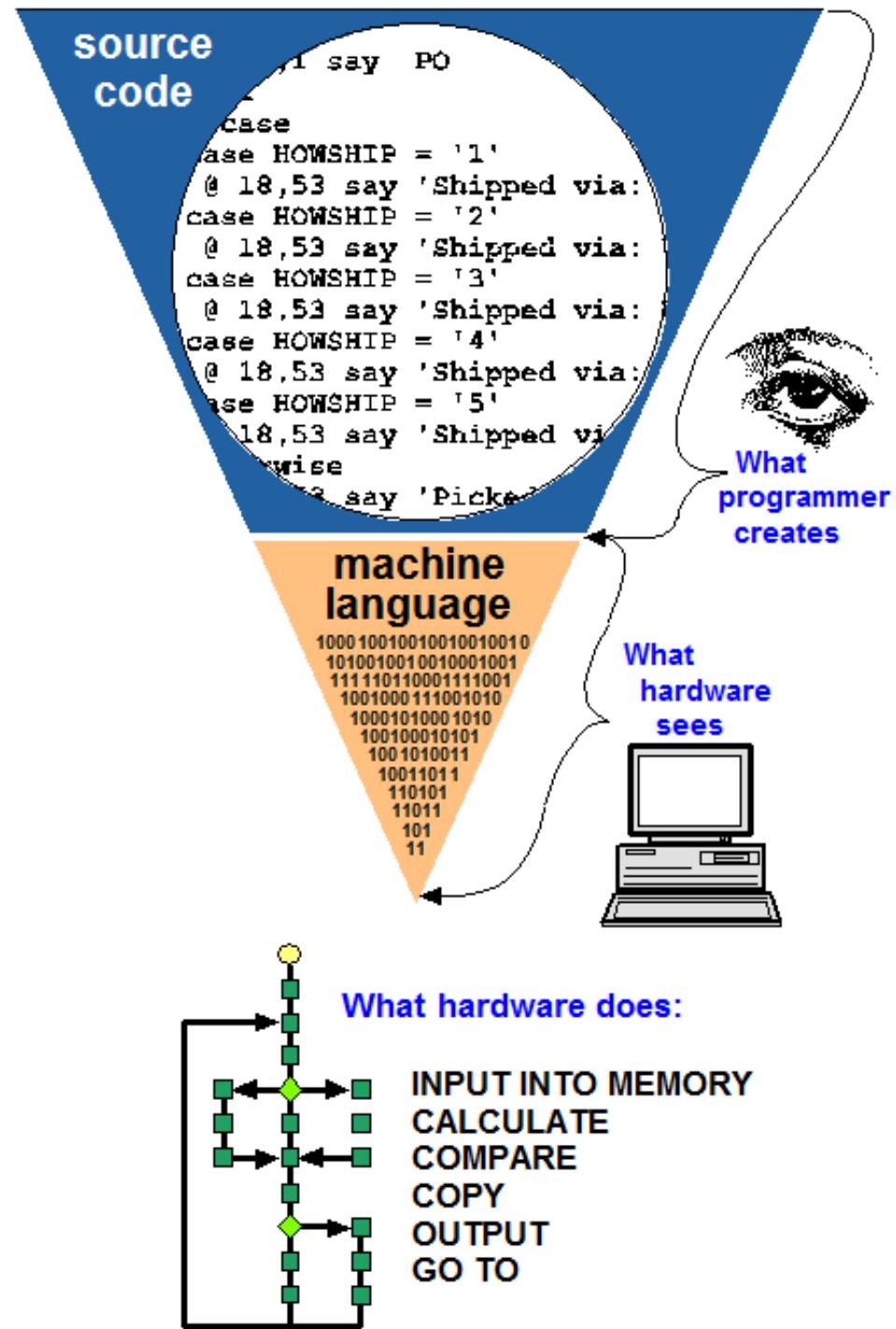


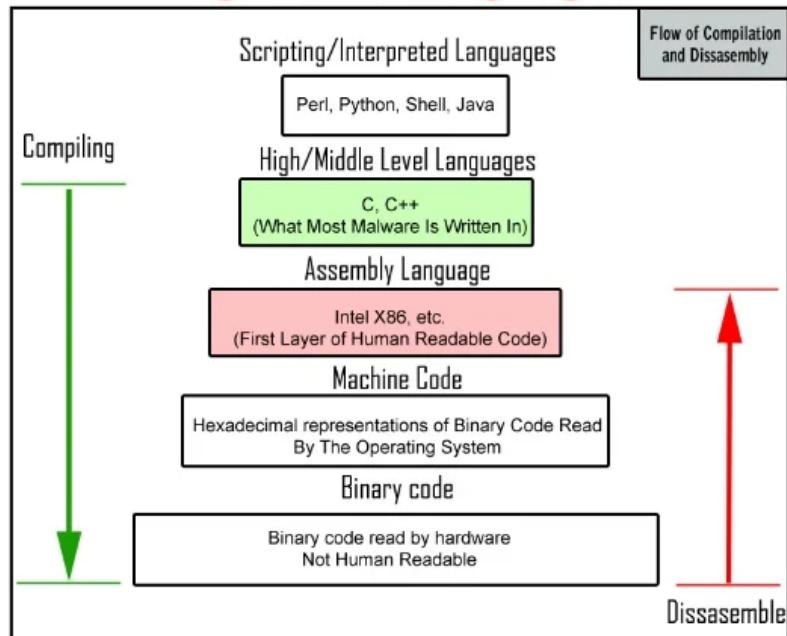
Computer Science

Programming Language -> Communicate with Computer

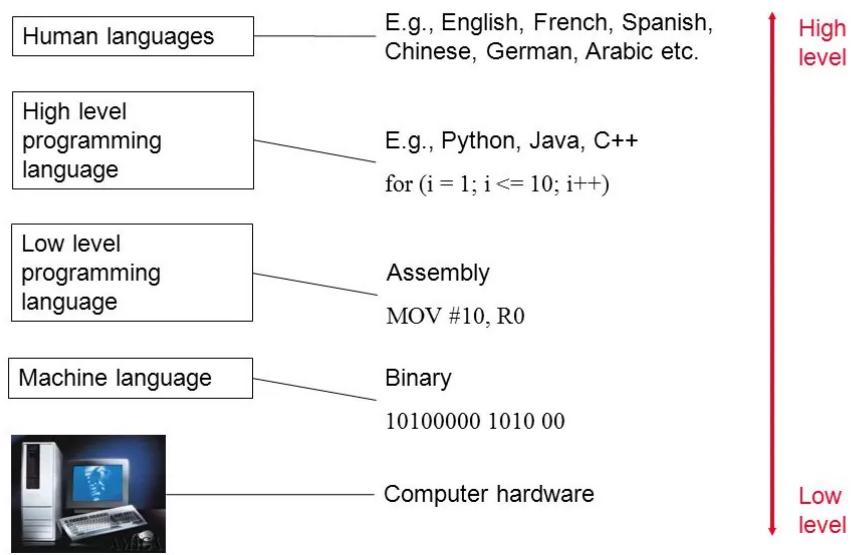
SOURCE CODE TO MACHINE LANGUAGE



High Level Languages

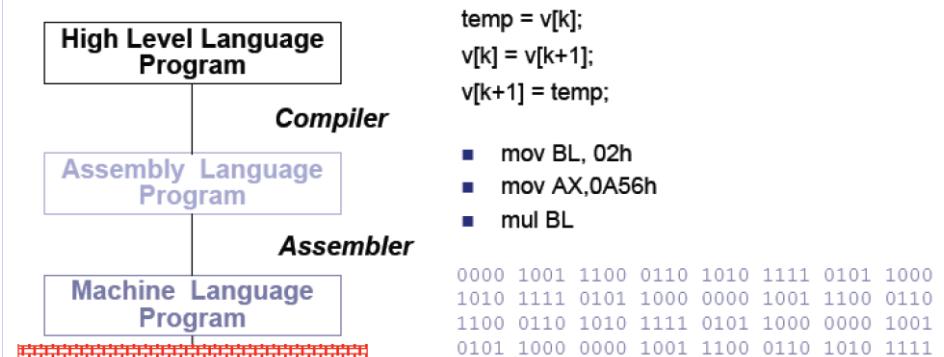


High Vs. Low Level Languages



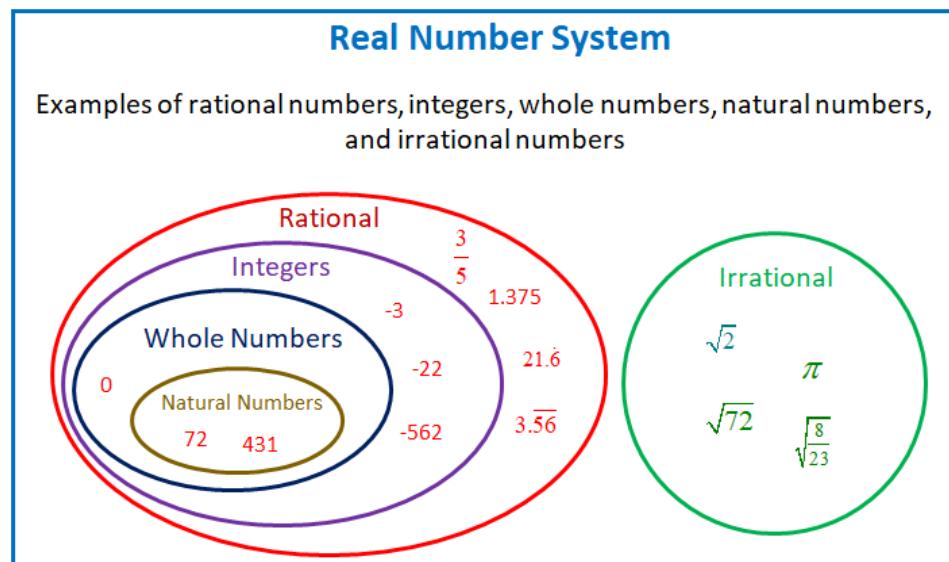
James Tam

Levels of Representation



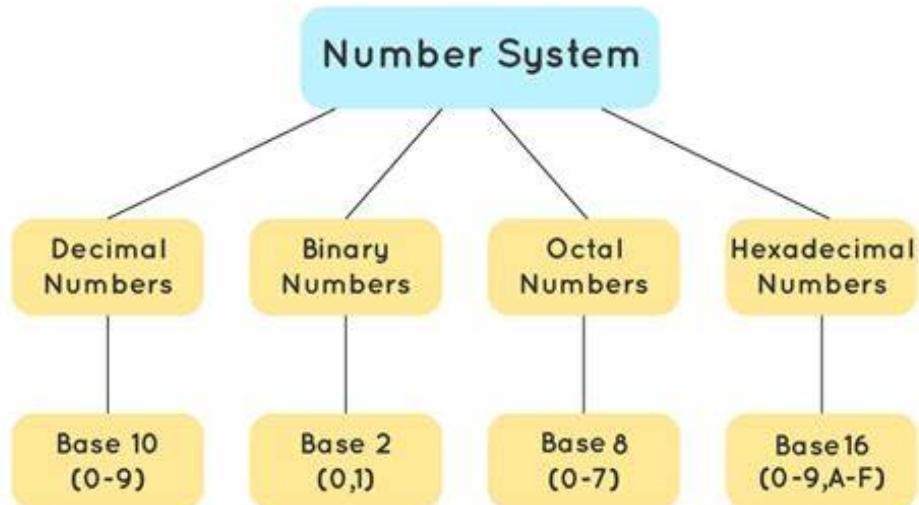
Number System

Real Number System



Number System Conversion

Types of Number System



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

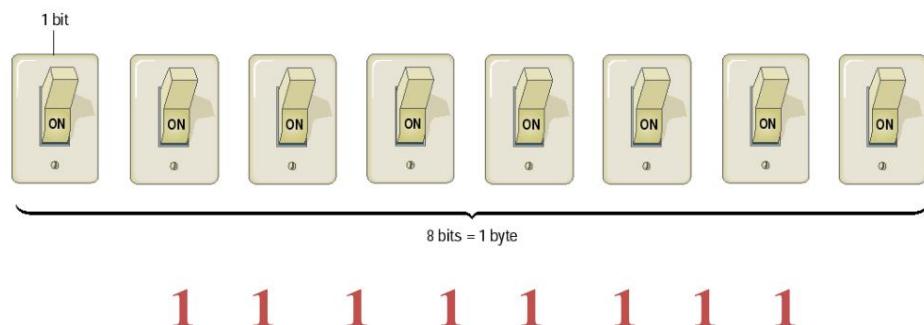
- Convert from decimal to binary
Divide the decimal by the largest binary weight it is divisible by and place a "1" in that column. Then select the next largest weight, if it is divisible put a "1" in that column otherwise place a "0" in the column. Continue until all the columns have either a "1" or "0" resulting in a binary expression.
- 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.
- 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

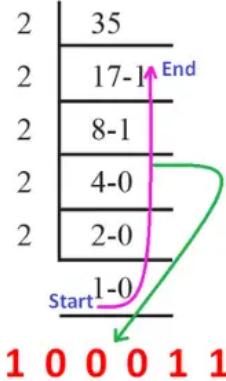
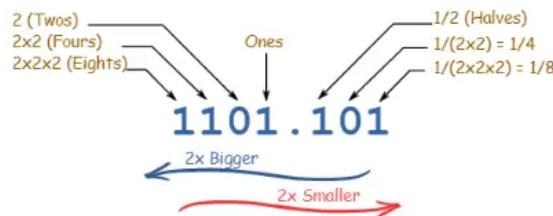
Number System

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.



What is the Binary Number System?



Electrical 4 U

Complement

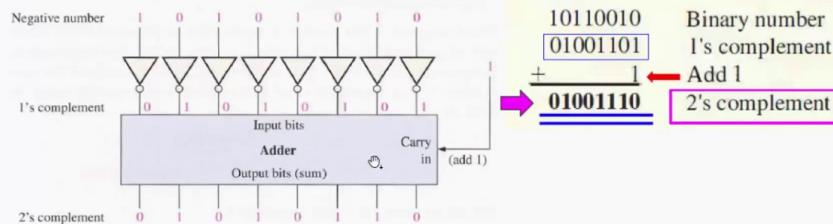
2.5) 1'S and 2'S Complements of Binary Numbers

Finding the 2's Complement

The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

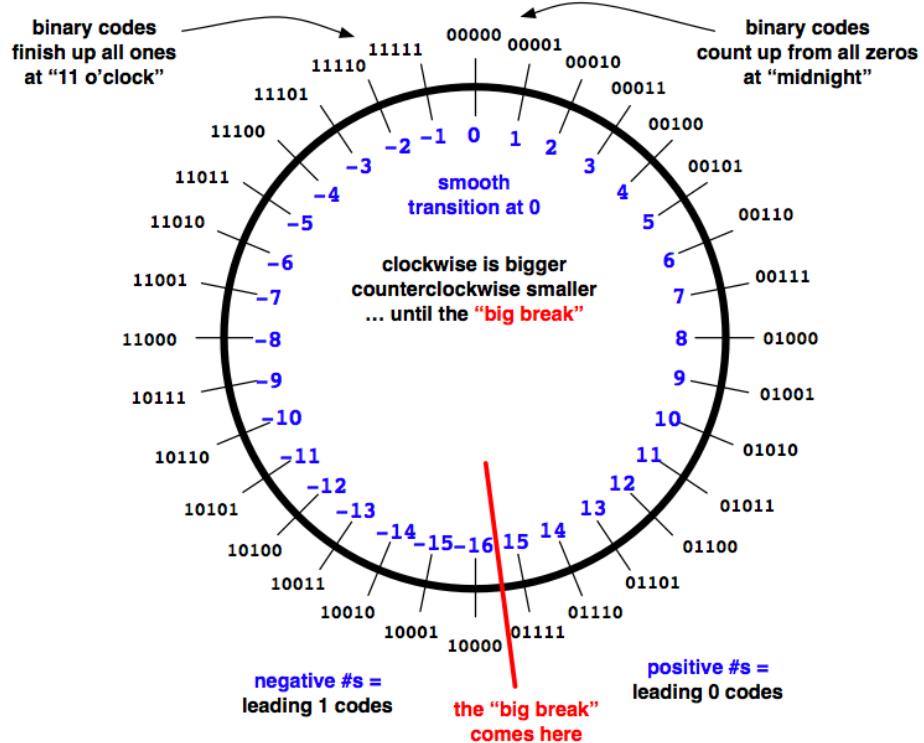
$$\text{2's complement} = (\text{1's complement}) + 1$$

Example 17: Find the 2's Complement of 10110010



Source: Digital Fundamentals by Thomas L. Floyd

27



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 $(10^n - 1) - N$

- $99999..... - N$

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

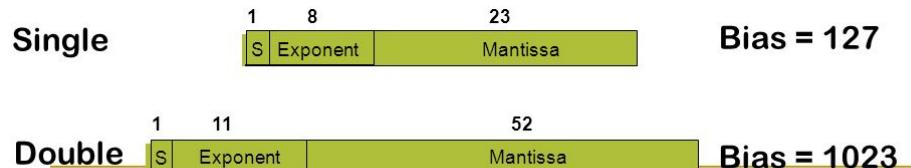
Floating point numbers – the IEEE standard

■ IEEE Standard 754

- ❑ Most (but not all) computer manufacturers use IEEE-754 format
 - ❑ Number represented:

$$(-1)^S * (1.M)^* 2^{(E - \text{Bias})}$$

2 main formats: single and double



Floating-Point Numbers

- ❖ Examples of floating-point numbers in base 10 ...
 - ✧ 5.341×10^3 , 0.05341×10^5 , -2.013×10^{-1} , -201.3×10^{-3}
 - ❖ Examples of floating-point numbers in base 2 ...
 - ✧ 1.00101×2^{23} , 0.0100101×2^{25} , -1.101101×2^{-3} , -1101.101×2^{-6}
 - ✧ Exponents are kept in decimal for clarity
 - ✧ The binary number $(1101.101)_2 = 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-3} = 13.625$
 - ❖ Floating-point numbers should be normalized
 - ✧ Exactly one non-zero digit should appear before the point
 - In a decimal number, this digit can be from 1 to 9
 - In a binary number, this digit should be 1
 - ✧ Normalized FP Numbers: 5.341×10^3 and -1.101101×2^{-3}
 - ✧ NOT Normalized: 0.05341×10^5 and -1101.101×2^{-6}

Normalized Floating Point Numbers

- ❖ For a normalized floating point number (S, E, F)



- ❖ Significand is equal to $(1.F)_2 = (1.f_1 f_2 f_3 f_4 \dots)_2$

◊ IEEE 754 assumes hidden 1. (not stored) for normalized numbers

◊ Significand is 1 bit longer than fraction

- ❖ Value of a Normalized Floating Point Number is

$$\begin{aligned} & (-1)^S \times (1.F)_2 \times 2^{\text{val}(E)} \\ & (-1)^S \times (1.f_1 f_2 f_3 f_4 \dots)_2 \times 2^{\text{val}(E)} \\ & (-1)^S \times (1 + f_1 \times 2^{-1} + f_2 \times 2^{-2} + f_3 \times 2^{-3} + f_4 \times 2^{-4} \dots)_2 \times 2^{\text{val}(E)} \end{aligned}$$

$(-1)^S$ is 1 when S is 0 (positive), and -1 when S is 1 (negative)

Floating-Point Representation

- ❖ A floating-point number is represented by the triple

◊ S is the Sign bit (0 is positive and 1 is negative)

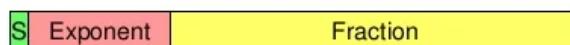
- Representation is called sign and magnitude

◊ E is the Exponent field (signed)

- Very large numbers have large positive exponents
- Very small close-to-zero numbers have negative exponents
- More bits in exponent field increases range of values

◊ F is the Fraction field (fraction after binary point)

- More bits in fraction field improves the precision of FP numbers



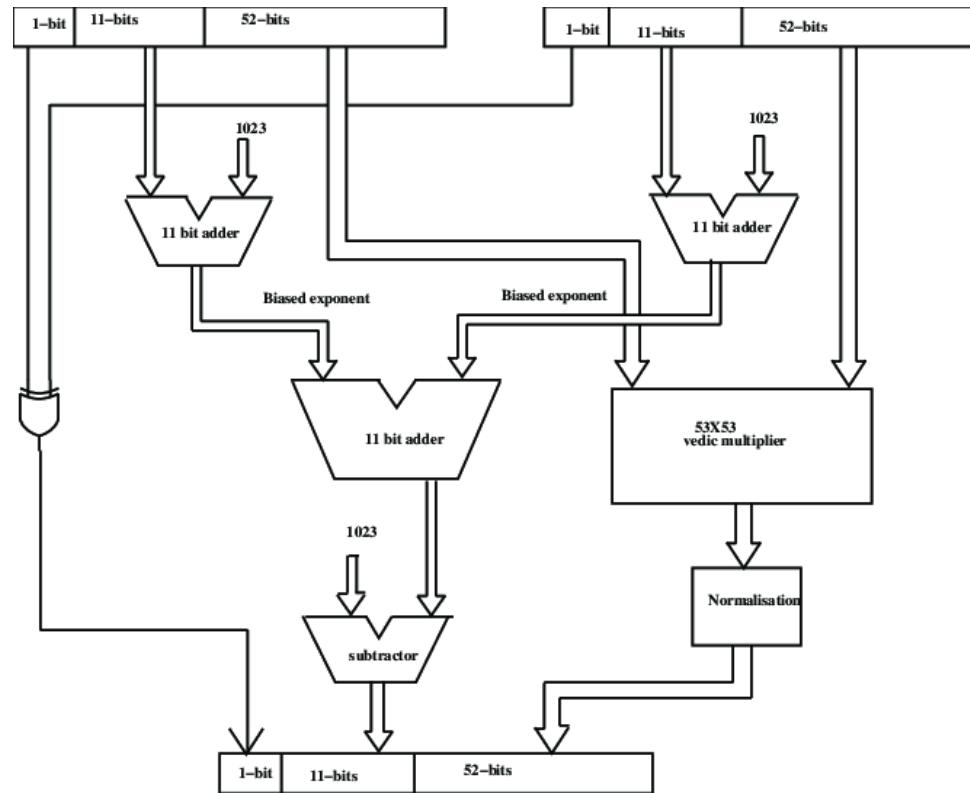
Value of a floating-point number = $(-1)^S \times \text{val}(F) \times 2^{\text{val}(E)}$

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), $\text{val}(E) = E - 1023$
 - ✧ $\text{val}(E=1) = -1022$, $\text{val}(E=1023) = 0$, $\text{val}(E=2046) = 1023$

- ❖ Value of a Normalized Floating Point Number is

$$\begin{aligned}
 & (-1)^S \times (1.F)_2 \times 2^{E-\text{Bias}} \\
 & (-1)^S \times (1.f_1 f_2 f_3 f_4 \dots)_2 \times 2^{E-\text{Bias}} \\
 & (-1)^S \times (1 + f_1 \times 2^{-1} + f_2 \times 2^{-2} + f_3 \times 2^{-3} + f_4 \times 2^{-4} \dots)_2 \times 2^{E-\text{Bias}}
 \end{aligned}$$



ASCII

ASCII control characters		ASCII printable characters										Extended ASCII characters									
00	NULL (Null character)	32	space	64	@	96	`	128	Ç	160	á	192	l	224	ó						
01	SOH (Start of Header)	33	!	65	A	97	a	129	ü	161	í	193	ł	225	þ						
02	STX (Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	ł	226	ö						
03	ETX (End of Text)	35	#	67	C	99	c	131	à	163	ú	195	ñ	227	ô						
04	EOT (End of Trans.)	36	\$	68	D	100	d	132	â	164	ñ	196	—	228	ô						
05	ENQ (Enquiry)	37	%	69	E	101	e	133	ä	165	N	197	+	229	ö						
06	ACK (Acknowledgement)	38	&	70	F	102	f	134	å	166	º	198	å	230	µ						
07	BEL (Bell)	39	'	71	G	103	g	135	ç	167	º	199	À	231	þ						
08	BS (Backspace)	40	(72	H	104	h	136	é	168	¿	200	£	232	þ						
09	HT (Horizontal Tab)	41)	73	I	105	i	137	ë	169	®	201	£	233	ú						
10	LF (Line feed)	42	*	74	J	106	j	138	è	170	¬	202	£	234	ó						
11	VT (Vertical Tab)	43	+	75	K	107	k	139	í	171	½	203	£	235	ú						
12	FF (Form feed)	44	,	76	L	108	l	140	í	172	¼	204	ý	236	ý						
13	CR (Carriage return)	45	-	77	M	109	m	141	í	173	í	205	=	237	ý						
14	SO (Shift Out)	46	.	78	N	110	n	142	À	174	«	206	+	238	-						
15	SI (Shift In)	47	/	79	O	111	o	143	Á	175	»	207	□	239	.						
16	DLE (Data link escape)	48	0	80	P	112	p	144	É	176	»	208	ø	240	≡						
17	DC1 (Device control 1)	49	1	81	Q	113	q	145	æ	177	¤	209	ð	241	±						
18	DC2 (Device control 2)	50	2	82	R	114	r	146	Æ	178	¤	210	€	242	≡						
19	DC3 (Device control 3)	51	3	83	S	115	s	147	ò	179	¤	211	€	243	¼						
20	DC4 (Device control 4)	52	4	84	T	116	t	148	ö	180	—	212	€	244	¶						
21	NAK (Negative acknowl.)	53	5	85	U	117	u	149	ð	181	À	213	—	245	§						
22	SYN (Synchronous idle)	54	6	86	V	118	v	150	ø	182	Á	214	—	246	÷						
23	ETB (End of trans. block)	55	7	87	W	119	w	151	ú	183	Á	215	—	247	.						
24	CAN (Cancel)	56	8	88	X	120	x	152	ÿ	184	®	216	—	248	°						
25	EM (End of medium)	57	9	89	Y	121	y	153	ö	185	—	217	—	249	.						
26	SUB (Substitute)	58	:	90	Z	122	z	154	ú	186	—	218	—	250	.						
27	ESC (Escape)	59	;	91	[123	{	155	ø	187	—	219	—	251	—						
28	FS (File separator)	60	<	92	\	124		156	£	188	—	220	—	252	—						
29	GS (Group separator)	61	=	93]	125	}	157	Ø	189	¢	221	—	253	—						
30	RS (Record separator)	62	>	94	^	126	~	158	×	190	¥	222	—	254	■						
31	US (Unit separator)	63	?	95	—			159	f	191	—	223	—	255	nbsp						
127	DEL (Delete)																				

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



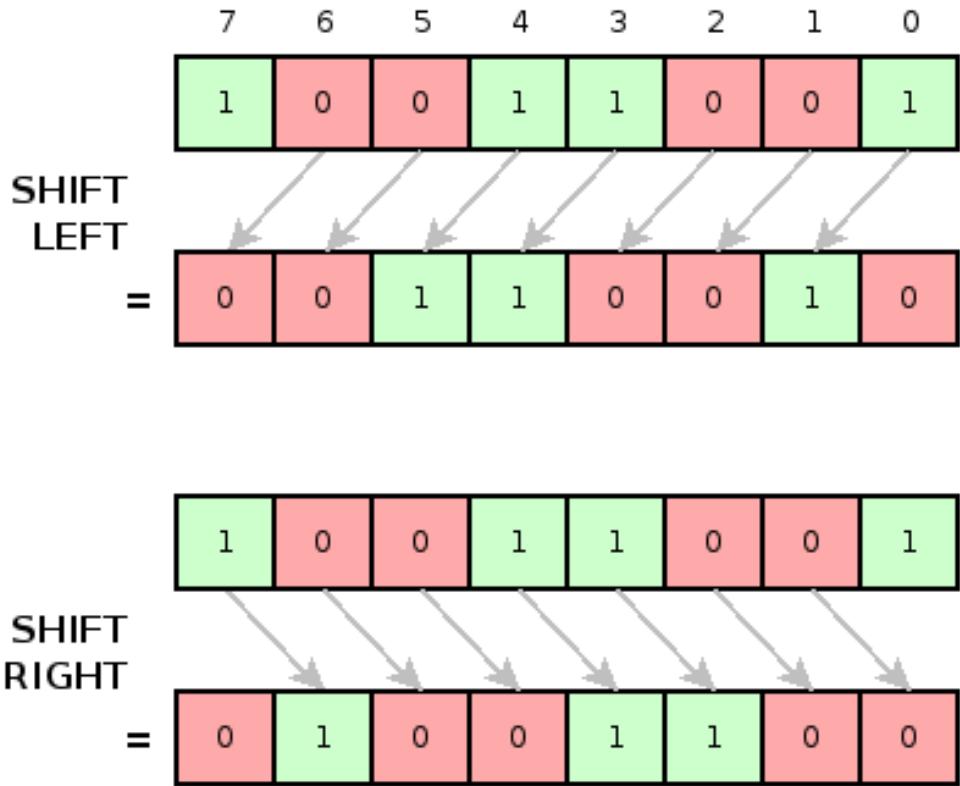
01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

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



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

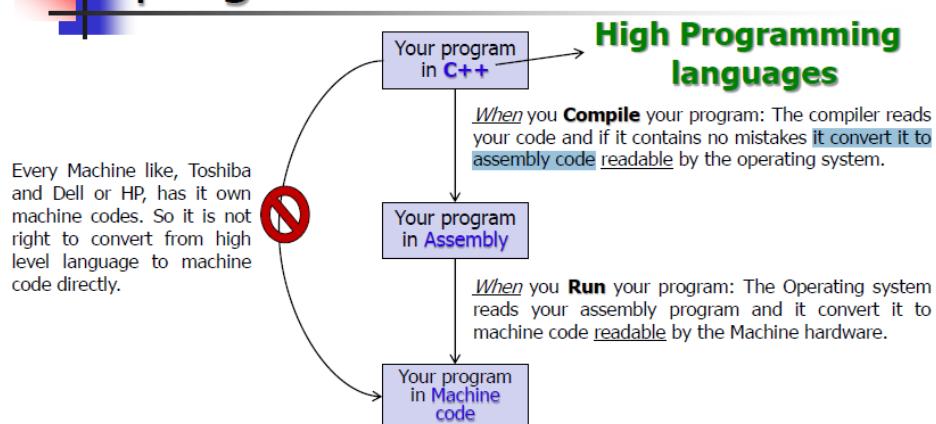
- As stated earlier, a program written in any programming language is a set of logically related instructions. These instructions have two parts, as shown in the following figure:
- The two parts of a programming language instruction are:
 - Operation code (opcode): This part instructs a computer about 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.

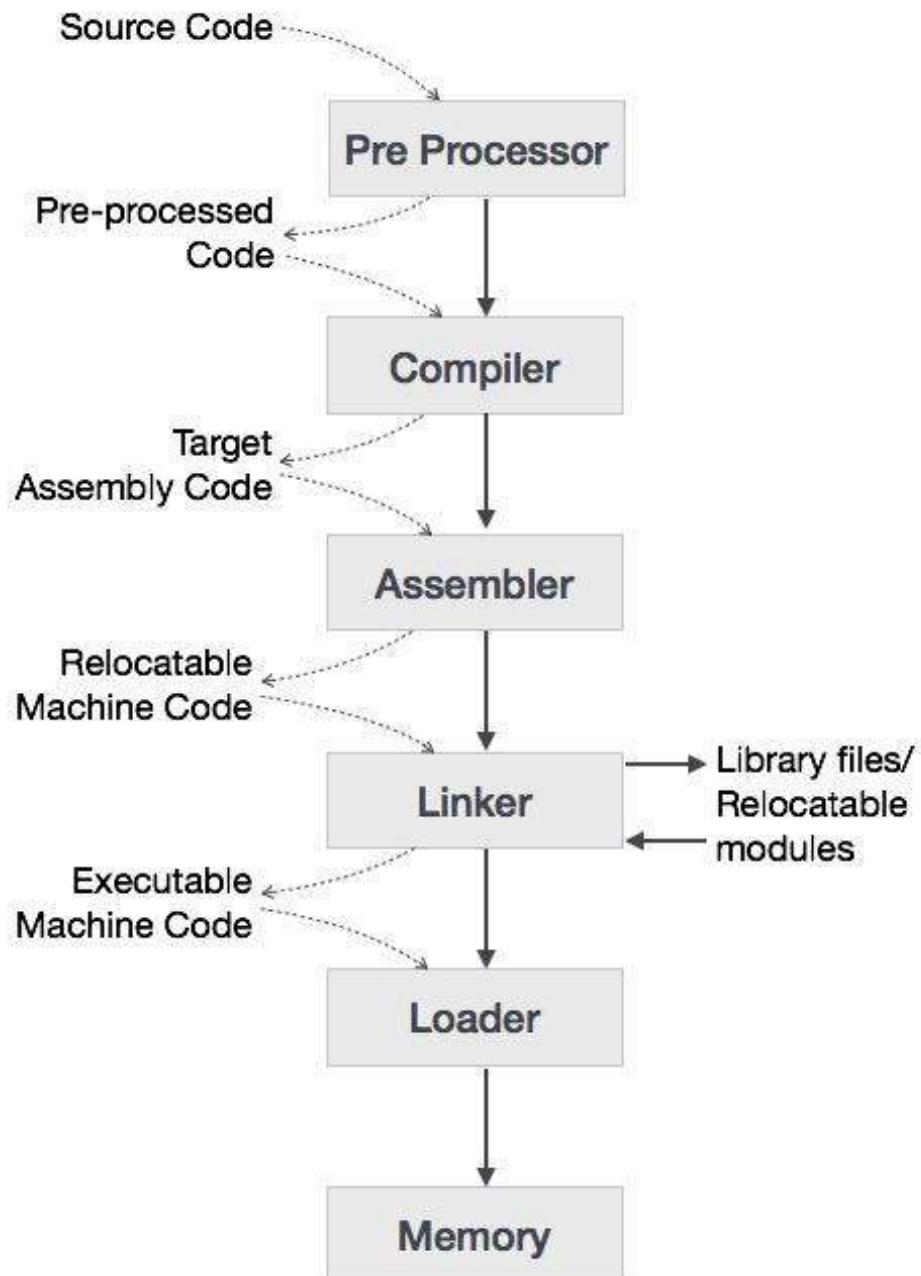


- For example, in the instruction Add A and B, Add is the opcode and A and B are operands

Compiler and Assembler

Compiling and running your program



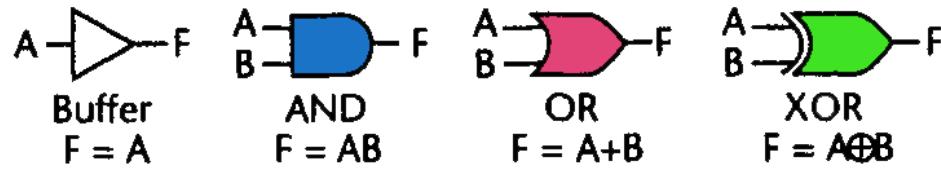


COMPILER VS LINKER VS LOADER

COMPILER	LINKER	LOADER
A software that transforms computer code written in one programming language into another programming language	A computer utility program that takes one or more object files generated by a compiler and combines them into a single executable file	A part of an operating system that is responsible for loading programs to memory
Transforms the source code into object code	Combines multiple object code and links them with libraries	Prepares the executable file for running

Logic Design

- Logic Gates

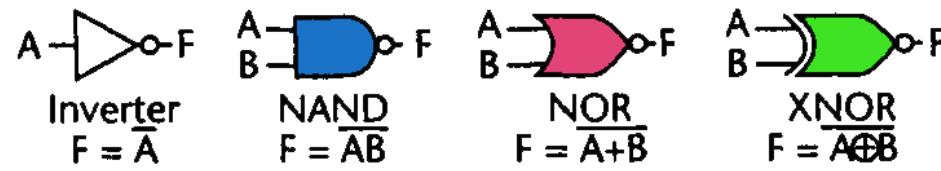


A	F
0	0
1	1

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



A	F
0	1
1	0

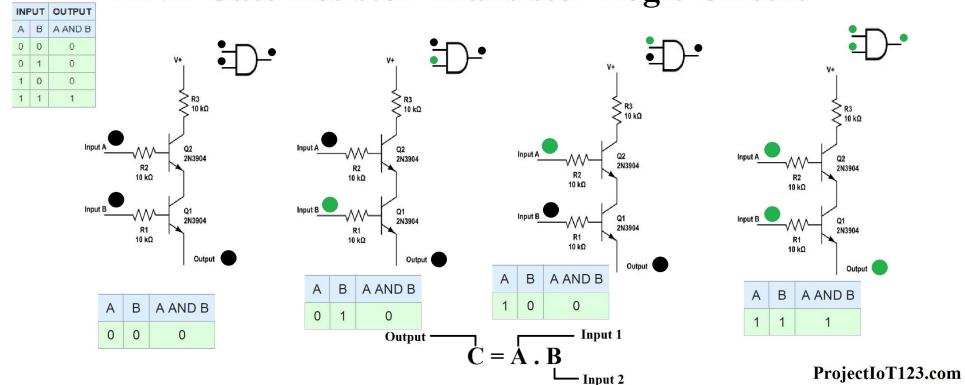
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

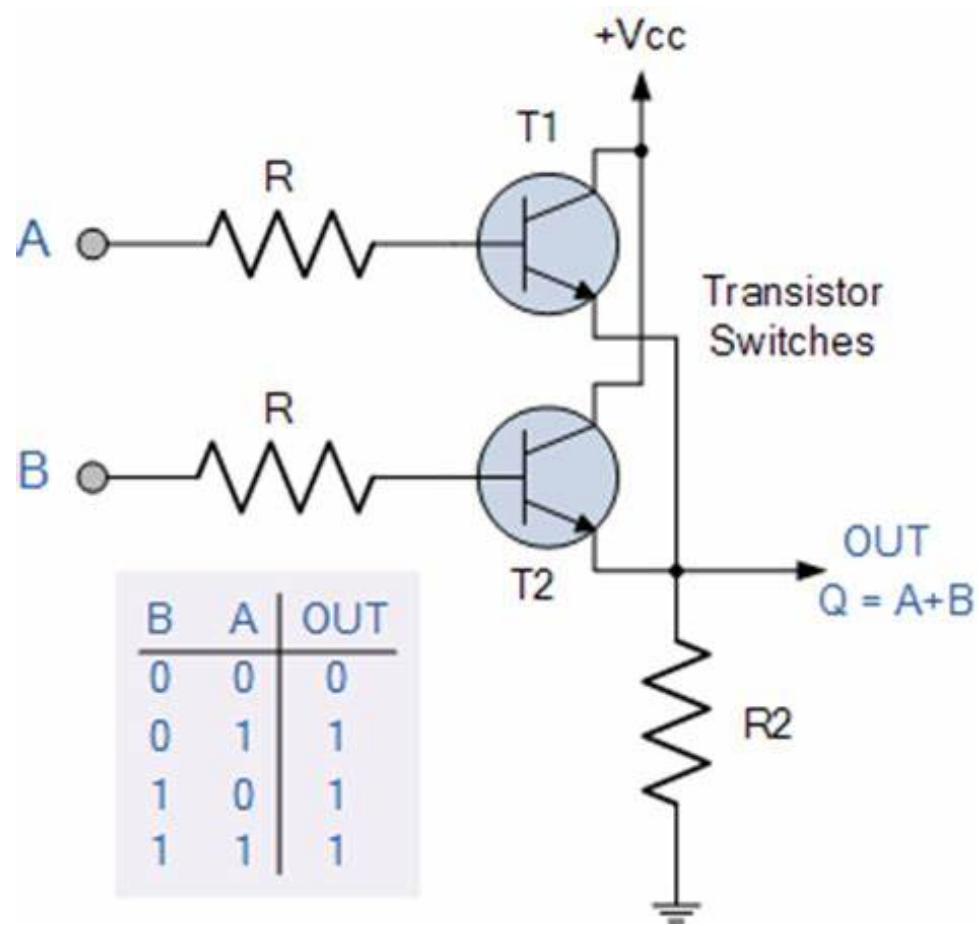
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

- AND Gate

AND Gate Resistor-Transistor Logic Circuit



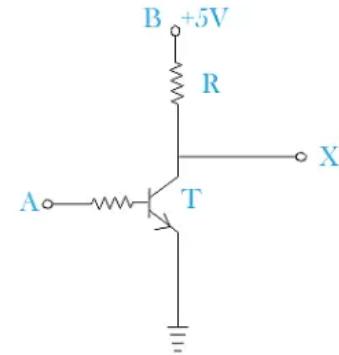
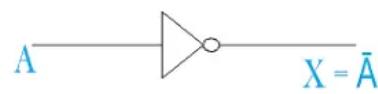
- OR Gate



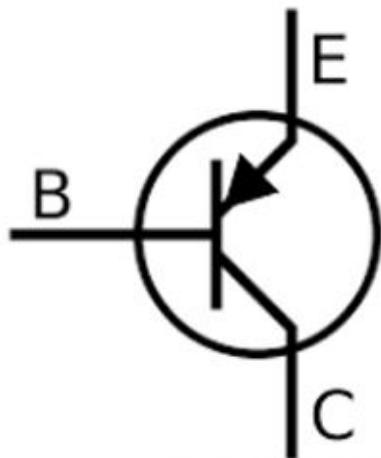
- NOT Gate

What is a NOT Gate?

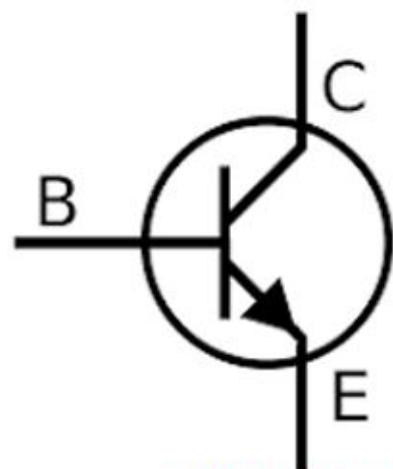
A	X = \bar{A}
0	1
1	0



Electrical 4 U



PNP BJT



NPN BJT

- XOR Gate

XOR GATE Truth Table



BOOLEAN EXPRESSION

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

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

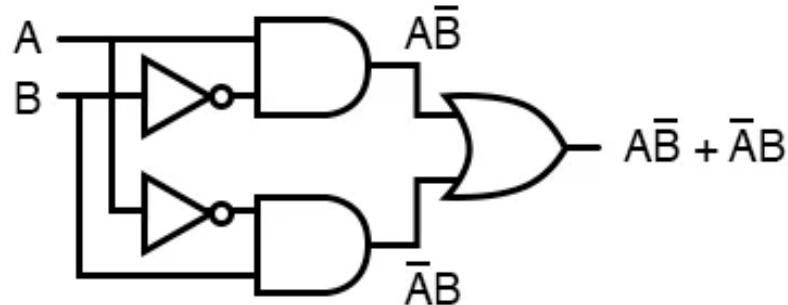
Output **C = A \oplus B** **Input1**
 | **Input2**

INPUT		OUTPUT
A	B	A \oplus B
0	0	0
0	1	1
1	0	1
1	1	0

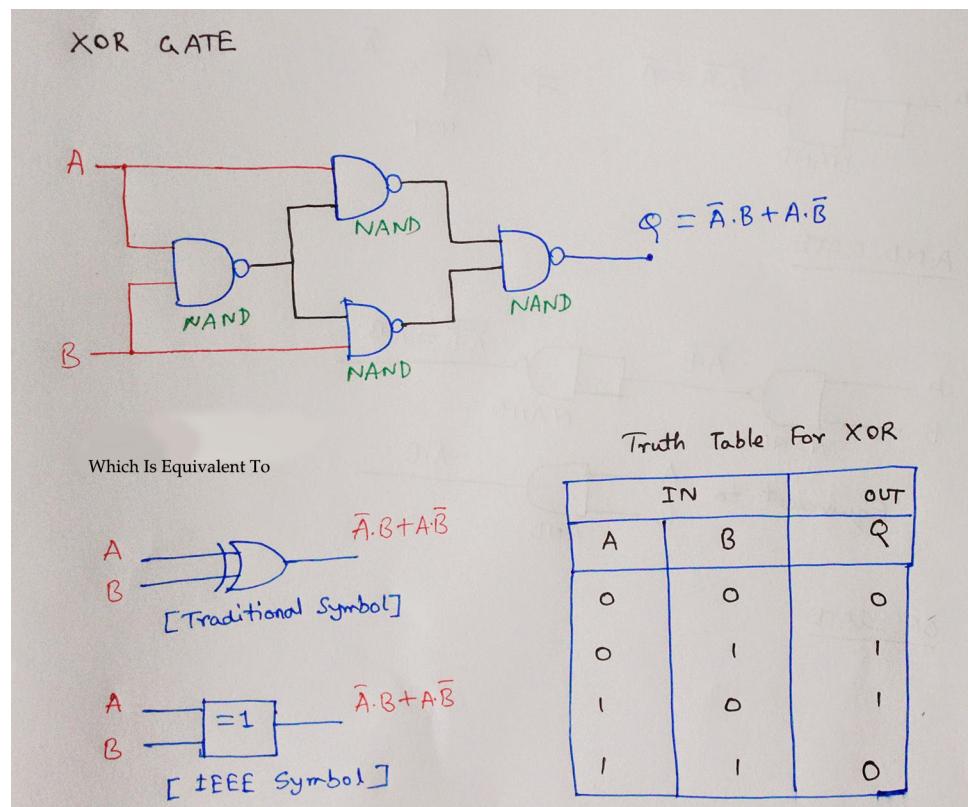
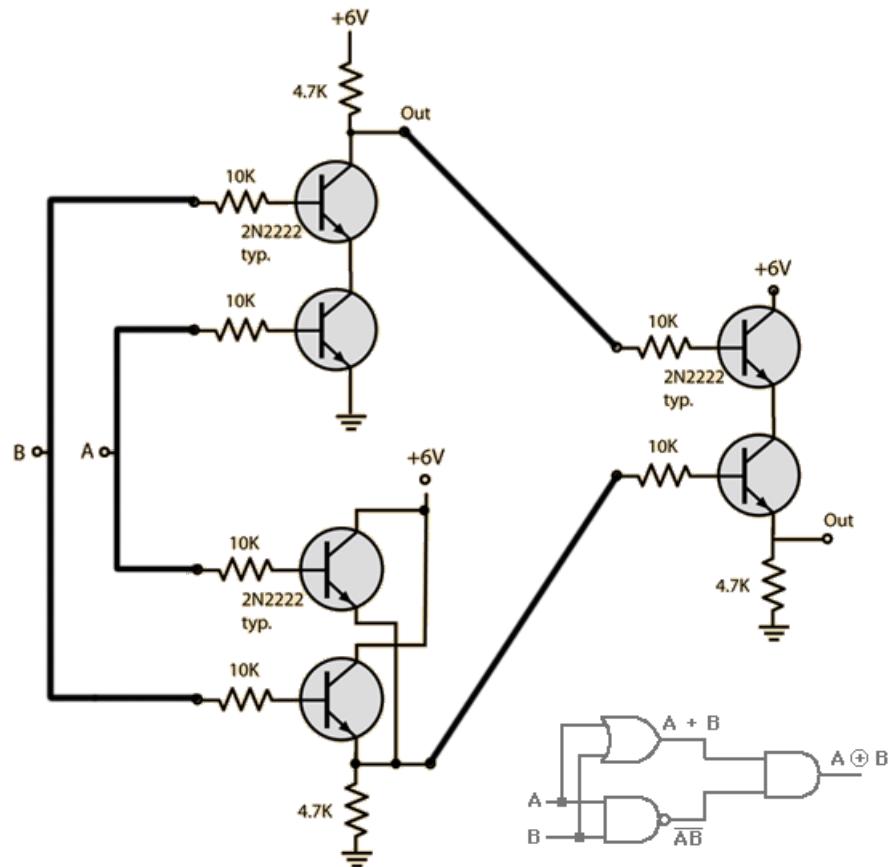
ProjectIoT123.com



. . . is equivalent to . . .

