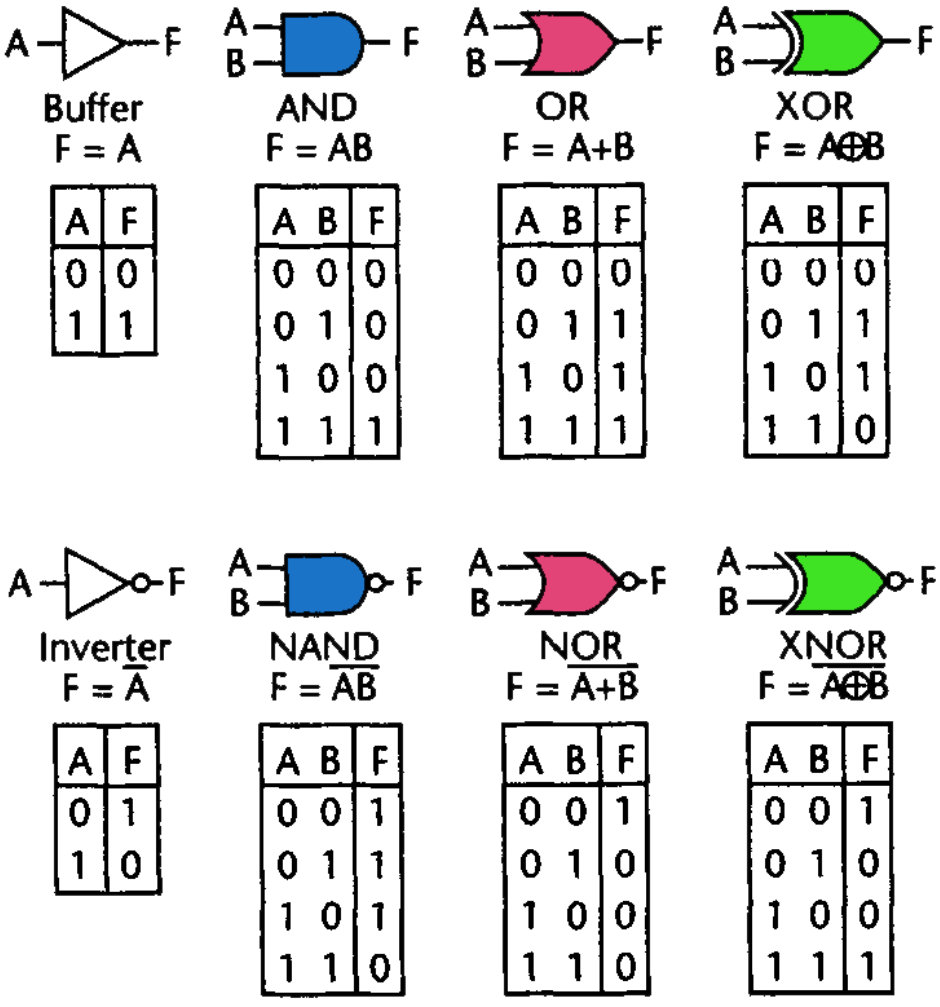

Source Code
Pre Processor
Pre-processed Code
Compiler
Target Assembly Code
Assembler
Relocatable Machine Code
Linker
Library files/ Relocatable modules
Executable Machine Code
Loader
Memory


Compiling and running your program
Your program in C++
High Programming languages
When you Compile your program: The compiler reads your code and if it contains no mistakes it convert it to assembly code readable by the operating system.
Every Machine like, Toshiba and Dell or HP, has it own machine codes. So it is not right to convert from high level language to machine code directly.
Your program in Assembly
When you Run your program: The Operating system reads your assembly program and it convert it to machine code readable by the Machine hardware.
Your program in Machine code

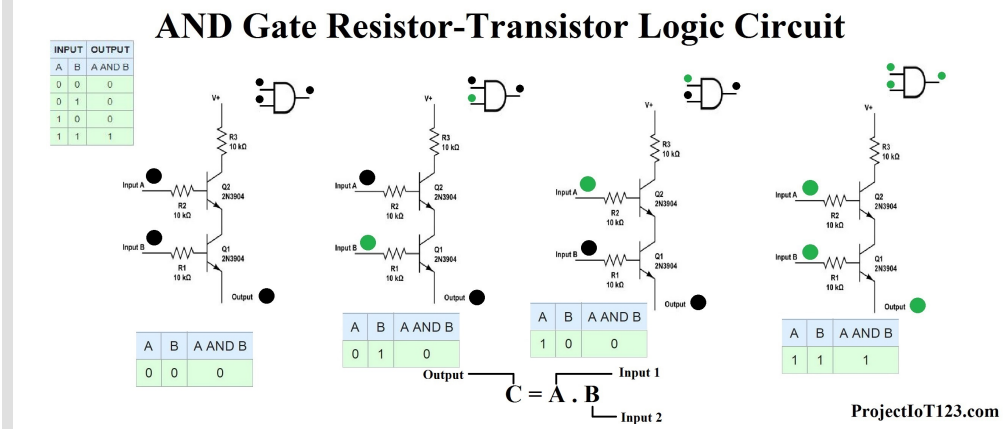


Logic Design

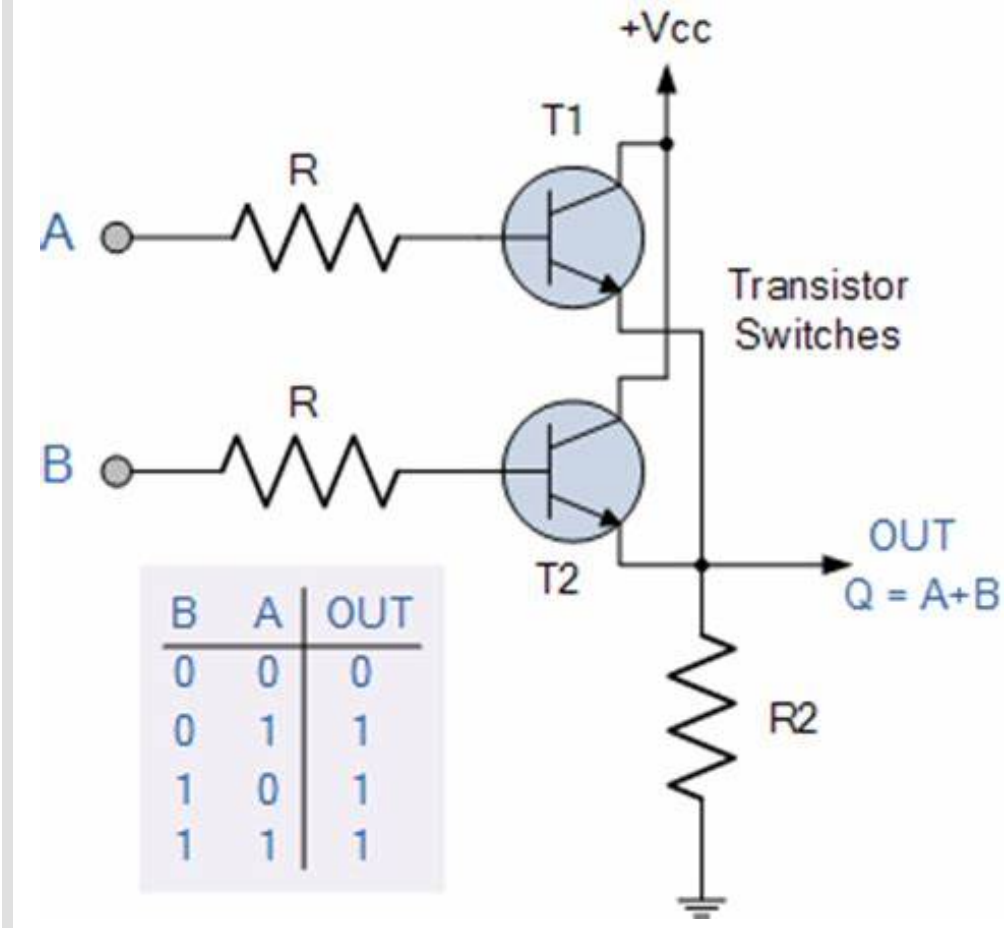
- **Logic Gates**



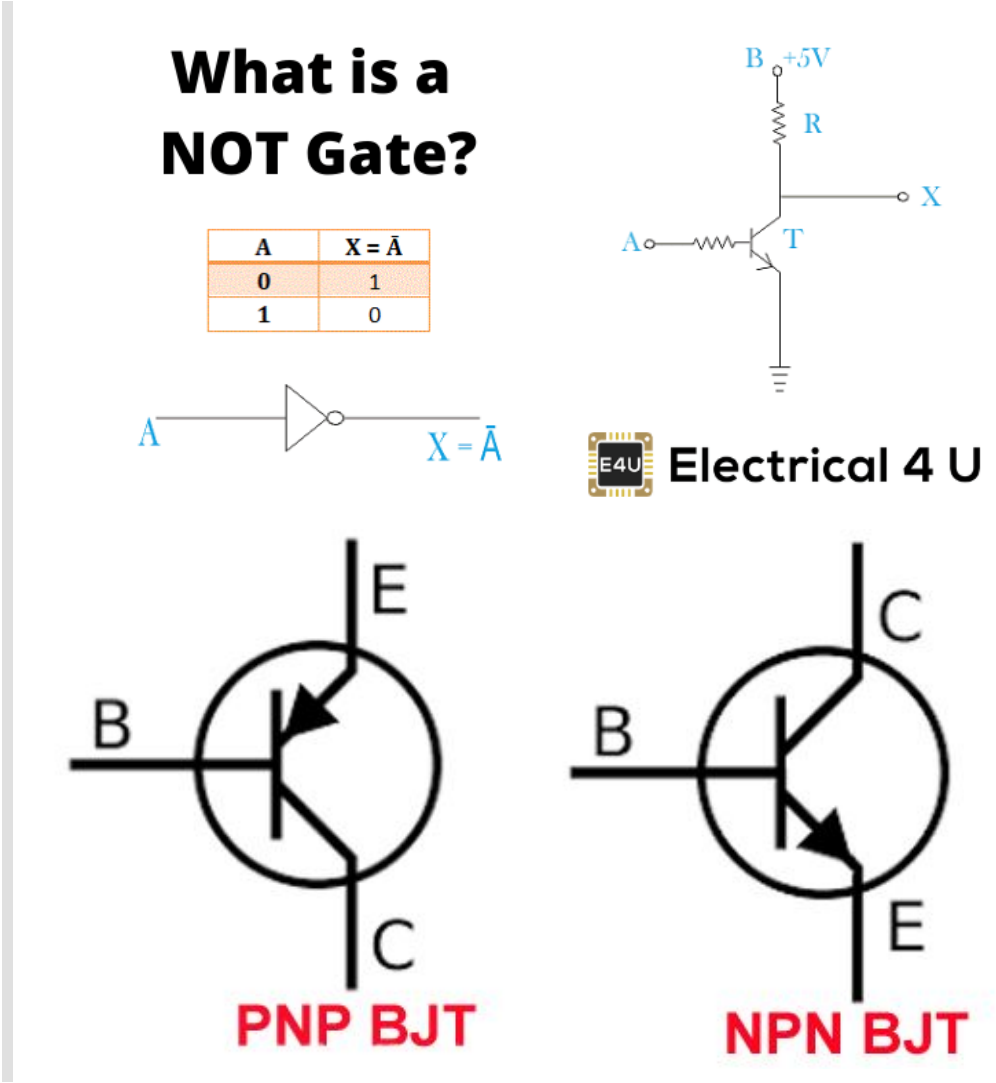
- **AND Gate**



- **OR Gate**



- **NOT Gate**



A	X = Ā
0	1
1	0

- **XOR Gate**

XOR GATE Truth Table

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

BOOLEAN EXPRESSION

$$A \cdot \overline{B} + \overline{A} \cdot B$$

$$(A + B) \cdot (\overline{A} + \overline{B})$$

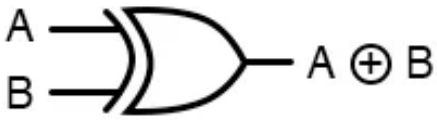
C = A⊕B

Input1

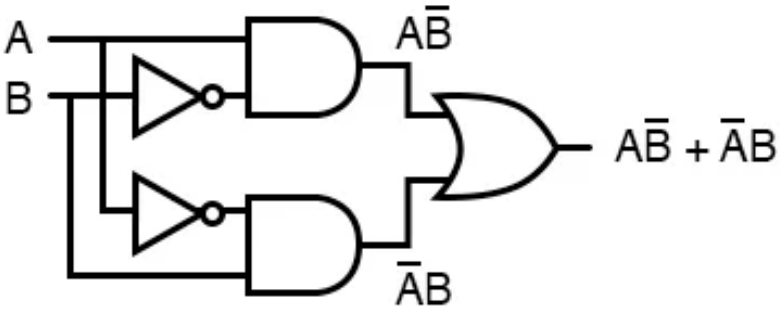
Input2

Output

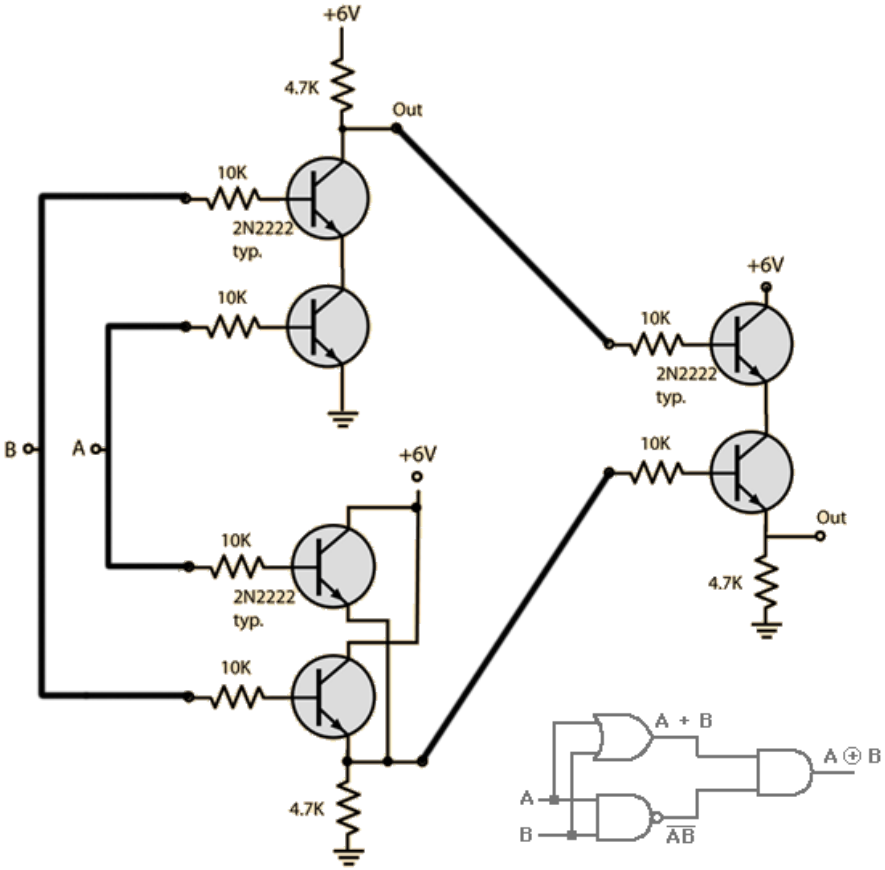
ProjectIoT123.com



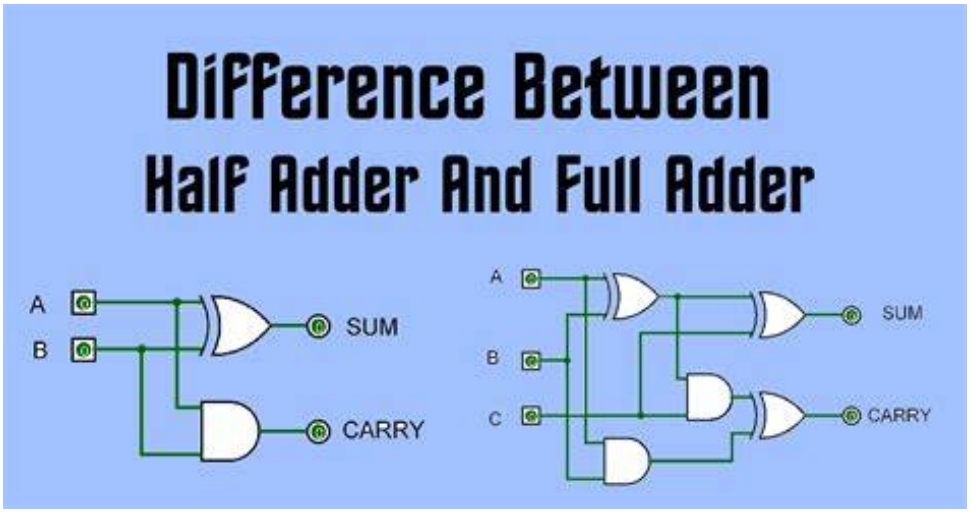
. . . is equivalent to . .



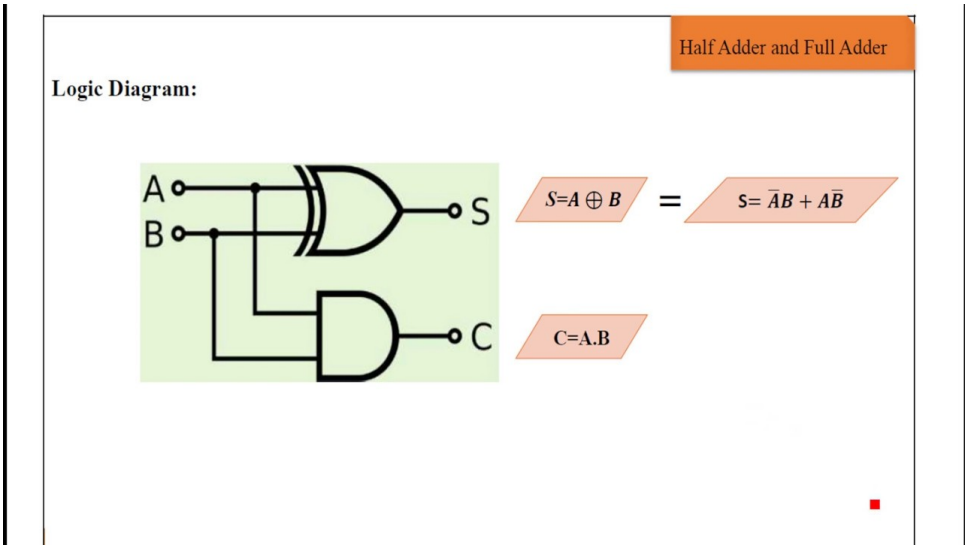
$$A \oplus B = A\overline{B} + \overline{A}B$$



- **Hal-Adder vs Full-Adder**



- **Half-Adder**



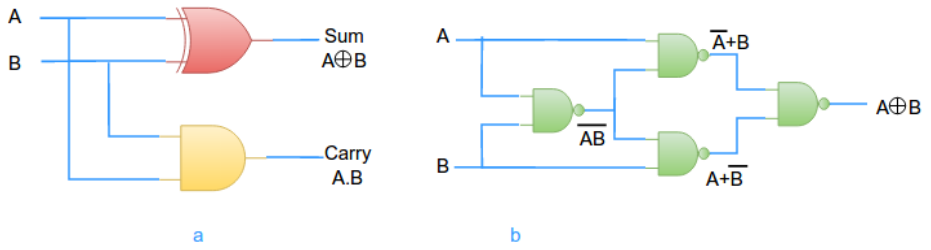
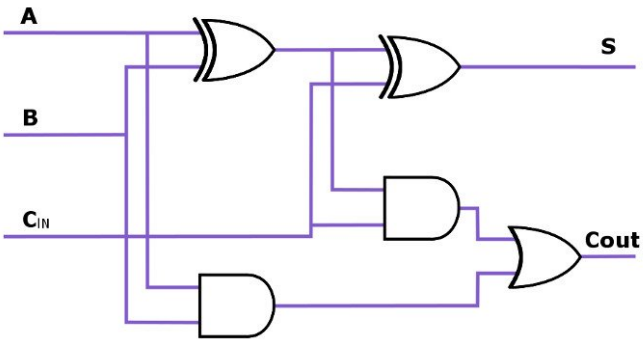
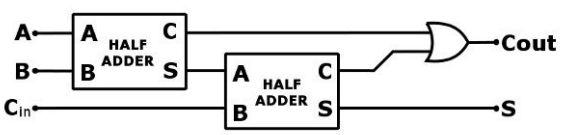


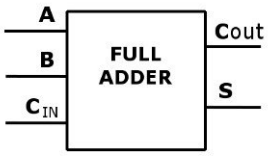
Figure-1 : a)Half Adder b)XOR implementation using NAND gates

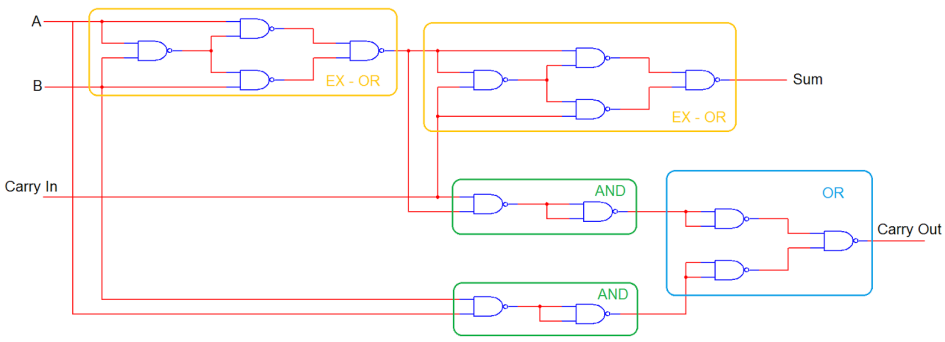
- **Full Adder**



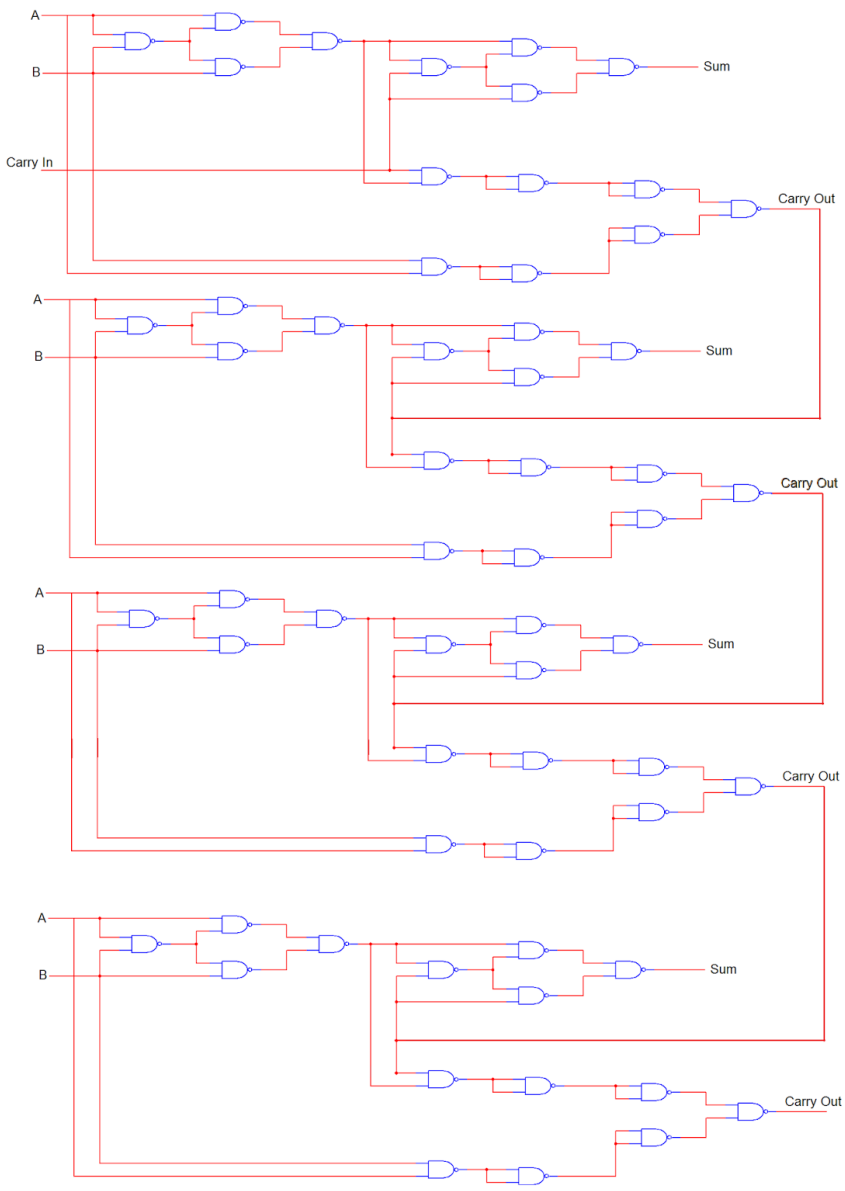
Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1







- **4 Bit Full Adder**



- **Substract**

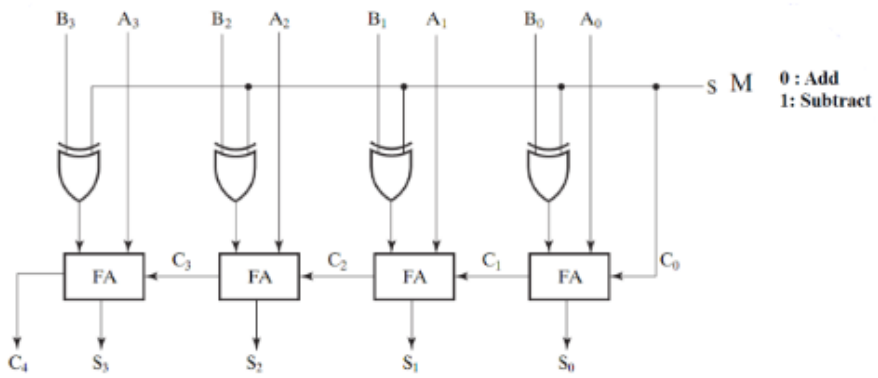
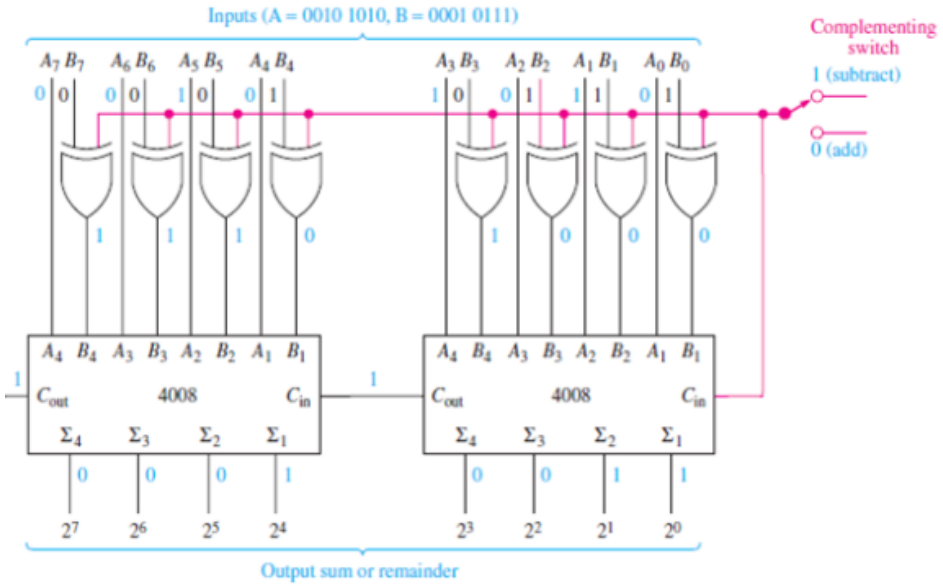
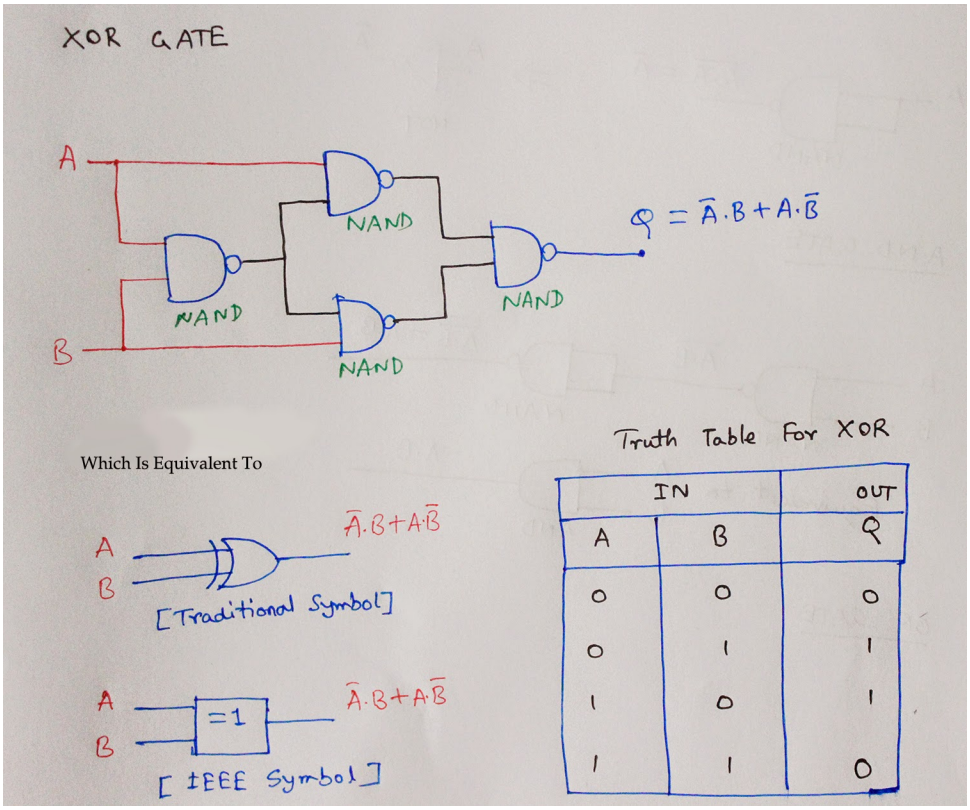


Fig. 1 Modular design of 4-bit adder/subtractor



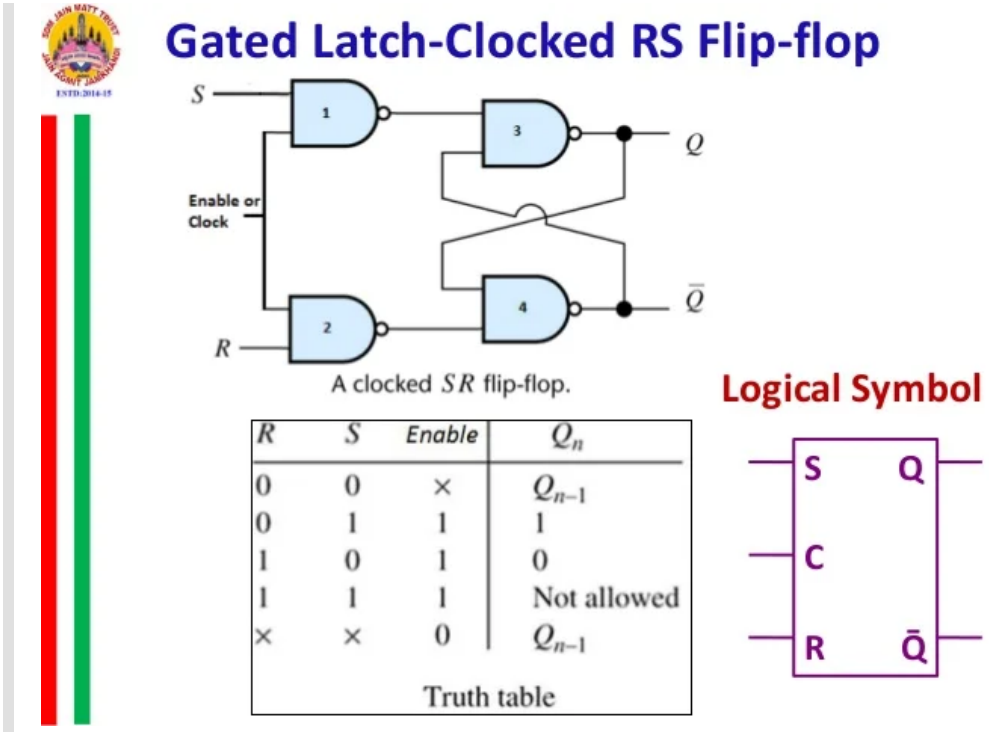
- **NAND Gate**

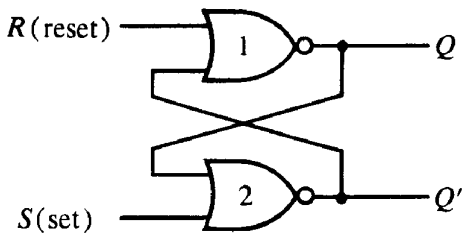
- **XOR and NAND**



Circuits as Memory

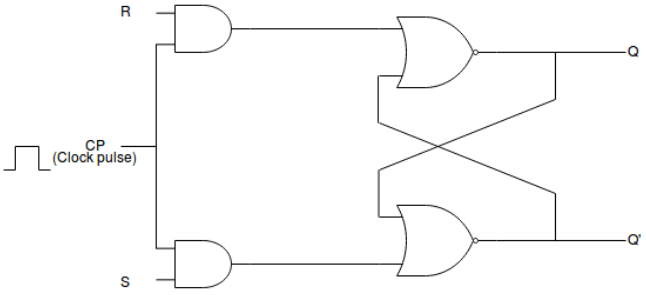
- **SR Filp Folp**





S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	

(a) Logic diagram (b) Truth table



a) Logic diagram

fig: Clocked SR flip flop

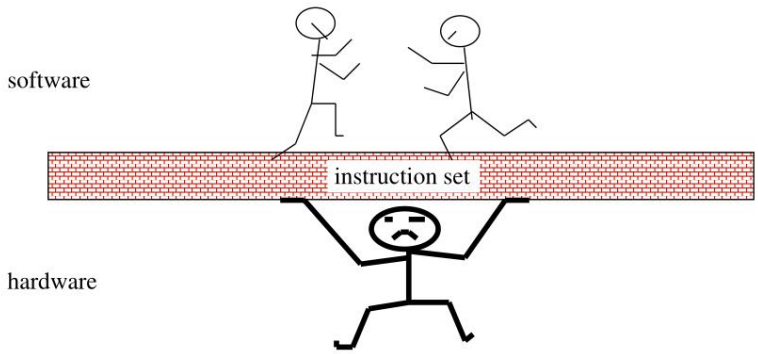
Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Intermediate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Intermediate

b) Truth table

Computer Orginization and Archetiture

Instruction Set Architecture - Interface of S/W and H/w

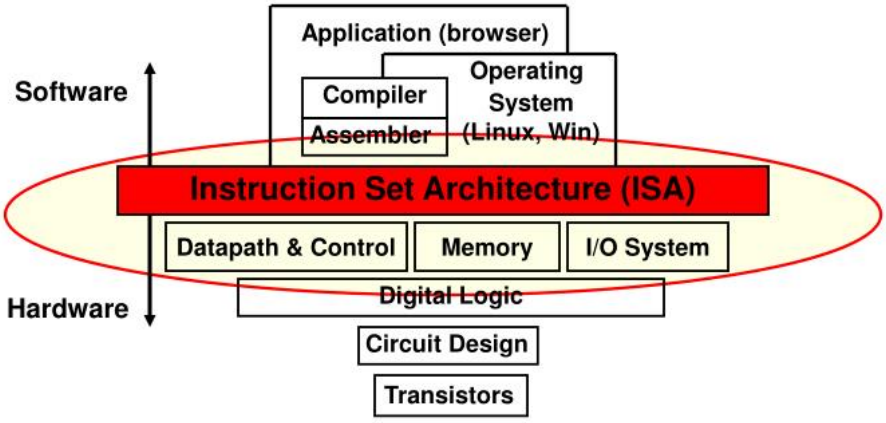
Instruction Set Architecture: Critical Interface



- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels

Computer Architecture- 8

The Instruction Set Architecture



ISA

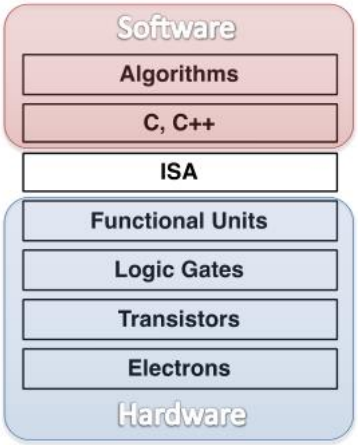
Instruction Set Architecture

- The computer ISA defines all the *programmer-visible* components and operations of the computer
  - Memory organization
    - address space -- how may locations can be addressed?
    - addressibility -- how many bits per location?
  - Register set
    - how many? what size? how are they used?
  - Instruction set
    - opcodes
    - data types
    - addressing modes
- ISA provides all information needed for someone that wants to write a program in machine language (or translate from a high-level language to machine language).

BYU CS 224 ISA 5

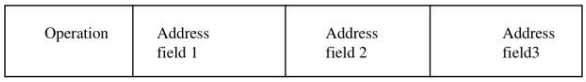
Instruction Set Architecture (ISA)

- ISA is the interface provided by the hardware to the software
  - Defines the available:
    - instructions
    - registers
    - addressing modes
    - memory architecture
    - interrupt and exception handling
    - external I/O
  - Syntax defined by assembly language
    - Symbolic representation of the machine instructions
  - Examples: x86, ARM, HCS12

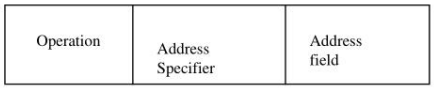


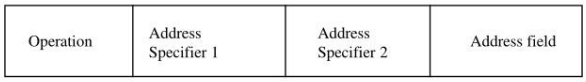
Three Examples of Instruction Set Encoding

Variable: VAX (1-53 bytes)



Fixed: DLX, MIPS, PowerPC, SPARC





Hybrid : IBM 360/370, Intel 80x86

EECC551 - Shaaban

#20 Lec # 2 Fall 2000 9-12-2000

CICS vs RICS

CISC vs. RISC

Complex Instruction Set Computer

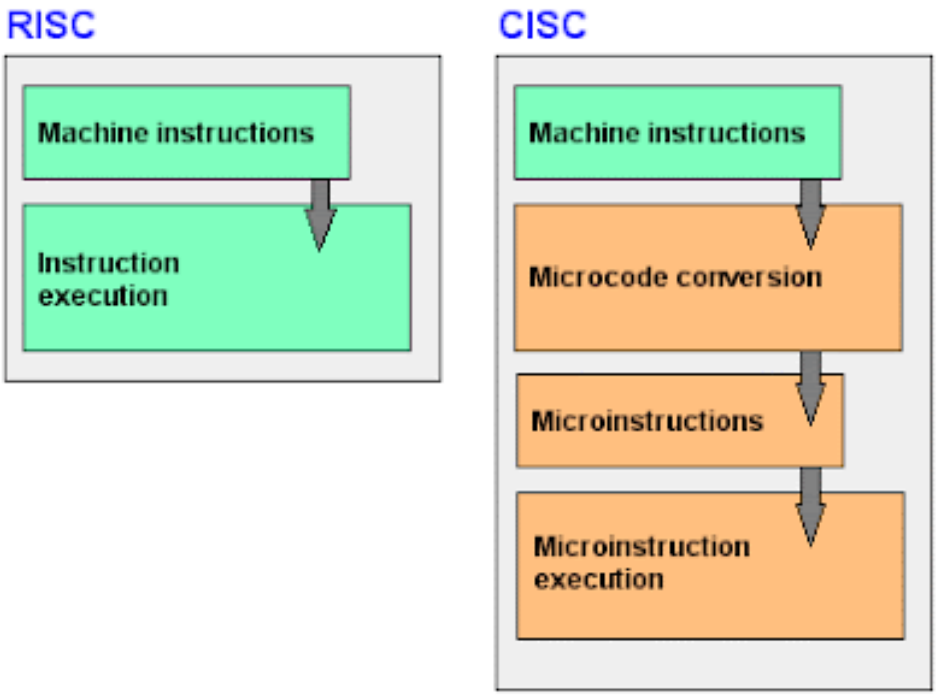
- Many instructions
  - e.g., 75-100
- Many instructions are macro-like
  - Simplifies programming
- Most microcontrollers are based on CISC concept
  - e.g., PDP-11, VAX, Motorola 68k
  - PIC is an exception

Reduced Instruction Set Computers

- Few instructions
  - e.g., 30-40
- Smaller chip, smaller pin count, & very low-power consumption
  - Simple but fast instructions
- Harvard architecture, instruction pipelining
- Industry trend for microprocessor design
  - e.g., Intel Pentium, PIC

24

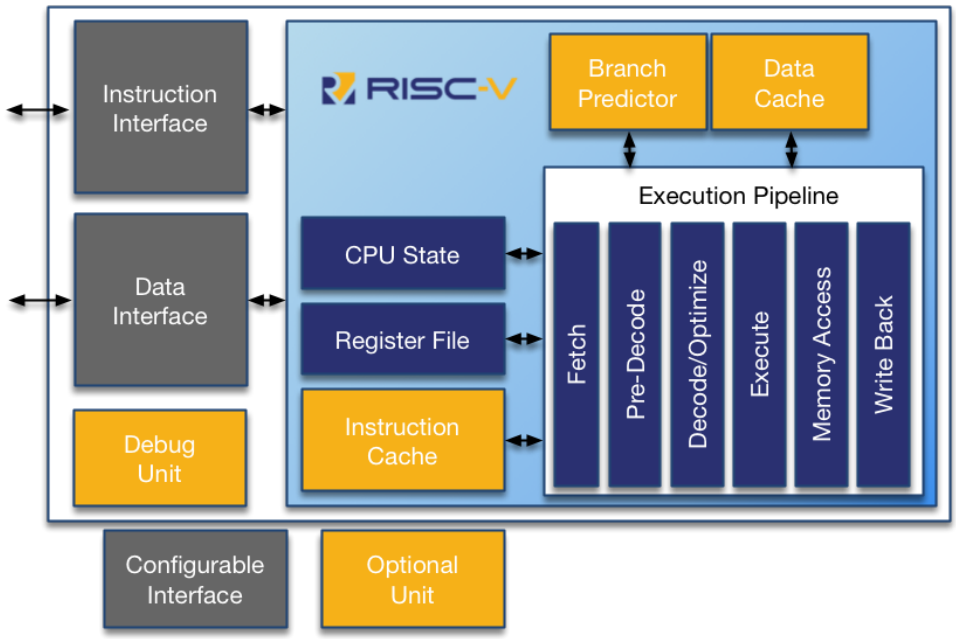
From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.



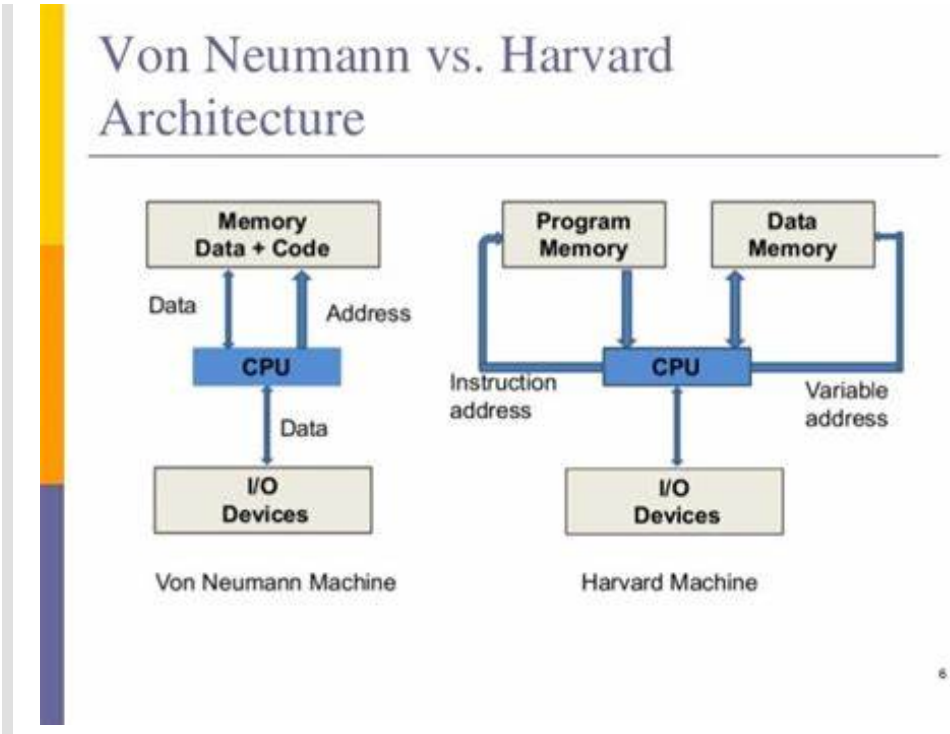
32-bit RISC-V Instruction Formats

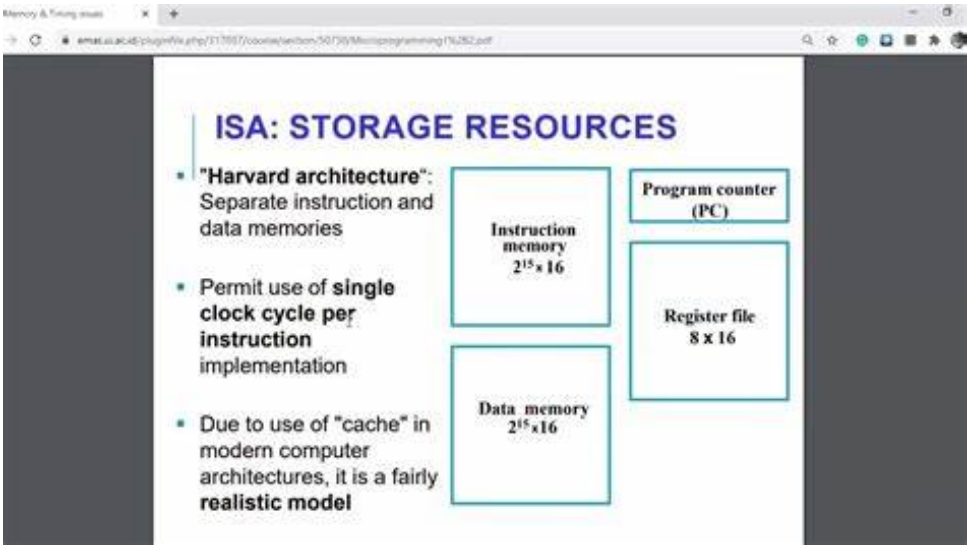
Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register/register	funct7							rs2					rs1					funct3			rd					opcode						
Immediate	imm[11:0]												rs1					funct3			rd					opcode						
Upper Immediate	imm[31:12]																				rd					opcode						
Store	imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						
Branch	[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				[11]	opcode						
Jump	[20]	imm[10:1]										[11]	imm[19:12]								rd					opcode						

- **opcode (7 bit):** partially specifies which of the 6 types of *instruction formats*
- **funct7 + funct3 (10 bit):** combined with **opcode**, these two fields describe what operation to perform
- **rs1 (5 bit):** specifies register containing first operand
- **rs2 (5 bit):** specifies second register operand
- **rd (5 bit):**: Destination register specifies register which will receive result of computation

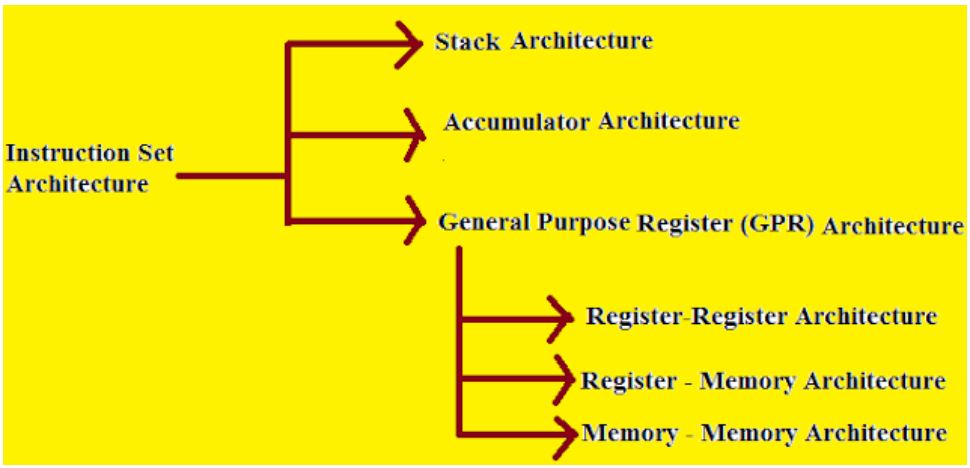


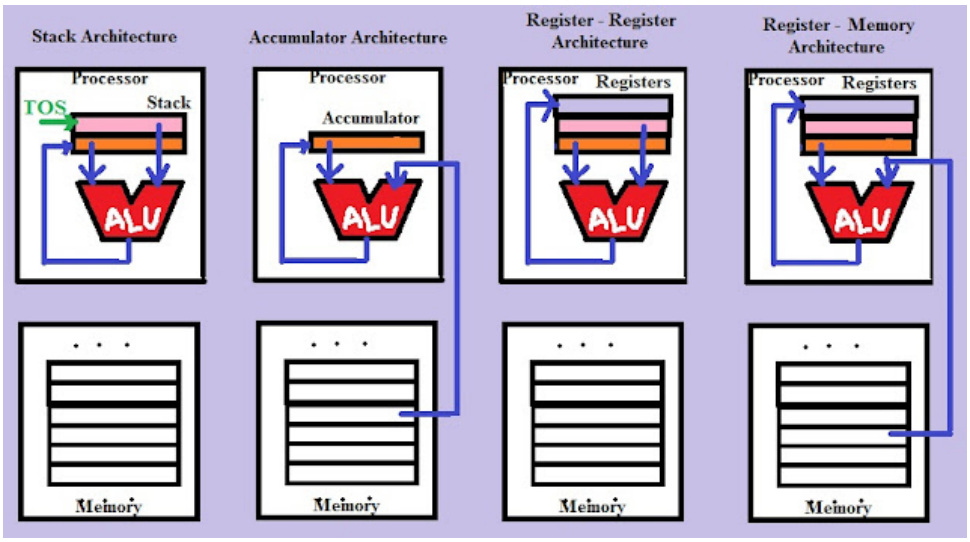
Memory Architecture





Classification of ISA (or) Types of ISA



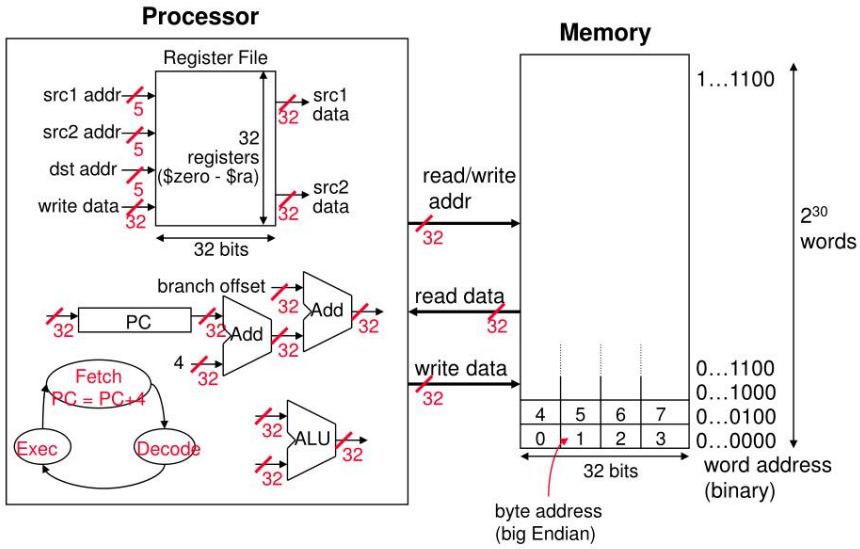


ISA - MIPS Instrucstion Foramt and Addressing Model

MIPS Design Paradigms

- Simplicity favors regularity
  - all instructions single size
  - three register operands in arithmetic instr.
  - keep register fields in the same place
- Smaller is faster
  - 32 registers
- Make good compromises
  - large addresses and constants versus unique instruction length
- Make the common case fast
  - PC-relative addressing for conditional branches

MIPS Organization So Far



34

The MIPS ISA – Register File

- When writing assembly, these registers can be referenced by their address (number) or name
- General purpose and special purpose registers

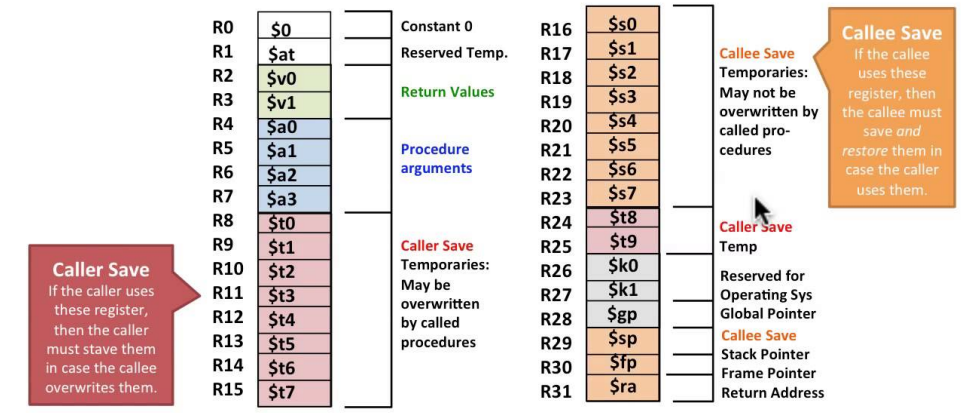
#	Name	Purpose
$0	$zero	Constant zero
$1	$at	Reserved for assembler
$2	$v0	Function return value
$3	$v1	
$4	$a0	Function parameter
$5	$a1	
$6	$a2	
$7	$a3	
$8	$t0	Temporary – Caller-saved
$9	$t1	
$10	$t2	
$11	$t3	
$12	$t4	
$13	$t5	
$14	$t6	
$15	$t7	

#	Name	Purpose
$16	$s0	Temporary – Callee-saved
$17	$s1	
$18	$s2	
$19	$s3	
$20	$s4	
$21	$s5	
$22	$s6	
$23	$s7	
$24	$t8	Temporary – Caller-saved
$25	$t9	
$26	$k0	Reserved for OS
$27	$k1	
$28	$gp	Global pointer
$29	$sp	Stack pointer
$30	$fp	Frame pointer
$31	$ra	Function return address

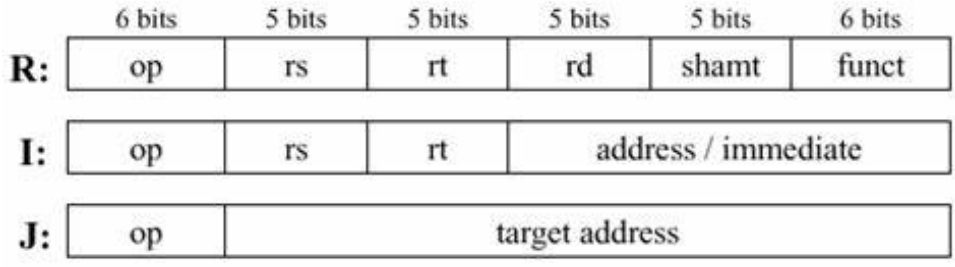
17

Who saves what?

53



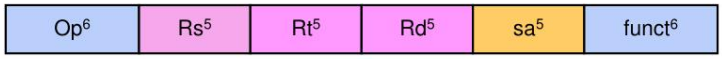
Instruction Format



op: basic operation of the instruction (opcode)
rs: first source operand register
rt: second source operand register
rd: destination operand register
shamt: shift amount
funct: selects the specific variant of the opcode (function code)
address: offset for load/store instructions (+/-2^{15})
immediate: constants for immediate instructions

Instruction Formats

- All instructions are 32-bit wide. Three instruction formats:
- Register (R-Type)
  - Register-to-register instructions
  - Op: operation code specifies the format of the instruction



- Immediate (I-Type)
  - 16-bit immediate constant is part in the instruction

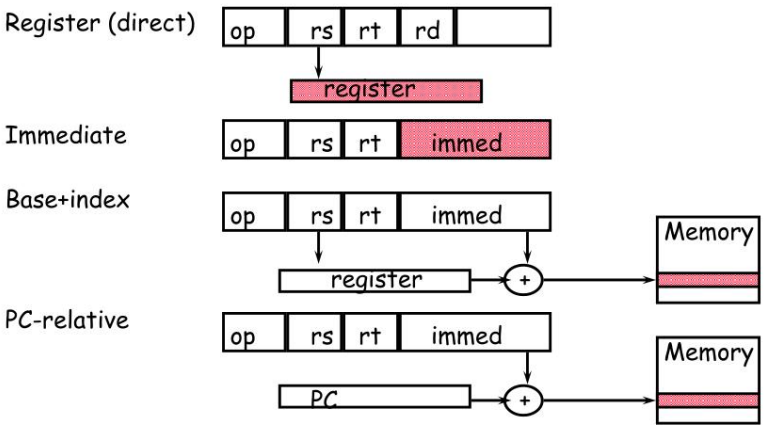
- Jump (J-Type)
  - Used by jump instructions

Instruction Set Architecture *ICS 233 – Computer Architecture and Assembly Language – KFUPM* *© Muhamed Mudawar* *slide 9*

Addressing Model

Example: MIPS Instruction Formats and Addressing Modes

- All instructions 32 bits wide



ECE 361 3-45

5.6 Addressing Modes

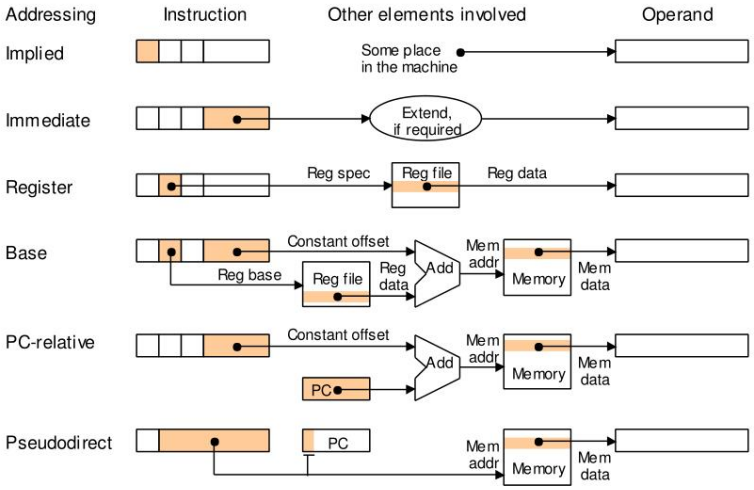


Figure 5.11 Schematic representation of addressing modes in MiniMIPS.

Computer Architecture, Instruction-Set Architecture Slide 22

Instruction Categories

MIPS: ISA

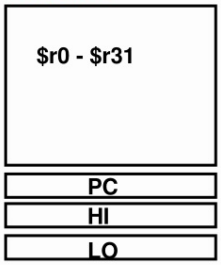
Category	Instruction	Op Code	Example	Meaning
Arithmetic	Add	0 and 32	add $s1, $s2, $s3	$s1 = $s2 + $s3
(R & I format)	Subtract	0 and 34	sub $s1, $s2, $s3	$s1 = $s2 - $s3
	add immediate	8	addi $s1, $s2, 6	$s1 = $s2 + 6
	or immediate	13	ori $s1, $s2, 6	$s1 = $s2 v 6
Logical (R & I format)	And	0 and 36	and $s1, $s2, $s3	$s1 = $s2 & $s3
	Or	0 and 37	or $s1, $s2, $s3	$s1 = $s2 | $s3
	Nor	0 and 39	nor $s1, $s2, $s3	$s1 = ~($s2 | $s3)
	And immediate	12	andi $s1, $s2, 100	$s1 = $s2 & 100
	Or immediate	13	ori $s1, $s2, 100	$s1 = $s2 | 100
	Shift left logical	0 and 0	sll $s1, $s2, 10	$s1 = $s2 << 10
	Shift right logical	0 and 2	srl $s1, $s2, 10	$s1 = $s2 >> 10
Data Transfer (I format)	load word	35	lw $s1, 24($s2)	$s1 = Memory[$s2+24]
	store word	43	sw $s1, 24($s2)	Memory[$s2+24] = $s1
	load byte	32	lb $s1, 25($s2)	$s1 = Memory[$s2+25]
	store byte	40	sb $s1, 25($s2)	Memory[$s2+25] = $s1
	load upper imm	15	lui $s1, 6	$s1 = 6 * 2^{16}
Cond. Branch (I & R format)	br on equal	4	beq $s1, $s2, L	if ($s1==$s2) go to L
	br on not equal	5	bne $s1, $s2, L	if ($s1 !=$s2) go to L
	set on less than	0 and 42	slt $s1, $s2, $s3	if ($s2<$s3) $s1=1 else $s1=0
	set on less than immediate	10	slti $s1, $s2, 6	if ($s2<6) $s1=1 else $s1=0
Uncond. Jump (J & R format)	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr $t1	go to $t1
	jump and link	3	jal 2500	go to 10000; $ra=PC+4

40

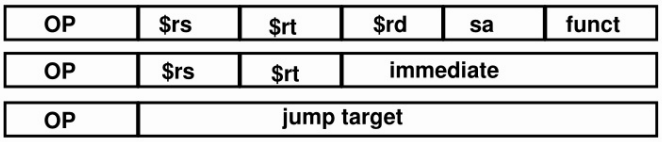
MIPS ISA as an Example

- Instruction categories:
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
  - Memory Management
  - Special

Registers



3 Instruction Formats: all 32 bits wide



National Tsing Hua University
COMPUTER ARCHITECTURE 6

ISA - Performance

Compiler Variations, MIPS, Performance: An Example (Continued)

MIPS = Clock rate / (CPI x 10^6) = 100 MHz / (CPI x 10^6)

CPI = CPU execution cycles / Instructions count

$$CPU\ clock\ cycles = \sum_{i=1}^{n} (CPI_i \times C_i)$$

CPU time = Instruction count x CPI / Clock rate

- For compiler 1:
  - $CPI_1 = (5 \times 1 + 1 \times 2 + 1 \times 3) / (5 + 1 + 1) = 10 / 7 = 1.43$
  - $MIP_1 = 100 / (1.428 \times 10^6) = 70.0$
  - $CPU\ time_1 = ((5 + 1 + 1) \times 10^6 \times 1.43) / (100 \times 10^6) = 0.10$ seconds
- For compiler 2:
  - $CPI_2 = (10 \times 1 + 1 \times 2 + 1 \times 3) / (10 + 1 + 1) = 15 / 12 = 1.25$
  - $MIP_2 = 100 / (1.25 \times 10^6) = 80.0$
  - $CPU\ time_2 = ((10 + 1 + 1) \times 10^6 \times 1.25) / (100 \times 10^6) = 0.15$ seconds

EECC551 - Shaaban
#47 Lec # 1 Winter 2003 12-1-2003

Speed Up Equation for Pipelining

$$CPI_{pipelined} = \text{Ideal CPI} + \text{Average Stall cycles per Inst}$$

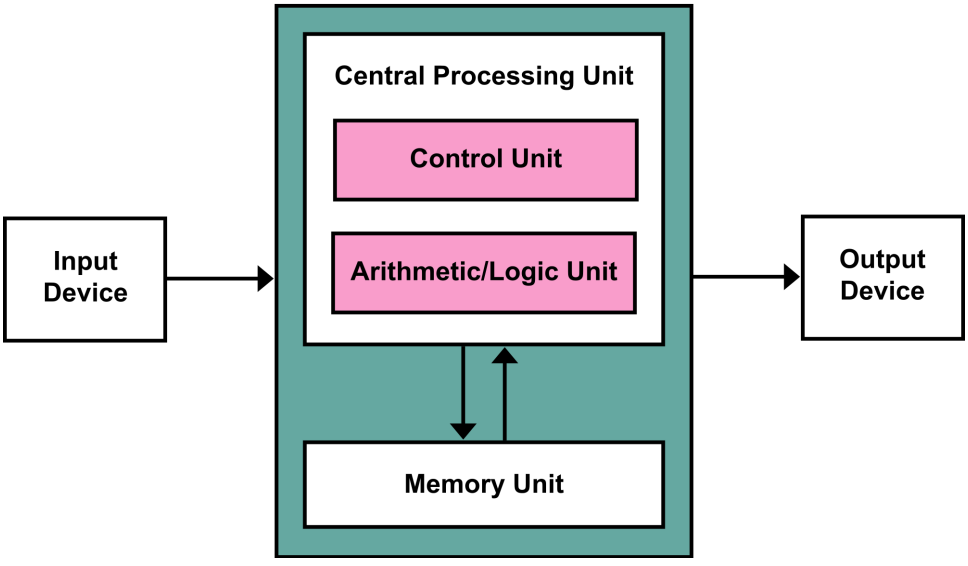
$$\text{Speedup} = \frac{\text{Ideal CPI} \times \text{Pipeline depth}}{\text{Ideal CPI} + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{unpipelined}}{\text{Cycle Time}_{pipelined}}$$

For simple RISC pipeline, CPI = 1:

$$\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{unpipelined}}{\text{Cycle Time}_{pipelined}}$$

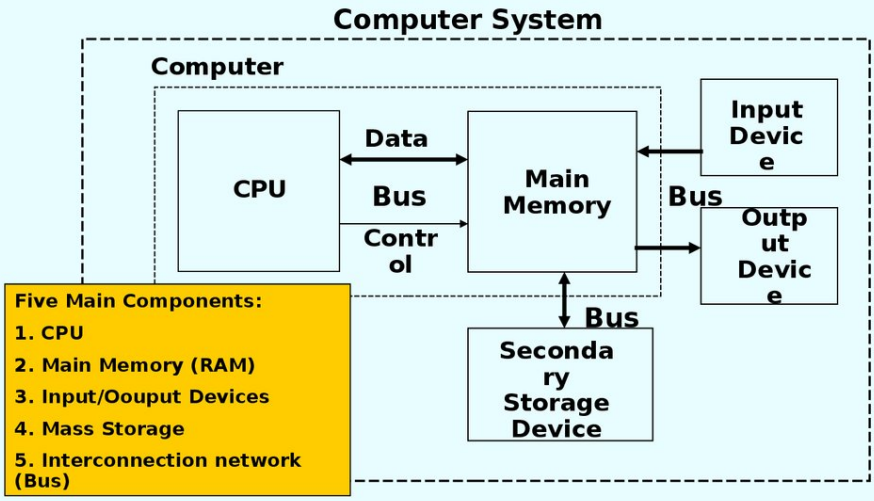
CS211 41

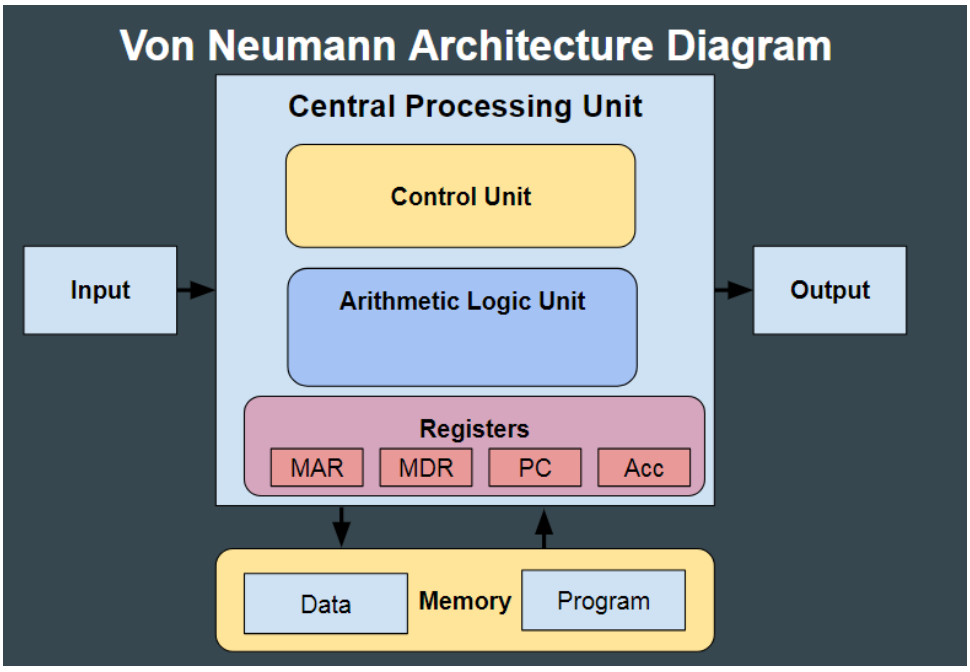
Processor - Von Neumann Architecture

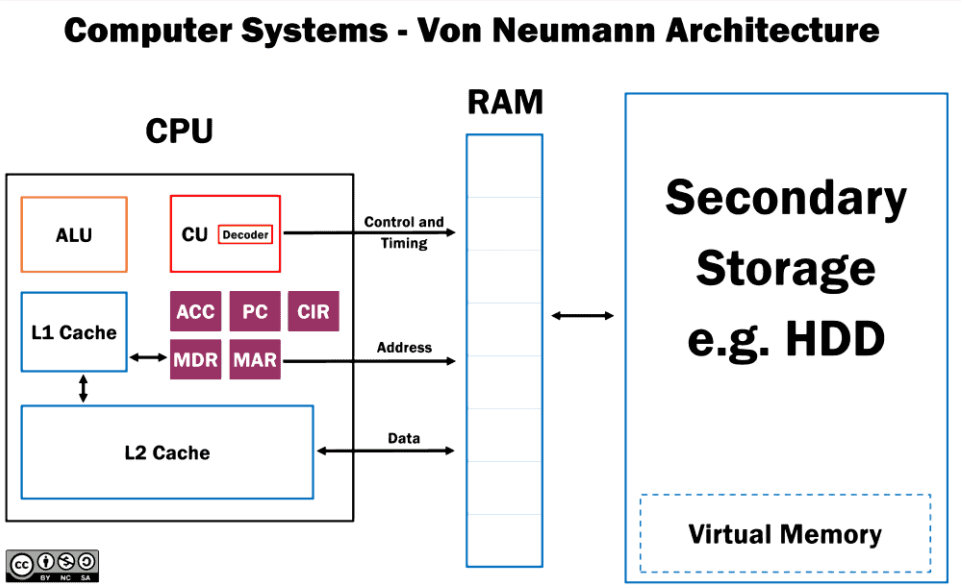


von Neumann Architecture

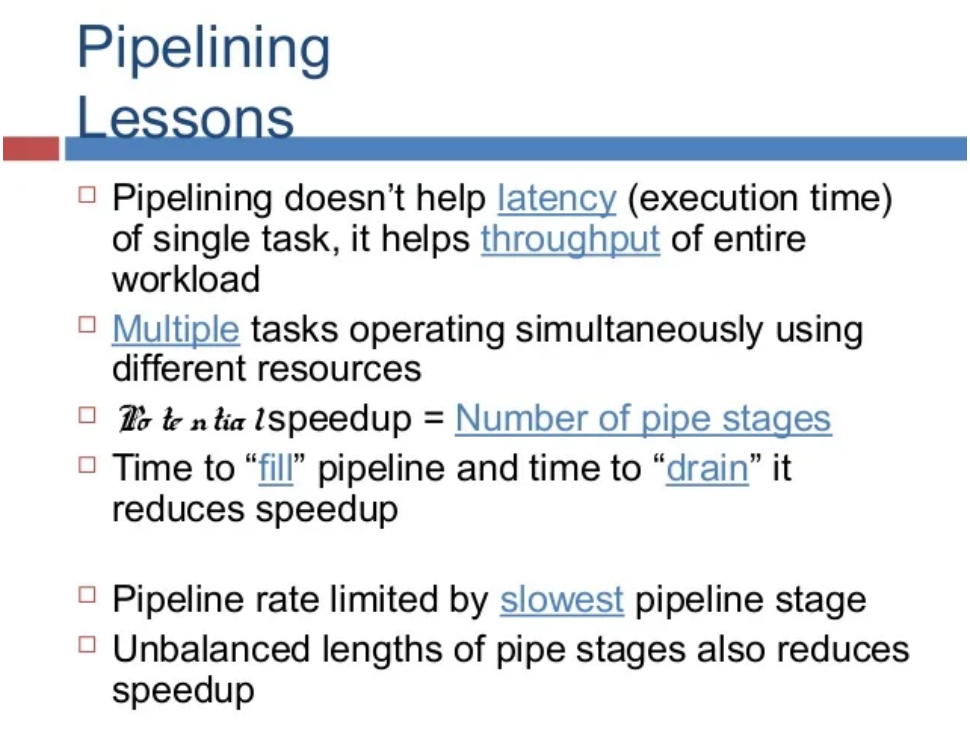
- A more complete view of the computer *system* architecture that integrates interaction (human or otherwise) consists of:







