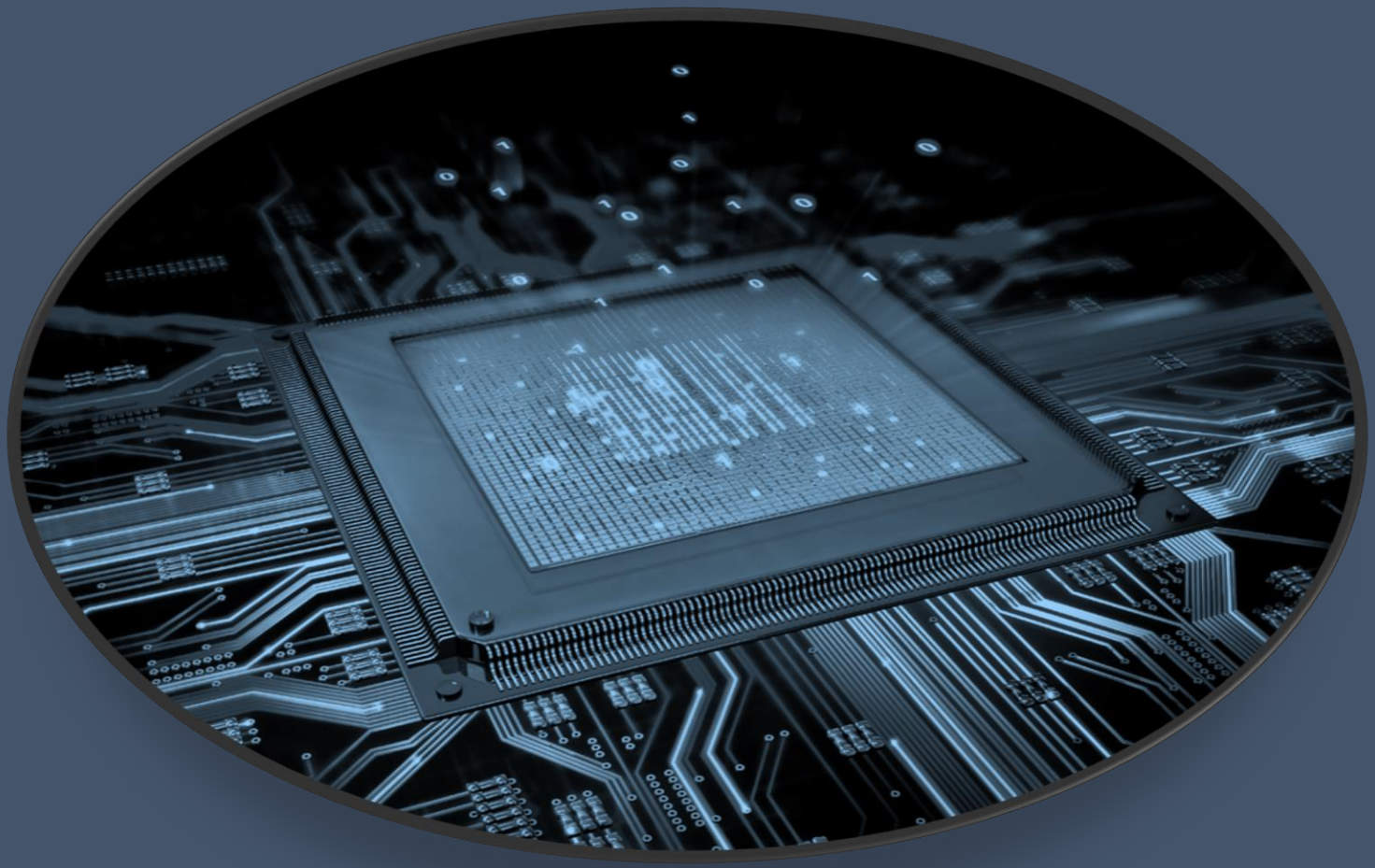
COMP10081

Foundations of Computing and Technology

Technical Strand



Matthew Robinson

CONTENTS

The Digital Logic Level

Introduction

Gates & Boolean Logic

Boolean Algebra

Number Systems

Computer Technology

The CPU

The CPU – Hardware View

The CPU – Microarchitecture View

Buses and I/O

Memory

File Systems

Operating Systems

Networks

Networking Concepts

CONTENTS

The Digital Logic Level

Introduction

Overview

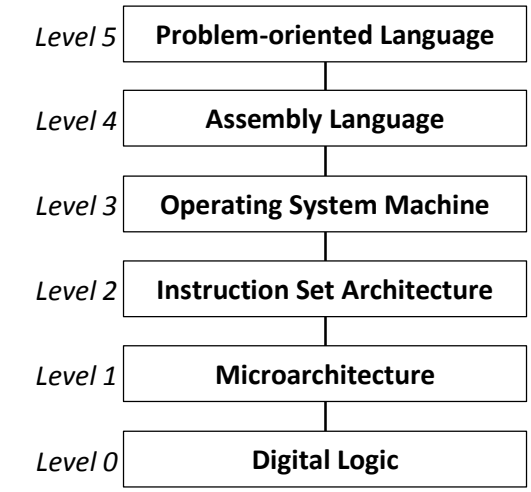
Definitions

A **digital computer** is a machine that can perform work for users by carrying out instructions provided in the form of a program.

A **program** is a sequence of instructions describing how to perform a certain task.

Levels

Diagram: Multi-Level Machine Abstraction



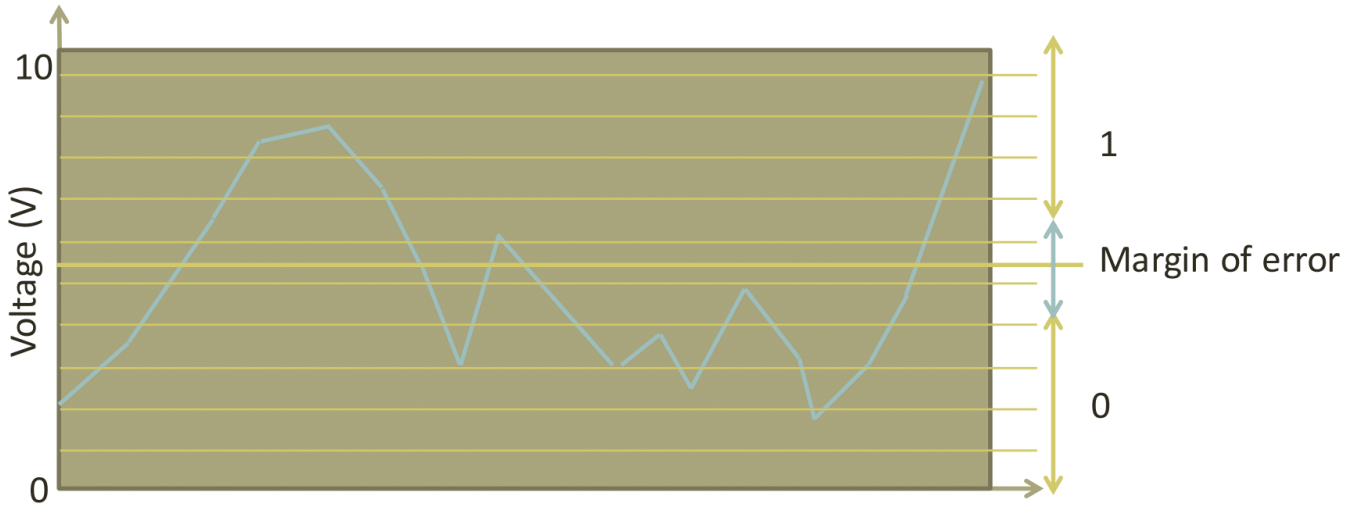
Introduction

Data representation in the CPU

Voltage

A computer system is a digital device. Computer systems can understand signals, of +5V or 0V, sent along a wire; these are represented by humans as a 1 or 0 respectively.

Diagram: Voltage in the CPU



This voltage range is divided into two halves, including a margin of error:

- any signal above the halfway point (>3v) is a 1; and
- any signal below the halfway point (<2v) is a 0.

The intermediate portion between 2v and 3v acts as the margin of error for fluctuations in the voltage.

Machine code or **machine language** is a computer programming language consisting of binary or hexadecimal instructions which a computer can respond to directly. Programs must be converted to machine code before execution. The electronic circuits of each computer can recognize and directly execute this machine code.

Transistors

A **transistor** is an electronic device that works by controlling the flow of the electrical current.

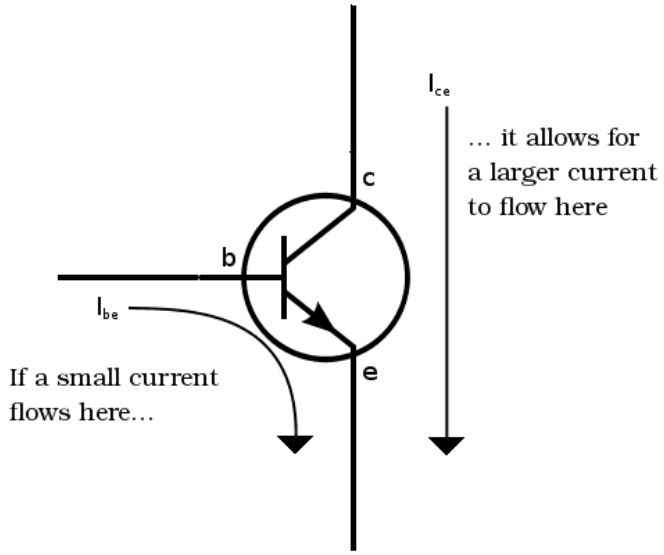
Transistors are used to build logic circuits and many logic circuits are used to make integrated circuit boards.

Transistors are effectively digital switches which have 3 regions of operation:

- off;
- on (active) – linear response; and
- on (saturated) – non-linear response.

Introduction

Diagram: Transistor Circuit



Introduction

Terminology

Term	Meaning
Bit	A single 1 or 0 value.
Bits	Multiple 1 or 0 values.
Nibble	4 bits.
Byte	8 bits.
Word	The number of bits a CPU can progress at a given time.

Gates & Boolean Logic

Boolean Logic and Logic Gates

Boolean Logic

Definitions

Boolean algebra is a set of rules to describe certain propositions whose outcome could be either True or False

Postulates

Boolean Postulates
$X = 0 \; OR \; X = 1$
$0 \; AND \; 0 = 0$
$1 \; OR \; 1 = 1$
$0 \; OR \; 0 = 0$
$1 \; AND \; 1 = 1$
$1 \; AND \; 0 = 0$
$1 \; OR \; 0 = 1 \quad , \quad 0 \; OR \; 1 = 0$

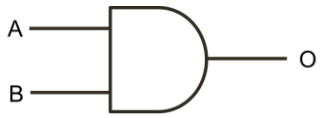
Logic Gates

Definitions

Logic gates perform basic logical functions and are the fundamental building blocks of digital integrated circuits.

AND Gate

Notation	Circuit	Truth Table			Code Representation
		A	B	O	Python

$A \,.\, B$		0	0	0	if A and B:
					  print("true")
					else:
					  print("false")

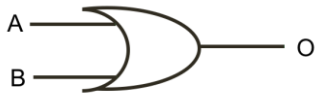
		0	1	0	
		1	0	0	
		1	1	1	
Explanation					Java / C++

An output is True if:
all inputs are True.
					if(A && B) {
					  // true
					} else {
					  // false
					}

Gates & Boolean Logic

OR Gate

Notation	Circuit	Truth Table			Code Representation
		A	B	O	Python

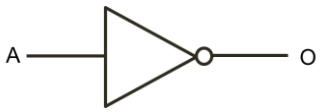
$A + B$		0	0	0	if A or B:
					  print("true")
					else:
					  print("false")

		0	1	1	
		1	0	1	
		1	1	0	
Explanation					**Java / C++**

An output is True if:
at least one input is True.
					if(A || B) {
					  // true
					} else {
					  // false
					}

NOT Gate

Notation	Circuit	Truth Table		Code Representation
		A	O	Python

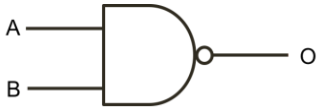
$\bar{A}$		0	1	if not A:
				 …
				else:
				  …

		1	0	
Explanation				**Java / C++**

An output is True if:
the input is False.
				if(!A) {
				 …
				} else {
				  …
				}

NAND Gate (NOT AND)

Notation	Circuit	Truth Table			Code Representation
		A	B	O	Python

$\overline{(A \,.\, B)}$		0	0	1	if not (A and B):
					  print("true")
					else:
					  print("false")

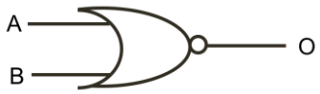
		0	1	1	
		1	0	1	
		1	1	0	
Explanation					**Java / C++**

An output is True if:
not all inputs are True.
					if(! (A && B)) {
					  // true
					} else {
					  // false
					}

Gates & Boolean Logic

NOR Gate (NOT OR)

Notation	Circuit	Truth Table			Code Representation
		A	B	O	Python

$\overline{(A+B)}$		0	0	1	if not (A or B):
					  print("true")
					else:
					  print("false")

		0	1	0	
		1	0	0	
		1	1	0	
Explanation					**Java / C++**

An output is True if:
no inputs are True.
					if(! (A || B)) {
					  // true
					} else {
					  // false
					}

XOR Gate (Exclusive OR)

Notation	Circuit	Truth Table			Code Representation
		A	B	O	Python

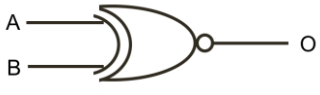
$A \oplus B$		0	0	0	if A != B:
					  print("true")
					else:
					  print("false")

		0	1	1	
		1	0	1	
		1	1	0	
Explanation					**Java / C++**

An output is True if:
only one input is True.
					if(A != B) {
					  // true
					} else {
					  // false
					}

XNOR Gate (Exclusive NOT OR)

Notation	Circuit	Truth Table			Code Representation
		A	B	O	Python

$\overline{A \oplus B}$		0	0	1	if not (A and B):
					  print("true")
					else:
					  print("false")

		0	1	1	
		1	0	1	
		1	1	0	
Explanation					**Java / C++**

An output is True if:
all inputs are True or all inputs are False.
					if(! (A && B)) {
					  // true
					} else {
					  // false
					}

When building circuits or evaluating Boolean expressions, the following order of precedence should be observed:

1) NOT;
2) AND; then
3) OR.

Gates & Boolean Logic

Truth Tables

Definitions

A **truth table** is a mathematical table used to compute the functional values of logical expressions on each of their functional arguments.

Truth tables are composed of one column for each input variable and one final column for all of the possible results of the logical operation.

Deriving Boolean Logic

Using Truth Tables to derive Boolean Expressions and Logic Circuits

Process: Deriving Boolean Expression	
Deriving Boolean Expressions	**Re-writing Using Boolean Notation**
1) Inspect each input where the output is True (1). 2) Construct a Boolean expression for the inputs on that row. 3) Repeat stages 1) and 2) for each row and separate the constructed Boolean expressions with ORs.	1) If the variable evaluates to 0, re-write the variable using the NOT notation. 2) If the variable evaluates to 1, do not modify the variable. 3) Replace other logic with the appropriate notation.

Example: Deriving Boolean Expression

Truth Table				Derived Boolean Expression	Boolean Notation Expression
A	**B**	**C**	**O**		
0	0	0	0	$O = A = 0\ AND\ B = 0\ AND\ C = 1\ OR\ A = 0\ AND\ B = 1\ AND\ C = 0\ OR\ A = 1\ AND\ B = 0\ AND\ C = 0\ OR\ A = 1\ AND\ B = 1\ AND\ C = 1$	$O = \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + A.B.C$
0	0	1	1		
0	1	0	1		
0	1	1	0		
1	0	0	1		
1	0	1	0		
1	1	0	0		
1	1	1	1		

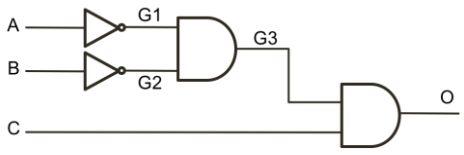
Gates & Boolean Logic

Using Logic Circuit Diagrams to derive Truth Tables and Boolean Expressions

Process: Deriving Truth Table and Boolean Expression		
Deriving Truth Table	**Deriving Boolean Expression**	**Re-writing Using Boolean Notation**
1) Mark points on the diagram at output end of each logic gate. (e.g. G1, G2, G3 etc.) 2) Construct a truth table with columns for the inputs and the points marked on the diagram. 3) Populate the input columns with the possible permutations of input values (0 and 1). 4) Calculate the result of the input to each of the logic circuits and write the result in the columns for the points on the diagram – do this for each row. 5) Calculate the output based on the values calculated.	1) Inspect each input where the output is True (1). 2) Construct a Boolean expression for the inputs on that row. 3) Repeat stages 1) and 2) for each row and separate the constructed Boolean expressions with ORs.	1) If the variable evaluates to 0, re-write the variable using the NOT notation. 2) If the variable evaluates to 1, do not modify the variable. 3) Replace other logic with the appropriate notation.

Example: Deriving Truth Table and Boolean Expression

Logic Circuit Diagram



Truth Table

A	B	C	G1	G2	G3	O
0	0	0	1	1	1	0
0	0	1	1	1	1	1
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0

Derived Boolean Expression

$$O = A = 0\ AND\ B = 0\ AND\ C = 1$$

Boolean Notation Expression

$$O = \bar{A} \,.\, \bar{B} \,.\, C$$

Calculating Values

Example: Calculating Values in Boolean Expressions

Calculate the value of F in the Boolean expression $F = \overline{(X \,.\, Y + X \,.\, Z)}$, assuming $X = 1$, $Y = 0$, and $Z = 0$.

$F = \overline{(1 \,.\, 0 + 1 \,.\, 0)}$	Substitute in values
$F = \overline{(0 + 0)}$	$1 \,.\, 0 = 0$
$F = \overline{(0)}$	$0 + 0 = 0$
$\boldsymbol{F = 1}$	$\overline{(0)} = 1$

Equivalent Circuits

Definitions

Equivalent circuits are ones where several statements have logical equivalence. This is a type of relationship where several circuits may be logically equivalent, in that they all have identical truth tables.

Example

Example: Two Equivalent Circuits										
$X = A + B\,.\,C$					$Y = (A + B)\,.\,(A + C)$					
A	**B**	**C**	**G1** (B . C)	**X**	**A**	**B**	**C**	**G1** (A + B)	**G2** (A + C)	**Y**
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	1	0
0	1	0	0	0	0	1	0	1	0	0
0	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	0	0	1	1	1
1	0	1	0	1	1	0	1	1	1	1
1	1	0	0	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1

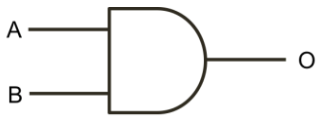
By abstracting the circuits to the inputs (A, B and C) and the output (X), it is shown that the circuits are equivalent.

NAND Gates

NAND gates are “universal gates”, meaning that they can be used to construct all other logic gates – this method is used to construct logic gates in most modern CPUs. This method of constructing logic gates is used because it is cheaper to construct CPUs using only one gate rather than many.

NAND Construction of AND Gate

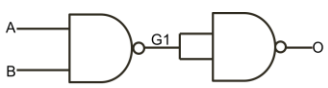
AND – Circuit



AND – Truth Table

A	B	O
0	0	0
0	1	0
1	0	0
1	1	1

NAND – Circuit



NAND – Truth Table

A	B	G1	O
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

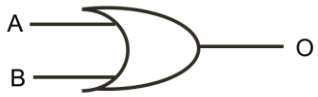
Boolean Expression

$$A\,.\,B = \overline{\overline{A}\,.\,\overline{B}}$$

Boolean Algebra

NAND Construction of OR Gate

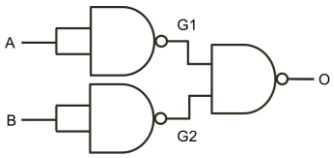
OR – Circuit



OR – Truth Table

A	B	O
0	0	0
0	1	1
1	0	1
1	1	1

NAND – Circuit



NAND – Truth Table

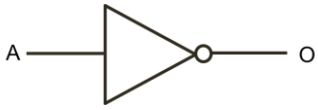
A	B	G1	G2	O
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

Boolean Expression

$$A + B = \overline{\overline{(A + B)}} = \overline{(\bar{A} \,.\, \bar{B})}$$

NAND Construction of NOT Gate

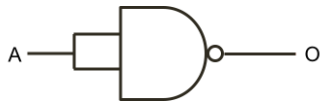
NOT – Circuit



NOT – Truth Table

A	O
0	1
1	0

NAND – Circuit



NAND – Truth Table

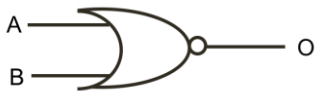
A	A	O
0	0	1
0	1	1
1	0	1
1	1	0

Boolean Expression

$$\bar{A} = \overline{(A \,.\, A)}$$

NAND Construction of NOR Gate (NOT OR)

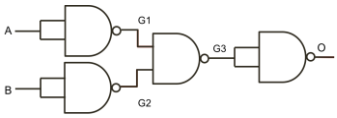
NOR – Circuit



NOR – Truth Table

A	B	O
0	0	1
0	1	0
1	0	0
1	1	0

NAND – Circuit



NAND – Truth Table

A	B	G1	G2	G3	O
0	0	1	1	0	1
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	1	0

Boolean Expression

$$\overline{(A + B)} = \overline{\overline{(A + B)}} = \overline{(\bar{A} \,.\, \bar{B})}$$

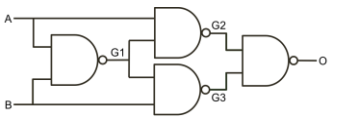
NAND Construction of XOR Gate (Exclusive OR)

XOR – Circuit

XOR – Truth Table

A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

NAND – Circuit



NAND – Truth Table

A	B	G1	G2	G3	O
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

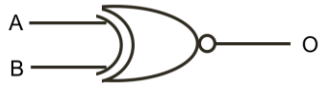
Boolean Expression

$$A \oplus \mathrm{B} = \overline{((A \,.\, \overline{(A \,.\, B)}) \,.\, \overline{(B \,.\, \overline{(A \,.\, B)})})} = A \,.\, \bar{B} + \bar{A} \,.\, B$$

Boolean Algebra

NAND Construction of XNOR Gate (Exclusive NOT OR)

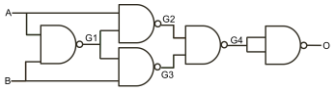
XNOR – Circuit



XNOR – Truth Table

A	B	O
0	0	1
0	1	0
1	0	0
1	1	1

NAND – Circuit



NAND – Truth Table

A	B	G1	G2	G3	G4	O
0	0	1	1	1	0	1
0	1	1	1	0	1	0
1	0	1	0	1	1	0
1	1	0	1	1	0	1

Boolean Expression

$$\overline{A \oplus B} = \overline{((A \,.\, \overline{(A \,.\, B)}) \,.\, (B \,.\, \overline{(A \,.\, B)}))} = \overline{A \,.\, \bar{B} + \bar{A} \,.\, B}$$

Boolean Algebra

Simplifying Boolean Algebra

Laws

Boolean Algebra Laws			
Law		**Algebra Representation**	**Explanation**
Commutative Law		$A + B = B + A$ $A \,.\, B = B \,.\, A$	The order of operations are not important as the result is identical.
Associative Law		$(A + B) + C = A + (B + C)$ $(A \,.\, B) \,.\, C = A \,.\, (B \,.\, C)$	The order of operations are not important as the result is identical.
Distributive Law		$A \,.\, (B + C) = (A \,.\, B) + (A \,.\, C)$ $A + (B \,.\, C) = (A + B) \,.\, (A + C)$	When multiplying the contents of a bracket, all terms within the bracket should be multiplied out.
Identity Laws	1	$A + A = A$ $A \,.\, A = A$	When performing an operation on the same variable, the result is the input.
	2	$A \,.\, B + A \,.\, \bar{B} = A$ $(A + B) \,.\, (A + \bar{B}) = A$	When there is a variable can be either present or omitted, the constant variable is the result.
Redundancy Laws	1	$A + A \,.\, B = A$ $A \,.\, (A + B) = A$	When there is a variable or the same variable and another variable, the second variable is not important. This works for both AND and OR operations.
	2	$1 + A = 1$ $1 \,.\, A = A$	When there is one (1) OR a variable, the result is one (1). When there is one (1) AND a variable, the result is the variable.
	3	$0 + A = A$ $0 \,.\, A = 0$	When there is zero (0) OR a variable, the result is the variable. When there is zero (0) AND a variable, the result is zero (0).
	4	$A + \bar{A} = 1$ $A \,.\, \bar{A} = 0$	When there is a variable OR the inverted variable, the result is one (1). When there is a variable AND the inverted variable, the result is zero (0).
De Morgan's Laws	1	$\overline{(A + B)} = \bar{A} \,.\, \bar{B}$	Inverting the output of any gate results in the same function is equivalent to the opposite gate with inverted inputs. *break the line, change the sign (left to right)* *make the line, change the sign (right to left)*
	2	$\overline{(A + B)} = \bar{A} + \bar{B}$	

Example

Example: Simplfying Boolean Expressions	
Simplify the Boolean expression $(\boldsymbol{X} \,.\, \boldsymbol{Y}) + (\boldsymbol{X} \,.\, \bar{\boldsymbol{Y}})$.	
$(X \,.\, Y) + (X \,.\, \bar{Y}) = X \,.\, (Y + \bar{Y})$	Distributive Law
$(X \,.\, Y) + (X \,.\, \bar{Y}) = X \,.\, (1)$	Redundancy Law 4
$(X \,.\, Y) + (X \,.\, \bar{Y}) = \boldsymbol{X}$	Redundancy Law 2

Number Systems

Number Systems

Definitions

A **number system** is used for representing numbers of a certain type.

Conversion Formula

$$value = \sum_{n=0}^{\infty} (weight_n \times digit_n)$$

Decimal Number System

Definition

The **decimal number system** is a number system that:

- uses a notation in which each number is expressed in base 10;
- uses the number range 0-9; and
- lets each place value be a power of 10.

Place Values

Decimal Place Values

Position	5	4	3	2	1
Weight *(base 10)*	10000	1000	100	10	1

Example

Example: Determining Decimal Numbers

Determine the decimal number represented below.

Position	5	4	3	2	1
Weight *(base 10)*	10000	1000	100	10	1
Decimal Number	1	2	6	8	7

$$value = \sum_{n=0}^{\infty} (weight_n \times digit_n)$$

$value = (10000 \times 1) + (1000 \times 2) + (100 \times 6) + (10 \times 8) + (7 \times 1)$
$value = 10000 + 2000 + 600 + 80 + 7$
$\boldsymbol{value = 12687}$

Binary Number System

Definition

The **binary number system** is a number system that:

- uses a notation in which each number is expressed in base 2;
- uses the number range 0-1; and
- lets each place value be a power of 2.

One binary character is represented by one (1) bit.

Place Values

Binary Place Values

Position	8	7	6	5	4	3	2	1
Weight *(base 2)*	128	64	32	16	8	4	2	1

The minimum decimal value for a binary number constructed using n bits is 0.
The maximum decimal value for a binary number constructed using n bits is calculated by $2^n - 1$.

An 8-bit binary number has:

- a minimum decimal value of 0, when the binary value is 00000000; and
- a maximum decimal value of 255, when the binary value is 11111111.

Example

Example: Determining Binary Numbers

Determine the decimal number represented below.

Position	8	7	6	5	4	3	2	1
Weight *(base 2)*	128	64	32	16	8	4	2	1
Binary Number	0	1	1	0	1	1	0	1

$$value = \sum_{n=0}^{\infty} (weight_n \times digit_n)$$
$value = (128 \times 0) + (64 \times 1) + (32 \times 1) + (16 \times 0) + (8 \times 1) + (4 \times 1) + (2 \times 0) + (1 \times 1)$
$value = 0 + 64 + 32 + 0 + 8 + 4 + 0 + 1$
$value = 64 + 32 + 8 + 4 + 1$
$\boldsymbol{value = 109}$

Number Systems

Converting Decimal to Binary

Division Method

Process: Conversion

1) Repeat until division evaluates to 0:
   - divide the decimal number by 2 (as binary is base 2); and
   - if there is a fraction left over, such as 0.5, set this as the remainder r.
2) Construct the binary number using the remainder r values in reverse order.

Example: Conversion

Convert 155_{10} to binary.

$\frac{155_{10}}{2_{10}} = 77_{10}\ (r = 1)$
$\frac{77_{10}}{2_{10}} = 38_{10}\ (r = 1)$
$\frac{38_{10}}{2_{10}} = 19_{10}\ (r = 0)$
$\frac{19_{10}}{2_{10}} = 9_{10}\ (r = 1)$
$\frac{9_{10}}{2_{10}} = 4_{10}\ (r = 1)$
$\frac{4_{10}}{2_{10}} = 2_{10}\ (r = 0)$
$\frac{2_{10}}{2_{10}} = 1_{10}\ (r = 0)$
$\frac{1_{10}}{2_{10}} = 0_{10}\ (r = 1)$

Repeat until division evaluates to 0:
- divide the decimal number by 2 (as binary is base 2); and
- if there is a fraction left over, such as 0.5, set this as the remainder r.

$\mathbf{10011011_2}$

Construct the binary number using the remainder r values in reverse order.

Subtraction Method

Process: Conversion

1) Start at the most significant bit (MSB) and repeat until the least significant bit (LSB) is reached:
   - subtract the place value from the decimal number;
   - if the result is positive, place a one (1) under that place value and perform the subtraction operation; and
   - if the result is negative, place a zero (0) under that place value and do not perform the subtraction operation.
2) Write the answer.

Number Systems

Example: Conversion

Convert 155_{10} to binary.

128	64	32	16	8	4	2	1

$155 - 128 = 27$

128	64	32	16	8	4	2	1
1							

$27 - 64 = -37$

128	64	32	16	8	4	2	1
1	0						

$27 - 32 = -5$

128	64	32	16	8	4	2	1
1	0	0					

$27 - 16 = 11$

128	64	32	16	8	4	2	1
1	0	0	1				

$11 - 8 = 3$

128	64	32	16	8	4	2	1
1	0	0	1	1			

$3 - 4 = -1$

128	64	32	16	8	4	2	1
1	0	0	1	1	0		

$3 - 2 = 1$

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	

$1 - 1 = 0$

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	1

$\mathbf{10011011_2}$

Start at the most significant bit (MSB) and repeat until the least significant bit (LSB) is reached:

- subtract the place value from the decimal number;
- if the result is positive, place a one (1) under that place value and perform the subtraction operation; and
- if the result is negative, place a zero (0) under that place value and do not perform the subtraction operation.

Write the answer.

Number Systems

Sign and Magnitude

Definitions

Sign and magnitude is a way of representing binary numbers where the most significant bit (MSB) stores the sign: 0 is positive; and 1 is negative.

Converting Decimal to Sign and Magnitude Binary

Examples: Conversion

Convert 88_{10} to a sign and magnitude binary number.

