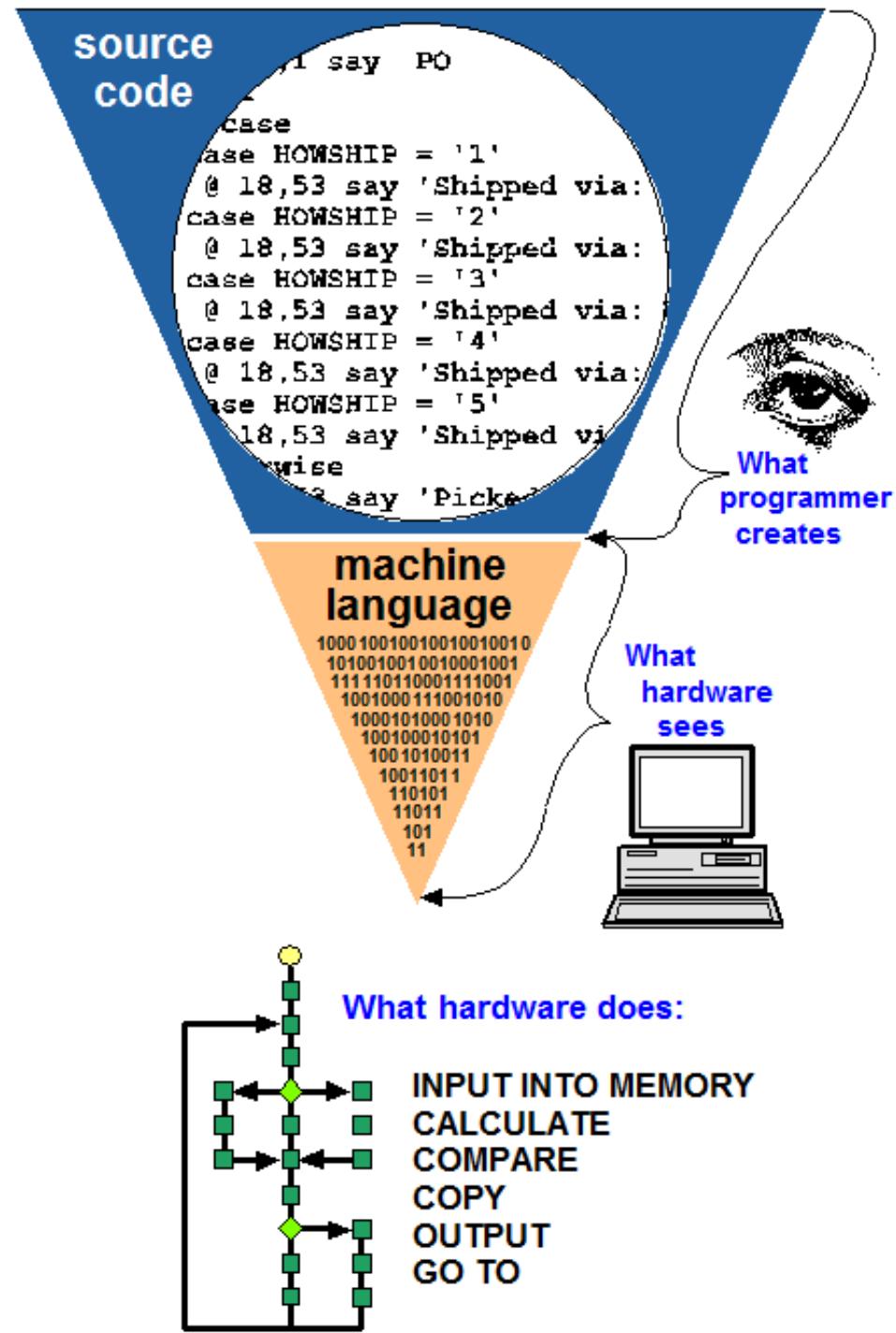


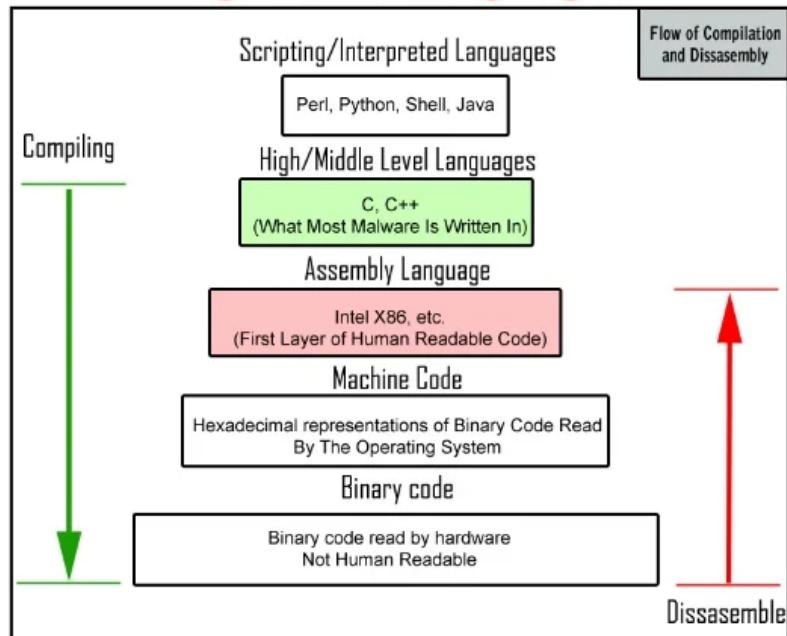
Computer Science

Programming Language -> Communicate with Computer

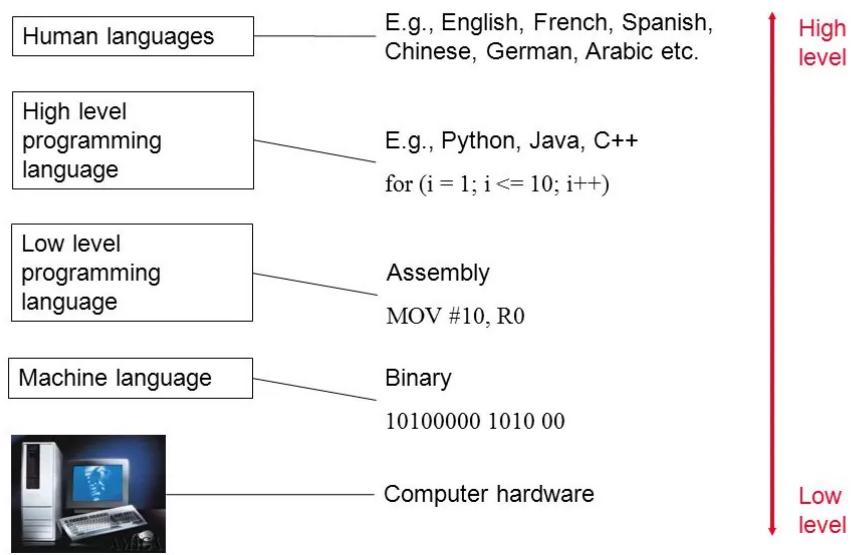
SOURCE CODE TO MACHINE LANGUAGE



High Level Languages

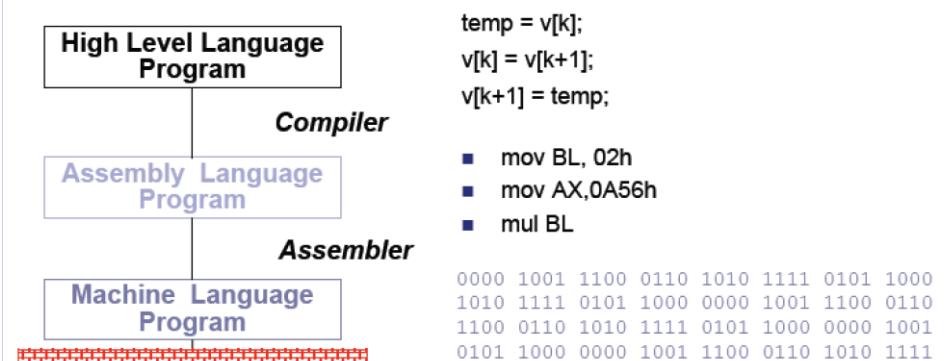


High Vs. Low Level Languages



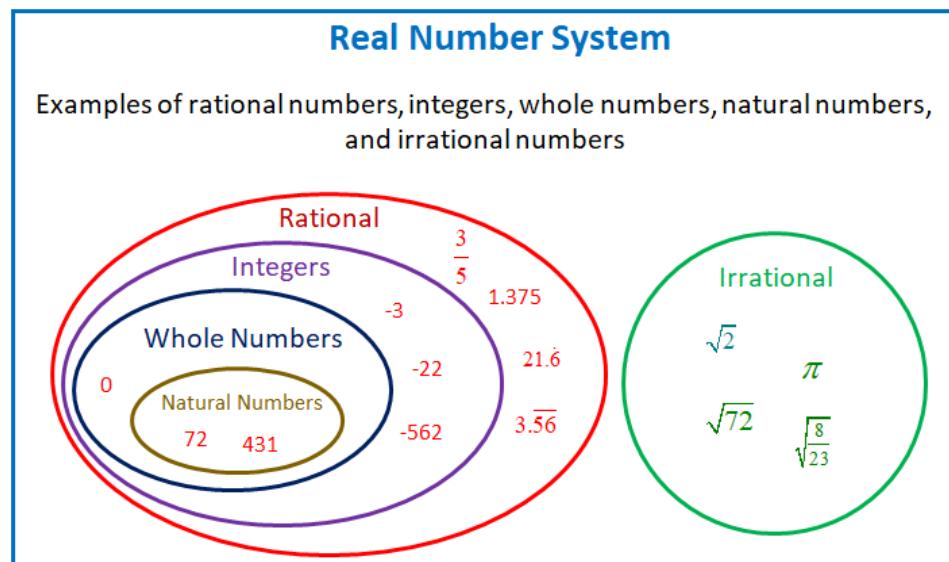
James Tam

Levels of Representation



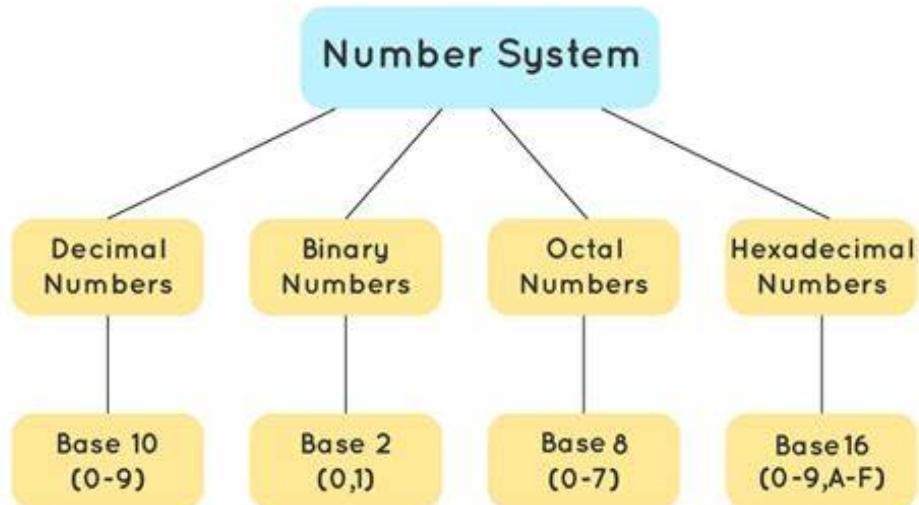
Number System

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

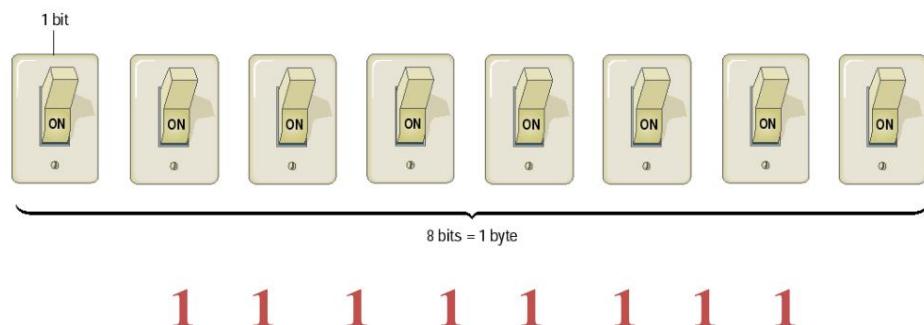
- Convert from decimal to binary
Divide the decimal by the largest binary weight it is divisible by and place a "1" in that column. Then select the next largest weight, if it is divisible put a "1" in that column otherwise place a "0" in the column. Continue until all the columns have either a "1" or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

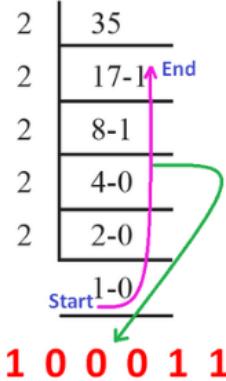
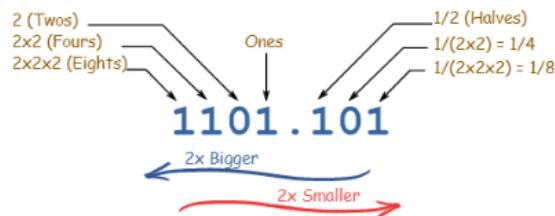
Number System

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

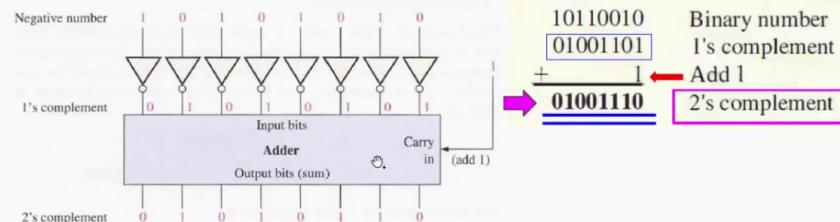
2.5) 1'S and 2'S Complements of Binary Numbers

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.

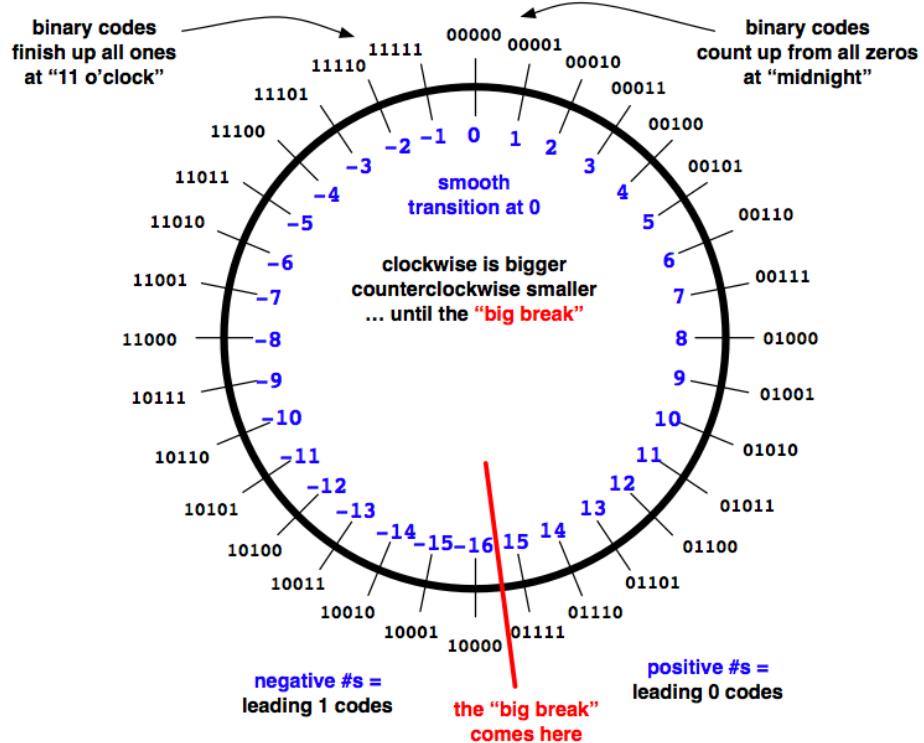
$$\text{2's complement} = (\text{1's complement}) + 1$$

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 $(10^n - 1) - N$

- $99999\dots\dots - N$

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

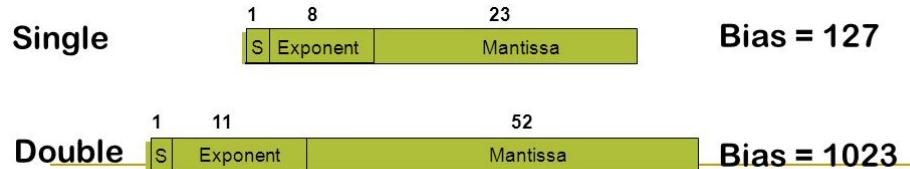
Floating point numbers – the IEEE standard

■ 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)

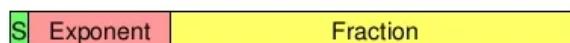
- Representation is called sign and magnitude

◊ 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

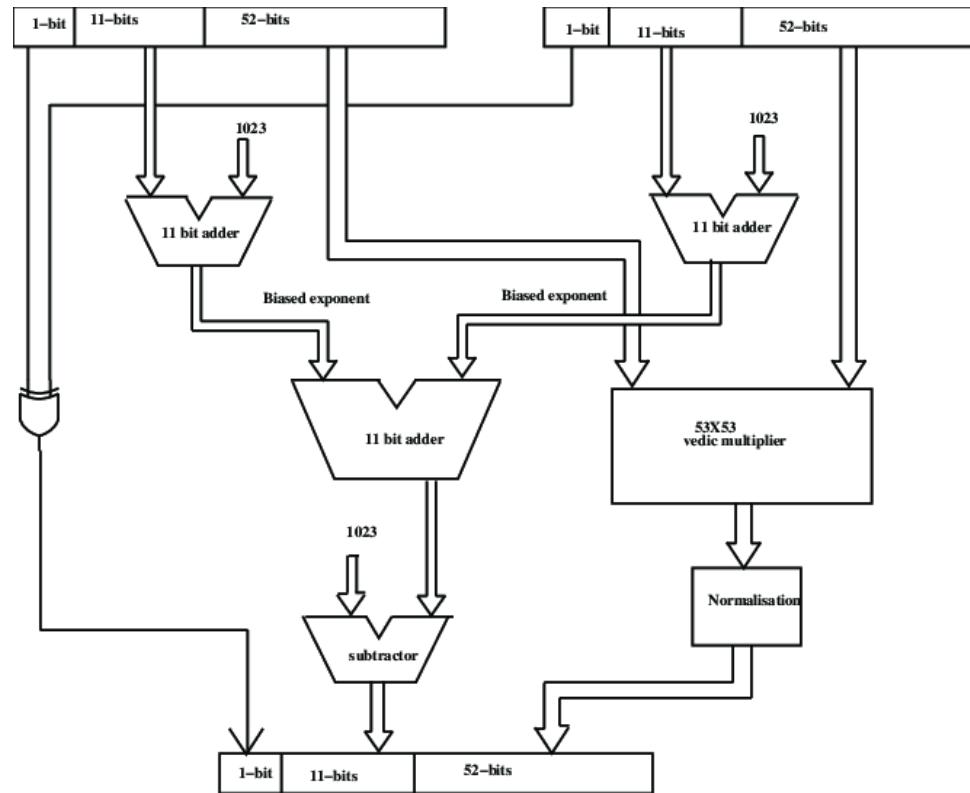


Value of a floating-point number = $(-1)^S \times \text{val}(F) \times 2^{\text{val}(E)}$

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	™	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	◊	240	≡						
17	DC1 (Device control 1)	49	1	81	Q	113	q	145	æ	177	„	209	¤	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	ö	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	ð	182	Á	214	—	246	÷						
23	ETB (End of trans. block)	55	7	87	W	119	w	151	ú	183	À	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	Ó	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



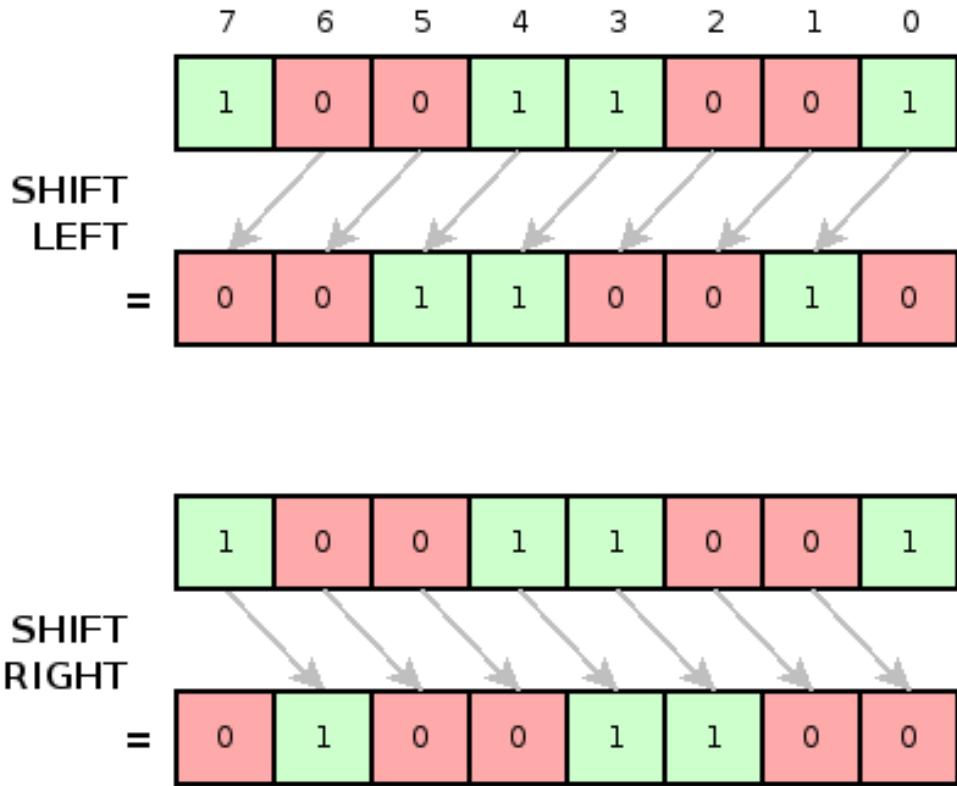
01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

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



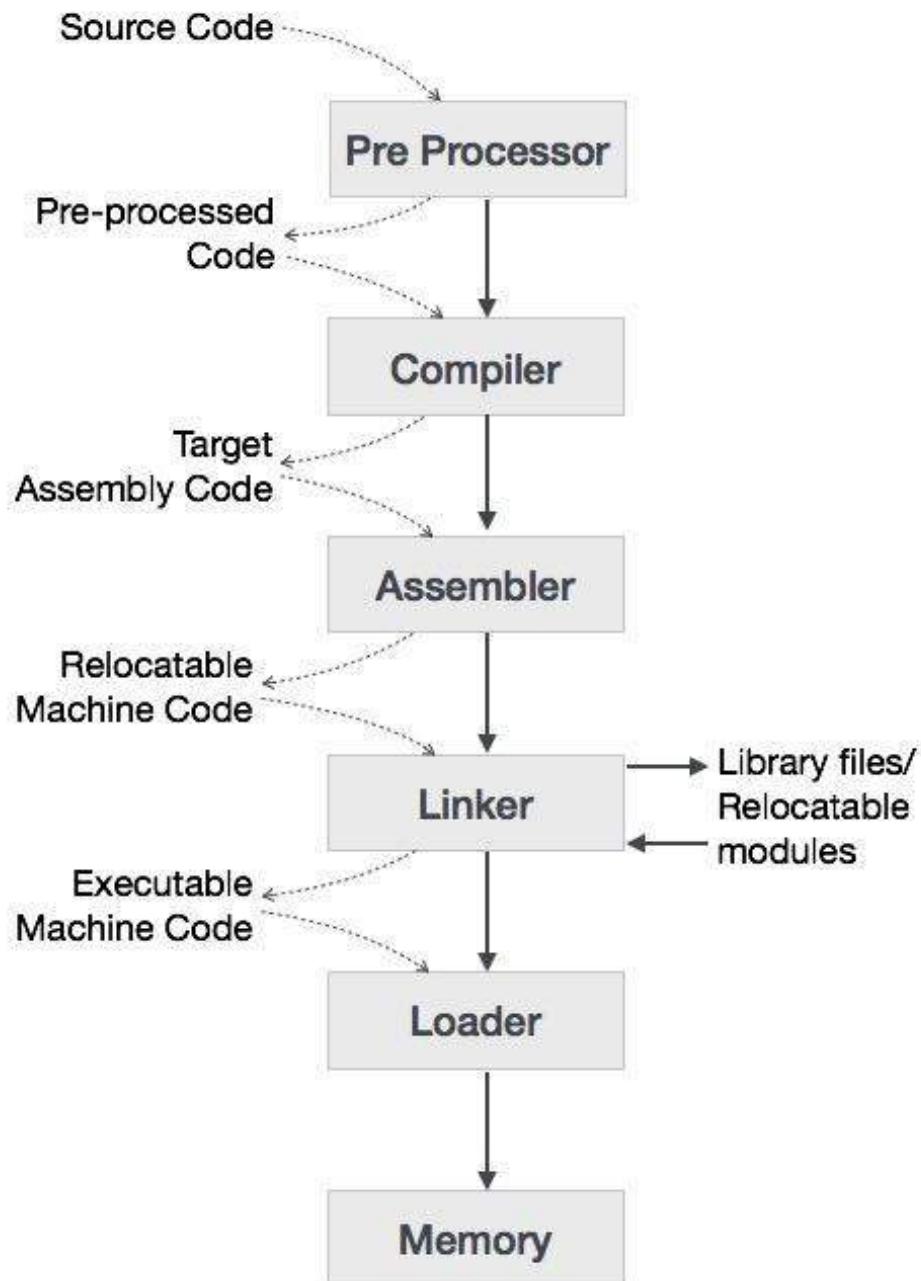
High level language

- 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.*

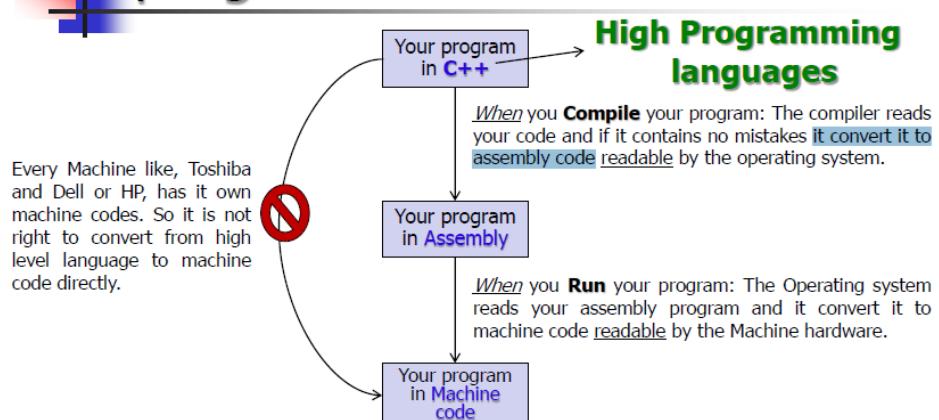


- For example, in the instruction Add A and B, Add is the opcode and A and B are operands

Compiler and Assembler



Compiling and running your program



LINKER VS LOADER VS COMPILER

LINKER	LOADER	COMPILER
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	A software that transforms computer code written in one programming language into another programming language
Combines multiple object code and links them with libraries	Prepares the executable file for running	Transforms the source code into object code

Visit www.pediaa.com

Logic Design

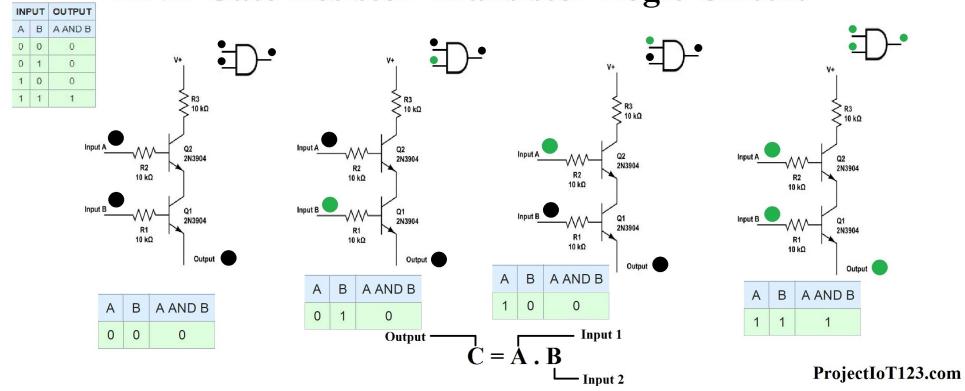
- Logic Gates

 Buffer $F = A$	 AND $F = AB$	 OR $F = A+B$	 XOR $F = A \oplus B$																																																			
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>F</th></tr> <tr> <td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td></tr> </table>	A	F	0	0	1	1	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
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																																																				

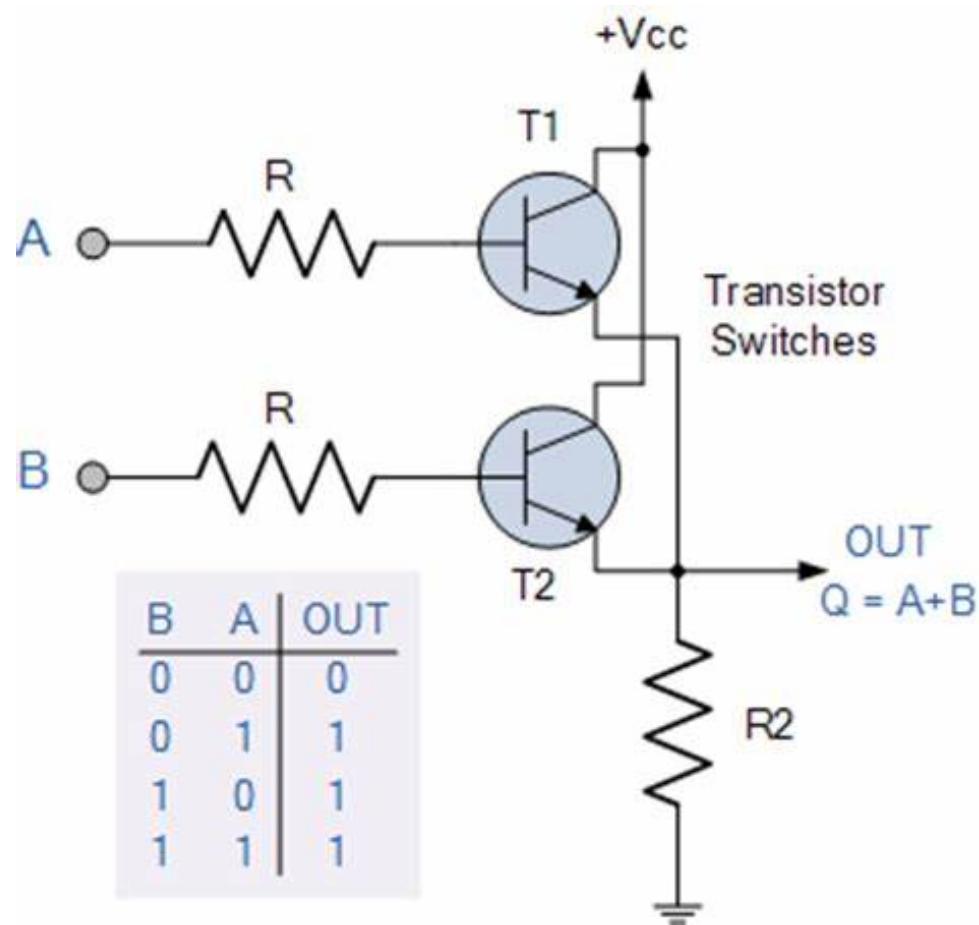
 Inverter $F = \bar{A}$	 NAND $F = \bar{AB}$	 NOR $F = \bar{A}+\bar{B}$	 XNOR $F = \bar{A} \oplus \bar{B}$																																																			
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>F</th></tr> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </table>	A	F	0	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>A</th><th>B</th><th>F</th></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	1
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



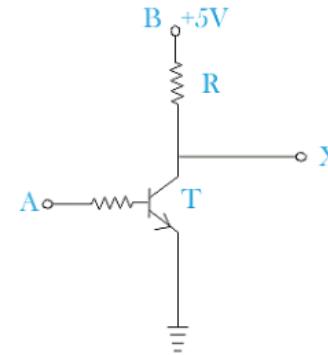
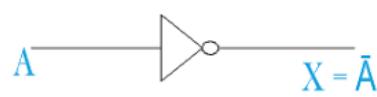
- OR Gate



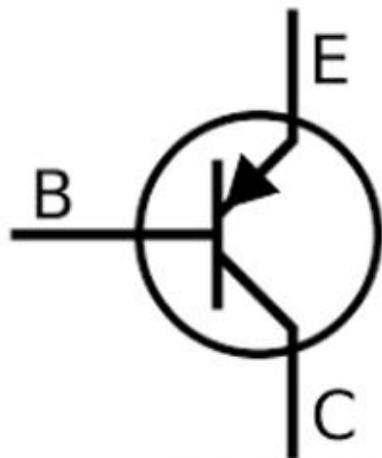
- NOT Gate

What is a NOT Gate?

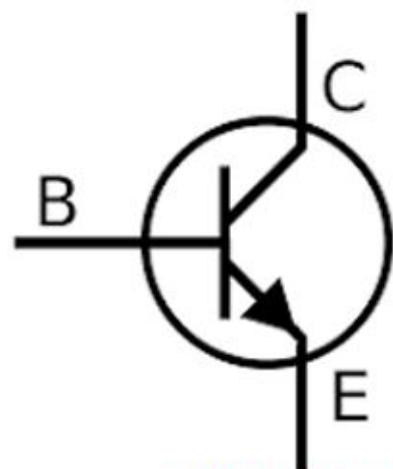
A	X = \bar{A}
0	1
1	0



Electrical 4 U



PNP BJT



NPN BJT

- XOR Gate

XOR GATE Truth Table



BOOLEAN EXPRESSION

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

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

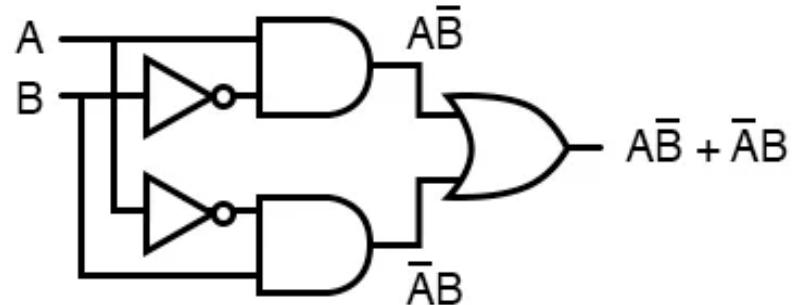
Output **C = A \oplus B** **Input1**
 | **Input2**

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

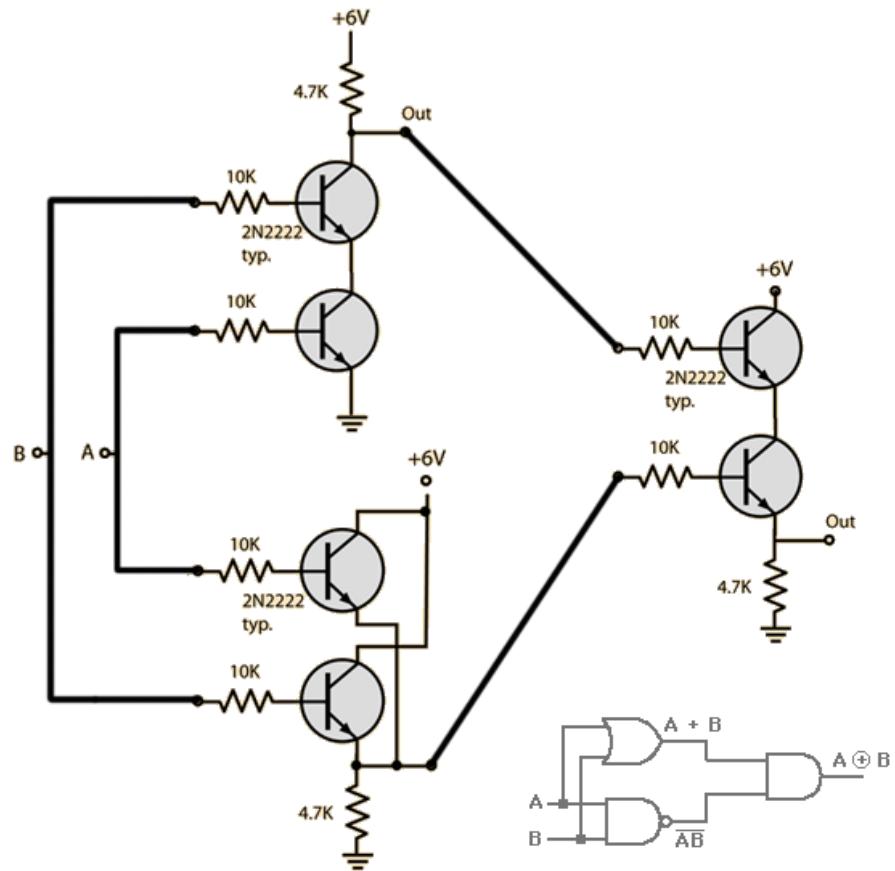
ProjectIoT123.com



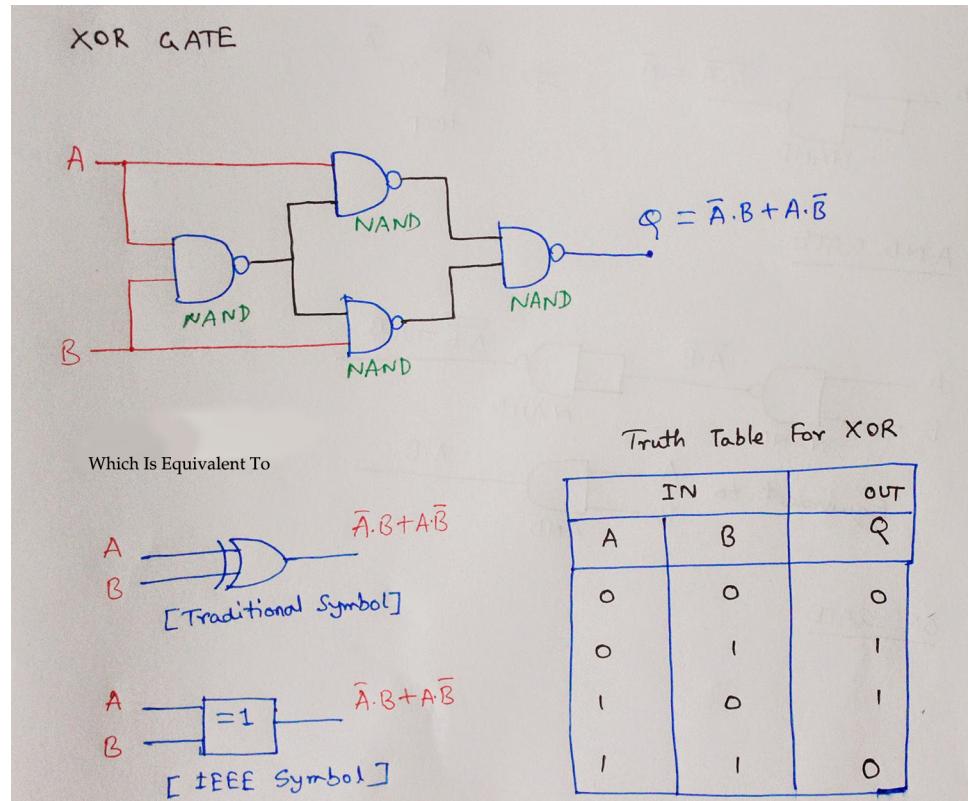
. . . is equivalent to . . .



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

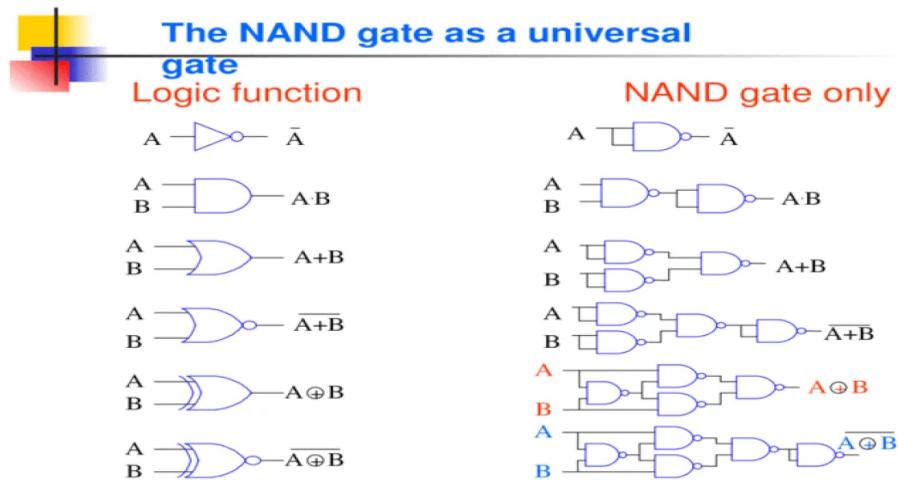
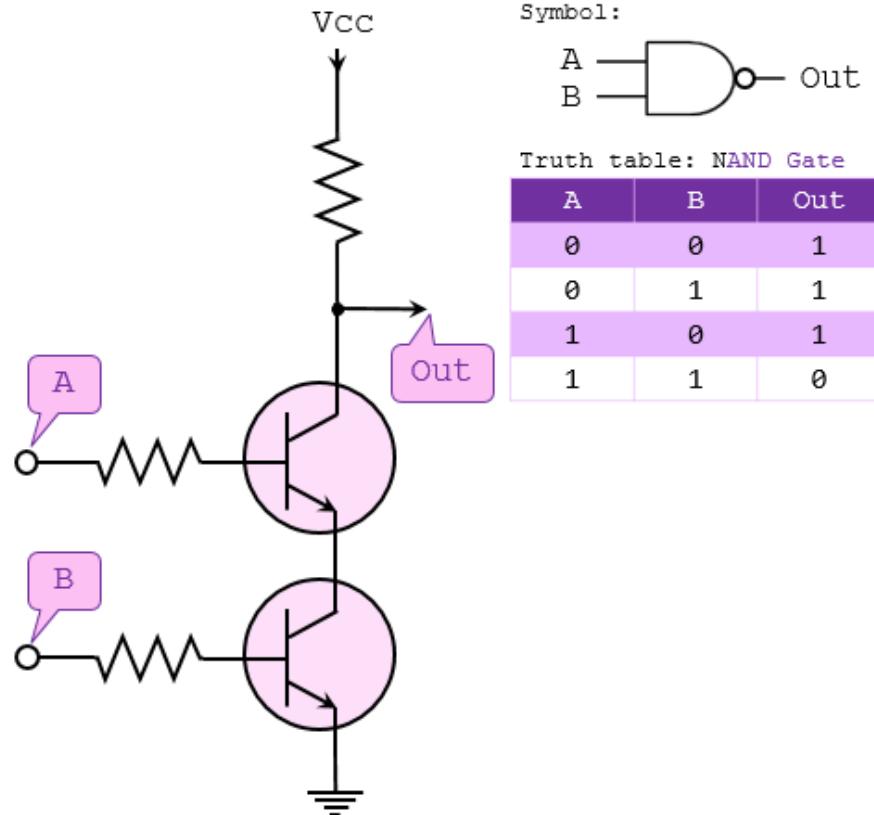


- **XOR Gate**



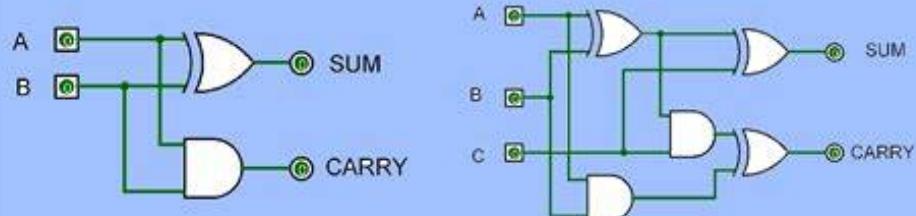
- **NAND Gate**

NAND Gate

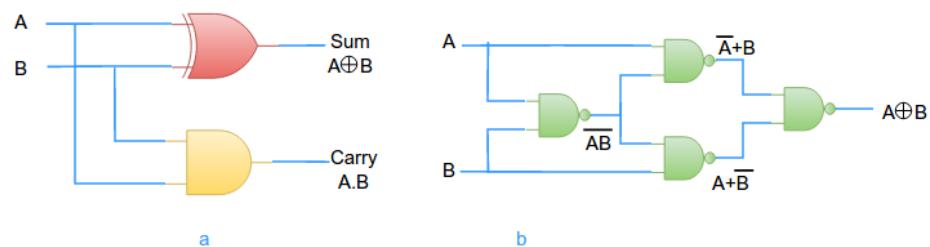
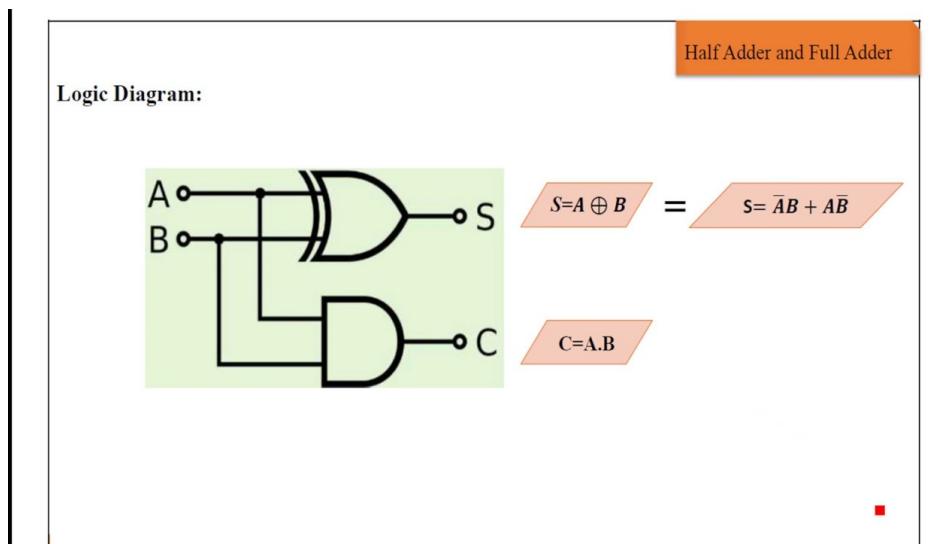


- Hal-Adder vs Full-Adder

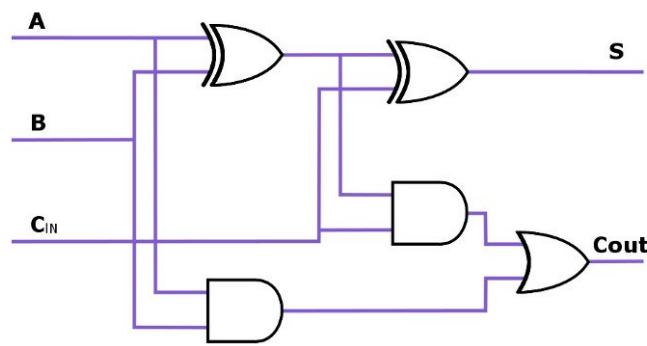
Difference Between Half Adder And Full Adder



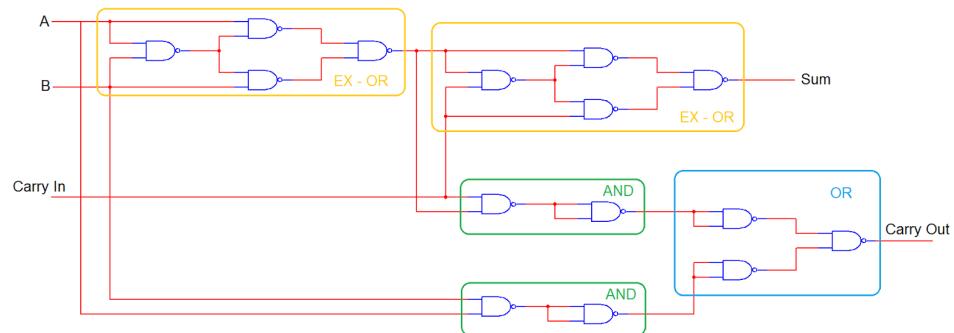
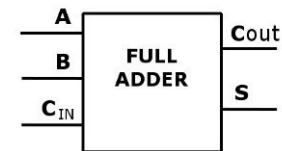
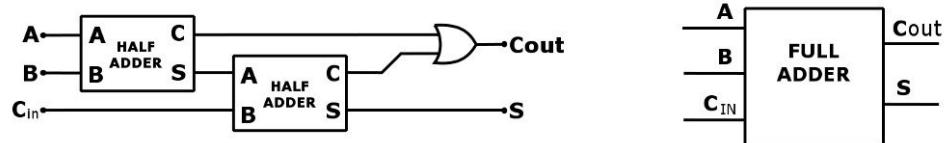
- **Half-Adder**



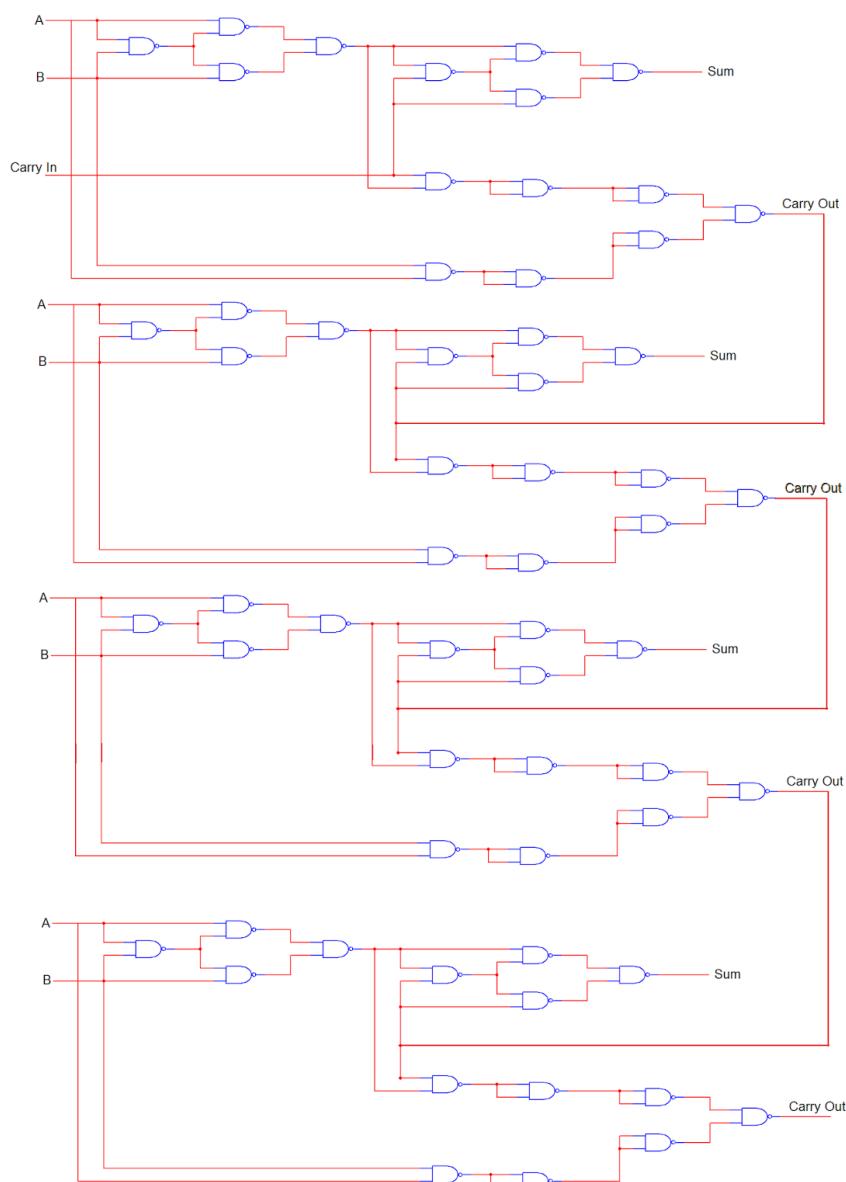
- **Full Adder**



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**

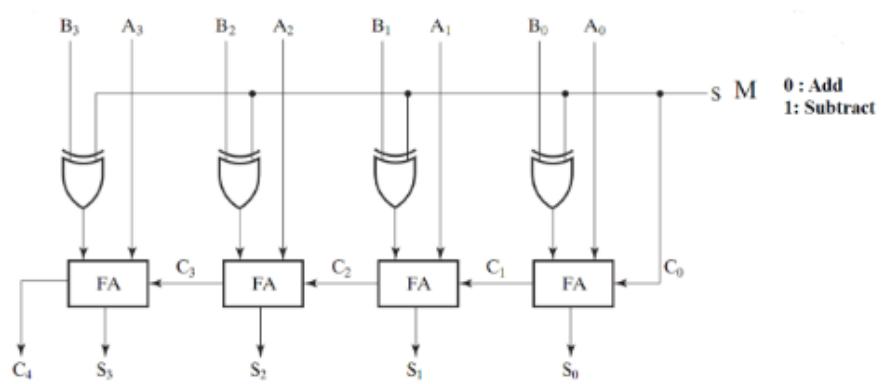
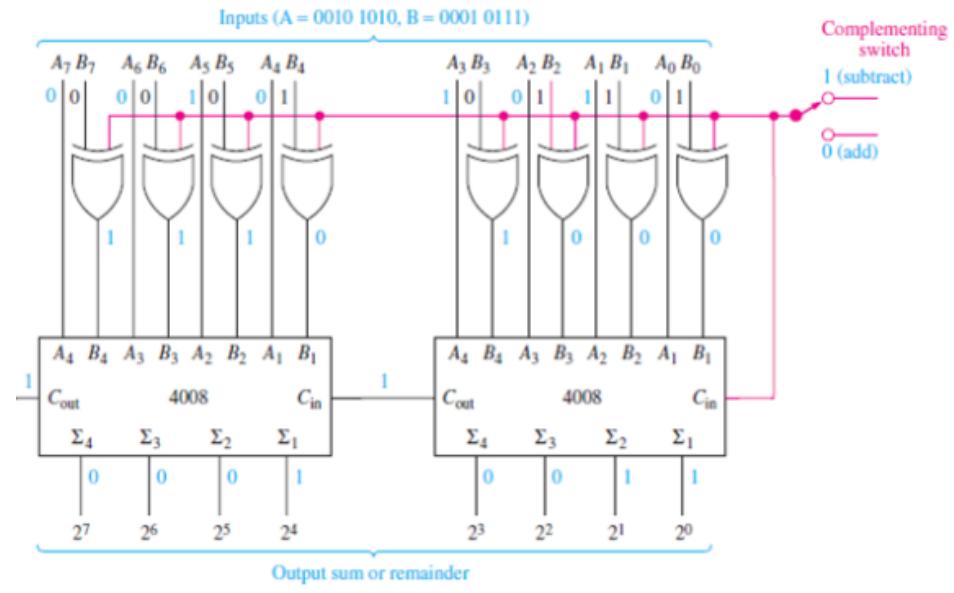


Fig. 1 Modular design of 4-bit adder/subtractor

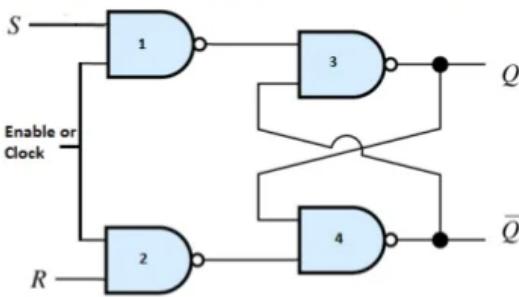


Circuits as Memory

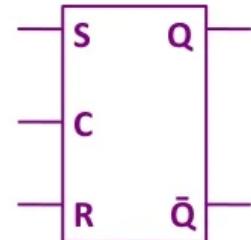
- SR Filp Fop



Gated Latch-Clocked RS Flip-flop

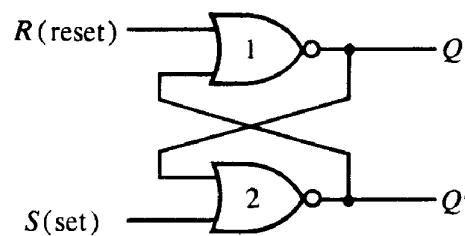


A clocked SR flip-flop.

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

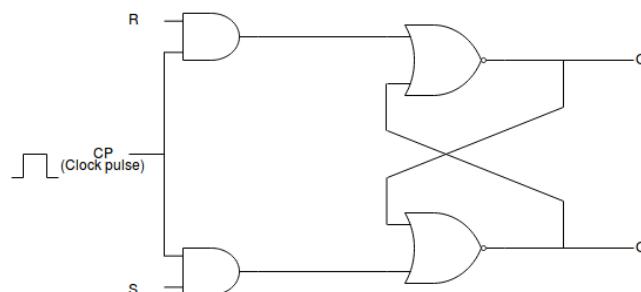


(a) Logic diagram

S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

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

(b) Truth table



a) Logic diagram

fig: Clocked SR flip flop

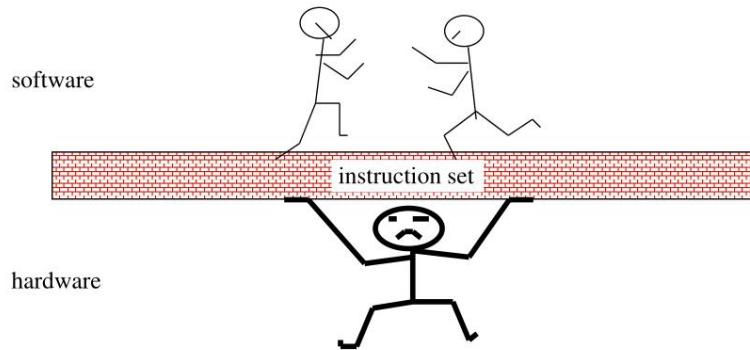
Q	S	R	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Intermediate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Intermediate

b) Truth table

Computer Organization and Architecture

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

Instruction Set Architecture: Critical Interface

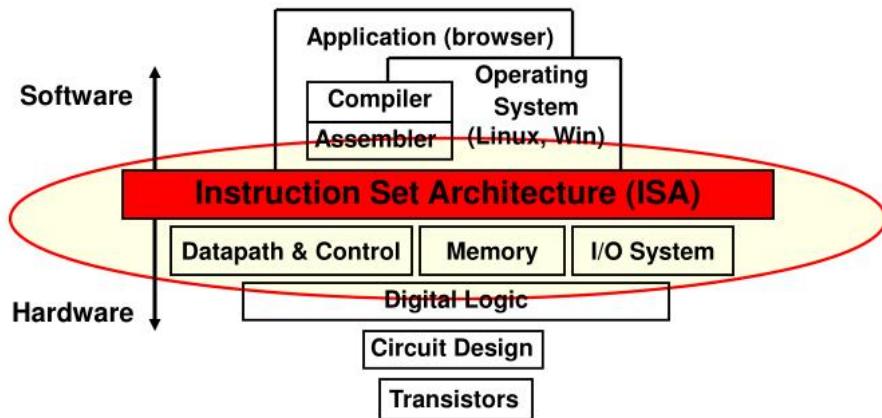


Properties of a good abstraction

- ◆ 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).