Processor - Pipeline



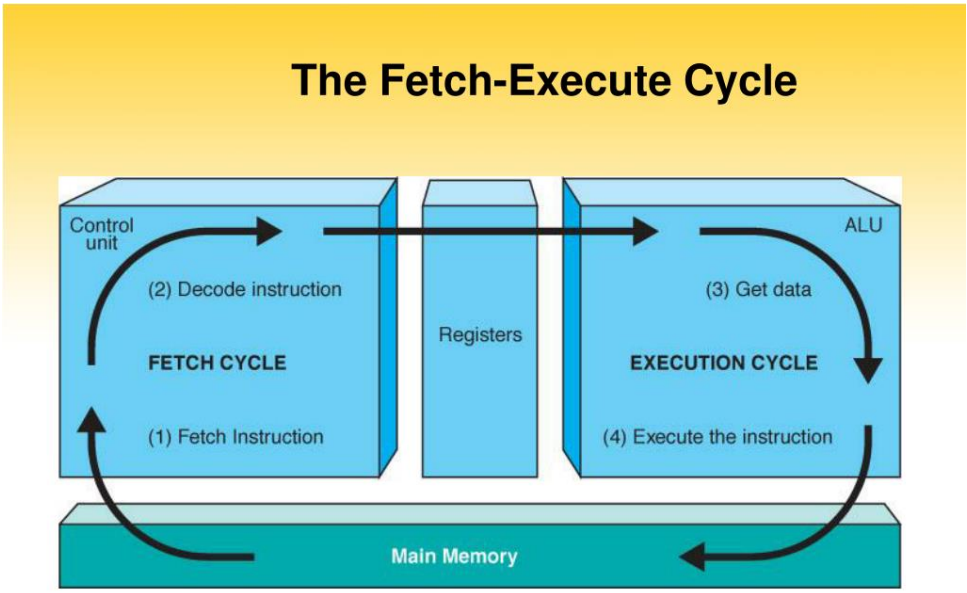
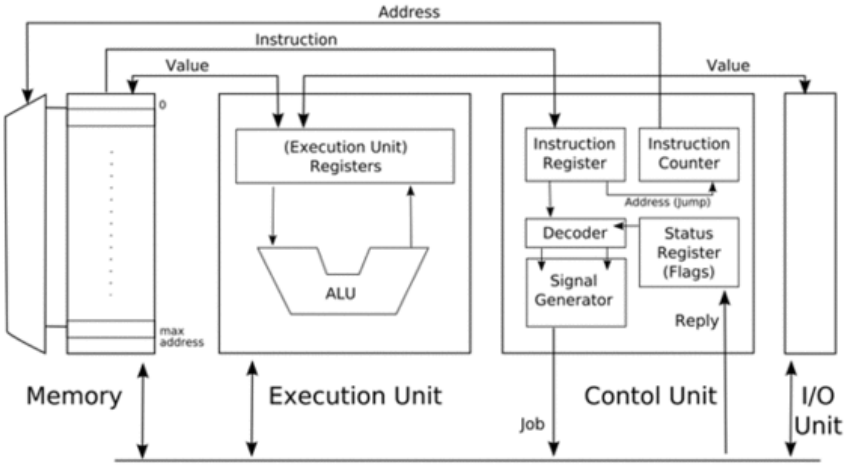
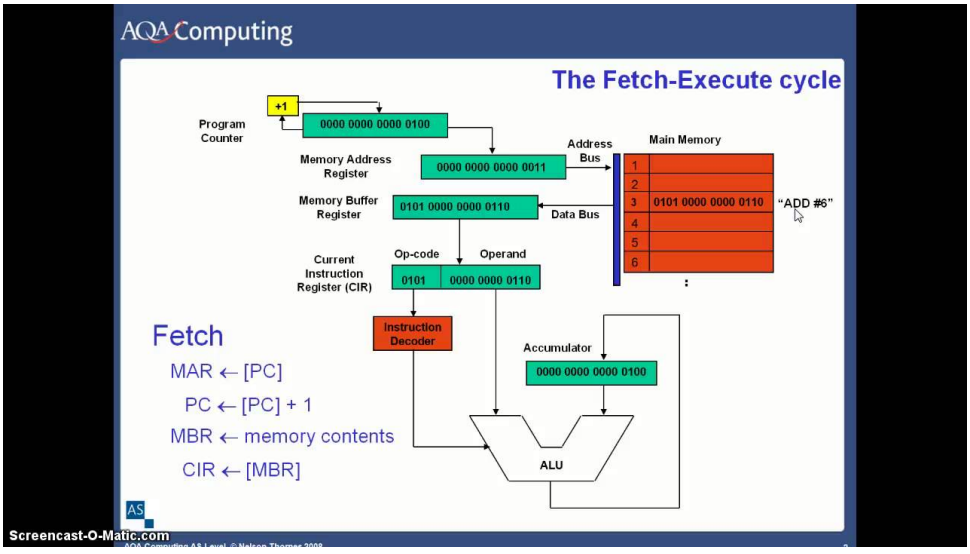
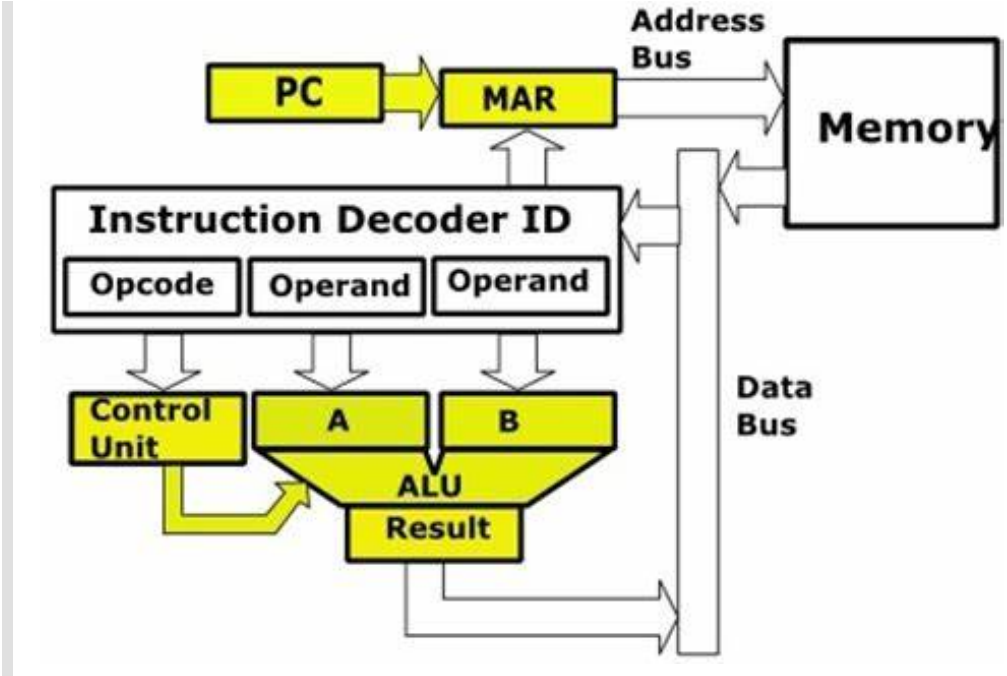


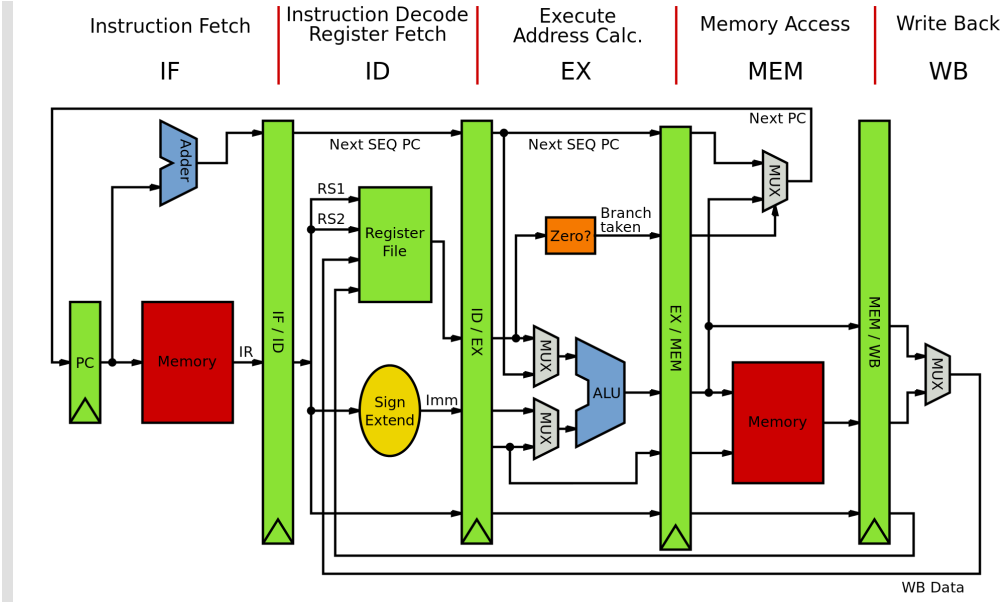
Figure 5.3 The Fetch-Execute Cycle

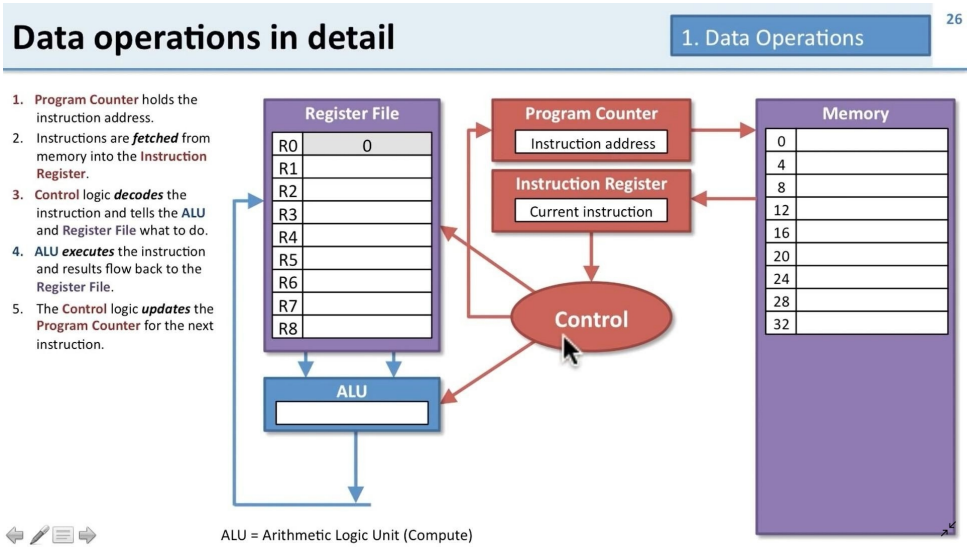
17

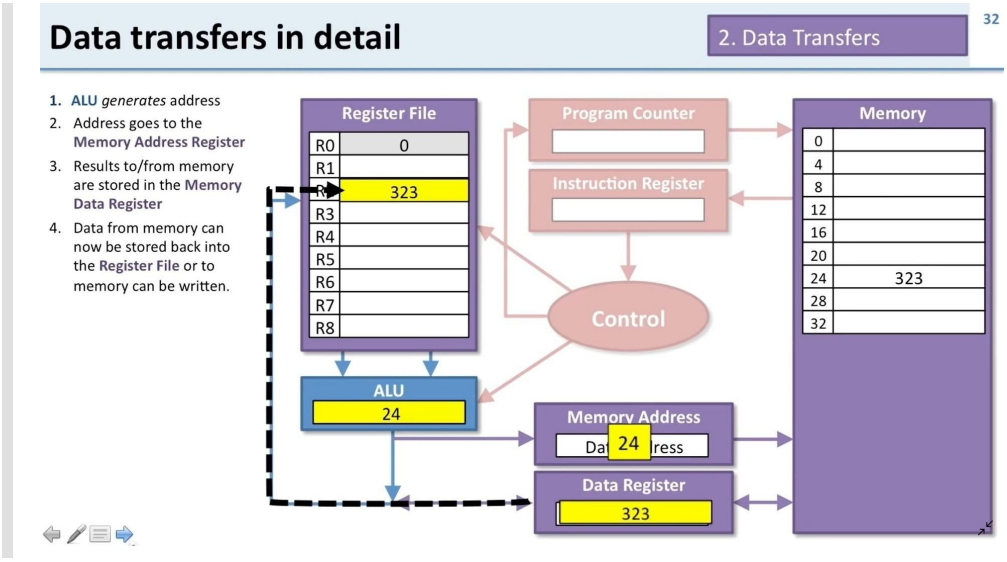


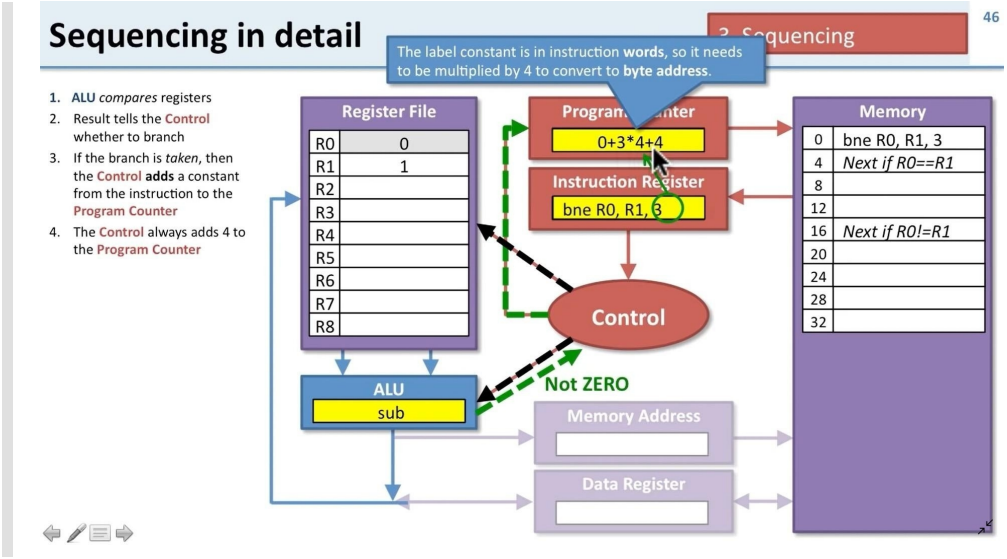



Address Bus
PC
MAR
Memory
Instruction Decoder ID
Opcode
Operand
Operand
Control Unit
A
B
ALU
Result
Data Bus

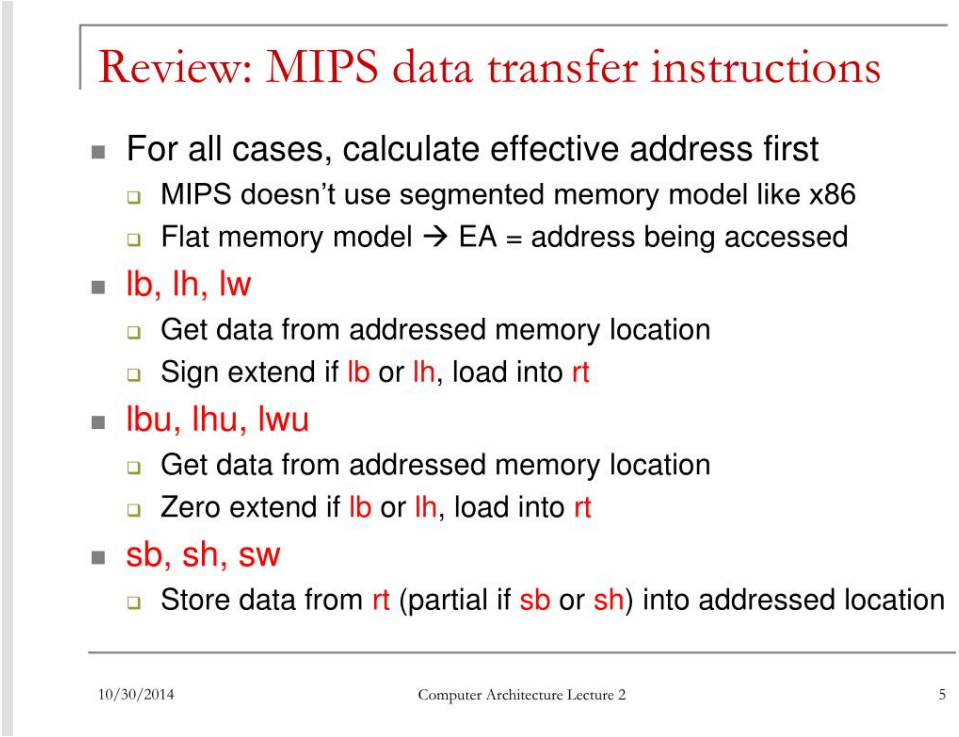

Instruction Fetch
IF
Instruction Decode Register Fetch
ID
Execute Address Calc.
EX
Memory Access
MEM
Write Back
WB
Adder
Next SEQ PC
Next SEQ PC
Next PC
RS1
RS2
Register File
Zero?
Branch taken
MUX
PC
Memory
IR
IF / ID
ID / EX
EX / MEM
MEM / WB
Sign Extend
Imm
ALU
Memory
WB Data


Data operations in detail
1. Data Operations
26
1. Program Counter holds the instruction address.
2. Instructions are fetched from memory into the Instruction Register.
3. Control logic decodes the instruction and tells the ALU and Register File what to do.
4. ALU executes the instruction and results flow back to the Register File.
5. The Control logic updates the Program Counter for the next instruction.
Register File
R0 0
R1
R2
R3
R4
R5
R6
R7
R8
Program Counter
Instruction address
Instruction Register
Current instruction
Control
ALU
Memory
0
4
8
12
16
20
24
28
32
ALU = Arithmetic Logic Unit (Compute)





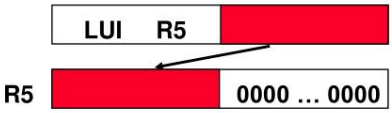
Processer - Datapath



MIPS Data Transfer Instructions

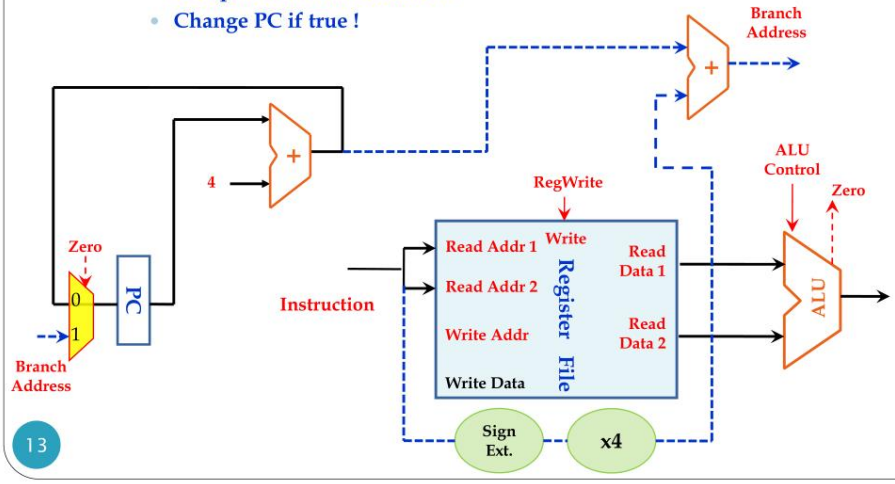
Instruction	Comment
SW R3, 500(R4)	Store word
SH R3, 502(R2)	Store half
SB R2, 41(R3)	Store byte
LW R1, 30(R2)	Load word
LH R1, 40(R3)	Load half word
LHU R1, 40(R3)	Load half word unsigned
LB R1, 40(R3)	Load byte
LBU R1, 40(R3)	Load byte unsigned
LUI R1, 40	Load Upper Immediate (16 bits shifted left by 16)

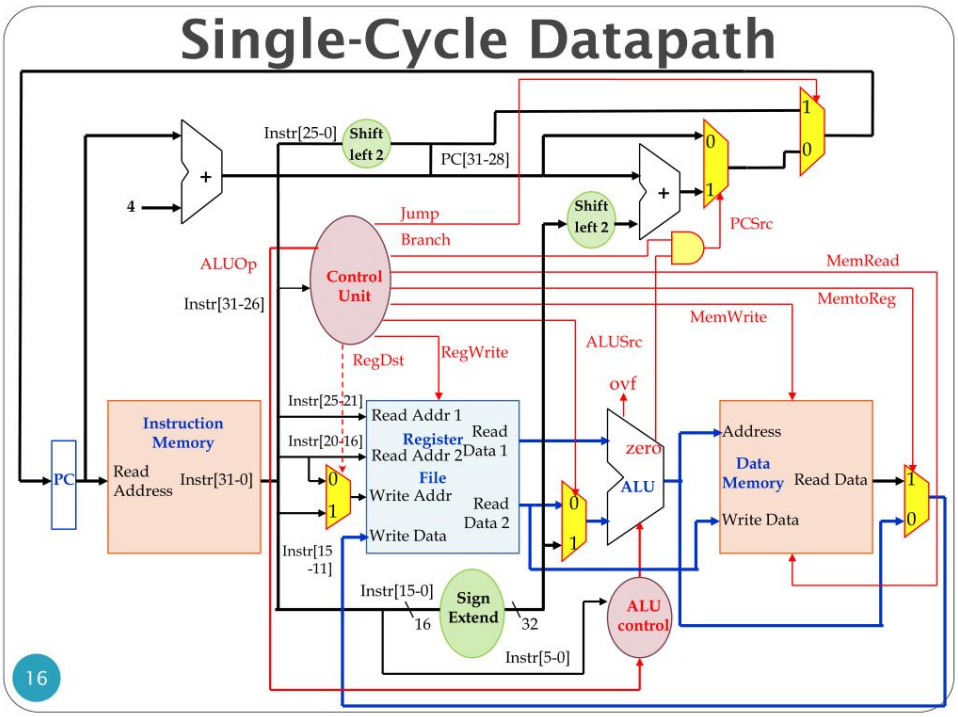
Why do we need LUI?



Single-Cycle Datapath

- Execution Datapath
  - Branch Instruction
    - Compare the two registers
    - Compute the branch address
    - Change PC if true !





Pipeline Hazard

Pipeline Hazards (1)

- ***Pipeline Hazards*** are situations that prevent the next instruction in the instruction stream from executing in its designated clock cycle
- Hazards reduce the performance from the ideal speedup gained by pipelining
- Three types of hazards
  - ***Structural hazards***
    - Arise from resource conflicts when the hardware can't support all possible combinations of overlapping instructions
  - ***Data hazards***
    - Arise when an instruction depends on the results of a previous instruction in a way that is exposed by overlapping of instruction in pipeline
  - ***Control hazards***
    - Arise from the pipelining of branches and other instructions that change the PC (Program Counter)

Pipeline Hazards (2)

- Hazards in pipeline can make the pipeline to ***stall***
- Eliminating a hazard often requires that some instructions in the pipeline to be allowed to proceed while others are delayed
  - When an instruction is stalled, instructions issued *latter* than the stalled instruction are stopped, while the ones issued *earlier* must continue
- No new instructions are fetched during the stall

Summary - Control Hazard Solutions

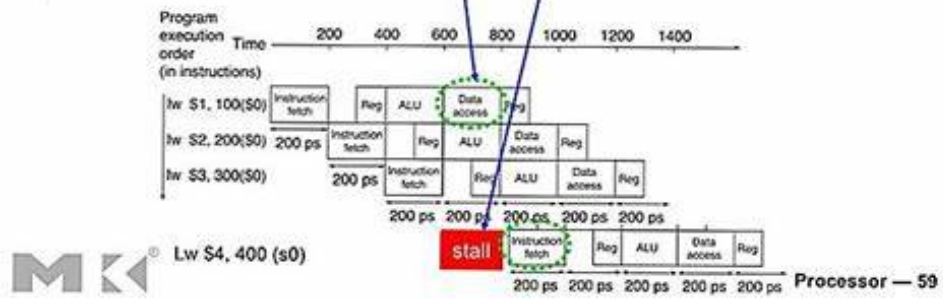
- **Stall** - stop fetching instr. until result is available
  - Significant performance penalty
  - Hardware required to stall
- **Predict** - assume an outcome and continue fetching (undo if prediction is wrong)
  - Performance penalty only when guess wrong
  - Hardware required to "squash" instructions
- **Delayed branch** - specify in architecture that following instruction is always executed
  - Compiler re-orders instructions into delay slot
  - Insert "NOP" (no-op) operations when can't use (~50%)
  - This is how original MIPS worked

CSCE430/830 Pipeline Hazards

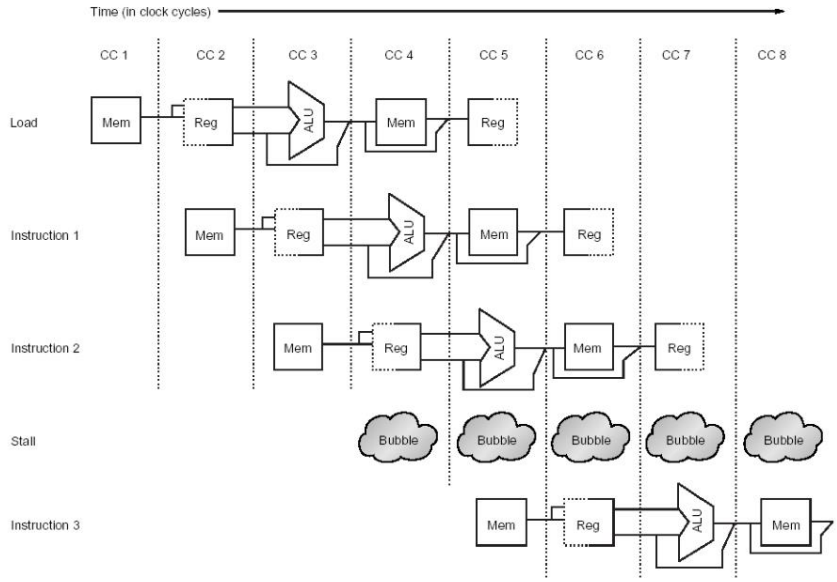
Solution for Pipeline Hazards

Structure Hazards

- Conflict for use of a resource
- Suppose that we has only a single memory instead of two memories (instruction and data) In the MIPS design
  - Load/store requires data access
  - Instruction fetch would have to stall for that cycle
    - Would cause a pipeline "bubble"
- Hence, pipelined datapaths require separate instruction/data memories



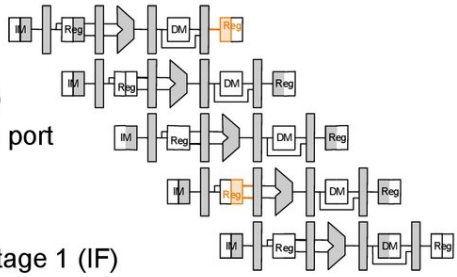
Structural Hazards (3)



- Stall cycle added (commonly called pipeline *bubble*)

Structural Hazard

- **Different instructions using the same resource at the same time**
- **Register File**:
  - Accessed in 2 stages:
    - Read during stage 2 (ID)
    - Write during stage 5 (WB)
  - Solution: 2 read ports, 1 write port
- **Memory**
  - Accessed in 2 stages:
    - Instruction Fetch during stage 1 (IF)
    - Data read/write during stage 4 (MEM)
  - Solution: separate instruction cache and data cache
- **Each functional unit can only be used once per instruction**
- **Each functional unit must be used at the same stage for all instructions**



Consider the following set of instructions in a 5-stage pipeline.
Operands are read in ID.
MEM is memory Write for result; RW is Register Write for result

R3 is accessed in READ mode; expect the result of ADD to be available in R3

ADD R3, R6, R5	- Results to be written in R3
SUB R4, R3, R5	- R3 has one of the operand
OR R6, R3, R7	- R3 has one of the operand
AND R8, R3, R7	- R3 has one of the operand
XOR R12, R3, R10	- R3 has one of the operand

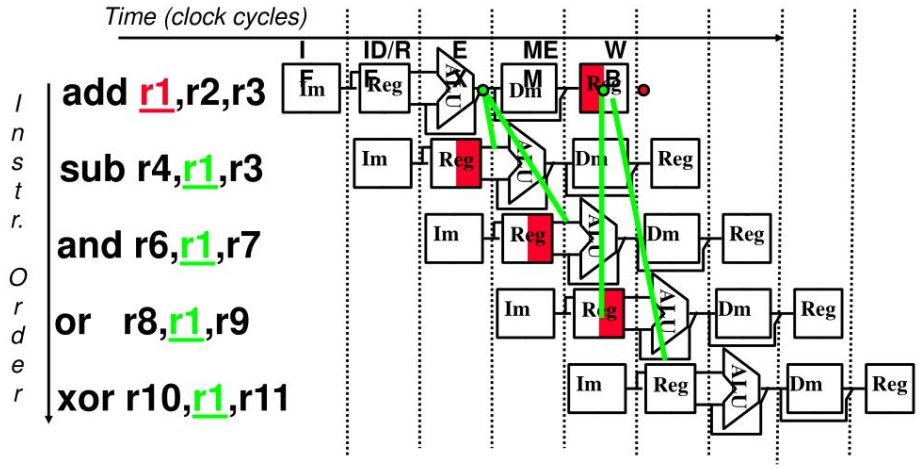
But result of ADD written in R3 at t5

	t1	t2	t3	t4	t5	t6	t7	t8	t9
ADD R3, R6, R5	IF	ID	IE	MEM	RW R3	--	--	--	--
SUB R4, R3, R5	--	IF	ID R3	IE	MEM	RW	--	--	--
OR R6, R3, R7	--	--	IF	ID R3	IE	MEM	RW	--	--
AND R8, R3, R9	--	--	--	IF	ID R3	IE	MEM	RW	--
XOR R10, R3, R11	--	--	--	--	IF	ID R3	IE	MEM	RW

Note when each instruction is accessing R3

Data Hazard Solution:

- **"Forward" result from one stage to another**

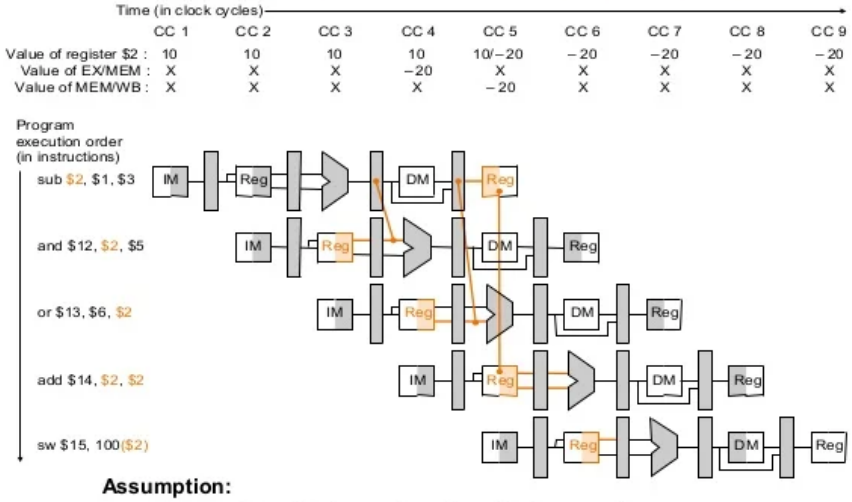


- **"or" OK if define read/write properly**

cs 152 L1 3 .13 DAP Fa97, © U.CB

Data Hazard Solution: Forwarding

- **Key idea: connect data internally before it's stored**



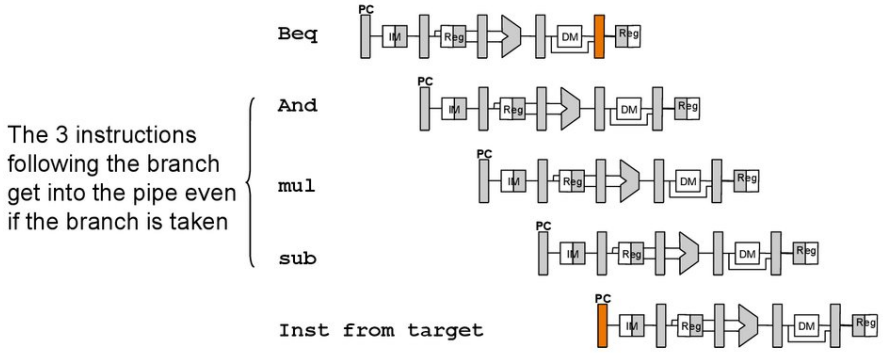
Assumption:

- **The register file forwards values that are read and written during the same cycle.**

Control Hazard

- Also known as *branch hazard*
- Pipeline makes wrong decision on branch prediction
- Brings instructions into pipeline that must subsequently be discarded
- Dealing with Branches
  - Multiple Streams
  - Prefetch Branch Target
  - Loop buffer
  - Branch prediction
  - Delayed branching

Control Hazard on Branches



Control Hazard Review

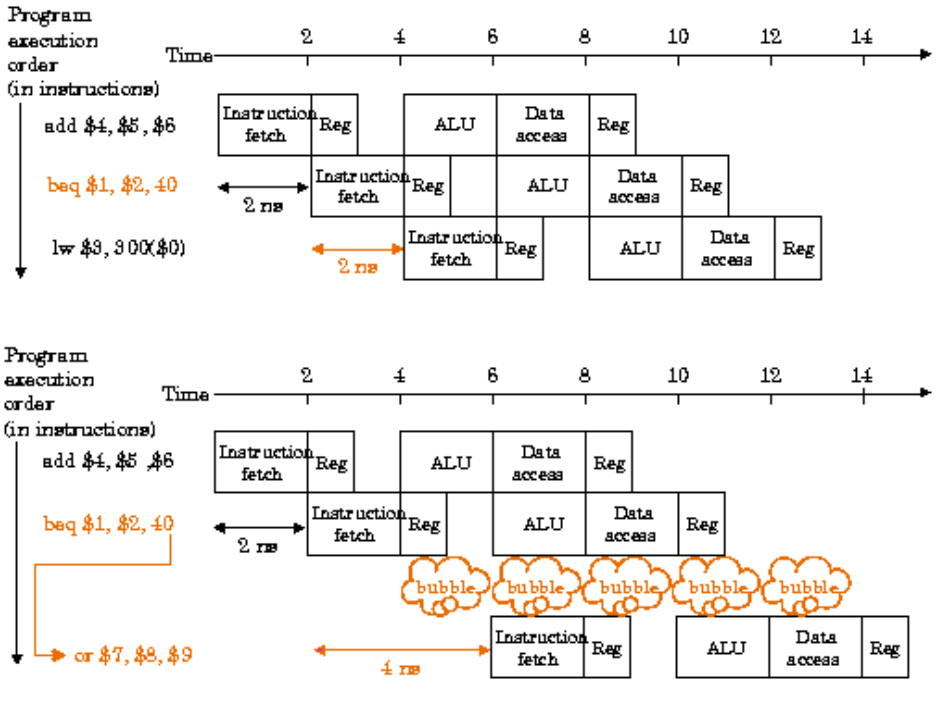
The nub of the problem:

- In what pipeline stage does the processor fetch the next instruction?
- If that instruction is a conditional branch, when does the processor know whether the conditional branch is taken (execute code at the target address) or not taken (execute the sequential code)?
- What is the difference in cycles between them?

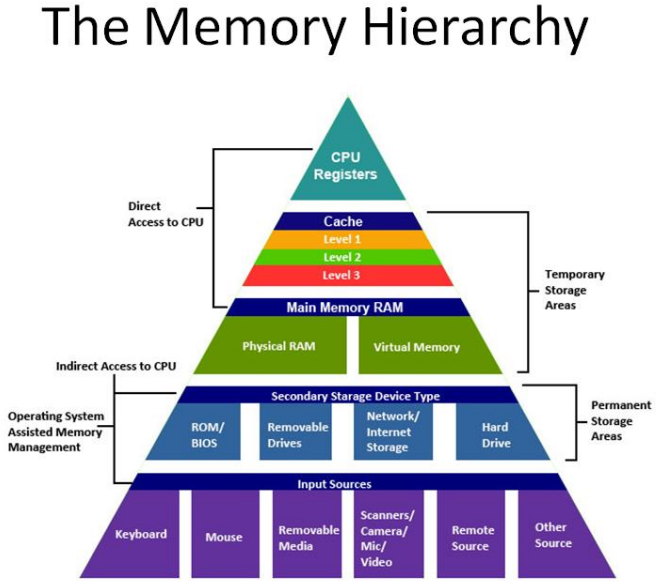
The cost of stalling until you know whether to branch

- number of cycles in between * branch frequency = the contribution to CPI due to branches

Predict the branch outcome to avoid stalling



Mermory Hierarchy

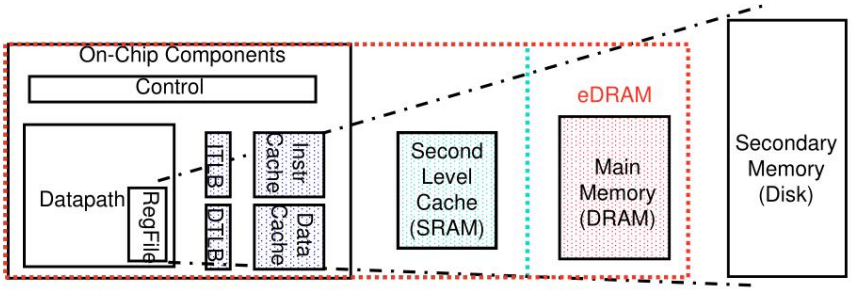


http://cse1.net/recaps/4-memory.html

2

A Typical Memory Hierarchy

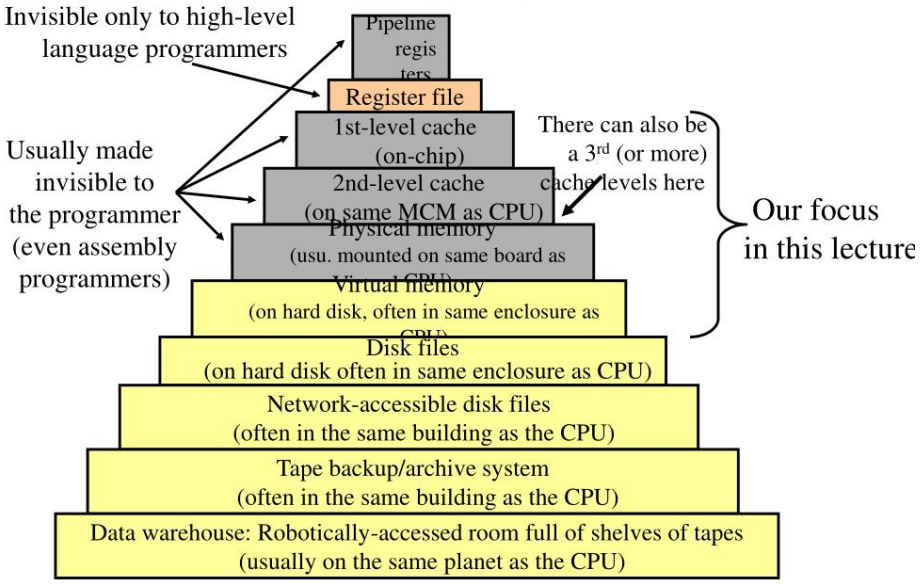
- By taking advantage of the principle of locality
  - Can present the user with as much memory as is available in the cheapest technology
  - at the speed offered by the fastest technology



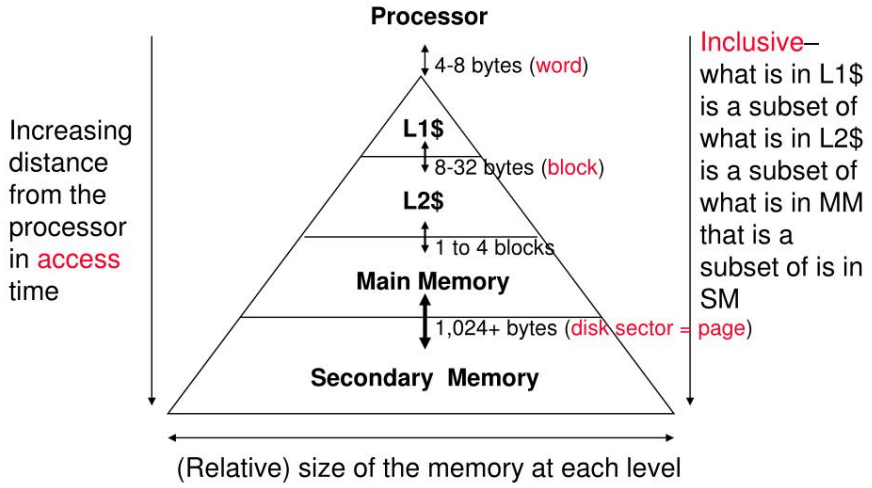
Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

CEG3420 L13.9 Qiang Xu CUHK, Spring 2011

Many Levels in Memory Hierarchy



Characteristics of the Memory Hierarchy

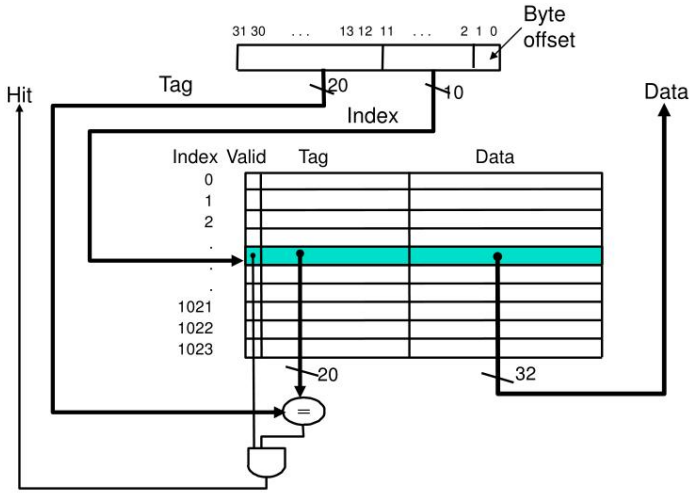


CPE432 Chapter 5A.7 Dr. W. Abu-Sufah, UJ

Block Size / Hit Rate / Miss Rate

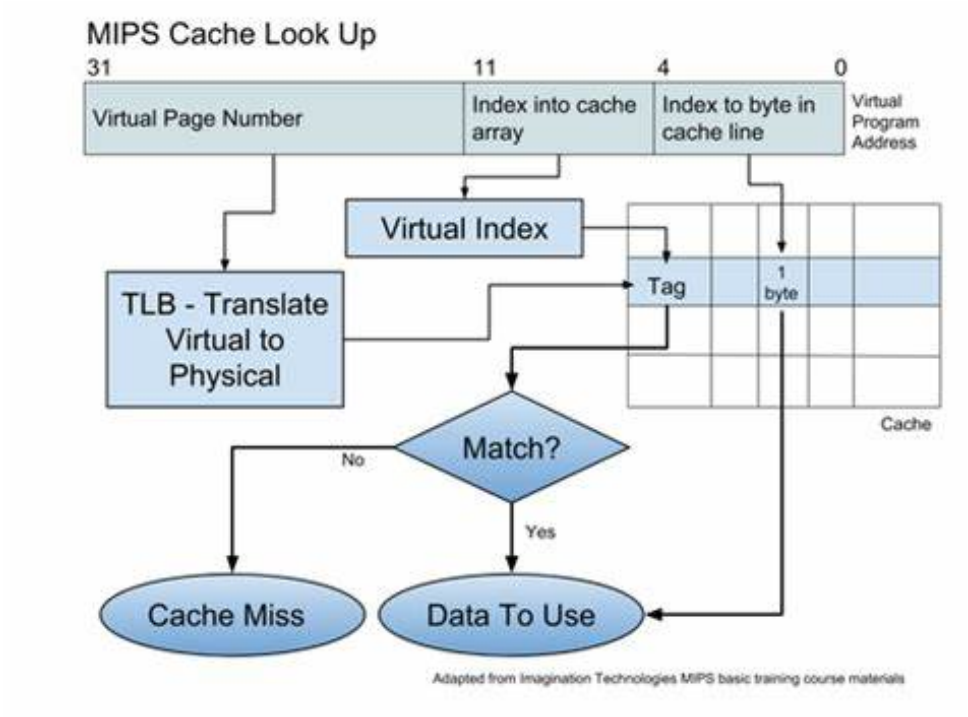
MIPS Direct Mapped Cache Example

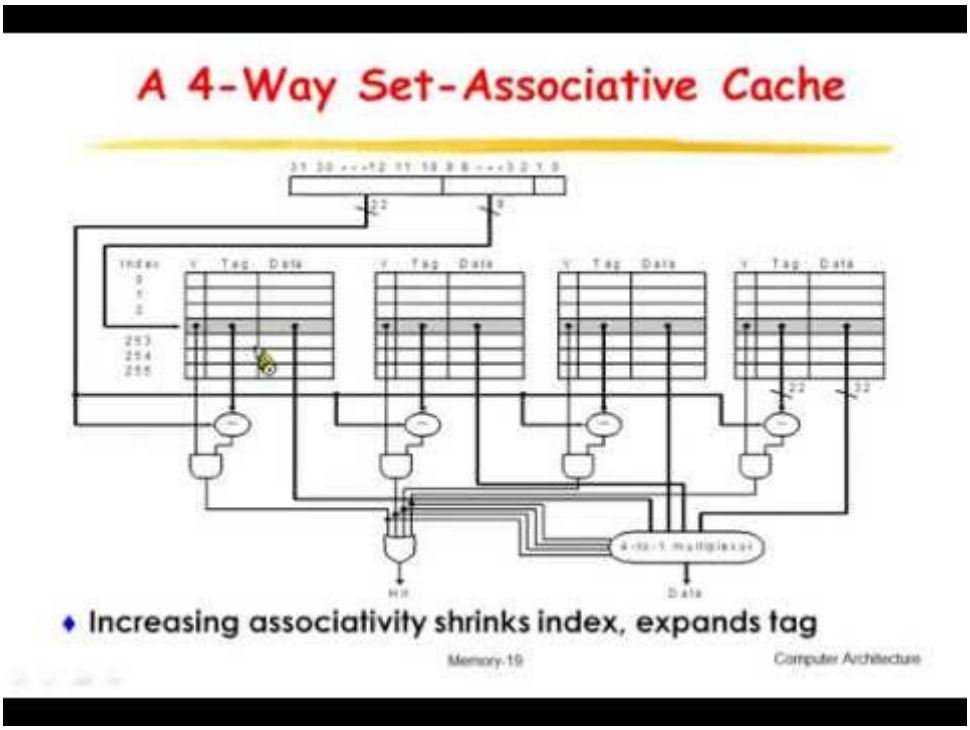
- One word/block, cache size = 1K words



What kind of locality are we taking advantage of?

CEG3420 L13.34 Qiang Xu CUHK, Spring 2011


MIPS Cache Look Up
31
11
4
0
Virtual Page Number
Index into cache array
Index to byte in cache line
Virtual Program Address
Virtual Index
TLB - Translate Virtual to Physical
Tag
1 byte
Cache
Match?
No
Yes
Cache Miss
Data To Use
Adapted from Imagination Technologies MIPS basic training course materials


A 4-Way Set-Associative Cache
Increasing associativity shrinks index, expands tag
Memory-19
Computer Architecture

Miss Rate vs. Block Size

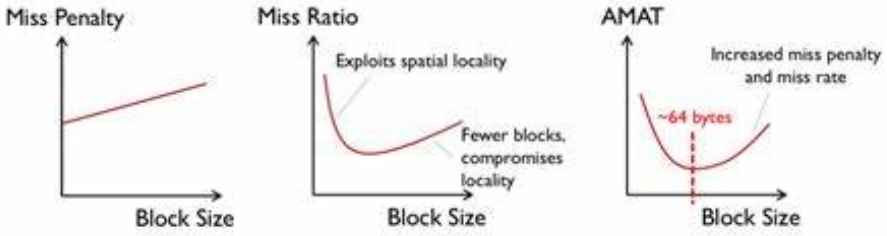
Block size	Cache size				
	1K	4K	16K	64K	256K
16	15.05%	8.57%	3.94%	2.04%	1.09%
32	13.34%	7.24%	2.87%	1.35%	0.70%
64	13.76%	7.00%	2.64%	1.06%	0.51%
128	16.64%	7.78%	2.77%	1.02%	0.49%
256	22.01%	9.51%	3.29%	1.15%	0.49%

FIGURE 5.12 Actual miss rate versus block size for five different-sized caches in Figure 5.11. Note that for a 1-KB cache, 64-byte, 128-byte, and 256-byte blocks have a higher miss rate than 32-byte blocks. In this example, the cache would have to be 256 KB in order for a 256-byte block to decrease misses.

2/15/99 CS520S99 Memory C. Edward Chow Page 34

Block Size Tradeoffs

- Larger block sizes...
  - Take advantage of spatial locality
  - Incur larger miss penalty since it takes longer to transfer the block into the cache
  - Can increase the average hit time and miss rate
- Average Access Time (AMAT) = HitTime + MissPenalty*MR



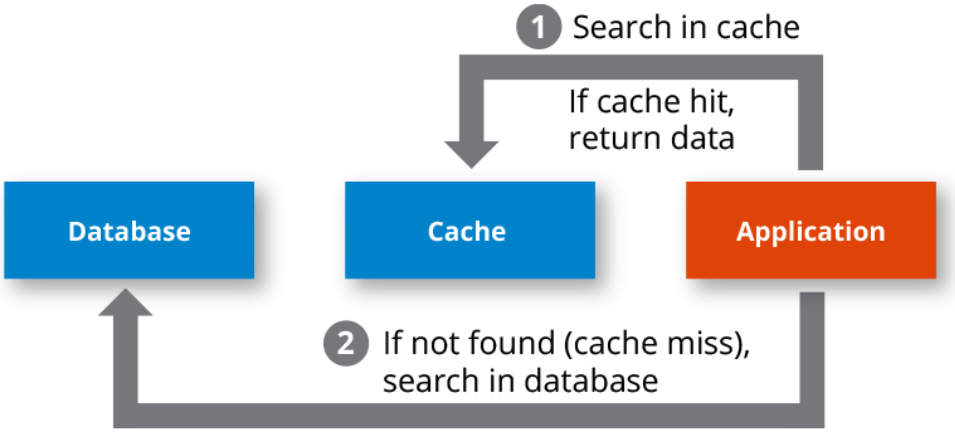
$$\frac{\#\text{ of cache hits}}{(\#\text{ of cache hits} + \#\text{ of cache misses})} = \text{Hit ratio}$$

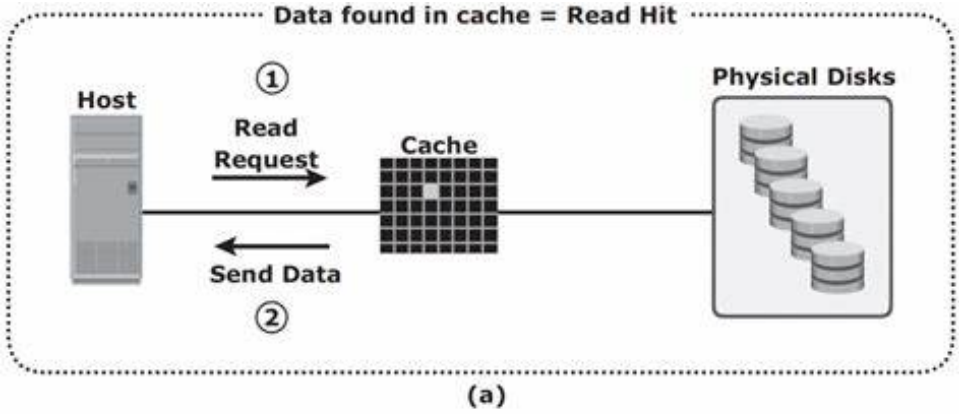
OR

Hit ratio = 1 - Miss ratio

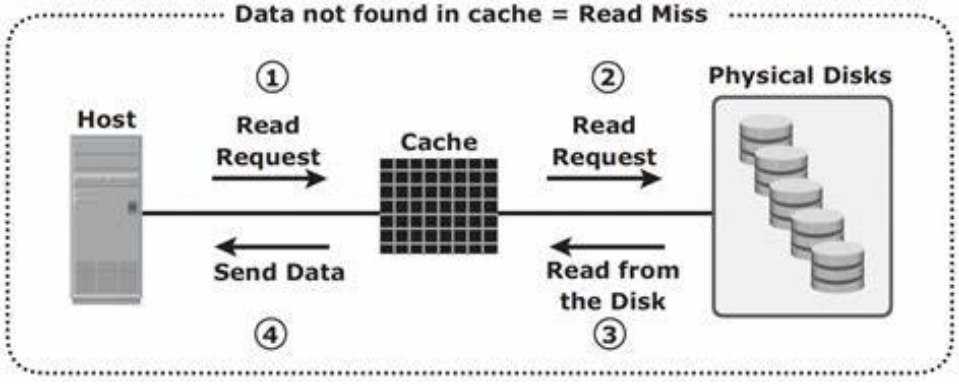
$$\text{Avg mem access time} = \text{Hit time}_{L1} + \text{Miss rate}_{L1} \times \text{Miss penalty}_{L1}$$
$$\text{Miss penalty}_{L1} = \text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2}$$

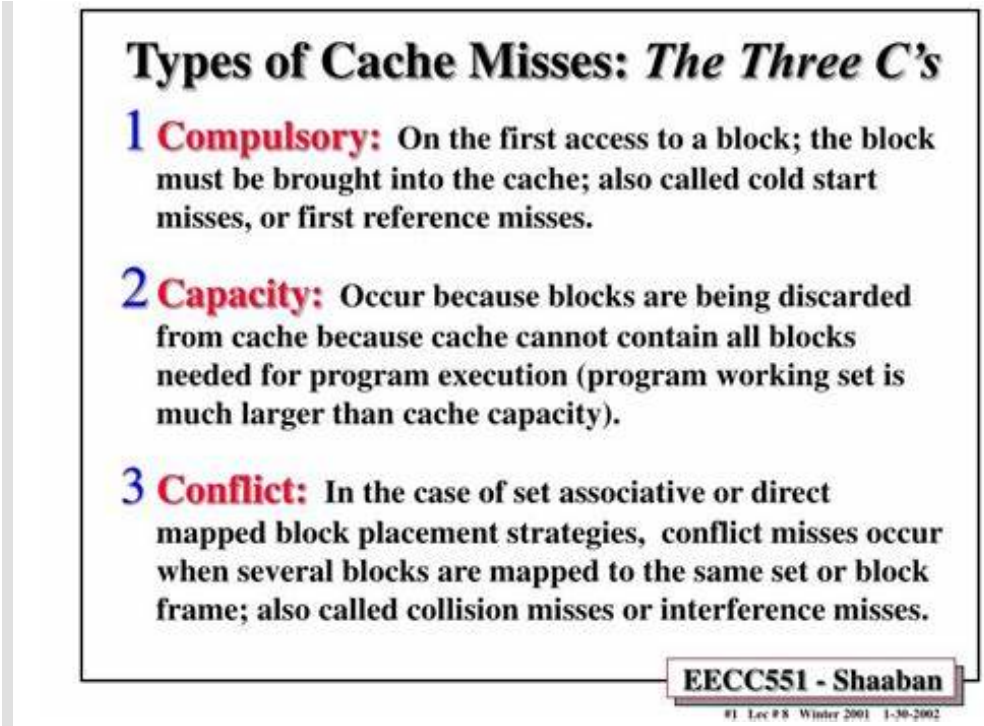
Handle Cache Miss

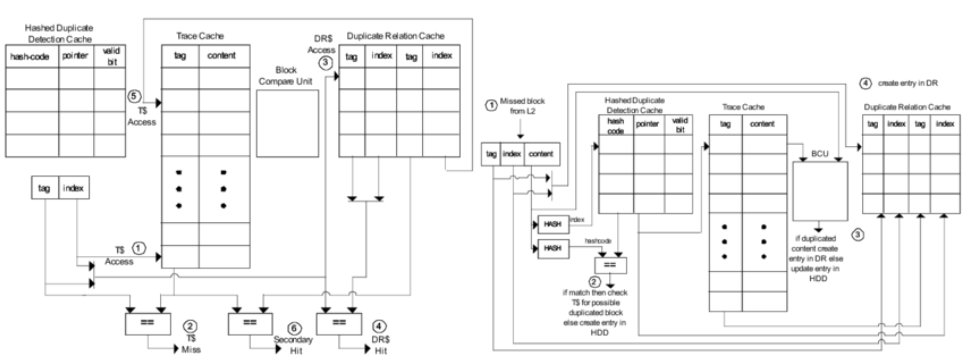


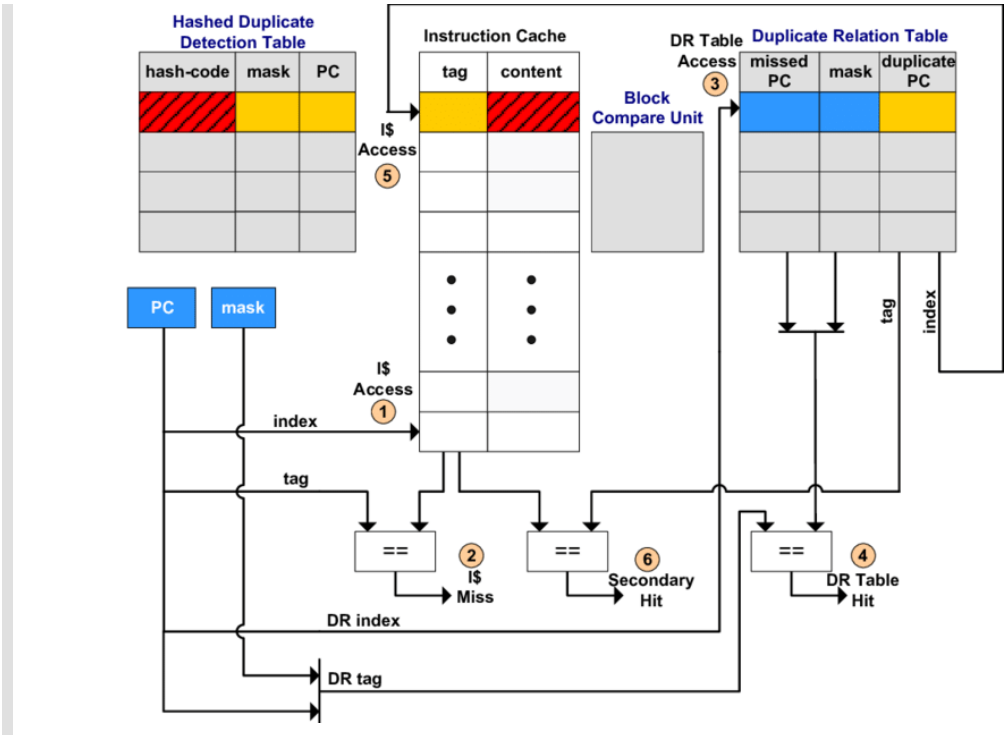


(a)









Cache Performance

Cache Performance Equations

- **Memory stalls per program (blocking cache):**

$$MemoryStallCycle = IC \times (\frac{MemoryAccesses}{Instruction}) \times MissRate \times MissPenalty$$

$$MemoryStallCycle = IC \times (\frac{Misses}{Instruction}) \times MissPenalty$$

- **CPU time formula:**

$$CPU\ Time = IC \times (CPI_{Execu} + \frac{Memory\ Stall\ Cycle}{Instruction}) \times Cycle\ Time$$

- **More cache performance will be given later!**

46

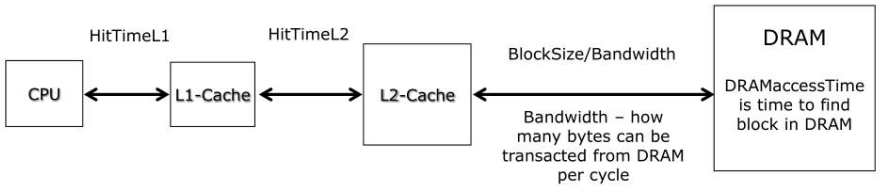
Cache Performance

- CPI $_{\text{contributed by cache}}$ = CPI_c
  = miss rate * number of cycles to handle the miss
- Another important metric
  Average memory access time = cache hit time * hit rate
  + Miss penalty * (1 - hit rate)

Caches CSE 471 15

2-level Cache Performance Equations

- L1 AMAT = HitTimeL1 + MissRateL1 * MissPenaltyL1
  - MissLatencyL1 is low, so optimize HitTimeL1
- MissPenaltyL1 = HitTimeL2 + MissRateL2 * MissPenaltyL2
  - MissLatencyL2 is high, so optimize MissRateL2
  - MissPenaltyL2 = DRAMaccessTime + (BlockSize/Bandwidth)
  - If DRAM time high or bandwidth high, use larger block size
- L2 miss rate:
  - Global: L2 misses / total CPU references
  - Local: L2 misses / CPU references that miss in L1
  - The equation above assumes local miss rate



18

Improving Cache Performance

- **Miss Rate Reduction Techniques:**
  - Increased cache capacity
  - Higher associativity
  - Hardware prefetching of instructions and data
  - Compiler-controlled prefetching
  - Larger block size
  - Victim caches
  - Pseudo-associative Caches
  - Compiler optimizations
- **Cache Miss Penalty Reduction Techniques:**
  - Giving priority to read misses over writes
  - Early restart and critical word first
  - Second-level cache (L_2)
  - Sub-block placement
  - Non-blocking caches
- **Cache Hit Time Reduction Techniques:**
  - Small and simple caches
  - Avoiding address translation during cache indexing
  - Pipelining writes for fast write hits

EECC551 - Shaaban

#7 Lec # 10 Winter2000 1-23-2000

Qualitative Cache Performance Model

Miss Types

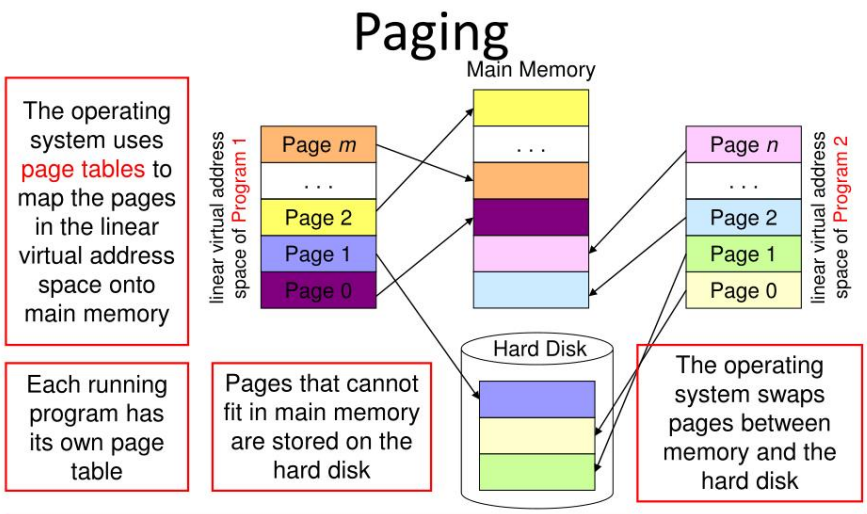
- Compulsory ("Cold Start") Misses
  - First access to line not in cache
- Capacity Misses
  - Active portion of memory exceeds cache size
- Conflict Misses
  - Active portion of address space fits in cache, but too many lines map to same cache entry
  - Direct mapped and set associative placement only
- Coherence Misses
  - Block invalidated by multiprocessor cache coherence mechanism

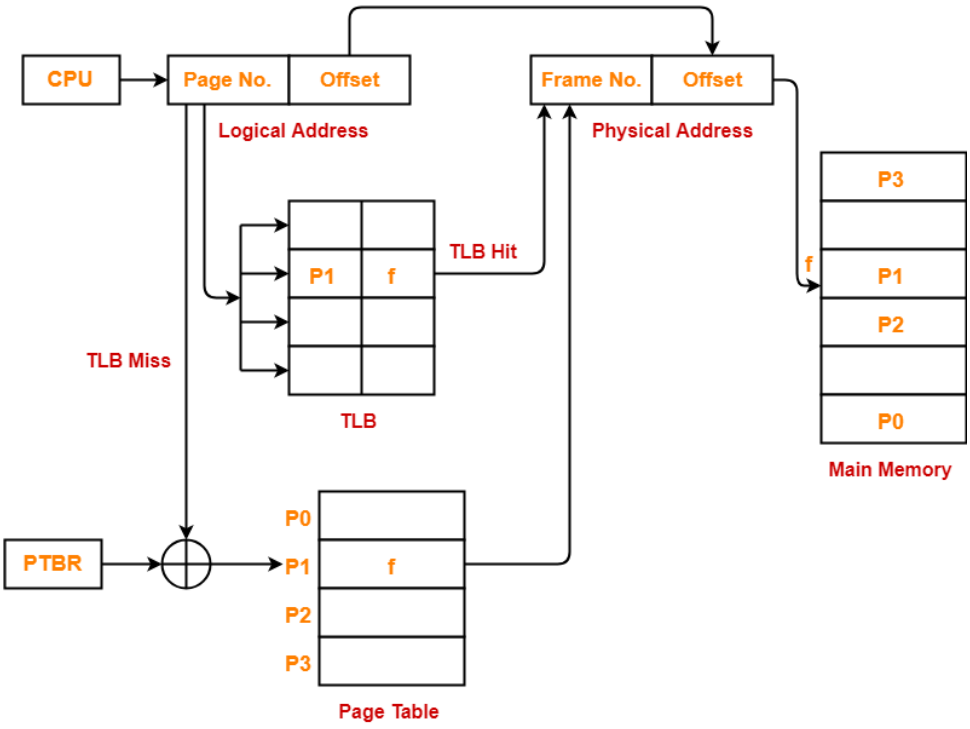
Hit Types

- Temporal locality hit
  - Accessing same word that previously accessed
- Spatial locality hit
  - Accessing word spatially near previously accessed word

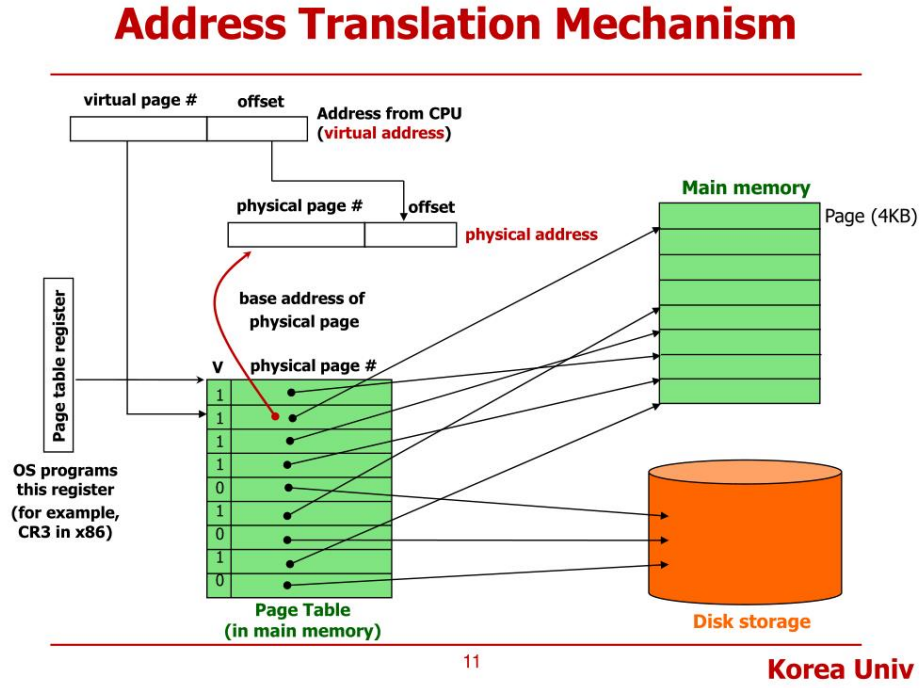
- 42 - CS 740 F'07

Address Translation Mechanism





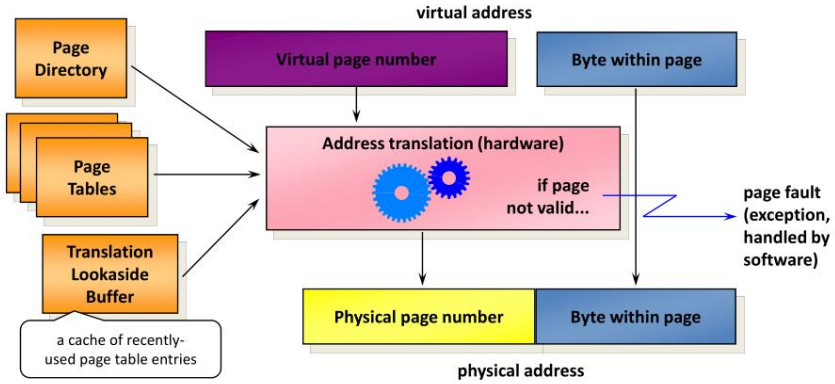
Translating Logical Address into Physical Address



Address Translation with Cache

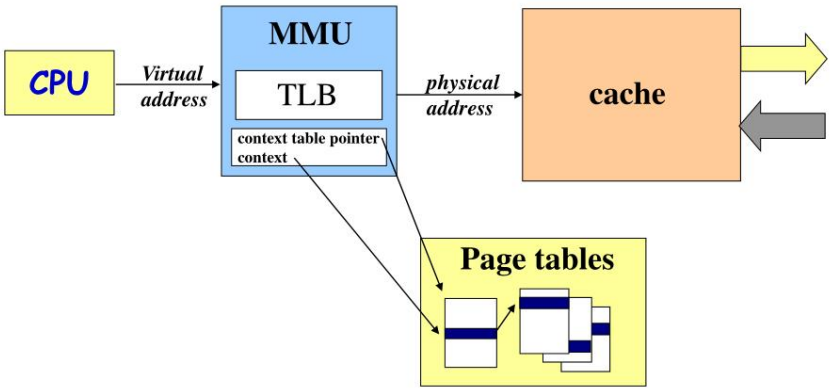
Virtual Address Translation

- The hardware converts each valid virtual address to a physical address



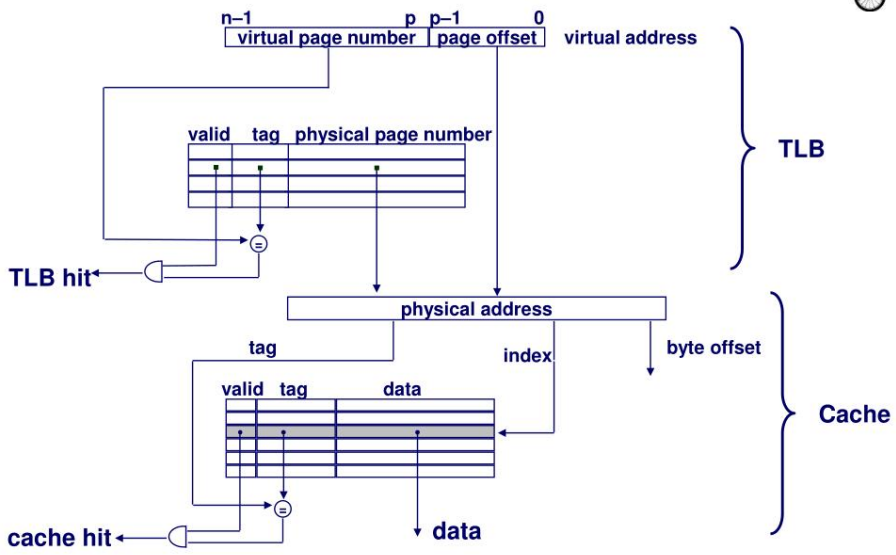
31

Address Translation Overview

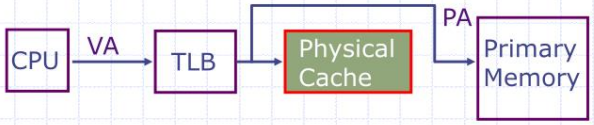


Fred Kuhns (3/11/2014) CS523 - Operating Systems 29

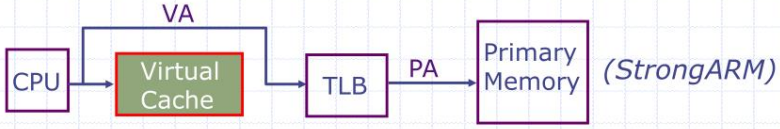
Address Translation With a TLB



Physical or Virtual Address Caches?