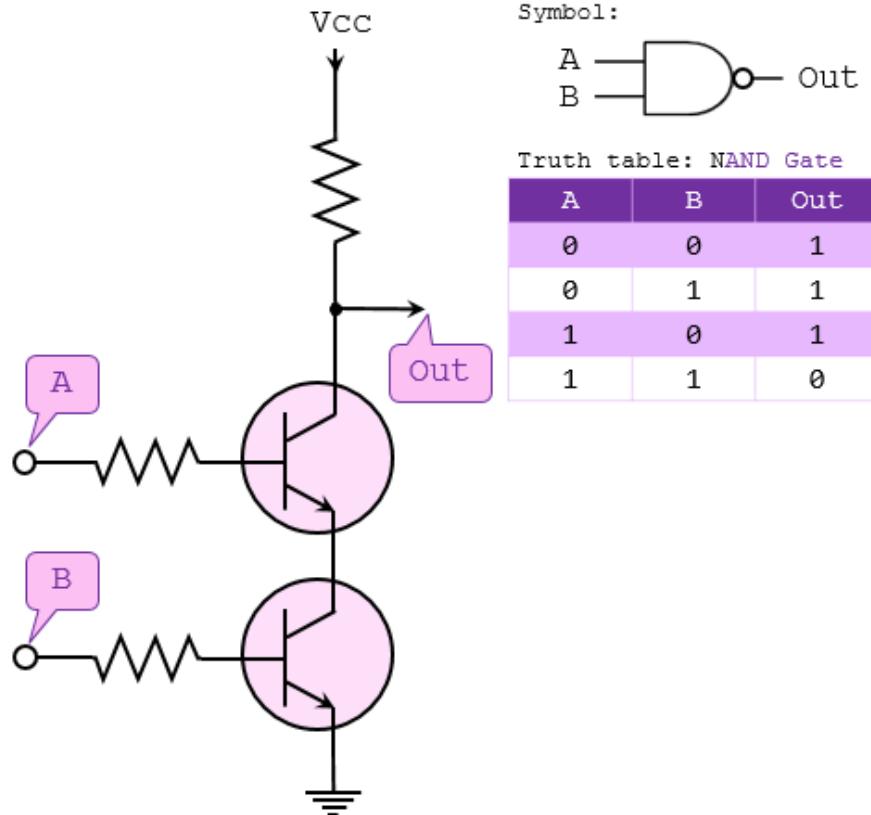


$$A \oplus B = A\overline{B} + \overline{A}B$$



- **NAND Gate**

NAND Gate



The NAND gate as a universal gate

Logic function

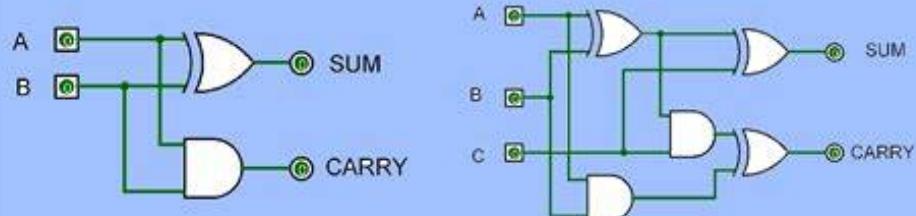
$$\begin{aligned} A &\rightarrow \text{NAND}(A, A) = \bar{A} \\ A, B &\rightarrow \text{NAND}(A, B) = A \cdot \bar{B} \\ A, B &\rightarrow \text{NAND}(B, A) = \bar{A} \cdot B \\ A, B &\rightarrow \text{NAND}(A, B) = \bar{A} + B \\ A, B &\rightarrow \text{NAND}(B, A) = A + \bar{B} \\ A, B &\rightarrow \text{NAND}(A, B) = A \oplus B \\ A, B &\rightarrow \text{NAND}(B, A) = \bar{A} \oplus B \end{aligned}$$

NAND gate only

$$\begin{aligned} A &\rightarrow \text{NAND}(A, A) = \bar{A} \\ A, B &\rightarrow \text{NAND}(A, \text{NAND}(A, B)) = A \cdot \bar{B} \\ A, B &\rightarrow \text{NAND}(B, \text{NAND}(A, B)) = \bar{A} \cdot B \\ A, B &\rightarrow \text{NAND}(\text{NAND}(A, B), \text{NAND}(A, B)) = A + B \\ A, B &\rightarrow \text{NAND}(\text{NAND}(A, B), \text{NAND}(B, A)) = \bar{A} + B \\ A, B &\rightarrow \text{NAND}(\text{NAND}(A, B), \text{NAND}(B, A)) = A \oplus B \\ A, B &\rightarrow \text{NAND}(\text{NAND}(A, \text{NAND}(A, B)), \text{NAND}(B, \text{NAND}(A, B))) = \bar{A} \oplus B \end{aligned}$$

- Half-Adder vs Full-Adder

Difference Between Half Adder And Full Adder



- **Half-Adder**

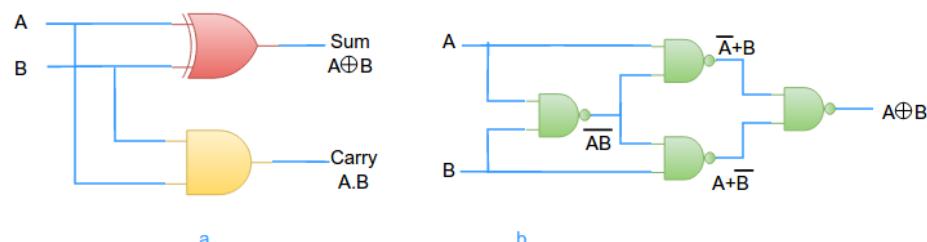
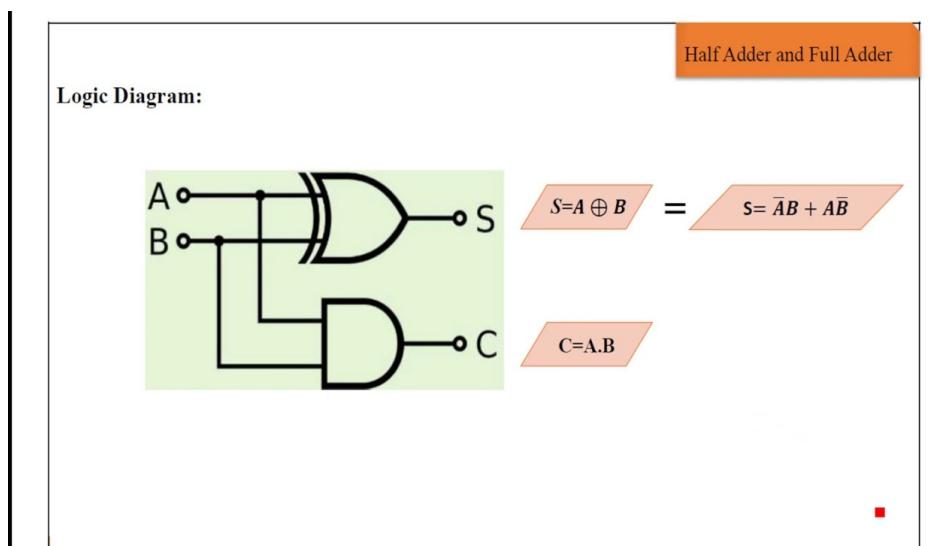
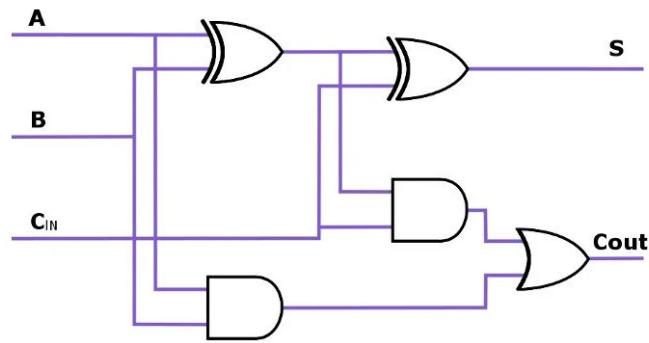
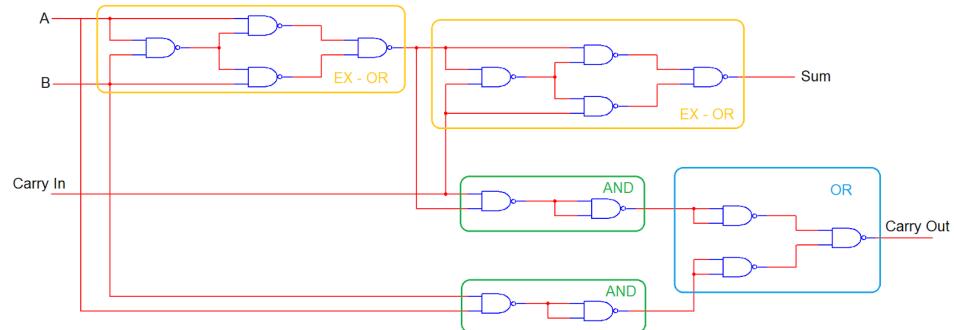
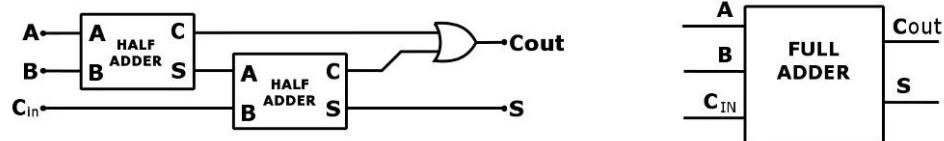


Figure-1 : a)Half Adder b)XOR implementation using NAND gates

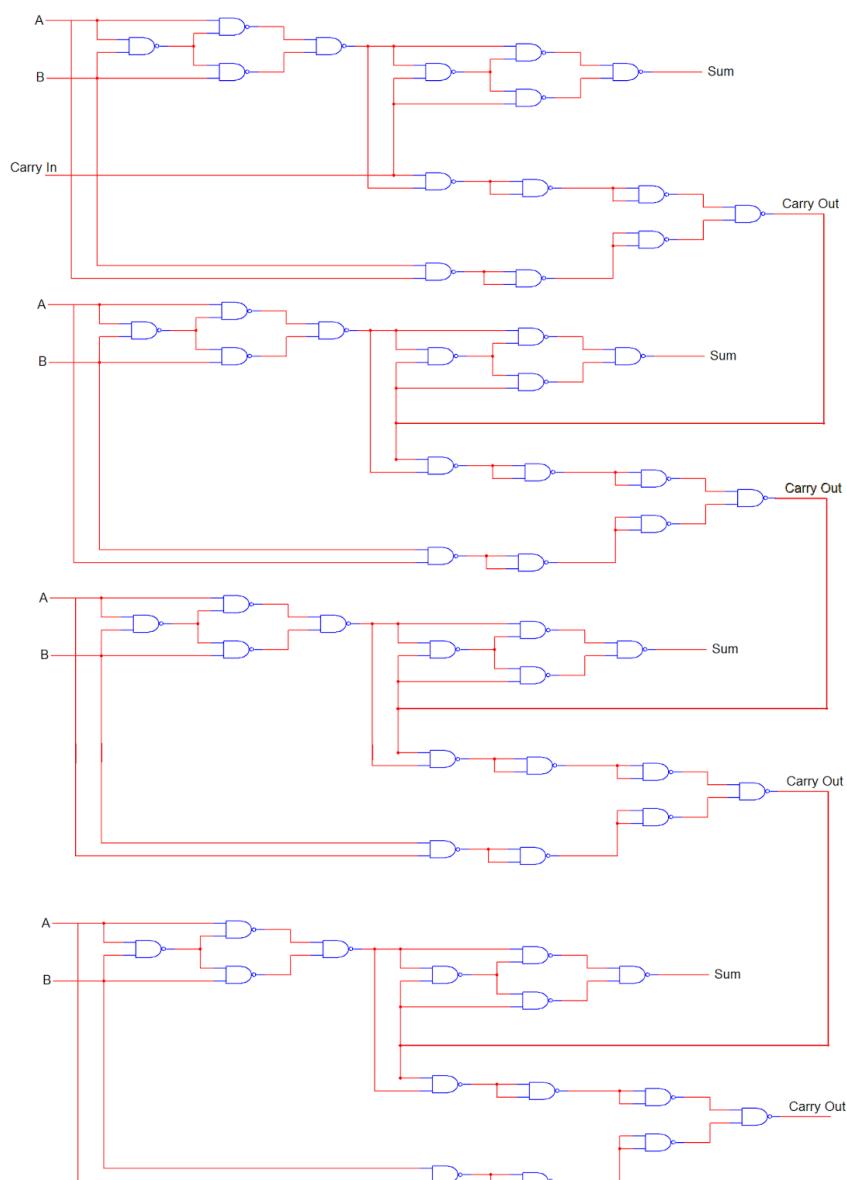
- **Full Adder**



Input			Output	
A	B	C _{in}	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



- **Subtract**

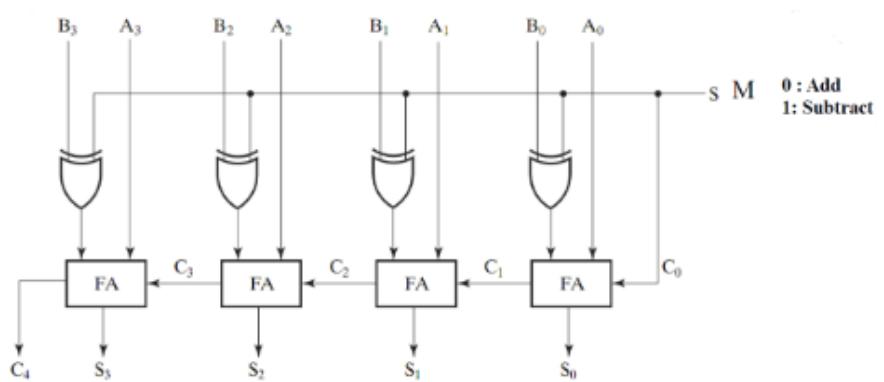
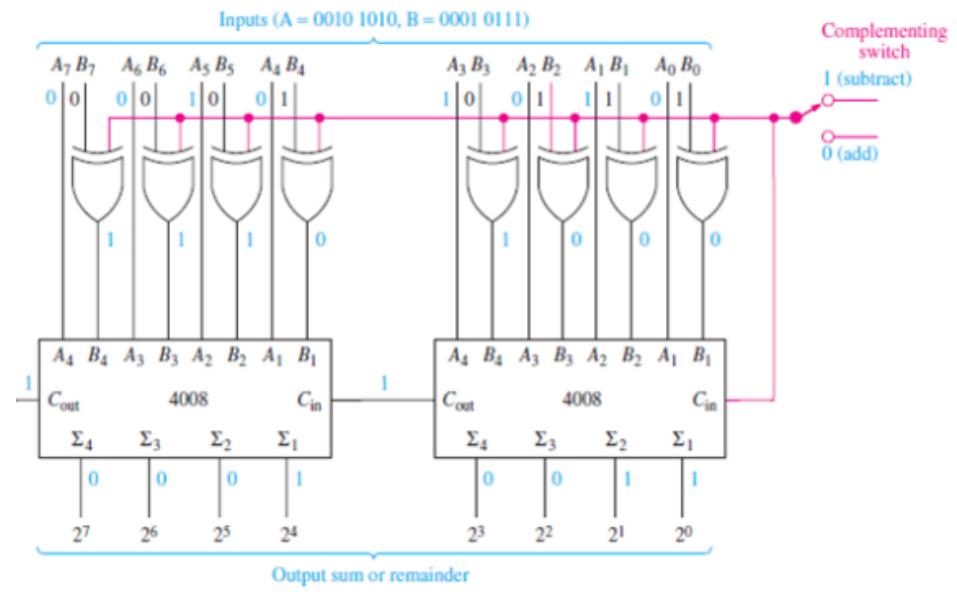


Fig. 1 Modular design of 4-bit adder/subtractor

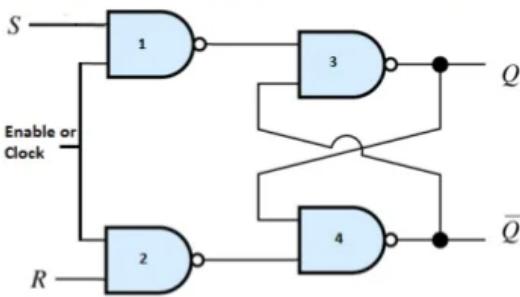


Circuits as Memory

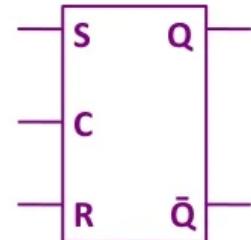
- SR Flip Flop



Gated Latch-Clocked RS Flip-flop

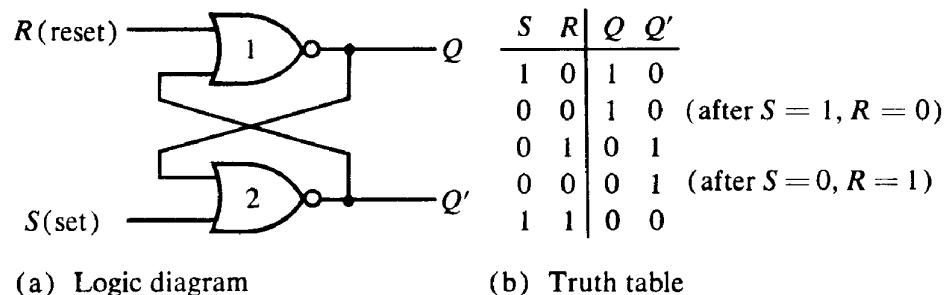


A clocked SR flip-flop.

Logical Symbol

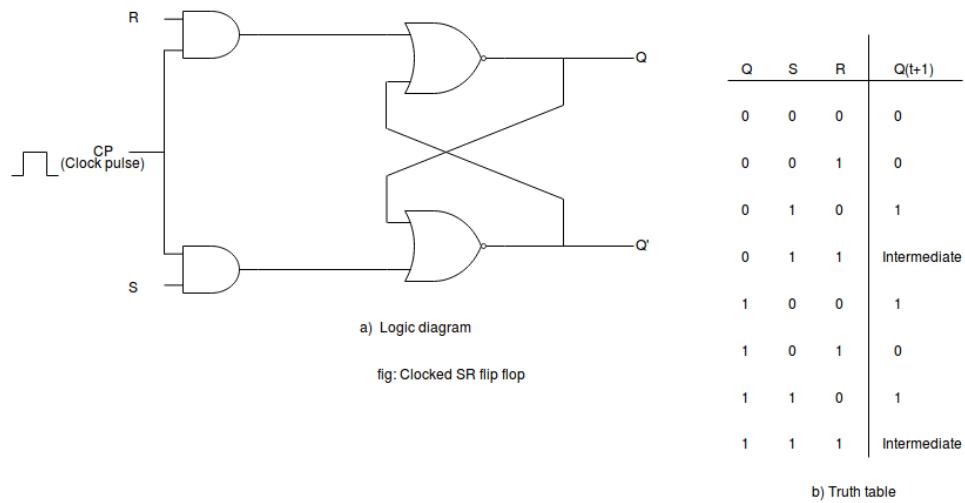
R	S	Enable	Q_n
0	0	\times	Q_{n-1}
0	1	1	1
1	0	1	0
1	1	1	Not allowed
\times	\times	0	Q_{n-1}

Truth table



(a) Logic diagram

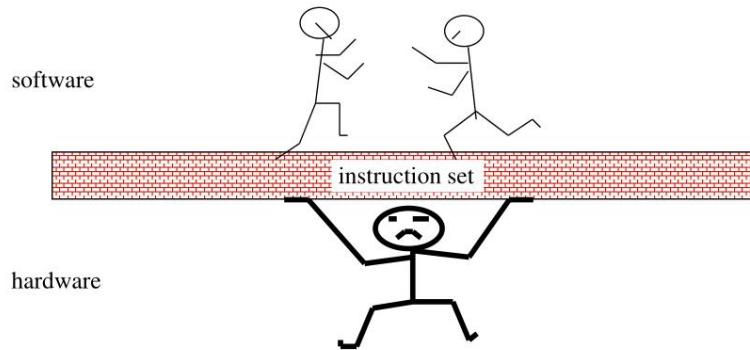
(b) Truth table



Computer Organization and Architecture

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

Instruction Set Architecture: Critical Interface

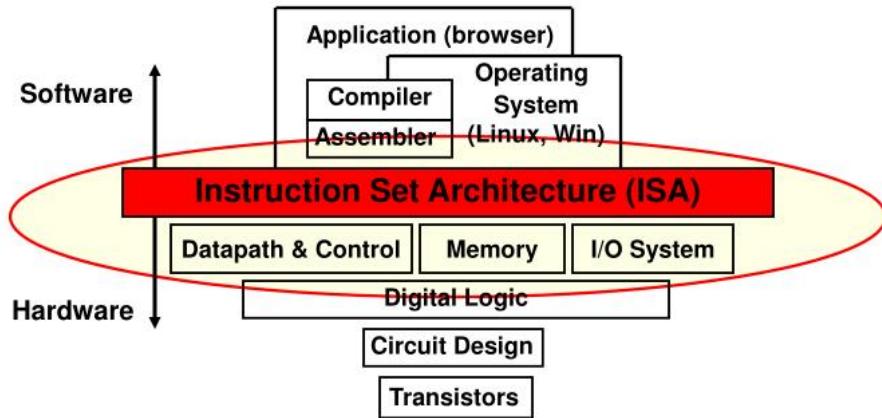


Properties of a good abstraction

- ◆ 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



Instruction Set Architecture

- The computer ISA defines all the *programmer-visible* components and operations of the computer
 - Memory organization
 - address space -- how many locations can be addressed?
 - addressability -- 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).

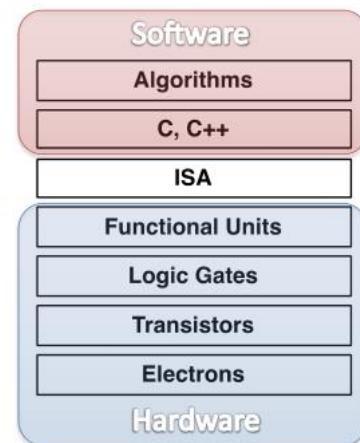
BYU CS 224

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

Operations & no of operands	Address specifier 1	Address field 1	...	Address specifier n	Address field n
-----------------------------	---------------------	-----------------	-----	---------------------	-----------------

Variable: VAX (1-53 bytes)

Operation	Address field 1	Address field 2	Address field3
-----------	-----------------	-----------------	----------------

Fixed: DLX, MIPS, PowerPC, SPARC

Operation	Address Specifier	Address field
-----------	-------------------	---------------

Operation	Address Specifier 1	Address Specifier 2	Address field
-----------	---------------------	---------------------	---------------

Operation	Address Specifier	Address field 1	Address field 2
-----------	-------------------	-----------------	-----------------

Hybrid : IBM 360/370, Intel 80x86

EECC551 - Shaaban

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

CICS vs RICS

CISC vs. RISC

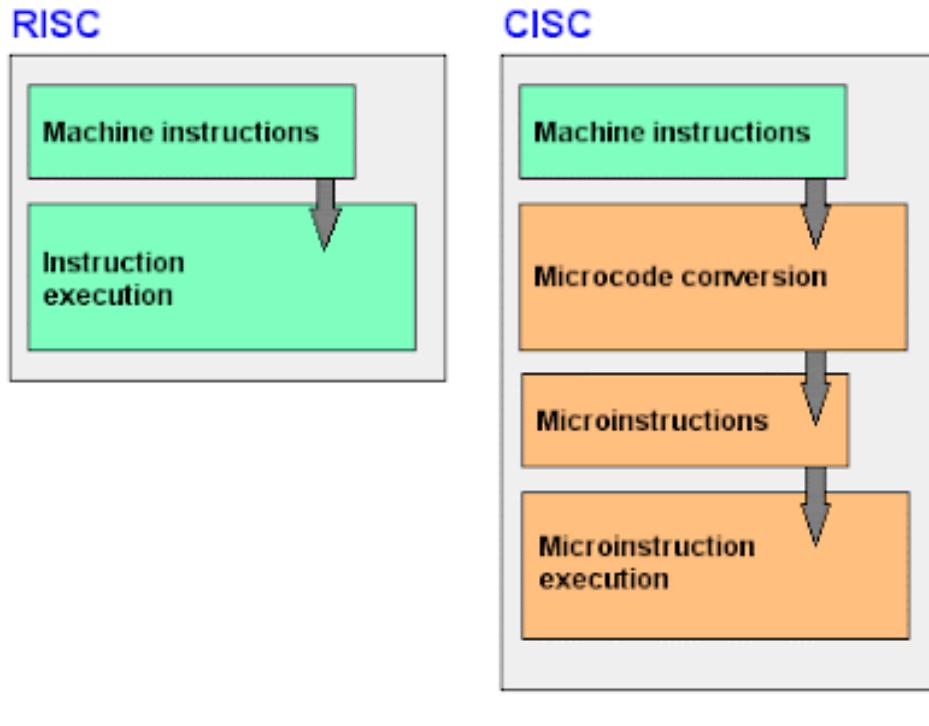
Complex Instruction Set Computer

- 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

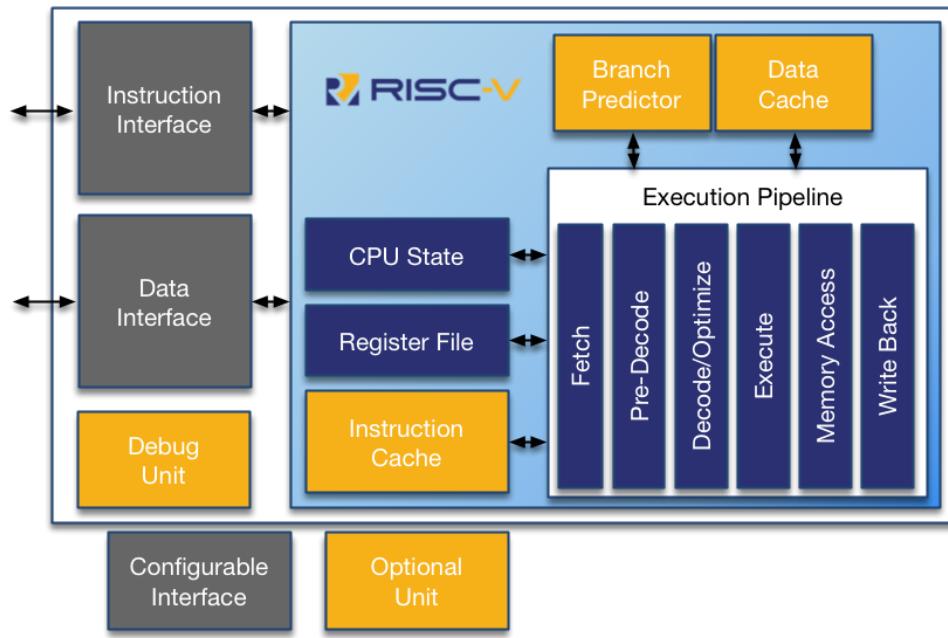
- Few instructions
 - e.g., 30-40
- Smaller chip, smaller pin count, & very low-power consumption
 - Simple but fast instructions
- Harvard architecture, instruction pipelining
- Industry trend for microprocessor design
 - e.g., Intel Pentium, PIC

24



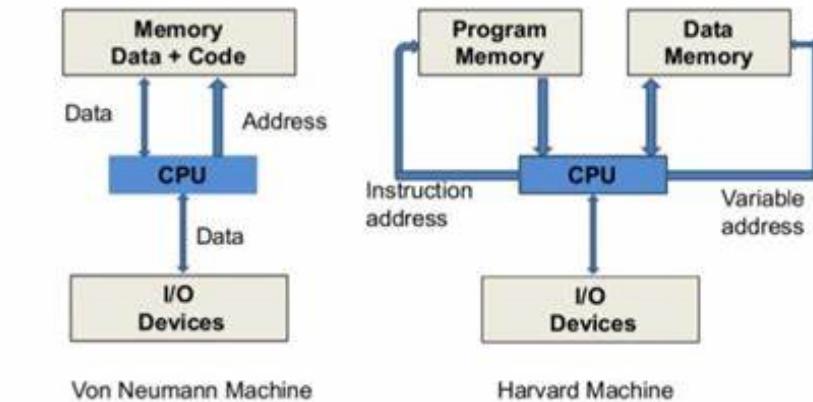
32-bit RISC-V Instruction Formats																																										
Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
Register/register	func7						rs2						rs1						func3						rd						opcode											
Immediate	imm[11:0]												rs1						func3						rd						opcode											
Upper Immediate	imm[31:12]												rd						rd						opcode						opcode											
Store	imm[11:5]						rs2						rs1						func3						imm[4:0]						opcode											
Branch	[12]	imm[10:5]						rs2						rs1						func3						imm[4:1]		[11]		opcode												
Jump	[20]	imm[10:1]						[11]	imm[19:12]						rd						rd						opcode						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

Von Neumann vs. Harvard Architecture



6

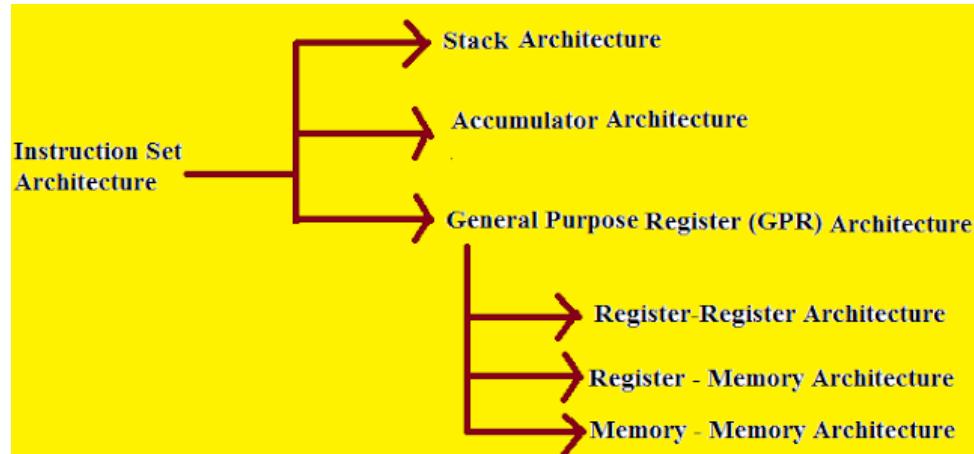
Memory & Timing model

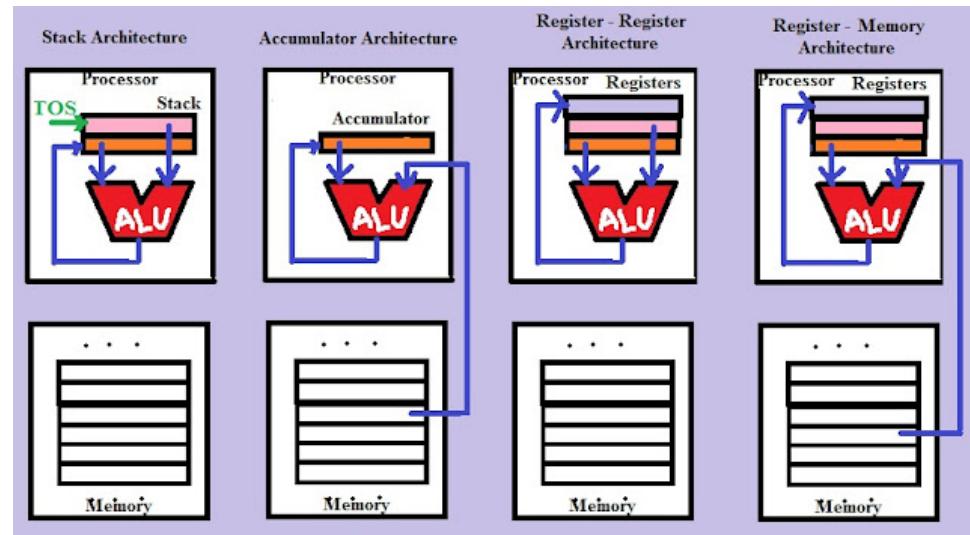
ISA: STORAGE RESOURCES

- "Harvard architecture": Separate instruction and data memories
- Permit use of **single clock cycle per instruction** implementation
- Due to use of "cache" in modern computer architectures, it is a fairly **realistic model**

Instruction memory $2^{15} \times 16$	Program counter (PC)
Data memory $2^{15} \times 16$	Register file 8×16

Classification of ISA (or) Types of ISA



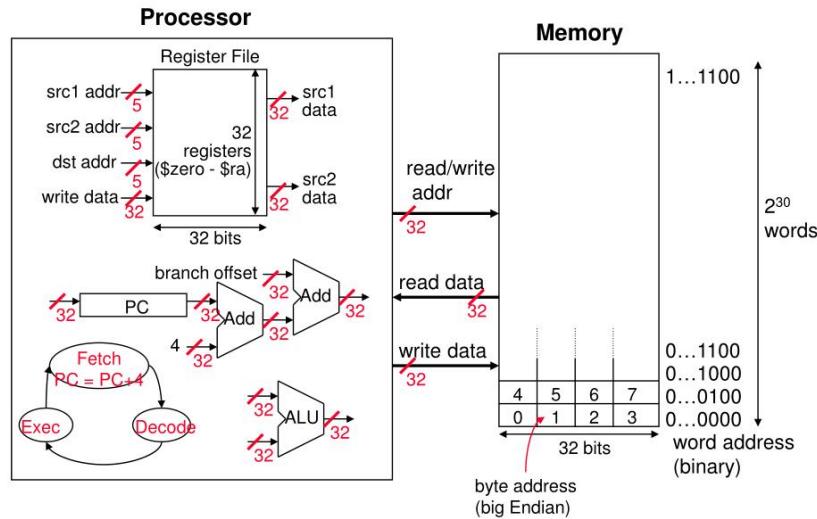


ISA - MIPS Instruction Format and Addressing Model

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

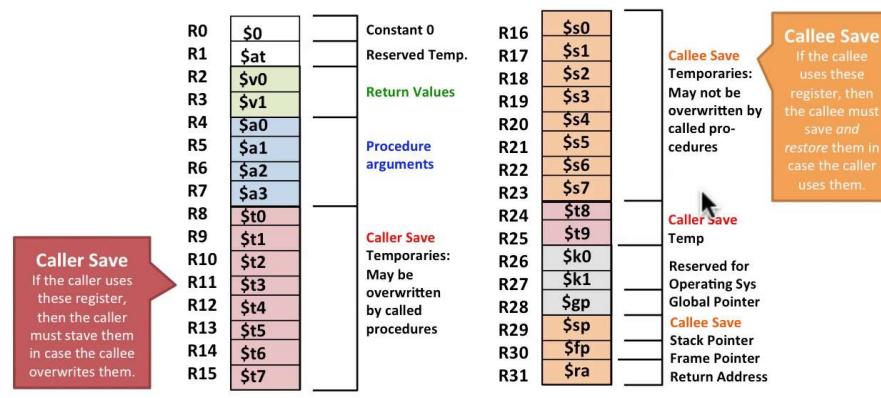
- When writing assembly, these registers can be referenced by their address (number) or name
- General purpose and special purpose registers

#	Name	Purpose	#	Name	Purpose
\$0	\$zero	Constant zero	\$16	\$s0	Temporary – Callee-saved
\$1	\$at	Reserved for assembler	\$17	\$s1	
\$2	\$v0	Function return value	\$18	\$s2	
\$3	\$v1		\$19	\$s3	
\$4	\$a0	Function parameter	\$20	\$s4	
\$5	\$a1		\$21	\$s5	
\$6	\$a2		\$22	\$s6	
\$7	\$a3		\$23	\$s7	
\$8	\$t0	Temporary – Caller-saved	\$24	\$t8	Temporary – Caller-saved
\$9	\$t1		\$25	\$t9	
\$10	\$t2		\$26	\$k0	Reserved for OS
\$11	\$t3		\$27	\$k1	
\$12	\$t4		\$28	\$gp	Global pointer
\$13	\$t5		\$29	\$sp	Stack pointer
\$14	\$t6		\$30	\$fp	Frame pointer
\$15	\$t7		\$31	\$ra	Function return address

17

Who saves what?

53



Instruction Format

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R:	op	rs	rt	rd	shamt	funct
I:	op	rs	rt		address / immediate	
J:	op			target address		

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

Op ⁶	Rs ⁵	Rt ⁵	Rd ⁵	sa ⁵	funct ⁶
-----------------	-----------------	-----------------	-----------------	-----------------	--------------------

❖ **Immediate (I-Type)**

◊ 16-bit immediate constant is part in the instruction

Op ⁶	Rs ⁵	Rt ⁵	immediate ¹⁶
-----------------	-----------------	-----------------	-------------------------

❖ **Jump (J-Type)**

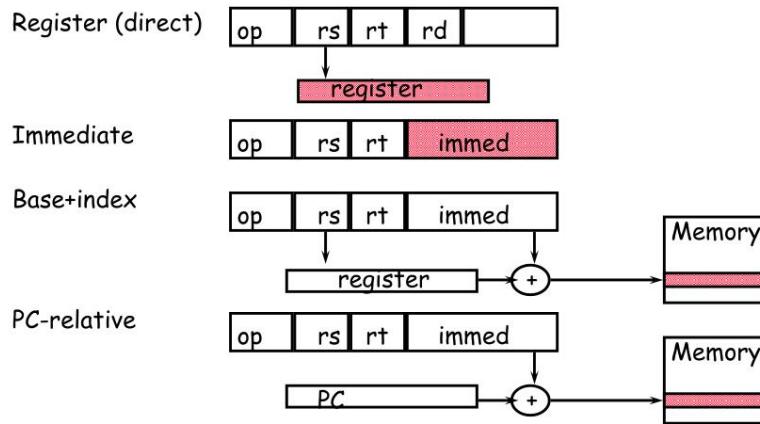
◊ Used by jump instructions

Op ⁶	immediate ²⁶
-----------------	-------------------------

Addressing Model

Example: MIPS Instruction Formats and Addressing Modes

- All instructions 32 bits wide



ECE 361

3-45

5.6 Addressing Modes

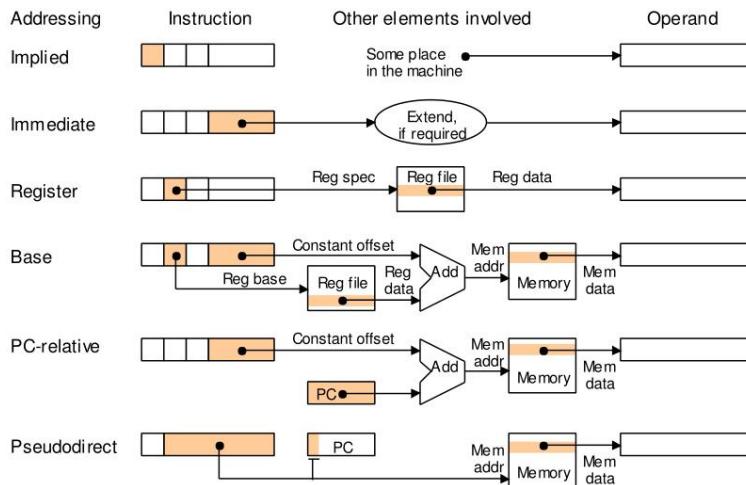


Figure 5.11 Schematic representation of addressing modes in MiniMIPS.