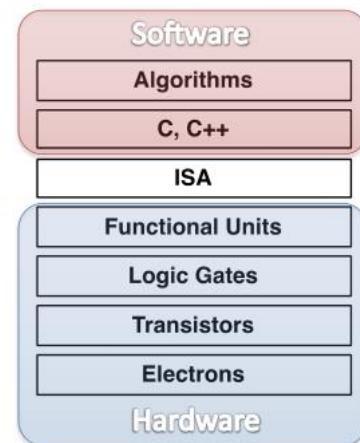
BYU CS 224

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

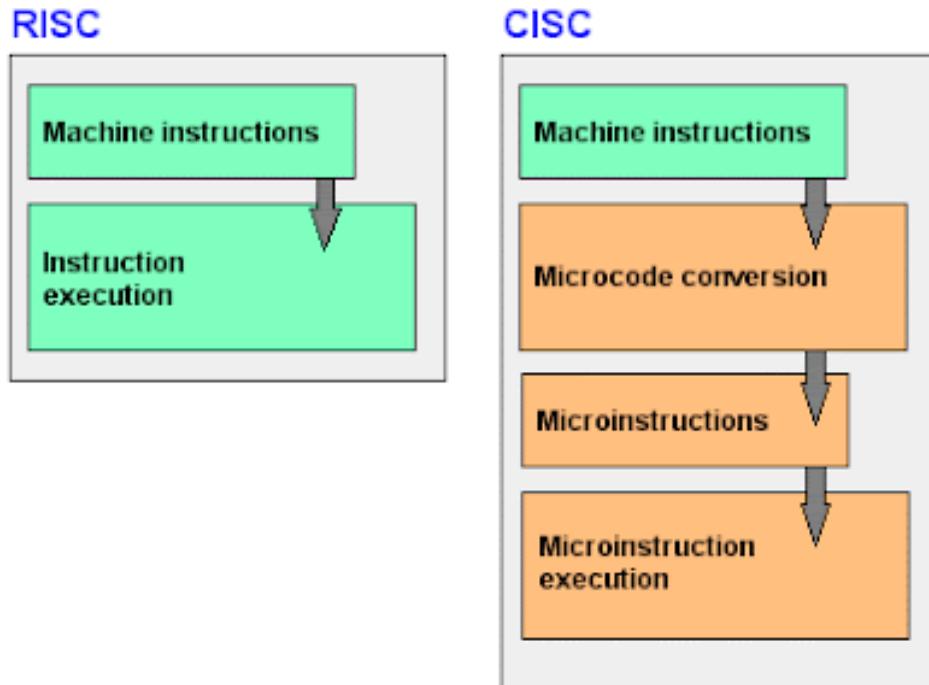
Complex Instruction Set Computer

- 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

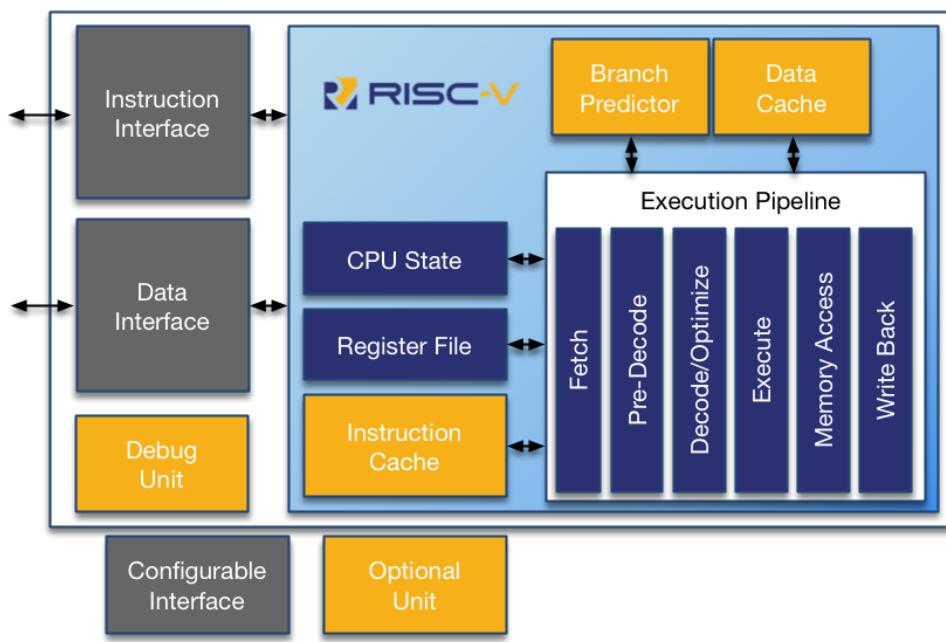
- Few instructions
 - e.g., 30-40
- Smaller chip, smaller pin count, & very low-power consumption
 - Simple but fast instructions
- Harvard architecture, instruction pipelining
- Industry trend for microprocessor design
 - e.g., Intel Pentium, PIC

24



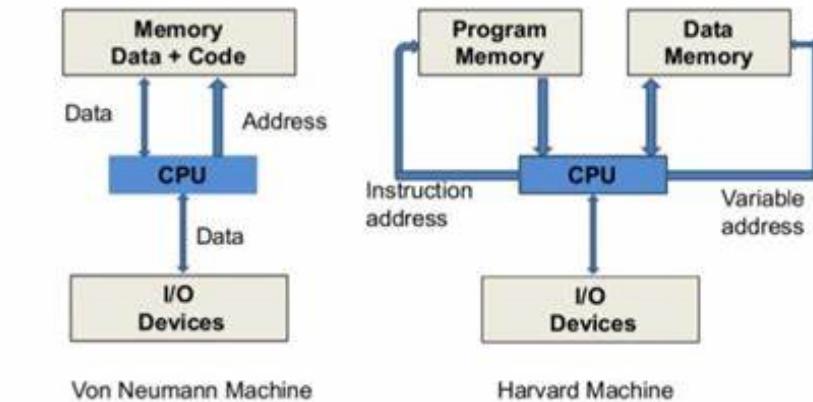
32-bit RISC-V Instruction Formats																																
Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register/register	func7							rs2				rs1				func3			rd			opcode										
Immediate	imm[11:0]							rs1				func3			rd			opcode		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

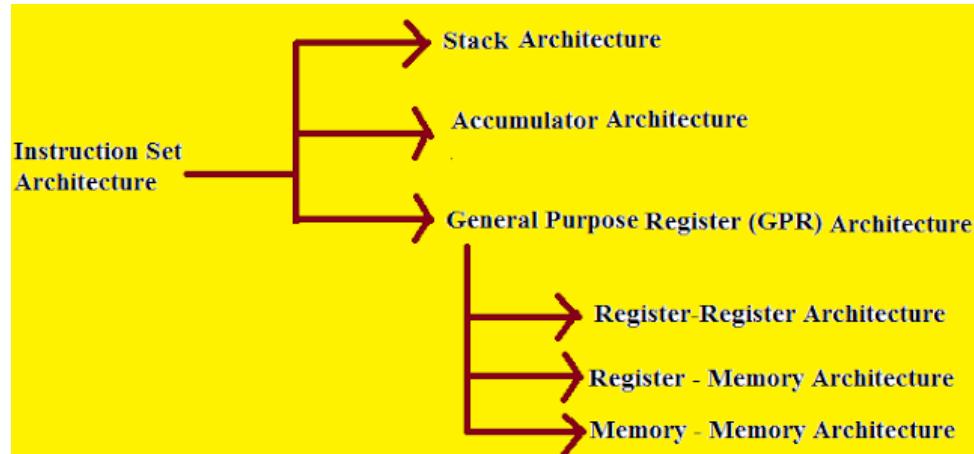
ISA: STORAGE RESOURCES

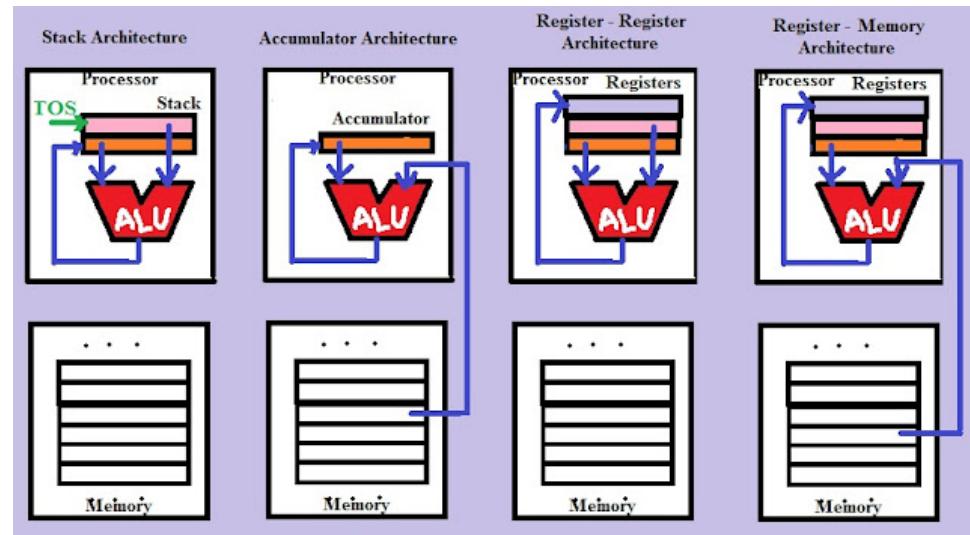
- "Harvard architecture": Separate instruction and data memories
- Permit use of **single clock cycle per instruction** implementation
- Due to use of "cache" in modern computer architectures, it is a fairly **realistic model**

The diagram shows the components of a Harvard architecture system:

- 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



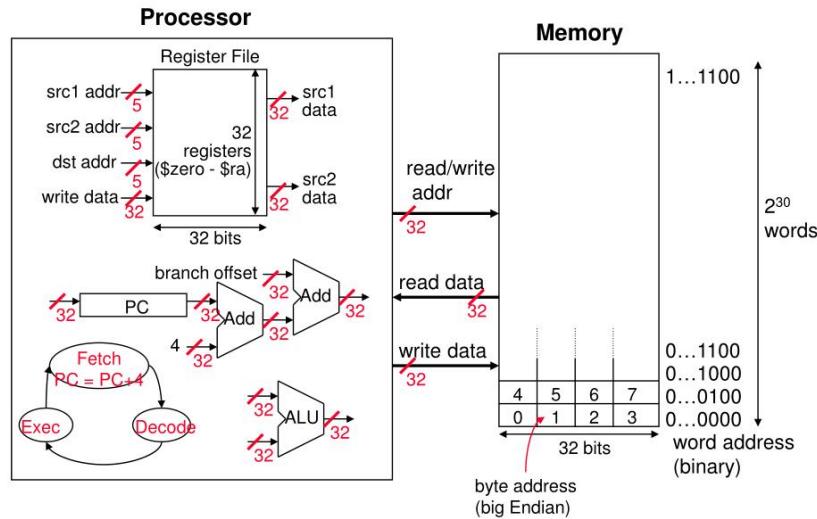


ISA - MIPS Instruction Format and Addressing Model

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

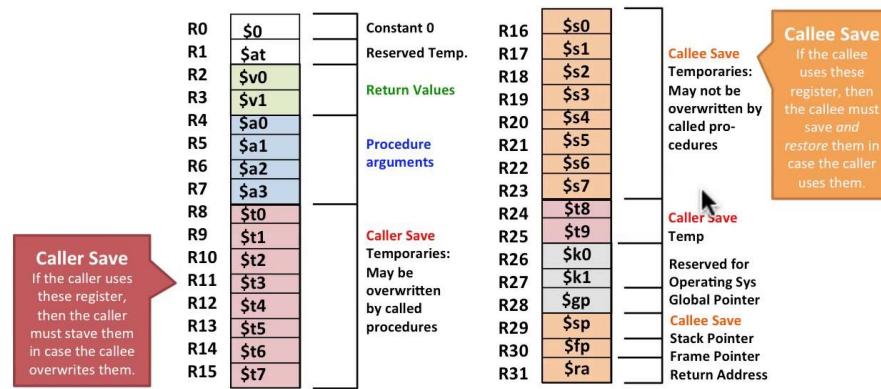
- When writing assembly, these registers can be referenced by their address (number) or name
- General purpose and special purpose registers

#	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



Caller Save
If the caller uses these registers, then the caller must save them in case the callee overwrites them.



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

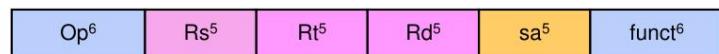
Instruction Formats

❖ All instructions are 32-bit wide. Three instruction formats:

❖ **Register (R-Type)**

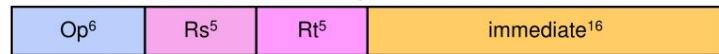
◊ Register-to-register instructions

◊ Op: operation code specifies the format of the instruction



❖ **Immediate (I-Type)**

◊ 16-bit immediate constant is part in the instruction



❖ **Jump (J-Type)**

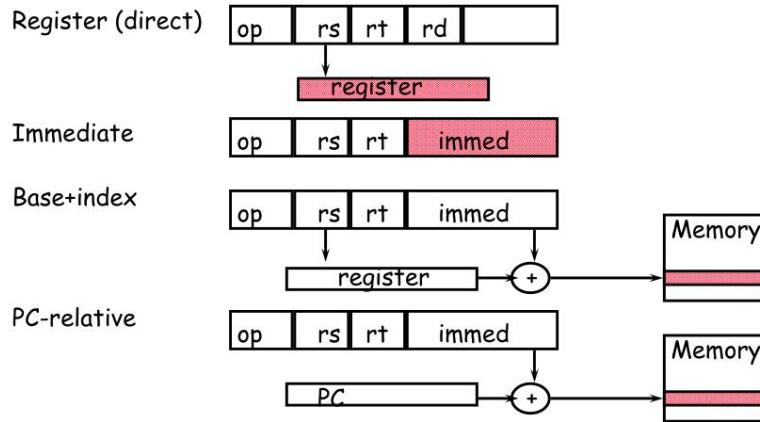
◊ Used by jump instructions



Addressing Model

Example: MIPS Instruction Formats and Addressing Modes

- All instructions 32 bits wide



ECE 361

3-45

5.6 Addressing Modes

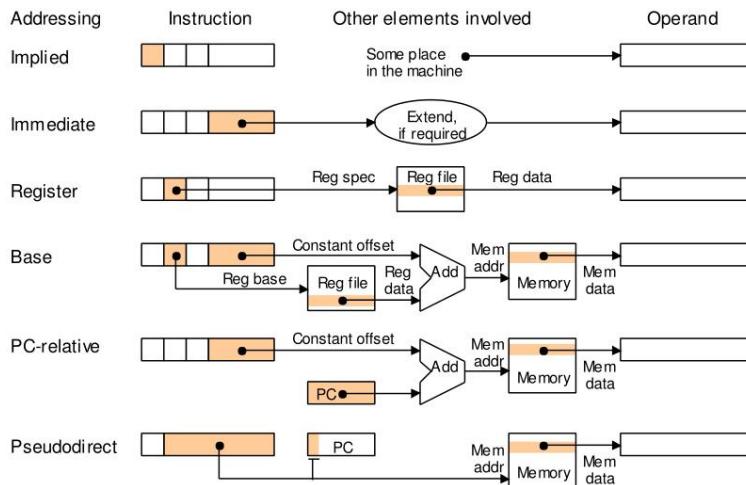


Figure 5.11 Schematic representation of addressing modes in MiniMIPS.

Computer Architecture, Instruction-Set Architecture

Slide 22

Instruction Categories

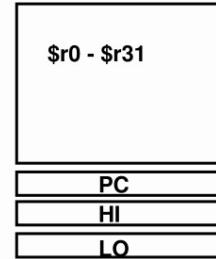
MIPS: ISA

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

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

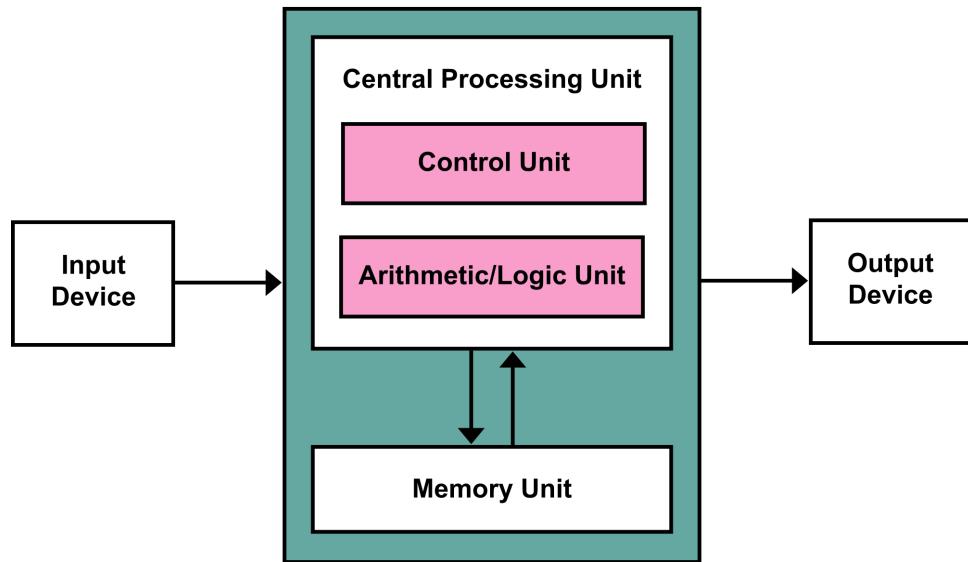
$$\text{Speedup} = \frac{\text{Ideal CPI} \times \text{Pipeline depth}}{\text{Ideal CPI} + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}}$$

For simple RISC pipeline, CPI = 1:

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

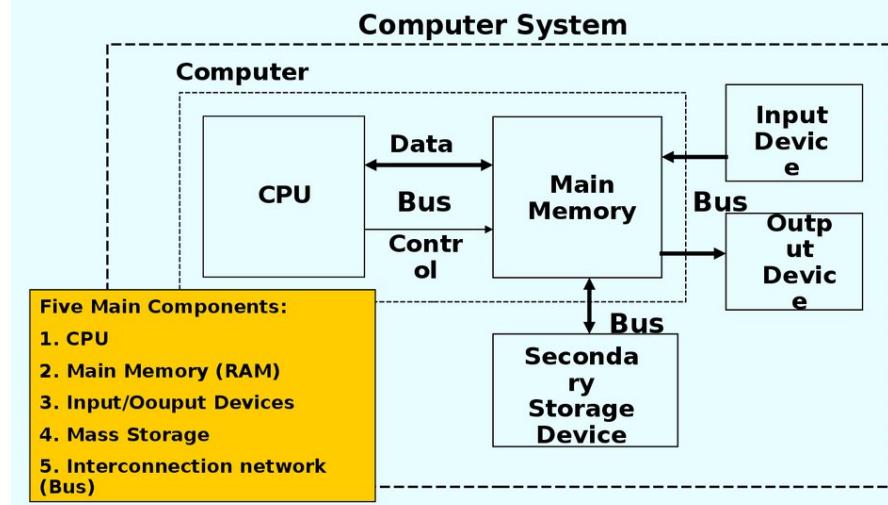
CS211 41

Processor - Von Neumann Architecture

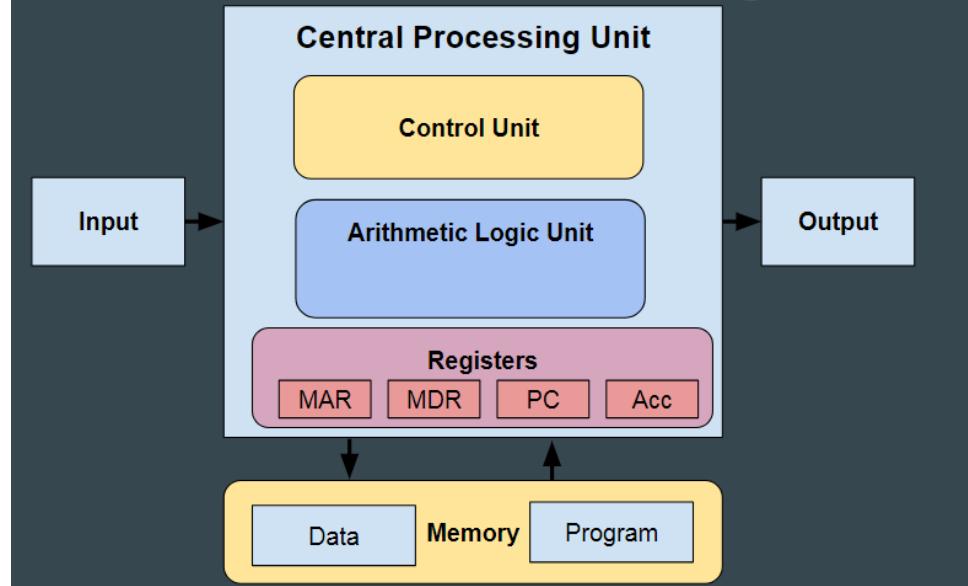


von Neumann Architecture

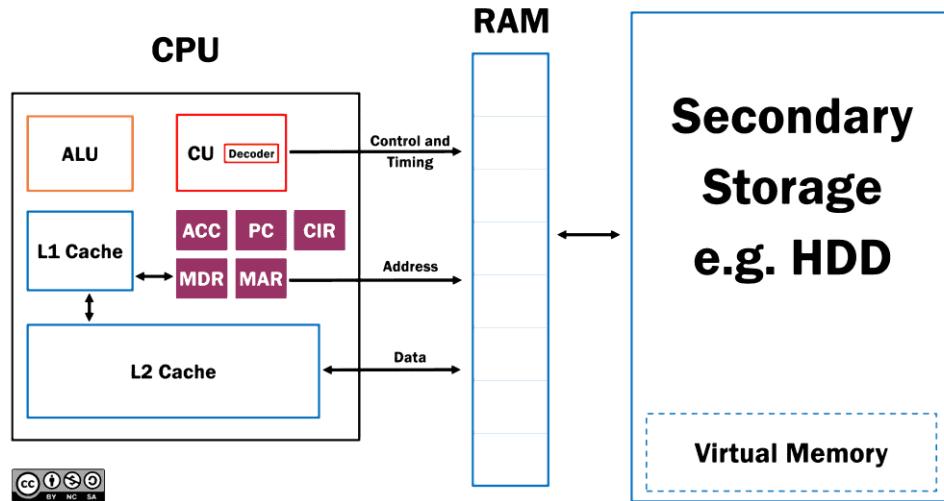
- A more complete view of the computer *system* architecture that integrates interaction (human or otherwise) consists of:



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 pipeline 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

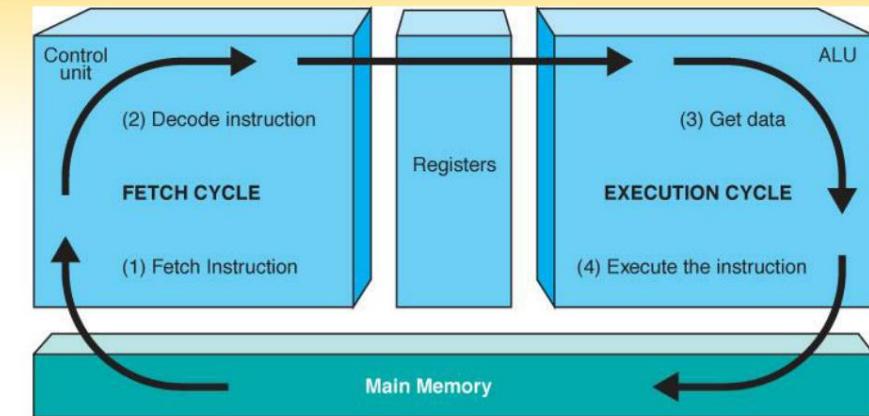
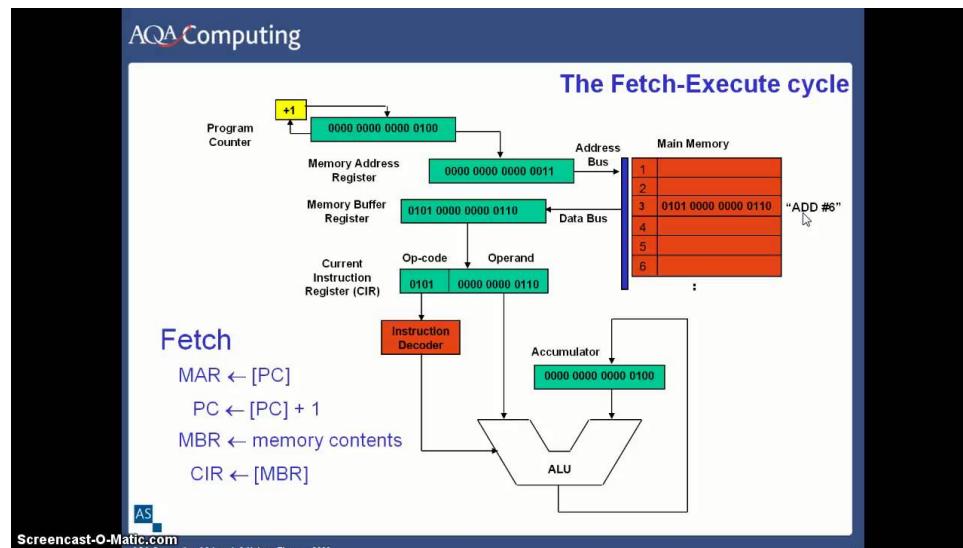
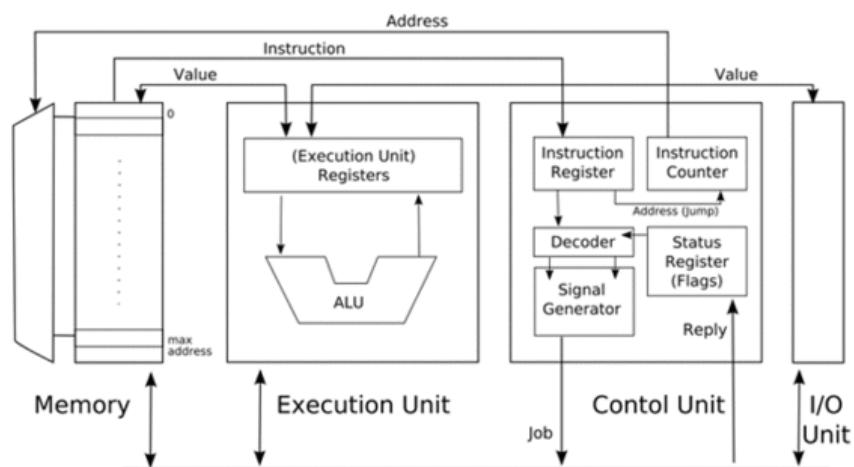
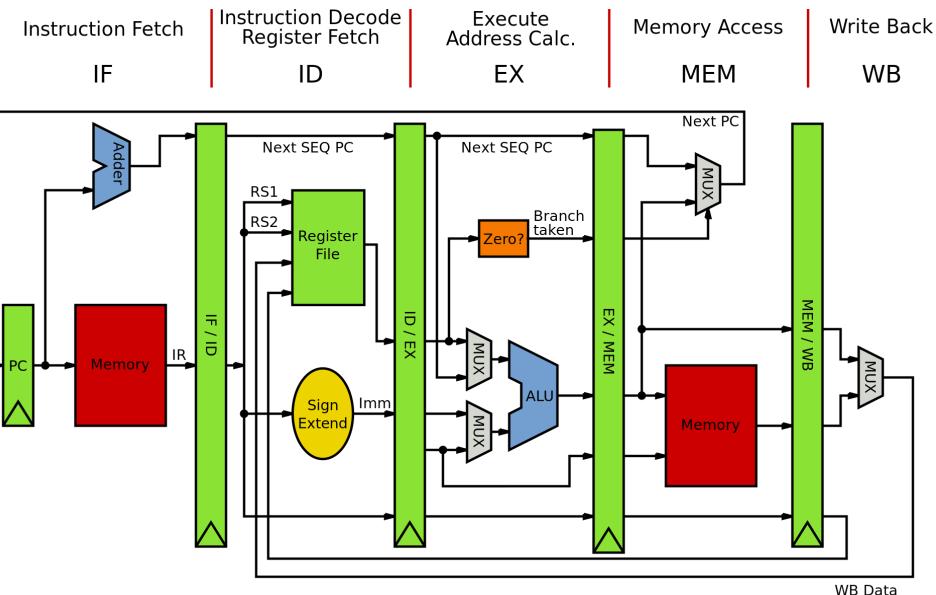
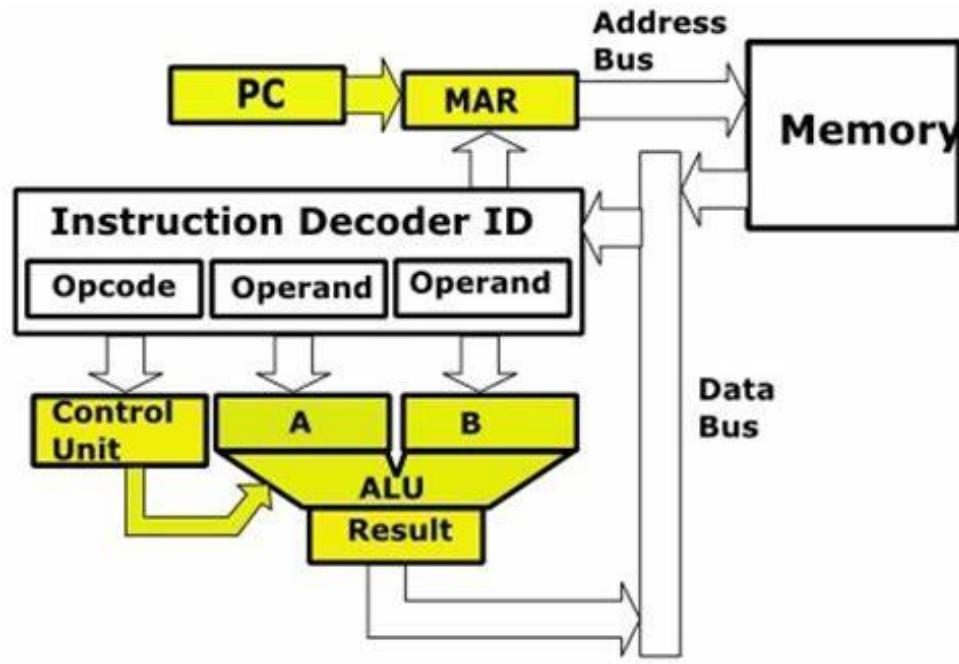


Figure 5.3 The Fetch-Execute Cycle

17

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



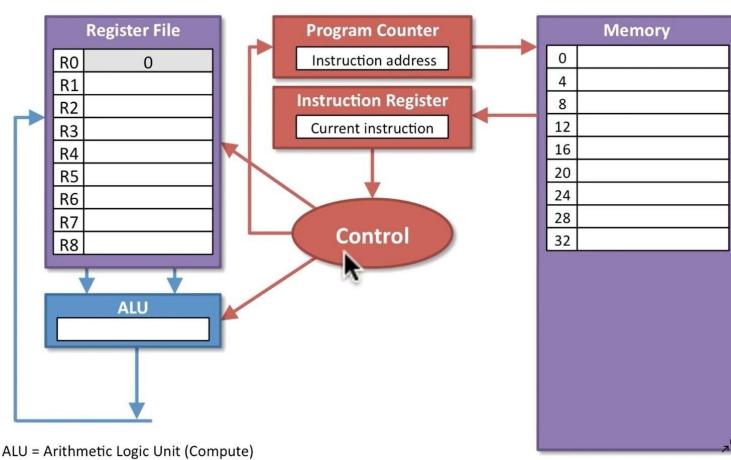


Data operations in detail

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.

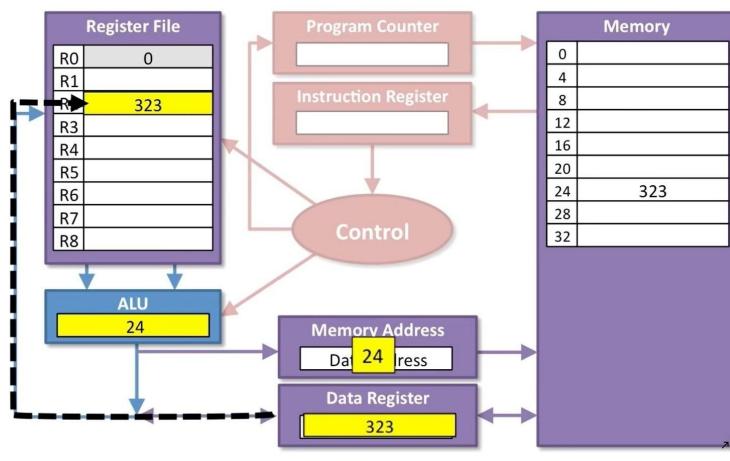


Data transfers in detail

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.

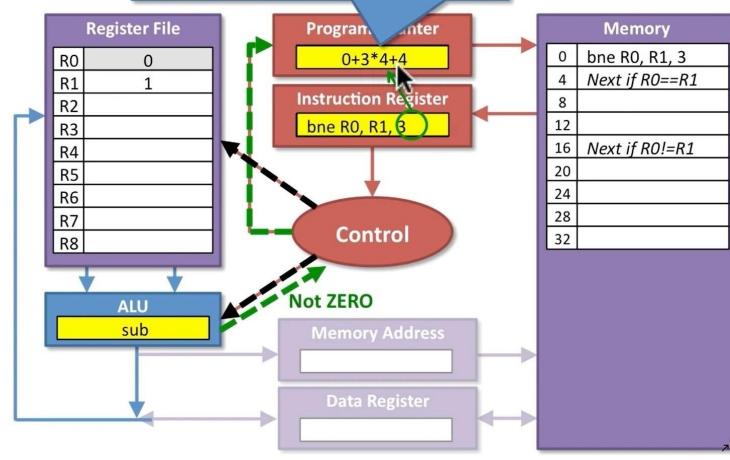
**Sequencing in detail**

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

The label constant is in instruction words, so it needs to be multiplied by 4 to convert to byte address.

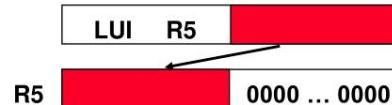
**Processor - Datapath****Review: MIPS data transfer instructions**

- 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?

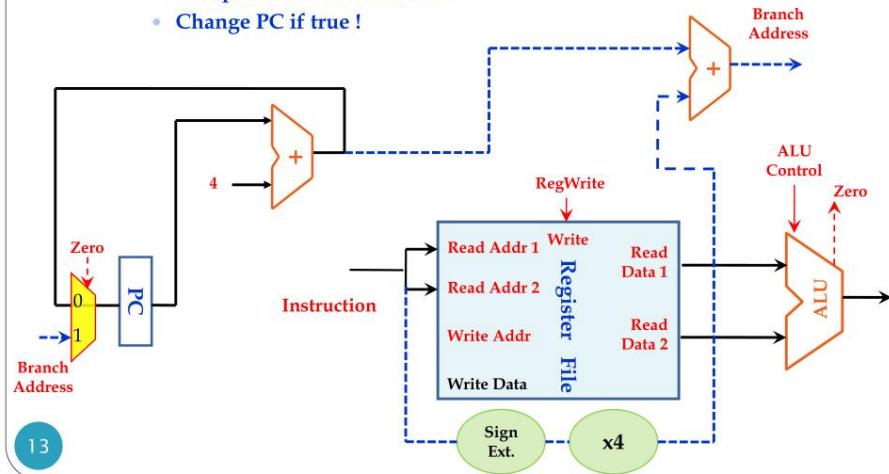


Single-Cycle Datapath

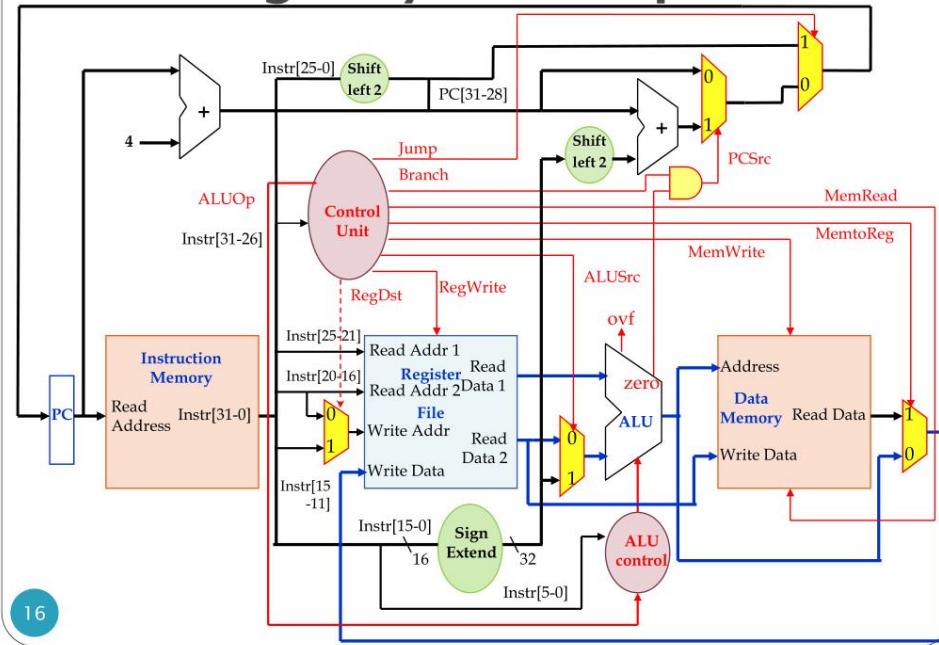
- Execution Datapath

- Branch Instruction

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



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

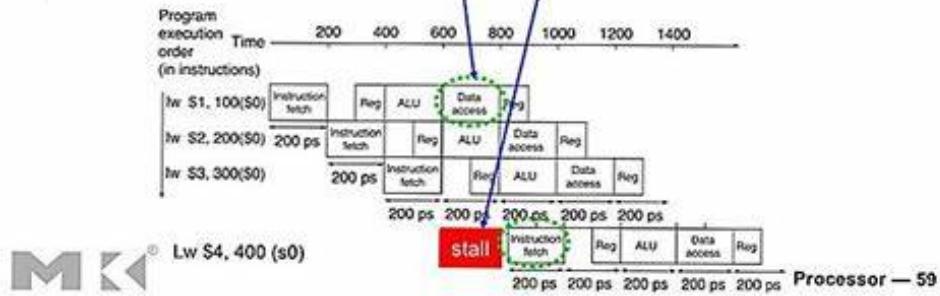
Summary - Control Hazard Solutions

- **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

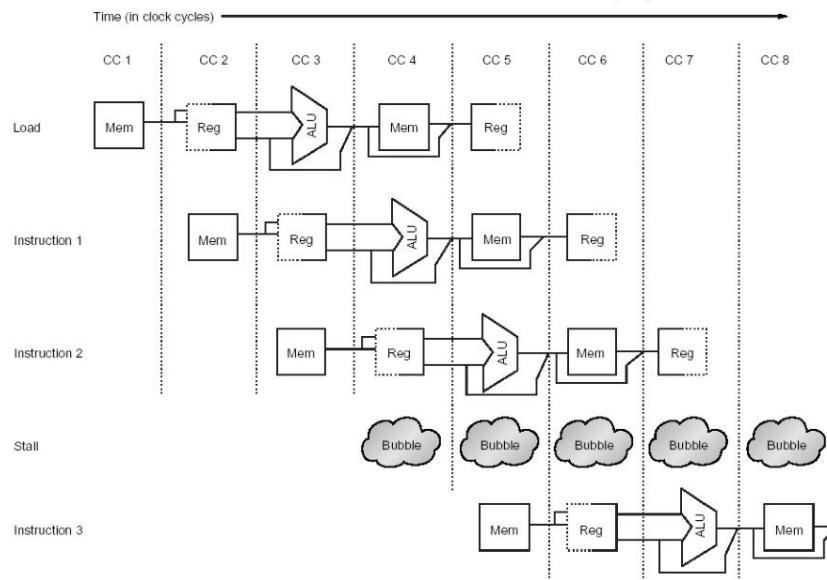
Solution for Pipeline Hazards

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



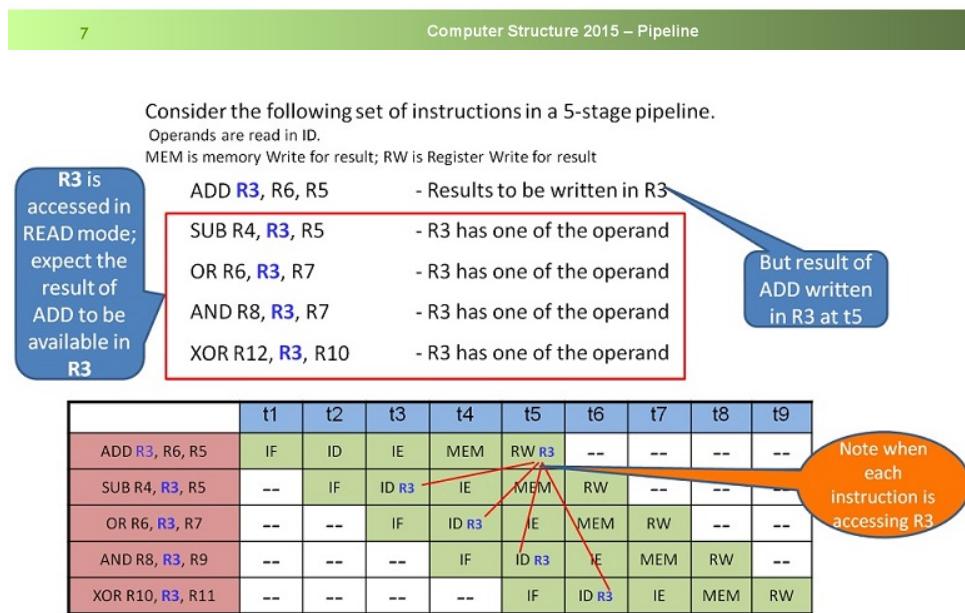
Structural Hazards (3)



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

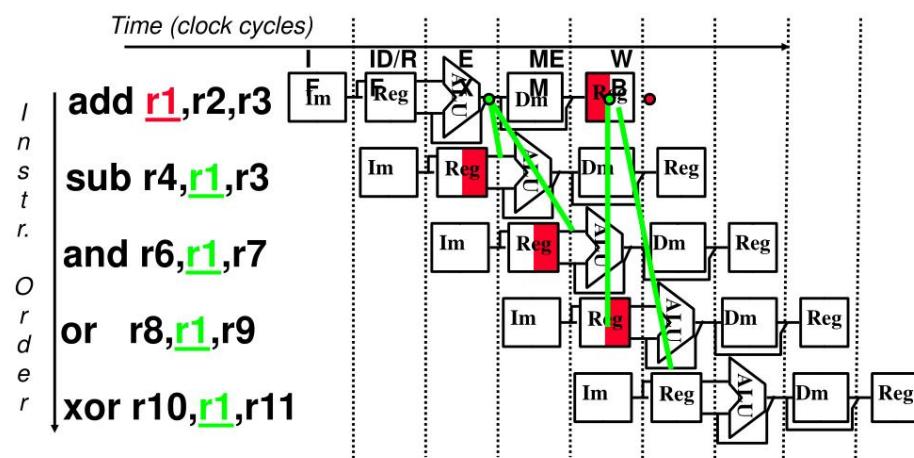
Structural Hazard

- ◆ 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



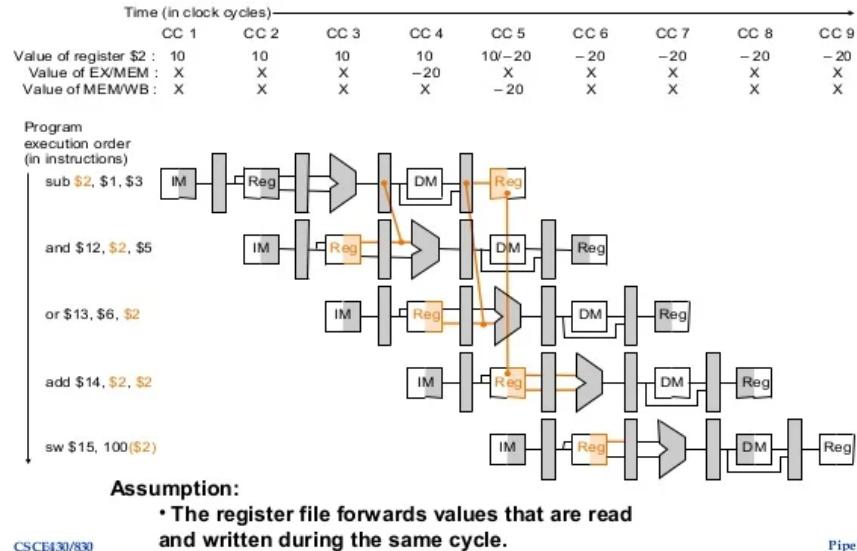
Data Hazard Solution:

- “Forward” result from one stage to another



Data Hazard Solution: Forwarding

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



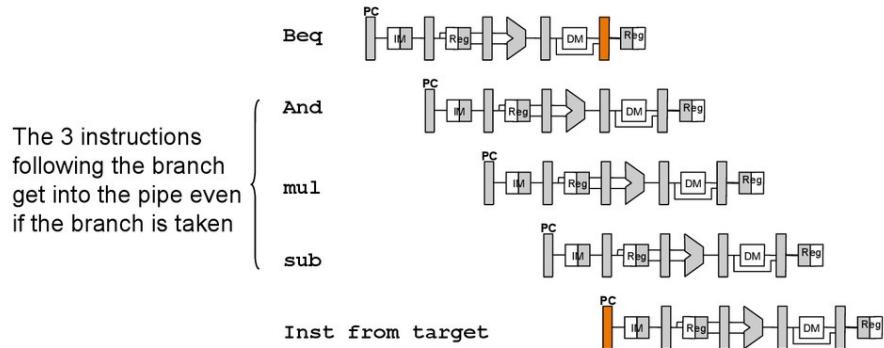
CSCE430/830

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



28

Computer Structure 2015 – Pipeline

Control Hazard Review

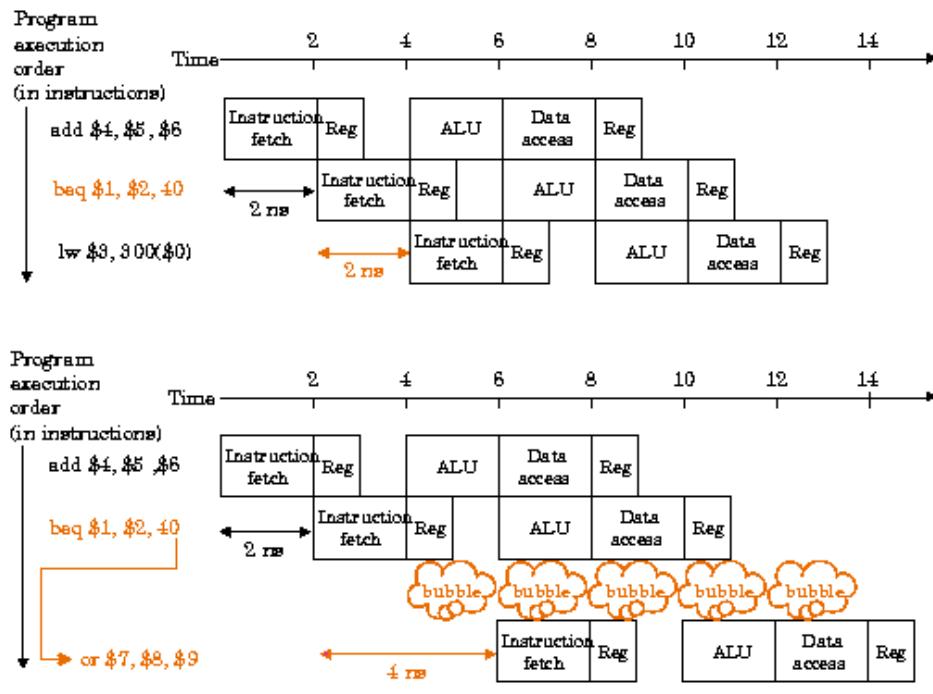
The nub of the problem:

- 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

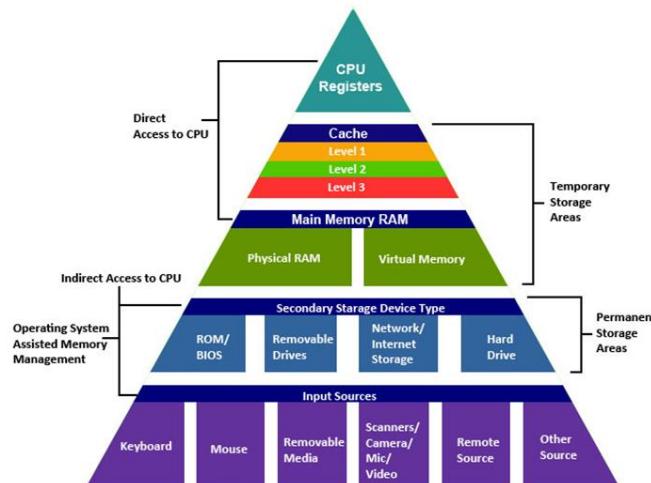
- number of cycles in between * branch frequency = the contribution to CPI due to branches

Predict the branch outcome to avoid stalling



Mermory Hierarchy

The Memory Hierarchy

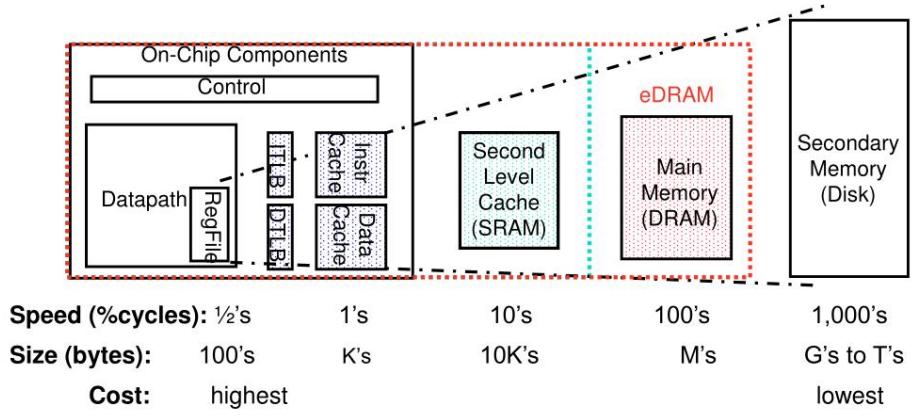


<http://cse1.net/recaps/4-memory.html>

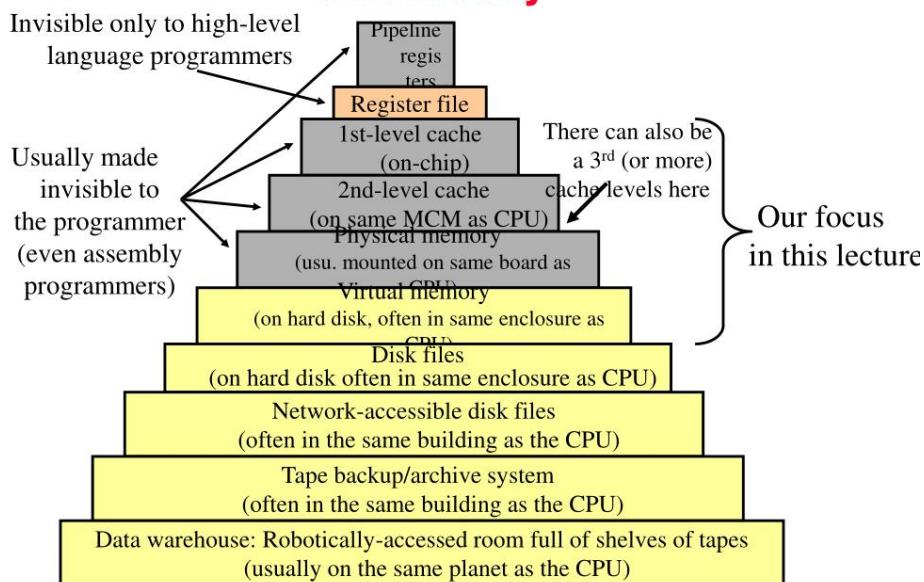
2

A Typical Memory Hierarchy

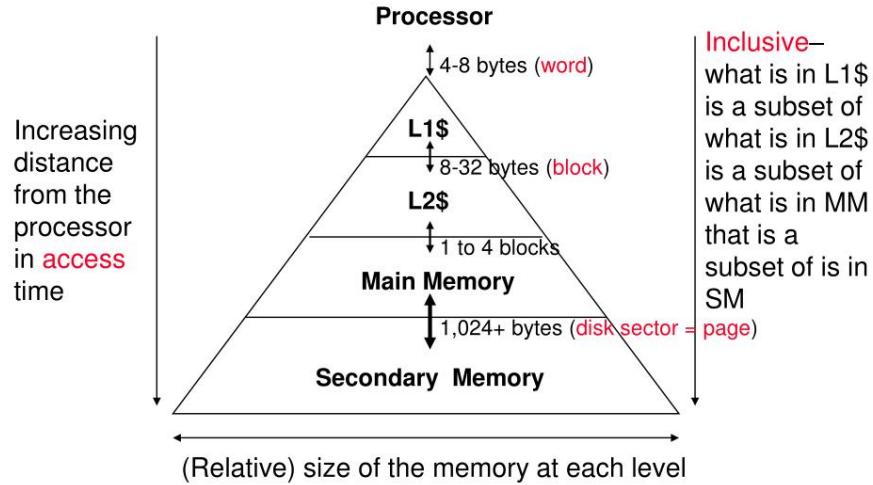
- ❑ By taking advantage of the principle of locality
 - Can present the user with as much memory as is available in the cheapest technology
 - at the speed offered by the fastest technology



Many Levels in Memory Hierarchy



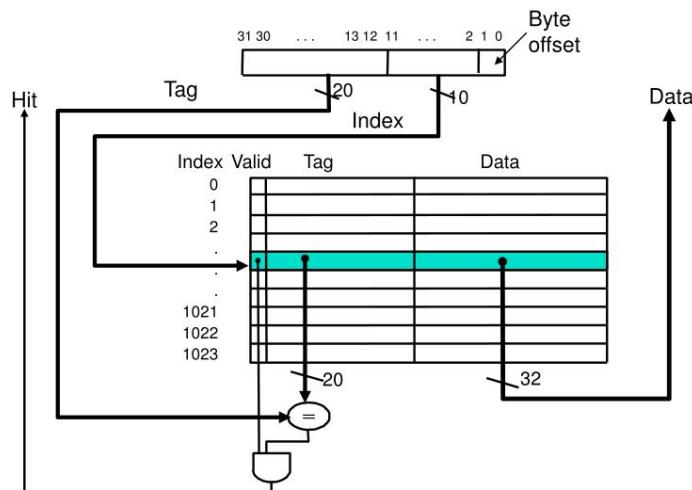
Characteristics of the Memory Hierarchy



Block Size / Hit Rate / Miss Rate

MIPS Direct Mapped Cache Example

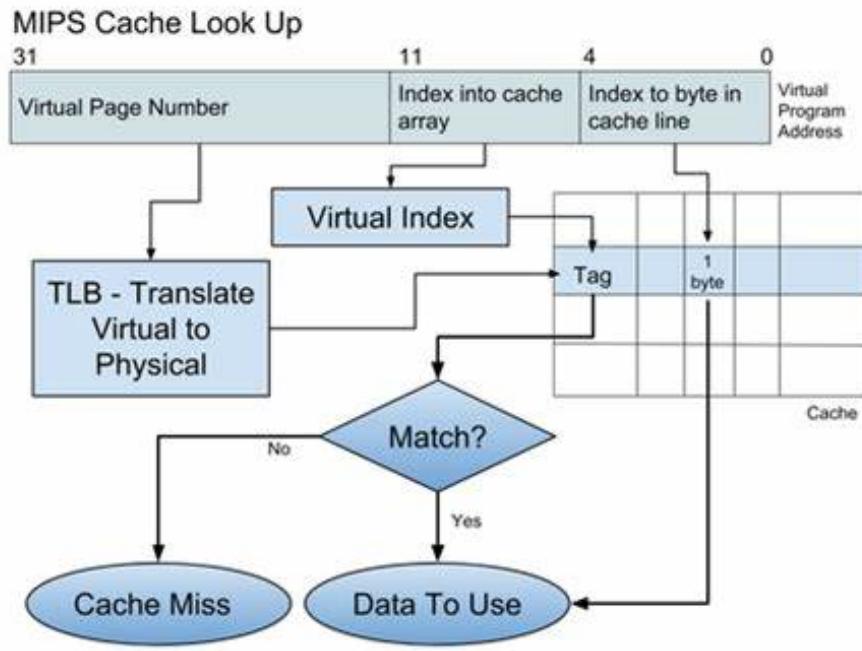
- ❑ One word/block, cache size = 1K words



What kind of locality are we taking advantage of?

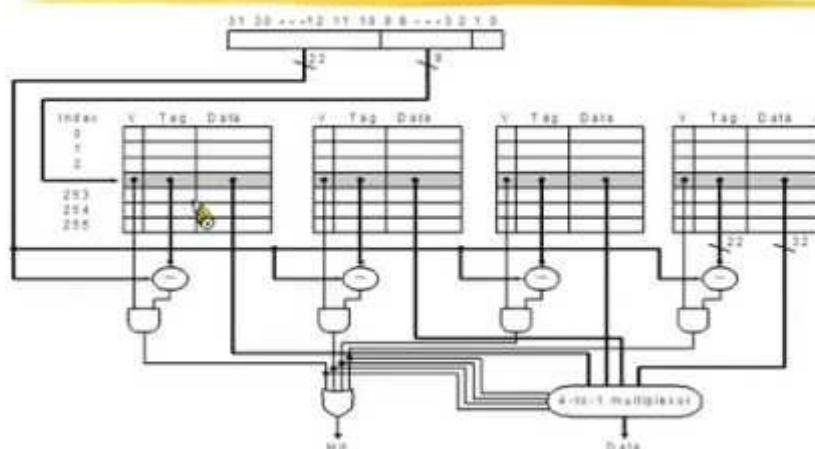
CEG3420 L13.34

Qiang Xu CUHK, Spring 2011



Adapted from Imagination Technologies MIPS basic training course materials

A 4-Way Set-Associative Cache



- Increasing associativity shrinks index, expands tag

Memory-19

Computer Architecture

Miss Rate vs. Block Size

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

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

2/15/99

CS520S99 Memory

C. Edward Chow Page 34

Block Size Tradeoffs

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



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

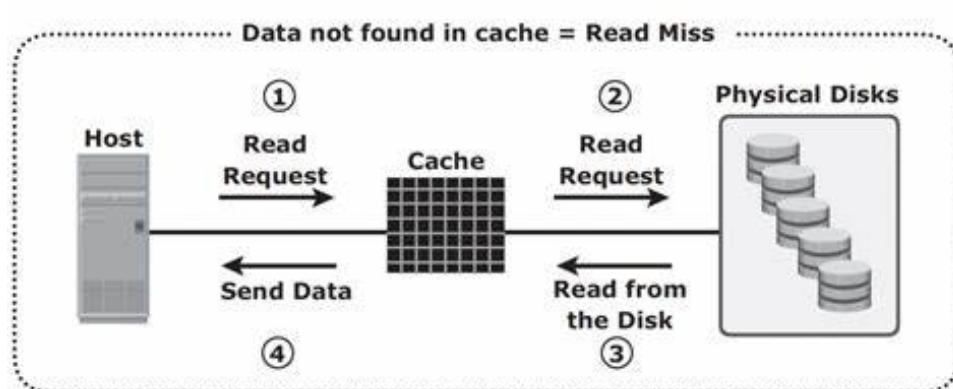
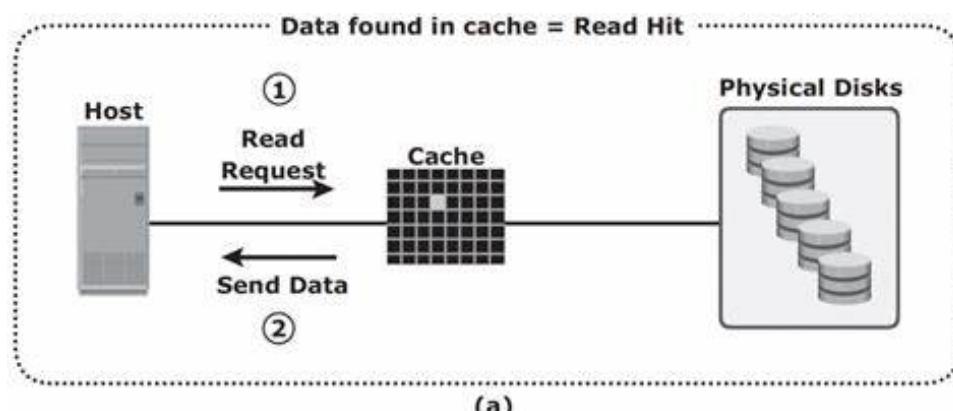
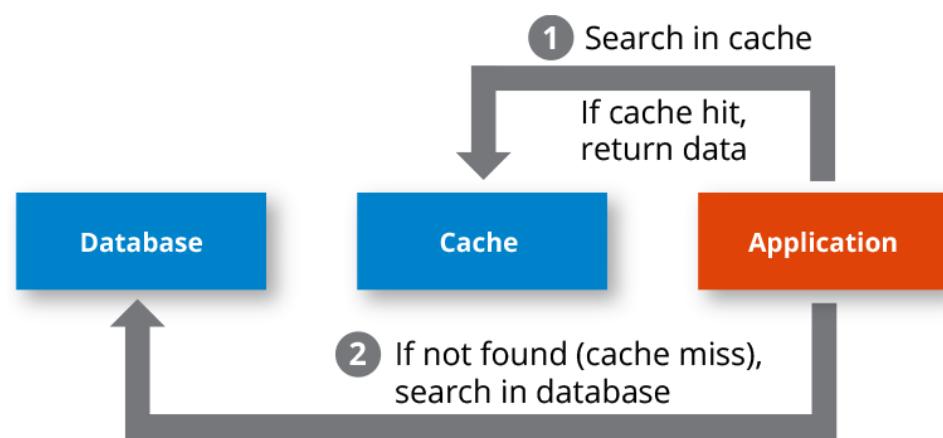
OR

$$\text{Hit ratio} = 1 - \text{Miss ratio}$$

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

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

Handle Cache Miss



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

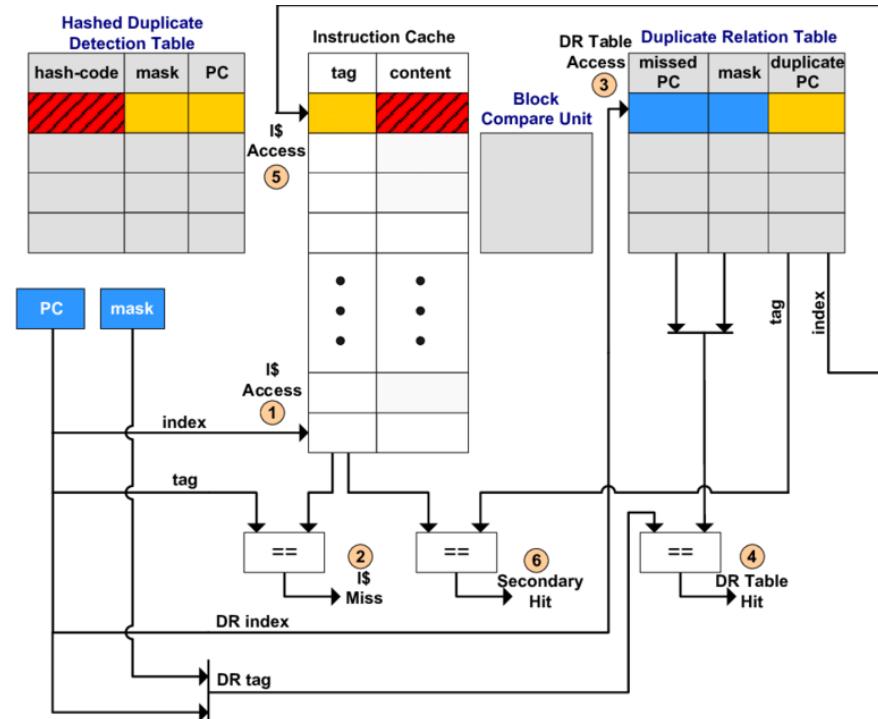
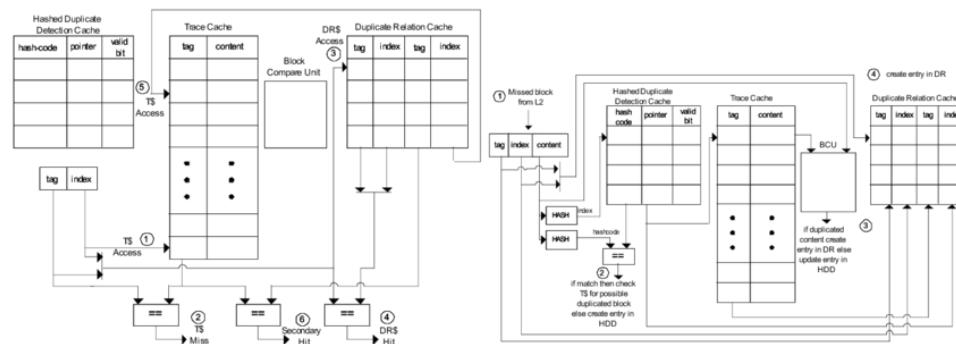
1 Compulsory: On the first access to a block; the block must be brought into the cache; also called cold start misses, or first reference misses.