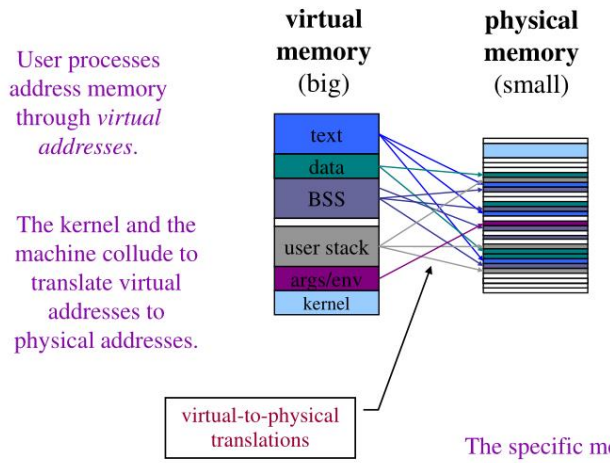


Alternative: place the cache before the TLB



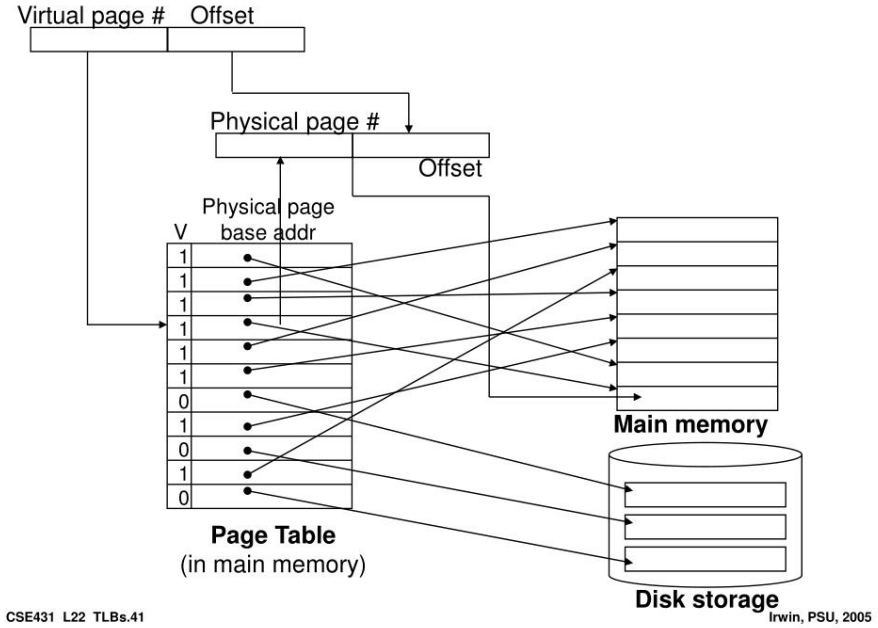
- one-step process in case of a hit (+)
- cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (-)
- *aliasing problems* due to the sharing of pages (-)

Review: Virtual Addressing



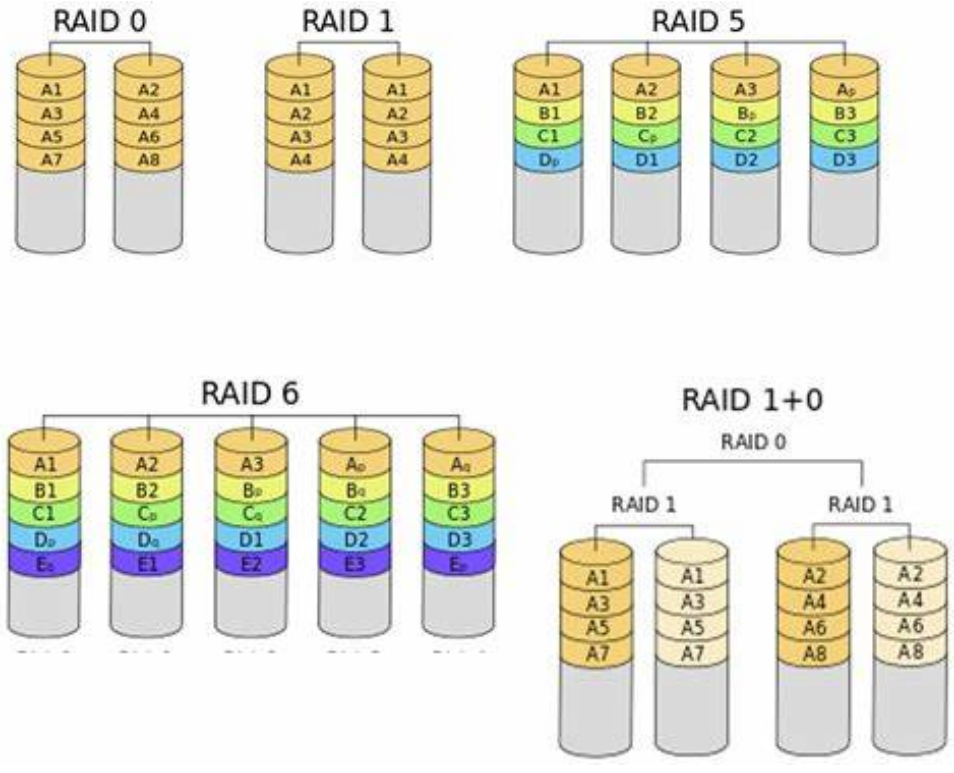
DUKE *Systems & Architecture*

Address Translation Mechanisms



CSE431 L22 TLBs.41 Irwin, PSU, 2005

Redundant Array of Inexpensive Disk



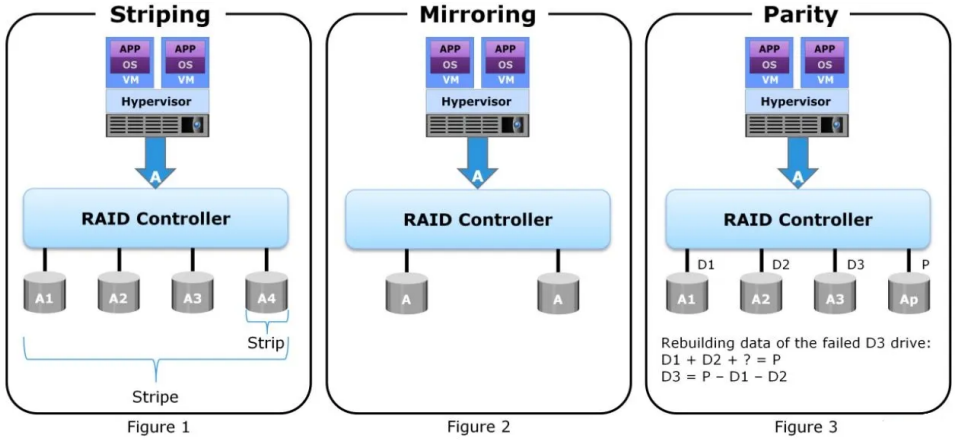
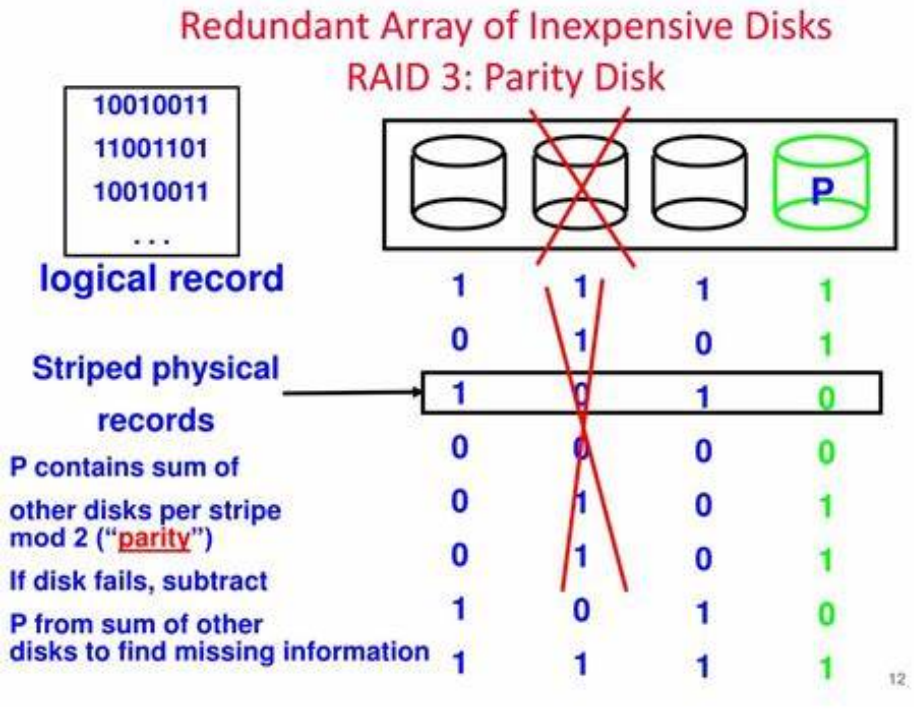


Figure 1 Figure 2 Figure 3



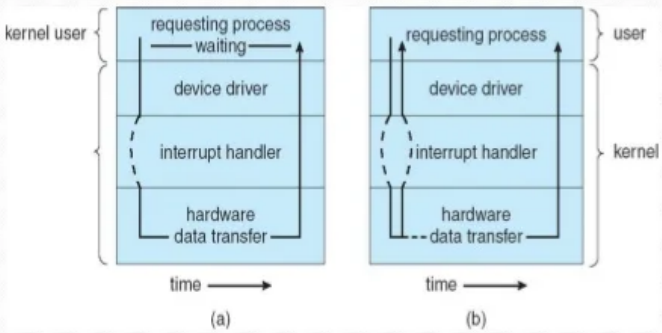
IO System Performance

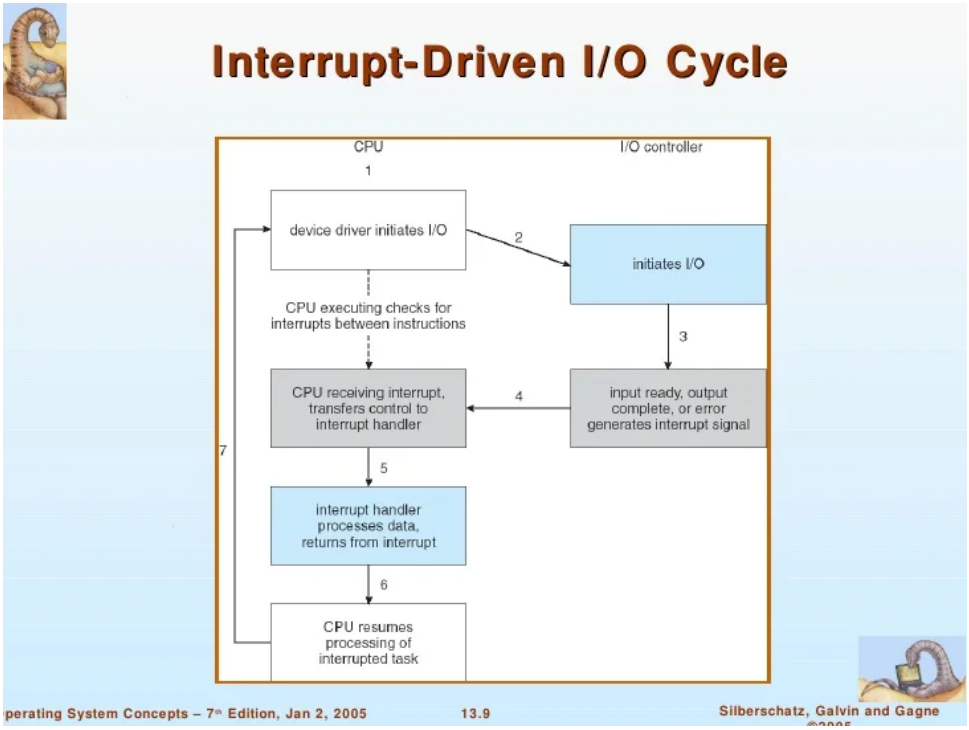
Common Concepts in I/O Hardware

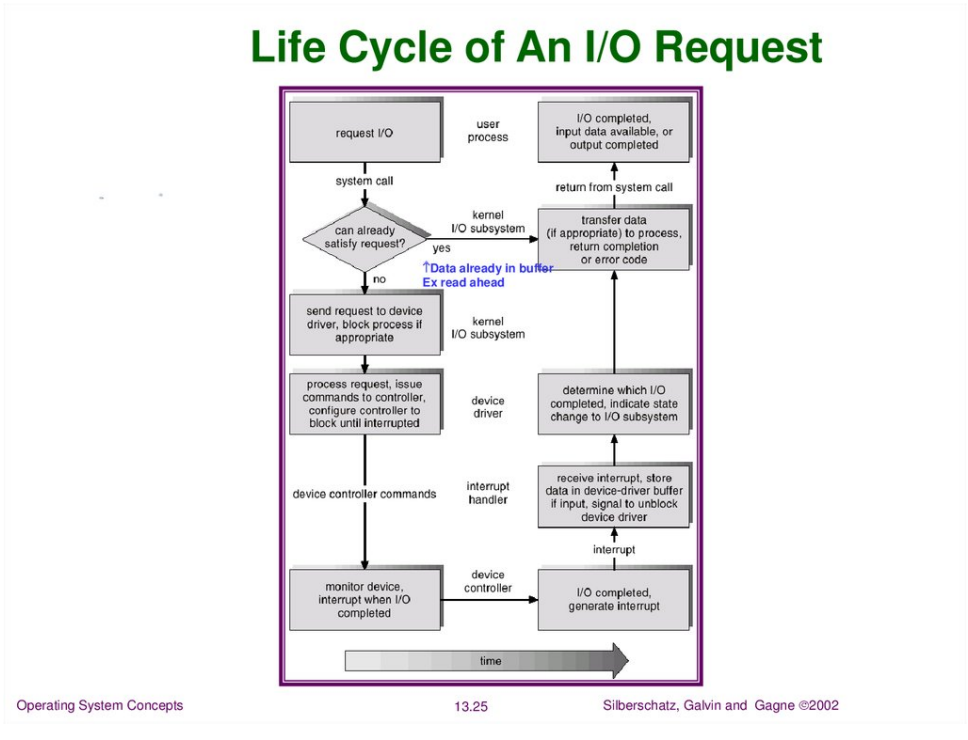
- Common concepts: signals from I/O devices interface with computer.
- Port: connection point for device
- Bus: set of wires and a protocol that specifies a set of messages that can be sent on the wires.
- Controller: a collection of electronics that can operate a port, a bus, or a device.
  - Sometimes integrated and sometimes separate circuit board (host adapter)
  - Contains processor, microcode, private memory, bus controller, etc

Amir H. Payberah (Tehran Polytechnic) I/O Systems 1393/9/15 6 / 57

Two I/O Methods

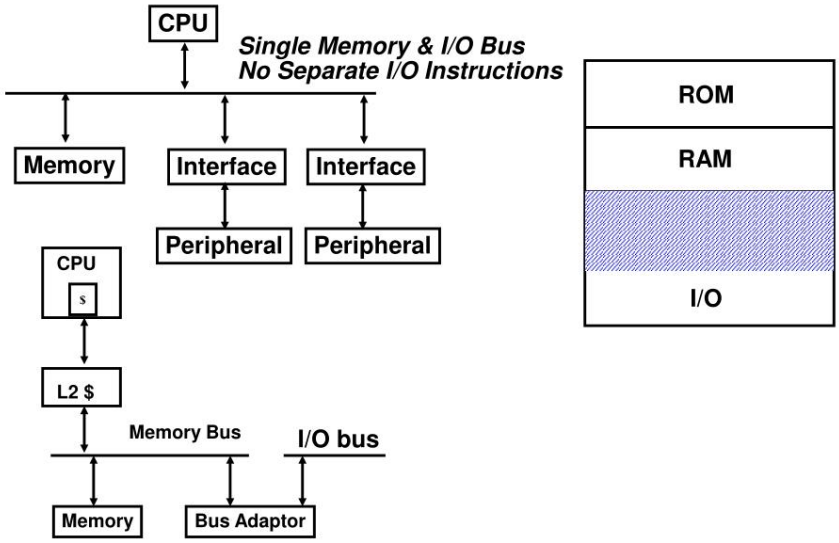


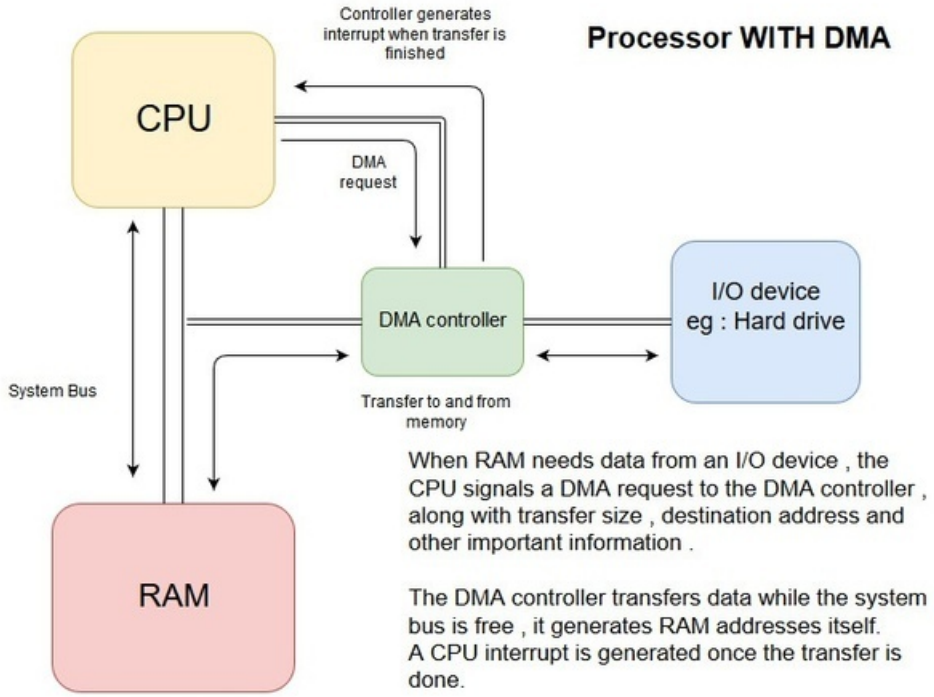




DMA(Direct Memroy Access)

Memory Mapped I/O

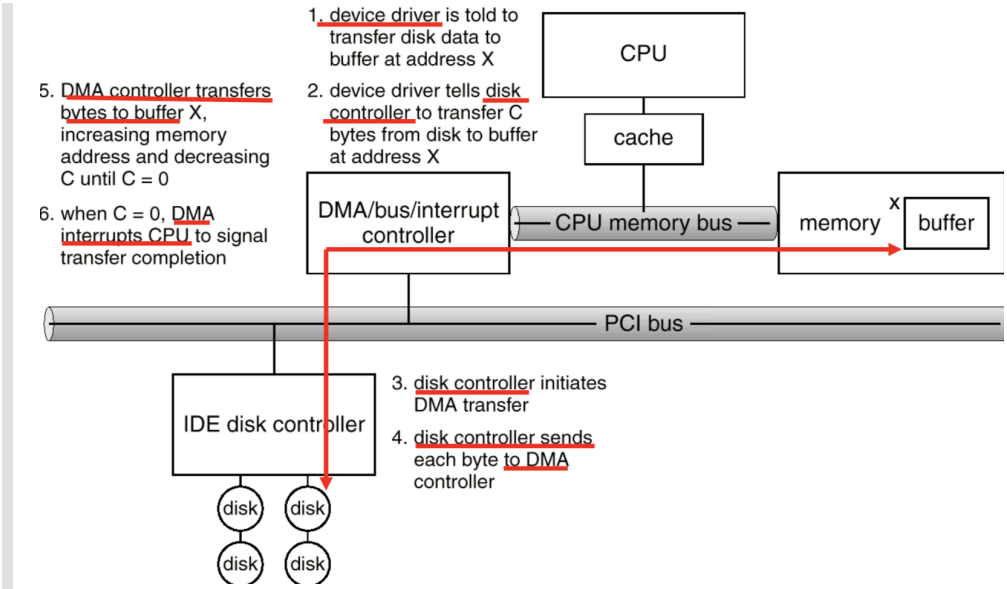


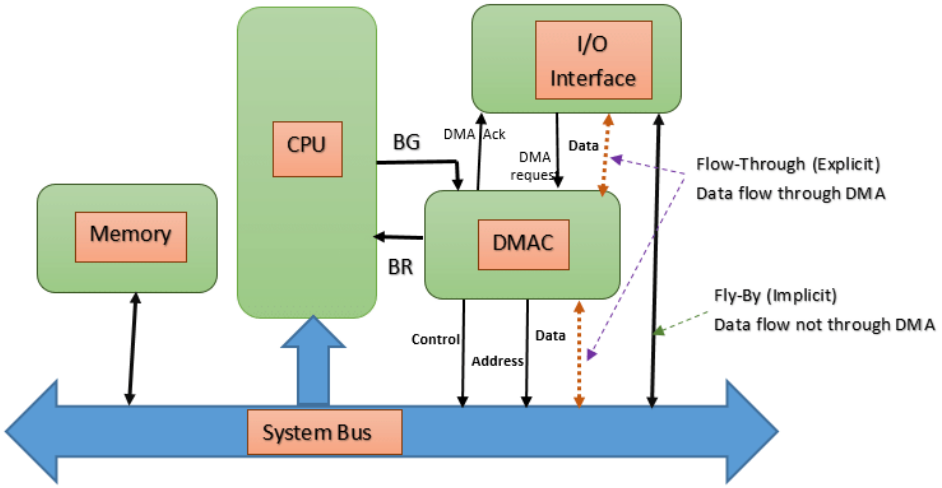


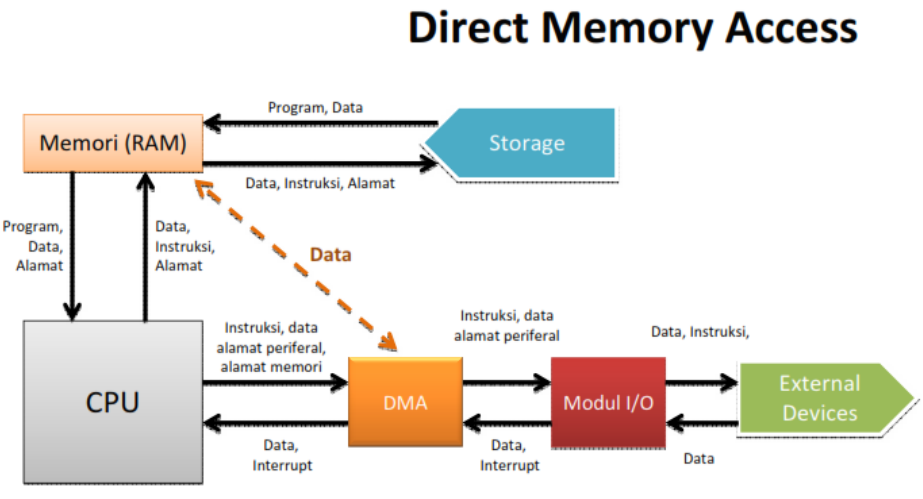
When RAM needs data from an I/O device , the CPU signals a DMA request to the DMA controller , along with transfer size , destination address and other important information .

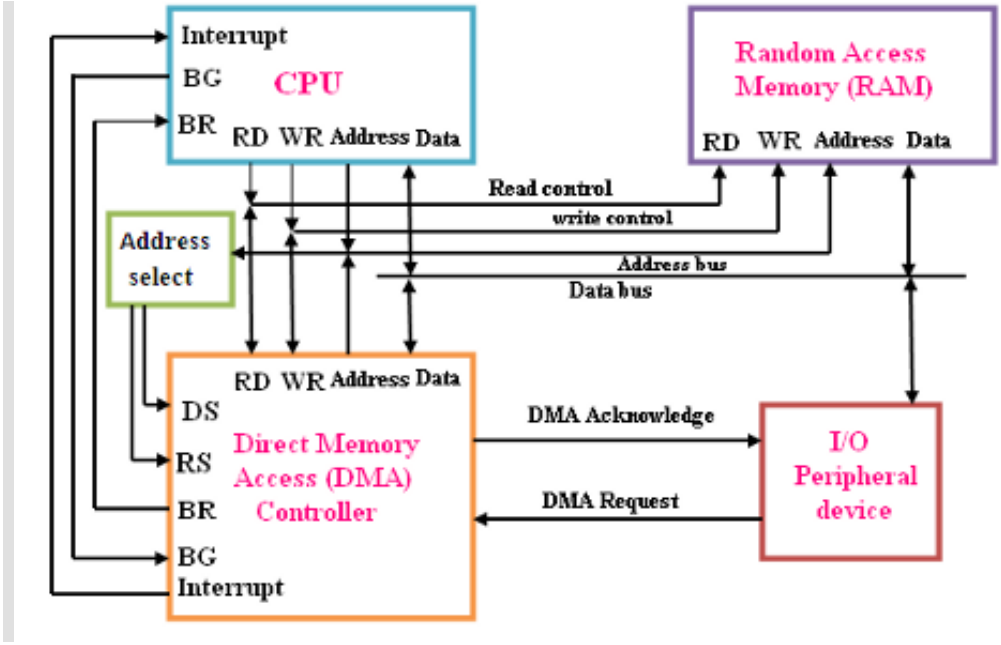
The DMA controller transfers data while the system bus is free , it generates RAM addresses itself.
A CPU interrupt is generated once the transfer is done.

Result : CPU is free to do other things !


1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion
CPU
cache
DMA/bus/interrupt controller
CPU memory bus
memory
x
buffer
PCI bus
IDE disk controller
disk
disk
disk
disk

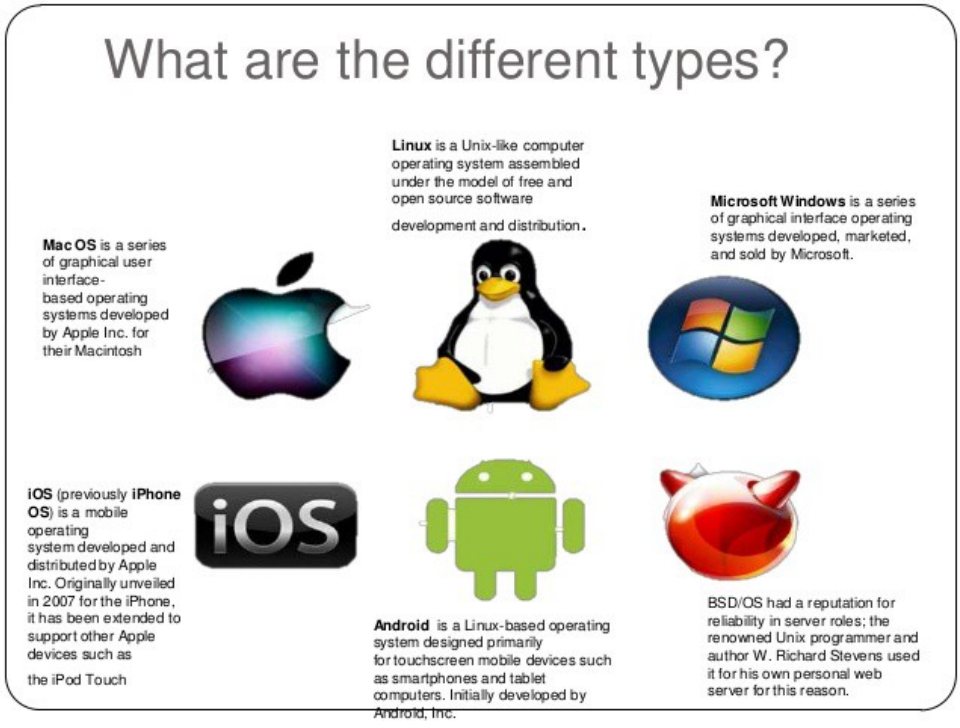

CPU
I/O Interface
BG
DMA Ack
DMA reques
Data
Flow-Through (Explicit)
Data flow through DMA
Memory
DMAC
BR
Fly-By (Implicit)
Data flow not through DMA
Control
Address
Data
System Bus


Direct Memory Access
Memori (RAM)
Program, Data
Storage
Data, Instruksi, Alamat
Program, Data, Alamat
Data, Instruksi, Alamat
Data
CPU
Instruksi, data alamat periferal, alamat memori
Instruksi, data alamat periferal
Data, Instruksi,
DMA
Modul I/O
External Devices
Data, Interrupt
Data, Interrupt
Data



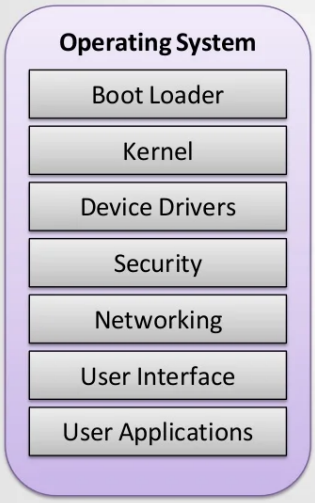
Operating System

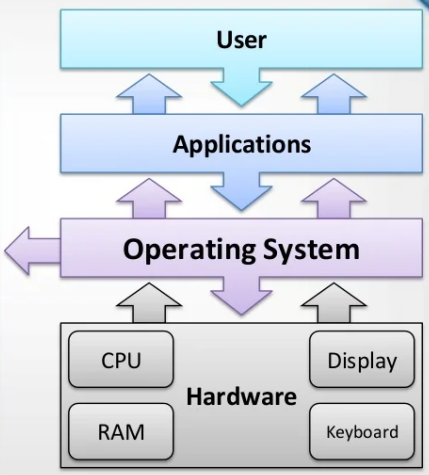
Types of OS



OS Structure

Components





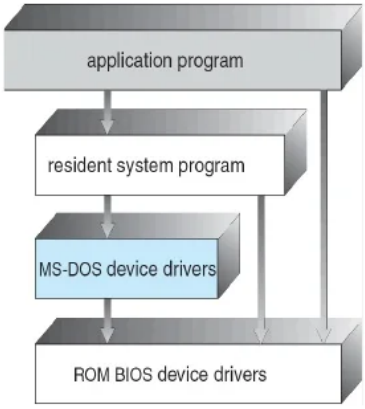
Computer Operating Systems 10

7. Operating-System Structure

- How OS components are interconnected and melded into a kernel.

7.1 Simple Structure

- Started as small, simple, and limited systems and then grown beyond their original scope
- Example : MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



MS-DOS layer structure

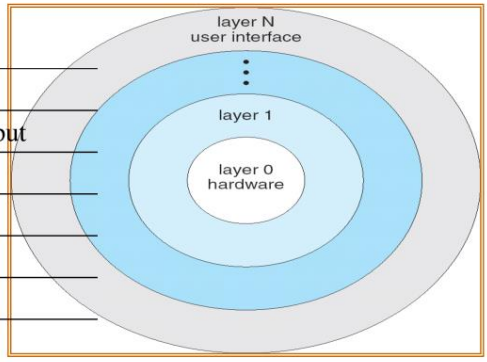
Loganathan R, CSE , HKBKCE 15

Layered Structure of the THE OS

- A layered design was first used in THE operating system.

Its six layers are as follows:

layer 5:	user programs
layer 4:	buffering for input and output
layer 3:	Process management
layer 2:	memory management
layer 1:	CPU scheduling
layer 0:	hardware

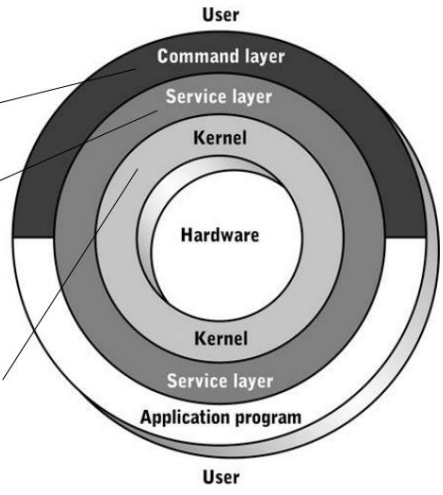


6

CIS410

Operating System Layers

Figure 11-3 ►
Operating system layers (shaded)



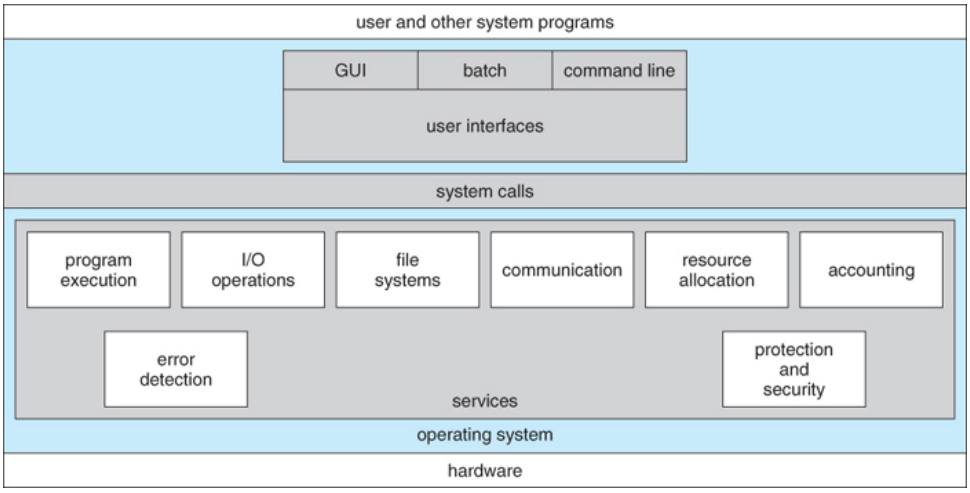
User's interface to OS

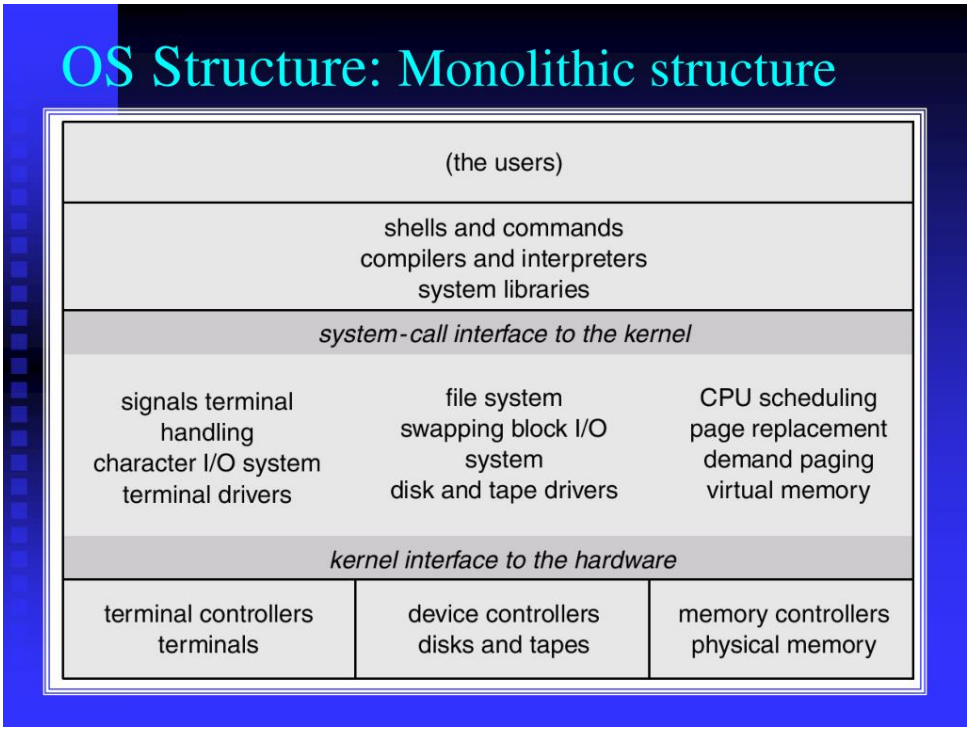
Contains set of functions executed by application programs and command layer

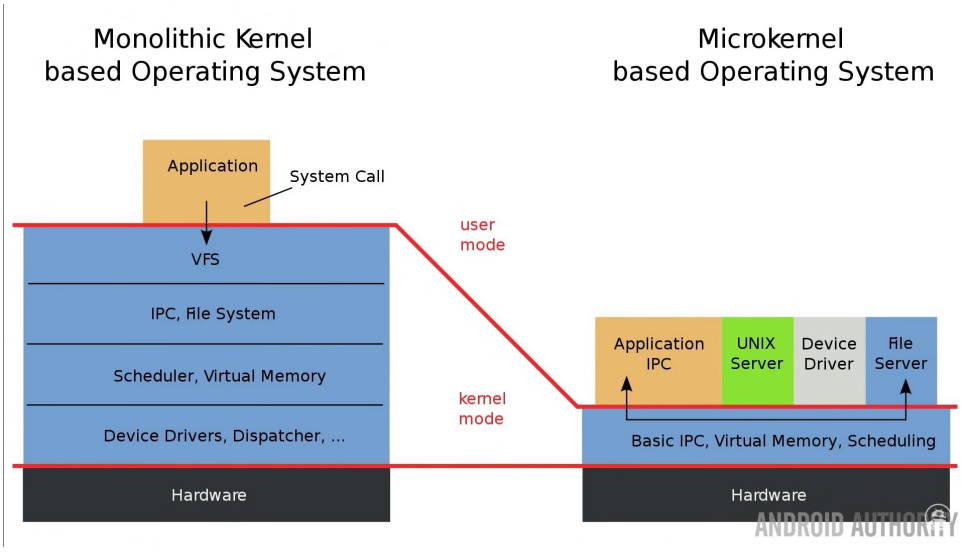
Manages resources; interacts directly with computer hardware

Hardware and Software Architecture

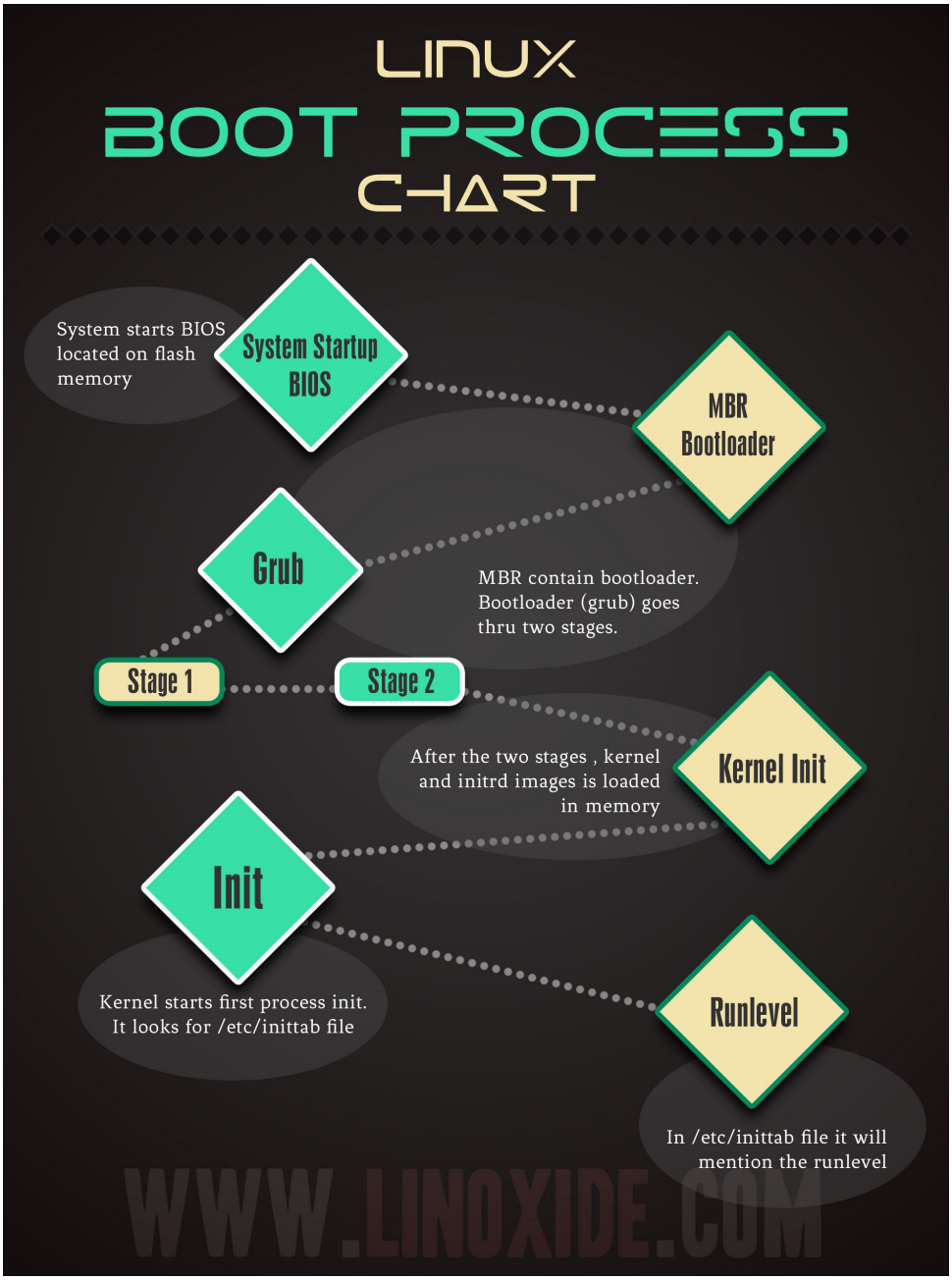
16

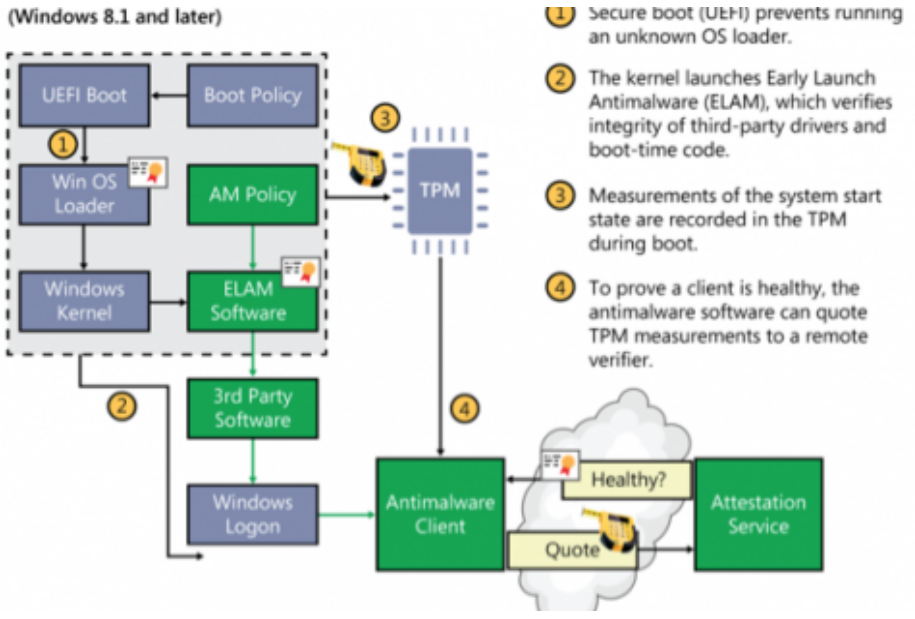


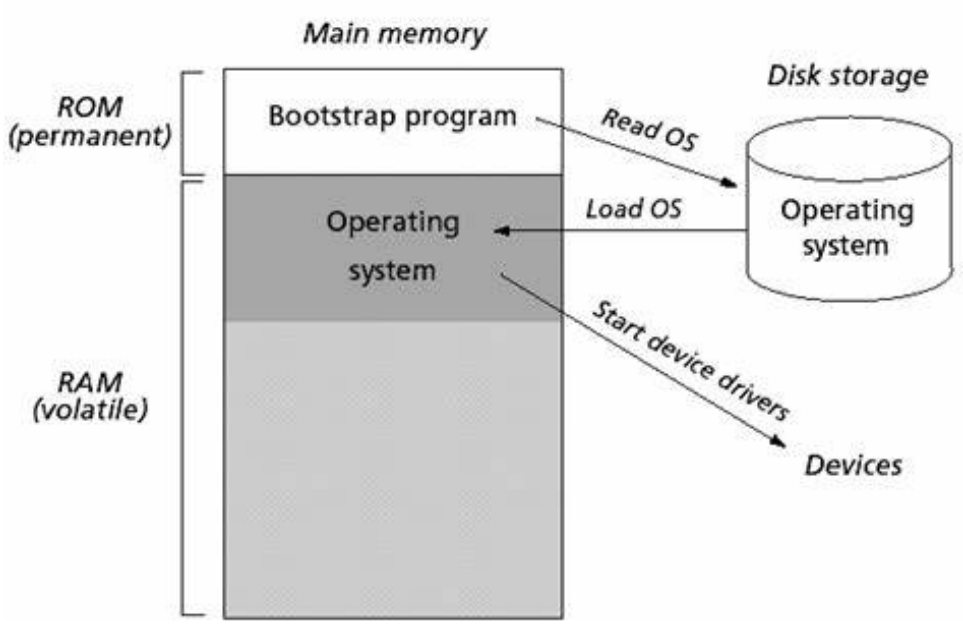

OS Structure: Monolithic structure
(the users)
shells and commands
compilers and interpreters
system libraries
system-call interface to the kernel
signals terminal handling
character I/O system
terminal drivers
file system
swapping block I/O system
disk and tape drivers
CPU scheduling
page replacement
demand paging
virtual memory
kernel interface to the hardware
terminal controllers
terminals
device controllers
disks and tapes
memory controllers
physical memory

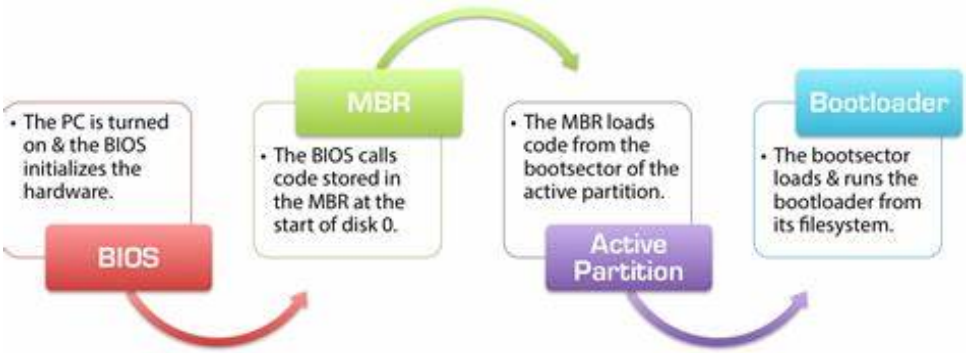

Monolithic Kernel based Operating System
Microkernel based Operating System
Application
System Call
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
user mode
kernel mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware
ANDROID AUTHORITY

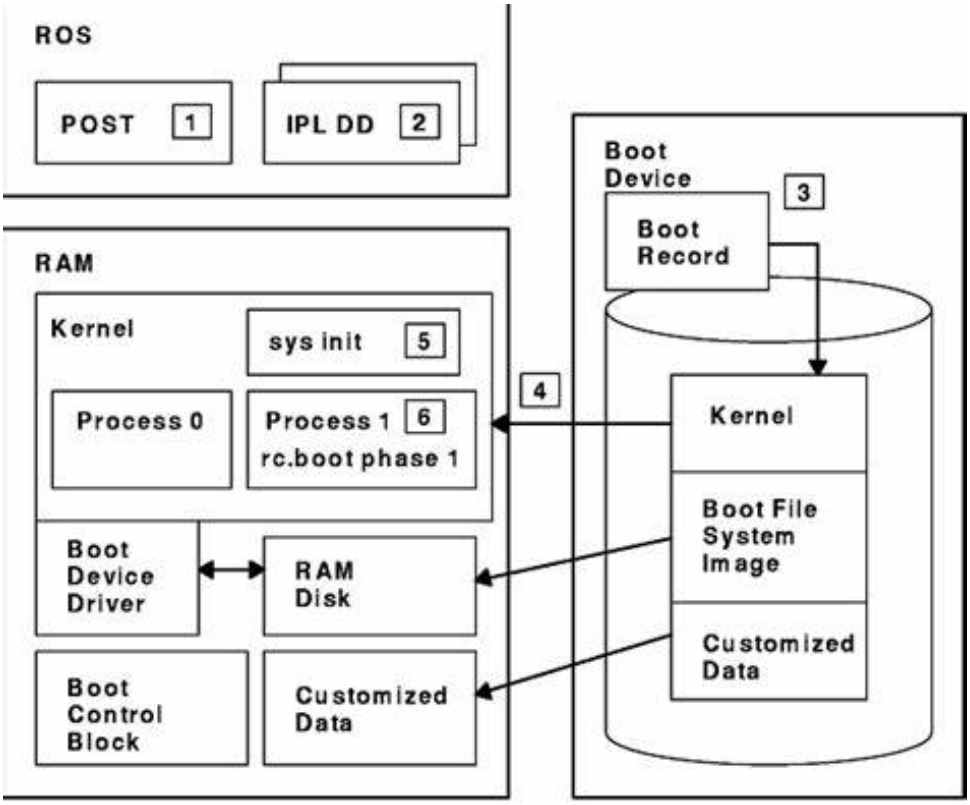
System Boot


LINUX
BOOT PROCESS
CHART
System starts BIOS located on flash memory
System Startup BIOS
MBR Bootloader
Grub
MBR contain bootloader. Bootloader (grub) goes thru two stages.
Stage 1
Stage 2
After the two stages , kernel and initrd images is loaded in memory
Kernel Init
Init
Kernel starts first process init. It looks for /etc/inittab file
Runlevel
In /etc/inittab file it will mention the runlevel
WWW.LINOXIDE.COM


(Windows 8.1 and later)
UEFI Boot
Boot Policy
Win OS Loader
AM Policy
TPM
Windows Kernel
ELAM Software
3rd Party Software
Windows Logon
Antimalware Client
Healthy?
Quote
Attestation Service
1 Secure boot (UEFI) prevents running an unknown OS loader.
2 The kernel launches Early Launch Antimalware (ELAM), which verifies integrity of third-party drivers and boot-time code.
3 Measurements of the system start state are recorded in the TPM during boot.
4 To prove a client is healthy, the antimalware software can quote TPM measurements to a remote verifier.

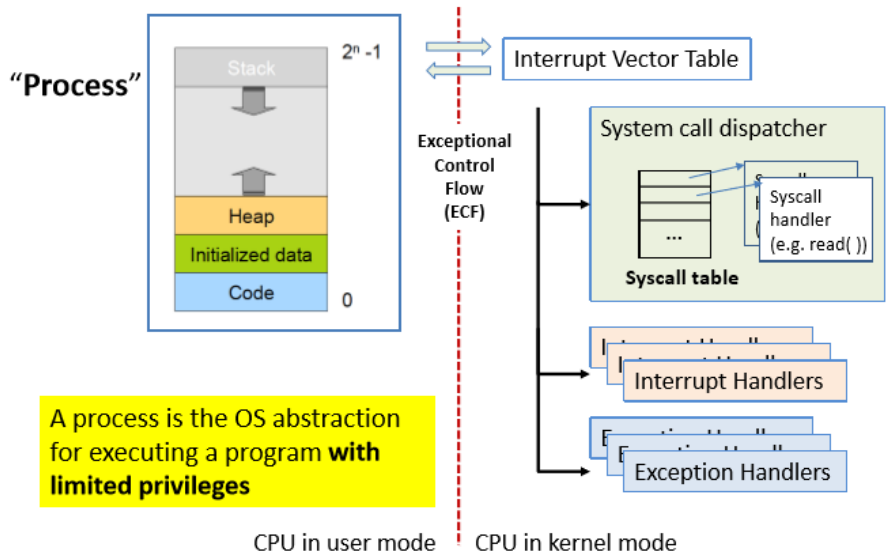


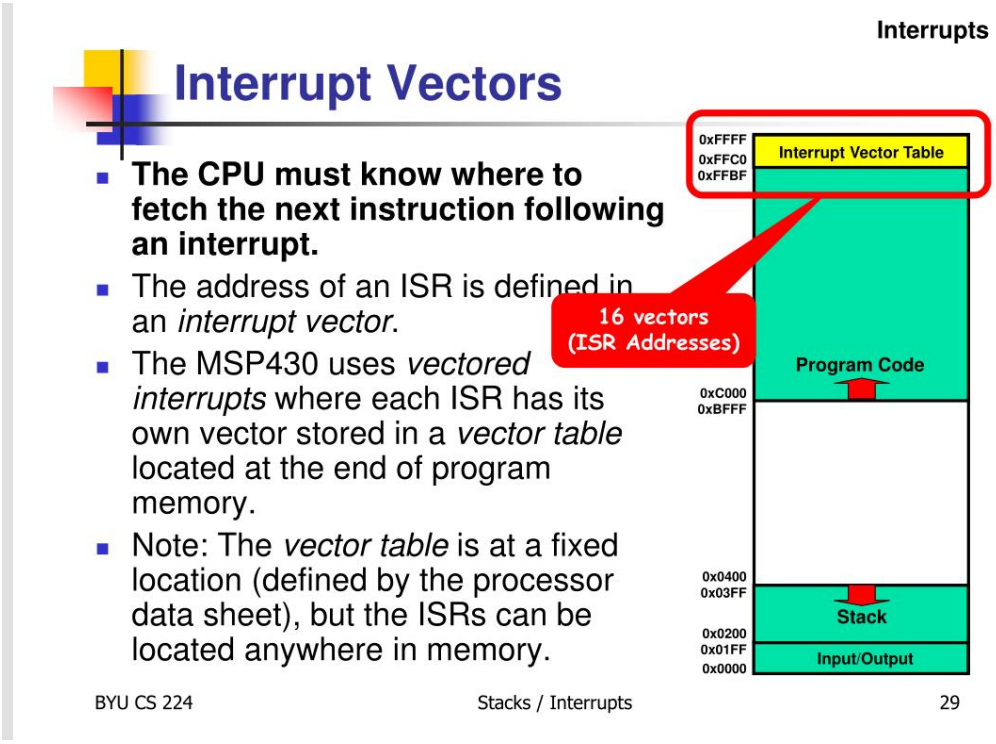




ROS Kernel Init Phase

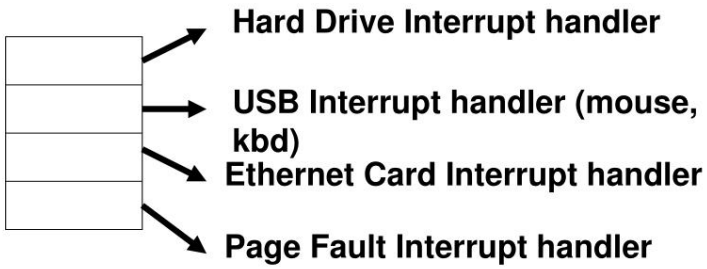
OS - Interrupt Driven



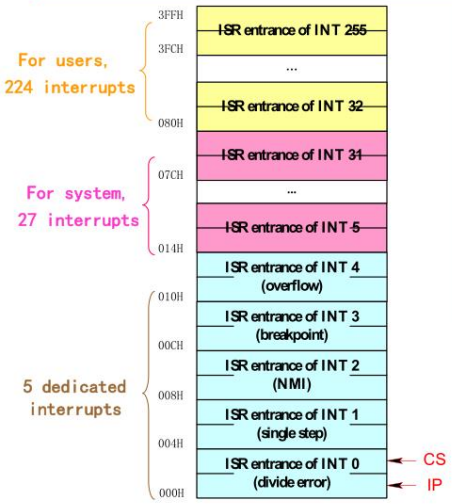


Interrupt Vector

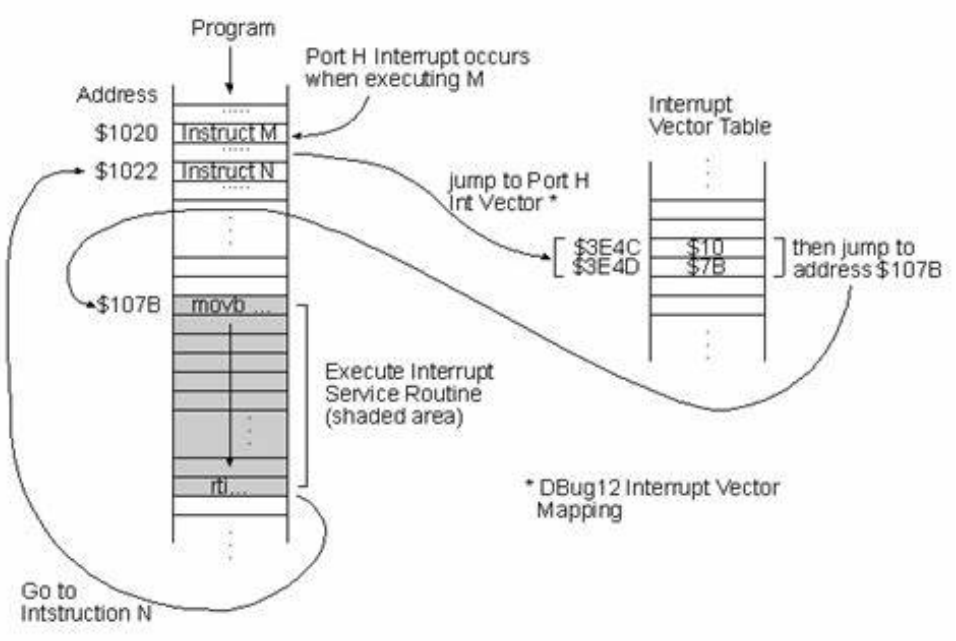
- It is an array of pointers that point to the different interrupt handlers of the different types of interrupts.



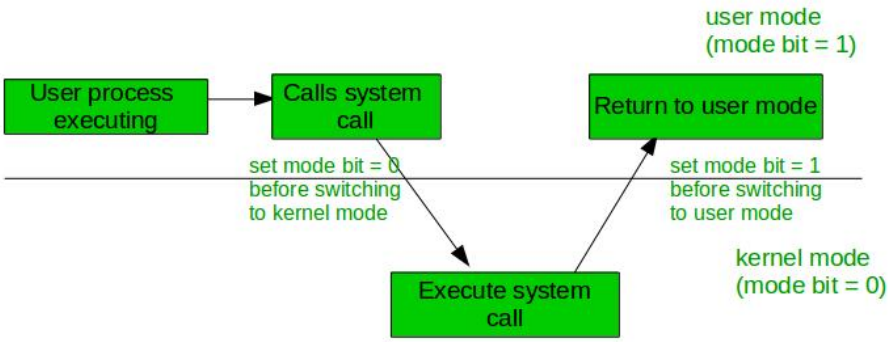
Interrupt Vector Table of 8086/8088

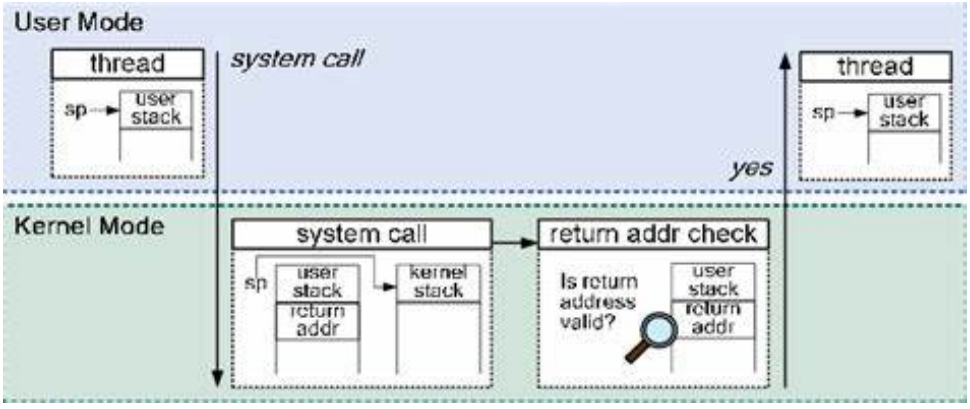


- 256 interrupts
  - 0～4 dedicated
  - 5～31reserved for system use
    - 08H～0FH：8259A
    - 10H～1FH：BIOS
  - 32～255 reserved for users
    - 20H～3FH：DOS
    - 40H～FFH：open

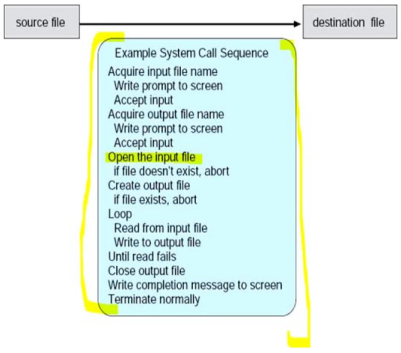


OS - Dual Mode Operation

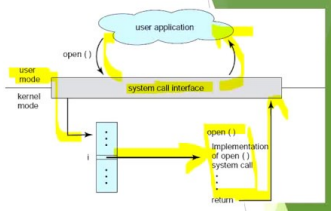




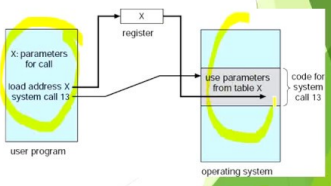
SYSTEM CALLS



Example of how system calls are used.

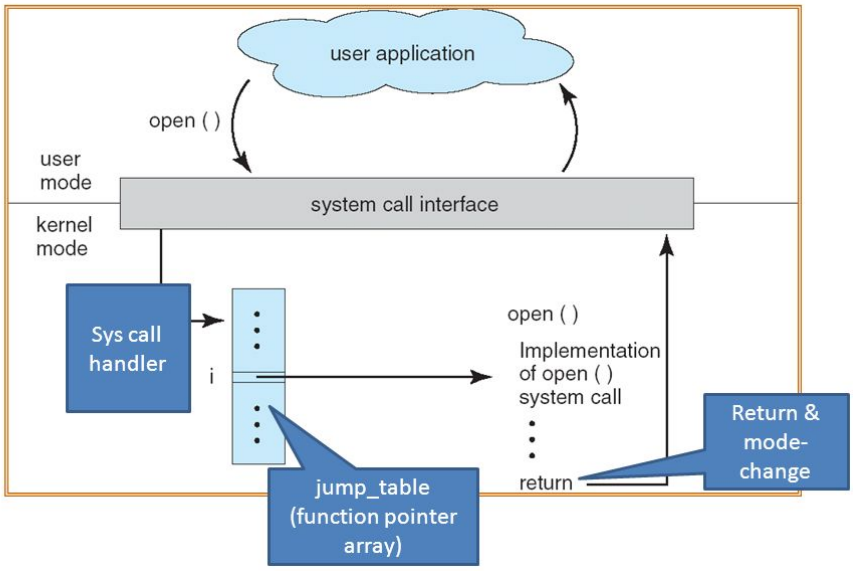


The handling of a user application invoking the open() system call



Passing of parameters as a table.

API – System Call – OS Relationship

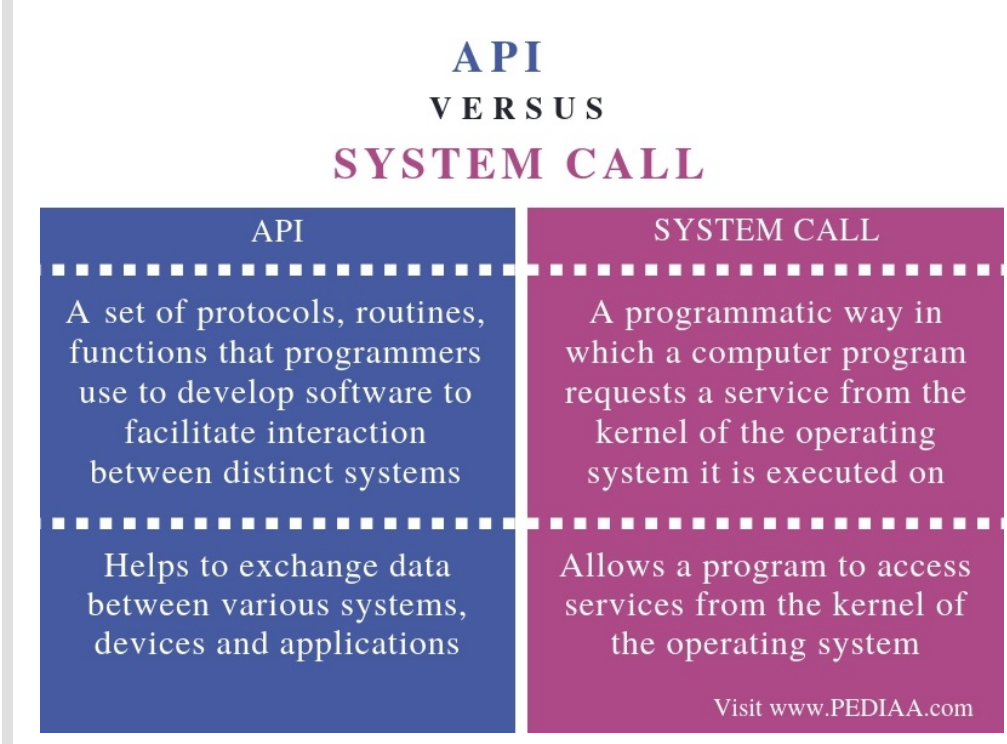


Types of System Calls

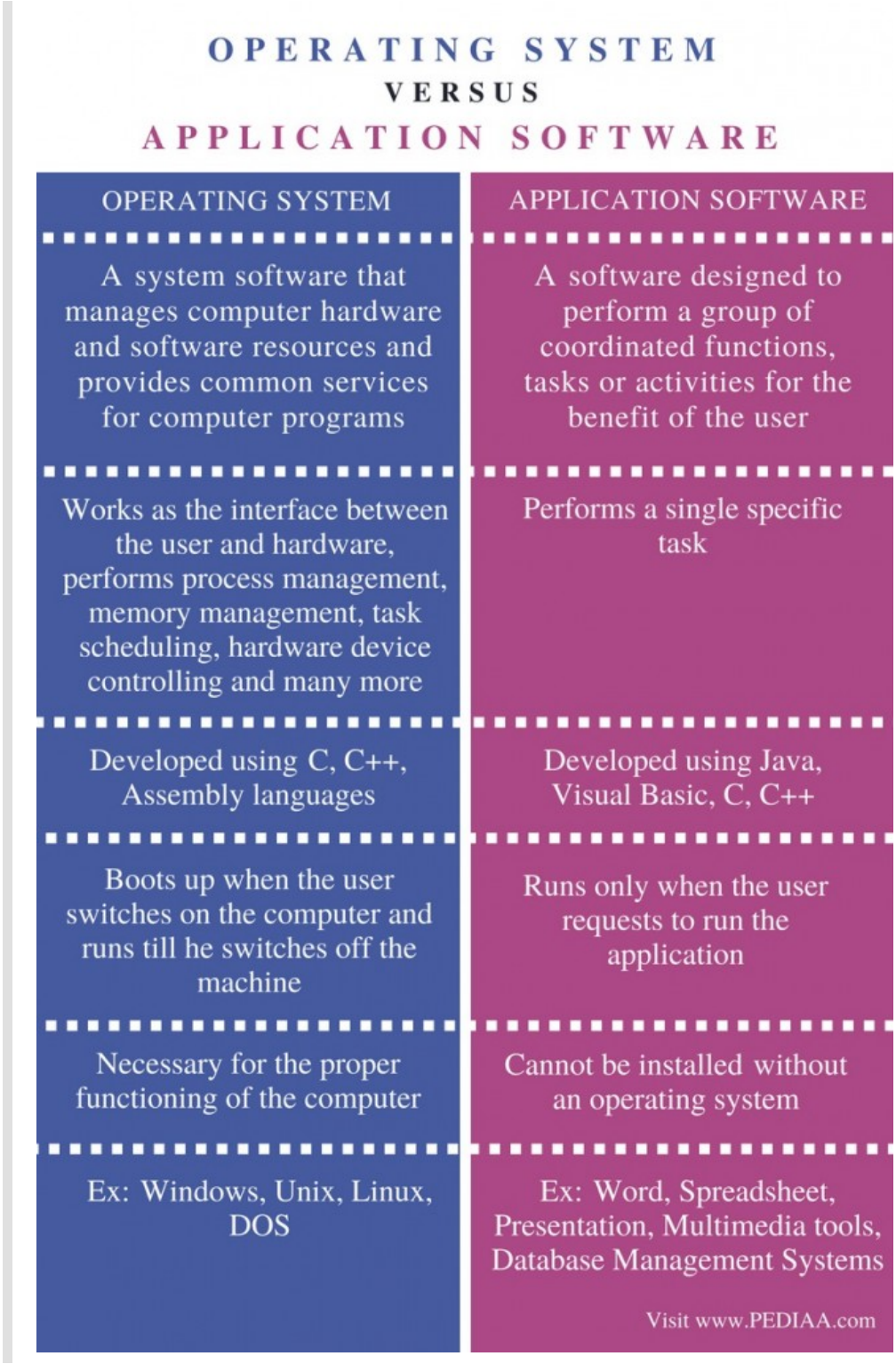
- Process control
  - Load, execute, create process, wait, etc.
  - Differs between single-tasking and multi-tasking.
- File management
  - Create/delete file, open/close, read/write, etc.
- Device management
  - Read, write, reposition, attach/detach device, etc.
- Information maintenance.
  - Get time/date/process/file, set time/date/process/file, etc.
- Communications
  - Send/receive messages , create/delete communication, etc.
  - Two models for IPC (interprocess communication): messages-passing and shared-memory.

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

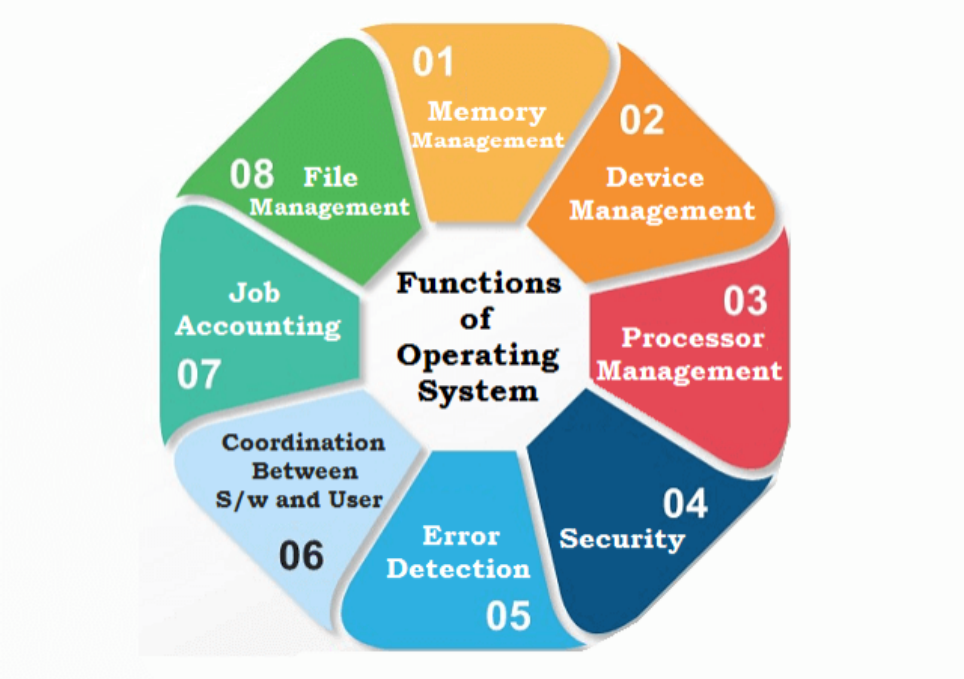
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()


API
VERSUS
SYSTEM CALL
API
SYSTEM CALL
A set of protocols, routines, functions that programmers use to develop software to facilitate interaction between distinct systems
A programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on
Helps to exchange data between various systems, devices and applications
Allows a program to access services from the kernel of the operating system
Visit www.PEDIAA.com

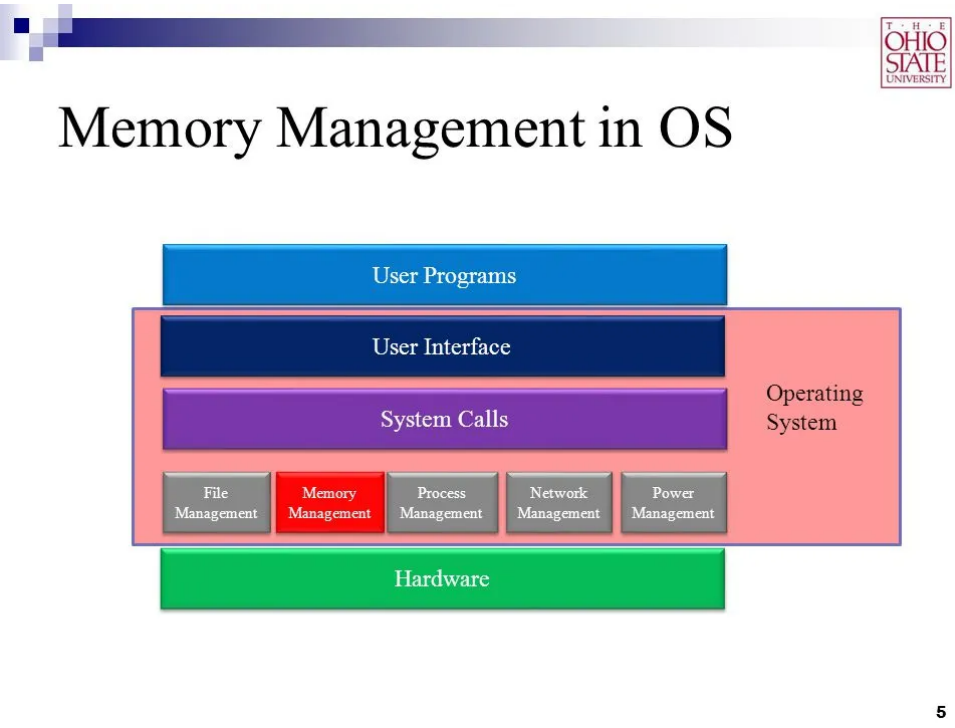

SYSTEM CALL
VERSUS
LIBRARY CALL
SYSTEM CALL
LIBRARY CALL
A request by the program to the kernel to enter kernel mode to access a resource
A request made by the program to access a function defined in a programming library
The mode changes from user mode to kernel mode
There is no mode switching
Not portable
Portable
Execute slower than library calls
Execute faster than system calls
System calls have more privileges than library calls
Library calls have less privileges than system calls
fork() and exec() are some examples for system calls
fopen(), fclose(), scanf() and prinf() are some examples for library calls
Visit www.PEDIAA.com

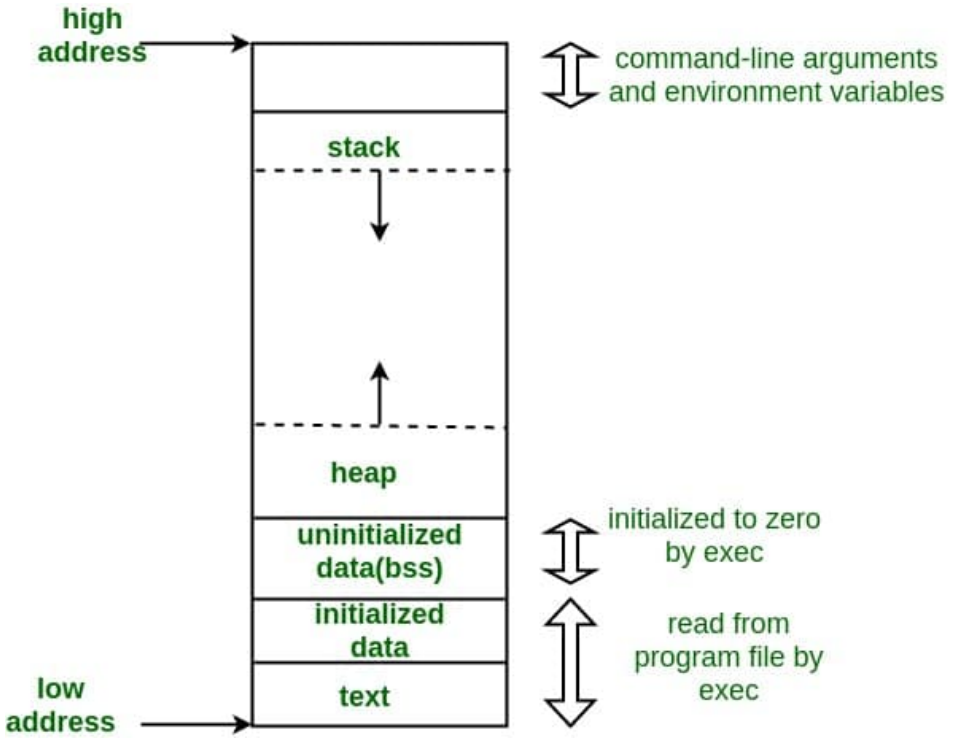


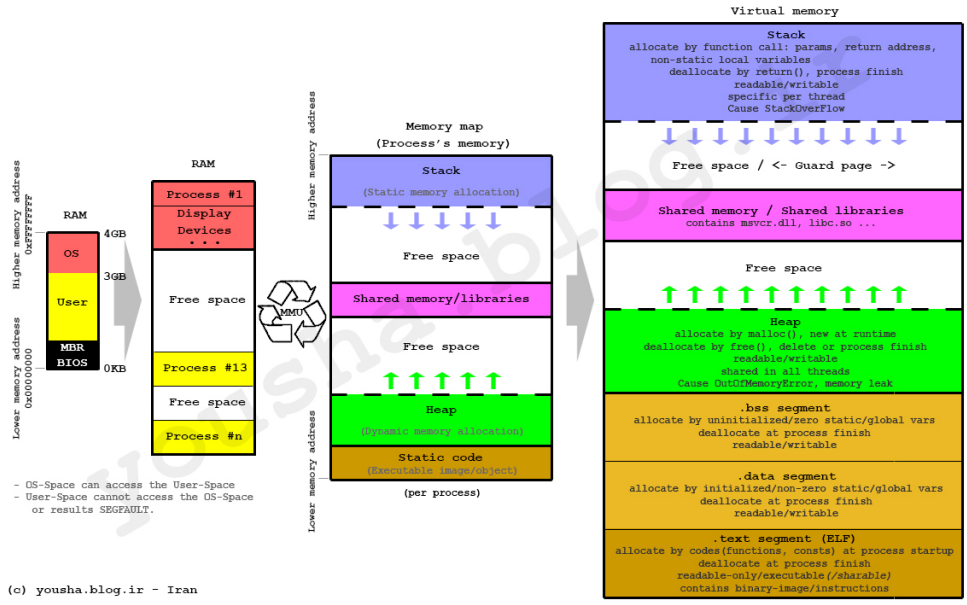
Functions of OS



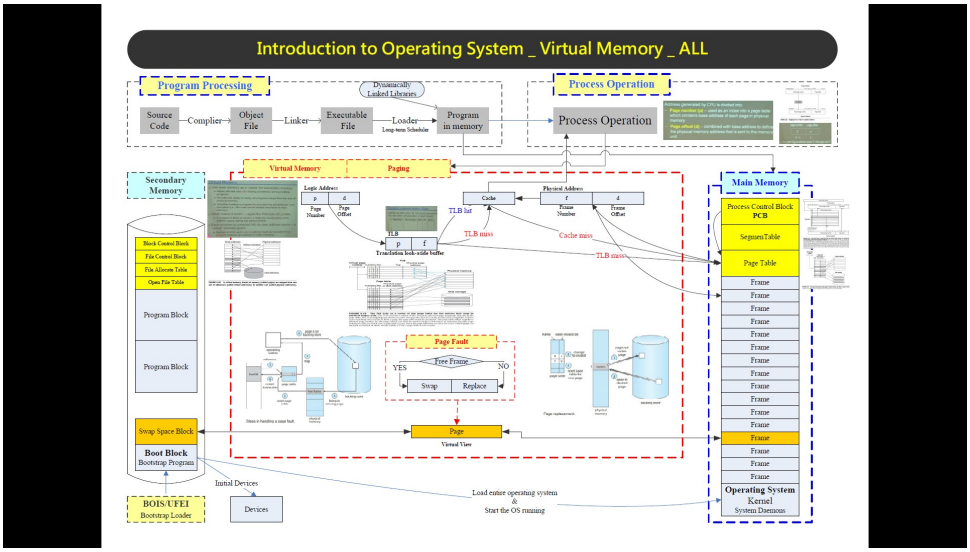
Memory Management

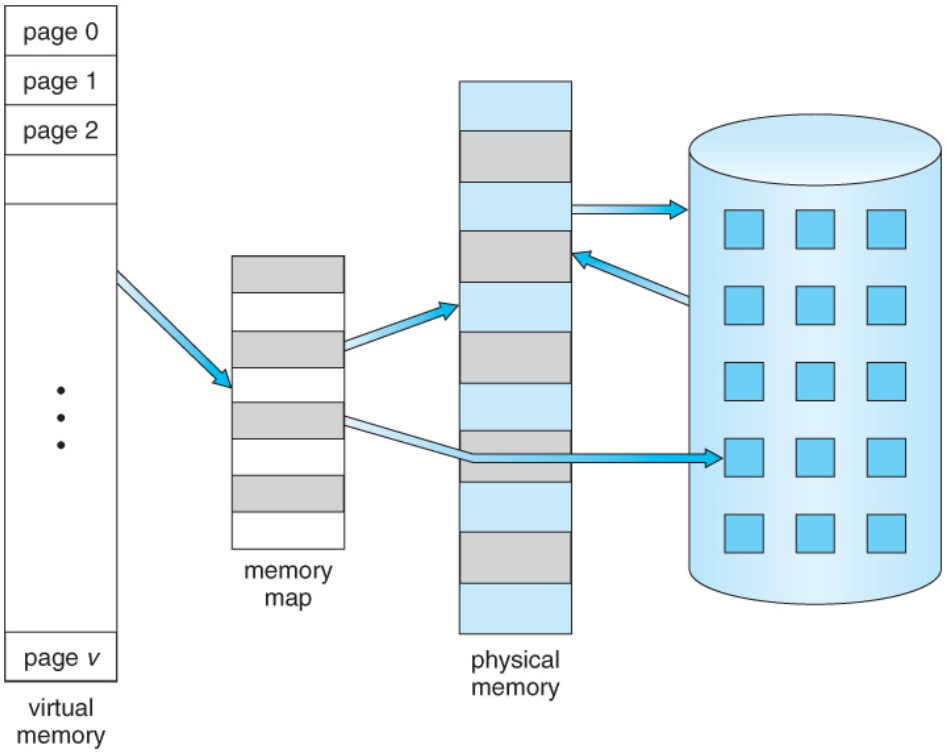


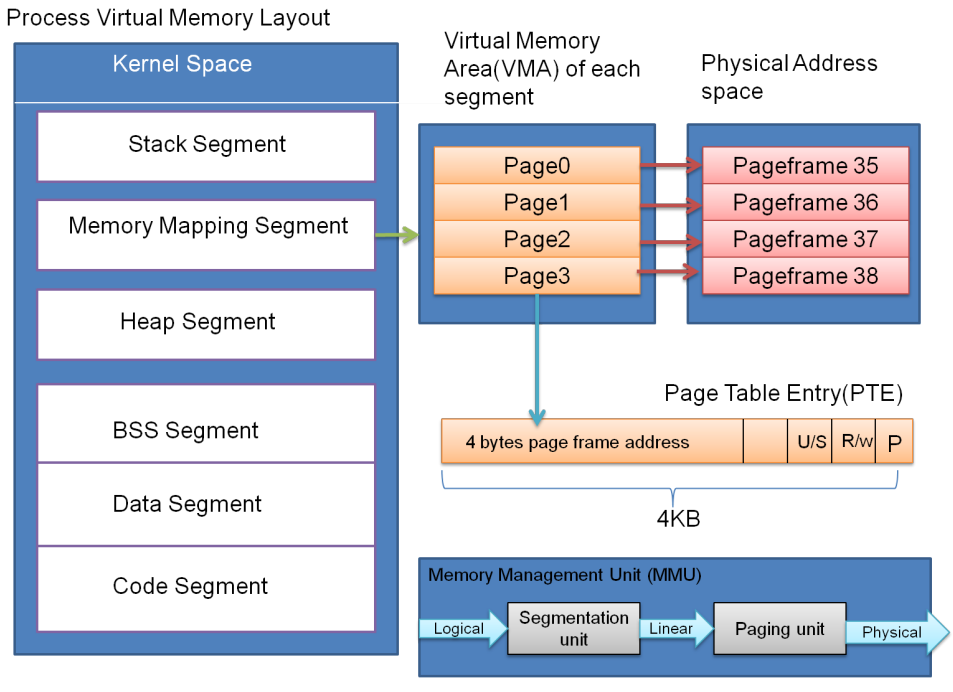




Virtual Memory

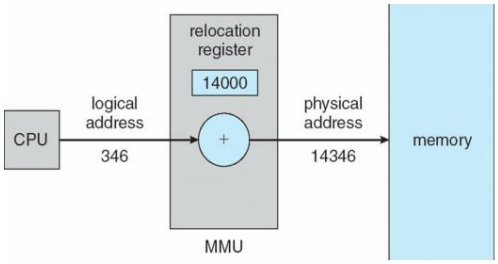



page 0
page 1
page 2
page v
virtual memory
memory map
physical memory


Process Virtual Memory Layout
Kernel Space
Stack Segment
Memory Mapping Segment
Heap Segment
BSS Segment
Data Segment
Code Segment
Virtual Memory Area(VMA) of each segment
Physical Address space
Page0
Page1
Page2
Page3
Pageframe 35
Pageframe 36
Pageframe 37
Pageframe 38
Page Table Entry(PTE)
4 bytes page frame address
U/S
R/w
P
4KB
Memory Management Unit (MMU)
Logical
Segmentation unit
Linear
Paging unit
Physical

Memory-Management Unit (MMU)

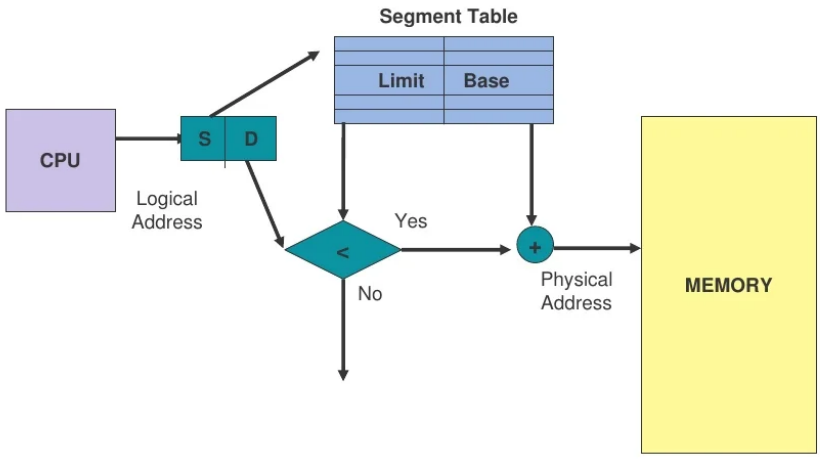
- Hardware device that maps **virtual** to **physical** address
- In MMU scheme, the value in the relocation register **is added to** every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses



Dynamic relocation using a relocation register

MEMORY MANAGEMENT Segmentation

HARDWARE -- Must map a dyad (segment / offset) into one-dimensional address.