MSB	64	32	16	8	4	2	1
9	1	0	1	1	0	0	0

$$64 \times 1 = 64$$
$$32 \times 0 = 0$$
$$16 \times 1 = 16$$
$$8 \times 1 = 8$$
$$4 \times 0 = 0$$
$$2 \times 0 = 0$$
$$1 \times 0 = 0$$
$$64 + 16 + 8 = 88$$

MSB is zero, so the number is positive.
Therefore, the number being represented is 88.

Convert -88_{10} to a sign and magnitude binary number.

MSB	64	32	16	8	4	2	1
1	1	0	1	1	0	0	0

$$64 \times 1 = 64$$
$$32 \times 0 = 0$$
$$16 \times 1 = 16$$
$$8 \times 1 = 8$$
$$4 \times 0 = 0$$
$$2 \times 0 = 0$$
$$1 \times 0 = 0$$
$$64 + 16 + 8 = 88$$

MSB is one, so the number is negative.
Therefore, the number being represented is -88.

Number Systems

Converting Sign and Magnitude Binary to Decimal

Examples: Conversion

Convert the sign and magnitude binary number 00000011_2 into denary.

MSB	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1

$64 \times 0 = 0$
$32 \times 0 = 0$
$16 \times 0 = 0$
$8 \times 1 = 0$
$4 \times 0 = 0$
$2 \times 1 = 2$
$1 \times 1 = 1$
$2 + 1 = 3$

MSB is zero, so the number is positive.
Therefore, the number being represented is 3.

Convert the sign and magnitude binary number 10000011_2 into denary.

MSB	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

$64 \times 0 = 0$
$32 \times 0 = 0$
$16 \times 0 = 0$
$8 \times 1 = 0$
$4 \times 0 = 0$
$2 \times 1 = 2$
$1 \times 1 = 1$
$2 + 1 = 3$

MSB is one, so the number is negative.
Therefore, the number being represented is -3.

Number Systems

One's Complement

Definitions

One's Complement is a way of representing binary number by inverting all the bits in the binary representation of the number.

Converting Decimal to One's Complement Binary

Example: Conversion

Convert -2_{10} to a One's Complement binary number.

	128	64	32	16	8	4	2	1
Binary	0	0	0	0	0	0	1	0
One's Complement	1	1	1	1	1	1	0	1

Issues

Using One's Complement introduces an issue when performing addition operations. In the event of an overflow, the overflow bit must be wrapped around as a carry bit.

Example: Addition

Perform addition on the binary equivalents of 13_{10} and -2_{10}.

	OVERFLOW	128	64	32	16	8	4	2	1
13		0	0	0	0	1	1	0	1
-2		1	1	1	1	1	1	0	1
$13 + (-2)$	1	0	0	0	0	1	0	1	0
CARRY	1	1	1	1	1	1		1	

There is an overflow and therefore the overflow bit must be wrapped around as a carry bit.

	OVERFLOW	128	64	32	16	8	4	2	1
$13 + (-2)$	~~1~~	0	0	0	0	1	0	1	0
$+1$									1
RESULT		0	0	0	0	1	0	1	1

This is an issue because handling the overflow requires more complex hardware.

Number Systems

Two's Complement

Definitions

Two's Complement is a way of representing binary numbers where the most significant bit (MSB) is always negative.

Converting Decimal to Two's Complement Binary

Example: Conversion

Convert -57_{10} to a two's complement binary number.

-128	64	32	16	8	4	2	1
0	0	1	1	1	0	0	1

Convert the denary number to binary.

-128	64	32	16	8	4	2	1
1	1	0	0	0	1	1	0

Convert to One's Complement by flipping the bits.

-128	64	32	16	8	4	2	1
1	1	0	0	0	1	1	1

Add one.

$$
\begin{aligned}
-128 \times 1 &= -128 \\
64 \times 1 &= 64 \\
32 \times 0 &= 0 \\
16 \times 0 &= 0 \\
8 \times 0 &= 0 \\
4 \times 1 &= 4 \\
2 \times 1 &= 2 \\
1 \times 1 &= 1
\end{aligned}
$$

$$-128 + 64 + 4 + 2 + 1 = -57$$

Number Systems

Binary Addition

Process: Addition

1) Add the first binary number and the second binary number together using the binary addition rules:
   - $0_2 + 0_2 = 00_2 = 0_{10}$
   - $0_2 + 1_2 = 01_2 = 1_{10}$
   - $1_2 + 0_2 = 01_2 = 1_{10}$
   - $1_2 + 1_2 = 10_2 = 2_{10}$
   - $1_2 + 1_2 + 1_2 = 11_2 = 3_{10}$
2) If the expression evaluates to an answer larger than one bit:
   - the first of the two bits is a carry bit;
   - the carry bit is moved to the next column; and
   - the addition in the next column must incorporate the carry bit.
3) If there is a carry bit present after performing addition on the most significant bit (MSB), there is an overflow error because there are not enough bits to represent all of the binary digits.

Example: Addition

Perform addition on the binary equivalents of 211_{10} and 54_{10}.

	128	64	32	16	8	4	2	1
211	1	1	0	1	0	0	1	1
54	0	0	1	1	0	1	0	0

Write 211 and 54 in binary.

	128	64	32	16	8	4	2	1
211	1	1	0	1	0	0	1	1
54	0	0	1	1	0	1	0	0
SUM	0	0	0	0	0	1	1	1
CARRY	1	1	1					

Add the binary digits together.

$1 \mid \mathbf{00000111_2}$

Write the answer.

There is an overflow error.

Number Systems

Binary Subtraction

Arithmetic Method

Process: Subtraction

1) Convert the denary number to binary.
2) For each place value, starting at the least significant bit (LSB), subtract the second binary number from the first binary number using the binary addition rules:
   - $0 - 0 = 0$;
   - $1 - 0 = 1$; and
   - $1 - 1 = 0$.
3) If there is an subtraction operation where $0 - 1$ occurs, borrow two from the column to the left:
   - cross out the one (1) in the column to the left;
   - place two ones (1s) above the current column; and
   - perform the subtraction operation.
4) If there is an subtraction operation where $0 - 1$ occurs, and there is a zero in the column to the left:
   - continue to the left until there is a column with a one (1);
   - cross out the one (1) in that column;
   - place two ones (1s) above the column to the right:
     - if the column to the right is the current column, perform the subtraction operation;
     - if the column to the right is not the current column, cross out one of the borrowed ones (1s) and place two ones (1s) above the column to the right – continue this process until the current column is reached and perform the subtraction operation.
5) Write the answer.

Example: Subtraction

Perform subtraction on the binary equivalents of $85_{10} - 16_{10}$.

	128	64	32	16	8	4	2	1
85	0	1	0	1	0	1	0	1
16	0	0	0	1	0	0	0	0

Convert the denary number to binary.

	128	64	32	16	8	4	2	1
85	0	1	0	1	0	1	0	1
16	0	0	0	1	0	0	0	0
Add one	0	1	0	0	0	1	0	1

Perform the subtraction operation.

$\mathbf{01000101_2}$

Write the answer.

Number Systems

Example: Subtraction

Perform subtraction on the binary equivalents of $16_{10} - 4_{10}$.

	128	64	32	16	8	4	2	1
16	0	0	0	1	0	0	0	0
4	0	0	0	0	0	1	0	0

Convert the denary number to binary.

	128	64	32	16	8	4	2	1
16	0	0	0	1	0	0	0	0
4	0	0	0	0	0	1	0	0
SUB							0	0

Perform the subtraction operation.

If there is a subtraction operation where $0 - 1$ occurs, and there is a zero in the column to the left:

- continue to the left until there is a column with a one (1);
- cross out the one (1) in that column;
- place two ones (1s) above the column to the right:
  - if the column to the right is the current column, perform the subtraction operation;
  - if the column to the right is not the current column, cross out one of the borrowed ones (1s) and place two ones (1s) above the column to the – continue this process until the current column is reached and perform the subtraction operation.

BORROW					1 1	1 1		
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
16	0	0	0	1	0	0	0	0
4	0	0	0	0	0	1	0	0
SUB	0	0	0	0	1	1	0	0

$\mathbf{00001100_2}$

Write the answer.

Number Systems

Two's Complement Method

Process: Subtraction

1) Convert both denary numbers to binary.
2) Convert the second binary number into a negative binary number using Two's Complement.
3) Add the first binary number and the converted second binary number together using the binary addition rules:
   - $0_2 + 0_2 = 00_2 = 0_{10}$
   - $0_2 + 1_2 = 01_2 = 1_{10}$
   - $1_2 + 0_2 = 01_2 = 1_{10}$
   - $1_2 + 1_2 = 10_2 = 2_{10}$
   - $1_2 + 1_2 + 1_2 = 11_2 = 3_{10}$
4) If the expression evaluates to an answer larger than one bit:
   - the first of the two bits is a carry bit;
   - the carry bit is moved to the next column; and
   - the addition in the next column must incorporate the carry bit.
5) If there is a carry bit present after performing addition on the most significant bit (MSB), there is an overflow error because there are not enough bits to represent all of the binary digits.
6) Write the answer and discard any overflow bits.

Example: Subtraction

Perform subtraction on the binary equivalents of $85_{10} - 16_{10}$.

	-128	64	32	16	8	4	2	1
85	0	1	0	1	0	1	0	1
16	0	0	0	1	0	0	0	0

Convert both denary numbers to binary.

	-128	64	32	16	8	4	2	1
16	0	0	0	1	0	0	0	0
Flip Bits	1	1	1	0	1	1	1	1
Add one	1	1	1	1	0	0	0	0
CARRY				1	1	1	1	

Convert the second binary number into a negative binary number using Two's Complement.

	-128	64	32	16	8	4	2	1
86	0	1	0	1	0	1	0	1
-16	1	1	1	1	0	0	0	0
SUM	0	1	0	0	0	1	0	1
CARRY	1	1	1					

Add the first binary number and the converted second binary number together using the binary addition rules.

$\mathbf{01000101_2}$

Write the answer.

Number Systems

Example: Subtraction

Perform subtraction on the binary equivalents of $16_{10} - 4_{10}$.

	128	64	32	16	8	4	2	1
16	0	0	0	1	0	0	0	0
4	0	0	0	0	0	1	0	0

Convert both denary numbers to binary.

	-128	64	32	16	8	4	2	1
4	0	0	0	0	0	1	0	0
Flip Bits	1	1	1	1	1	0	1	1
Add one	1	1	1	1	1	1	0	0
CARRY						1	1	

Convert the second binary number into a negative binary number using Two's Complement.

	-128	64	32	16	8	4	2	1
16	0	0	0	1	0	0	0	0
-4	1	1	1	1	1	1	0	0
SUM	0	0	0	0	1	1	0	0
CARRY	1	1	1					

Add the first binary number and the converted second binary number together using the binary addition rules.

$\mathbf{00001100_2}$

Write the answer.

Number Systems

Binary Multiplication

Process: Multiplication

1) Convert the denary numbers to binary.
2) Starting at the least significant bit (LSB) of the top binary number and repeat until all binary digits have been multiplied:
   - multiply the binary digit from the top binary number by each of the digits in the bottom binary number;
   - write the result in the corresponding column below; and
   - after the first binary digit from the top binary number, shift the result left one position.
3) Perform addition on all of the results.
4) Write the answer.

Example: Multiplication

Perform multiplication on the binary equivalents of 11_{10} and 5_{10}.

	128	64	32	16	8	4	2	1
11_{10}	0	0	0	0	1	0	1	1
5_{10}	0	0	0	0	0	1	0	1

Convert the denary numbers to binary.

Starting at the least significant bit (LSB) of the top binary number and repeat until all binary digits have been multiplied:

- multiply the binary digit from the top binary number by each of the digits in the bottom binary number;
- write the result in the corresponding column below; and
- after the first binary digit from the top binary number, shift the result left one position.

	128	64	32	16	8	4	2	1
11_{10}	0	0	0	0	1	0	1	1
5_{10}	0	0	0	0	0	1	0	1
RESULT 1	0	0	0	0	0	1	0	1
RESULT 2	0	0	0	0	1	0	1	**0**
RESULT 3	0	0	0	0	0	0	**0**	**0**
RESULT 4	0	0	1	0	1	**0**	**0**	**0**

RESULT 1	0	0	0	0	0	1	0	1
RESULT 2	0	0	0	0	1	0	1	**0**
RESULT 3	0	0	0	0	0	0	**0**	**0**
RESULT 4	0	0	1	0	1	**0**	**0**	**0**
SUM	0	0	1	1	0	1	1	1
CARRY				1				

Perform addition on all of the results

$$\mathbf{00110111_2}$$

Write the answer.

Number Systems

Binary Fractions

Place Values

Fractional Binary Place Values

Position	8	7	6	5	4	3	2	1 ●	-1	-2	-3	-4	-5	-6	-7
Weight *(base 2)*	128	64	32	16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$

The decimal point is placed between the weights 1 and $\frac{1}{2}$.

To calculate weights:

- to the left of the decimal point – double the previous weight; and
- to the right of the decimal point – half the previous weight.

Fraction to Decimal

While the decimal place values are expressed as fractions, it is easier to convert these to decimal numbers in order to perform calculations.

Fractional Binary Place Values

Fraction	Decimal
$\frac{1}{2}$	0.5
$\frac{1}{4}$	0.25
$\frac{1}{8}$	0.125
$\frac{1}{16}$	0.0625
$\frac{1}{32}$	0.03125
$\frac{1}{64}$	0.015625
$\frac{1}{128}$	0.0078125

Number Systems

Example

Example: Determining Decimal Numbers

Determine the decimal number represented below.

Position	8	7	6	5	4	3	2	1	-1	-2	-3	-4	-5	-6	-7
Weight *(base 2)*	128	64	32	16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$
Binary Number	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0

$$value = \sum_{n=0}^{\infty}(weight_n \times digit_n)$$

$$value = (128 \times 0) + (64 \times 0) + (32 \times 0) + (16 \times 0) + (8 \times 1) + (4 \times 1) + (2 \times 0) + (1 \times 0) + \left(\frac{1}{2} \times 1\right) + \left(\frac{1}{4} \times 0\right) + \left(\frac{1}{8} \times 1\right) + \left(\frac{1}{16} \times 0\right) + \left(\frac{1}{32} \times 0\right) + \left(\frac{1}{64} \times 0\right) + \left(\frac{1}{128} \times 0\right)$$

$$value = 0 + 0 + 0 + 0 + 8 + 4 + 0 + 0 + 0.5 + 0 + 0.125 + 0 + 0 + 0 + 0$$

$$value = 8 + 4 + 0.5 + 0.125$$

$$\boldsymbol{value = 12.625}$$

Converting Fractional Decimal to Fractional Binary

Process: Conversion

Use either the division method or subtraction method to perform the conversions.

1) Convert the integer part of the decimal number first, disregarding the fractional part.
2) Convert the fractional part of the decimal number.
3) Write the answer.

Example: Conversion

Convert 119.875_{10} to binary.

128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	1

Convert the integer part of the decimal number first, disregarding the fractional part

128	64	32	16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$
1	0	0	0	1	0	0	1	1	1	1	0	0	0	0

Convert the fractional part of the decimal number.

$$\mathbf{10001001.1110000_2}$$

Write the answer.

Number Systems

Hexadecimal

Definition

The **hexadecimal number system**, or **hex**, is a number system that:

- uses a notation in which each number is expressed in base 16;
- uses the number range 0-9, A-F; and
- lets each place value be a power of 16.

One hexadecimal character is represented by a nibble (4 bits).

Place Values

Hexadecimal Place Values

Position	4	3	2	1
Weight *(base 16)*	4096	256	16	1

The minimum decimal value for a binary number constructed using n bits is 0.
The maximum decimal value for a binary number constructed using n bits is calculated by $16^n - 1$.

Example

Example: Determining Hexadecimal Numbers

Determine the hexadecimal number represented below.

Position	2	1
Weight *(base 16)*	16	1
Binary Number	4	1

$$value = \sum_{n=0}^{\infty} (weight_n \times digit_n)$$
$$value = (16 \times 4) + (1 \times 1)$$
$$value = 64 + 1$$
$$\boldsymbol{value = 65}$$

Number Systems

Symbols

Hexadecimal Symbols

Decimal	Hexadecimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Converting Decimal to Hexadecimal

Division Method

Process: Conversion

1) Set the remainder equal to the decimal number.
2) Starting at the leftmost power and repeat until division leaves no remainder:
   - divide the remainder by the power;
   - if the result is 0, place the result under that place value and do not change the remainder; and
   - if the result is not 0, place the result under that place value and set the remainder r to the value left over after the division.
3) Write the answer.

Example: Conversion

Convert 56_{10} to hexadecimal.

$r = 56$

4096	256	16	1

$\frac{56_{10}}{4096_{10}} = 0_{10}\ (r = 56)$

4096	256	16	1
0			

Set the remainder equal to the decimal number.

Starting at the leftmost power and repeat until division leaves no remainder:

- divide the remainder by the power;
- if the result is 0, place the result under that place value and do not change the remainder; and
- if the result is not 0, place the result under that place value and set the remainder r to the value left over after the division.

Number Systems

$\frac{56_{10}}{256_{10}} = 0_{10}\ (r = 56)$

4096	256	16	1
0	0		

$\frac{56_{10}}{16_{10}} = 3_{10}\ (r = 8)$

4096	256	16	1
0	0	3	

$\frac{8_{10}}{1_{10}} = 8_{10}\ (r = 0)$

4096	256	16	1
0	0	3	8

Starting at the leftmost power and repeat until division leaves no remainder:

- divide the remainder by the power;
- if the result is 0, place the result under that place value and do not change the remainder; and
- if the result is not 0, place the result under that place value and set the remainder r to the value left over after the division.

$\mathbf{0038_{16}}$

Write the answer.

Via Binary Method

Process: Conversion

1) Convert the decimal number to binary.
2) Split the binary number into octets (4 bits) and give each octet its own place values – if necessary zero's (0's) can be added before the most significant bit (MSB) if the number of characters in the binary number is not divisible by four.
3) Either:
   - compare each binary octet to the hexadecimal symbols table to find its hexadecimal equivalent; or
   - convert each binary octet to decimal and compare the decimal number to the hexadecimal symbols table to find its hexadecimal equivalent.
4) Write the answer.

Number Systems

Example: Conversion

Convert 56_{10} to hexadecimal.

128	64	32	16	8	4	2	1
0	0	1	1	1	0	0	0

Convert the decimal number to binary.

8	4	2	1
0	0	1	1

8	4	2	1
1	0	0	0

Split the binary number into octets (4 bits) and give each octet its own place values – if necessary zero's (0's) can be added before the most significant bit (MSB) if the number of characters in the binary number is not divisible by four.

Decimal	Hexadecimal	Binary
3	3	11

$\therefore 0011_2 = 3_{16}$

Decimal	Hexadecimal	Binary
8	8	1000

$\therefore 1000_2 = 8_{16}$

Compare each binary octet to the hexadecimal symbols table to find its hexadecimal equivalent.

Either:

8	4	2	1
0	0	1	1

$0011_2 = 3_{10}$

Decimal	Hexadecimal	Binary
3	3	11

$\therefore 3_{10} = 3_{16}$

8	4	2	1
1	0	0	0

$1000_2 = 8_{10}$

Decimal	Hexadecimal	Binary
8	8	1000

$\therefore 8_{10} = 8_{16}$

Convert each binary octet to decimal and compare the decimal number to the hexadecimal symbols table to find its hexadecimal equivalent.

$\mathbf{0038_{16}}$

Write the answer.

Computer Technology

The CPU

Components of a Computer System

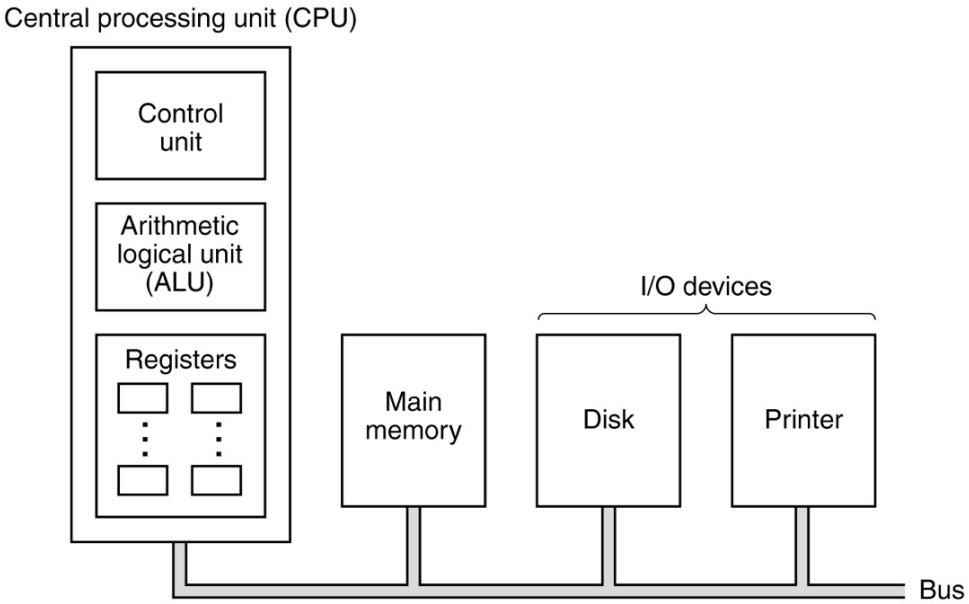
Definition

A **computer system** allows users to input, manipulate and store data.

Von Neumann Architecture

The Von Neumann Architecture consists of a CPU, memory and input / output that are connected by a set of buses.

Diagram: Computer System



Component	Use
Central processing unit (CPU)	The Central Processing Unit (CPU) is the hardware within a computer that carries out the instructions of a computer program by performing the basic arithmetical, logical, and input/output operations of the system.
Main memory	Main memory stores data and instructions frequently being used by the CPU. Random access memory (RAM) is an example of main memory.
Input/Output (I/O) Devices	Input/Output (I/O) devices are those which can either input or output data from a computer system: • input devices provide input to a computer system; and • output devices provide a way for a computer system to output data for communication with users or other computer systems. Examples of I/O devices include: disks; printers; Ethernet; and wireless cards.
System Bus	The system bus is comprised of the address bus, data bus and control bus.

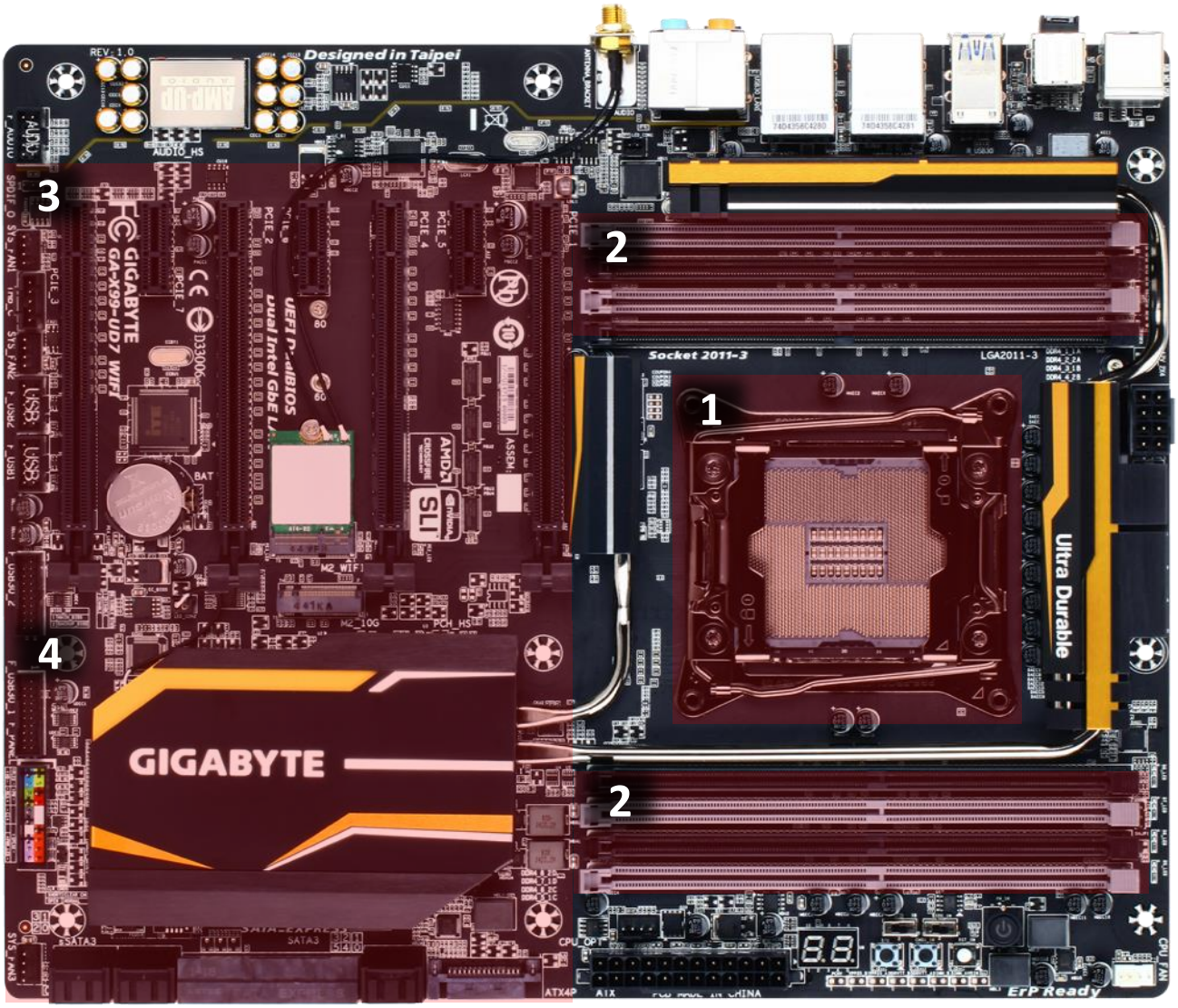
- Program instructions and program data reside within memory.
- The CPU has to fetch instructions from memory into the CPU.
- Instructions are processed, decoded and executed by the CPU and then the next instruction is fetched.

The CPU is an essential part of the computer system as it is capable of understanding binary instructions.

The CPU

Components of a Desktop Computer System

Image: Gigabyte x99 Motherboard

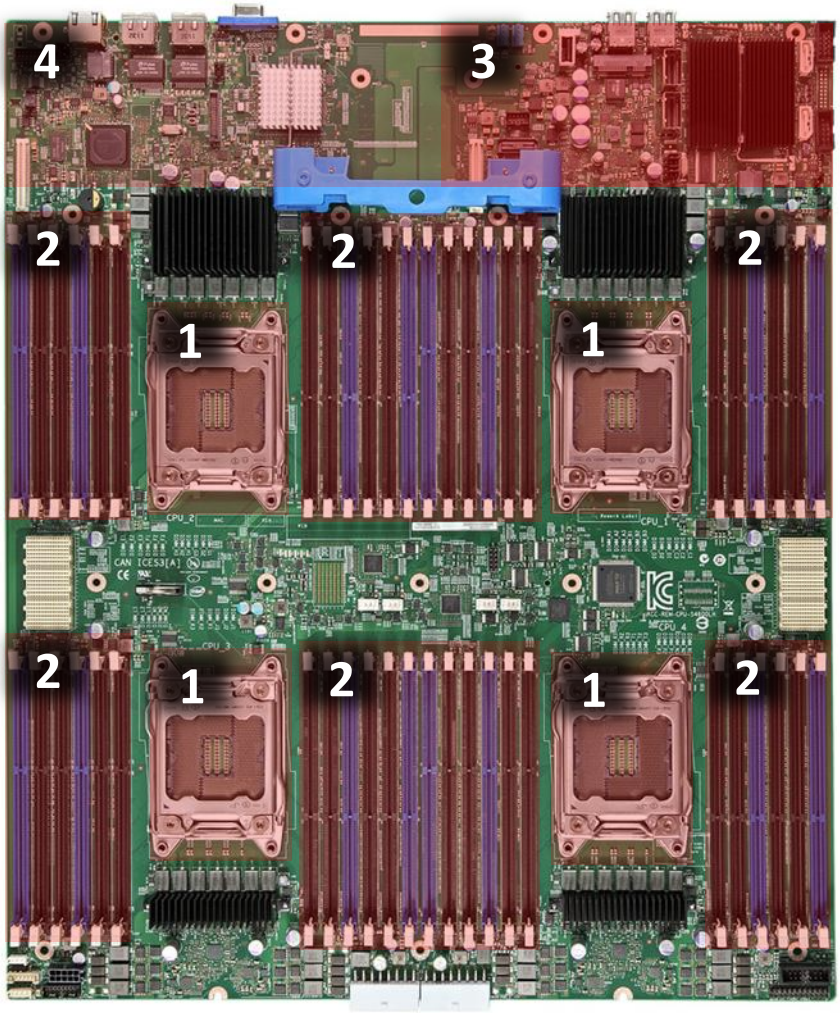


Number	Component
1	Central processing unit (CPU)
2	Main memory (RAM)
3	Bus & I/O
4	Bus control circuitry & BIOS

The CPU

Components of a Server Computer System

Image: Intel Server Board S4600LH2

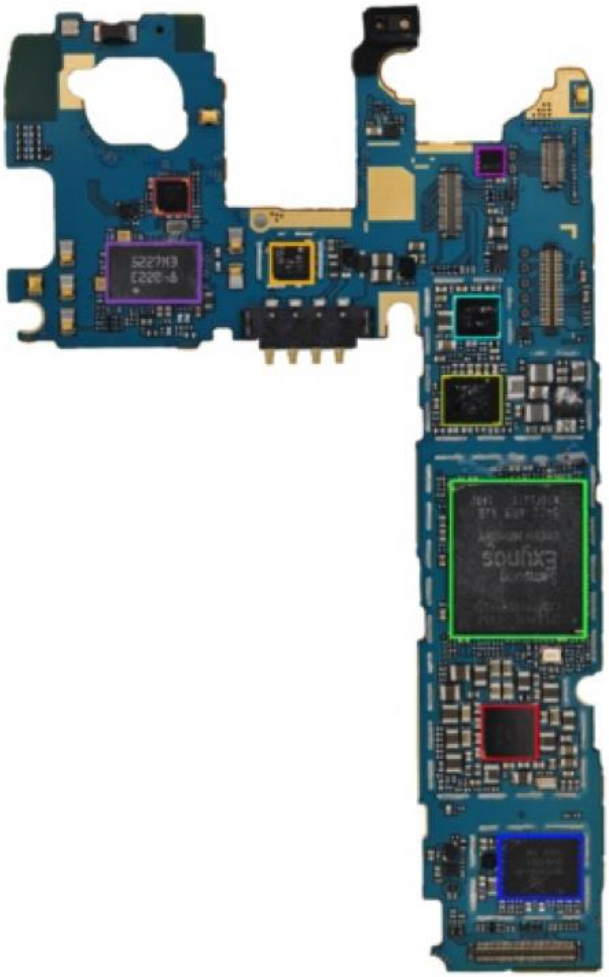


Number	Component
1	Central processing unit (CPU)
2	Main memory (RAM)
3	Bus & I/O
4	Bus control circuitry & BIOS

The CPU

Components of a Smartphone

Image: Samsung Galaxy S5 Motherboard



Number	Component
1	Central processing unit (CPU)
2	Main memory (RAM)
3	Bus & I/O
4	Bus control circuitry & BIOS

Due to the smaller form factor of smartphones when compared to other computer systems, multiple small motherboards have to be used.