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!**

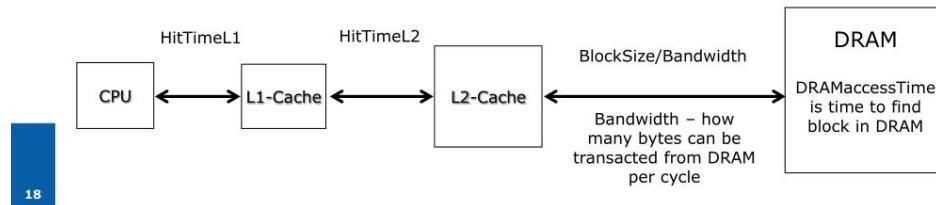
46

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



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
 - * Sub-block placement
 - * Non-blocking caches
 - * Second-level cache (L₂)
- **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

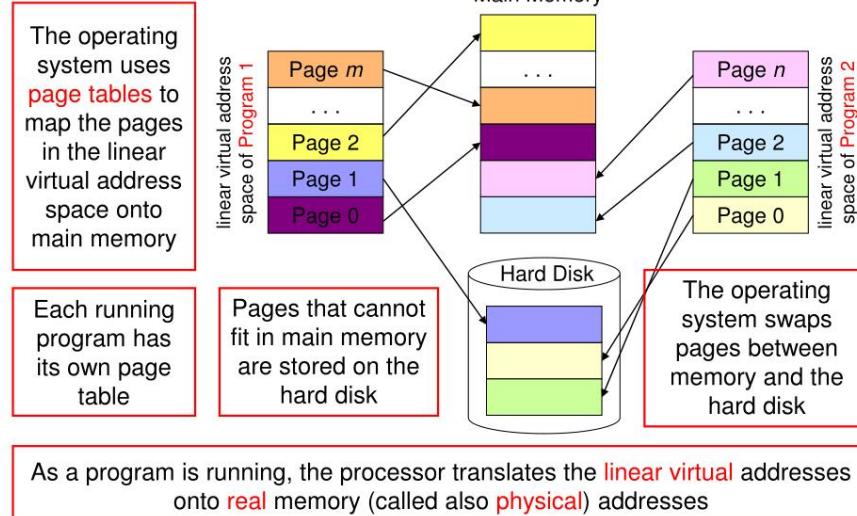
- **Temporal locality hit**
 - Accessing same word that previously accessed
- **Spatial locality hit**
 - Accessing word spatially near previously accessed word

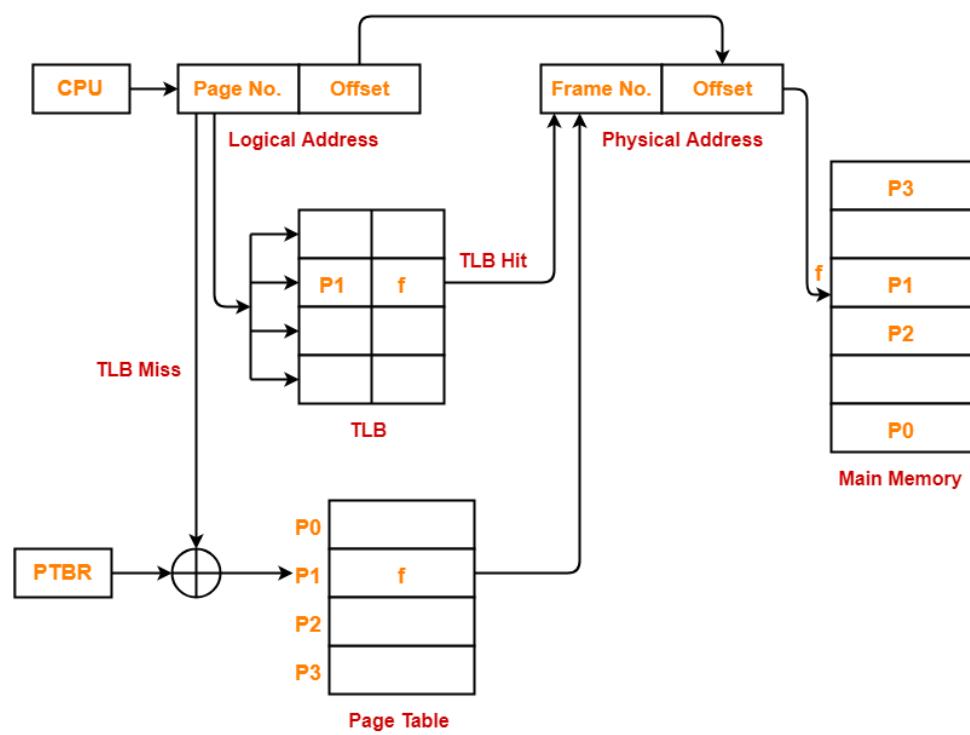
- 42 -

CS 740 F'07

Address Translation Mechanism

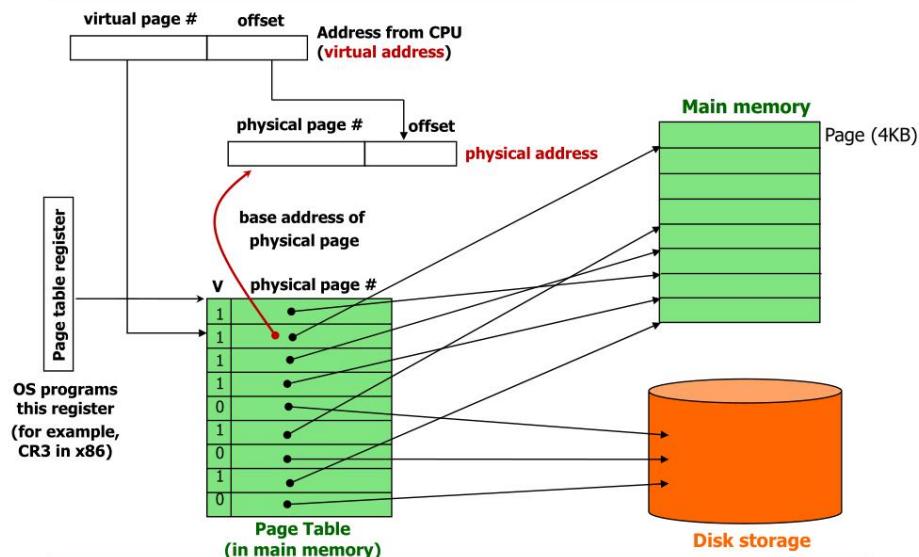
Paging





Translating Logical Address into Physical Address

Address Translation Mechanism



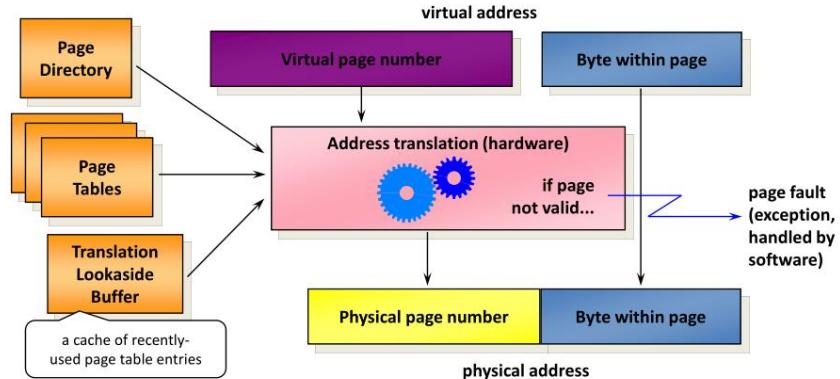
11

Korea Univ

Address Translation with Cache

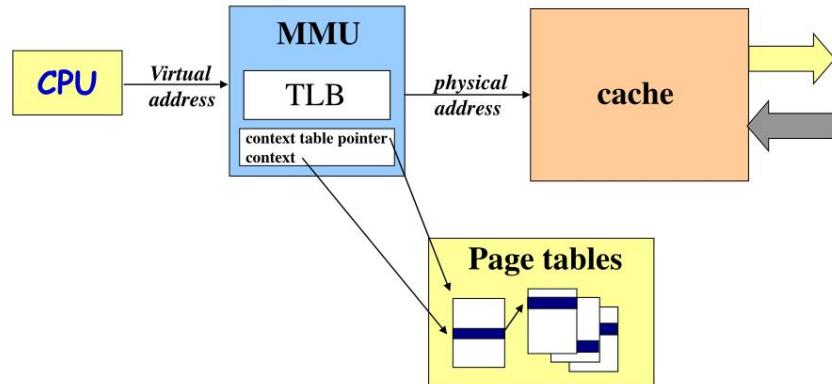
Virtual Address Translation

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

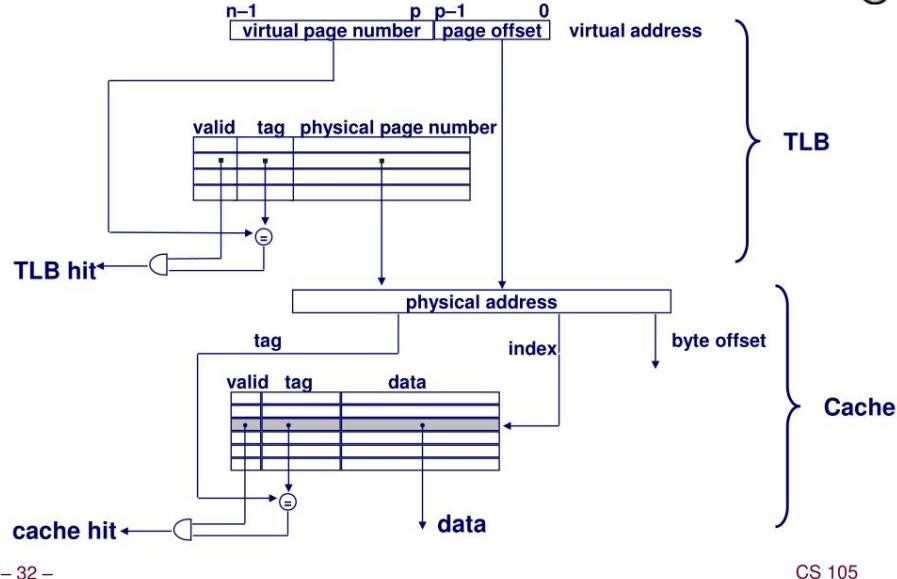


31

Address Translation Overview



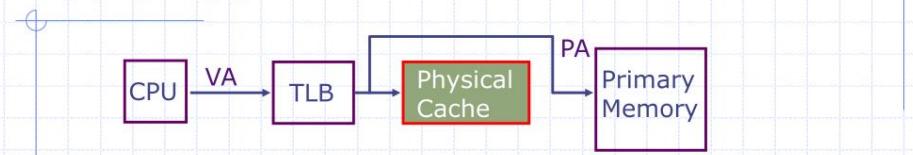
Address Translation With a TLB



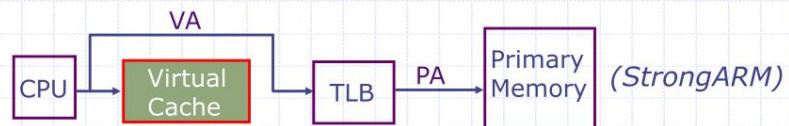
- 32 -

CS 105

Physical or Virtual Address Caches?

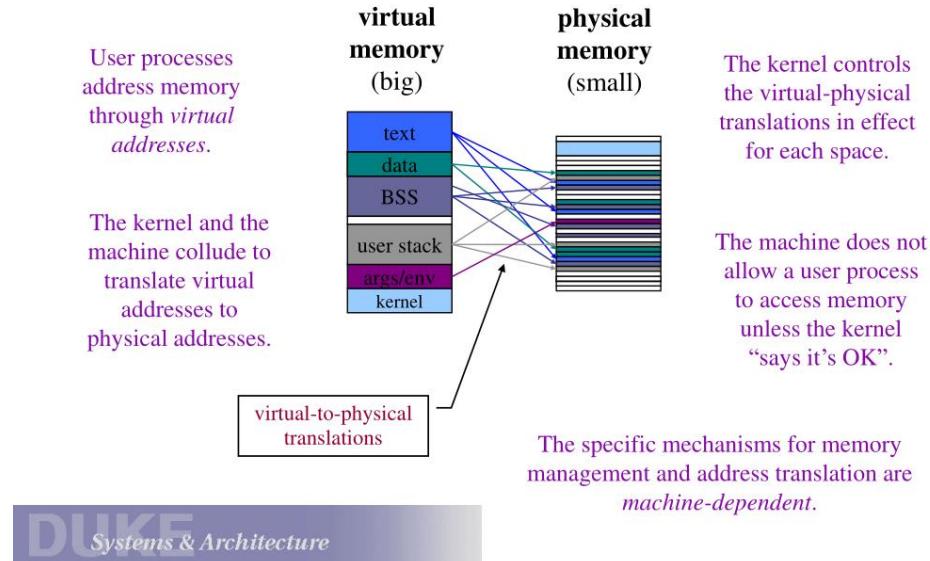


Alternative: place the cache before the TLB



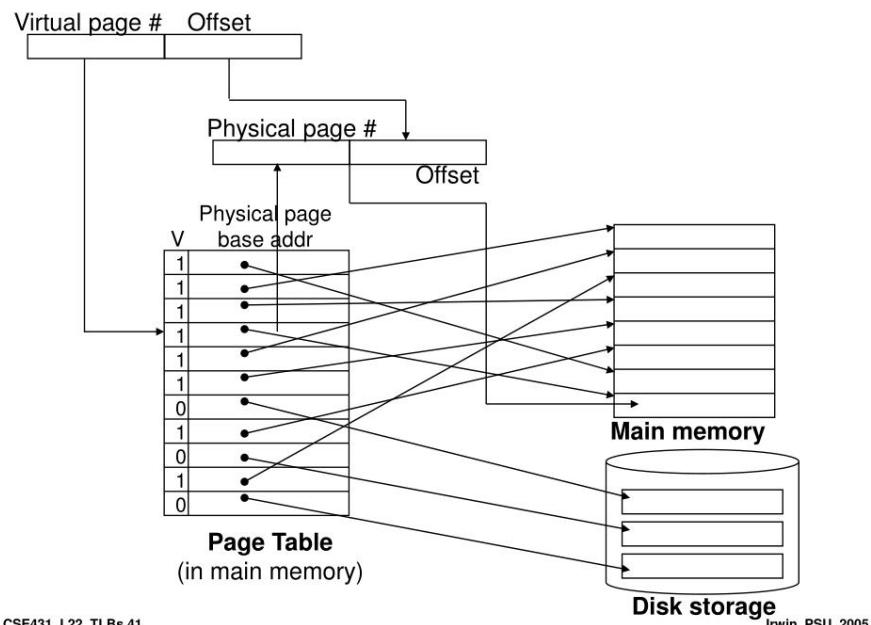
- ◆ one-step process in case of a hit (+)
- ◆ cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (-)
- ◆ aliasing problems due to the sharing of pages (-)

Review: Virtual Addressing

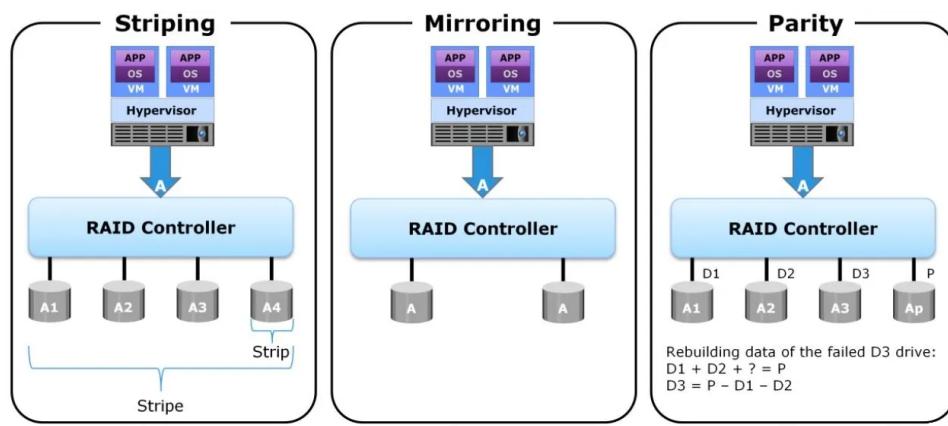
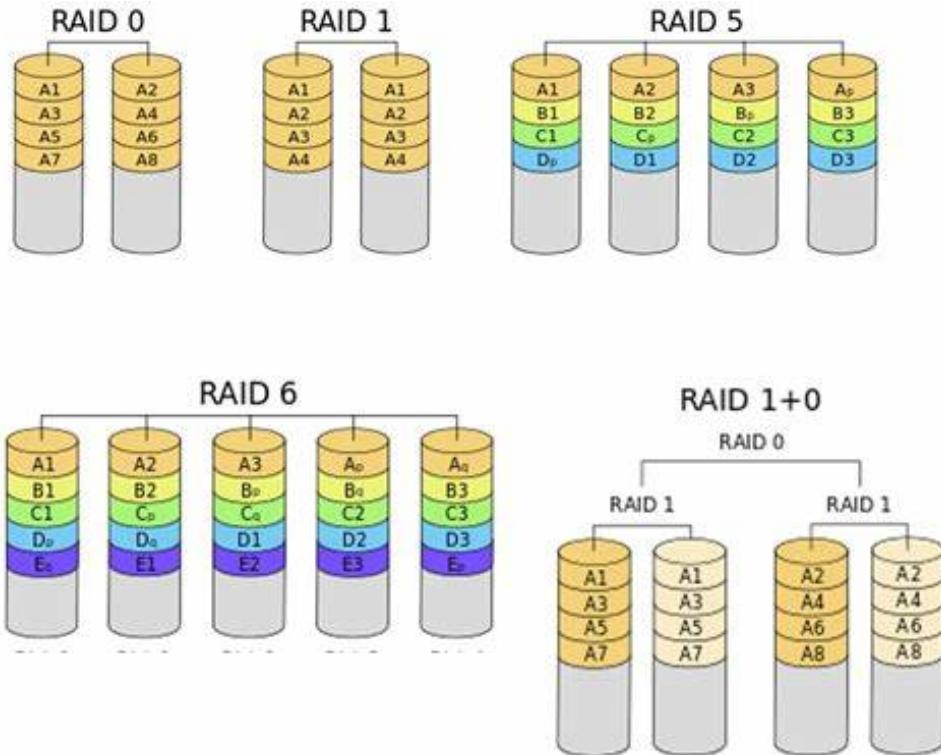


DUKE Systems & Architecture

Address Translation Mechanisms

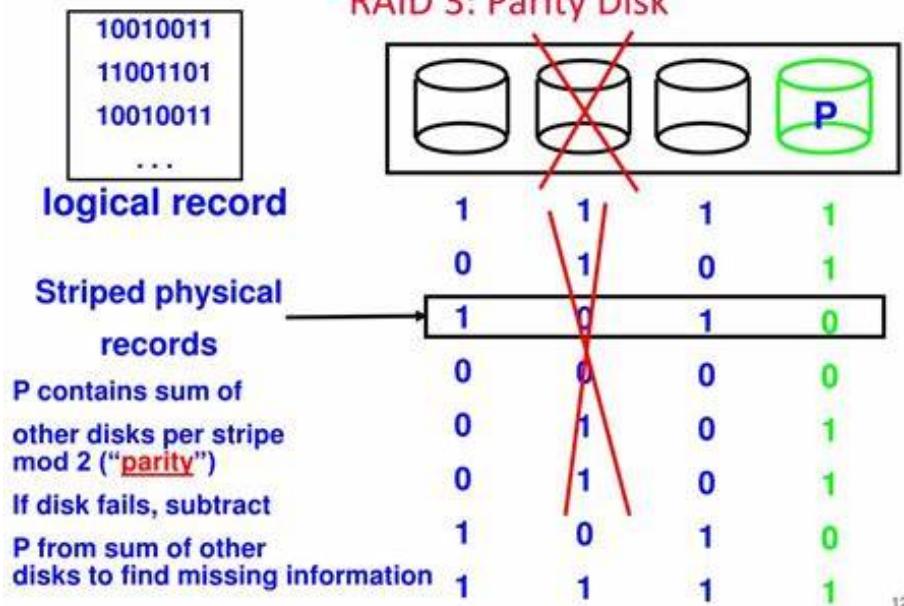


Redundant Array of Inexpensive Disk



Redundant Array of Inexpensive Disks

RAID 3: Parity Disk

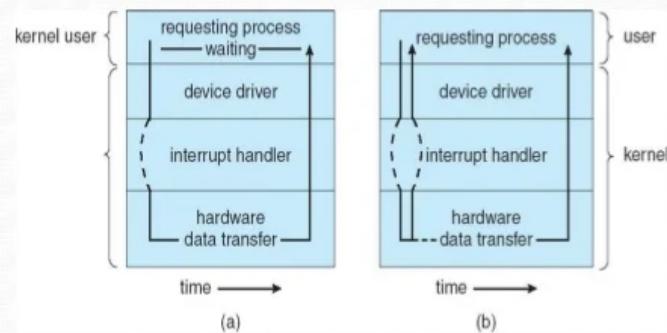


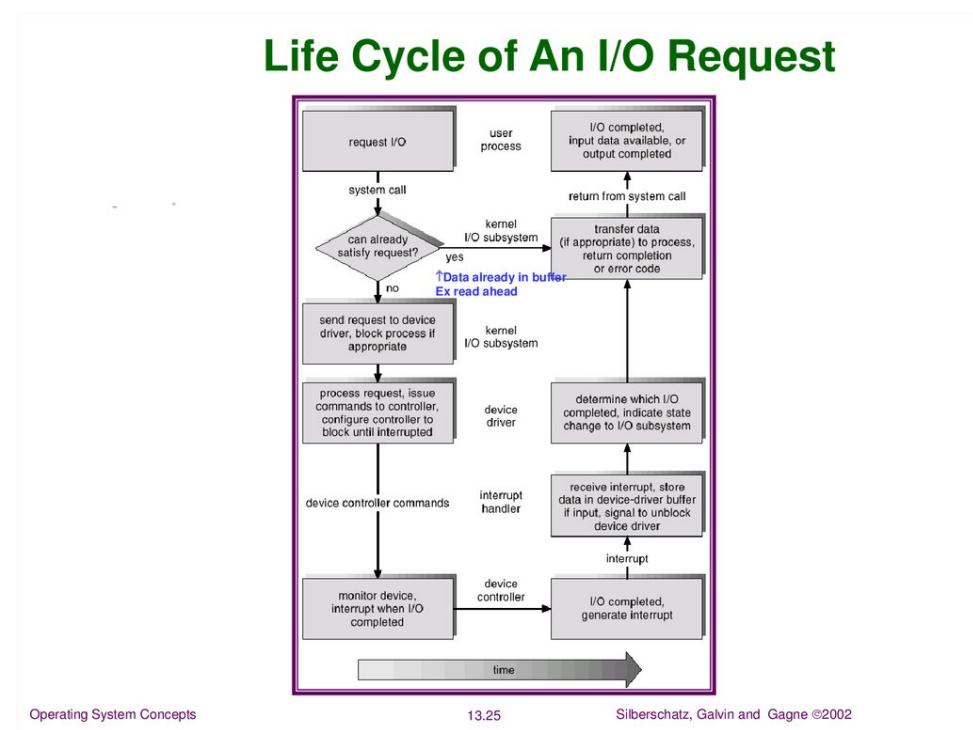
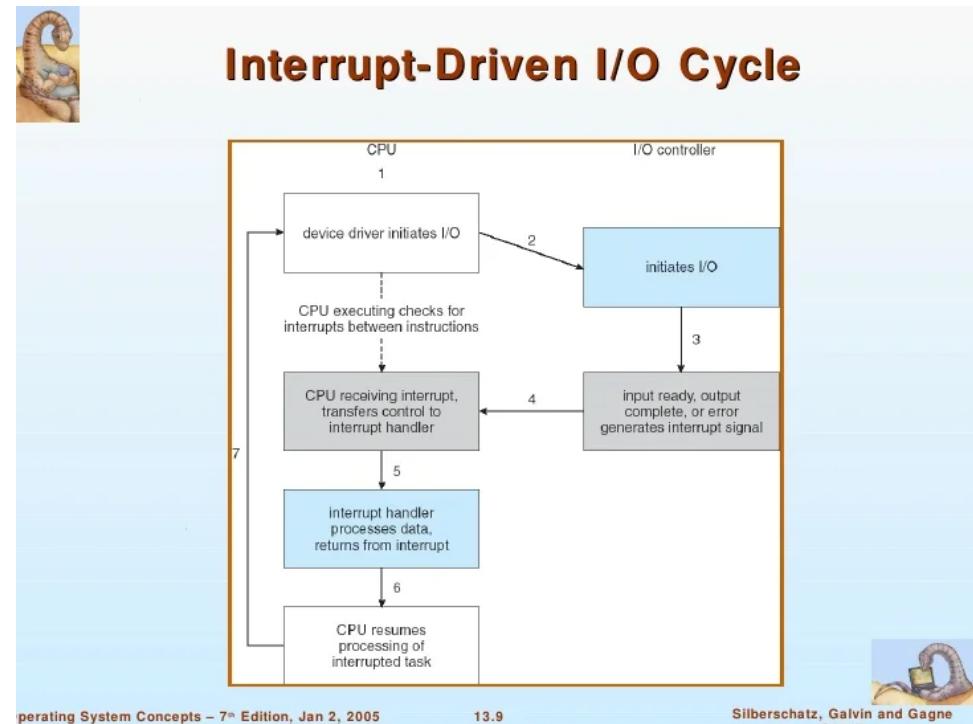
IO System Performance

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

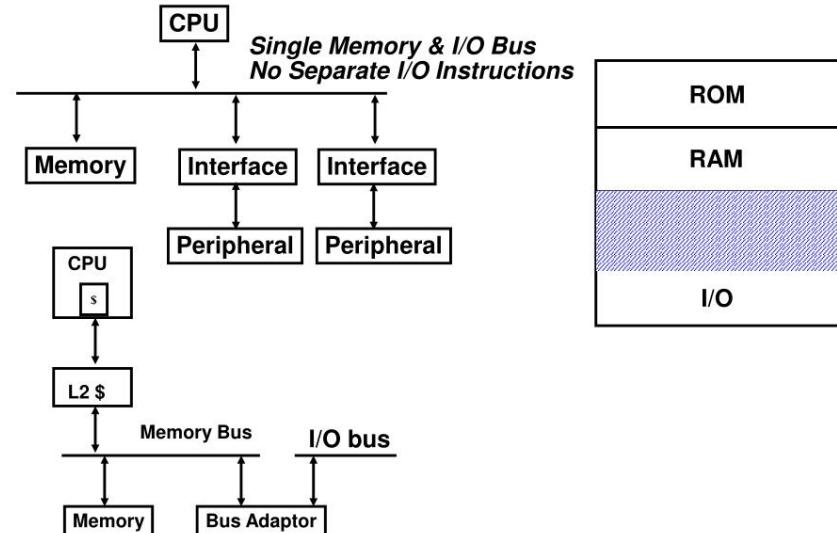
Two I/O Methods





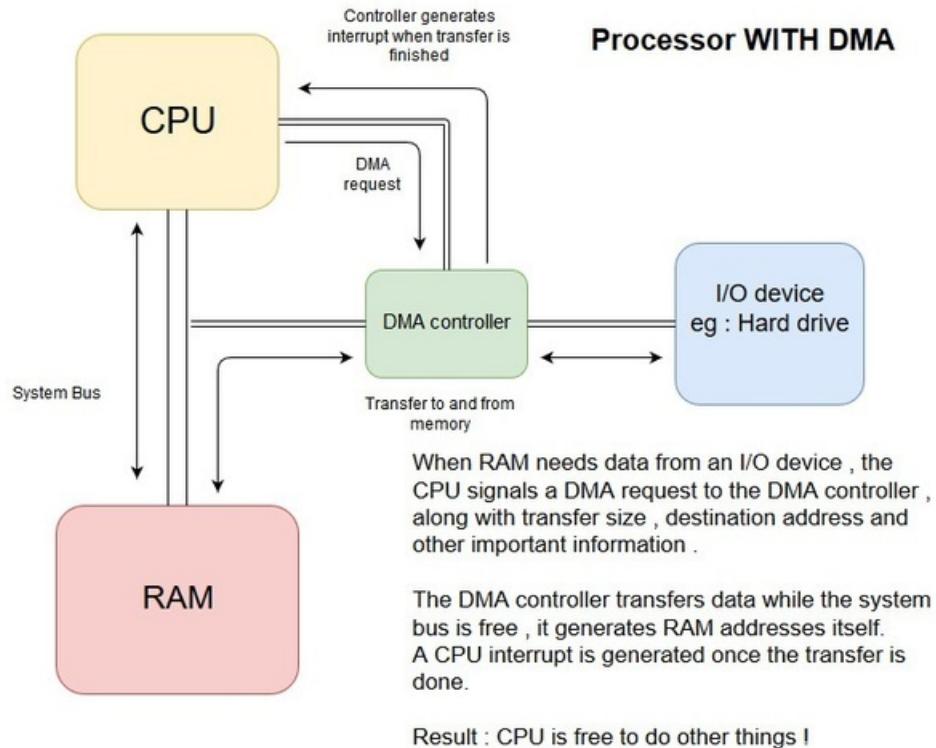
DMA(Direct Memory Access)

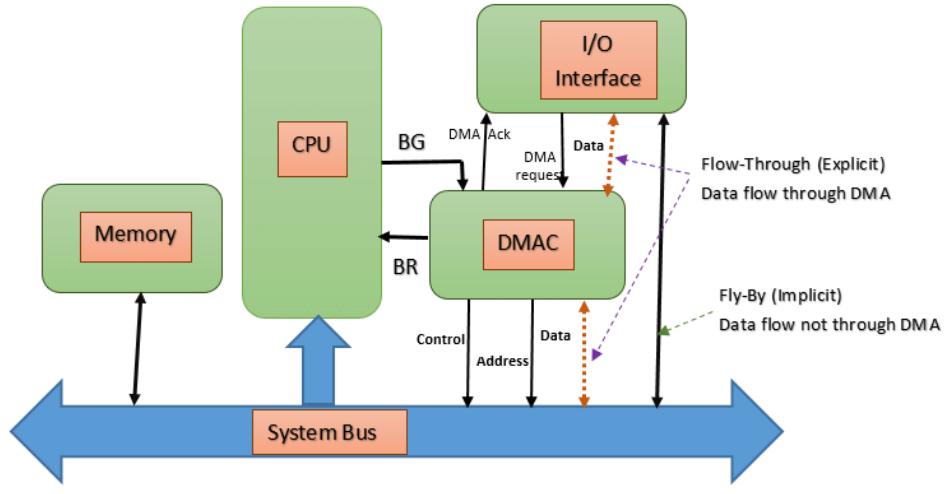
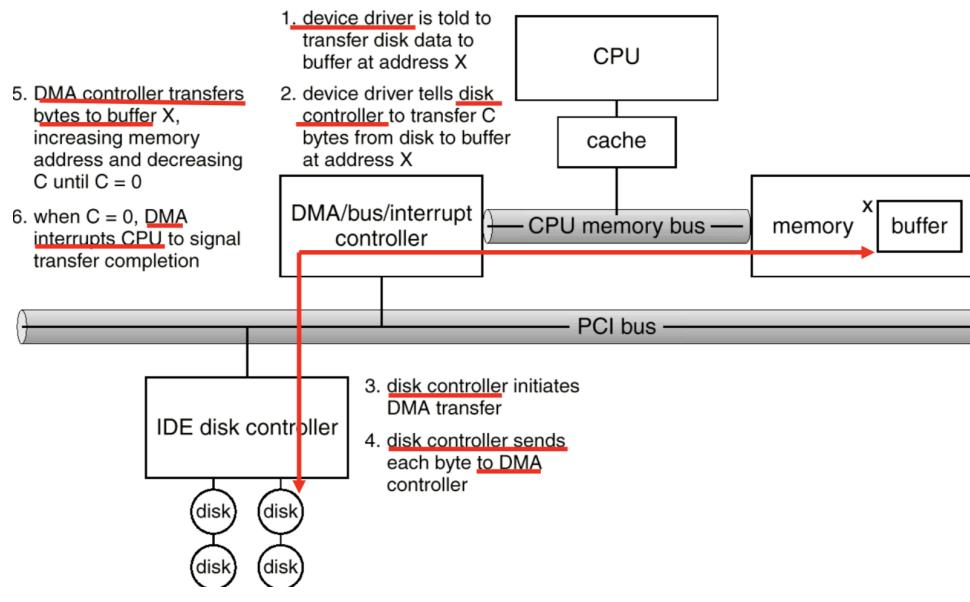
Memory Mapped I/O



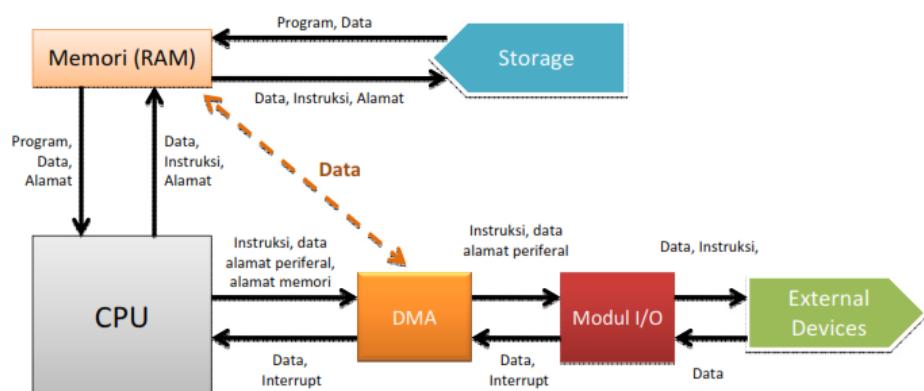
204521 Digital Computer Architecture

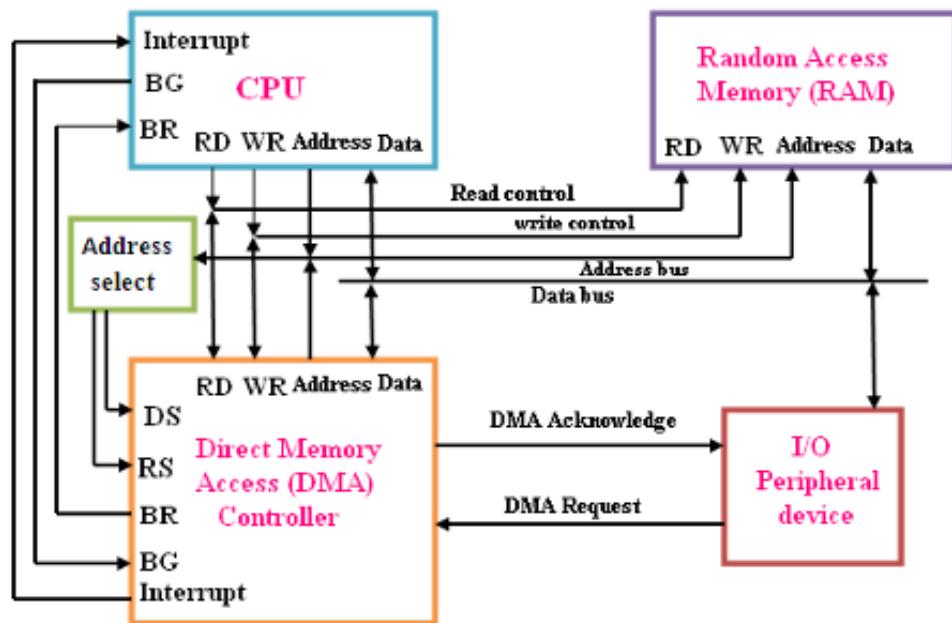
47





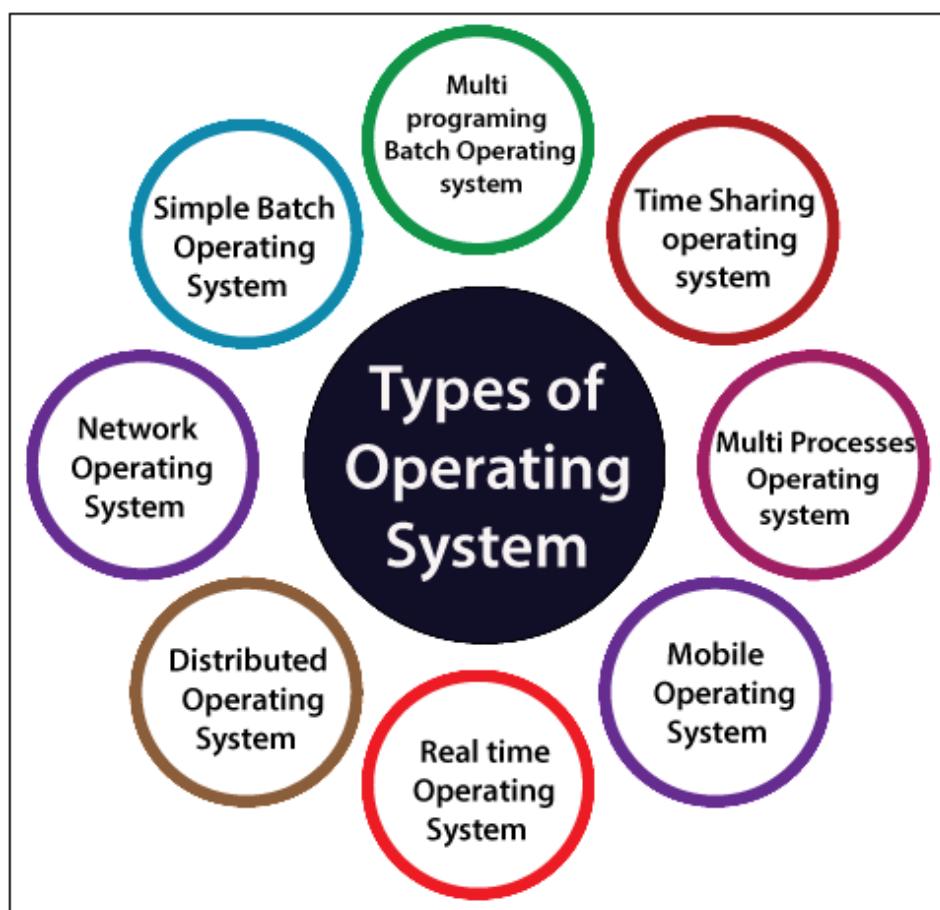
Direct Memory Access





Operating System

Types of OS



What are the different types?

Mac OS is a series of graphical user interface-based operating systems developed by Apple Inc. for their Macintosh



Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution.



Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.



iOS (previously iPhone OS) is a mobile operating system developed and distributed by Apple Inc. Originally unveiled in 2007 for the iPhone, it has been extended to support other Apple devices such as the iPod Touch



Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc.



BSD/OS had a reputation for reliability in server roles; the renowned Unix programmer and author W. Richard Stevens used it for his own personal web server for this reason.



Real time Operating system

- Real time operating systems are used as OS in real time system.
- In RTOS tasks are completed in given time constraints.
- RTOS is a multitasking system where multiple tasks run concurrently
 - system shifts from task to task
 - must remember key registers of each task
(this is called context of task)



EmbeddedCraft
crafting of intelligent systems

Real-Time Operating System

- ◆ An RTOS is an OS for response time-controlled and event-controlled processes. It is very essential for large scale embedded systems.
- ◆ RTOS occupy little space from 10 KB to 100KB
- ◆ The main task of a RTOS is to manage the resources of the computer such that a particular operation executes in precisely the same amount of time every time it occurs.

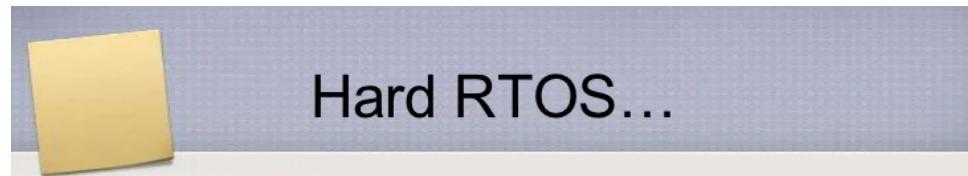


Renesas automotive dashboard platform.

Soft RTOS...

- In a soft real-time system, it is considered undesirable, but not catastrophic, if deadlines are occasionally missed.
- Also known as “best effort” systems
- Most modern operating systems can serve as the base for a soft real time systems.
- Examples:
 - multimedia transmission and reception,
 - networking, telecom (cellular) networks,
 - web sites and services
 - computer games.

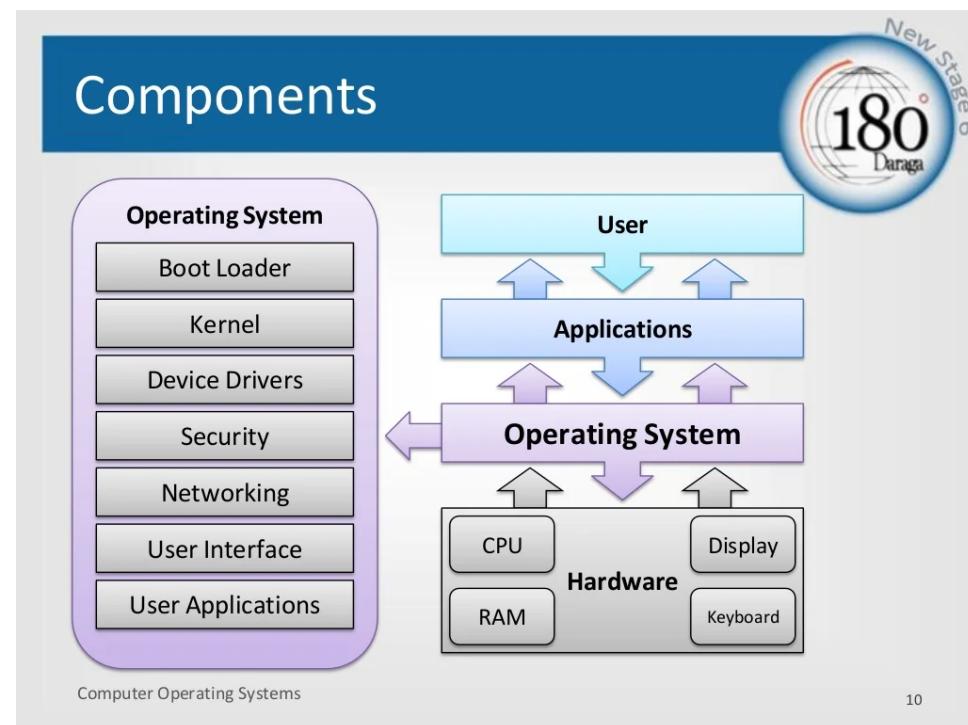




Hard RTOS...

- A hard real-time system has time-critical deadlines that must be met; otherwise a catastrophic system failure can occur.
- Absolutely, positively, first time every time
- Requires formal verification/guarantees of being to always meet its hard deadlines (except for fatal errors).
- Examples:
 - air traffic control
 - vehicle subsystems control
 - Nuclear power plant control

OS Structure

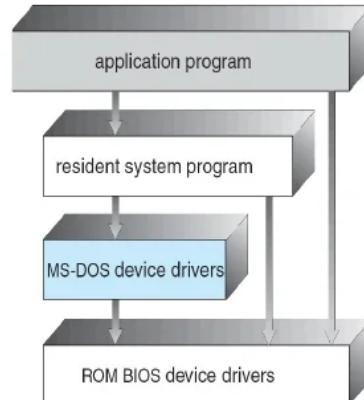


7. Operating-System Structure

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

7.1 Simple Structure

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



MS-DOS layer structure

Loganathan R, CSE , HKBKCE

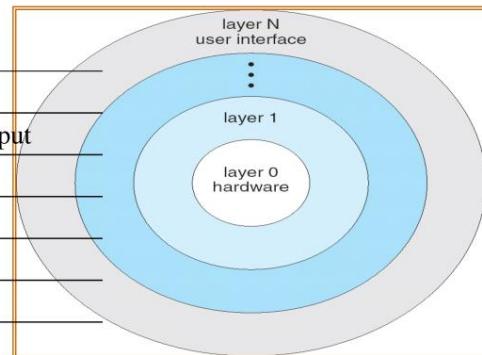
15

Layered Structure of the THE OS

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

Its six layers are as follows:

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



6

Operating System Layers

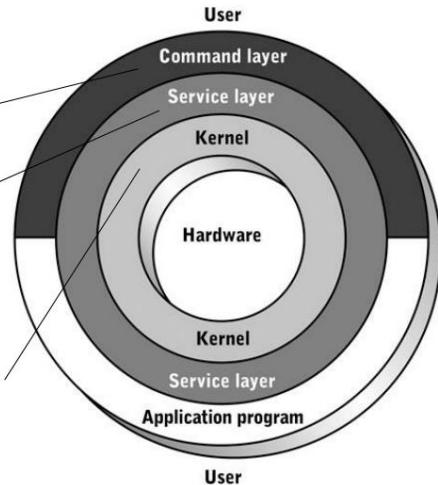
Figure 11-3 ►

Operating system layers (shaded)

User's interface to OS

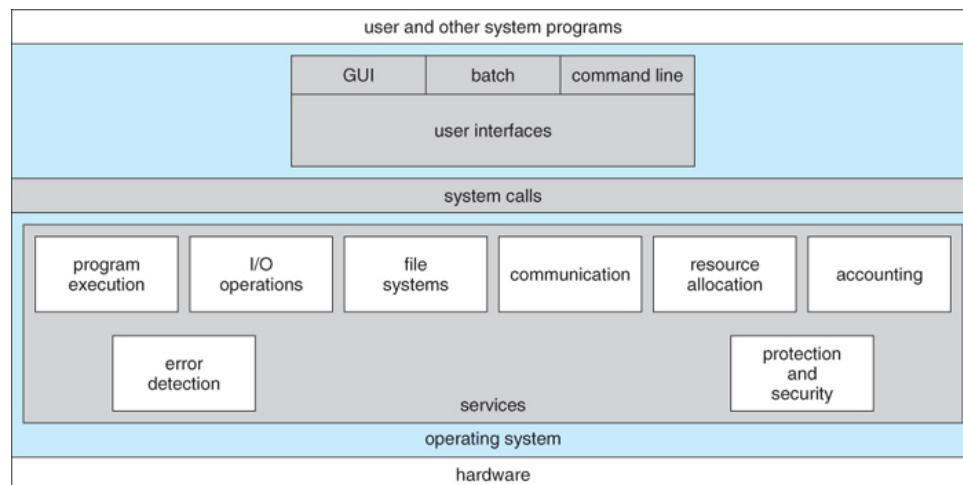
Contains set of functions executed by application programs and command layer

Manages resources; interacts directly with computer hardware

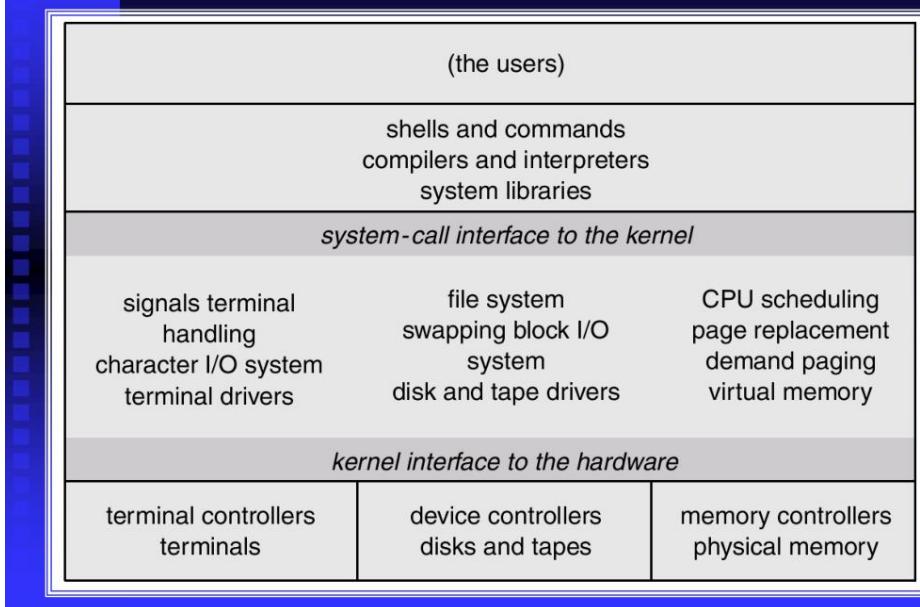


Hardware and Software Architecture

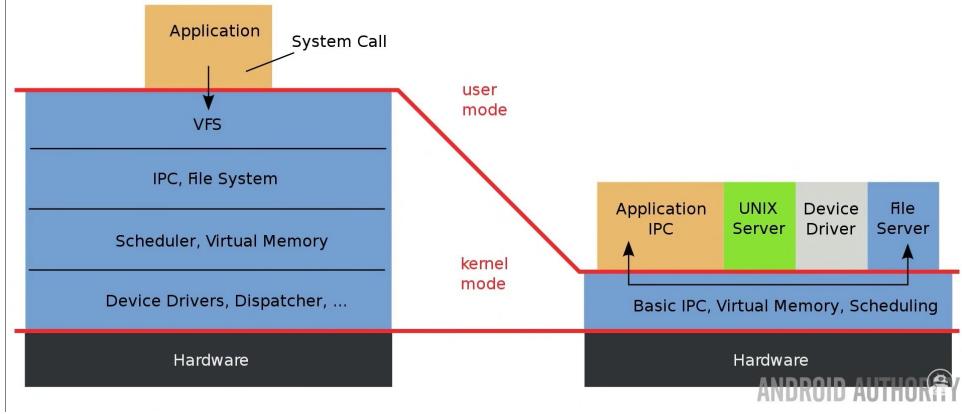
16



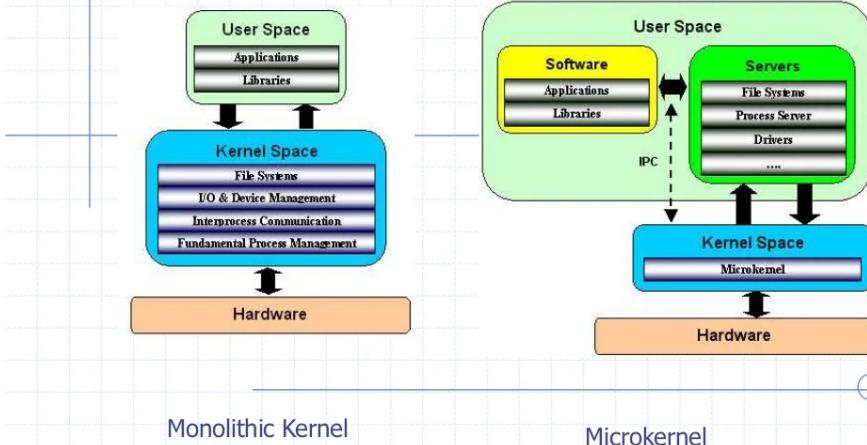
OS Structure: Monolithic structure



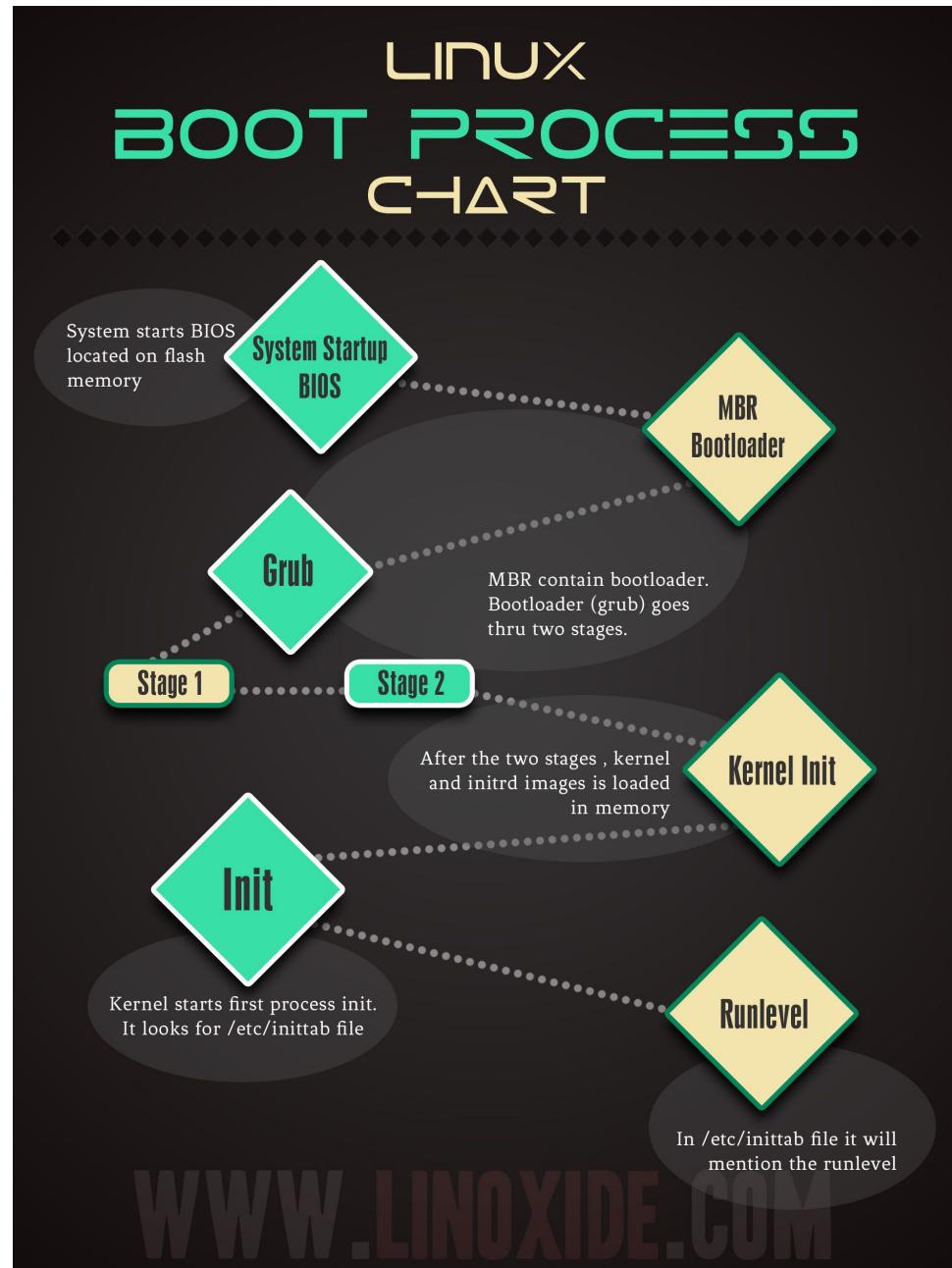
Monolithic Kernel based Operating System Microkernel based Operating System



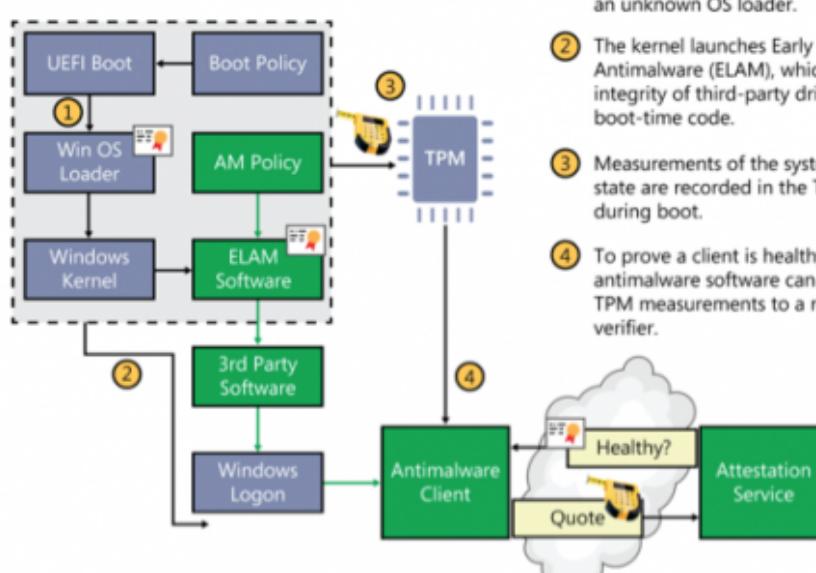
Monolithic kernel Vs Microkernel

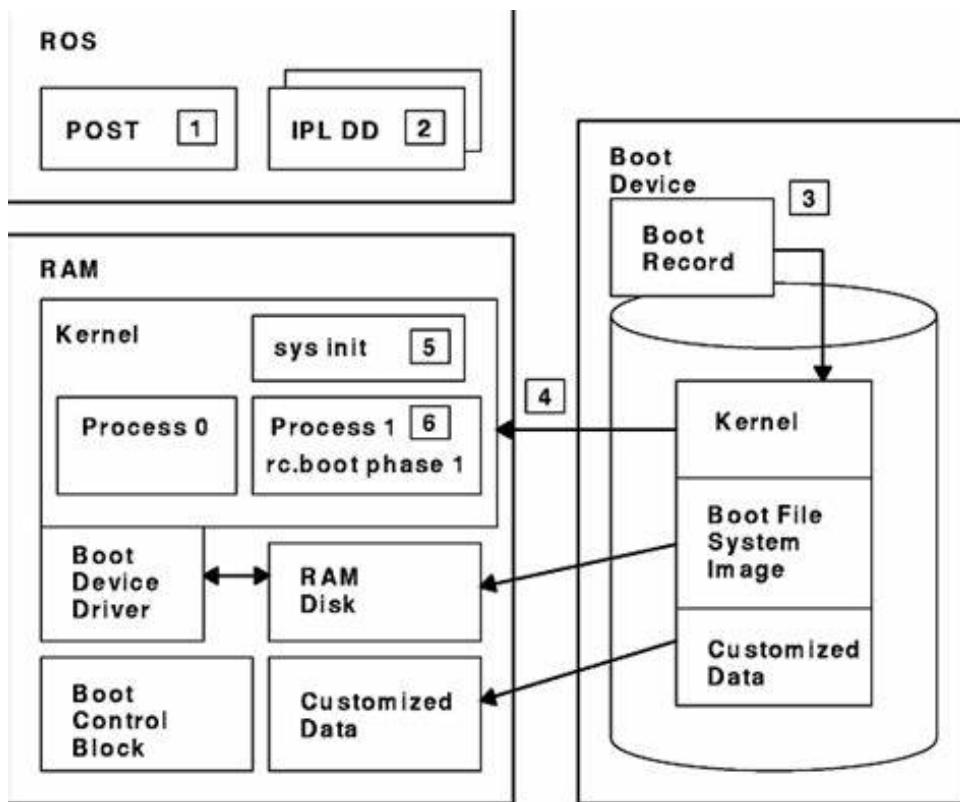
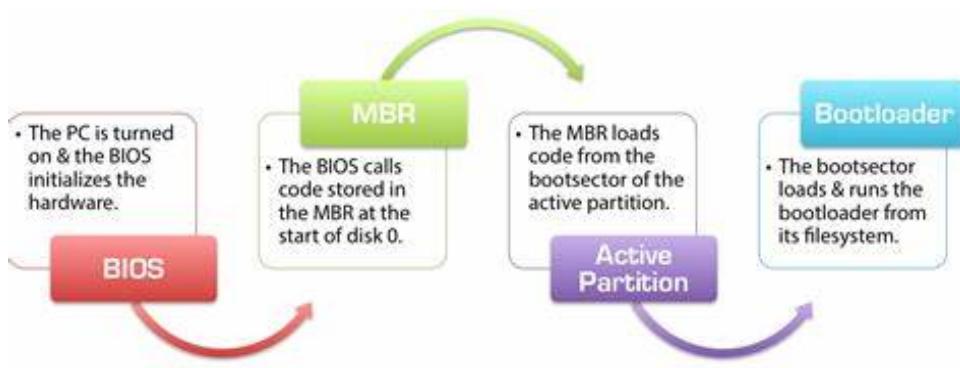
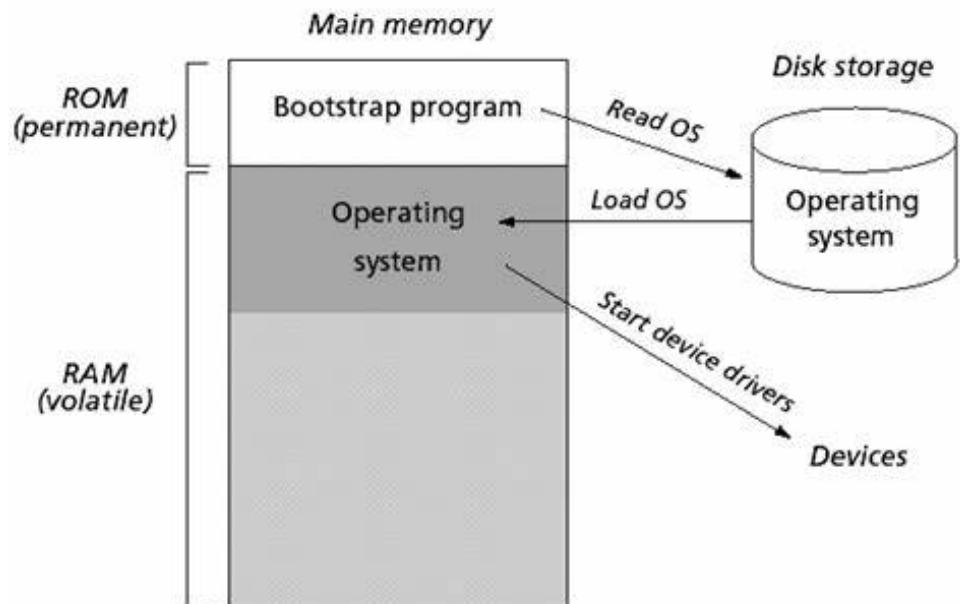


System Boot



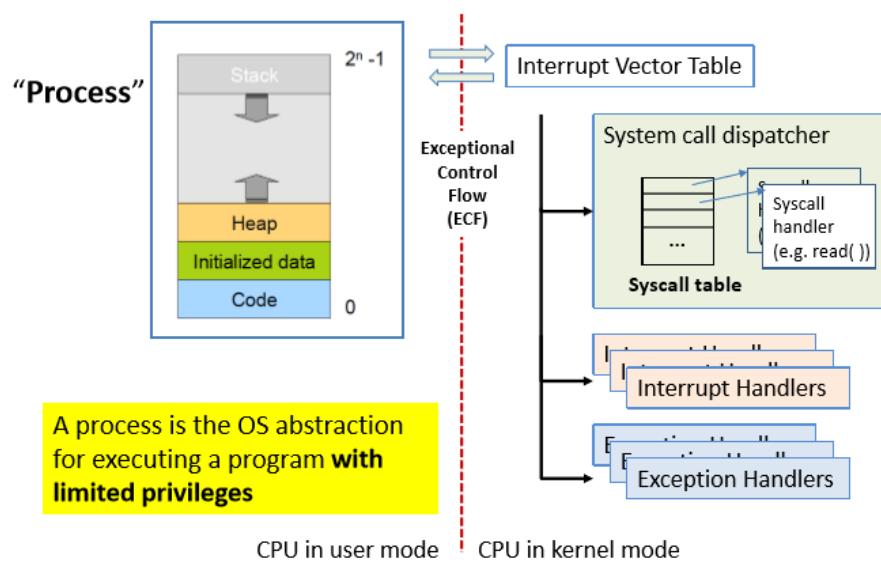
(Windows 8.1 and later)





ROS Kernel Init Phase

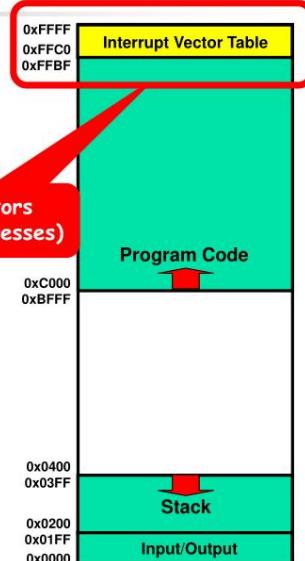
OS - Interrupt Driven



Interrupts

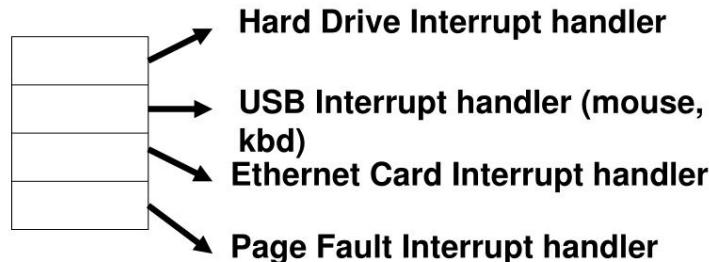
Interrupt Vectors

- The CPU must know where to fetch the next instruction following an interrupt.
- The address of an ISR is defined in an *interrupt vector*.
- The MSP430 uses *vectorized interrupts* where each ISR has its own vector stored in a *vector table* located at the end of program memory.
- Note: The *vector table* is at a fixed location (defined by the processor data sheet), but the ISRs can be located anywhere in memory.

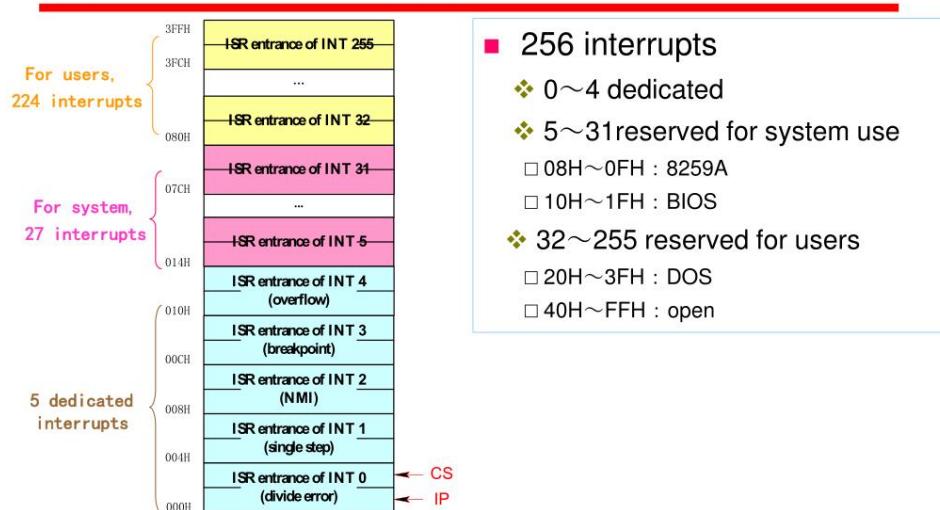


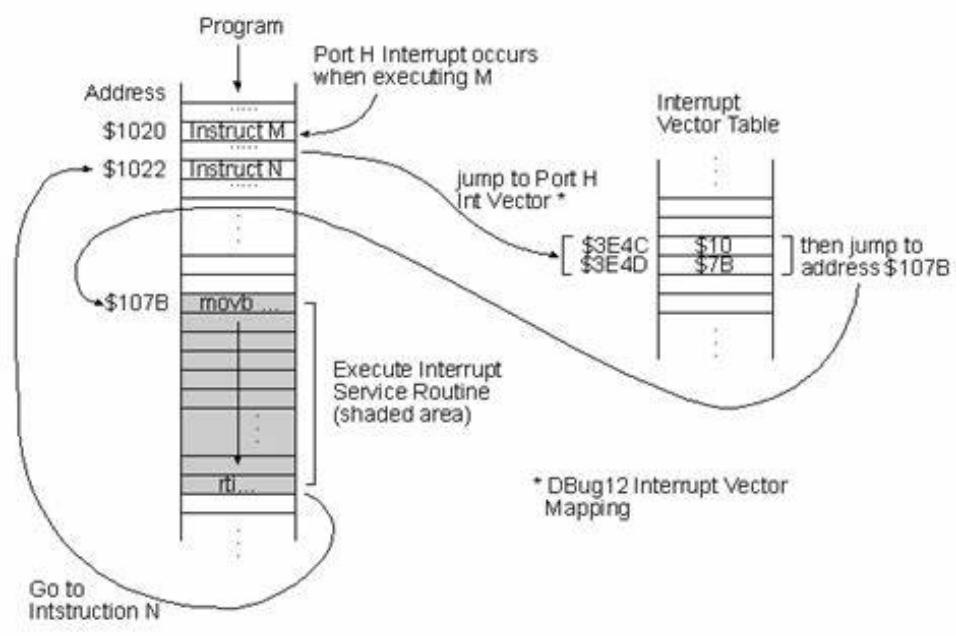
Interrupt Vector

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

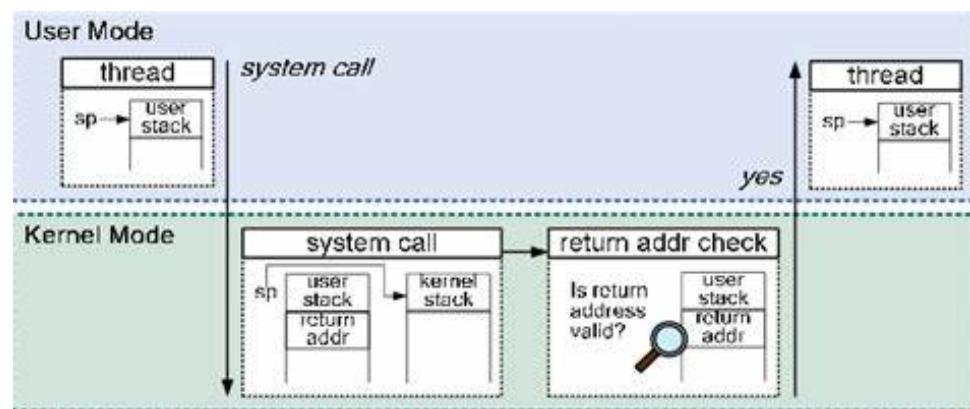
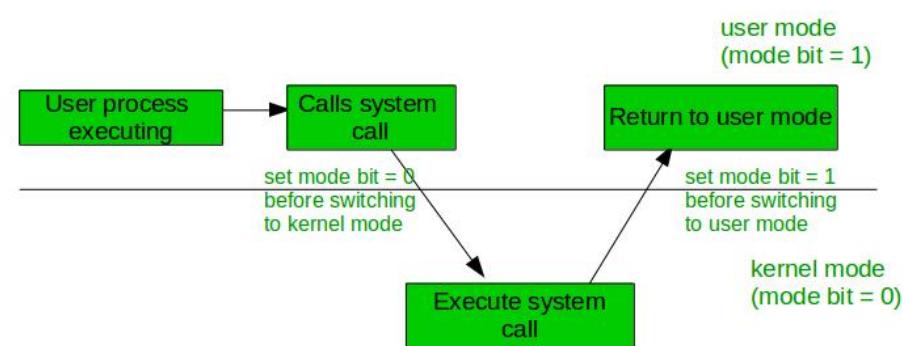


Interrupt Vector Table of 8086/8088





OS - Dual Mode Operation



SYSTEM CALLS



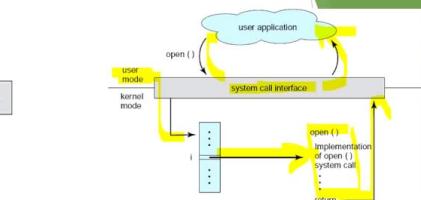
Example System Call Sequence

```

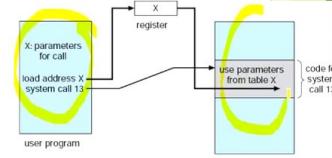
Acquire Input file name
Write prompt to screen
Accept Input
Acquire output file name
Write prompt to screen
Accept input
Open the input file
  if file doesn't exist, abort
  Create output file
  if file exists, abort
Loop
Read from input file
Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally

```

Example of how system calls are used.