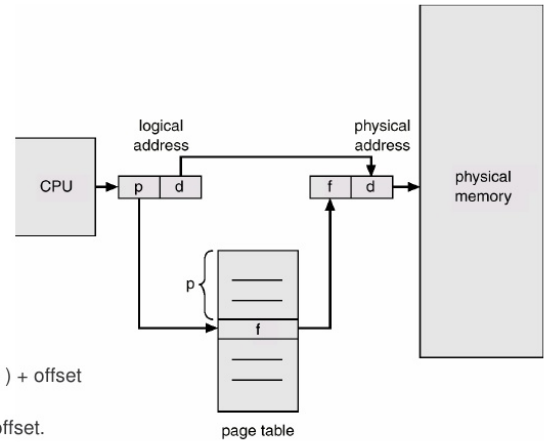
MEMORY MANAGEMENT

PAGING

Permits a program's memory to be physically noncontiguous so it can be allocated from wherever available. This avoids fragmentation and compaction.

Frames = physical blocks
Pages = logical blocks

Size of frames/pages is defined by hardware (power of 2 to ease calculations)



HARDWARE
An address is determined by:

page number (index into table) + offset
---> mapping into --->
base address (from table) + offset.

8: Memory Management **22**

Paging

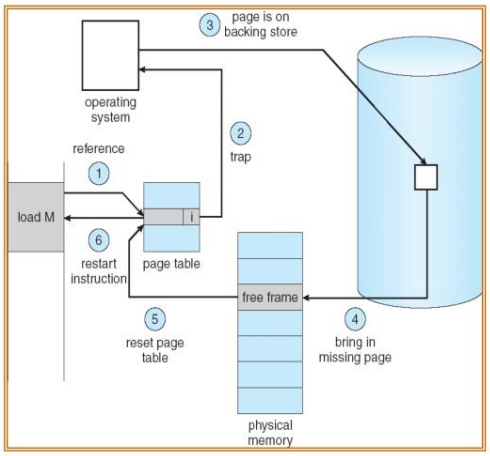
Page Fault

Page Fault

A reference to a page with valid bit set to 0 will trap to OS => page fault

OS looks at PCB to decide
- invalid reference => abort
- just no in memory
  . Get free frame
  . Swap into frame
  . Reset tables

What if there is no free frame?
- evict a victim page in memory



Page Fault Handling – a different perspective

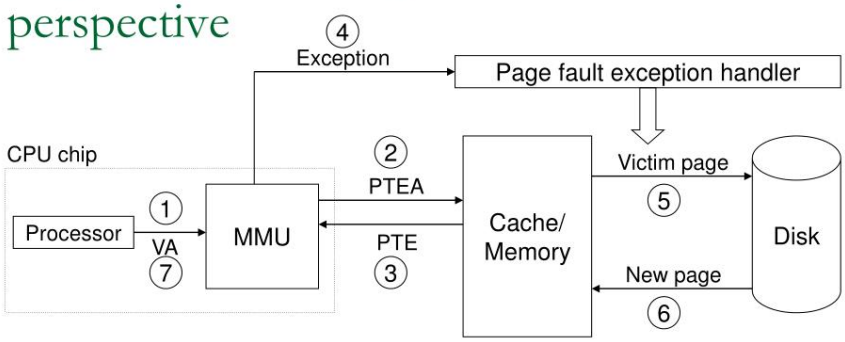


Fig. 10.14 (Bryant)

MÄLARDALENS HÖGSKOLA

Page fault handling

A page fault handling sequence:

1. Find the requested page on the disk
2. If there is free frame space in memory, jump to 4.
3. If not,
    1. Choose a page to evict from memory
    2. Adjust the page table entry for the evicted page
    3. If the evicted page is altered, write it to disk
4. Load the requested page
5. Adjust the page table
6. Let the process scheduler decide what happens next

MRTC
MÄLARDALEN REAL-TIME RESEARCH CENTRE

20

Page Fault Handling (1)

1. Hardware traps to kernel
2. General registers saved
3. OS determines which virtual page needed
4. OS checks validity of address, seeks page frame
5. If selected frame is dirty, write it to disk

32

Page Fault Handling (2)

6. OS brings new page in from disk
7. Page tables updated

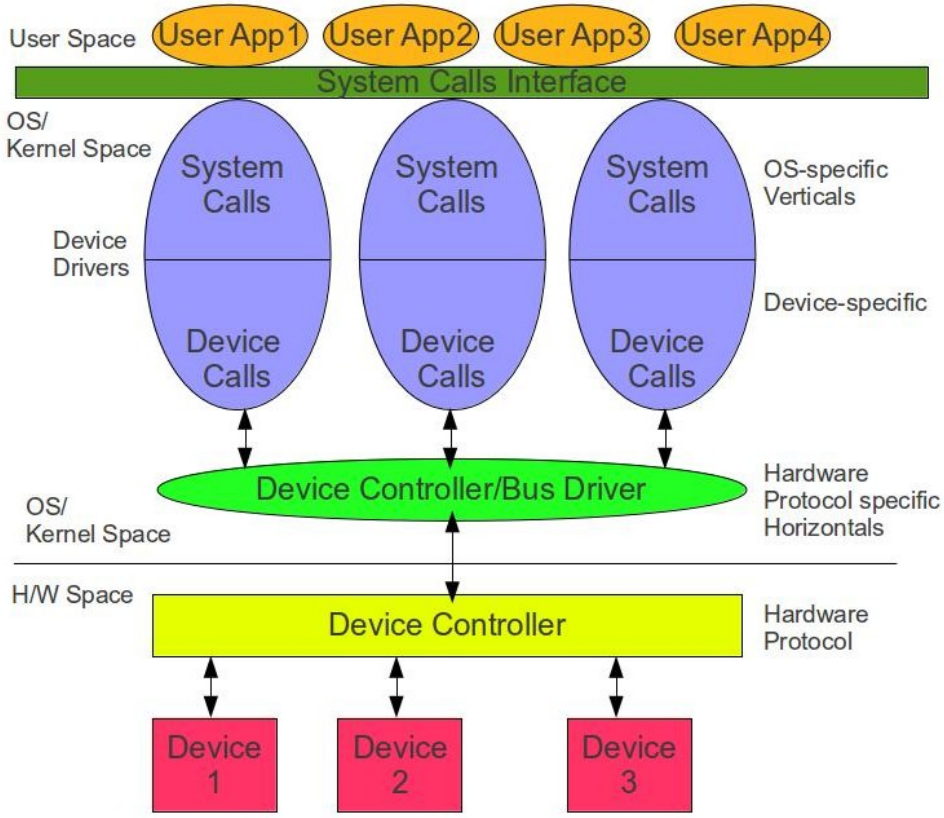
Faulting instruction backed up to when it began

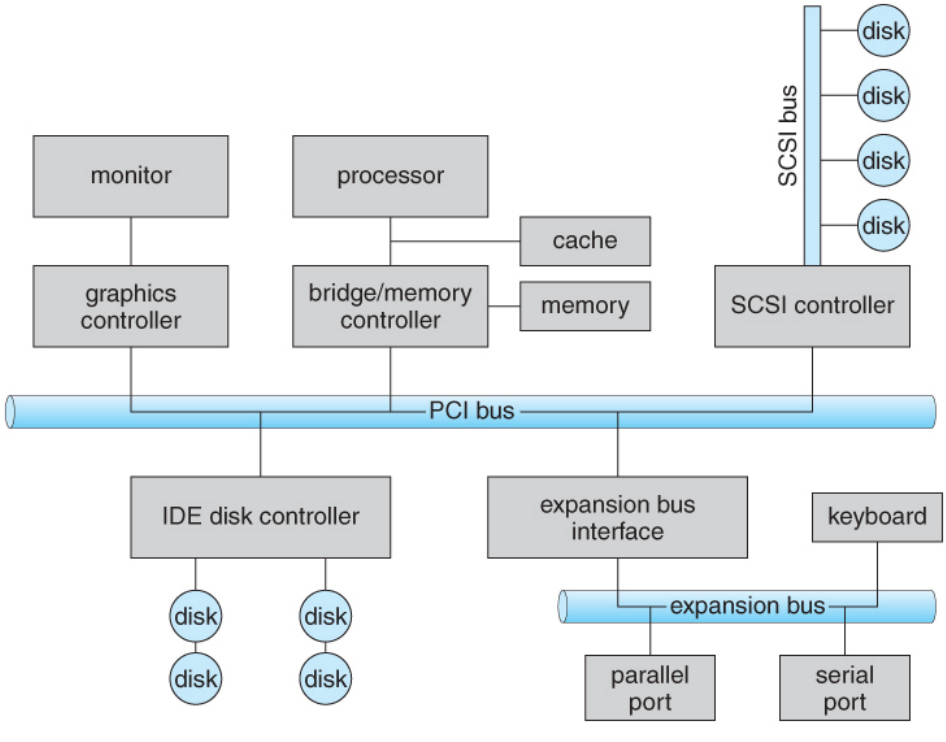
6. Faulting process scheduled
7. Registers restored

Program continues

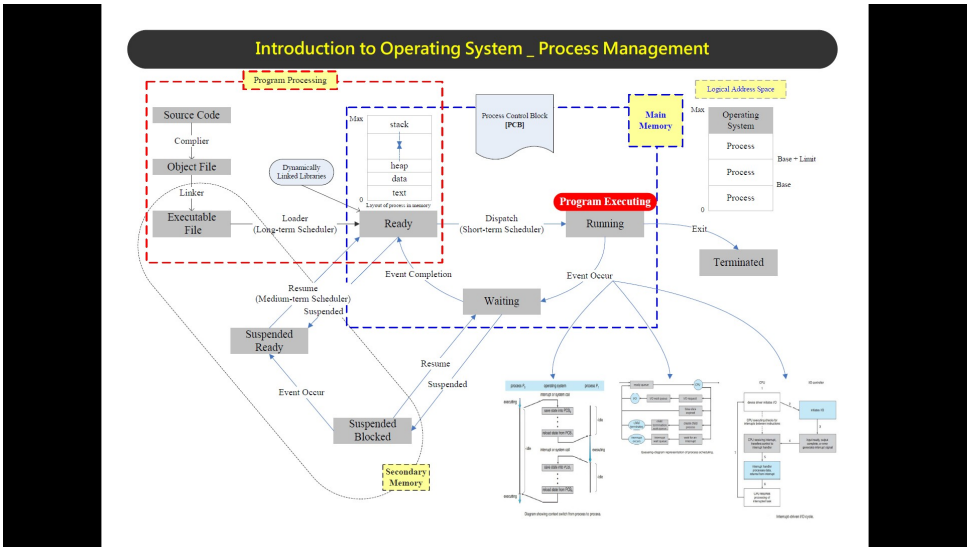
47

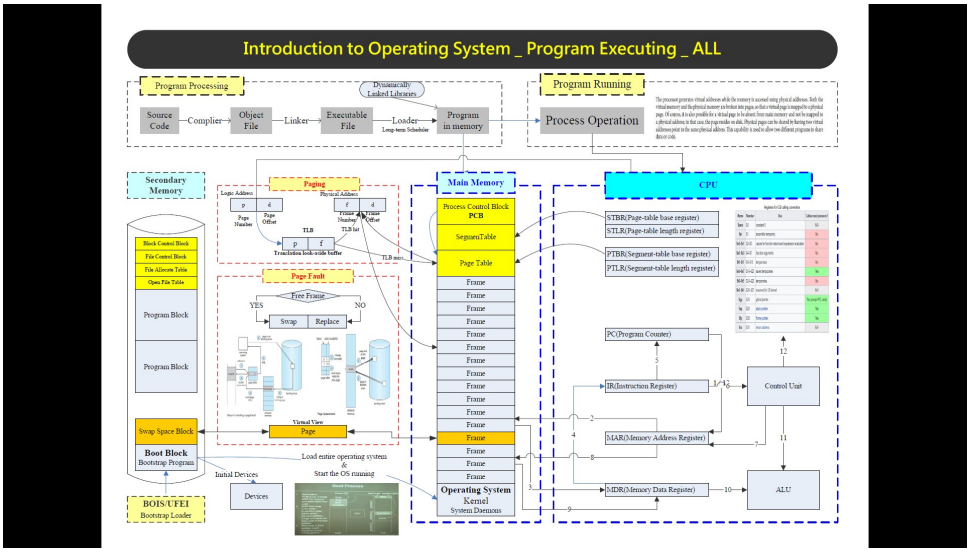
Device Mangement



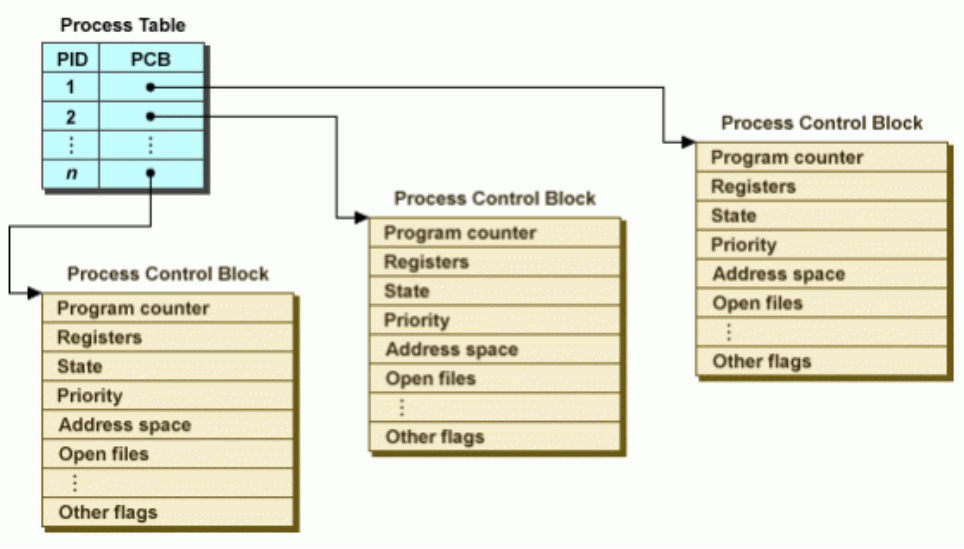


Process Management

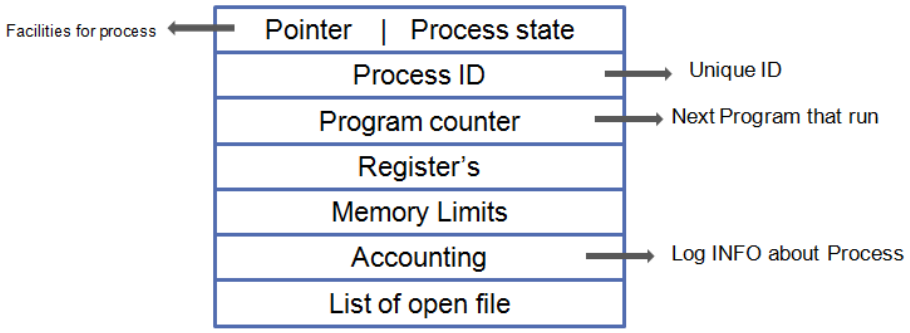




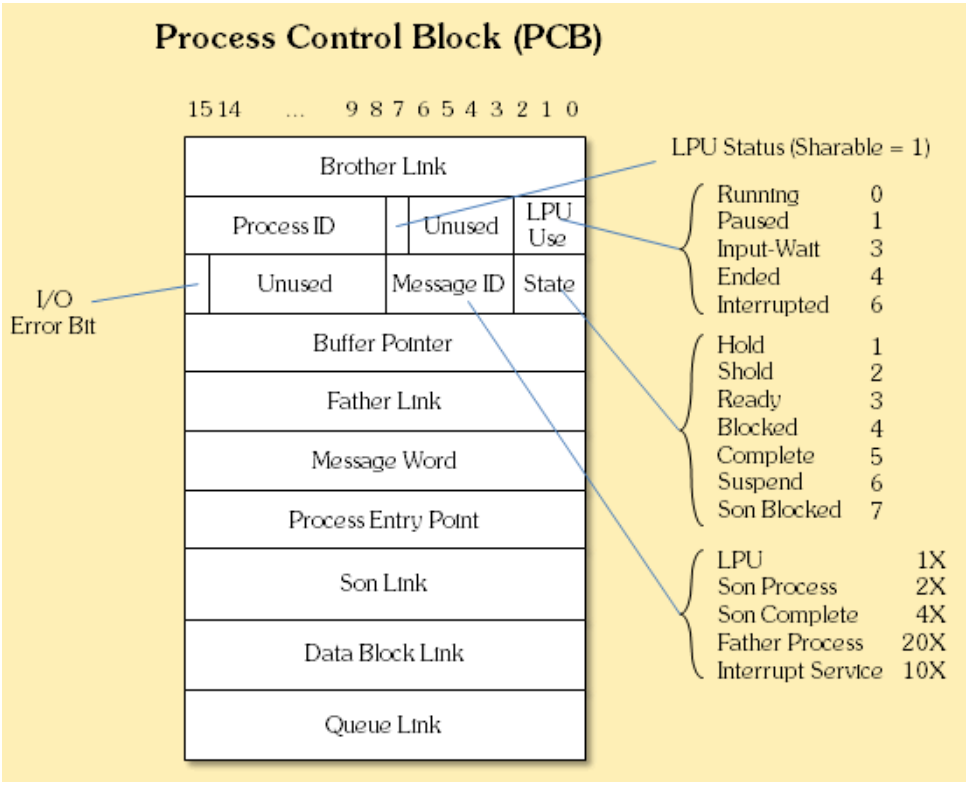
- PCB(Process Control Block)



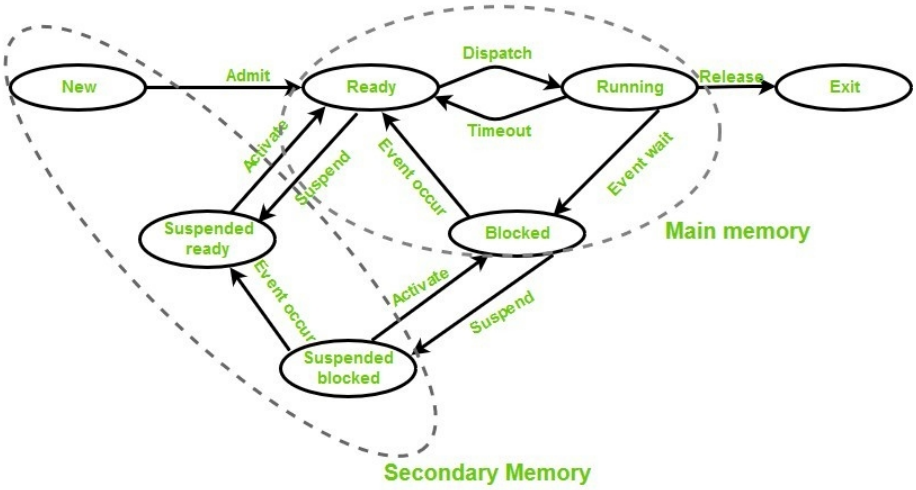
PROCESS CONTROL BLOCK (PCB)



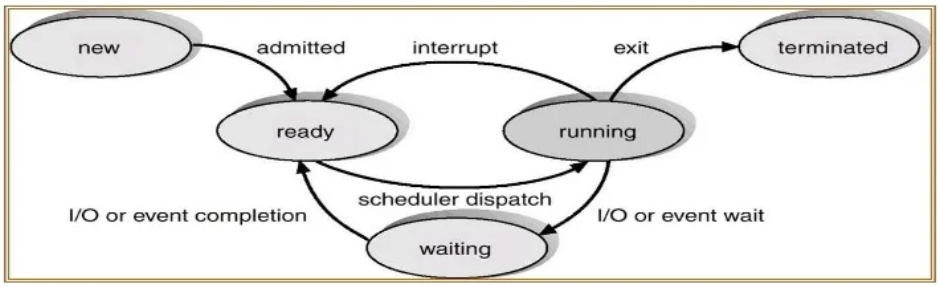
PCB Diagram



- Process State Flow

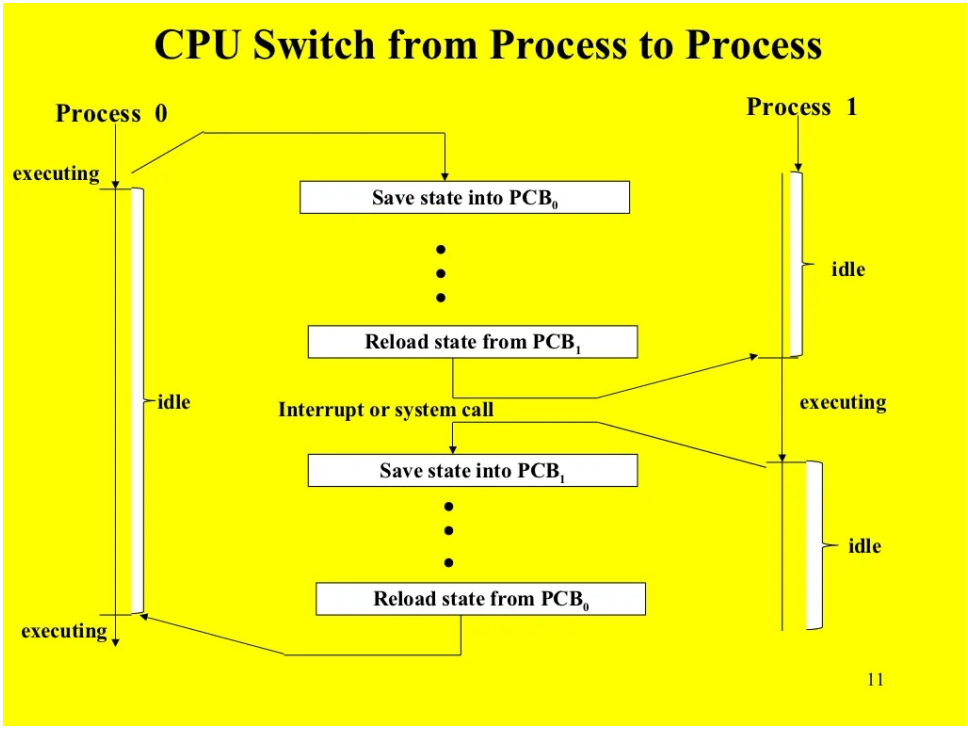


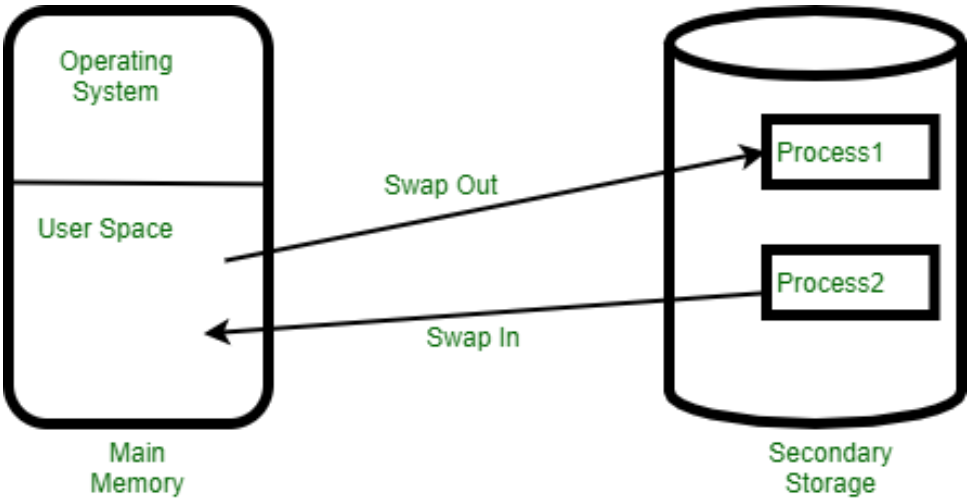
Process Management

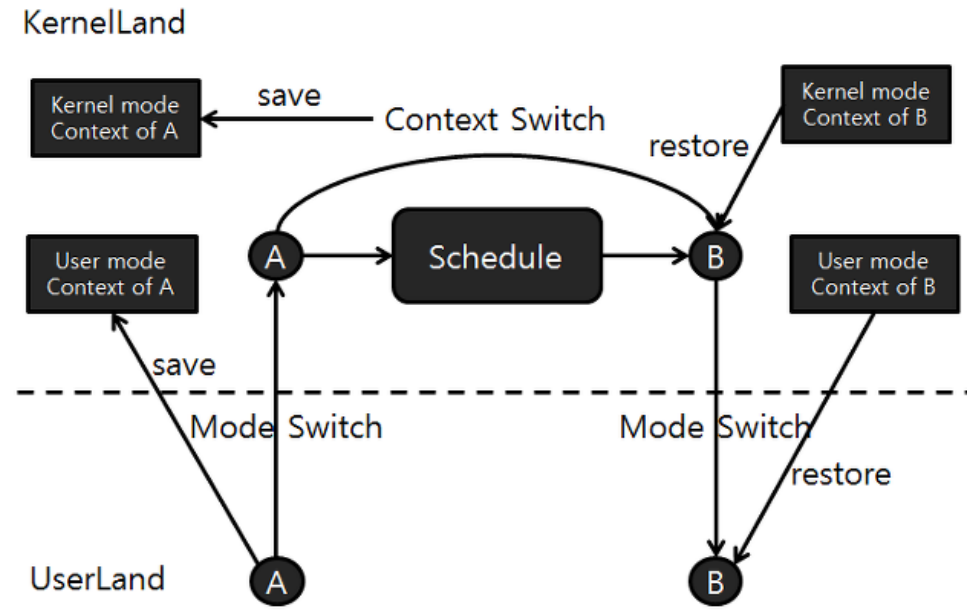


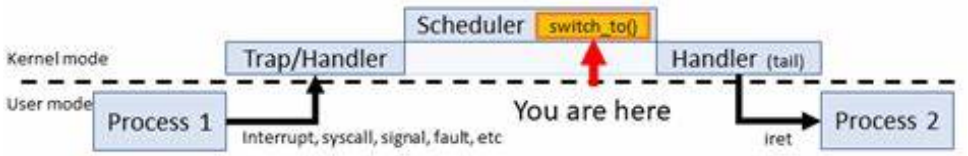
Process States
•New- The process is being created.
•Running- Instructions are being executed.
•Waiting- The process is waiting for some event to occur.
•Ready- The process is waiting to be assigned to a processor.
•Terminated- The process has finished execution.
Process Control Block- contains all information associated with a specific process like process state, program counter, CPU registers and info regarding CPU scheduling algorithms, memory, I/O and accounting.

- Context Switch and SWAP

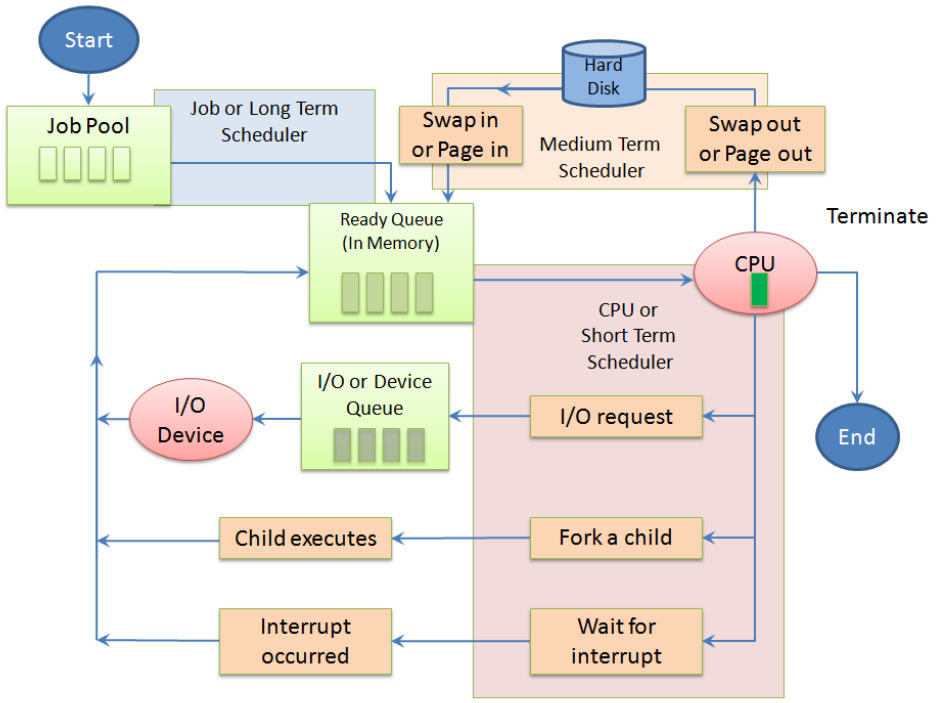

CPU Switch from Process to Process
Process 0
Process 1
executing
Save state into PCB0
idle
Reload state from PCB1
Interrupt or system call
executing
Save state into PCB1
idle
Reload state from PCB0
executing
11

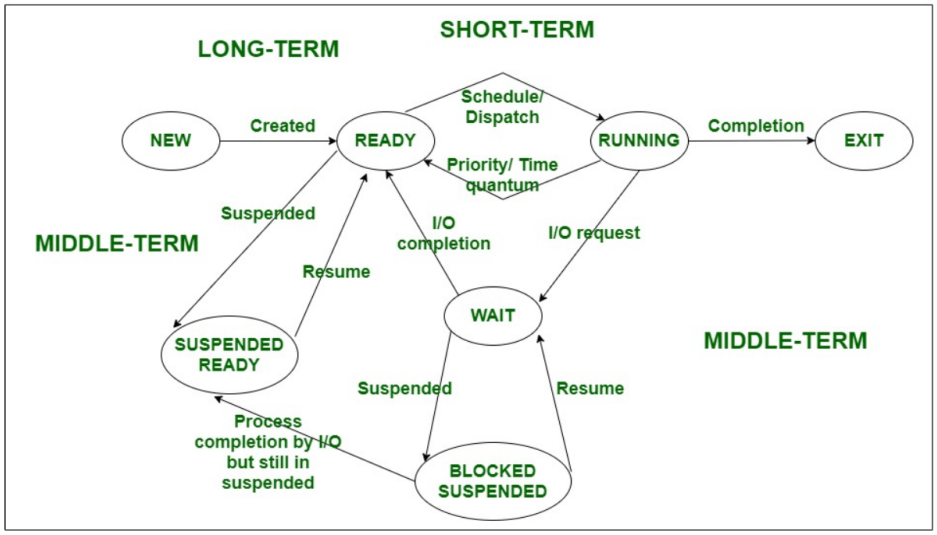

Operating System
User Space
Swap Out
Swap In
Process1
Process2
Main Memory
Secondary Storage

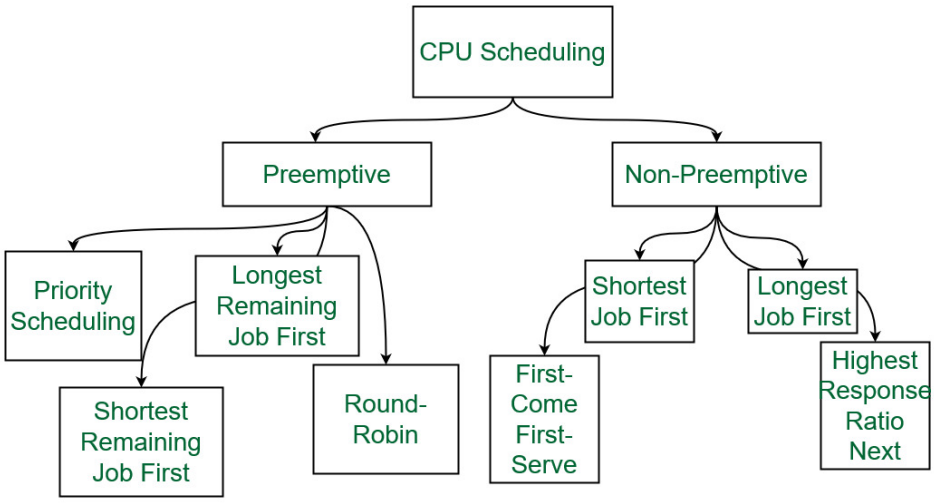

KernelLand
Kernel mode Context of A
save
Context Switch
restore
Kernel mode Context of B
User mode Context of A
A
Schedule
B
User mode Context of B
save
Mode Switch
Mode Switch
restore
UserLand
A
B


Scheduler
switch_to()
Kernel mode
Trap/Handler
Handler (tail)
User mode
Process 1
Interrupt, syscall, signal, fault, etc
You are here
iret
Process 2

- Scheduling


Start
Job Pool
Job or Long Term Scheduler
Hard Disk
Swap in or Page in
Medium Term Scheduler
Swap out or Page out
Terminate
Ready Queue (In Memory)
CPU
CPU or Short Term Scheduler
I/O Device
I/O or Device Queue
I/O request
End
Child executes
Fork a child
Interrupt occurred
Wait for interrupt


LONG-TERM
SHORT-TERM
NEW
Created
READY
Schedule/ Dispatch
RUNNING
Completion
EXIT
Priority/ Time quantum
Suspended
MIDDLE-TERM
I/O completion
I/O request
Resume
WAIT
SUSPENDED READY
MIDDLE-TERM
Suspended
Resume
Process completion by I/O but still in suspended
BLOCKED SUSPENDED


CPU Scheduling
Preemptive
Non-Preemptive
Priority Scheduling
Longest Remaining Job First
Shortest Remaining Job First
Round-Robin
Shortest Job First
Longest Job First
First-Come First-Serve
Highest Response Ratio Next

CPU SCHEDULING CRITERIA

- CPU Utilization: Percent of time that the CPU is busy executing a process.
- Throughput: Number of processes executed per unit time.
- Turnaround Time: The interval of time between submission of a process and its completion.
- Waiting Time: The amount of time the process spends in the ready queue waiting for the CPU.
- Response Time: The time between submission of requests and first response to the request.

CPU Scheduling (40 points)

Process	Burst Time	Priority	Arrival Time
P1	12	3	0
P2	6	4	2
P3	4	1	4
P4	18	2	6

Table 1: Process Information.

Consider the processes described in Table 1.

Questions: What is the average waiting time of those processes for each of the following scheduling algorithms? (Draw a Gantt chart for each algorithm.)

(a) First Come First Serve (FCFS)

(b) Non-preemptive Shortest Job First (NP-SJF)

(c) Preemptive Shortest Job First (P-SJF)

(d) Priority Scheduling

(e) Round Robin, with the following assumptions:

Assumption (1). The scheduling time quantum is 5 time units.

Assumption (2). If a new process arrives at the same time as the time slice of the executing process expires, the OS puts the executing process in the ready queue, followed by the new process.

Question 3: Consider the following set of processes, their arrival times, length of the CPU burst time given in milliseconds:

Process	Arrival Time	Burst Time	Priority
P1	1	12	3
P2	2	3	1
P3	2	16	4
P4	3	10	2
P5	7	1	1

Let us schedule the execution of these processes using the following scheduling algorithms: First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), and a new type of scheduling algorithm called non-preemptive priority scheduling. The following are the assumptions:

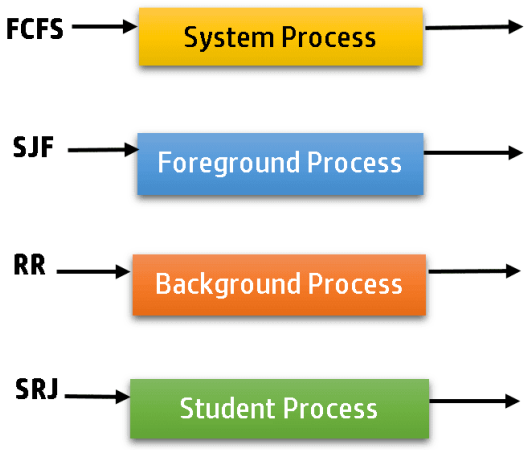
1) Larger priority number implies higher priority
2) Assume the quantum (aka time slice) of 2 for RR algorithm
3) Non-Preemptive means once scheduled, a process cannot be preempted (i.e. it runs to completion).

3a [10 points] What is the response (completion) time of each process for each of the scheduling algorithms. Write the times in the table below.

	FCFS	SJF	Priority	RR
P1				
P2				
P3				
P4				
P5				

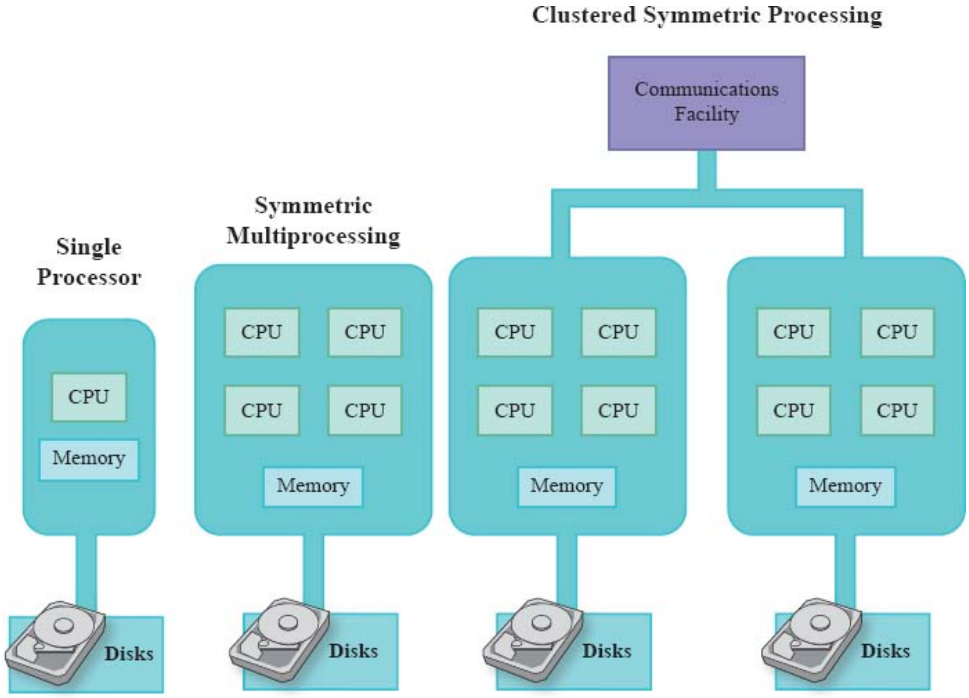
3b [10 points] Which of the above methods results in the longest average wait time. Show calculations and explain.

Highest Priority

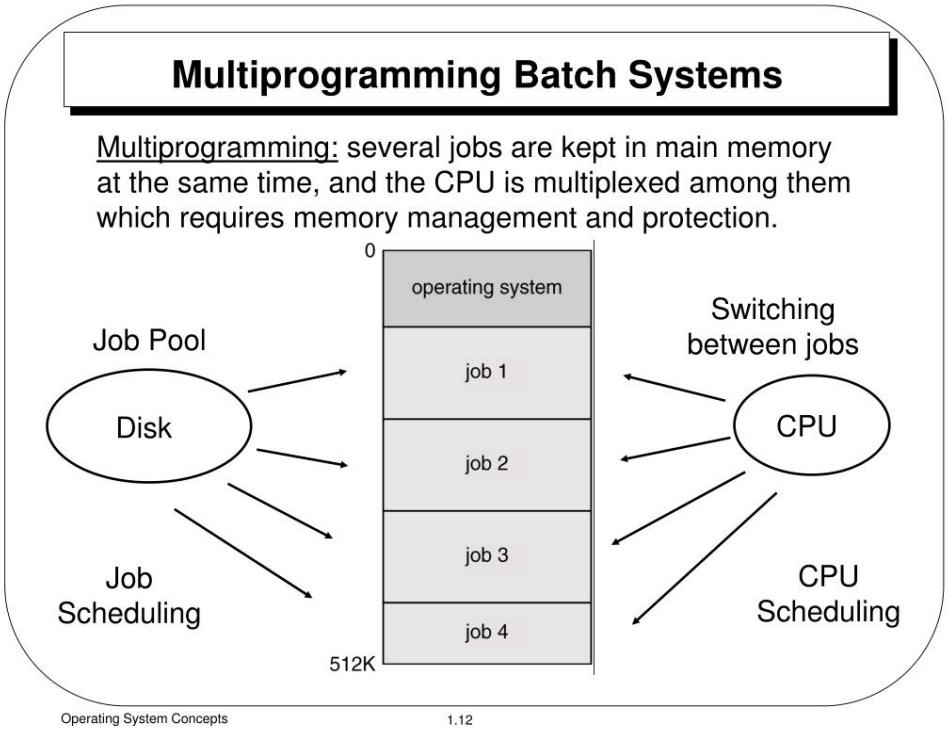


Lowest Priority

- SP vs MP vs Cluster



- Multiprogramming

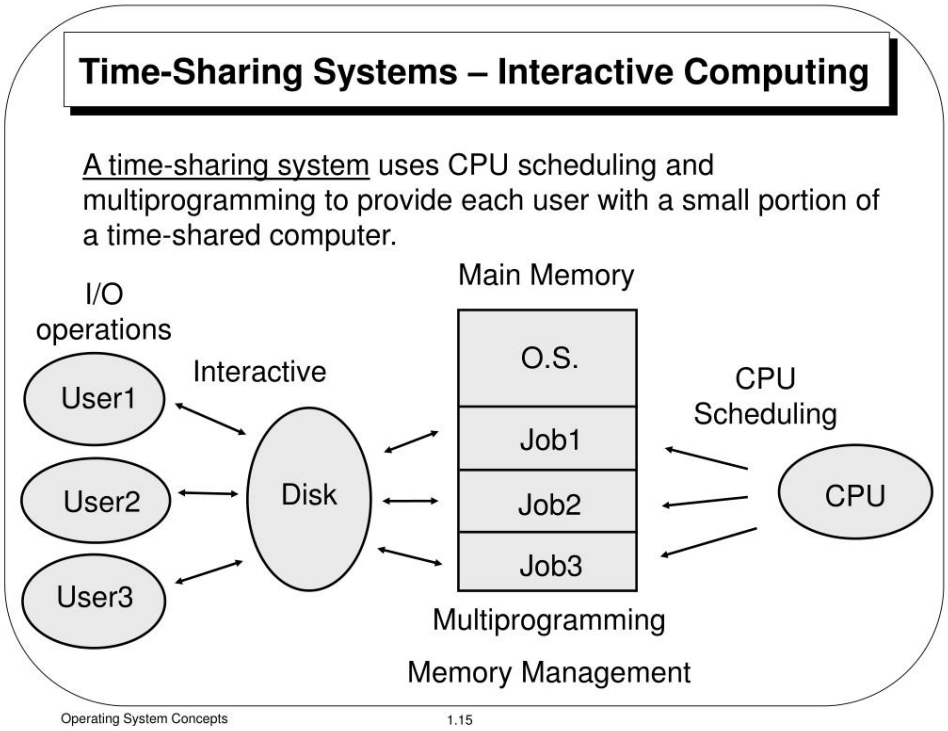


MULTIPROGRAMMING VERSUS MULTITASKING

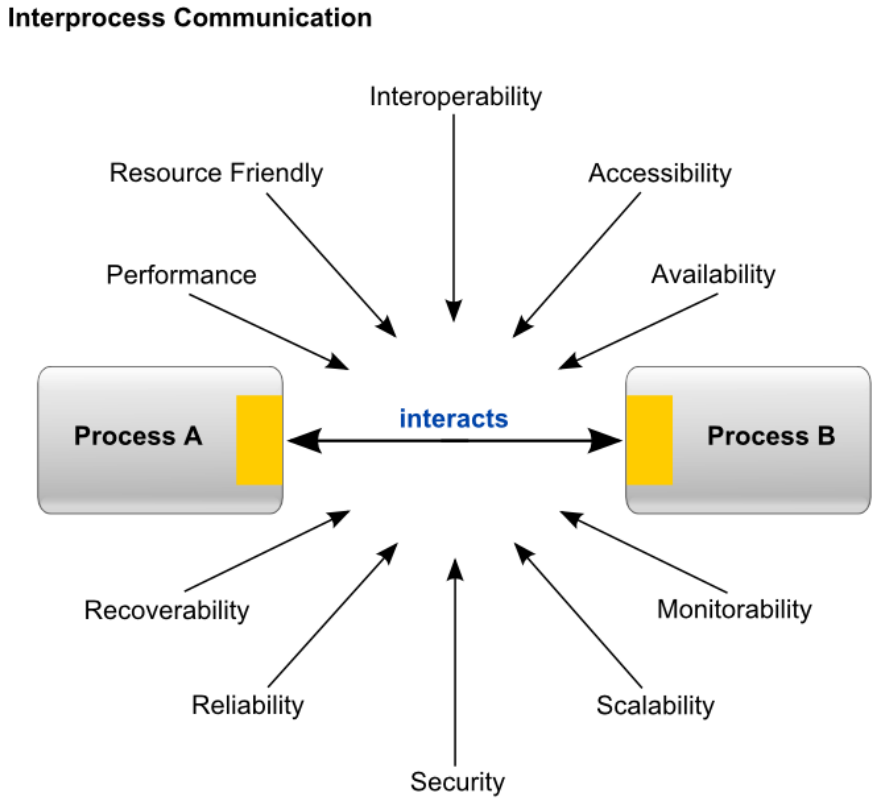
Multiprogramming	Multitasking
In multiprogramming, multiple processes run concurrently at the same time on a single processor.	Multitasking is when more than one task is executed at a single time utilizing multiple CPUs
It is based on the concept of context switching.	It is based on the concept of time sharing.
Multiple programs reside in the main memory simultaneously to improve CPU utilization so that CPU doesn't sit idle for a long time.	It enables execution of multiple tasks and processes at the same time to increase CPU performance.
It utilizes single CPU for execution of processes.	It utilizes multiple CPUs for task allocation.
It takes more time to execute the processes.	It takes less time to execute the tasks or processes.
The idea is to reduce the CPU idle time for as long as possible.	The idea is to allow multiple processes to run simultaneously via time sharing.

DB Difference Between.net

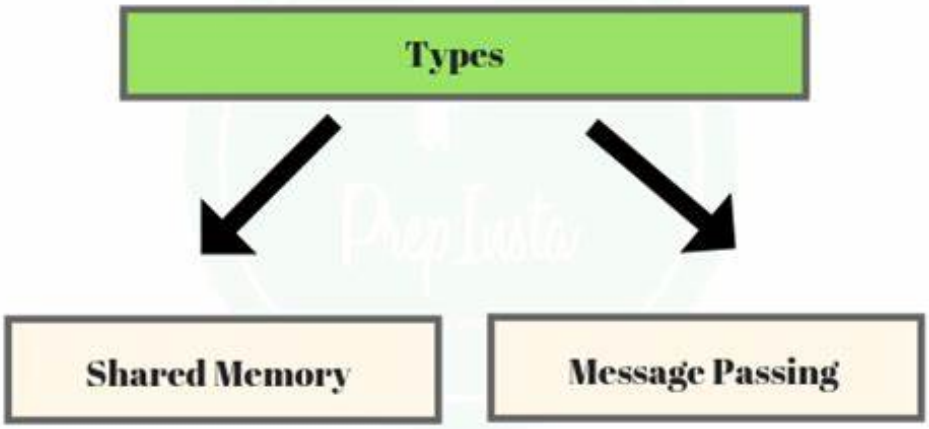
- Time Sharing



Interprocess Communication(IPC)

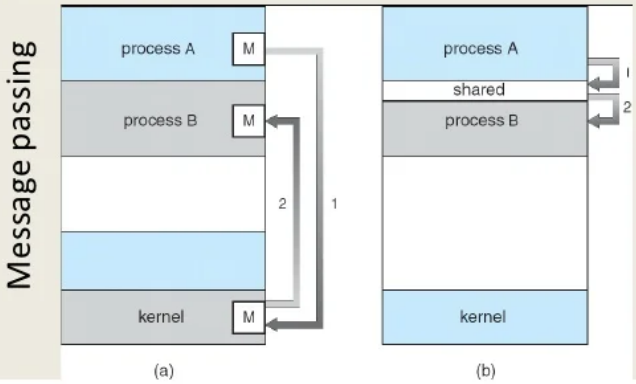


Inter-Process Communication Types



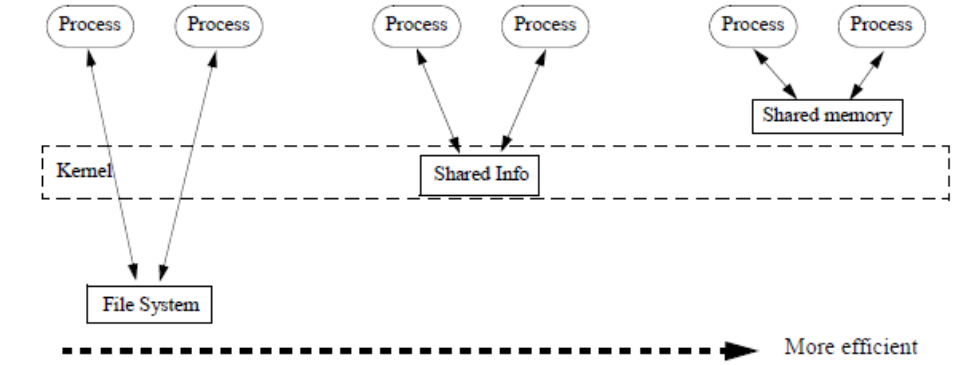
4. Interprocess Communication Contd...

- Two fundamental models of Interprocess communication
- **Shared memory**
  - a region of memory that is shared by cooperating processes is established then exchange information takes place by reading and writing data to the shared region
- **Message passing**
  - communication takes place by means of messages exchanged between the cooperating processes



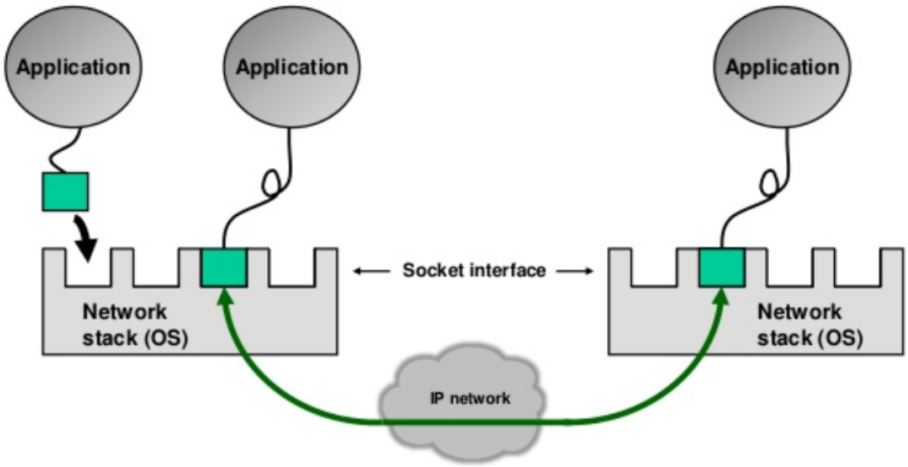
IPC Mechanisms

IPC classified by implementation mechanisms.

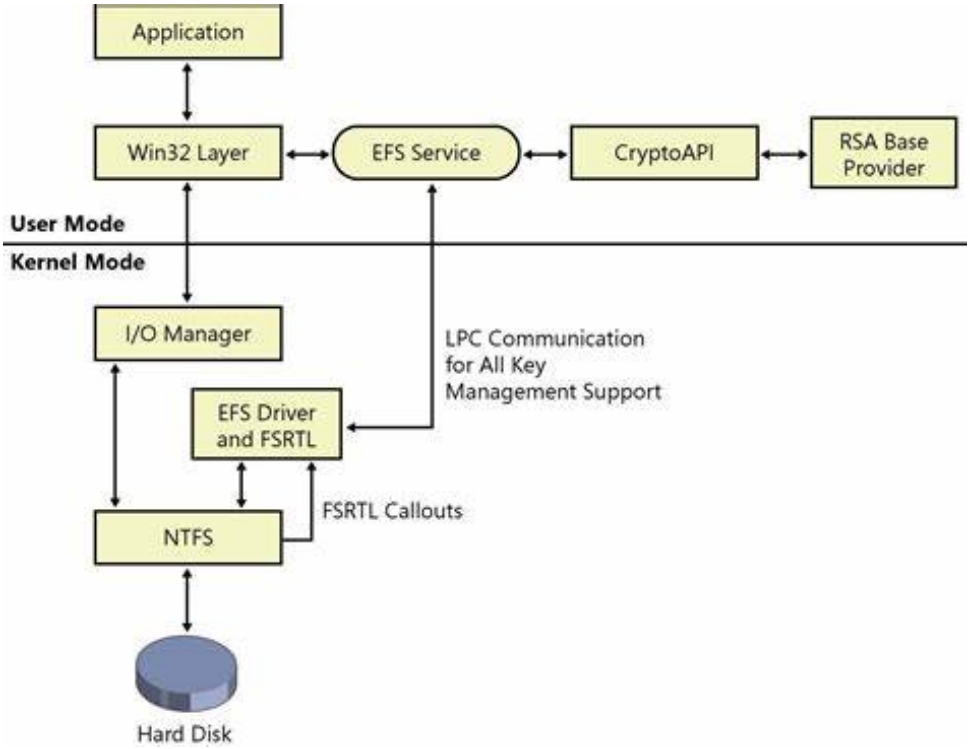


Examples:

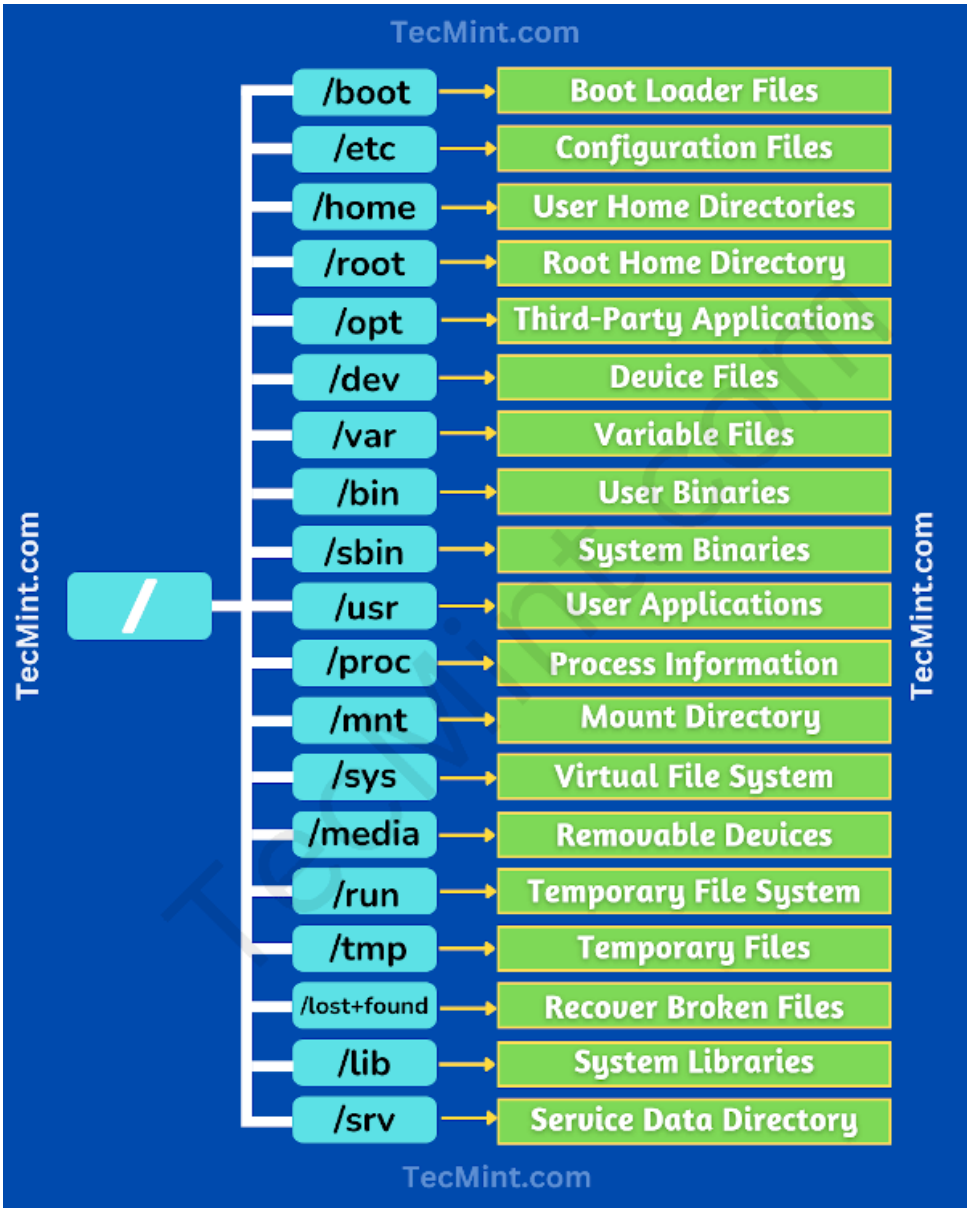
files | semaphore, socket, pipe, message queue, signal | shared memory

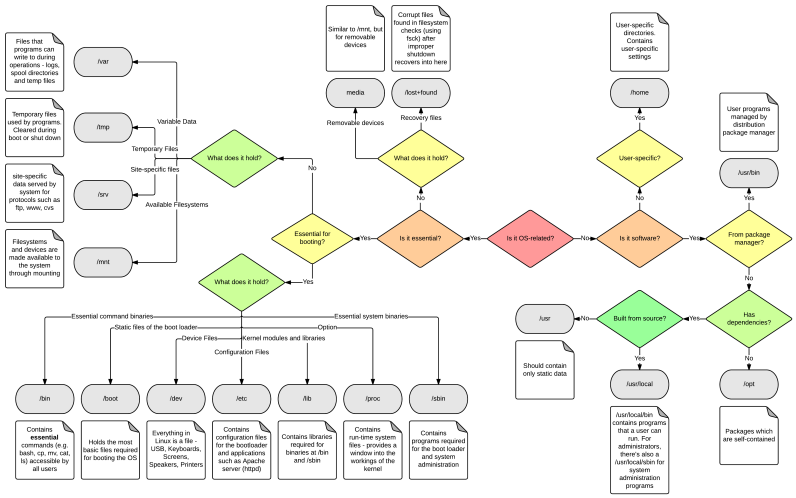


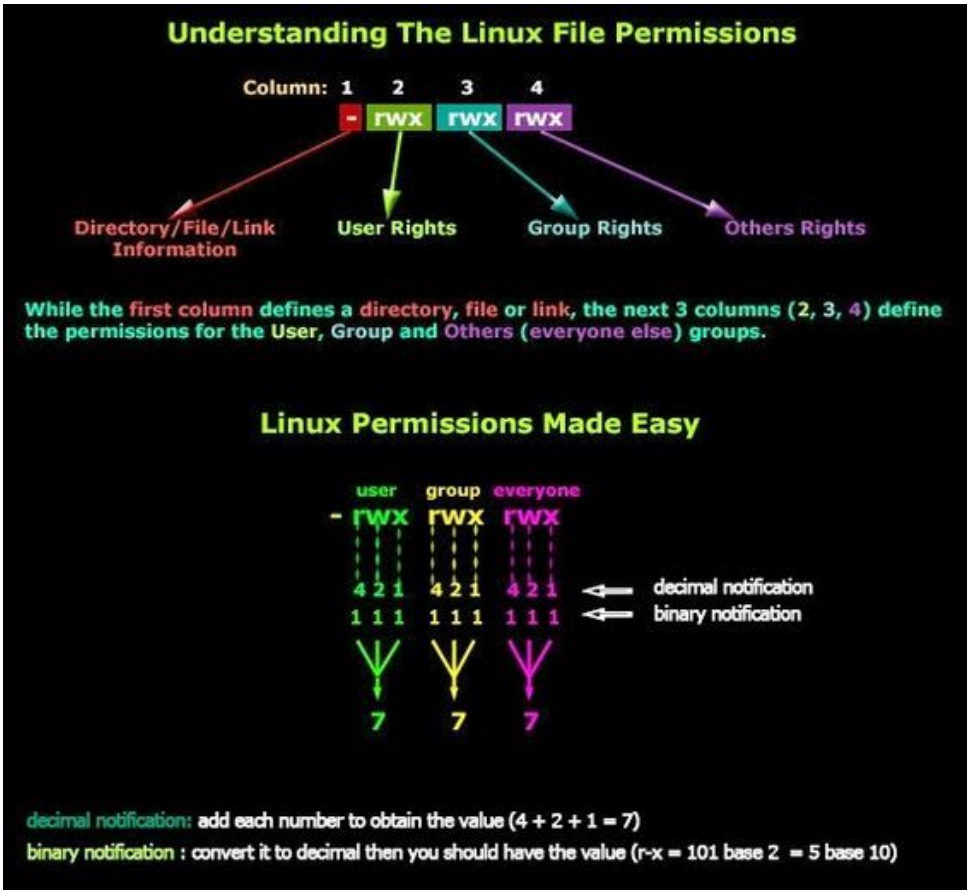
File Management

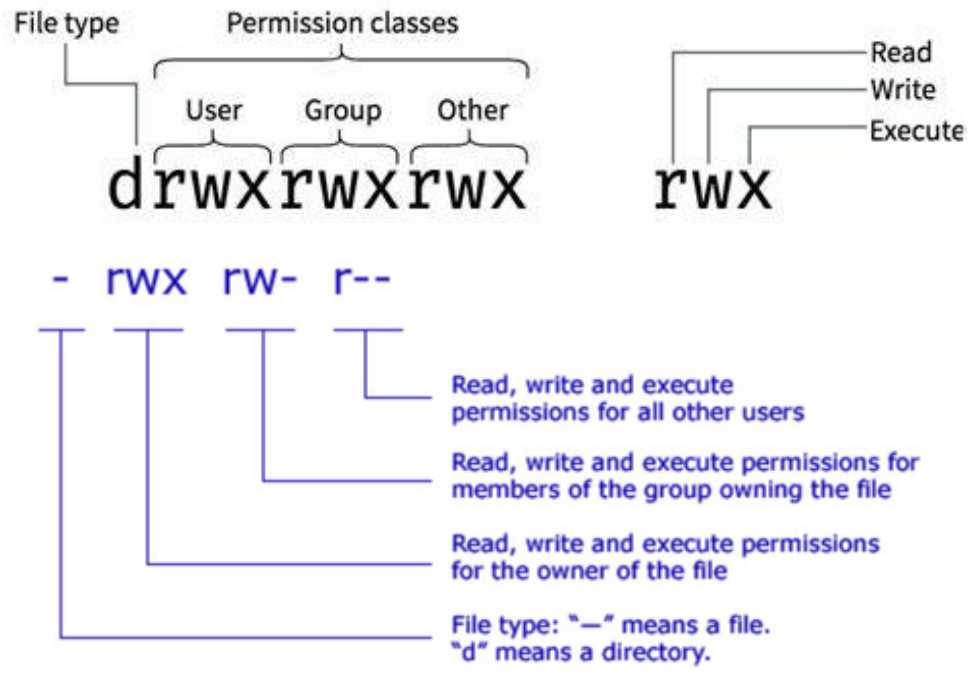


File Structure in Linux

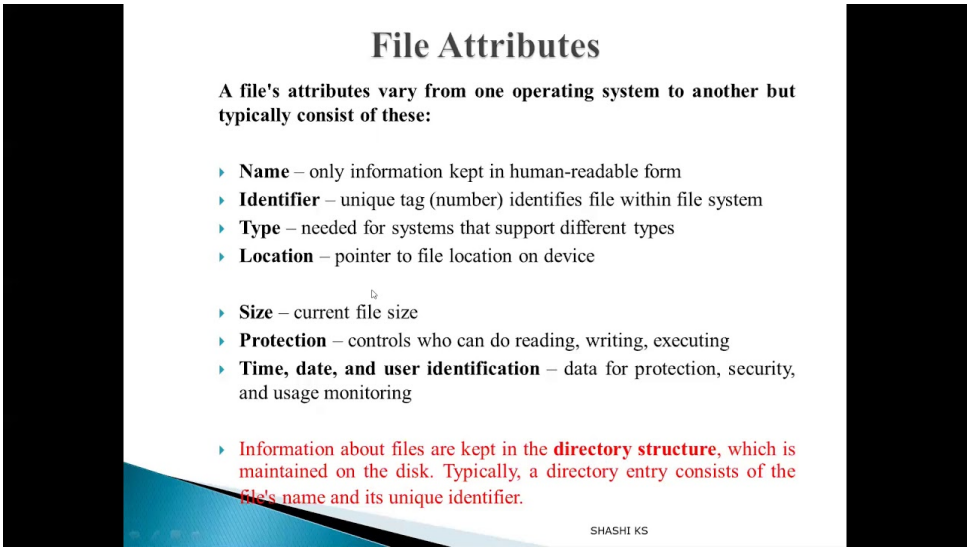

TecMint.com
/
/boot
Boot Loader Files
/etc
Configuration Files
/home
User Home Directories
/root
Root Home Directory
/opt
Third-Party Applications
/dev
Device Files
/var
Variable Files
/bin
User Binaries
/sbin
System Binaries
/usr
User Applications
/proc
Process Information
/mnt
Mount Directory
/sys
Virtual File System
/media
Removable Devices
/run
Temporary File System
/tmp
Temporary Files
/lost+found
Recover Broken Files
/lib
System Libraries
/srv
Service Data Directory
TecMint.com
TecMint.com
TecMint.com


/var
/tmp
/srv
/mnt
Variable Data
Temporary Files
Site-specific files
Available Filesystems
What does it hold?
Essential for booting?
Is it essential?
Is it OS-related?
Is it software?
From package manager?
Has dependencies?
Built from source?
User-specific?
/home
/usr
/usr/bin
/usr/local
/opt
media
/lost+found
Removable devices
Recovery files
Essential command binaries
Essential system binaries
Static files of the boot loader
Device Files
Kernel modules and libraries
Configuration Files
Option
/bin
/boot
/dev
/etc
/lib
/proc
/sbin





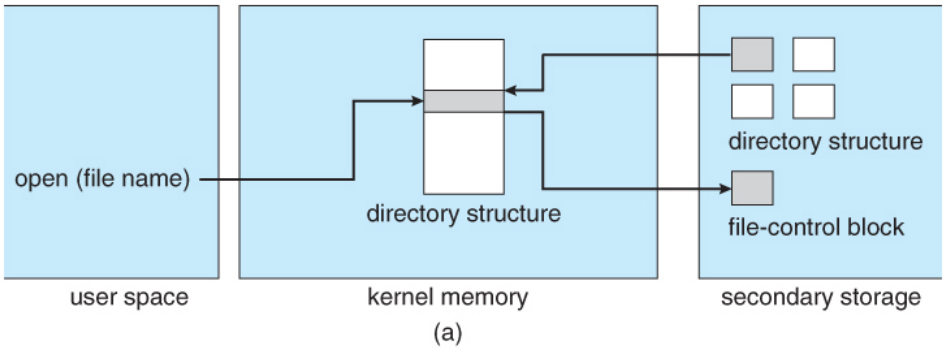
File Attributes



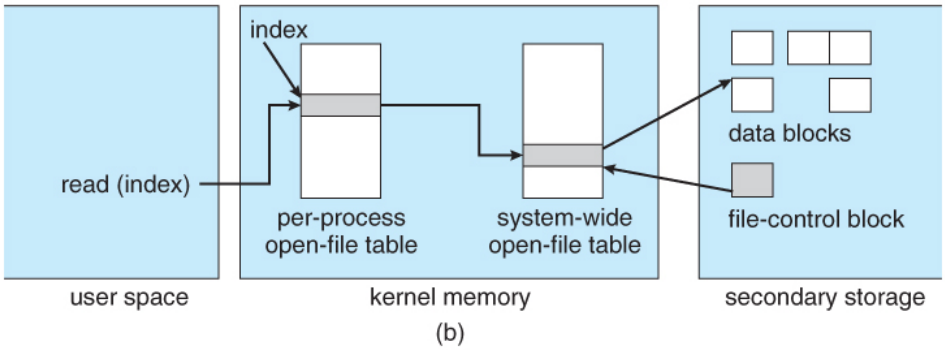
File Type

File type	Usual extension	Function
Executable	exe,com,bin	Read to run machine language program
Object	obj,o	Compiled,machine language not linked
Source code	C,java,pas,asm,a	Source code in various languages
Batch	bat,sh	Commands to the command interpreter
Text	txt,doc	Textual data,documents
Word processor	Wp,tex,rrf,doc	Various word processor formats
Archive	arc,zip,tar	Related files grouped into one file compressed

File Access



(a)

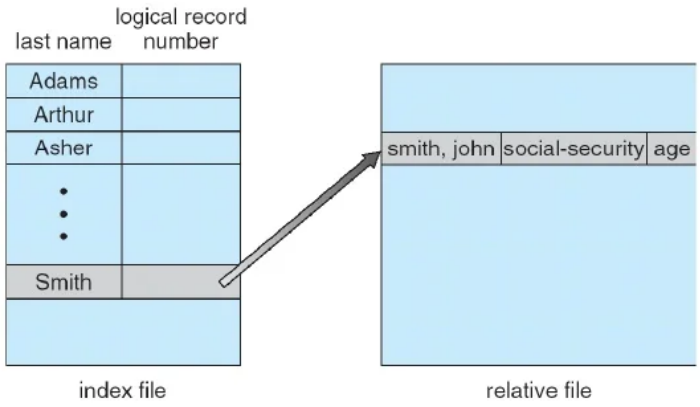


(b)

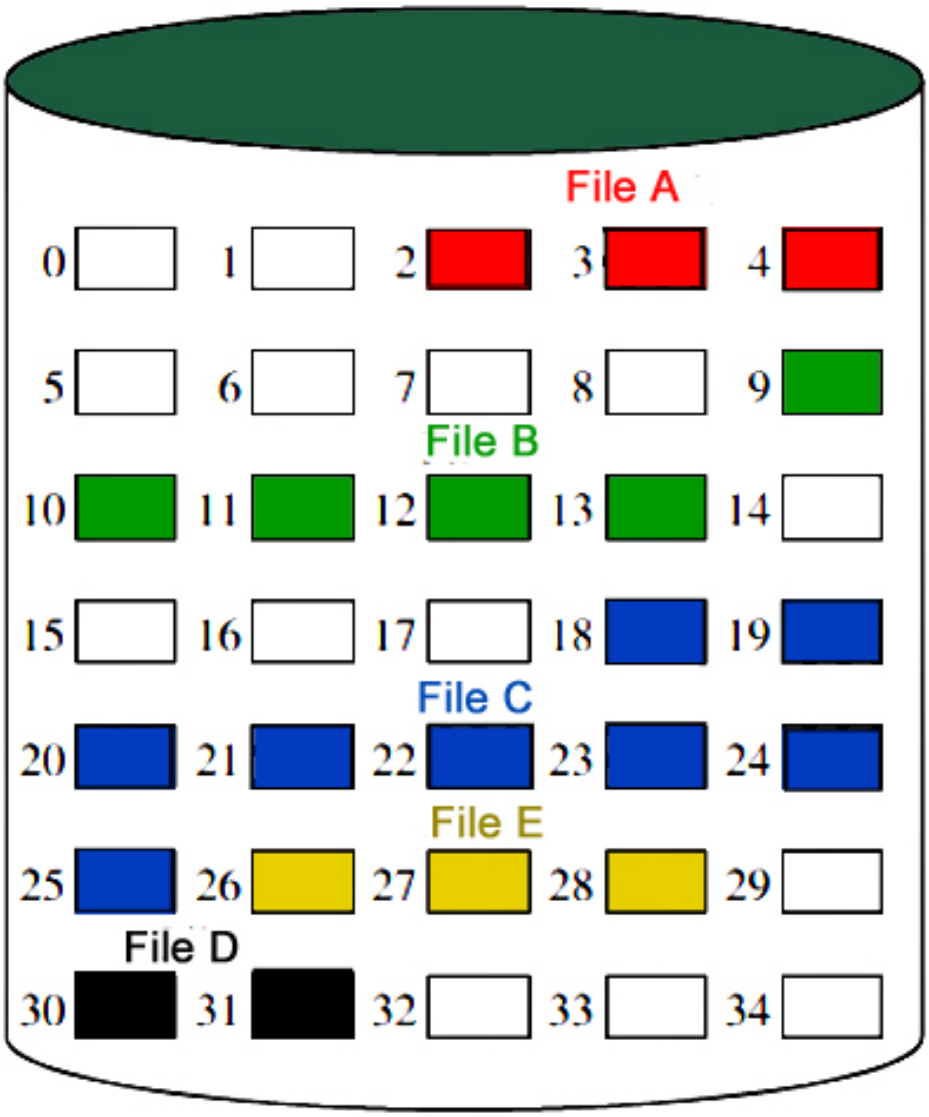
2. Access Methods Contd...