Computer Architecture, Instruction-Set Architecture

Slide 22

Instruction Categories

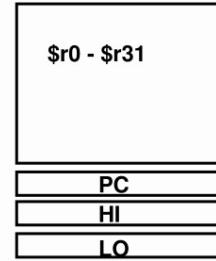
MIPS: ISA

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

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

OP	\$rs	\$rt	\$rd	sa	funct
OP	\$rs	\$rt	immediate		
OP	jump target				

ISA - Performance

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

$$\text{MIPS} = \text{Clock rate} / (\text{CPI} \times 10^6) = 100 \text{ MHz} / (\text{CPI} \times 10^6)$$

$$\text{CPI} = \text{CPU execution cycles} / \text{Instructions count}$$

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

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} / \text{Clock rate}$$

- For compiler 1:

- $\text{CPI}_1 = (5 \times 1 + 1 \times 2 + 1 \times 3) / (5 + 1 + 1) = 10 / 7 = 1.43$

- $\text{MIP}_1 = 100 / (1.43 \times 10^6) = 70.0$

- $\text{CPU time}_1 = ((5 + 1 + 1) \times 10^6 \times 1.43) / (100 \times 10^6) = 0.10 \text{ seconds}$

- For compiler 2:

- $\text{CPI}_2 = (10 \times 1 + 1 \times 2 + 1 \times 3) / (10 + 1 + 1) = 15 / 12 = 1.25$

- $\text{MIP}_2 = 100 / (1.25 \times 10^6) = 80.0$

- $\text{CPU time}_2 = ((10 + 1 + 1) \times 10^6 \times 1.25) / (100 \times 10^6) = 0.15 \text{ seconds}$

EECC551 - Shaaban

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

Speed Up Equation for Pipelining

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

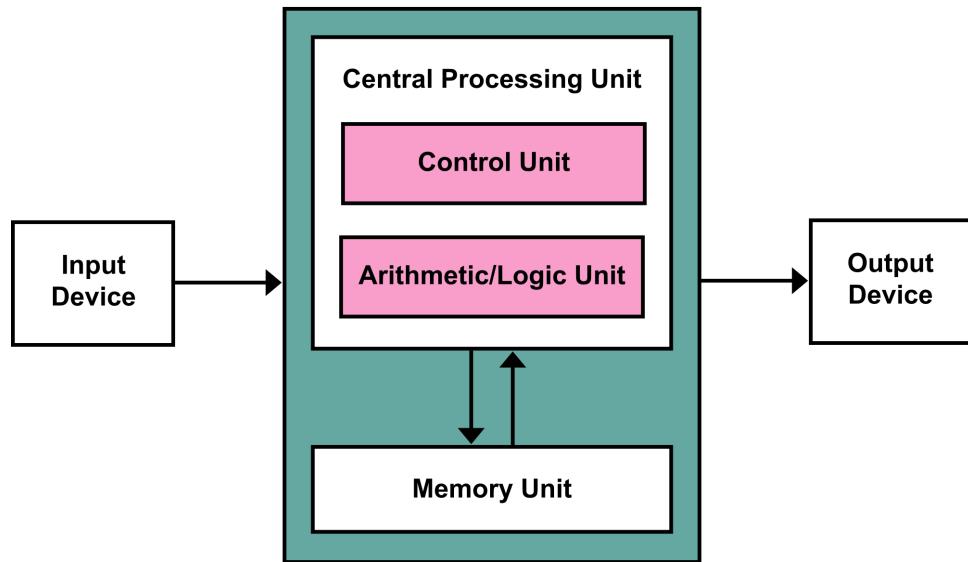
$$\text{Speedup} = \frac{\text{Ideal CPI} \times \text{Pipeline depth}}{\text{Ideal CPI} + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}}$$

For simple RISC pipeline, CPI = 1:

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

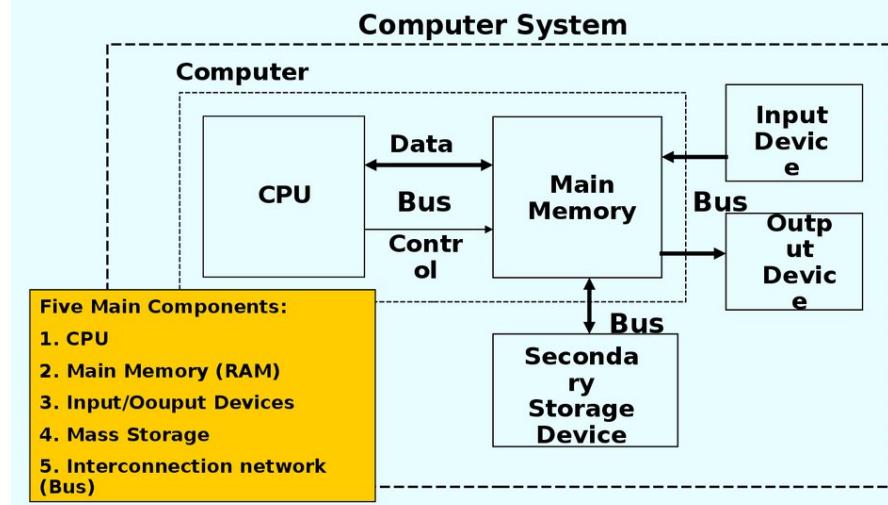
CS211 41

Processor - Von Neumann Architecture

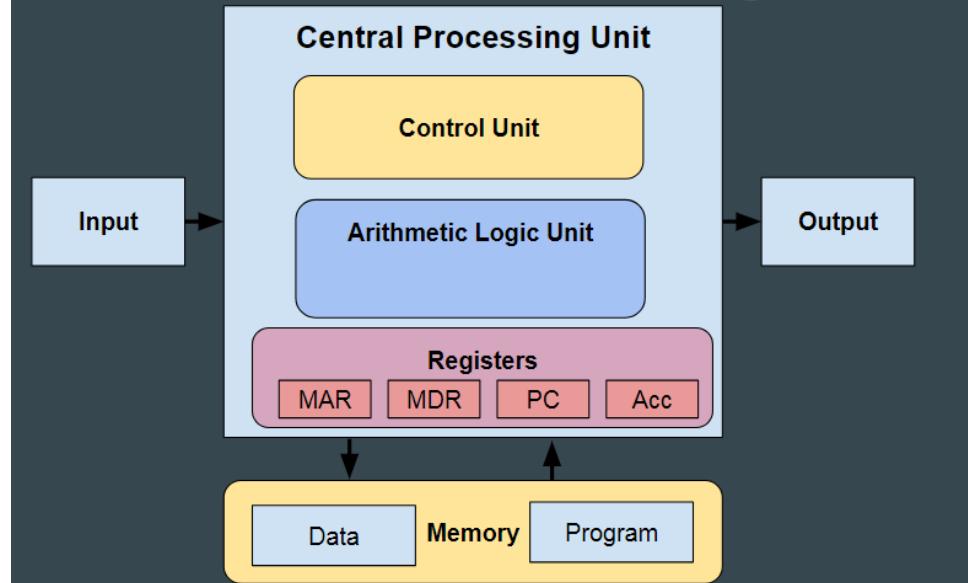


von Neumann Architecture

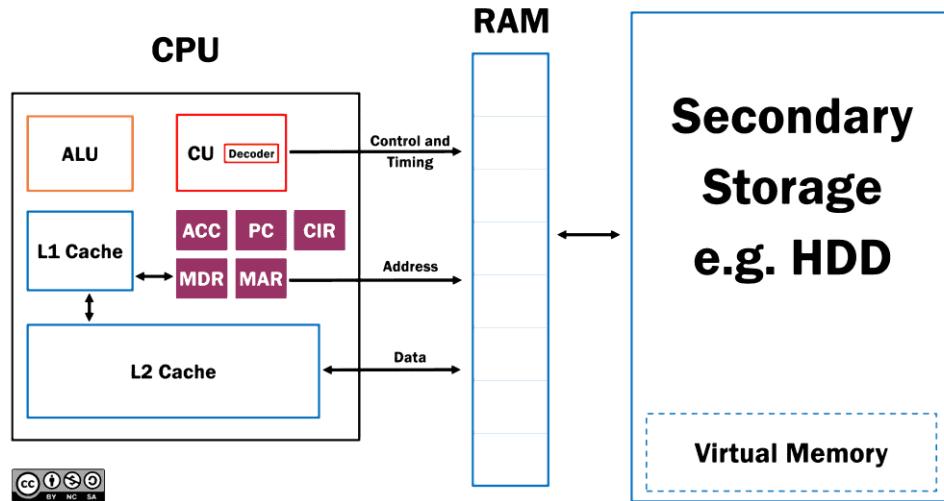
- A more complete view of the computer *system* architecture that integrates interaction (human or otherwise) consists of:



Von Neumann Architecture Diagram



Computer Systems - Von Neumann Architecture



Processor - Pipeline

Pipelining Lessons

- Pipelining doesn't help latency (execution time) of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- For a given speedup = Number of pipe stages
- Time to "fill" pipeline and time to "drain" it reduces speedup
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages also reduces speedup

The Fetch-Execute Cycle

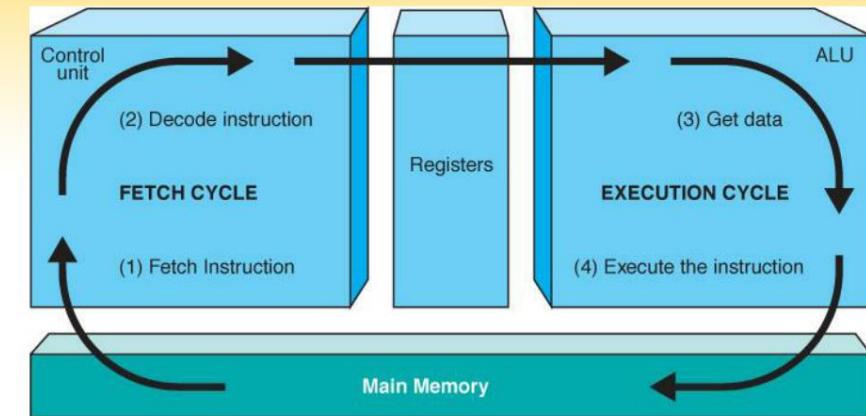
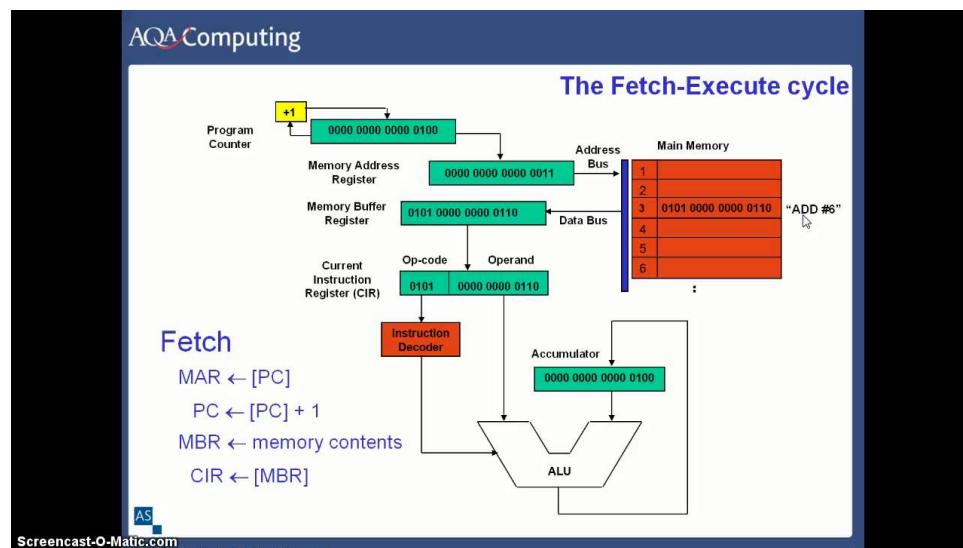
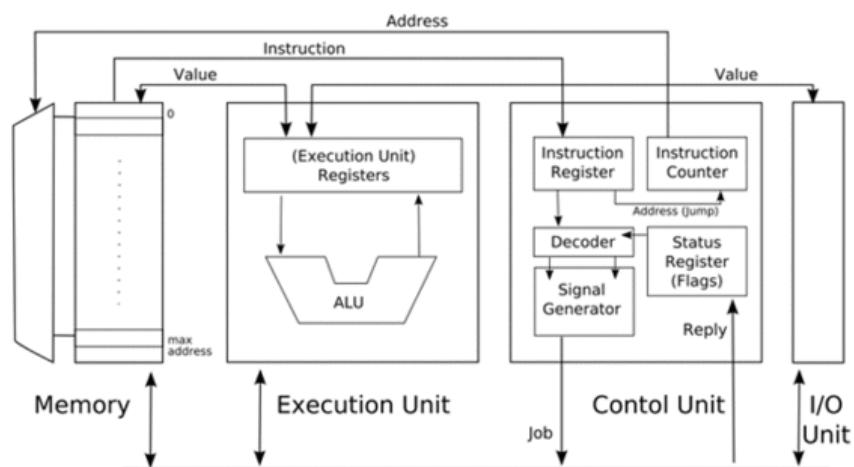
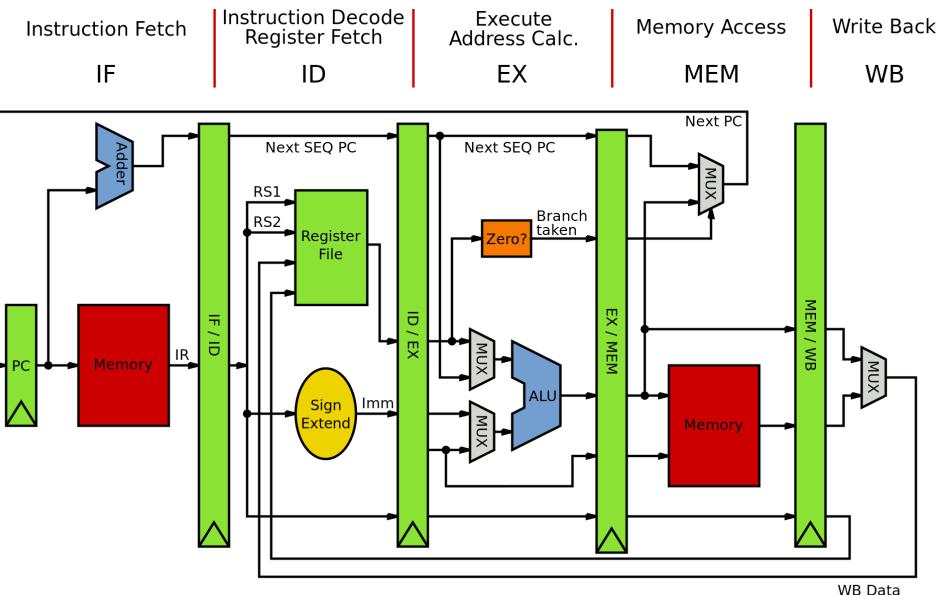
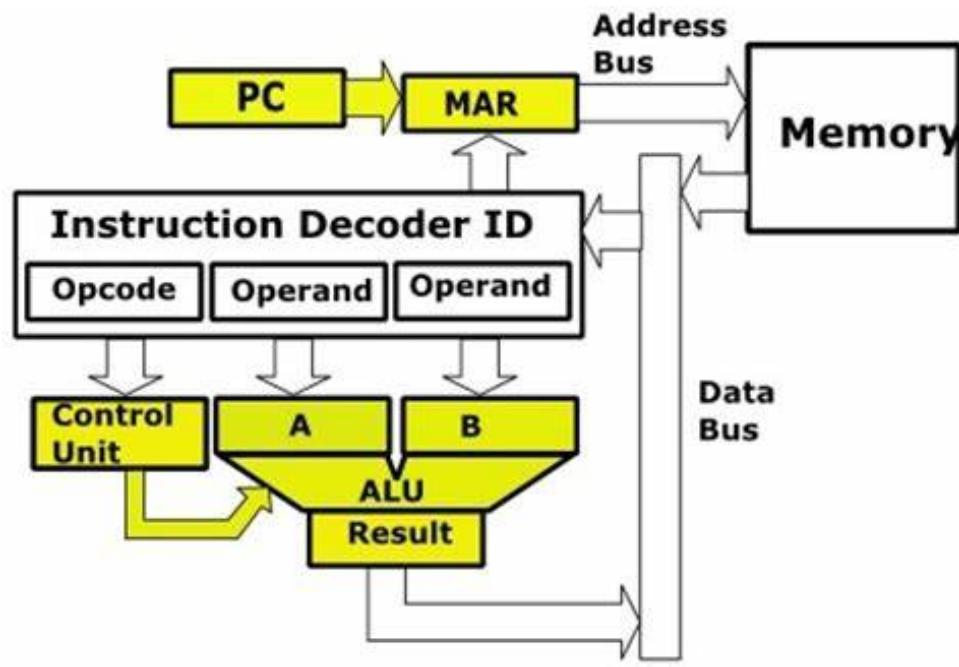


Figure 5.3 The Fetch-Execute Cycle

17

© 2011 Jones and Bartlett Publishers, LLC (www.jbpub.com)



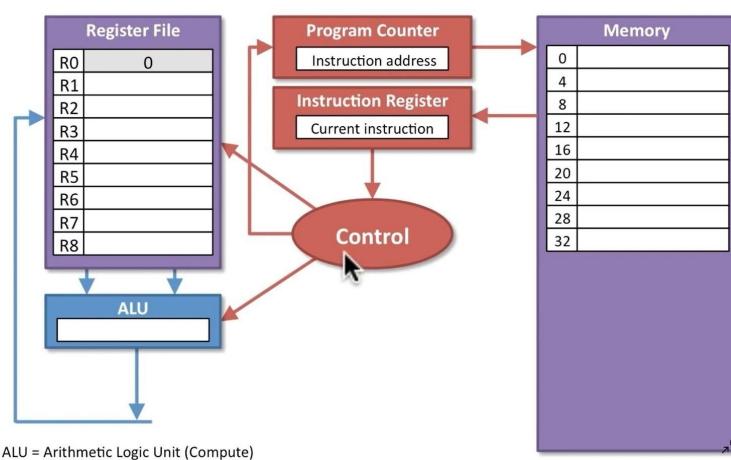


Data operations in detail

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.

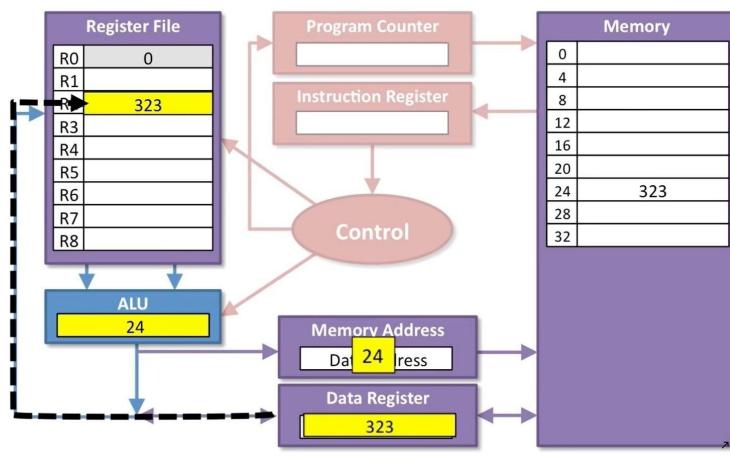


Data transfers in detail

2. Data Transfers

32

1. ALU generates address
2. Address goes to the Memory Address Register
3. Results to/from memory are stored in the Memory Data Register
4. Data from memory can now be stored back into the Register File or to memory can be written.

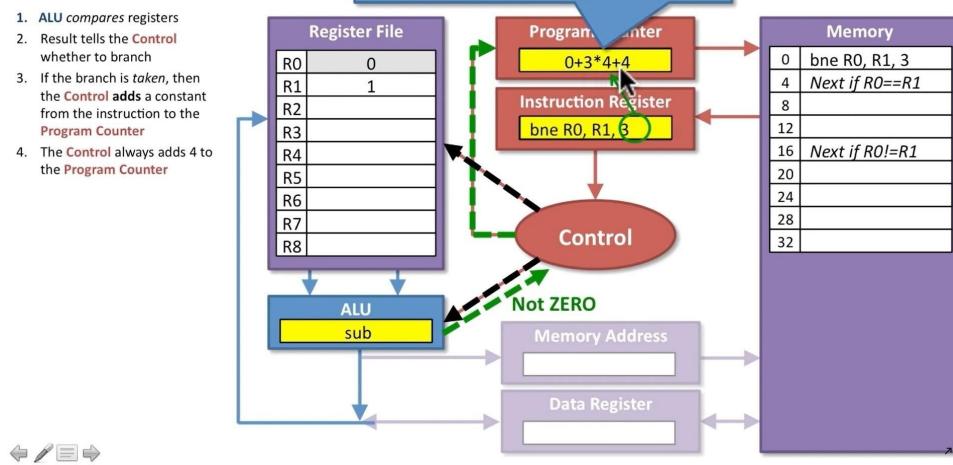


Sequencing in detail

3. Sequencing

46

1. ALU compares registers
2. Result tells the Control whether to branch
3. If the branch is taken, then the Control adds a constant from the instruction to the Program Counter
4. The Control always adds 4 to the Program Counter



Processor - Datapath

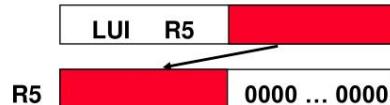
Review: MIPS data transfer instructions

- For all cases, calculate effective address first
 - MIPS doesn't use segmented memory model like x86
 - Flat memory model → EA = address being accessed
- **lb, lh, lw**
 - Get data from addressed memory location
 - Sign extend if **lb** or **lh**, load into **rt**
- **lbu, lhu, lwu**
 - Get data from addressed memory location
 - Zero extend if **lb** or **lh**, load into **rt**
- **sb, sh, sw**
 - Store data from **rt** (partial if **sb** or **sh**) into addressed location

MIPS Data Transfer Instructions

<u>Instruction</u>	<u>Comment</u>
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?

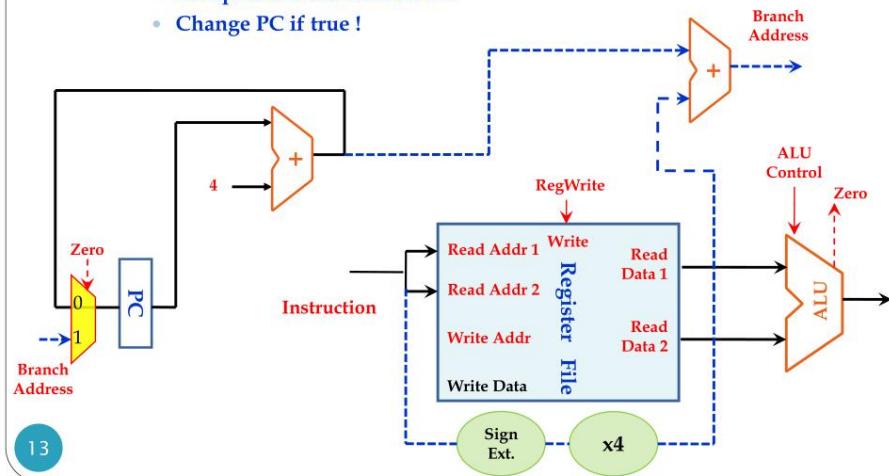


Single-Cycle Datapath

- Execution Datapath

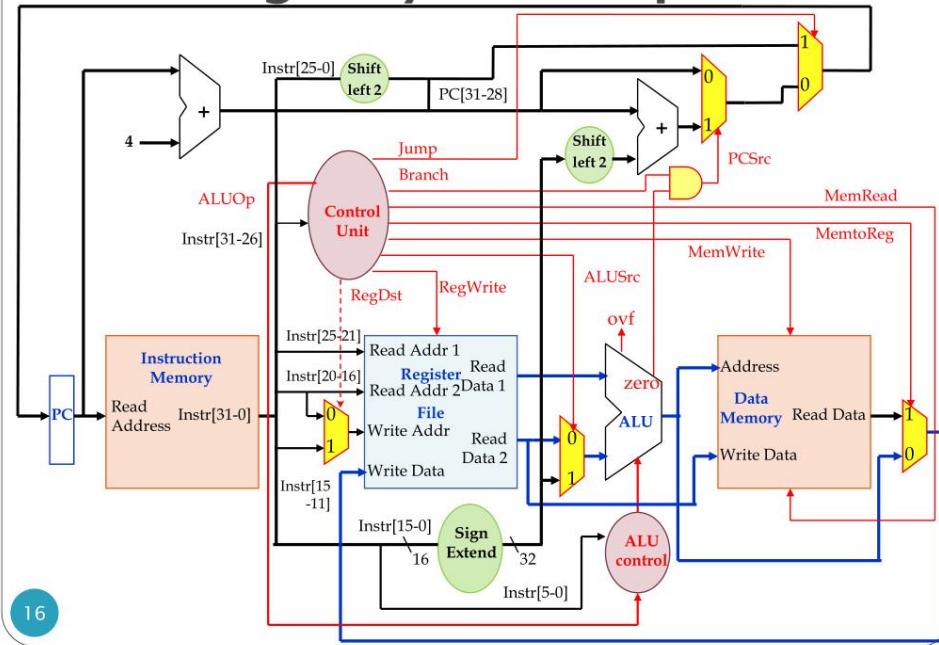
- Branch Instruction

- Compare the two registers
 - Compute the branch address
 - Change PC if true !



13

Single-Cycle Datapath



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)

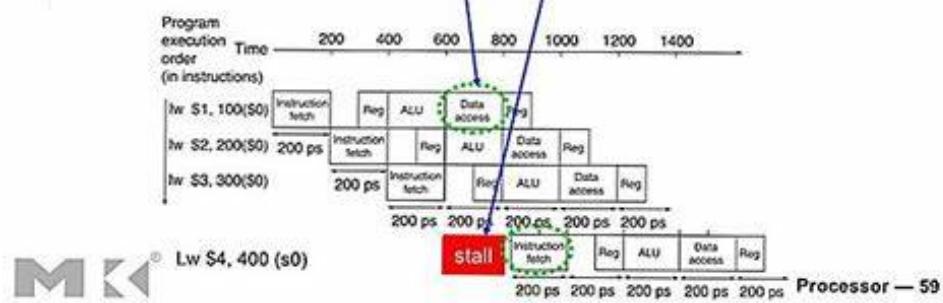
Pipeline Hazards (2)

- 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

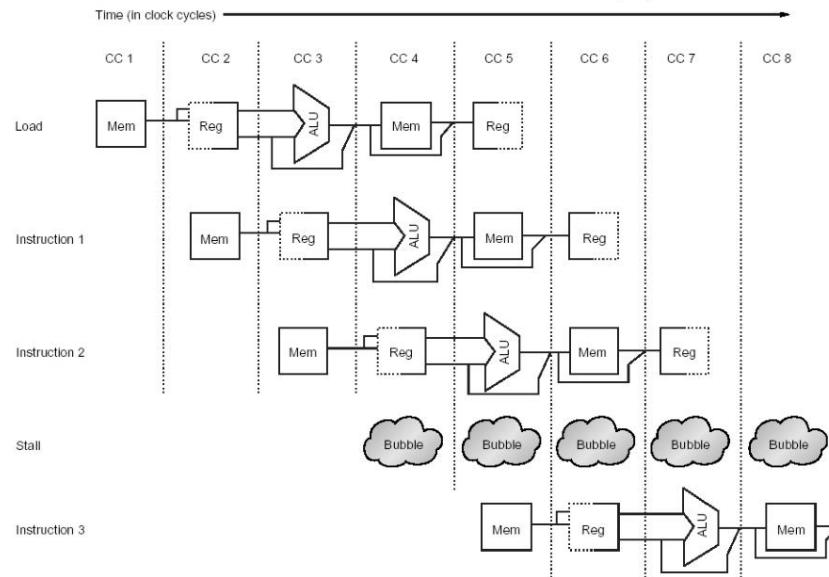
Solution for Pipeline Hazards

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



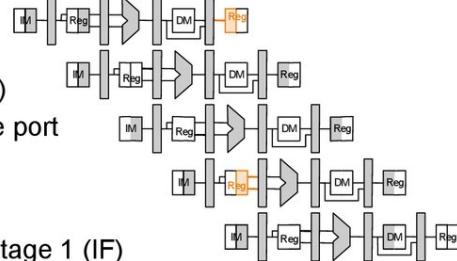
Structural Hazards (3)



- 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



7

Computer Structure 2015 – Pipeline

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

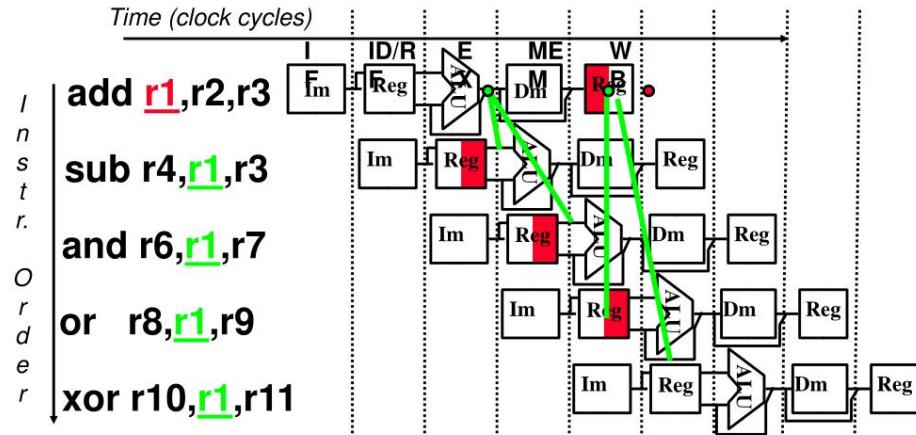
But result of ADD written in R3 at t5

	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



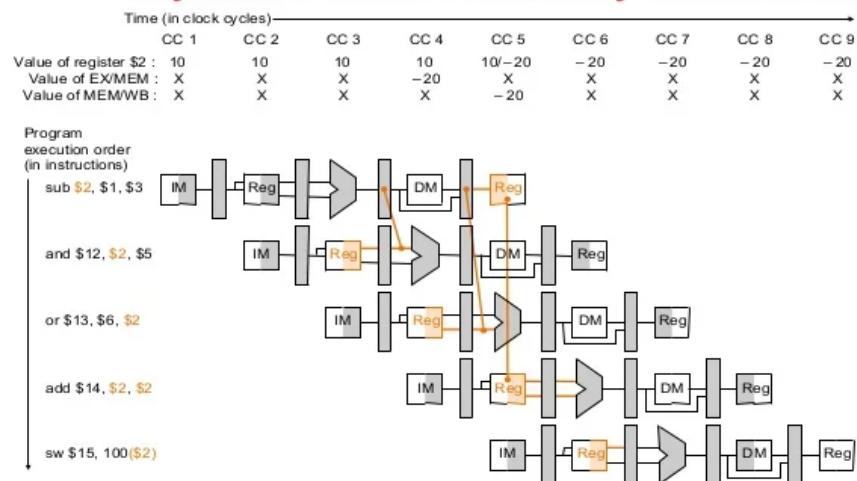
- “or” OK if define read/write properly

cs 152 L1 3.13

DAP Fa97, © U.CB

Data Hazard Solution: Forwarding

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



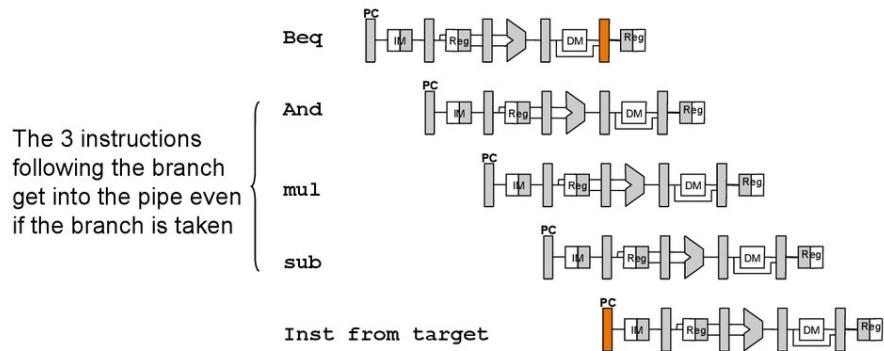
CSCE430/830

Pipeline Hazards

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



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

Control Hazard Review

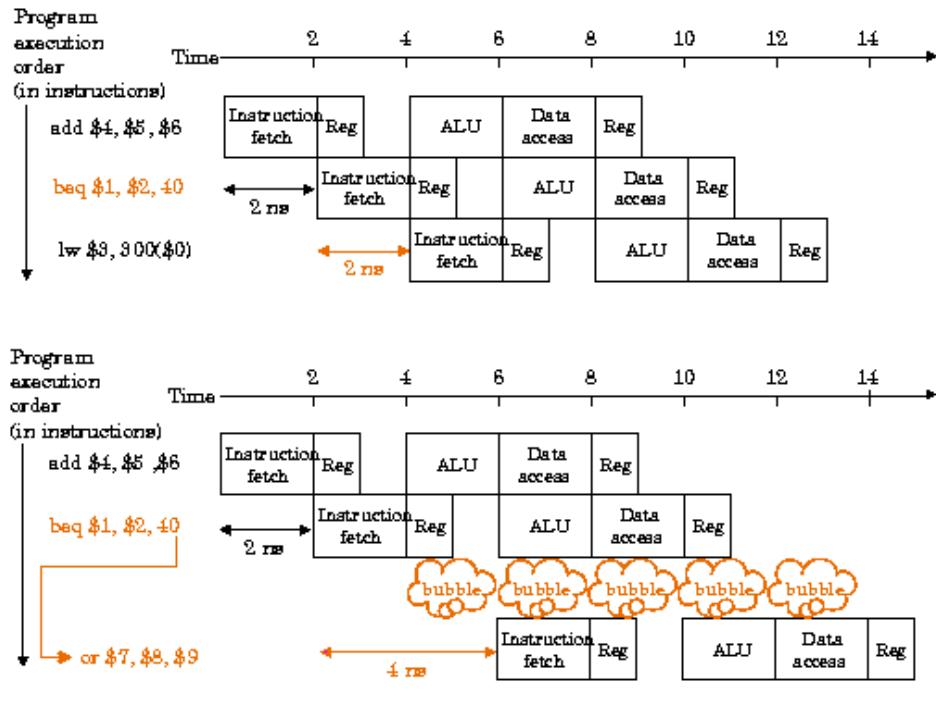
The nub of the problem:

- 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

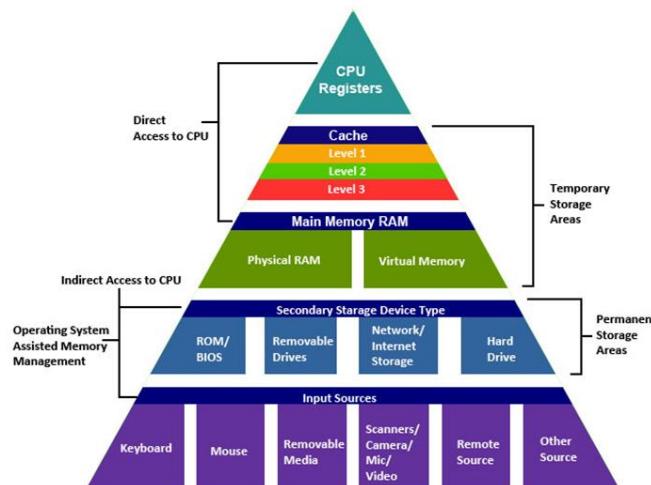
- number of cycles in between * branch frequency = the contribution to CPI due to branches

Predict the branch outcome to avoid stalling



Mermory Hierarchy

The Memory Hierarchy

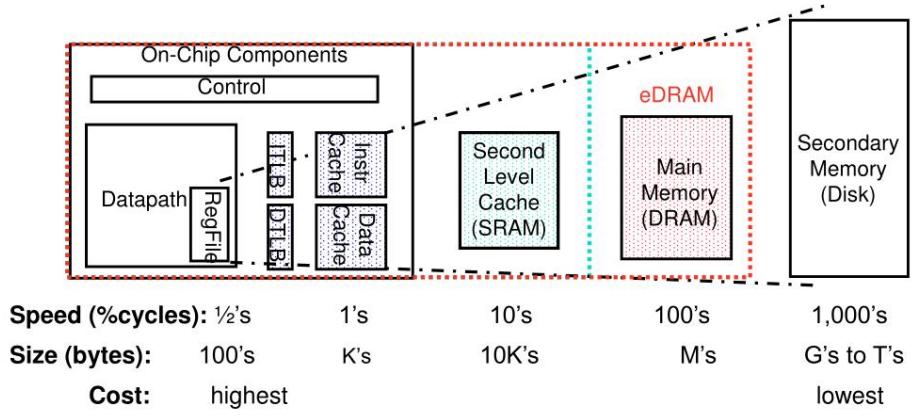


<http://cse1.net/recaps/4-memory.html>

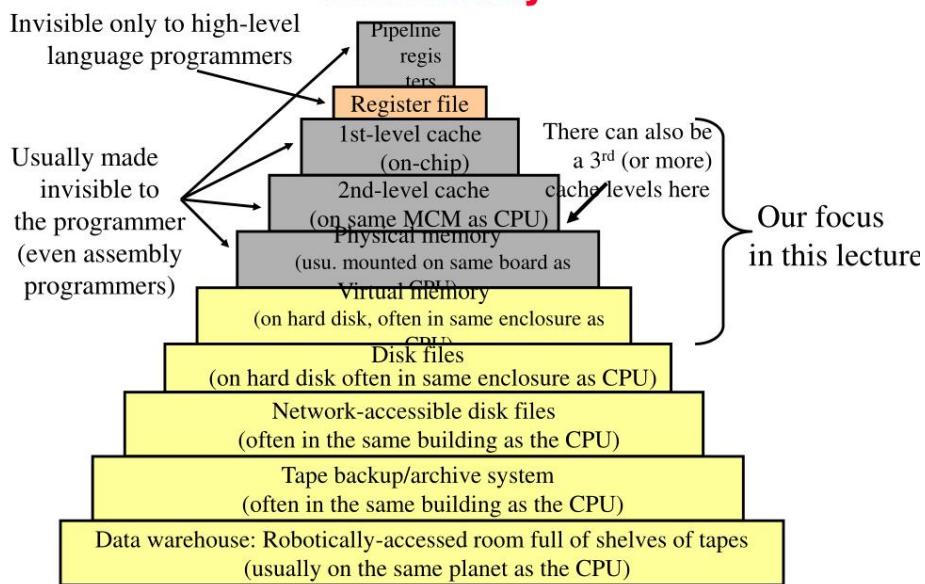
2

A Typical Memory Hierarchy

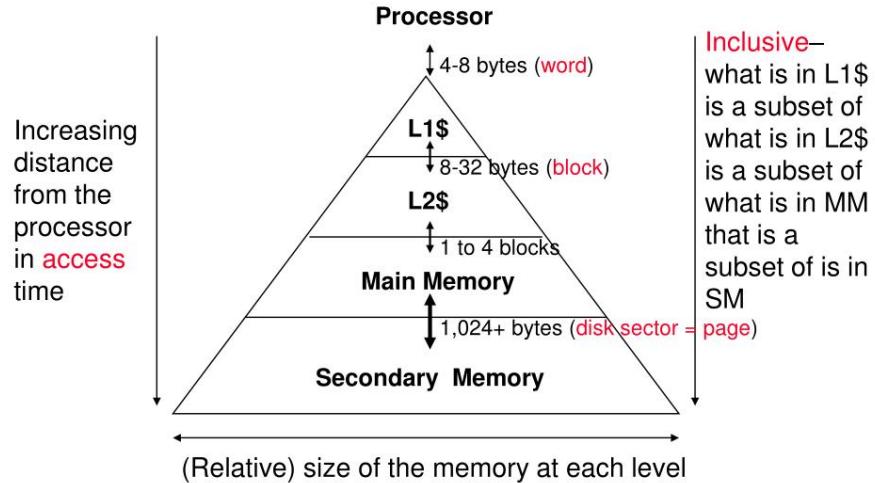
- ❑ By taking advantage of the principle of locality
 - Can present the user with as much memory as is available in the cheapest technology
 - at the speed offered by the fastest technology



Many Levels in Memory Hierarchy



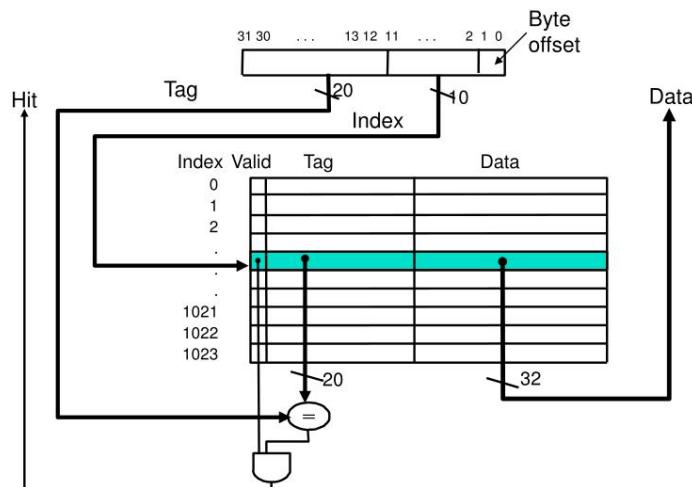
Characteristics of the Memory Hierarchy



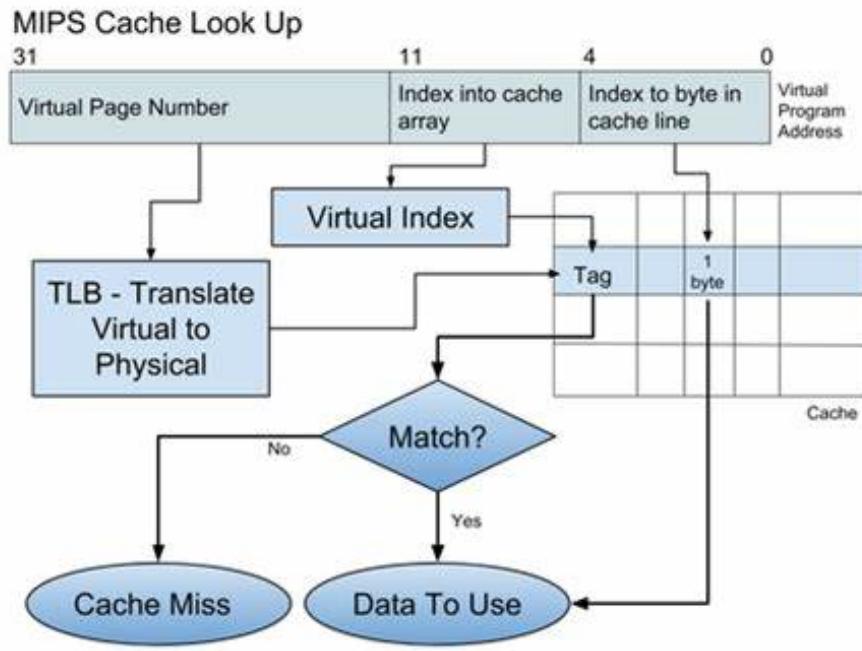
Block Size / Hit Rate / Miss Rate

MIPS Direct Mapped Cache Example

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

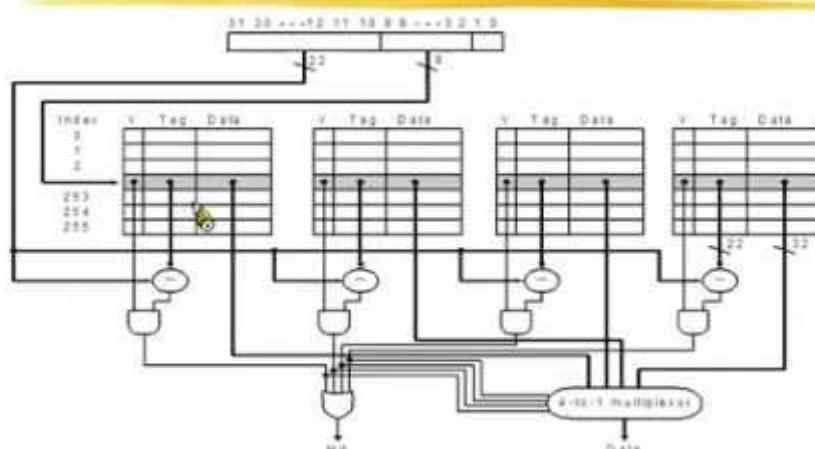


What kind of locality are we taking advantage of?



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



of cache hits

= Hit ratio

(# of cache hits + # of cache misses)

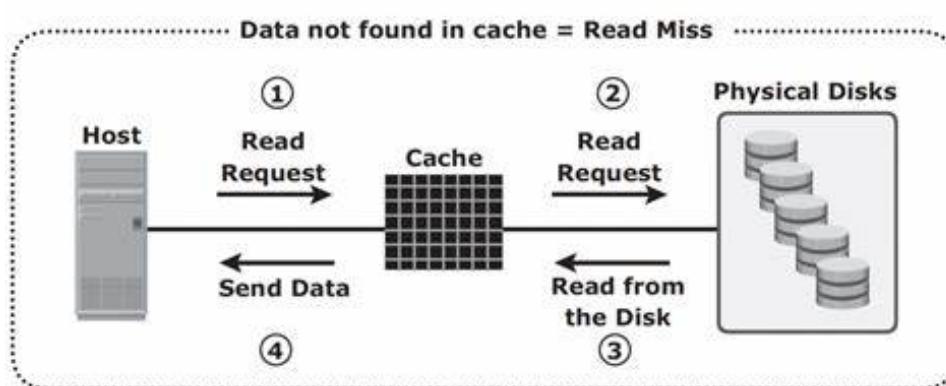
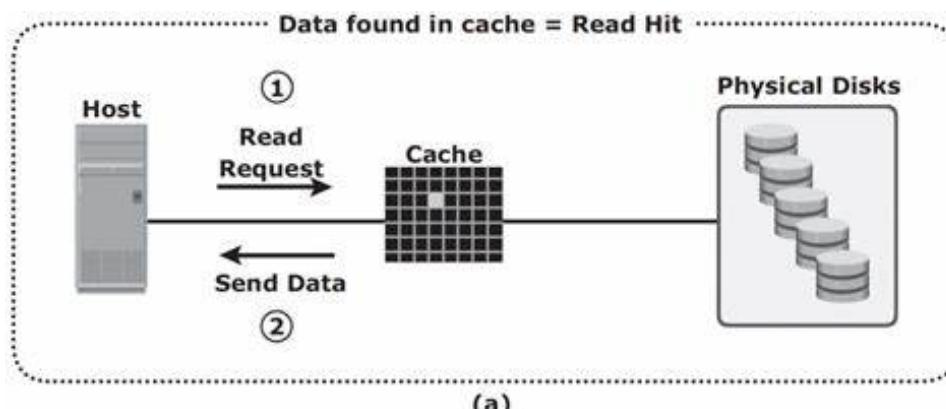
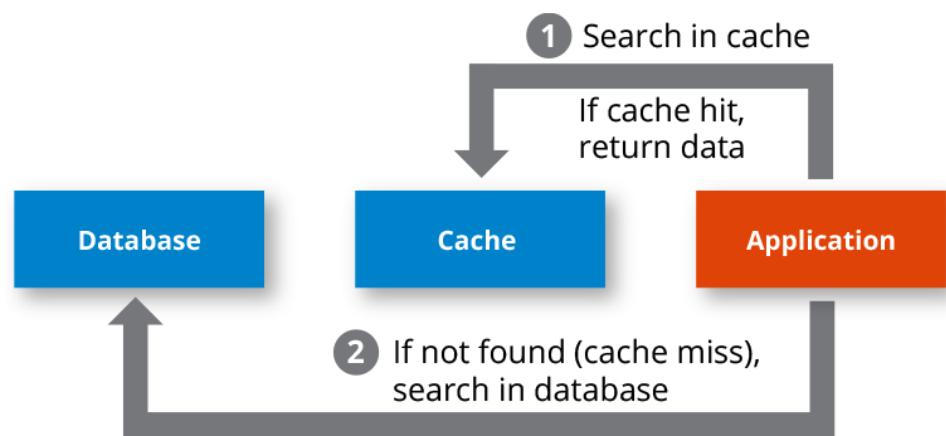
OR

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



Types of Cache Misses: *The Three C's*

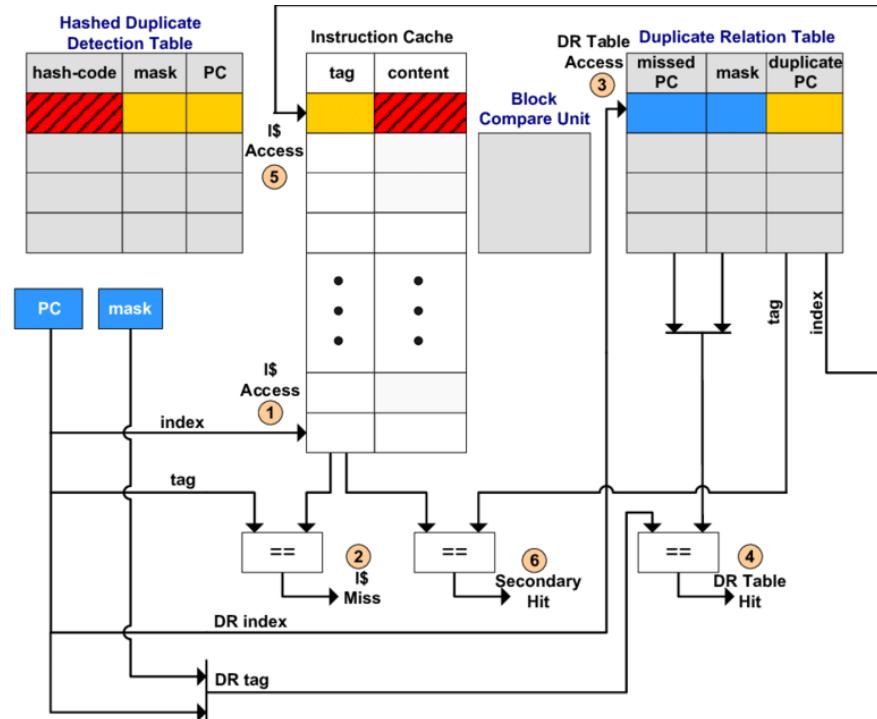
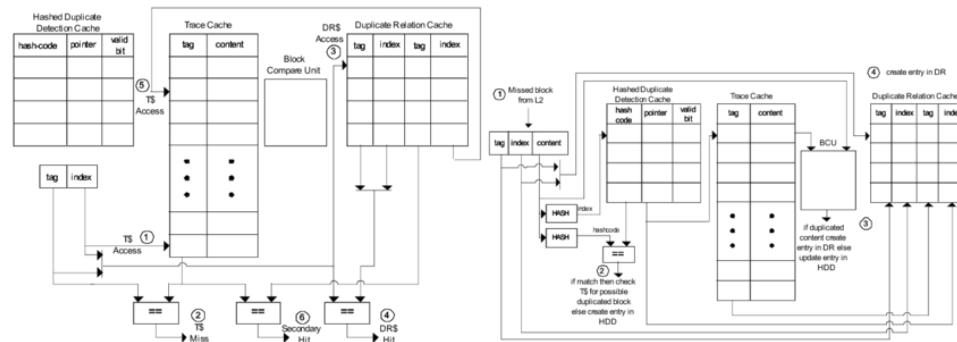
1 Compulsory: On the first access to a block; the block must be brought into the cache; also called cold start misses, or first reference misses.

2 Capacity: Occur because blocks are being discarded from cache because cache cannot contain all blocks needed for program execution (program working set is much larger than cache capacity).

3 Conflict: In the case of set associative or direct mapped block placement strategies, conflict misses occur when several blocks are mapped to the same set or block frame; also called collision misses or interference misses.