The CPU

Importance of the CPU

Characteristics

- The CPU (central processing unit) is the core of a modern computer.
- It is the functional brain of the computer performing operations and redirecting data between peripherals.
- Without the CPU the rest of the components would be of little use.

Capabilities

The CPU can perform a small set of basic operations:

- arithmetic – add, subtract, multiply, divide, etc.;
- memory access – fetch data from memory, store results back in memory;
- decision making – compare numbers, letters, etc. and decide what to do next according to the result; and
- control – controls other components of the computer system.

The CPU operates by performing sequences of very simple operations very fast:

- memory operations – load, store, move, jump and offset;
- arithmetic operations – add and subtract;
- comparison operations – Equals Zero and Not Equals Zero; and
- branch operations – branch if zero and branch if not zero.

The CPU

Views of the CPU

Hardware View

- The CPU (central processing unit) is the core of a modern computer.
- It is the functional brain of the computer performing operations and redirecting data between peripherals.
- Without the CPU the rest of the components would be of little use.

Microarchitecture View

- Processor designer view - how all the bits fit together to make the machine
- Logical organization that implements the ISA
- Pipelining, functional units, caches, physical registers etc.

Instruction Set Architecture View

- Programmer/compiler view - drive all the bits connected together to perform a task
- Uses machine code
- Instructions visible to the (system) programmer
- Opcode, addressing mode, architectural registers etc.

The CPU – Hardware View

Components

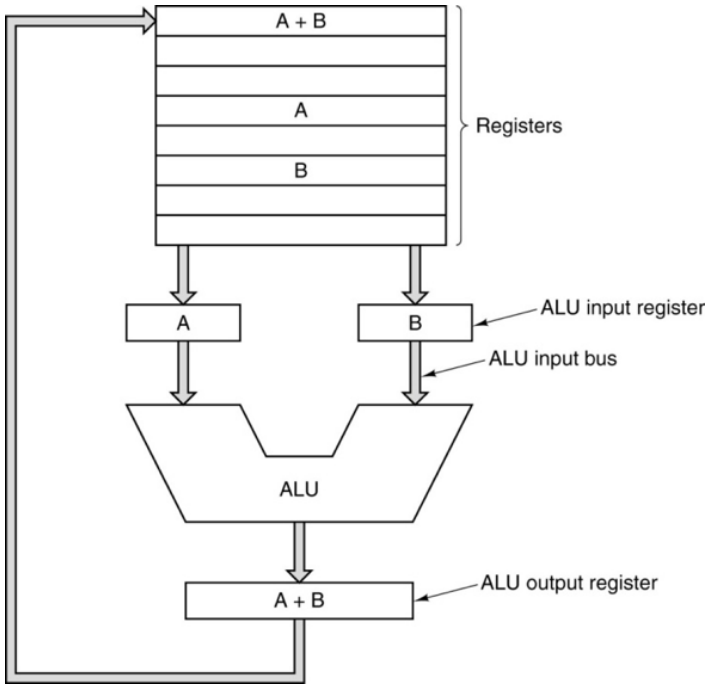
- **Control unit** – responsible for managing the flow of data around the computer and controlling the operation of the other units.
- **Arithmetic and Logic Unit (ALU)** – responsible for performing arithmetic and logic operations, it allows the AND'ing, OR'ing and SUM'ing (addition/subtraction) of two words.
- **Registers** – very fast memory locations stored in the processor itself which are used to temporarily store values for processing.
- **Memory** – longer term storage than registers which are much larger, and correspondingly slower than registers.
- **Input and output** – devices outside of the computer system.

The CPU – Hardware View

Control Unit and Arithmetic and Logic Unit (ALU)

Performing Operations in the ALU

Diagram: Visual Representation of Data Flow

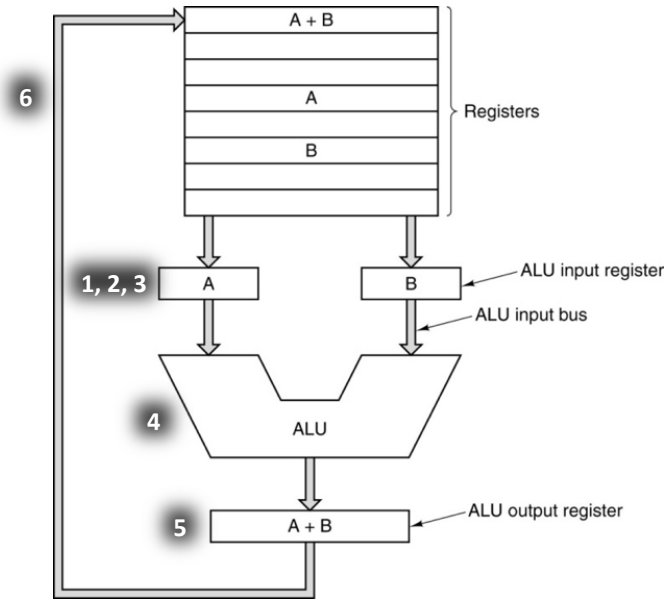


The data path of the control unit shows how data flows through each of the components. Registers are used to store operations and data as well as the output of the operation. A bus connects the registers and ALU together. This process follows the fetch-decode-execute cycle.

To perform the operation, there three components are required:

- the first operand (value);
- the opcode (operation); and
- the second operand (value).

Diagram: Performing an Operation



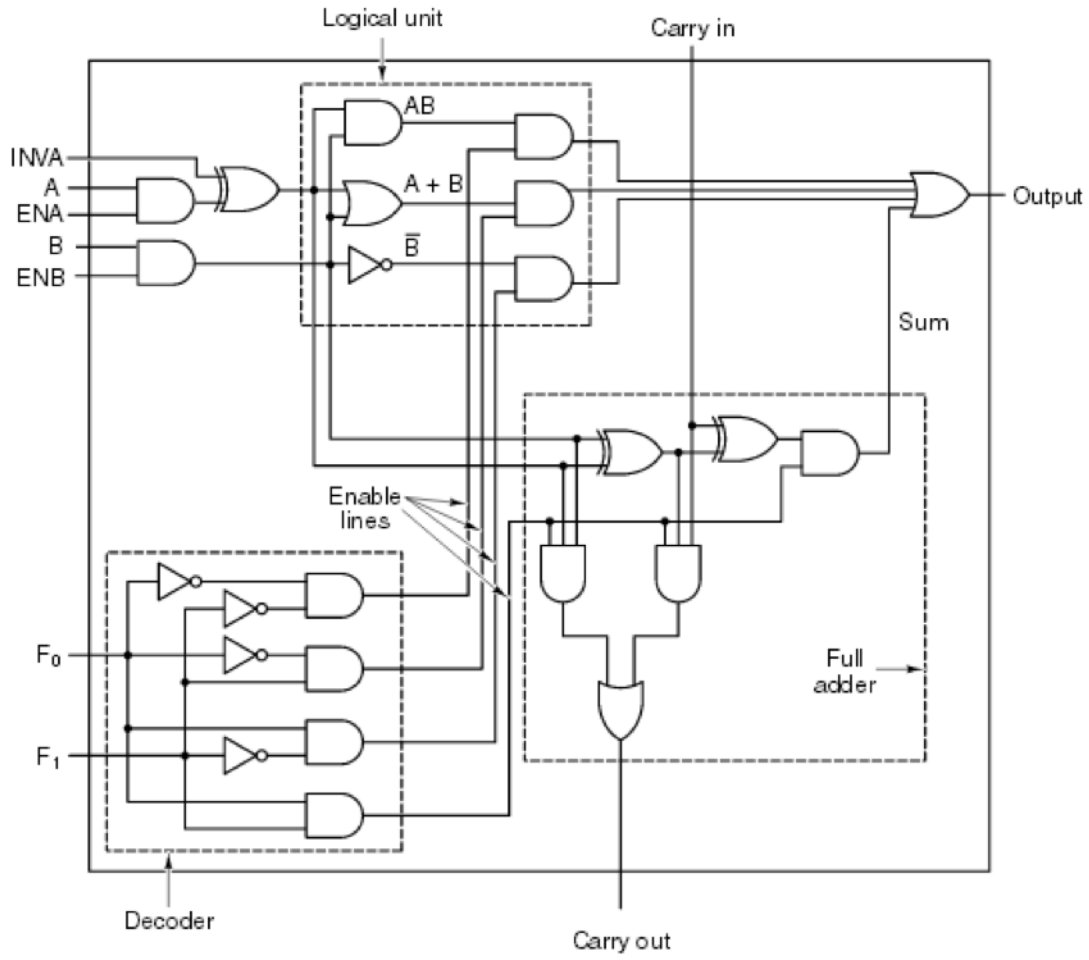
1) Both operands (A and B) are stored in the ALU input registers – these are used to store the register value, immediate value, memory address etc. in order to access the operand in memory.
2) The operation for the ALU is selected.
3) An operand is selected (A or B) and One's Complement or Two's Complement is performed on the values in the ALU input registers.
4) The operation is performed in the ALU.
5) The result of the operation is stored in the ALU output register.
6) The contents of the ALU output register is moved to a destination register so that the result can be kept as the contents of the ALU output register will be overwritten on subsequent operations.

This process is repeated for each operation.

The CPU – Hardware View

ALU Circuitry

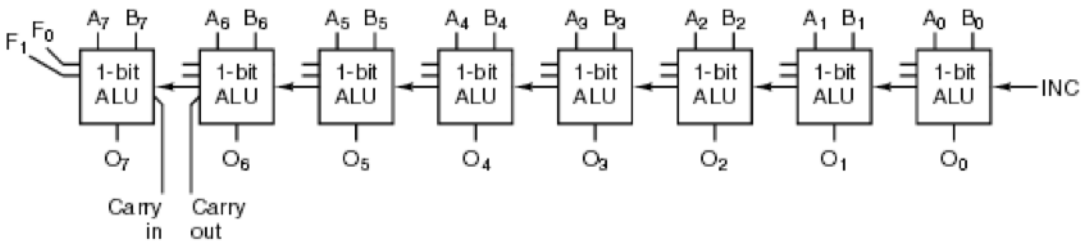
Diagram: 1-bit ALU Circuitry



The decoder decodes instruction to determine what segments will be active in the data path. A binary code can be put on F0 and F1 in order to control what happens to the data, this specifies which part of the CPU is active. For example, if some data must be fetched first, the ALU will not be activated.

In order to perform larger operations, multiple ALUs must be concatenated together.

Diagram: 8-bit ALU



The CPU – Hardware View

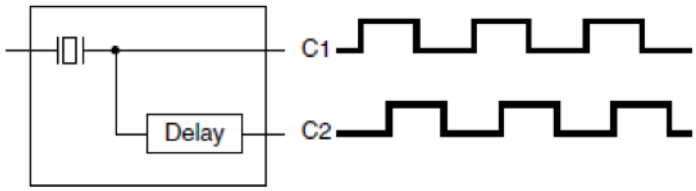
Clocks

Definition

Clock speed refers to the frequency at which the CPU is running and is used as an indicator of the processor's speed. It is measured in clock cycles per second which is written in hertz (Hz). The first generation of computers was measured in hertz or kilohertz (kHz), but the speed of modern CPUs is commonly measured in gigahertz (GHz).

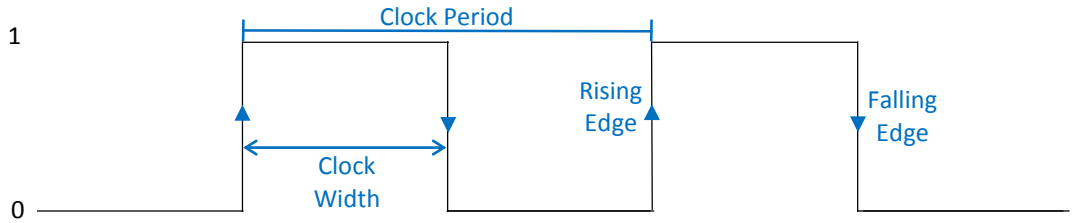
Circuitry

Diagram: A Clock



Clock Signal

Diagram: Clock Signal



A clock emits a series of pulses which have a fixed length and delay. The clock-cycle is the interval between two consecutive pulses. For example, the clock inside a CPU with a clock speed of 3.4GHz will emit 3400000000 pulses per second

The CPU – Hardware View

Registers

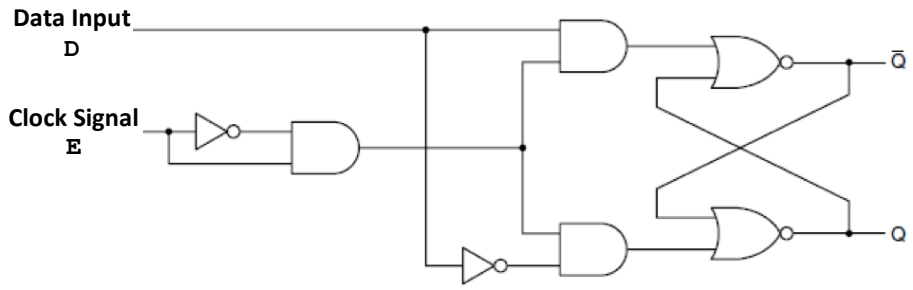
Definition

Registers are very fast memory locations stored in the processor itself which are used to temporarily store values for processing. Registers must be very fast as they would otherwise slow down the CPU, causing a bottleneck.

Circuitry

Single Bit Register

Circuitry



Truth Table

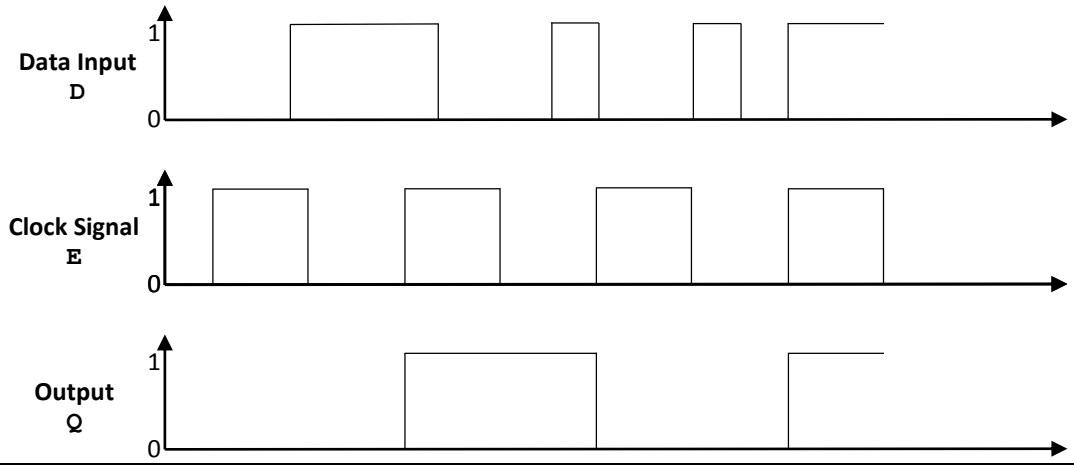
D	E	Q	¬Q	Operation
0	0	0	1	RESET
0	1	0	1	RESET
1	0	0	1	RESET
1	1	1	0	SET

Registers are typically implemented using flip-flops. Many single bit registers can be setup in parallel to achieve more bits. Their management is under the direct control of the instruction set architecture (ISA).

Flip Flops

A **flip flop** is a sequential logic unit that can store the state of one bit, 0 or 1. It has two inputs: a control input (D) and a clock signal (E). They are used in registers so that the contents of a register will only change when clock signal is present.

Diagram: Output Graph



The output of a flip flop shows that:

- when the clock signal is at a rising edge, the data input can change the output; and
- when the clock signal is not at a rising edge, the output remains constant.

The CPU – Hardware View

A flip flop has the capability to be store one bit of data. This means that they can be used in a central processor unit's (CPU's) registers to synchronise data and instructions with the clock signals.

In the fetch-decode-execute cycle, an address for data or instructions is transmitted across the address bus in order to be fetched from memory. The data or instructions are then transmitted across the data bus to the memory data register (MDR). However, the data or instructions may arrive at the memory data register (MDR) while a current fetch-decode-execute cycle is in progress.

The contents of the registers must not be changed by an external influence during a fetch-decode-execute cycle, otherwise incorrect instructions can be performed and incorrect calculations can be made. Therefore, the use of the flip flop allows registers to maintain the same value until a new fetch-decode-execute cycle begins, at the next rising edge of the clock signal – therefore preventing the contents of the current instruction register (CIR) to be changed.

A register may contain many concatenated flip flops, for example an 8-bit register would require eight concatenated flip flops.

The CPU – Microarchitecture View

Datapath

Definition

A **datapath** is a collection of functional units, such as arithmetic logic units or multipliers, that perform data processing operations, registers, and buses.

Combinatorial Inputs

In this type of datapath, the output follows the input and there is a combination of arithmetic / logic operations

Examples:

- ALU
- Sign Extender
- Number format translator
- Adder
- Multiplexer (MUX)

Diagram: Datapath Visualisation

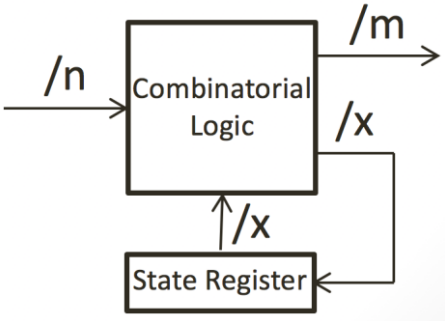
State Elements

In this type of datapath, outputs and inputs change only on the clock edge.

Examples:

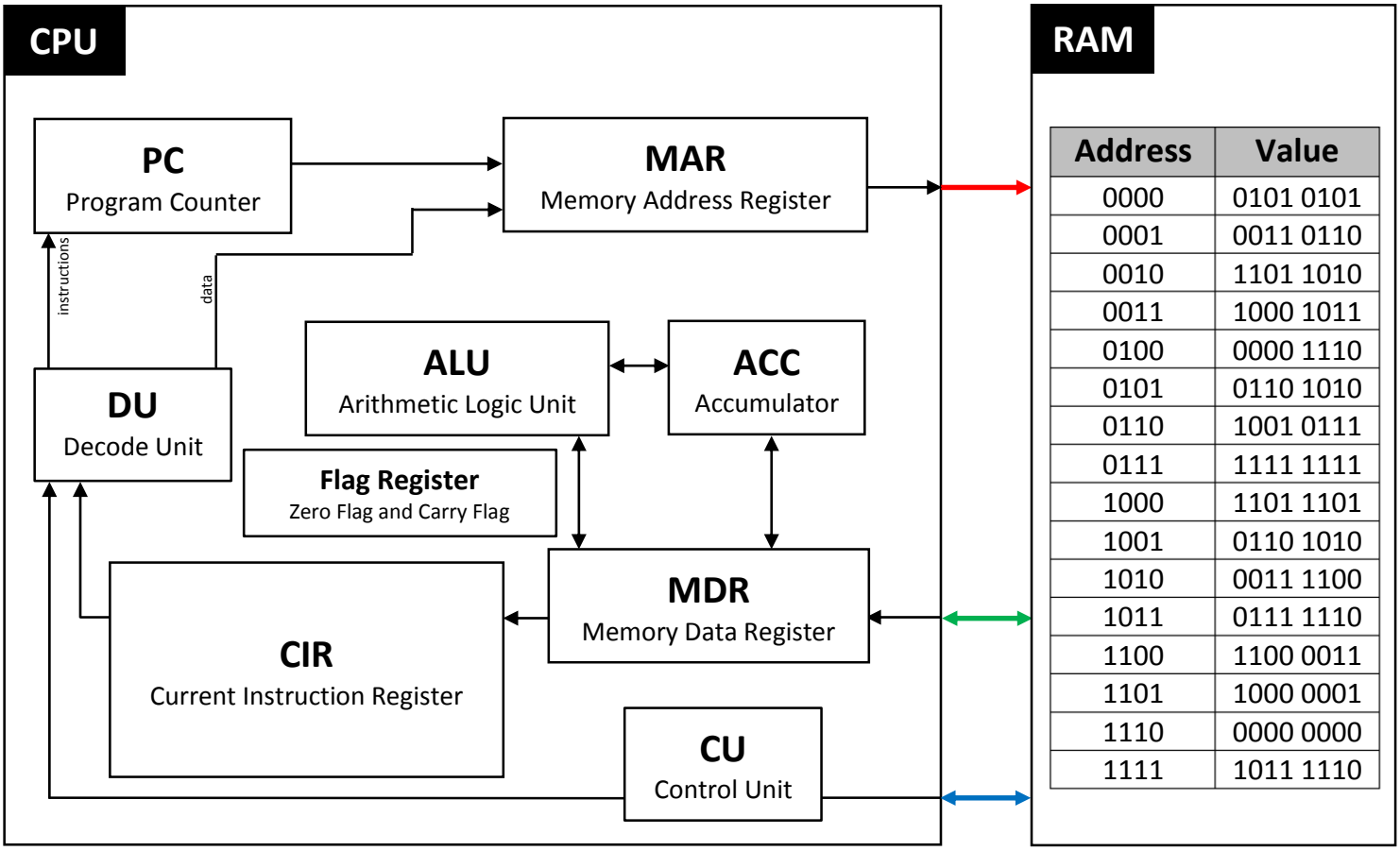
- Registers
- Clocks
- Memory
- Program Counter

Diagram: Datapath Visualisation



The CPU – Microarchitecture View

CPU Diagram



The system bus is comprised of the address bus, data bus and control bus:

Address Bus	The CPU sends the memory location along this bus to allow access to that data or instruction.
Data Bus	Transfers data or instructions from input devices to the CPU to be processed, transfers data from the CPU to RAM and transfers data to output devices.
Control Bus	Keeps track of all parts of the computer system by sending various control signals, such as read and write access.

The system bus connects to a multitude of registers:

PC Program Counter	Stores the memory address location of the instruction which will be fetched on the next cycle.
MAR Memory Address Register	Stores the memory address location of the where data or instructions will be read or written.
CU Control Unit	Sends control signals to all parts of the CPU and peripherals which coordinate their actions.
DU Decode Unit	Contains the instruction set to identify instructions.
ALU Arithmetic Logic Unit	Carries out any arithmetic or logic operations, this is tightly coupled with the ACC.
ACC Accumulator	Stores the result of operations from the ALU, this is tightly coupled with the ALU.
CIR Current Instruction Register	Stores current instruction to be executed, having been fetched from memory.
MDR Memory Data Register	Stores data that has been read from RAM or waiting to be written to RAM.
Flag Registers Zero Flag and Carry Flag	Used by branch instructions and when carries occur.

A clock is externally generated and controlled by the CPU. This goes out from the CPU to all internal parts of the computer system to create a synchronised system; all parts have a common clock signal and therefore know when to place data on the buses and read data from the buses.

The CPU – Microarchitecture View

Fetch-Decode-Execute Cycle

Definition

The **fetch-decode-execute cycle** is an operational process in which a computer system retrieves a program instruction from its memory, determines what actions the instruction dictates, and carries out those actions. This cycle is always being completed while the computer system has power.

Processing an Instruction

Process: Fetch-Decode-Execute Cycle

Fetch

Retrieving the next instruction from memory.

- ***Program instructions are stored in memory.***
- Memory address of next instruction copied from program counter (PC) to the memory address register (MAR).
- Memory address of the next instruction transferred along the address bus to main memory (RAM).
- A signal is sent along the control bus to the memory controller to allow a read memory operation.
- Contents of the memory address transferred along the data bus to the memory data register (MDR).
- Contents of the memory data register (MDR) are either:
  - copied to the current instruction register (CIR) if it is an instruction; or
  - copied to the memory data register (MDR) if it is a datum.
- The program counter (PC) is incremented to allow the next instruction to be fetched.

↓

Decode

Understanding the instruction.

- ***The instruction is decoded by the control unit to determine what the instruction requires the CPU to perform.***
- ***There may be a case where more information is to be fetched, such as operands, before the instruction can be executed.***
- The instruction stored in the current instruction register (CIR) is decoded and split into:
  - an opcode – used to determine the type of instruction will be executed and what hardware will be used in the execution; and
  - an operand – this contains either:
    - the address of the data to be operated on, which is copied to the memory address register (MAR); or
    - the actual data to be operated on, which is copied to the memory data register (MDR), the arithmetic logic unit (ALC) or accumulator.

↓

Execute

Carrying out the instruction.

- ***When all data is available, the CPU will execute the instruction on operands.***
- If the instruction requires data to be fetched:
  - transfer memory address of the instruction to the memory address register (MAR); and
  - carry out fetch process.

  If this occurs, the program counter (PC) and the memory address register (MAR) will contain different values because the program counter (PC) is storing the memory address of the next instruction while the memory address register (MAR) is storing the memory address of the data to be fetched in order to complete the current instruction. When fetched, the value is transferred to the accumulator (ACC) and may be transferred to the arithmetic logic unit (ALU) if an arithmetic operation is to be performed.
- If a branch instruction is performed, the contents of the program counter (PC) must be changed to the memory address of instruction to be performed.
- ***Increment the program counter to point to the next instruction.***

↓

Interrupts

Checking to see if there are any interrupts.

- Run the process to check for interrupts (see page 75).

The CPU – Microarchitecture View

Modern Design Principles

Definitions

Modern design principles are criteria that should be met by modern CPUs.

Common Principles

- All instructions directly executed by hardware.
- Maximise rate at which instructions are issued.
- Instructions should be easy to decode.
- Only loads, stores should reference memory.
- Provide plenty of registers.

The CPU – Microarchitecture View

Performance

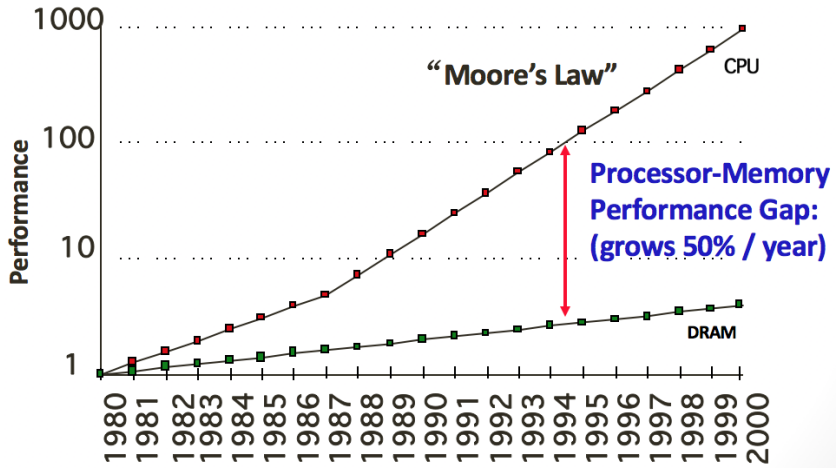
Limited Performance

A single stage computer is limited in the speed it can operate at due to the depth of gates.

Moore's Law

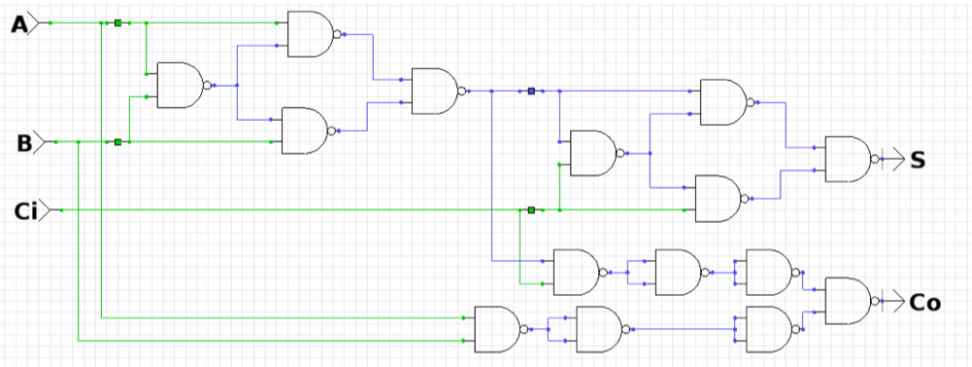
Moore's Law states that processor speeds, or overall processing power, for computers will double every two years.

Graph: Moore's Law



Case Study

Diagram: Single Bit Full Adder



In this circuit, the longest path (critical path) is seven NAND gates long.

For example, if each NAND gate takes 5ns to operate, this circuit takes 35ns to give us an output (based on a TI NAND gate device). Therefore, in the best case scenario, the circuit can operate at 1/35ns (= 28.5 MHz) – this is much slower than the typical clock speed of a modern CPU.

As a result, it is necessary to introduce methods to improve performance.

The CPU – Microarchitecture View

Improving Performance – Caches

Definition

Cache memory consists of blocks of small, expensive and volatile memory located closer to the CPU than main memory. It holds blocks of data that the cache believes:

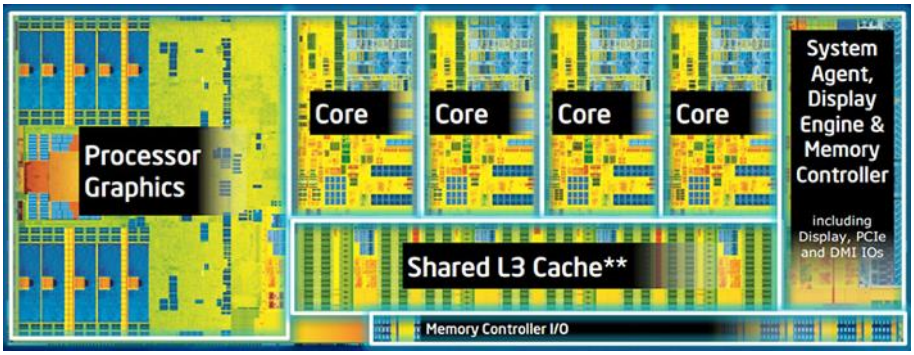
- is currently in use;
- has just been used; or
- will likely be used by the processor and memory locations near that (principle of locality).

Cache Location

There are multiple levels to cache memory:

- level 1 cache – very fast and small size (~2KB – 64KB);
- level 2 cache – fast and medium size (~256KB – 2MB); and
- level 3 cache – Slower and large size (~8MB – 64MB).

Diagram: 4th Generation Intel Core Processor Die Map



The diagram shows a quad-core CPU.

Level 1 and 2 caches are typically on the CPU chip. Level 3 cache can be off chip but with newer designs, can be found on chip. In this example, the level 1 cache and level 2 cache are located on each core while the level 3 cache is shared between all four cores.

A combination of circuit design and being physically closer to the processor make these caches faster than main memory.

Access

When querying cache for data or instructions, there are two possible outcomes:

- a "cache hit" – data or instructions found in cache; or
- a "cache miss" – data or instructions not found in cache.

$$average\ access\ time = (hit\ time + miss\ rate) \times corresponding\ access\ times$$

If a "cache miss" occurs, the next level of memory must be queried for the data or instructions.