2.3 Other Access Methods

- Built on top of a direct-access method
- Example of Index and Relative Files



Loganathan R, CSE, HKBKCE 9



File allocation table

File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Disk Scheduling Algorithm

Selecting a Disk-Scheduling Algorithm

- Simulation results
  - low load --> SCAN
  - medium to heavy load --> C-SCAN
- influenced by the file-allocation method
  - contiguously allocated file
  - a linked or indexed file
- the location of directories and index blocks
  - placing the directories halfway between the inner and outer edge of the disk
  - placing the directories at either end

SunMoon University 19

Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

16

Disk Scheduling Algorithms

Selection according to requestor		
RSS	Random scheduling	For analysis & simulation
FIFO	First in first out	Fairest of them all
Priority	Priority by process	No disk optimization
LIFO	Last in first out	Max locality & resource
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of N records at a time	Service guarantee
FSCAN	NsS w/N=queue at beginning of SCAN cycle	Load sensitive

Example

Trace the policies FIFO, SSTF, SCAN, C-SCAN and FSCAN for the following disk requests. Each I/O request on a track takes 5 time units. At time 0, the disk starts reading track 10, and the read/write head was moving to the larger track number direction .

Time	0	1	2	3	6	7
Request to access track ..	10	19	3	14	12	9

	Track access order	Average seek length
FIFO	10,19,3,14,12,9	(9+16+11+2+3)/5 = 8.2
SSTF	10,14,12,9,3,19	(4+2+3+6+16)/5 = 6.2
SCAN	10,14,19,12,9,3	(4+5+7+3+6)/5 = 5
C-SCAN	10,14,19,3,9,12	(4+5+16+6+3)/5 = 6.8
FSCAN	10,14,19,3,9,12	(4+5+16+6+3)/5 = 6.8

11

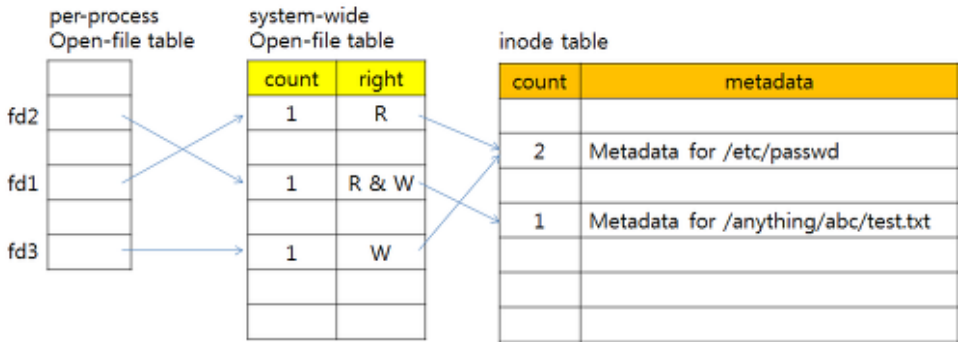
Disk Scheduling Algorithms

Table 11.2 Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

38

File Operations



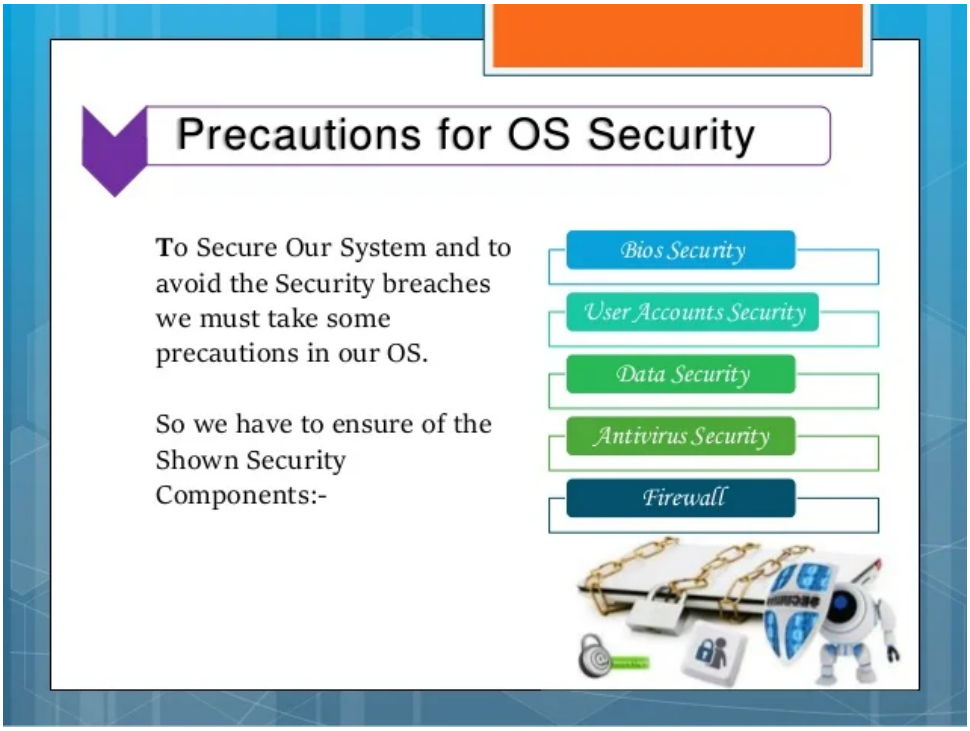
File operation	Declaration & Description
fopen() – To open a file	Declaration: FILE *fopen (const char *filename, const char *mode) fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist. FILE *fp; fp=fopen (“filename”, ”‘mode”); Where, fp – file pointer to the data type “FILE”. filename – the actual file name with full path of the file. mode – refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.
fclose() – To close a file	Declaration: int fclose(FILE *fp); fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below. fclose (fp);
fgets() – To read a file	Declaration: char *fgets(char *string, int n, FILE *fp) fgets function is used to read a file line by line. In a C program, we use fgets function as below. fgets (buffer, size, fp); where, buffer – buffer to put the data in. size – size of the buffer fp – file pointer
fprintf() – To write into a file	Declaration: int fprintf(FILE *fp, const char *format, ...);fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, “some data”); or fprintf (fp, “text %d”, variable_name);

File operation	Declaration & Description
fopen() – To open a file	Declaration: FILE *fopen (const char *filename, const char *mode) fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist. FILE *fp; fp=fopen (“filename”, ”‘mode”); Where, fp – file pointer to the data type “FILE”. filename – the actual file name with full path of the file. mode – refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.
fclose() – To close a file	Declaration: int fclose(FILE *fp); fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below. fclose (fp);
fgets() – To read a file	Declaration: char *fgets(char *string, int n, FILE *fp) fgets function is used to read a file line by line. In a C program, we use fgets function as below. fgets (buffer, size, fp); where, buffer – buffer to put the data in. size – size of the buffer fp – file pointer
fprintf() – To write into a file	Declaration: int fprintf(FILE *fp, const char *format, …);fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, “some data”); or fprintf (fp, “text %d”, variable_name);

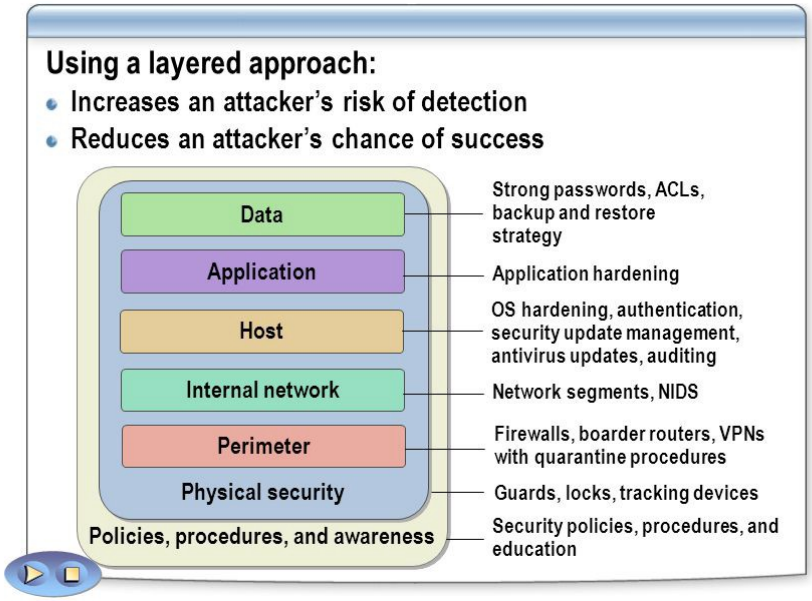
File operation	Declaration & Description
fopen() – To open a file	Declaration: FILE *fopen (const char *filename, const char *mode) fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist. FILE *fp; fp=fopen (“filename”, ”‘mode”); Where, fp – file pointer to the data type “FILE”. filename – the actual file name with full path of the file. mode – refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.
fclose() – To close a file	Declaration: int fclose(FILE *fp); fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below. fclose (fp);
fgets() – To read a file	Declaration: char *fgets(char *string, int n, FILE *fp) fgets function is used to read a file line by line. In a C program, we use fgets function as below. fgets (buffer, size, fp); where, buffer – buffer to put the data in. size – size of the buffer fp – file pointer
fprintf() – To write into a file	Declaration: int fprintf(FILE *fp, const char *format, …);fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, “some data”); or fprintf (fp, “text %d”, variable_name);

'

- OS Security



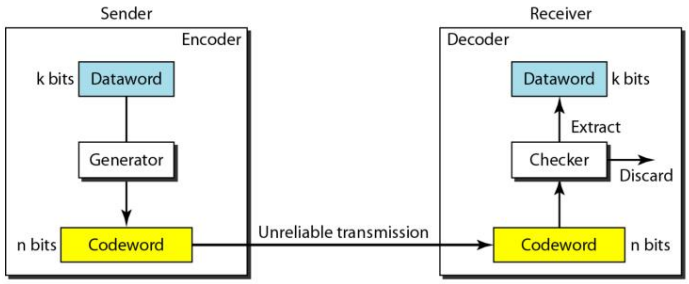
Understanding Defense-in-Depth



Error Detection and Correction

Error Detection

- A receiver can detect a change if the original codeword if
  - The receiver has a list of valid codewords, and
  - The original codeword has changed to an invalid one.

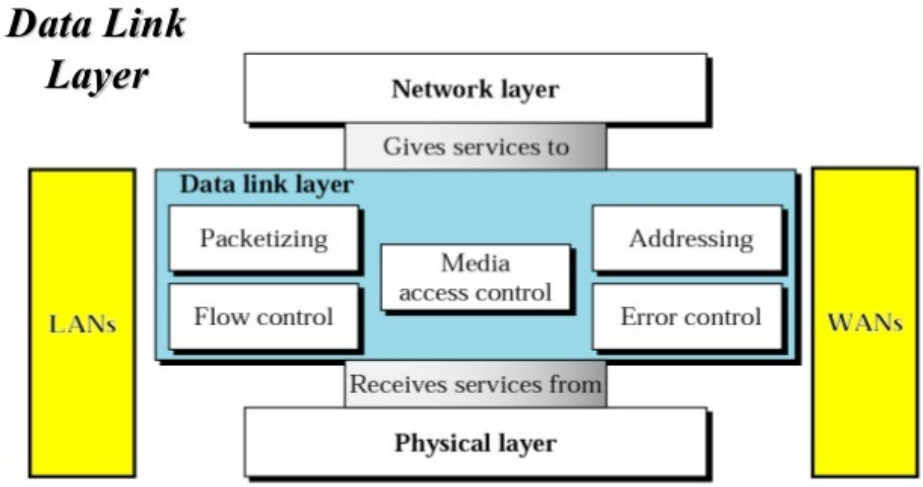


10.12

Er. M.S.Kuthar

Error Detection and Correction

Data can be corrupted during transmission. Some applications require that errors be detected and corrected.

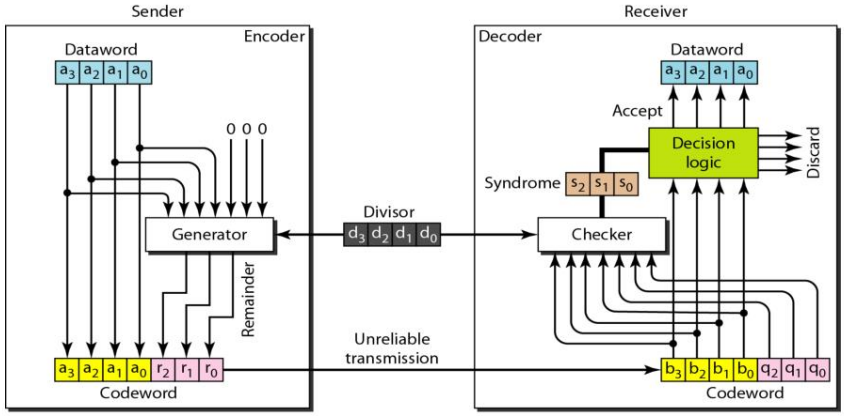


BCS Error Detection & Correction

- Error Detection
  - Check if any error has occurred
  - Don't care the number of errors
  - Don't care the positions of errors
- Error Correction
  - Need to know the number of errors
  - Need to know the positions of errors
  - More complex
  - The number of errors and the size of the message are important factors in error correction

BCS Error Detection & Correction

Figure 1.6 CRC encoder and decoder



1. User Interface

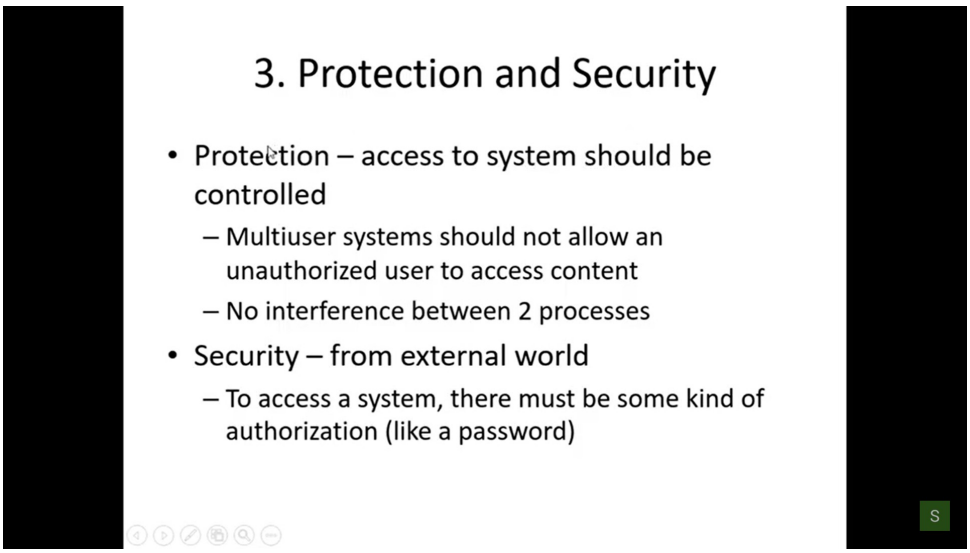
- Command Line Interface (CLI)
  - Interaction with text/commands
- Batch Files
  - Interaction with the help of batch files (.bat)
- Graphical User Interface (GUI)
  - User friendly
  - Easier to use : pointing device, menus etc
- Hybrid UI

1. Resource Allocation

- In case of multiple users accessing same resource
- In case of multiple processes accessing same resource
- Example:
  - Multiple users may be accessing same resource (say printer), then OS allocates the printer based on some algo (like FIFO for ex)
  - CPU Scheduling algos (FIFO, SJF etc)

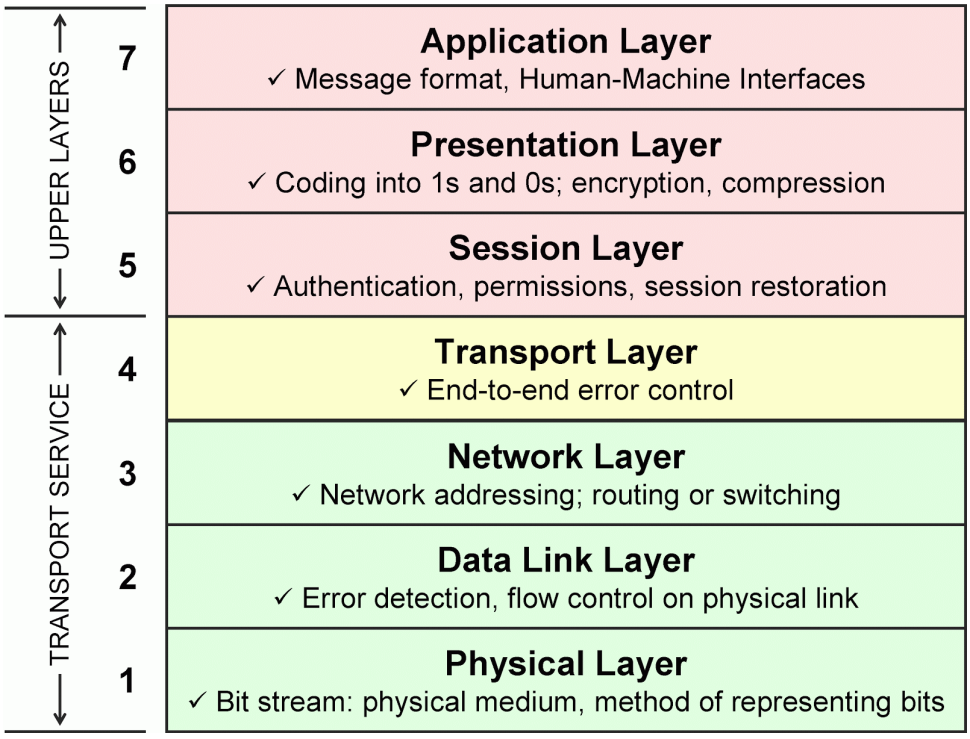
2. Accounting

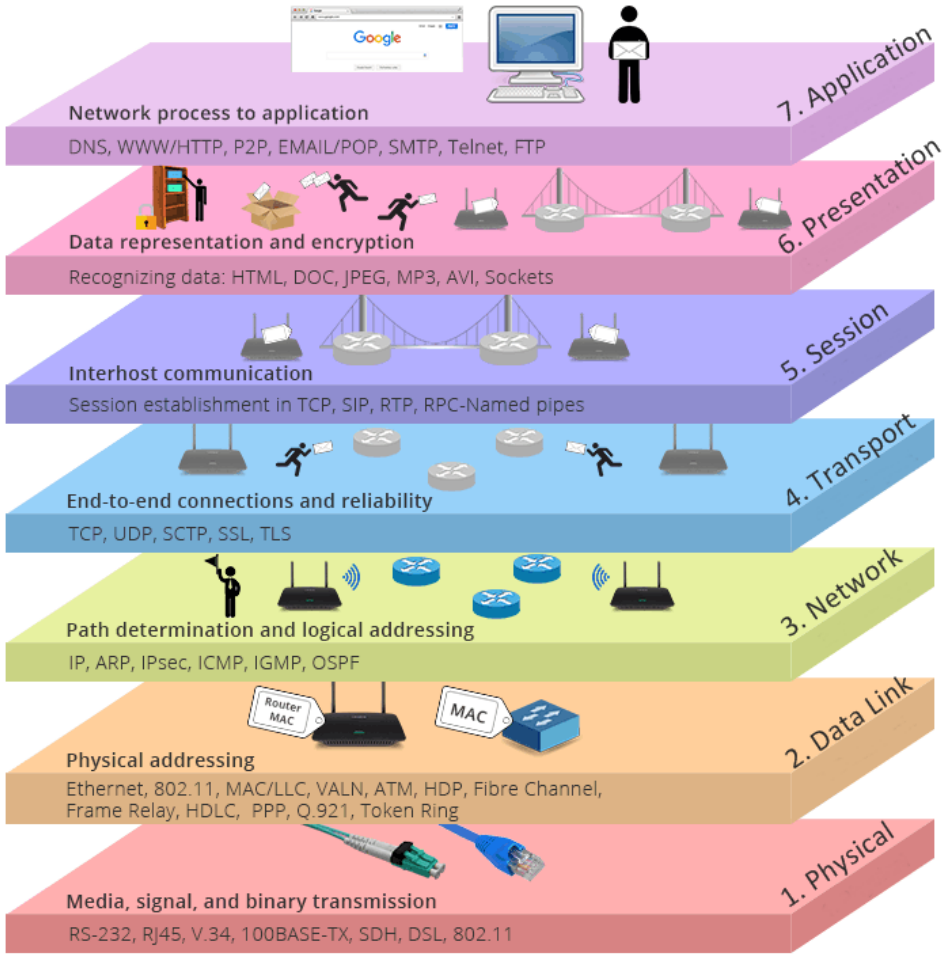
- Statistics related to the resource usage by the users – how much resource each user consumes, what all resources a user uses etc.
- Can be used for billing purposes
- Can also help in reconfiguring system to improve computing services



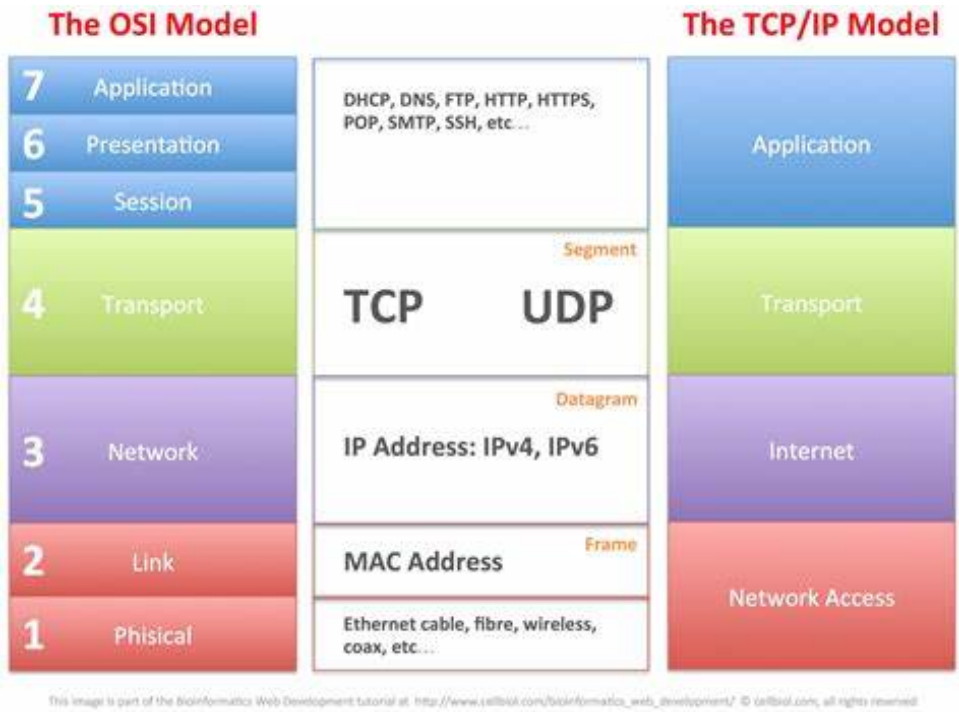
Network

Network OSI Model

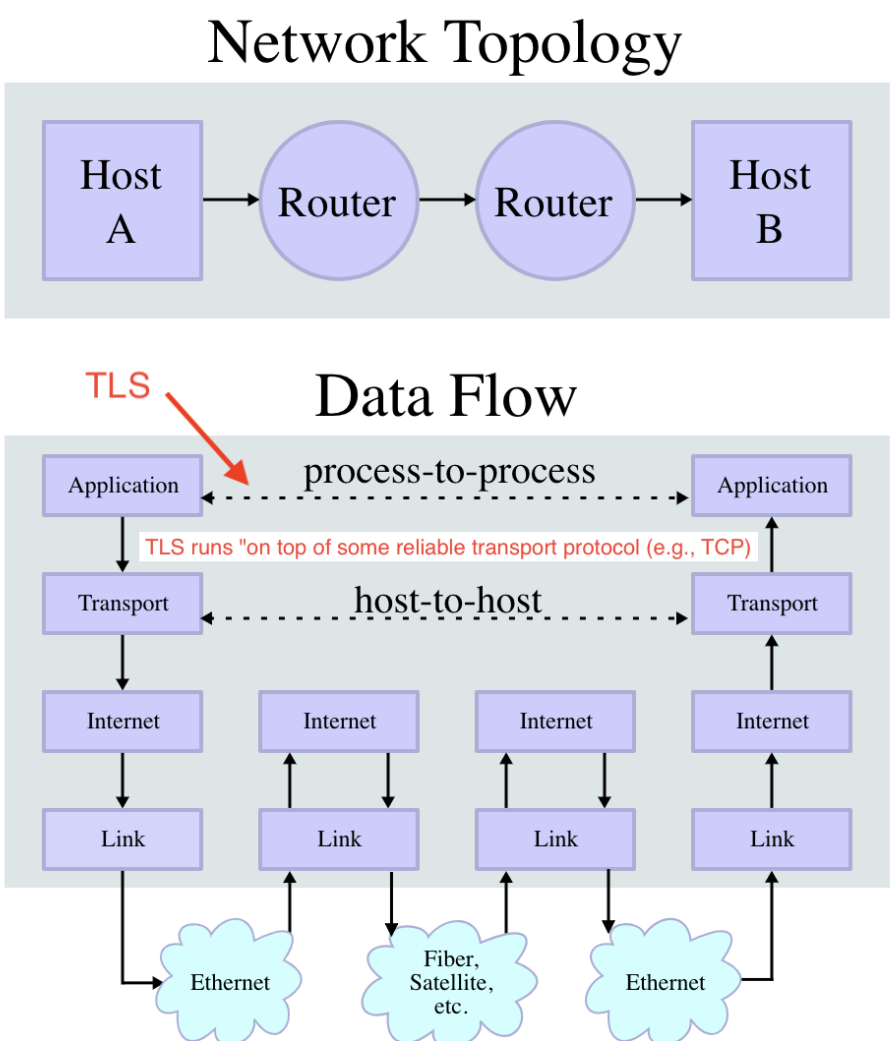




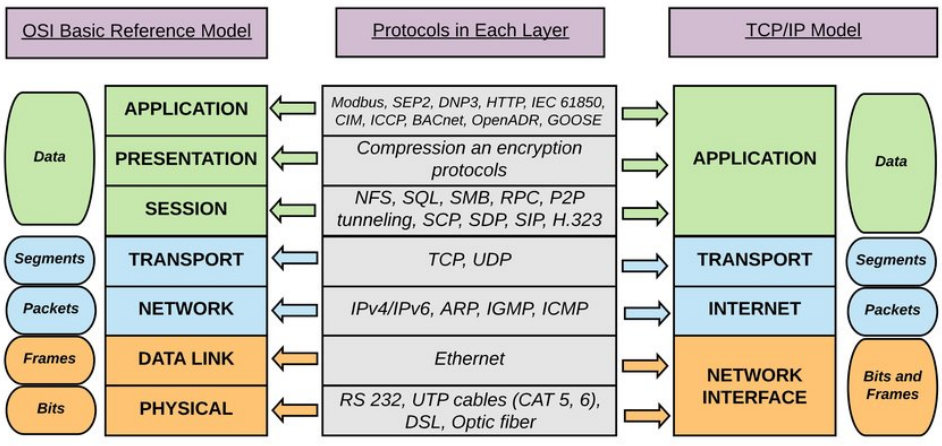
TCP/IP

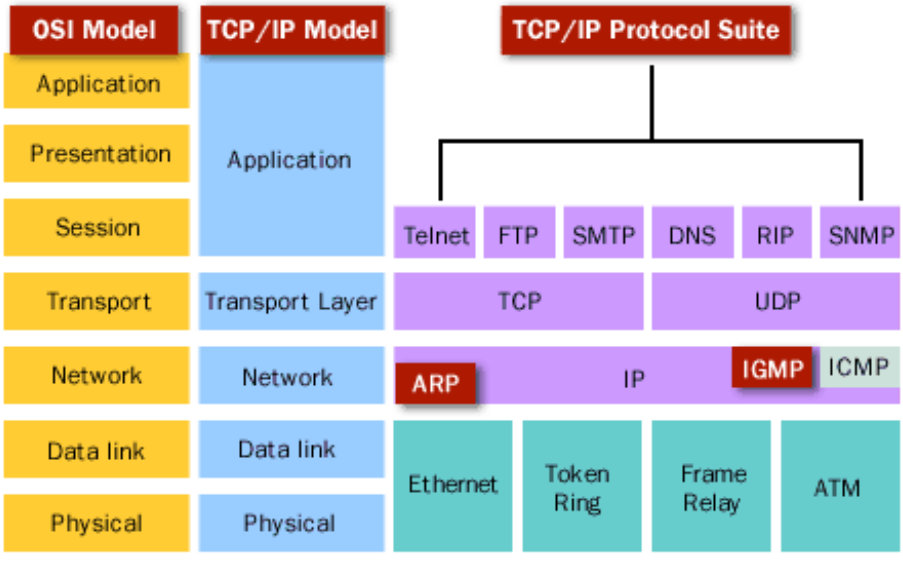


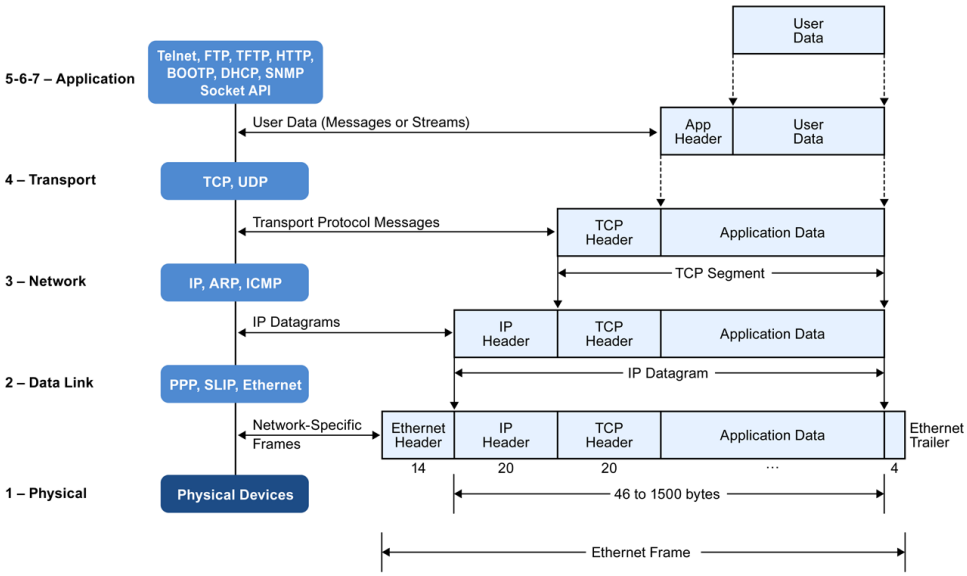
Data Flow

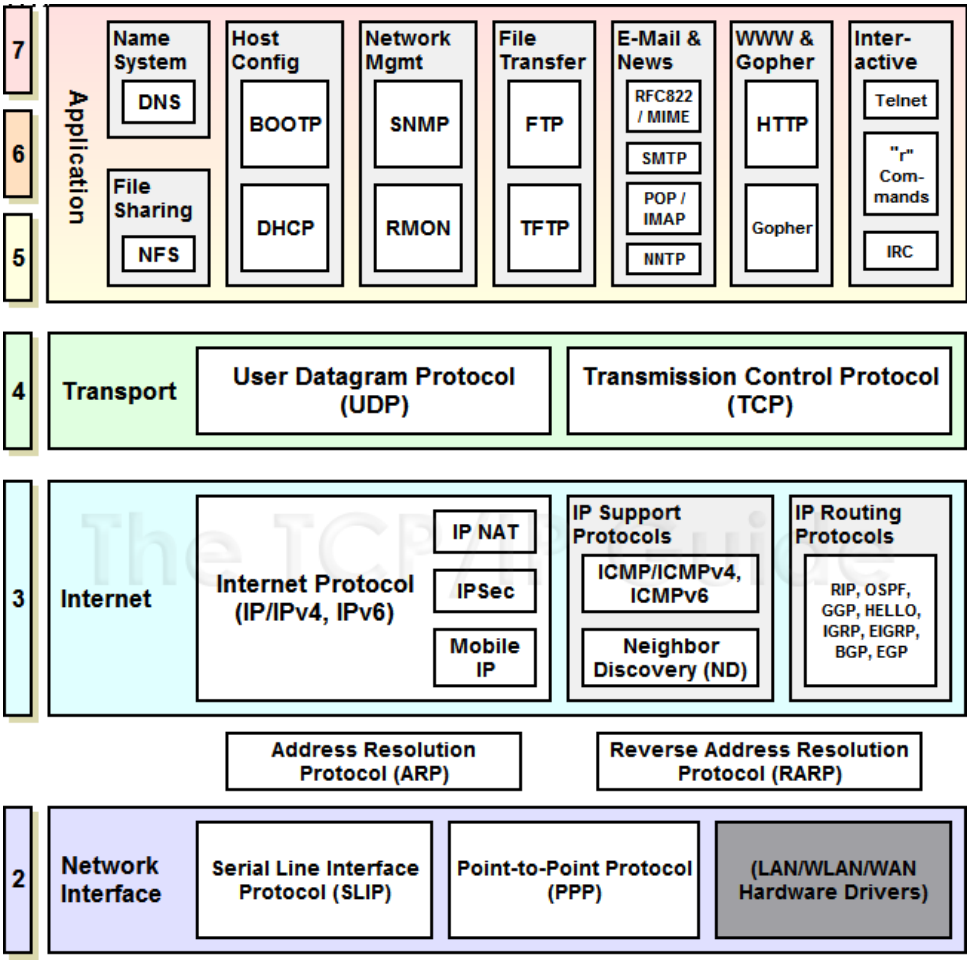

Network Topology
Host A
Router
Router
Host B
TLS
Data Flow
Application
process-to-process
Application
TLS runs "on top of some reliable transport protocol (e.g., TCP)
Transport
host-to-host
Transport
Internet
Internet
Internet
Internet
Link
Link
Link
Link
Ethernet
Fiber, Satellite, etc.
Ethernet

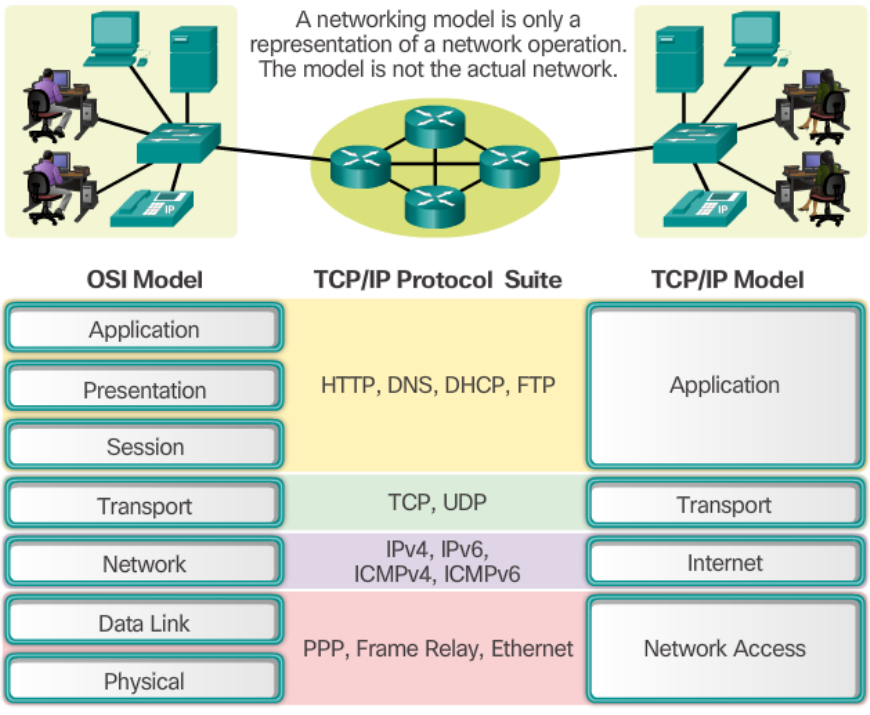
Protocol

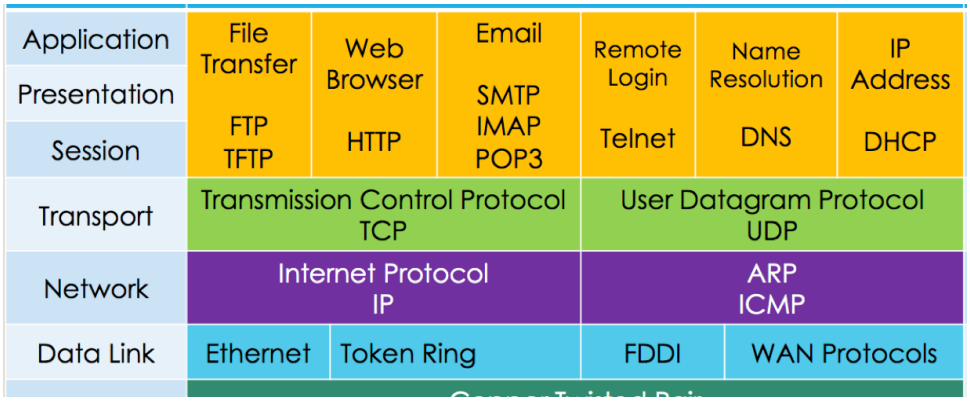

OSI Basic Reference Model
Protocols in Each Layer
TCP/IP Model
Data
Segments
Packets
Frames
Bits
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
Modbus, SEP2, DNP3, HTTP, IEC 61850, CIM, ICCP, BACnet, OpenADR, GOOSE
Compression an encryption protocols
NFS, SQL, SMB, RPC, P2P tunneling, SCP, SDP, SIP, H.323
TCP, UDP
IPv4/IPv6, ARP, IGMP, ICMP
Ethernet
RS 232, UTP cables (CAT 5, 6), DSL, Optic fiber
APPLICATION
TRANSPORT
INTERNET
NETWORK INTERFACE
Data
Segments
Packets
Bits and Frames

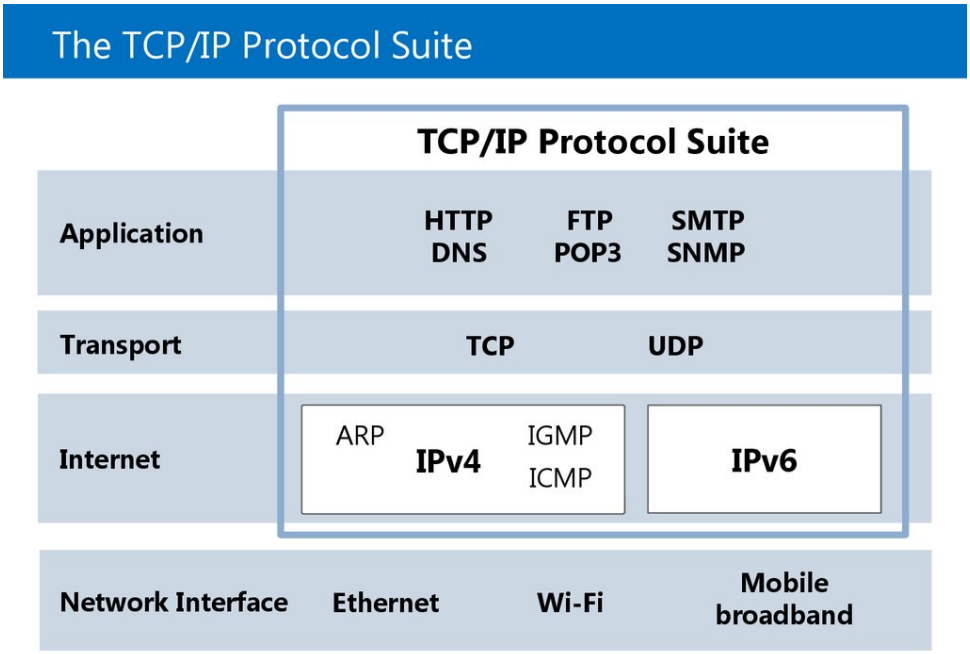

OSI Model
TCP/IP Model
TCP/IP Protocol Suite
Application
Presentation
Session
Application
Telnet
FTP
SMTP
DNS
RIP
SNMP
Transport
Transport Layer
TCP
UDP
Network
Network
ARP
IP
IGMP
ICMP
Data link
Data link
Physical
Physical
Ethernet
Token Ring
Frame Relay
ATM


5-6-7 – Application
Telnet, FTP, TFTP, HTTP, BOOTP, DHCP, SNMP Socket API
User Data
User Data (Messages or Streams)
App Header
User Data
4 – Transport
TCP, UDP
Transport Protocol Messages
TCP Header
Application Data
TCP Segment
3 – Network
IP, ARP, ICMP
IP Datagrams
IP Header
TCP Header
Application Data
IP Datagram
2 – Data Link
PPP, SLIP, Ethernet
Network-Specific Frames
Ethernet Header
IP Header
TCP Header
Application Data
Ethernet Trailer
14
20
20
...
4
1 – Physical
Physical Devices
46 to 1500 bytes
Ethernet Frame

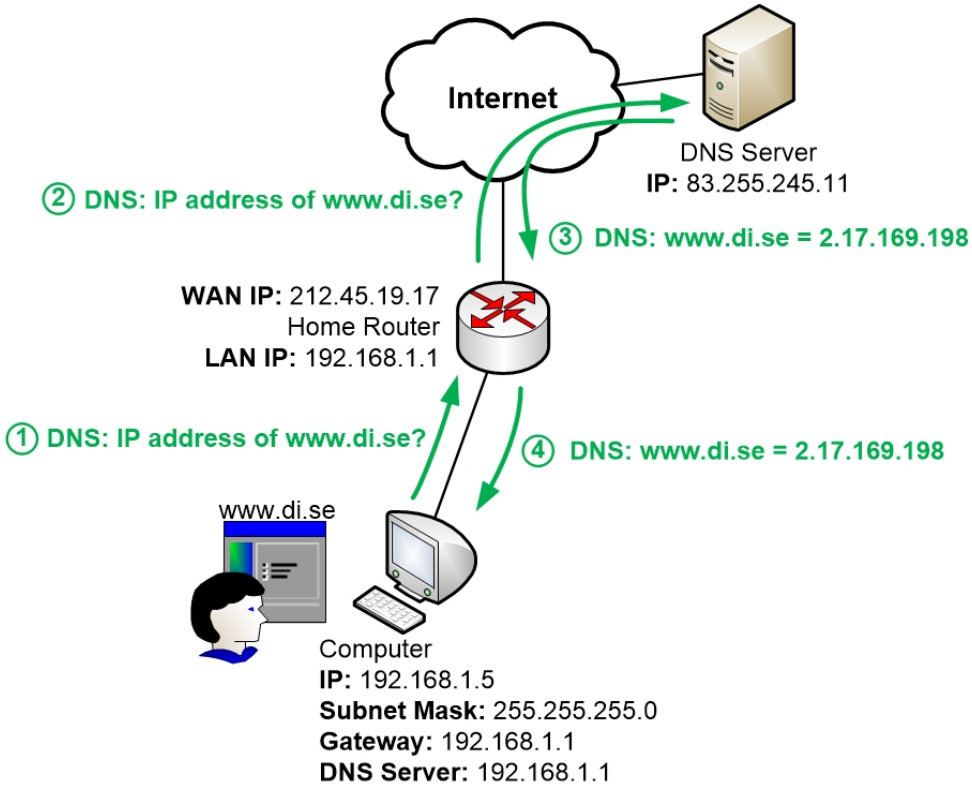

7
6
5
Application
Name System
DNS
File Sharing
NFS
Host Config
BOOTP
DHCP
Network Mgmt
SNMP
RMON
File Transfer
FTP
TFTP
E-Mail & News
RFC822 / MIME
SMTP
POP / IMAP
NNTP
WWW & Gopher
HTTP
Gopher
Inter-active
Telnet
"r" Com-mands
IRC
4
Transport
User Datagram Protocol (UDP)
Transmission Control Protocol (TCP)
3
Internet
Internet Protocol (IP/IPv4, IPv6)
IP NAT
IPSec
Mobile IP
IP Support Protocols
ICMP/ICMPv4, ICMPv6
Neighbor Discovery (ND)
IP Routing Protocols
RIP, OSPF, GGP, HELLO, IGRP, EIGRP, BGP, EGP
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)
2
Network Interface
Serial Line Interface Protocol (SLIP)
Point-to-Point Protocol (PPP)
(LAN/WLAN/WAN Hardware Drivers)

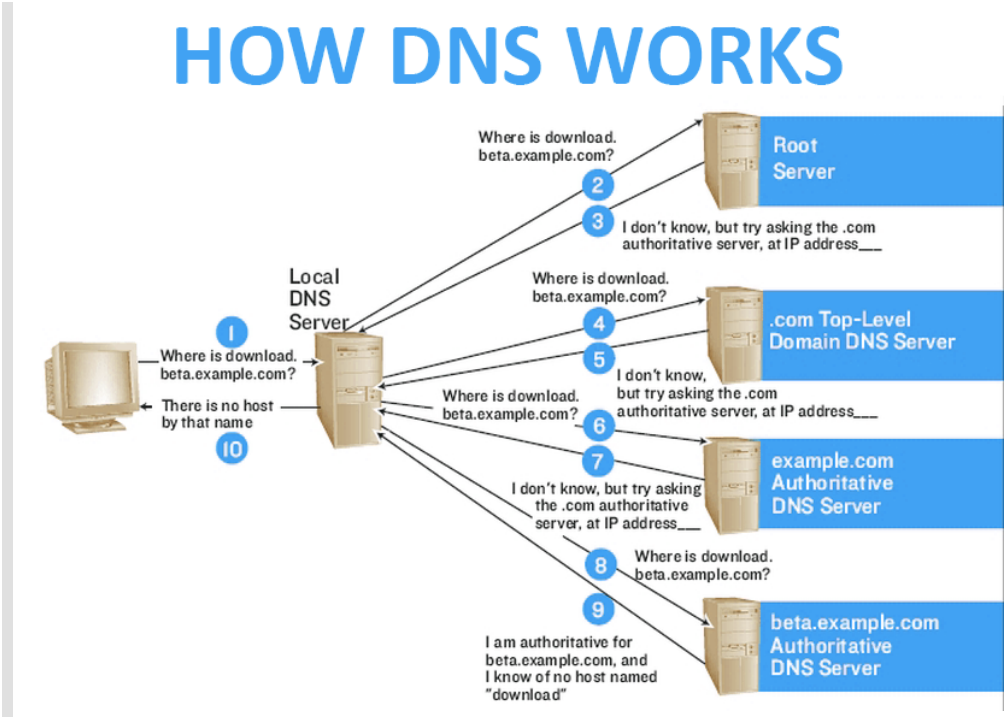

A networking model is only a representation of a network operation. The model is not the actual network.
OSI Model
TCP/IP Protocol Suite
TCP/IP Model
Application
Presentation
Session
HTTP, DNS, DHCP, FTP
Application
Transport
TCP, UDP
Transport
Network
IPv4, IPv6, ICMPv4, ICMPv6
Internet
Data Link
Physical
PPP, Frame Relay, Ethernet
Network Access

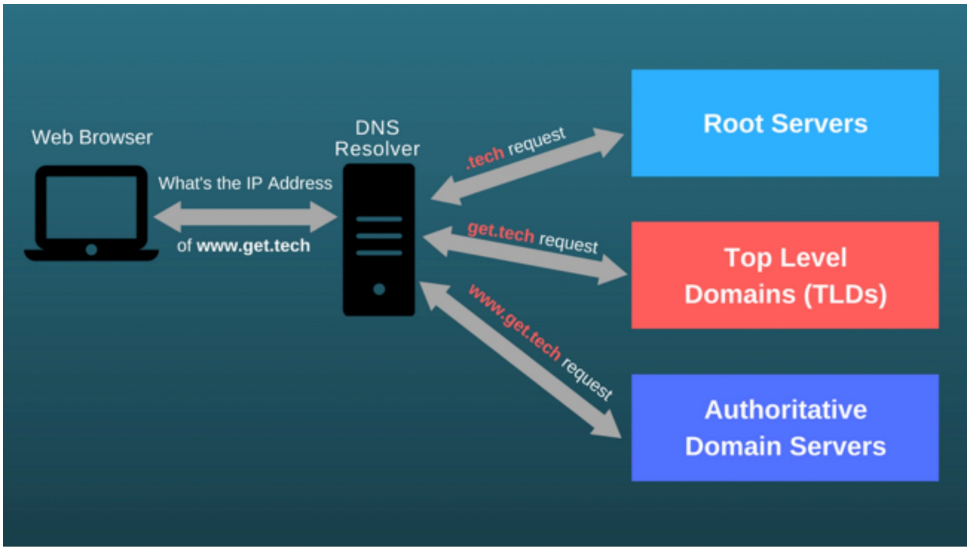


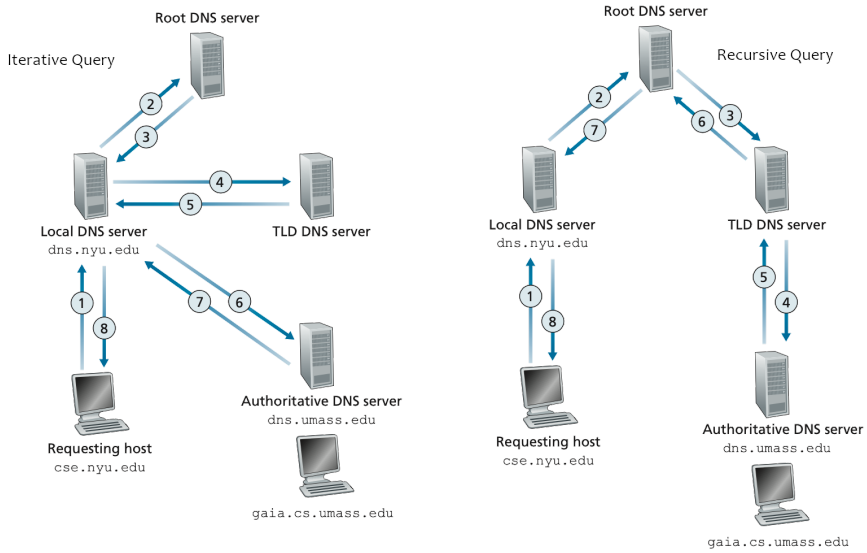


DNS(Domain Name System)


Internet
DNS Server
IP: 83.255.245.11
② DNS: IP address of www.di.se?
③ DNS: www.di.se = 2.17.169.198
WAN IP: 212.45.19.17
Home Router
LAN IP: 192.168.1.1
① DNS: IP address of www.di.se?
④ DNS: www.di.se = 2.17.169.198
www.di.se
Computer
IP: 192.168.1.5
Subnet Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server: 192.168.1.1


HOW DNS WORKS
Where is download. beta.example.com?
Root Server
I don't know, but try asking the .com authoritative server, at IP address___
Local DNS Server
Where is download. beta.example.com?
.com Top-Level Domain DNS Server
I don't know, but try asking the .com authoritative server, at IP address___
Where is download. beta.example.com?
There is no host by that name
Where is download. beta.example.com?
example.com Authoritative DNS Server
I don't know, but try asking the .com authoritative server, at IP address___
Where is download. beta.example.com?
beta.example.com Authoritative DNS Server
I am authoritative for beta.example.com, and I know of no host named "download"





Port

Port #	Application Layer Protocol	Type	Description
20	FTP	TCP	File Transfer Protocol - data
21	FTP	TCP	File Transfer Protocol - control
22	SSH	TCP/UDP	Secure Shell for secure login
23	Telnet	TCP	Unencrypted login
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP/UDP	Domain Name Server
67/68	DHCP	UDP	Dynamic Host
80	HTTP	TCP	HyperText Transfer Protocol
123	NTP	UDP	Network Time Protocol
161,162	SNMP	TCP/UDP	Simple Network Management Protocol
389	LDAP	TCP/UDP	Lightweight Directory Authentication Protocol
443	HTTPS	TCP/UDP	HTTP with Secure Socket Layer

TCP/UDP Port Numbers

7 Echo	554 RTSP	2745 Bagle.H	6891-6901 Windows Live
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970 Quicktime
20-21 FTP	560 rmonitor	3050 Interbase DB	7212 GhostSurf
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649 CU-SeeMe
23 Telnet	587 SMTP	3124 HTTP Proxy	8000 Internet Radio
25 SMTP	591 FileMaker	3127 MyDoom	8080 HTTP Proxy
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087 Kaspersky AV
43 WHOIS	631 Internet Printing	3222 GLBP	8118 Privoxy
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200 VMware Server
53 DNS	639 MSDP (PIM)	3306 MySQL	8500 Adobe ColdFusion
67-68 DHCP/BOOTP	646 LDP (MPLS)	3389 Terminal Server	8767 TeamSpeak
69 TFTP	691 MS Exchange	3689 iTunes	8866 Bagle.B
70 Gopher	860 iSCSI	3690 Subversion	9100 HP JetDirect
79 Finger	873 rsync	3724 World of Warcraft	9101-9103 Bacula
80 HTTP	902 VMware Server	3784-3785 Ventrilo	9119 MXit
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898 Dabber
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 Rbot/Spybot
113 Ident	1025 Microsoft RPC	4672 eMule	9999 Urchin
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000 Webmin
123 NTP	1080 SOCKS Proxy	5000 UPnP	10000 BackupExec
135 Microsoft RPC	1080 MyDoom	5001 Slingbox	10113-10116 NetIQ
137-139 NetBIOS	1194 OpenVPN	5001 iperf	11371 OpenPGP
143 IMAP4	1214 Kazaa	5004-5005 RTP	12035-12036 Second Life
161-162 SNMP	1241 Nessus	5050 Yahoo! Messenger	12345 NetBus
177 XDMCP	1311 Dell OpenManage	5060 SIP	13720-13721 NetBackup
179 BGP	1337 WASTE	5190 AIM/ICQ	14567 Battlefield
201 AppleTalk	1433-1434 Microsoft SQL	5222-5223 XMPP/Jabber	15118 Dipnet/Oddbob
264 BGMP	1512 WINS	5432 PostgreSQL	19226 AdminSecure
318 TSP	1589 Cisco VQP	5500 VNC Server	19638 Ensim
381-383 HP Openview	1701 L2TP	5554 Sasser	20000 Usermin
389 LDAP	1723 MS PPTP	5631-5632 pcAnywhere	24800 Synergy
411-412 Direct Connect	1725 Steam	5800 VNC over HTTP	25999 Xfire
443 HTTP over SSL	1741 CiscoWorks 2000	5900+ VNC Server	27015 Half-Life
445 Microsoft DS	1755 MS Media Server	6000-6001 X11	27374 Sub7
464 Kerberos	1812-1813 RADIUS	6112 Battle.net	28960 Call of Duty
465 SMTP over SSL	1863 MSN	6129 DameWare	31337 Back Orifice
497 Retrospect	1985 Cisco HSRP	6257 WinMX	33434+ traceroute
500 ISAKMP	2000 Cisco SCCP	6346-6347 Gnutella	
512 rexec	2002 Cisco ACS	6500 GameSpy Arcade	
513 rlogin	2049 NFS	6566 SANE	
514 syslog	2082-2083 cPanel	6588 AnalogX	
515 LPD/LPR	2100 Oracle XDB	6665-6669 IRC	
520 RIP	2222 DirectAdmin	6679/6697 IRC over SSL	
521 RIPng (IPv6)	2302 Halo	6699 Napster	
540 UUCP	2483-2484 Oracle DB	6881-6999 BitTorrent	

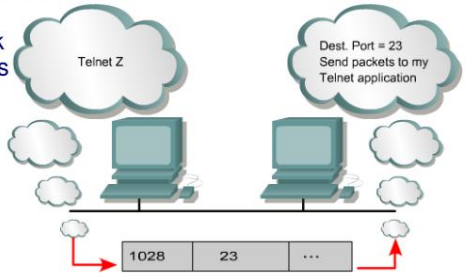
Legend

- Chat
- Encrypted
- Gaming
- Malicious
- Peer to Peer
- Streaming

IANA port assignments published at **http://www.iana.org/assignments/port-numbers**

Transport Layer Ports

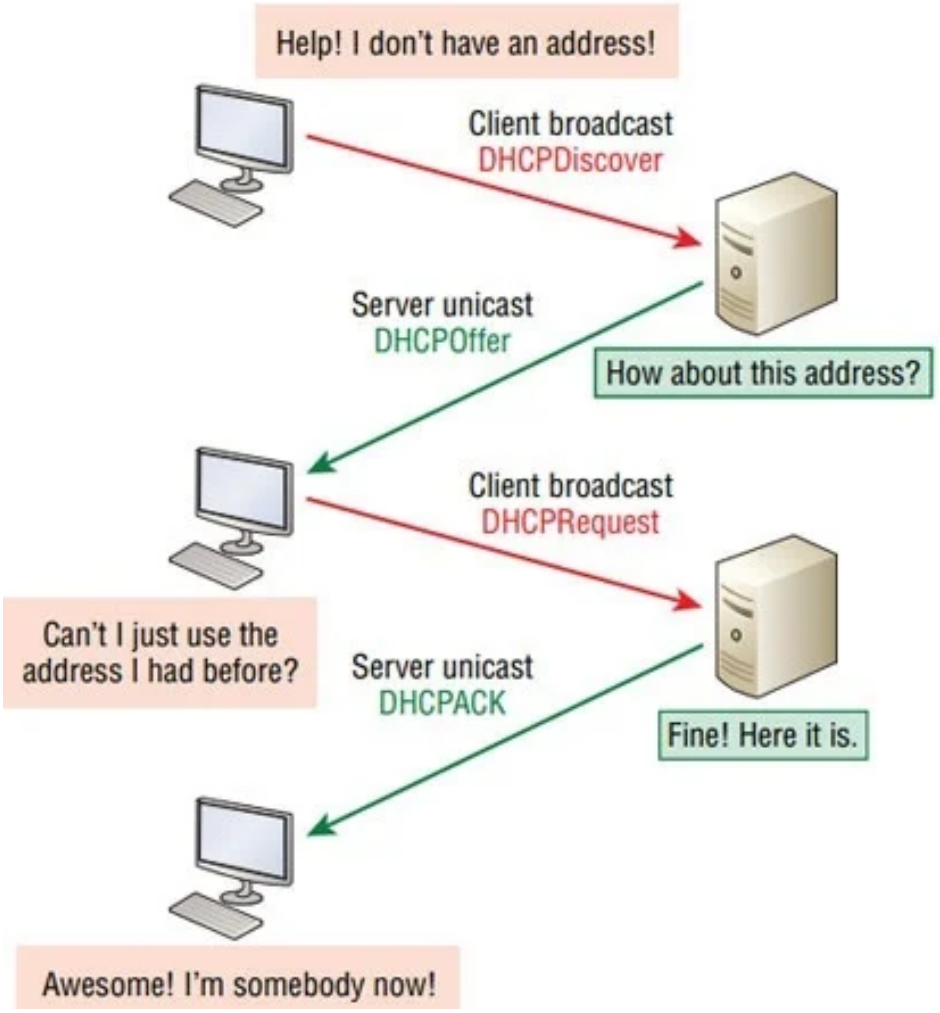
- Port numbers are used to keep track of different **conversations** that cross the network at the same time.
- Port numbers identify which upper layer service is needed, and are needed when a host communicates with a server that uses multiple services.

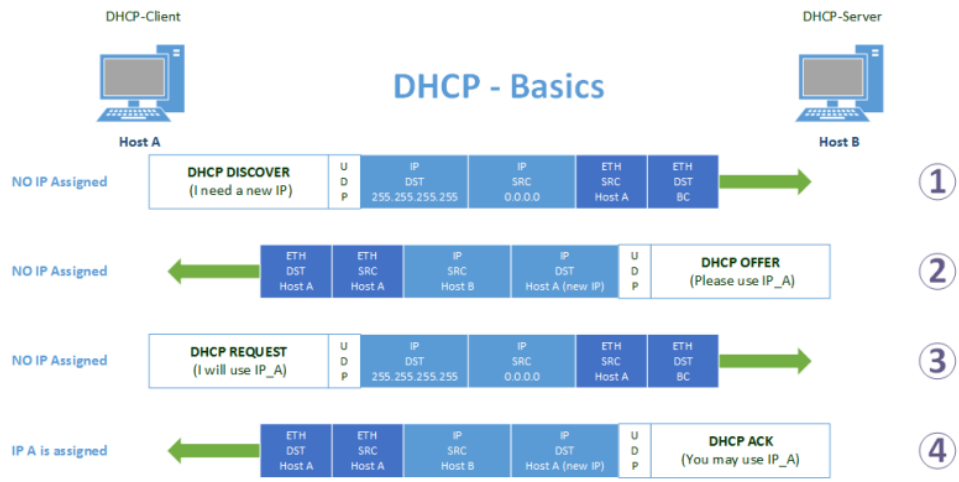


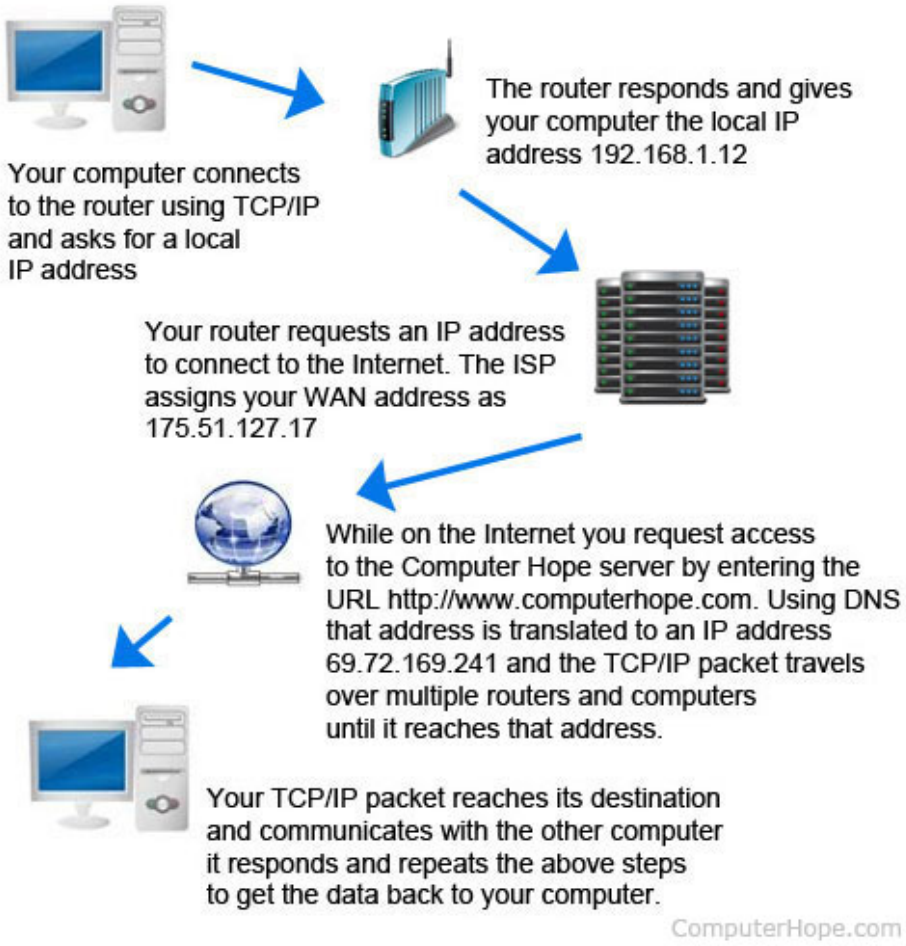
- Both TCP and UDP use port numbers to pass to the upper layers.
- Port numbers have the following **ranges**:
  - 0-255 used for public applications, 0-1023 also called **well-known ports,** regulated by IANA (Internet assigned numbers authority)**.**
  - Numbers from 255-1023 are assigned to marketable applications
  - 1024 through 49151 Registered Ports, not regulated.
  - 49152 through 65535 are Dynamic and/or Private Ports .

Dr. Muazzam A. Khan 4

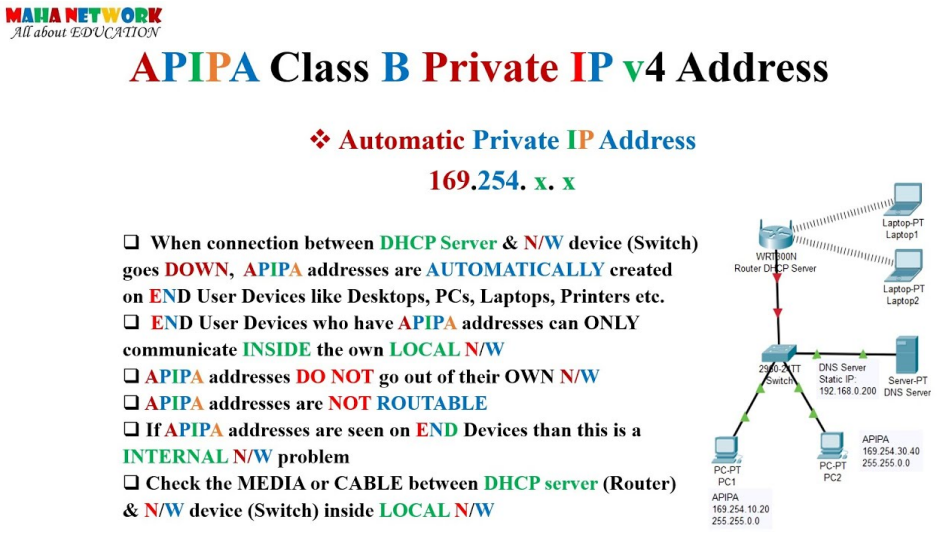
DHCP


Help! I don't have an address!
Client broadcast
DHCPDiscover
Server unicast
DHCPOffer
How about this address?
Client broadcast
DHCPRequest
Can't I just use the address I had before?
Server unicast
DHCPACK
Fine! Here it is.
Awesome! I'm somebody now!