EECC551 - Shaaban

#1 Lec #8 Winter 2001 1-30-2002



Cache Performance

Cache Performance Equations

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

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

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

- **CPU time formula:**

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

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

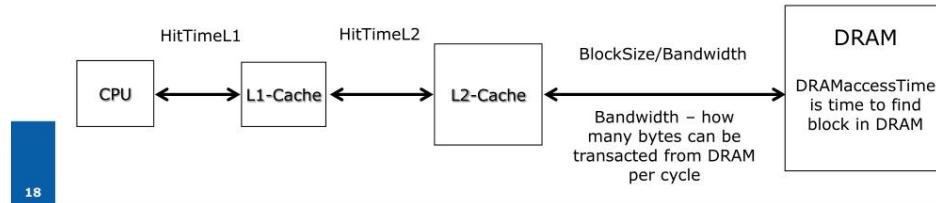
46

Cache Performance

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

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 (L₂)
 - * 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

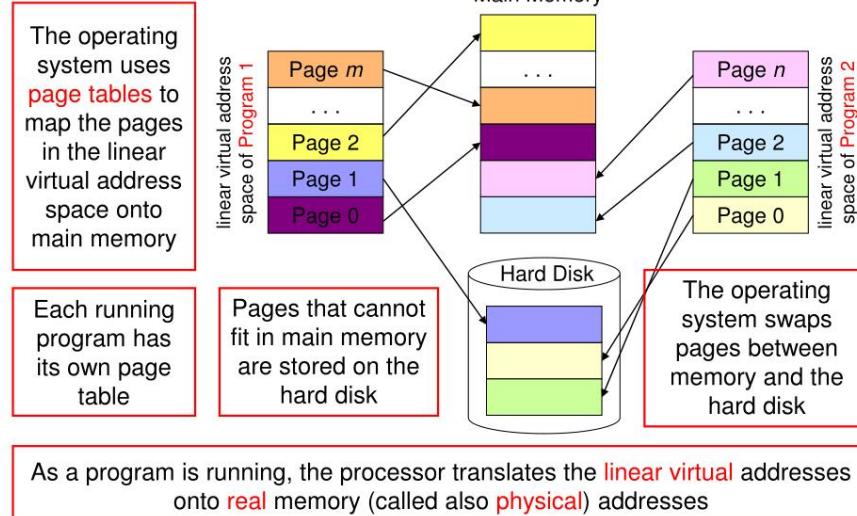
- **Temporal locality hit**
 - Accessing same word that previously accessed
- **Spatial locality hit**
 - Accessing word spatially near previously accessed word

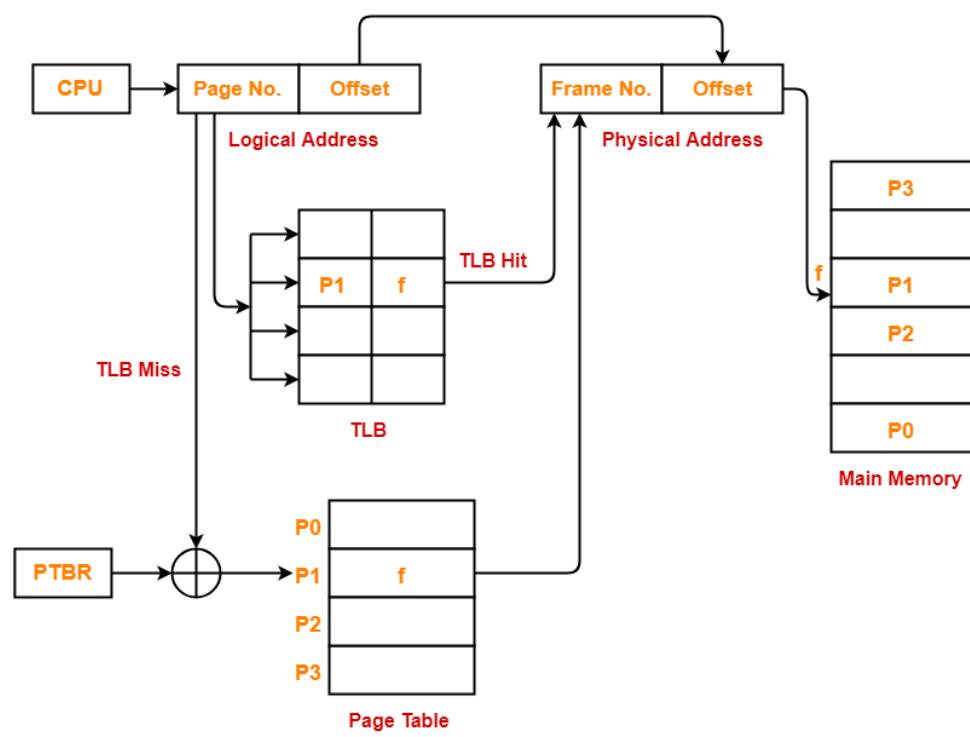
- 42 -

CS 740 F'07

Address Translation Mechanism

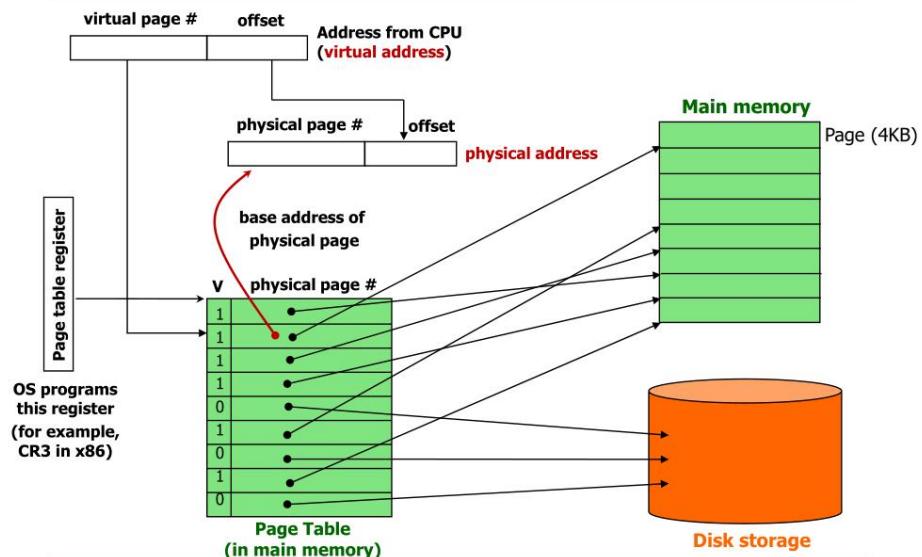
Paging





Translating Logical Address into Physical Address

Address Translation Mechanism



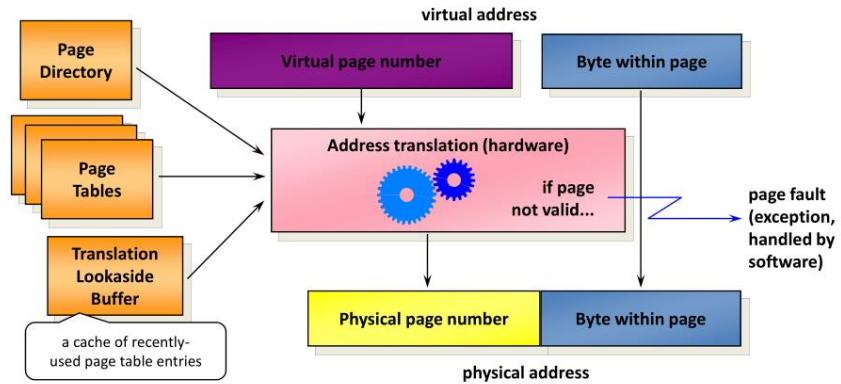
11

Korea Univ

Address Translation with Cache

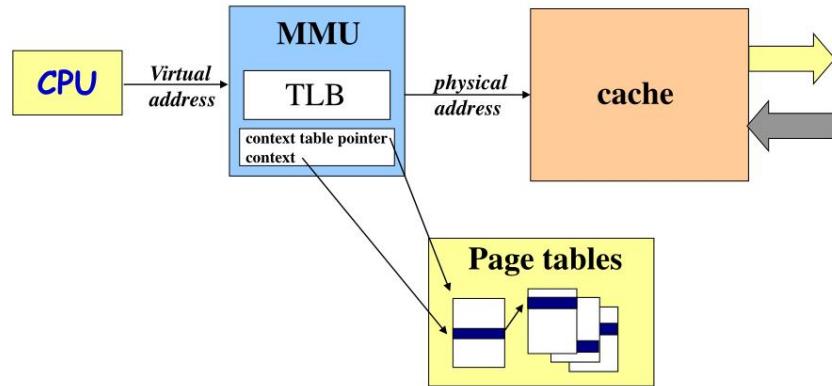
Virtual Address Translation

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

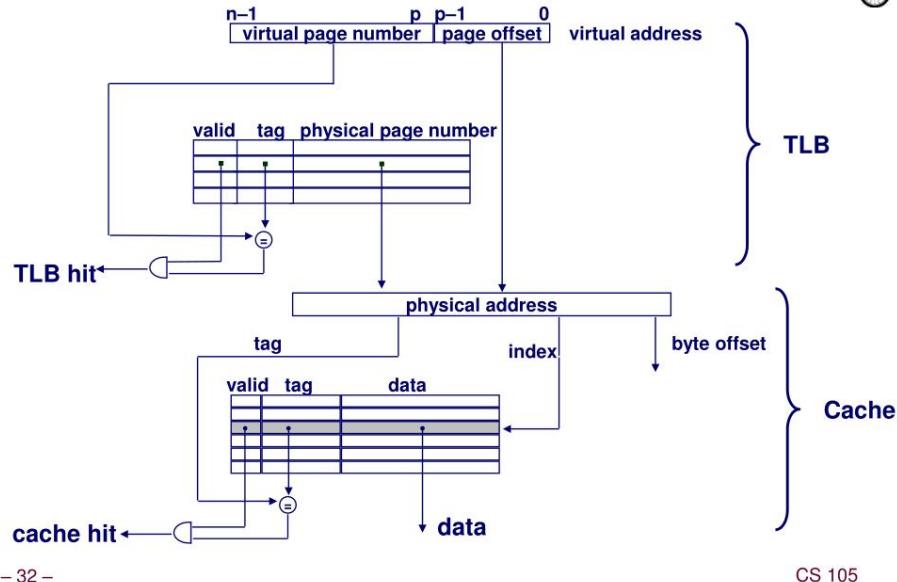


31

Address Translation Overview



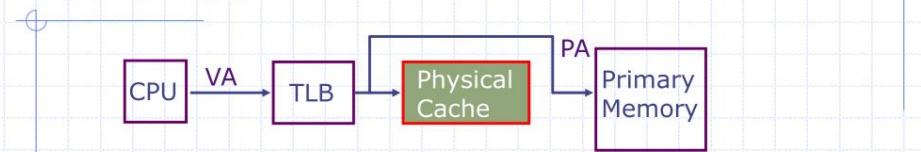
Address Translation With a TLB



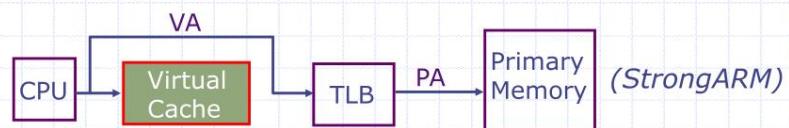
- 32 -

CS 105

Physical or Virtual Address Caches?

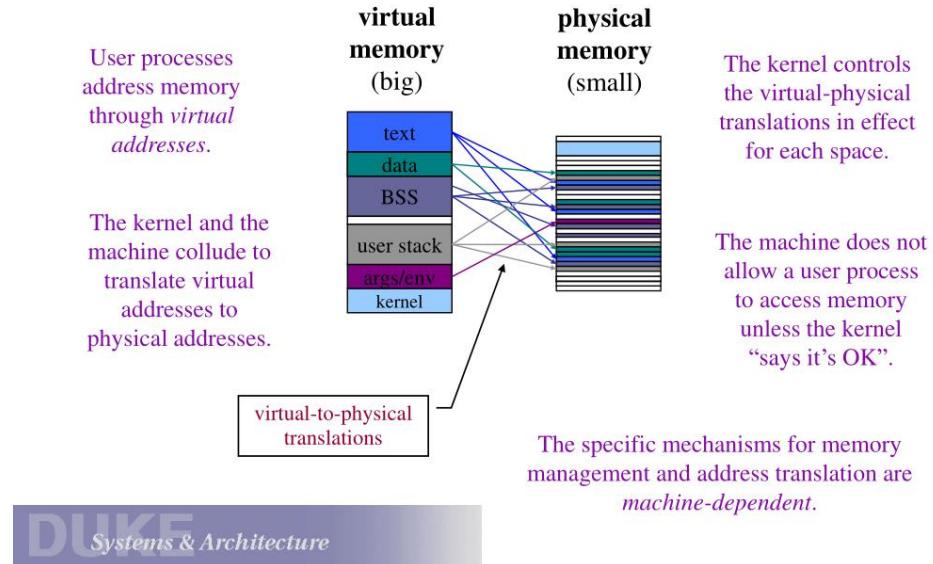


Alternative: place the cache before the TLB

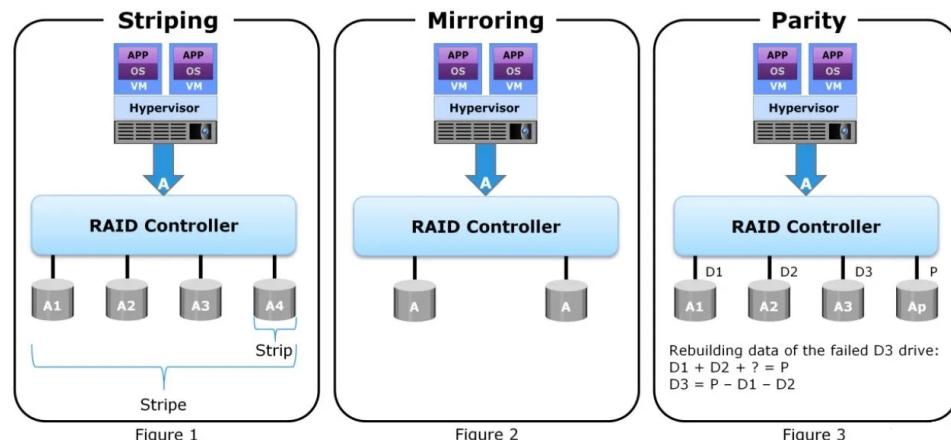
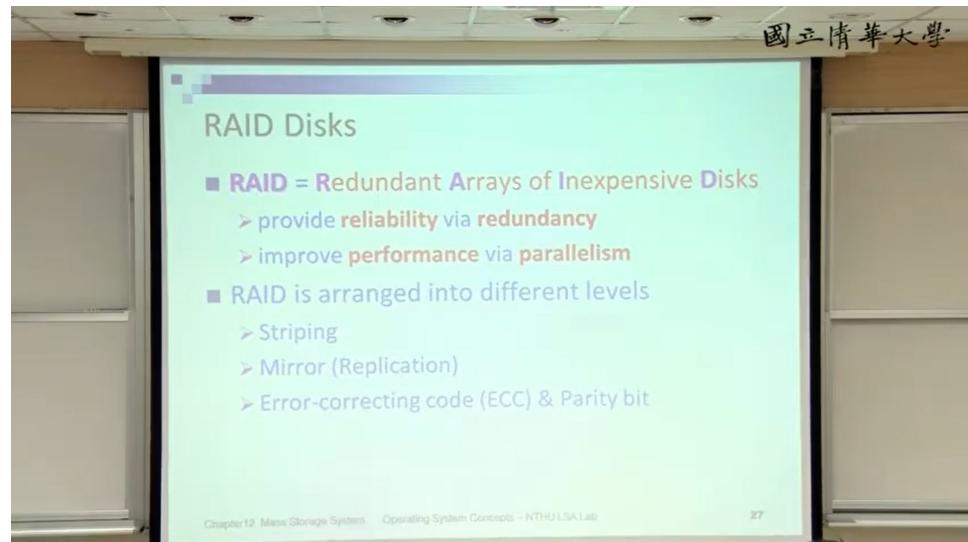


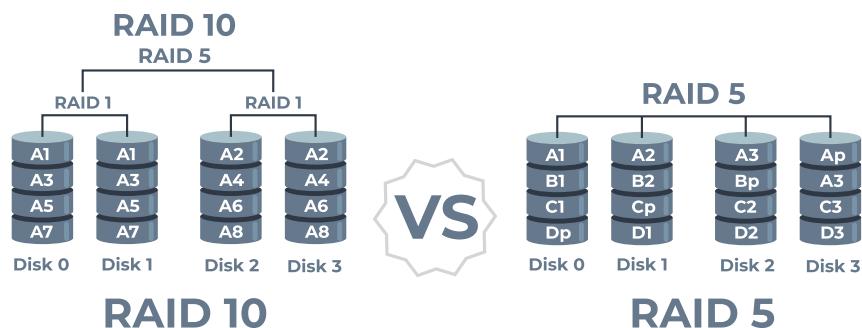
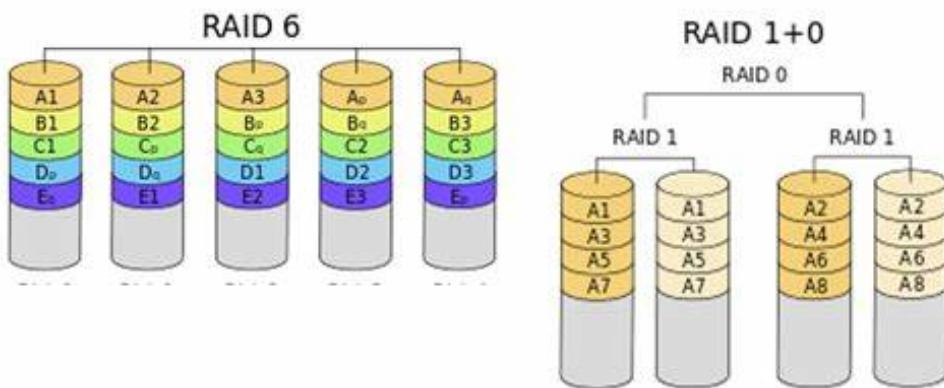
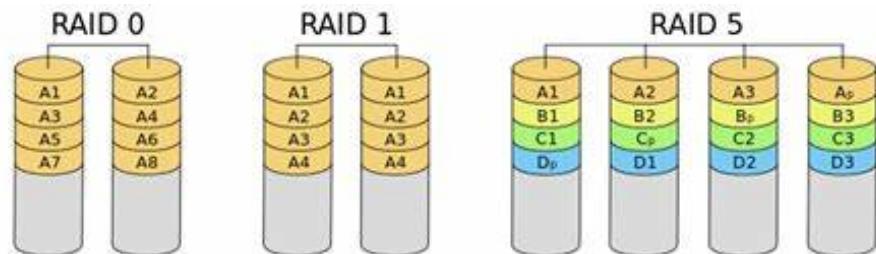
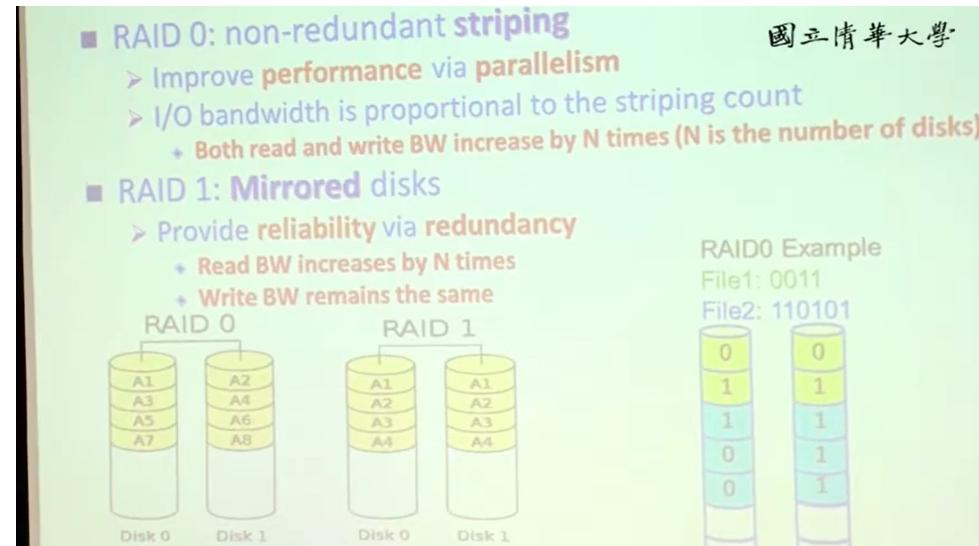
- ◆ one-step process in case of a hit (+)
- ◆ cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (-)
- ◆ aliasing problems due to the sharing of pages (-)

Review: Virtual Addressing



Redundant Array of Inexpensive Disk



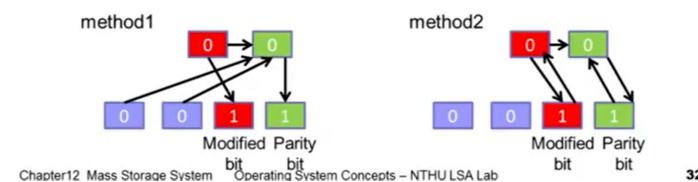




RAID 5: Distributed Parity

國立清華大學

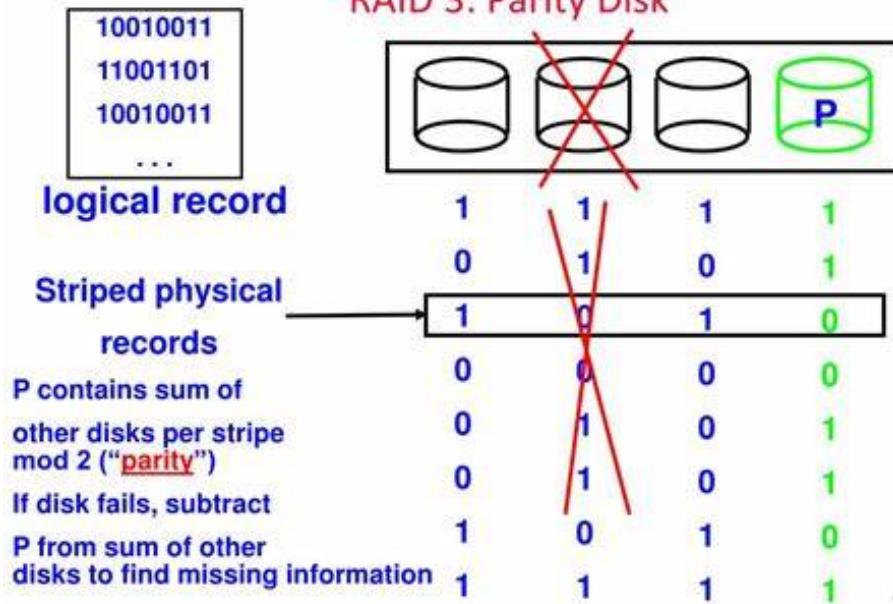
- Read BW increases by N times, because all four disks can serve a read request
- Write BW
 - Method1: (1)read out all unmodified (N-2) data bits. (2) re-compute parity bit. (3) write both modified bit and parity bit to disks.
→ write BW = $N / ((N-2)+2) = 1 \rightarrow$ remains the same
 - Method2: (1)only read the parity bit and modified bit. (2) re-compute parity bit by the difference. (3) write both modified bit and parity bit.
→ write BW = $N / (2+2) = N/4$ times faster



32

Redundant Array of Inexpensive Disks

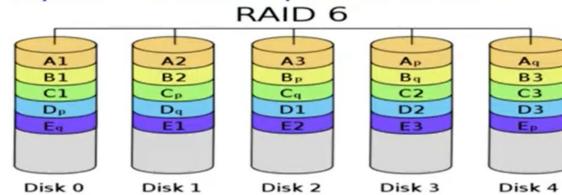
RAID 3: Parity Disk



12.

RAID 6: P+Q Dual Parity Redundancy

- Like RAID 5, but stores extra redundant information to guard against **multiple disk failure**
- Use **ECE code** (i.e. Error Correction Code) instead of single parity bit
- Parity bits are also striped across disks

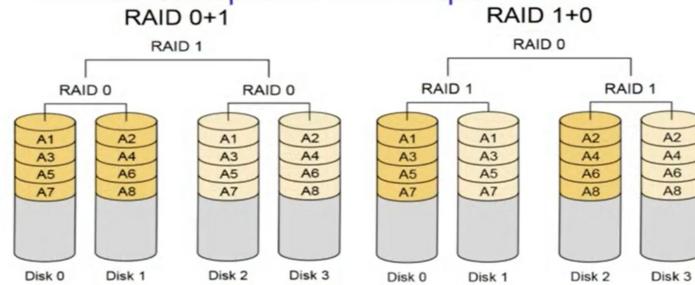


Chapter12 Mass Storage System Operating System Concepts – NTHU LSA Lab

33

Hybrid RAID

- RAID 0+1: Stripe then replicate
- RAID 1+0: Replicate then stripe



*First level often controlled by a controller. Therefore, RAID 10 has better fault tolerance than RAID 01 when multiple disk fails

<http://www.thegeekstuff.com/2011/10/raid10-vs-raid01/>

Chapter12 Mass Storage System Operating System Concepts – NTHU LSA Lab

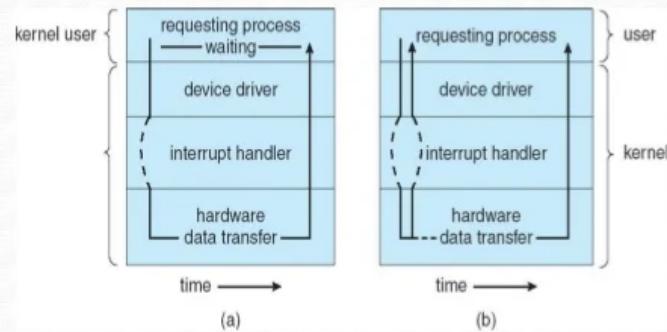
34

IO System Performance

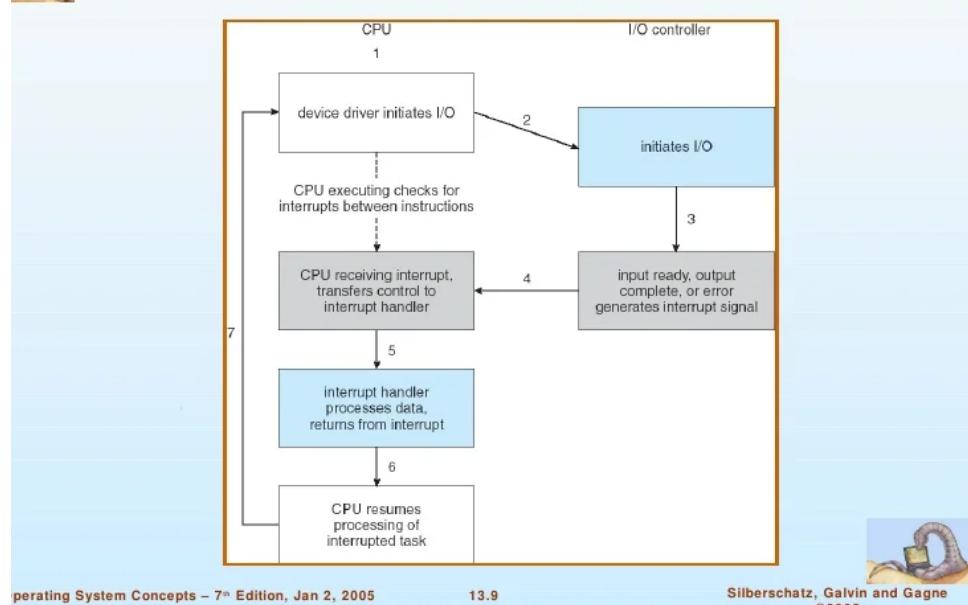
Common Concepts in I/O Hardware

- ▶ **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

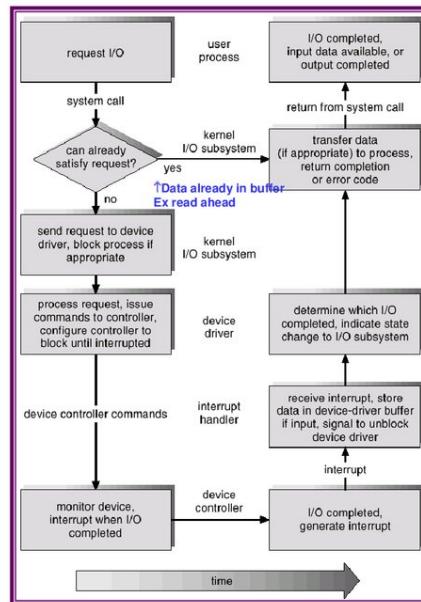
Two I/O Methods



Interrupt-Driven I/O Cycle



Life Cycle of An I/O Request



Operating System Concepts

13.25

Silberschatz, Galvin and Gagne ©2002

I/O Subsystem

國立清華大學

I/O Subsystem

- **I/O Scheduling** – improve system performance by ordering the jobs in I/O queue
 - e.g. disk I/O order scheduling
- **Buffering** – store data in memory while transferring between I/O devices
 - Speed mismatch between devices
 - Devices with different data-transfer sizes
 - Support copy semantics

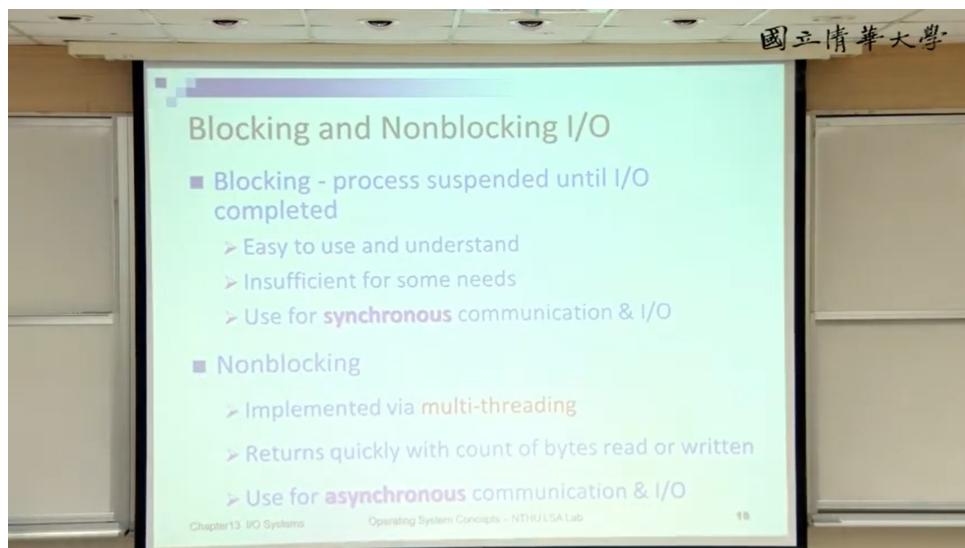
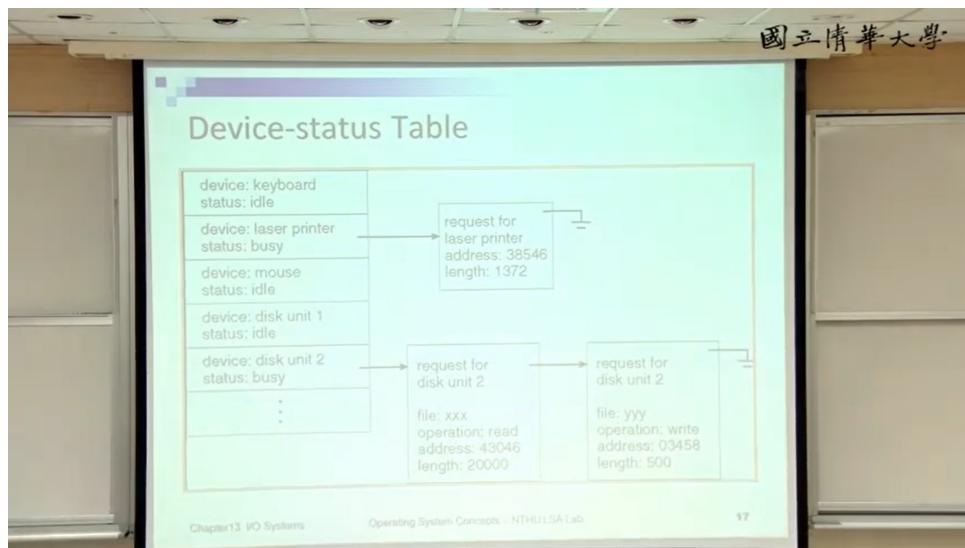
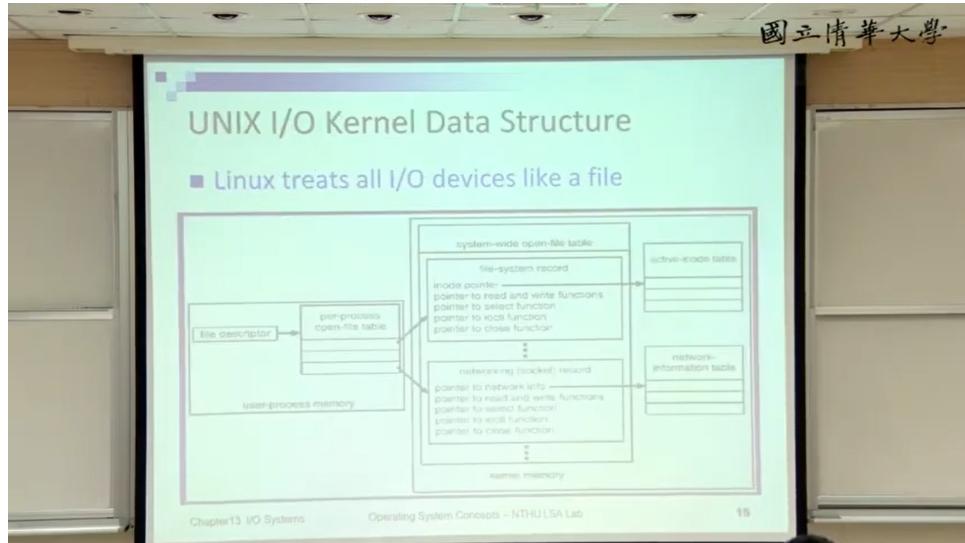
Chapter13. I/O Systems Operating System Concepts - NTHU LSA Lab 14

國立清華大學

I/O Subsystem

- **Caching** – fast memory that holds copies of data
 - Always just a copy
 - Key to performance
- **Spooling** – holds output for a **device**
 - e.g. printing (cannot accept interleaved files)
- **Error handling** – when I/O error happens
 - e.g. SCSI devices returns error information
- **I/O protection**
 - Privileged instructions

Chapter13. I/O Systems Operating System Concepts - NTHU LSA Lab 15



國立清華大學

Performance

- I/O is a major factor in **system** performance
 - It places heavy demands on the CPU to execute **device driver code**
 - The resulting **context switches** stress the CPU and its hardware caches
 - I/O loads down the memory bus during **data copy** between controllers and physical memory, ...
 - **Interrupt handling** is a relatively expensive task
 - Busy-waiting could be more efficient than interrupt-driven if I/O time is small

Chapter13 I/O Systems Operating System Concepts - NTHU LSA Lab 20

國立清華大學

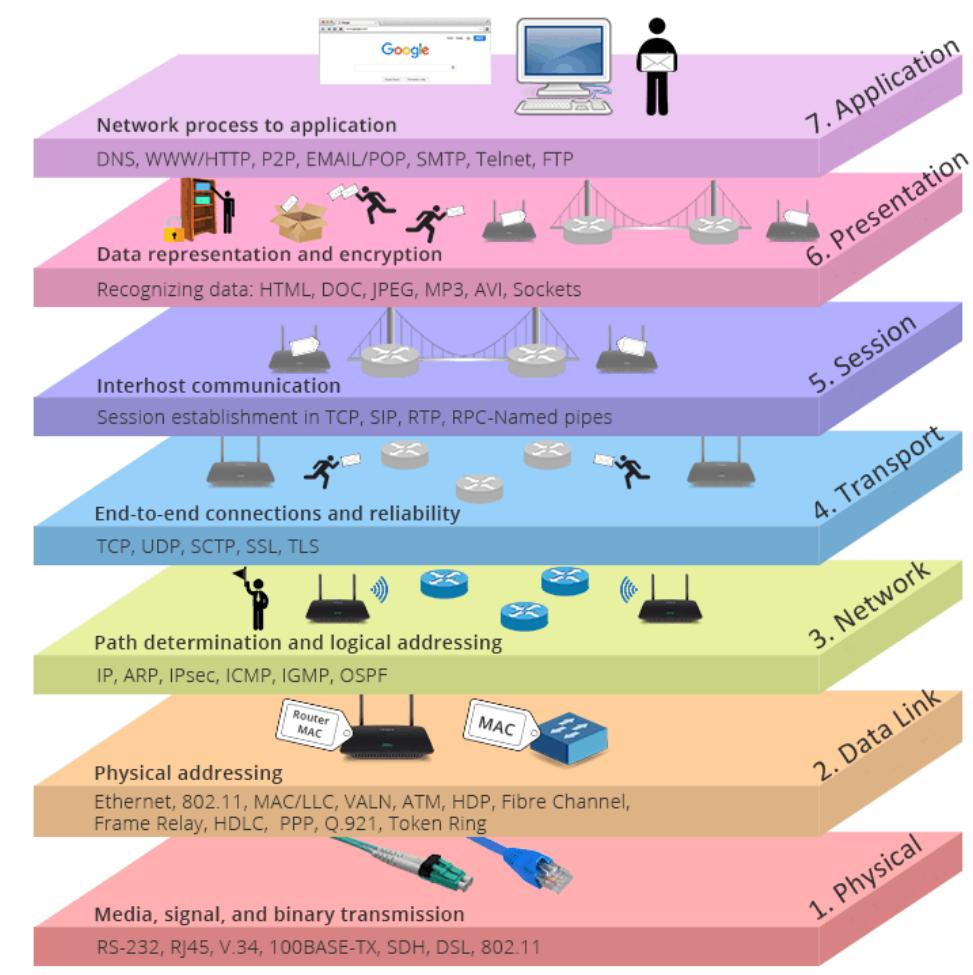
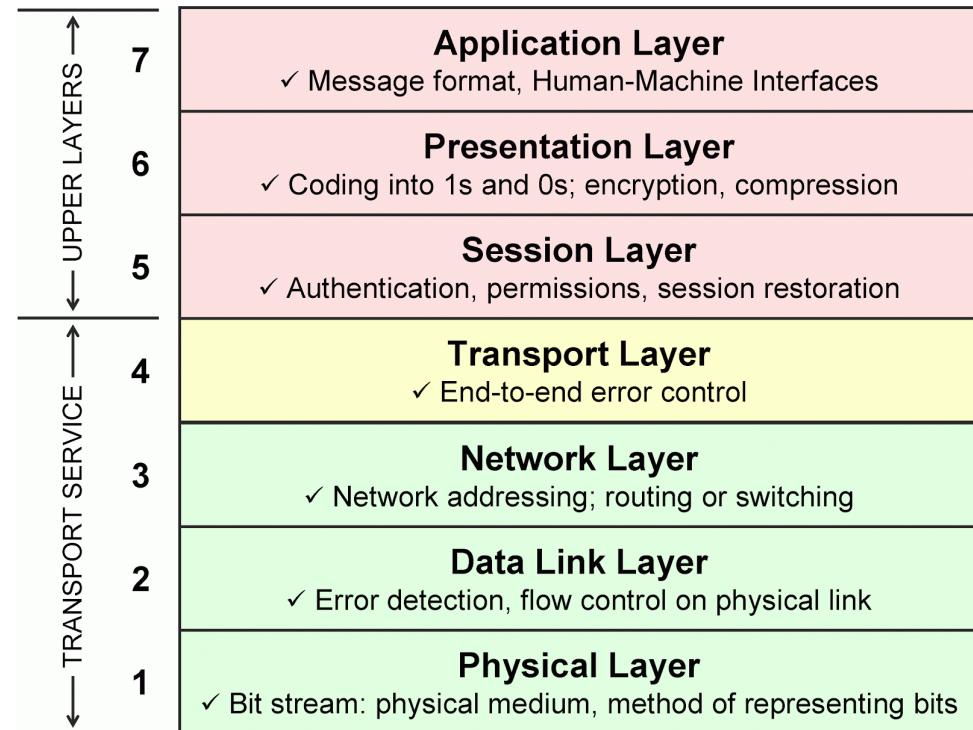
Improving Performance

- Reduce **number of context switches**
- Reduce **data copying**
- Reduce **interrupts** by using large transfers, smart controllers, polling
- Use **DMA**
- Balance CPU, memory, bus, and I/O performance for highest throughput

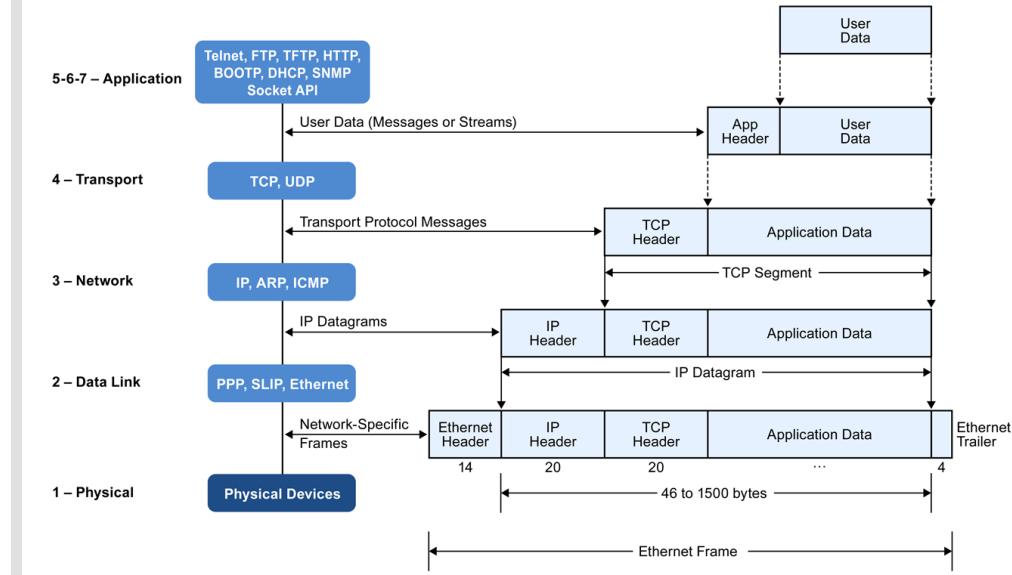
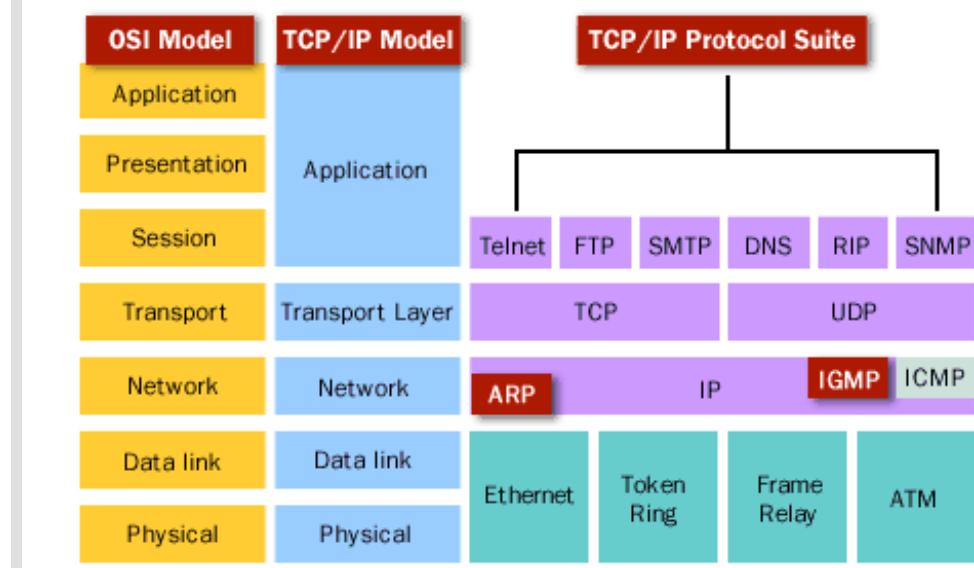
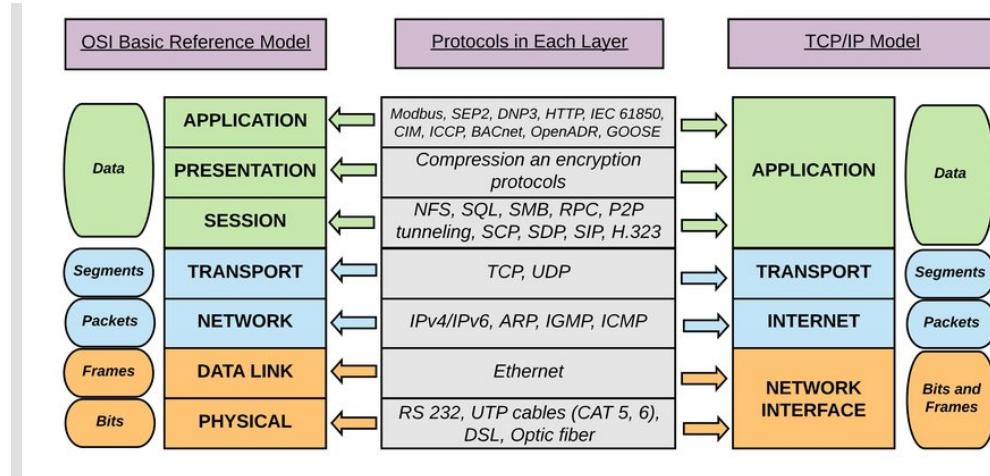
Chapter13 I/O Systems Operating System Concepts - NTHU LSA Lab 21