The CPU – Microarchitecture View

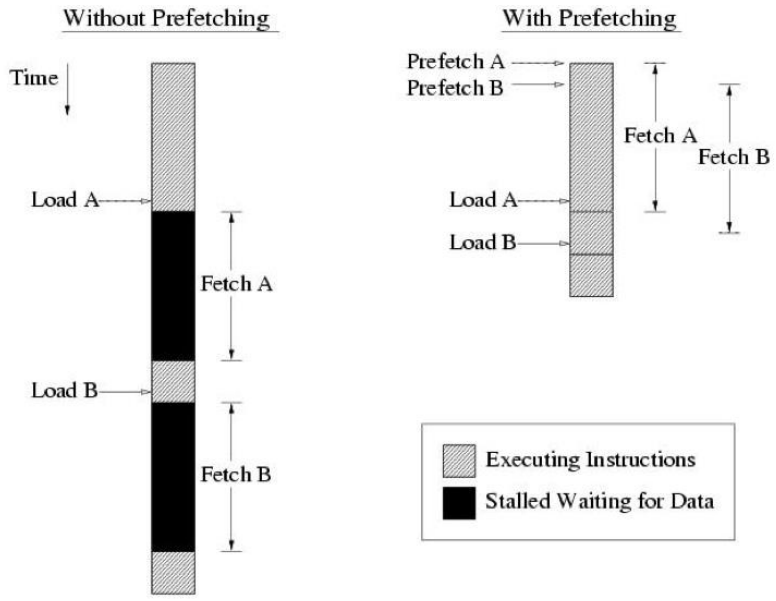
Improving Performance – Pre-fetch Buffers

Definition

Cache pre-fetching is a technique used by computer processors to boost execution performance by fetching instructions or data from their original storage in slower memory, such as main memory, to a faster local memory, such as cache, before it is required.

How it works

Diagram: Comparison



Pre-fetched data and instructions are stored in cache. Pre-fetching can be performed when a piece of data is known to be required soon by the CPU.

Evaluation

Advantages	Disadvantages
Helps to improve performance as it tolerates memory latency as the CPU does not have to wait for the data to be fetched from memory.	**The cache may need to be flushed if a "cache miss" occurs**; this can happen if a branch instruction is present as these are outside of the current program flow, the cache will often be flushed and the data and/or instructions from the code where the branch has taken the program will have to be fetched from main memory.

The CPU – Microarchitecture View

Improving Performance – Pipelining

Definition

Pipelining is a technique which can be used to overlap multiple instructions during the fetch-decode-execute cycle so that different stages of the cycle can be completed in one cycle.

How it works

Without pipelining, the CPU would complete the fetch-decode-execute cycle for instructions sequentially.

	Fetch	Decode	Execute
Processor Cycle 1	Instruction 1		
Processor Cycle 2		Instruction 1	
Processor Cycle 3			Instruction 1
Processor Cycle 4	Instruction 2		
Processor Cycle 5		Instruction 2	
Processor Cycle 6			Instruction 2

With pipelining, the CPU can schedule instructions so that stages of the fetch-decode-execute cycle can be completed in the same CPU cycle.

	Fetch	Decode	Execute
Processor Cycle 1	Instruction 1		
Processor Cycle 2	Instruction 2	Instruction 1	
Processor Cycle 3	Instruction 3	Instruction 2	Instruction 1
Processor Cycle 4	Instruction 4	Instruction 3	Instruction 2
Processor Cycle 5	Instruction 5	Instruction 4	Instruction 3
Processor Cycle 6	Instruction 6	Instruction 5	Instruction 4

Evaluation

Advantages	Disadvantages
Helps to improve performance as instructions can be processed at a quicker rate.	**Unnecessary processing can occur**; subsequent instructions must be predictable otherwise the incorrect instruction may be fetched and then discarded by flushing the pipeline. For example, a branch operation in an assembly language program may result in the pipeline being flushed.

The CPU – Microarchitecture View

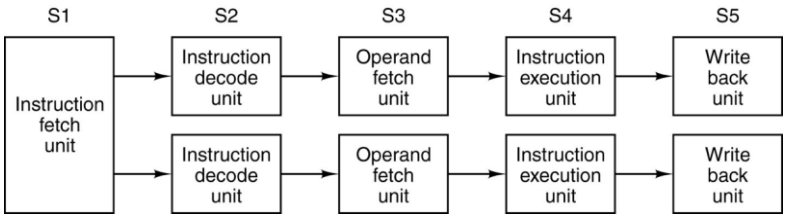
Improving Performance – Superscalar Architectures

Definition

A **superscalar processor** is a CPU that implements a form of parallelism called instruction-level parallelism within a single processor. In contrast to a scalar processor that can execute at most one single instruction per clock cycle, a superscalar processor can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different execution units on the processor.

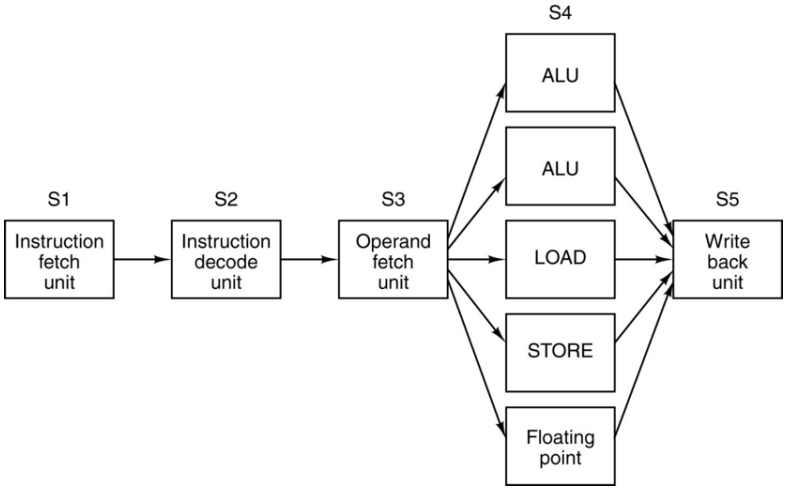
How it works

Diagram: Superscalar Architecture



Superscalar architectures act by having multiple physical hardware units. The CPU has multiple copies of each pipeline stage. Each pipeline is independent of the other pipelines. This allows the CPU executes multiple instructions at once and is effectively a pre-cursor to multi-threading.

Diagram: Superscalar Architecture



Alternatively, multiple functional units can be added as the decode and fetch / store stages are likely to be simpler.

Evaluation

Advantages	Disadvantages
Helps to improve performance as it allows for more throughput (the number of instructions that can be executed in a unit of time) than would otherwise be possible at a given clock rate.	**Increases complexity of the system** however this is a more hardware and cost efficient method than implementing completely separate pipelines.

The CPU – Microarchitecture View

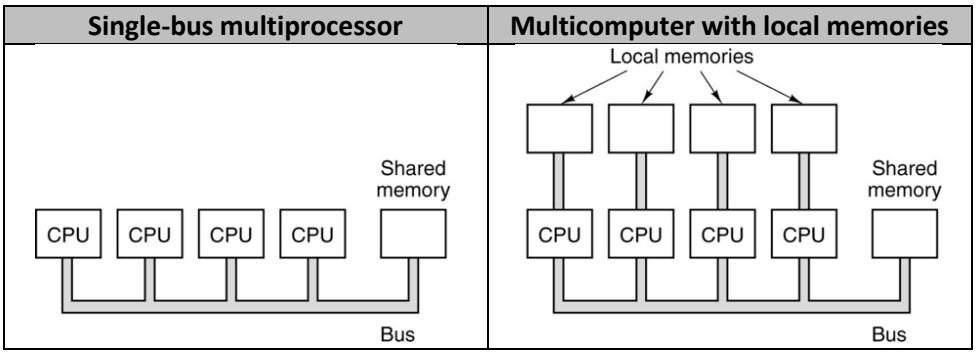
Improving Performance – Multi-Processor Systems

Definition

Multi-processor systems involve is the use of two or more central processing units (CPUs) within a single computer system. The computer system has the ability to support more than one processor and allocate tasks between them.

How it works

Diagram: Comparison



A multicore system can be thought of as a being a natural evolution from superscalar and multithreading systems. Multiple cores are capable of running multiple physical threads at once (like multithreading systems). Multiple cores have multiple functional units / pipelines like superscalar systems.

Evaluation

Advantages	Disadvantages
Helps to improve performance as it allows for more throughput (the number of instructions that can be executed in a unit of time) than would otherwise be possible at a given clock rate.	
Does not require redesign of the processor.	

The CPU – Microarchitecture View

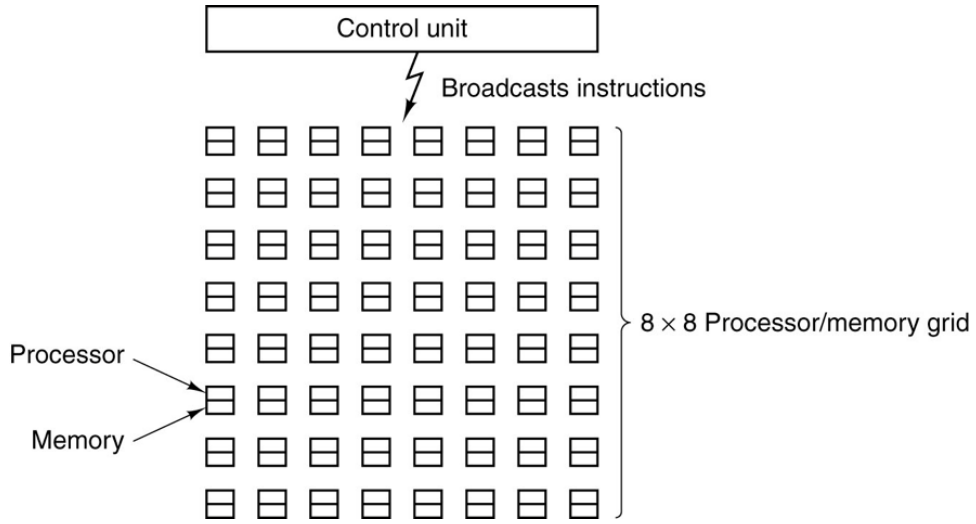
Improving Performance – Manycore Processors

Definition

Manycore processors are specialist multi-core processors designed for a high degree of parallel processing, containing a large number of simpler, independent processor cores (e.g. 10s, 100s, or 1,000s).

How it works

Diagram: Manycore Processory



Evaluation

Advantages	Disadvantages
Helps to improve performance as it allows for more throughput (the number of instructions that can be executed in a unit of time) than would otherwise be possible at a given clock rate.	

Buses and I/O

Types of Buses

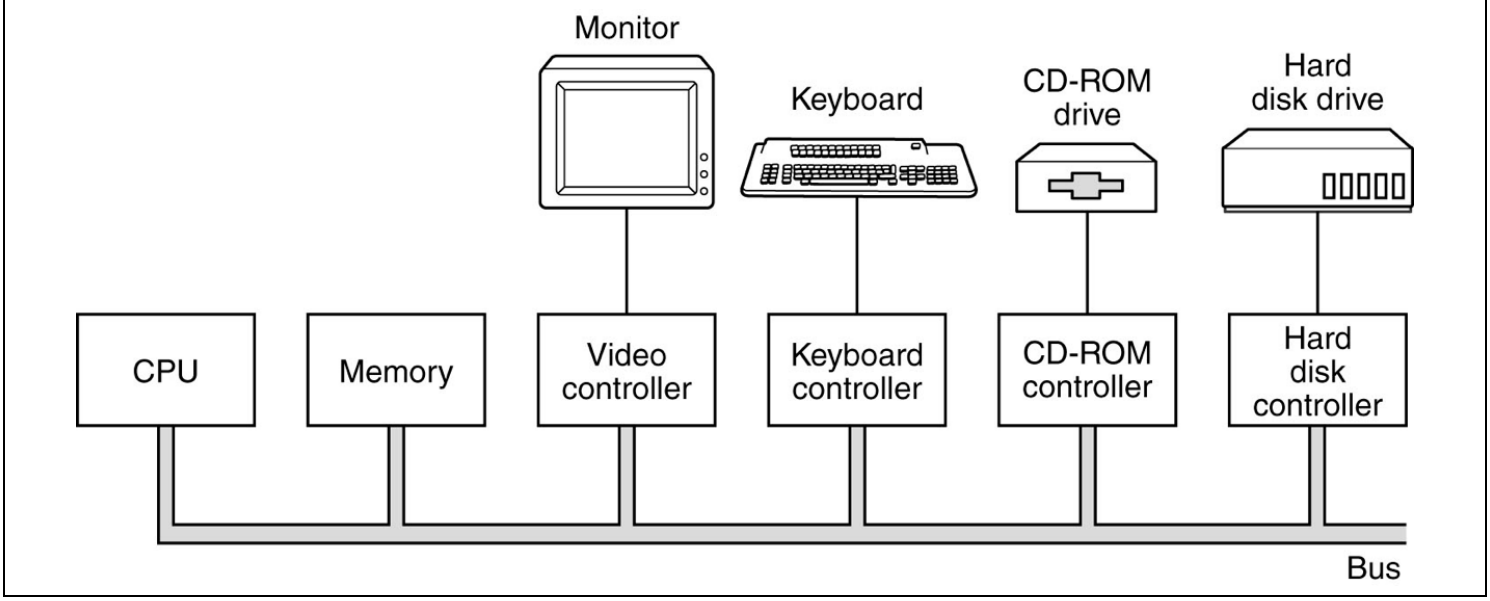
Definition

A **bus** is used for moving data around a computer system.

Single Bus Systems

The simplest computer systems place all devices on a single bus.

Diagram: Single Bus System



The bus acts in one of interrupted, master / slave, or arbitrated configuration.

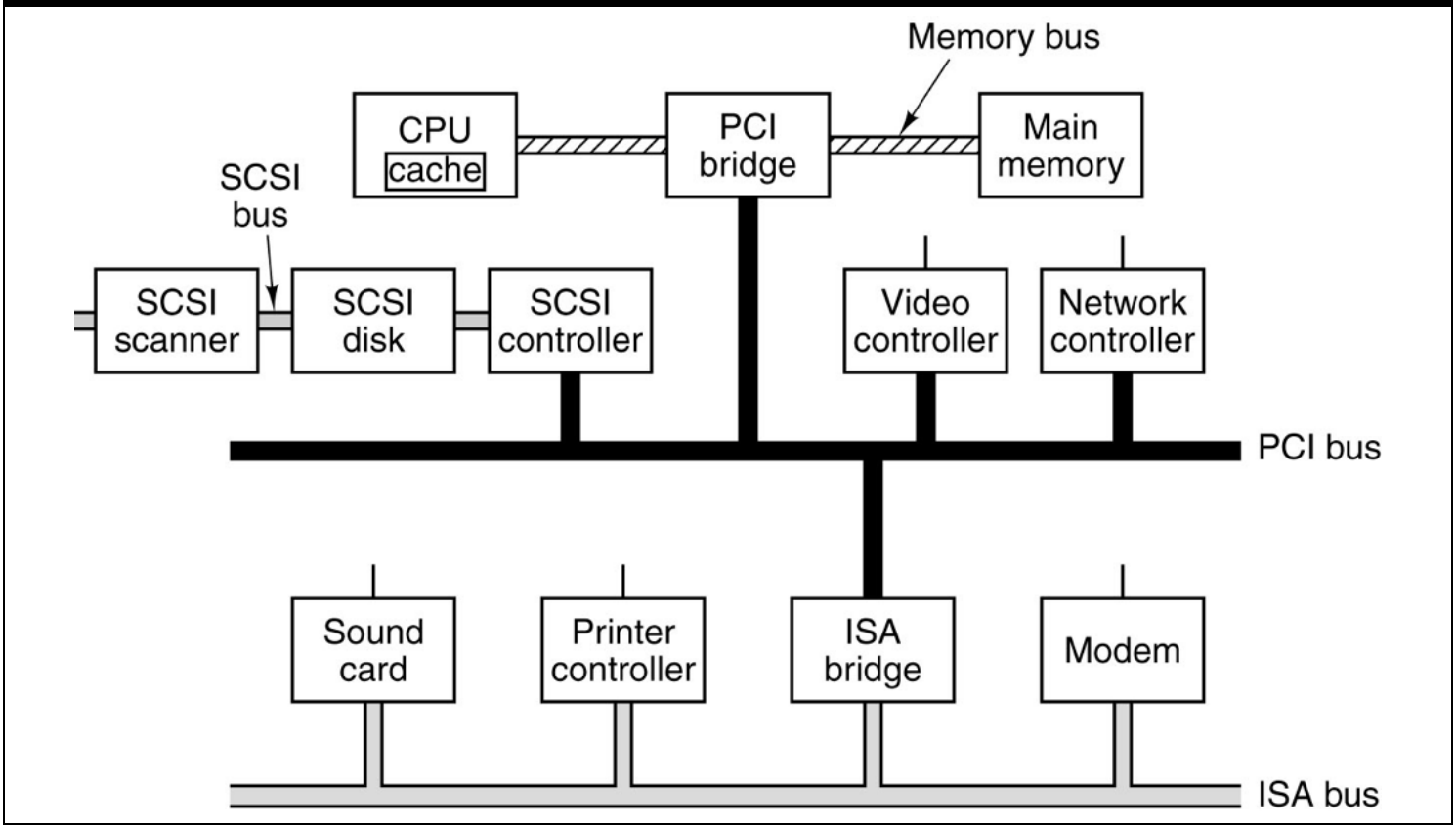
Each memory location has a unique address, only one location can be read from or written to at a given time. The memory location is put on the address bus, a read/write signal (on the read not write line) is then sent and the data in that memory location is put on the data bus.

Buses and I/O

Hierarchical Buses

In reality, we use multiple buses in a hierarchical structure.

Diagram: Hierarchical Bus Configuration



This allows the system to be subdivided and therefore reduces conflict. Controllers and bridges are used in the system to manage the inter-bus communication. PCI standard (32/64 bit parallel bus) has superseded ISA. PCI express has superseded PCI.

Bottlenecks can be caused:

- the CPU is trying to run faster than the speed of the buses;
- memory is a bottleneck; and
- I/O devices are less of a bottleneck as other operations can be completed while waiting for user input or output.

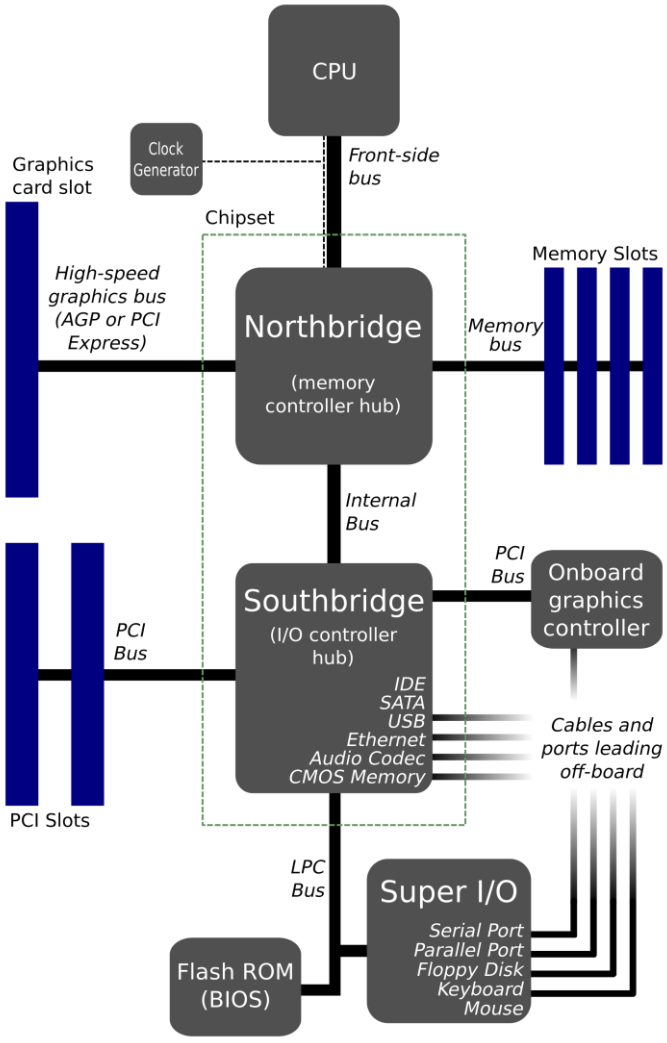
Buses and I/O

Modern Buses

Northbridge and Southbridge

Modern desktop computers are typically based on this model.

Diagram: Bridges



There are two chips in the core logic chipset architecture on a computer system's motherboard:

- Northbridge *(or host bridge)*
  - directly connected to the CPU via the front-side bus (FSB);
  - responsible for tasks that require the highest performance; and
  - paired with the sourthbridge.
- Southbridge *(or I/O controller hub)*
  - implements the slower capabilities of the motherboards; and
  - connected to I/O devices.

These two chips manage communications between the CPU and other parts of the motherboard, and constitute the core logic chipset of the PC motherboard.

The northbridge ties the southbridge to the CPU. Through the use of controller integrated channel circuitry, the northbridge can directly link signals from the I/O units to the CPU for data control and access.

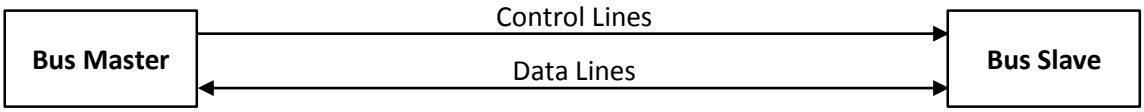
Buses and I/O

Bus Characteristics

Bus Master and Bus Slave

In a bus, there is a bus master and a bus slave.

Diagram: Buses



The master has unidirectional controller over one or more other devices, the other devices act in the role of slaves.

The bus operates using two groups of lines:

- Control lines
  - sends signals for requests from the bus master and acknowledgments from either the bus master or bus slave; and
  - indicates what type of information is on the data lines.
- Data lines
  - carries data, addresses, and complex commands between the bus master and the bus slave.

Bus Transactions

A bus transaction consists of:

- the bus master initiating the request by issuing the command (and address); and
- the bus slave performing the action by receiving the data from the bus master or sending the data to the bus master.

A bus transaction is defined by what operation the transaction performs on memory:

- input – inputs data from the I/O device to the memory; or
- output – outputs data from the memory to the I/O device.

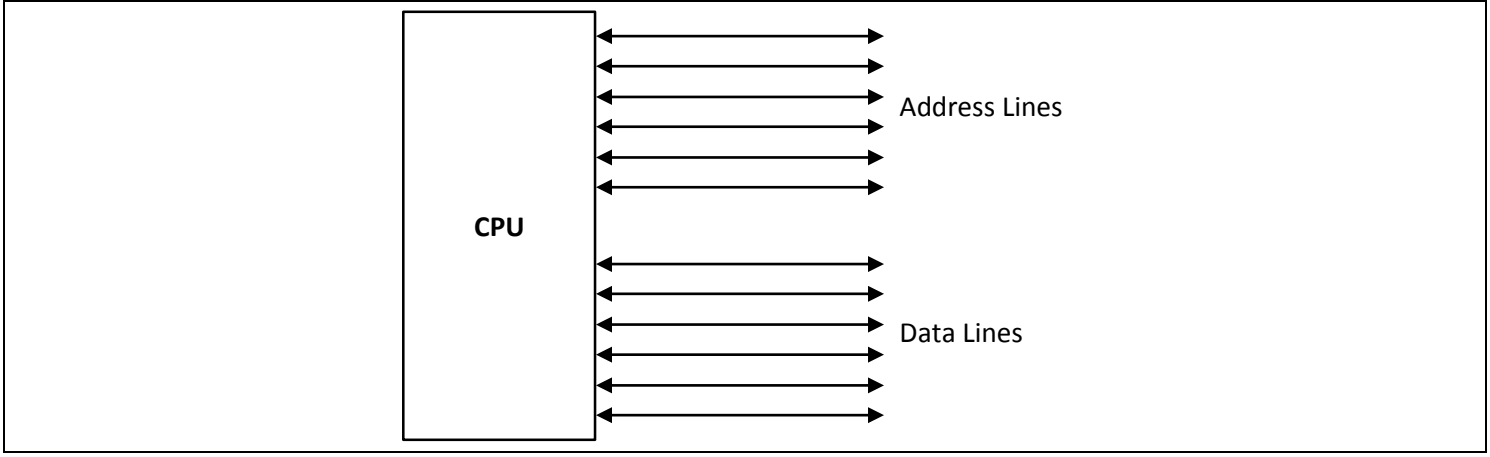
Buses and I/O

Bus Width

Address and Data Lines

Buses are defined by the number of lines they contain.

Diagram: Bus Lines



In the address bus, 2^n tells you how many unique memory locations can be addressed because it tells you how many binary numbers you can represent, where n is the number of lines on the bus. Adding more lines to the address bus allows more memory locations to be addressed.

In the data bus, 2^n tells you how much data can be transferred on a read or write operation. Adding more lines to the data bus helps to improve speed as more instruction data can be transferred and therefore less read/write operations required for a given instruction.

The bus width is typically defined as the largest of either the address lines or the data lines (the control lines are not included).

Increasing Bus Width

How it is achieved

Increasing bus width can be achieved by adding more address lines or data lines.

Evaluation

Advantages	Disadvantages
Improved performance.	**More wires are required on the motherboard** and therefore it is more expensive and may not be suitable for smaller motherboards, such as those found in smartphones.

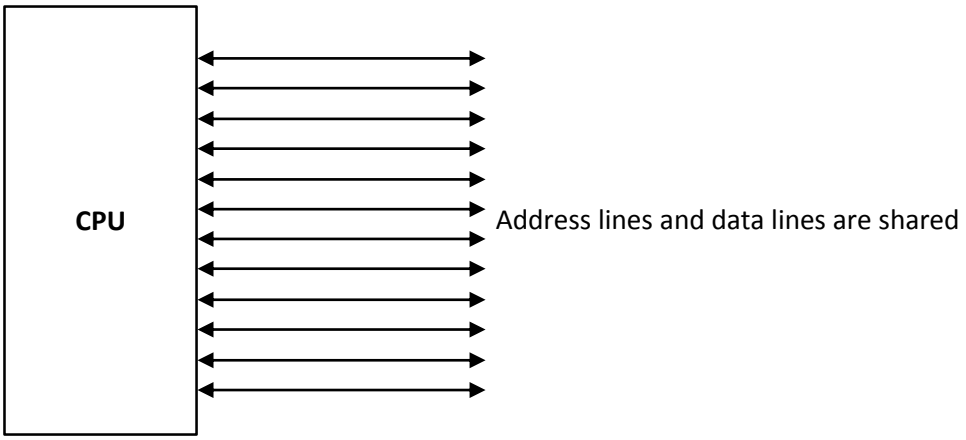
Buses and I/O

Multiplexed Buses

How it is achieved

As an alternative to increasing the bus width, multiplexed buses can be used.

Diagram: Bus Lines



Evaluation

Advantages	Disadvantages
Requires less lines than increasing the bus width.	**Speed can be slower** as the address must be put on then used and removed before the data can be put on the lines.

Buses and I/O

Synchronous and Asynchronous Buses

Comparison

Characteristic	Synchronous Buses	Asynchronous Buses
Clock Signal	Includes a clock signal as one of the control lines.	Un-clocked, devices can run at their own speeds.
Communication	Has a fixed protocol for communication.	Requires a handshaking protocol to let communicating devices sync as different computer systems may run at different speeds and therefore requires additional control lines (ReadReq, Ack, DataRdy).
Data Transfer	Data transfer fixed relative to clock rate.	Sends bits one bit at a time.
Other	During the wait time between clock signals, the registers are setup for the next operation The common clock signal means that it is known when data will be placed on the bus or removed from the bus.	Often external to the computer system, such as Ethernet. There is no common clock as devices connected on a network (such as via Ethernet) may not be running at the same clock speed.

The buses connected to the CPU and memory are synchronous buses.

The buses connected to I/O devices are asynchronous buses.

Evaluation

Bus Type	Advantages	Disadvantages
Synchronous Bus	**Involves very little logic**.	**Every device on the bus must run at the same clock rate**.
	Can run at high speeds.	**Clock skew can occur,** this is phenomenon in which the same sourced clock signal arrives at different components at different times.
Asynchronous Bus	**Can be lengthened without concern of clock skew** as the bus is un-clocked.	**Protocol overhead** as the handshaking protocol is required.

Buses and I/O

Serial and Parallel Buses

Comparison

Characteristic	Serial Bus	Parallel Bus
Lines	Includes two lines; signal and return.	Includes multiple data lines.
Sending Data	Data is sent one bit at a time.	Data is sent with one bit on each data line.
Uses	Universal Serial Bus (USB).	PCI is becoming more commonly implemented as a serial bus rather than a parallel bus. This is because data transfer can be faster using serial transmission rather than parallel transmission.

Buses and I/O

Bus Arbitration

Definition

A **bus arbiter** is a device used in a multi-master bus system to decide which bus master will be allowed to control the bus for each bus cycle. There are processes that are used to decide who gets priority when two things are trying to access a bus. Multiple devices may need to use the bus at the same time so must have a way to arbitrate multiple requests.

How it works

When an I/O device requires access to the CPU, an interrupt is generated. It is may be possible that there are many interrupts present at a given time.

Bus arbitration schemes attempt to balance:

- bus priority – the highest priority device should be serviced first; and
- fairness – even the lowest priority device should never be completely locked out from the bus.