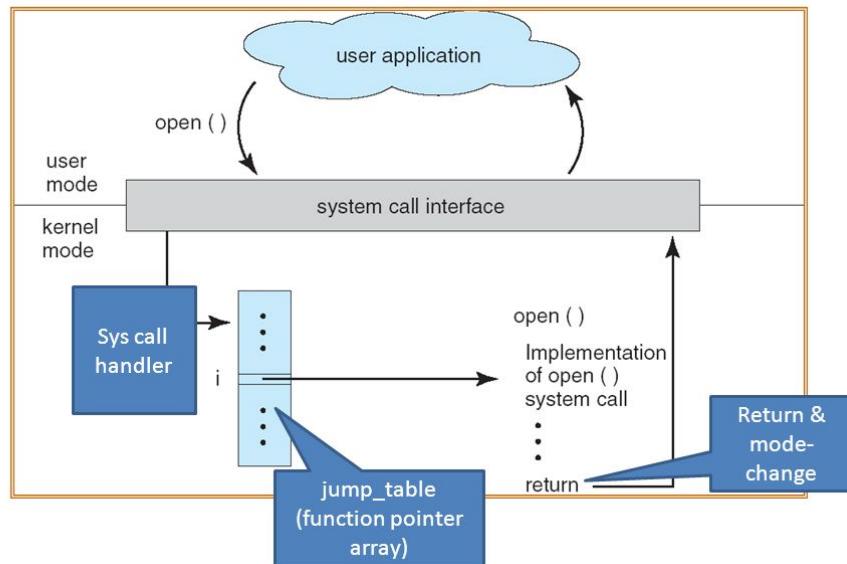


The handling of a user application invoking the `open()` system call



Passing of parameters as a table.

API – System Call – OS Relationship





Types of System Calls

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



Operating System Concepts - 7th Edition, Jan 14, 2005

2.21

Silberschatz, Galvin and Gagne ©2005

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

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

API VERSUS SYSTEM CALL

API

A set of protocols, routines, functions that programmers use to develop software to facilitate interaction between distinct systems

Helps to exchange data between various systems, devices and applications

SYSTEM CALL

A programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on

Allows a program to access services from the kernel of the operating system

Visit www.PEDIAA.com

SYSTEM CALL VERSUS LIBRARY CALL

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

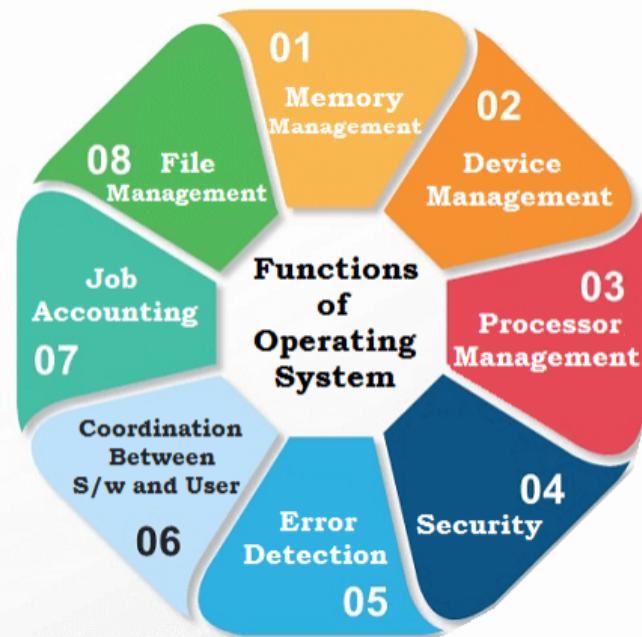
Visit www.PEDIAA.com

OPERATING SYSTEM VERSUS APPLICATION SOFTWARE

OPERATING SYSTEM	APPLICATION SOFTWARE
A system software that manages computer hardware and software resources and provides common services for computer programs	A software designed to perform a group of coordinated functions, tasks or activities for the benefit of the user
Works as the interface between the user and hardware, performs process management, memory management, task scheduling, hardware device controlling and many more	Performs a single specific task
Developed using C, C++, Assembly languages	Developed using Java, Visual Basic, C, C++
Boots up when the user switches on the computer and runs till he switches off the machine	Runs only when the user requests to run the application
Necessary for the proper functioning of the computer	Cannot be installed without an operating system
Ex: Windows, Unix, Linux, DOS	Ex: Word, Spreadsheet, Presentation, Multimedia tools, Database Management Systems

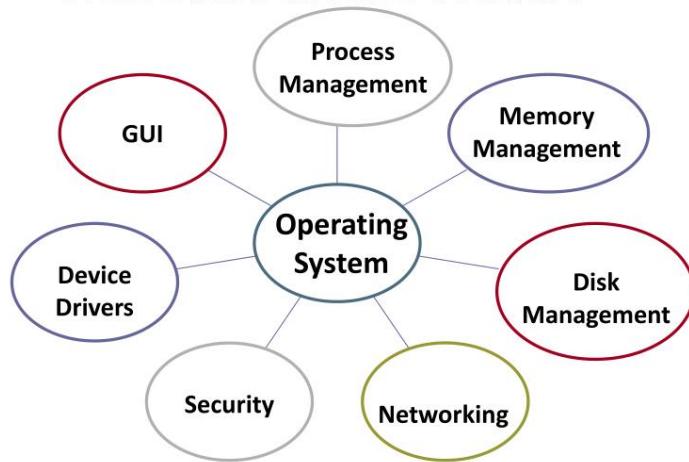
Visit www.PEDIAA.com

Functions of OS



The Operating System

- The OS is responsible for all the functions of hardware and also software



GUI

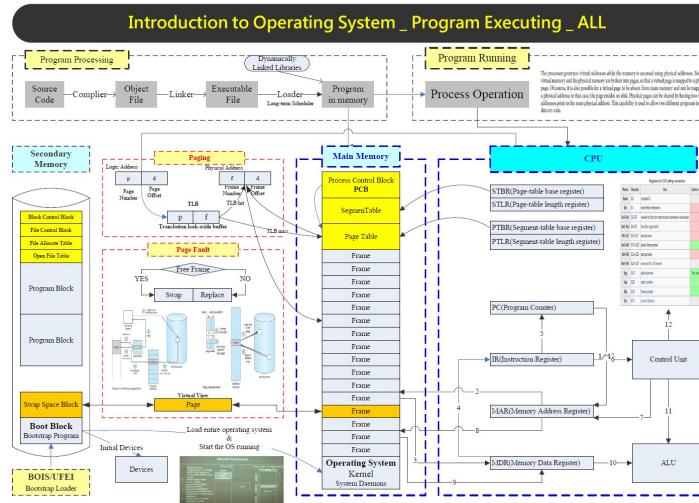
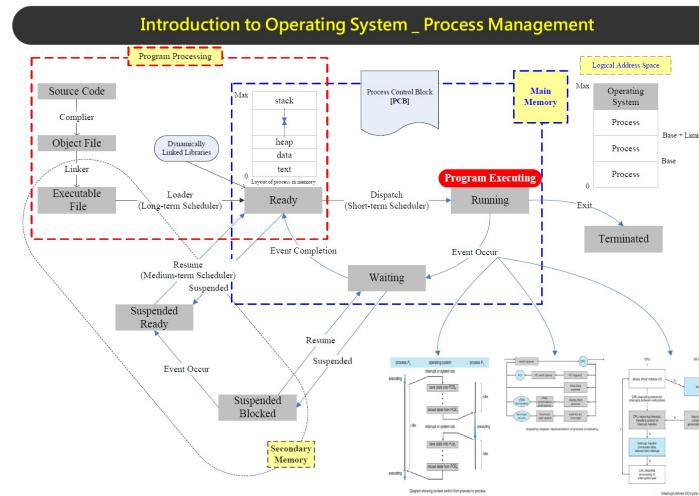
1. User Interface

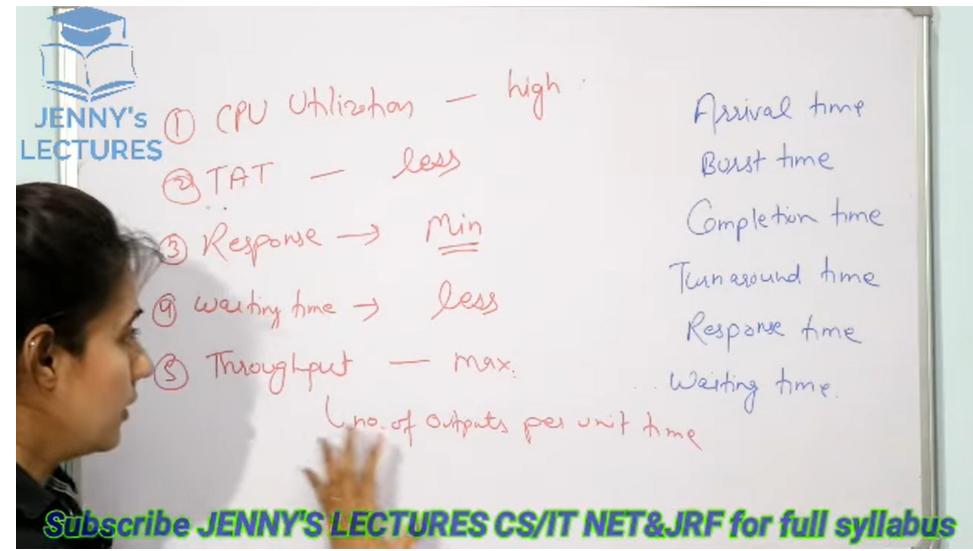
- Command Line Interface (CLI)
 - Interaction with text/commands
- Batch Files
 - Interaction with the help of batch files (.bat)
- Graphical User Interface (GUI)
 - User friendly
 - Easier to use : pointing device, menus etc
- Hybrid UI



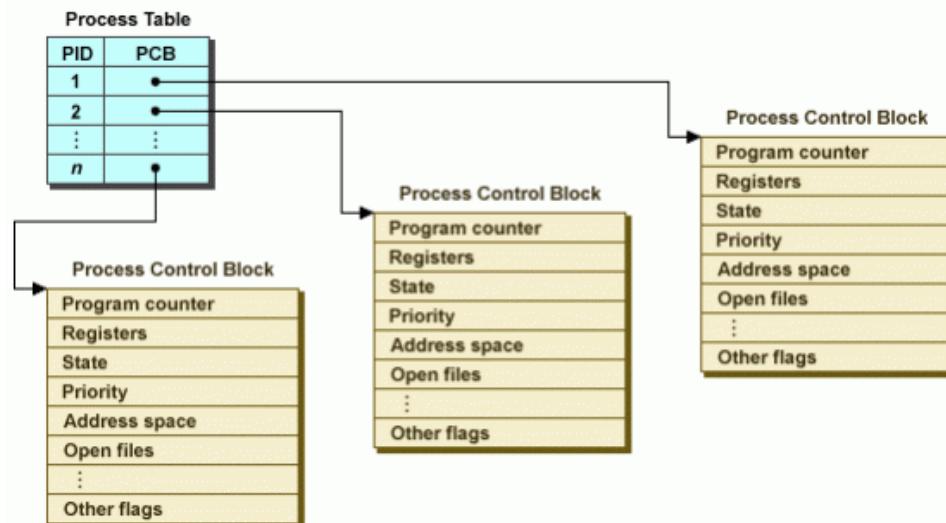
S

Process Management

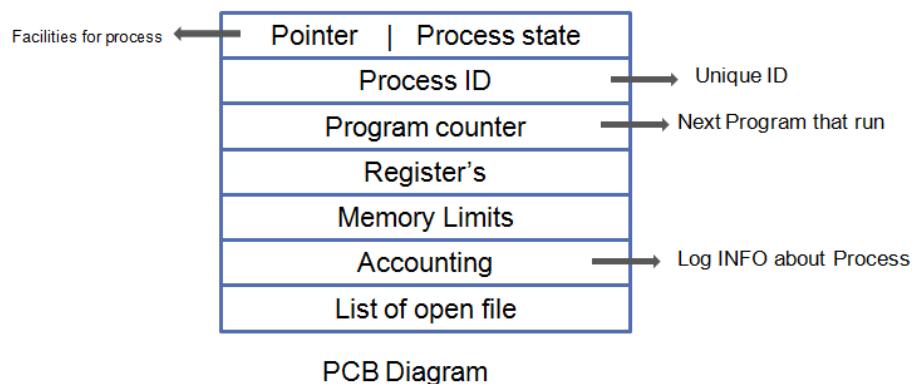




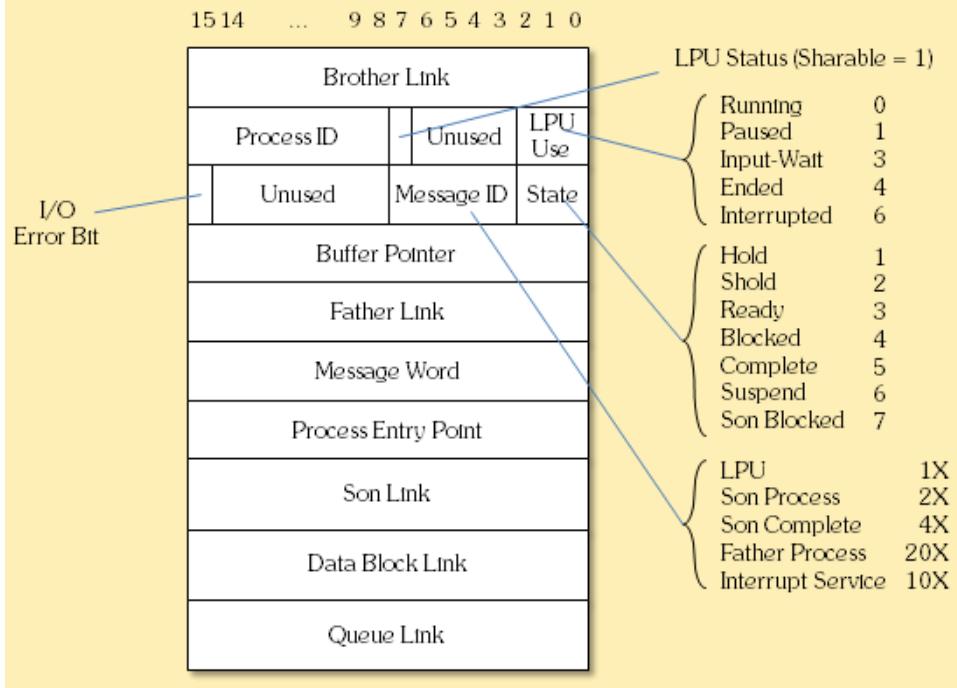
- PCB(Process Control Block)



PROCESS CONTROL BLOCK (PCB)



Process Control Block (PCB)



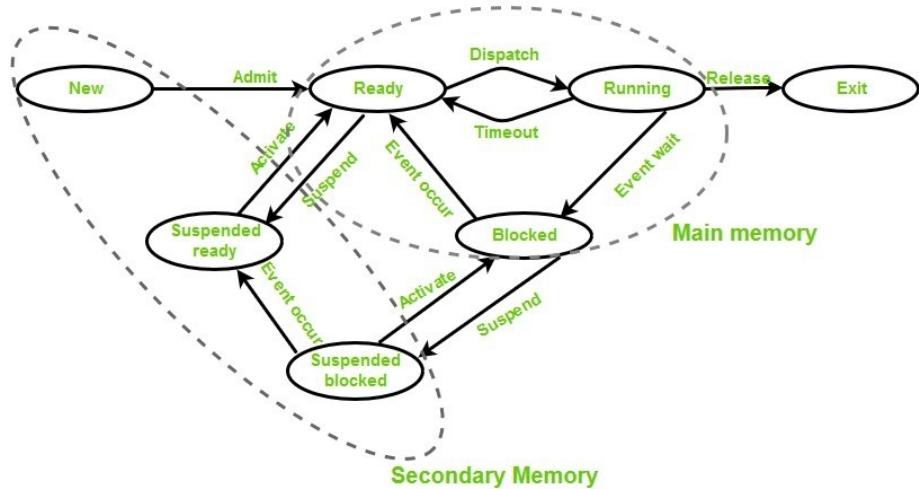
LPU Status (Sharable = 1)

Running	0
Paused	1
Input-Wait	3
Ended	4
Interrupted	6

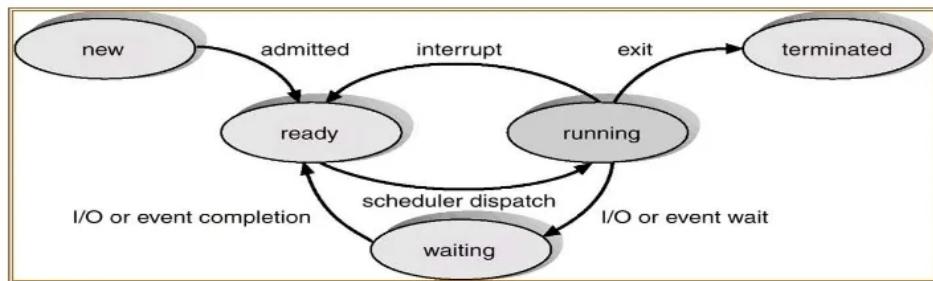
Hold	1
Shold	2
Ready	3
Blocked	4
Complete	5
Suspend	6
Son Blocked	7

LPU	1X
Son Process	2X
Son Complete	4X
Father Process	20X
Interrupt Service	10X

- Process State Flow



Process Management

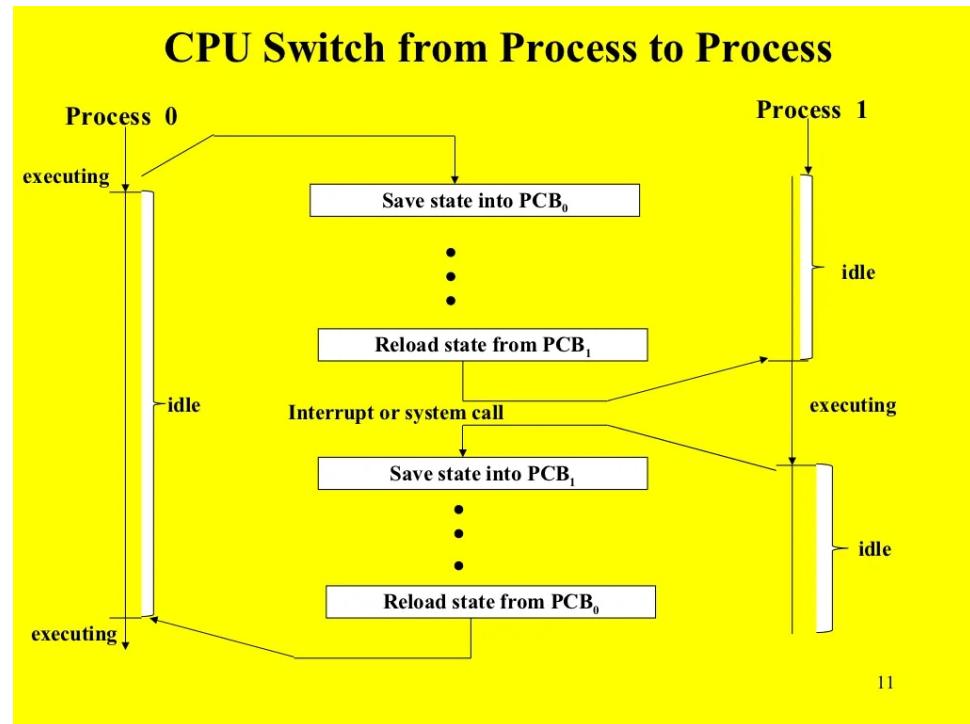


Process States

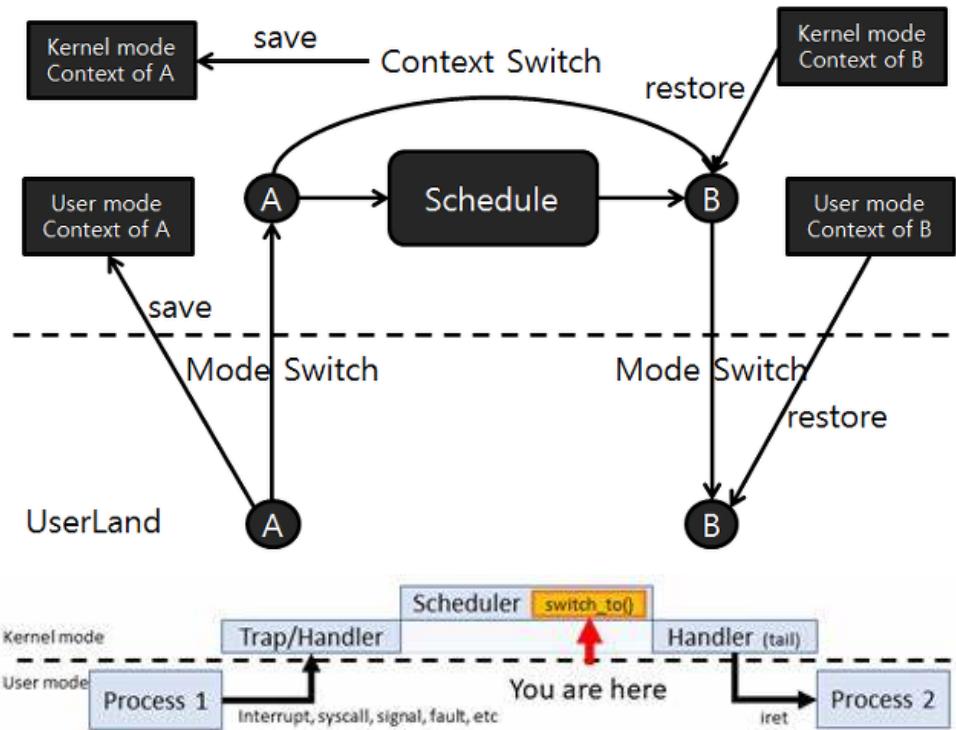
- New- The process is being created.
- Running- Instructions are being executed.
- Waiting- The process is waiting for some event to occur.
- Ready- The process is waiting to be assigned to a processor.
- Terminated- The process has finished execution.

Process Control Block- contains all information associated with a specific process like process state, program counter, CPU registers and info regarding CPU scheduling algorithms, memory, I/O and accounting.

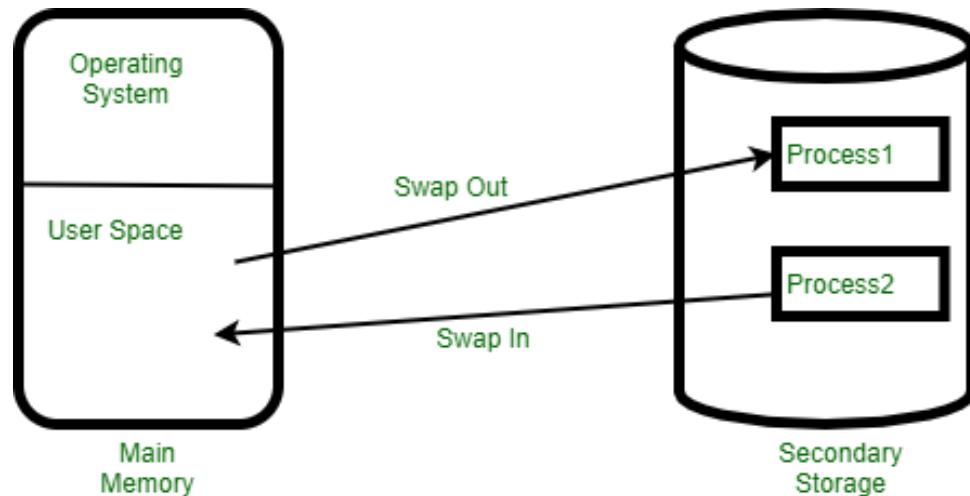
- Context Switch



KernelLand



- SWAP



Page replacement algorithm

- * Page replacement algorithms decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated.
- * Paging happens when a page fault occurs and a free page cannot be used to satisfy the allocation, either because there are none, or because the number of free pages is lower than some threshold.

Lecture Slides By Adil Aslam

FIFO Example

- Reference String is: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

7	7	7	2	2	2	4	4	4	0	0	0
0	0	0	3	3	3	2	2	2	1	1	
	1	1	1	0	0	0	3	3	3	2	

A callout box points to the value '1' in the third row, stating: "1 is Present in table so hit the page".

Page Fault : 1+1+1+1+1+1+1+1+1+1

Check the oldest page and replaced it. If it is not present in table

LEAST RECENTLY USED (LRU) ALGORITHM

reference string

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	4	4	4	0	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	3	3	3	2	2	2	2	2	2	2	2
1	1	1	1	1	3	3	2	2	2	2	2	2	2	2	2	2	2	2	7

page frames

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1	
7	7	7	2	2	4	4	4	0				1	1	1						
0	0	0	0	0	0	0	0	3	3		3	3	0	0						
1	1	1	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	

2

Problem-05: Optimal Page Replacement Algorithms

Optimal is a type of cache algorithm used to manage memory within a computer. OS replaces the page that will not be used for the longest period of time in future.

A system uses 4-page frames for storing process pages in main memory. It uses the **Optimal** page replacement policy. Assume that the first 4 frames have references 6,7,3,4. What is the total number of page faults that will occur while processing the page reference string given below;

6,7,1,2,3,4,1,2,4,3

5,3,2,5,6,7,1,2,3,3

so calculate the hit ratio and miss ratio.

frames	6	7	3	4	6	7	1	2	3	4	1	2	4	3	5	3	2	5	6	7	1	2	3	3
1	6	6	6	6	6	6	2	2	2	2	2	2	2	2										
2		7	7	7	7	7	1	1	1	1	1	1	1	1										
3			3	3	3	3	3	3	3	3	3	3	3	3										
4				4	4	4	4	4	4	4	4	4	4	4										
PF	x	x	x	x	/	/	x	x	/	/	/	/	/	/										

= miss =

I

: hit =

Belady's Anomaly

For FIFO algorithm, as the following counter-example shows, increasing m from 3 to 4 increases faults

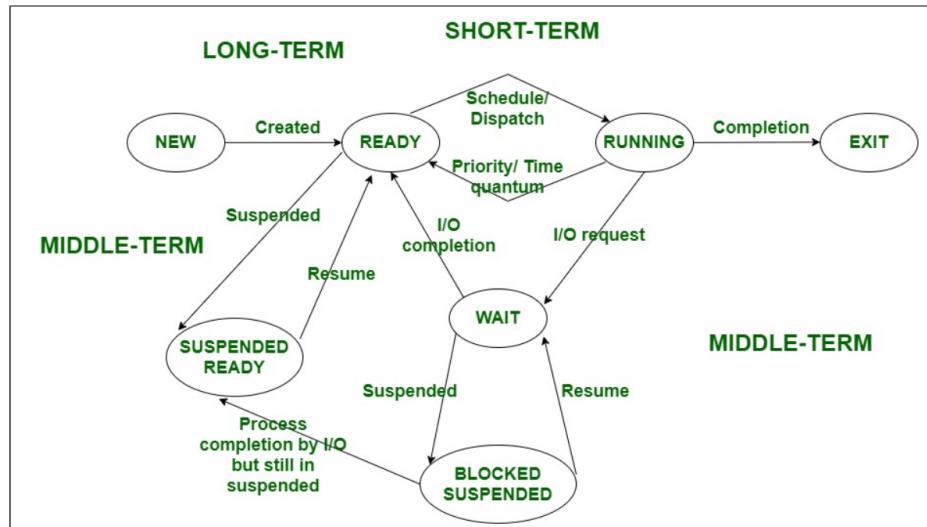
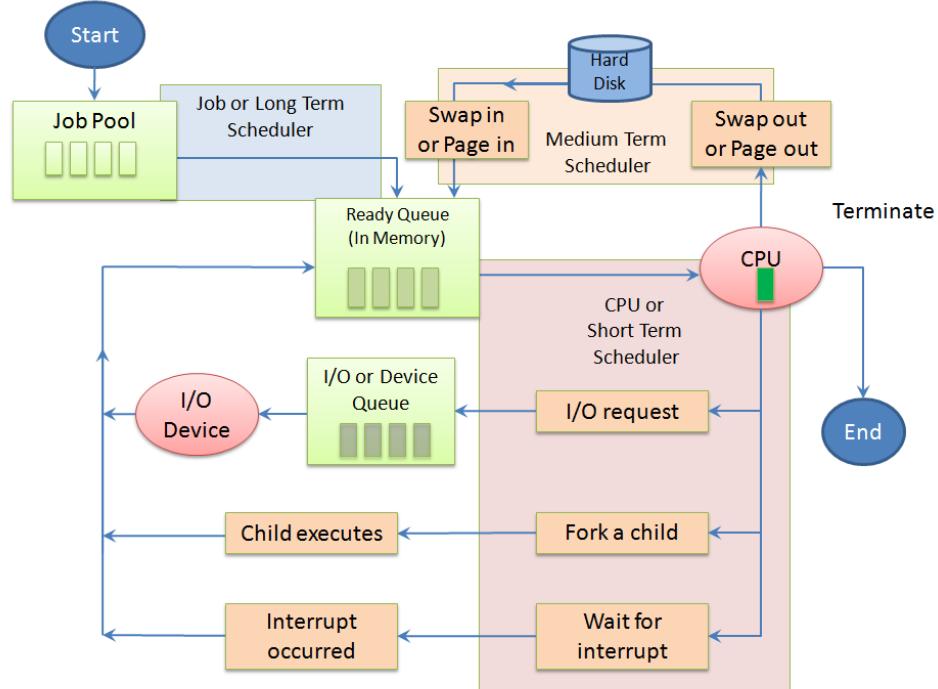
w		1	2	3	4	1	2	5	1	2	3	4	5											

		1	2	3	4	1	2	5	5	5	3	4	4	9	page									
m=3			1	2	3	4	1	2	2	2	5	3	3		faults									
			1	2	3	4	1	1	1	1	2	5	5											

		1	2	3	4	4	4	5	1	2	3	4	5	10	page									
m=4			1	2	3	3	3	4	5	1	2	3	4		faults									
			1	2	2	2	3	4	5	1	2	3	2											
			1	1	1	2	3	4	5	1	2	1	2											

36

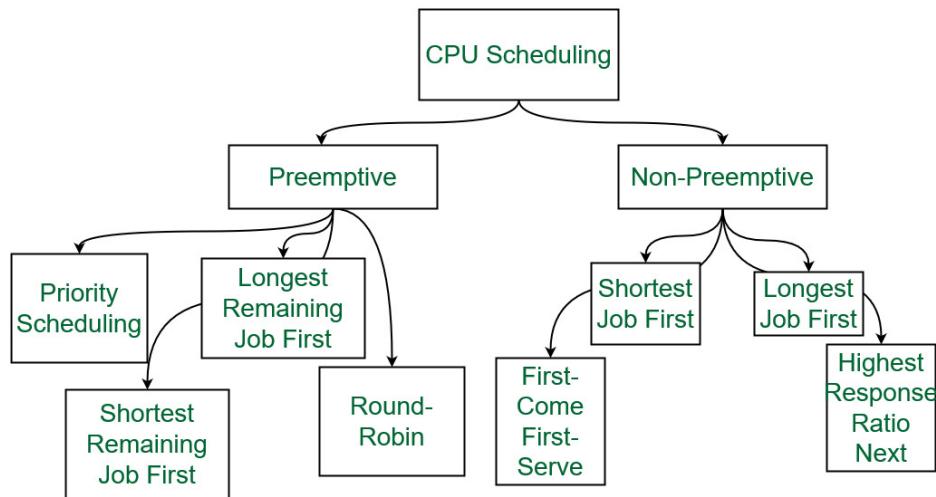
- Scheduling



Differentiate between short term and long term scheduler.

{**Note: Any other relevant difference shall be considered**}

Sr. No	Short term scheduler	Long term scheduler
1	It is a CPU scheduler	It is a job scheduler
2	It selects processes from ready queue which are ready to execute and allocates CPU to one of them.	It selects processes from job pool and loads them into memory for execution.
3	Access ready queue and CPU.	Access job pool and ready queue
4	It executes frequently. It executes when CPU is available for allocation.	It executes much less frequently. It executes when memory has space to accommodate new process.
5	Speed is fast	Speed is less than short term scheduler
6	It provides lesser control over degree of multiprogramming	It controls the degree of multiprogramming



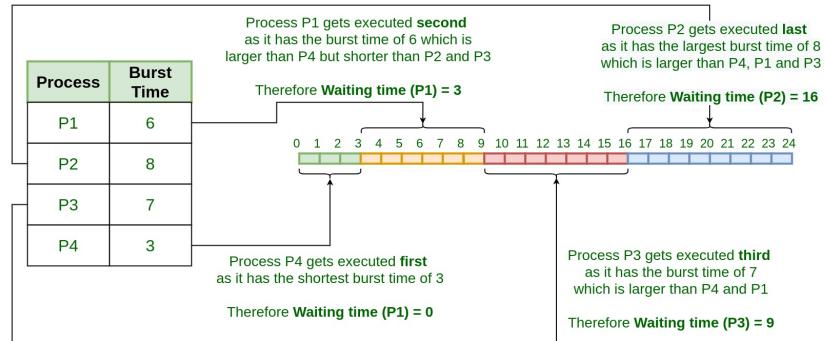
CPU SCHEDULING CRITERIA

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

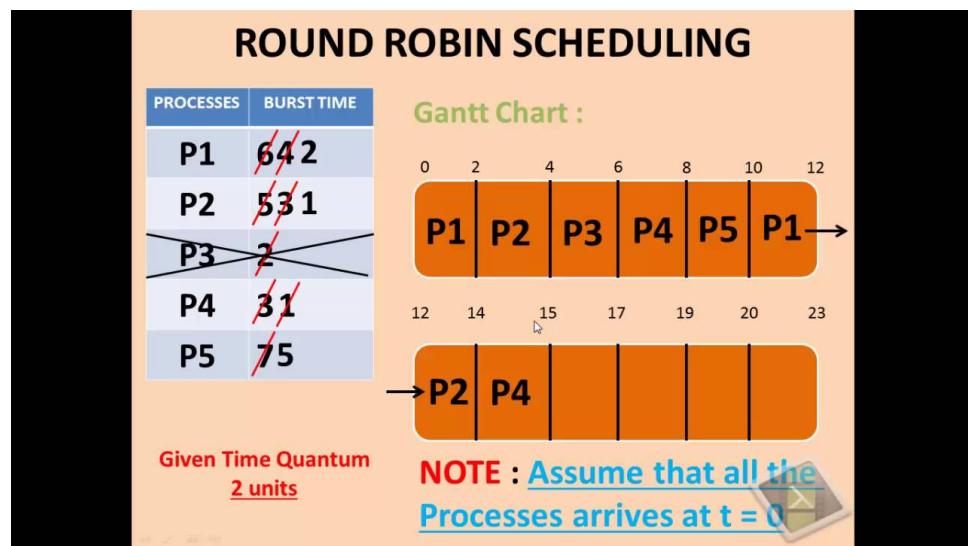
Round Robin(RR)

- Each process is allotted a time slot(q). After this time has elapsed, the process is pre-empted and added to the end of the ready queue.
- Performance of the round robin algorithm
 - q large \rightarrow FCFS
 - q small \rightarrow q must be greater than the context switch time; otherwise, the overhead is too high

Shortest Job First (SJF) Scheduling Algorithm



DG



CPU Scheduling (40 points)

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

Table 1: Process Information.

Consider the processes described in Table 1.

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

- (a) First Come First Serve (FCFS)
- (b) Non-preemptive Shortest Job First (NP-SJF)
- (c) Preemptive Shortest Job First (P-SJF)
- (d) Priority Scheduling
- (e) Round Robin, with the following assumptions:

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

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

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

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

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

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

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

	FCFS	SJF	Priority	RR
P1				
P2				
P3				
P4				
P5				

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

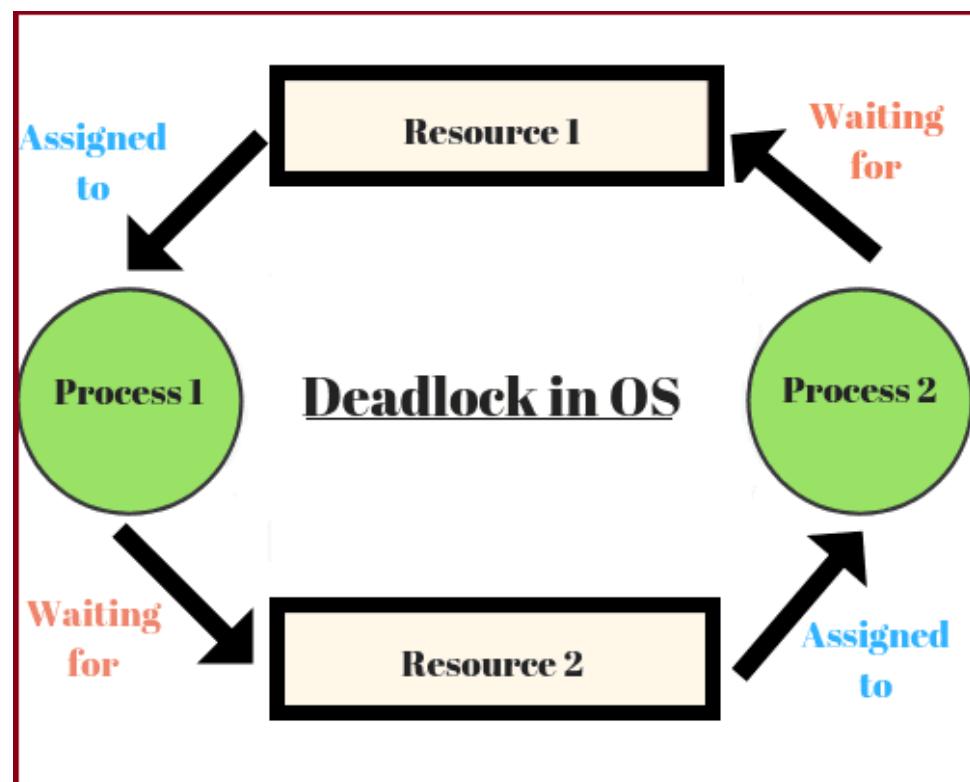
Highest Priority



Lowest Priority

Created by Notes Jam

-Deadlock



Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion:** only one process at a time can use a resource
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes
- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task
- **Circular wait:** there exists a set $\{P_0, P_1, \dots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1 , P_1 is waiting for a resource that is held by P_2, \dots, P_{n-1} is waiting for a resource that is held by P_n , and P_n is waiting for a resource that is held by P_0 .

7.5

- Resource Allocation Graph

Resource-Allocation Graph: Definition

A set of vertices V and a set of edges E .

- V is partitioned into two types:
 - $P = \{P_1, P_2, \dots, P_n\}$, the set consisting of all the processes in the system.
 - $R = \{R_1, R_2, \dots, R_m\}$, the set consisting of all resource types in the system.
- request edge – directed edge $P_i \rightarrow R_j$
- assignment edge – directed edge $R_j \rightarrow P_i$

The resource-allocation graph is therefore a bipartite directed graph. *What would be the graph representing the bridge-crossing example?*

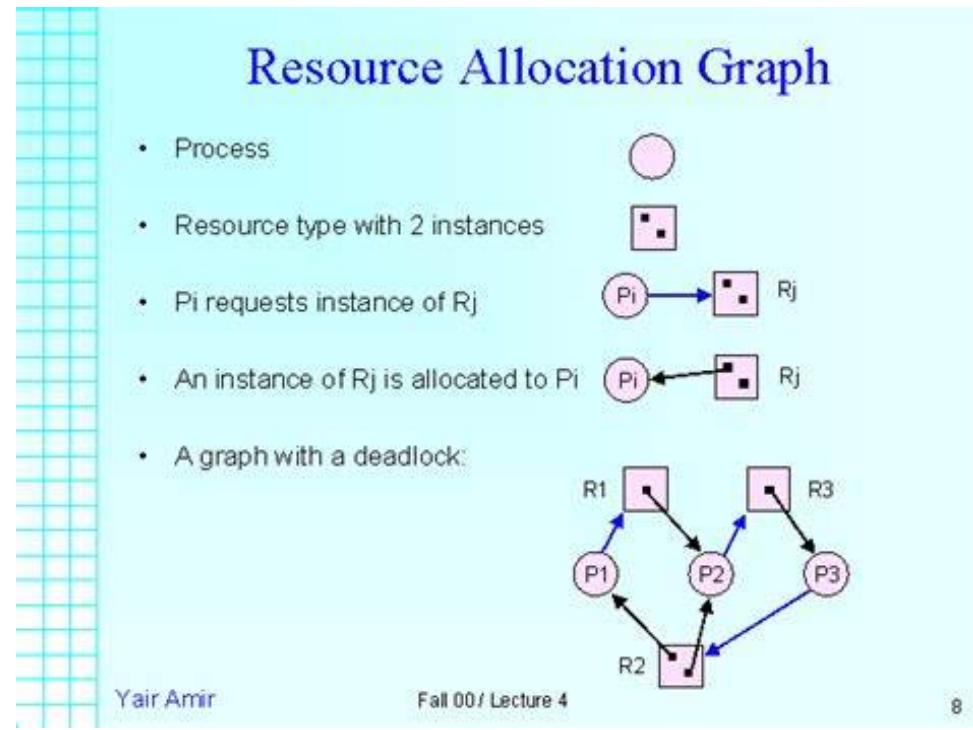
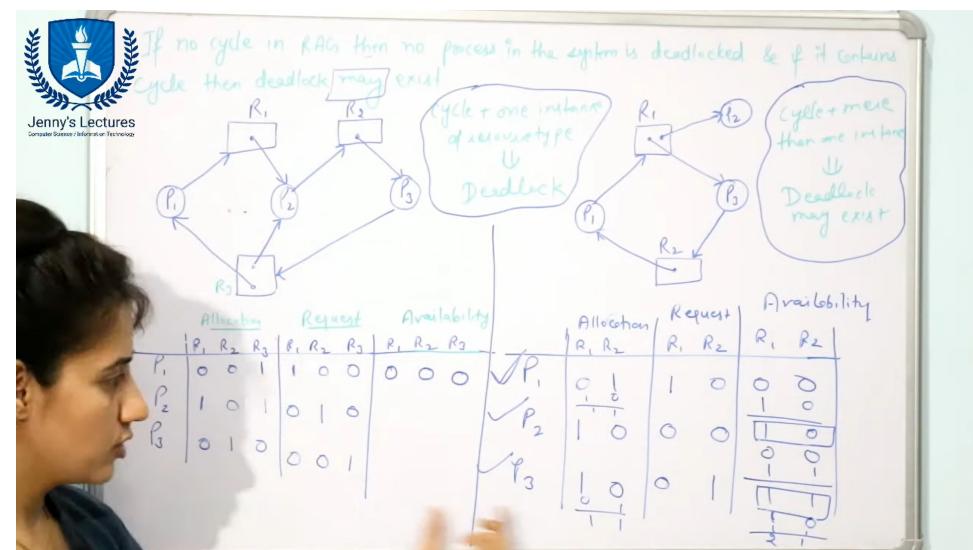
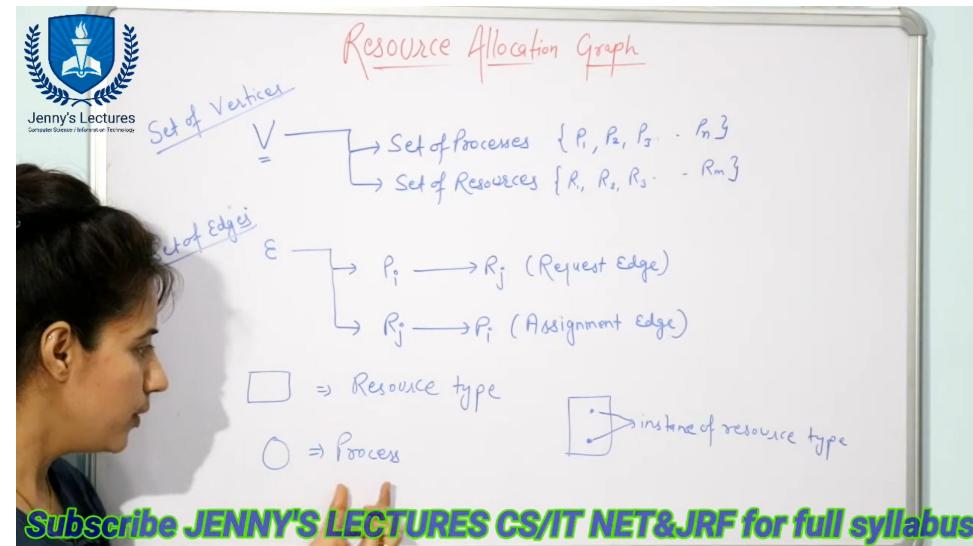


Operating System Concepts with Java – 7th Edition, Nov 15, 2006

7.8

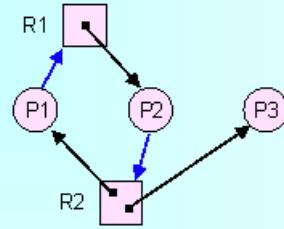


Silberschatz, Galvin and Gagne ©2007



Resource Allocation Graph (cont.)

- A graph with a cycle but without a deadlock:



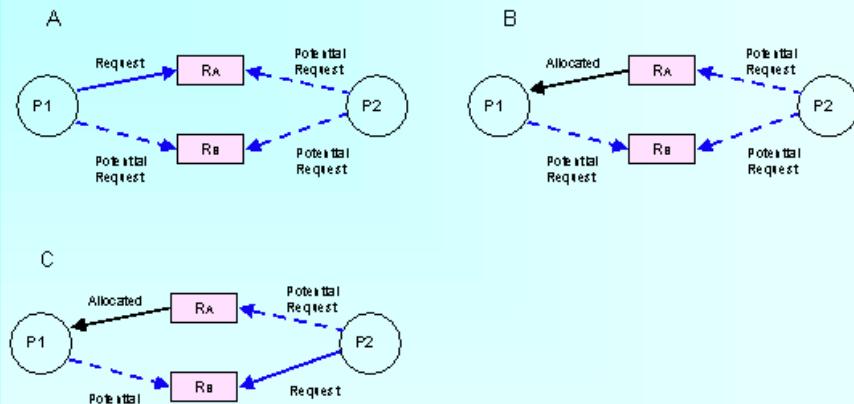
- If there are no cycles then there is no deadlock.
- If there is a cycle:
 - If there is only one instance per resource type then there is a deadlock.**
 - If there is more than one instance for some resource type, there may or may not be a deadlock.**

Yair Amir

Fall 00 / Lecture 4

9

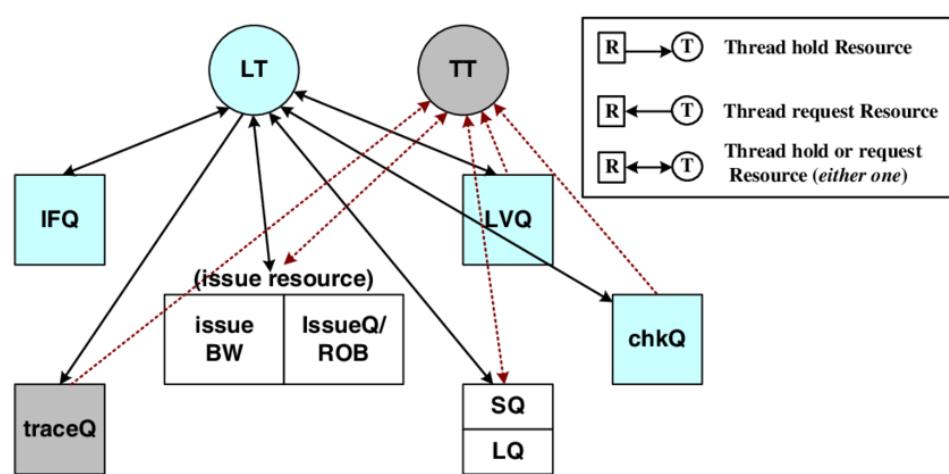
Resource Allocation Graph Algorithm (cont.)



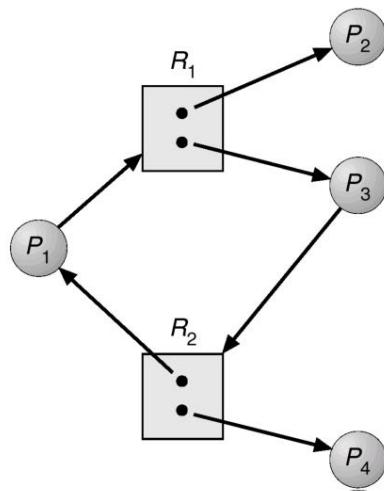
Yair Amir

Fall 00 / Lecture 4

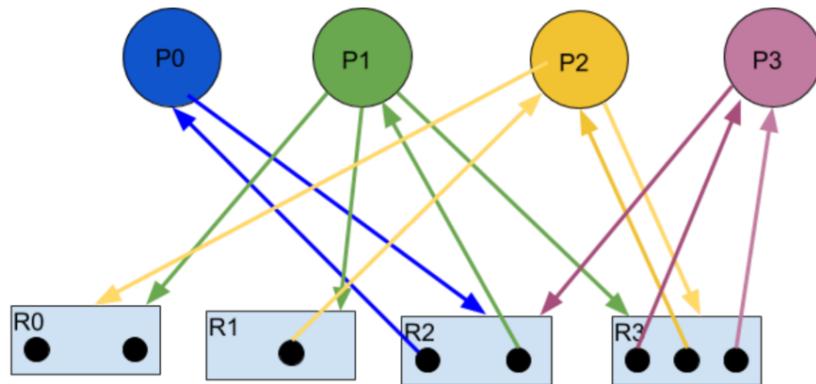
20



Resource Allocation Graph With A Cycle But No Deadlock



- If graph contains no cycles \Rightarrow no deadlock.
- If graph contains a cycle \Rightarrow
 - if only one instance per resource type, then deadlock.
 - if several instances per resource type, possibility of deadlock.



[6 pts] Fill in the resource matrix according to the graph above:

	Allocation				Need				Available			
	R0	R1	R2	R3	R0	R1	R2	R3	R0	R1	R2	R3
P0												
P1												
P2												
P3												

Approaches to Deadlock Prevention

Condition	Approach
Mutual exclusion	Spool everything
Hold and wait	Request all resources initially
No preemption	Take resources away
Circular wait	Order resources numerically

Figure 6-14. Summary of approaches to deadlock prevention.

Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. All rights reserved. 0-13-6006639

Deadlock Avoidance

- Deadlock avoidance is a technique used to avoid deadlock.
- It requires information about how different processes would request different resources.
- **Safe state:** if deadlock not occur then safe state.
- **Unsafe state:** if deadlock occur then unsafe state.
- The idea of avoiding a deadlock is simply not allow the system to enter an unsafe state the may cause a deadlock.

3.5 Deadlock Detection

- If a system does not employ either a deadlock-prevention or a deadlock avoidance algorithm, then a deadlock situation may occur.
- In this environment, the system may provide:
 - An algorithm that examines the state of the system to determine whether a deadlock has occurred.
 - An algorithm to recover from the deadlock.
 - Single Instance of Each Resource Type.
 - Several Instances of a Resource Type.
 - When should we invoke the detection algorithm?
 - The answer depends on two factors:
 1. How *often* is a deadlock likely to occur?
 2. How *many* processes will be affected by deadlock when it happens?

20

The Difference Between Deadlock Prevention and Deadlock Avoidance

- **Deadlock Prevention:**
 -  – Preventing deadlocks by constraining how requests for resources can be made in the system and how they are handled (system design).
 - The goal is to ensure that at least one of the necessary conditions for deadlock can never hold.
- **Deadlock Avoidance:**
 -  – The system dynamically considers every request and decides whether it is safe to grant it at this point;
 - The system requires additional apriori information regarding the overall potential use of each resource for each process.
 - Allows more concurrency

Similar to the difference between a traffic light and a police officer directing traffic.

Yair Amir

Fall 00 / Lecture 4

11

Comparison Deadlock

- Definition
- Deadlock occurs when none of the processes in the set is able to move ahead due to occupancy of the required resources by some other process. Or Deadlock is where no process proceeds, and get blocked
- Other name
- Circular waiting
- Arising conditions
- These four conditions arising simultaneously – mutual exclusion, hold and wait, no-preemption and circular wait
- Avoidance/ prevention Techniques
- Infinite resources, Waiting is not allowed, Sharing is not allowed, Preempt the resources, All Requests made at the starting

Starvation

- Starvation occurs when a process waits for an indefinite period of time to get the resource it requires. Or Starvation is where low priority processes get blocked, and high priority process proceeds.
- Lived lock
- Uncontrolled management of resources, Process priorities being strictly enforces Use of random selection, Scarcity of resources

BetweenMates.com

Bankers' Algorithm

The banker's algorithm

- A state is safe iff there exists a sequence $\{P_1..P_n\}$ where each P_i is allocated all of its needed resources to be run to completion
 - ◆ i.e.: we can always run all the processes to completion from a safe state
- The safety algorithm is the part that determines if a state is safe
- Initialization:
 - ◆ all processes are said to be “unfinished”
 - ◆ set the work vector to the amount resources available: $W(i) = V(i)$ for all i ;

41

Banker's Algorithm

1. Look for a new row in R which is smaller than A . If no such row exists the system will eventually deadlock → not safe.
2. If such a row exists, the process may finish. mark that process (row) as terminate and add all of its resources to A .
3. Repeat Steps 1 and 2 until all rows are marked → safe state
If some are not marked → not safe.

30

Example of Banker's Algorithm

	Allocation A B C				Need A B C				Available A B C		
P0	0	1	0	P0	7	4	3	P0	3	3	2
P1	2	0	0	P1	1	2	2	P1			
P2	3	0	2	P2	6	0	0	P2			
P3	2	1	1	P3	0	1	1	P3			
P4	0	0	2	P4	4	3	1	P4			

Try to find a row in *Need*, that is \leq Available.

- P1. run completion. Available becomes = $[3\ 3\ 2] + [2\ 0\ 0] = [5\ 3\ 2]$
- P3. run completion. Available becomes = $[5\ 3\ 2] + [2\ 1\ 1] = [7\ 4\ 3]$
- P4. run completion. Available becomes = $[7\ 4\ 3] + [0\ 0\ 2] = [7\ 4\ 5]$
- P2. run completion. Available becomes = $[7\ 4\ 5] + [3\ 0\ 2] = [10\ 4\ 7]$
- P0. run completion. Available becomes = $[10\ 4\ 7] + [0\ 1\ 0] = [10\ 5\ 7]$

We found a sequence of execution: P1, P3, P4, P2, P0. State is safe

40

Banker's Algorithm for a single resource



Has Max		
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10	Free: 2	Free: 1
Any sequence finishes	C,B,A,D finishes	Deadlock (unsafe state)

A	1	6
B	1	5
C	2	4
D	4	7

- Bankers' algorithm: before granting a request, ensure that a sequence exists that will allow all processes to complete
 - Use previous methods to find such a sequence
 - If a sequence exists, allow the requests
 - If there's no such sequence, deny the request
- Can be slow: must be done on each request!

-
1. Using the banker's algorithm, determine whether the following state is unsafe based on the snapshot of the system below. If the state is safe, provide a safe sequence of execution. Otherwise explain why it is unsafe. Show your calculations for full credit.

[10 points]

	<u>Allocation</u>				<u>Max (Demand)</u>				Available = (1, 0, 0, 2)
	A	B	C	D	A	B	C	D	
P ₀	3	0	1	4	5	1	1	7	
P ₁	2	2	1	0	3	2	1	1	
P ₂	3	1	2	1	3	3	2	1	
P ₃	0	5	1	0	4	6	1	2	
P ₄	4	2	1	2	6	3	2	5	

Sample question (bankers algorithm)

	Allocation				Max				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P0	4	0	0	1	7	0	2	1					3	2	2	1
P1	1	1	0	0	1	6	5	0								
P2	1	0	4	5	3	3	4	6								
P3	0	4	2	1	1	5	6	2								
P4	0	3	1	2	2	4	3	2								

Using Bankers algorithm answer the following:

1. How many resources of type A, B, C and D are there?
2. What are the contents of the Need matrix?
3. Is the system in a safe state? Provide reasoning for your answer (show the sequence in which the processes would finish)
4. If a request from process P2 arrives for additional resources of {0, 2, 0, 0}, can the Bankers algorithm grant the request immediately? Provide reasoning for your answer. (14 Marks)

Q1. Deadlocks. The Banker's algorithm is used for deadlock avoidance. Consider the state of resource availability and allocation defined by the following matrices.

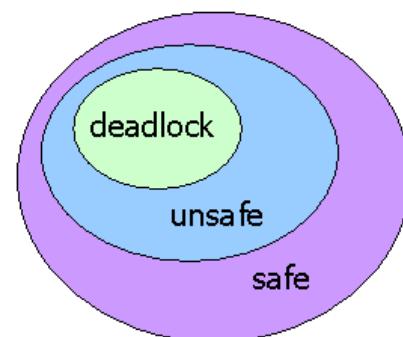
Claim Matrix			
	R1	R2	R3
P1	3	1	4
P2	6	1	3
P3	3	2	2
P4	4	2	2

Allocation Matrix			
	R1	R2	R3
P1	2	1	1
P2	5	1	1
P3	2	0	1
P4	0	0	2

- (1) Assuming that the total amounts for resources R1, R2, and R3 are 10, 2, and 10, should a new request to the Banker's algorithm by process P3 to acquire one additional resource from R1 and one additional resource from R3 be approved or denied? Explain why or why not.
- (2) Assuming that the total amounts for resources R1, R2, and R3 are 10, 2, and 10, should a new request to the Banker's algorithm by process P4 to acquire one additional resource from R3 be approved or denied? Explain why or why not.
- (3) Assuming that the total amounts for resources R1 and R2 are 10 and 2, what is the minimum amount for resource R3 that would render the above state a safe state under the Banker's algorithm?
- (4) Given your answer for part (3) what are all the possible orderings for the four processes P1, P2, P3, and P4 to complete their execution subject to the Banker's algorithm.
- (5) Assuming that the total amounts for resources R1 and R2 are 10 and 2, what is the minimum amount for resource R3 that would make it possible for the Banker's algorithm to allow process P1 to complete its execution before all other three processes?

The Banker's Algorithm

- Idea: know what each process *might* ask for
- Only make allocations that leave the system in a *safe* state
- Inefficient



Resource allocation state space



Jenny's Lectures
Computer Science / Information Technology

	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P ₁	0	0	1	2	0	0	1	2	1	5	2	0	0	0	0	0
P ₂	1	0	0	0	1	7	5	0	1	5	3	2	0	7	5	0
P ₃	0	1	3	5	4	2	3	5	6	2	8	8	6	1	0	2
P ₄	0	0	1	4	0	6	5	6	2	1	4	1	2	0	6	4
	2	9	10	12												

$\text{Need}_i = \text{Allocation}_i - \text{Max}_i$

total :- $\begin{array}{cccc} A & B & C & D \\ \hline 3 & 14 & 12 & 12 \end{array}$



Jenny's Lectures
Computer Science / Information Technology

	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P ₁	0	0	1	2	0	0	1	2	1	5	2	0	0	0	0	0
P ₂	1	0	0	0	1	7	5	0	1	5	3	2	0	7	5	0
P ₃	0	1	3	5	4	2	3	5	6	2	8	8	6	1	0	2
P ₄	0	0	1	4	0	6	5	6	2	1	4	1	2	0	6	4
	2	9	10	12												

Sequence :- $\boxed{P_0 \ P_2 \ P_3 \ P_4 \ P_1}$

$\text{Need}_i = \text{Allocation}_i - \text{Max}_i$



Jenny's Lectures
Computer Science / Information Technology

	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P ₁	0	0	1	2	0	0	1	2	1	5	2	0	0	0	0	0
P ₂	1	0	0	0	1	7	5	0	1	5	3	2	0	7	5	0
P ₃	0	1	3	5	4	2	3	5	6	2	8	8	6	1	0	2
P ₄	0	0	1	4	0	6	5	6	2	1	4	1	2	0	6	4
	2	9	10	12												

also:- Input :- Processes

- any 2 out of 3 (Max, Need, Allocation)
- available or total no. of resources
- flag[i]=0 for i=0 to (n-1) & find $\text{Need}[n][m] = \text{Max}[n][m] - \text{Allocation}[n][m]$
- if such i exists then
 - flag[i]=1 & Need_i <= Available
 - if such i exists then
 - flag[i]=1, available = available + Allocate_i
 - goto step 1

n : no. of processes
 m : no. of resources

 Jenny's Lectures Computer Science / Information Technology

	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P ₀	2	0	0	1	4	2	1	2	3	3	2	1	2	2	1	1
P ₁	3	1	2	1	5	2	5	2	2	1	3	1	1	2	1	1
P ₂	2	1	0	3	2	3	1	6	0	2	1	3	1	2	1	3
P ₃	1	3	1	2	1	4	2	4	0	1	1	2	1	2	1	2
P ₄	1	4	3	2	3	6	6	5	2	2	3	3	1	2	2	3
					9	9	6	9								

Resource Request algo:-

Step 1:- if Request_i ≤ Need_i; then go to step 2
otherwise error

Step 2:- if Request_i ≤ Available then go to step 3
otherwise P_i will wait

Step 3:- System pretend as if request has been granted
by modifying the state as follows:-

$\Delta \text{Available} = \text{Request}_i$
 $\Delta \text{Allocation}_i = \text{Request}_i$
 $\Delta \text{Needs}_i = \text{Request}_i$

① Need Matrix? Yes
② Is system in Safe state? If Yes find safe sequence
③ If request from P_i arrives for
Yes (1,1,0,0) can request be immediately granted?
No (0,0,2,0) can't be immediately granted?