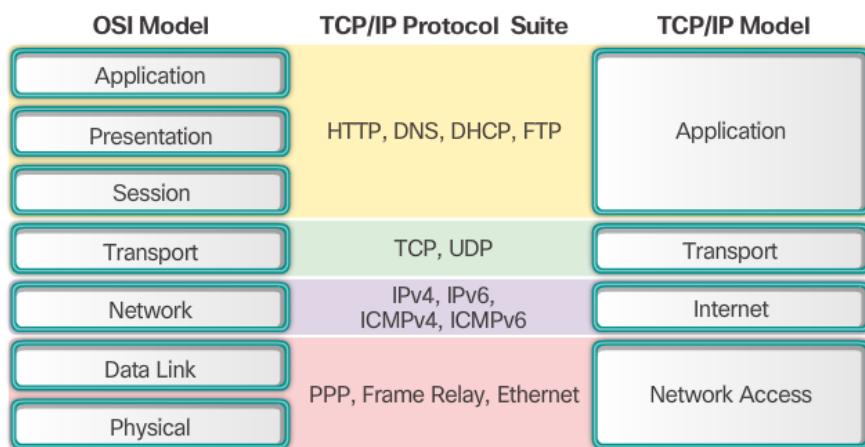
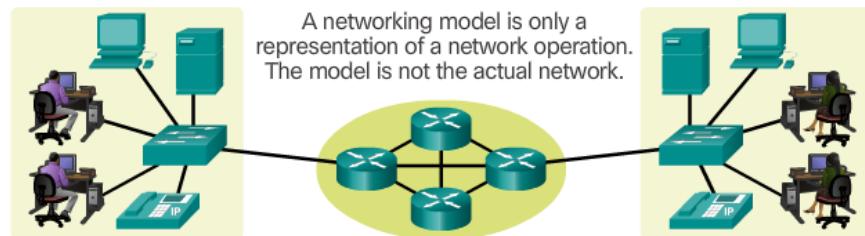
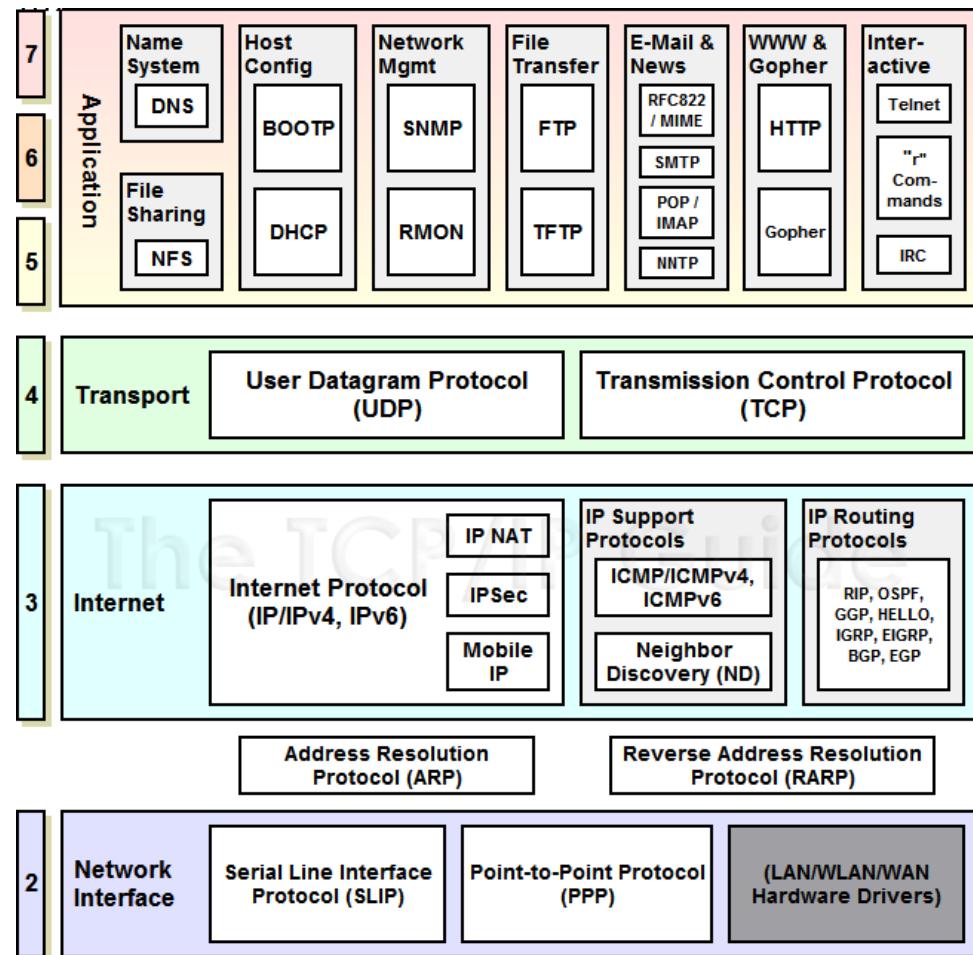
Network

Network OSI Model



TPC/IP & Protocol

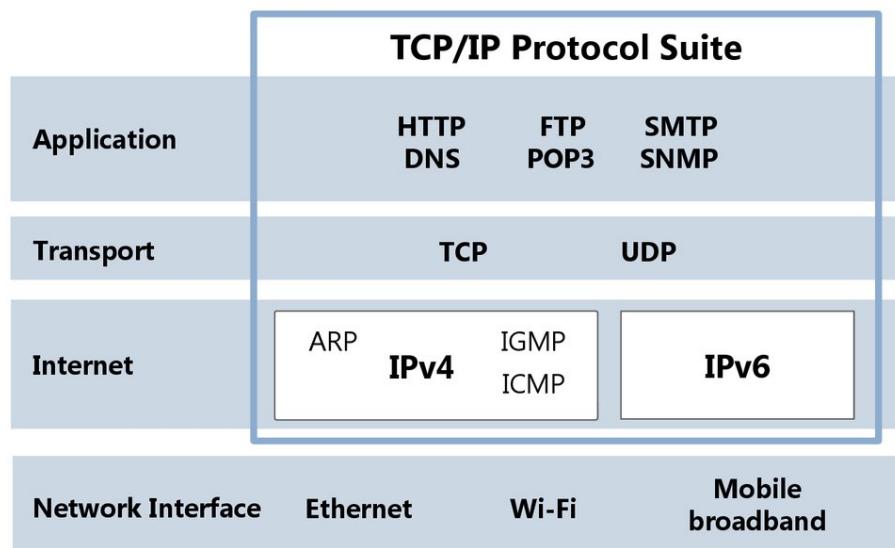




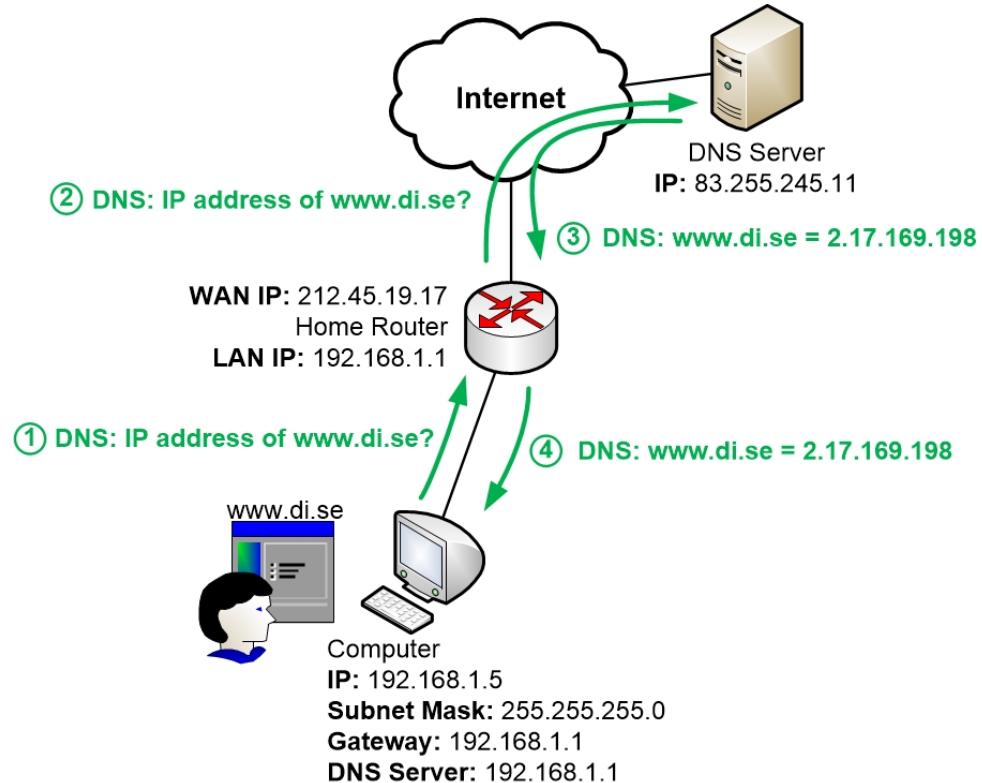
Application	File Transfer	Web Browser	Email	Remote Login	Name Resolution	IP Address
Presentation	FTP TFTP	HTTP	SMTP IMAP POP3	Telnet	DNS	DHCP
Session						
Transport	Transmission Control Protocol TCP				User Datagram Protocol UDP	
Network	Internet Protocol IP				ARP ICMP	
Data Link	Ethernet	Token Ring		FDDI	WAN Protocols	

Computer Network Basic

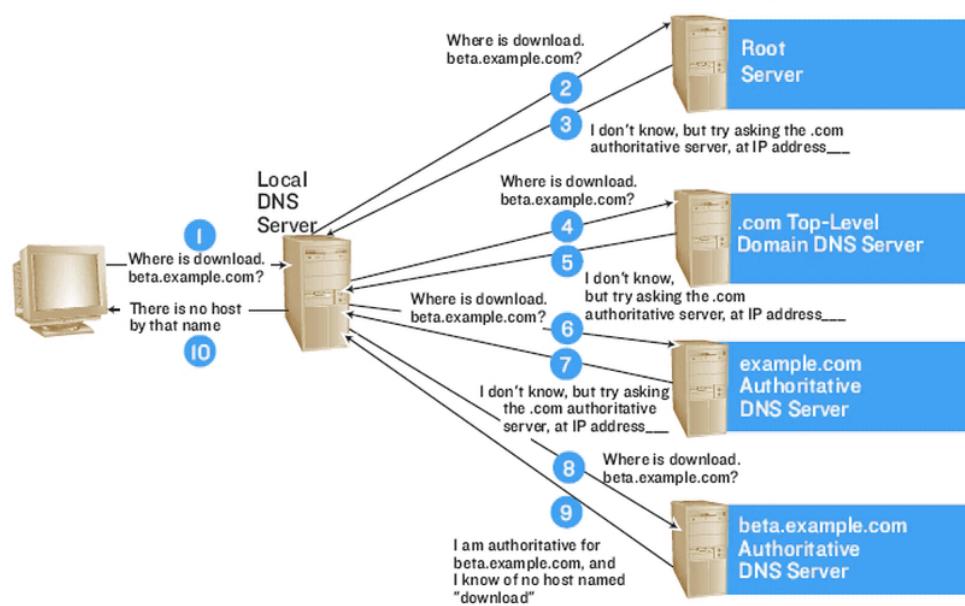
The TCP/IP Protocol Suite

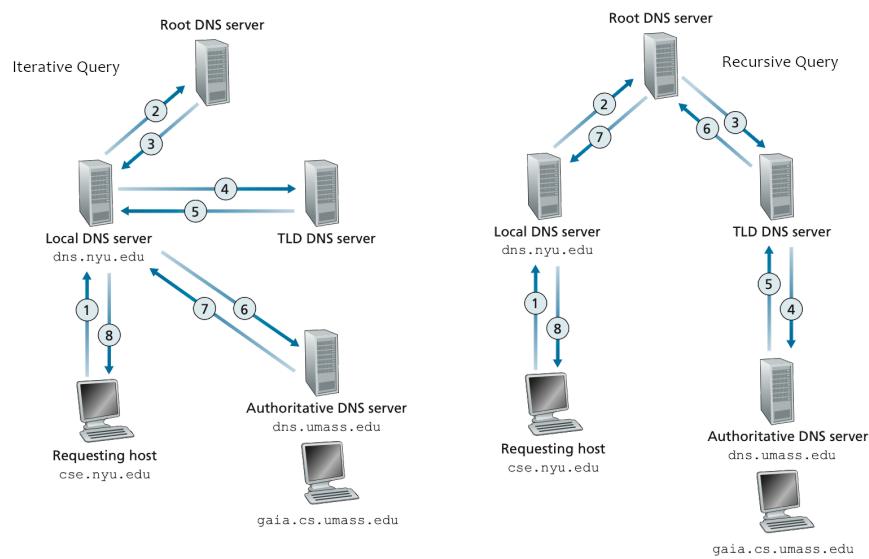
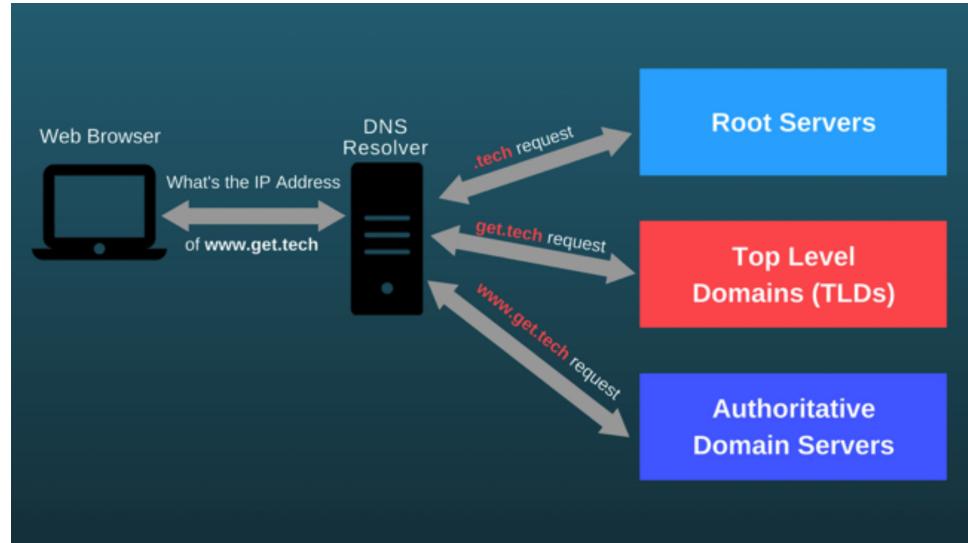


DNS(Domain Name System)

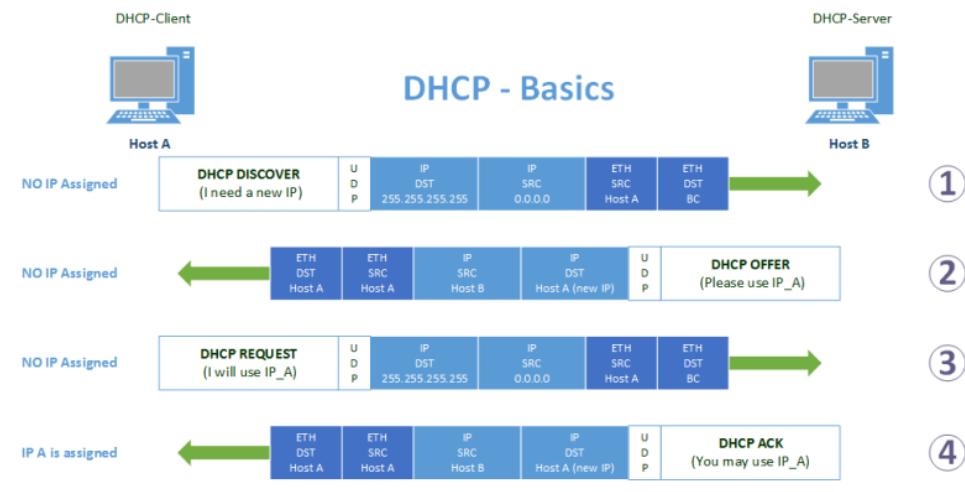
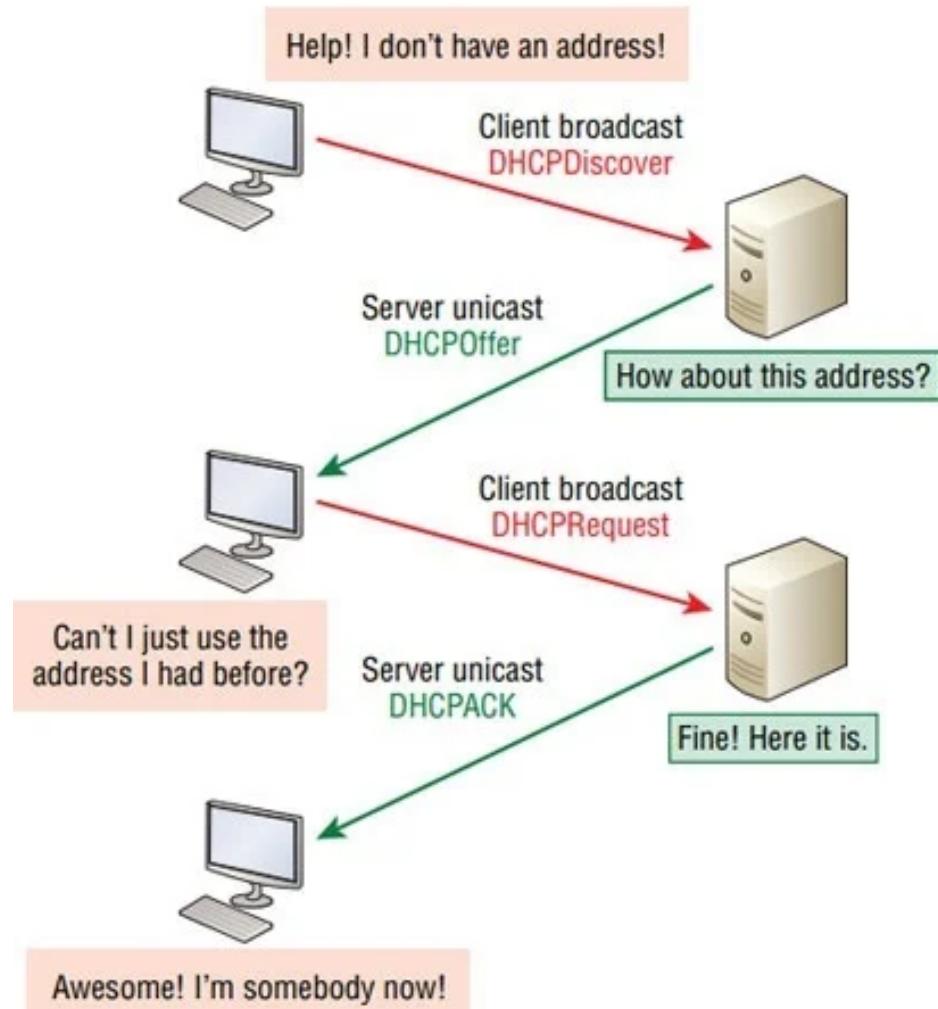


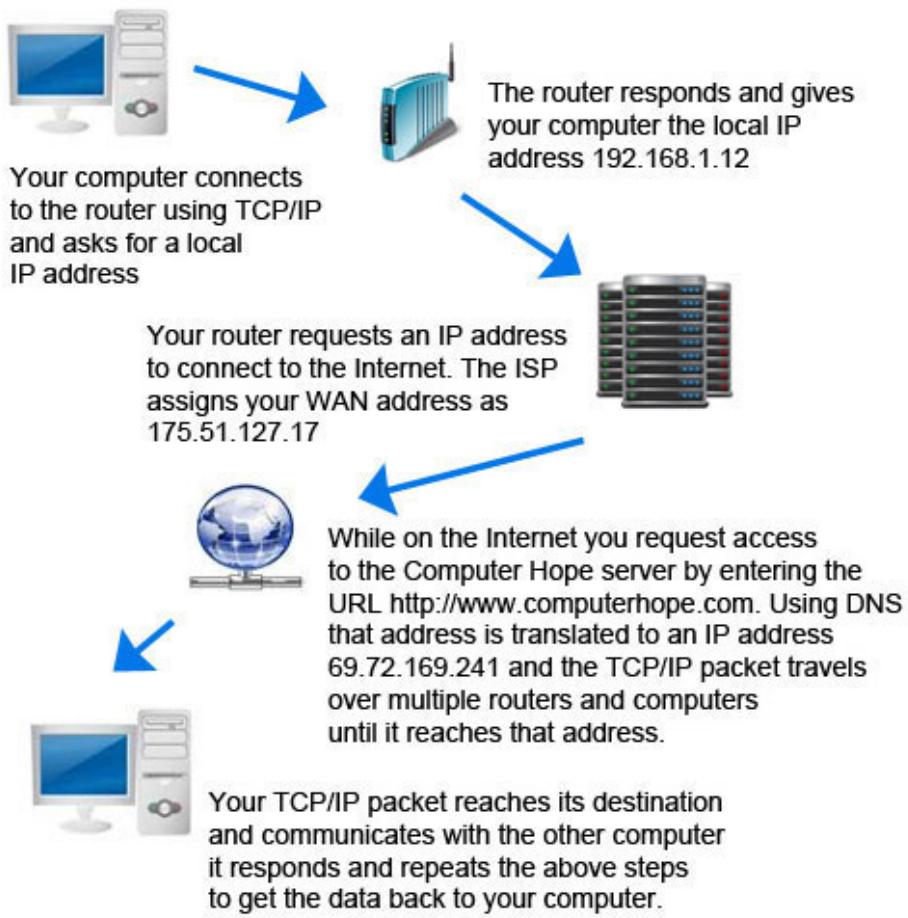
HOW DNS WORKS





DHCP





ComputerHope.com

APIPA

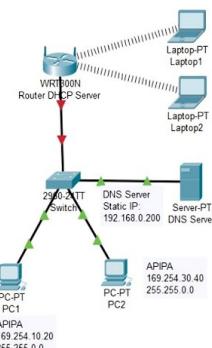
MANA NETWORK
All about EDUCATION

APIPA Class B Private IP v4 Address

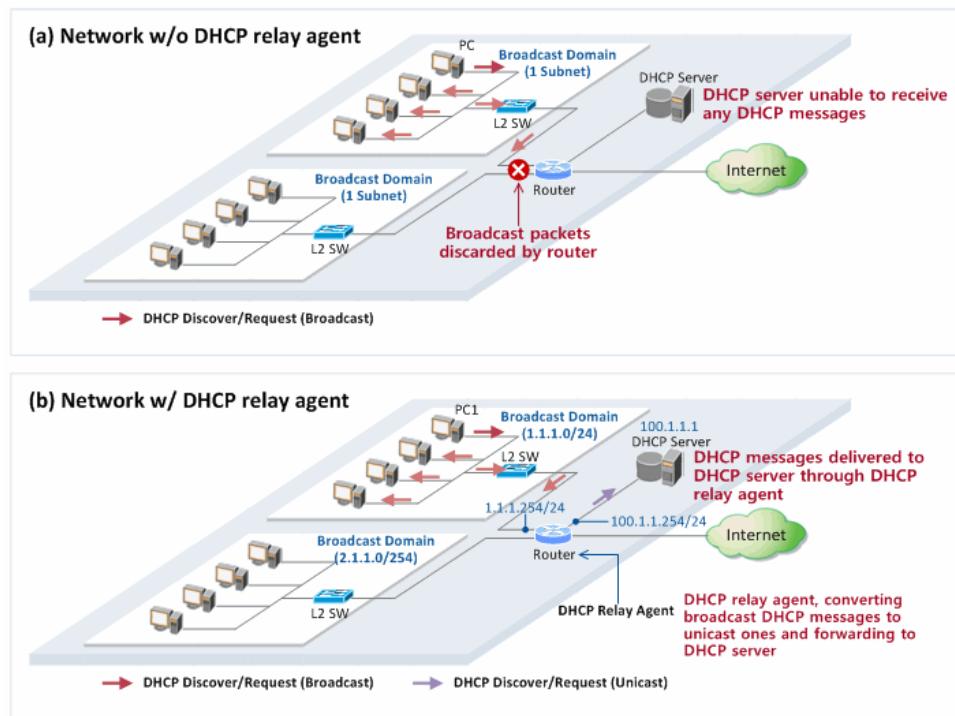
❖ Automatic Private IP Address

169.254. x. x

- ❑ When connection between **DHCP Server & N/W device (Switch)** goes **DOWN**, **APIPA** addresses are **AUTOMATICALLY** created on **END User Devices** like Desktops, PCs, Laptops, Printers etc.
- ❑ **END User Devices** who have **APIPA** addresses can **ONLY** communicate **INSIDE** the own **LOCAL N/W**
- ❑ **APIPA** addresses **DO NOT** go out of their **OWN N/W**
- ❑ **APIPA** addresses are **NOT ROUTABLE**
- ❑ If **APIPA** addresses are seen on **END Devices** than this is a **INTERNAL N/W** problem
- ❑ Check the **MEDIA or CABLE** between **DHCP server (Router) & N/W device (Switch)** inside **LOCAL N/W**



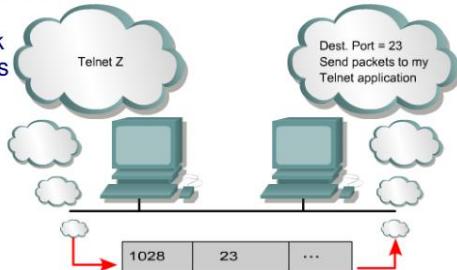
DHCP Relay = IP Helper



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	Legend		
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			

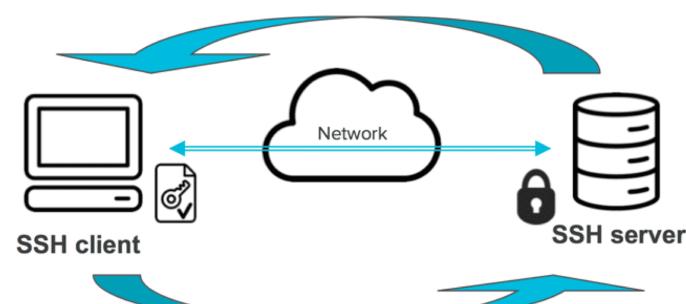
IANA port assignments published at <http://www.iana.org/assignments/port-numbers>

Transport Layer Ports

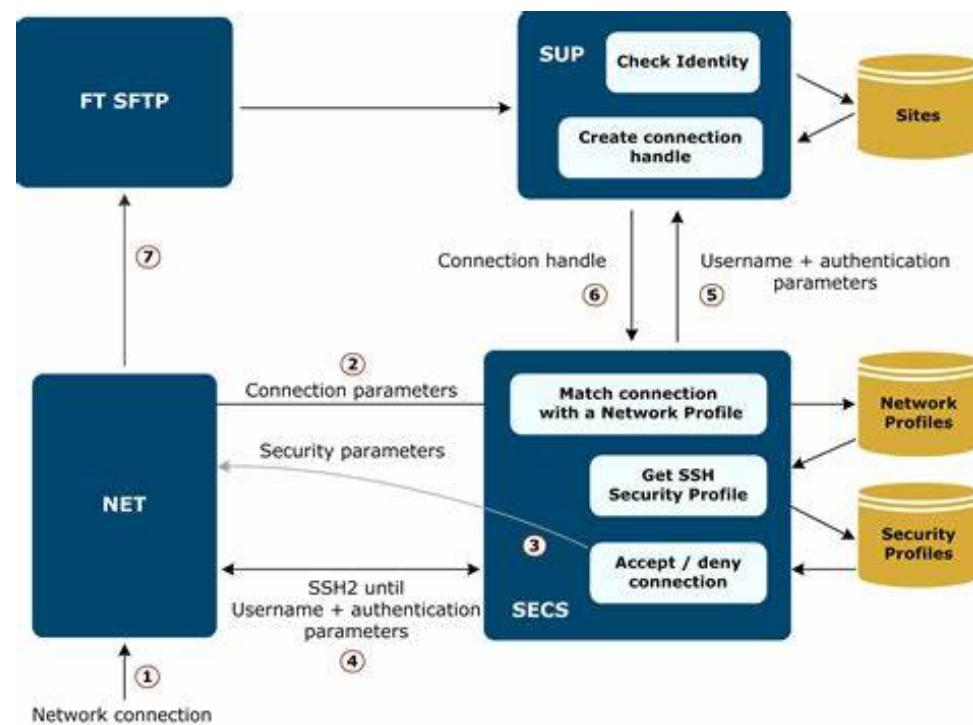
- Port numbers are used to keep track of different **conversations** that cross the network at the same time.
- Port numbers identify which upper layer service is needed, and are needed when a host communicates with a server that uses multiple services.
- Both TCP and UDP use port numbers to pass to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

SSH(Secure Shell)

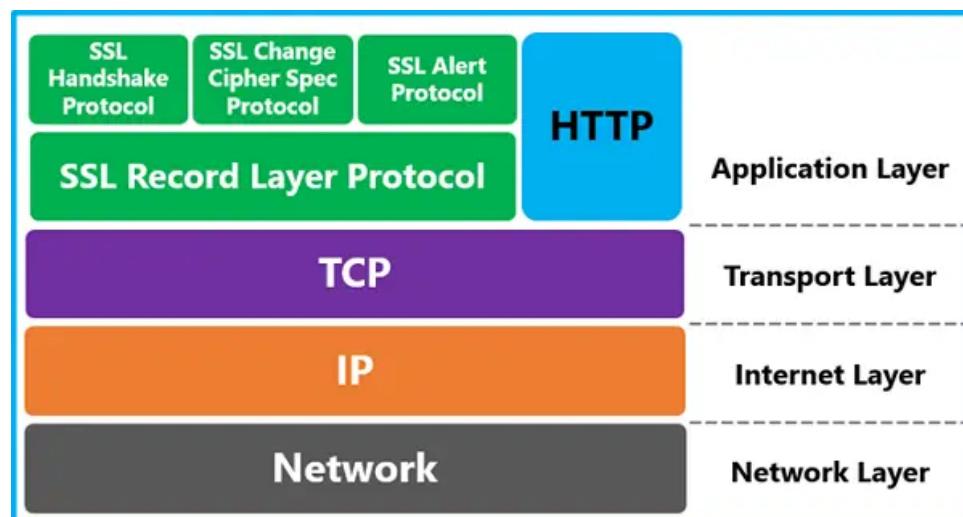
- 1) **Server authentication:**
Server proves its identity to the client

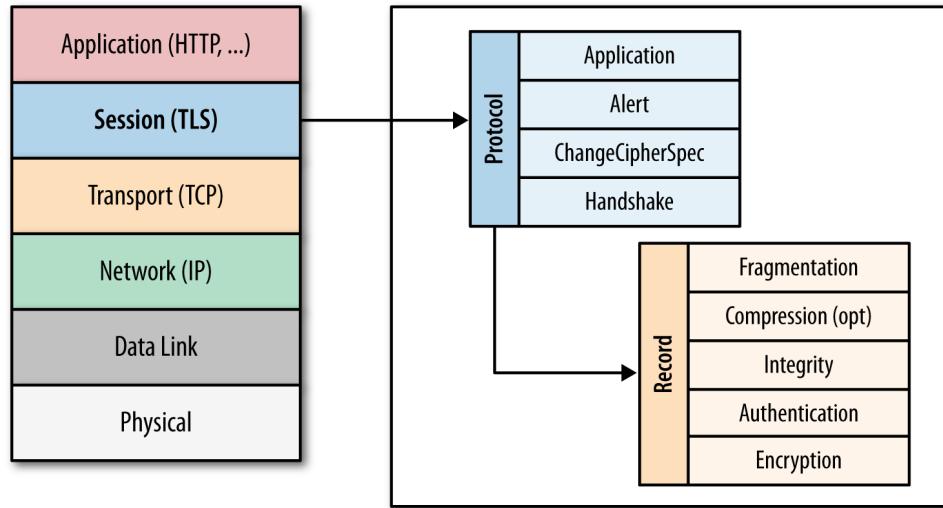
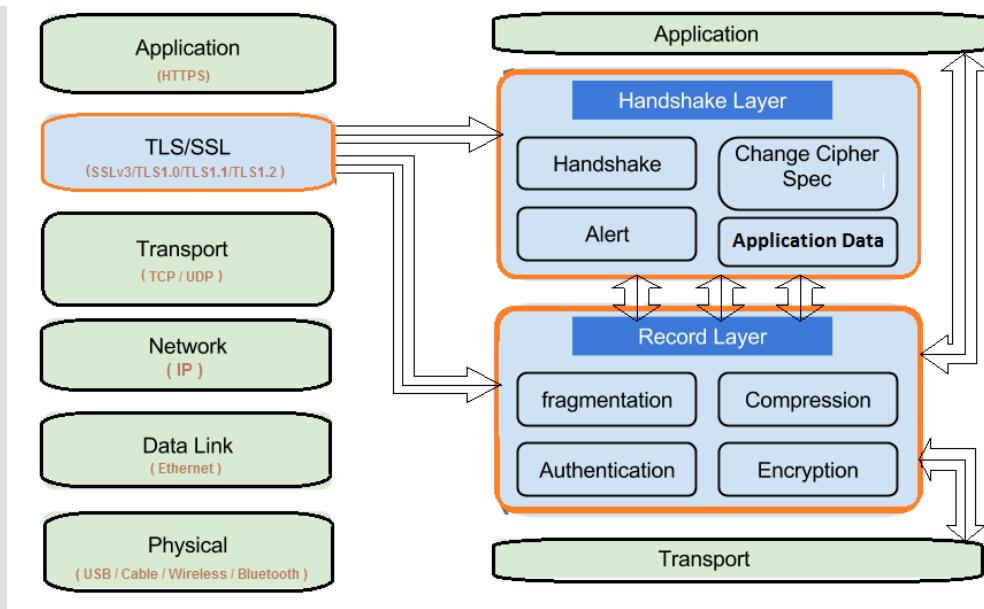


- 2) **User authentication:**
Client proves user's identity to the server

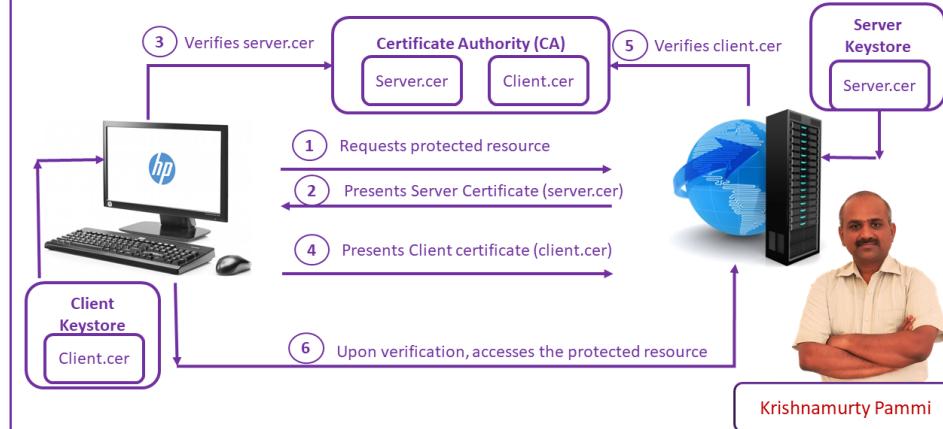


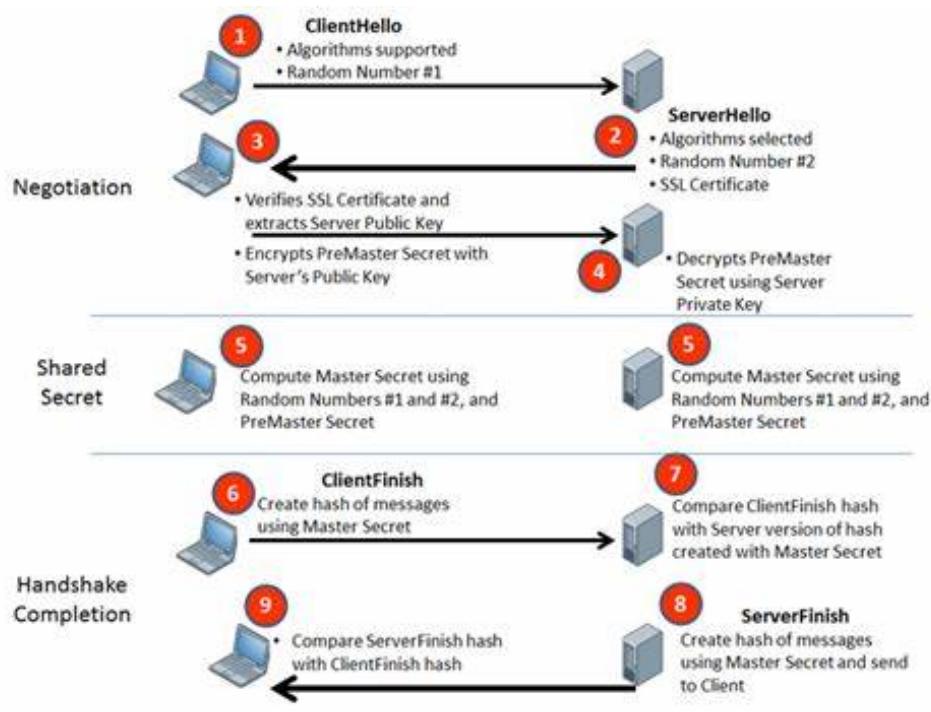
SSL(Secure Sockets Layers)





Web Security: Secure Socket Layer (SSL)





SSL/TLS VPNs vs. IPsec VPNs

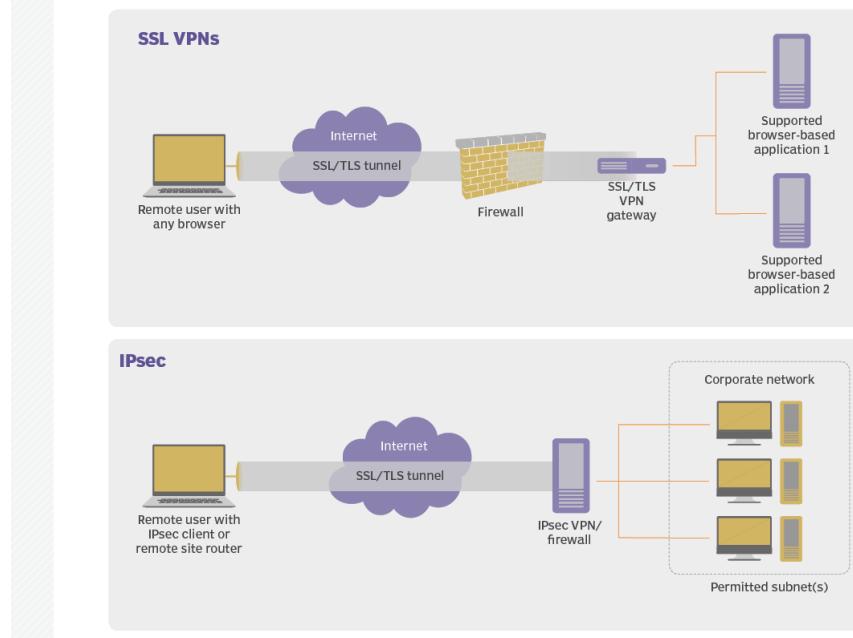
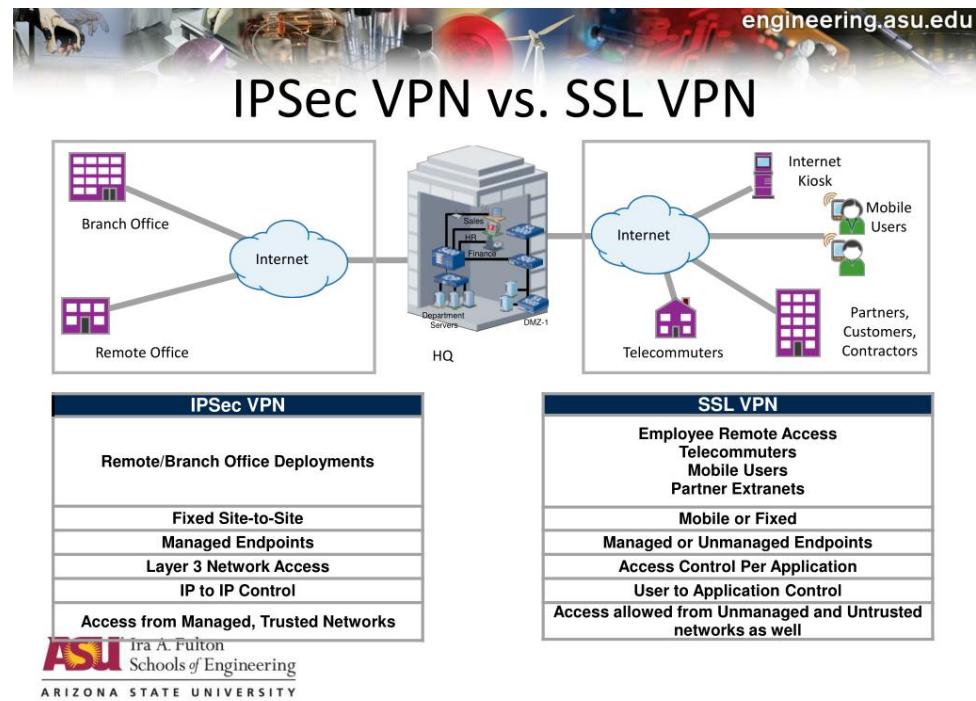
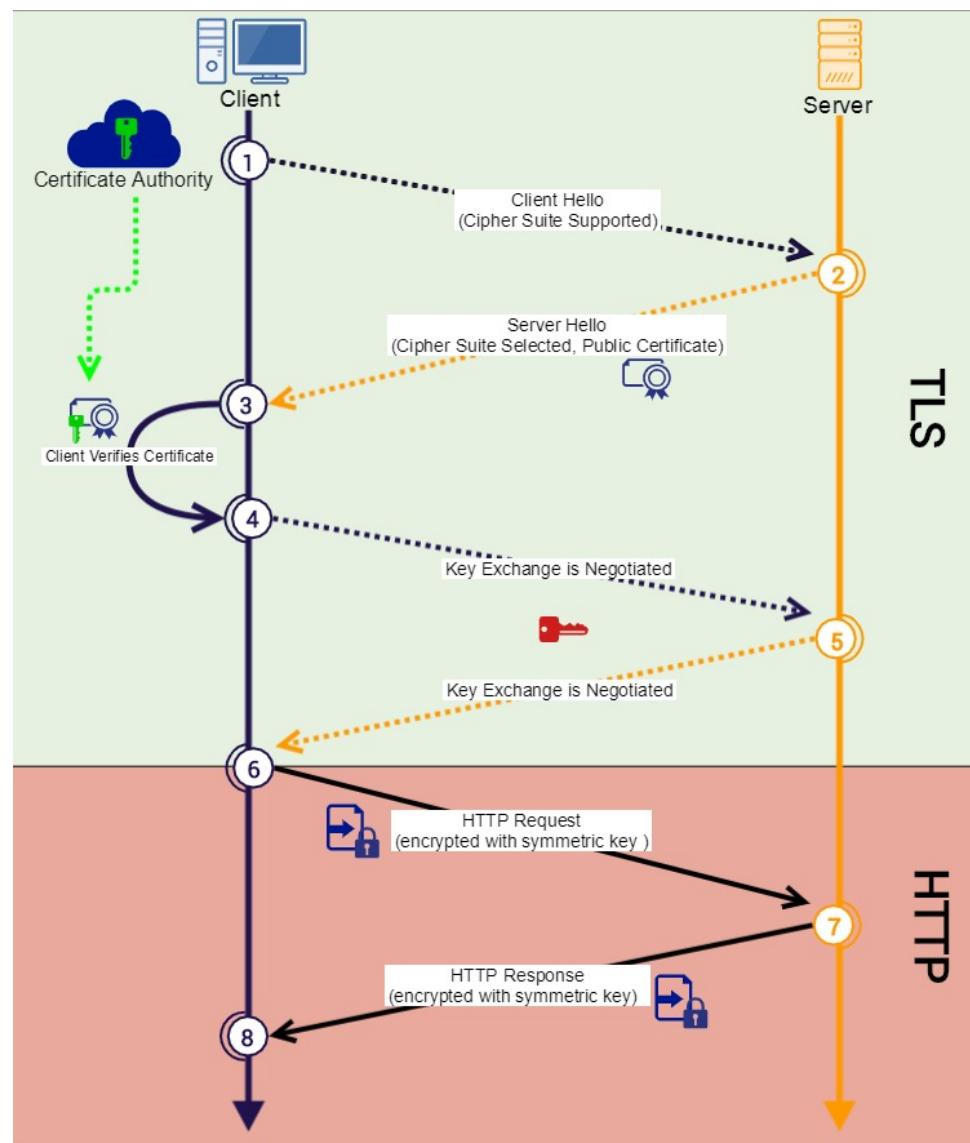


ILLUSTRATION CREDIT: VALLEPYFOTOLIA

© 2009 TechTarget. All rights reserved. TechTarget



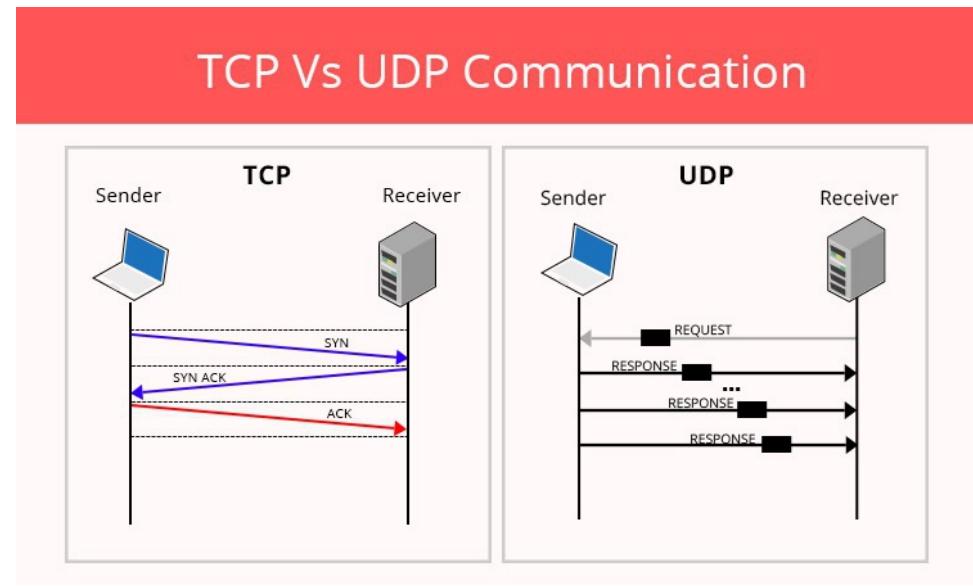
TLS

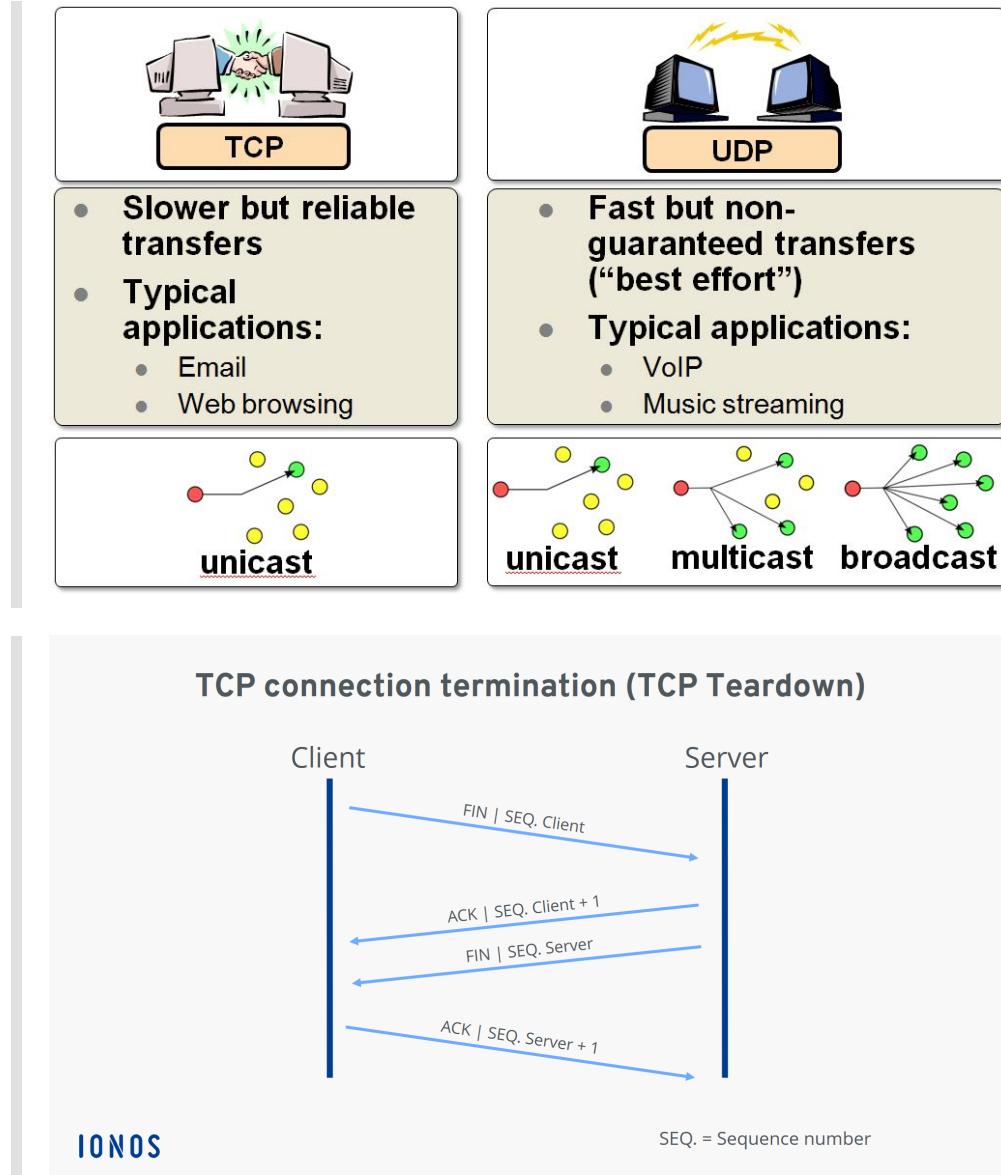


SSL vs TSL

S S L	V E R S U S	T L S
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 internet
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