Classes of Bus Arbitration Schemes

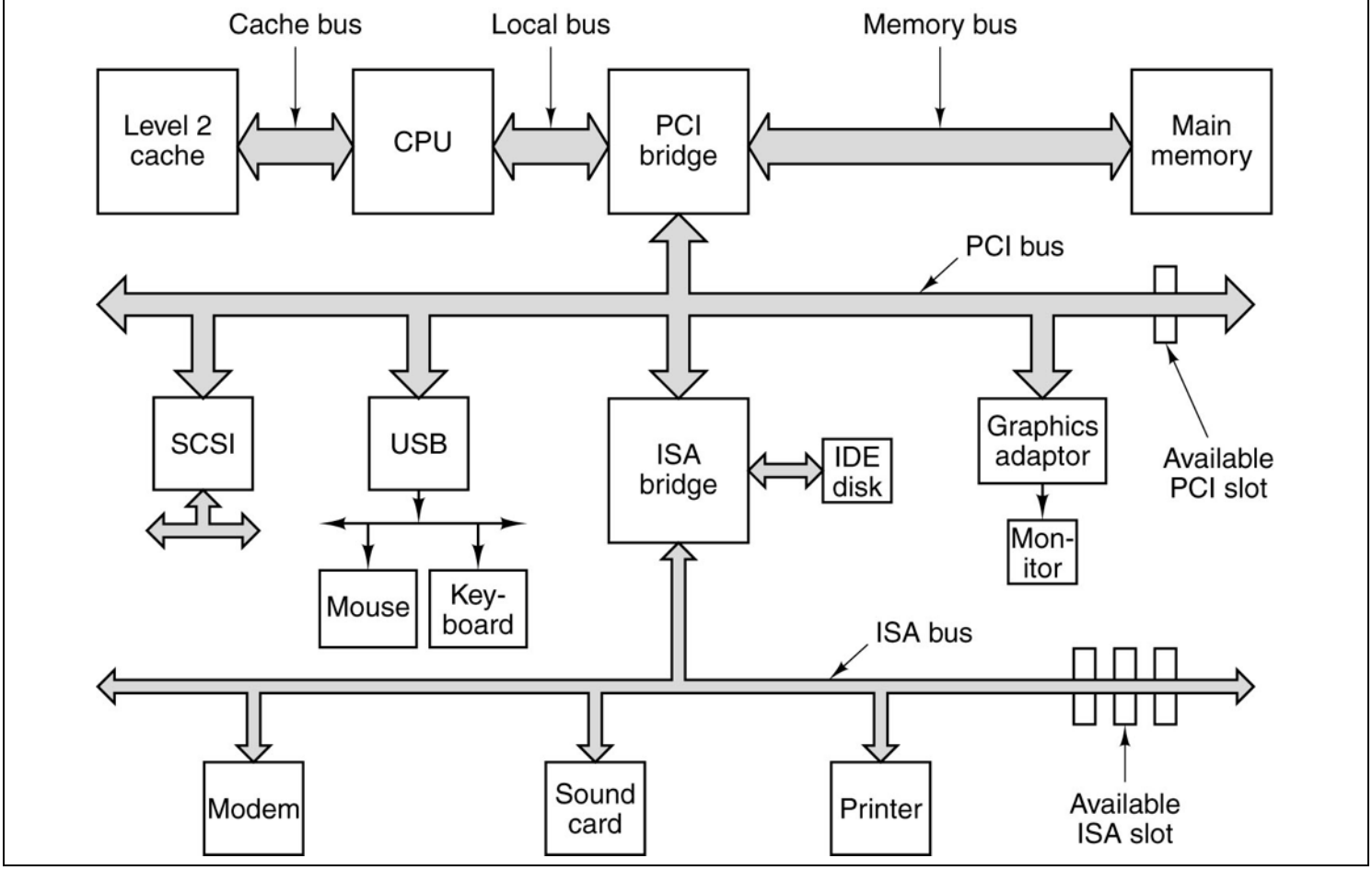
Bus arbitration schemes can be divided into four classes.

- Daisy chain arbitration – There is a hierarchy where the first device will have priority and then subsequent devices along the "daisy chain" will have priority after the device before them. However, this may mean that subsequent devices do not get serviced if the first device is very busy.
- Centralised, parallel arbitration – Each device has its own bus request and bus grant lines. The CPU inspects the priority of the devices and therefore the CPU decides which request to handle first. This allows the CPU to stop a high priority device if it is making too many requests, service another device and then revisit the previous device.
- Distributed arbitration by self-selection – Each device wanting the bus places a code indicating its identity on the bus.
- Distributed arbitration by collision detection – Device uses the bus when it is not busy and if a collision happens, as a result of some other device trying to use the bus, then the device tries again later (such as Ethernet).

Buses and I/O

Generalised PCI Bus Structure

Diagram: PCI Bus Configuration



The PCI bridge acts as the northbridge and the ISA bridge acts as the southbridge.

- CPU is connected with a local bus to the PCI bridge.
- PCI bridge converts between the data transfer standard, size and speed, between the local bus used by the CPU and the PCI bus.

- PCI bus connects devices that require fast speed to the CPU.
- ISA bridge converts between the data transfer standard, size and speed, between the PCI bus and the ISA bus.

- ISA bus connects devices that are relatively slow.

Buses and I/O

Universal Serial Bus (USB)

Definition

A **Universal Serial Bus (USB)** is a common interface that enables communication between devices and a host controller such as a computer system. It connects peripheral devices such as digital cameras, mice, keyboards, printers, scanners, media devices, external hard drives and flash drives.

How it works

USB allows 127 devices (8 bit address) to share one bus. It utilises stop and wait flow control. To achieve “full speed”, the length must be no longer than 5 metres.

Table: USB Pins

Pin Number	Role
1	+5V
2	-Data
3	+Data
4	Gnd *(ground)*

Speeds

Different USB versions permit different speeds.

- USB 1.0 –
  - 12Mbps full.
  - 1.5Mbps slow.
- USB 2.0 –
  - Up to 480Mbps transfer speed.
  - Backwards compatible connector.
- USB 3.0 –
  - Up to 5Gbps transfer speed.
  - Backwards compatible, extended connector to allow legacy cables.

Buses and I/O

Handling I/O

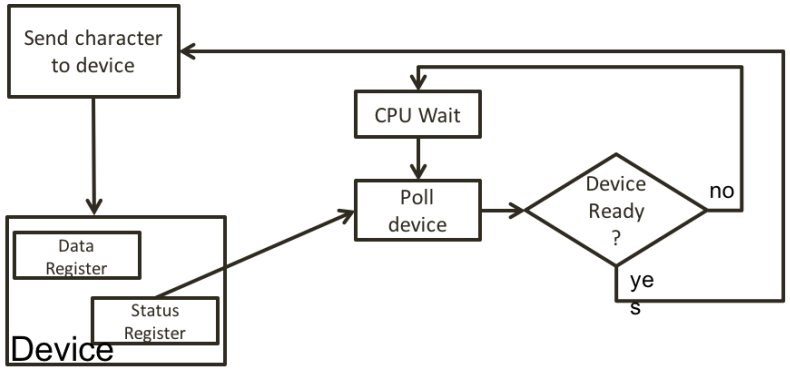
I/O Devices

I/O occurs when the computer system is exchanging data with devices outside of the computer system. These devices are slower than the internal speed of the computer system and therefore a method is required to access these devices only when they are ready to be accessed otherwise they would act as a large bottleneck to the CPU.

Polling

Devices can be written to or read from using busy-wait I/O.

Diagram: Polling



- The CPU checks the device status register to see if operation has completed.
- Once completed, next operation will be performed.

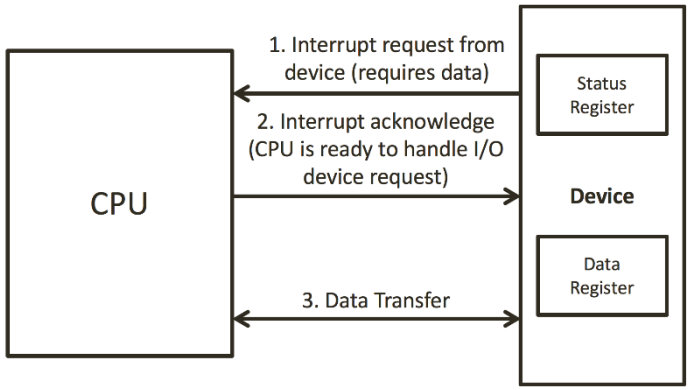
This method not efficient as it involves the process of constantly asking the device if it is ready when most of the time the device will not be ready and therefore it is a wasted operation.

Buses and I/O

Interrupts

Devices can be written to or read from when then generate an interrupt signal.

Diagram: Interrupts



This is achieved using an interrupt service routine (ISR), which is a program that makes use of a stack to manage interrupts.

An interrupt is generated in the following way:

- an I/O device activates its interrupt line, containing the interrupt and interrupt ID, when it is ready to send information to the CPU; and
- when the CPU finishes the current instruction, it checks the interrupt lines of the I/O devices.

If an interrupt line is active:

- CPU stops the current process;
- state of the current process is preserved by pushing the contents of the program counter (PC), the other registers and the return address on to the stack;
- CPU uses the interrupt ID sent from the I/O device to determine the correct interrupt service routine (ISR) to run; and
- the start address for the ISR is loaded into the program counter (PC) in order to change the position of the CPU from the current program to the beginning of the ISR.

An **Interrupt Service Routine (ISR)** contains the operations that are performed to deal with an interrupt. Different types of interrupts can have different routines.

Once the ISR reaches its return address:

- contents of the program counter, the other registers and the return address from the original program are popped from the stack and loaded back in to the respective locations in the CPU; and
- CPU has now been returned to the state before the interrupt occurred and can continue processing the original program.

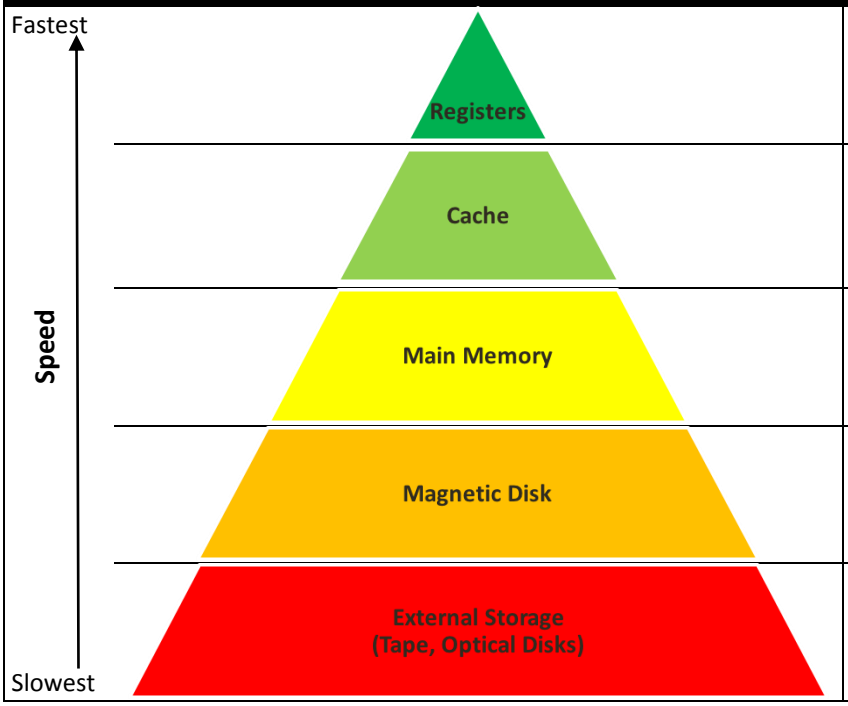
This method is more efficient as CPU can continue performing other tasks.

Memory

Hierarchy

Memory Hierarchy

Diagram: Memory Hierarchy



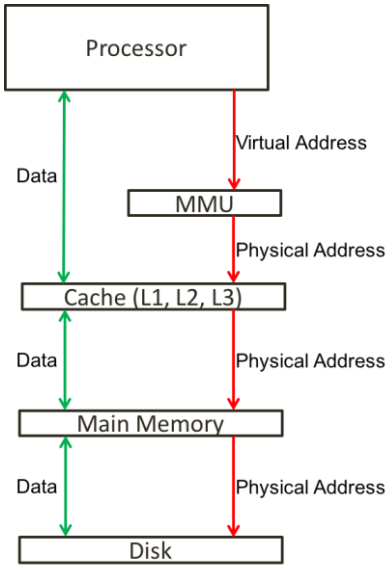
Memory type:	On CPU
Access time:	1cc
Memory type:	Hierarchical storage designed to alleviate access time issues to lower levels
Access time:	L1 – 4cc, L2 – 10cc, L3 – 75cc
Memory type:	Main working memory
Access time:	100cc or more
Memory type:	Typical non-volatile storage
Access time:	1000cc or more
Memory type:	Additional storage devices
Access time:	10000cc or more

cc = clock cycle

How it works

Memory acts as a hierarchy.

Diagram: Data Flow



Memory

When data or instructions must be fetched from memory:

- the CPU requests the data or instructions from the virtual address (VA) location;
- the memory management unit (MMU) translates the virtual address (VA) to a physical address (PA) in memory;
- cache memories determine if they have a copy, if so they return it otherwise request from main memory;
- main memory determines if it has a copy of the data, otherwise requests it from disk;
- disk maps the physical address (PA) to the disk address and returns the data or instructions data up stack.

If the memory management unit (MMU) is performing efficiently, it should not have to visit memory past the cache level.

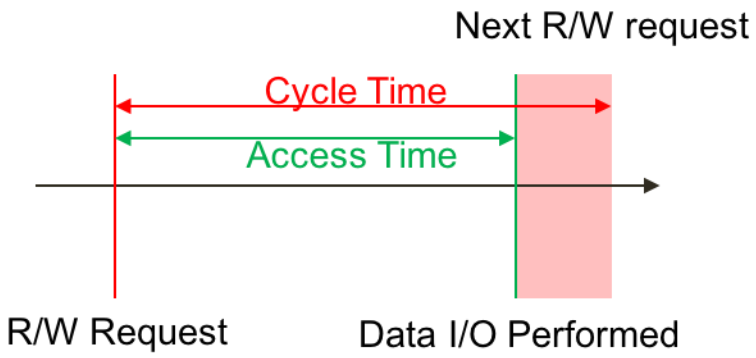
Memory

Memory Timing

Timing Numbers

Memory has two important timing numbers, access time and cycle time.

Diagram: Timing



- Access time – Time for operation to be performed.
- Cycle time – Time before next operation can begin.

Timing can also be defined in clock cycles. This means that a faster processor has effectively faster registers however, the latency to other memories seems longer.

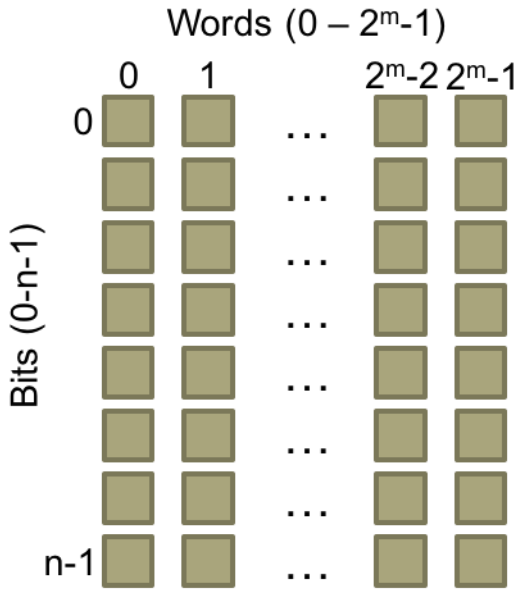
Memory

Building Memory Circuits

Memory Circuits

Memory is built in grids.

Diagram: Memory Circuits



- Each grid is one WORD long, composed of n bits.
- The number of physically addressable words of memory is defined by the address bus width of m bits: $2m - 1$.

Bandwidth is defined as the number of number of bytes written / read per second. This can be increased by using faster memory (SRAM vs. DRAM), or wider memory (64-bit / 128-bit vs. 32-bit processor).

The number of memory locations is defined by the number of wires in the address bus. Therefore, the more memory locations require a wider address bus.

Memory

Types of Memory

Registers

What are registers?

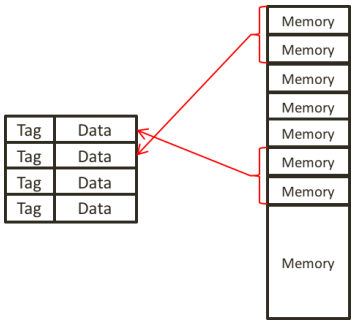
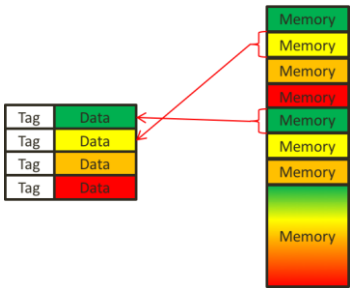
See page 47/48.

Cache Memory

What is cache memory?

See page 54.

Types of cache memory

Fully Associative Cache	Direct-Mapped Cache	Set-Associative Cache
		
• No restriction on memory to cache mappings. • Associative search of tags is 'expensive'. • Feasible for small caches only (L0 / L1).	• Each memory block can only be mapped into one cache line. • Associative search is not needed. • Miss rate can increase.	• N- way associative, this means memory blocks are mapped to a set of N lines of the cache (e.g. 4 entries per cache line). • Cheaper than fully associative cache. • Lower miss rate than direct mapped cache.

Level 1 and Level 2 cache are likely to use fully associative or direct mapped.

Why cache memory is effective

Cache memory works from the premises of program locality and data locality. When a small reference to memory is used, the data and instructions surrounding that in memory reference are loaded into the cache as well as they may be needed in subsequent operations.

Memory

It is possible that a cached value has changed while main memory retains the old value. Therefore, it is necessary to maintain data continuity between cache memory and main memory.

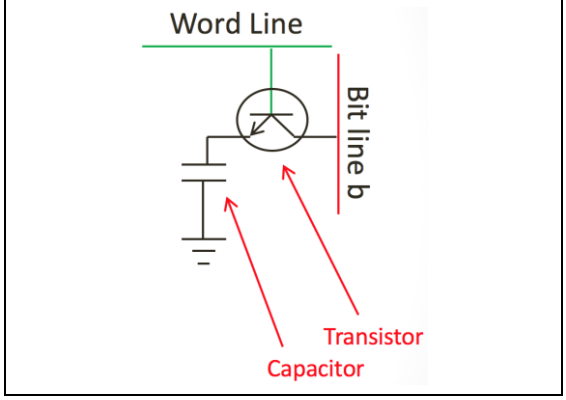
There are two methods of achieving data continuity.

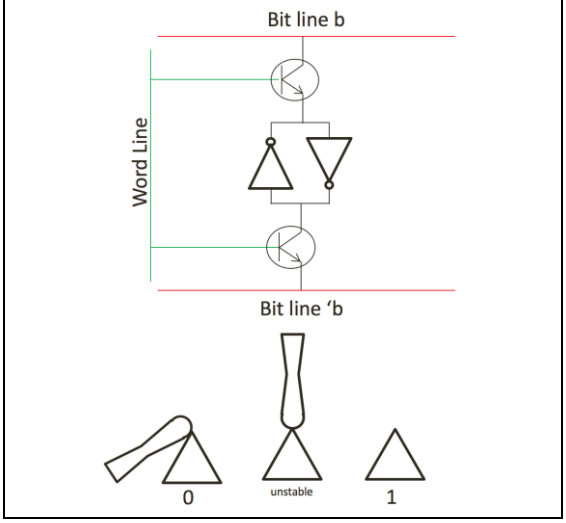
Method	Write Through		Write Back / Deferred	
How it works	Cached value is immediately written back to main memory (at same time).		Cached value is only written back when the original values in main memory are about to be modified. This means that write operations only occur if a subsequent instruction is about to change the value in main memory; the old change, performed in cache memory, is written back to main memory before performing the new change.	
Evaluation	**Advantages**	**Disadvantages**	**Advantages**	**Disadvantages**
		Introduces a significant amount of write traffic.	**Less write traffic than the write through method** as less write operations are required.	**More complex to implement** as it requires a dirty flag to keep track of when to put data or instructions back into memory.

Memory

DRAM and SRAM Comparison

Random Access Memory (RAM) is a fast and volatile storage location that contains data and instructions frequently being used by the CPU.

<table>
<tr><th>Type of Memory</th><th colspan="2">DRAM</th><th colspan="2">SRAM</th></tr>
<tr><td>Definition</td><td colspan="2">Dynamic Random Access Memory (DRAM) is a common type of random access memory (RAM) used in personal computers (PCs), workstations and servers.</td><td colspan="2">Static Random Access Memory (SRAM) a type of memory chip which is faster than dynamic memory.</td></tr>
<tr><td>How it works</td><td colspan="2">DRAM stores bits in cells consisting of a capacitor and a transistor.
Diagram: DRAM
 

The representation of bits is based on the charge of the capacitor:
• charged – one (1); or
• discharged – zero (0).
When reading DRAM, the current/voltage is drawn off the capacitor to discharge it.</td><td colspan="2">SRAM stores bits using transistors and inverters and could use flip-flops.
Diagram: SRAM
 

The representation of bits is based on the state:
• stable – Zero (0) and one (1); or
• unstable – intermediate stage will move towards 0 or 1.</td></tr>
<tr><td>Speed</td><td colspan="2">~10x slower than SRAM.</td><td colspan="2">~10x faster than DRAM.</td></tr>
<tr><td rowspan="4">Evaluation</td><td>Advantages</td><td>Disadvantages</td><td>Advantages</td><td>Disadvantages</td></tr>
<tr><td>Smaller and therefore can be fit into smaller form factors as more memory can be fit in the same silicon.</td><td>The capacitors can discharge and therefore its value will have been lost therefore refreshing is required; must have circuitry capable of maintaining the data by keeping the capacitors charged when the CPU is not accessing the data.</td><td>Does not need to be refreshed as there are no capacitors used.</td><td>Higher power consumption than DRAM as each bit is always on.</td></tr>
<tr><td>Lower power consumption than SRAM.</td><td>Slower than SRAM (10’s of nanoseconds access time).</td><td></td><td>Expensive to manufacture as the memory cells take up a lot of space.</td></tr>
<tr><td>Low cost.</td><td></td><td></td><td></td></tr>
</table>

Memory

Error Correction

Definition

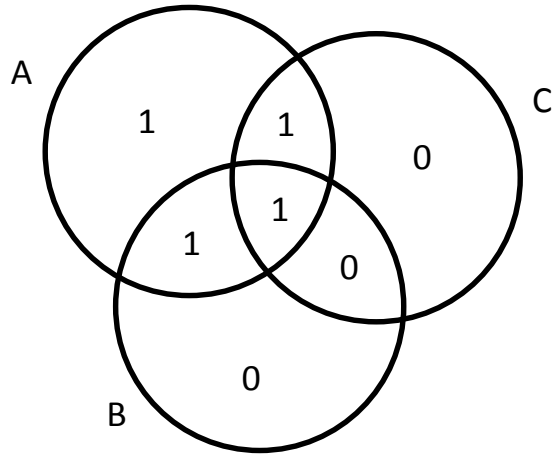
Error-correcting code memory (ECC memory) is a type of computer data storage that can detect and correct the most common kinds of internal data corruption.

How it works

Memories often have an error correction code attached to them to enable them to correct errors. A common scheme is a parity check bit.

Parity Check Bit

A **parity check bit** is a bit added to a string of binary code to ensure that the total number of 1-bits in the string is even or odd, thereby identifying data corruption.

Example: Parity Check Bit	
Diagram: Parity Check Bit using Even Scheme	**Explanation**
	1) The binary number 1110 is stored in the intersections between A, B and C in alphabetical order. 2) Parity information can be stored in regions A, B and C. 3) The parity information can follow either the: • odd scheme – that sum of regions is an odd number; or • even scheme – that sum of regions is an even number. If the sum of the regions is not in accordance with the odd or even scheme, it shows that an error has occurred and the data is likely corrupted.

Where is it used?

ECC memory is often used in servers or other computer systems where data corruption cannot be tolerated under any circumstances, such as for scientific or financial computing. While non-ECC memory is common is personal computers (PCs).

Evaluation

Advantages	Disadvantages
Helps to protect high-value data by ensuring that its contents is correct.	**More expensive** than non-ECC memory.

Memory

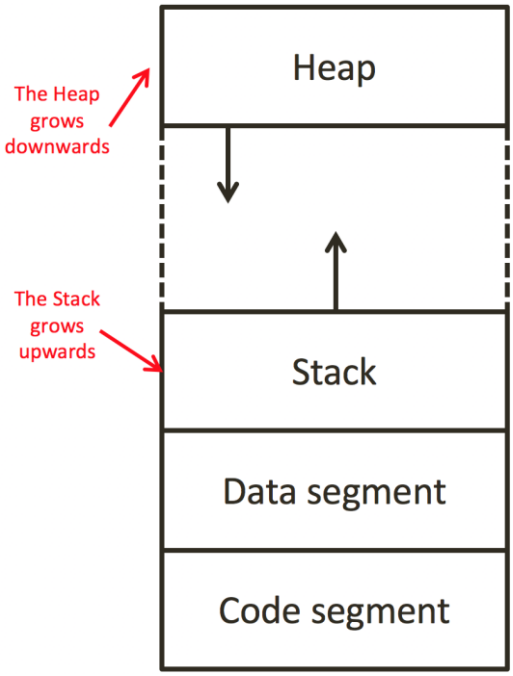
Role of Memory in an Operating System

Running a Program

Application programs will be executed via the operating system layer. Memory is required to run these programs.

For this purpose, memory will be divided into sections:

Diagram: Memory



Stack

Definition

A **stack** is a "Last In, First Out" (LIFO) data structure, where the last item added is the first item to be accessed.

Use in Programs

The stack occupies a static and dedicated portion of memory and allows the operating system to buffer a stream of objects:

- a push method is used to add items to the stack; and
- a pop method is used to read and remove the last item added from the stack.

The stack is used if a program wishes to perform a common task, such as a function call:

- the address of the instruction that is currently being performed and the variables currently in use are pushed on to the stack;
- the program counter (PC) is loaded with the location of the function call; and
- once the function has completed, such as a return statement, the contents of the stack must be retrieved:
    - the address of the previous instruction is popped from the stack and loaded into the program counter (PC); and
    - the variables are popped from the stack and loaded back in to the registers.

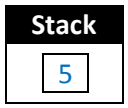
Memory

Use in Calculating Sums

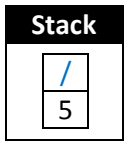
The stack can be used to help perform calculations.

Example: Calculating Sums

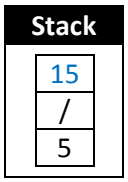
Using the stack to perform the operation $\frac{15+(13\times8)}{5}$.



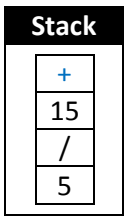
Push 5 on to the stack.



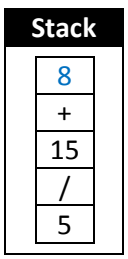
Push the division operation on to the stack.



Push 15 on to the stack.

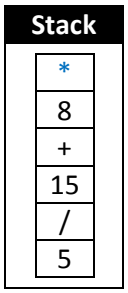


Push the addition operation on to the stack.

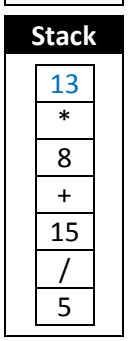


Push 8 on to the stack.

Push all items from the operation to the stack.

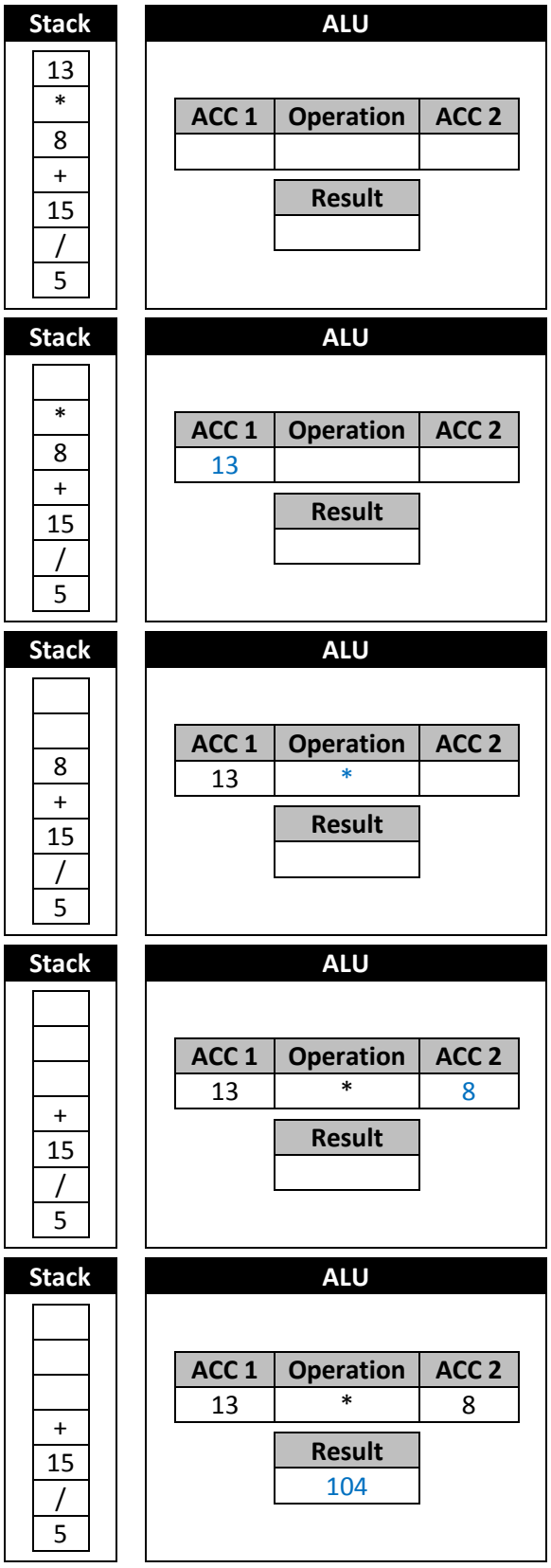


Push the multiplication operator on to the stack.



Push 13 on to the stack.

Memory



The contents of the stack will be popped off in order to perform the operation in the ALU.

13 is popped off the stack and loaded into accumulator 1.

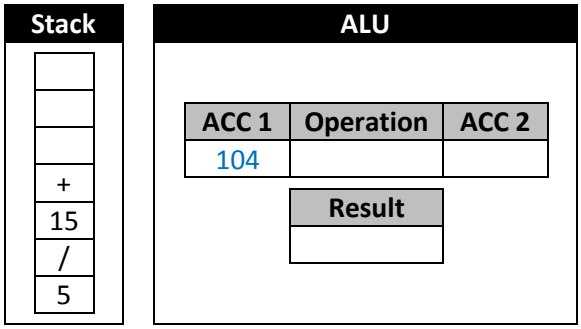
The multiplication operator is popped off the stack and loaded as an operation.

Perform arithmetic.

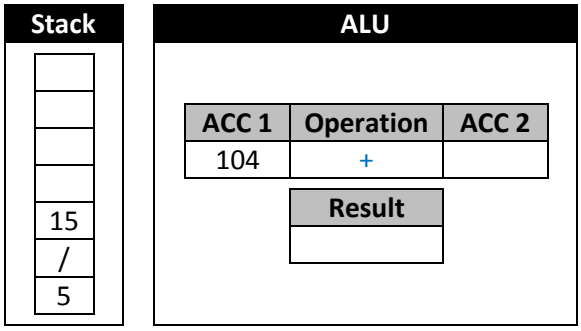
8 is popped off the stack and loaded into accumulator 2.

The arithmetic logic unit calculates the operation 13×8 giving a result of 104.

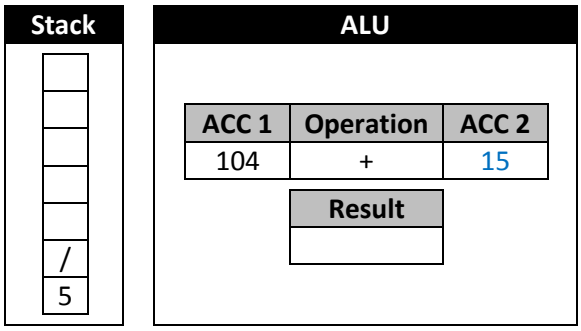
Memory



The previous result, 104, is loaded into accumulator 1.

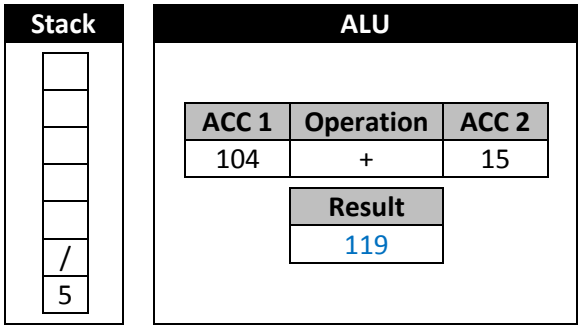


The addition operator is popped off the stack and loaded as an operation.