If modified resource-allocation state is safe then request granted
Otherwise P_i will wait & old allocation state is restored

 Jenny's Lectures Computer Science / Information Technology

Deadlock Detection & Recovery

- Allow the system to enter into deadlocked state
- Deadlock detection algorithms (2 types)
 - Recovery techniques
 - Single instance (Wait-for graph)
(Detect cycle)
 - Multiple instances (Banker's algo)

Subscribe JENNY'S LECTURES CS/IT NET&JRF for full syllabus

 Jenny's Lectures Computer Science / Information Technology

Deadlock Recovery

- ① Optimistic approach
(Pre-emption of Resources & Processes)
 - ↳ Pre-empt some resources from process & give these resources to other processes until the deadlock cycle is broken
- ② Pessimistic Approach
(Process Termination)
 - ↳ Abort all deadlocked processes
 - ↳ Abort one process at a time and decide next to abort after deadlock detection
 - ↳ overhead of calling detection algo again & again

Selecting a victim

Rollback

Starvation

→ Priority of process

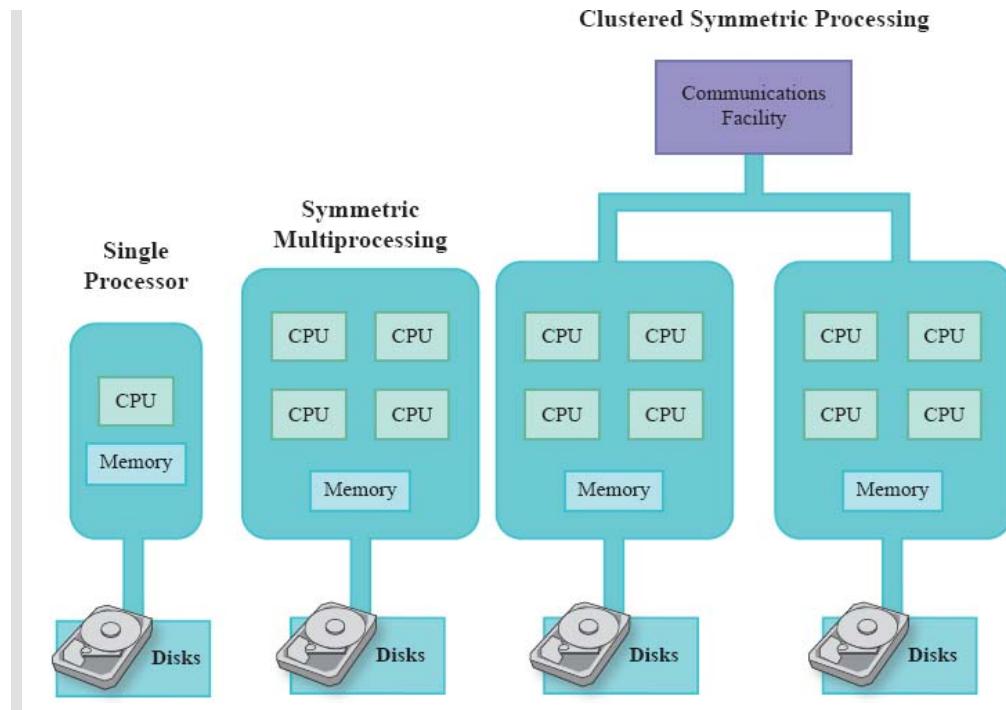
→ How long the process has computed?

→ how much longer a process will compute before completion

→ How many & what type of resources process has used

→ How many resources the process needs to complete its execution etc.

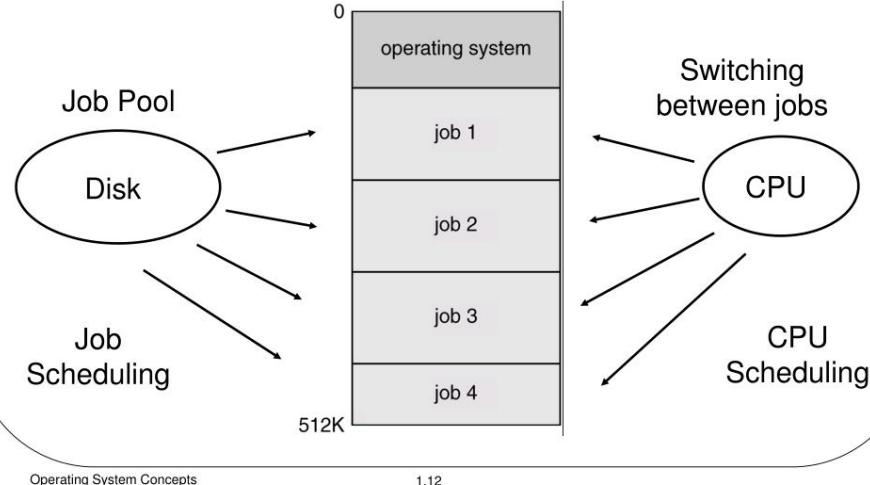
- SP vs MP vs Cluster

Clustered Symmetric Processing

- Multiprogramming

Multiprogramming Batch Systems

Multiprogramming: several jobs are kept in main memory at the same time, and the CPU is multiplexed among them which requires memory management and protection.



Operating System Concepts

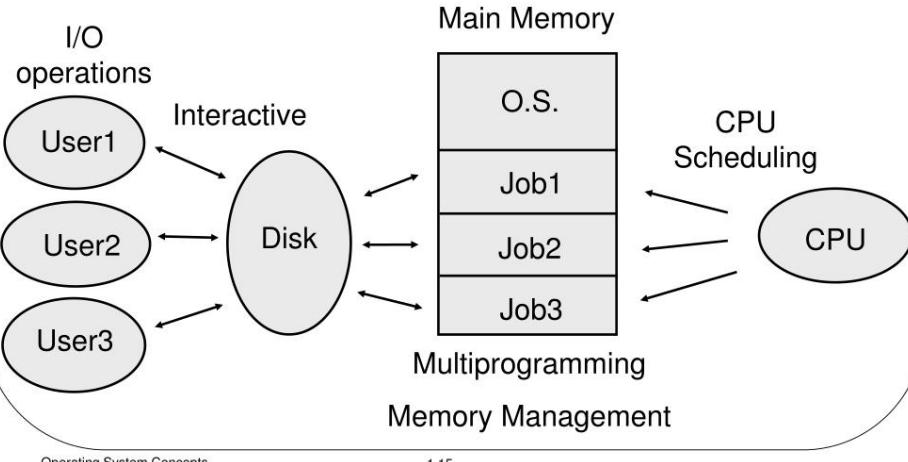
1.12

MULTIPROGRAMMING VERSUS MULTITASKING

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

Time-Sharing Systems – Interactive Computing

A time-sharing system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.

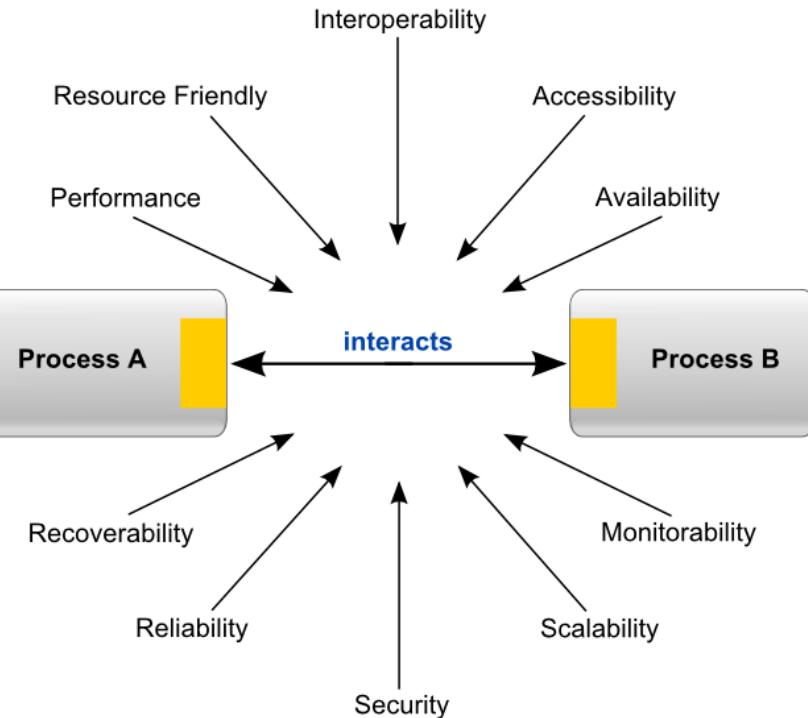


Operating System Concepts

1.15

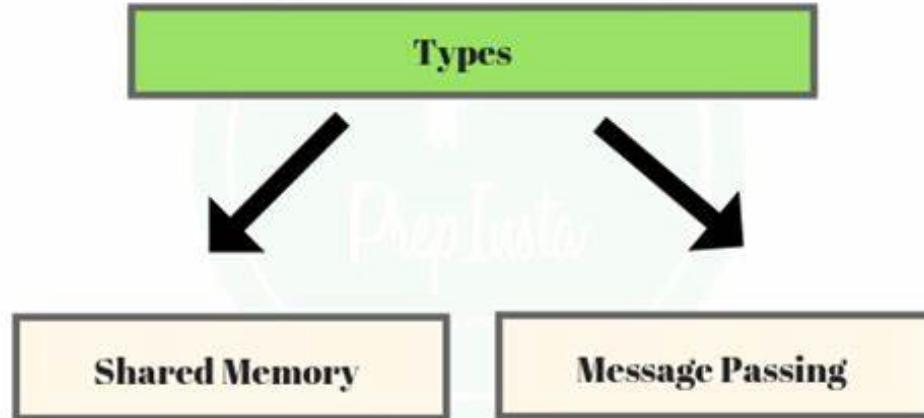
Interprocess Communication(IPC)

Interprocess Communication



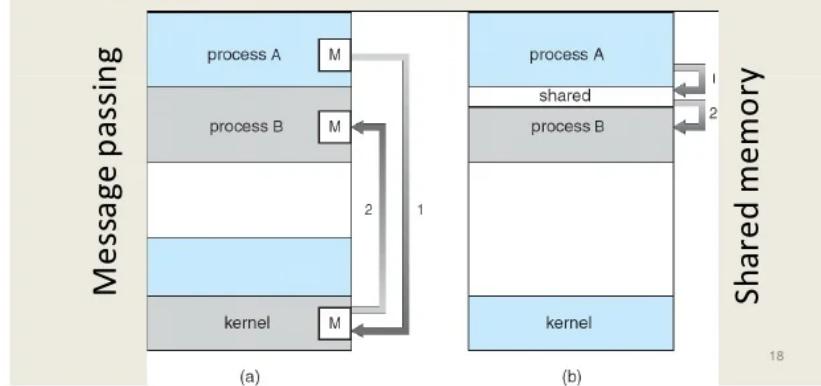


Inter-Process Communication Types



4. Interprocess Communication Contd...

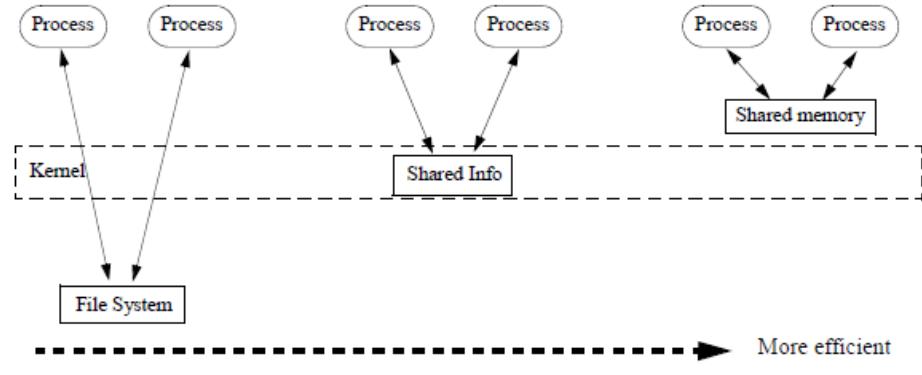
- Two fundamental models of Interprocess communication
- **Shared memory**
 - a region of memory that is shared by cooperating processes is established then exchange information takes place by reading and writing data to the shared region
- **Message passing**
 - communication takes place by means of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



18

IPC Mechanisms

IPC classified by implementation mechanisms.

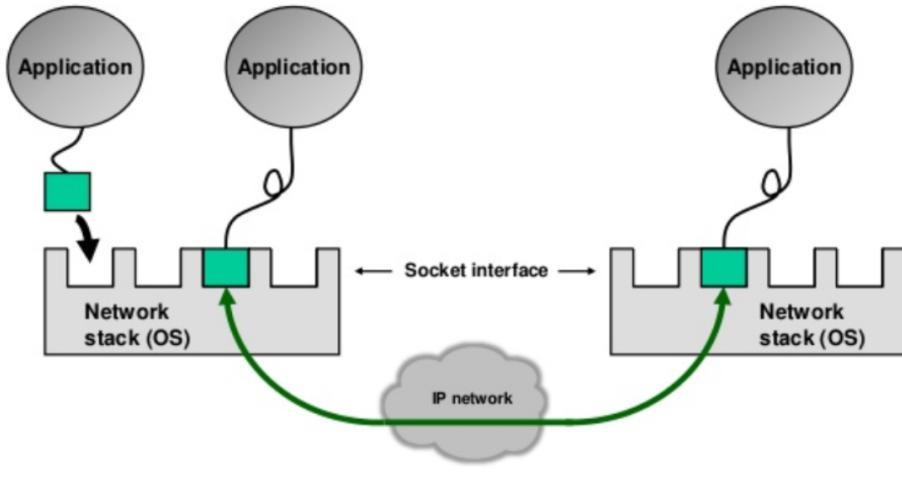


Examples:

files

semaphore, socket,
pipe, message queue,
signal

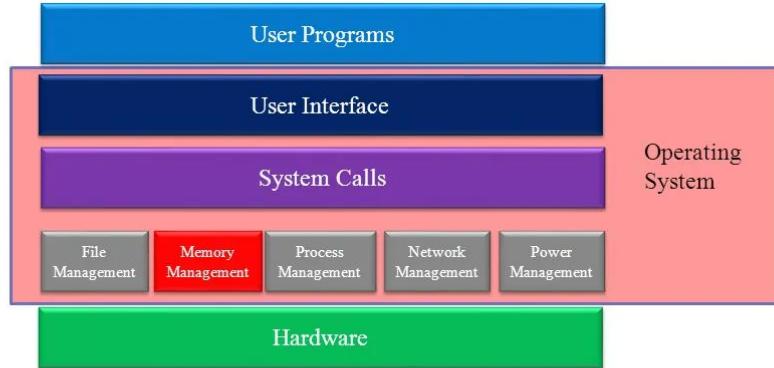
shared memory



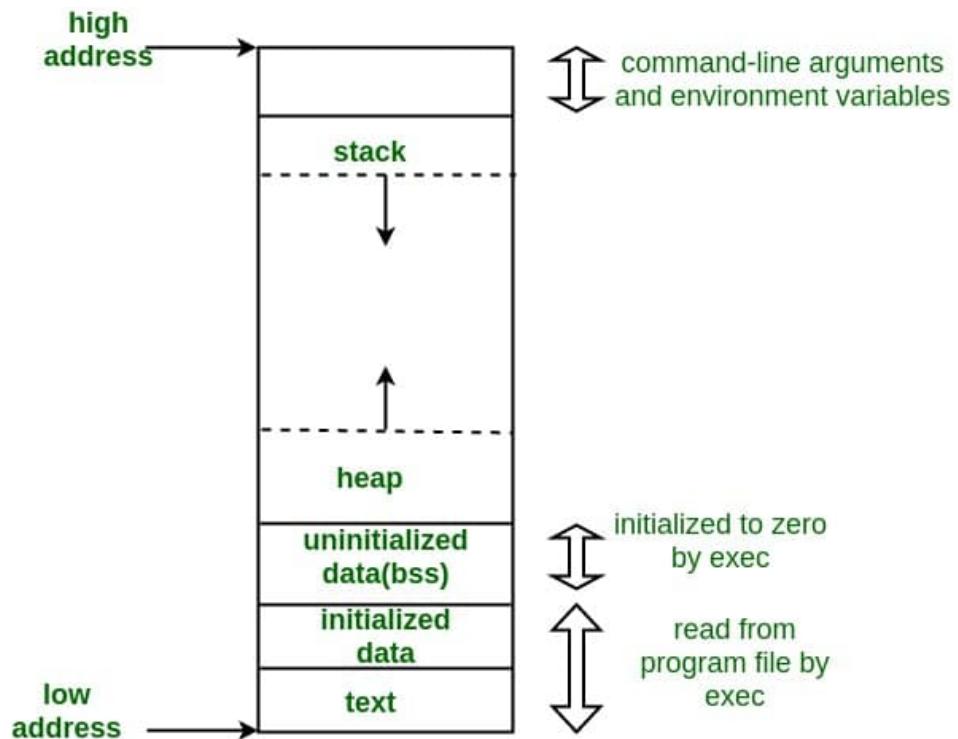
Memory Management

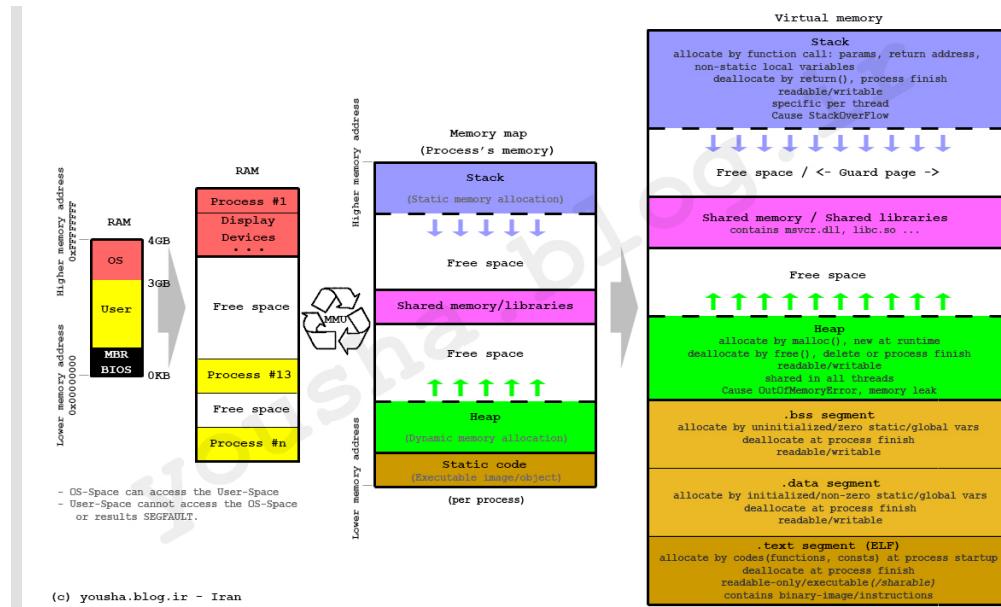


Memory Management in OS

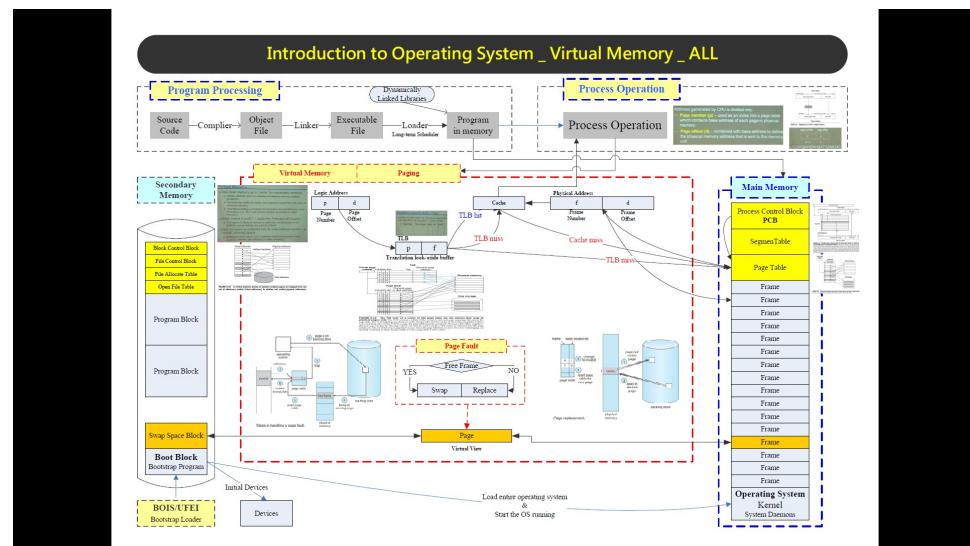


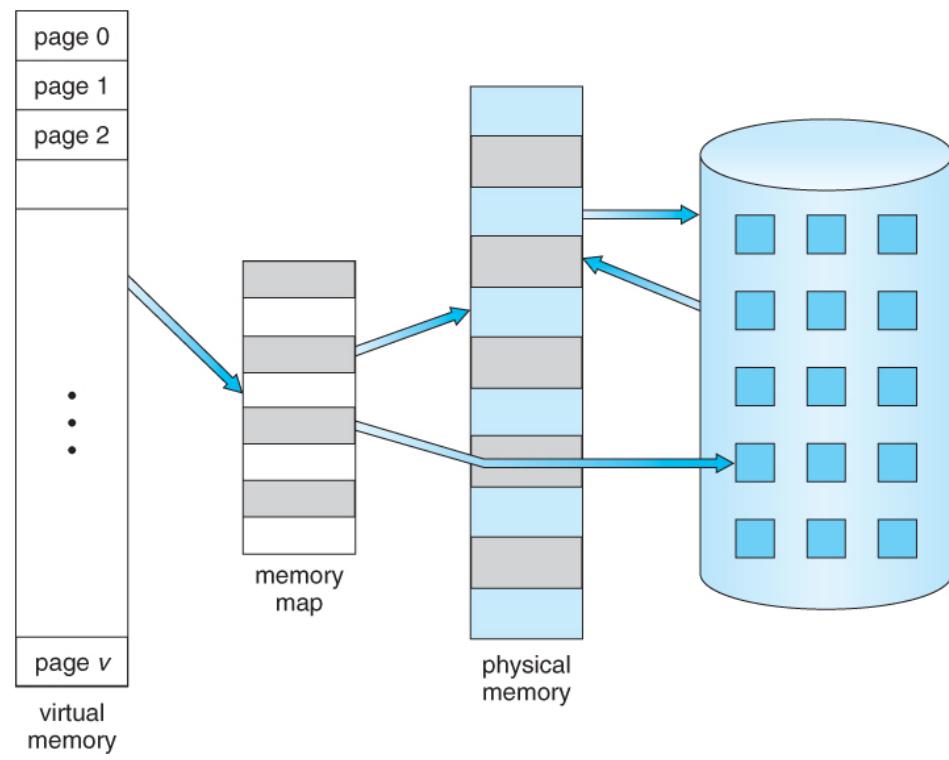
5



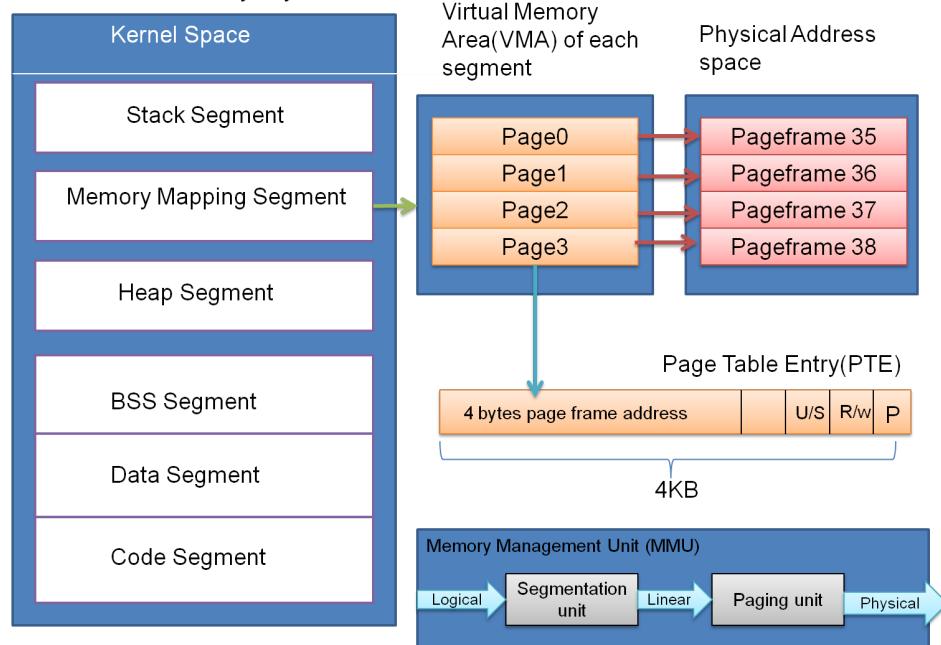


Virtual Memory





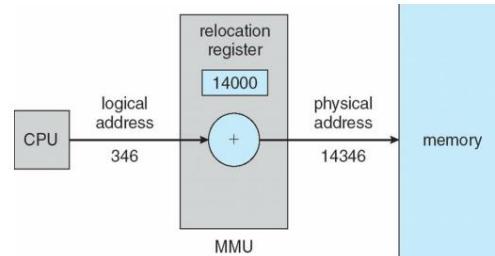
Process Virtual Memory Layout





Memory-Management Unit (MMU)

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



Dynamic relocation using a relocation register



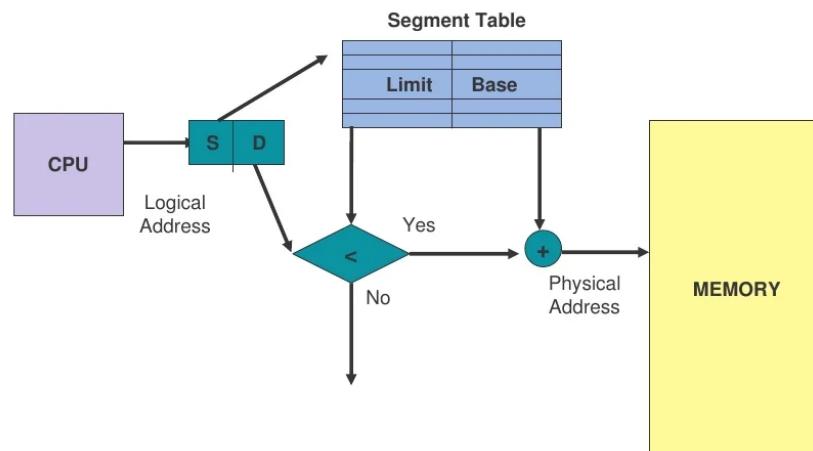
Operating System Concepts with Java – 8th Edition

8.8

Silberschatz, Galvin and Gagne ©2009

MEMORY MANAGEMENT Segmentation

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



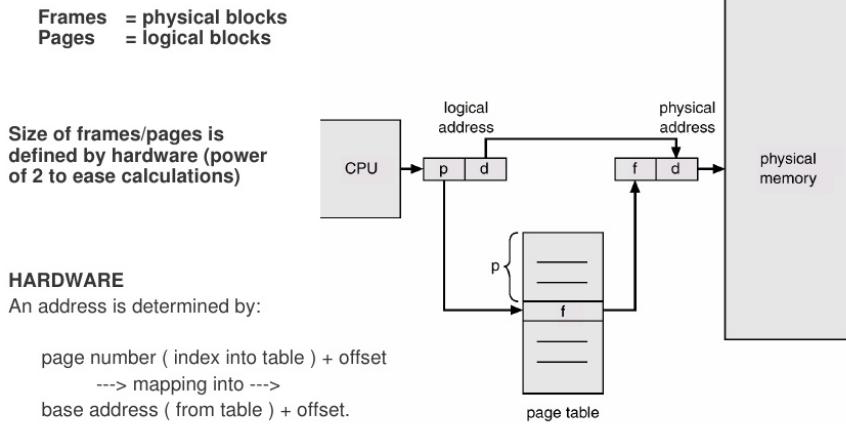
8: Memory Management

31

MEMORY MANAGEMENT

PAGING

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



8: Memory Management

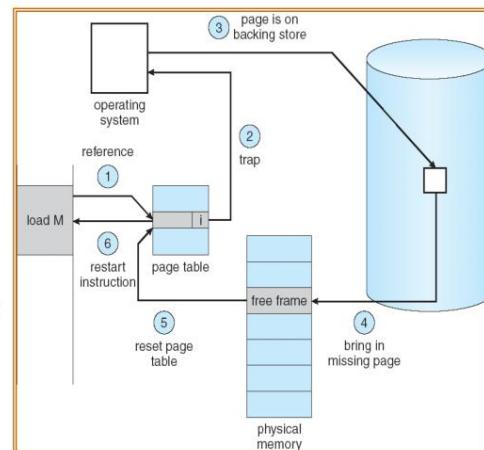
22

Page Fault

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

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

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



Page Fault Handling – a different perspective

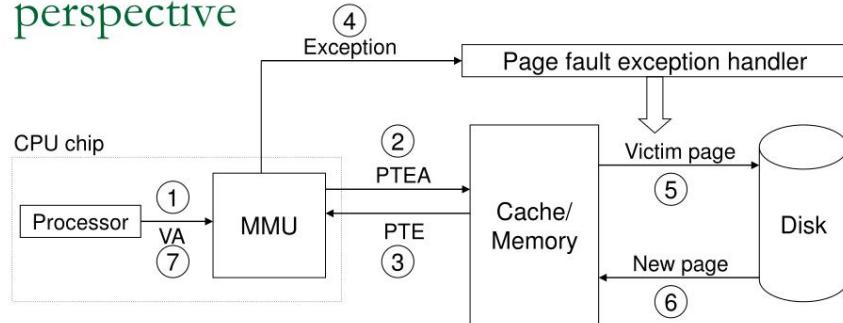


Fig. 10.14 (Bryant)



Page fault handling

A page fault handling sequence:

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

Page Fault Handling (1)

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

32

Page Fault Handling (2)

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

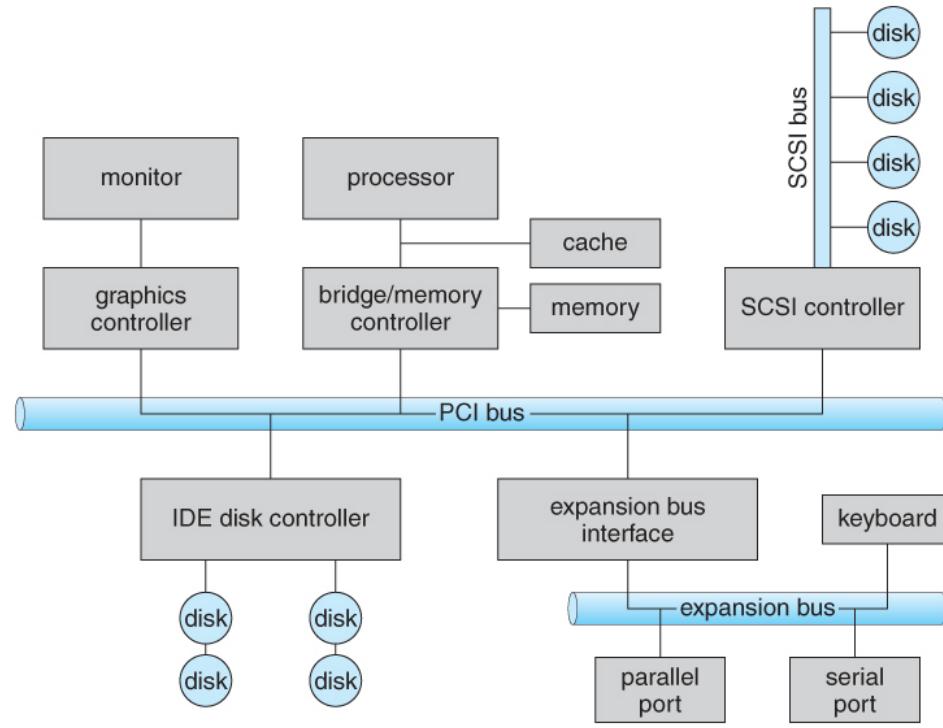
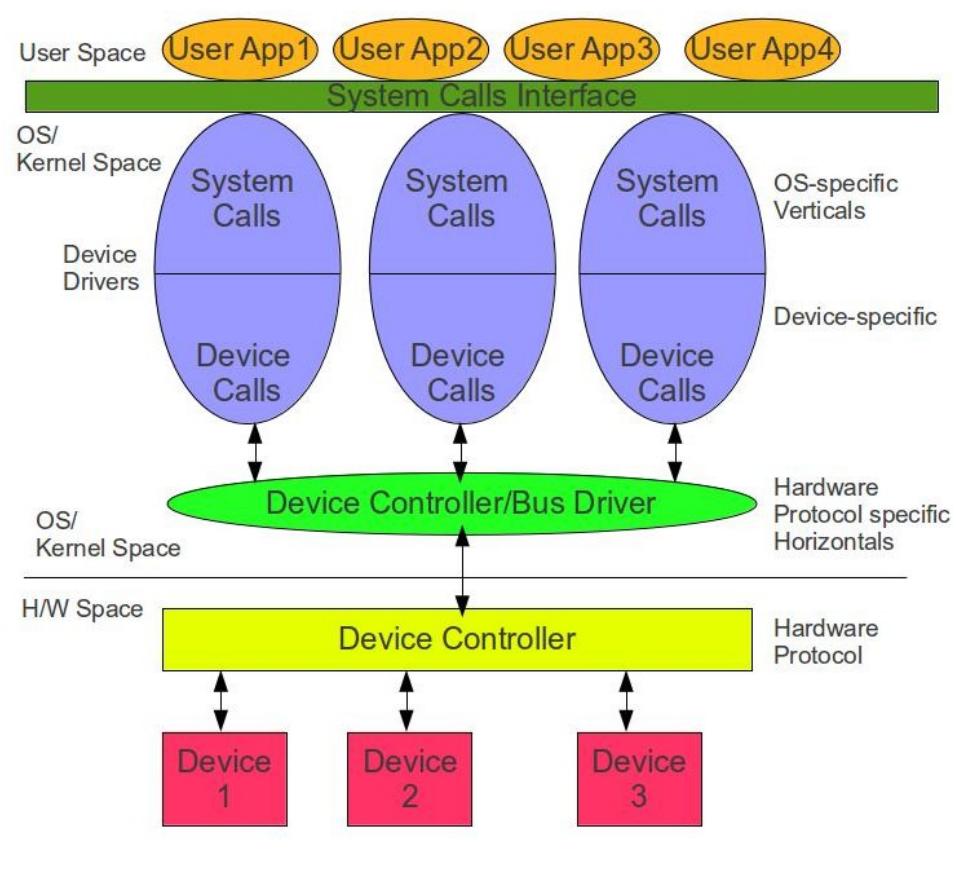
Faulting instruction backed up to when it began

6. Faulting process scheduled
7. Registers restored

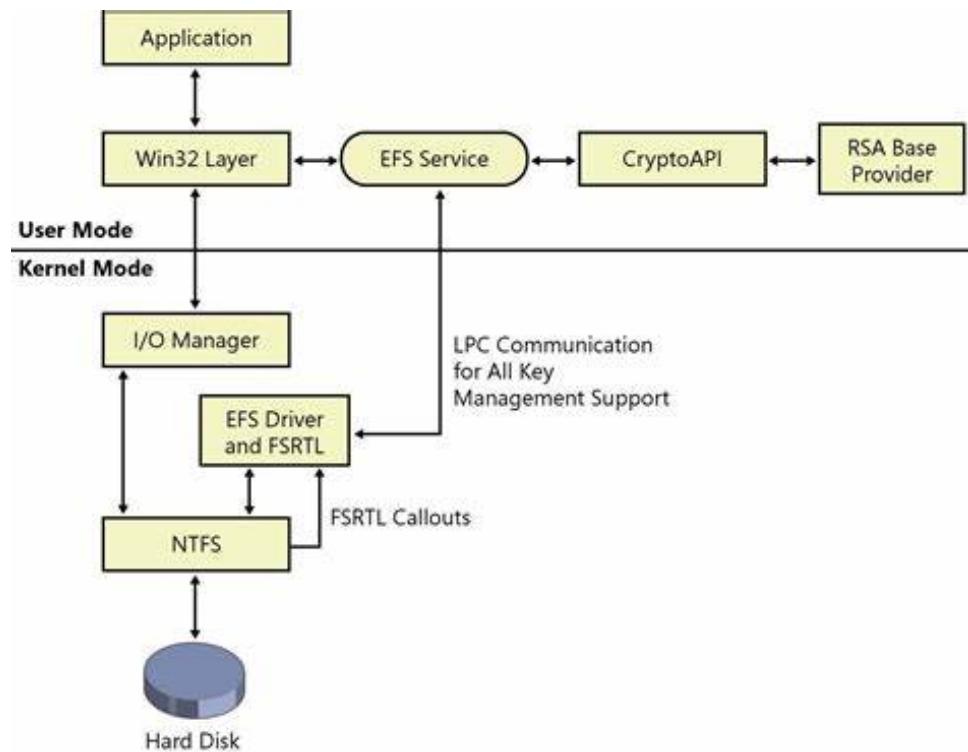
Program continues

47

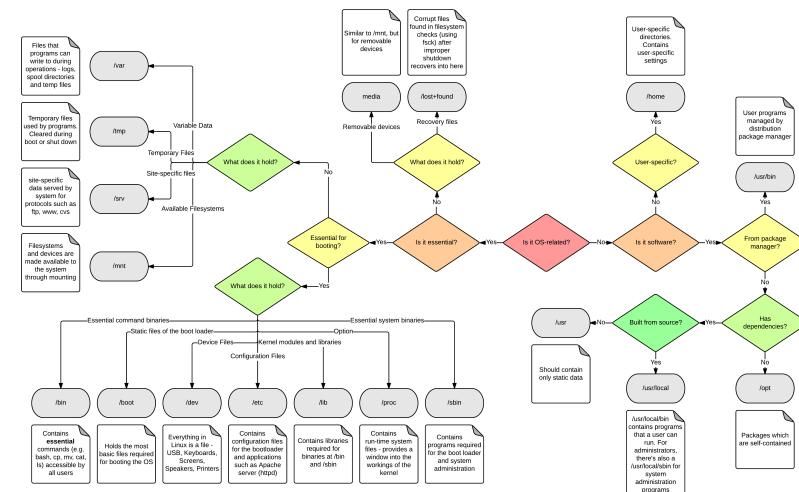
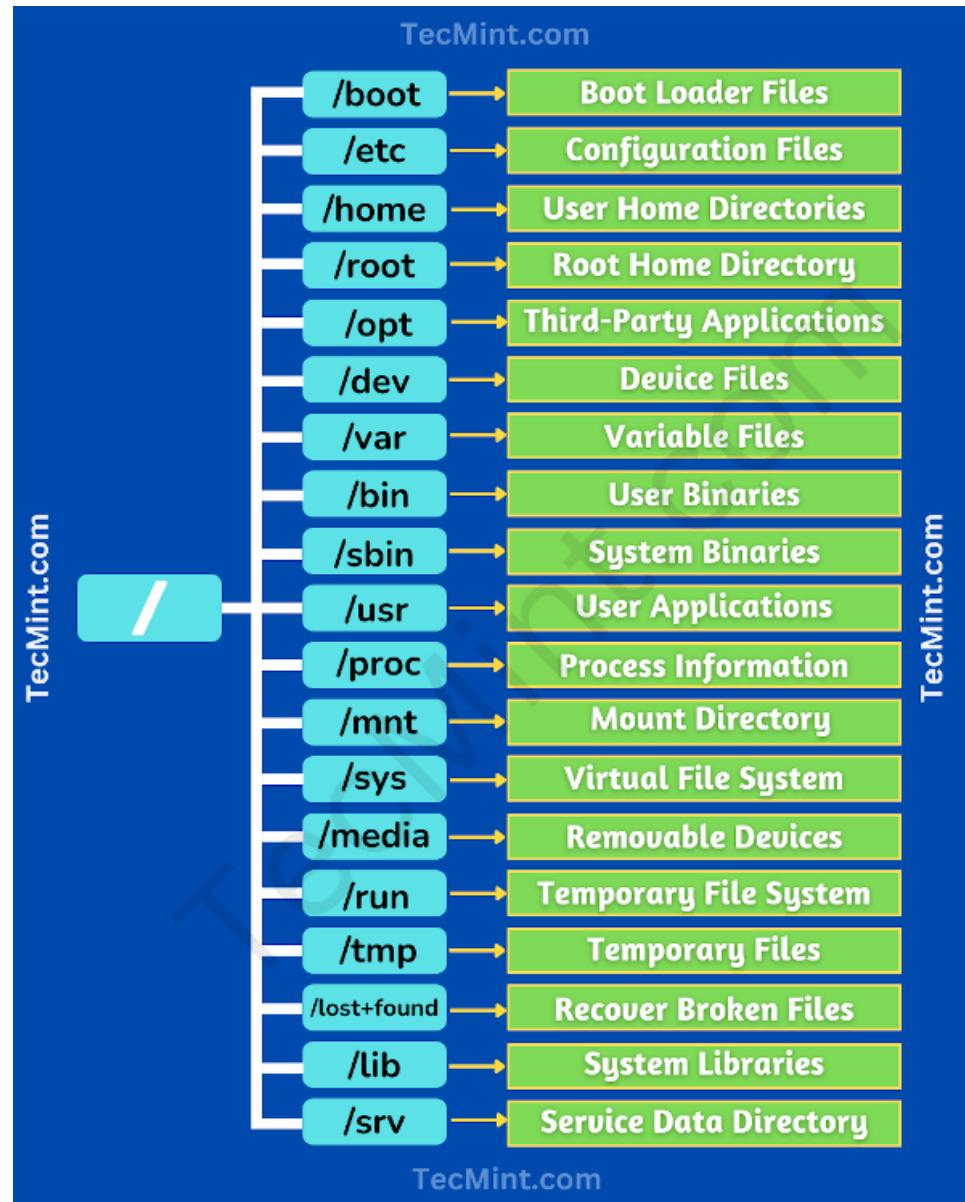
Device Management

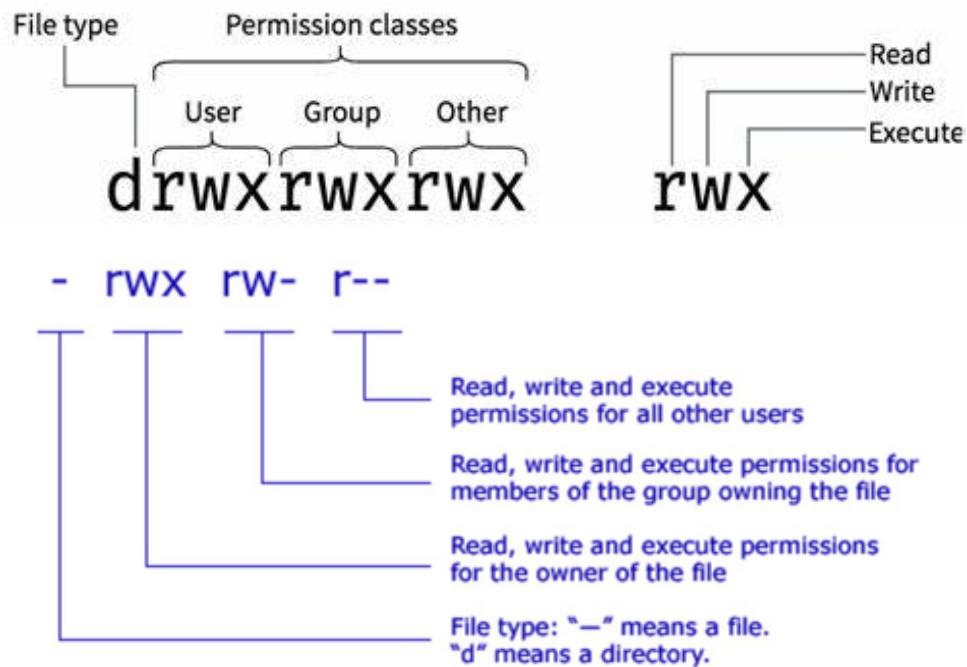
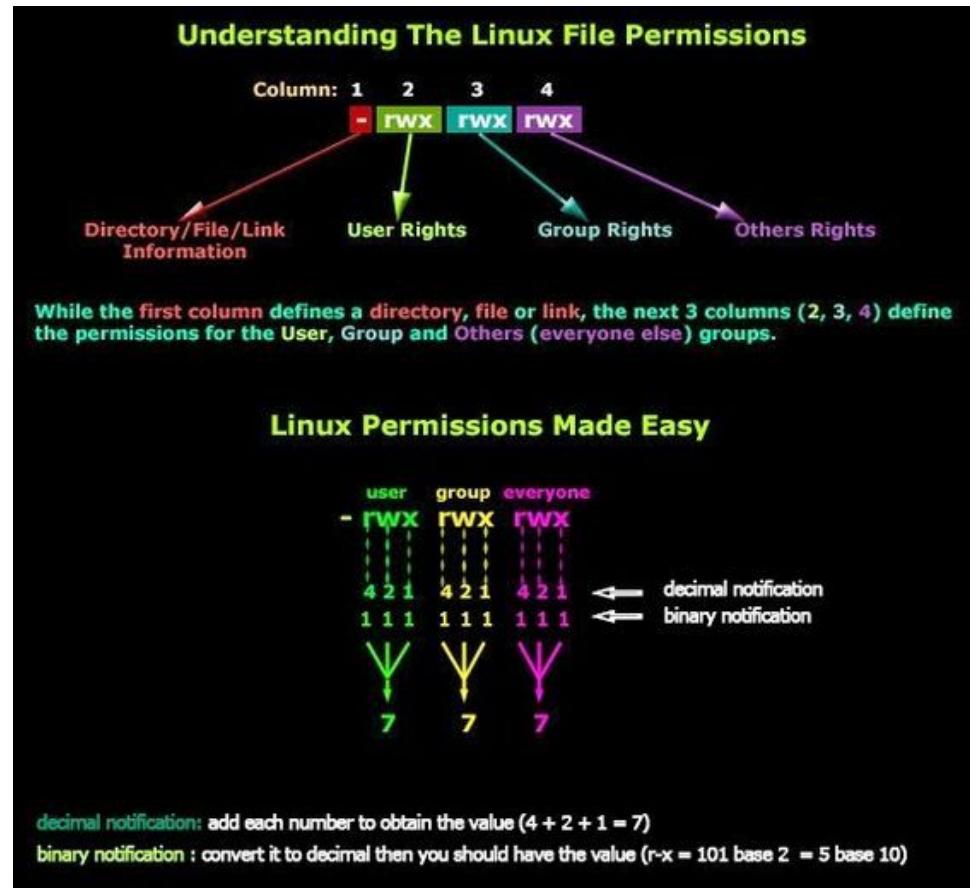


File Management



File Structure in Linux





File Attributes

File Attributes

A file's attributes vary from one operating system to another but typically consist of these:

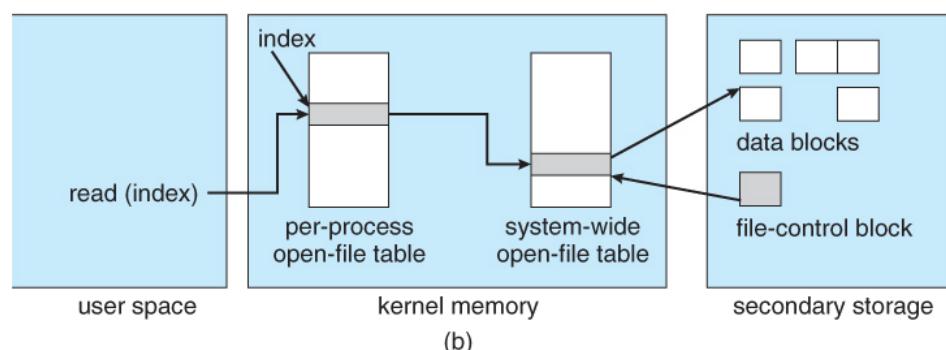
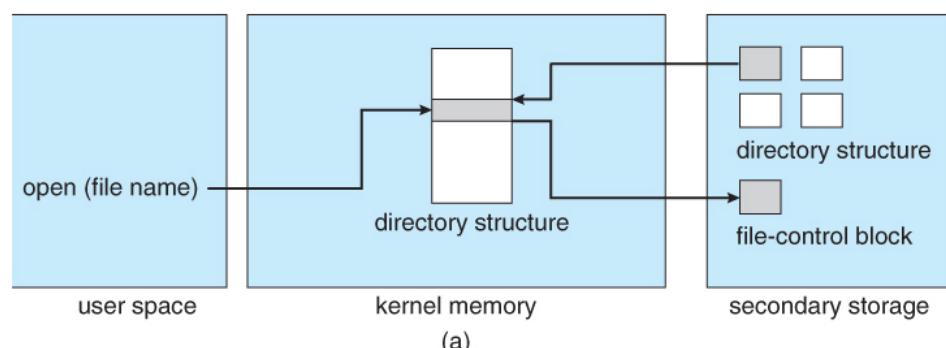
- ▶ **Name** – only information kept in human-readable form
- ▶ **Identifier** – unique tag (number) identifies file within file system
- ▶ **Type** – needed for systems that support different types
- ▶ **Location** – pointer to file location on device
- ▶ **Size** – current file size
- ▶ **Protection** – controls who can do reading, writing, executing
- ▶ **Time, date, and user identification** – data for protection, security, and usage monitoring
- ▶ Information about files are kept in the **directory structure**, which is maintained on the disk. Typically, a directory entry consists of the file's name and its unique identifier.

SHASHI KS

File Type

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

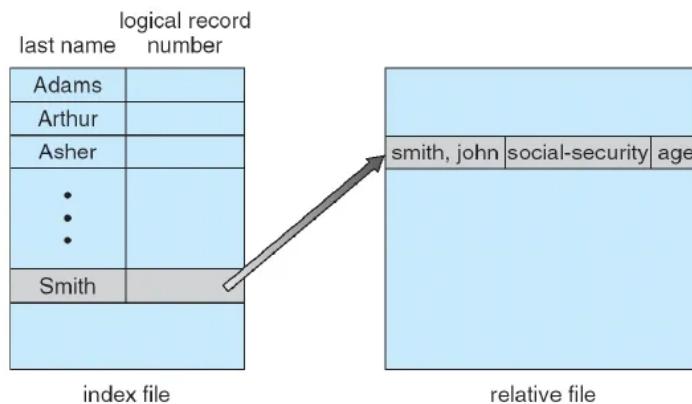
File Access

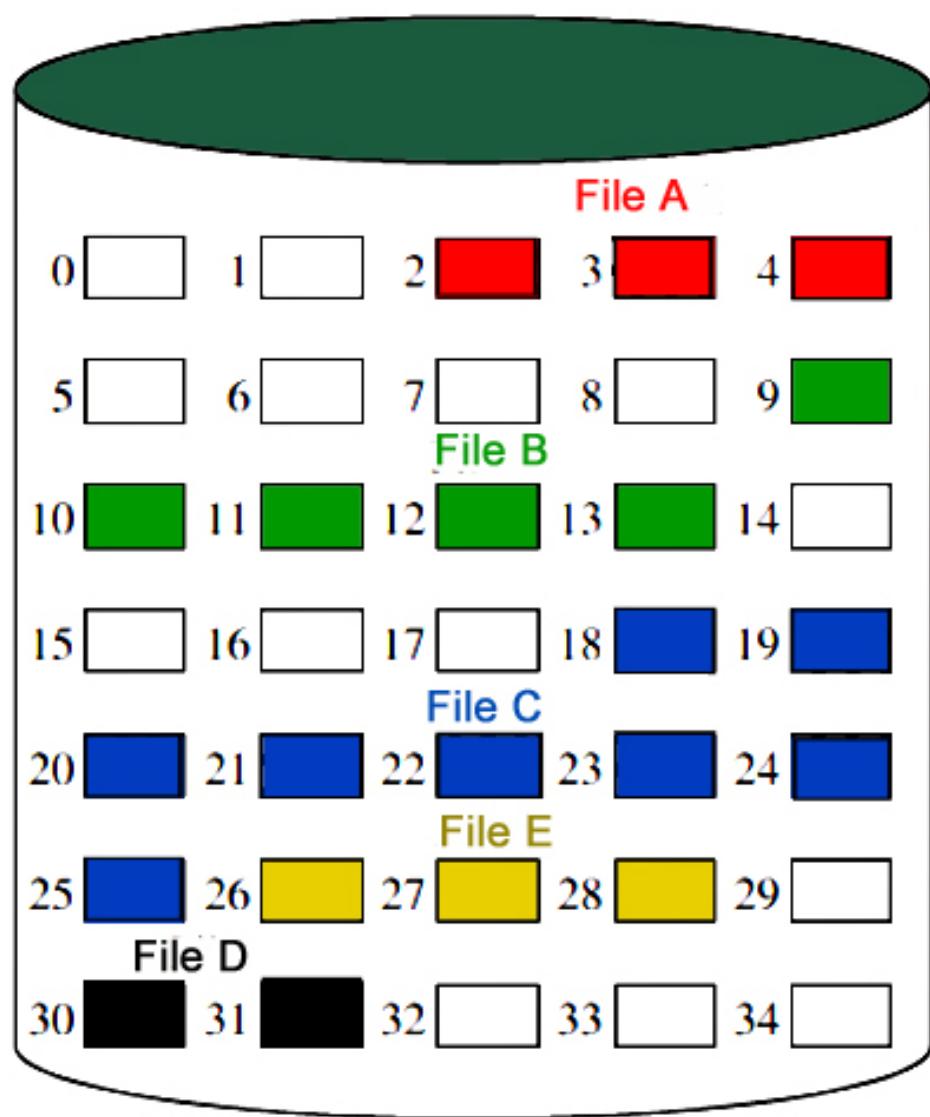


2. Access Methods Contd...

2.3 Other Access Methods

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





File allocation table

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

Disk Scheduling Algorithm



Selecting a Disk-Scheduling Algorithm

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

SunMoon University

19

Selecting a Disk-Scheduling Algorithm

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

16

Disk Scheduling Algorithms

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

Example

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

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

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



Disk Scheduling Algorithms

Table 11.2 Comparison of Disk Scheduling Algorithms

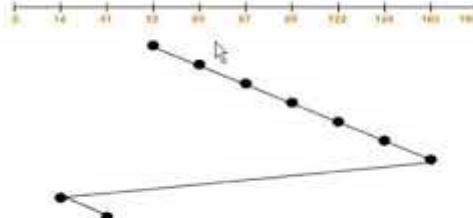
(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number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

38

Problem - C-LOOK

Disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The head is initially at cylinder number 53 and moving towards the end. The cylinders are numbered from 0 to 199. The total head movement in number of cylinders incurred while servicing these requests is _____

Solution-



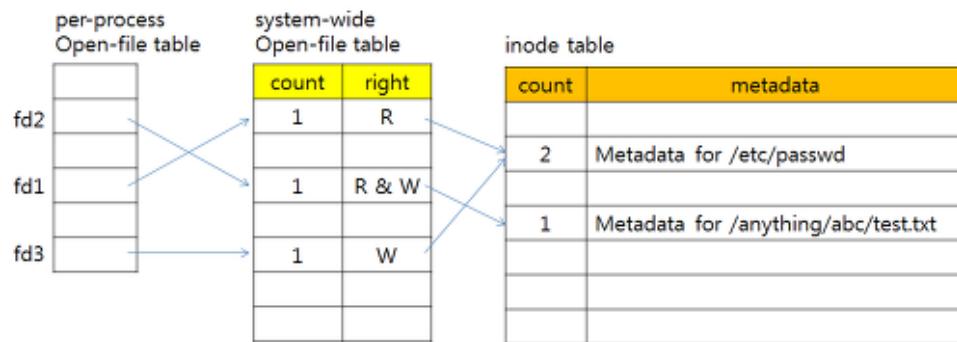
Order of head movements = 53 → 65 → 67 → 98 → 122 → 124 → 183 → 41 → 14

Total head movements incurred while servicing these requests

$$\begin{aligned}
 &= |53 - 65| + |65 - 67| + |67 - 98| + |98 - 122| + |122 - 124| + |124 - 183| + |183 - 14| \\
 &+ |14 - 41| = 12 + 2 + 31 + 24 + 2 + 59 + 169 + 27 = 326
 \end{aligned}$$

$$\text{Verification: } |183 - 53| + |183 - 14| + |41 - 14| = 130 + 169 + 27 = 326$$

File Operations



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

File operation	Declaration & Description
fopen() - To open a file	<p>Declaration: FILE *fopen (const char *filename, const char *mode)</p> <p>fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist.</p> <pre data-bbox="817 316 1091 370">FILE *fp; fp=fopen ("filename", "mode");</pre> <p>Where,</p> <p>fp - file pointer to the data type "FILE".</p> <p>filename - the actual file name with full path of the file.</p> <p>mode - refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.</p>
fclose() - To close a file	<p>Declaration: int fclose(FILE *fp);</p> <p>fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.</p> <pre data-bbox="906 574 1006 601">fclose (fp);</pre>
fgets() - To read a file	<p>Declaration: char *fgets(char *string, int n, FILE *fp)</p> <p>fgets function is used to read a file line by line. In a C program, we use fgets function as below.</p> <pre data-bbox="858 698 1049 752">fgets (buffer, size, fp);</pre> <p>where,</p> <p>buffer - buffer to put the data in.</p> <p>size - size of the buffer</p> <p>fp - file pointer</p>
fprintf() - To write into a file	<p>Declaration:</p> <pre data-bbox="596 844 1309 929">int fprintf(FILE *fp, const char *format, ...);</pre> <p>fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, "some data"); or</p> <pre data-bbox="790 900 1112 927">fprintf (fp, "text %d", variable_name);</pre>

File operation	Declaration & Description
fopen() - To open a file	<p>Declaration: FILE *fopen (const char *filename, const char *mode)</p> <p>fopen() function is used to open a file to perform operations such as reading, writing etc. In a C program, we declare a file pointer and use fopen() as below. fopen() function creates a new file if the mentioned file name does not exist.</p> <pre data-bbox="817 1199 1091 1253">FILE *fp; fp=fopen ("filename", "mode");</pre> <p>Where,</p> <p>fp - file pointer to the data type "FILE".</p> <p>filename - the actual file name with full path of the file.</p> <p>mode - refers to the operation that will be performed on the file. Example: r, w, a, r+, w+ and a+. Please refer below the description for these mode of operations.</p>
fclose() - To close a file	<p>Declaration: int fclose(FILE *fp);</p> <p>fclose() function closes the file that is being pointed by file pointer fp. In a C program, we close a file as below.</p> <pre data-bbox="898 1473 1006 1500">fclose (fp);</pre>
fgets() - To read a file	<p>Declaration: char *fgets(char *string, int n, FILE *fp)</p> <p>fgets function is used to read a file line by line. In a C program, we use fgets function as below.</p> <pre data-bbox="858 1605 1049 1632">fgets (buffer, size, fp);</pre> <p>where,</p> <p>buffer - buffer to put the data in.</p> <p>size - size of the buffer</p> <p>fp - file pointer</p>
fprintf() - To write into a file	<p>Declaration:</p> <pre data-bbox="596 1736 1309 1821">int fprintf(FILE *fp, const char *format, ...);</pre> <p>fprintf() function writes string into a file pointed by fp. In a C program, we write string into a file as below. fprintf (fp, "some data"); or</p> <pre data-bbox="790 1799 1112 1825">fprintf (fp, "text %d", variable_name);</pre>

OS Security

Precautions for OS Security

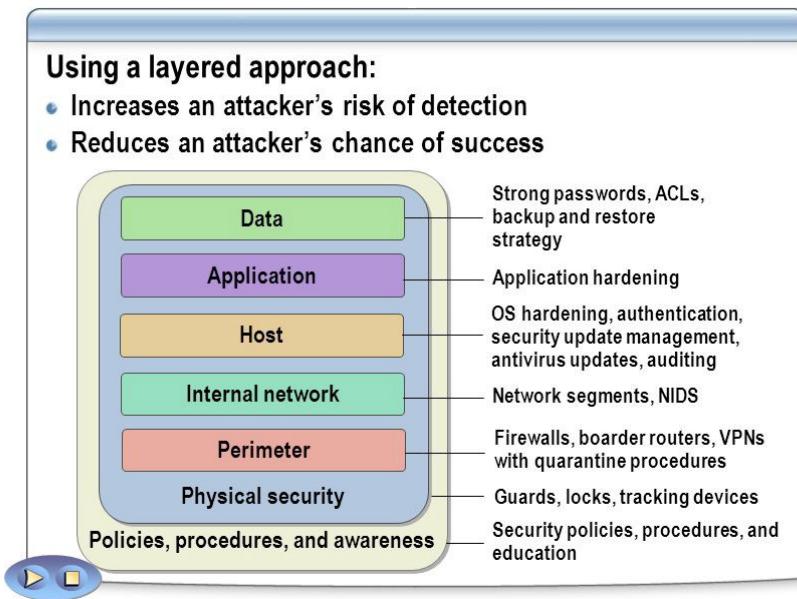
To Secure Our System and to avoid the Security breaches we must take some precautions in our OS.