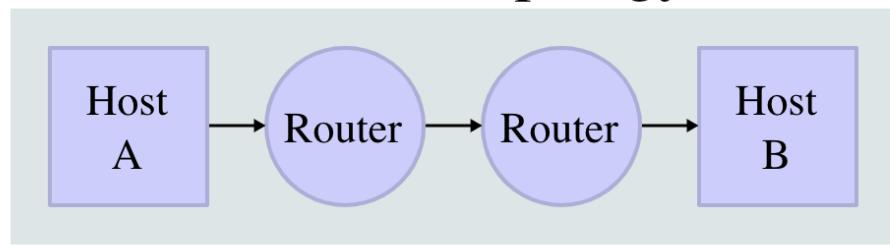
Network Layer - TCP vs UDP



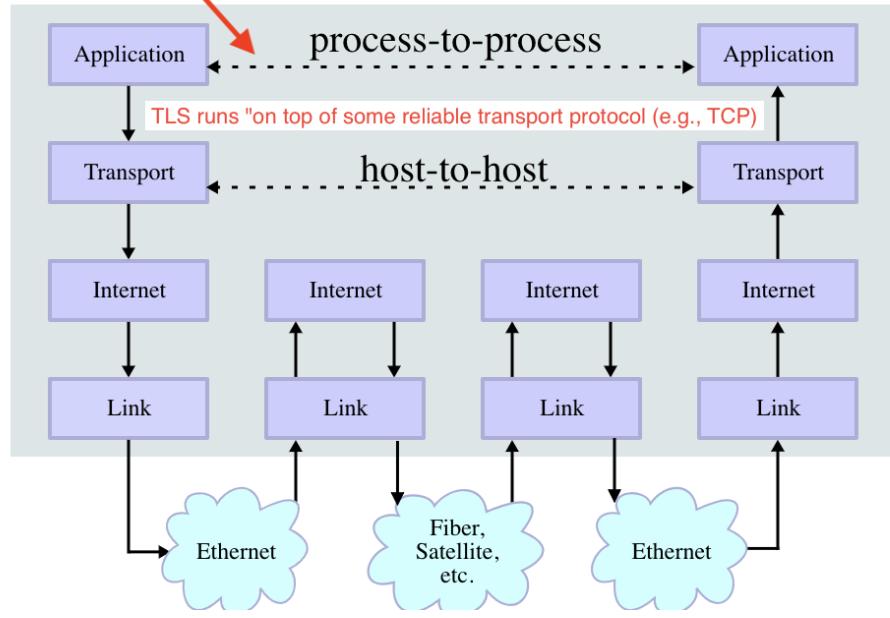


Data Flow

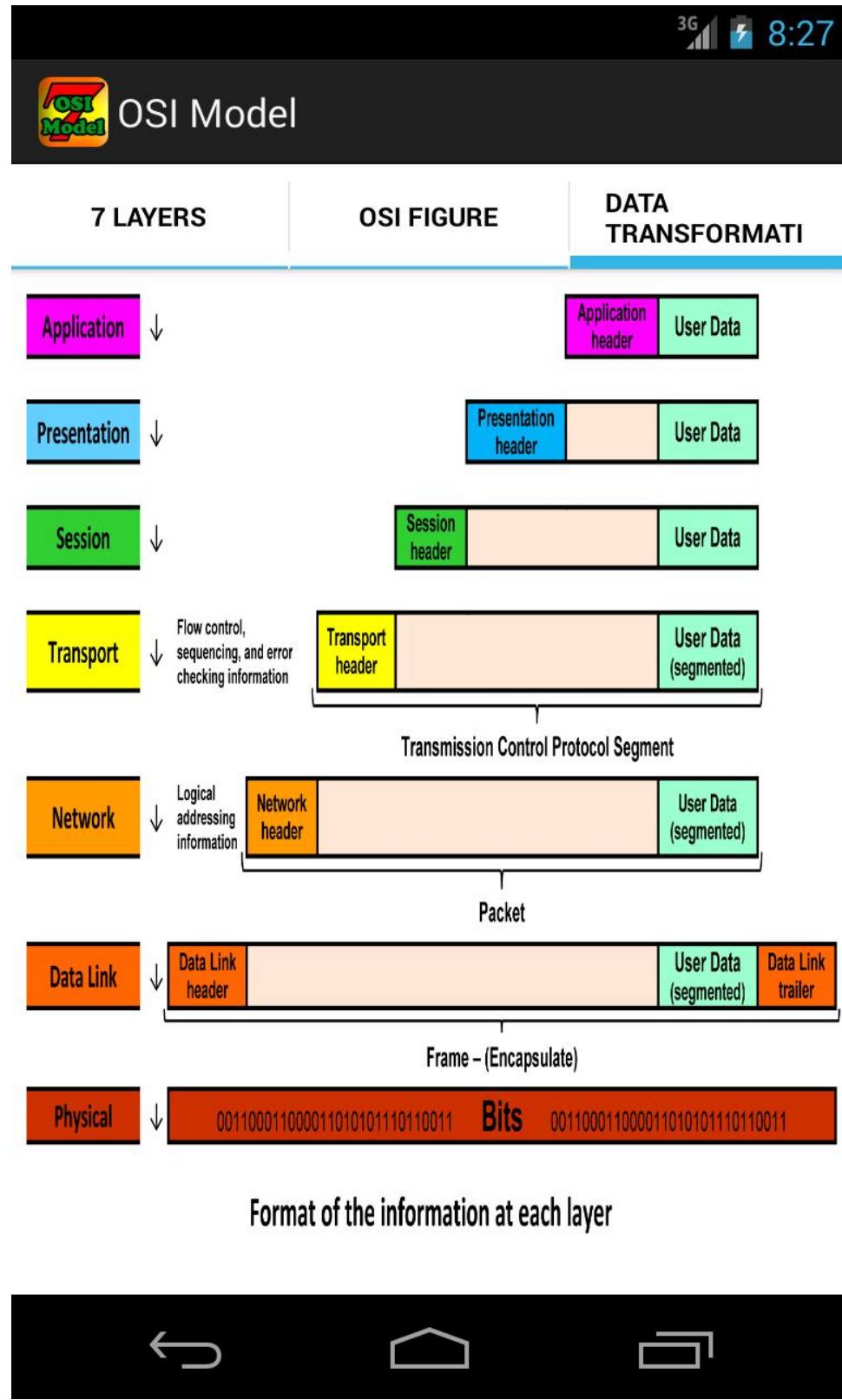
Network Topology

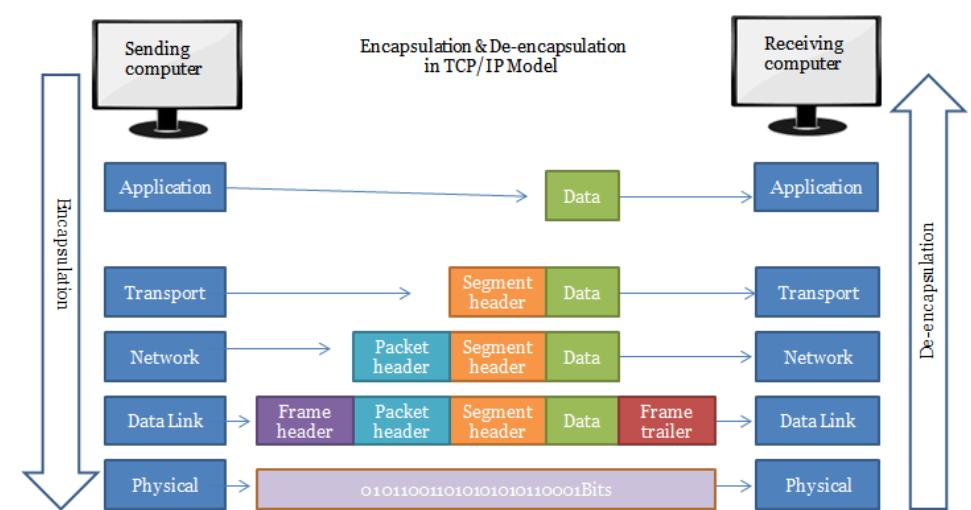
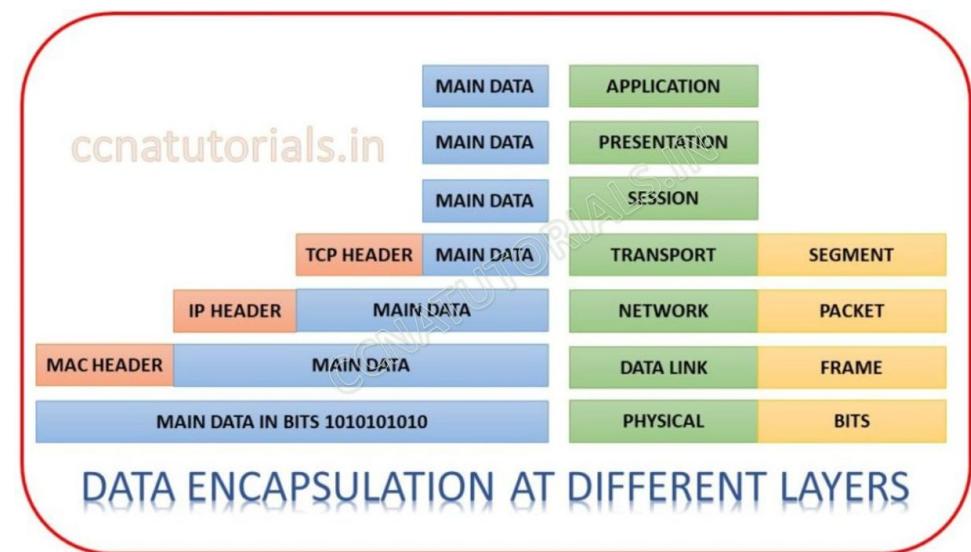
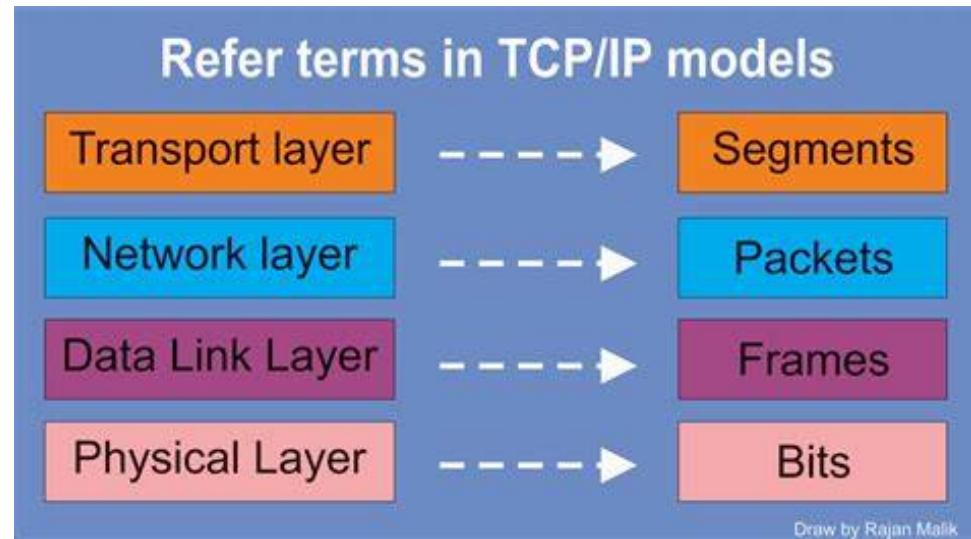


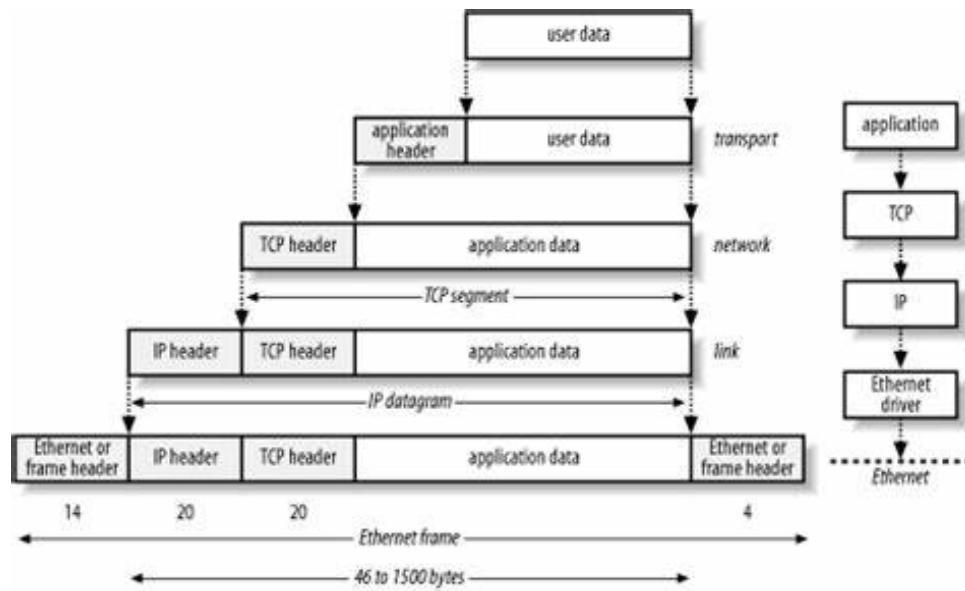
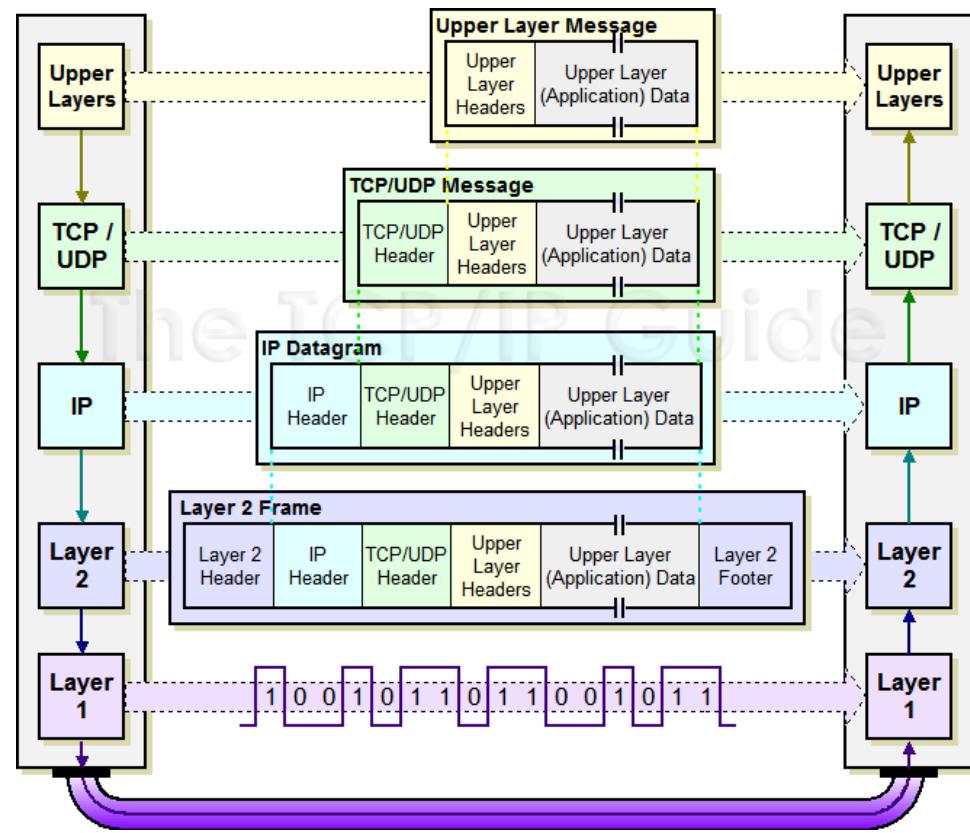
Data Flow

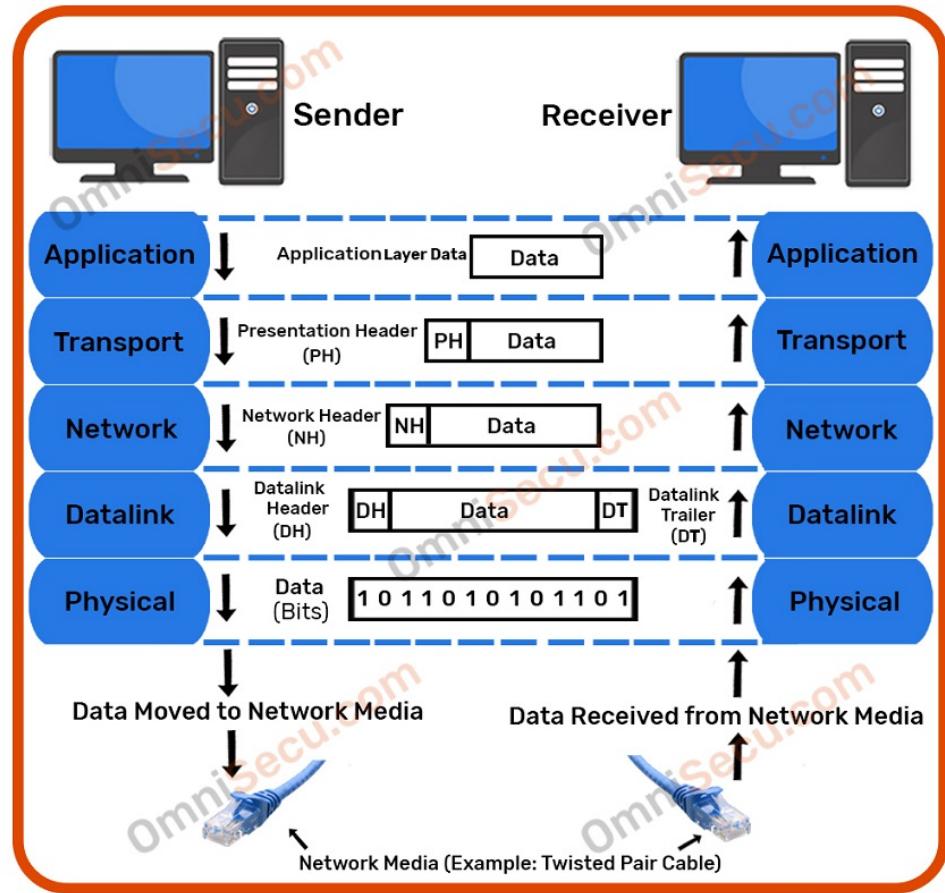


Encapsulation





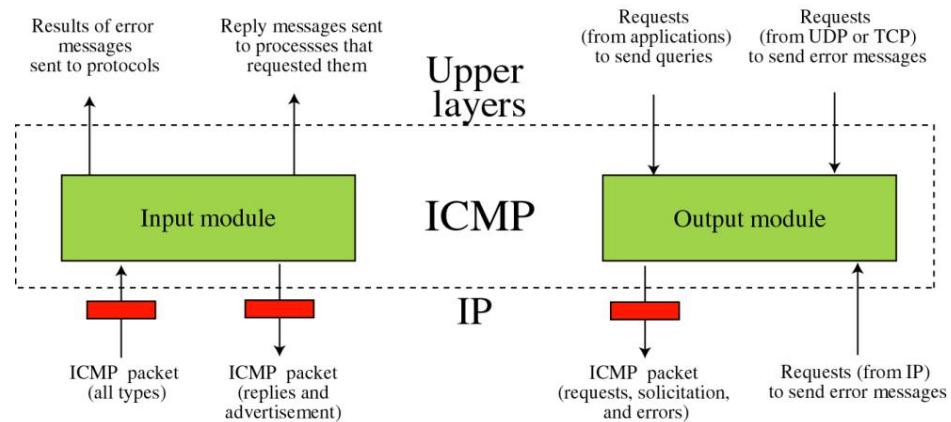




© OmniSecu.com

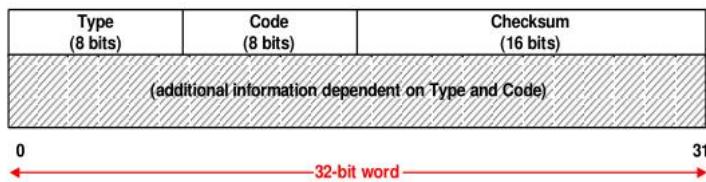
ICMP vs IGMP

ICMP package



ICMP: A helper protocol to IP

- The **Internet Control Message Protocol (ICMP)** is the protocol used for error and control messages in the Internet.
 - ICMP provides an error reporting mechanism of routers to the sources.
 - All ICMP packets are encapsulated as IP datagrams.
 - The packet format is simple:

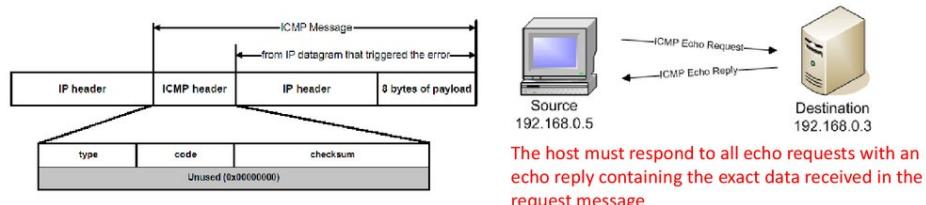


© Jörg Liebeherr (modified by M. Veeraraghavan)

1

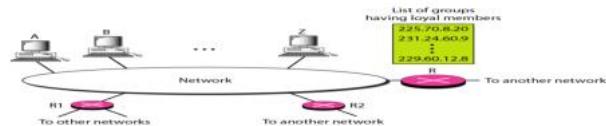
Internet Control Message Protocol (ICMP)

- The Internet Control Message Protocol (**ICMP**) is one of the main IP protocols; it is used by network devices, like routers, to send error messages (e.g., a requested service is not available or a host or router could not be reached)



IGMP Operation

- A multicast router connected to a network has a list of multicast addresses of the groups with at least one loyal member in that network.
- For each group, there is one router that has the duty of distributing the multicast packets destined for that group.
- This means that if there are three multicast routers connected to a network, their lists of group ids are mutually exclusive.

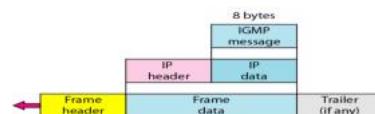


Computer Networks

21-17

IGMP: Encapsulation at Network Layer

- The IGMP message is encapsulated in an IP datagram, which is itself encapsulated in a frame.



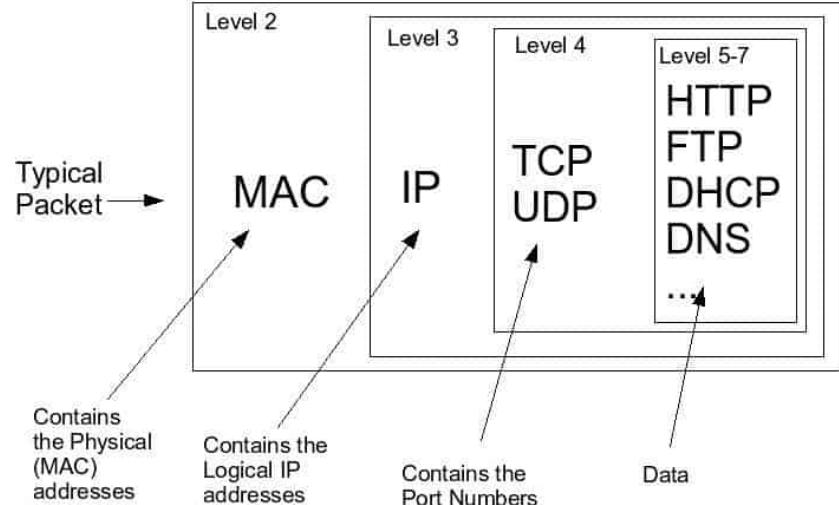
- The IP packet that carries an IGMP packet has a value of 1 in its TTL field

Type	IP Destination Address
Query	224.0.0.1 All systems on this subnet
Membership report	The multicast address of the group
Leave report	224.0.0.2 All routers on this subnet

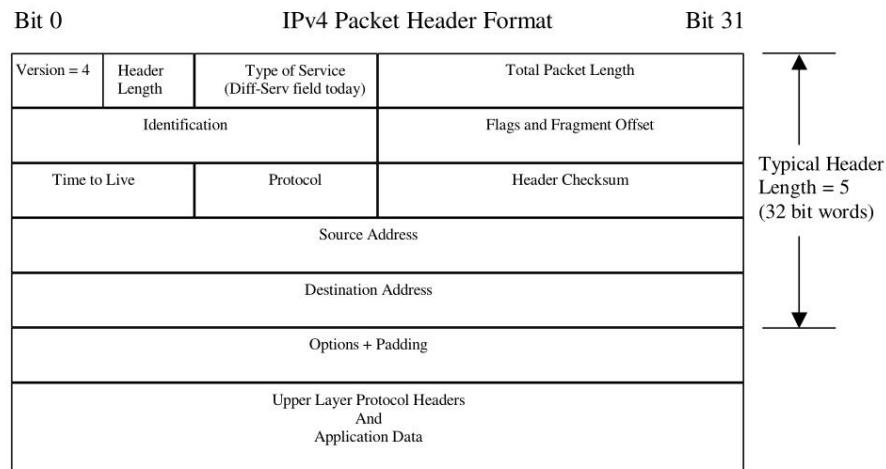
Computer Networks

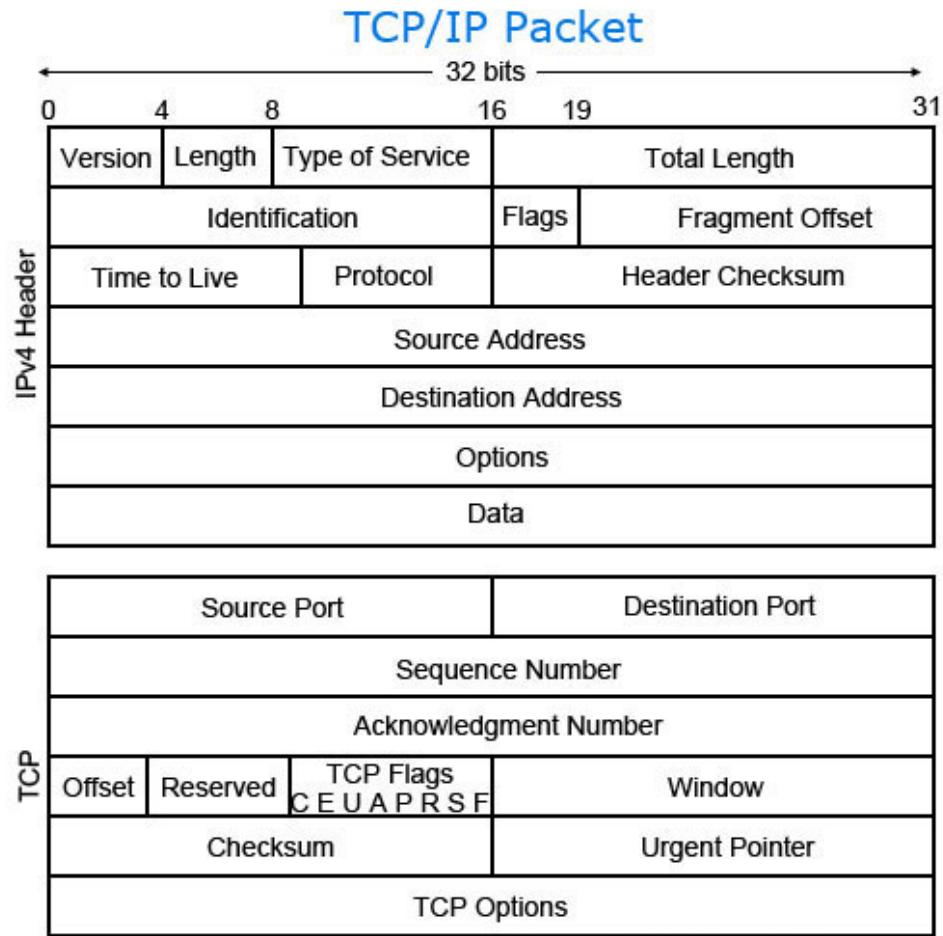
21-21

Packet



IP Packet Format





ComputerHope.com

Packet Switch

How TCP/IP Works

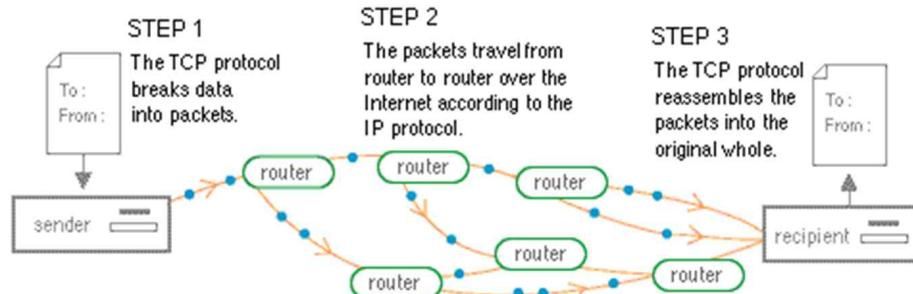


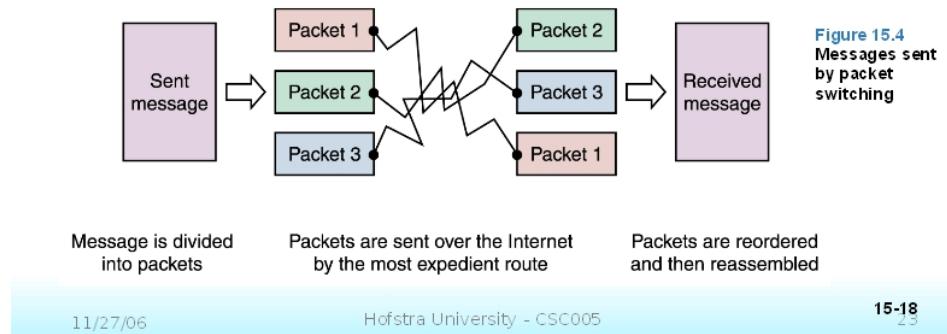
Figure 2. How data travels over the Net.

Dr. Vinton Cerf



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



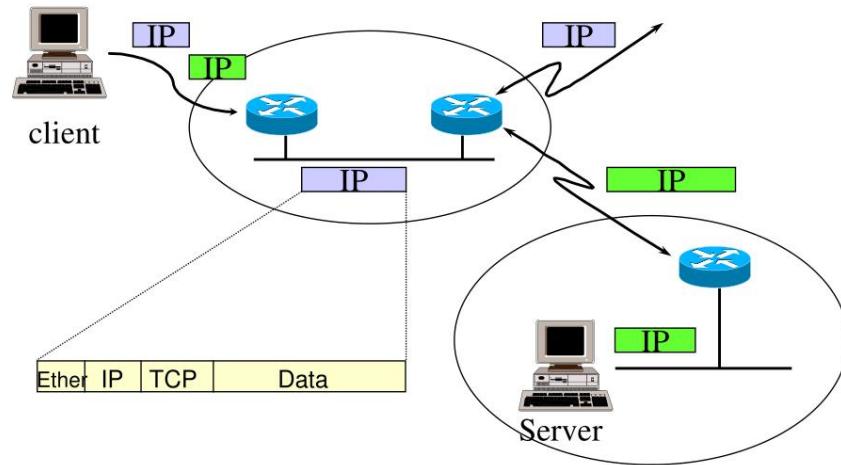
11/27/06

Hofstra University - CSC005

15-18

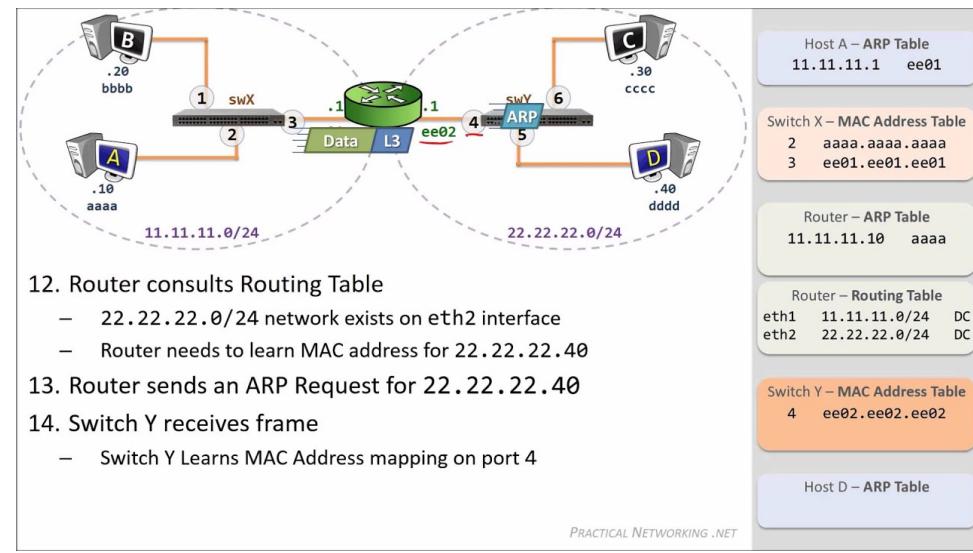
23

Packet Switch Network



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

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



IP4 vs IP6

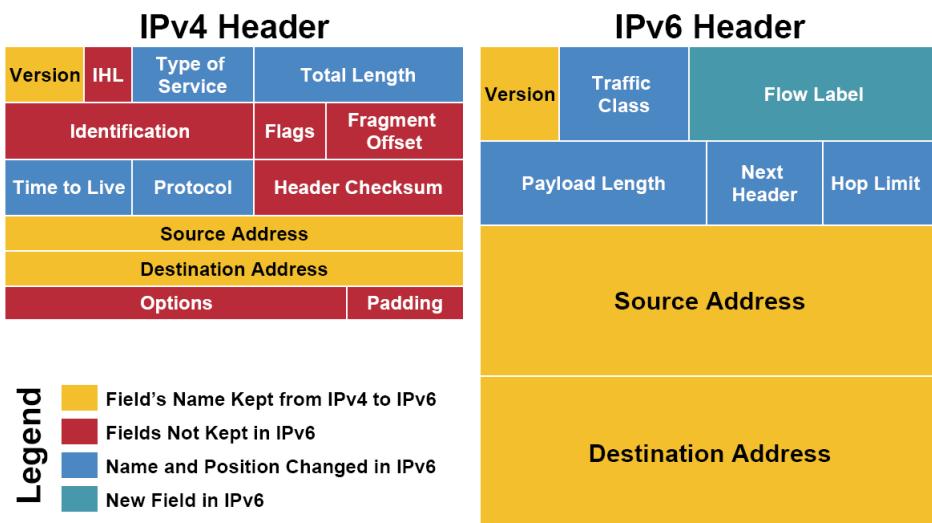
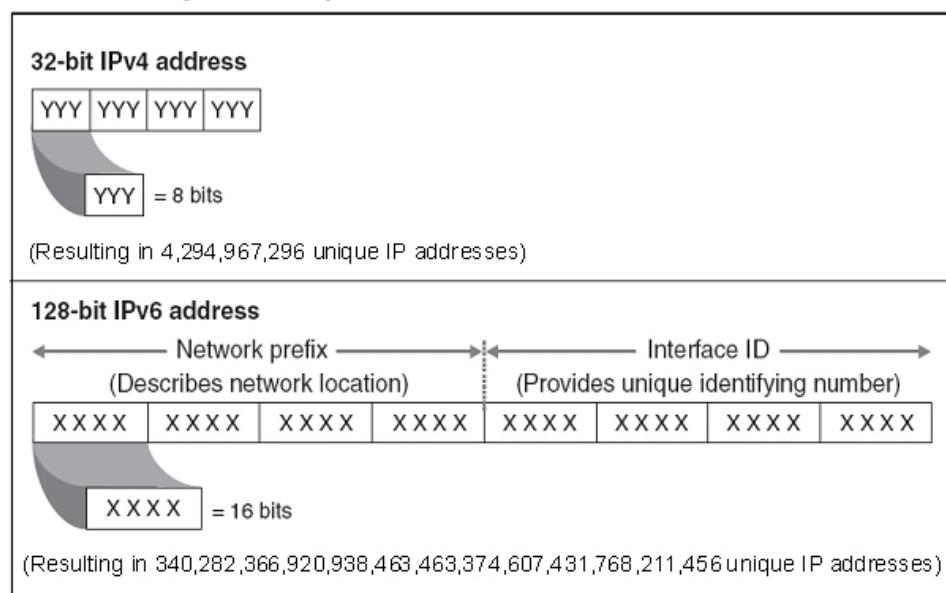


Figure 1: Comparison of IPv6 and IPv4 Address Scheme



Source: GAO.

Differences Between IPv4 and IPv6

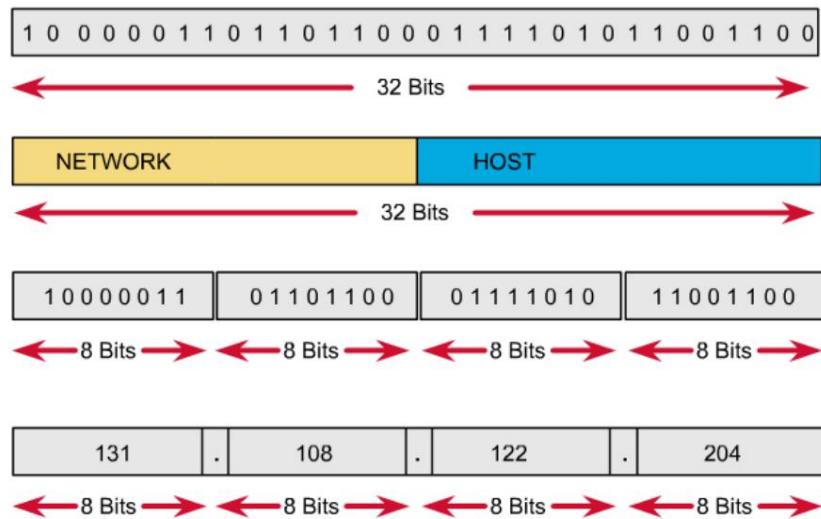
Feature	IPv4	IPv6
Fragmentation	Performed by routers and sending host	Performed only by sending host
Address Resolution	Broadcast ARP Request frames	Multicast Neighbor Solicitation messages
Manage multicast group membership	IGMP	Multicast listener discovery
Router Discovery	ICMP Router Discovery (optional)	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 Includes a checksum Includes options up to 40 bytes	Contains Flow Label field that specifies packet flow for QoS handling Does not include a checksum Extension headers used for optional data
DNS records	Address (A) records, maps host names Pointer (PTR) records, IN-ADDR.ARPA DNS domain	Address (AAAA) records, maps host names 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

IP address format

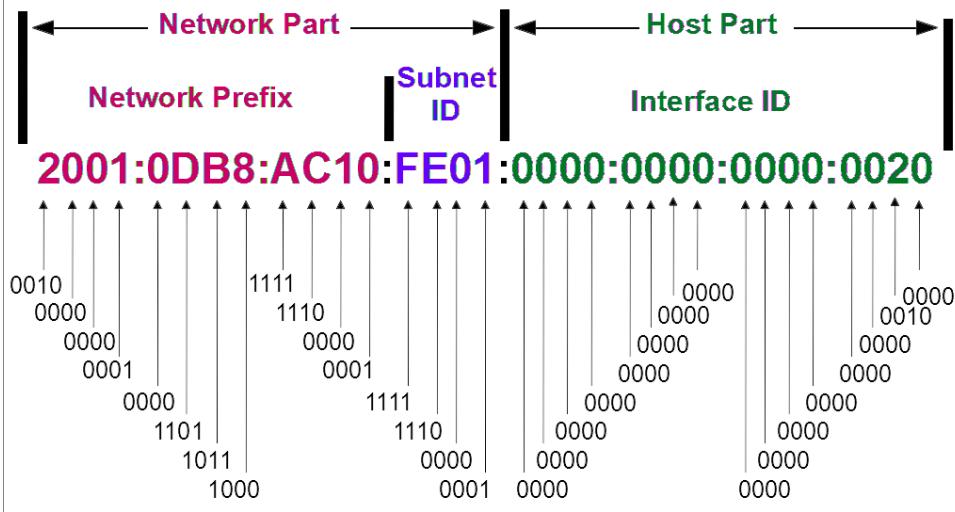


Class	Private Address Ranges
Class A	10.0.0.0 – 10.255.255.255
Class B	172.16.0.0 – 172.31.255.255
Class C	192.168.0.0 – 192.168.255.255
Loopback	127.0.0.0 – 127.255.255.255 (127.0.0.1)

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

2001:0DB8:AAAA:1111:0000:0000:0000:0100/64

1 2 3 4 5 6 7 8
 2001 : 0DB8 : AAAA : 1111 : 0000 : 0000 : 0000 : 0100
 16 bits 16 bits

- 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	<u>00000000</u> – <u>01111111</u> (0 – 127)	1.0.0.1 – 126.255.255.254	126	$2^{24} - 2$
B	<u>10000000</u> – <u>10111111</u> (128 – 191)	128.1.0.1 – 191.255.255.254	16000	65000
C	<u>11000000</u> – <u>11011111</u> (192 – 223)	192.0.1.1 – 223.255.255.254	2 Million	254
D	<u>11100000</u> – <u>11101111</u> (224 – 239)	224.0.0.0 – 239.255.255.255		Multicast addresses
E	<u>11110000</u> – <u>11111111</u> (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	<u>00000001</u> to <u>01111110</u> *	16,777,214
Class B	128-191	<u>10000000</u> to <u>10111111</u>	65,534
Class C	192-223	<u>11000000</u> to <u>11011111</u>	254

*127 (01111111) is a Class A address reserved for loopback testing and cannot be assigned to a network.



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

What Do You Need To Know About Private 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

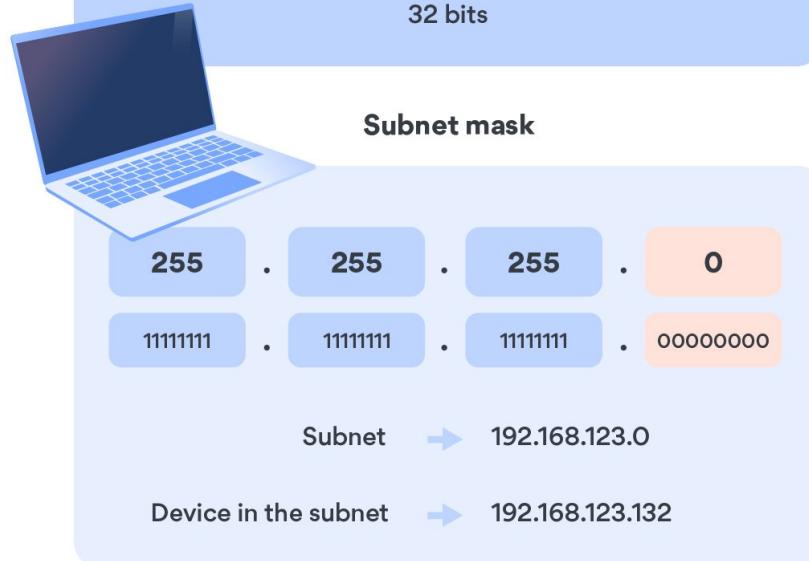
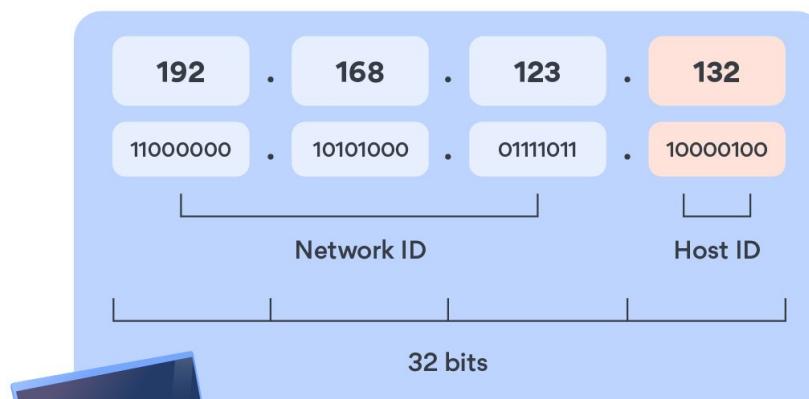
127.0.0.0 – 127.255.255.255
(127.0.0.1)

Subnet Mask

Subnet Mask

Prefix	Hosts	32-Borrowed=CIDR	2^Borrowed = Hosts	Binary=> dec = Prefix
.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

IP address explained



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

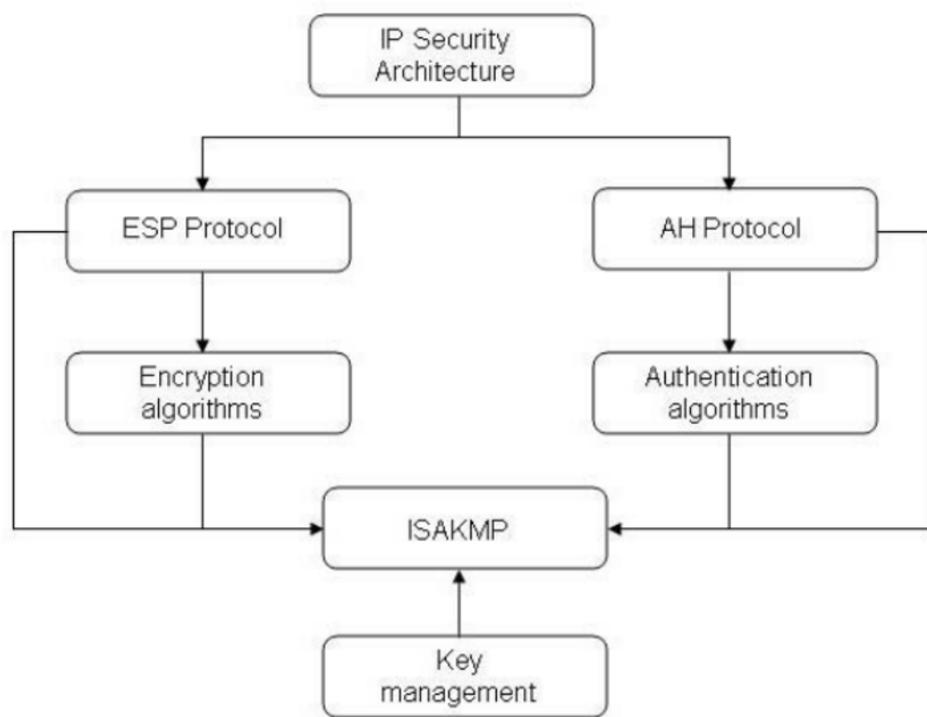
- A "1" bit in the subnet mask means that the corresponding bit in the IP address should be read as a network number
- A "0" bit in the subnet mask means that the corresponding bit in the IP address should be read as a host bit.
- /n "slash" tells us how many "1" bits are in the subnet mask.