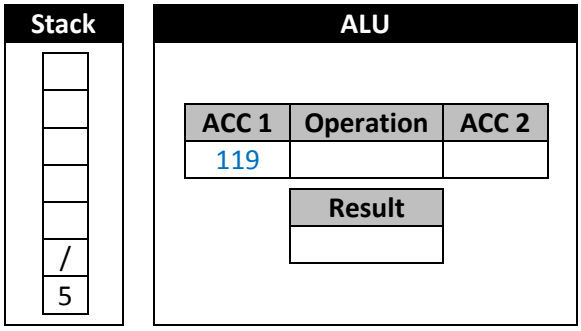


15 is popped off the stack and loaded into accumulator 2.

Perform arithmetic (cont.).



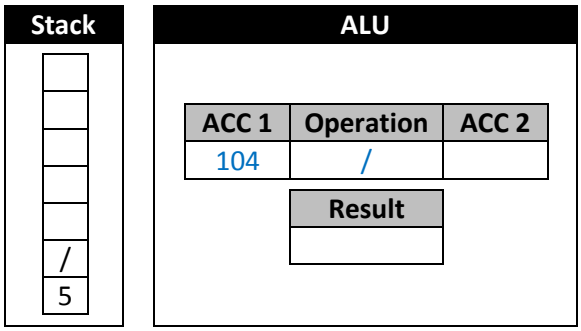
The arithmetic logic unit calculates the operation $104 + 15$ giving a result of 119.



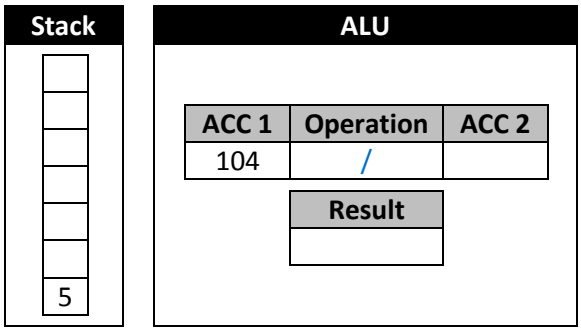
The previous result, 119, is loaded into accumulator 1.

Memory

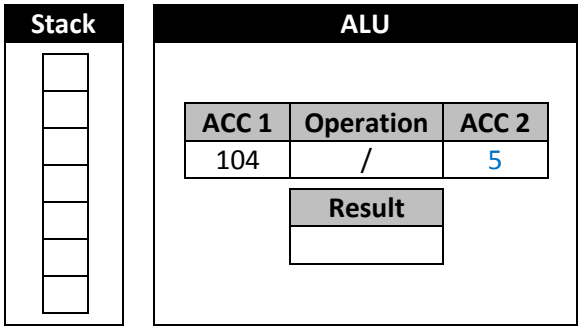
Perform arithmetic (cont.).



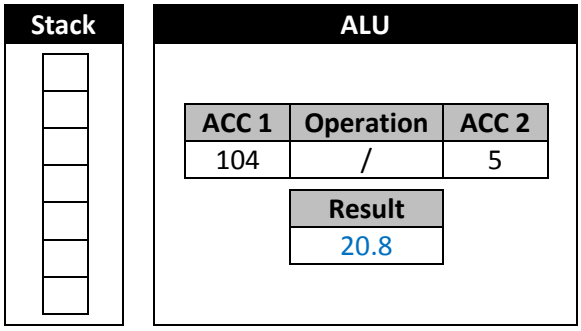
The previous result, 104, is loaded into accumulator 1.



The division operator is popped off the stack and loaded as an operation.



5 is popped off the stack and loaded into accumulator 2.



The arithmetic logic unit calculates the operation $\frac{104}{5}$ giving a result of 20.8.

The stack is now empty meaning that there are no further components to the operation. The final result of the equation is 20.8.

Memory

Heap

Definition

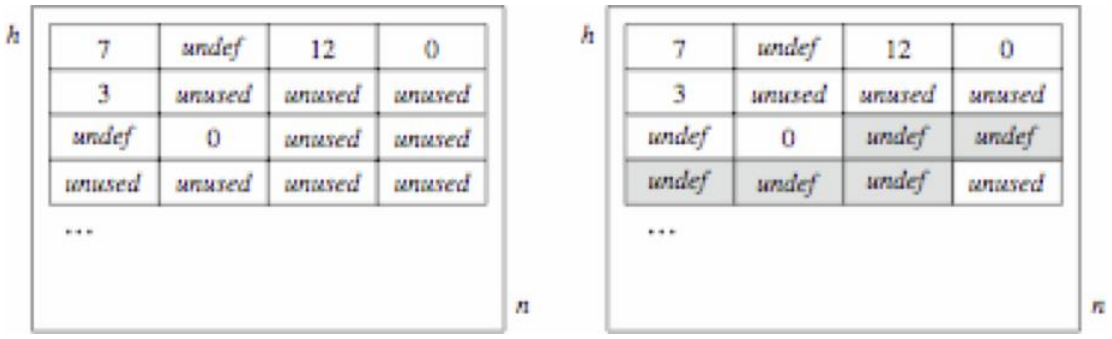
The **heap** is a pre-reserved and dynamic area of main memory that a program process can use to store data.

Use in Programs

A program may store structures in the stack:

- structures whose size varies dynamically, such as variable length arrays or strings;
- structures that are allocated dynamically, such as records in a linked list; and
- structures created by a function call that must be kept after the call returns.

Diagram: Heap



Issues

Allocation and free space management must be managed as data can be taken from any point in the heap.
De-allocation and garbage collection must also be managed in order to clean up the heap of unreferenced or dead objects.

Memory

Secondary Memory

Definition

Secondary memory is a non-volatile and persistent store of data, which is used for items such as user files, programs and the operating system.

Why is secondary memory required?

Main memory is not suitable for long term storage as:

- it is volatile, meaning that the contents are lost when power is removed;
- space is limited as main memory is relatively expensive; and
- information is not normally shared between different running programs.

Secondary memory solves this problem as:

- it is non-volatile, meaning that the contents are retained when power is removed;
- space is larger allowing the storage of very large amounts of data; and
- information can be accessed by multiple programs concurrently.

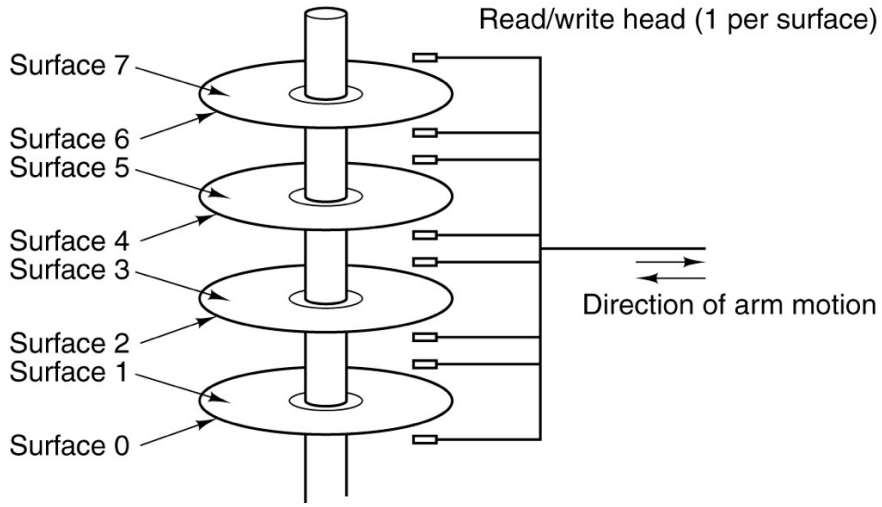
Hard Disk Drives (HDD)

Definition

A **Hard Disk Drive (HDD)** is a type of secondary memory that stores data on spinning platters whereby the data is accessed through a disk head.

How it works

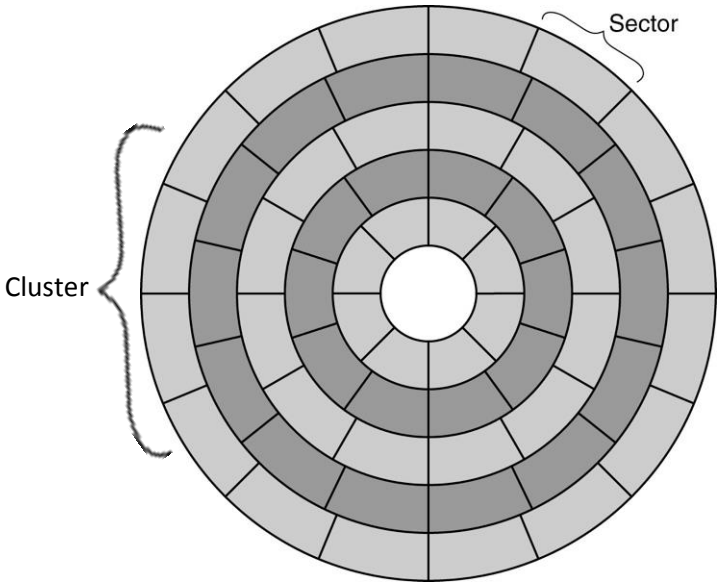
Diagram: HDD Surfaces



A HDD contains multiple surfaces comprised of disk platters, each platter has a read/write head.

Memory

Diagram: HDD Disk Platter



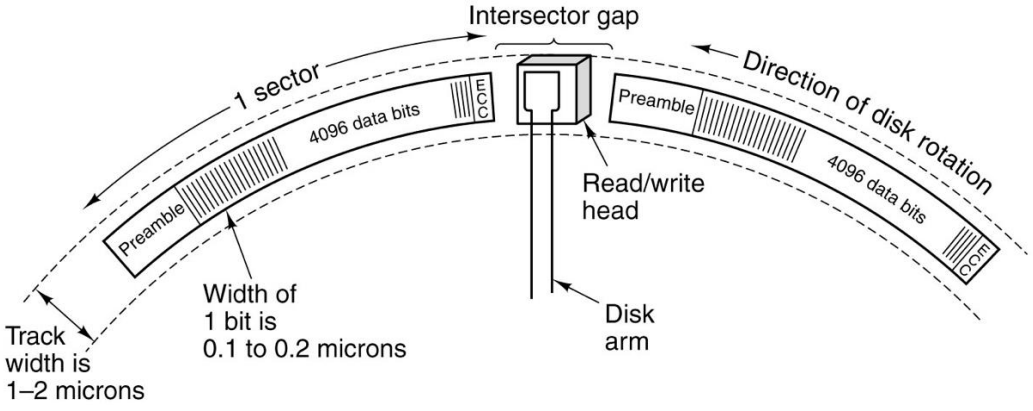
In the disk platter:

- the surface is split into tracks, shown by the concentric circles.
- each track is made out of a number of sectors, which each have a finite size (e.g. 1K, 2K, 4K or 8K); and
- a cluster, usually between 2KB and 32KB, can have a number of sectors associated with them (e.g. 8 sectors).

The platters are coated with magnetic material. The magnetic material contains ferrous particles, containing iron, which are polarised to become either a north or south state; these states can correspond to either a 0 or a 1.

Platters spin below the read/write head. The read/write head must move to the track containing the data. The sector then passes below the read/write head and it can be read or the head must magnetize the surface to write data.

Diagram: Portion of Disk Track



The diagram shows two sectors on a portion of the disk track.

The sector comprises of the following parts:

- preamble – setup the sector (formatting a disk sets the preamble to contain cylinder and sector numbers) and synchronise the read/write head;
- 4096 data bits – used to store the data in the sector; and
- ECC – error correcting code to determine whether the data has been corrupted – this is necessary as the read/write head is floating nanometres above the disk and therefore it can be possible that movement can cause the read/write head to "crash" against the platter and destroy data.

File Systems

File System Formats

Definition

A **file system format** is a standard way that information is encoded for storage in a file on a computer system. It specifies how bits are used to encode information in a digital storage medium.

Types of File System Formats

File System Format	FAT	FAT32	NTFS
Characteristics	A **file allocation table (FAT)** is a file system developed for hard drives that originally used 12 or 16 bits for each cluster entry into the file allocation table. It is used by the operating system (OS) to manage files on hard drives and other computer systems.	**File allocation table 32 (FAT32)** increases the number of bits used to address clusters and also reduces the size of each cluster. It is used by the operating system (OS) to manage files on hard drives and other computer systems.	**New Technology File System (NTFS)** is a proprietary file system developed by Microsoft. It has improved support for metadata and advanced data structures to improve performance, reliability, and disk space use.
Use	Becoming historical	Still regularly used	Necessary for servers
Compatibility	DOS and Windows	Windows 95 R2 and later	Windows NT 4.0 and later
Maximum Size	2GB or 4GB (depending on OS version)	32GB (limit imposed by Microsoft)	16EB
Features	• No file-level or folder-level security.	• No file-level or folder-level security.	• File-level or folder-level security. • Compression • Encryption • Disk quotas • Mounted volumes

File Systems

Files and File Names

Definition

Files are used to store information on secondary memory.

File Types

File types include:

- binary executables – a program;
- text files – stores plain text; and
- media files – such as images or videos.

File Names

From 1981 to 1993 (advent of Windows NT), file names followed an “8.3” format, such as command.com. However, since Windows 95 and Win NT file names are no longer restricted; names can be up to 255 characters in length.

File Systems

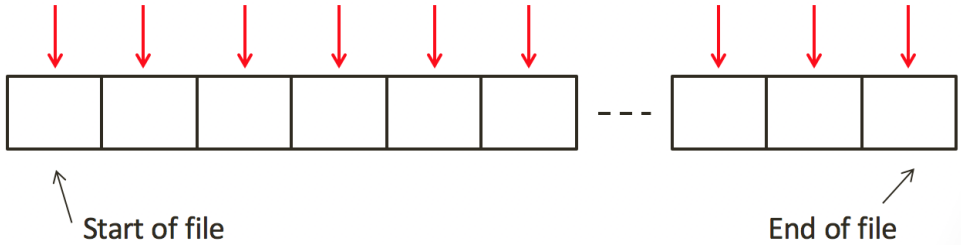
File Access

Why is file access necessary?

Data that is stored on a Hard Disk Drive (HDD) needs to be read off the disk and stored in main memory.

Sequential Access

Diagram: Sequential Access



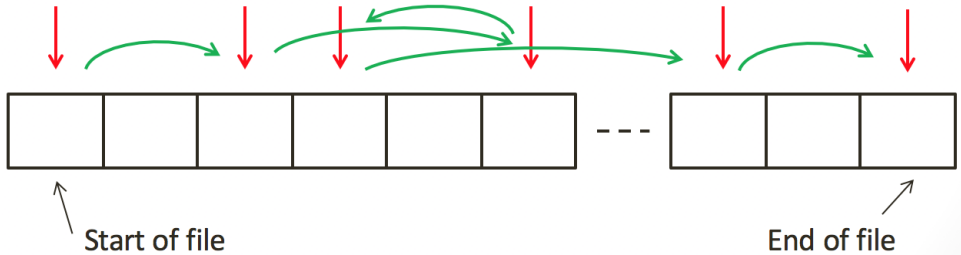
In sequential access, files are stored sequentially.

The read/write head moves to first block of file and reads every byte until the end of file has been reached such that data is read sequentially and in order. The head can be re-wound to beginning of file.

This method of file access is similar to accessing a file on a magnetic tape. This is not very efficient as changes to files require the entire file to be accessed before reaching the portion where the change has occurred.

Random Access

Diagram: Random Access



In random access, files are stored as blocks of data.

Random access uses the following operations:

- seek – allows the read/write head to move to a different position; and
- read – allows the read/write head to read the data.

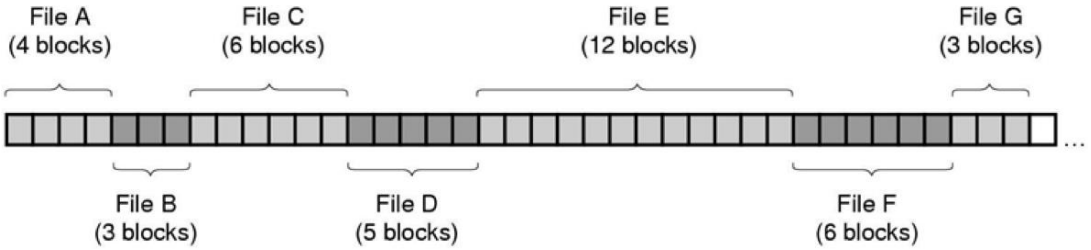
This method of file access is more efficient than sequential access and is commonly used.

File Systems

File Allocation Strategies

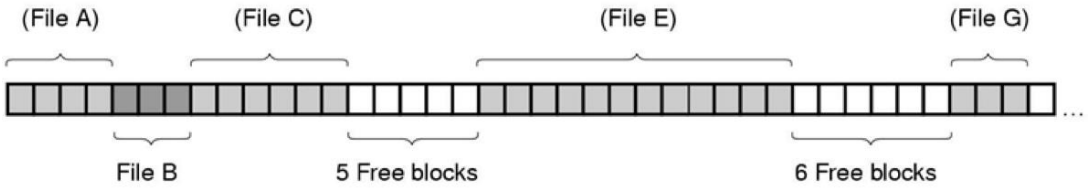
Contiguous Allocations

Diagram: Contiguous Allocations



Files are stored in free blocks.

Diagram: Contiguous Allocations



When a file is removed, the free blocks created can only be used by files where the $file\ size \leq number\ of\ free\ blocks$. This means that it is necessary to break files into blocks; to link these blocks together in order to determine in what sequence the blocks of data should be fetched so that the original file is re-created., file allocation tables are required.

File Allocation Tables

A **file allocation tables (FAT)** is a way in which the system knows where data has been stored (track, cluster, sector) by using a table which shows the next cluster to visit after the current cluster has been accessed.

The FAT is stored on track 0, which is an early part of disk. This contains information as to where a given file is stored on the disk in blocks. The file allocation table is stored in a separate area on the disk to the data.

Example: Use of a File Allocation Table

File Allocation Table	
Cluster	**Next Cluster**
2	
3	
4	7
5	
6	FFFF
7	10
8	
9	
10	6
11	
12	
13	

Clusters	
Cluster Number	**Data**
2	00 00 00 00 00 00 00 00 00 …
3	00 00 00 00 00 00 00 00 00 …
4	56 A3 AA 09 7C 32 C0 0C 8A …
5	00 00 00 00 00 00 00 00 00 …
6	C3 FA 77 09 4A 32 1A BB FF FF
7	3F 33 6E 23 5D 10 19 FB 84 …
8	00 00 00 00 00 00 00 00 00 …
9	00 00 00 00 00 00 00 00 00 …
10	56 33 77 09 77 32 11 00 84 …
11	00 00 00 00 00 00 00 00 00 …
12	87 45 22 56 78 44 88 99 34 …
13	00 00 00 00 00 00 00 00 00 …

File Systems

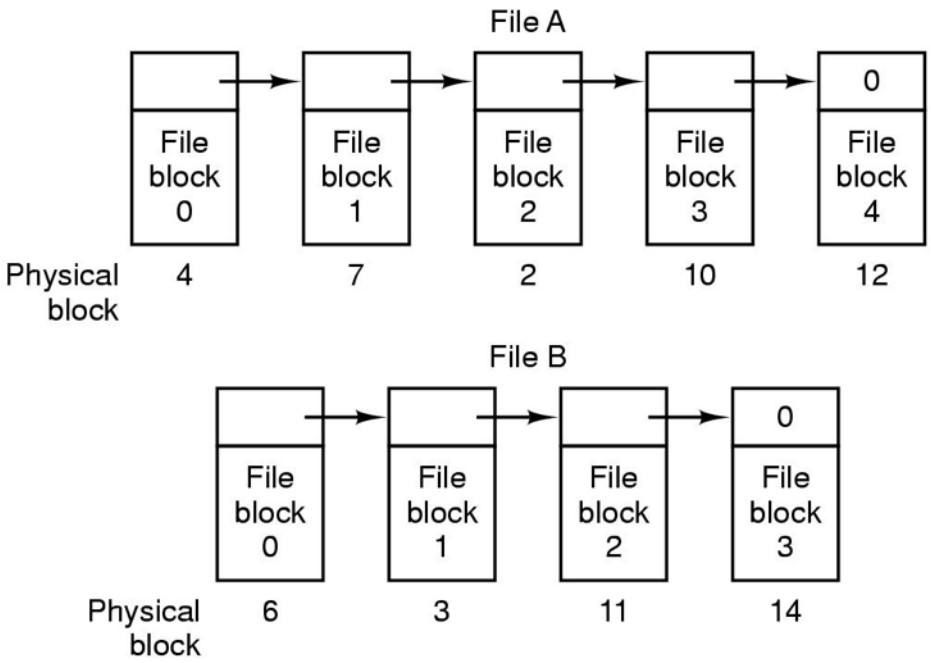
At the end of the data there can be an End of File (EOF) marker, denoted by "FF FF" – this is also used in the FAT.
Data containing all zeros (0's) show that the cluster is free.

Linked List Allocations

A **file allocation tables (FAT)** is a way in which the system knows where data has been stored by having each file as a linked list of disk blocks.

The information to retrieve the next block is stored with the data, rather than in a separate area on the disk.

Diagram: Linked List Allocations



Each file block has two fields:

- pointer field – contains a pointer to the next file block in the sequence; and
- data field – contains the actual data associated with the file.

This method of file allocation works similar to a file allocation table however, it does not require a table and therefore does not necessitate a separate area on the disk to store a table.

File Systems

File Operations and File Attributes

File Operations

The operating system needs to provide functionality to seek, read and write to a file system.

The typical system calls are:

- create;
- delete;
- open;
- close;
- read;
- write;
- append;
- seek; and
- rename.

File Attributes

There are a number of possible pieces of information that can be associated to a file, these are stored on disk.

Typical File Attributes

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

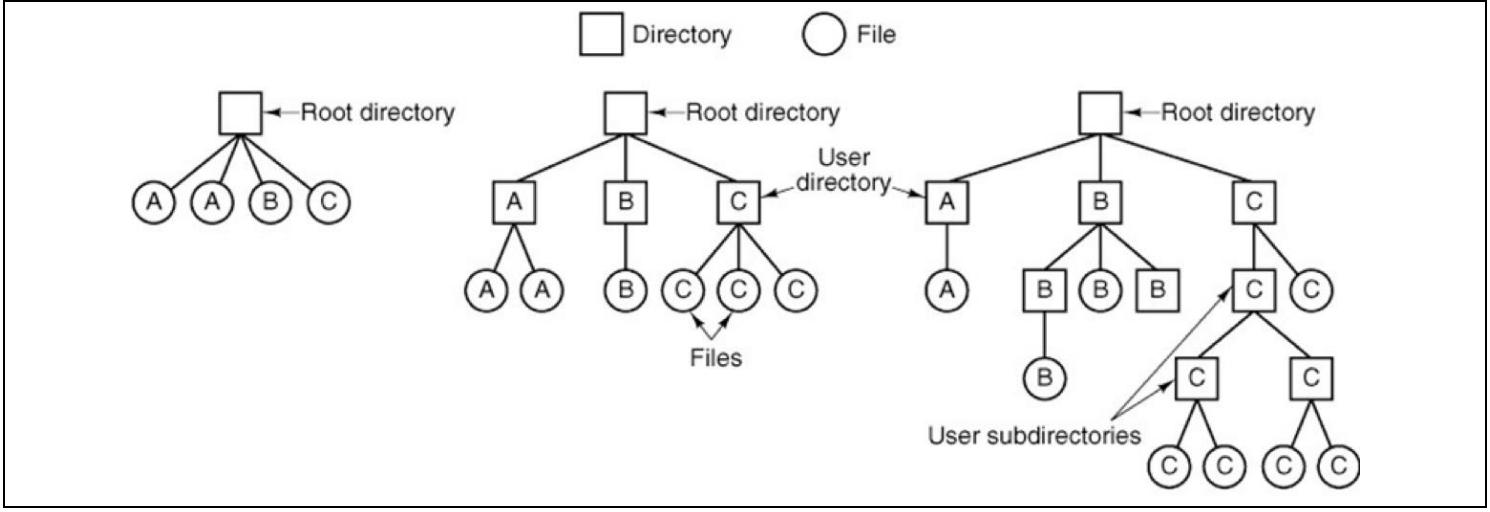
File Systems

Hierarchical Directories

Format

A **tree data structure** is a hierarchical data structure which organises data in hierarchical structure with a root value and subtrees of children with a parent node, represented as a set of linked nodes. These can be used to represent a hierarchical directory file system.

Diagram: Hierarchical Directories in a Tree Data Structure



A simple directory (single-layer) consists of:

- a root directory as the root value; and
- files as the child nodes.

More complex directories (multi-layered, hierarchical) consist of:

- a root directory as the root value;
- directories as the child nodes; and
- directories are sub-trees, meaning they are parent nodes which have child nodes of more directories or files.

Directory Operations

The operating system needs to provide the following system calls:

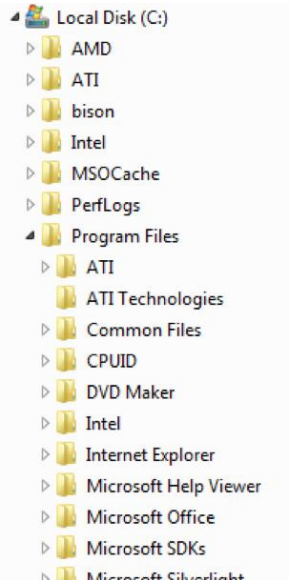
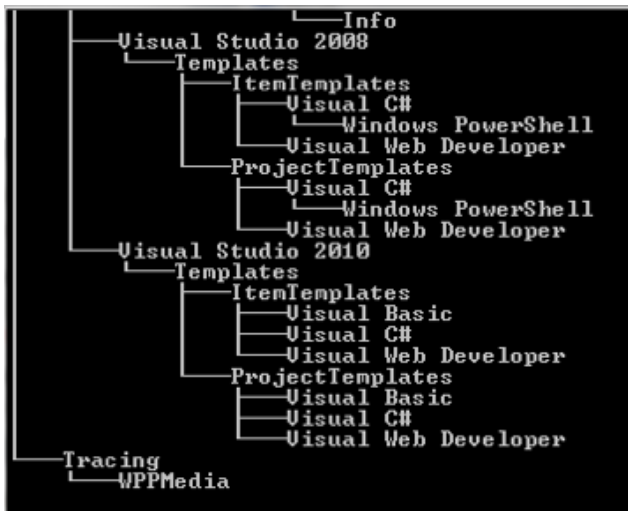
- create;
- delete;
- open directory;
- close directory;
- read directory; and
- rename.

File Systems

Uses

Hierarchical directories can be found in many operating systems.

Example: Hierarchical Directories

Windows File Explorer	`tree` Command in Command Prompt
	

File Systems

Searching Trees

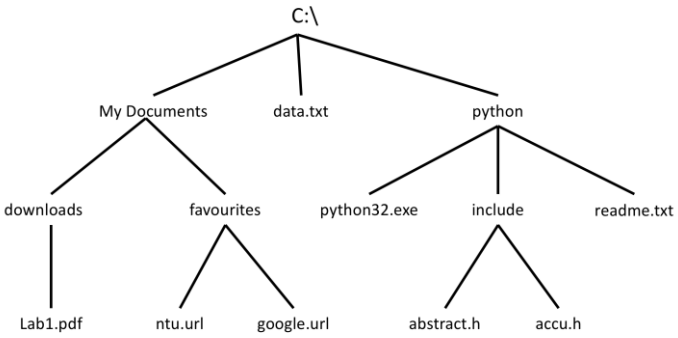
There are two traversal methods used for searching a tree:

- breadth-first – visits all of the nodes directly attached to a node and backtracks to the next node; and
- depth-first – visits all of the nodes on one route before backtracking and beginning the next route.

Example: Tree Traversal

Serach the tree using a breadth-first traversal and a depth-first traversal.

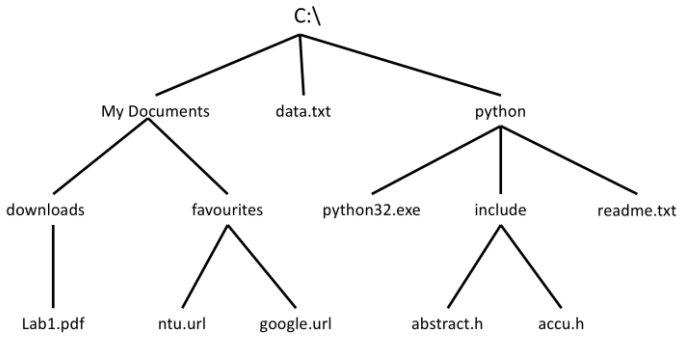
Breadth-first Traversal



The items in the tree would be accessed in the following order:

- C:\
- My Documents
- data.txt
- python
- downloads
- favourites
- python32.exe
- include
- readme.txt
- Lab1.pdf
- ntu.url
- google.url
- abstrach.h
- accu.h

Depth-first Traversal



The items in the tree would be accessed in the following order:

- C:\
- My Documents
- Downloads
- Lab1.pdf
- Favourites
- ntu.url
- google.url
- data.txt
- python
- python32.exe
- include
- abstract.h
- accu.h
- readme.txt

Operating Systems

Purpose of an Operating System

Definition

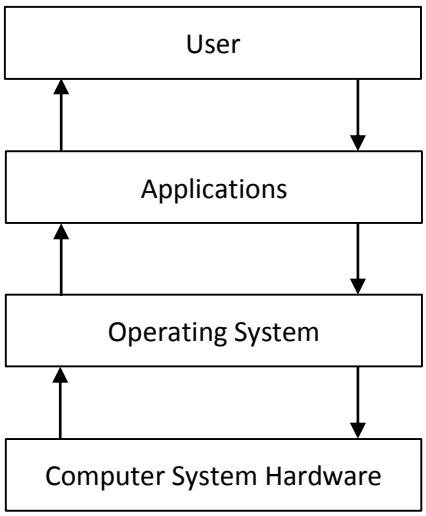
An **operating system** is a piece of software that runs on a computer system, it is responsible for:

- managing the hardware;
- the graphical user interface or command line interface; and
- all other software running on the computer system.

Layers of a Computer System

The operating system allows a user to interact with the computer system.

Diagram: Computer System Layers



Kernel

The **operating system kernel** hides the complexity of how a computer system works from users.

It is responsible for managing the computer's hardware including:

- memory management;
- CPU scheduling;
- handling interrupts; and
- input/output devices.

Operating Systems

Types of Operating Systems

Popular Operating Systems

Type of Operating System	Client	Server	Embedded

Characteristics	• Stand-alone operating system.
	• Used by single users.
	• Does not need to be connected to any other system to run.
		• Normally complete operating systems with a file and task manager.
		• Additional features such as:
		  ○ a web server;
		  ○ directory services; and
		  ○ a messaging system
		may also be included.
			• Stored on ROM chips in computer systems.
			• Designed for portable or dedicated devices.