So we have to ensure of the Shown Security Components:-

- Bios Security*
- User Accounts Security*
- Data Security*
- Antivirus Security*
- Firewall*



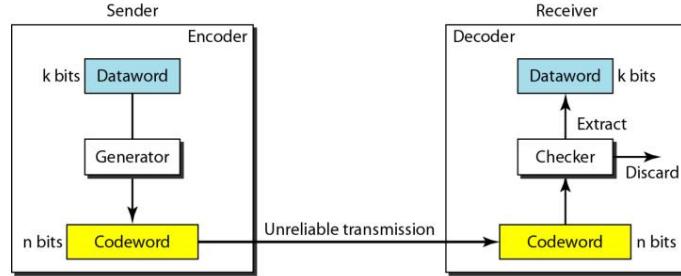
Understanding Defense-in-Depth



Error Detection and Correction

Error Detection

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



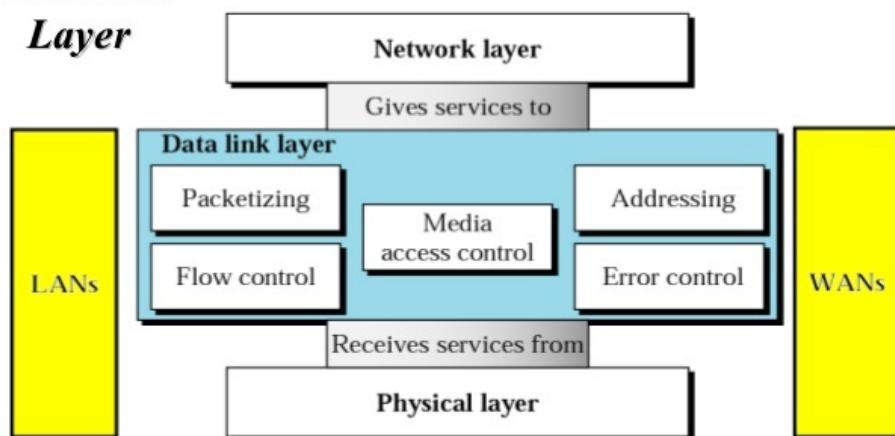
10.12

Er. M.S.Kuthar

Error Detection and Correction

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

Data Link Layer



BCS Error Detection & Correction

■ Error Detection

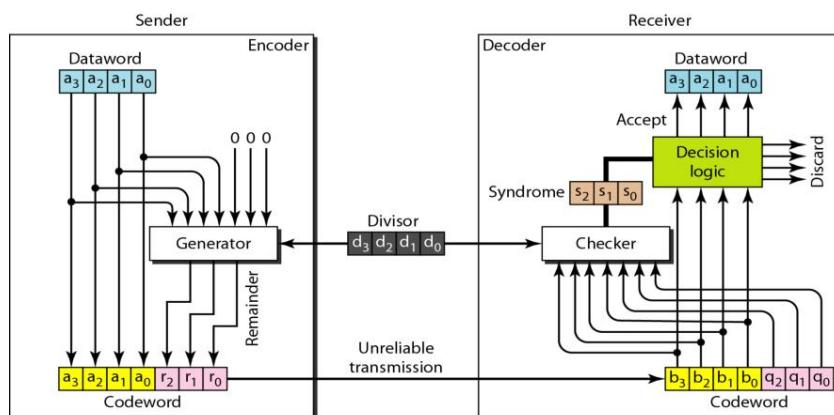
- Check if any error has occurred
- Don't care the number of errors
- Don't care the positions of errors

■ Error Correction

- Need to know the number of errors
- Need to know the positions of errors
- More complex
- The number of errors and the size of the message are important factors in error correction

BCS Error Detection & Correction

Figure 1.6 CRC encoder and decoder



1. Resource Allocation

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



S

2. Accounting

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

S

3. Protection and Security

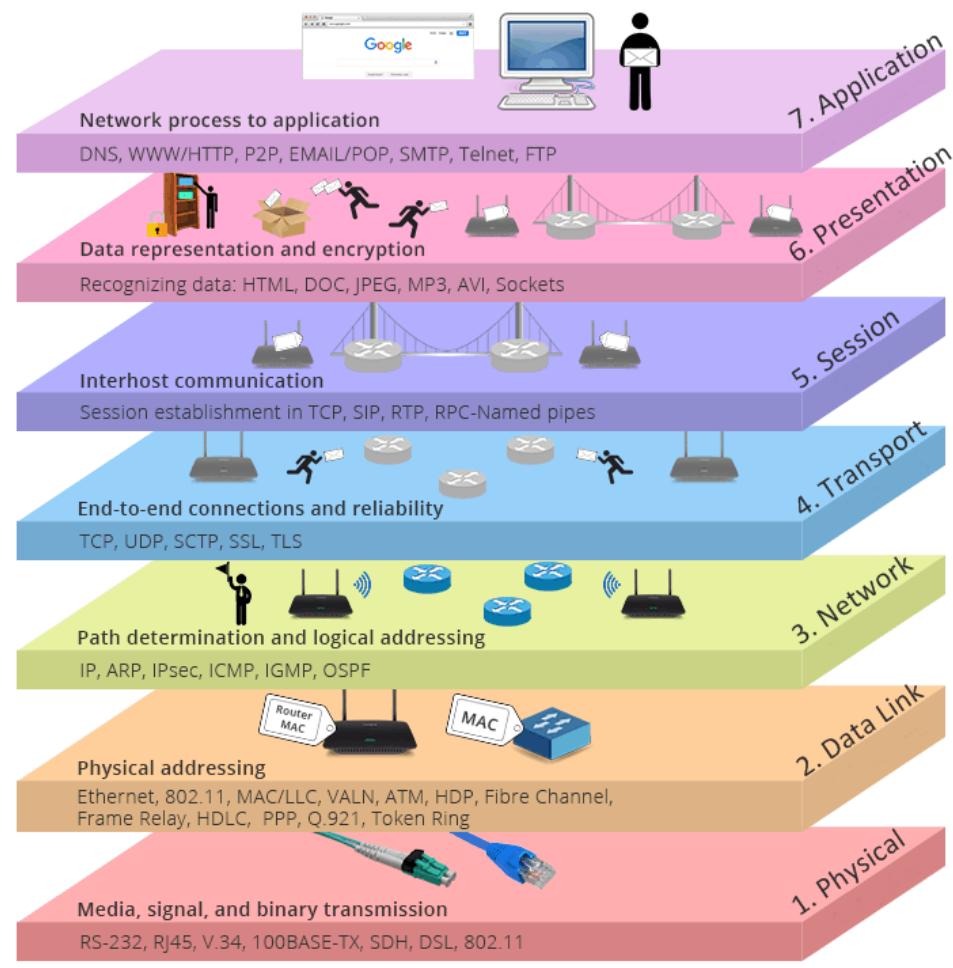
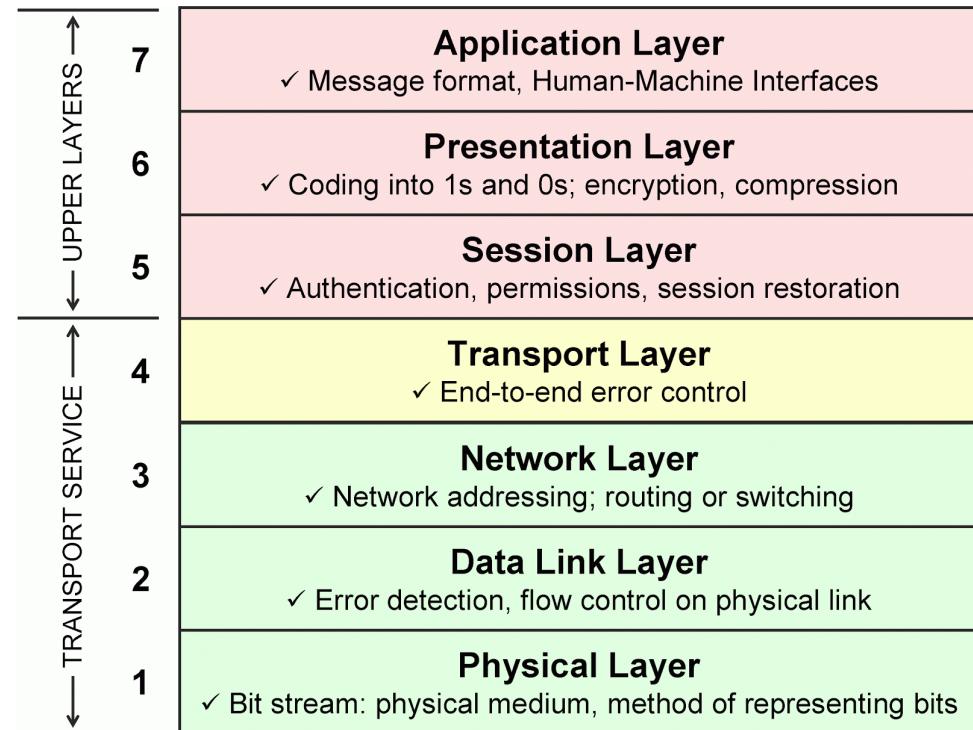
- Protection – access to system should be controlled
 - Multiuser systems should not allow an unauthorized user to access content
 - No interference between 2 processes
- Security – from external world
 - To access a system, there must be some kind of authorization (like a password)



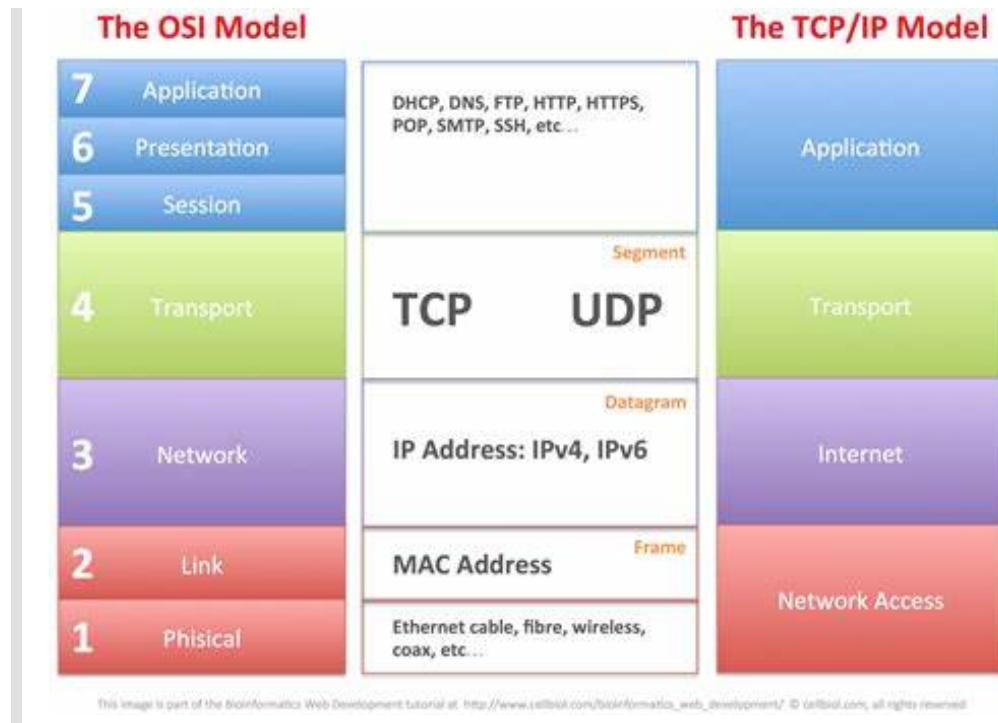
S

Network

Network OSI Model



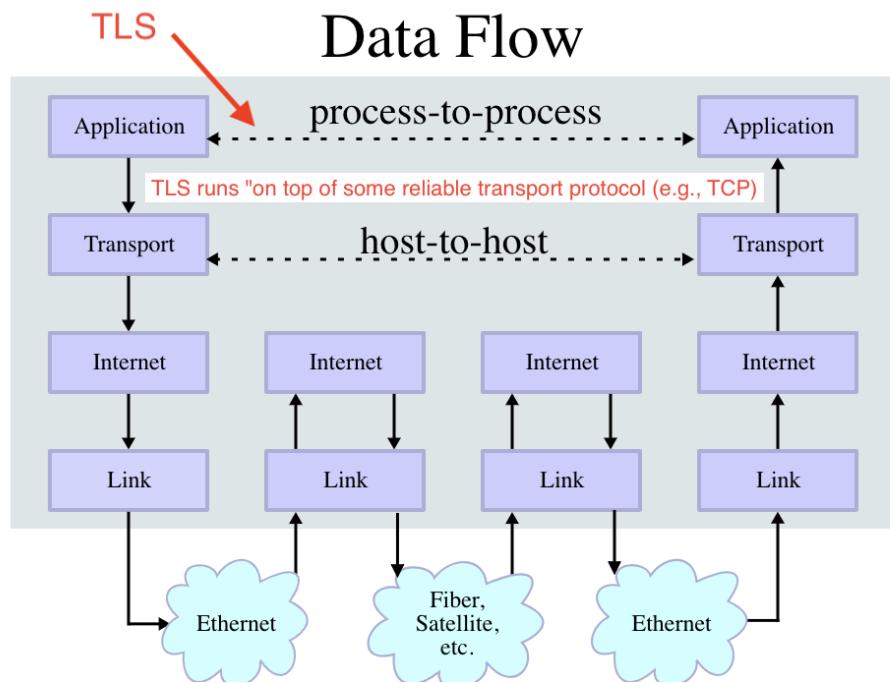
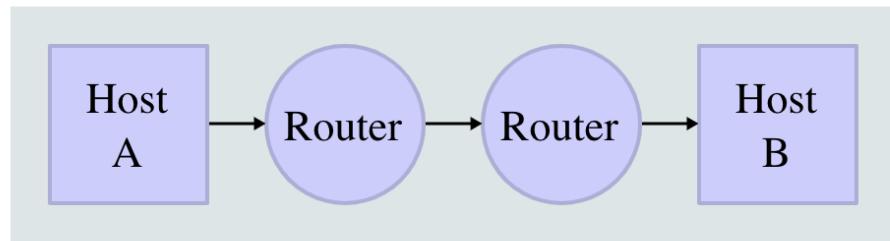
TCP/IP



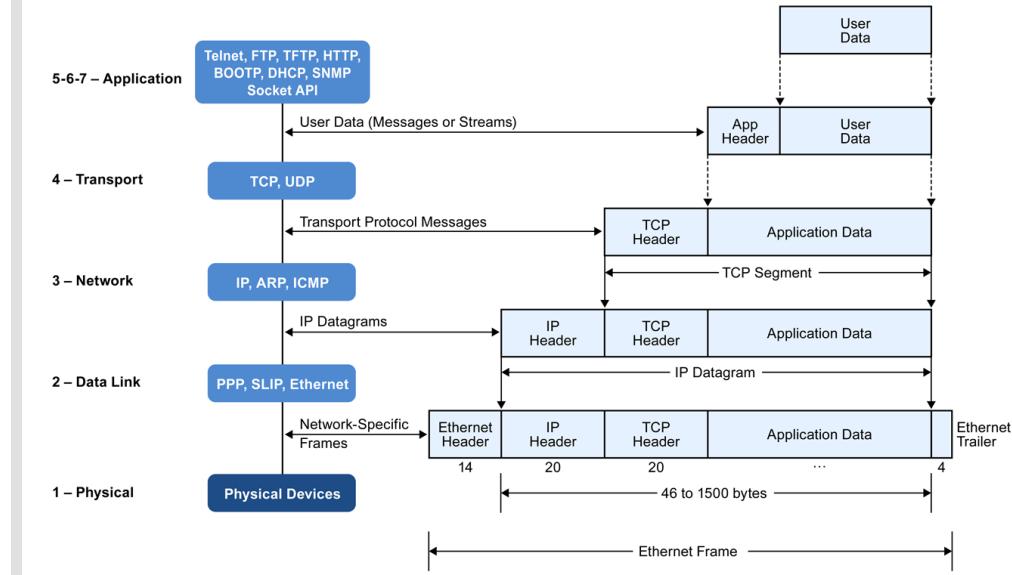
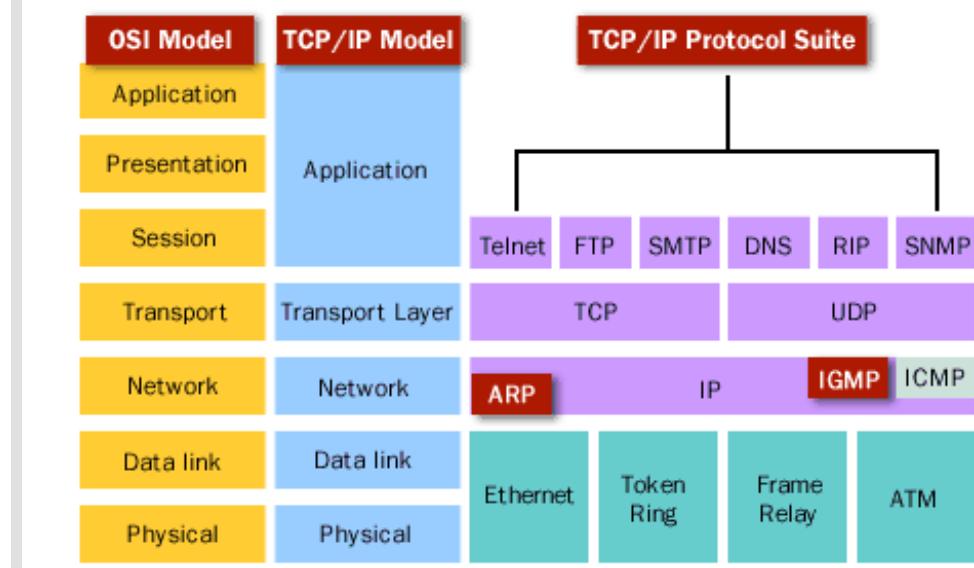
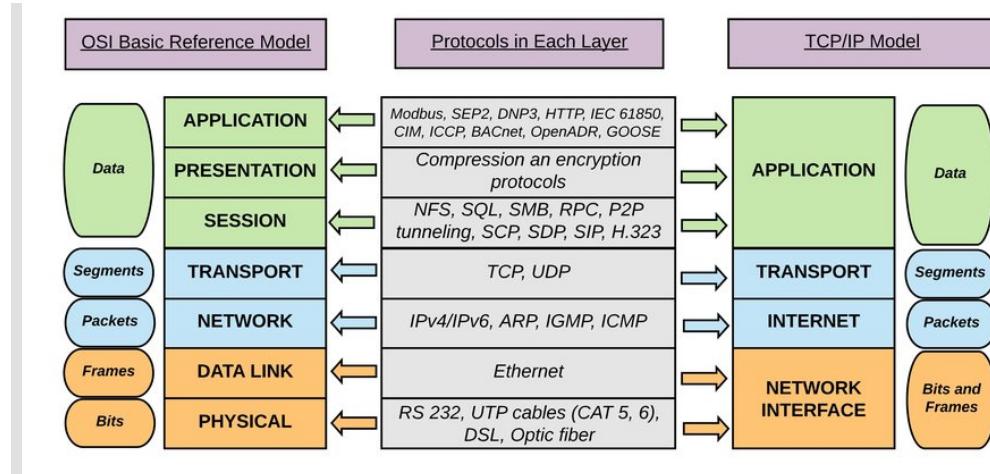
This image is part of the Bioinformatics Web Development tutorial at http://www.cellbiok.com/bioinformatics_web_development/ © cellbiok.com, all rights reserved.

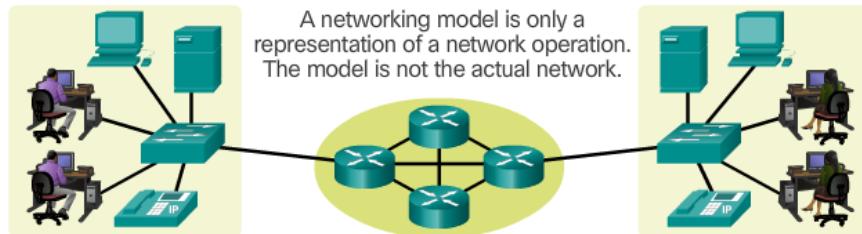
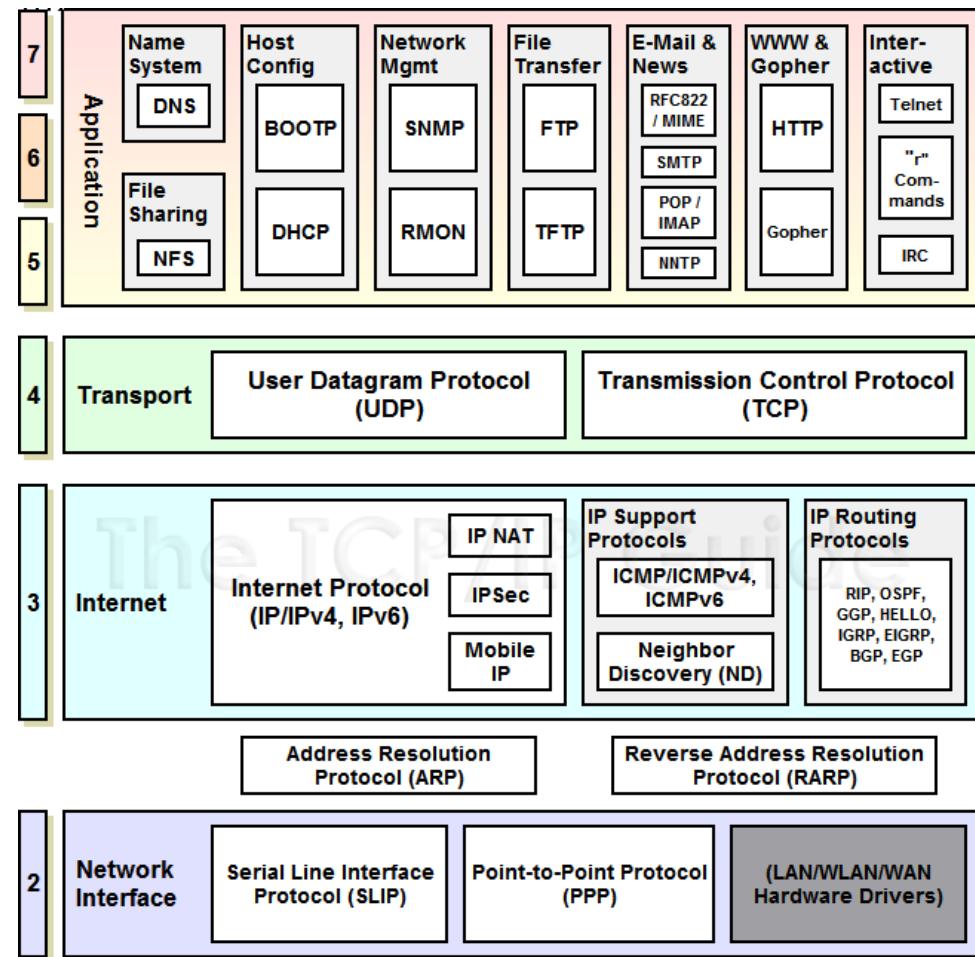
Data Flow

Network Topology



Protocol





OSI Model

TCP/IP Protocol Suite

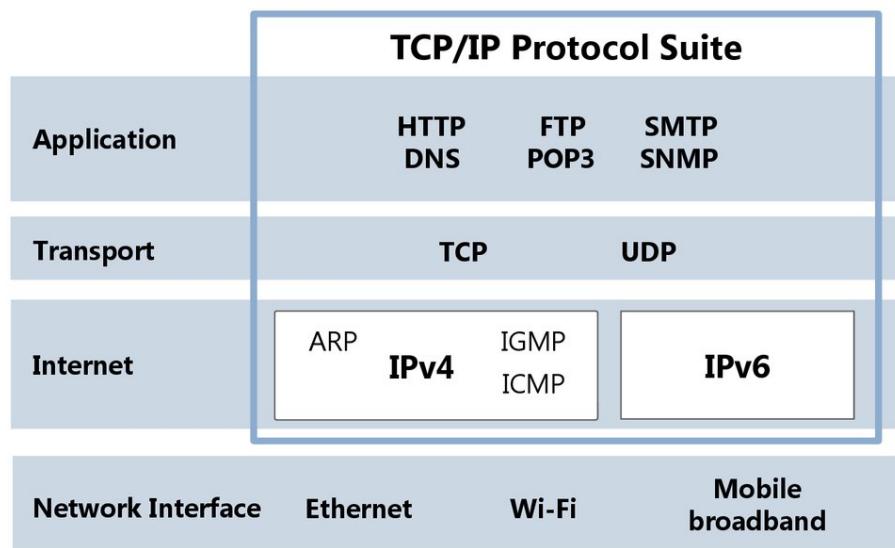
TCP/IP Model

Application		Application
Presentation	HTTP, DNS, DHCP, FTP	
Session		
Transport	TCP, UDP	Transport
Network	IPv4, IPv6, ICMPv4, ICMPv6	Internet
Data Link	PPP, Frame Relay, Ethernet	Network Access
Physical		

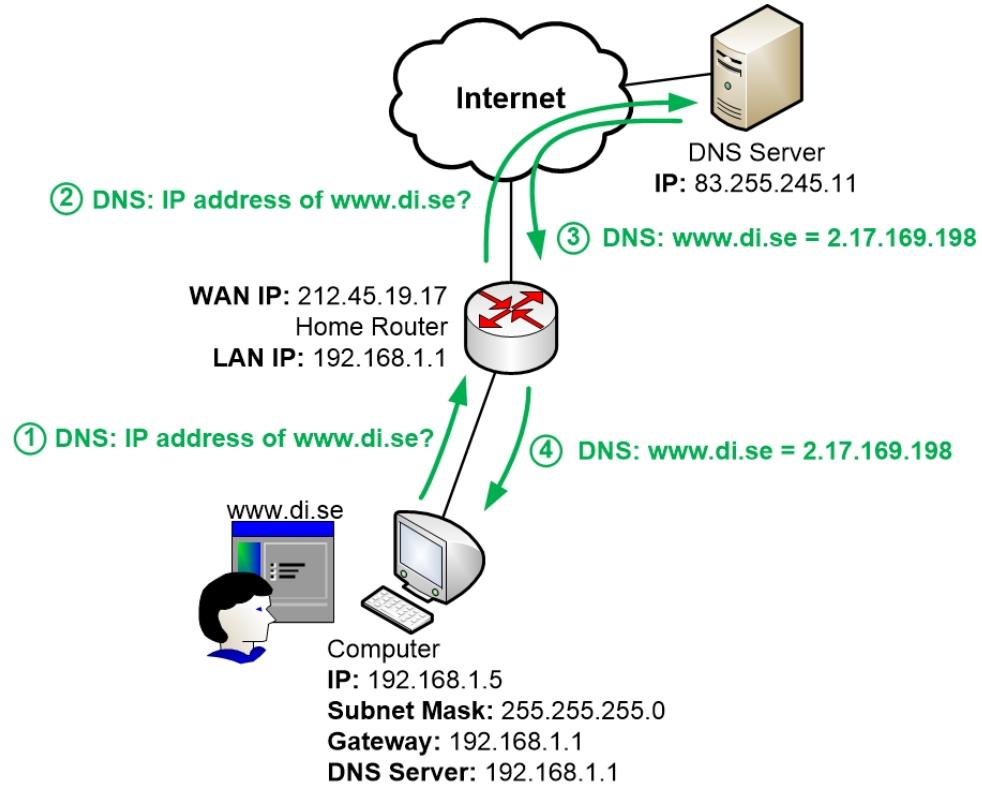
Application	File Transfer	Web Browser	Email	Remote Login	Name Resolution	IP Address
Presentation	FTP TFTP	HTTP	SMTP IMAP POP3	Telnet	DNS	DHCP
Session						
Transport	Transmission Control Protocol TCP				User Datagram Protocol UDP	
Network	Internet Protocol IP				ARP ICMP	
Data Link	Ethernet	Token Ring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

Computer Network Basic

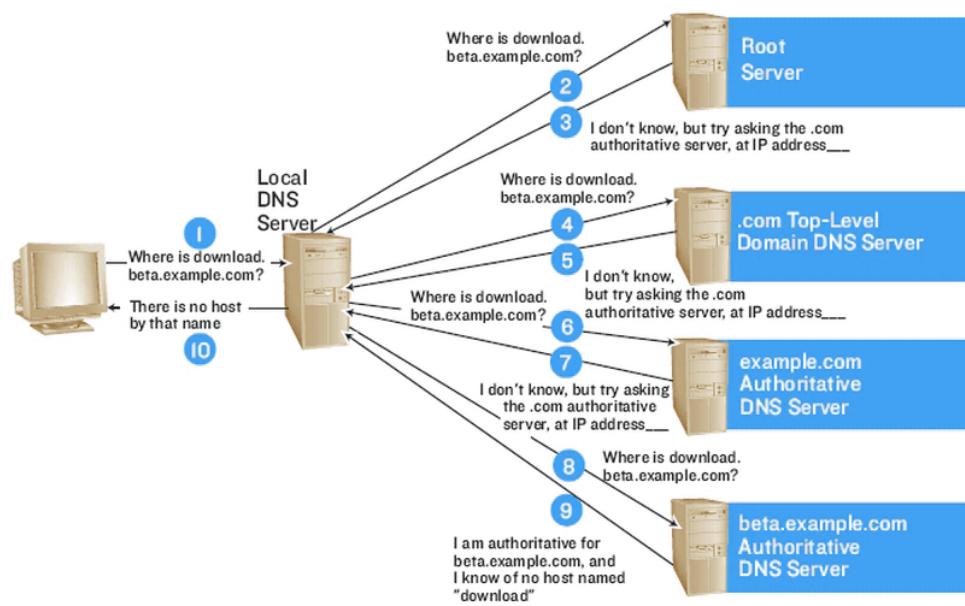
The TCP/IP Protocol Suite

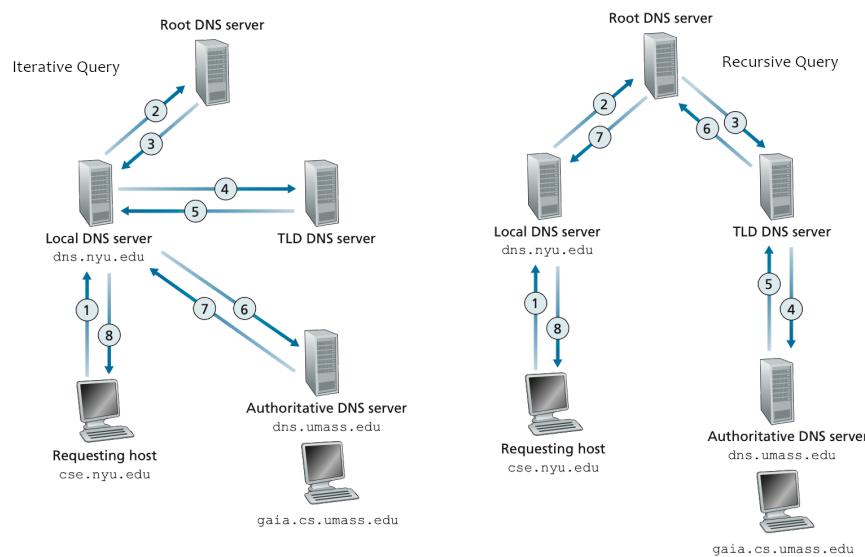
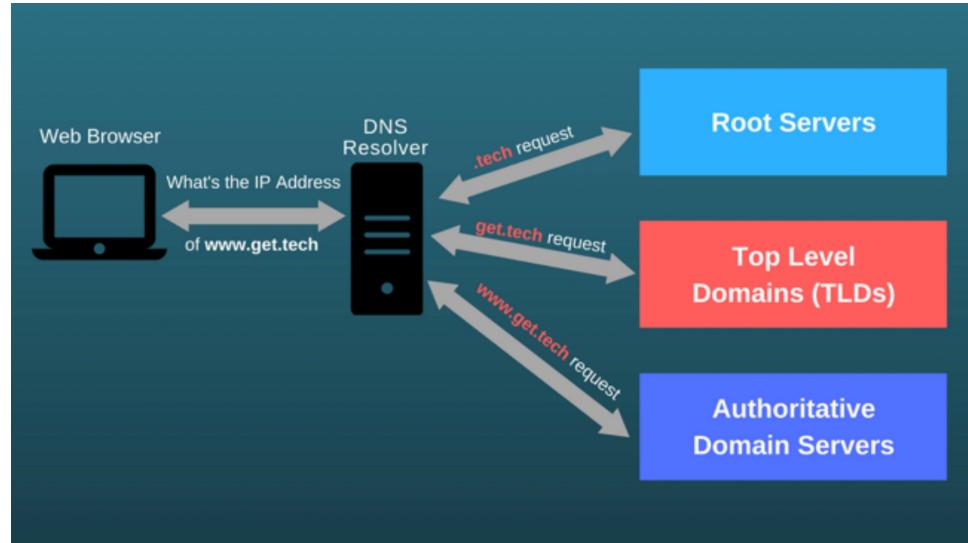


DNS(Domain Name System)



HOW DNS WORKS





Port

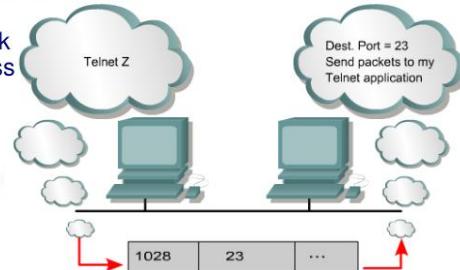
Port #	Application Layer Protocol	Type	Description
20	FTP	TCP	File Transfer Protocol - data
21	FTP	TCP	File Transfer Protocol - control
22	SSH	TCP/UDP	Secure Shell for secure login
23	Telnet	TCP	Unencrypted login
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP/UDP	Domain Name Server
67/68	DHCP	UDP	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			

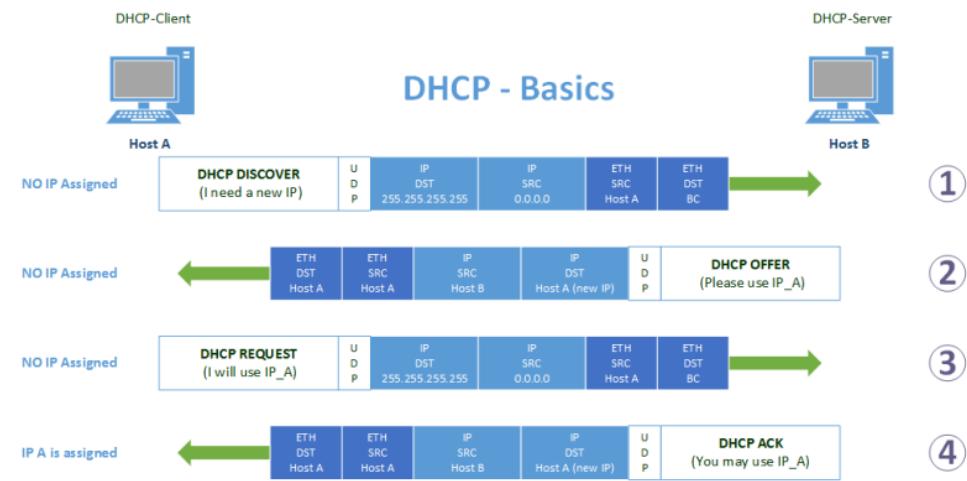
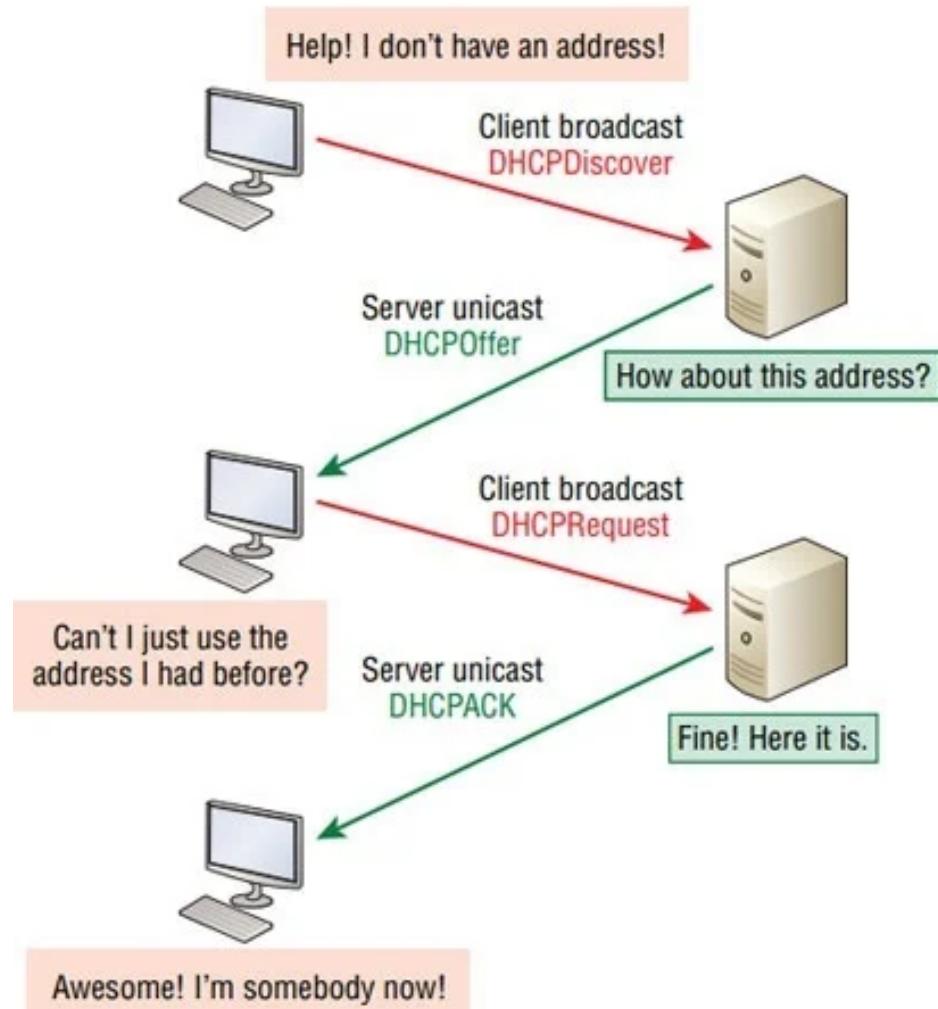
IANA port assignments published at <http://www.iana.org/assignments/port-numbers>

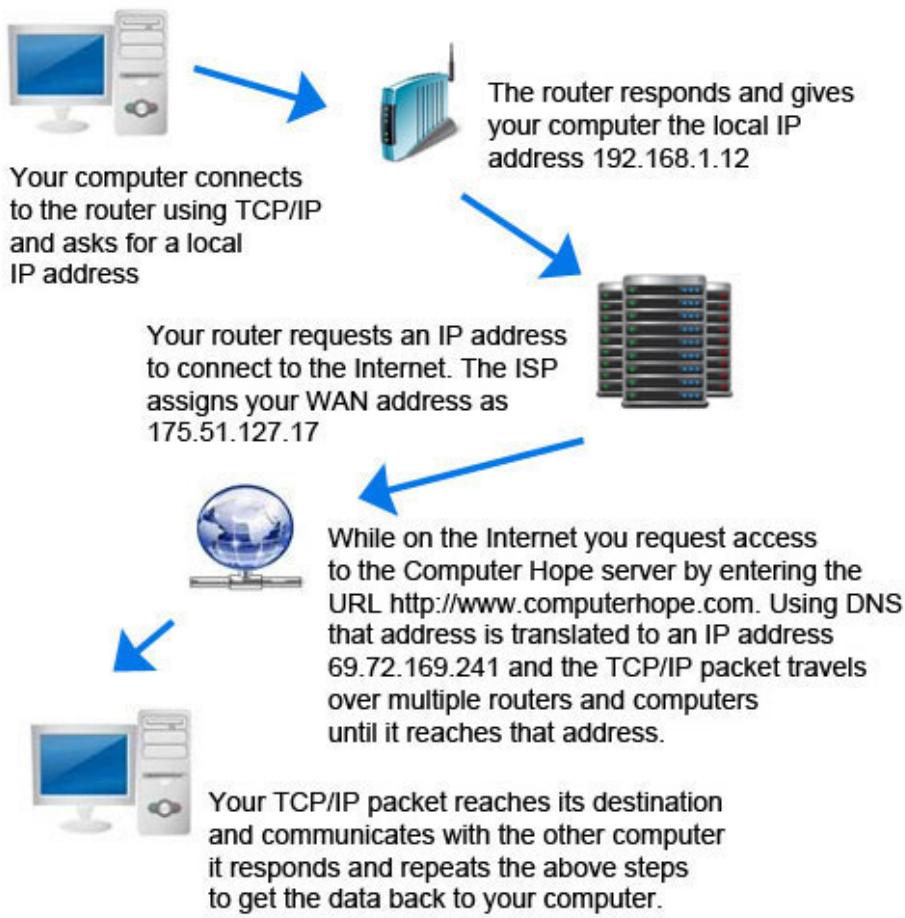
Transport Layer Ports

- 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 .



DHCP





ComputerHope.com

APIPA

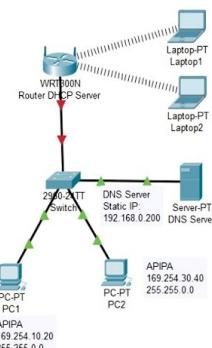
MANA NETWORK
All about EDUCATION

APIPA Class B Private IP v4 Address

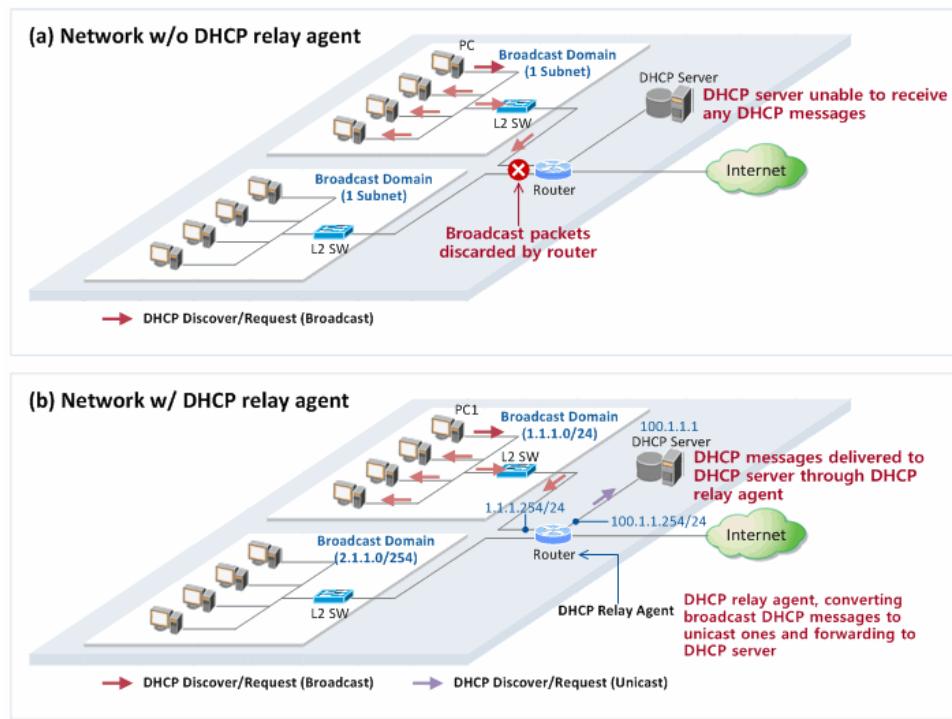
❖ Automatic Private IP Address

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**

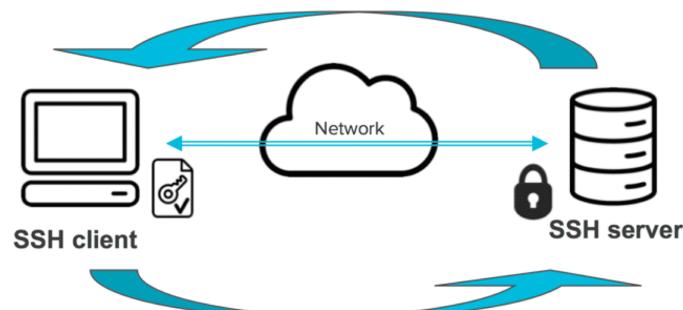


DHCP Relay = IP Helper

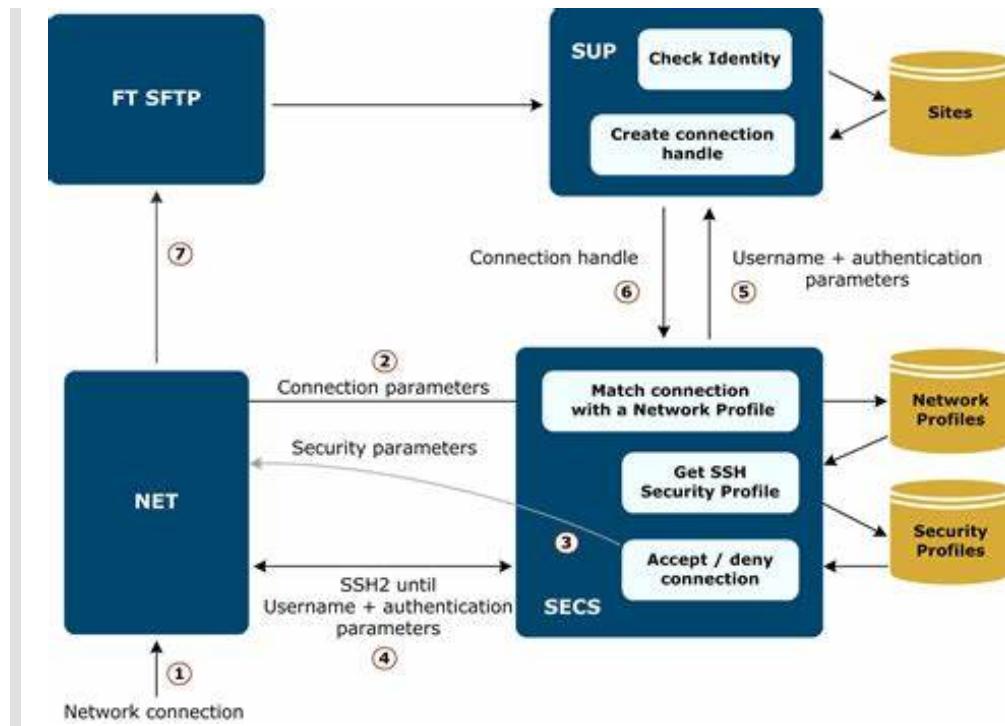


SSH(Secure Shell)

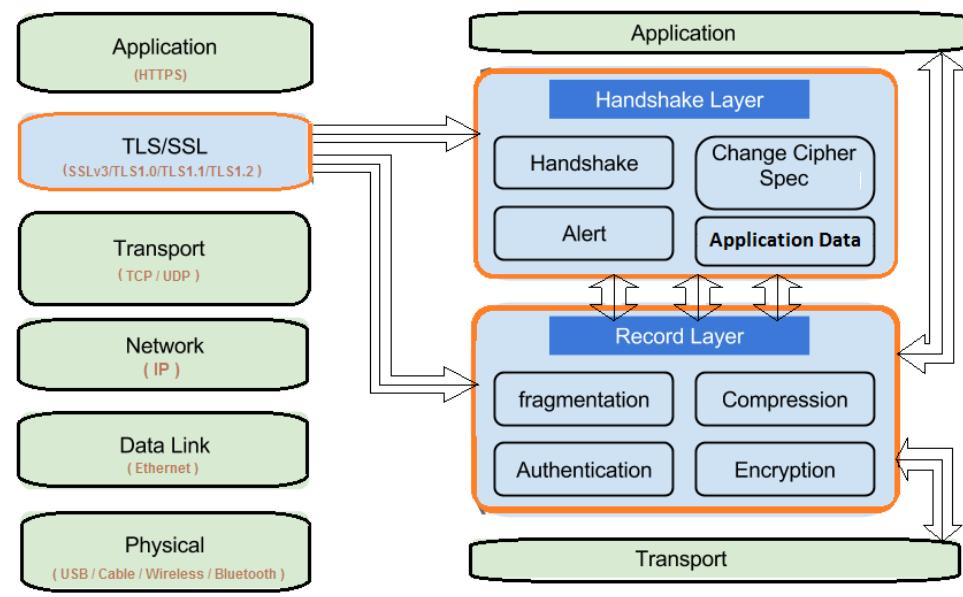
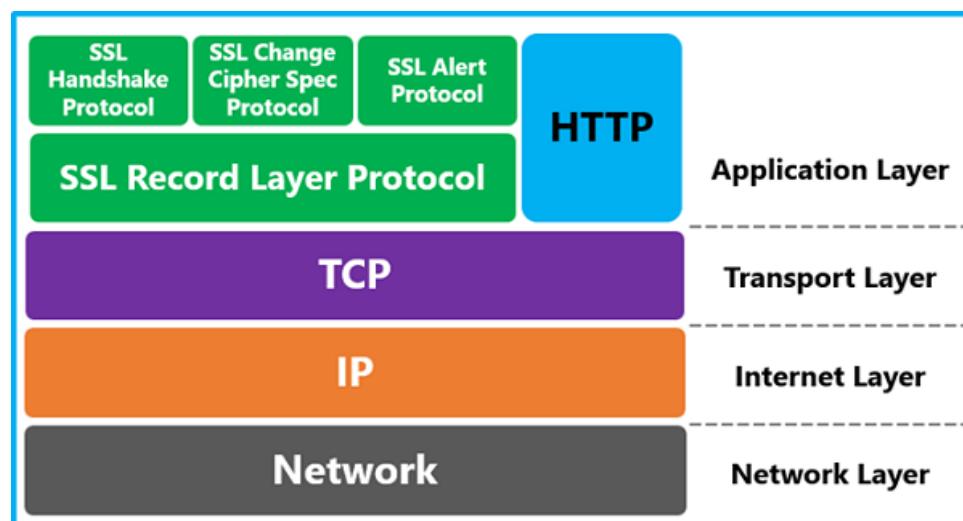
- 1) **Server authentication:**
Server proves its identity to the client

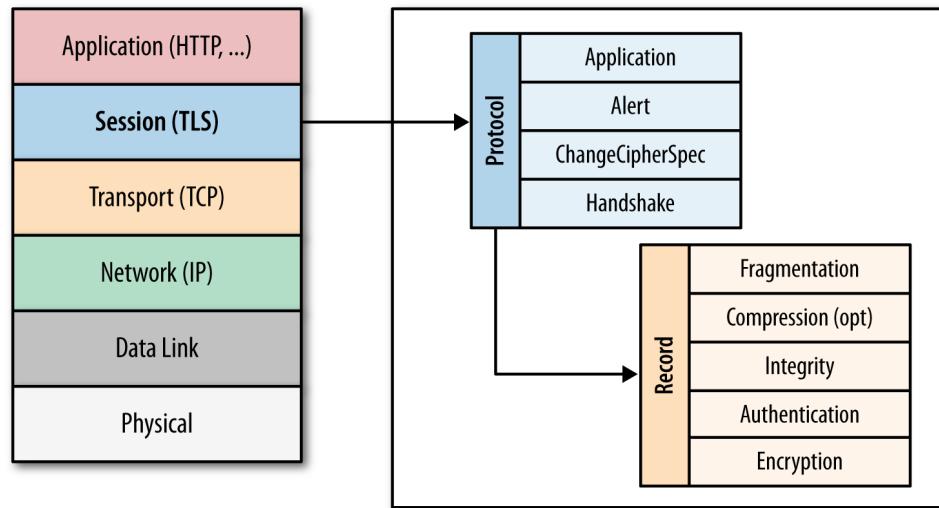


- 2) **User authentication:**
Client proves user's identity to the server

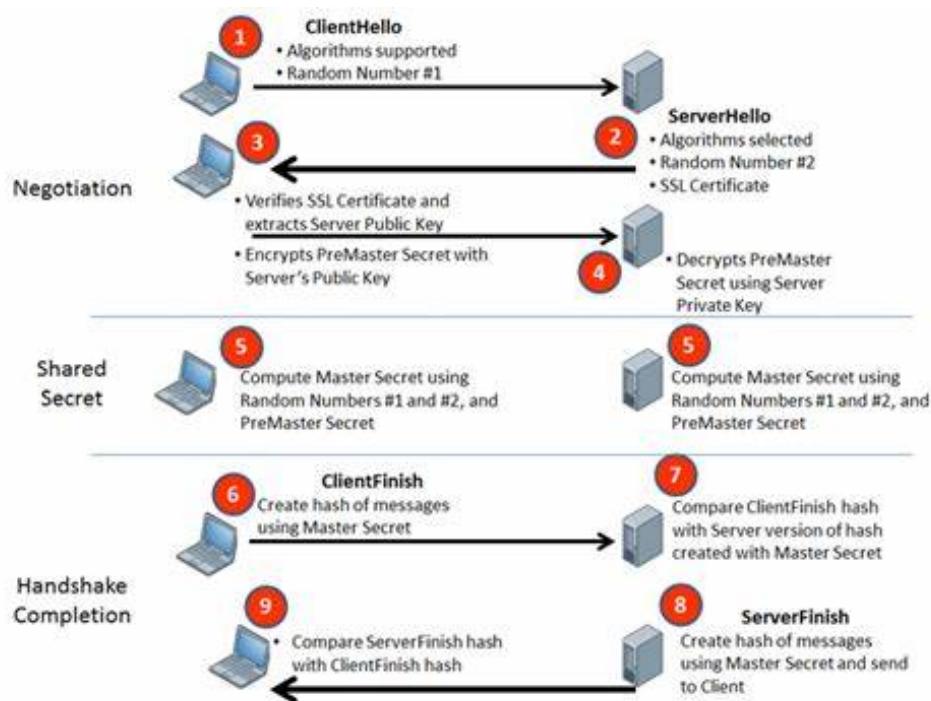
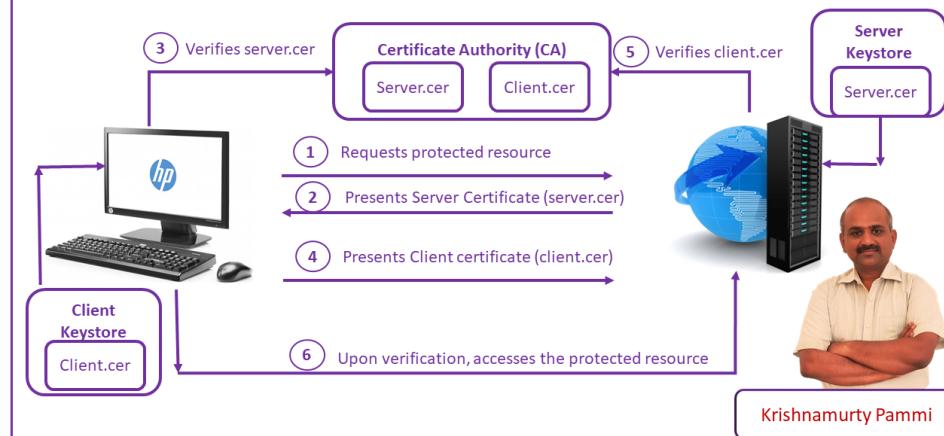


SSL(Secure Sockets Layers)

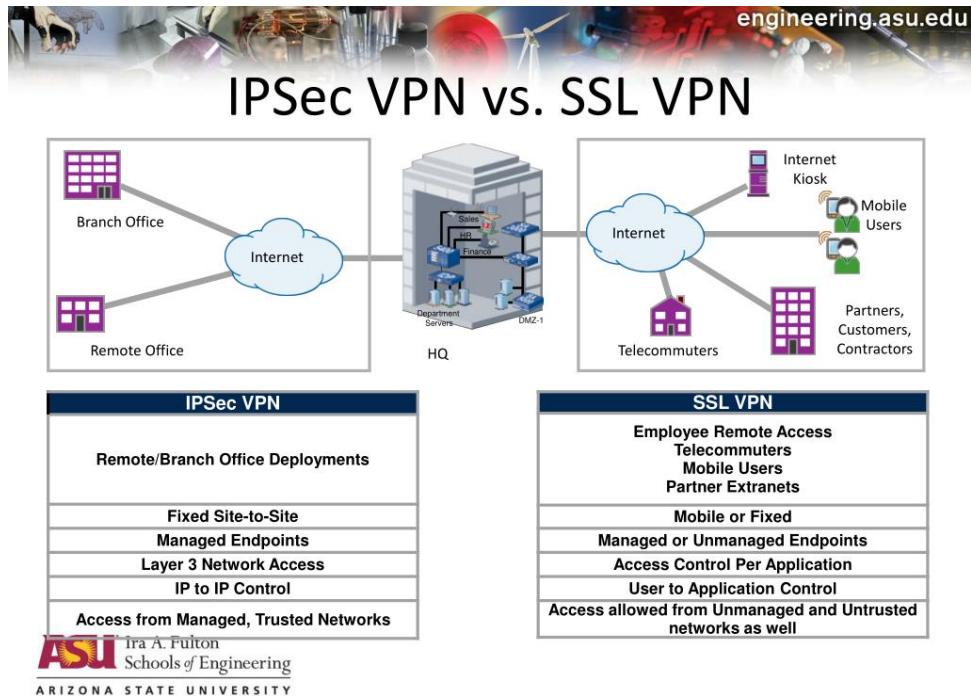
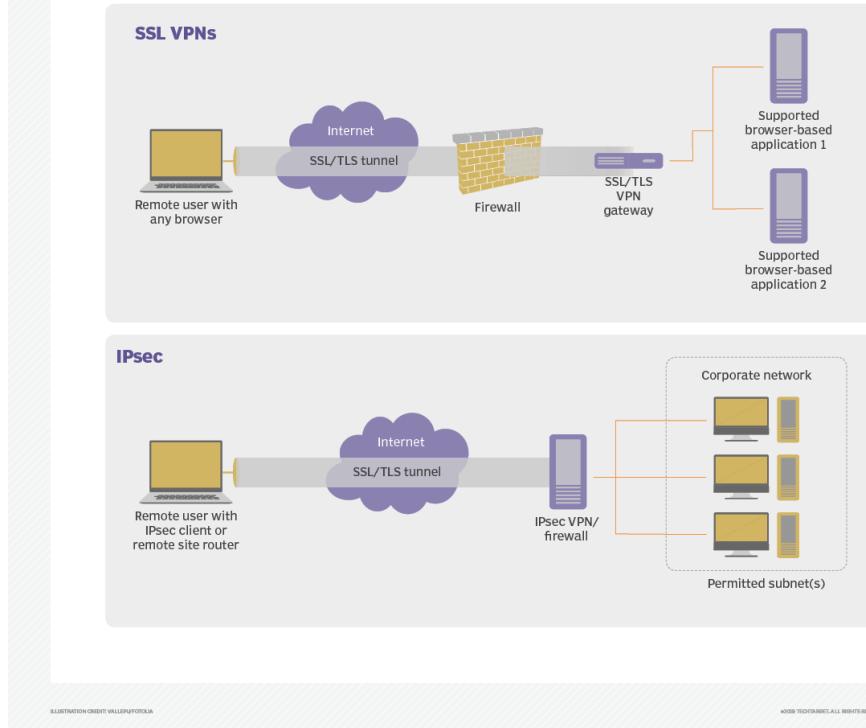




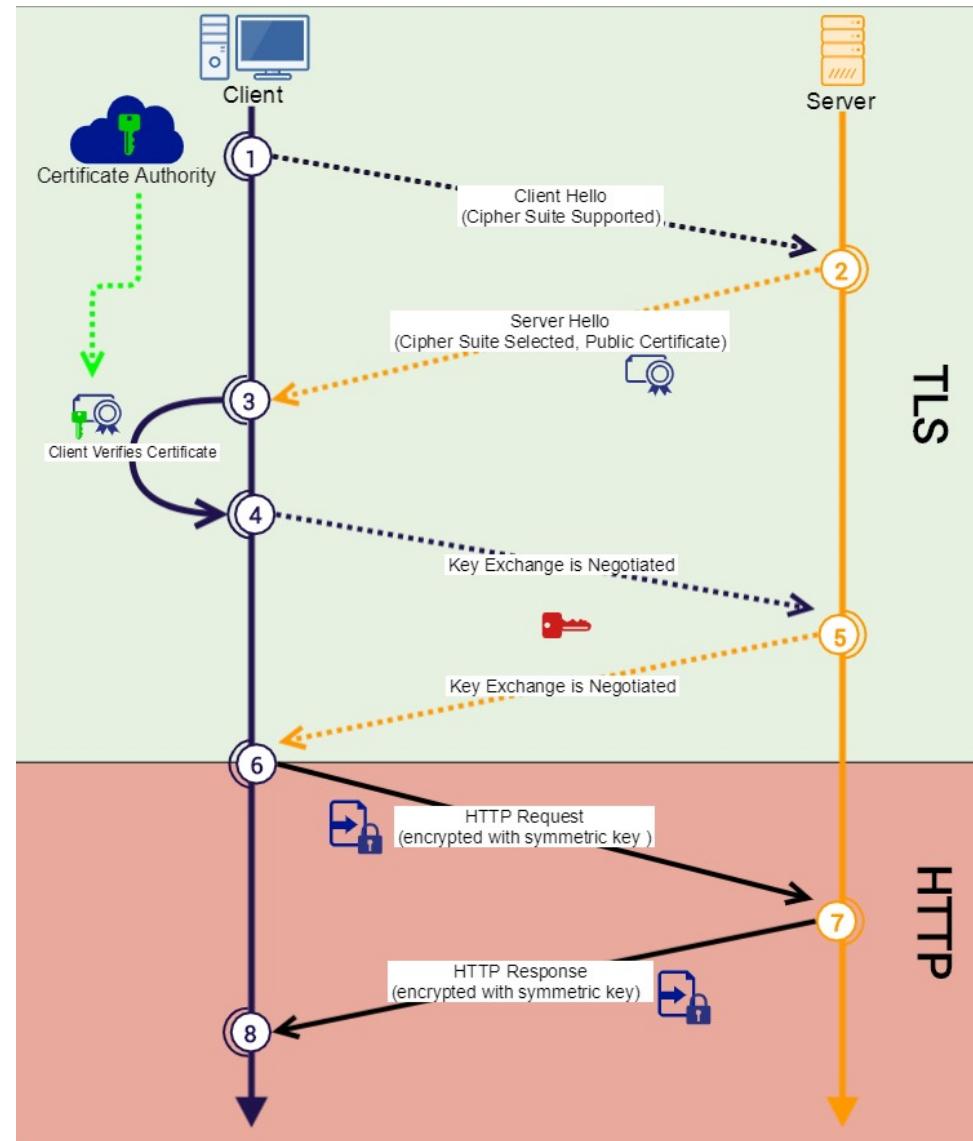
Web Security: Secure Socket Layer (SSL)



SSL/TLS VPNs vs. IPsec VPNs



TLS



SSL vs TSL

S S L**V E R S U S****T L S****SSL**

Standard security protocol for establishing an encrypted link between a web server and a browser

Introduced in the year 1994 by Netscape Communications

Stands for Secure Socket Layer

Not as secure as TSL

Comparatively less complex

TLS

Protocol that provides communication security between client/server applications that communicate with each other over the interne

Introduced in 1999 by Internet Engineering Task Force (IETF)