The Subnet Mask

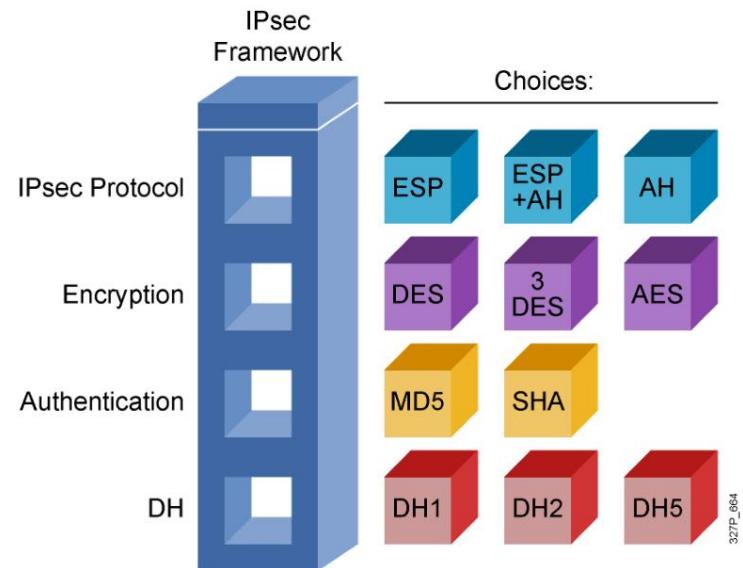
- Subnet Mask:
 - Let's not forget about the subnet mask.
 - Each class has a **default or "natural"** subnet mask based on the default number of bits used for the network and host portion.

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

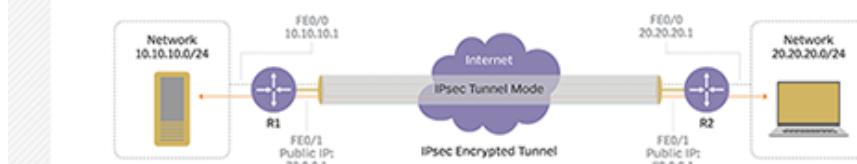
IPSec



IPsec Protocol Framework



IPsec tunnel mode



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

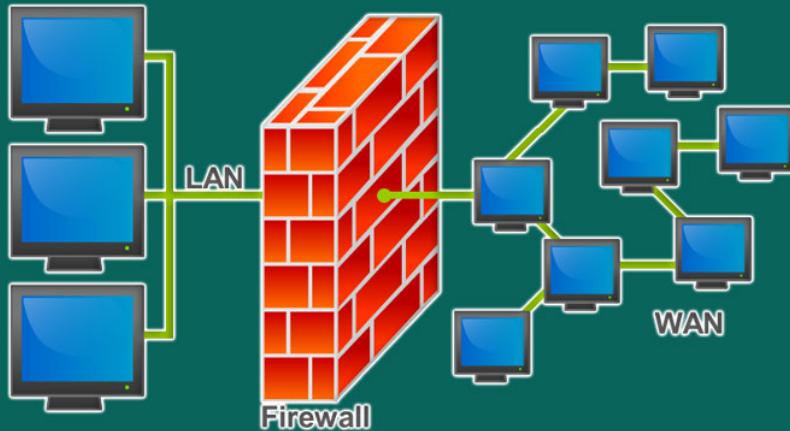
8: Network Security

8-1

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.

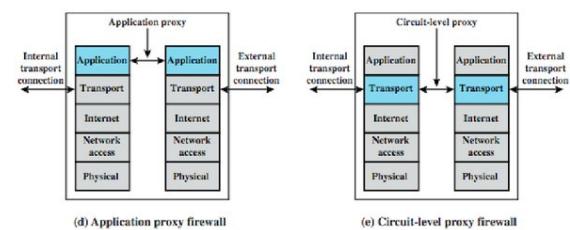
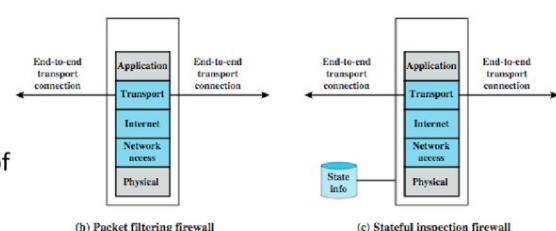
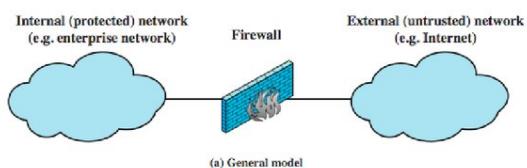
It simply prevents unauthorized access to or from a private network. Used to enhance the security of computers connected to a network, such as LAN or the Internet. Considered as an integral part of a comprehensive security framework for your network.

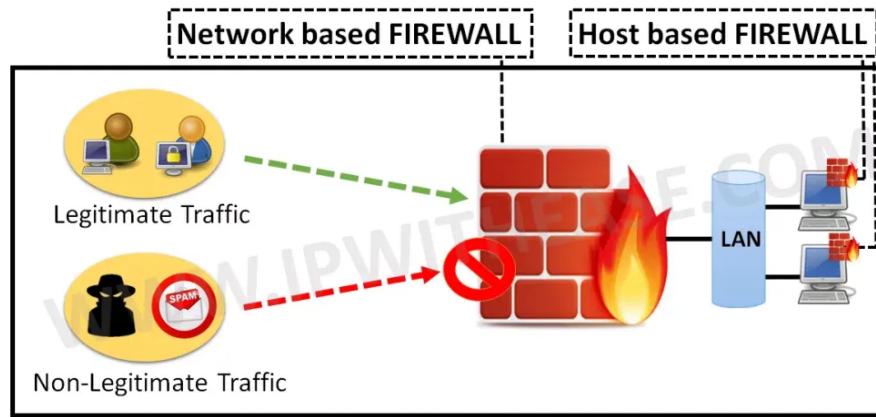


Types of Firewalls

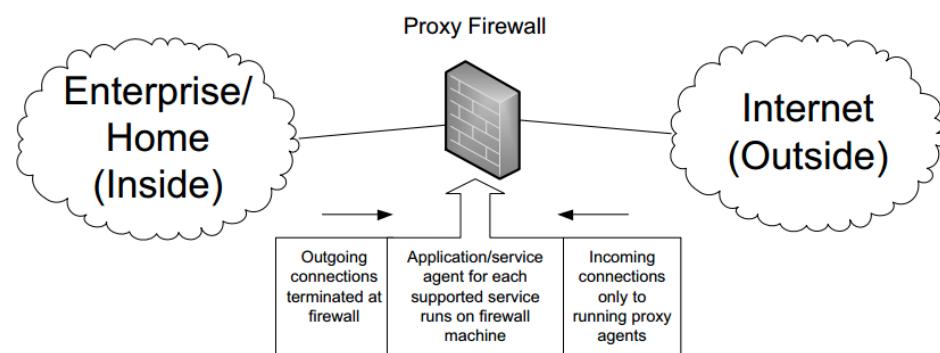
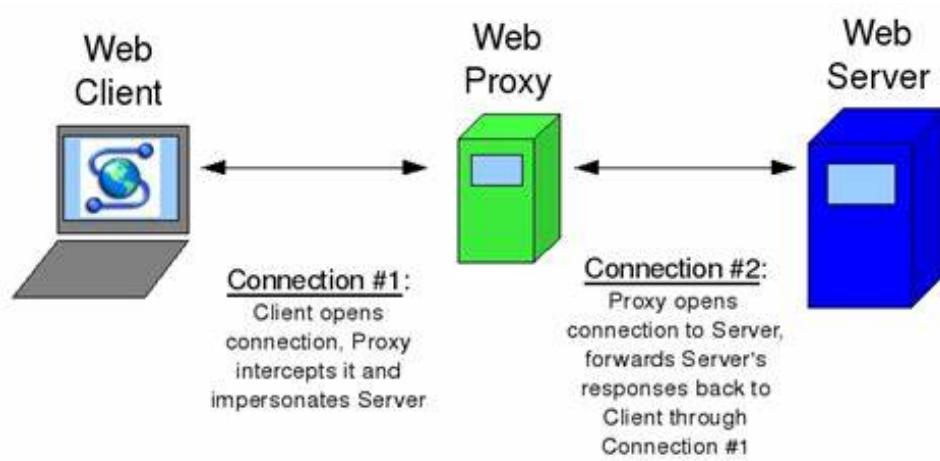
Positive (negative) filter:
Allow (reject) packets that meet a criteria

Stateful inspection: Keeps track of TCP connections





Proxy



Proxy Cache

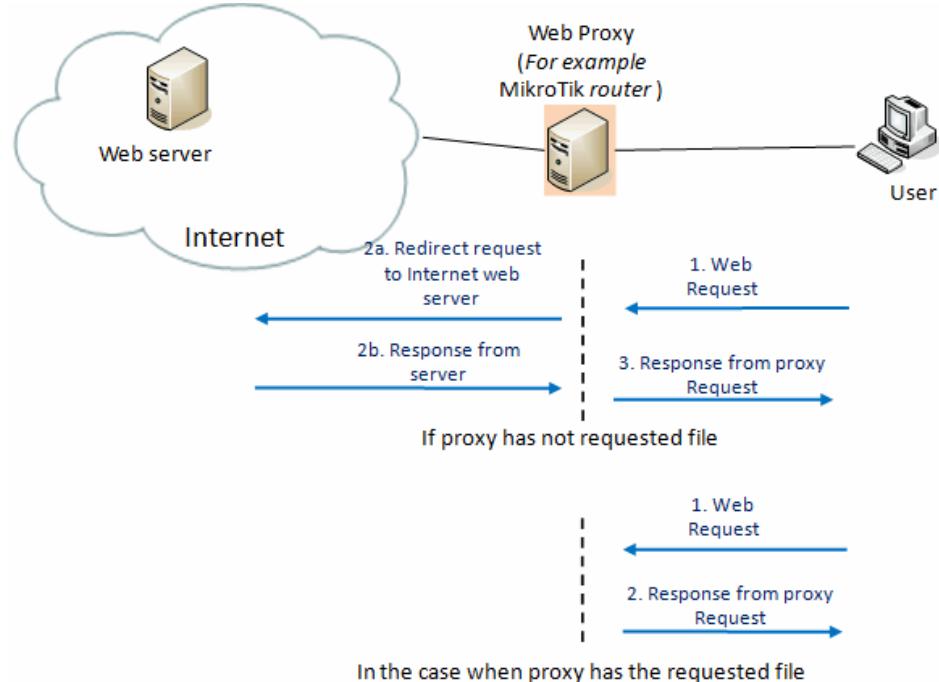
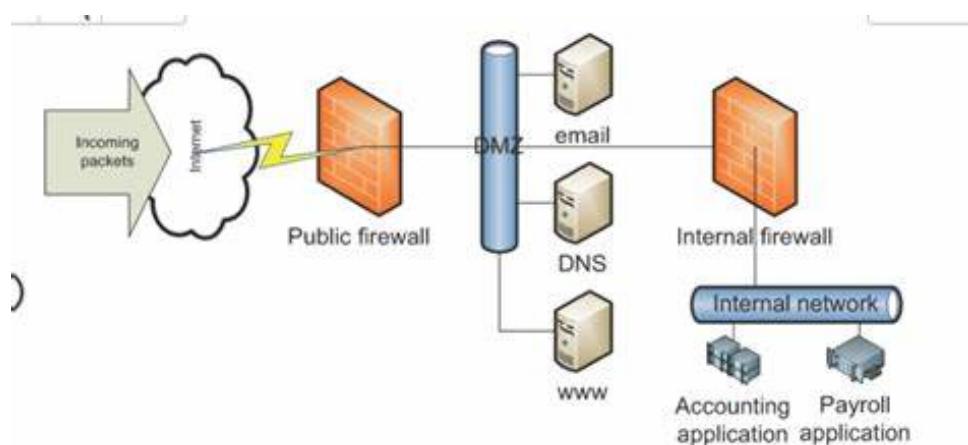


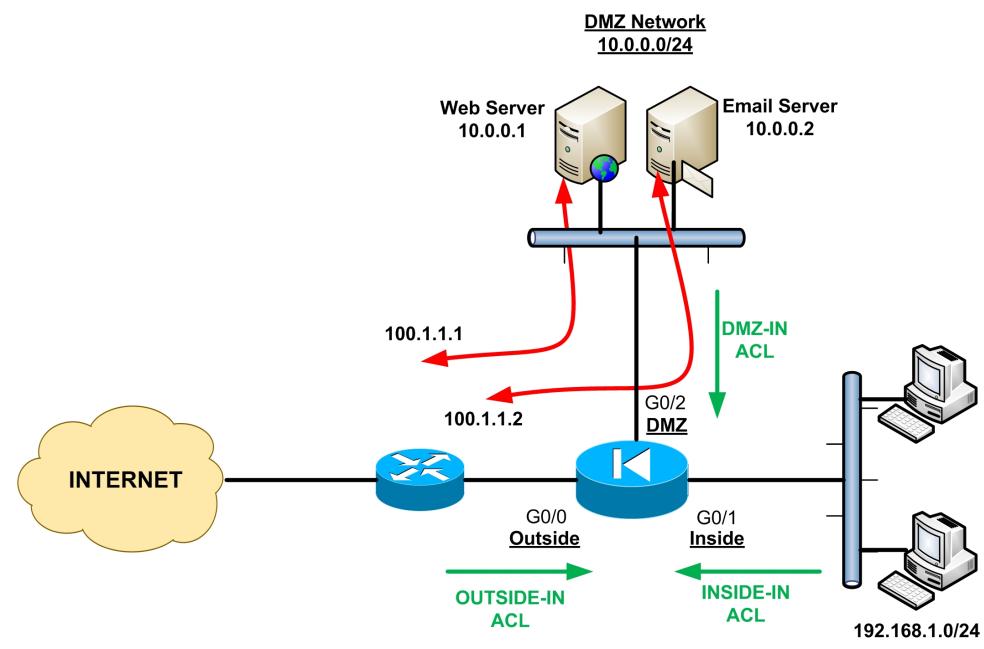
Figure 10.1. Web proxy basic operation scheme

ACL (Access Control List)

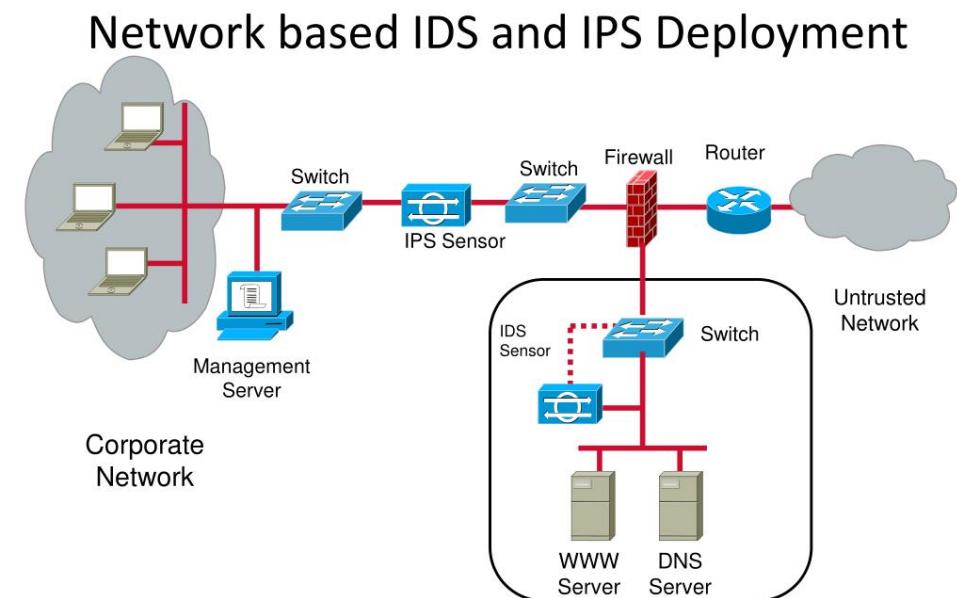
Parameter	ACL	Firewall
Asset Type	Feature on Layer 3 devices and Firewalls	Hardware or Software
Stateful/Stateless inspection	Performs stateless inspection	Performs Stateful inspection
Scope wrt OSI	Upto Layer 4	Upto Layer 7
Security	Low	High
Intrusion detection	Not possible	Possible
Target deployment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

DMZ





IPS/IDS



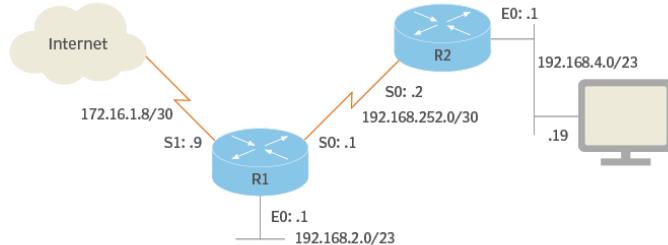
Engineering and Management of Secure Computer Networks

15

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

©2019 TECHTARGET. ALL RIGHTS RESERVED TechTarget

Routing Protocol

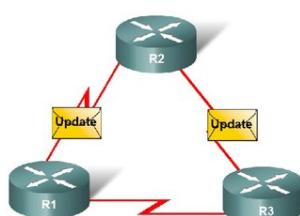
Dynamic IP Routing Protocols

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

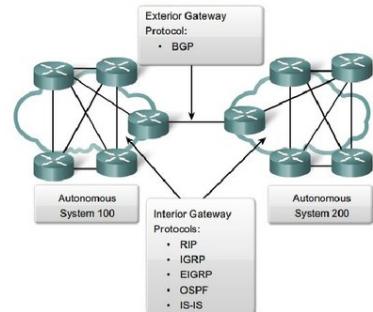
There are quite many dynamic routing protocols for routing IP packets. The most common protocols are:

- **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



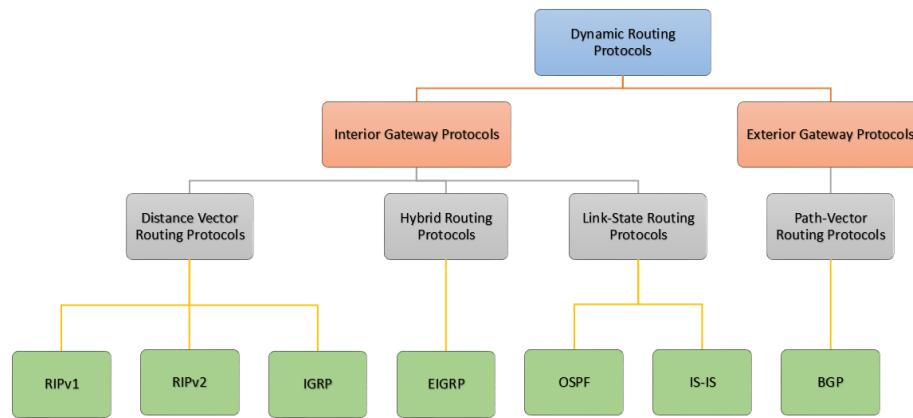
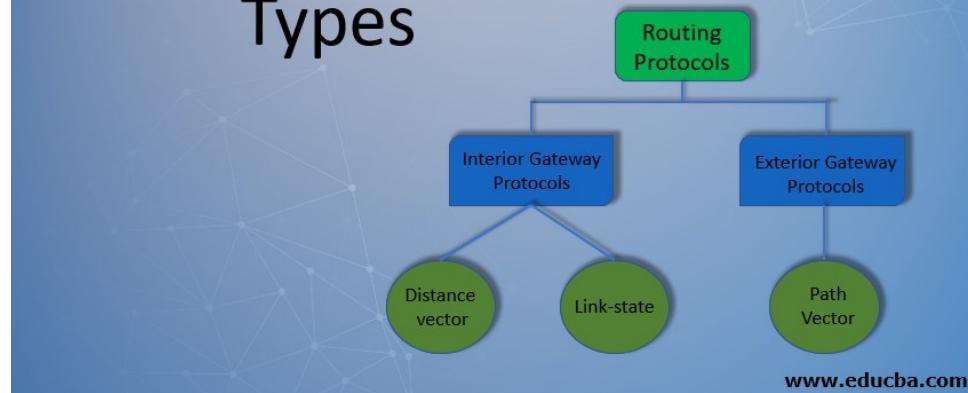
IGP vs. EGP Routing Protocols



16

Routing Protocols

Types



Routing Protocols for IP Networks

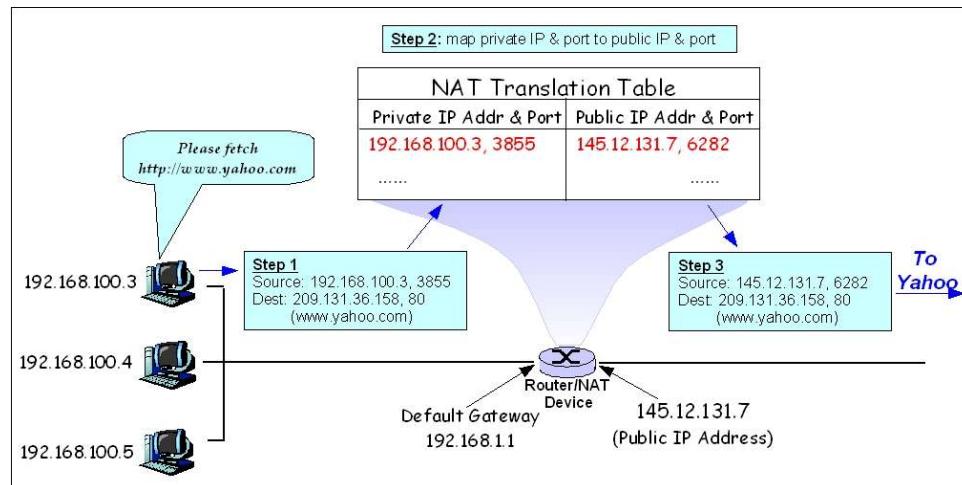
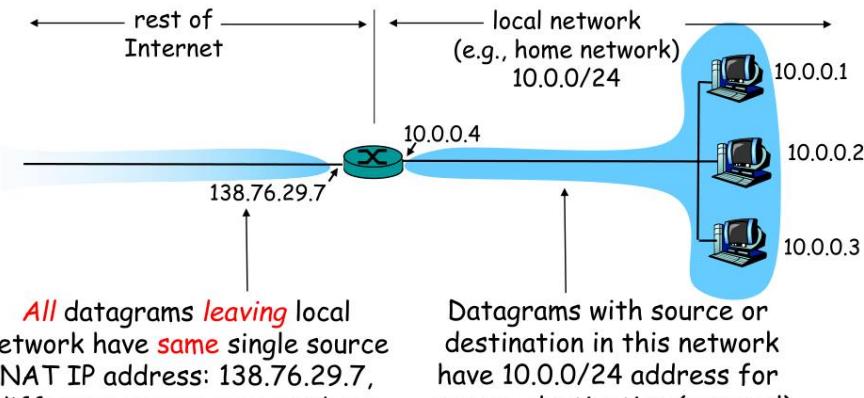
Protocol	Type	Scalability	Metric	IP classes
RIP-1	Distance vector	Small	Hop count	Classful
RIP-2	Distance vector	Small	Hop count	Classless
OSPF-2	Link state	Large	Cost	Classless
IS-IS	Link state	Very large	Cost	Classless
IGRP	Distance vector	Medium	Bandwidth, delay, load, MTU, reliability	Classful
EIGRP	Dual	Large	Bandwidth, delay, load, MTU, reliability	Classless
BGP	Distance vector	Large	Vector of attributes	Classless



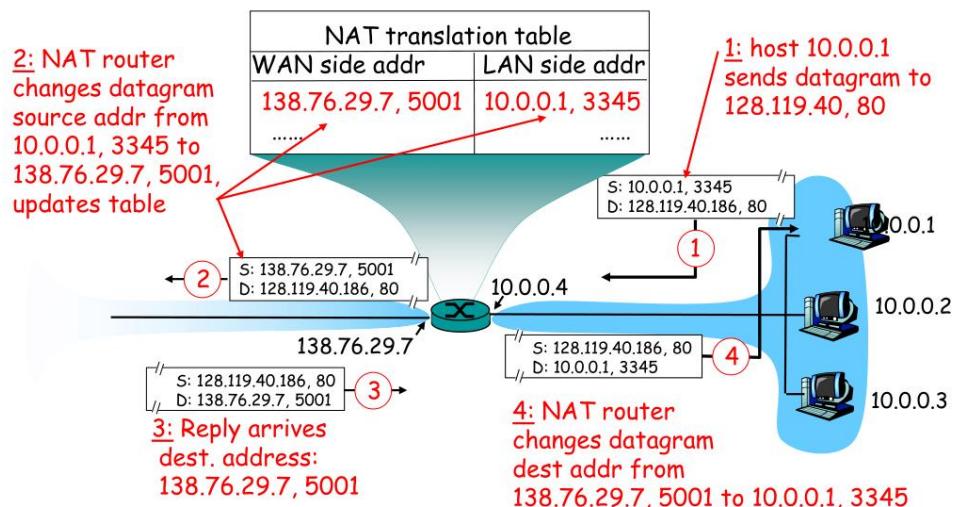
7

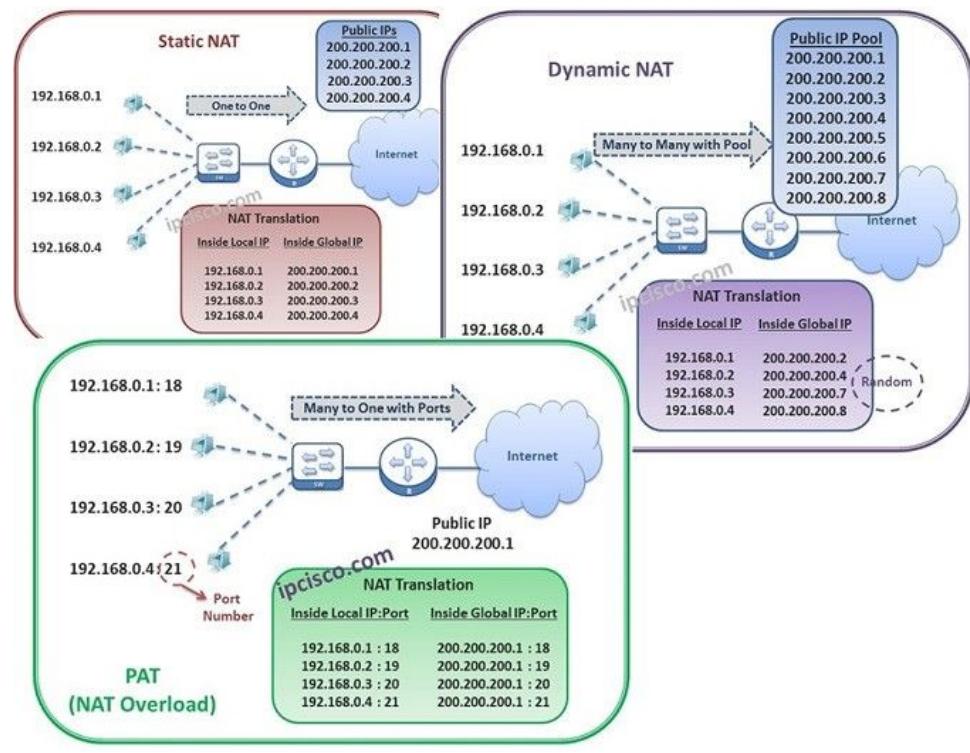
NAT(Network Address Translation)

NAT: Network Address Translation

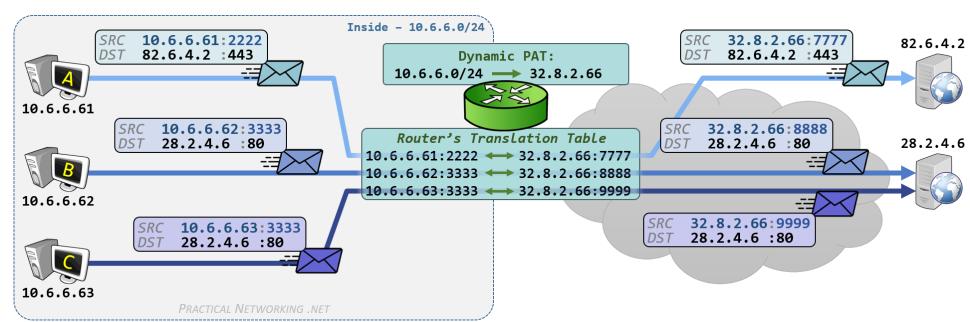
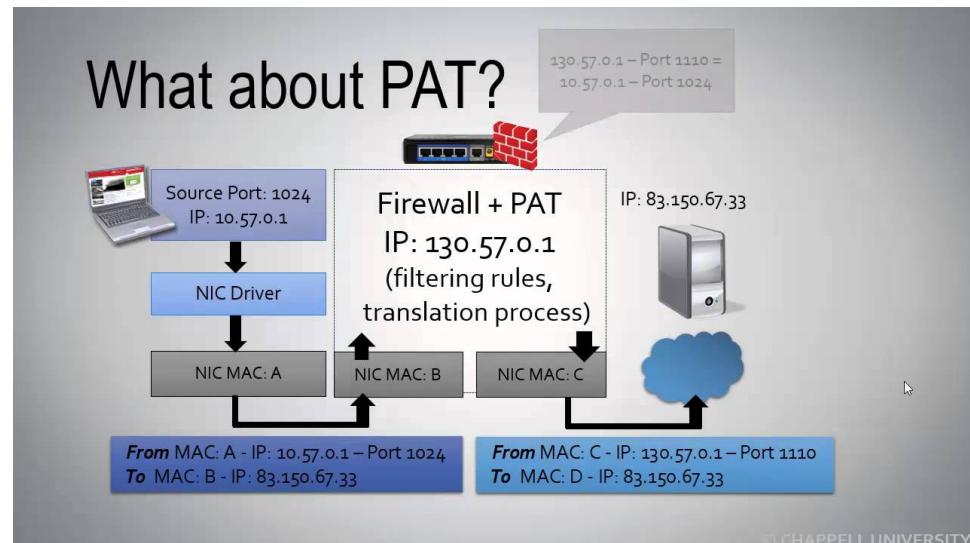


NAT: Network Address Translation





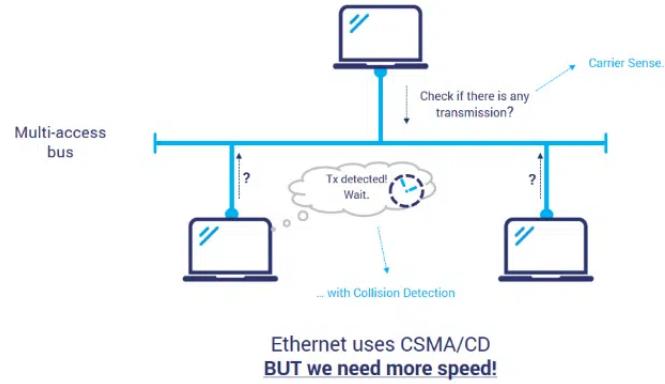
PAT (Port Address Translation)



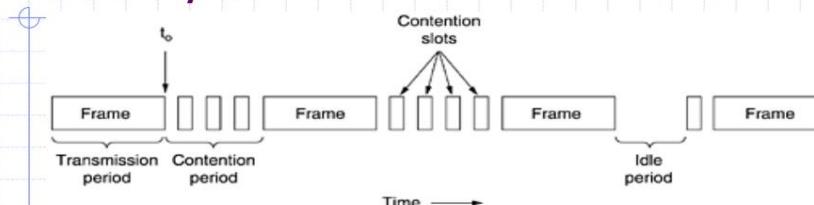
CSMA/CD

CSMA/CD

CSMA/CD – mechanism for collision detection introduced to detect transmission

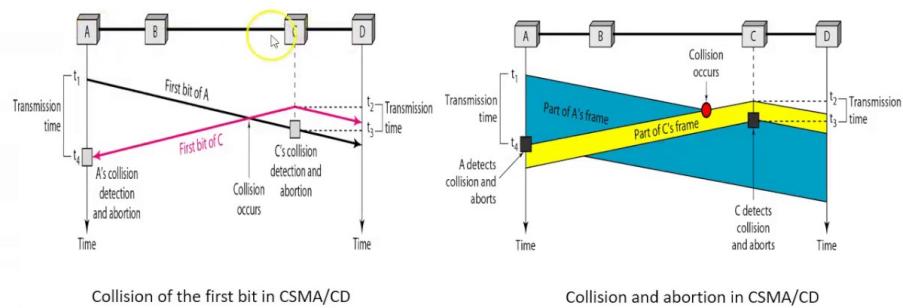


CSMA/CD



- ◆ Sense the channel
- ◆ Stop sending when detecting collision
- ◆ After collision wait a random amount of time and try again.

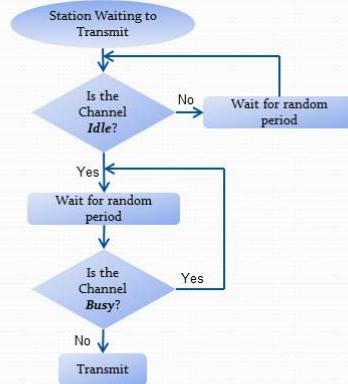
Collision in CSMA /CD



CSMA/CA

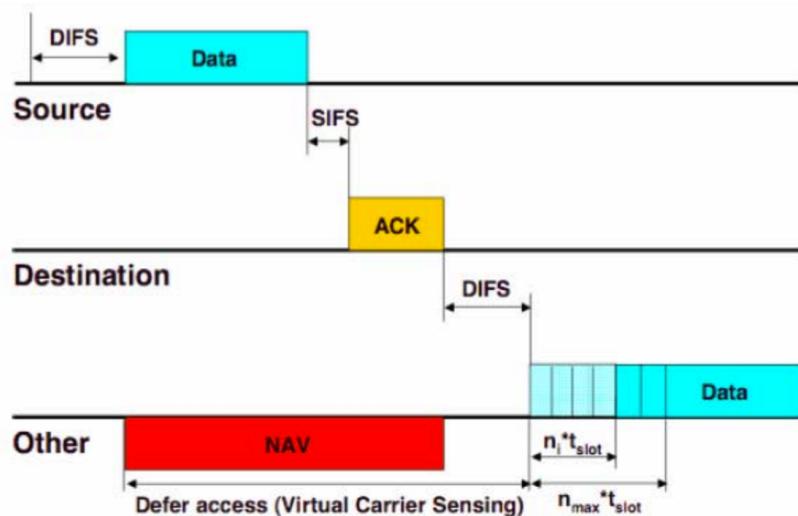
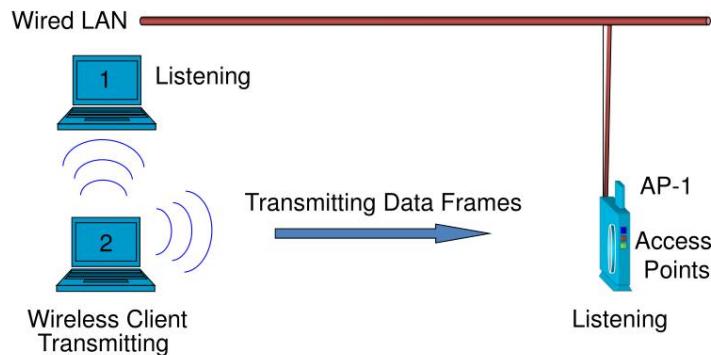
CSMA/CA

- 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 be explained later in the presentation

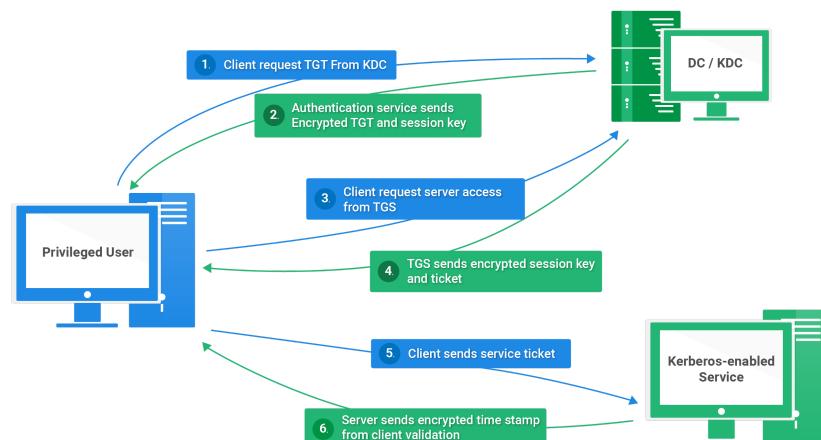
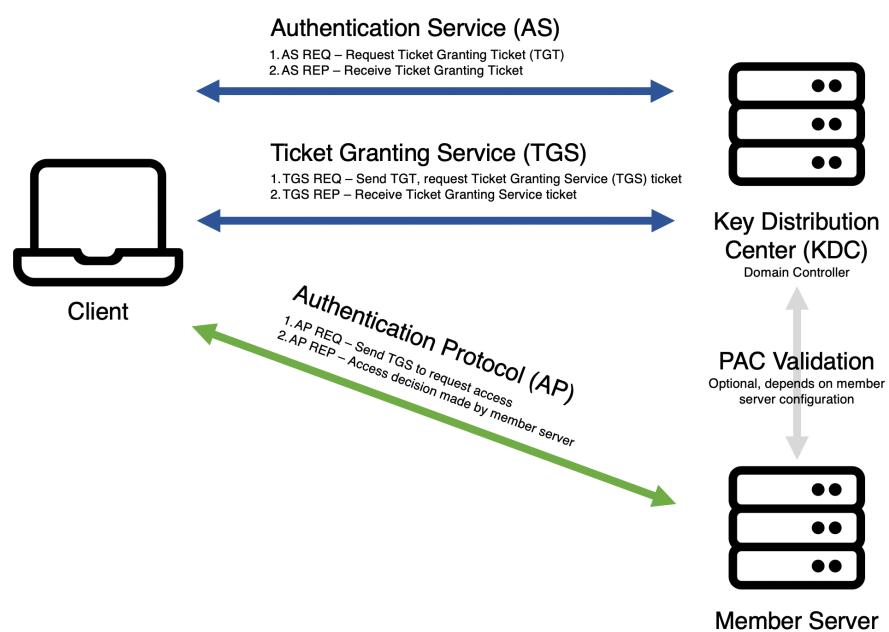


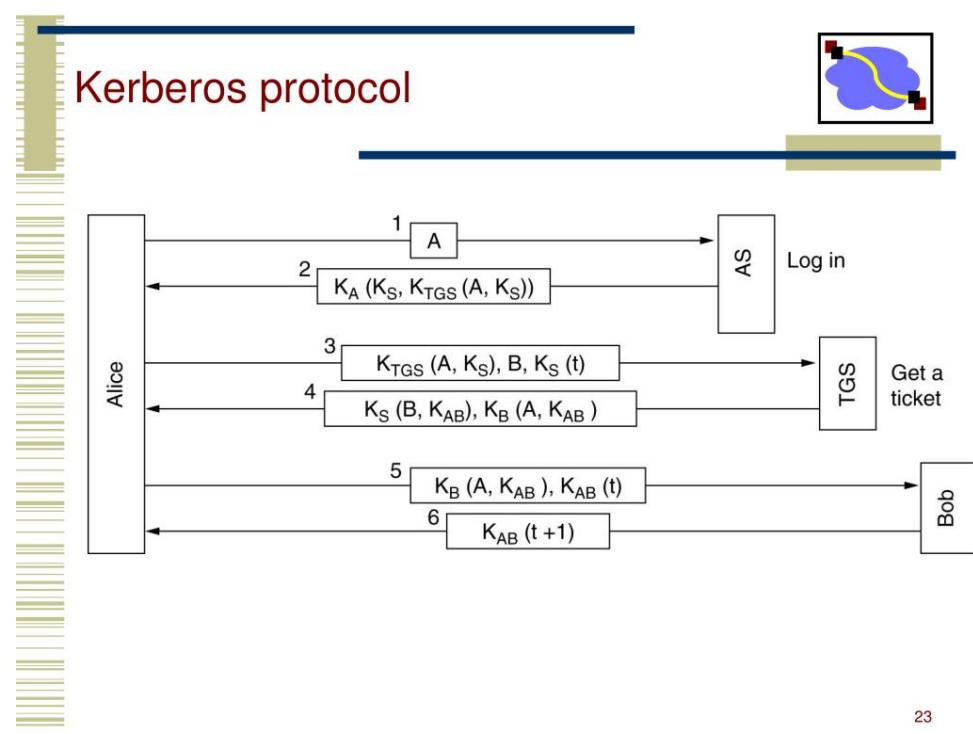
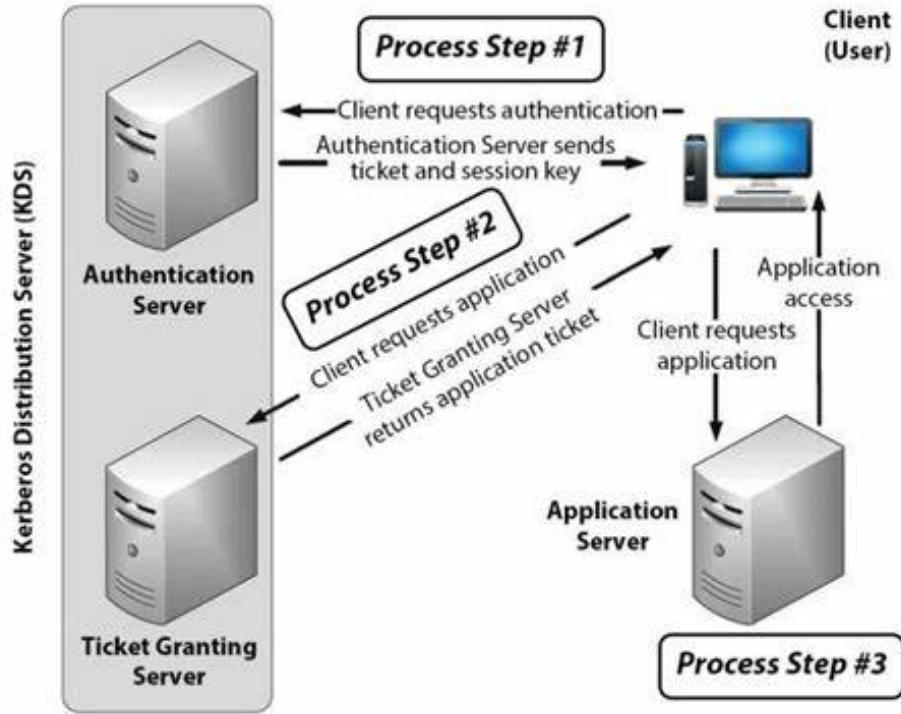
CSMA/CA Collision Handling