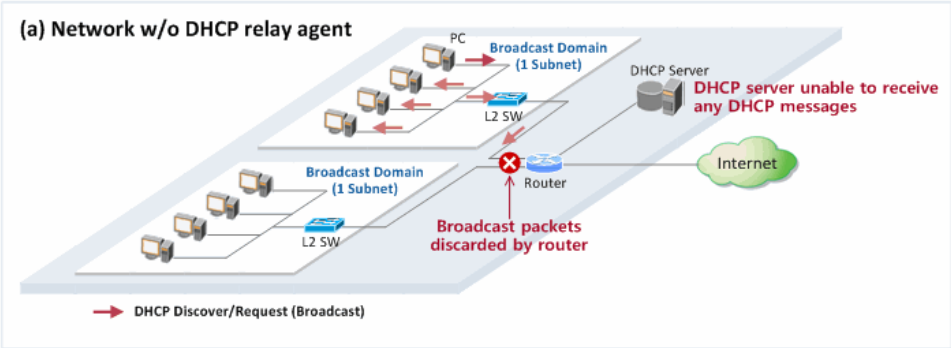

DHCP-Client
DHCP-Server
DHCP - Basics
Host A
Host B
NO IP Assigned
DHCP DISCOVER
(I need a new IP)
U D P
IP DST 255.255.255.255
IP SRC 0.0.0.0
ETH SRC Host A
ETH DST BC
1
NO IP Assigned
ETH DST Host A
ETH SRC Host A
IP SRC Host B
IP DST Host A (new IP)
U D P
DHCP OFFER
(Please use IP_A)
2
NO IP Assigned
DHCP REQUEST
(I will use IP_A)
U D P
IP DST 255.255.255.255
IP SRC 0.0.0.0
ETH SRC Host A
ETH DST BC
3
IP A is assigned
ETH DST Host A
ETH SRC Host A
IP SRC Host B
IP DST Host A (new IP)
U D P
DHCP ACK
(You may use IP_A)
4

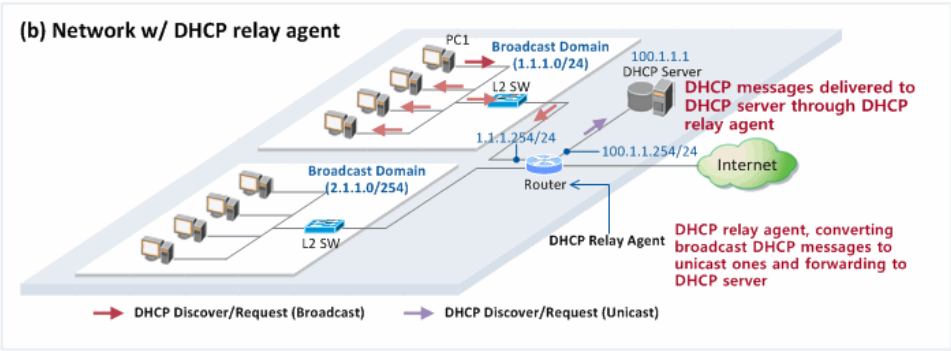


APIPA

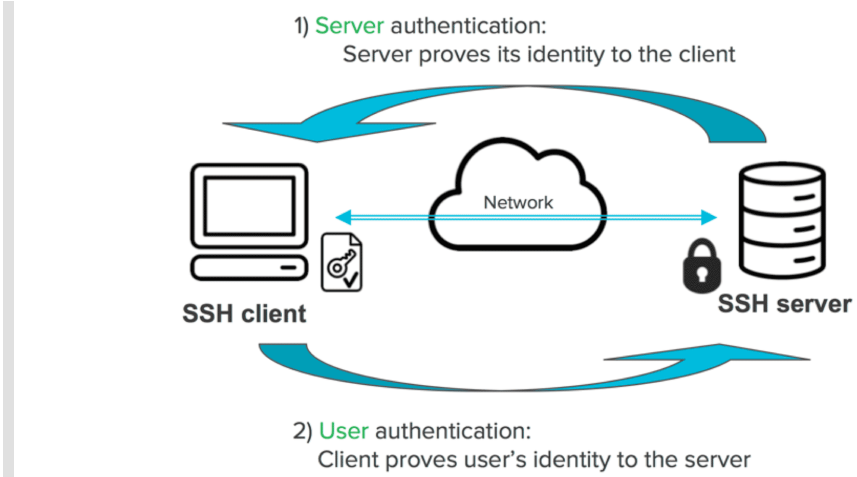


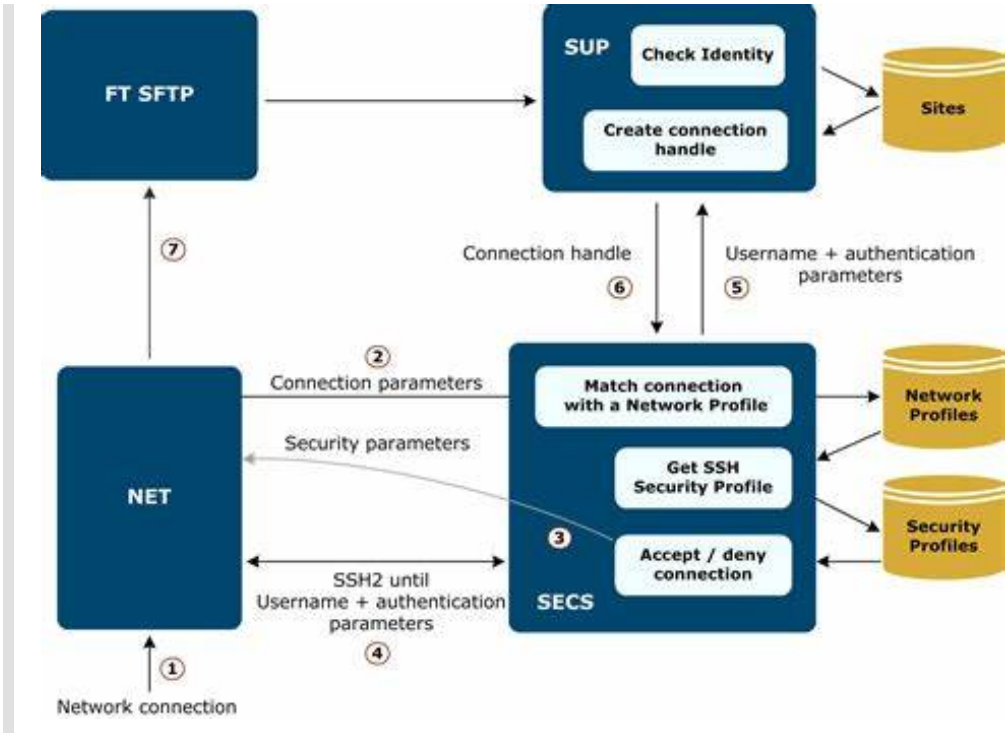
DHCP Relay = IP Helper



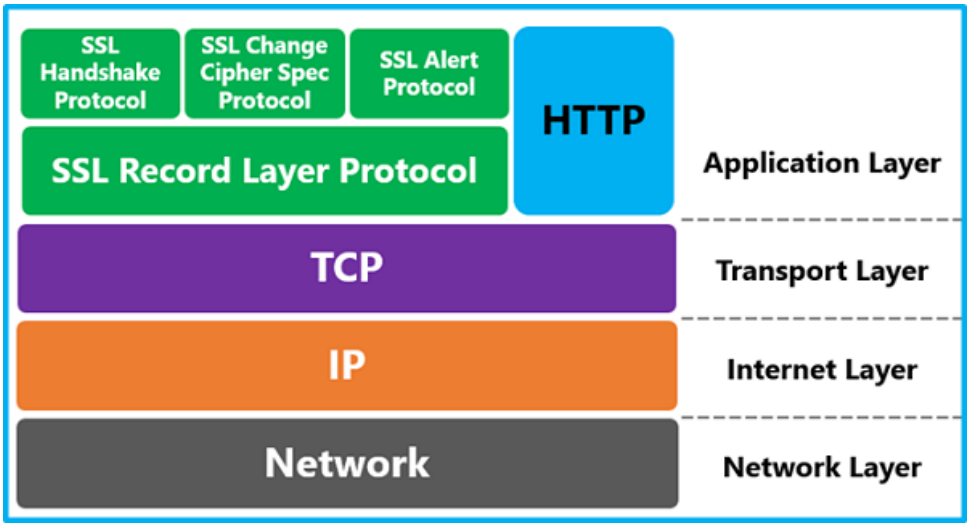


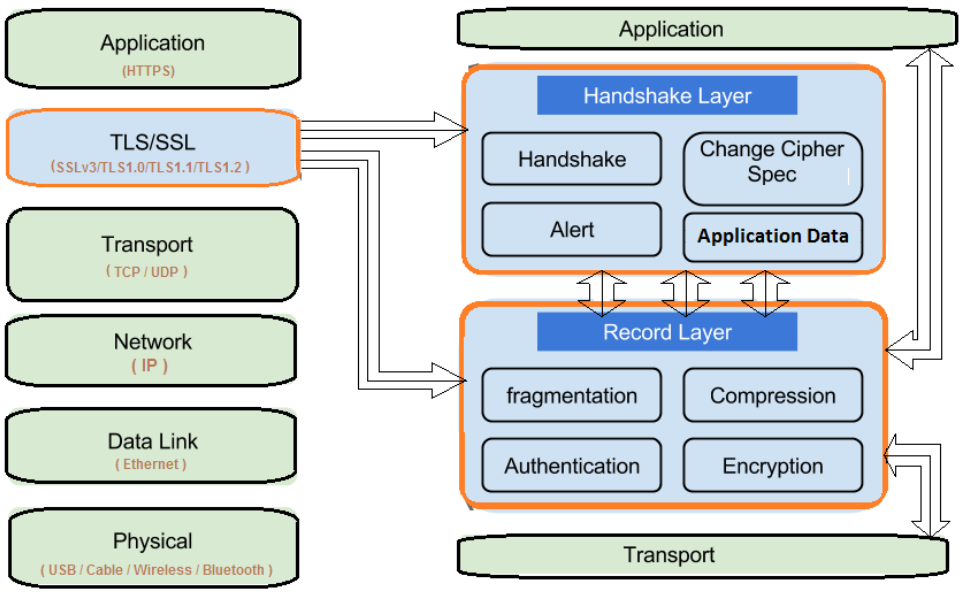
SSH(Secure Shell)

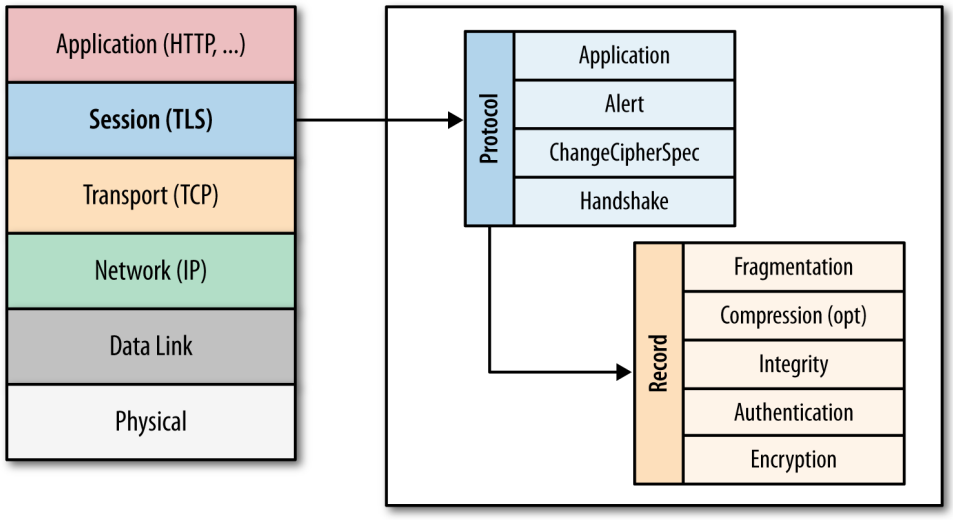


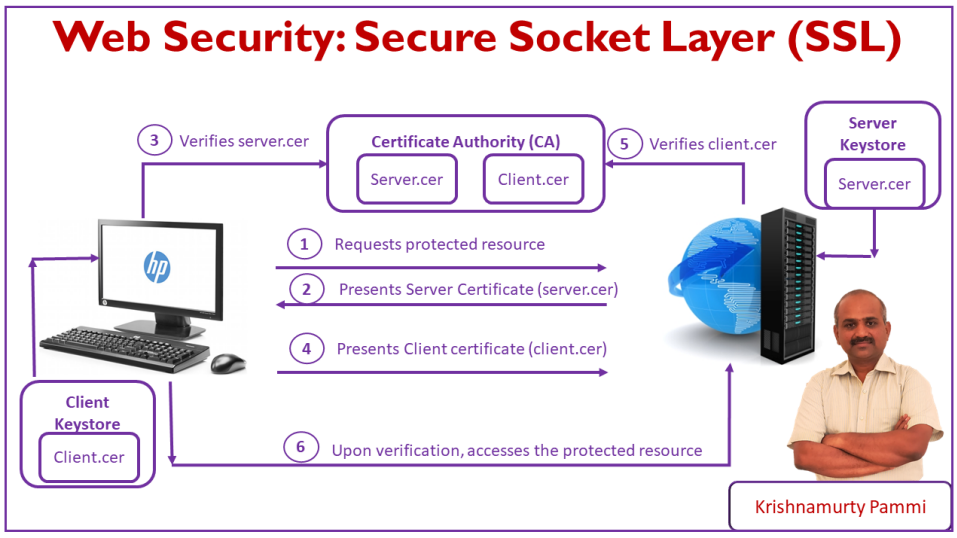


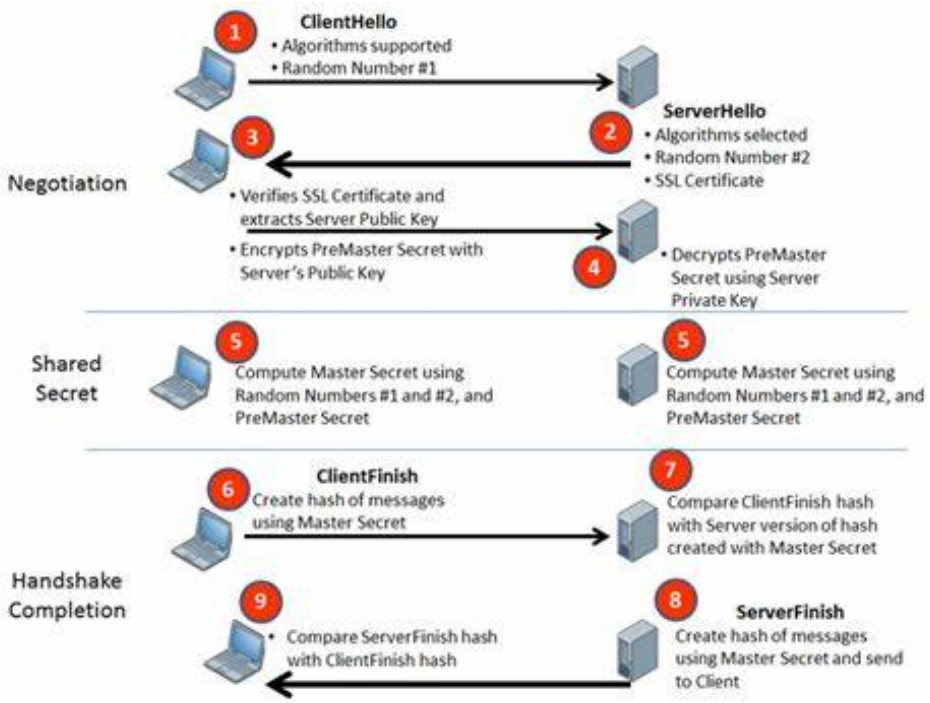
SSL(Secure Sockets Layers)

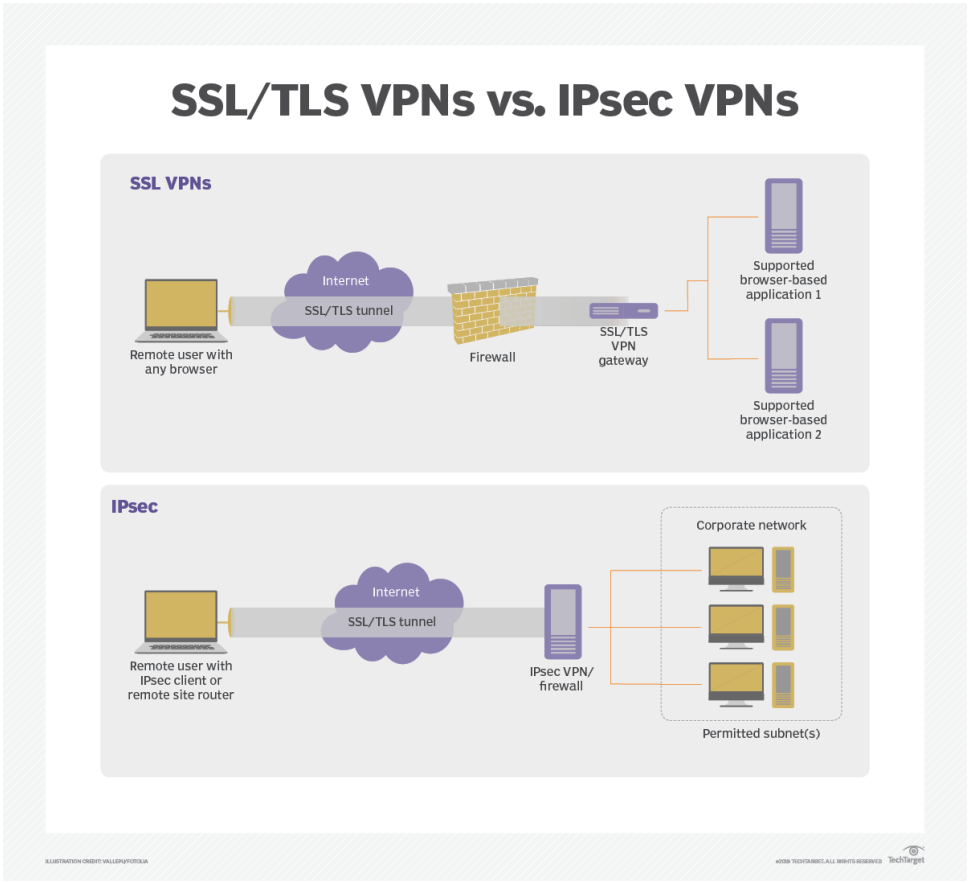


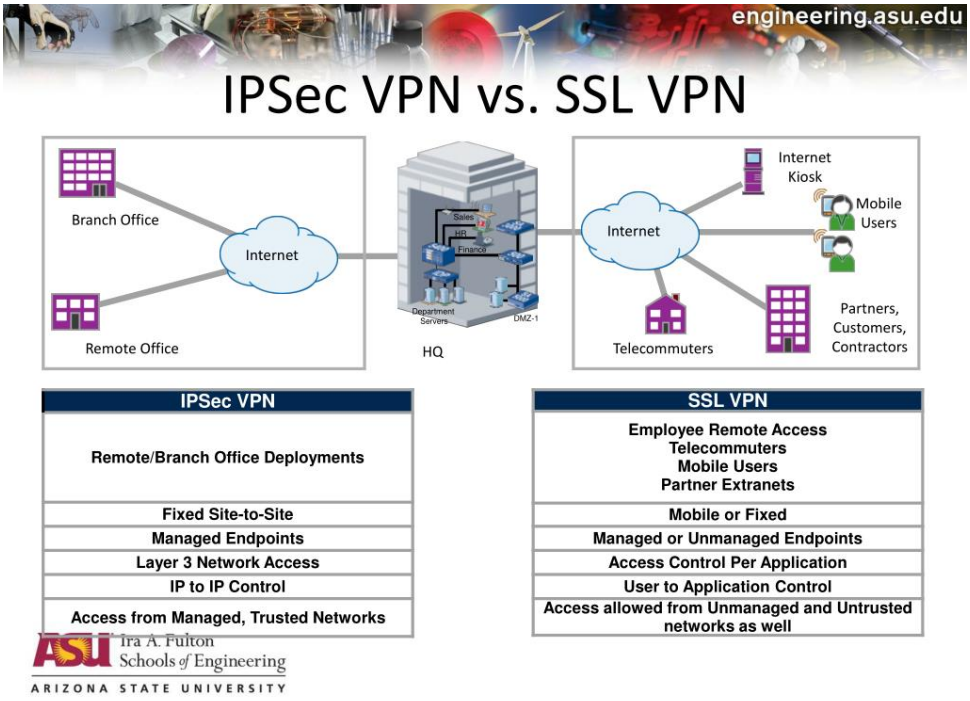



Application (HTTP, ...)
Session (TLS)
Transport (TCP)
Network (IP)
Data Link
Physical
Protocol
Application
Alert
ChangeCipherSpec
Handshake
Record
Fragmentation
Compression (opt)
Integrity
Authentication
Encryption

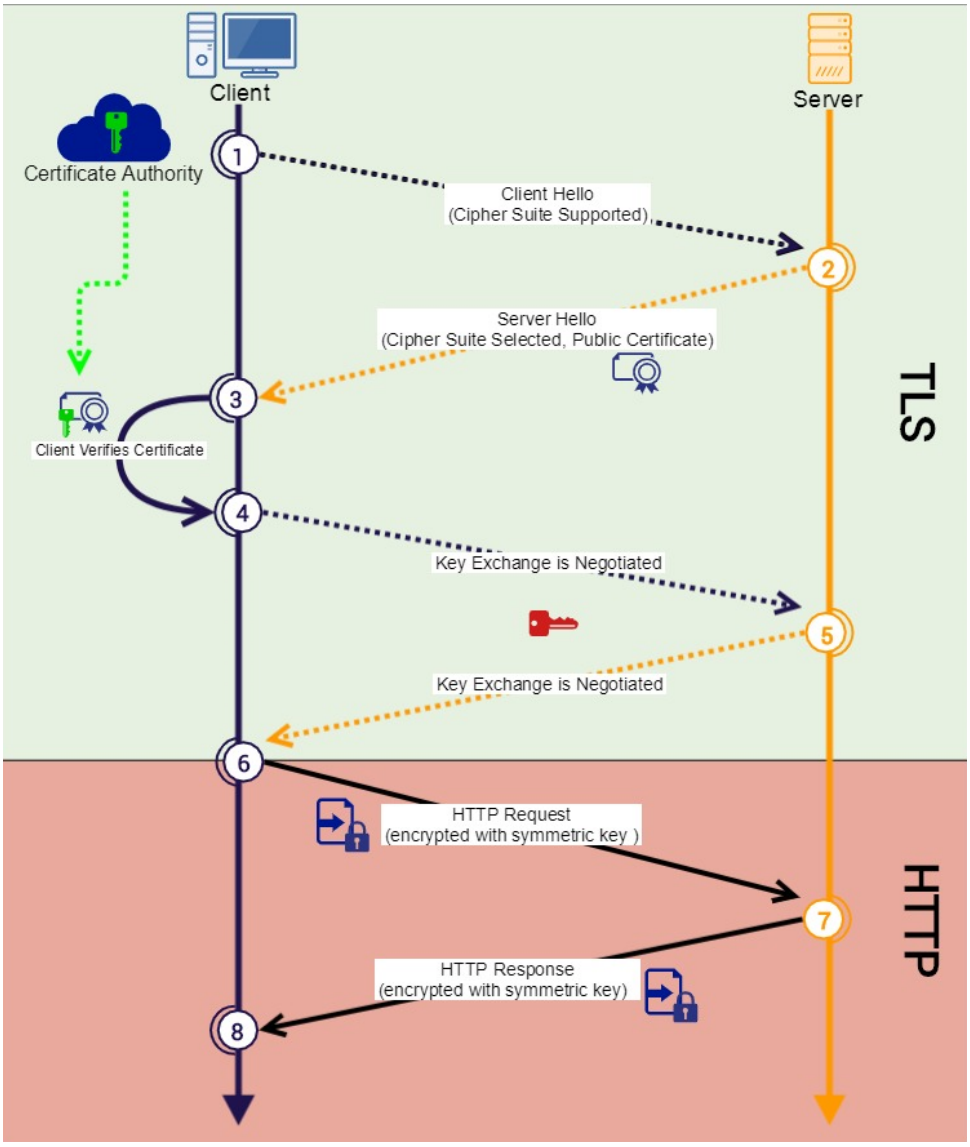

Web Security: Secure Socket Layer (SSL)
3 Verifies server.cer
Certificate Authority (CA)
Server.cer
Client.cer
5 Verifies client.cer
Server Keystore
Server.cer
1 Requests protected resource
2 Presents Server Certificate (server.cer)
4 Presents Client certificate (client.cer)
Client Keystore
Client.cer
6 Upon verification, accesses the protected resource
Krishnamurty Pammi


Negotiation
1 ClientHello
• Algorithms supported
• Random Number #1
2 ServerHello
• Algorithms selected
• Random Number #2
• SSL Certificate
3
• Verifies SSL Certificate and extracts Server Public Key
• Encrypts PreMaster Secret with Server's Public Key
4
• Decrypts PreMaster Secret using Server Private Key
Shared Secret
5 Compute Master Secret using Random Numbers #1 and #2, and PreMaster Secret
5 Compute Master Secret using Random Numbers #1 and #2, and PreMaster Secret
Handshake Completion
6 ClientFinish
Create hash of messages using Master Secret
7 Compare ClientFinish hash with Server version of hash created with Master Secret
9
• Compare ServerFinish hash with ClientFinish hash
8 ServerFinish
Create hash of messages using Master Secret and send to Client

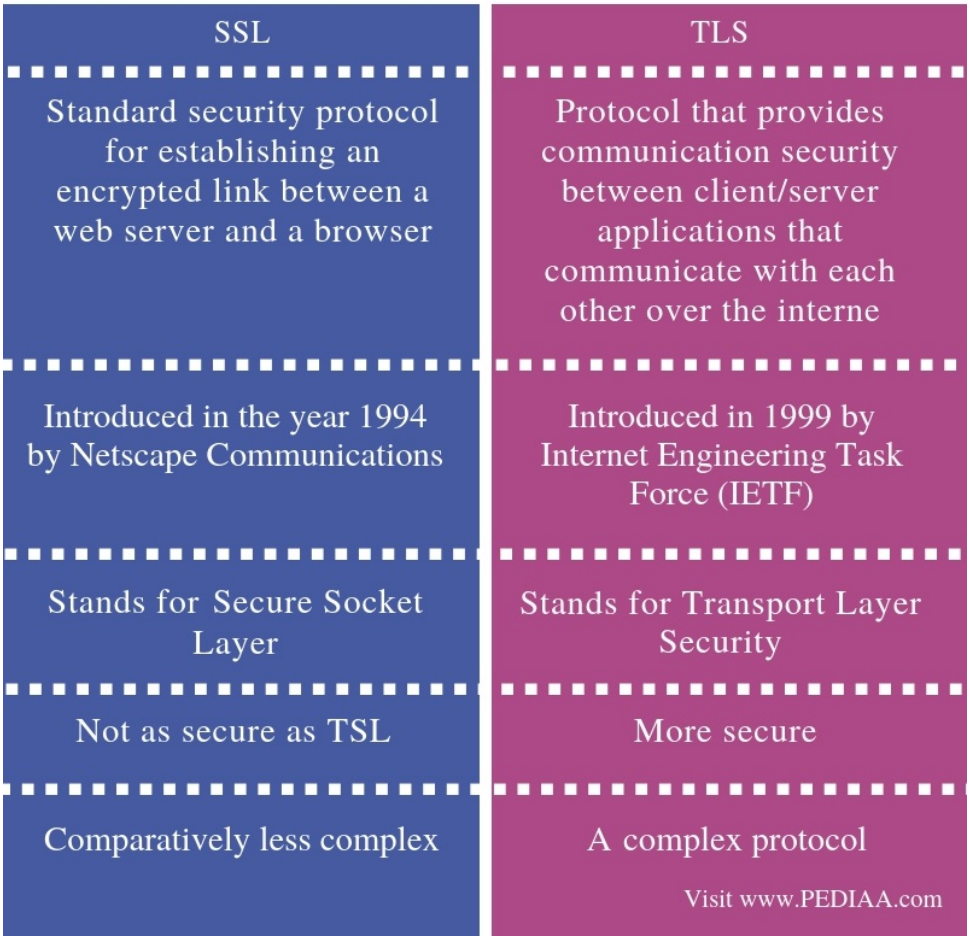

SSL/TLS VPNs vs. IPsec VPNs
SSL VPNs
Internet
SSL/TLS tunnel
Remote user with any browser
Firewall
SSL/TLS VPN gateway
Supported browser-based application 1
Supported browser-based application 2
IPsec
Corporate network
Internet
SSL/TLS tunnel
Remote user with IPsec client or remote site router
IPsec VPN/ firewall
Permitted subnet(s)


engineering.asu.edu
IPSec VPN vs. SSL VPN
Branch Office
Internet
Remote Office
HQ
Internet
Internet Kiosk
Mobile Users
Telecommuters
Partners, Customers, Contractors
IPSec VPN
Remote/Branch Office Deployments
Fixed Site-to-Site
Managed Endpoints
Layer 3 Network Access
IP to IP Control
Access from Managed, Trusted Networks
SSL VPN
Employee Remote Access
Telecommuters
Mobile Users
Partner Extranets
Mobile or Fixed
Managed or Unmanaged Endpoints
Access Control Per Application
User to Application Control
Access allowed from Unmanaged and Untrusted networks as well
Ira A. Fulton Schools of Engineering
ARIZONA STATE UNIVERSITY

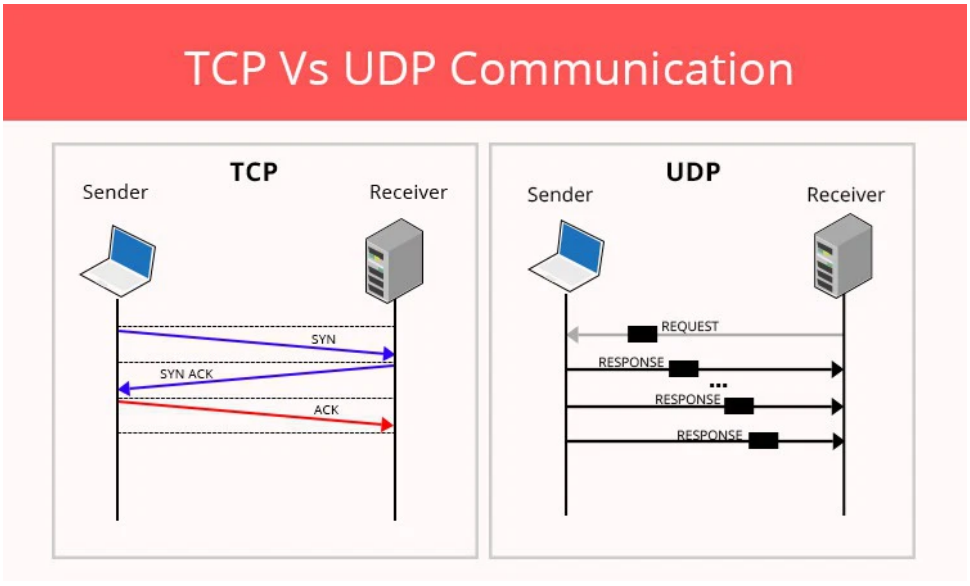
TLS

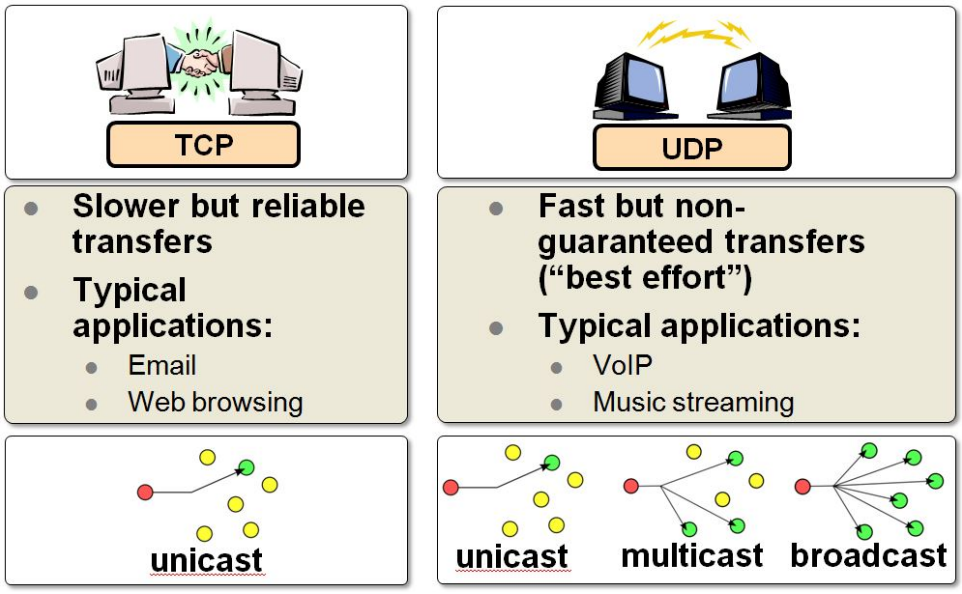

Client
Server
Certificate Authority
1
Client Hello
(Cipher Suite Supported)
2
Server Hello
(Cipher Suite Selected, Public Certificate)
3
Client Verifies Certificate
4
Key Exchange is Negotiated
5
Key Exchange is Negotiated
6
HTTP Request
(encrypted with symmetric key)
7
HTTP Response
(encrypted with symmetric key)
8
TLS
HTTP

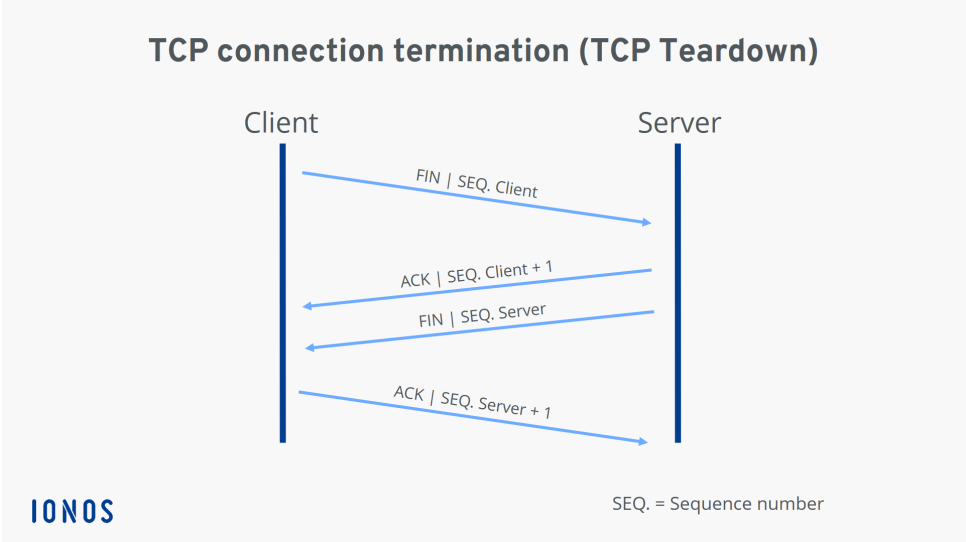
SSL vs TSL


SSL VERSUS TLS
SSL
TLS
Standard security protocol for establishing an encrypted link between a web server and a browser
Protocol that provides communication security between client/server applications that communicate with each other over the interne
Introduced in the year 1994 by Netscape Communications
Introduced in 1999 by Internet Engineering Task Force (IETF)
Stands for Secure Socket Layer
Stands for Transport Layer Security
Not as secure as TSL
More secure
Comparatively less complex
A complex protocol
Visit www.PEDIAA.com

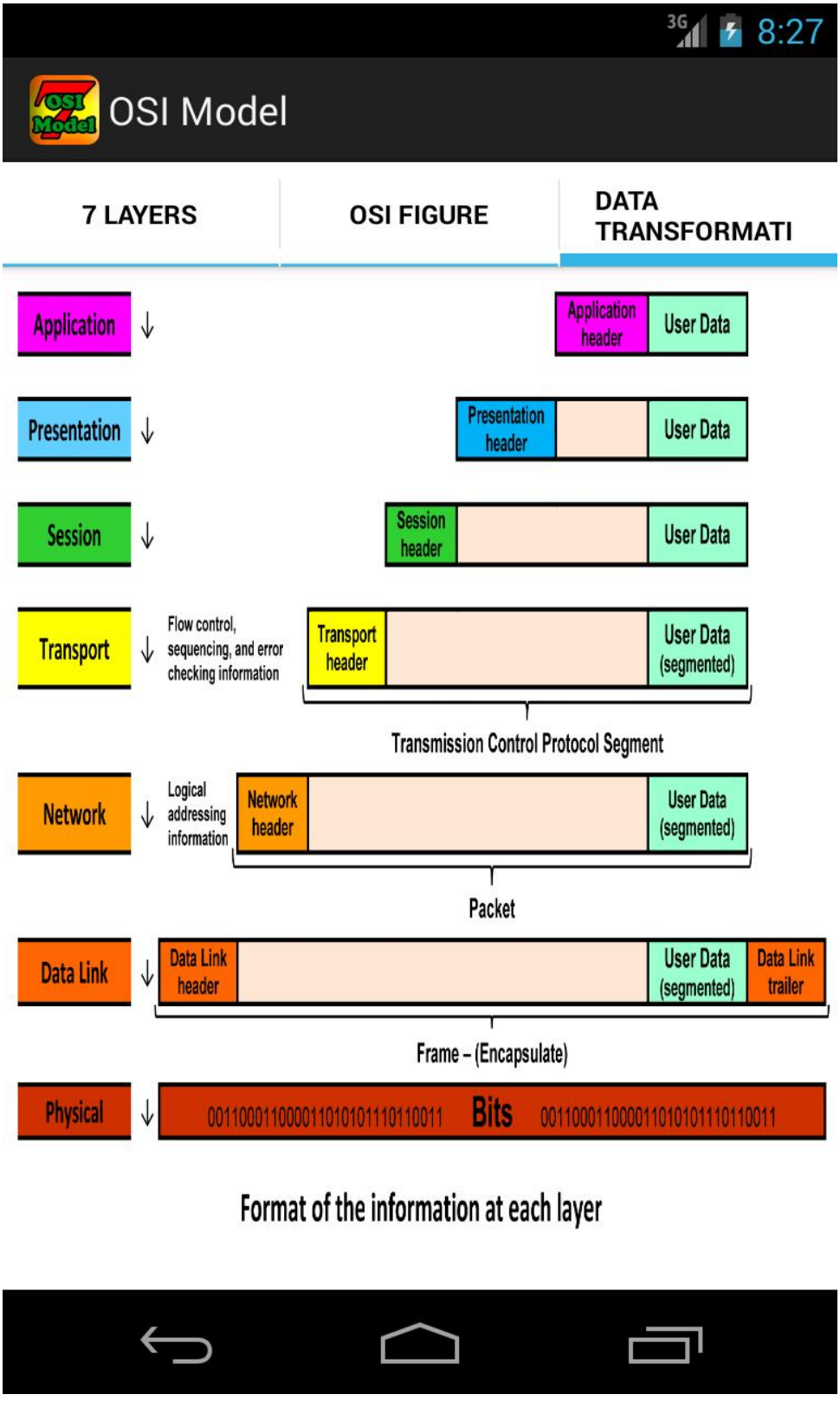
TCP vs UDP

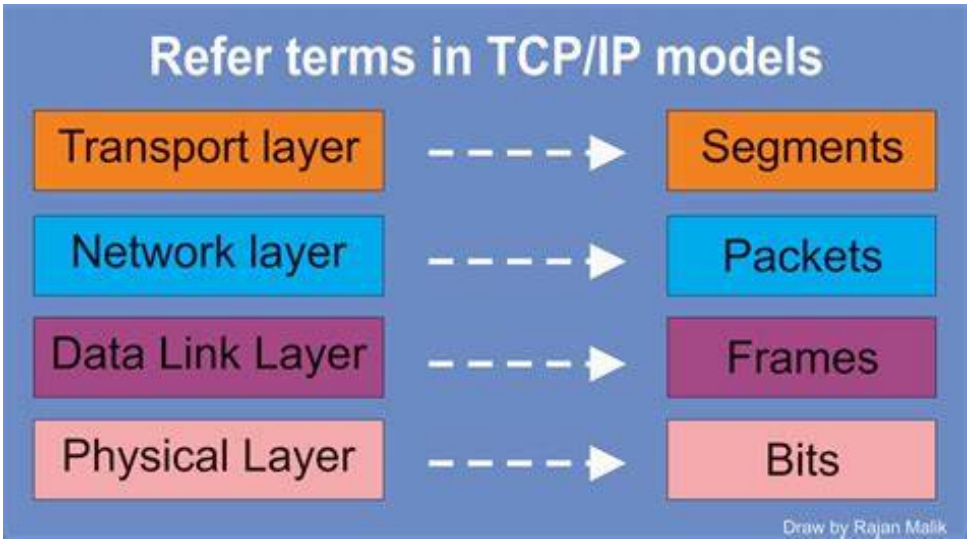

TCP Vs UDP Communication
TCP
Sender
Receiver
SYN
SYN ACK
ACK
UDP
Sender
Receiver
REQUEST
RESPONSE
RESPONSE
RESPONSE

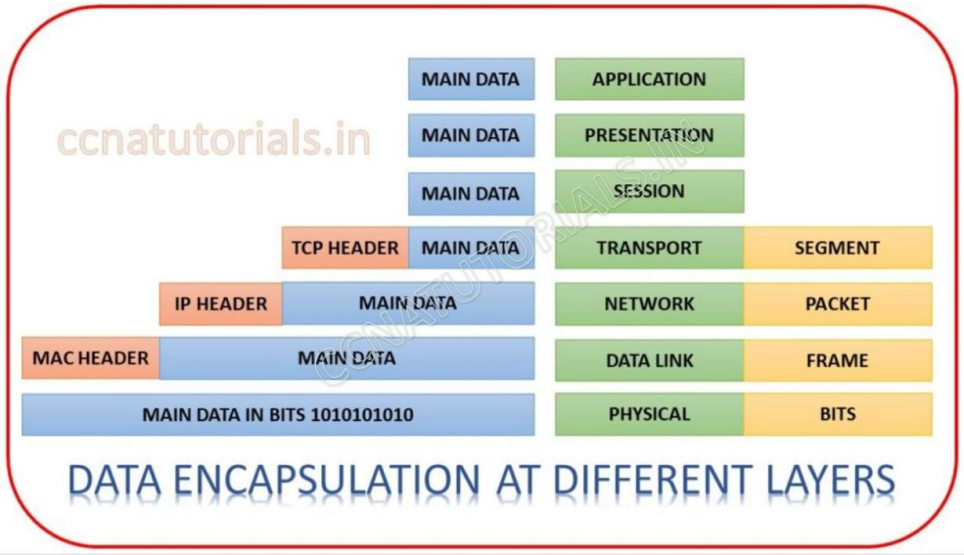


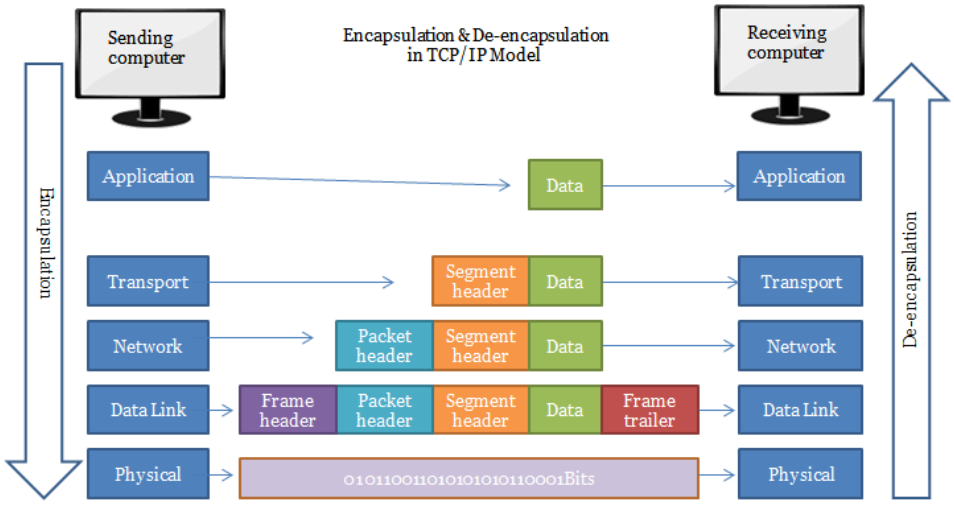


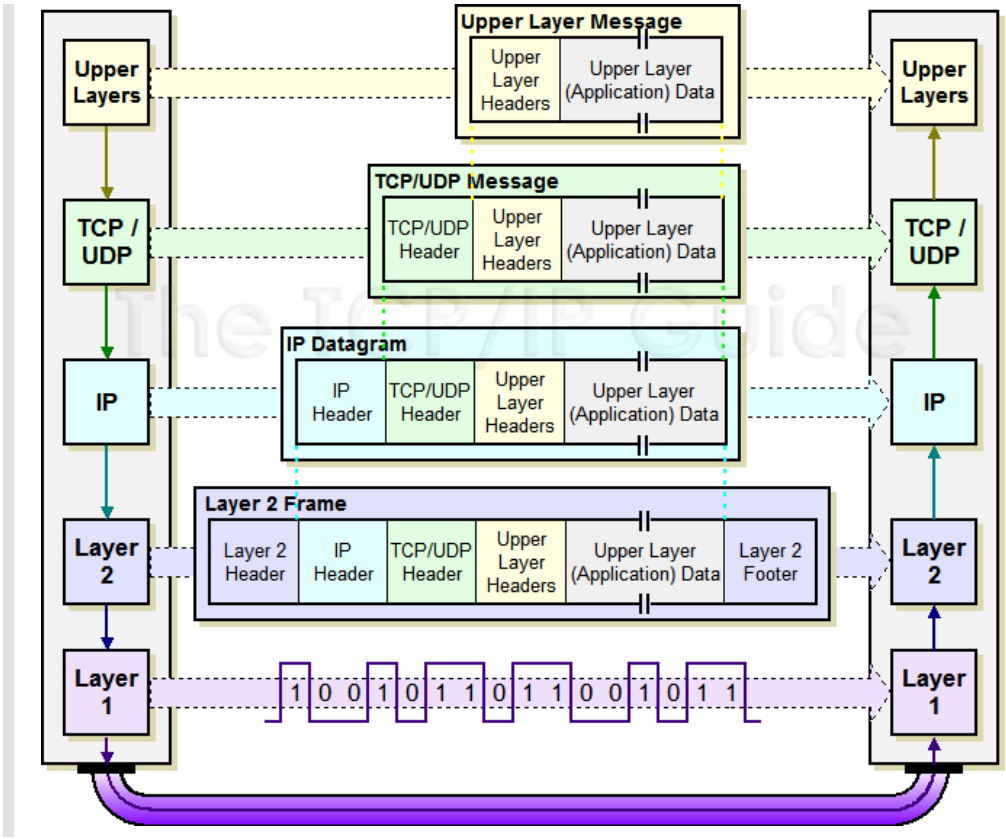
Encapsulation

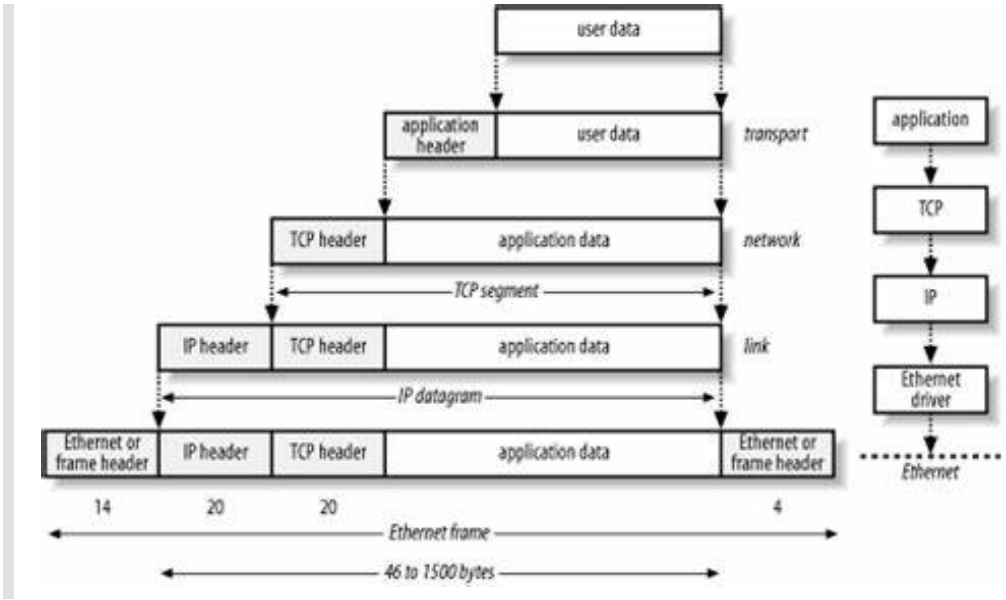

8:27
OSI Model
7 LAYERS
OSI FIGURE
DATA TRANSFORMATI
Application
Application header
User Data
Presentation
Presentation header
User Data
Session
Session header
User Data
Transport
Flow control, sequencing, and error checking information
Transport header
User Data (segmented)
Transmission Control Protocol Segment
Network
Logical addressing information
Network header
User Data (segmented)
Packet
Data Link
Data Link header
User Data (segmented)
Data Link trailer
Frame – (Encapsulate)
Physical
00110001100001101010111010011 Bits 00110001100001101010111010011
Format of the information at each layer

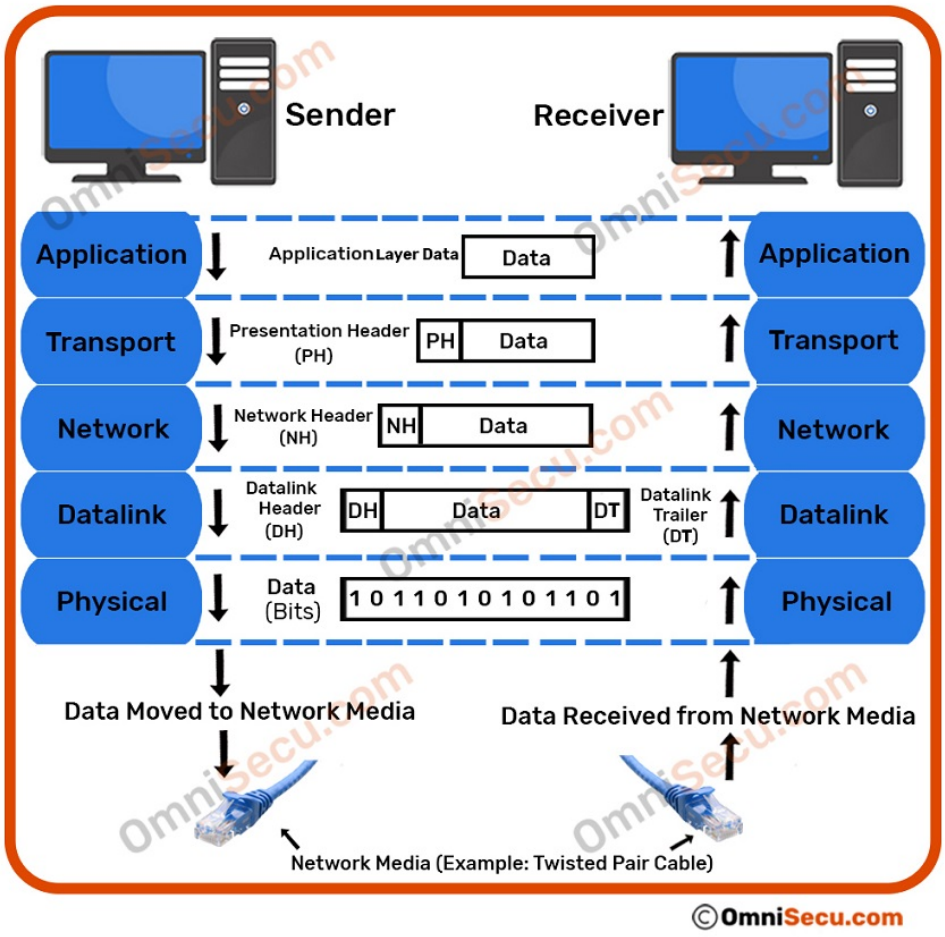

Refer terms in TCP/IP models
Transport layer
Segments
Network layer
Packets
Data Link Layer
Frames
Physical Layer
Bits
Draw by Rajan Malik


ccnatutorials.in
MAIN DATA
APPLICATION
MAIN DATA
PRESENTATION
MAIN DATA
SESSION
TCP HEADER
MAIN DATA
TRANSPORT
SEGMENT
IP HEADER
MAIN DATA
NETWORK
PACKET
MAC HEADER
MAIN DATA
DATA LINK
FRAME
MAIN DATA IN BITS 1010101010
PHYSICAL
BITS
DATA ENCAPSULATION AT DIFFERENT LAYERS

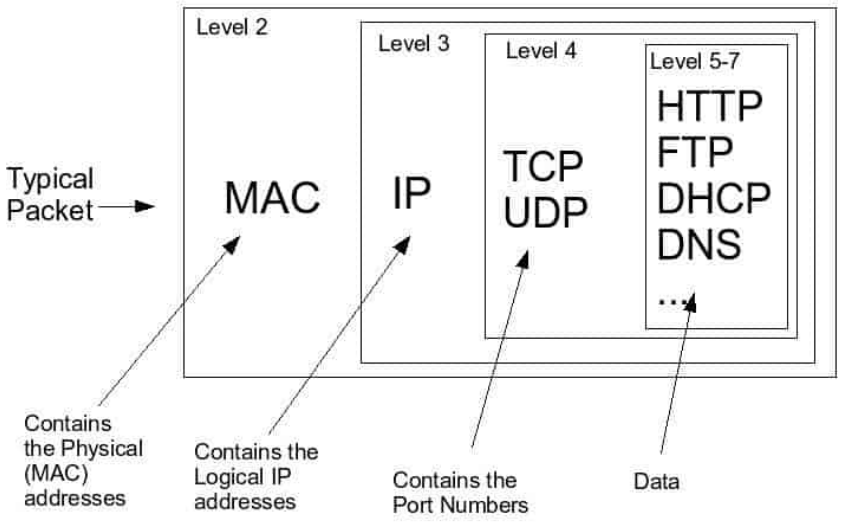

Sending computer
Encapsulation & De-encapsulation in TCP/IP Model
Receiving computer
Encapsulation
De-encapsulation
Application
Data
Application
Transport
Segment header
Data
Transport
Network
Packet header
Segment header
Data
Network
Data Link
Frame header
Packet header
Segment header
Data
Frame trailer
Data Link
Physical
0101100110101010101100001Bits
Physical


Upper Layer Message
Upper Layer Headers
Upper Layer (Application) Data
TCP/UDP Message
TCP/UDP Header
Upper Layer Headers
Upper Layer (Application) Data
IP Datagram
IP Header
TCP/UDP Header
Upper Layer Headers
Upper Layer (Application) Data
Layer 2 Frame
Layer 2 Header
IP Header
TCP/UDP Header
Upper Layer Headers
Upper Layer (Application) Data
Layer 2 Footer
Upper Layers
TCP / UDP
IP
Layer 2
Layer 1
1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1
The TCP/IP Guide

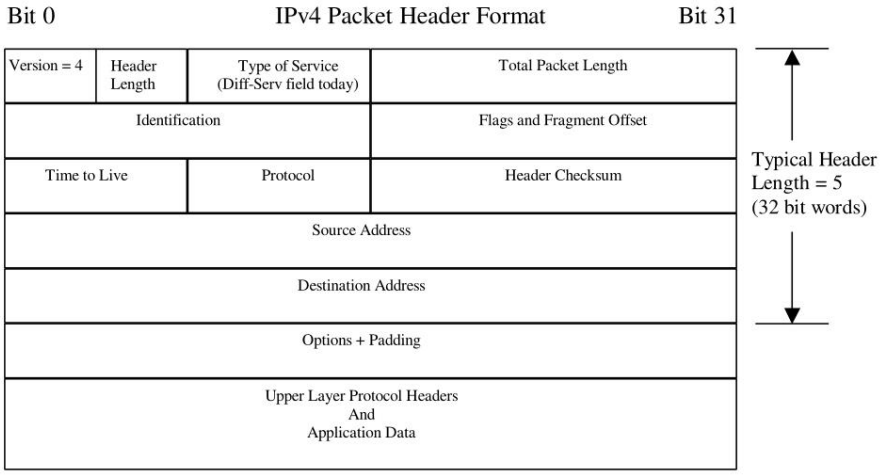

user data
application header
user data
transport
TCP header
application data
network
TCP segment
IP header
TCP header
application data
link
IP datagram
Ethernet or frame header
IP header
TCP header
application data
Ethernet or frame header
14
20
20
4
Ethernet frame
46 to 1500 bytes
application
TCP
IP
Ethernet driver
Ethernet

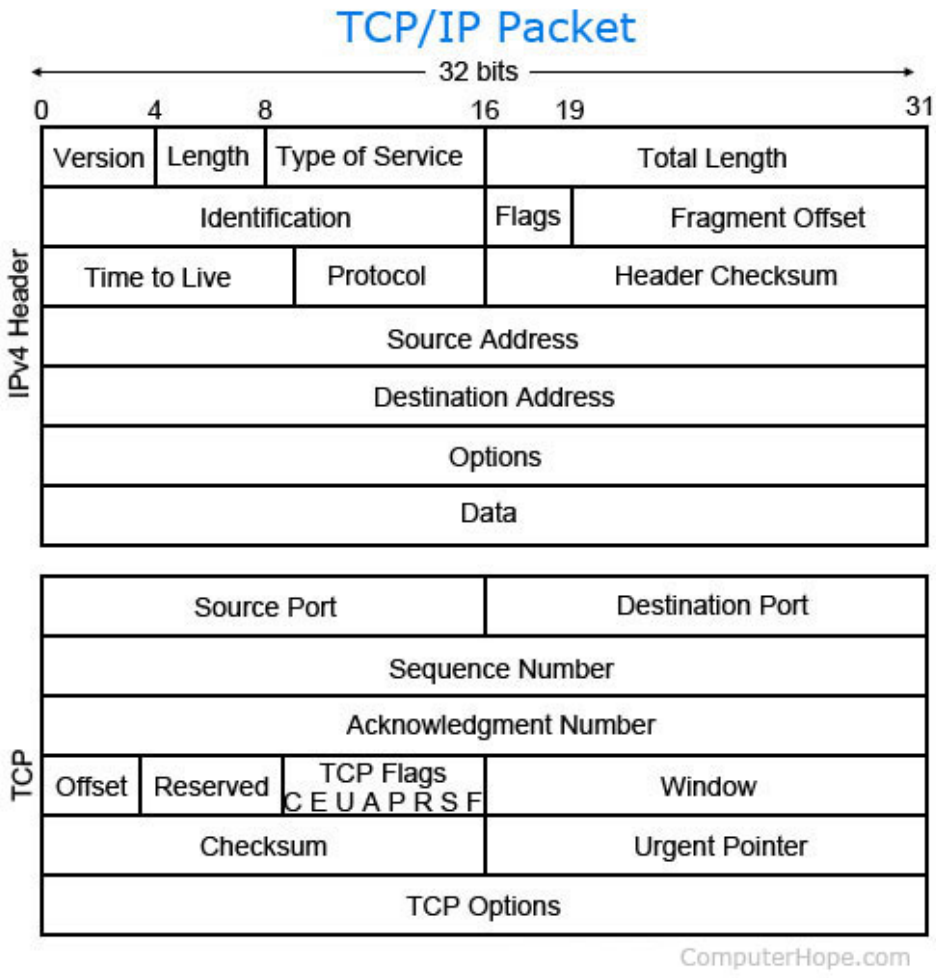

Sender
Receiver
Application
Application Layer Data
Data
Application
Transport
Presentation Header
(PH)
PH
Data
Transport
Network
Network Header
(NH)
NH
Data
Network
Datalink
Datalink
Header
(DH)
DH
Data
DT
Datalink
Trailer
(DT)
Datalink
Physical
Data
(Bits)
1 0 1 1 0 1 0 1 0 1 1 0 1
Physical
Data Moved to Network Media
Data Received from Network Media
Network Media (Example: Twisted Pair Cable)
©OmniSecu.com

Packet


Level 2
Level 3
Level 4
Level 5-7
Typical
Packet
MAC
IP
TCP
UDP
HTTP
FTP
DHCP
DNS
...
Contains
the Physical
(MAC)
addresses
Contains the
Logical IP
addresses
Contains the
Port Numbers
Data

IP Packet Format





Packet Switch

How TCP/IP Works

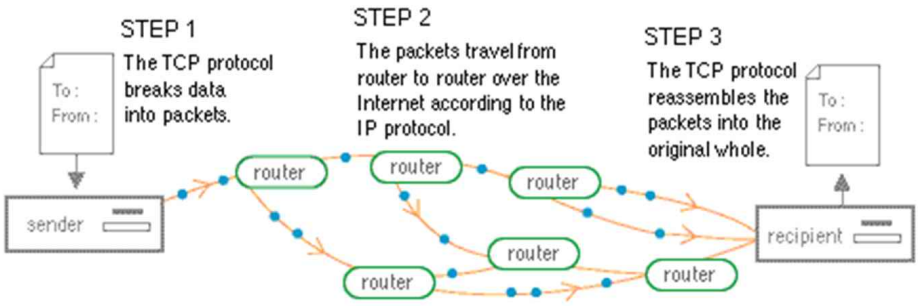


Figure 2. How data travels over the Net.

Dr. Vinton Cerf

MCI

Packet Switching

- To improve the efficiency of transferring information over a shared communication line, messages are divided into fixed-sized, numbered **packets**
- Network devices called routers are used to direct packets between networks

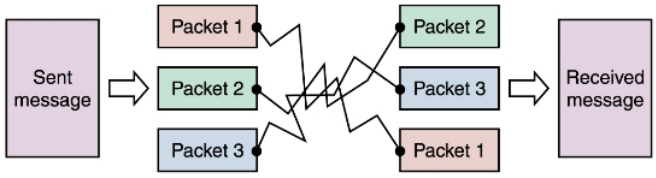


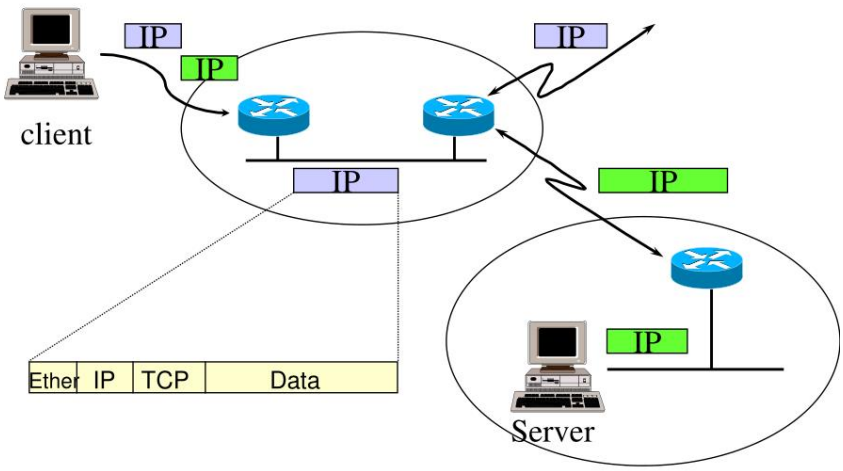
Figure 15.4
Messages sent by packet switching

Message is divided into packets

Packets are sent over the Internet by the most expedient route

Packets are reordered and then reassembled

11/27/06 Hofstra University - CSC005 15-18


Packet Switch Network
client
IP
IP
IP
IP
IP
IP
Ether
IP
TCP
Data
Server

IP4 vs IP6

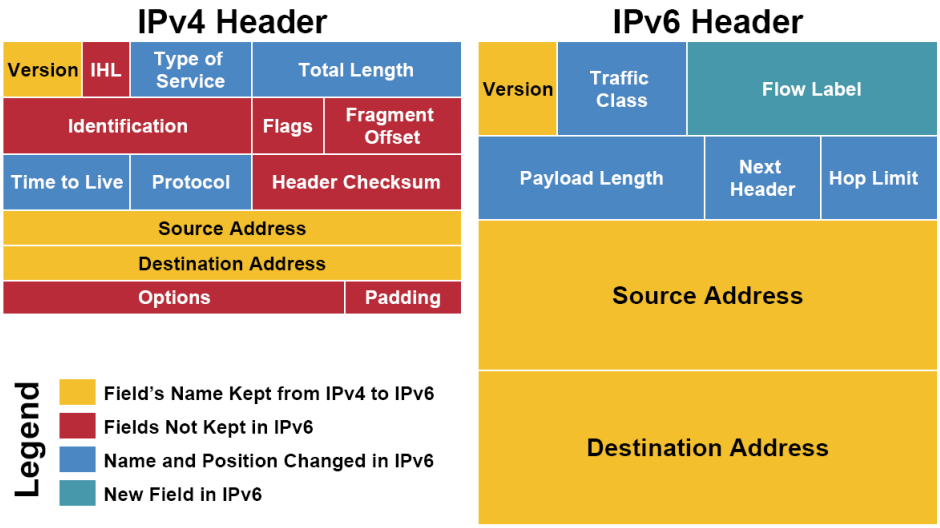
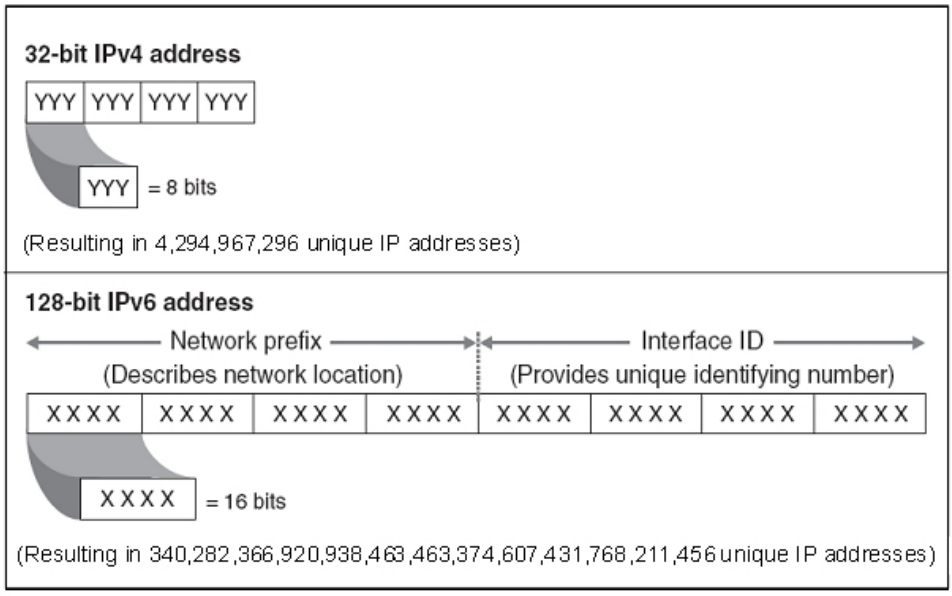

IPv4 Header
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding
IPv6 Header
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
Legend
Field's Name Kept from IPv4 to IPv6
Fields Not Kept in IPv6
Name and Position Changed in IPv6
New Field in IPv6

Figure 1: Comparison of IPv6 and IPv4 Address Scheme



Source: GAO.

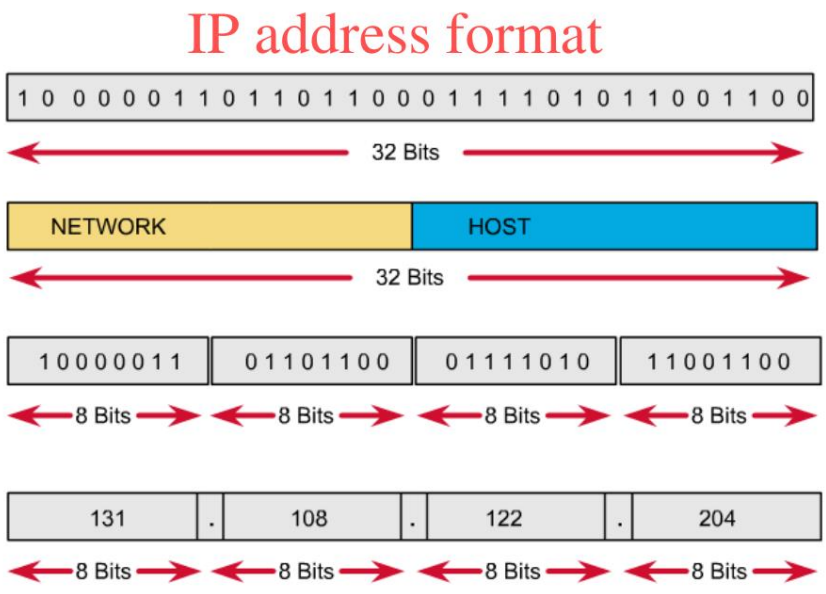
Differences Between IPv4 and IPv6

Feature	IPv4	IPv6
Fragmentation	Performed by routers and sending host	Performed only by sending host
Address Resolution	Broadcast ARP Request frames	Multicast Neighbor Solicitation messages
Manage multicast group membership	IGMP	Multicast listener discovery
Router Discovery	ICMP Router Discovery (optional)	ICMPv6 Router Solicitation and Router Advertisement (required)
DNS host records	A records	AAAA records
DNS reverse lookup zones	IN-ADDR.ARPA	IP6.ARPA
Minimum packet size	576 bytes	1280 bytes

IPv4/IPv6 Differences

	IPv4	IPv6
Address	32 bits (4 bytes) 12:34:56:78	128 bits (16 bytes) 1234:5678:9abc:defo:1234:5678:9abc:defo
Packet size	576 bytes required, fragmentation optional	1280 bytes required without fragmentation
Packet fragmentation	Routers and sending hosts	Sending hosts only
Packet header	Does not identify packet flow for QoS handling	Contains Flow Label field that specifies packet flow for QoS handling
	Includes a checksum	Does not include a checksum
	Includes options up to 40 bytes	Extension headers used for optional data
DNS records	Address (A) records, maps host names	Address (AAAA) records, maps host names
	Pointer (PTR) records, IN-ADDR.ARPA DNS domain	Pointer (PTR) records, IP6.ARPA DNS domain
Address configuration	Manual or via DHCP	Stateless address autoconfiguration (SLAAC) using Internet Control Message Protocol version 6 (ICMPv6) or DHCPv6
IP to MAC resolution	broadcast ARP	Multicast Neighbor Solicitation
Local subnet group management	Internet Group Management Protocol (IGMP)	Multicast Listener Discovery (MLD)
Broadcast	Yes	No
Multicast	Yes	Yes
IPSec	optional, external	required

IP Address

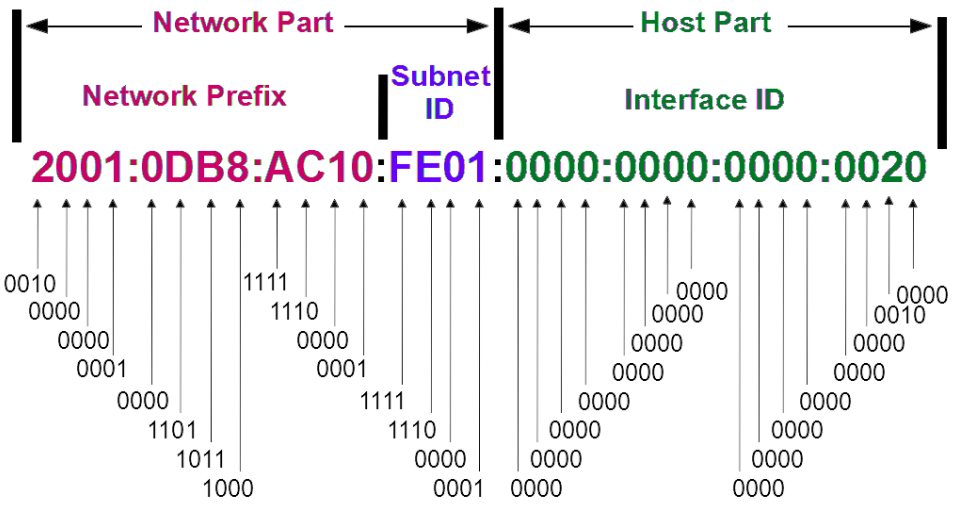


Class	Private Address Ranges
Class A	10.0.0.0 – 10.255.255.255
Class B	172.16.0.0 – 172.31.255.255
Class C	192.168.0.0 – 192.168.255.255
Loopback	127.0.0.0 – 127.255.255.255 (127.0.0.1)

IPv6 Address Structure

128 Bits, Expressed in Hex (Hexadecimal) with 3 parts

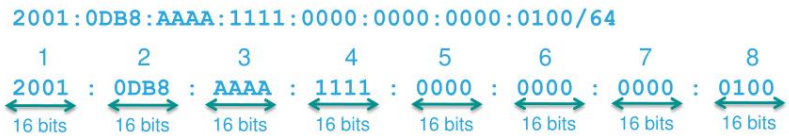
This is the usual breakdown but it can be broken down in other ways



IPv6 Address Notation

One Hex digit = 4 bits

Dec.	Hex.	Binary	Dec.	Hex.	Binary
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111



- IPv6 addresses are 128-bit addresses represented in:
  - Eight 16-bit segments or “hextets” (not a formal term)
  - Hexadecimal (non-case sensitive) between 0000 and FFFF

IP Range

Global Addresses

Class	First Octet value	Range	No. of Network	No. of Hosts / Network
A	00000000 – 01111111 (0 – 127)	1.0.0.1 – 126.255.255.254	126	2^{24} - 2
B	10000000 – 10111111 (128 – 191)	128.1.0.1 – 191.255.255.254	16000	65000
C	11000000 – 11011111 (192 – 223)	192.0.1.1 – 223.255.255.254	2 Million	254
D	11100000 – 11101111 (224 – 239)	224.0.0.0 – 239.255.255.255	Multicast addresses	
E	11110000 – 11111111 (240 – 255)	240.0.0.0 – 254.255.255.254	Future use	

In class **A 127.0.0.1 – 127.255.255.255** addresses are reserved for loopback & diagnostic purpose.

IP Address Ranges

IP Address Class	First Octet Binary Value	First Octet Decimal Value	Possible Number of Hosts
Class A	1-126	00000001 to 01111110*	16,777,214
Class B	128-191	10000000 to 10111111	65,534
Class C	192-223	11000000 to 11011111	254

*127 (011111111) is a Class A address reserved for loopback testing and cannot be assigned to a network.