Stands for Transport Layer Security

More secure

A complex protocol

Visit www.PEDIAA.com

TCP vs UDP**TCP Vs UDP Communication**

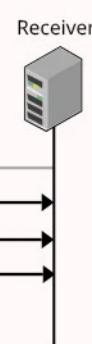
Sender

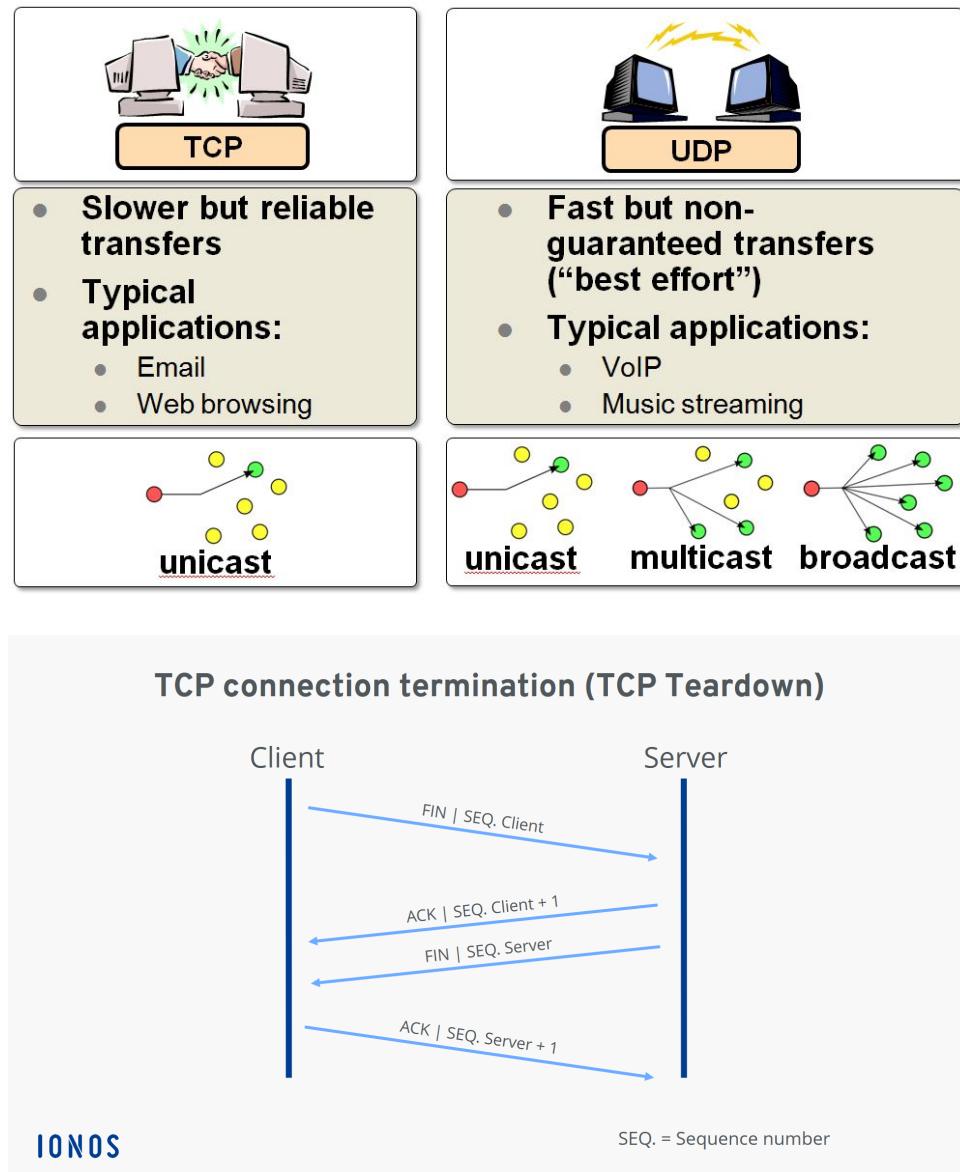
TCP

Receiver

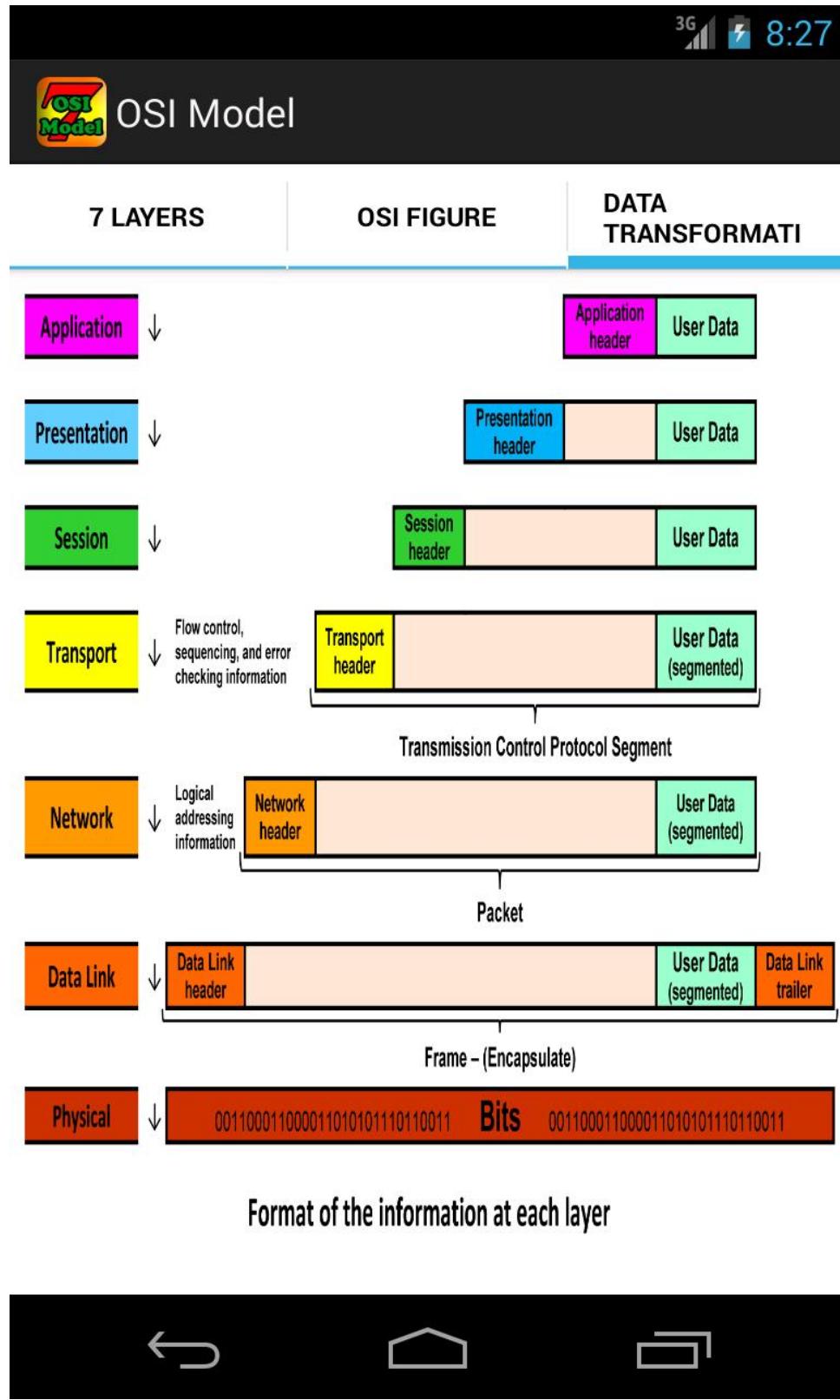


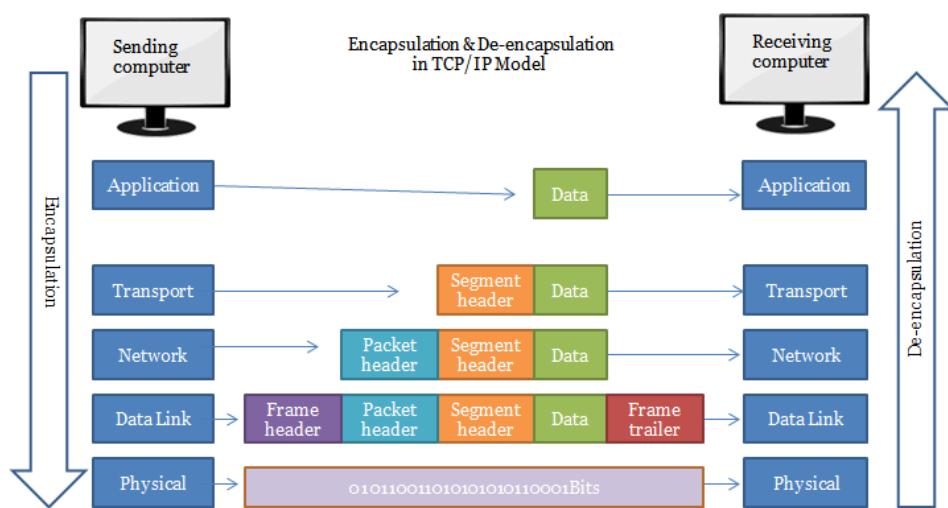
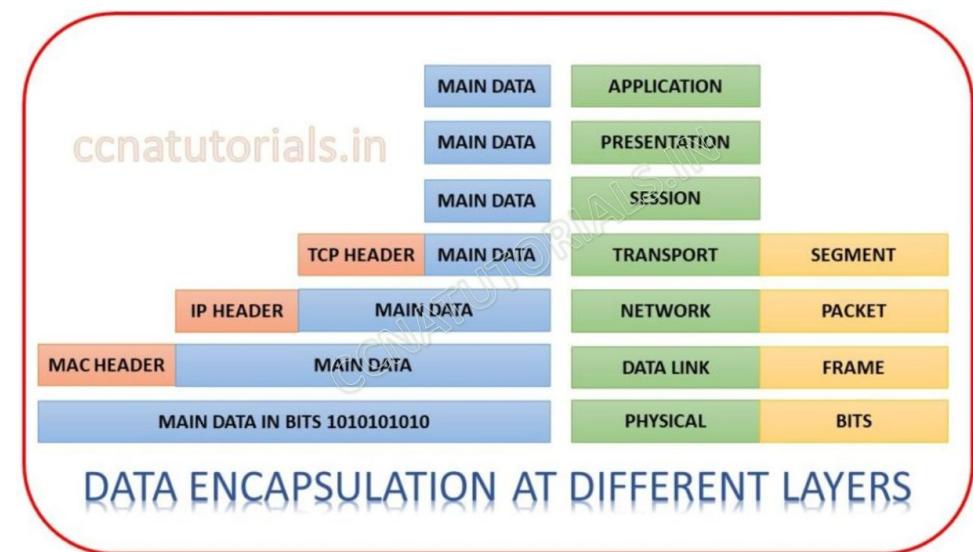
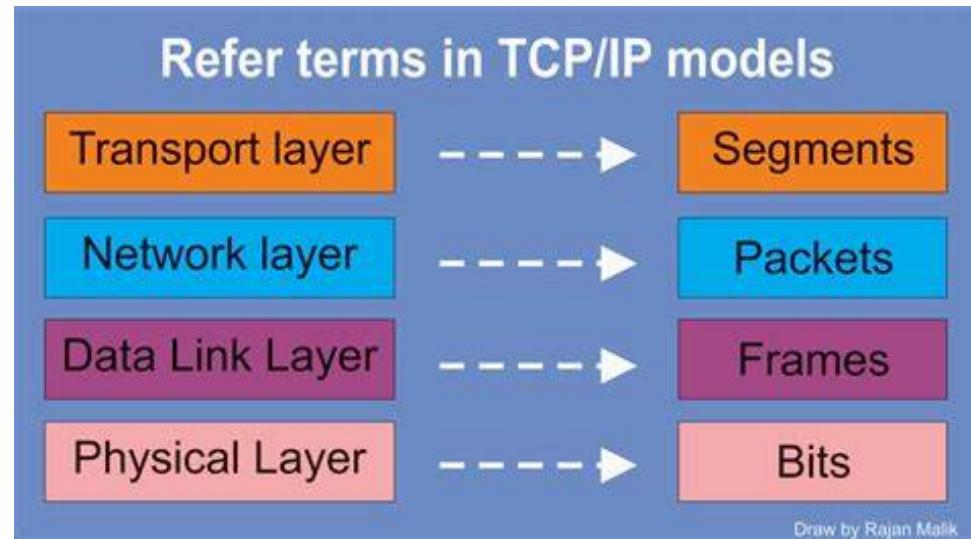
Sender

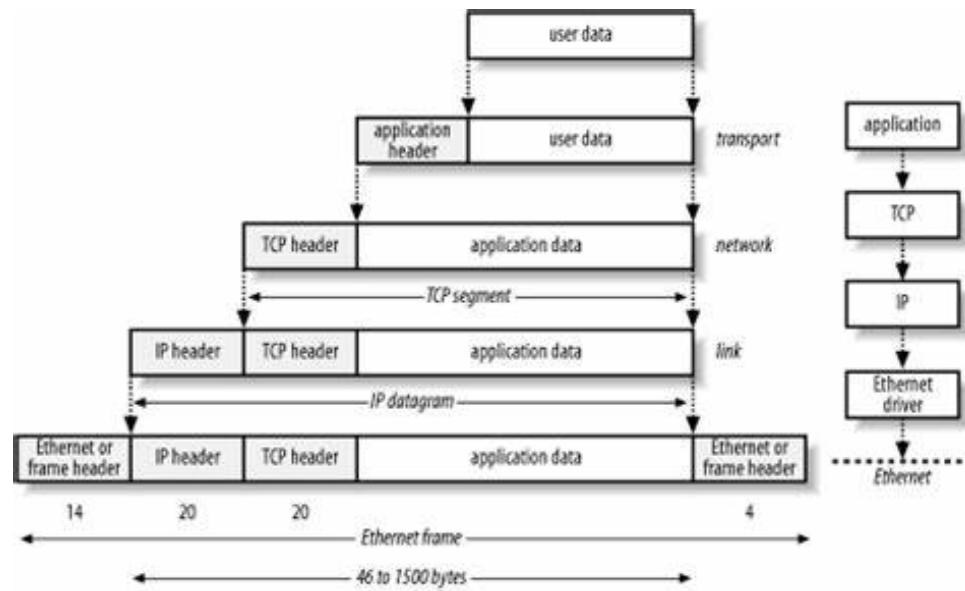
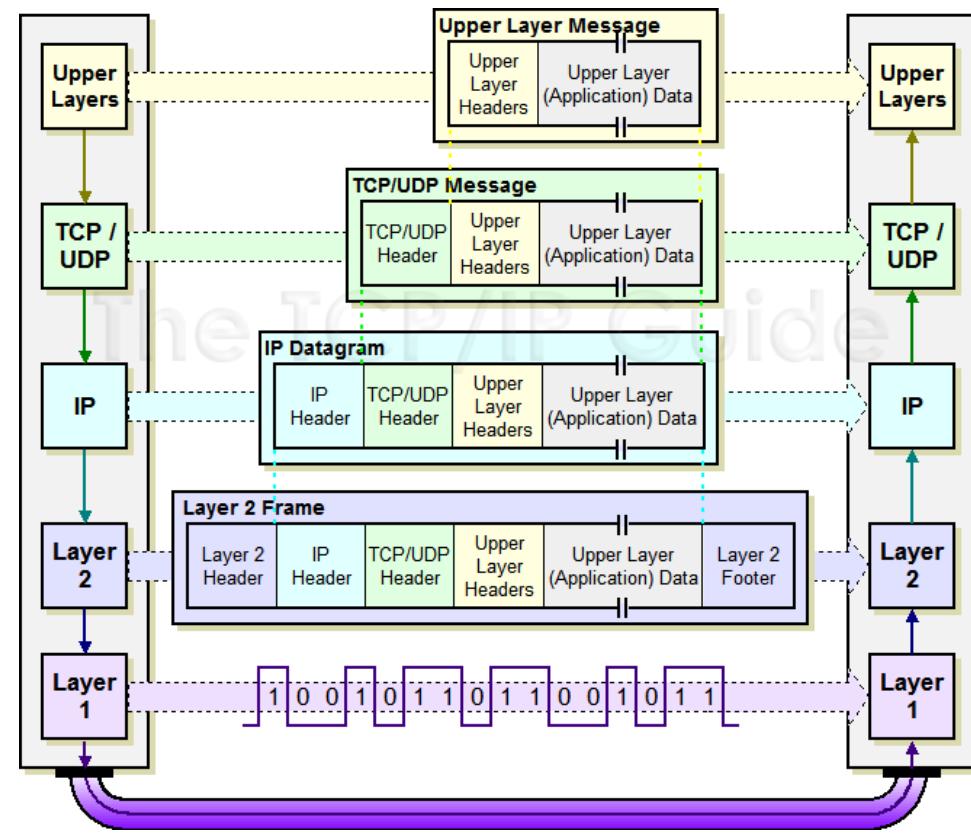
UDP

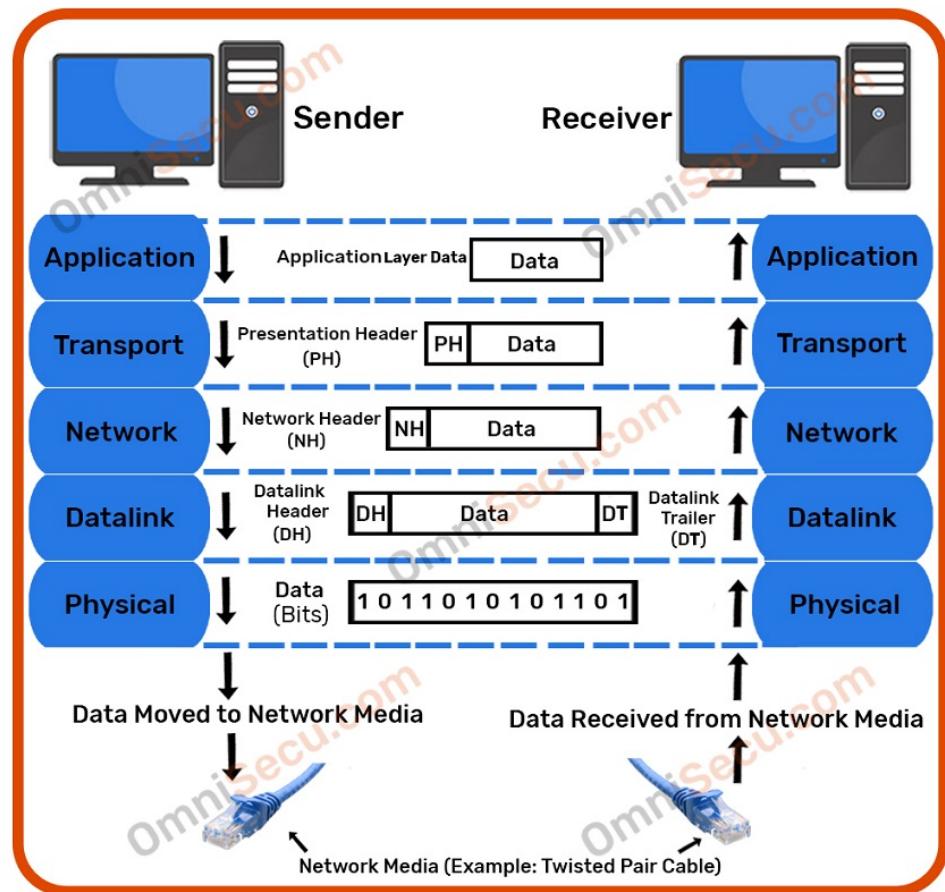


Encapsulation



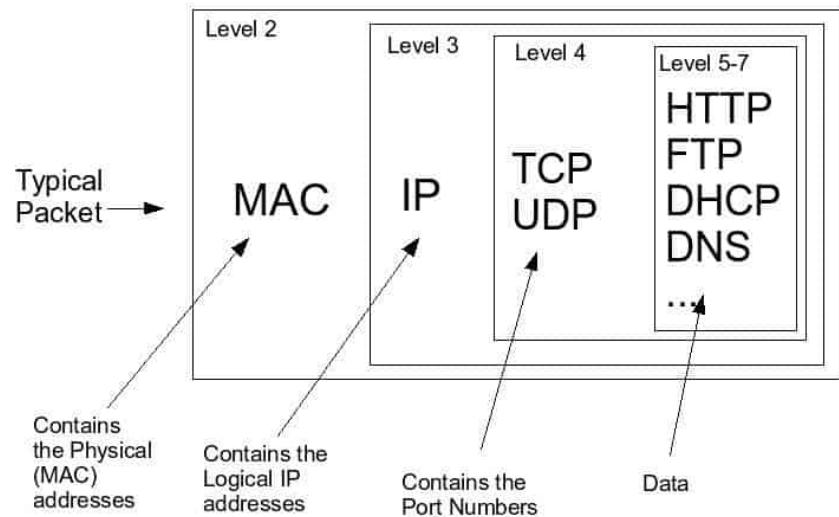




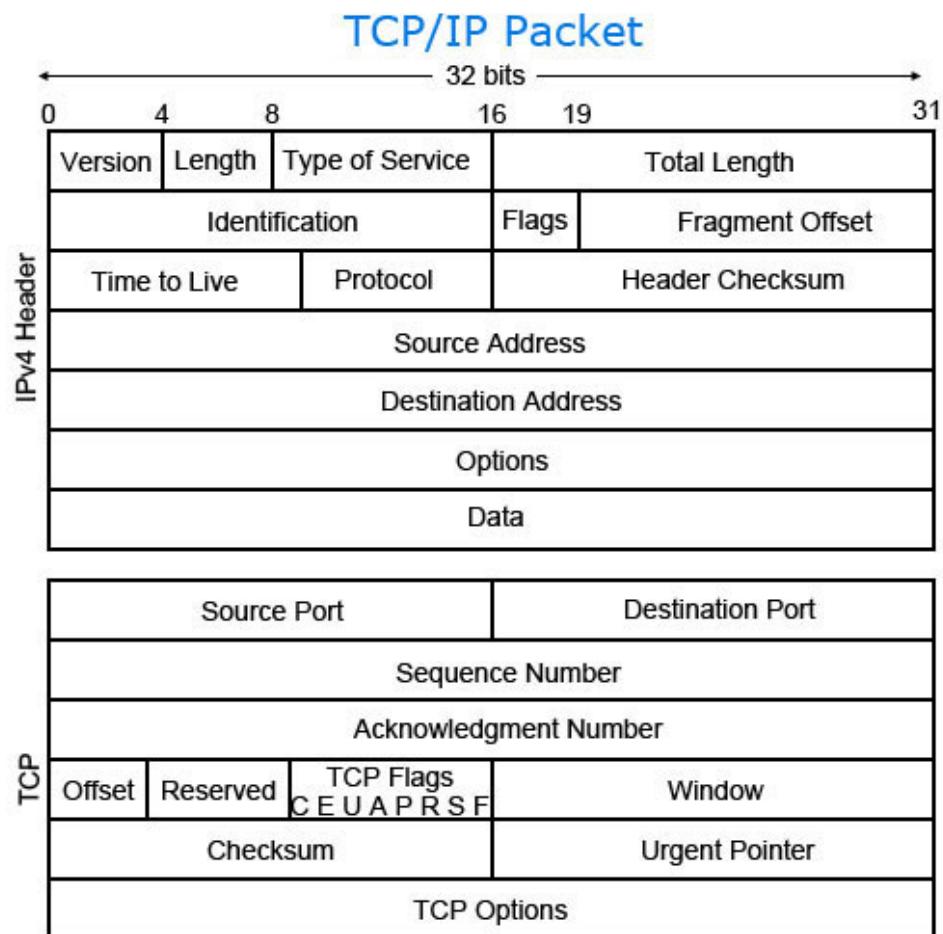
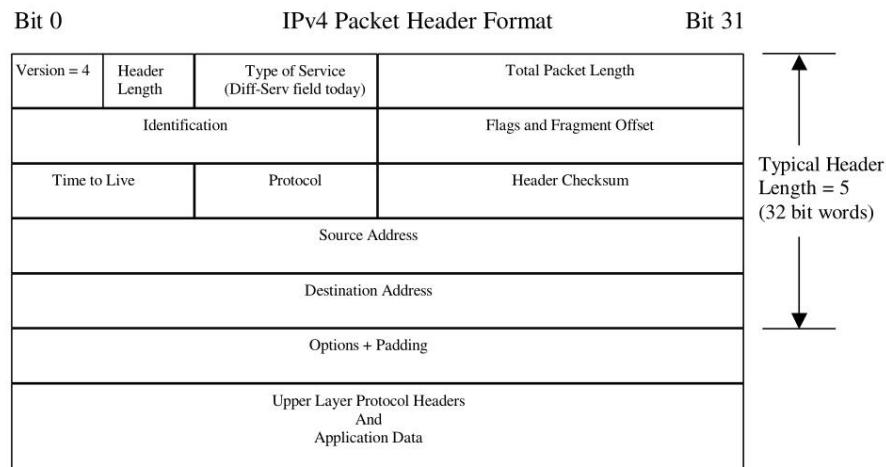


© OmniSecu.com

Packet



IP Packet Format



ComputerHope.com

Packet Switch

How TCP/IP Works

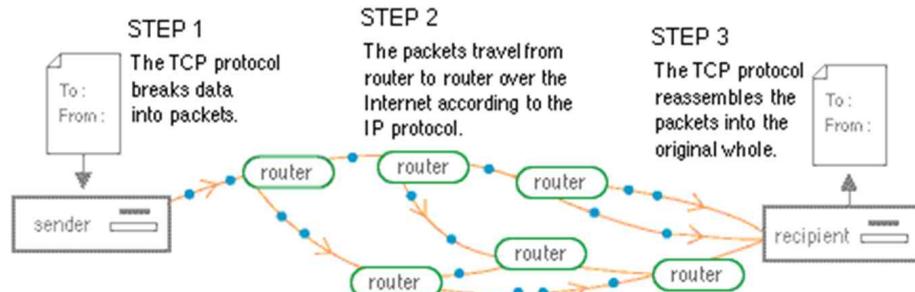


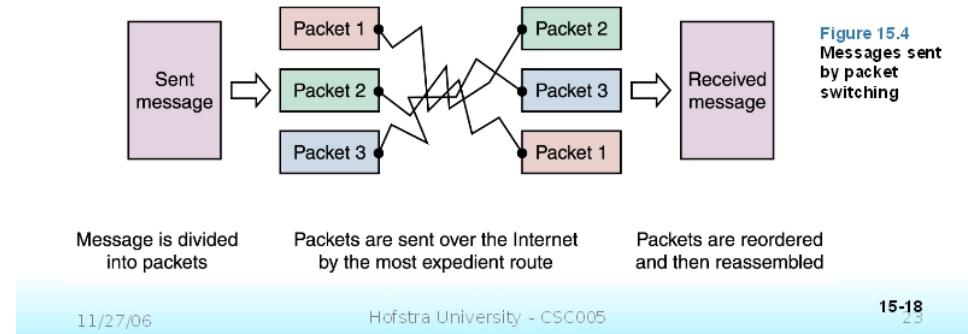
Figure 2. How data travels over the Net.

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

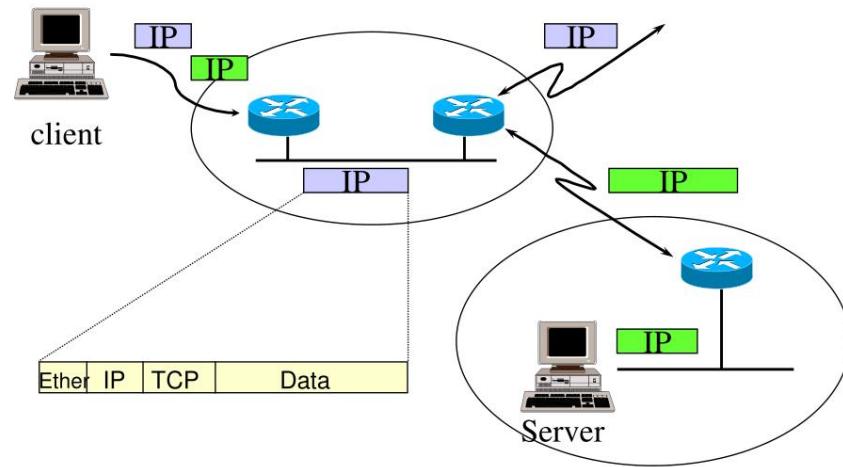


11/27/06

Hofstra University - CSC005

15-18

Packet Switch Network



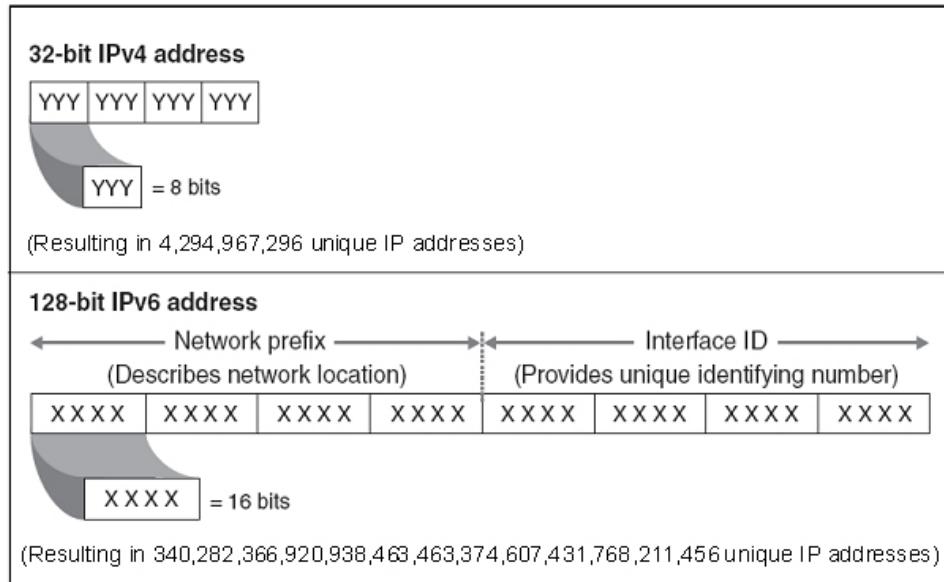
IP4 vs IP6

IPv4 Header				IPv6 Header			
Version	IHL	Type of Service	Total Length	Version		Traffic Class	Flow Label
Identification		Flags	Fragment Offset	Payload Length		Next Header	Hop Limit
Time to Live	Protocol	Header Checksum		Source Address			
Destination Address				Destination Address			
Options		Padding		Source Address			

Legend

- Yellow: Field's Name Kept from IPv4 to IPv6
- Red: Fields Not Kept in IPv6
- Blue: Name and Position Changed in IPv6
- Teal: New Field in IPv6

Figure 1: Comparison of IPv6 and IPv4 Address Scheme



Source: GAO.

Differences Between IPv4 and IPv6

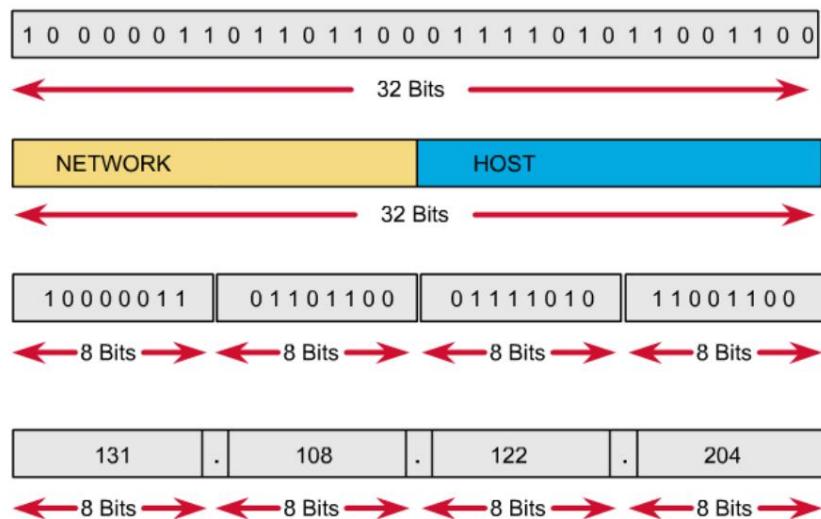
Feature	IPv4	IPv6
Fragmentation	Performed by routers and sending host	Performed only by sending host
Address Resolution	Broadcast ARP Request frames	Multicast Neighbor Solicitation messages
Manage multicast group membership	IGMP	Multicast listener discovery
Router Discovery	ICMP Router Discovery (optional)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

IPv4/IPv6 Differences

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

IP Address

IP address format

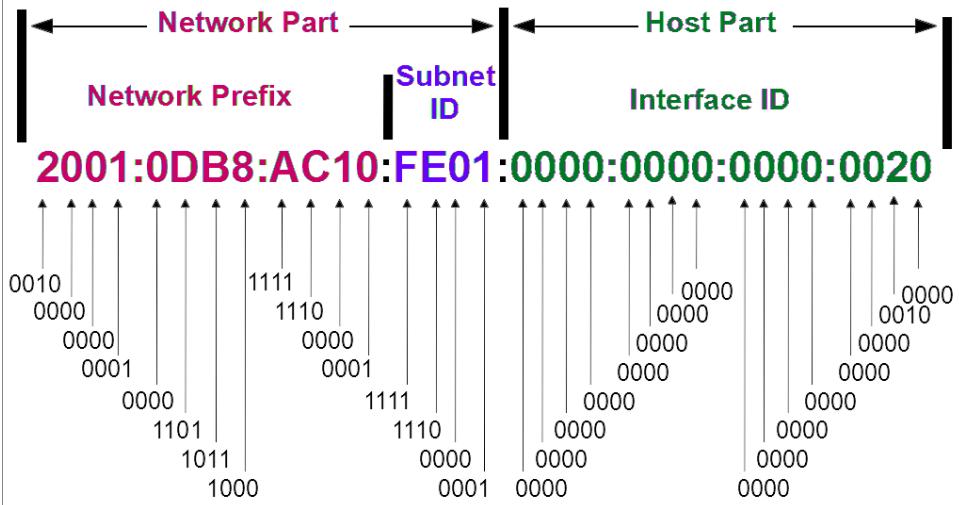


Class	Private Address Ranges
Class A	10.0.0.0 – 10.255.255.255
Class B	172.16.0.0 – 172.31.255.255
Class C	192.168.0.0 – 192.168.255.255
Loopback	127.0.0.0 – 127.255.255.255 (127.0.0.1)

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

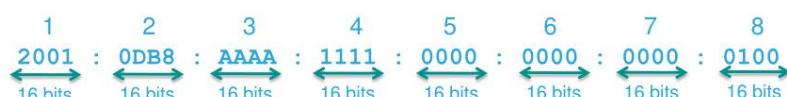


IPv6 Address Notation

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



- 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>0</u> 0000000 – <u>0</u> 1111111 (0 – 127)	1.0.0.1 – 126.255.255.254	126	$2^{24} - 2$
B	<u>1</u> 0000000 – <u>1</u> 0111111 (128 – 191)	128.1.0.1 – 191.255.255.254	16000	65000
C	<u>1</u> 1000000 – <u>1</u> 1011111 (192 – 223)	192.0.1.1 – 223.255.255.254	2 Million	254
D	<u>1</u> 1100000 – <u>1</u> 1101111 (224 – 239)	224.0.0.0 – 239.255.255.255		Multicast addresses
E	<u>1</u> 1110000 – <u>1</u> 1111111 (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>0</u> 0000001 to <u>0</u> 1111110*	16,777,214
Class B	128-191	<u>1</u> 0000000 to <u>1</u> 0111111	65,534
Class C	192-223	<u>1</u> 1000000 to <u>1</u> 1011111	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

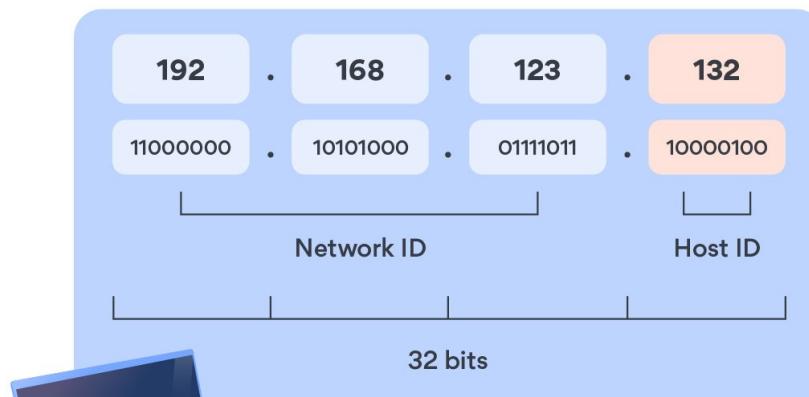
127.0.0.0 – 127.255.255.255
(127.0.0.1)

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

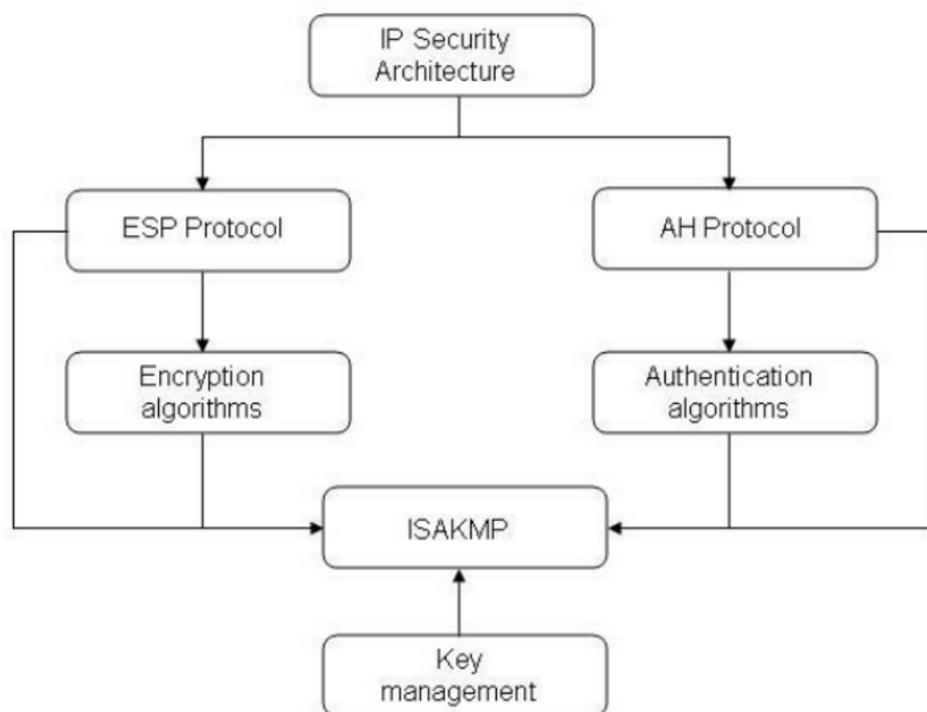
- A "1" bit in the subnet mask means that the corresponding bit in the IP address should be read as a network number
- A "0" bit in the subnet mask means that the corresponding bit in the IP address should be read as a host bit.
- /n "slash" tells us how many "1" bits are in the subnet mask.

The Subnet Mask

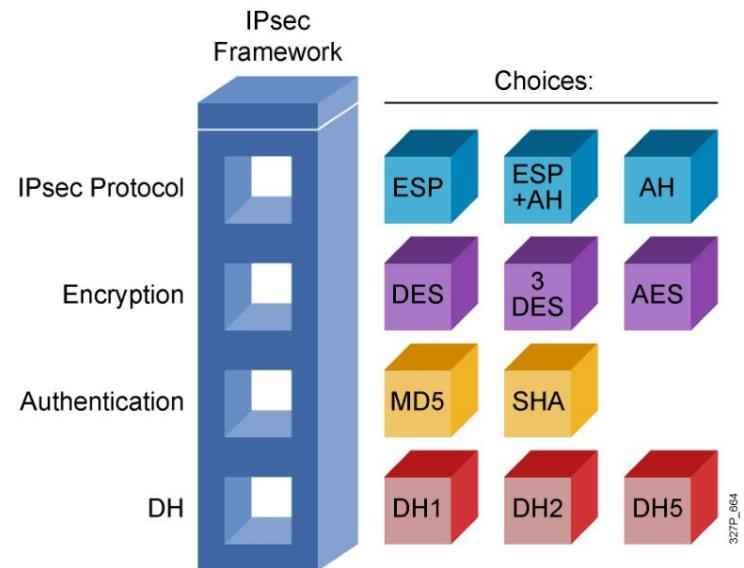
- Subnet Mask:
 - Let's not forget about the subnet mask.
 - Each class has a **default or "natural"** subnet mask based on the default number of bits used for the network and host portion.

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

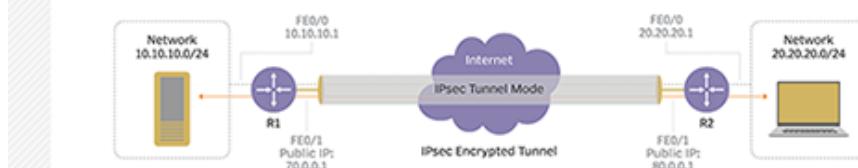
IPSec



IPsec Protocol Framework



IPsec tunnel mode



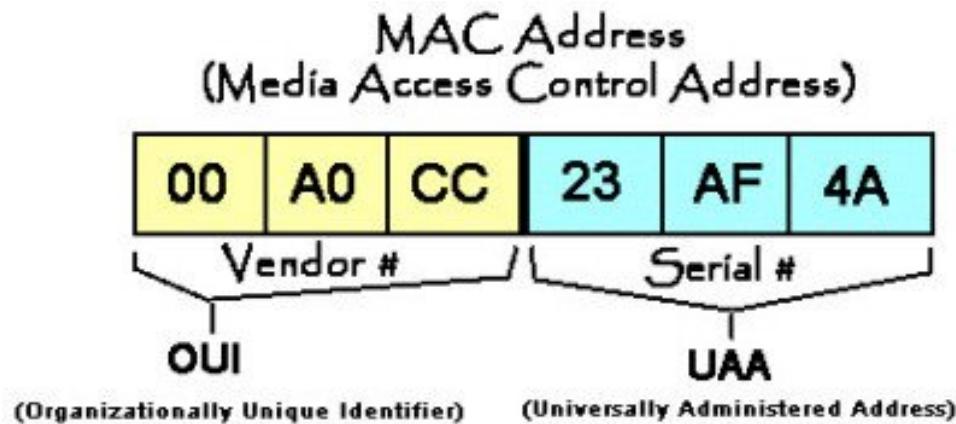
IPsec: Network Layer Security

- **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

MAC Address - DataLink Layer



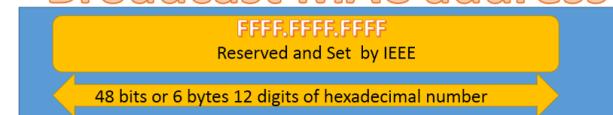
Unicast MAC address



Multicast MAC address

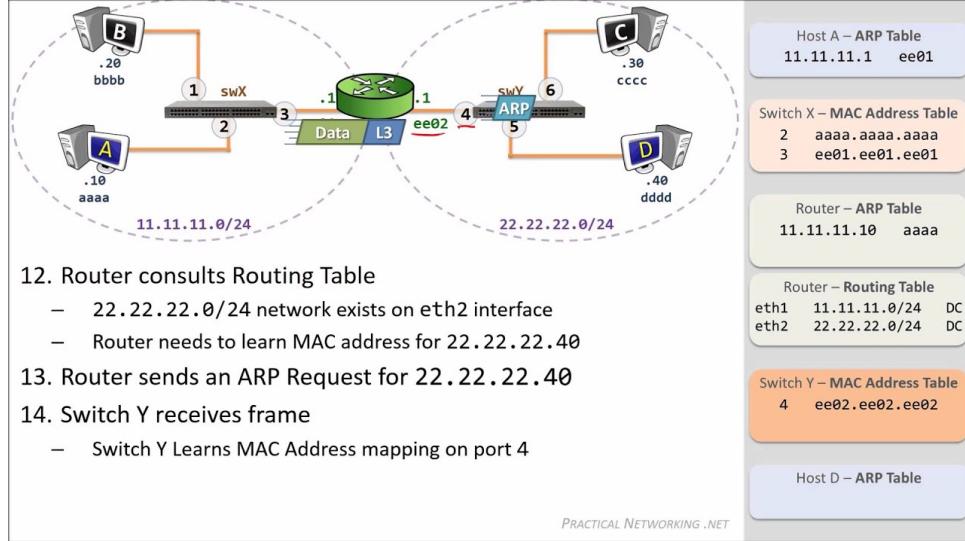


Broadcast MAC address



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

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

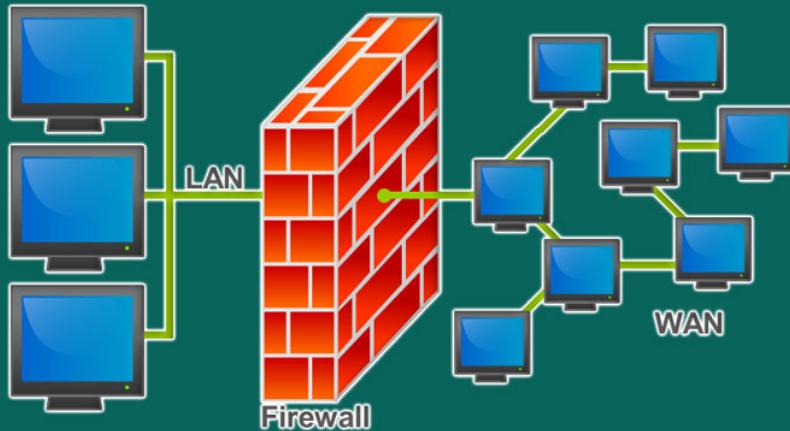


12. Router consults Routing Table
 - 22.22.22.0/24 network exists on eth2 interface
 - Router needs to learn MAC address for 22.22.22.40
13. Router sends an ARP Request for 22.22.22.40
14. Switch Y receives frame
 - Switch Y Learns MAC Address mapping on port 4

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.

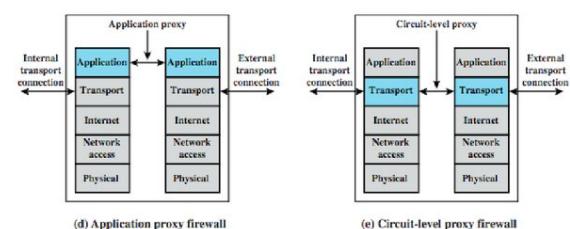
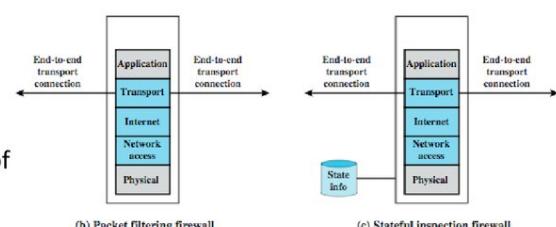
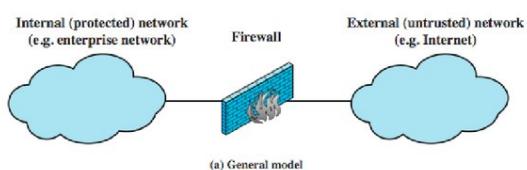
It simply prevents unauthorized access to or from a private network. Used to enhance the security of computers connected to a network, such as LAN or the Internet. Considered as an integral part of a comprehensive security framework for your network.

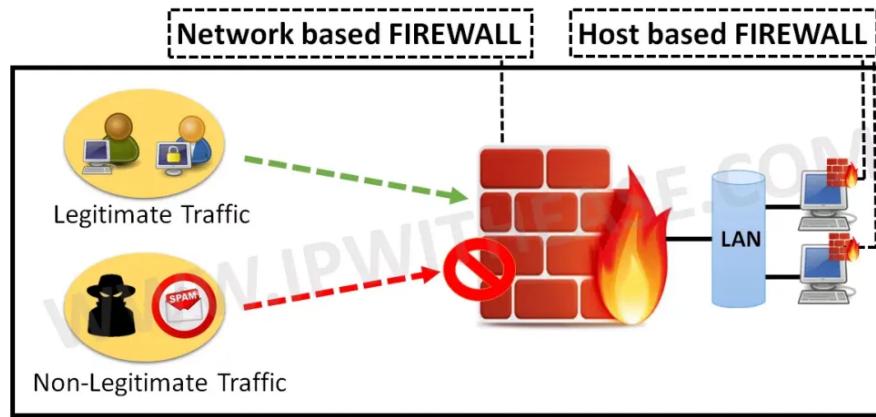


Types of Firewalls

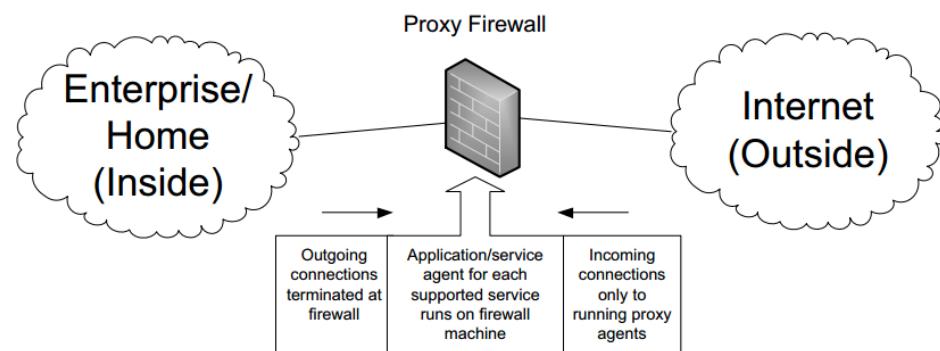
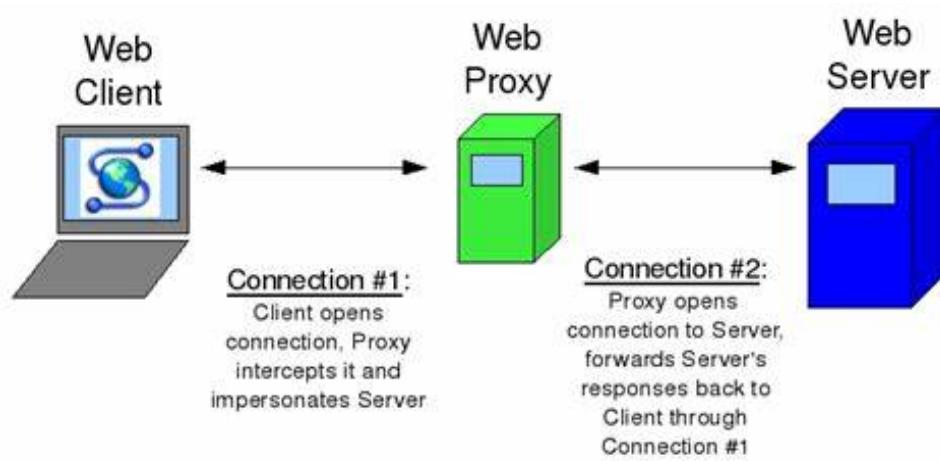
Positive (negative) filter:
Allow (reject) packets that meet a criteria

Stateful inspection: Keeps track of TCP connections





Proxy



Proxy Cache

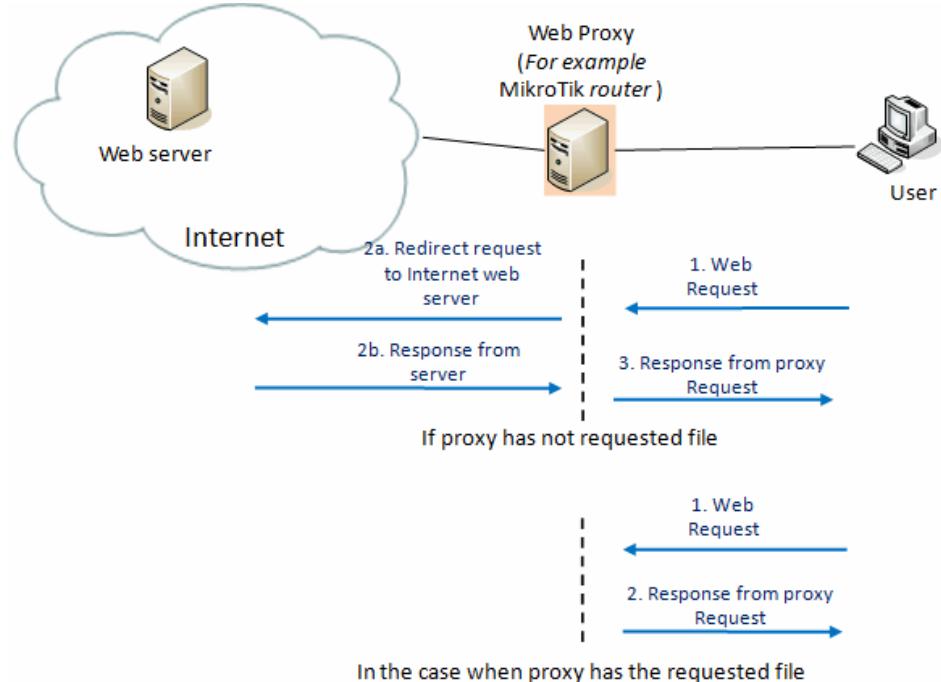
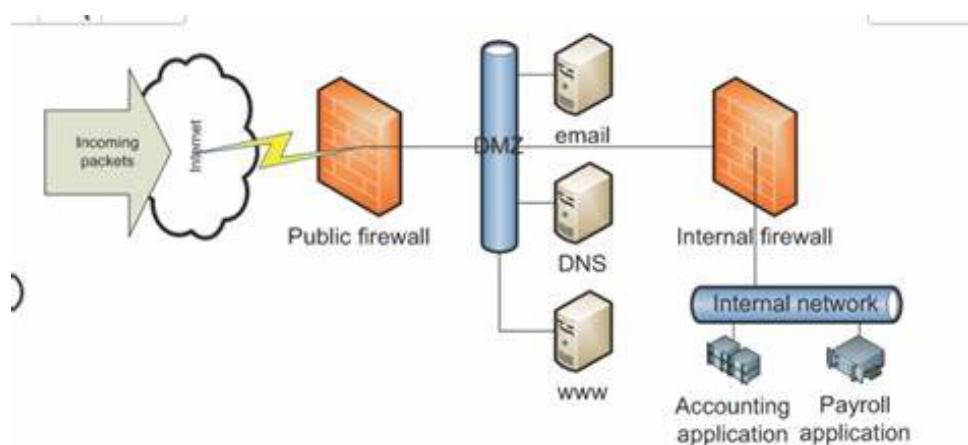


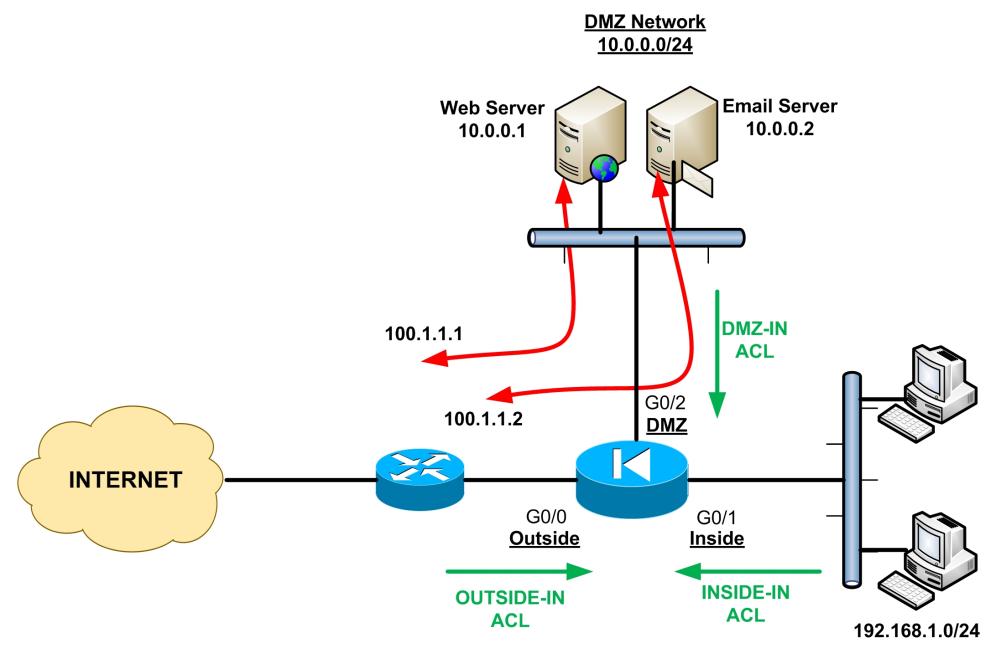
Figure 10.1. Web proxy basic operation scheme

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

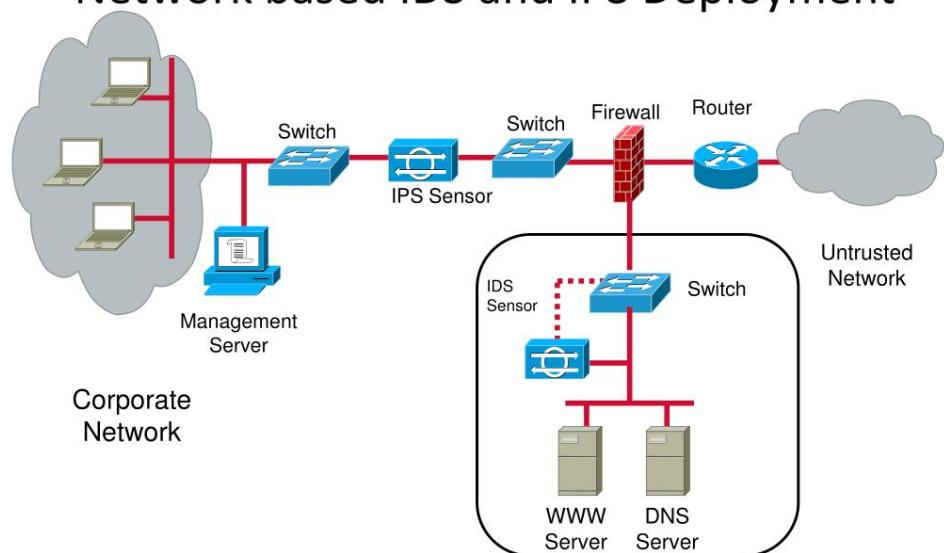
DMZ





IPS/IDS

Network based IDS and IPS Deployment



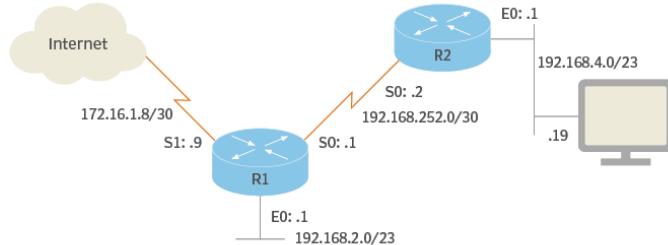
Engineering and Management of Secure Computer Networks

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

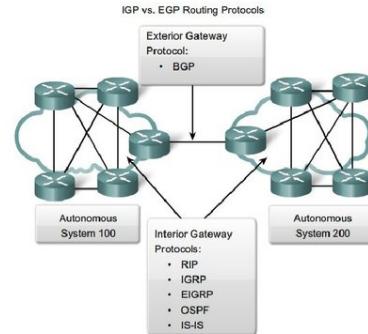
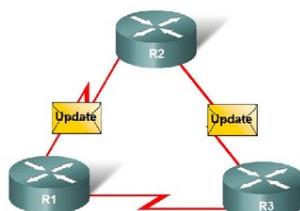
Dynamic IP Routing Protocols

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).

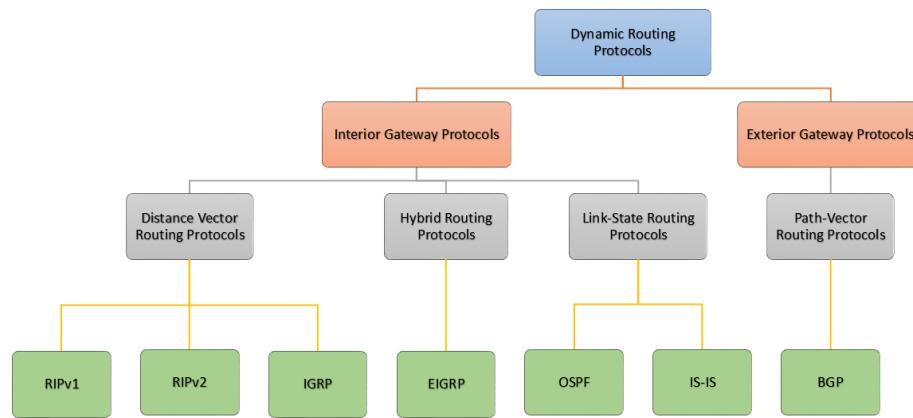
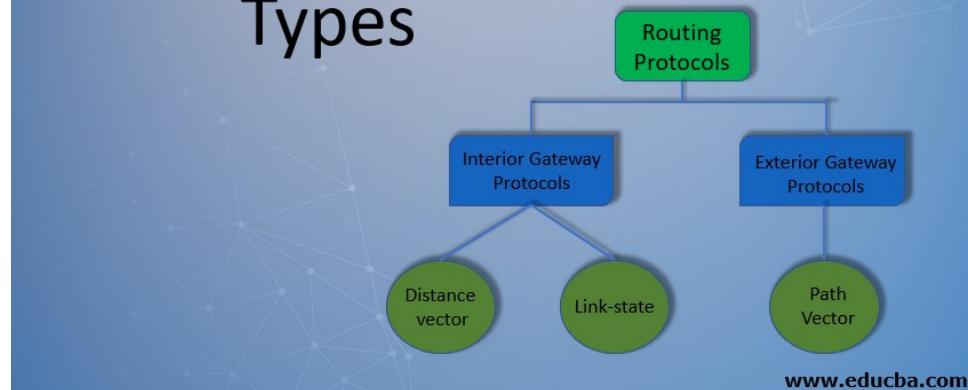
Routers Dynamically Pass Updates



16

Routing Protocols

Types



Routing Protocols for IP Networks

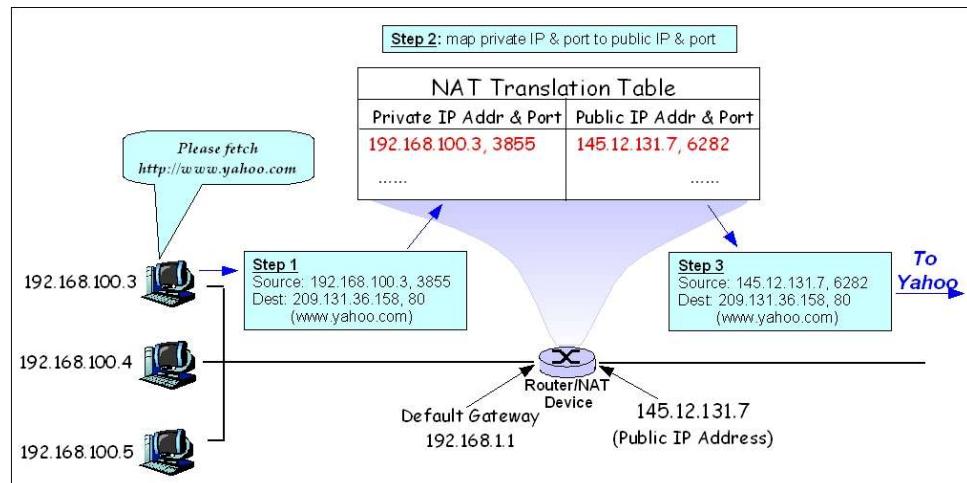
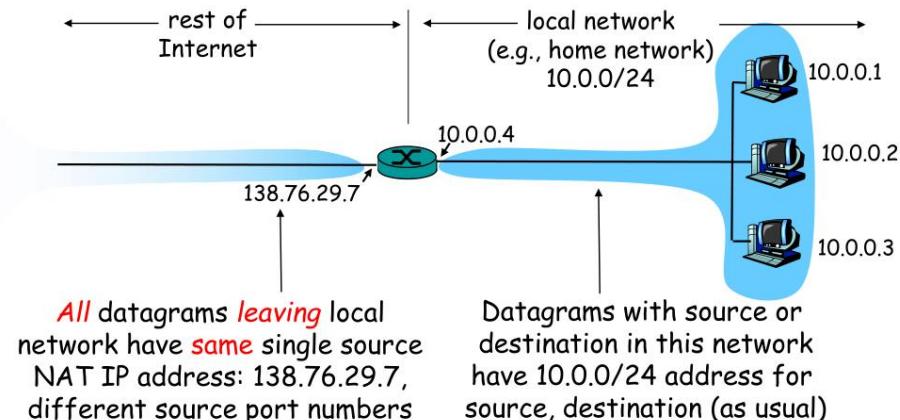
Protocol	Type	Scalability	Metric	IP classes
RIP-1	Distance vector	Small	Hop count	Classful
RIP-2	Distance vector	Small	Hop count	Classless
OSPF-2	Link state	Large	Cost	Classless
IS-IS	Link state	Very large	Cost	Classless
IGRP	Distance vector	Medium	Bandwidth, delay, load, MTU, reliability	Classful
EIGRP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



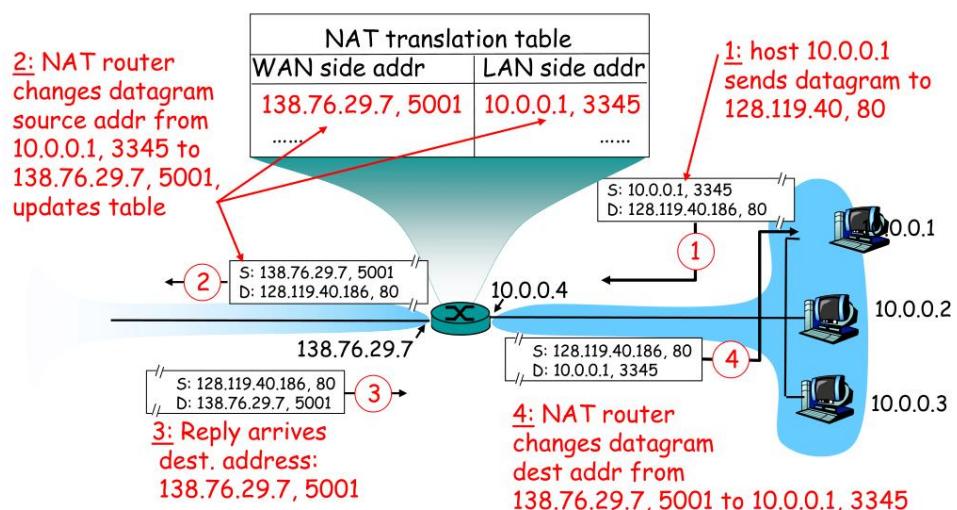
7

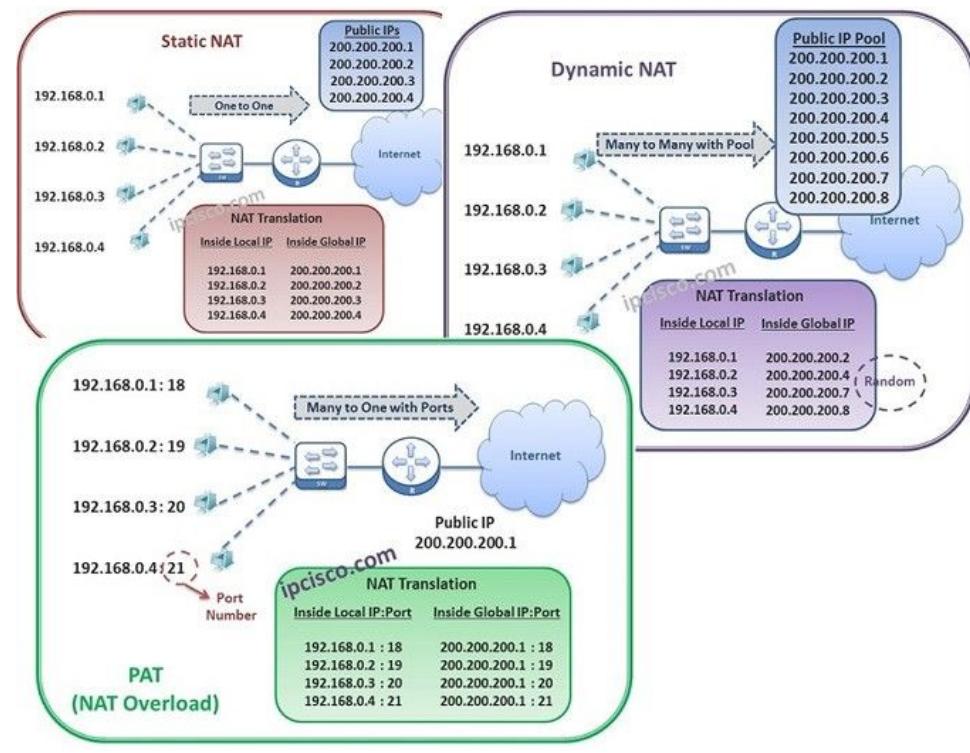
NAT(Network Address Translation)