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



Kerberos

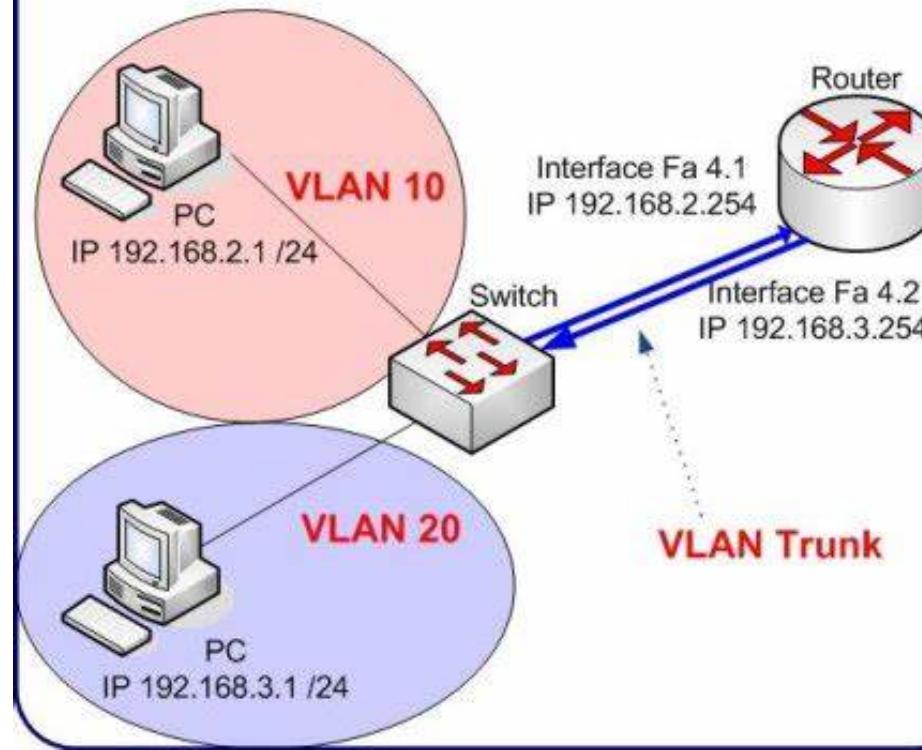




23

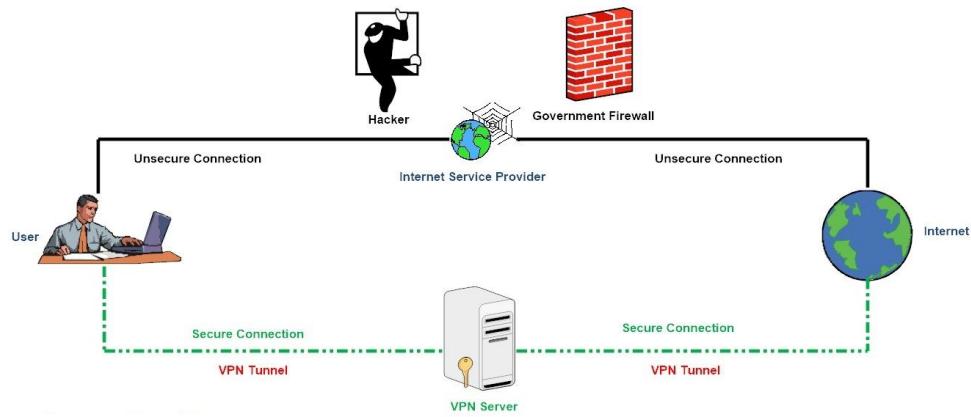
VLAN

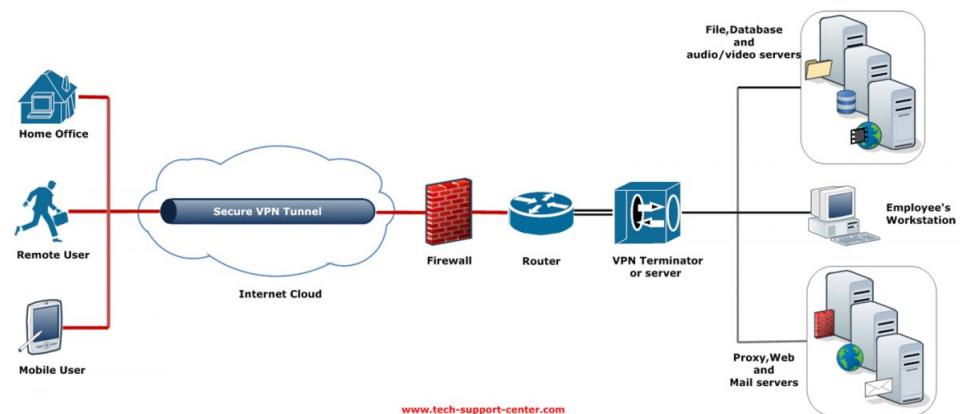
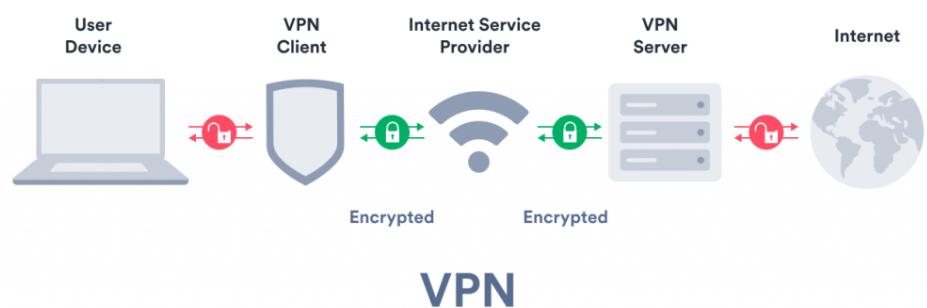
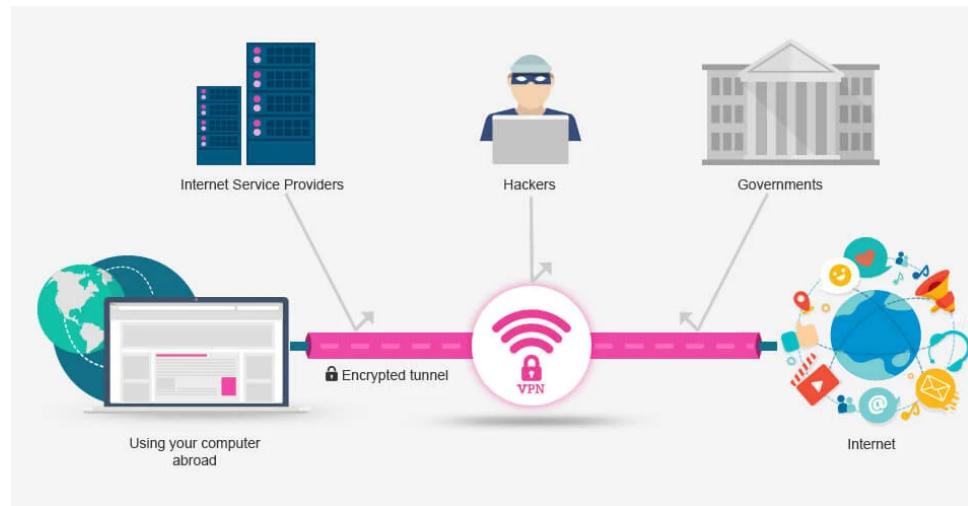
Sample network using VLANs



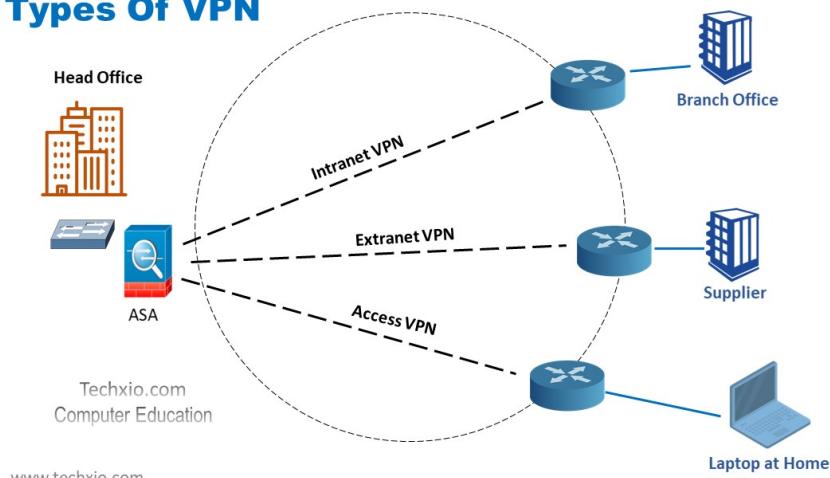
VPN

How VPN Works

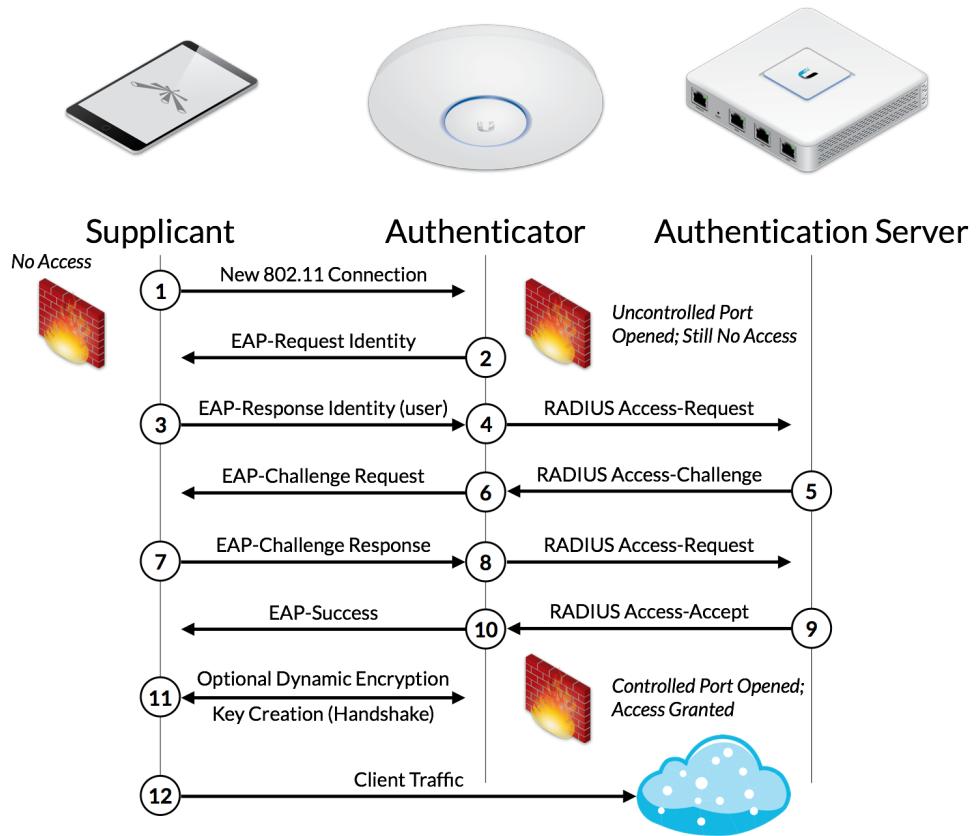




Types Of VPN

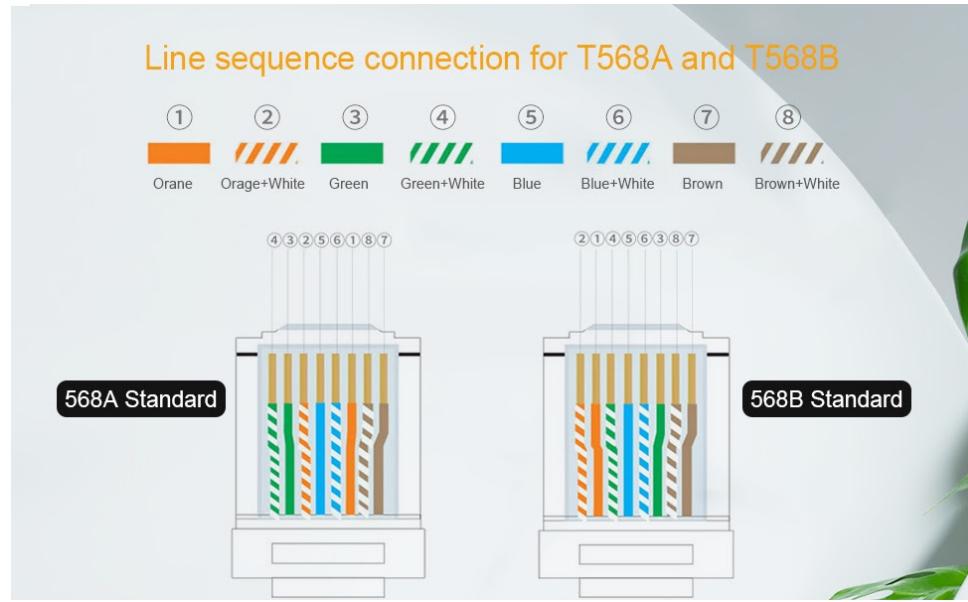


802.1X Authentication (EAP & RADIUS)

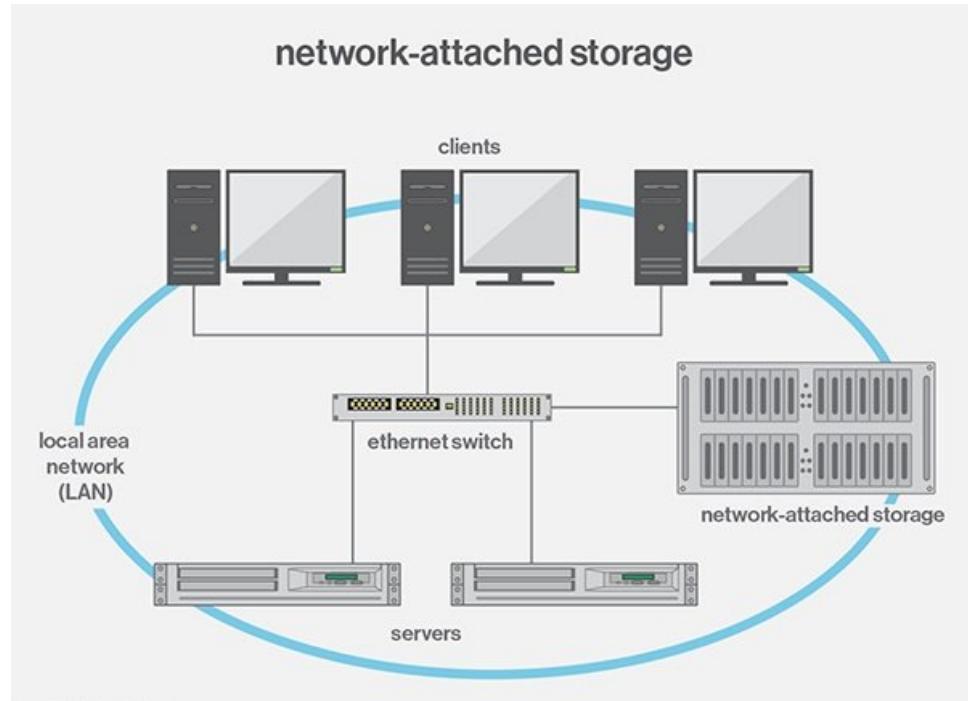


Cable

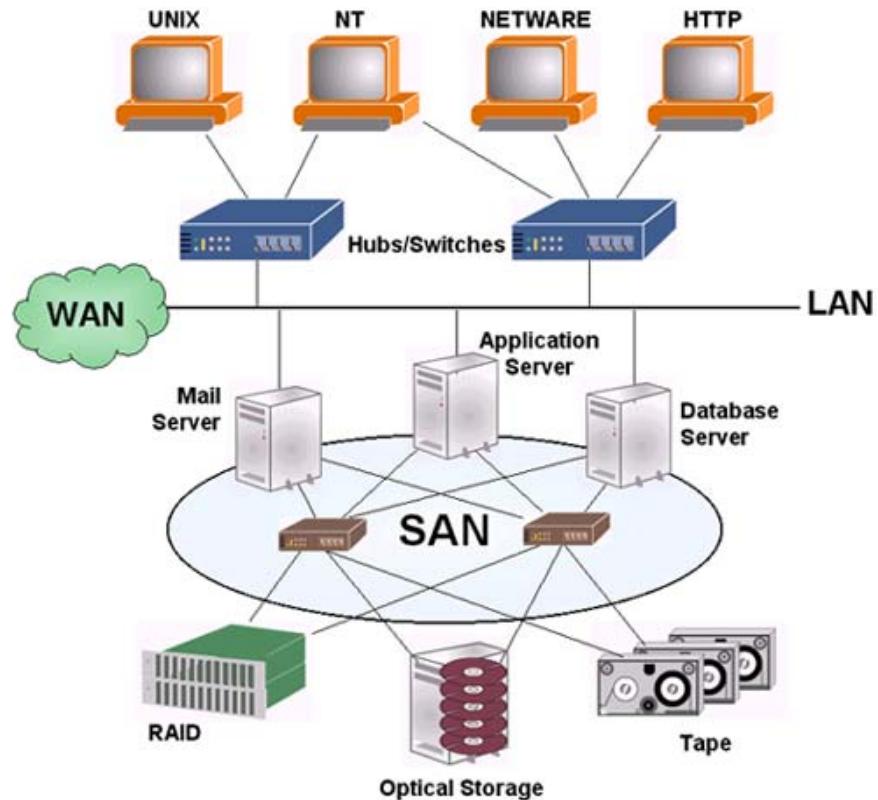
Category	Standard Bandwidth	Max Data Rate	Shielding
Cat5e	100MHz (up to 350)	1000Mbps	UTP or STP
Cat6	250MHz (up to 550)	1000Mbps	UTP or STP
Cat6A	500MHz (up to 550)	10Gbps	UTP or STP
Cat7	600MHz	10Gbps	Shielded only
Cat8	2000MHz	25Gbps or 40Gbps	Shielded only



NAS vs SAN



① Storage Area Networks



Source: allSAN Report 2001

Copyright © 2000 allSAN.com Inc 

SAN components

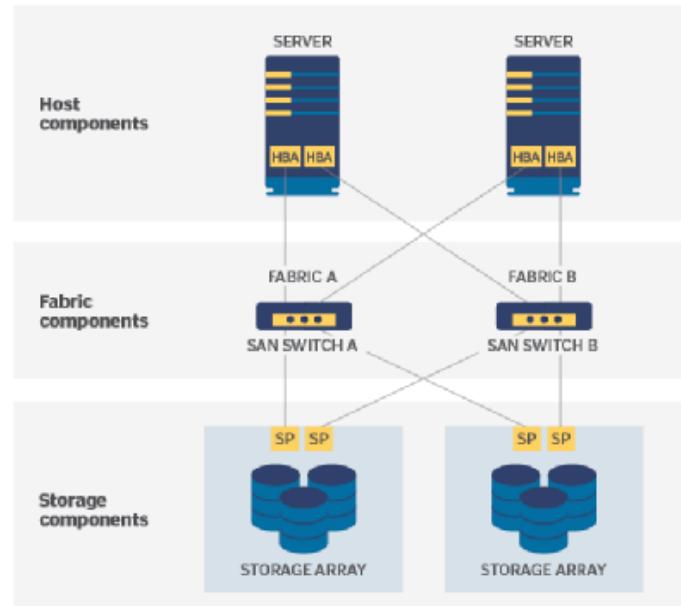
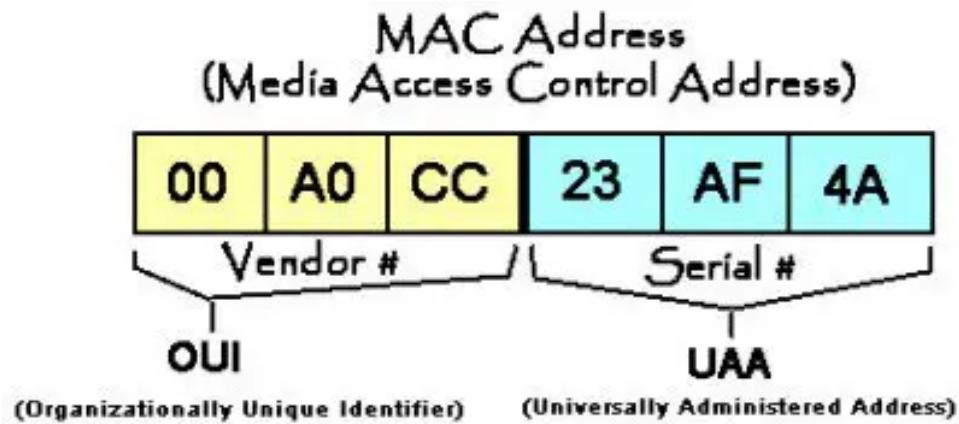


ILLUSTRATION: MAGLARA/ANDOE STOCK

©2006 TECHTARGET. ALL RIGHTS RESERVED.  TechTarget

Datalink Layer MAC Address



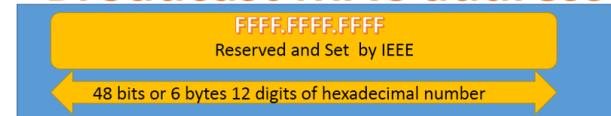
Unicast MAC address



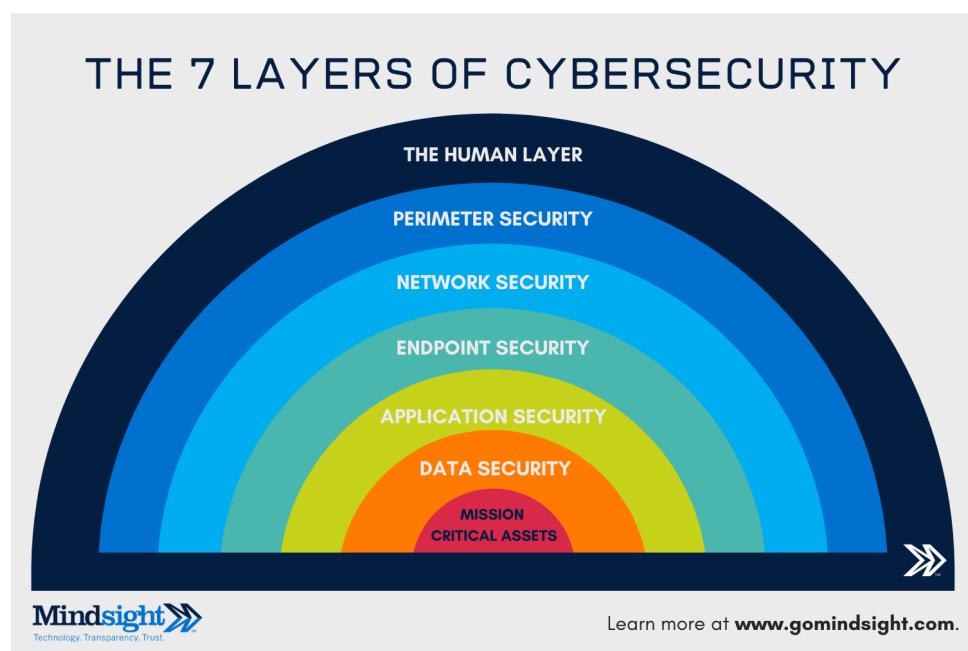
Multicast MAC address



Broadcast MAC address



CyberSecurity

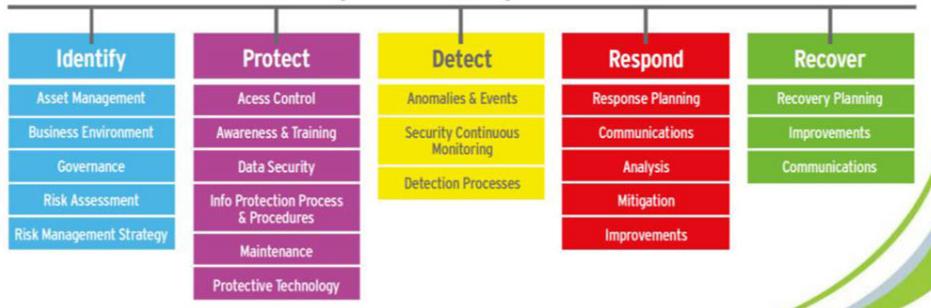


**1 IDENTIFY**

- Identify and control who has access to your business information
- Conduct background checks
- Require individual user accounts for each employee
- Create policies and procedures for cybersecurity

**2 PROTECT**

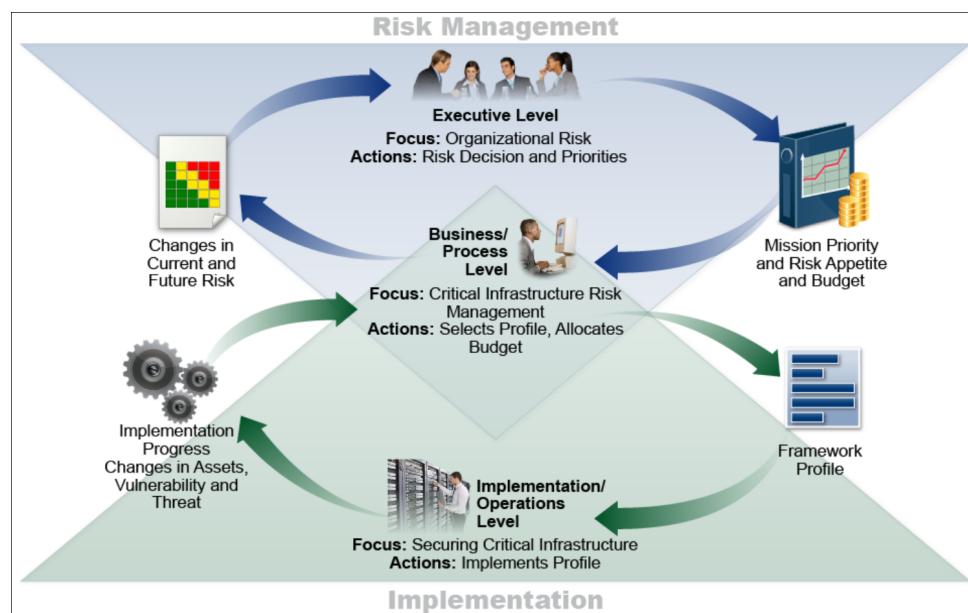
- Limit employee access to data and information
- Install Surge Protectors and Uninterruptible Power Supplies (UPS)
- Patch your operating systems and applications routinely
- Install and activate software and hardware firewalls on all your business networks
- Secure your wireless access point and networks
- Set up web and email filters
- Use encryption for sensitive business information
- Dispose of old computers and media safely
- Train your employees

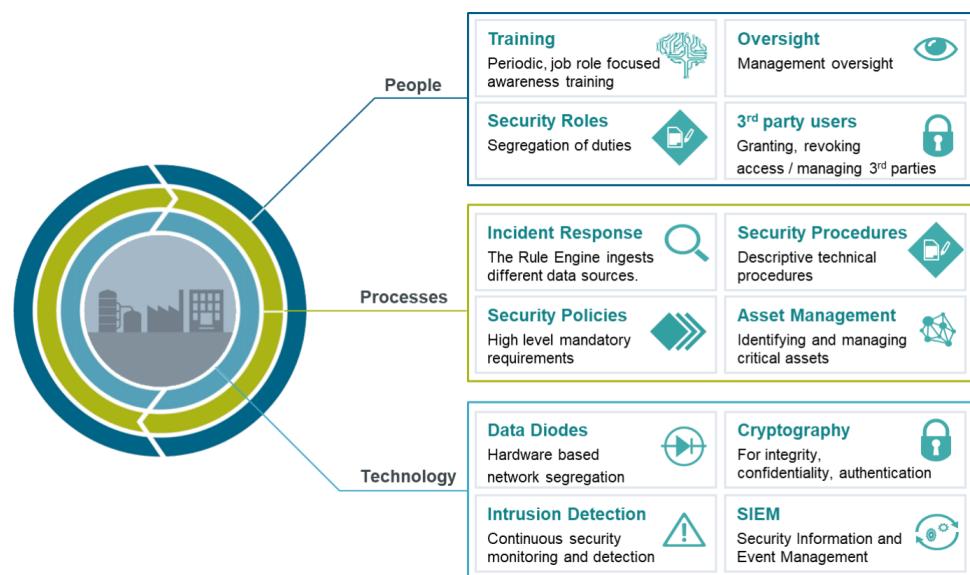
**NIST Cybersecurity Framework****NIST Cyber Security Framework**

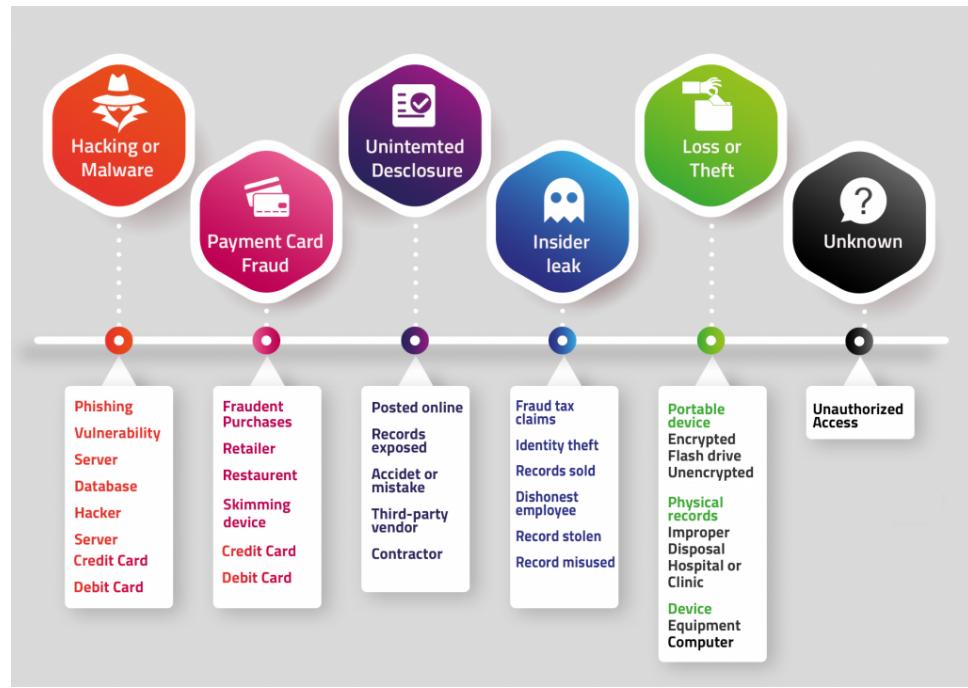


Risk Management

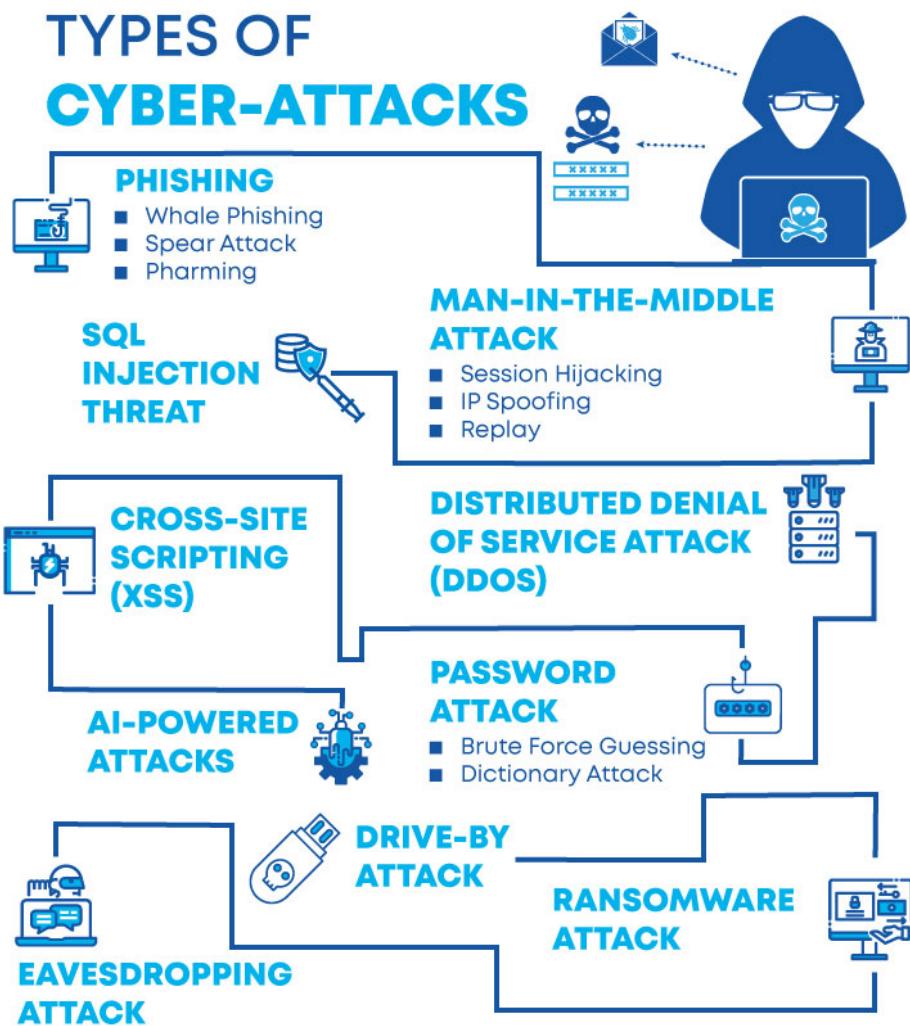
BEST CYBERSECURITY PRACTICES

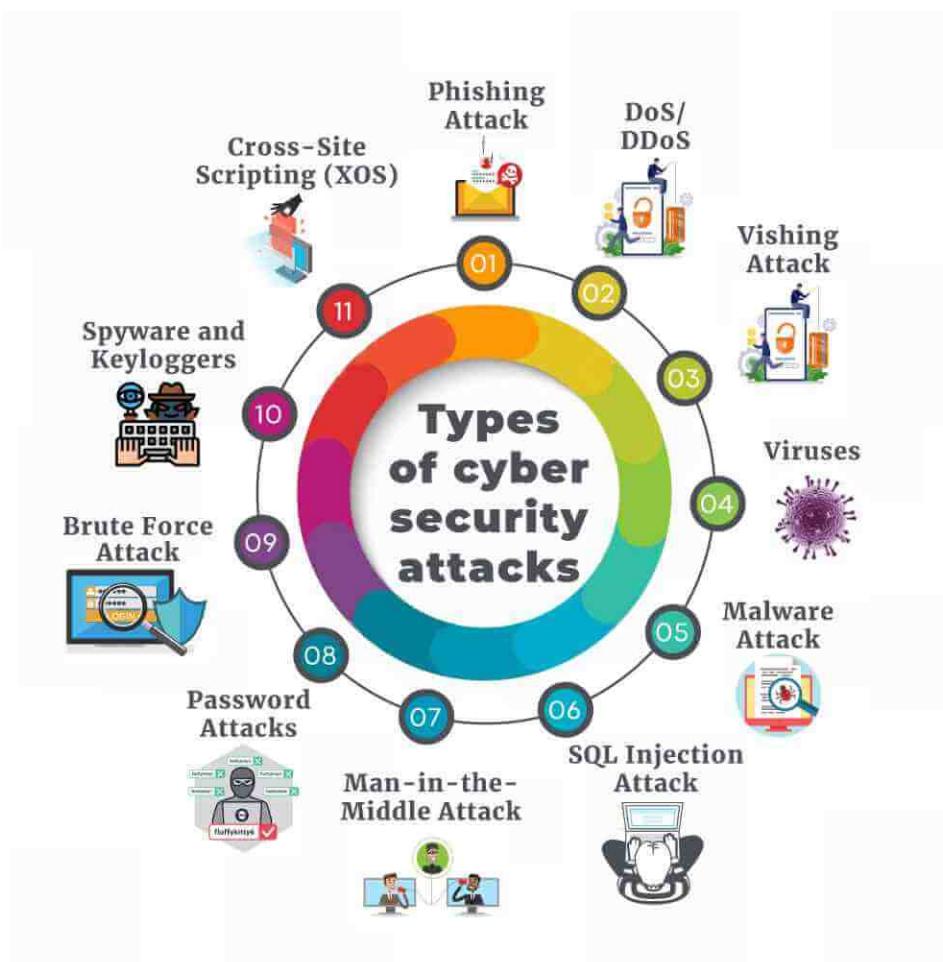
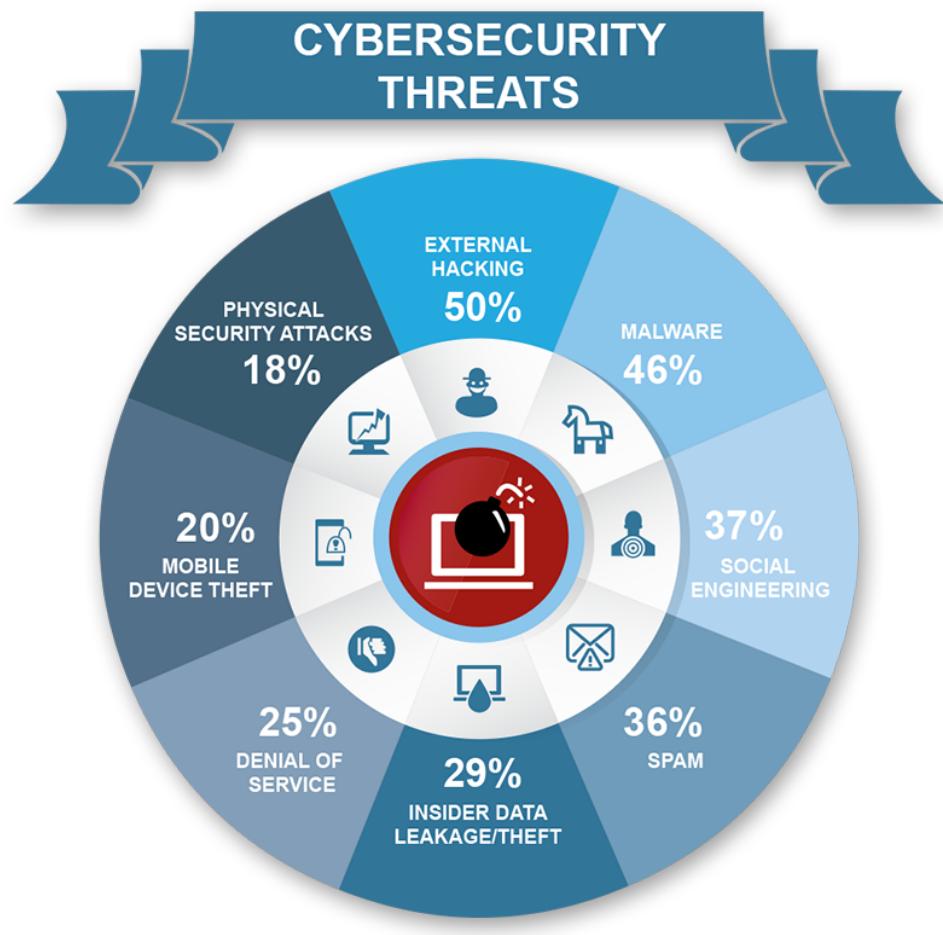




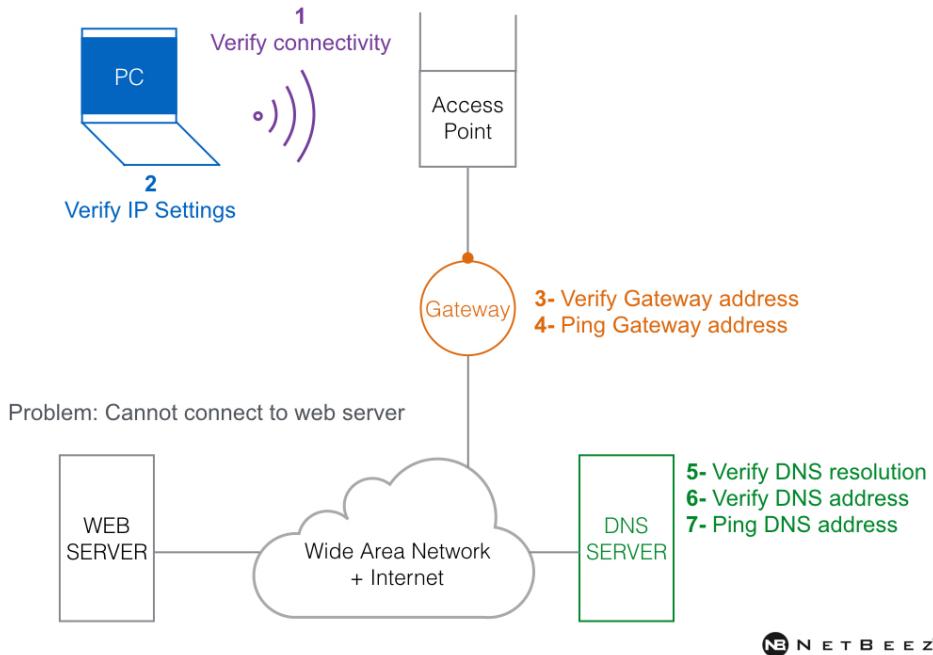


Cybersecurity Attacks

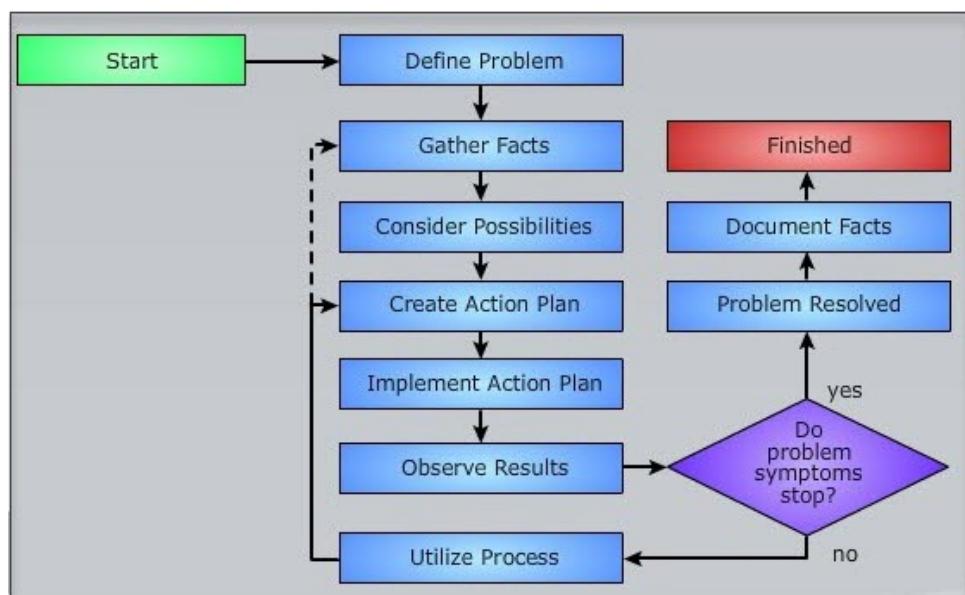
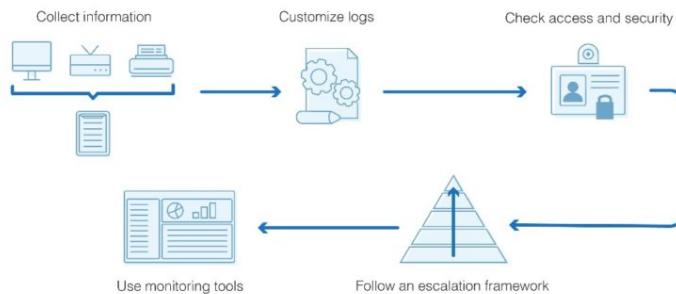




Network Troubleshooting



Network Troubleshooting Flowchart



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



CompTIA Network+
PowerCert

Troubleshooting *STRATEGY*



PowerCert

PowerCert

Icon of a computer monitor with a network cable.

7. Identify the results and effects of the solution.

- ✓ Has your plant of action solved the problem or not?
- ✓ What effect did it have on everyone else?
- ✓ Do the results show a temporary fix or a permanent one?

CompTIA Network+
PowerCert

Troubleshooting *STRATEGY*



PowerCert

PowerCert

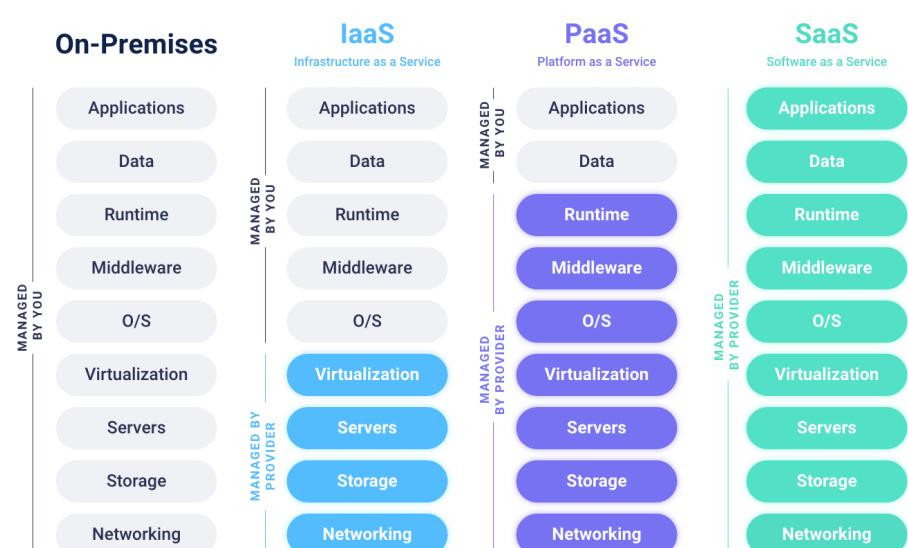
Icon of a computer monitor with a network cable.

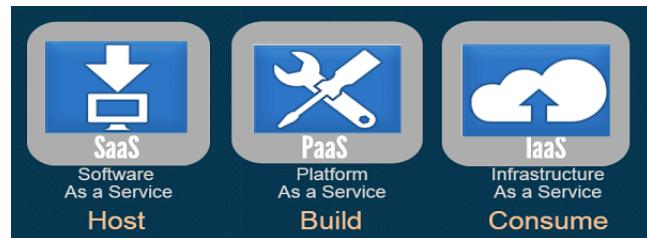
8. Document the solution and process.

- ✓ Document the problem.
- ✓ Document what caused the problem.
- ✓ Document how the problem was fixed.

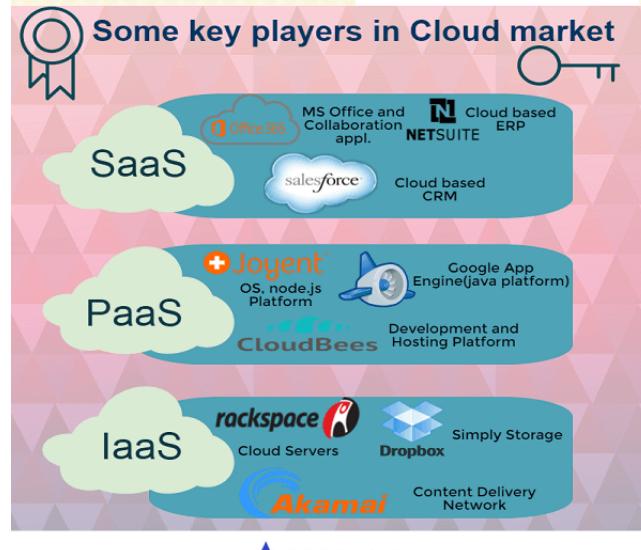
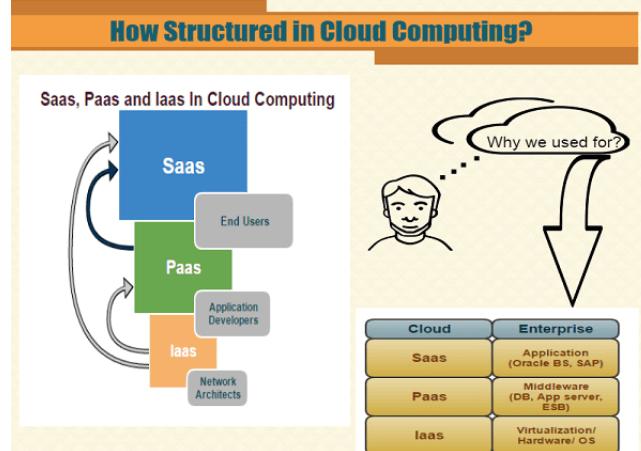
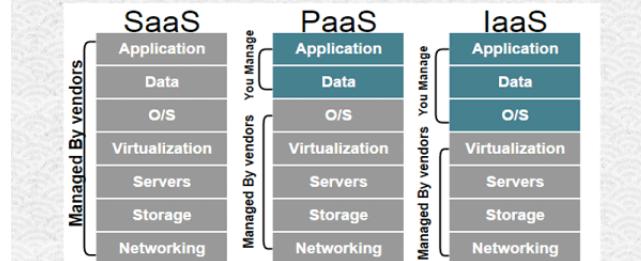
Network Administrator

Cloud Computing - IaaS Paas Saas





Difference between SaaS, PaaS and IaaS



-- Memo End --