Uses	Used by single users on: • desktop computer; • notebook; or • any portable computing device.	Used in client/server network environments.	Used on portable or dedicated devices, such as: • PDAs; • cell phones; • point-of-sale devices; • VCRs; • industrial robot controls; and • modern toasters.
Examples	• DOS (developed for original IBM PC) • Windows 3.x and later • MacOS • UNIX • Linux	• Windows NT Server • Windows 2000 Server and later • UNIX • Linux • Novell Netware • Solaris • Red Hat Enterprise Server	• Windows CE (variations include Windows Mobile and Pocket PC) • iPhone OS / iOS • Palm OS • BlackBerry OS • Embedded Linux • Google Android • Symbian OS

Is an Operating System always required?

Not all processors require an operating system. Some dedicated embedded applications can be stored on non-volatile memory and control the system.

This method may be implemented in:

- car engine management systems;
- building climate control systems; and
- microprocessor controlled home electronic devices.

These embedded systems are not designed to perform tasks concerned with general purpose computing due to the physical constraints of the system, such as small RAM and limited CPU power.

However, embedded processors are getting more powerful for the same cost and embedded operating systems are getting lighter, such as embedded Linux varieties. The tasks the embedded operating systems are required to do are getting more complex. Due to this, companies often use an off-the-shelf processor and operating system for an embedded application.

Operating Systems

Functions of an Operating System

Start the Computer System

There are two types of boot:

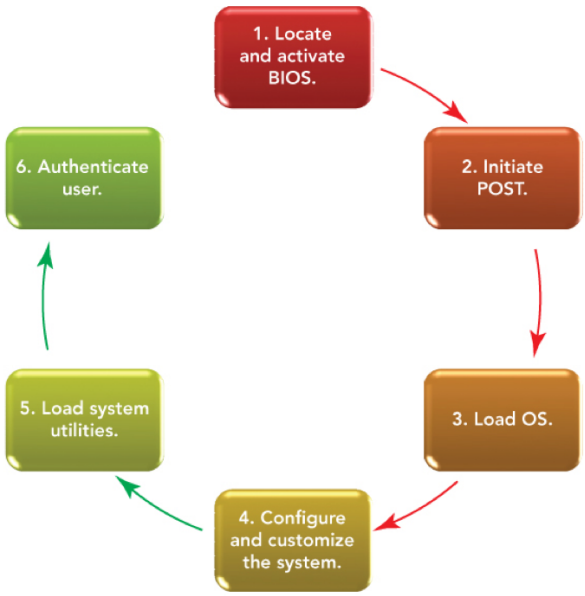
- cold boot – the computer system has been started from no power; and
- warm boot / soft boot – the computer system has been started but was already receiving power, such as a restart.

When a computer system boots, the operating system's kernel is loaded into the computer's RAM. The kernel transfers files from secondary memory to main memory (RAM).

After the operating system has been loaded in to main memory (RAM), the operating system:

- checks the registry which holds configuration information about software and other devices;
- checks driver configurations, for example a connected printer may require drivers to be loaded;
- detects plug and play devices;
- loads system utilities such as anti-virus software, volume controls and power management; and
- verifies authorised users during login session.

Diagram: Booting a Computer System



BIOS – Basic Input/Output System
POST – Power-on Self-Test

Input / Output Devices

The operating system must handle messages from input / output devices. This is achieved by using device drivers, which are programs that allow communication between the operating system and the devices.

These devices include:

- printers;
- scanners;
- mice; and
- keyboards.

Operating Systems

Managing Applications

The operating system must run and manage multiple applications as these are the functions that have the most dramatically affect to the overall performance.

Managing Memory

The operating system must allocation each program its own portion of RAM and attempt to keep the programs from interfering with each other's memory allocation.

Provide a User Interface

The operating system must then provide a user interface. This is the part of the operating system that the user sees, interacts with, and uses to communicate with programs.

Operating Systems

Processes and Threads

Processes

Definition

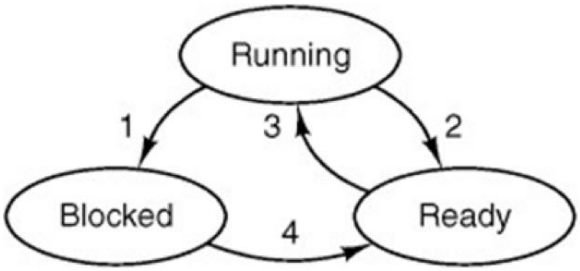
A **process** is an instance of a computer program that is being executed, it contains the program code and its current activity. These may consist of multiple threads.

Process States

A process can be in one of three states:

- running – actually using the CPU to perform a task;
- ready – temporarily stopped to let another process run; or
- blocked – unable to run until some external event occurs, such as:
  - waiting for an interrupt, this is a message from the hardware saying that a resource is now available to read from; and
  - waiting for another process to finish accessing a shared resource, such as a file or memory.

Diagram: Process States



Transitions between process states:

- running / ready ⇒ blocked – the process is waiting for some input from I/O;
- blocked ⇒ ready – the process has received input from I/O;
- blocked ⇒ running – the process sends an interrupt and the interrupt service routine (ISR) is executed, the scheduler is called to transition the process from blocked to ready;
- running ⇒ ready – the process has had opportunity to run and is flagged as no longer currently running, so that another process can run; and
- ready ⇒ running – the next process that is ready to run is set to running to allow access to the CPU.

The transition between process states is dependent on the scheduling algorithm used. A clock is used to send a signal to stop the current process, move it from running to ready and then run the scheduler to find out what process should be processed next and the next process will be made to running.

Operating Systems

Starting and Stopping Processes

Principal events that cause processes to be created include:

- system initialisation;
- execution of a process creation system call by a running process; and
- a user request to create a new process.

Conditions that cause a process to terminate include:

- normal exit (voluntary);
- error exit (voluntary), these should be handled by the code in the program;
- fatal error (involuntary), these are not handled by the code in the program; and
- killed by another process (involuntary).

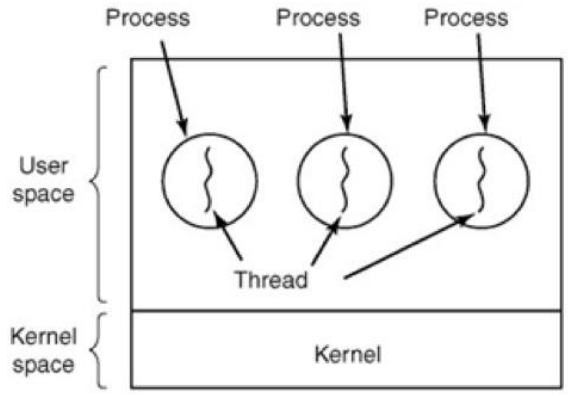
Threads

Definition

A **thread** is an independent bit of code inside a single process. Threads can be run independent of one another but are completing part of a process.

Threads as part of a process

Diagram: Threads inside a process



Processes are often decomposed in to smaller component parts, threads, that can be performed independent of one another.

Examples of threads include:

- outputting to the screen; and
- dealing with the disk drive.

If one of the threads becomes blocked, the other threads can continue. Therefore, the process is not blocked if a thread is blocked.

Operating Systems

Multi-tasking

Definition

Multi-tasking allows multiple processes to run alongside each other and appear to be processed simultaneously and achieves this without any apparent delay.

Scheduling

Definitions

A **scheduler** uses a scheduling algorithm to determine each process's time quantum.

A **scheduling algorithm** consist of code which allows each process access to the CPU so that instructions can be executed.

Time quantum is the amount of time that the process can run on the CPU is called the time quantum.

Process Manager

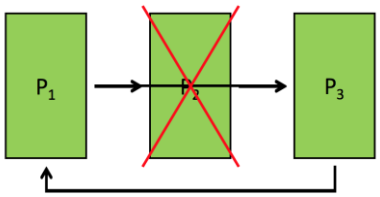
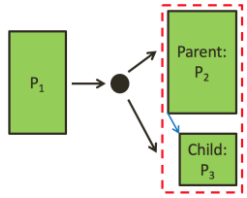
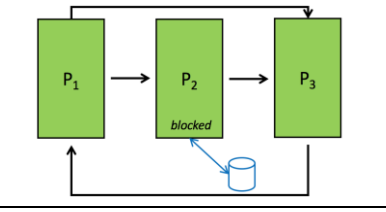
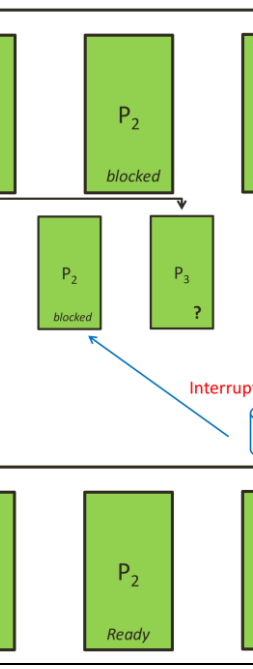
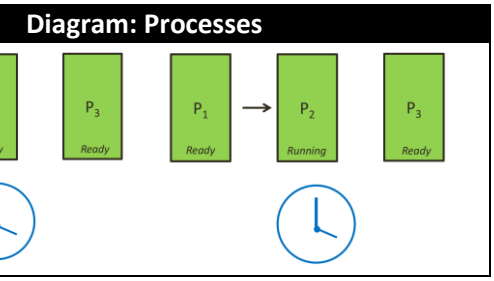
The **process manager** is a part of the operating system's kernel, it keeps track of running processes and uses a scheduling algorithm to deal with multi-tasking.

Why is scheduling required?

- Multi-tasking operating systems allow multiple processes to run alongside each other and appear to be processed simultaneously. However, since a processor unit can only perform one task at any given time, a scheduler must be used to give each process a slice of processor time to give the appearance of multi-tasking without any apparent delay.
- Promote efficient processing by making maximum use of processor time, maximise throughput and maximise use of hardware resources such as input and output devices.
- Ensure fairness to all users on a multi-user operating system.
- Provide acceptable response times to all users.

Operating Systems

When does scheduling occur?

When required	When usually done (but not absolutely required)
When a process exits	**When a new process is created**
Diagram: Processes 	**Diagram: Processes** 
The process may have completed or it has been terminated by the user. The process may have completed before time quantum completed.	A decision has to be made whether to execute the parent process or child process first as both can be in the ready state meaning that either can be run.
When a process becomes blocked as it is waiting for I/O	**When an I/O interrupt occurs**
Diagram: Processes 	**Diagram: Processes** 
A process may be become blocked if it is waiting for the user to input on an I/O device, the process cannot continue until this has occurred.	A process that is blocked can become ready when receiving an interrupt from an I/O device. A decision has to be made whether to continue executing P_3 or stop executing P_3 and resume P_2.
	When a clock interrupt occurs
	Diagram: Processes 
	A clock signal may be used as part of pre-emptive multitasking, where scheduler allows process to run for a fixed time (such as 50ms). When an interrupt is received, the next process is run.

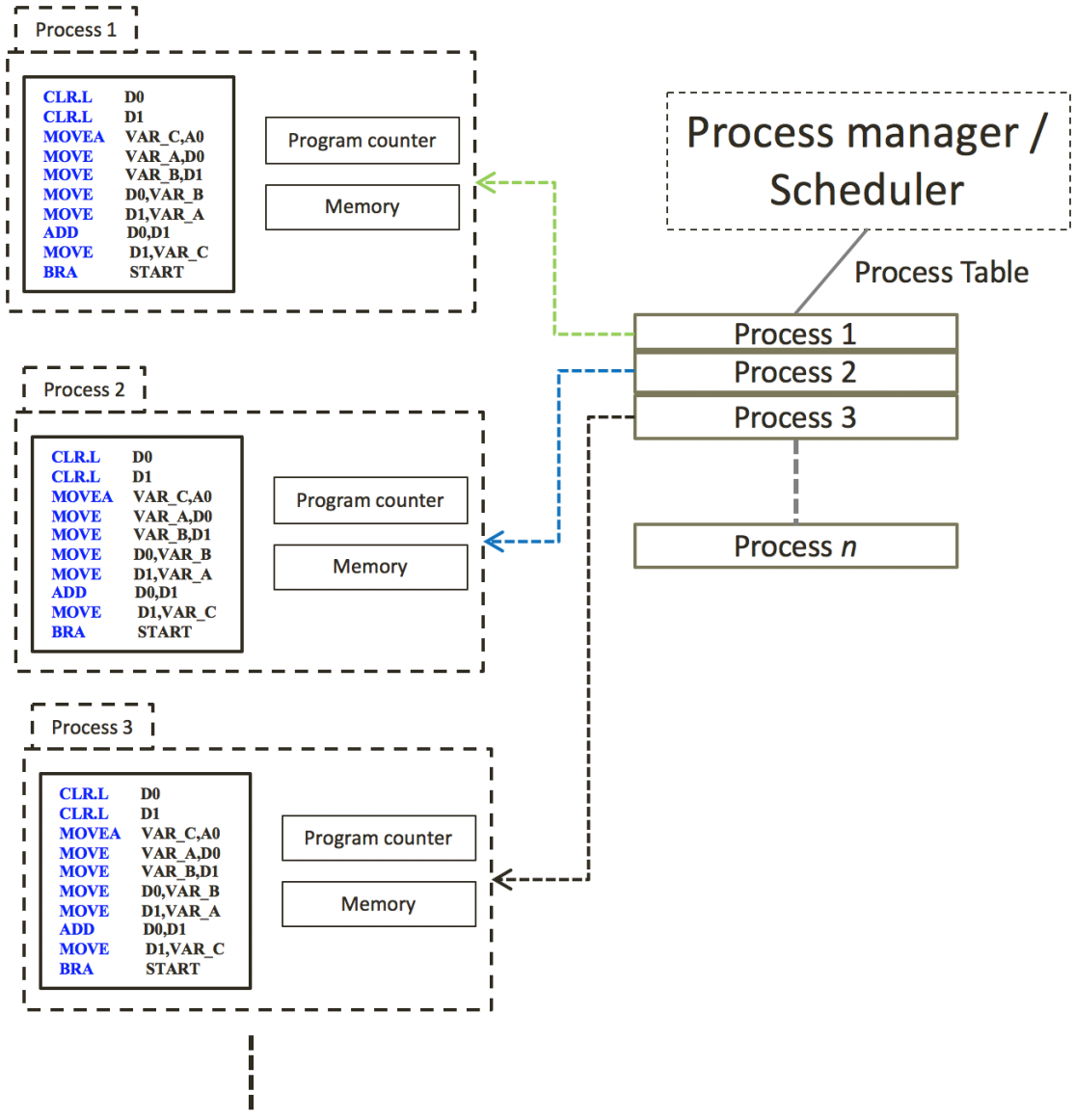
Operating Systems

Scheduling Algorithm: Round Robin

In round robin, each process is dispatched on a first in first out (FIFO) basis and given time slice, which is a fixed amount of processor time. If the process is not completed by the end of the time period, it goes to the back of the queue so that the next process in line can be processed.

Diagram: Round Robin

Diagram: Process Manager in Round Robin



With the assumption that none of the processes are in a blocked state, the round robin scheduling algorithm will perform in the following manner:

- the first process is run for the amount of time defined by a given time slice (such as 50ms); and
- the subsequent processes are run, in turn, for the same amount of time.

This process will continue for any number, n, of processes and then the scheduling algorithm will restart at the first process.

Operating Systems

Scheduling Algorithm: Shortest Job First

In shortest job first, the process that will take the shortest time is processed first, in its entirety. This algorithm needs to know the time each job will take in advance.

Diagram: Round Robin

Evaluation

Scheduling Algorithm	Advantages	Disadvantages
Round Robin	**Simple to implement**.	**Does not consider the priority of a process** and therefore the important processes may not be completed quickly.
	Suitable for some types of computer systems, such as those which will be running processes of similar priority and size.	**Does not consider the size of a process** and therefore, if processes are of varying sizes, there may be inefficiencies since a single process may take a long time to complete thus leaving the user waiting before they can perform any other actions.
Shortest Job First	**High throughput** because the number of processes completed is high due to the shortest processes taking precedence.	**Does not consider the priority of a process** and therefore the important processes may not be completed quickly.
	Shorter processes are processed quickly because they take precedence.	**Relies on an estimation of how long a process will take** which could be incorrect.
	Minimises the average time taken to complete a process because the shortest processes take precedence.	

Operating Systems

Multi-threading

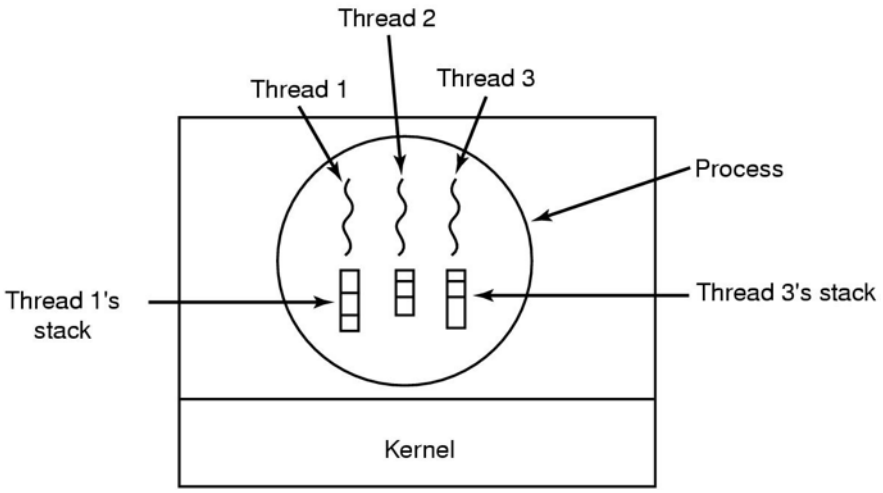
Definition

Multi-threading allows multiple threads to exist within the context of a process such that they execute independently but share their process resources.

Resources

Threads have their own stacks but can also share resources, such as global variables and files.

Diagram: Thread Resources



If two or more threads try to access a shared resource at the same, it could lead to the race condition.

Race Condition

The race condition can occur if one thread is currently accessing a variable or file when it's quantum time is finishing. The next thread must not change the variable until the previous thread has finished with the variable. The next thread must not change the variable until the previous thread has finished with the variable or file; this can be handled using "flags" which show whether the variable or file can be accessed depending on whether it is currently in use.

Operating Systems

Memory Management

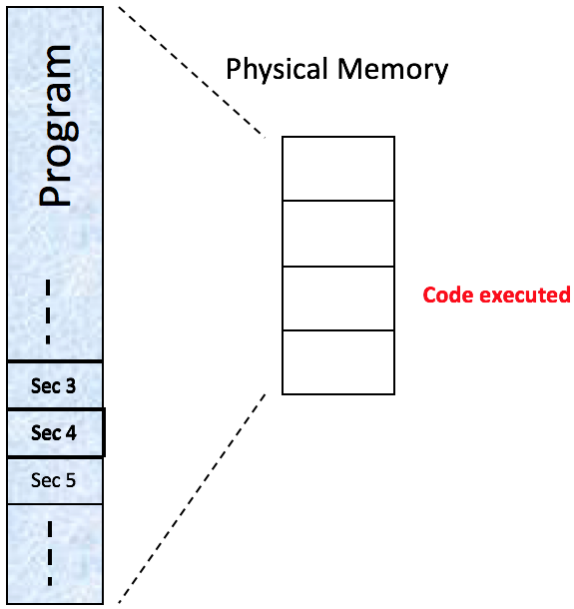
Definition

Memory management is the process of controlling and coordinating computer memory by assigning blocks to various running programs to optimise overall system performance.

Overlays

In the 1950's mainframes had limited addressable memory to run programs and therefore it was common that a given program would be too large to store in the computer system's physical memory. This meant that programs had to be stored on secondary memory.

Diagram: Thread Resources



This method was a problem as programmers were required to split programs into overlays and load them in to physical memory when required.

Operating Systems

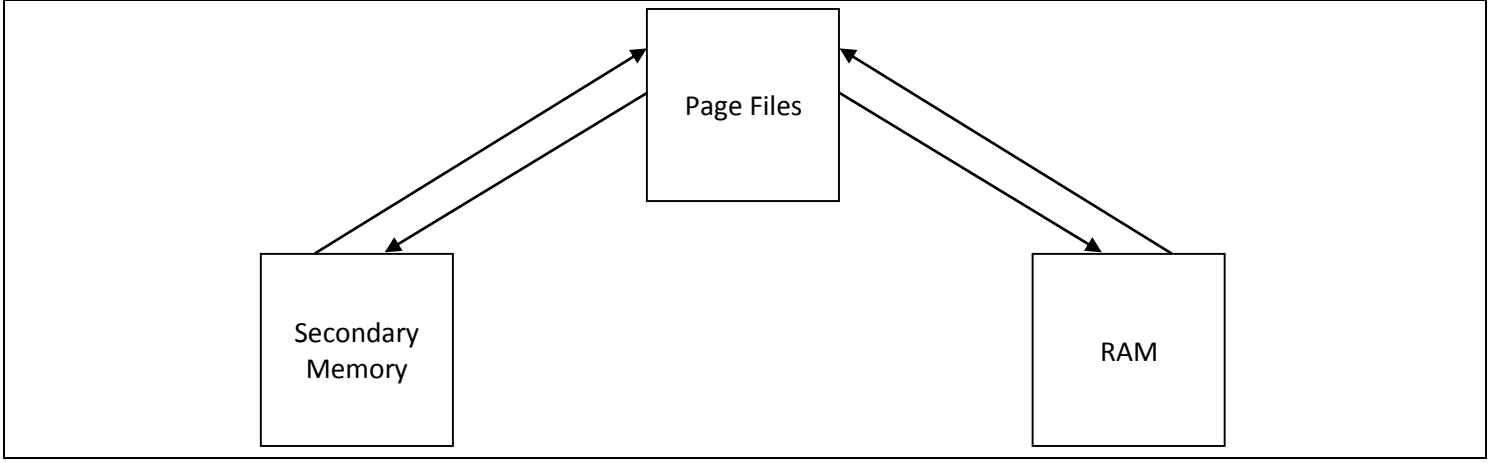
Paging

Why is paging required?

The address space of a computer system is usually much greater than the physical memory installed and therefore virtual memory must be utilised.

Virtual Memory

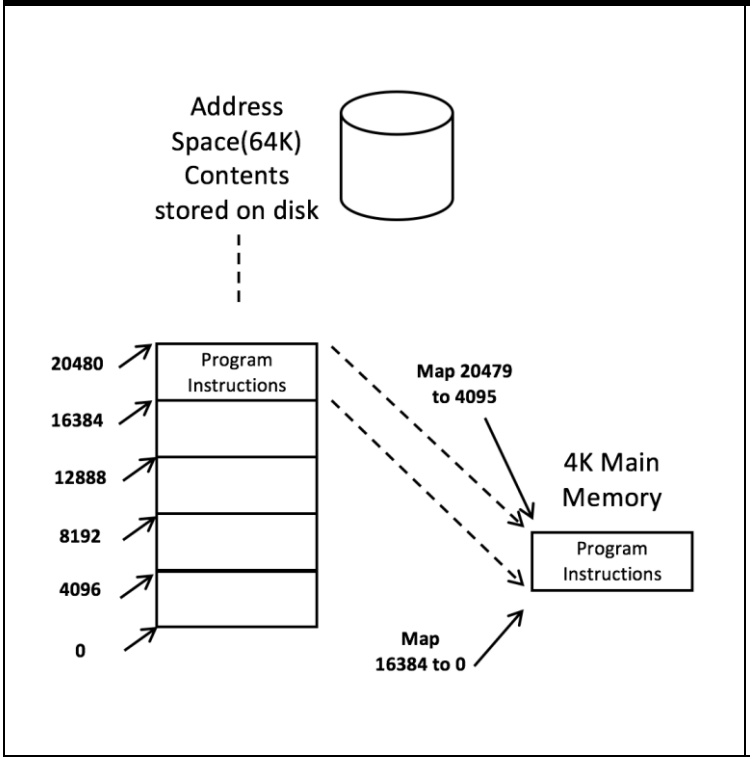
Diagram: Virtual Memory

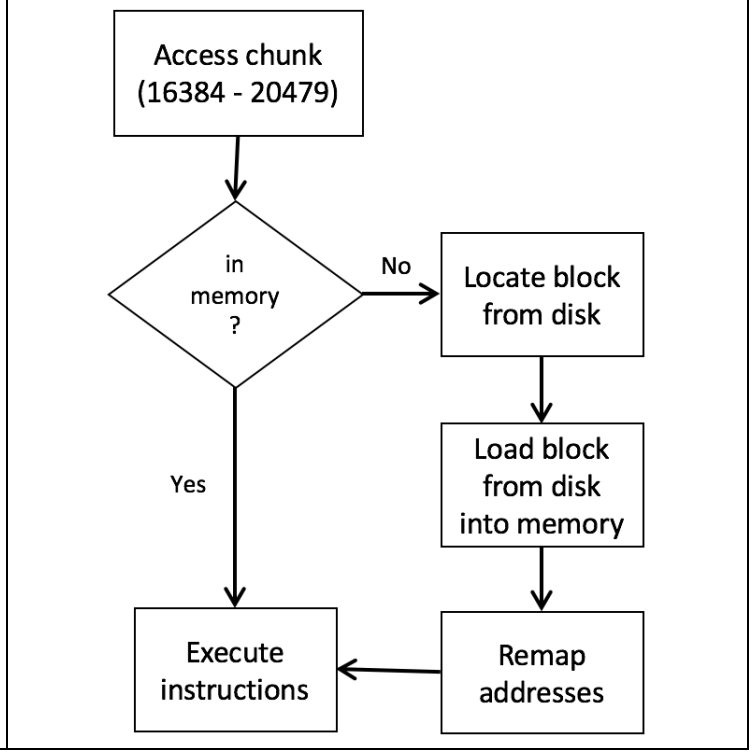


When RAM becomes full, data and instructions which are being used less frequently are moved to the virtual storage partition on the secondary storage device through page files which contain segments of data and instructions.

The space allocated on secondary storage for this use can be specified by the computer system's user; it is usually xx'MB or x'GB.

Example: Paging



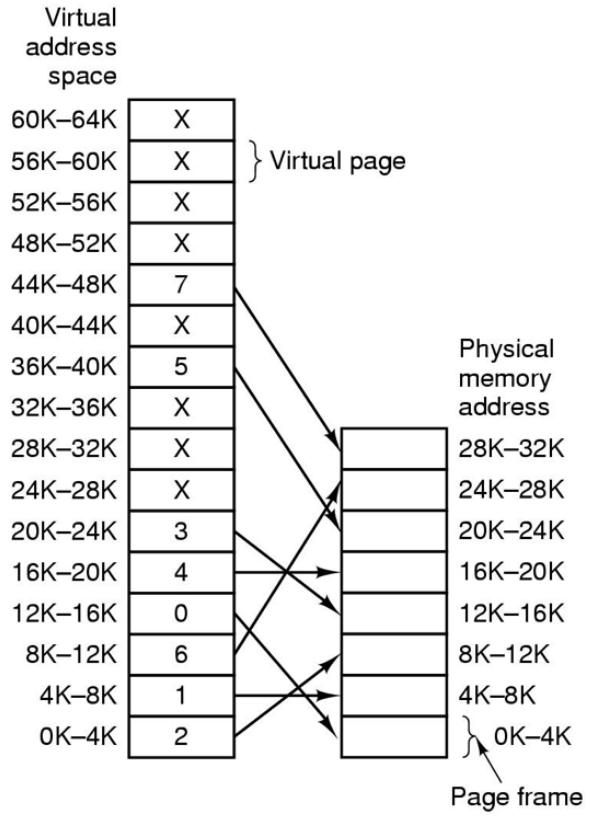


Operating Systems

Virtual Memory and Page Tables

Page tables are used to link virtual pages and physical page frames.

Diagram: Page Tables



When a page is required and is not in memory a page fault occurs. This causes the page to be loaded into corresponding page frame.

This method allows all programs and data to fit into memory, regardless of physical memory size.

Evaluation

Advantage	Disadvantage
Helps to prevent some computer crashes because a computer system which has no space left in memory can crash.	**Slower than accessing data directly from cache or RAM** because secondary storage has relatively slow read and write speeds than cache and RAM and is further away from the CPU so data and instructions must be transferred further.
	Disk thrashing is likely to occur, this is where page files are constantly exchanging between virtual memory and main memory and can cause the performance of the computer system to be seriously degraded.

Operating Systems

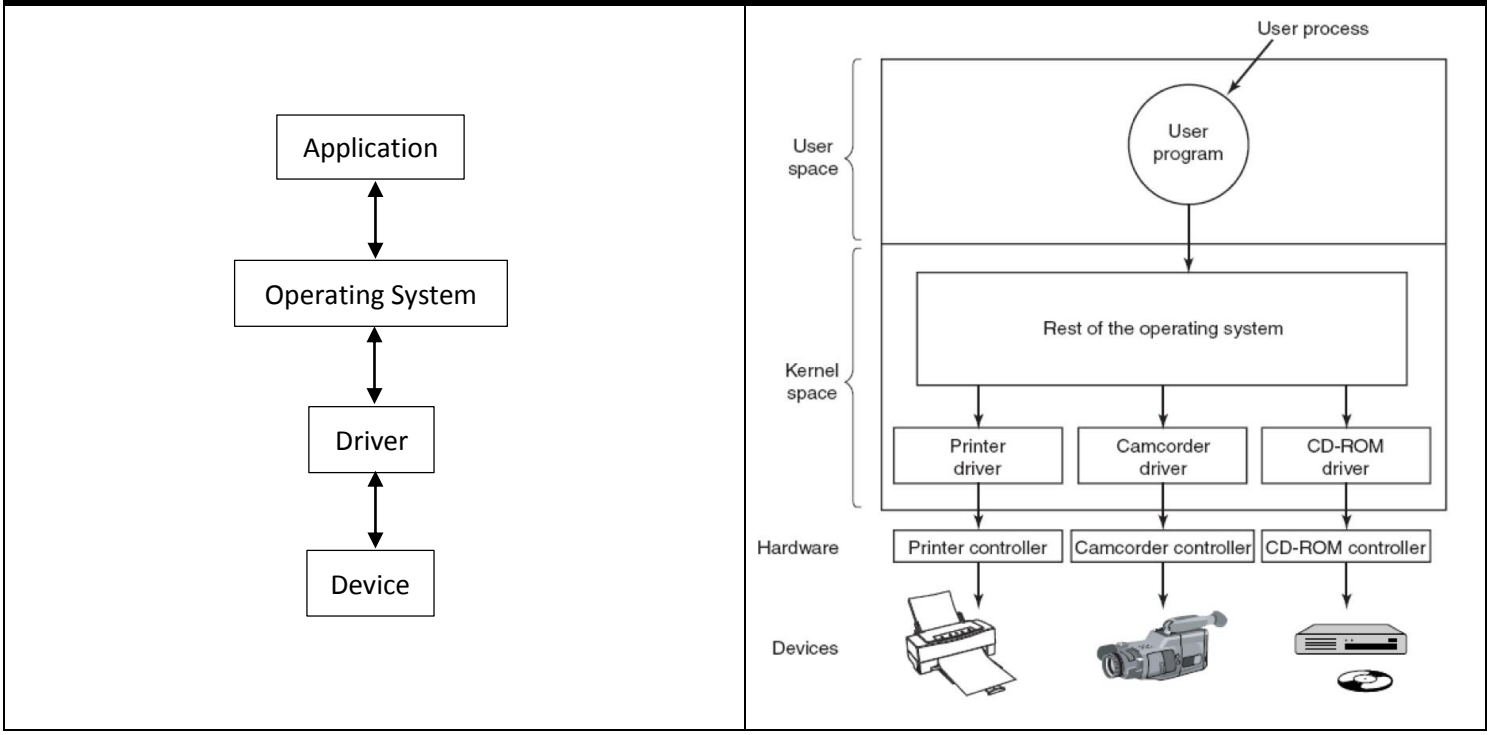
Device Drivers

Definition

A device driver is a piece of software which enables a computer system's operating system to control and communicate with a piece of hardware. This is written by the manufacturer of the piece of hardware and is specifically designed for that piece of hardware.