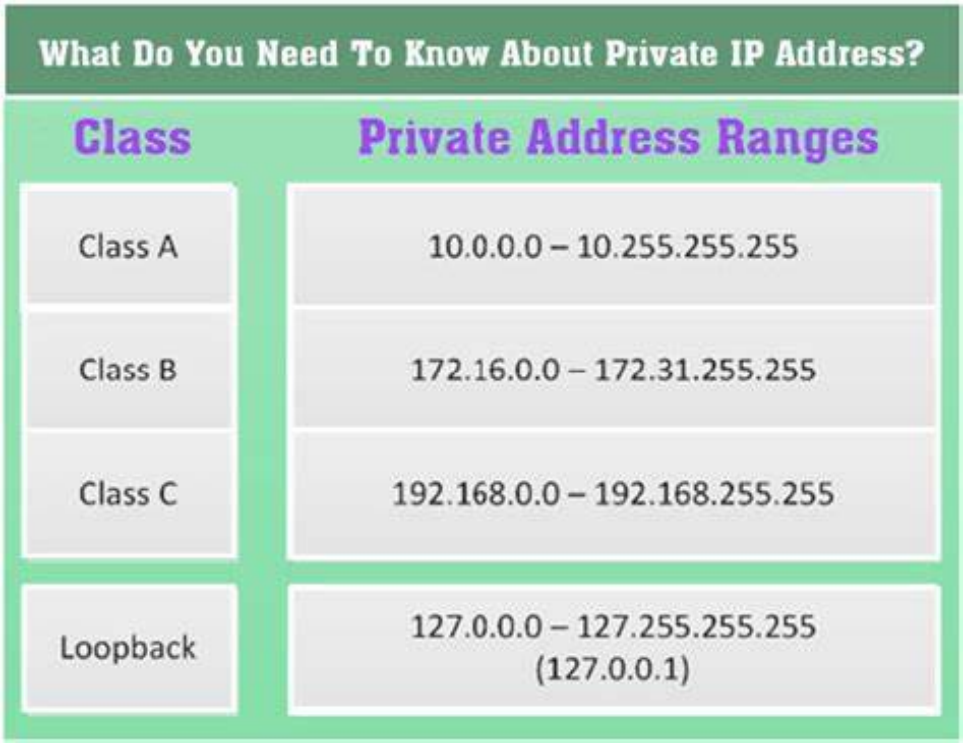
MyShared

Private IP ranges

- Often it is necessary to connect devices to the network, but not to the internet. RFC 1918 manages the private IP addresses that cannot appear on the internet, but are reserved for private use.
- Private IP ranges managed by IANA:

Class	From	To	No. Of hosts
1 x A class	10.0.0.0	10.255.255.255	2^{24} = 16.777.216
16 x B class	172.16.0.0	172.31.255.255	2^{20} = 1.048.576
256 x C class	192.168.0.0	192.168.255.255	2^{16} = 65.536

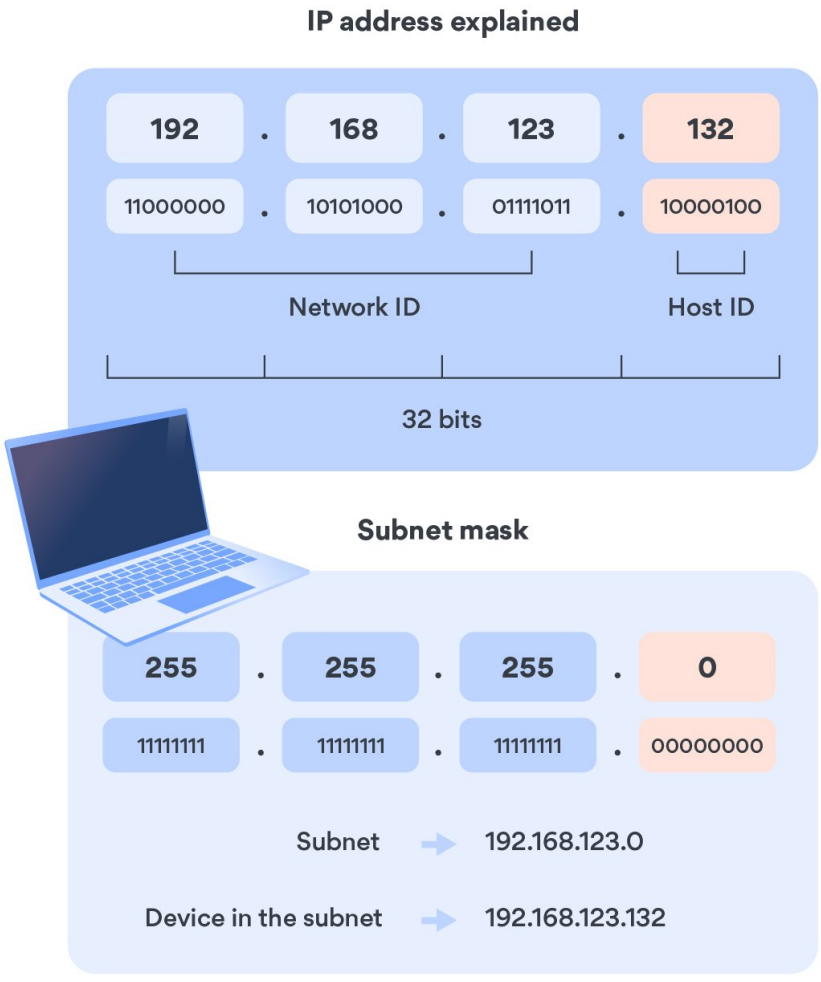
- example:
  - 192.168.1.0/24 (mask: 255.255.255.0 | 256 hosts) - 256 networks
  - 172.17.0.0/16 (mask: 255.255.0.0 | 65.536 hosts) 256 networks

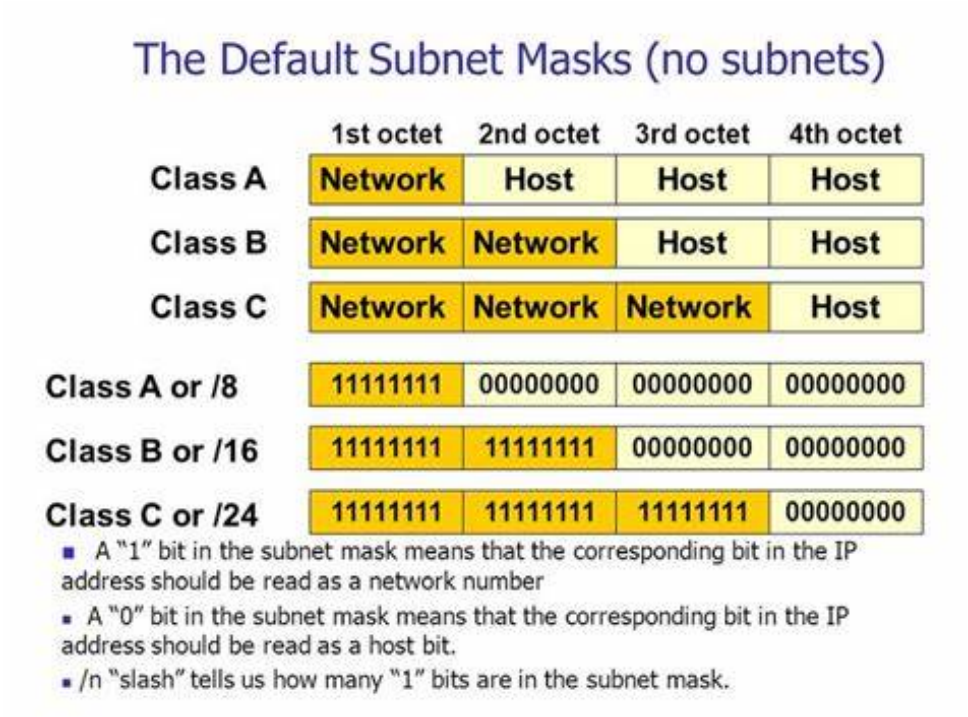


Subnet Mask

Subnet Mask

Suffix	Hosts	32-Borrowed=CIDR	2^Borrowed = Hosts	Binary=> dec = Suffix
.255	1	/32	0	11111111
.254	2	/31	1	11111110
.252	4	/30	2	11111100
.248	8	/29	3	11111000
.240	16	/28	4	11110000
.224	32	/27	5	11100000
.192	64	/26	6	11000000
.128	128	/25	7	10000000

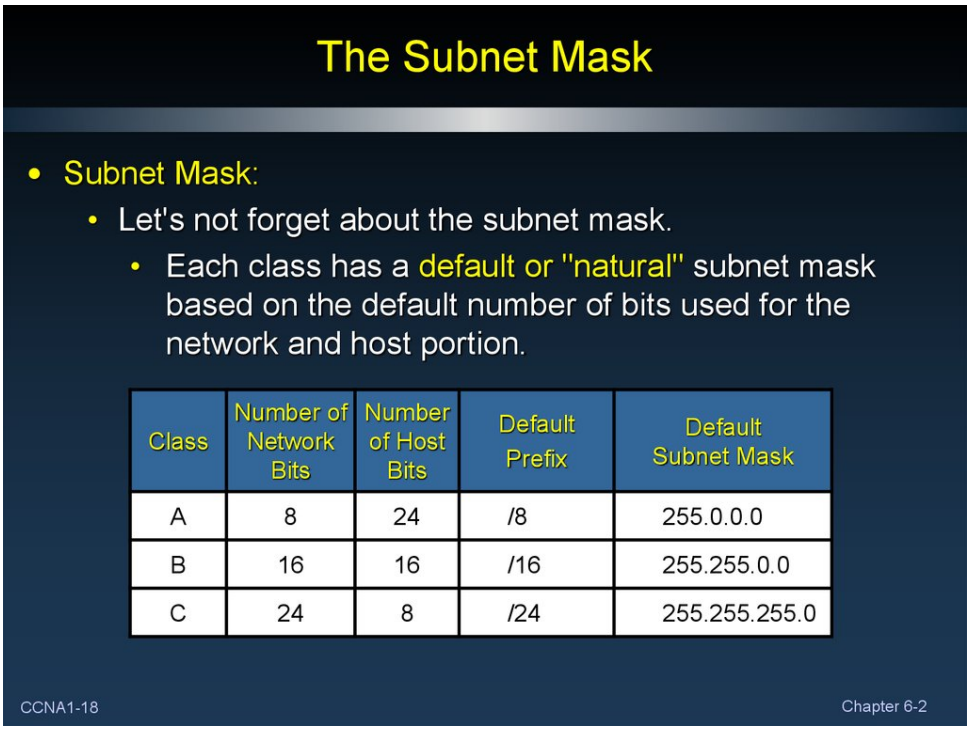




The Default Subnet Masks (no subnets)

	1st octet	2nd octet	3rd octet	4th octet
Class A	Network	Host	Host	Host
Class B	Network	Network	Host	Host
Class C	Network	Network	Network	Host
Class A or /8	11111111	00000000	00000000	00000000
Class B or /16	11111111	11111111	00000000	00000000
Class C or /24	11111111	11111111	11111111	00000000

- A "1" bit in the subnet mask means that the corresponding bit in the IP address should be read as a network number
- A "0" bit in the subnet mask means that the corresponding bit in the IP address should be read as a host bit.
- /n "slash" tells us how many "1" bits are in the subnet mask.



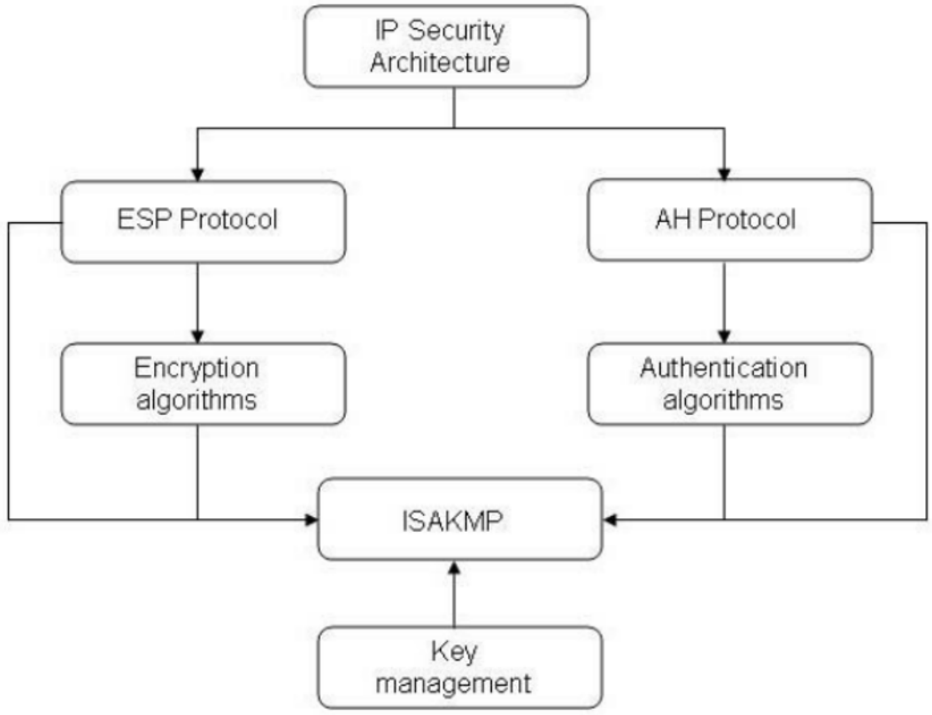
The Subnet Mask

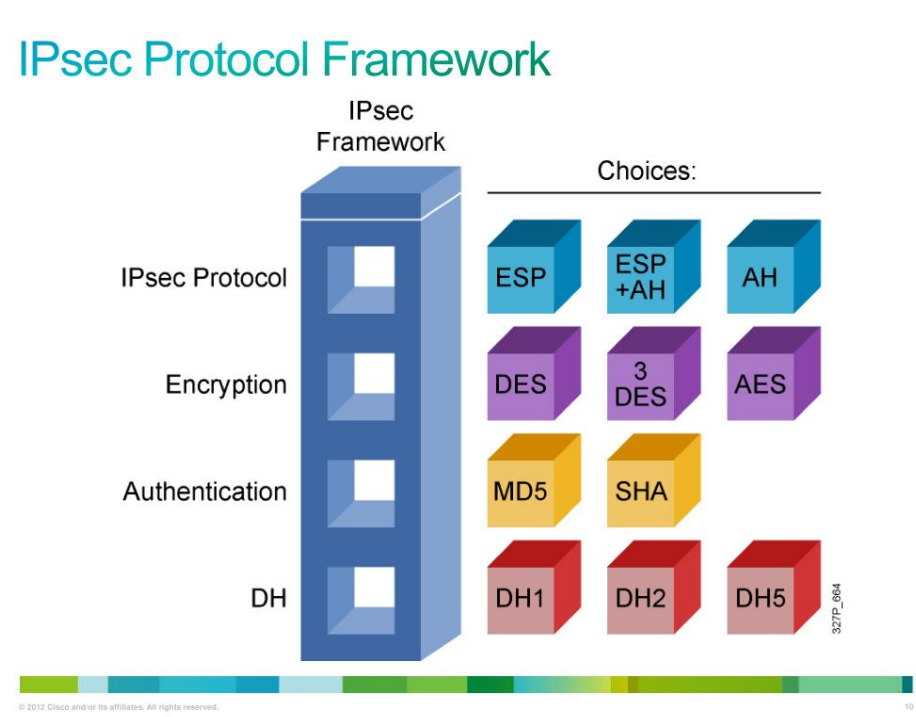
- Subnet Mask:
  - Let's not forget about the subnet mask.
    - Each class has a default or "natural" subnet mask based on the default number of bits used for the network and host portion.

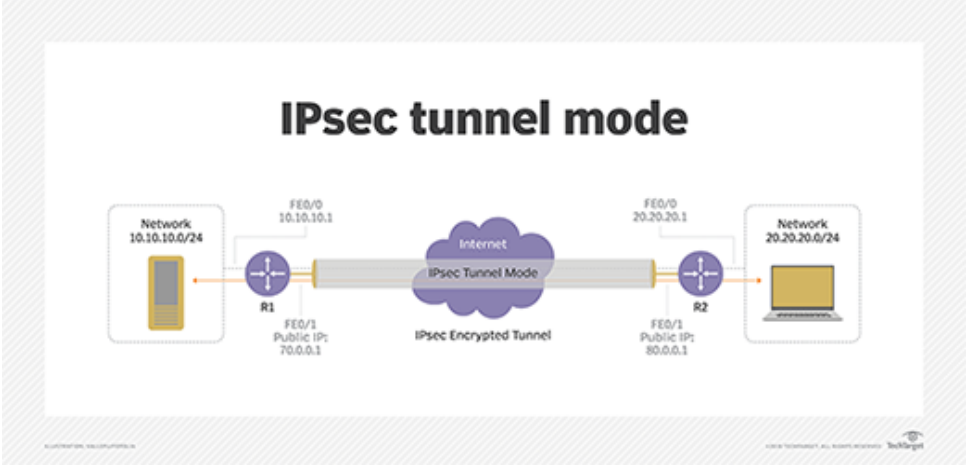
Class	Number of Network Bits	Number of Host Bits	Default Prefix	Default Subnet Mask
A	8	24	/8	255.0.0.0
B	16	16	/16	255.255.0.0
C	24	8	/24	255.255.255.0

CCNA1-18 Chapter 6-2

IPSec


IP Security Architecture
ESP Protocol
AH Protocol
Encryption algorithms
Authentication algorithms
ISAKMP
Key management


IPsec Protocol Framework
IPsec Framework
Choices:
IPsec Protocol
Encryption
Authentication
DH
ESP
ESP +AH
AH
DES
3 DES
AES
MD5
SHA
DH1
DH2
DH5
327P_864
© 2012 Cisco and/or its affiliates. All rights reserved.
10

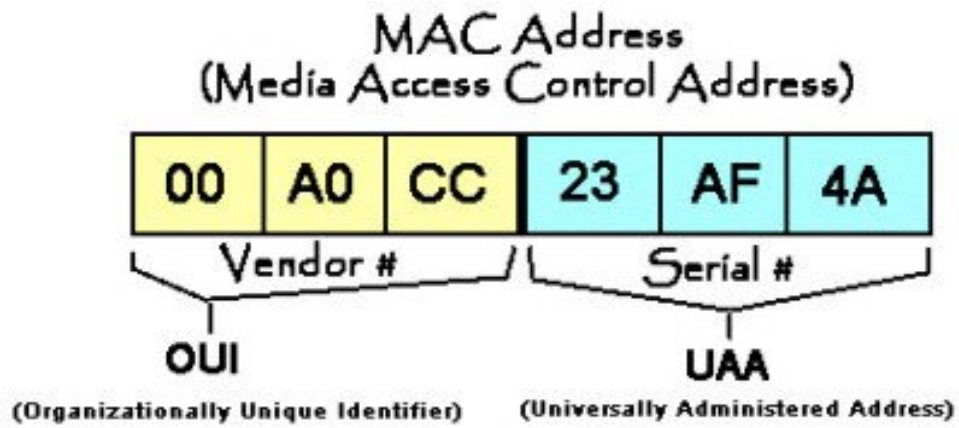

IPsec tunnel mode
Network 10.10.10.0/24
FE0/0 10.10.10.1
R1
FE0/1 Public IP: 70.0.0.1
Internet
IPsec Tunnel Mode
IPsec Encrypted Tunnel
FE0/0 20.20.20.1
R2
FE0/1 Public IP: 80.0.0.1
Network 20.20.20.0/24

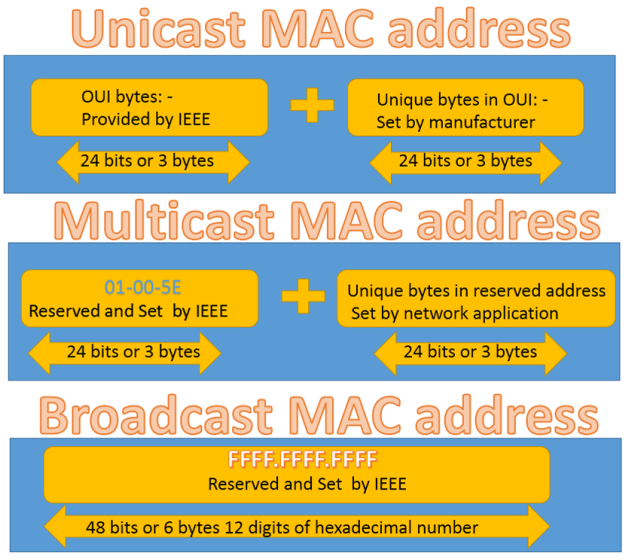
IPsec: Network Layer Security

- network-layer secrecy:
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- network-layer authentication
  - destination host can authenticate source IP address
- two principal protocols:
  - authentication header (AH) protocol
  - encapsulation security payload (ESP) protocol
- for both AH and ESP, source, destination handshake:
  - create network-layer logical channel called a security association (SA)
- each SA unidirectional.
- uniquely determined by:
  - security protocol (AH or ESP)
  - source IP address
  - 32-bit connection ID

8: Network Security 8-1

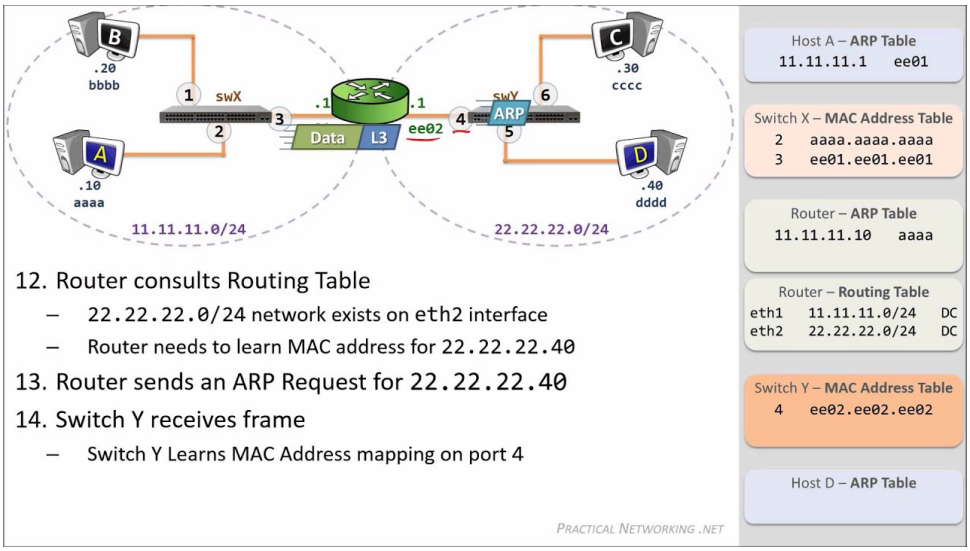
MAC Address - DataLink Layer





ARP(Address Resolution Protocl) / RARP (ReverseAddress Resolution Protocl)

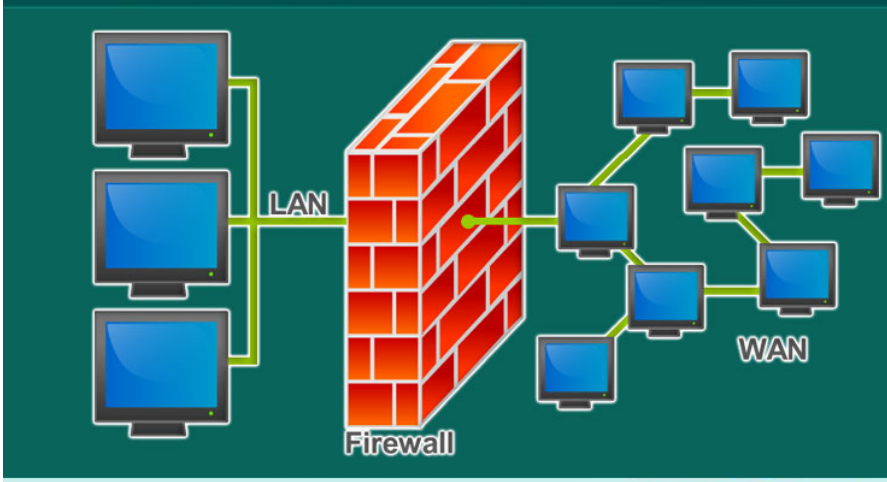
- ARP: resolve IP Address to MAC Address
- RARP: resolve MAC Address to IP Address



Network Equipment

Firewall

Configure Firewall & Internet Security of the QuickBooks Desktop



What is firewall?

A firewall is nothing but a network security system that monitors and controls over all your incoming and outgoing network traffic based on advanced and a defined set of security rules.

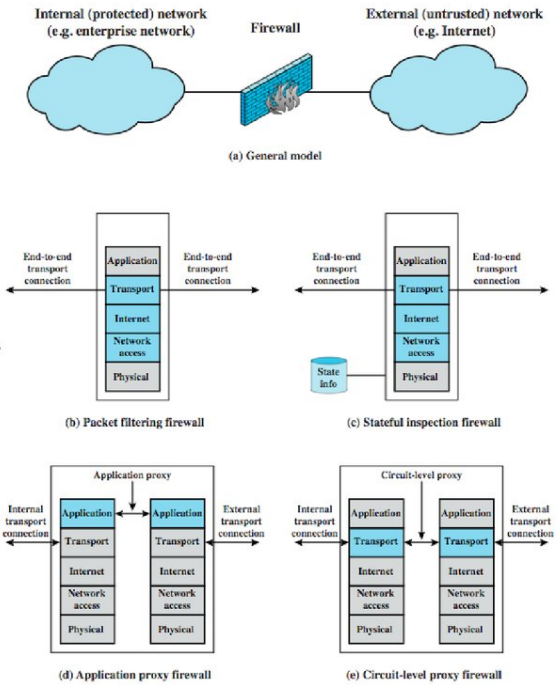
It simply prevents unauthorized access to or from a private network. Used to enhance the security of computers connected to a network, such as LAN or the Internet. Considered as an integral part of a comprehensive security framework for your network.

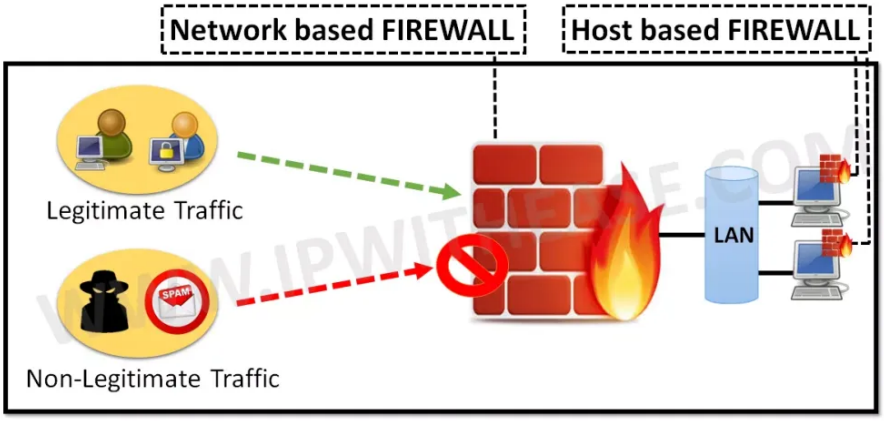
WiZXPERT

Types of Firewalls

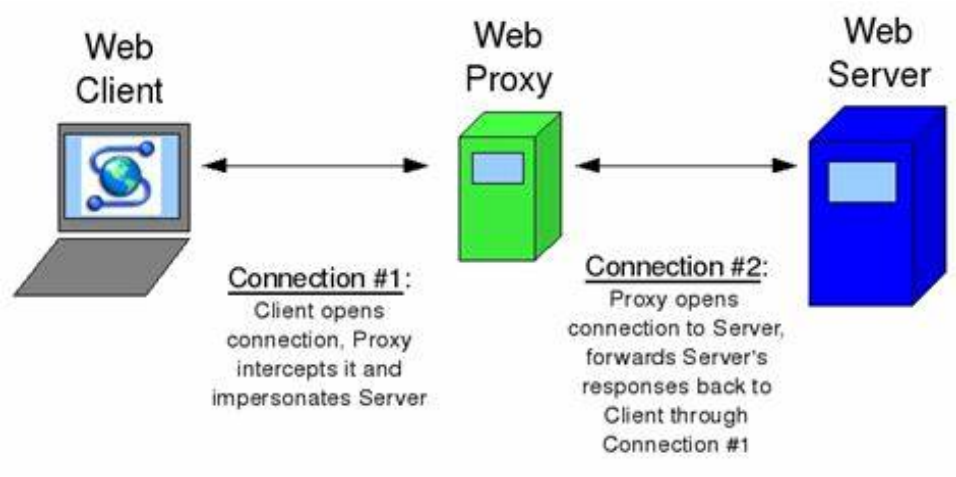
Positive (negative) filter: Allow (reject) packets that meet a criteria

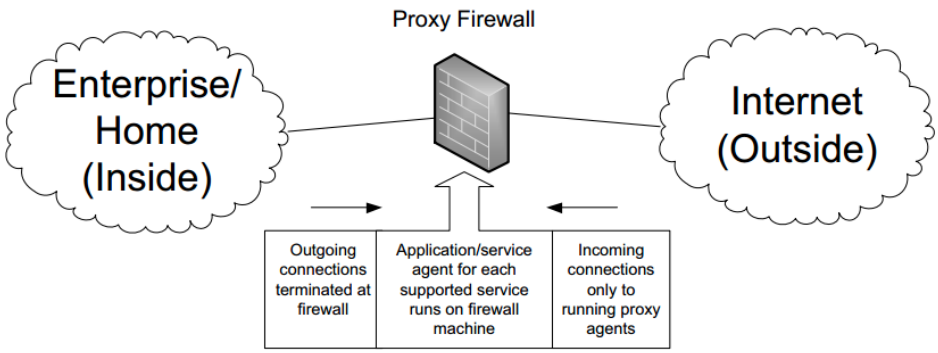
Stateful inspection: Keeps track of TCP connections





Proxy





Proxy Cache

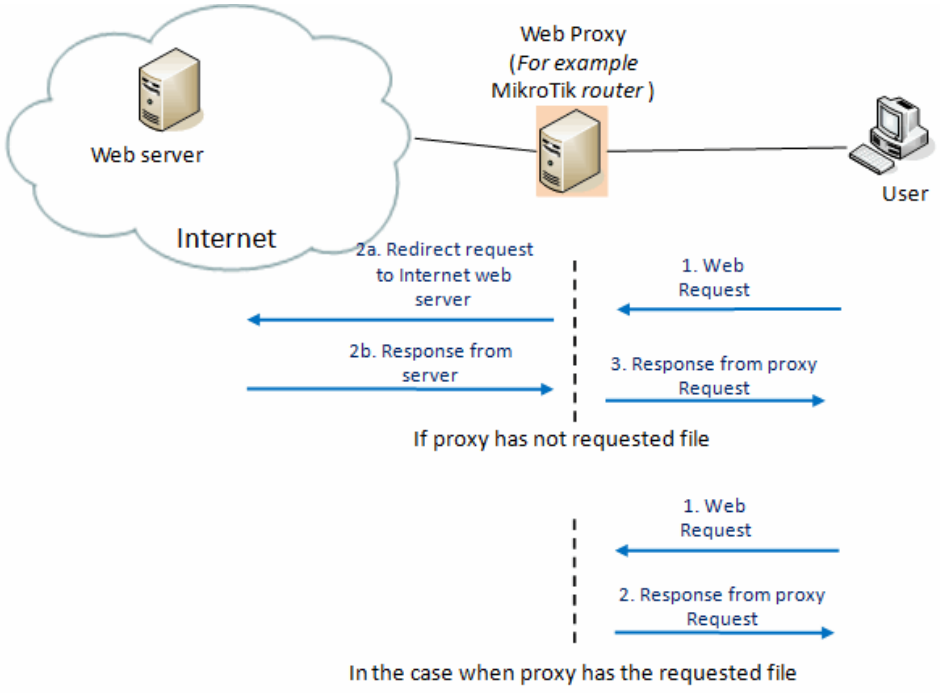
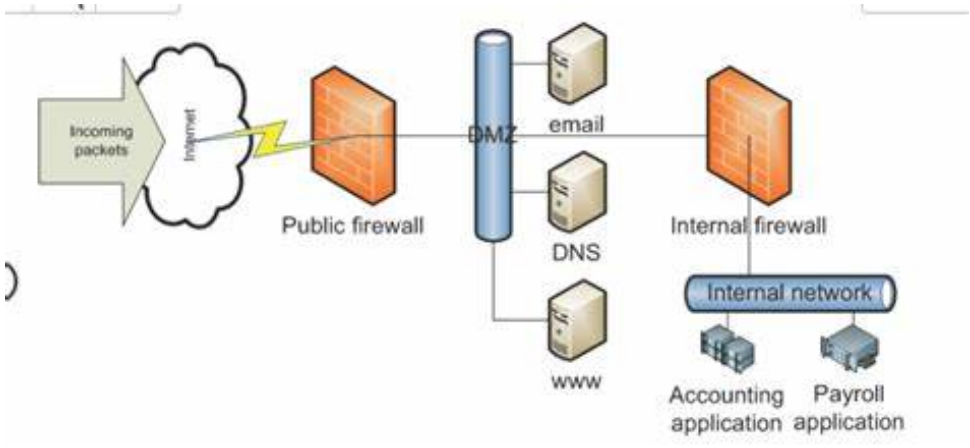


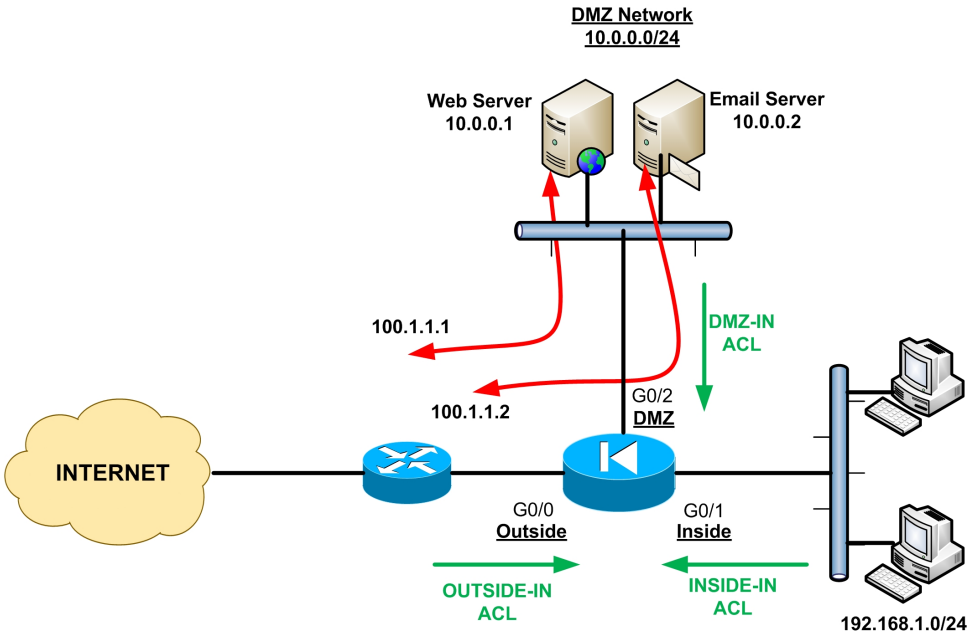
Figure 10.1. Web proxy basic operation scheme

ACL (Access Control List)

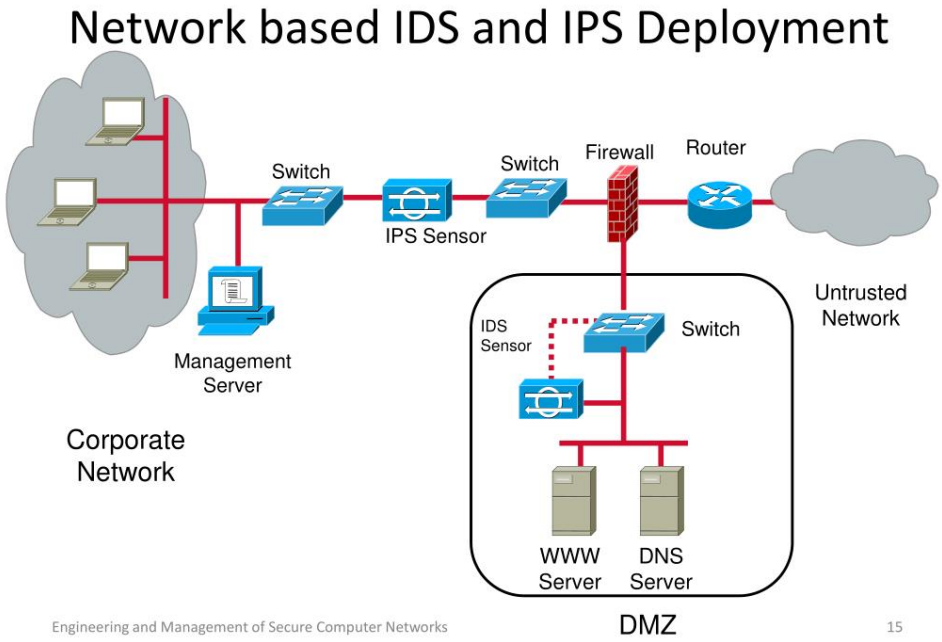
Parameter	ACL	Firewall
Asset Type	Feature on Layer 3 devices and Firewalls	Hardware or Software
Stateful/Stateless inspection	Performs stateless inspection	Performs Stateful inspection
Scope wrt OSI	Upto Layer 4	Upto Layer 7
Security	Low	High
Intrusion detection	Not possible	Possible
Target deployment	Setups requiring low level of security	Setups requiring higher level of security

DMZ





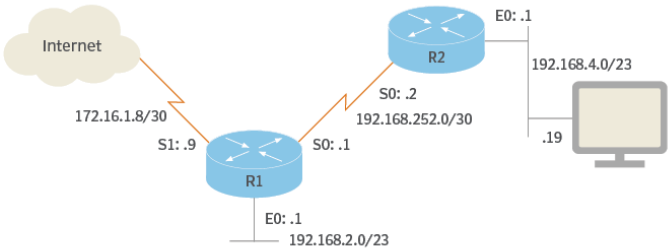
IPS/IDS



Routing Table

Subnet masks, prefixes and routing

In this diagram, R1 receives a packet addressed to 192.168.5.19, a host that's connected to R2's LAN. Using a binary AND operation on the address and its mask, R1 finds 192.168.4.0 and forwards the packet out the S0 interface to R2, which will perform the same prefix calculation. R2 determines it should send the packet on interface E0 and deliver it to host 5.19.



R1'S ROUTING TABLE

Prefix	192.168.2.0	192.168.4.0	192.168.252.0	0.0.0.0
Mask	255.255.254.0	255.255.254.0	255.255.255.252	0.0.0.0
Outgoing interface	E0	S0 to R2	S0	S1 to internet (default)

SOURCE: NETWORK ARCHITECT TERRY SLATTERY

Routing Protocol

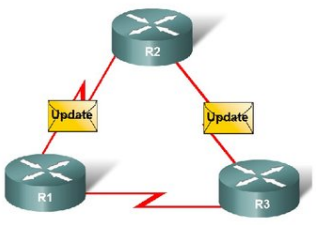
Dynamic IP Routing Protocols

Routing Protocols learn and **dynamically** share information about the networks connected to each other therefore these protocols are called **dynamic protocols**.

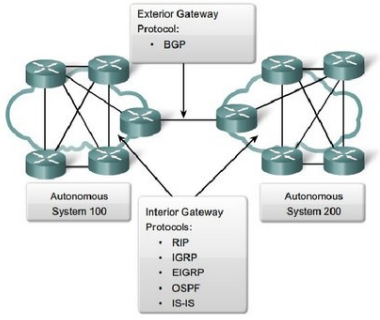
There are quite many dynamic routing protocols for routing IP packets. The most common protocols are:

- **RIP** (Routing Information Protocol);
- **IGRP** (Interior Gateway Routing Protocol);
- **EIGRP** (Enhanced Interior Gateway Routing Protocol);
- **OSPF** (Open Shortest Path First);
- **IS-IS** (Intermediate System-to-Intermediate System) *(pronounced "i-s i-s" or more commonly "Eye-Sis")*;
- **BGP** (Border Gateway Protocol).

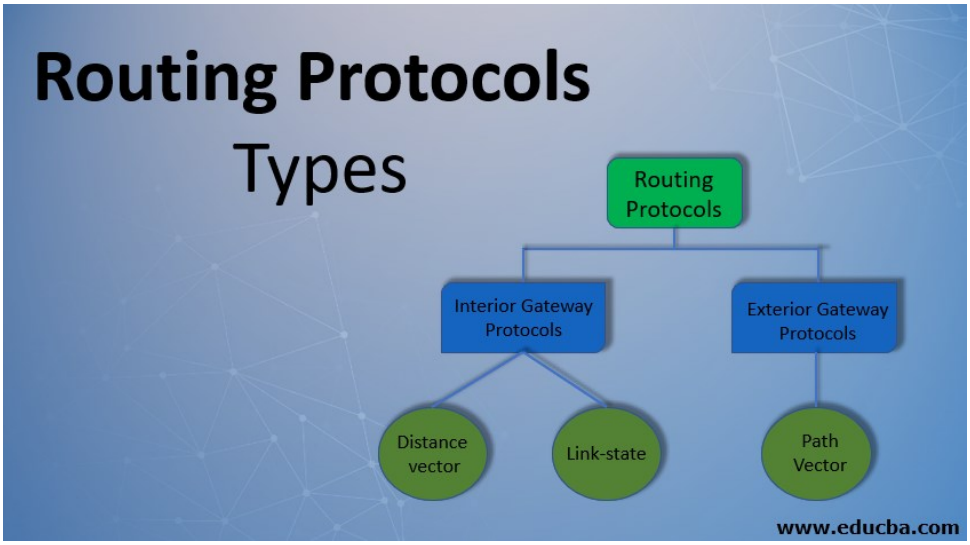
Routers Dynamically Pass Updates

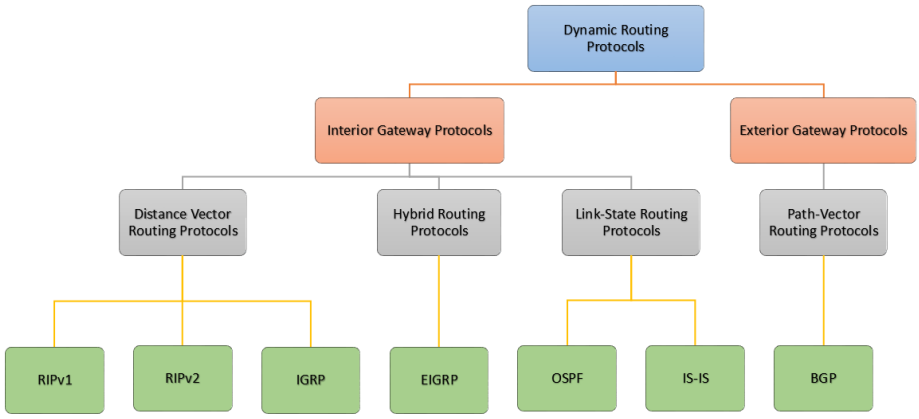


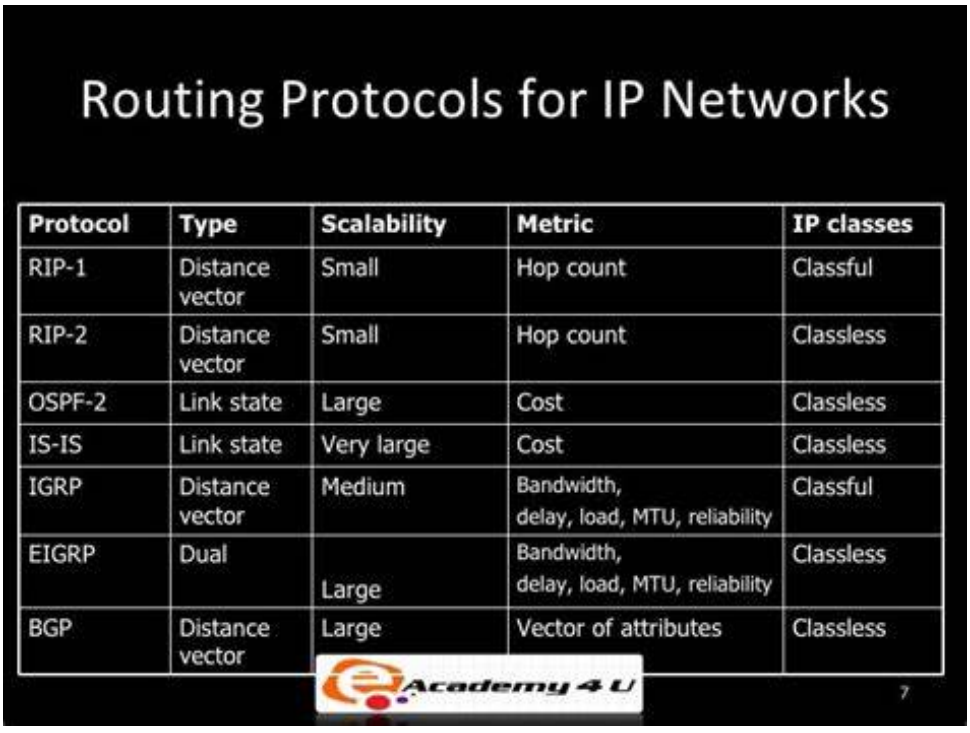
IGP vs. EGP Routing Protocols



16

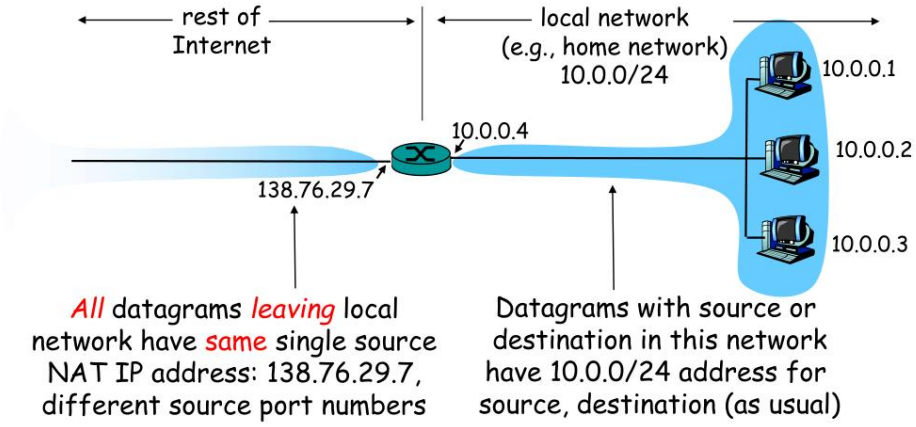






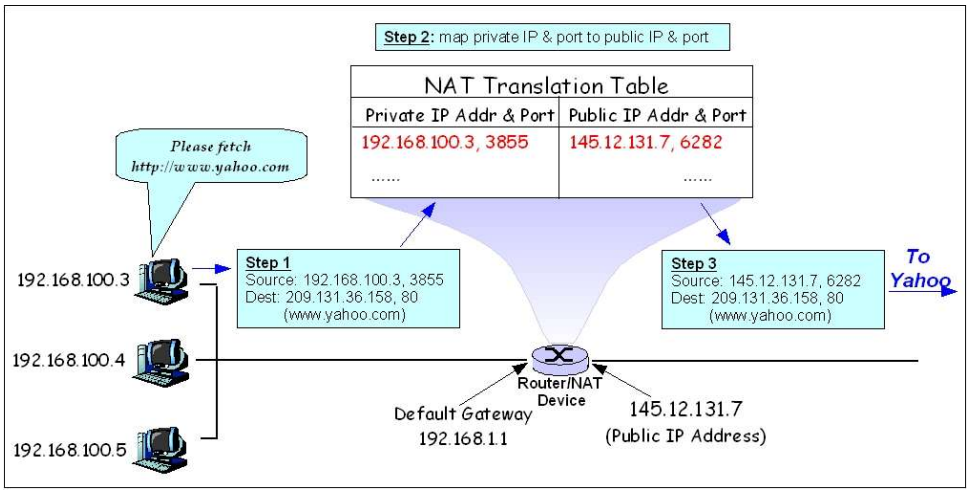
NAT(Network Address Translation)

NAT: Network Address Translation

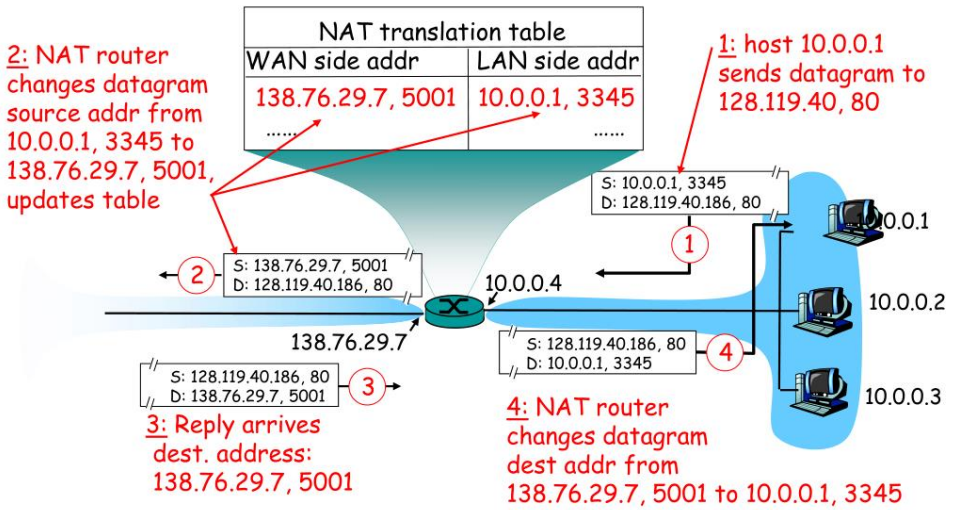


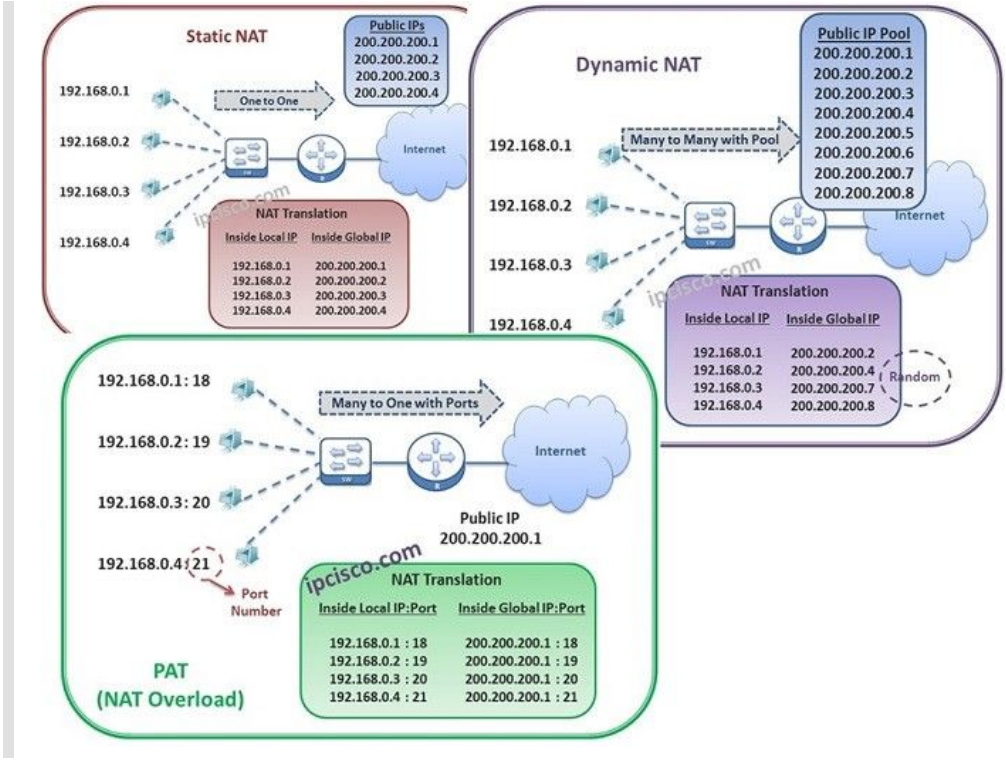
All datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

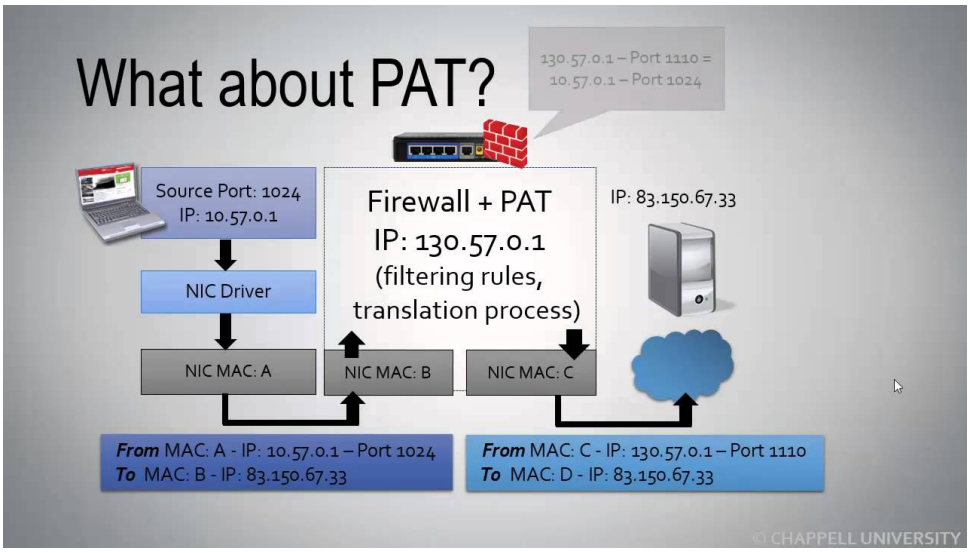


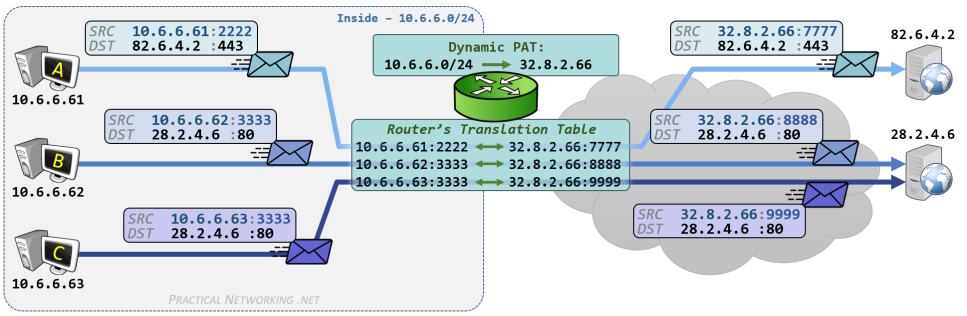
NAT: Network Address Translation



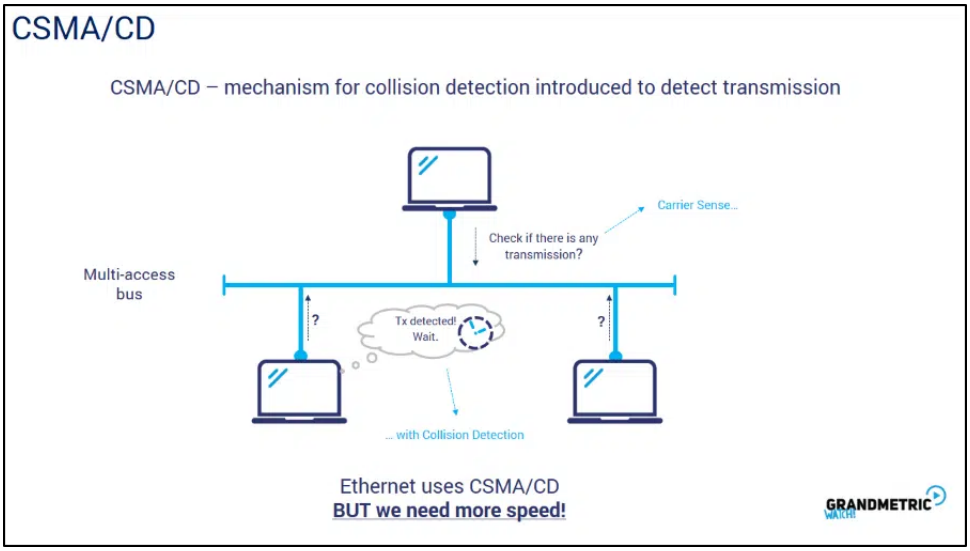


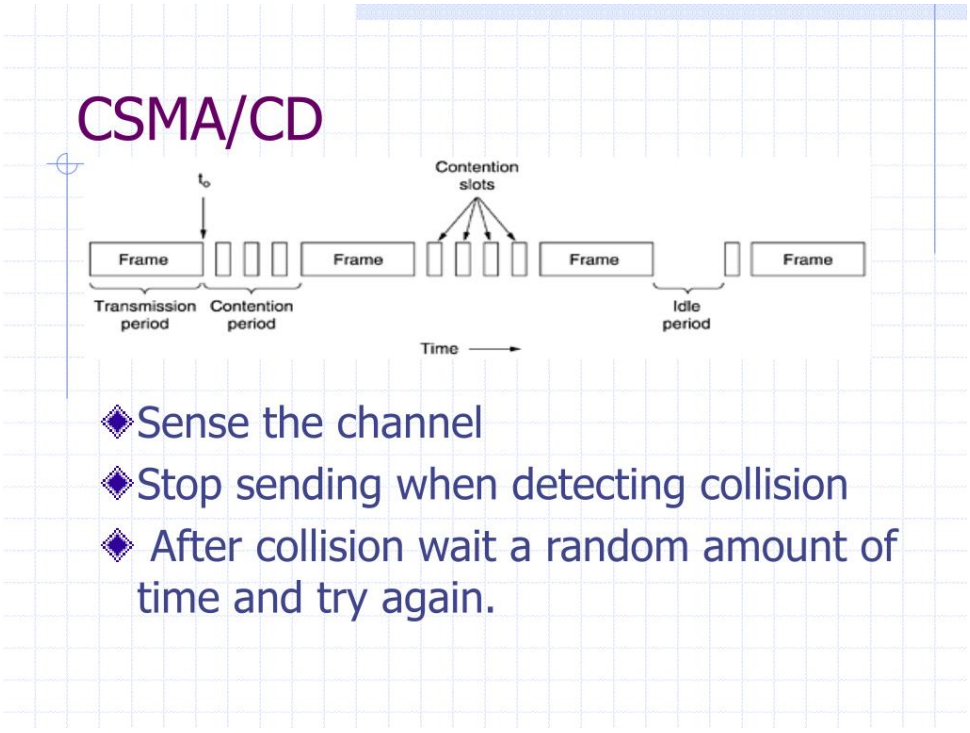
PAT(Port Address Translation)

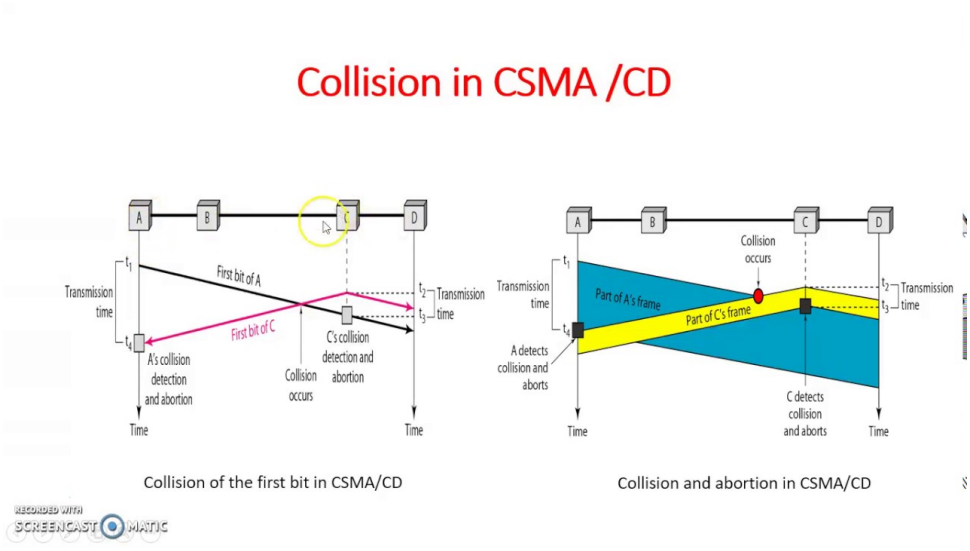




CSMA/CD


CSMA/CD
CSMA/CD – mechanism for collision detection introduced to detect transmission
Carrier Sense...
Check if there is any transmission?
Multi-access bus
Tx detected! Wait.
... with Collision Detection
Ethernet uses CSMA/CD
BUT we need more speed!
GRANDMETRIC

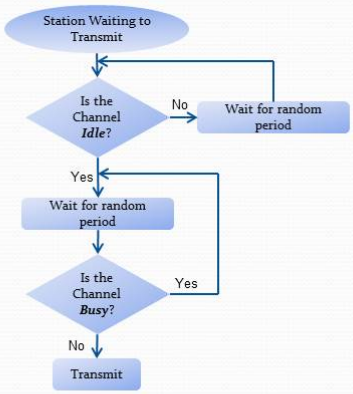

CSMA/CD
Contention slots
Frame
Frame
Frame
Frame
Transmission period
Contention period
Idle period
Time
Sense the channel
Stop sending when detecting collision
After collision wait a random amount of time and try again.


Collision in CSMA /CD
A
B
C
D
Transmission time
First bit of A
First bit of C
A's collision detection and abortion
Collision occurs
C's collision detection and abortion
Time
Collision occurs
Part of A's frame
Part of C's frame
A detects collision and aborts
C detects collision and aborts
Collision of the first bit in CSMA/CD
Collision and abortion in CSMA/CD
RECORDED WITH SCREENCAST-O-MATIC

CSMA/CA

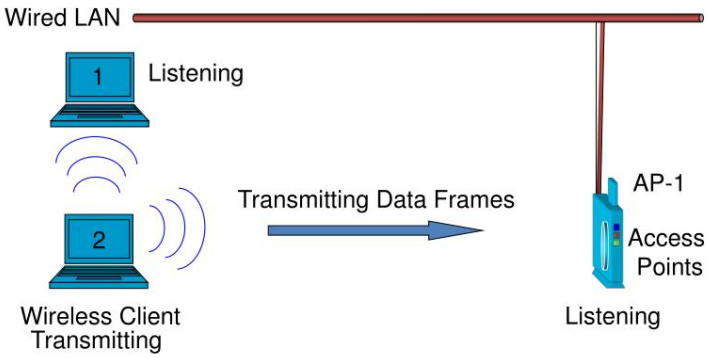
CSMA/CA

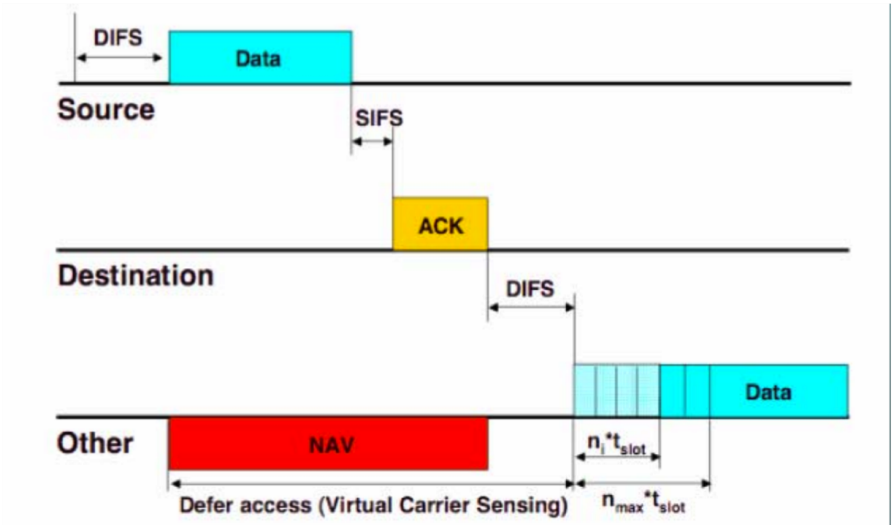
- CSMA/CA is a wireless network multiple access method in which:
  - A carrier sensing scheme is used.
  - A node wishing to transmit data has to first listen to the channel for a predetermined amount of time whether or not another node is transmitting on channel within the wireless range. If the channel is sensed *"idle"*, then the node is permitted to begin the transmission process. If the channel is sensed as *"busy"*, the node defers its transmission for a random period of time.
  - State of channel *"Idle"* or *"Busy"* is based on CS mechanism, which will explained later in the presentation



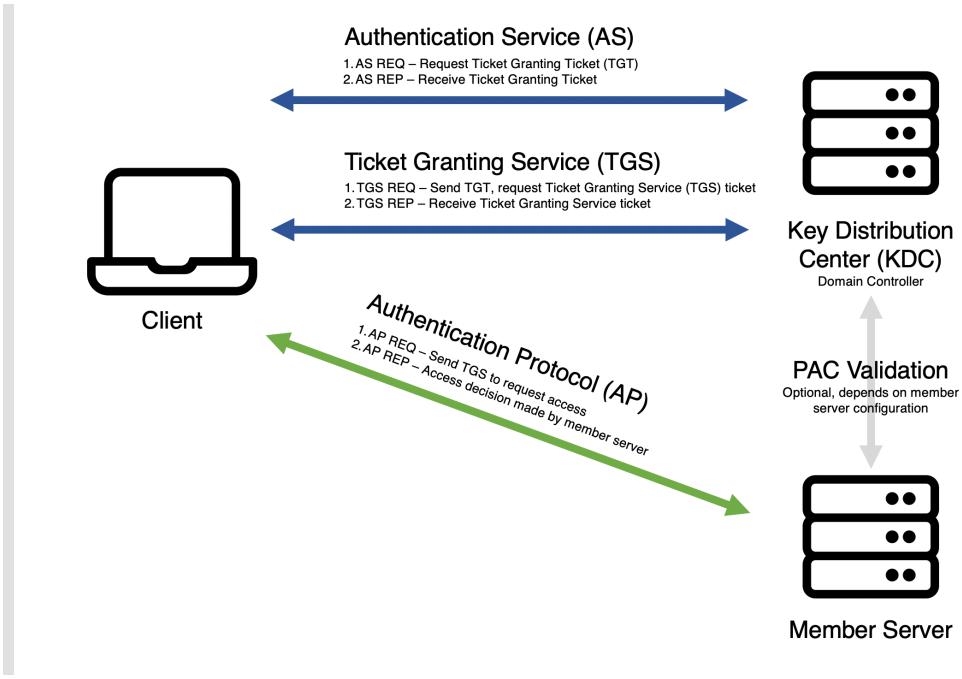
CSMA/CA Collision Handling

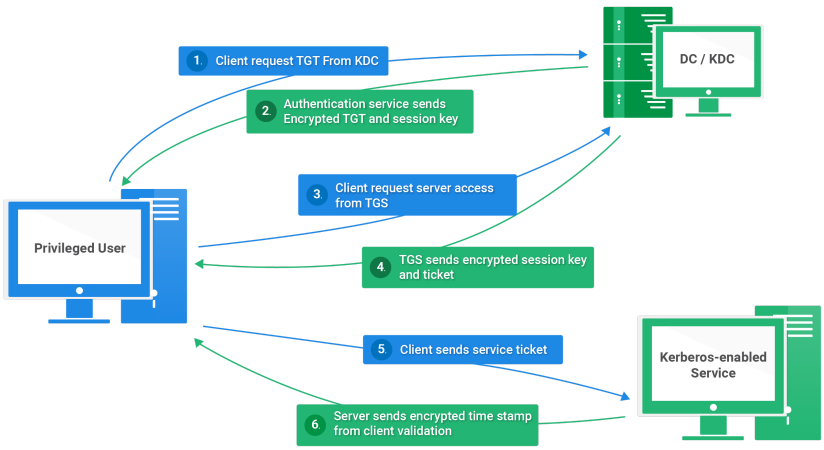
- 802.11 standard employs half-duplex radios-radios capable of transmission or reception-but not both simultaneously

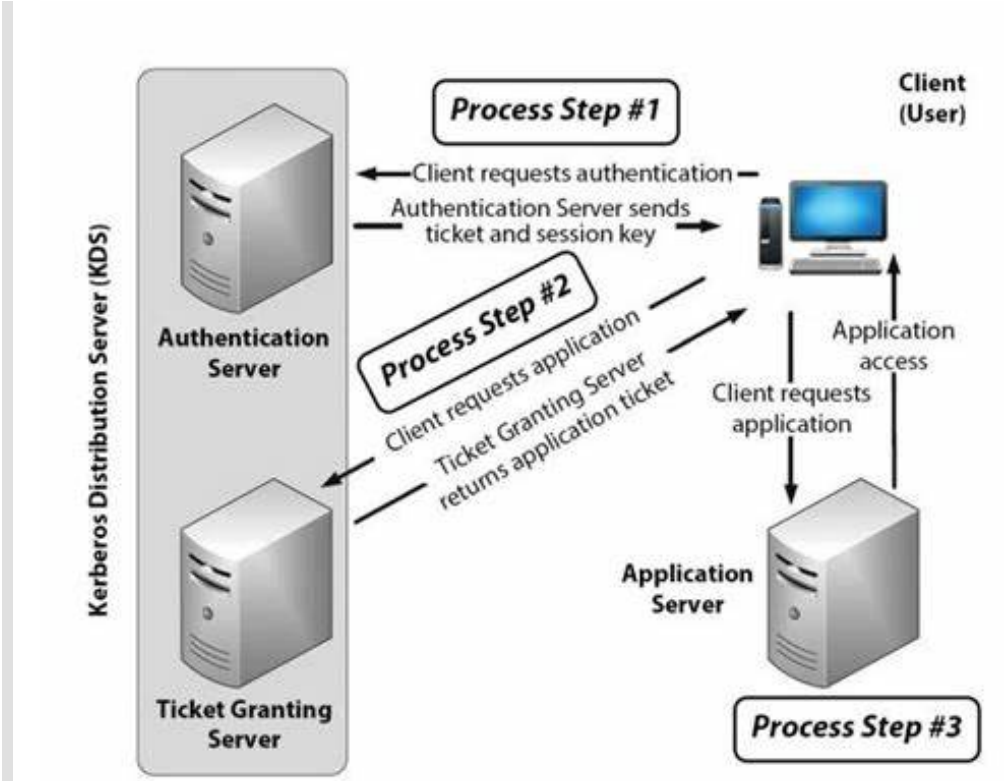


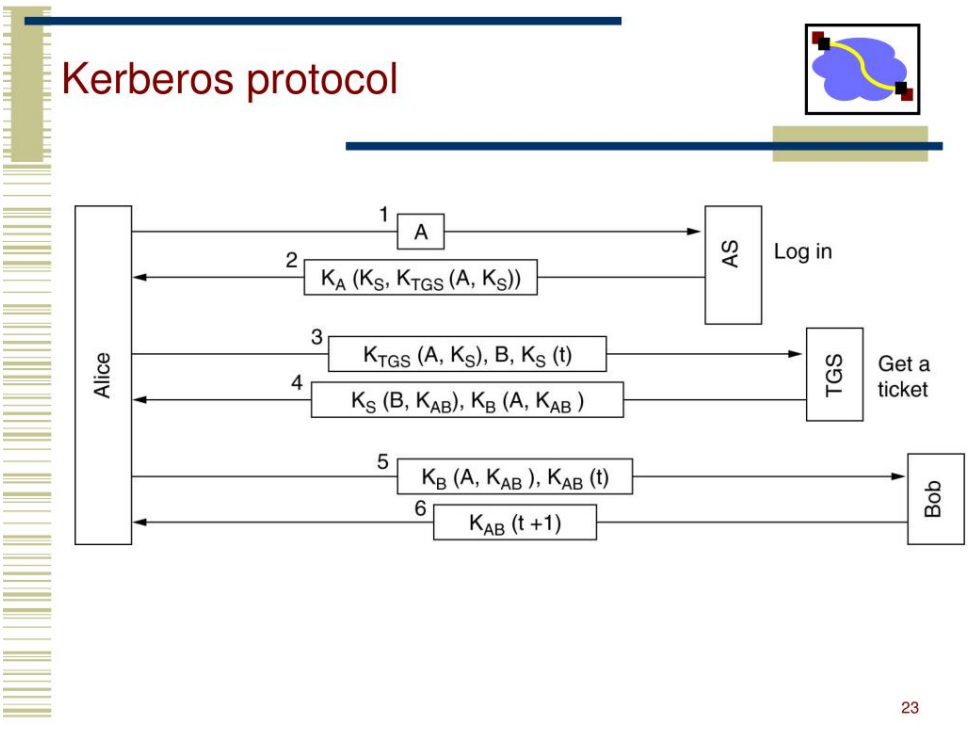


Kerberos

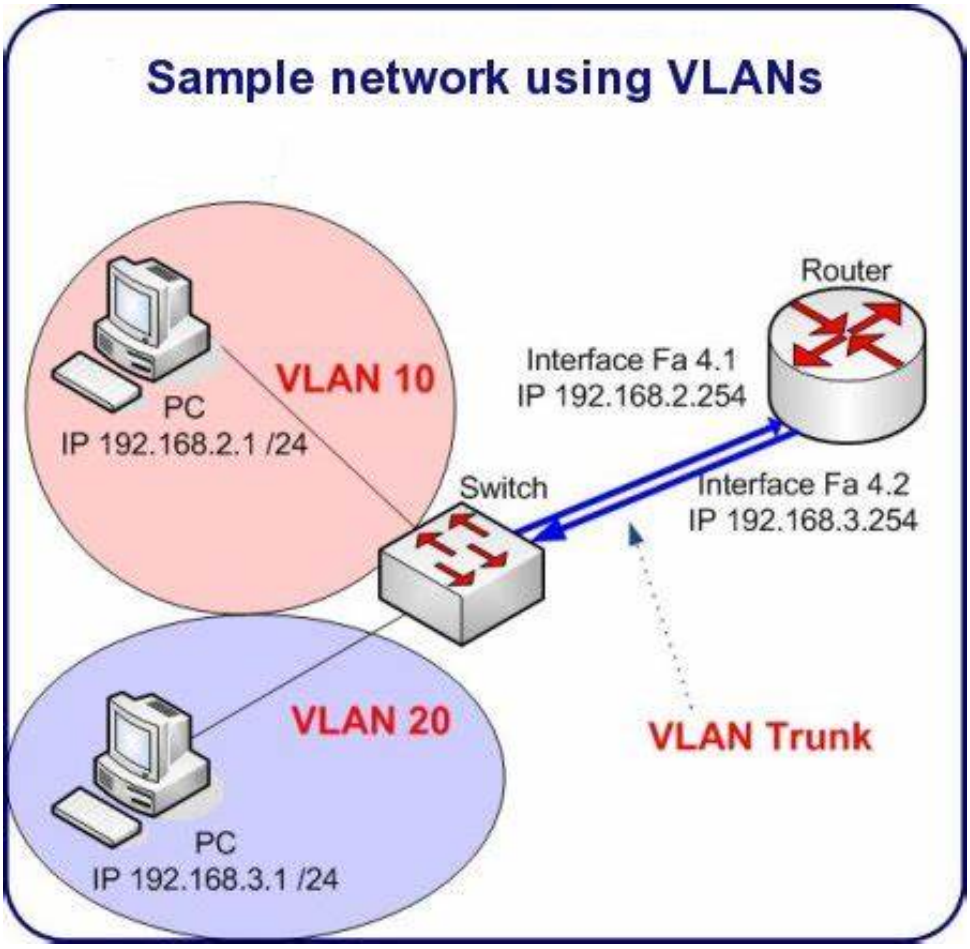

Authentication Service (AS)
1. AS REQ – Request Ticket Granting Ticket (TGT)
2. AS REP – Receive Ticket Granting Ticket
Ticket Granting Service (TGS)
1. TGS REQ – Send TGT, request Ticket Granting Service (TGS) ticket
2. TGS REP – Receive Ticket Granting Service ticket
Client
Key Distribution Center (KDC)
Domain Controller
Authentication Protocol (AP)
1. AP REQ – Send TGS to request access
2. AP REP – Access decision made by member server
PAC Validation
Optional, depends on member server configuration
Member Server


1 Client request TGT From KDC
DC / KDC
2 Authentication service sends Encrypted TGT and session key
3 Client request server access from TGS
Privileged User
4 TGS sends encrypted session key and ticket
5 Client sends service ticket
Kerberos-enabled Service
6 Server sends encrypted time stamp from client validation