NAT: Network Address Translation

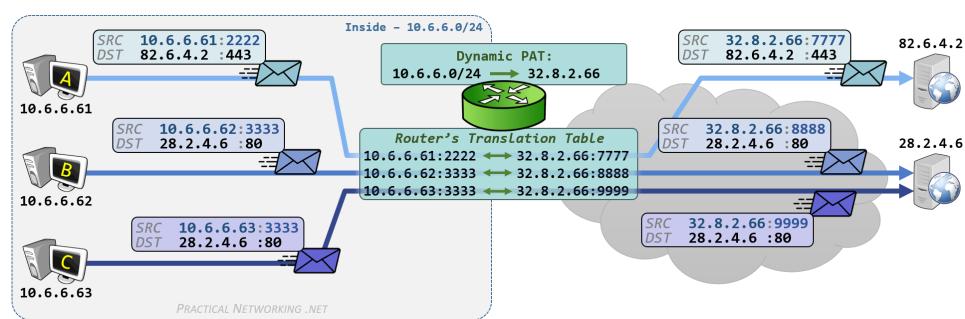
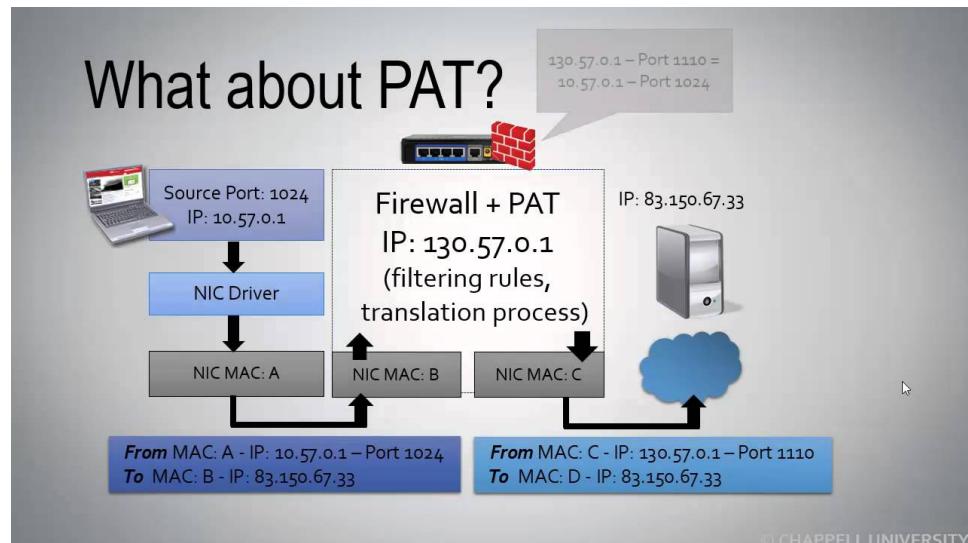


NAT: Network Address Translation





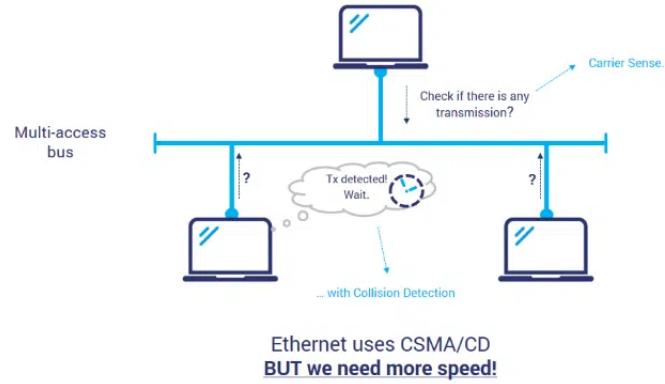
PAT (Port Address Translation)



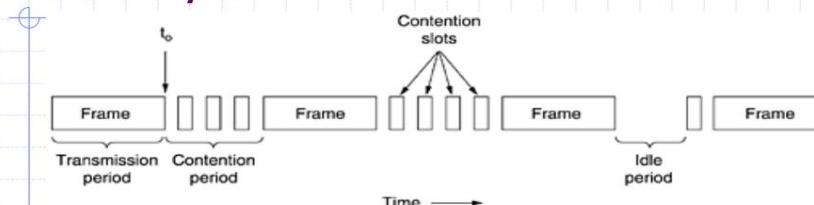
CSMA/CD

CSMA/CD

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

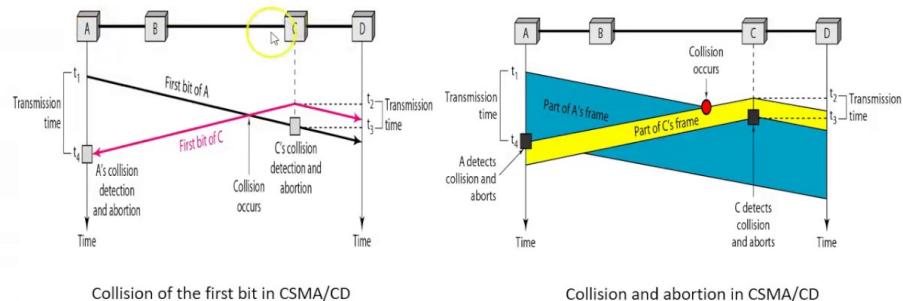


CSMA/CD



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

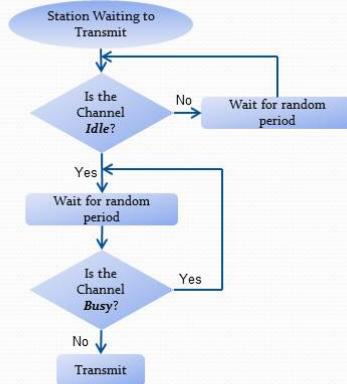
Collision in CSMA /CD



CSMA/CA

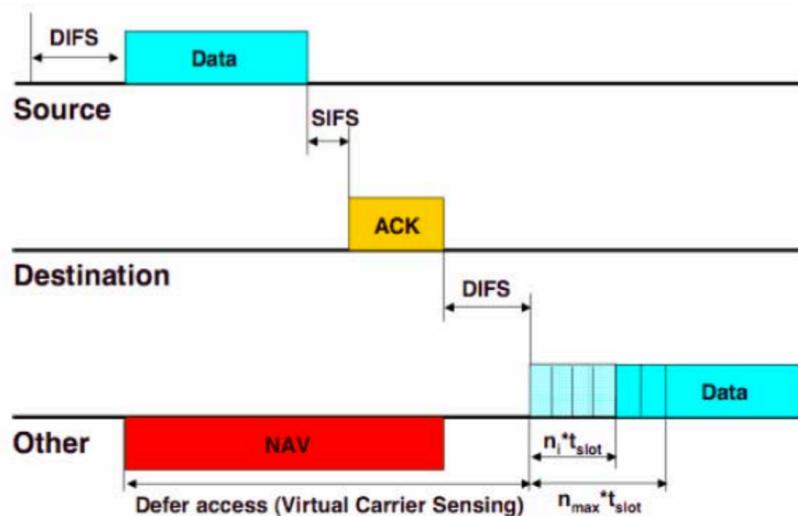
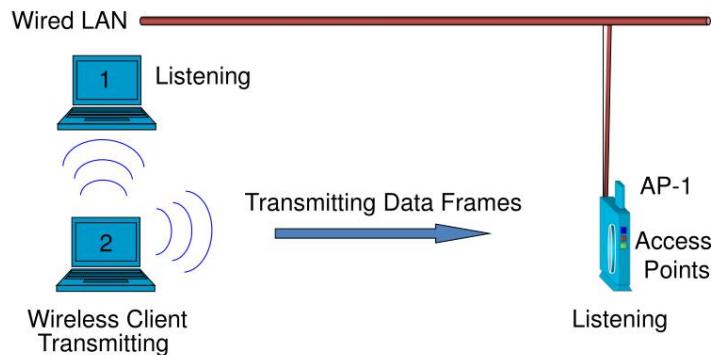
CSMA/CA

- 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 explained later in the presentation

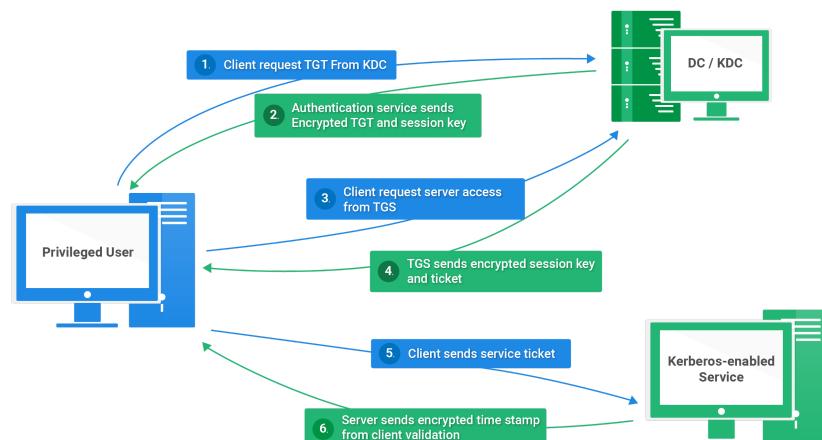
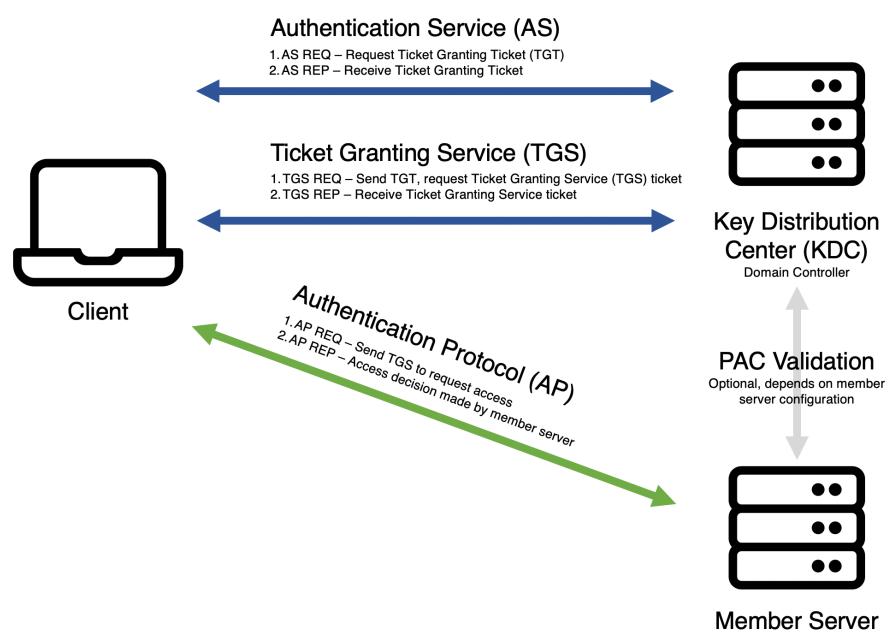


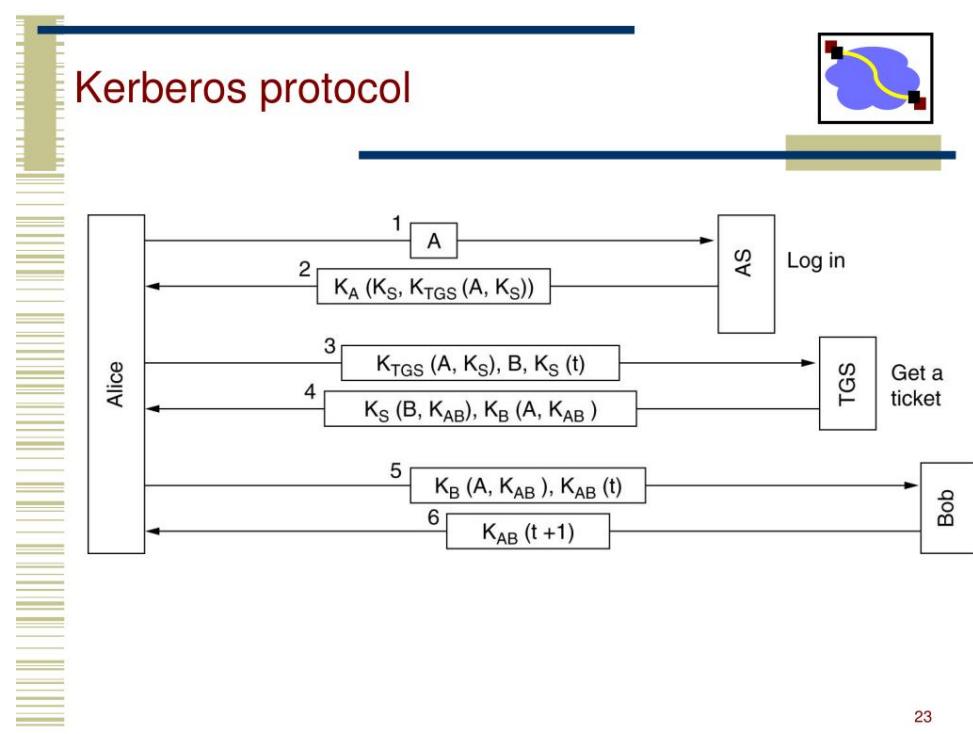
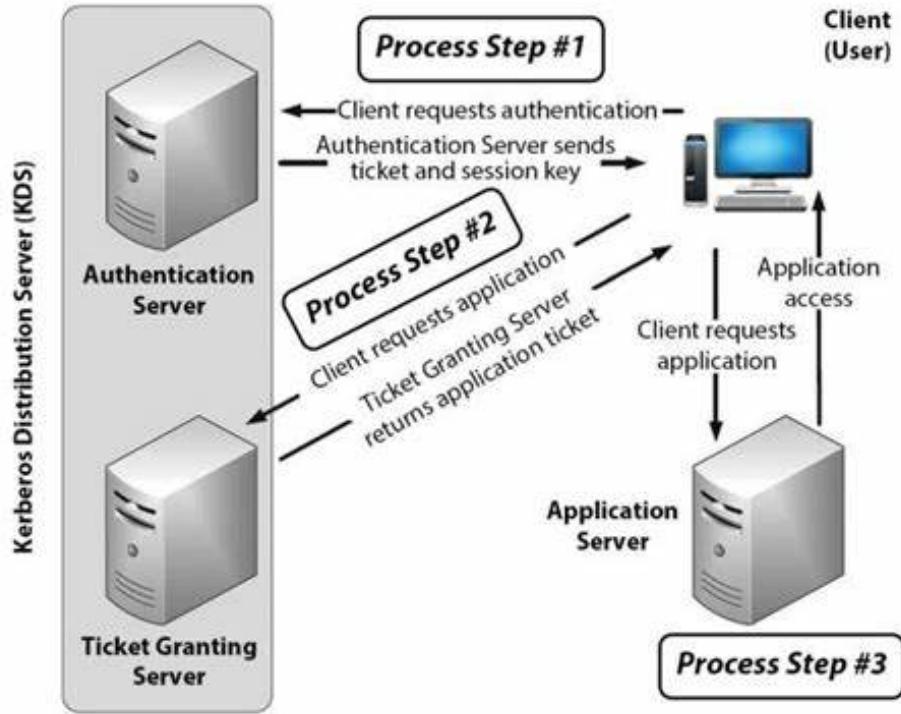
CSMA/CA Collision Handling

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



Kerberos

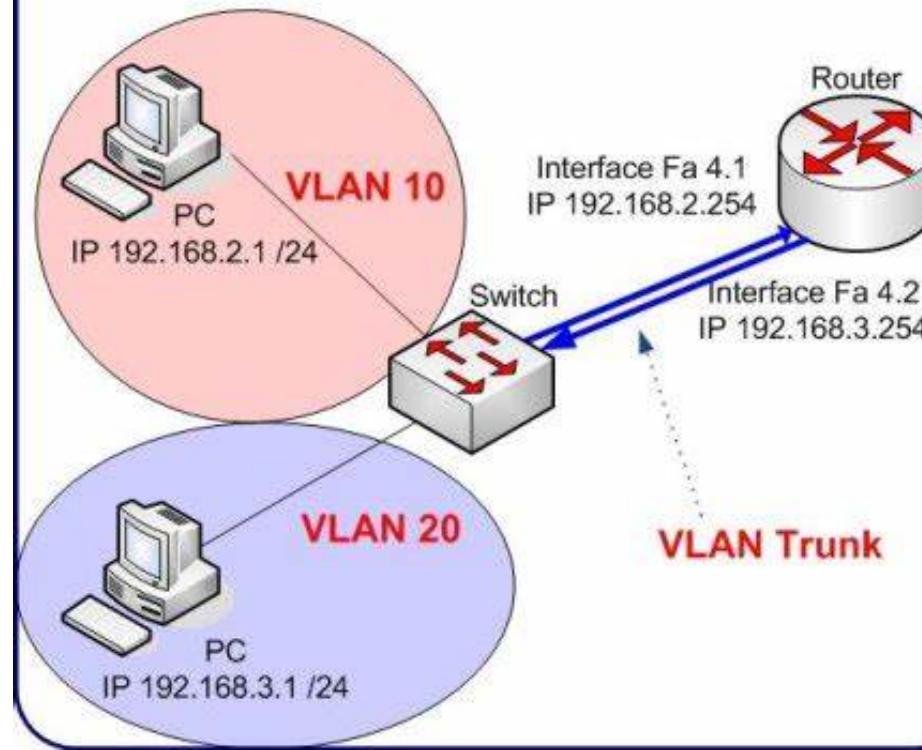




23

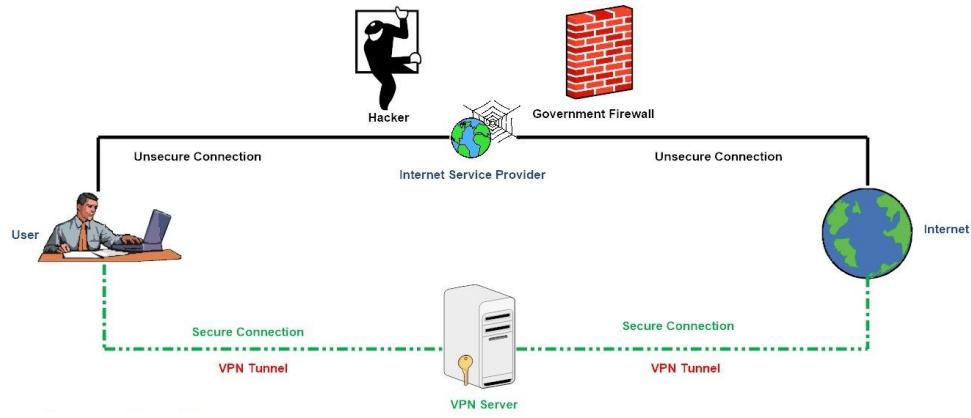
VLAN

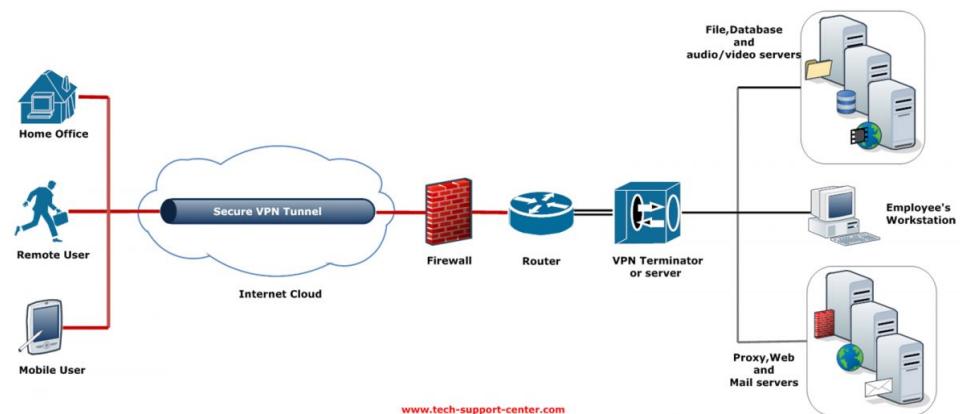
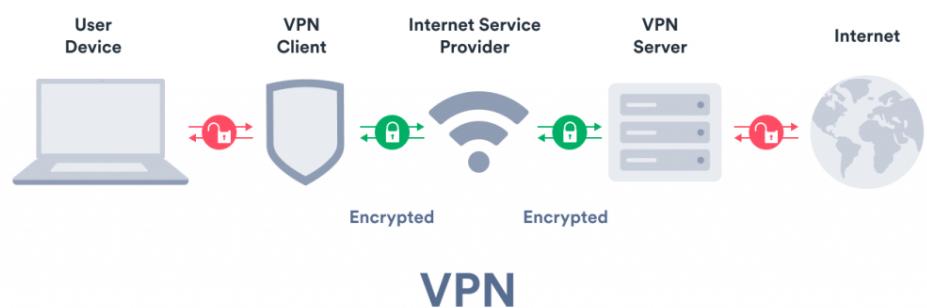
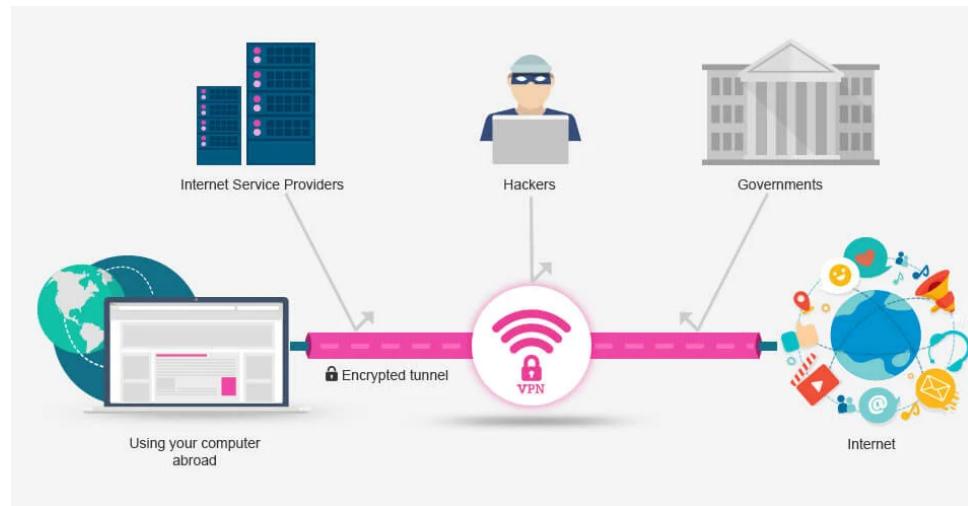
Sample network using VLANs



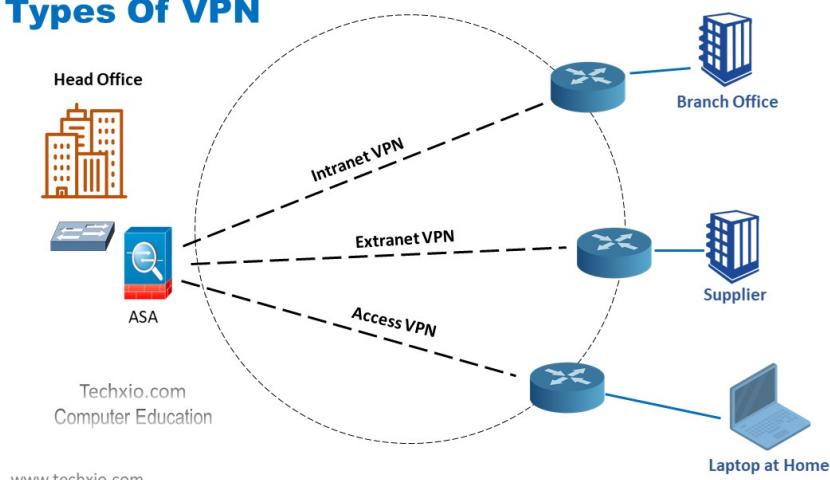
VPN

How VPN Works

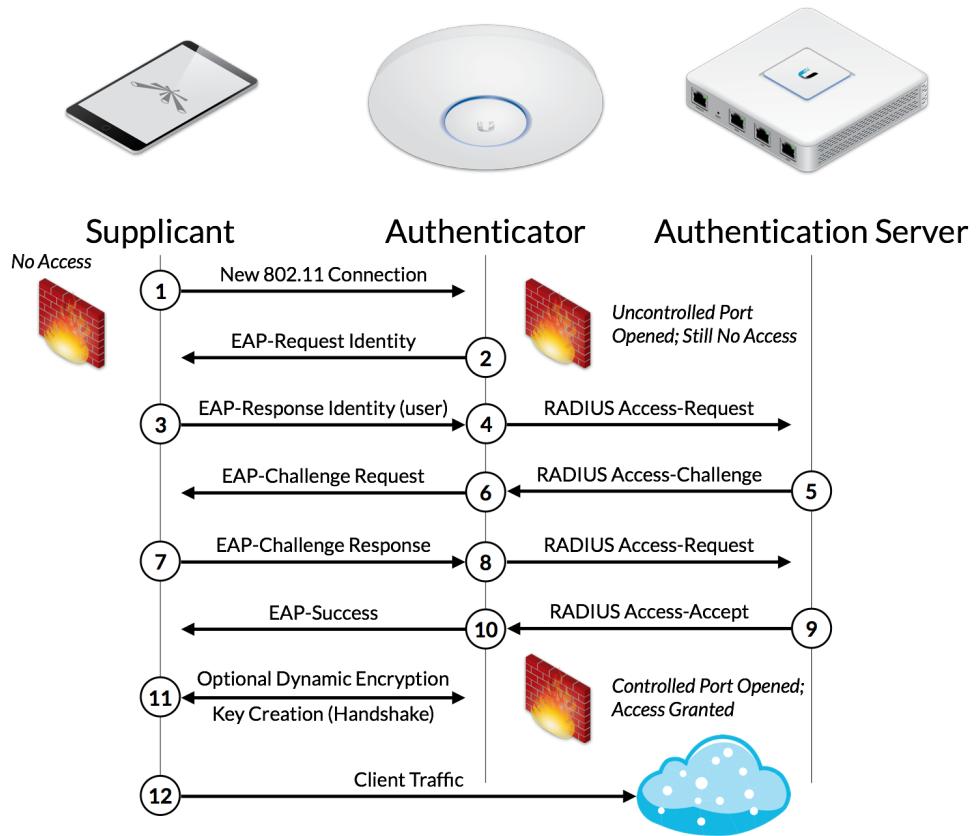




Types Of VPN

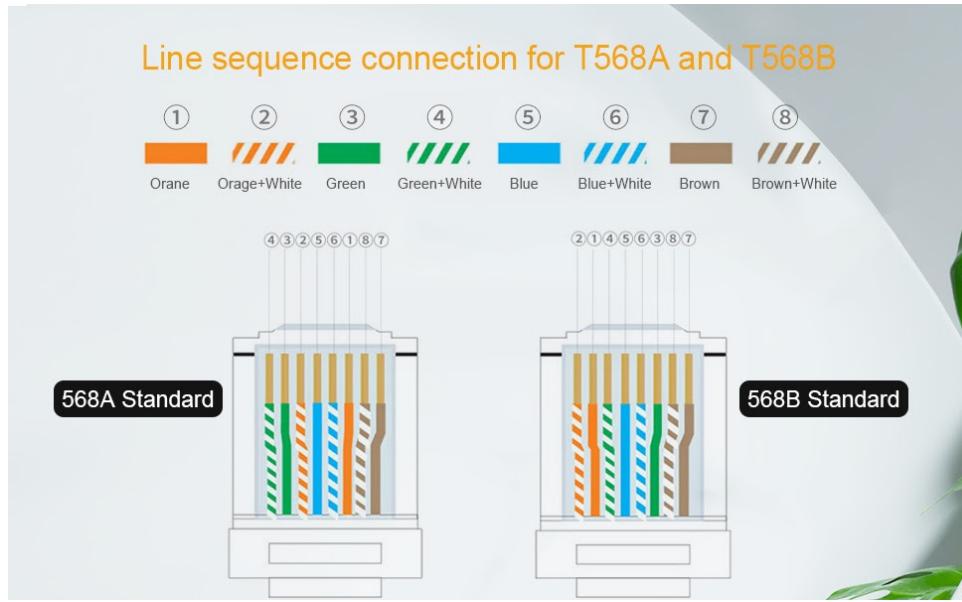


802.1X Authentication (EAP & RADIUS)

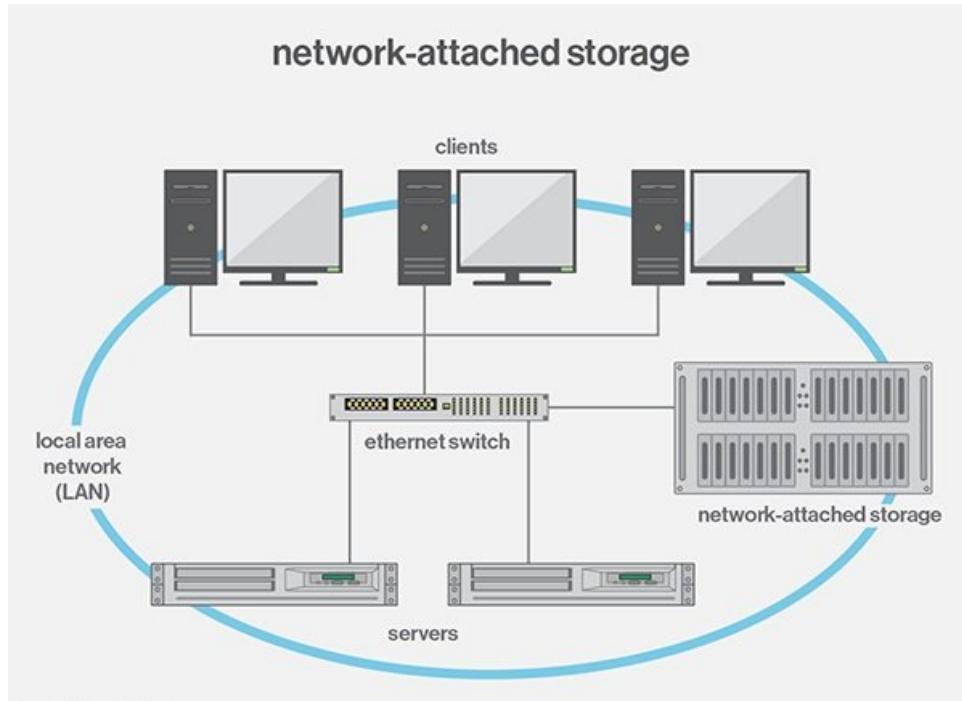


Cable

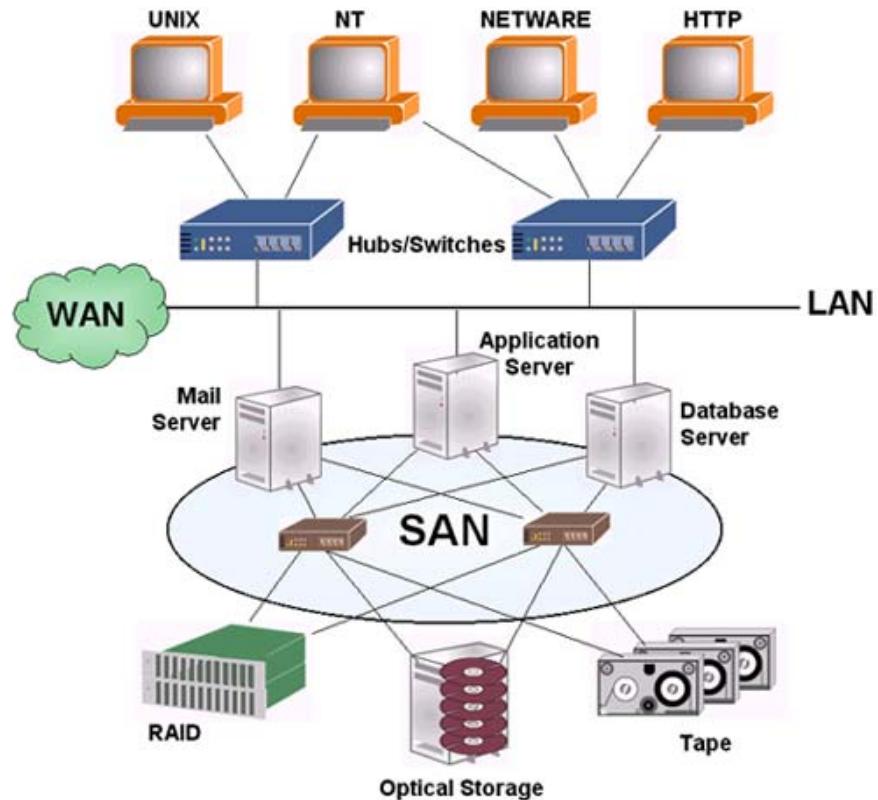
Category	Standard Bandwidth	Max Data Rate	Shielding
Cat5e	100MHz (up to 350)	1000Mbps	UTP or STP
Cat6	250MHz (up to 550)	1000Mbps	UTP or STP
Cat6A	500MHz (up to 550)	10Gbps	UTP or STP
Cat7	600MHz	10Gbps	Shielded only
Cat8	2000MHz	25Gbps or 40Gbps	Shielded only



NAS vs SAN



① Storage Area Networks



Source: allSAN Report 2001

Copyright © 2000 allSAN.com Inc 

SAN components

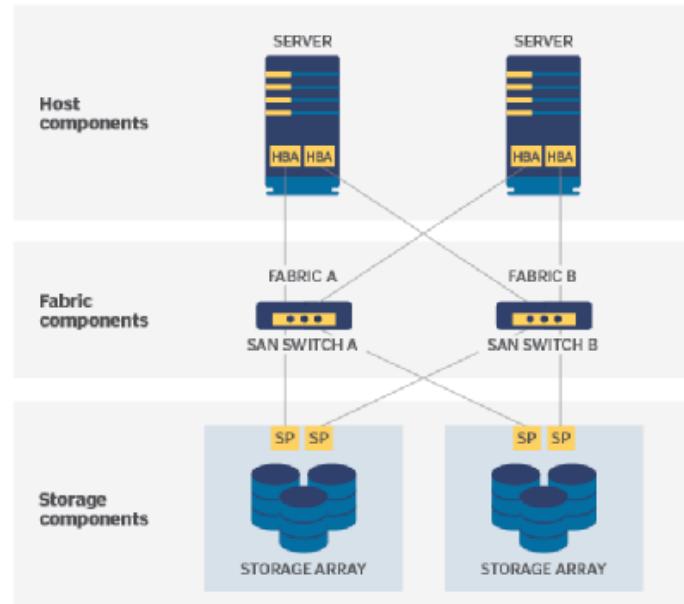
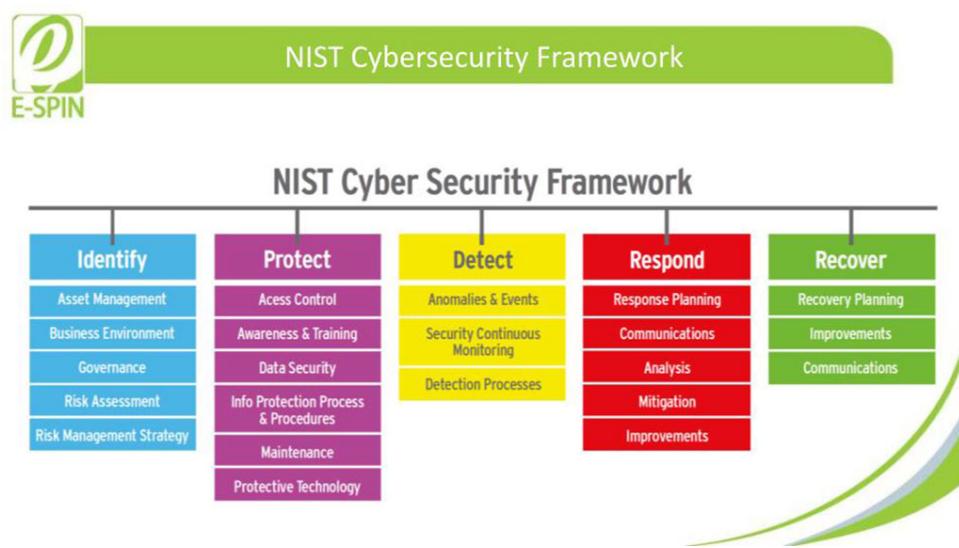
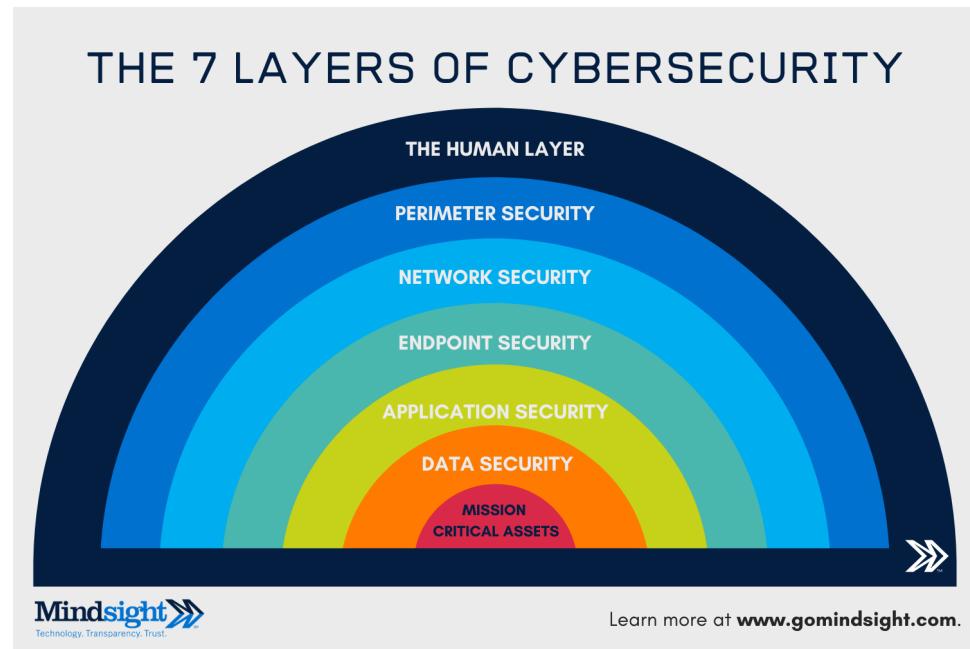


ILLUSTRATION: MAGLARA/ANDOIE STOCK

©2006 TECHTARGET. ALL RIGHTS RESERVED  TechTarget

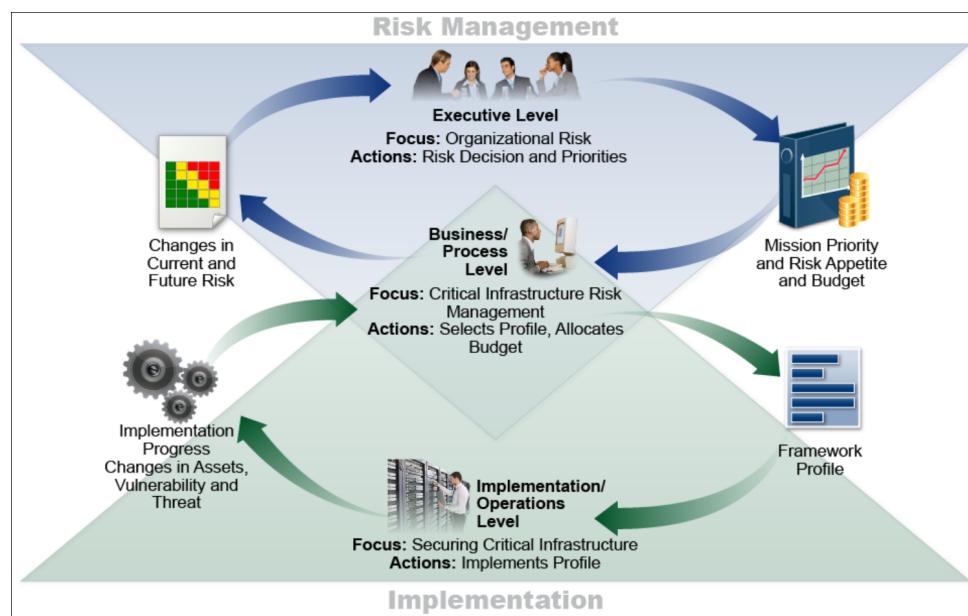
CyberSecurity

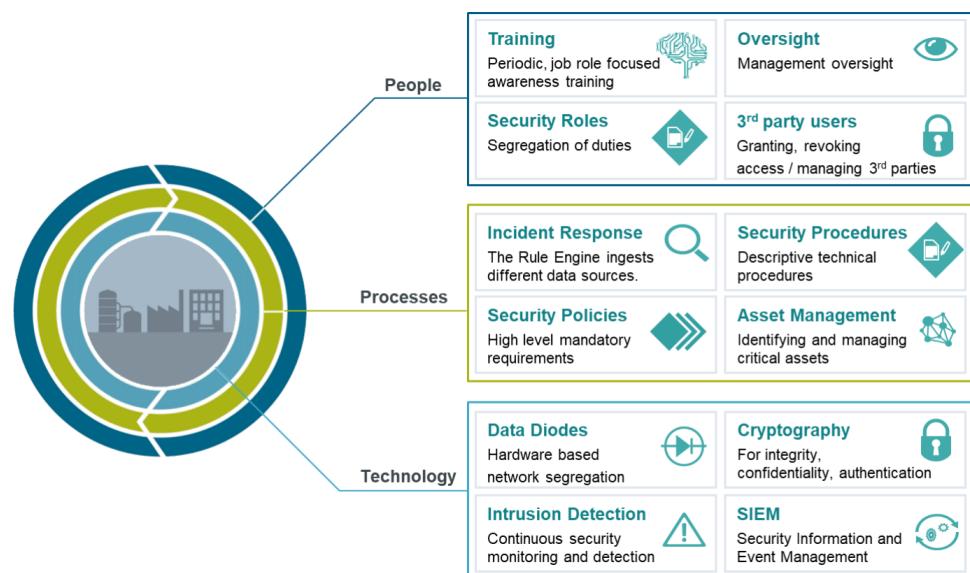
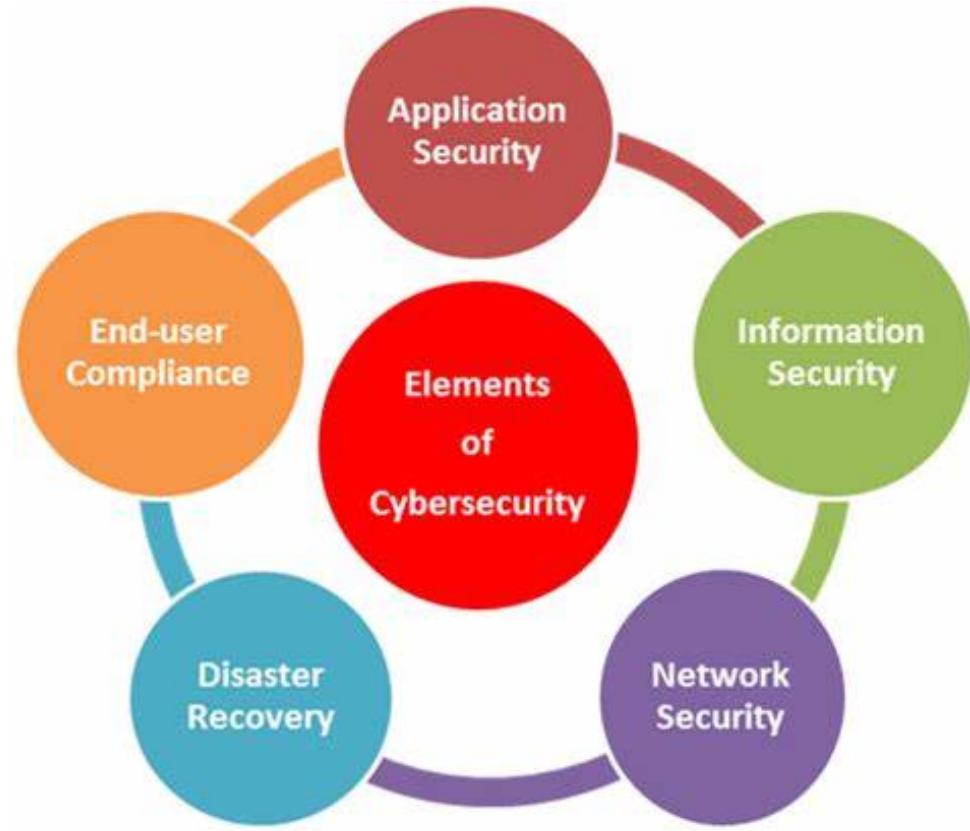




Risk Management

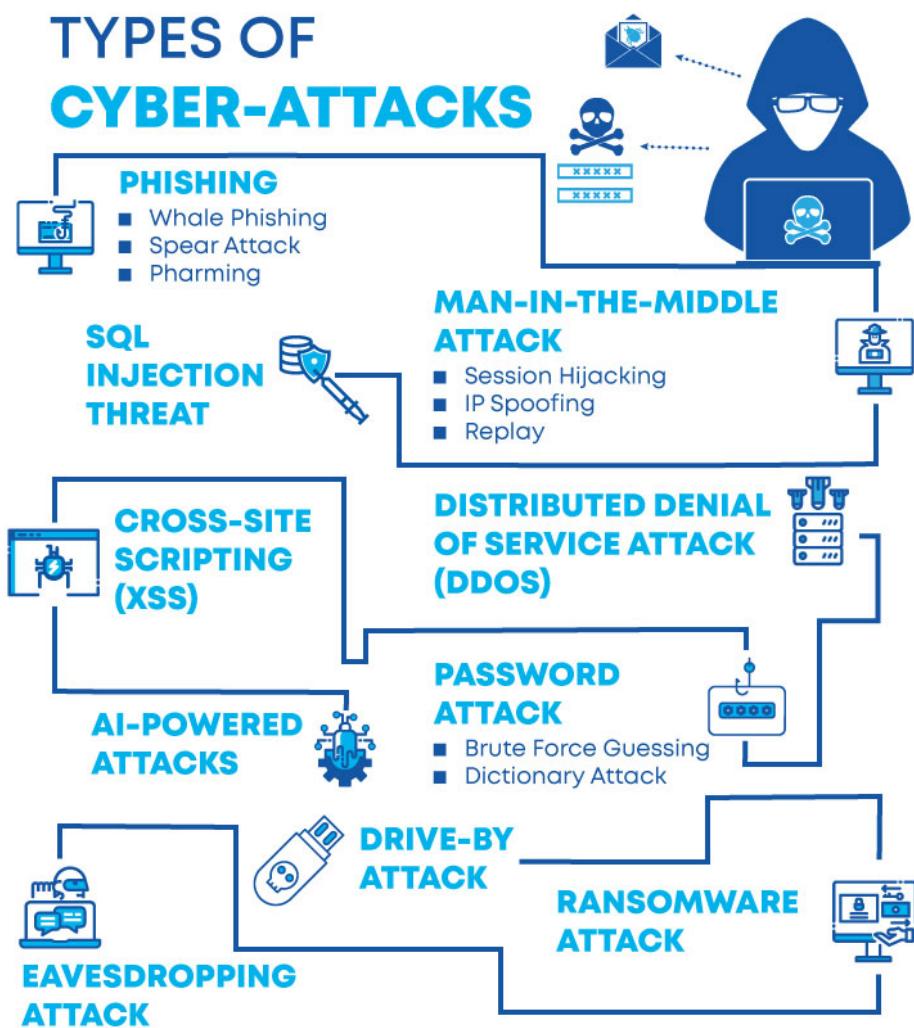
BEST CYBERSECURITY PRACTICES

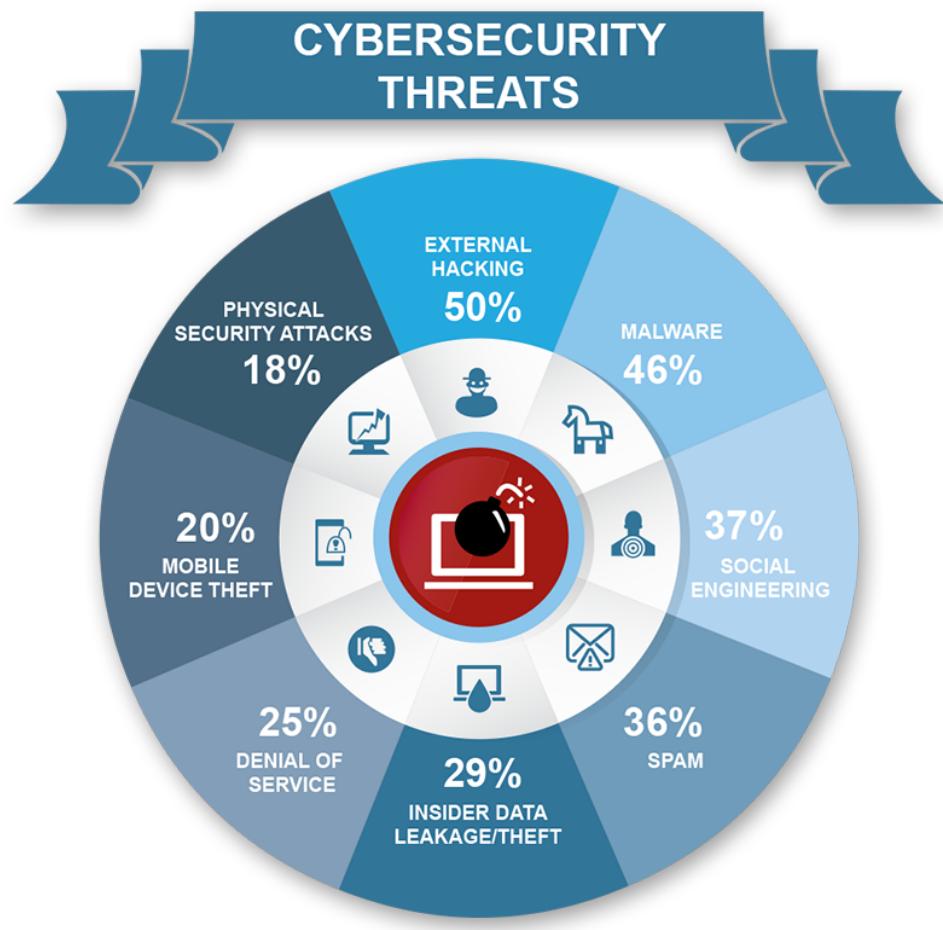




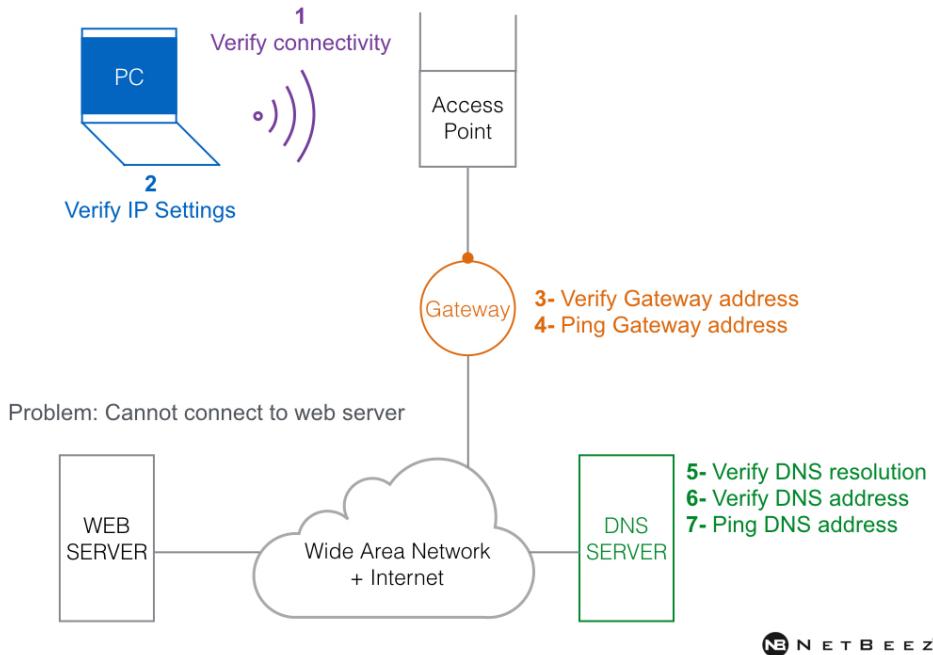


Cybersecurity Attacks

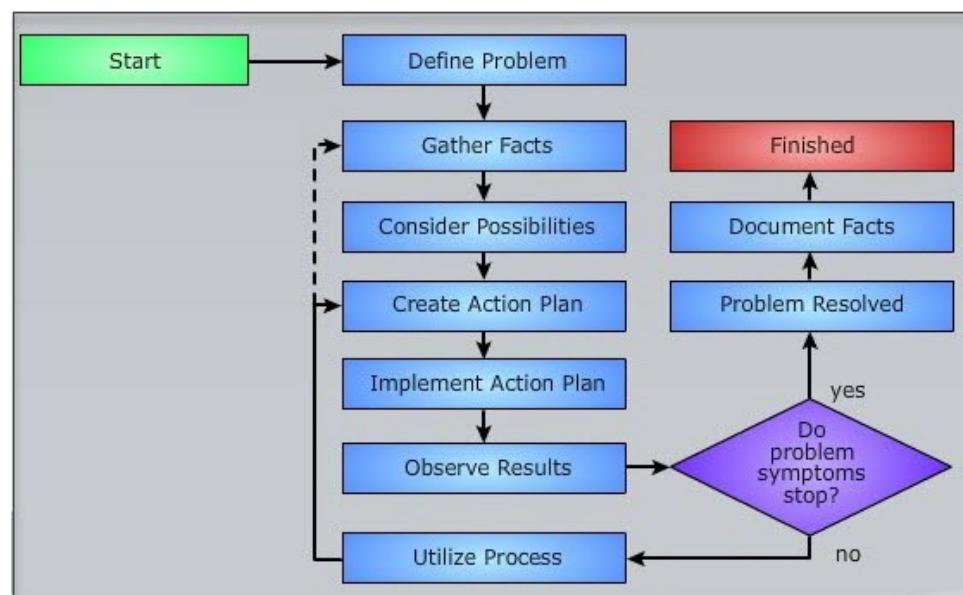
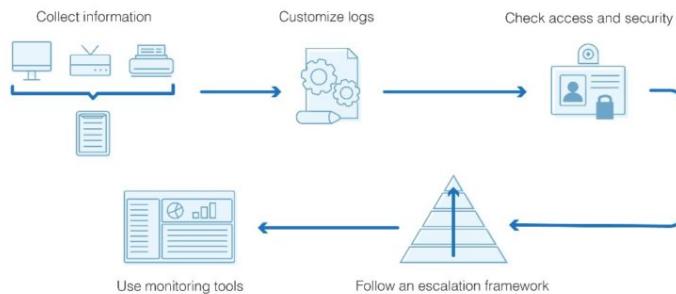




Network Troubleshooting



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

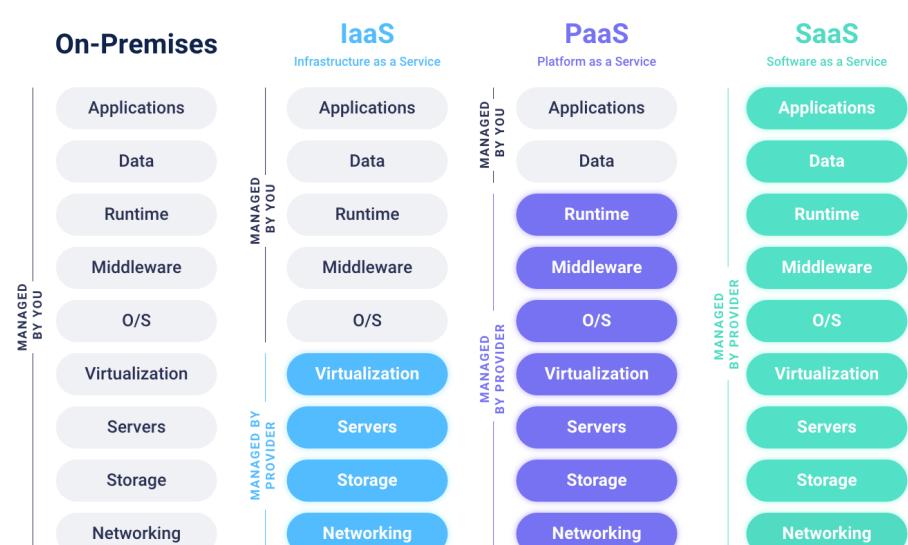
Icon of a computer monitor with a network cable.

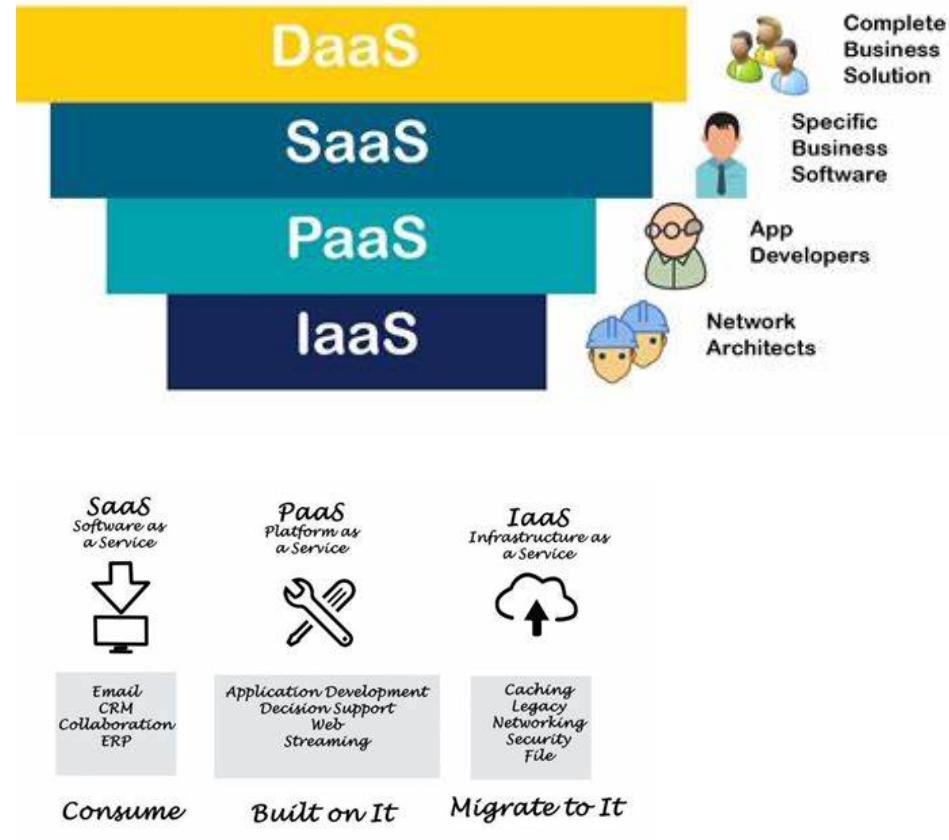
8. Document the solution and process.

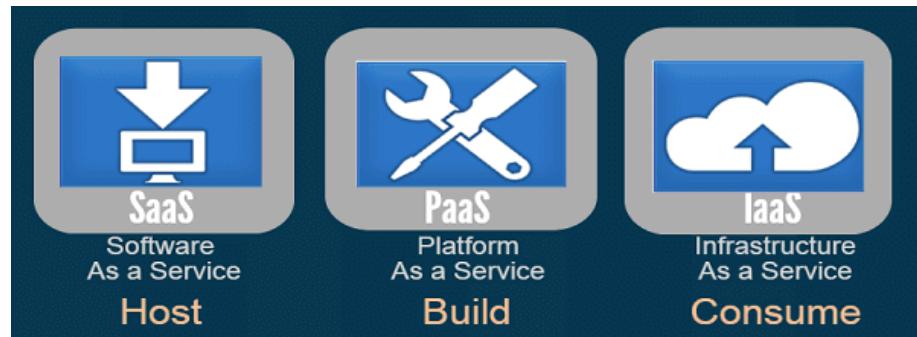
- ✓ Document the problem.
- ✓ Document what caused the problem.
- ✓ Document how the problem was fixed.

Network Administrator

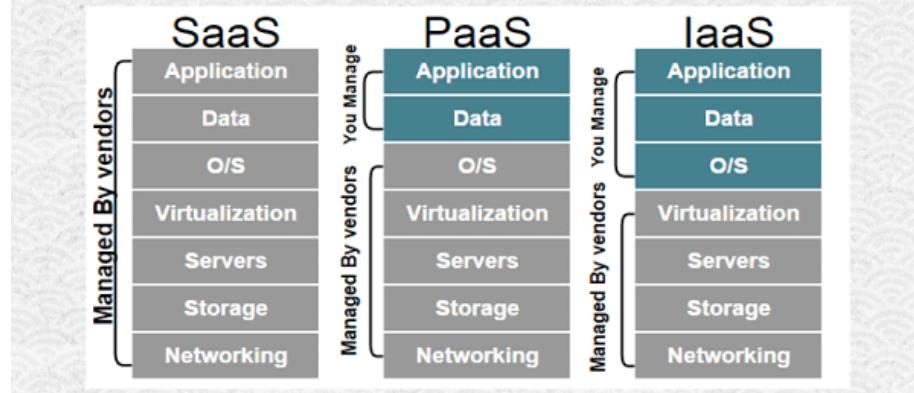
Cloud Computing - IaaS Paas Saas



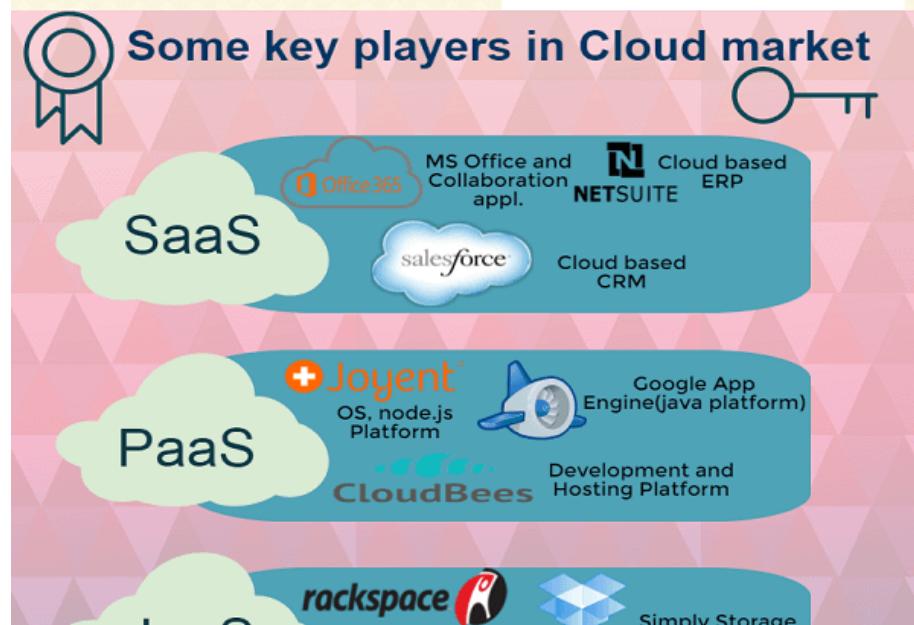
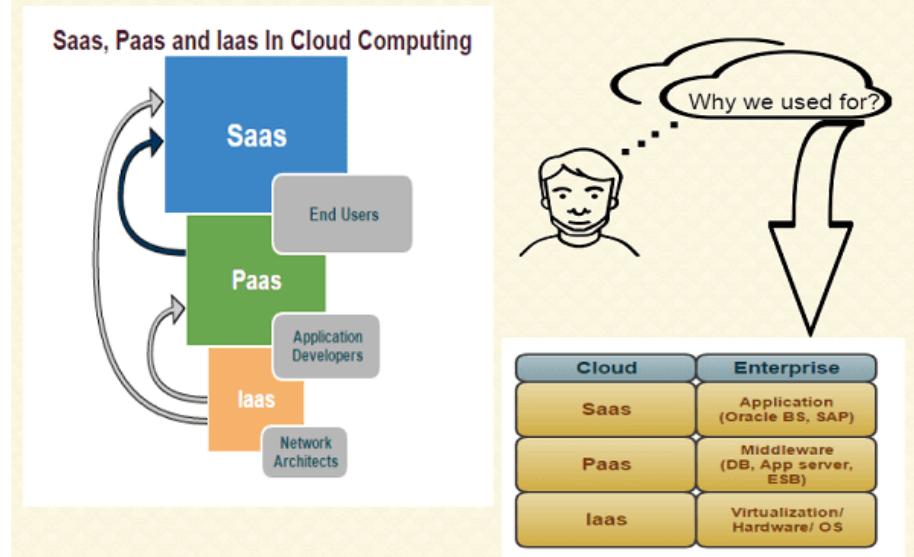




Difference between SaaS, PaaS and IaaS



How Structured in Cloud Computing?





-- Memo End --