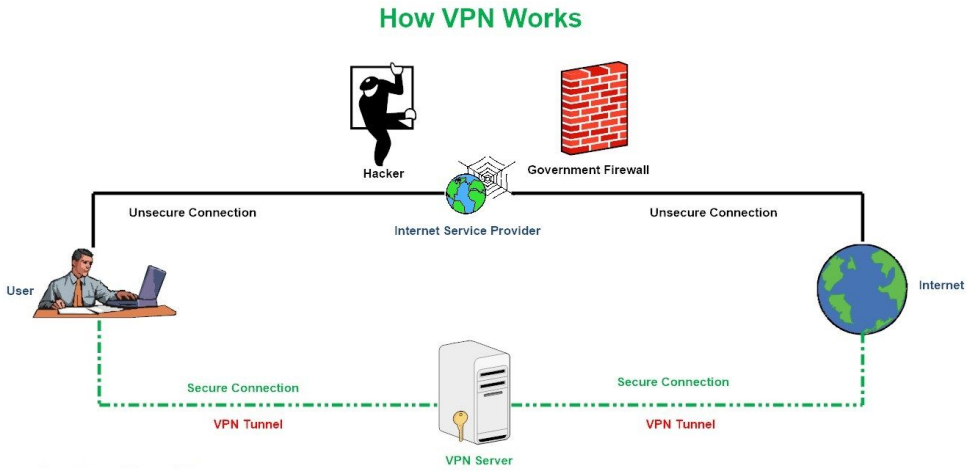

Kerberos Distribution Server (KDS)
Authentication Server
Ticket Granting Server
Process Step #1
Client (User)
Client requests authentication
Authentication Server sends ticket and session key
Process Step #2
Client requests application
Ticket Granting Server returns application ticket
Application access
Client requests application
Application Server
Process Step #3

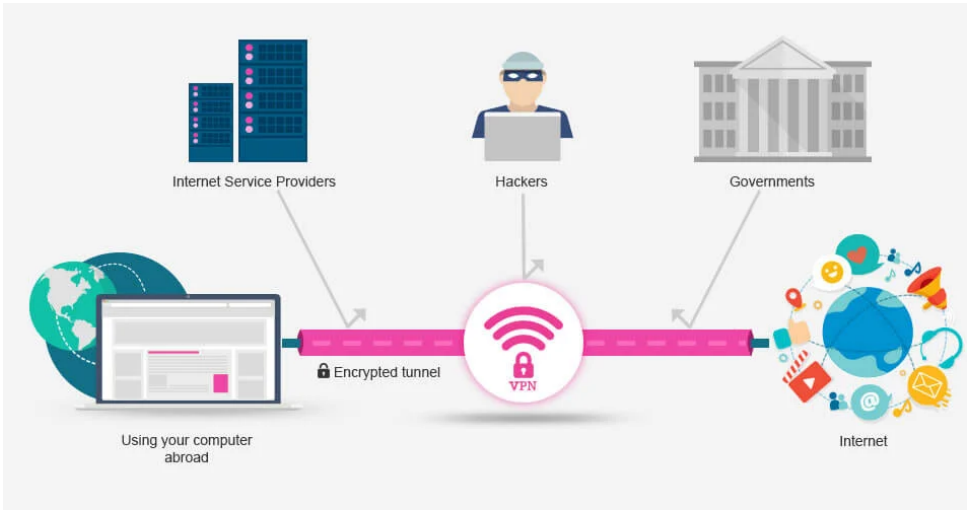

Kerberos protocol
Alice
AS
TGS
Bob
1 A
2 K_A (K_S, K_{TGS} (A, K_S))
Log in
3 K_{TGS} (A, K_S), B, K_S (t)
4 K_S (B, K_{AB}), K_B (A, K_{AB})
Get a ticket
5 K_B (A, K_{AB}), K_{AB} (t)
6 K_{AB} (t +1)
23

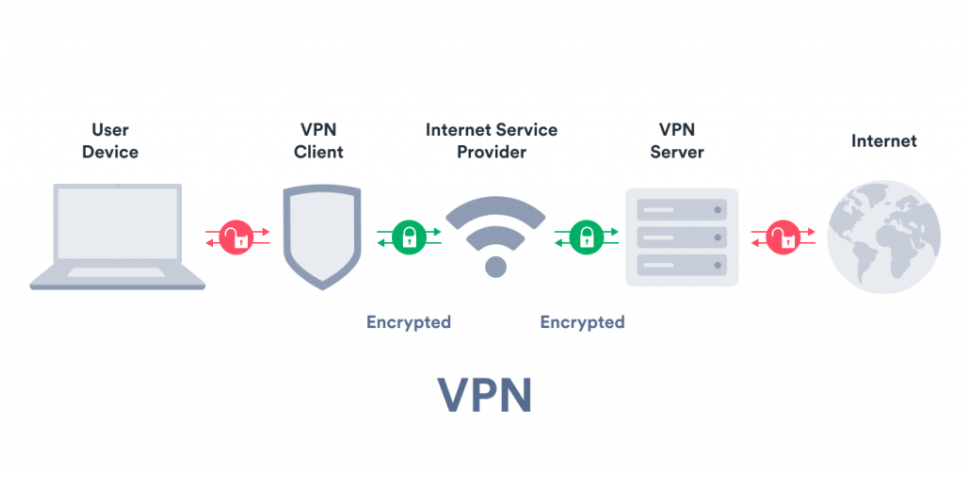
VLAN

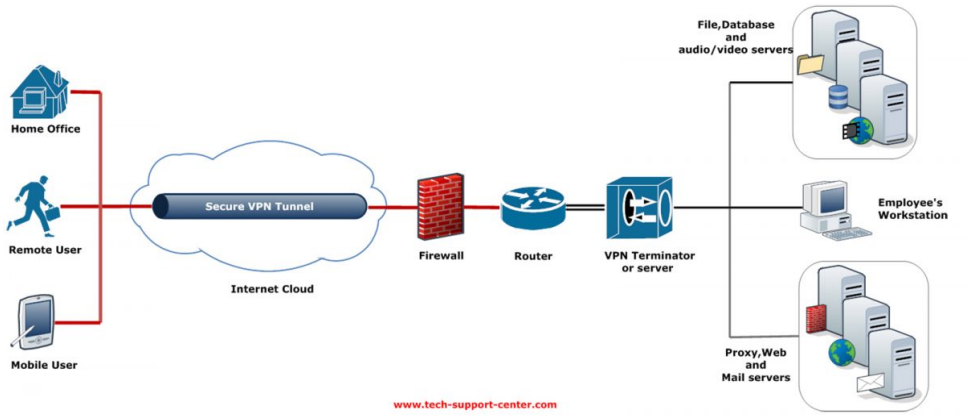

Sample network using VLANs
Router
Interface Fa 4.1
IP 192.168.2.254
VLAN 10
PC
IP 192.168.2.1 /24
Switch
Interface Fa 4.2
IP 192.168.3.254
VLAN 20
VLAN Trunk
PC
IP 192.168.3.1 /24

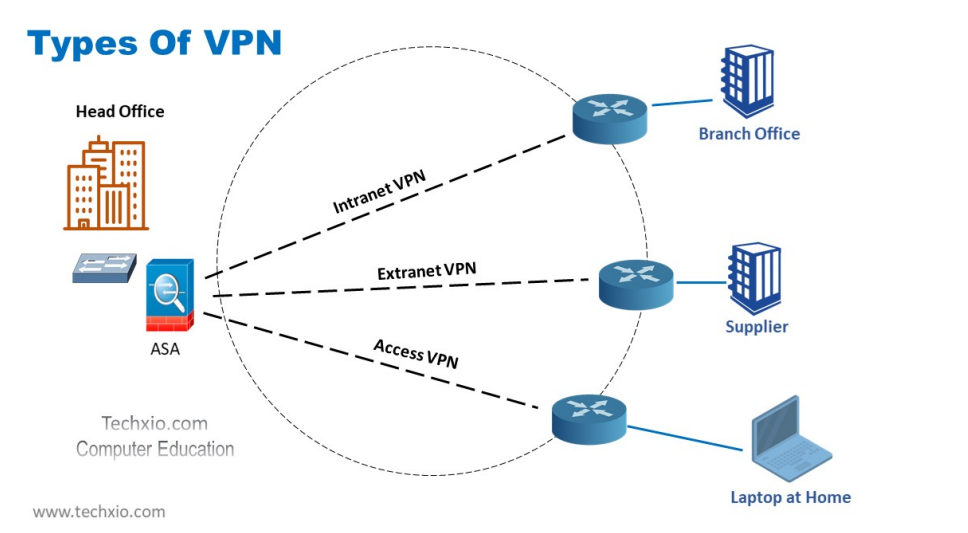
VPN


How VPN Works
Hacker
Government Firewall
Unsecure Connection
Internet Service Provider
Unsecure Connection
User
Internet
Secure Connection
VPN Tunnel
Secure Connection
VPN Tunnel
VPN Server

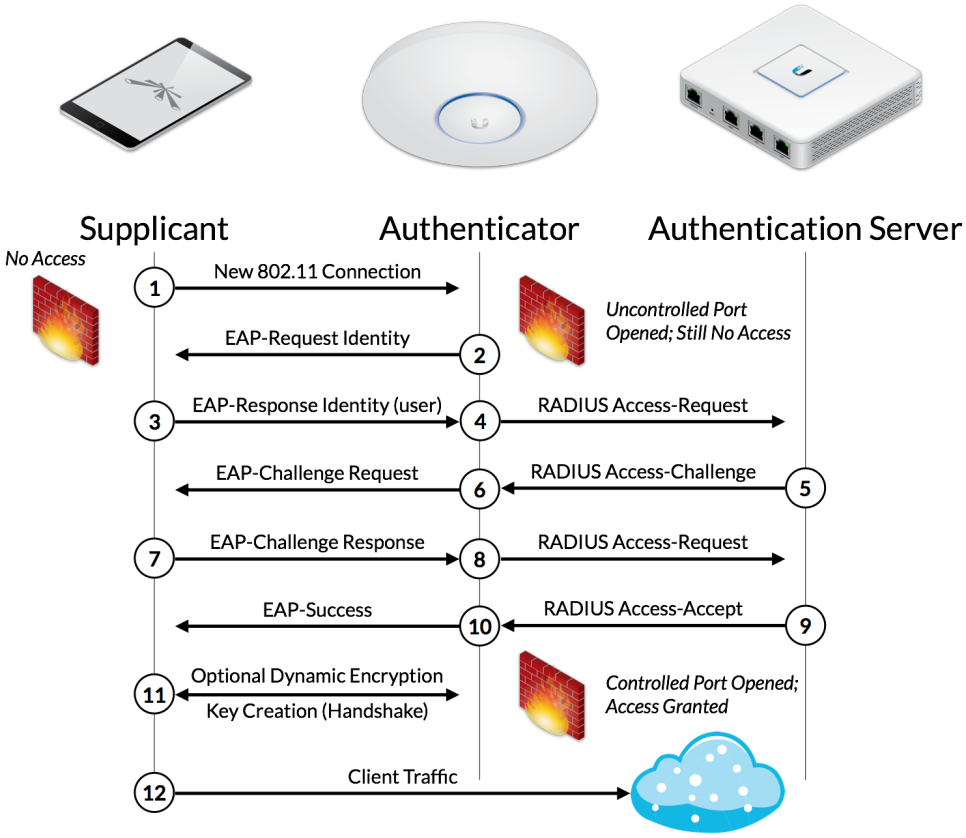

Internet Service Providers
Hackers
Governments
Encrypted tunnel
VPN
Using your computer abroad
Internet


User Device
VPN Client
Internet Service Provider
VPN Server
Internet
Encrypted
Encrypted
VPN


Home Office
Remote User
Mobile User
Secure VPN Tunnel
Internet Cloud
Firewall
Router
VPN Terminator or server
File, Database and audio/video servers
Employee's Workstation
Proxy, Web and Mail servers
www.tech-support-center.com

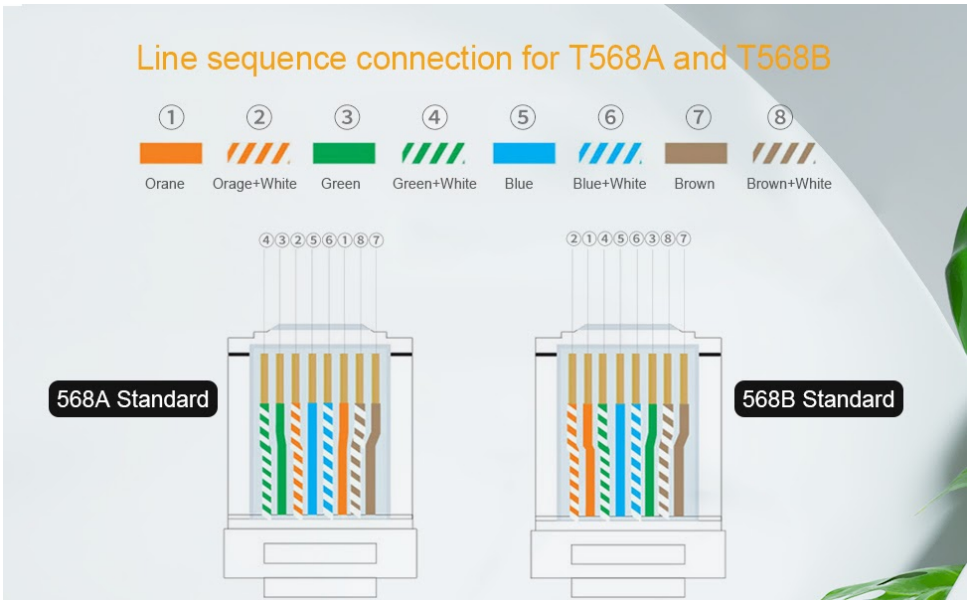

Types Of VPN
Head Office
ASA
Intranet VPN
Extranet VPN
Access VPN
Branch Office
Supplier
Laptop at Home
Techxio.com
Computer Education
www.techxio.com

802.1X Authentication (EAP & RADIUS)

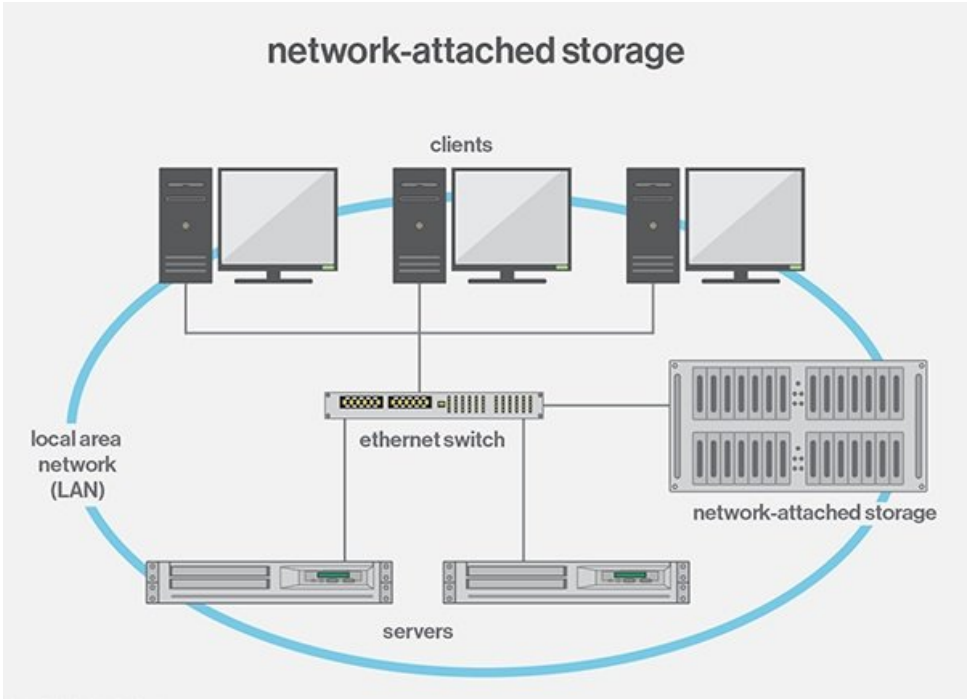


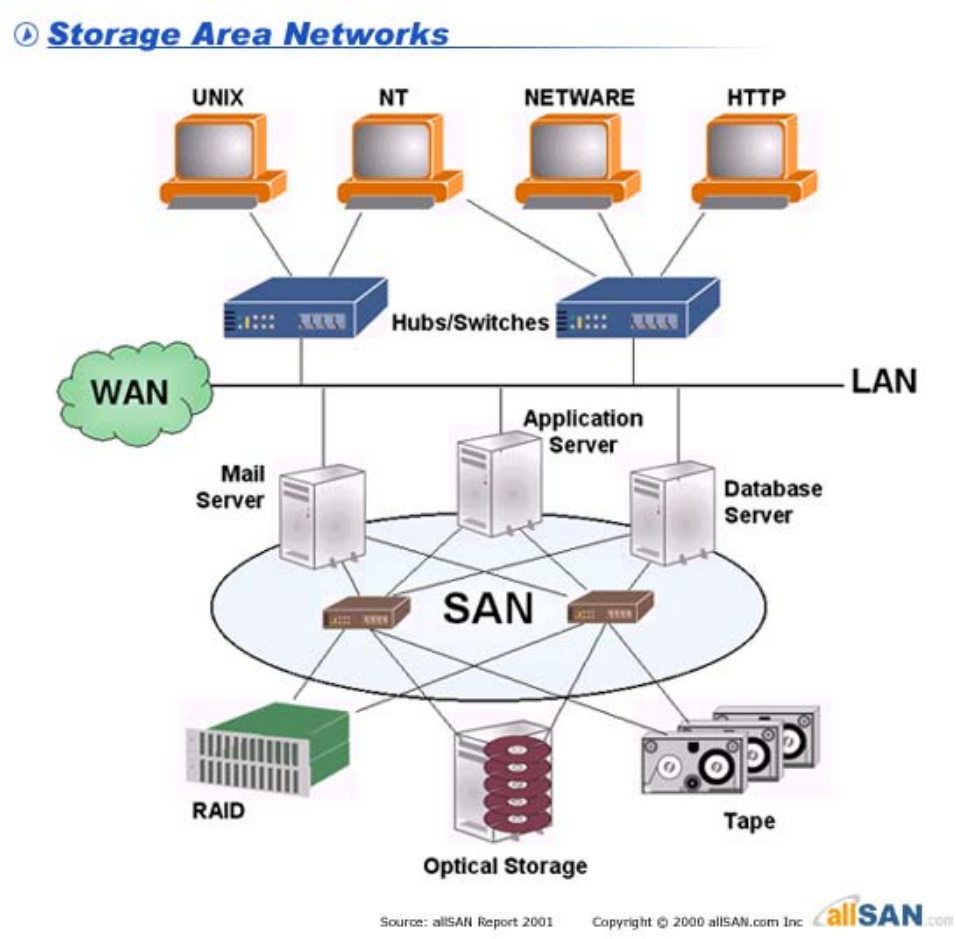
Cable

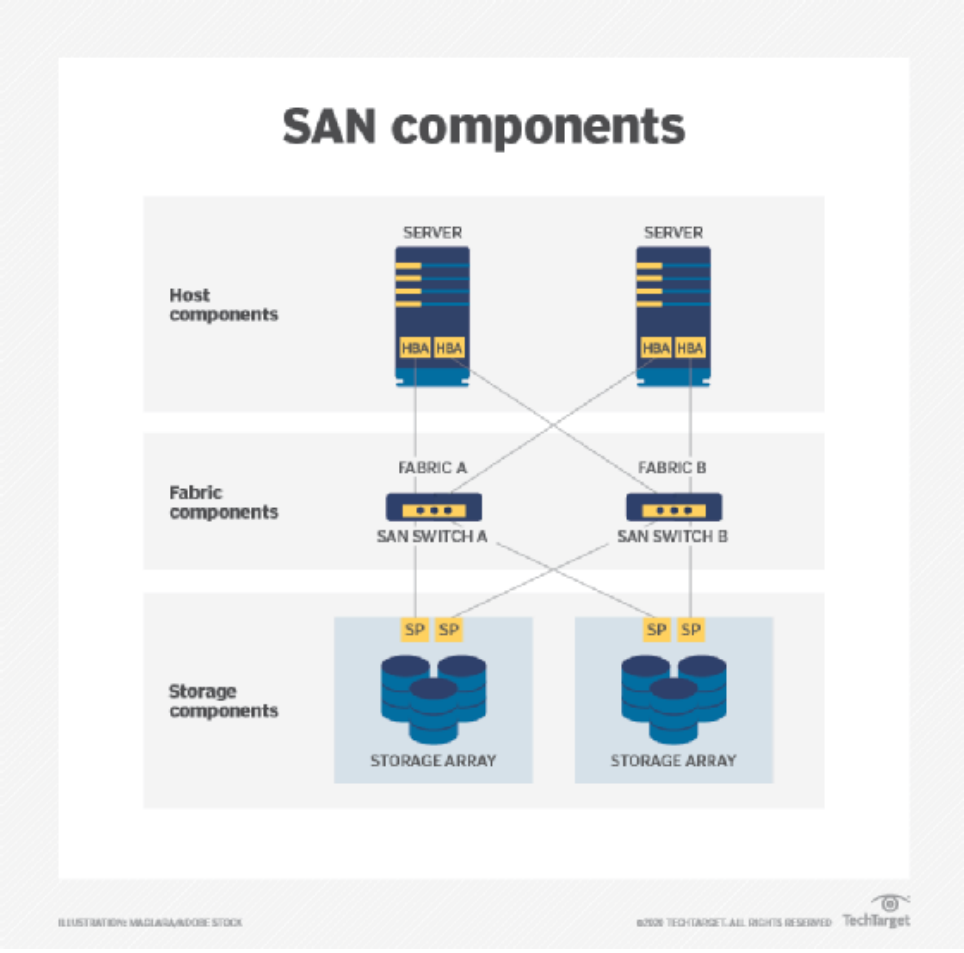
Category	Standard Bandwidth	Max Data Rate	Shielding
Cat5e	100MHz (up to 350)	1000Mbps	UTP or STP
Cat6	250MHz (up to 550)	1000Mbps	UTP or STP
Cat6A	500MHz (up to 550)	10Gbps	UTP or STP
Cat7	600MHz	10Gbps	Shielded only
Cat8	2000MHz	25Gbps or 40Gbps	Shielded only



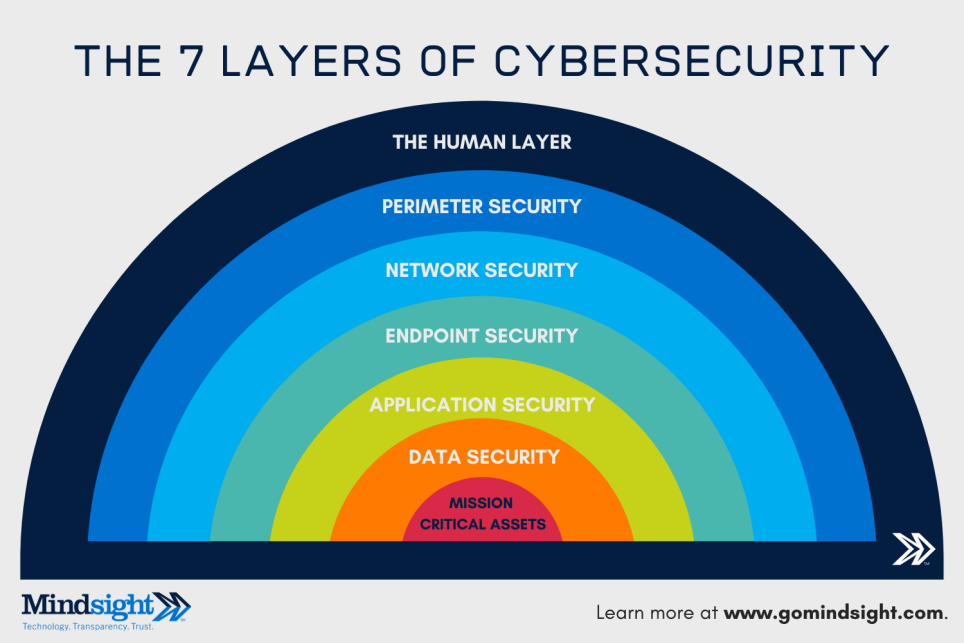
NAS vs SAN

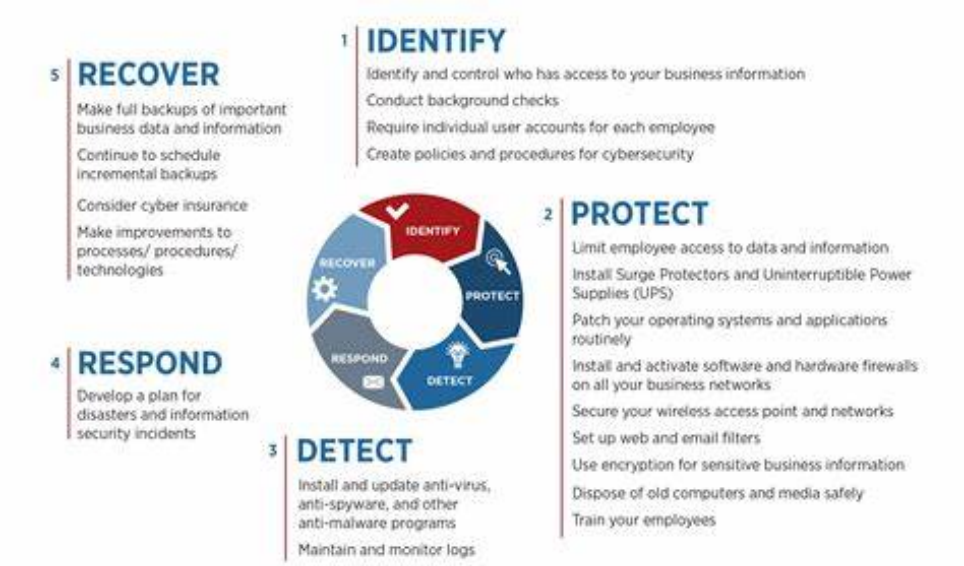


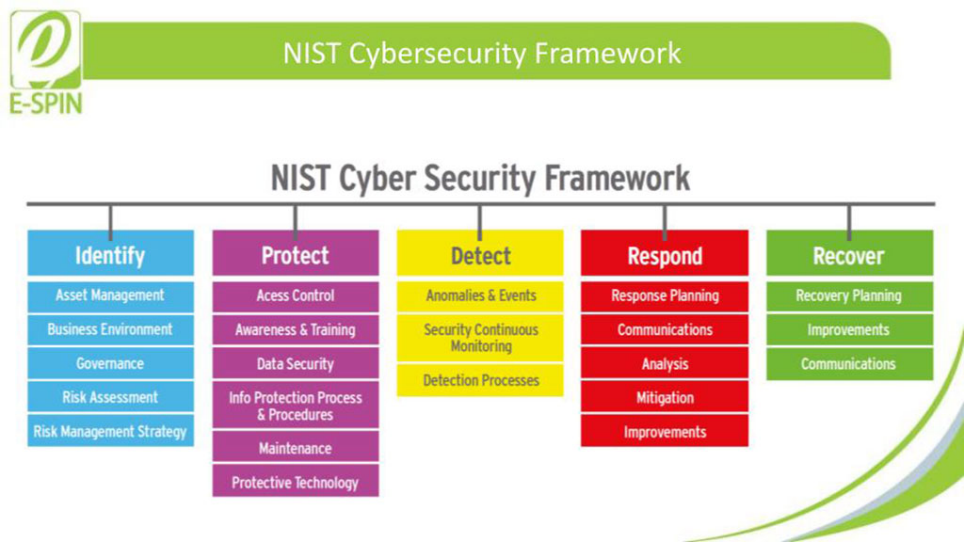

Storage Area Networks
UNIX
NT
NETWARE
HTTP
Hubs/Switches
WAN
LAN
Application Server
Mail Server
Database Server
SAN
RAID
Optical Storage
Tape
Source: allSAN Report 2001
Copyright © 2000 allSAN.com Inc
allSAN


SAN components
Host components
SERVER
SERVER
HBA HBA
HBA HBA
Fabric components
FABRIC A
FABRIC B
SAN SWITCH A
SAN SWITCH B
Storage components
SP SP
SP SP
STORAGE ARRAY
STORAGE ARRAY
TechTarget

CyberSecurity

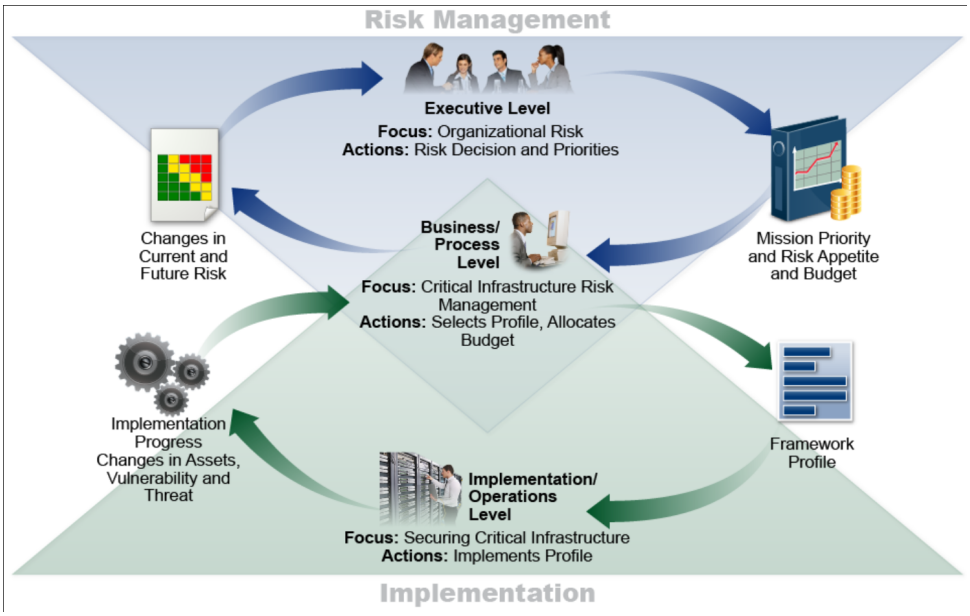




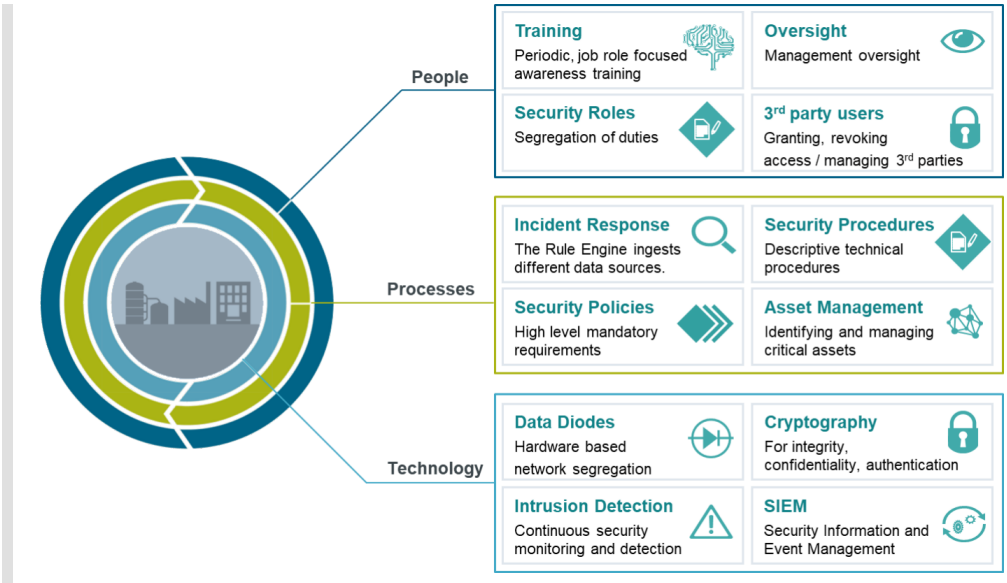


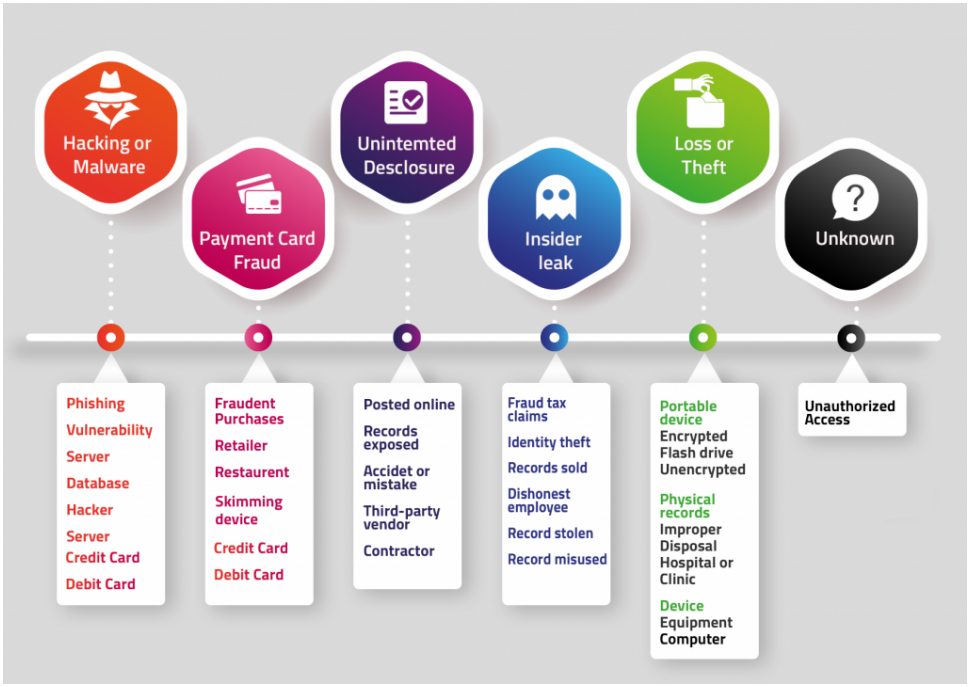
Risk Management


BEST CYBERSECURITY PRACTICES
01
Install and regularly update protective software
02
Apply a strong password policy
Use multi-factor authentication
03
Make regular backups
04
Drudesk
Drupal Support On Demand

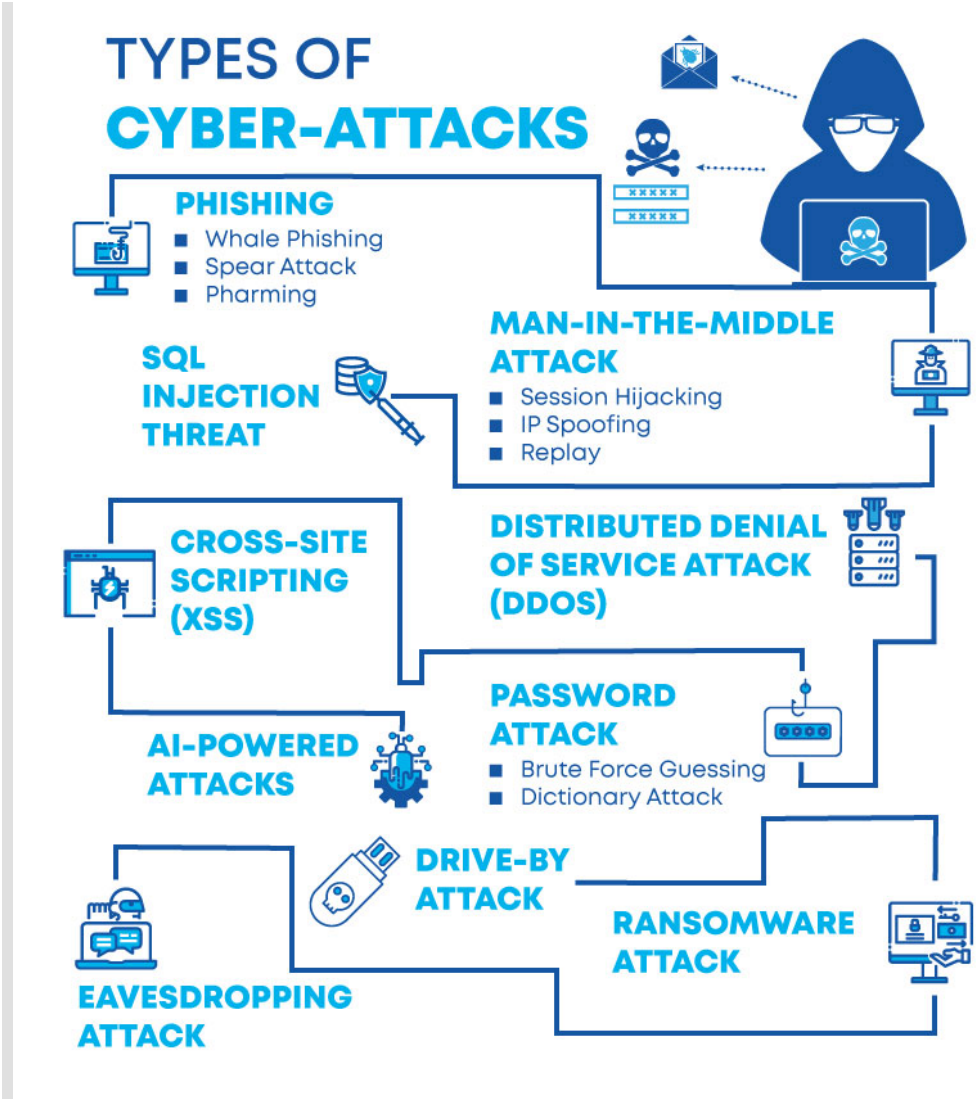

Risk Management
Executive Level
Focus: Organizational Risk
Actions: Risk Decision and Priorities
Changes in Current and Future Risk
Business/ Process Level
Focus: Critical Infrastructure Risk Management
Actions: Selects Profile, Allocates Budget
Mission Priority and Risk Appetite and Budget
Implementation Progress Changes in Assets, Vulnerability and Threat
Framework Profile
Implementation/ Operations Level
Focus: Securing Critical Infrastructure
Actions: Implements Profile
Implementation

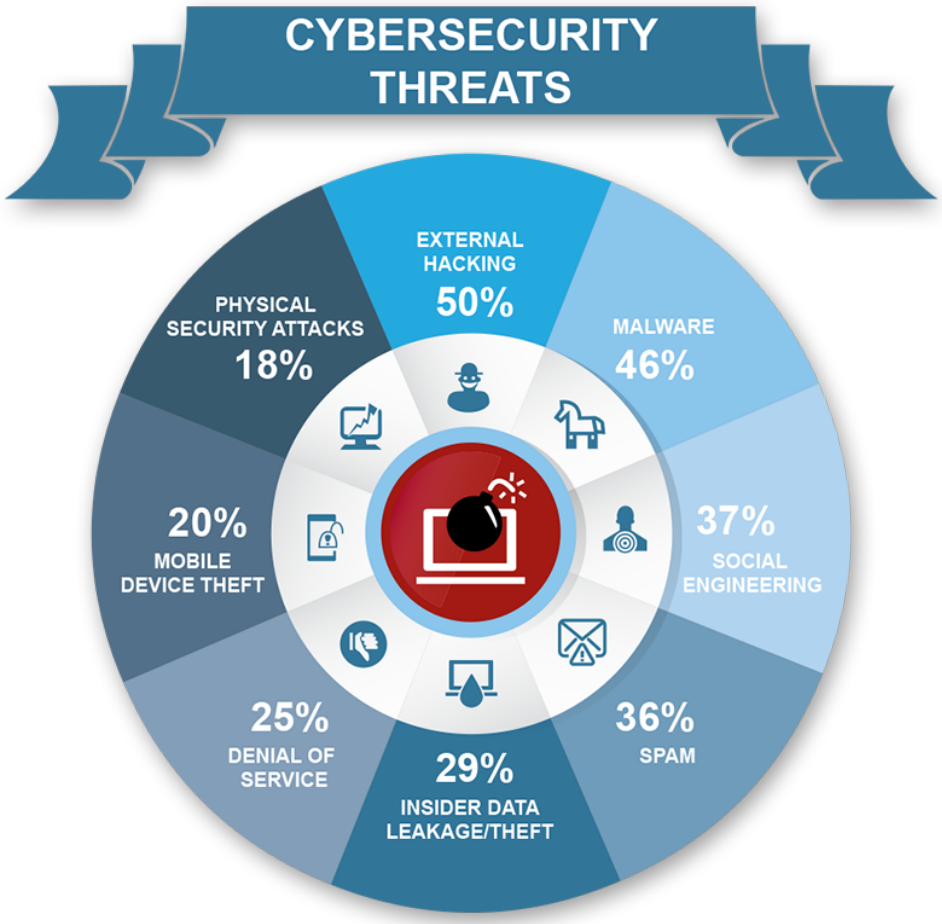

Application Security
Information Security
Network Security
Disaster Recovery
End-user Compliance
Elements of Cybersecurity

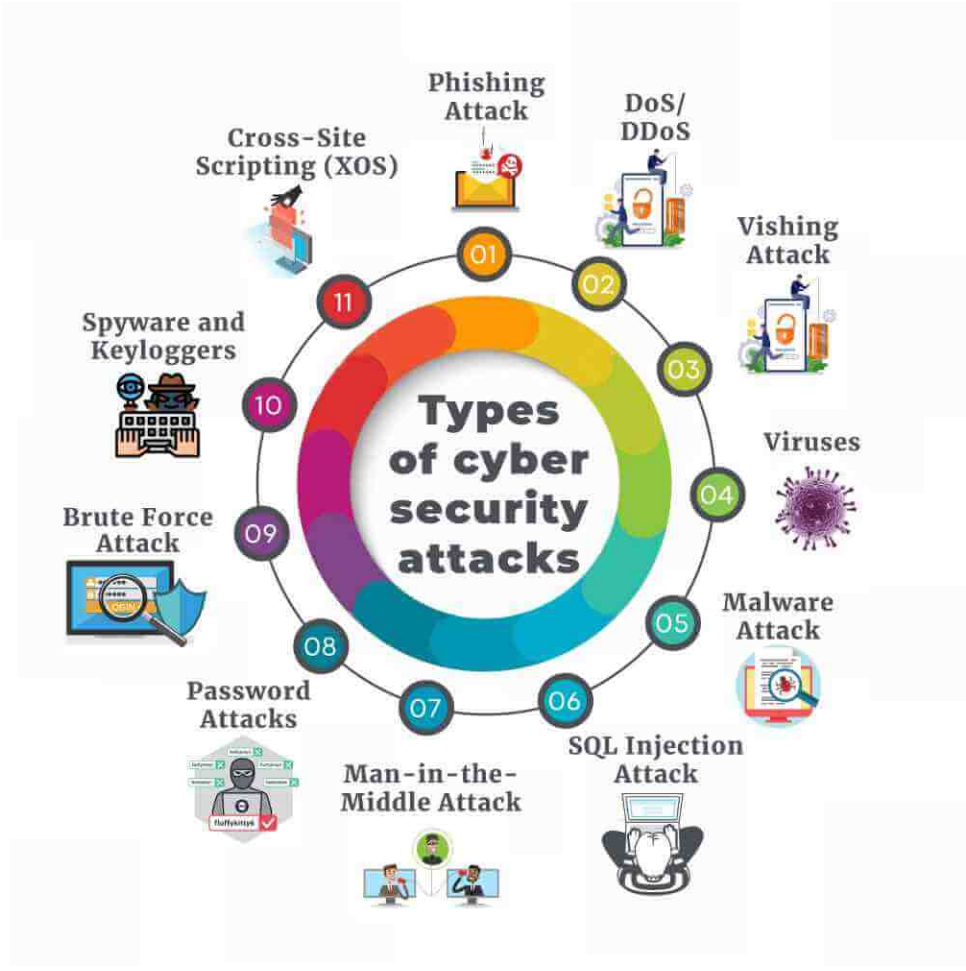

People
Processes
Technology
Training
Periodic, job role focused awareness training
Oversight
Management oversight
Security Roles
Segregation of duties
3rd party users
Granting, revoking access / managing 3rd parties
Incident Response
The Rule Engine ingests different data sources.
Security Procedures
Descriptive technical procedures
Security Policies
High level mandatory requirements
Asset Management
Identifying and managing critical assets
Data Diodes
Hardware based network segregation
Cryptography
For integrity, confidentiality, authentication
Intrusion Detection
Continuous security monitoring and detection
SIEM
Security Information and Event Management



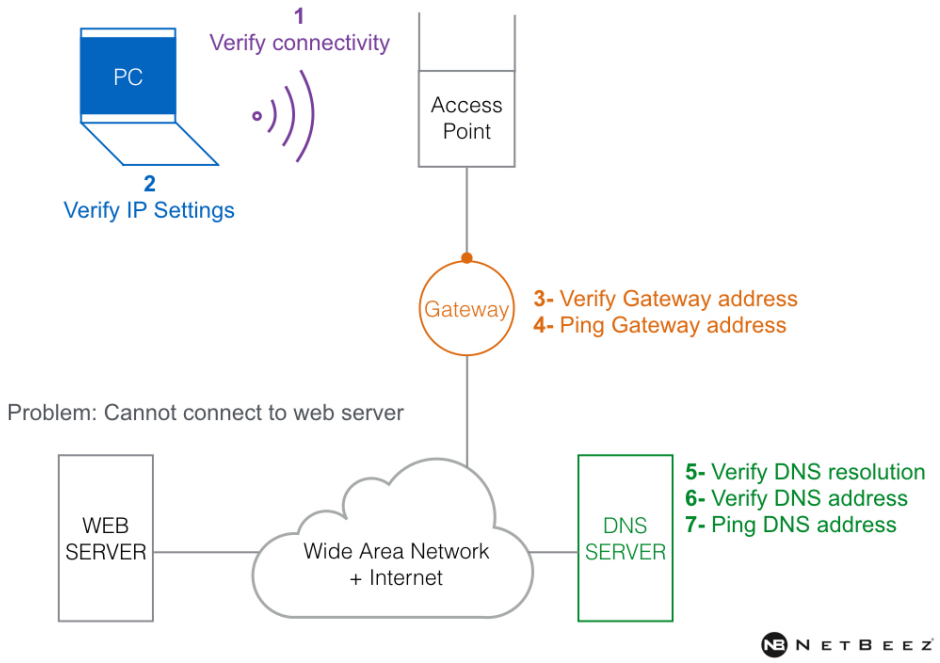
Cybersecurity Attacks

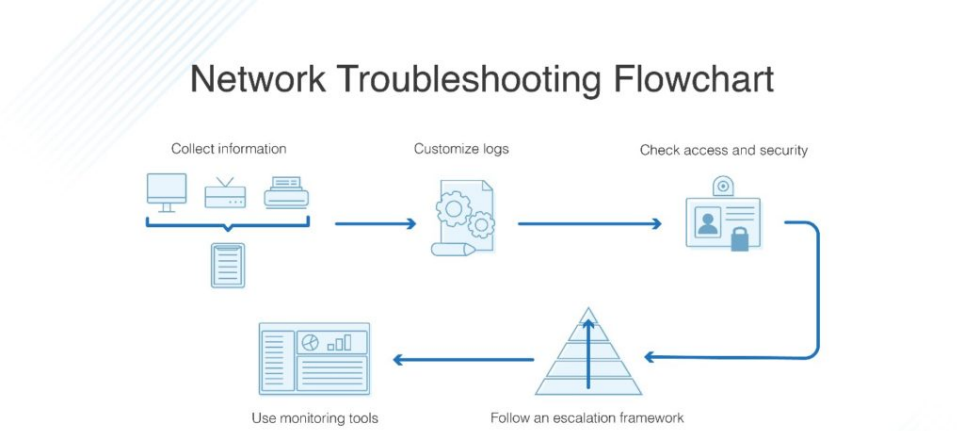


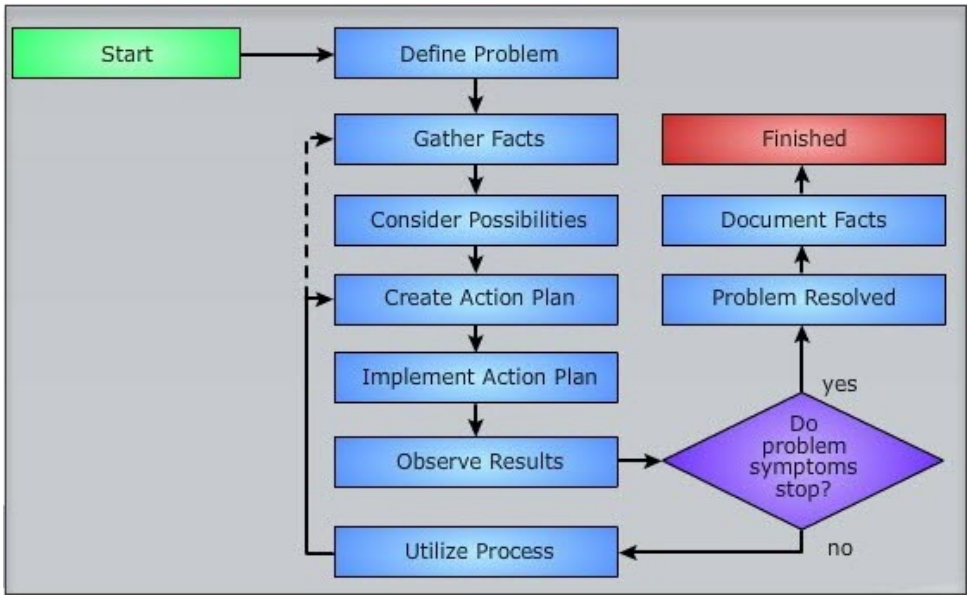




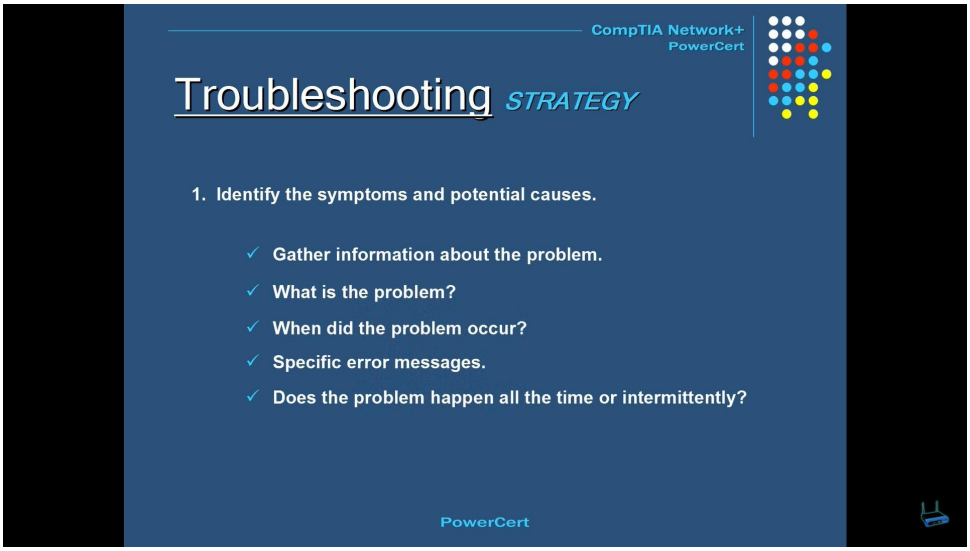
Network Troubleshooting

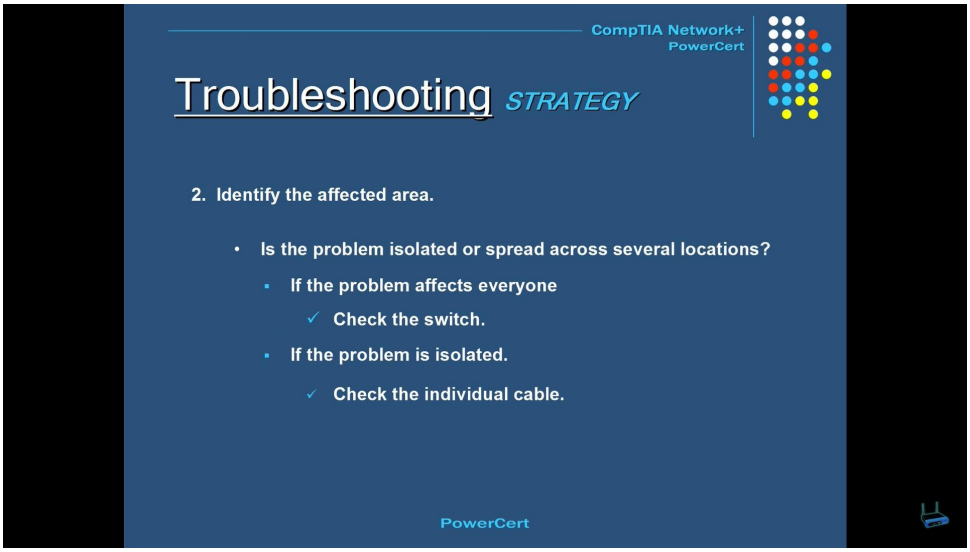


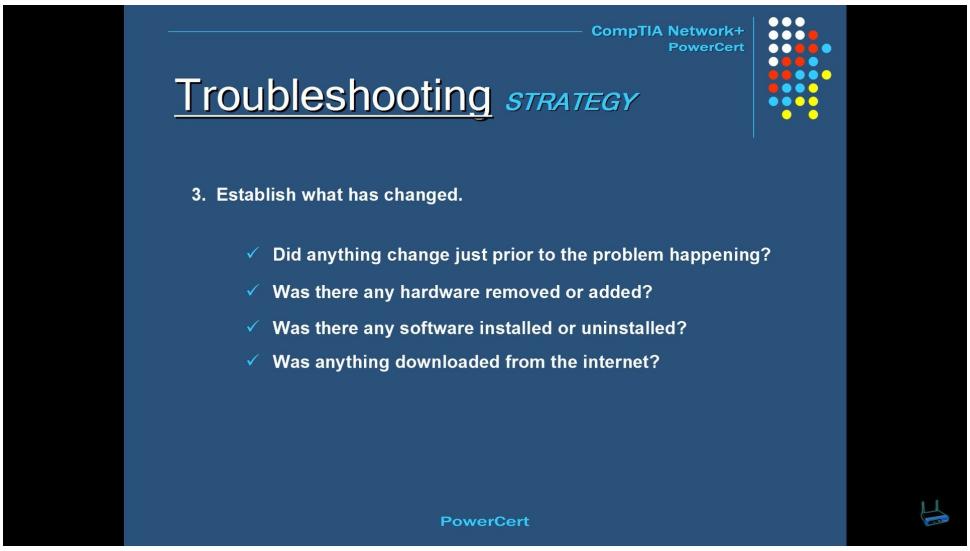


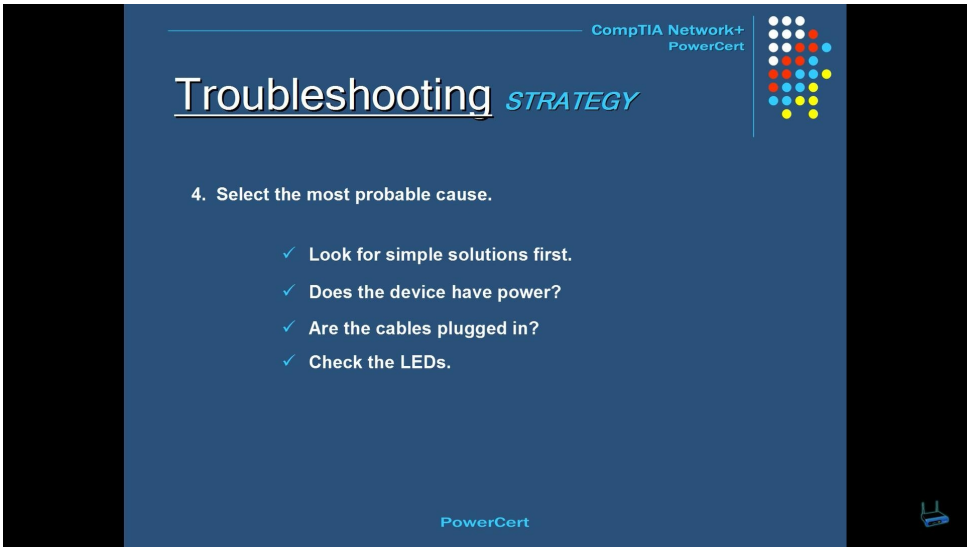


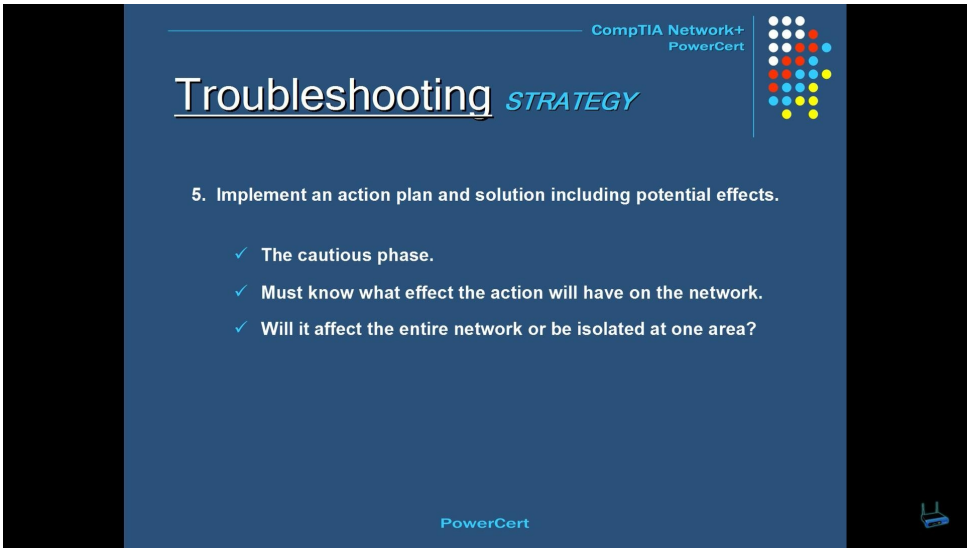
Troubleshooting Strategy

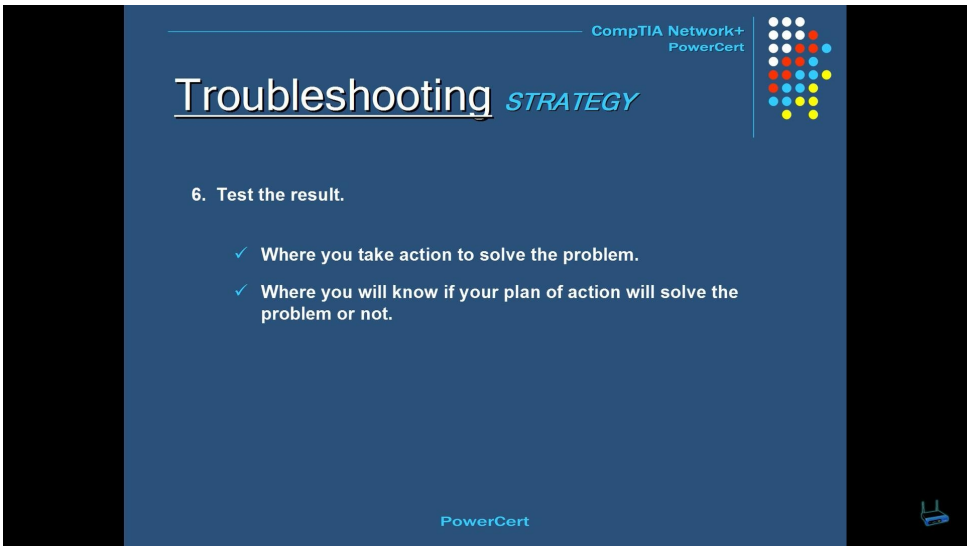

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
1. Identify the symptoms and potential causes.
✓ Gather information about the problem.
✓ What is the problem?
✓ When did the problem occur?
✓ Specific error messages.
✓ Does the problem happen all the time or intermittently?
PowerCert

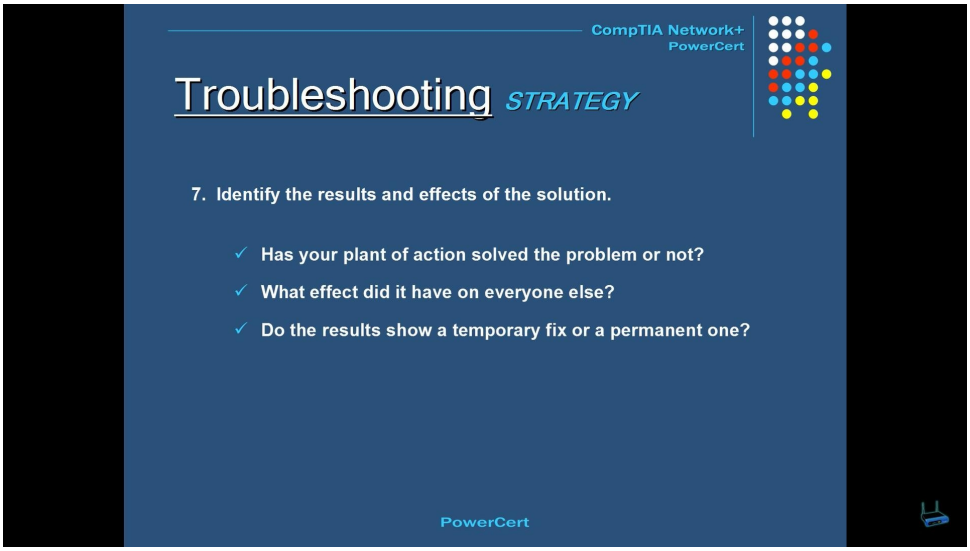

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
2. Identify the affected area.
• Is the problem isolated or spread across several locations?
▪ If the problem affects everyone
✓ Check the switch.
▪ If the problem is isolated.
✓ Check the individual cable.
PowerCert

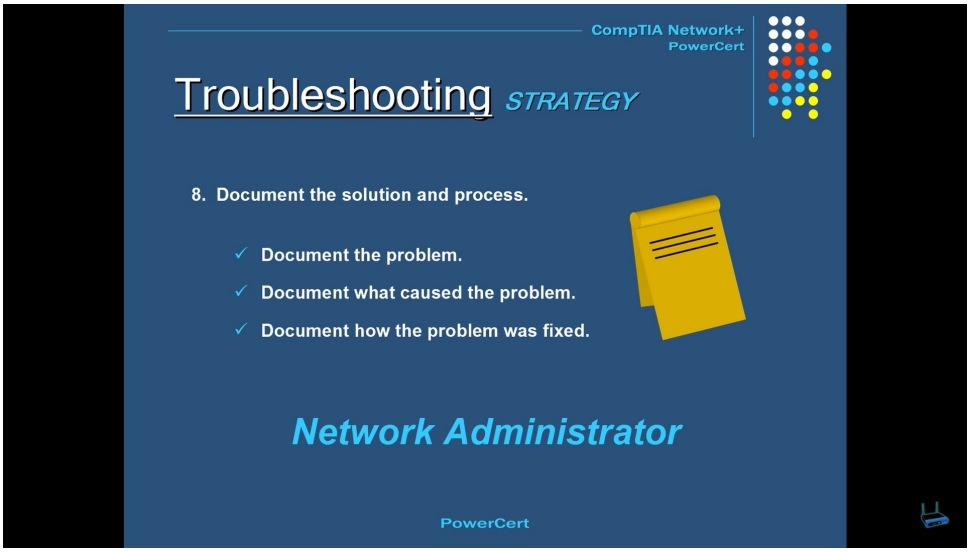

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
3. Establish what has changed.
✓ Did anything change just prior to the problem happening?
✓ Was there any hardware removed or added?
✓ Was there any software installed or uninstalled?
✓ Was anything downloaded from the internet?
PowerCert


CompTIA Network+
PowerCert
Troubleshooting STRATEGY
4. Select the most probable cause.
✓ Look for simple solutions first.
✓ Does the device have power?
✓ Are the cables plugged in?
✓ Check the LEDs.
PowerCert

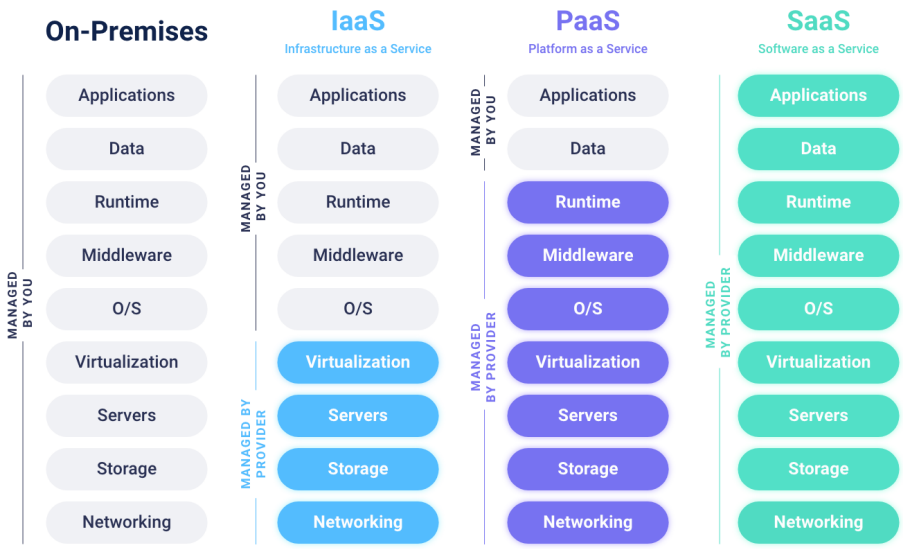

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
5. Implement an action plan and solution including potential effects.
✓ The cautious phase.
✓ Must know what effect the action will have on the network.
✓ Will it affect the entire network or be isolated at one area?
PowerCert

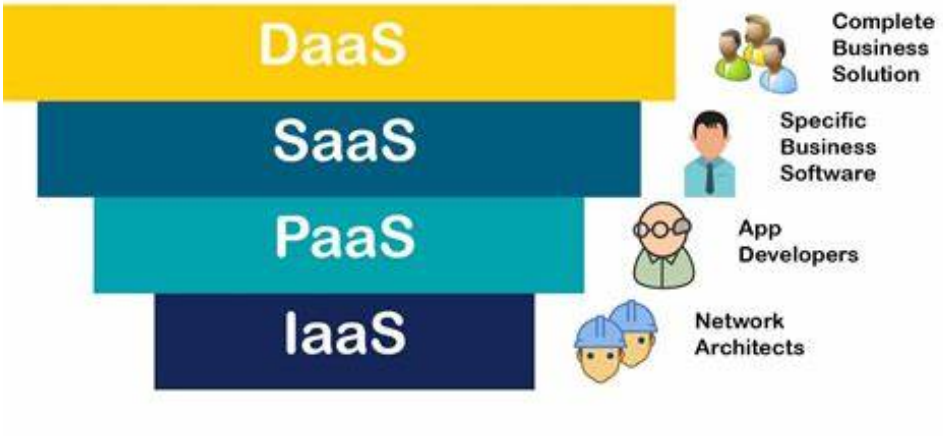

CompTIA Network+
PowerCert
Troubleshooting STRATEGY
6. Test the result.
✓ Where you take action to solve the problem.
✓ Where you will know if your plan of action will solve the problem or not.
PowerCert

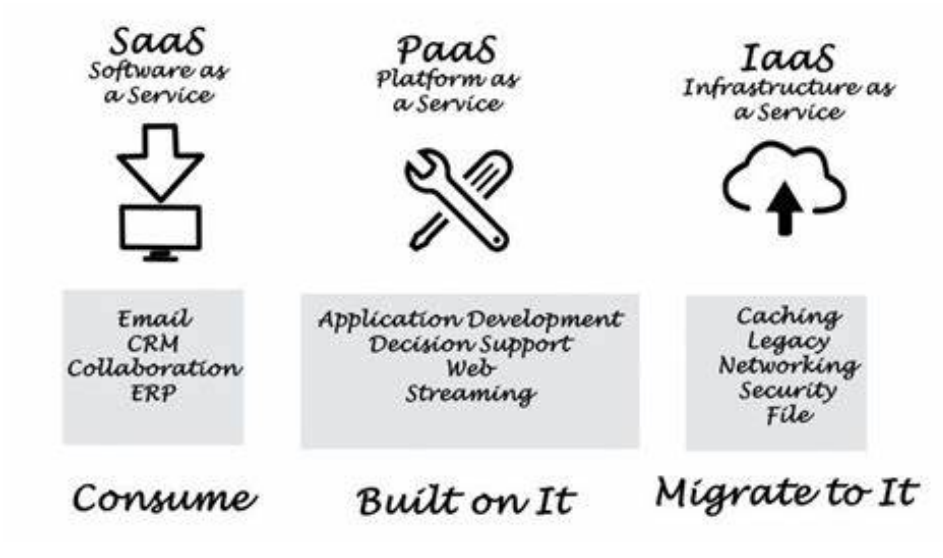


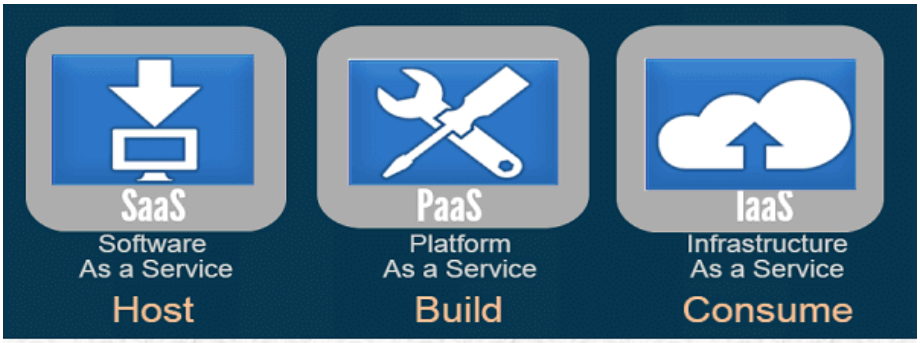


Could Computing - Iaas Paas Saas

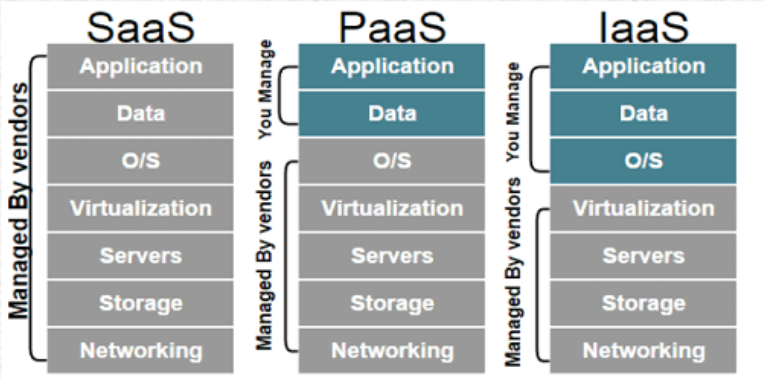



DaaS
SaaS
PaaS
IaaS
Complete Business Solution
Specific Business Software
App Developers
Network Architects

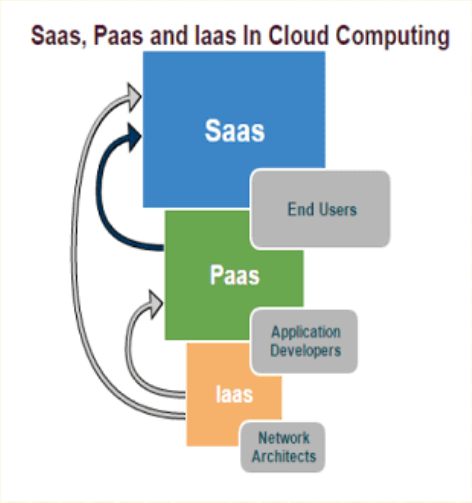

SaaS
Software as a Service
PaaS
Platform as a Service
IaaS
Infrastructure as a Service
Email
CRM
Collaboration
ERP
Application Development
Decision Support
Web
Streaming
Caching
Legacy
Networking
Security
File
Consume
Built on It
Migrate to It

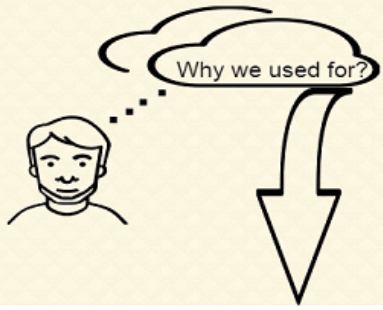


Difference between SaaS, PaaS and IaaS



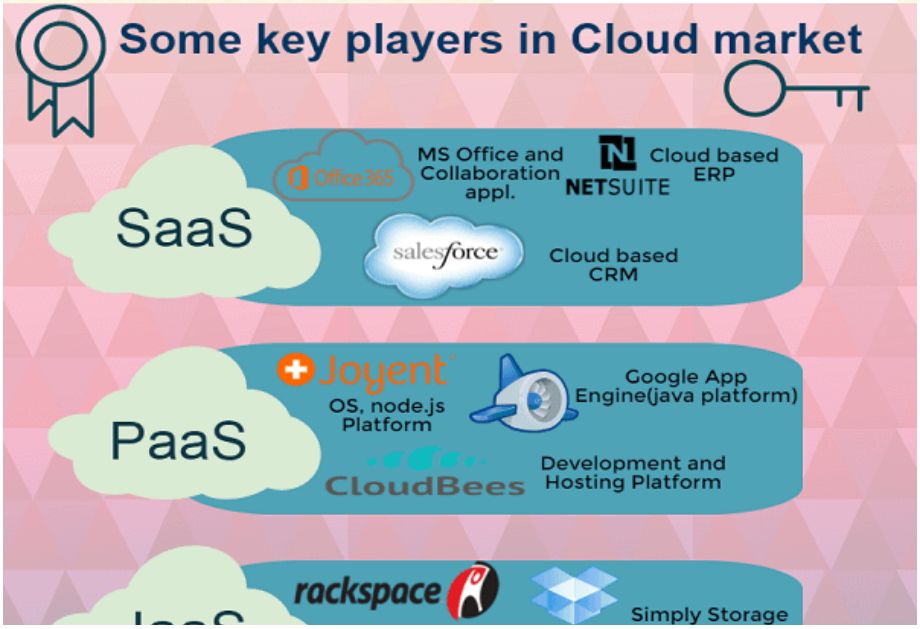
How Structured in Cloud Computing?





Cloud	Enterprise
Saas	Application (Oracle BS, SAP)
Paas	Middleware (DB, App server, ESB)
Iaas	Virtualization/ Hardware/ OS

Some key players in Cloud market




IaaS
Cloud Servers
Dropbox
Akamai
Content Delivery
Network

-- Memo End --