How they work

Diagram: Computer System Layers



Specific device drivers are provided by hardware manufactures on installation discs or Internet download links. Hardware manufacturers must create device drivers for each operating system for which the hardware must be compatible.

Generic device drivers, or **non-specific device drivers**, are often provided in operating systems. These may not be as good as specific device drivers because they may not be supported by the device or provide all of the necessary features to allow the hardware device to operate properly.

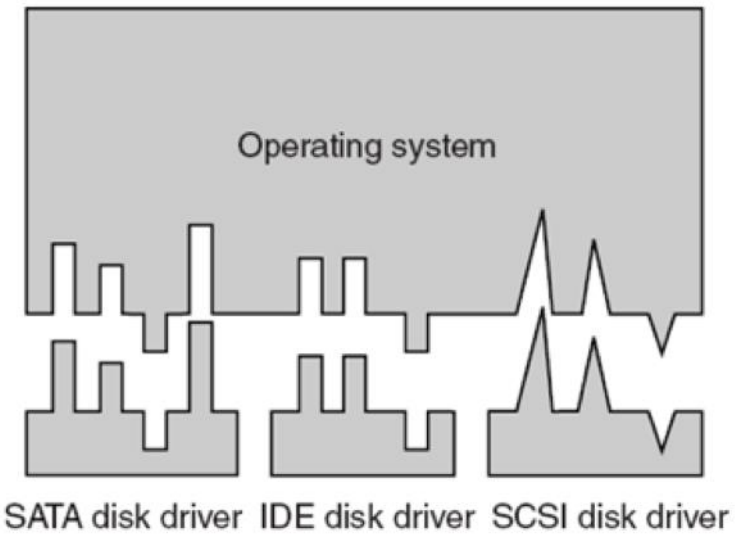
The exact detail of which device driver is required by the operating system is stored in a file and the drivers for the hardware connected are loaded when the computer system is booted.

Operating Systems

Device Driver Interfacing

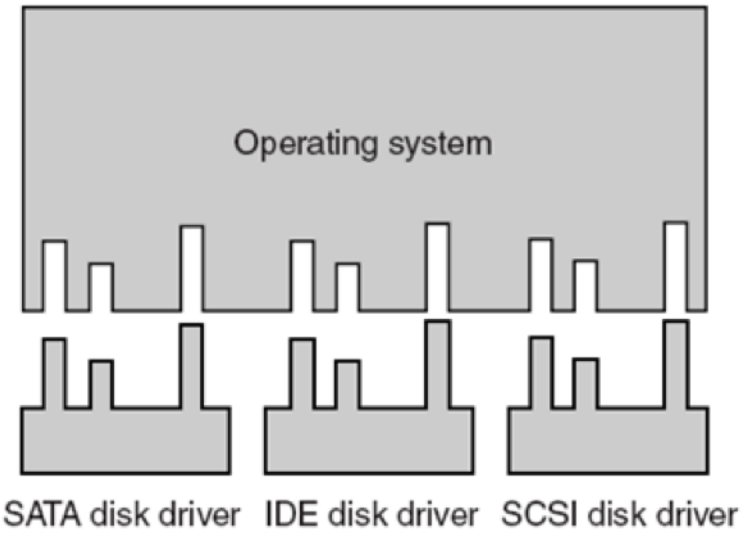
A problem arises as many different manufactures create device drivers for their devices.

Diagram: Device Drivers



This means that the operating system would need to be modified to be compatible with each device driver after a manufacturer develops a new device. This would yield a high amount of maintenance and code required in the operating system.

Diagram: Device Drivers with Standard Driver Interface

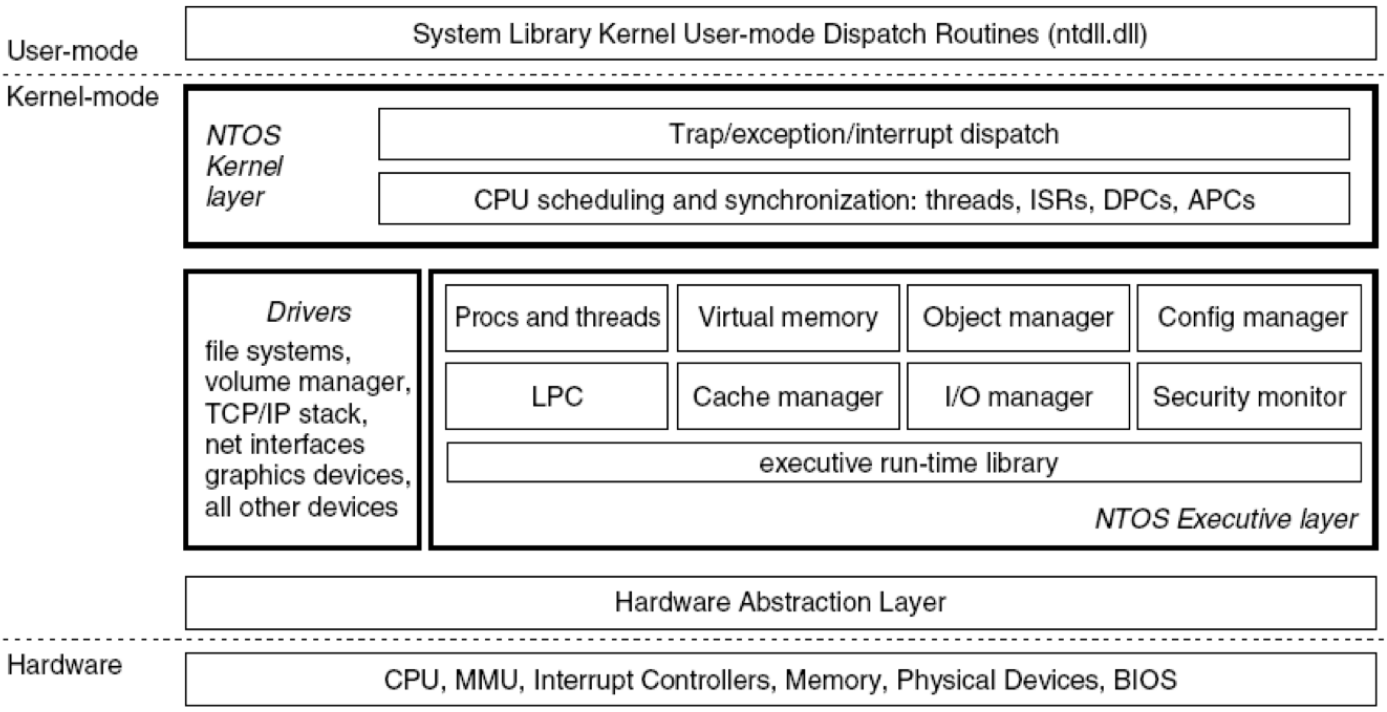


To overcome this problem, a standard driver interface is used. This allows the operating system to expose an application programming interface (API) to device manufacturers. The device manufacturers must adhere to the API and therefore, all drivers follow this API definition so the operating system code does not have to be modified.

Operating Systems

Windows Operating System Structure

Diagram: Windows NT Structure



- **Hardware abstraction layer** – Maps between generic hardware commands and responses and those unique to a specific platform. The HAL makes each machine's system bus, DMA controller, interrupt controller, system timers and memory module look the same to the kernel. It also delivers the support needed for symmetric multiprocessing. The HAL makes the OS system platform independent.

- **Kernel** – The kernel mode has full access to the hardware and system resources of the computer and runs code in a protected memory area. It controls access to scheduling, thread prioritization, memory management and the interaction with hardware. The Windows Executive services make up the low-level kernel-mode portion, and are contained in the file NTOSKRNL.EXE It deals with I/O, object management, security and process management.

- **User mode** – User mode is made up of various system-defined processes and DLLs and provide an interface to user-mode software. Includes a variety of modules for specific functions, which make use of the basic services provided by the kernel.

Networking

Networking Concepts

Networks

Definitions

A **communication network** is where two or more devices able to exchange signals and/or data through the use of network technology.

A **computer network** is where two or more computer systems are able to exchange data through the use of network technology.

Data can be anything capable of being represented in digital form, such as words, numbers, dates, images and sounds, without context.

Protocols

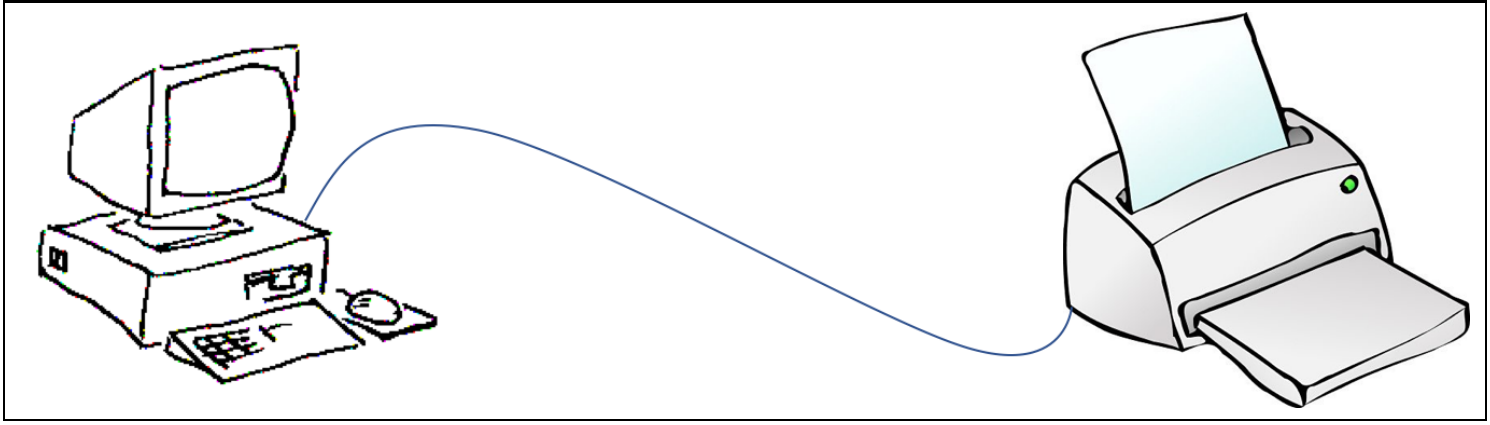
Definition

A **protocol** is a formal standard comprised of rules, procedures and formats that define how communication will take place between two or more devices over a network. Network protocols govern the end-to-end communication between networked devices and promotes timely, secure and managed network communication.

How data is sent

Protocols ensure that the sending network device is transmitting data in the same format that the receiving network device is expecting. If this was not agreed before communication, the application receiving the data would be unable to decipher and display the data.

Diagram: Case Study



Data can be sent from a computer system to a peripheral, such as a printer:

- text characters are typically represented using 8 bits, in a mapping called ASCII, and stored in memory as 8-bit bytes;
- the bits are converted to electrical signals by interface;
- the signals travel along the wire, connecting the computer system and the printer, and identified by the printer's interface; and
- the signals are converted to bits and interpreted as ASCII characters.

Part of the protocol for this communication must agree in what order the bits will be transmitted so that the correct ASCII character is represented.

Networking Concepts

Topologies

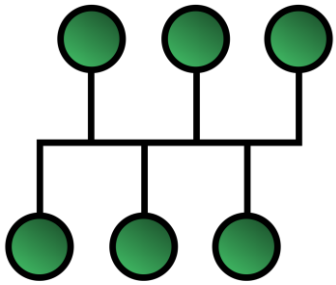
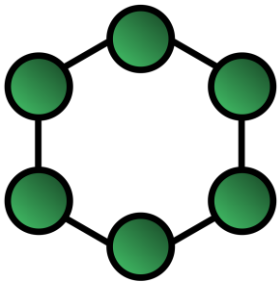
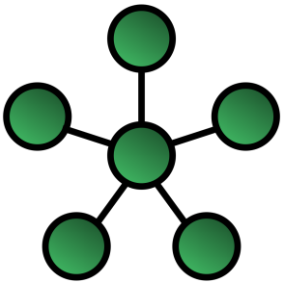
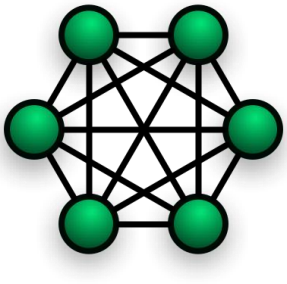
Definitions

The **physical topology** of a network is the design layout of the network infrastructure. This is important when deciding a wiring layout for a new network.

The **logical topology** of a network is the path in which the data travels and describes how networked components communicate across the physical topology.

Physical topologies and logical topologies are independent of one another. This means that a network can be one physical topology and perform as a different logical topology, such as Ethernet which always performs as a logical bus topology regardless of the physical topology implemented.

Comparison

Diagrams: Topology Visualisation			
Physical Bus	**Physical Ring**	**Physical Star**	**Wireless Mesh**
			

A **scalable** network can be changed in size easily by adding additional hardware and network infrastructure. Planning can be put in place to allow scalability in the future, such as purchasing switches with more ports than currently necessary so that the empty ports can be utilised as the network increases in size.

Companies which are likely to grow over time are likely to require a scalable network in order to meet increasing demands.

Networking Concepts

<table>
<tr><th>Topology</th><th colspan="2">Physical Bus</th><th colspan="2">Physical Ring</th></tr>
<tr><td>Definition</td><td colspan="2">A physical bus topology involves a single cable connecting all networked devices, with a terminator at the end of the bus. The terminator is a piece of hardware which prevents signals in the cable from echoing, where they bounce back and forth.

This topology is half-duplex, meaning that data and information cannot travel in opposite directions at the same time.</td><td colspan="2">A physical ring topology involves connecting networked devices in a circle. The data passes from one computer system to a sequential computer system until the data reaches its destination.

This topology is simplex, meaning that data and information can only travel in one direction.</td></tr>
<tr><td>Uses</td><td colspan="2">• Small offices.</td><td colspan="2">• Telecommunication networks.</td></tr>
<tr><td rowspan="5">Evaluation</td><td>Advantages</td><td>Disadvantages</td><td>Advantages</td><td>Disadvantages</td></tr>
<tr><td>Easy to install.</td><td>Can be slow if there are many connected devices because data must visit more nodes before finding the destination node.</td><td>No data collisions because data only travels in one direction.</td><td>Can be slow because data must pass through each node until its destination is reached.</td></tr>
<tr><td>Easy to add additional nodes because they can be connected using a single cable.</td><td>Low security because all devices can access all of the data being transmitted across the network.</td><td>Scalable because new nodes can be connected with little impact on performance.</td><td>If any of the nodes become unavailable or there is a problem with any of the cables, network data cannot be transmitted to any of the nodes.</td></tr>
<tr><td>Suitable for some networks, such as those which are setup on a temporary basis.</td><td>Data collisions are likely because all data is transferred using a single cable, this may be more significant on a network with many connected nodes.</td><td>Transmission of data is relatively simple as it only travels in one direction.</td><td>Can be difficult to add additional nodes because the network must be switched off.</td></tr>
<tr><td></td><td>Little scalability because the single cable has a maximum length.</td><td></td><td>Troubleshooting issues as the cause of a problem can be difficult to identify.</td></tr>
<tr><td></td><td>If there is a problem with the central cable, network data cannot be transmitted to any of the nodes.</td><td></td><td></td></tr>
</table>

Networking Concepts

<table>
<tr><th>Topology</th><th colspan="2">Physical Star</th><th colspan="2">Wireless Mesh</th></tr>
<tr><td>Definition</td><td colspan="2">A physical star topology involves a central node, such as a switch or computer acting as a router, which keeps a record of the unique MAC addresses on the network and can route data to the correct computer system.

This topology is duplex, meaning that data and information can travel in opposite directions at the same time.</td><td colspan="2">A wireless mesh topology involves nodes which have all have a connection to one another. Data is transmitted by using intermediate nodes until its destination is reached. Only one node is required to have an active Internet connection as the other nodes can share this connection.

This topology is duplex, meaning that data and information can travel in opposite directions at the same time.</td></tr>
<tr><td>Uses</td><td colspan="2">• Local area networks, such as a school.</td><td colspan="2">• Mobile hotspots.
• Music streaming between devices in a building.
• Home automation, such as smart controls for lighting and heating devices.</td></tr>
<tr><td rowspan="4">Evaluation</td><td>Advantages</td><td>Disadvantages</td><td>Advantages</td><td>Disadvantages</td></tr>
<tr><td>Reliable because a single failure can be isolated.</td><td>Expensive because more hardware is required and large amounts of cabling is required.</td><td>Less expensive because there are no cabling costs.</td><td>Higher power consumption because each devices must act as a router, this may be more significant on portable devices such as a smartphone.</td></tr>
<tr><td>Secure because all communications must go through the server meaning that nodes cannot interact with each other directly.</td><td>Expertise required to set up and maintain the network.</td><td>Self-healing because if a node becomes unavailable, other nodes can route the data transmissions.</td><td></td></tr>
<tr><td>Few data collisions because each node has its own cable connection to the server.</td><td>If the central node becomes unavailable, network data cannot be transmitted to any of the nodes.</td><td>Scalable because new nodes can be easily connected using wireless medium.</td><td></td></tr>
</table>

Networking Concepts

Network Media

Definitions

Network media refers to the communication channels used to interconnect notes on a computer network or data communications network.

Examples

- Copper Cables
- Optic Fibre
- Wireless

Speed

Data transmission speed, or **bandwidth**, is how many bits per second are sent along the network.

Signal propagation speed is the speed at which signals travel along a cable or through the air (3x10$_8$m/s for wireless) – this remains constant regardless of the data transmission speed.

Examples: Speeds		
Data Transmission Speed	**Standard Form Power**	**Common Usage**
kilobits / second	$\times 10^3$	Enough to transmit voice
10 – 100 megabits / second	$\times 10^6$	Broadband speeds
gigabits / second	$\times 10^9$	Desktop Ethernet
terabits / second	$\times 10^{12}$	Some optical fibres

Networking Concepts

Packets

Definition

A **packet**, or **datagram**, is a piece of a message transmitted over a packet-switching network.

Packet Contents

Diagram: IPv4 Packet

Header				Payload	Trailer	
source address	destination address	packet sequence number	protocol	data	error checking information	end of packet marker

Routing a Packet

Packets are routed by “hopping” between routers; a hop is the act of traversing between one router and another router across a network. Packets hop between the Internet layer and the link layer when being transmitted between routers connected to the Internet. Each time a packet “hops”, it is assigned a new MAC address for the next router to which it will be transmitted. This process continues until the packet reaches the router with the destination IP address. Each router has a record of other close routers in a routing table and sharing this information allows an algorithm to determine the optimum next path in the route for a packet, such as Dijkstra’s Shortest Path algorithm.

Networking Tools (Lab)

Command Prompt

Definitions

A **command line interface (CLI)** allows a user to interact with a computer system by typing in commands. The computer displays a prompt, the user keys in the command and presses enter or return.

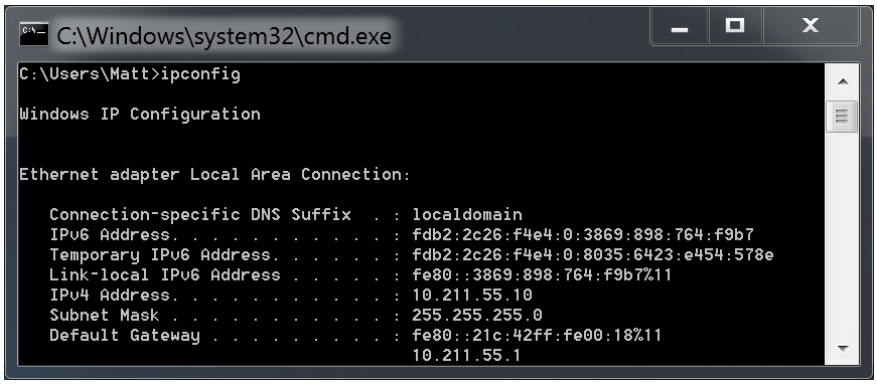
Command Prompt is the Windows derivative of a command line interface (CLI).

Networking Commands

ipconfig

ipconfig is used to display all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.

Example: Using `ipconfig`

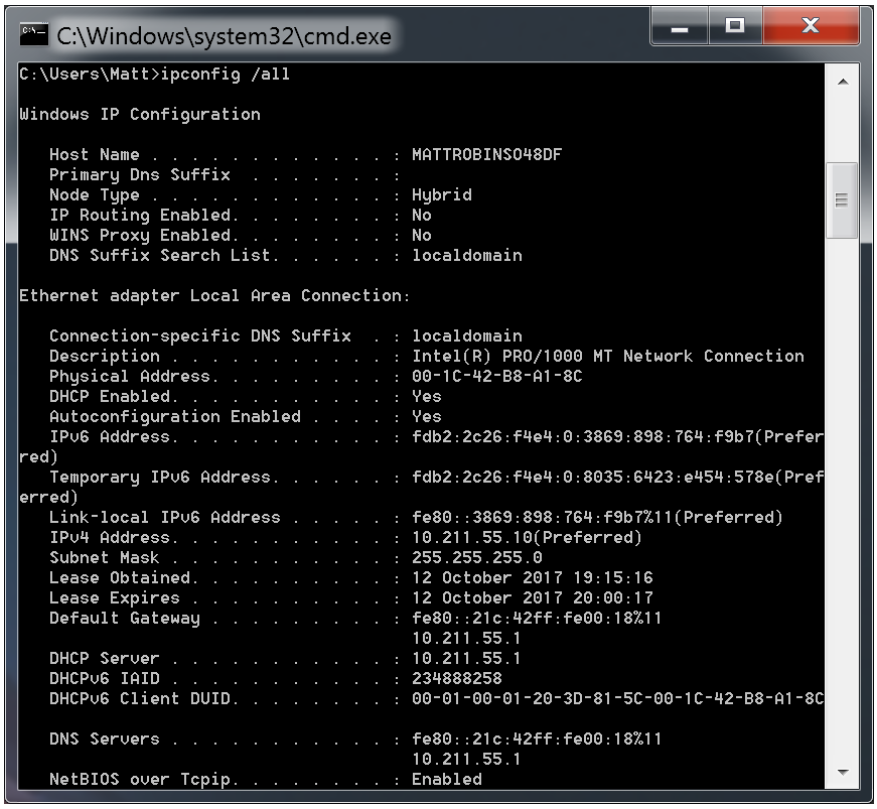


Output	Value	Meaning
IPv4 Address	10.211.55.10	The numerical label assigned to each device connected to the computer network that uses the Internet Protocol for communication.
Connection-specific DNS Suffix	localdomain	The DNS service used for resolving domain names.
Default Gateway	10.211.55.1	The device through which the computer system accesses the Internet, such as a router.

Networking Tools (Lab)

ipconfig /all can be used to view more information about the network configuration values.

Example: Using `ipconfig /all`

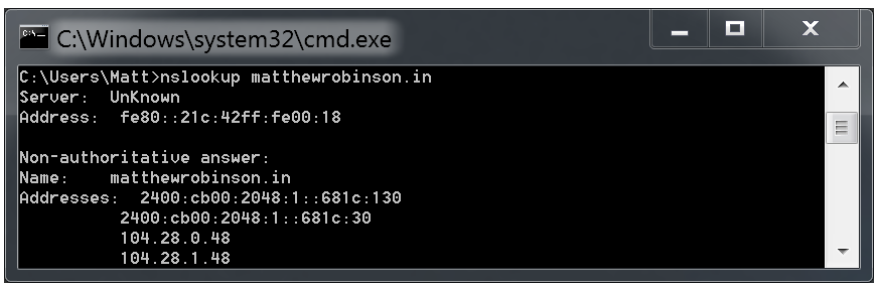


Output	Value	Meaning
Physical Address	00-1C-42-B8-A1-8C	The MAC address for the network card attached to the computer system.

nslookup

nslookup is used for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or for any other specific DNS record.

Example: Using `nslookup`



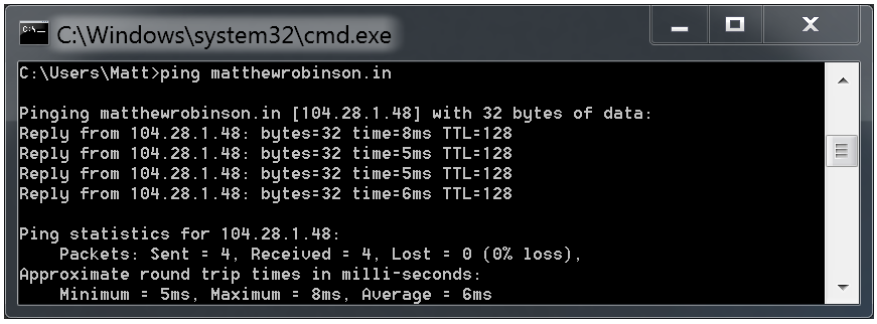
Networking Tools (Lab)

Output	Value	Meaning
Name	matthewrobinson.in	The domain name.
Connection-specific DNS Suffix	localdomain	The DNS service used for resolving domain names.
Default Gateway	10.211.55.1	The device through which the computer system accesses the Internet, such as a router.

ping

ping is used to test the reachability of a host on an Internet Protocol (IP) network. This is achieved by sending a request, in the form of a data packet, to an IP address that echoes (sends back) a packet (response). The "round trip" time is then calculated based on the time taken for the echo to occur after the request has been sent, this is how long it takes for the packet to be sent to the server and for the echo to be received.

Example: Using `ping`



Domain	Average Round Trip Time	Meaning
matthewrobinson.in	6ms	Normal time taken for a web server.
localhost / 127.0.0.1	<1ms	Very quick as the packet does not leave the computer system as localhost/127.0.0.1 references the local machine.
sydney.edu.au	283ms	Slow because the web server is far away.
sure.co.sh	550ms	Very slow because the web server is situated on a small island; the island may have inferior network infrastructure and therefore more routers must be used to send the echo back.
bt.com	Request timed out.	Some networking devices have been configured to prevent ping requests. This practice may be carried out by organisations or individuals in an attempt to prevent malicious attacks; some attackers may use probing to find out information about their network configuration.

Networking Tools (Lab)

The distance a packet has travelled can be calculated using the formula:

$$distance = time\ taken \times speed$$

where distance is measured in metres (m)
time taken is measured in seconds (sec)
speed is measured in metres per second (m/s) and is assumed to be $2 \times 10^8\ m/s$

Example: Calculating Distance from Nottingham to Sydney

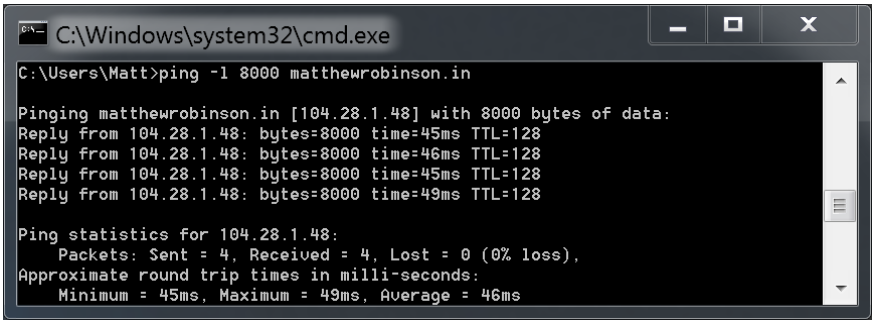
$distance = (283 \times 10^{-3}) \times (2 \times 10^8)$
$distance = 0.283 \times 200000000$
$distance = 56000000m$

The distance calculated using the round trip time is subject to inaccuracies because:

- the signal does not travel in a straight line, instead the hop between routers; and
- networking devices, such as routers, require a certain amount of time to process requests being made by signals.

By default, the command ping sends data to the server of size 32 bytes - however, this can be altered.

Example: Using `ipconfig`

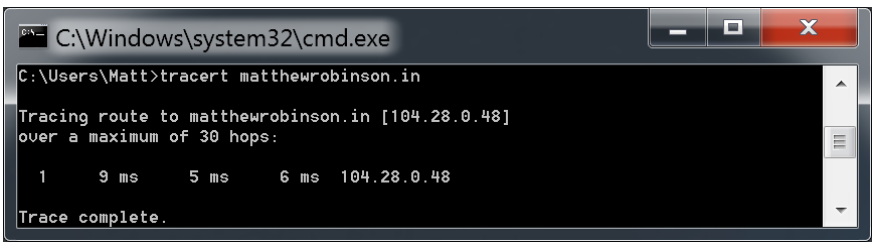


The average round trip time is likely to be increased as multiple packets may be required. This is because more data is being sent to the server and there is a limit to the amount of data that can be stored in a single packet.

tracert

tracert is used to see the path a packet takes when being routed from the source to its destination. Packets "hop" from one router to another and the number of hops can be viewed using tracert.

Example: Using `tracert`



Networking Tools (Lab)

Domain	Hops	Meaning
matthewrobinson.in	1	-
localhost / 127.0.0.1	1, no router used	No routers used as the packet does not leave the computer system as localhost/127.0.0.1 references the local machine.
ntu.ac.uk	18	-
sydney.edu.au	25, 2 timed out	Some networking devices have been configured to prevent ping requests. This practice may be carried out by organisations or individuals in an attempt to prevent malicious attacks; some attackers may use probing to find out information about their network configuration.

Networking Principles

Wide Area Network (WAN)

Definition

A **Wide Area Network (WAN)** consists of two or more inter-connected Local Area Networks (LANs) that are in different geographical locations.

Characteristics

- Can spread over any geographical area.
- Make use of external telecom infrastructure services for long distance network communications, these can comprise of: fiber optic lines; satellite communication links; leased telephone lines; or microwave links.
- External operators are responsible for maintaining the WAN infrastructure
- The cables and equipment are usually not owned by the business or private individual however some multi-national companies, such as Google, may own their own WAN infrastructure.

Examples

- The Internet (the largest WAN and is not owned by any single organisation or private individual).
- Defense organisations.
- Banking firms.

Networking Principles

Packets

Packets

Definition

A **packet**, or **datagram**, is a piece of a message transmitted over a packet-switching network.

The term packet was introduced by Donald Davies in late 1960's:
"I thought it was important to have a new word for one of the short pieces of data which travelled separately. This would make it easier to talk about them. I hit on the word packet in the sense of the small package."

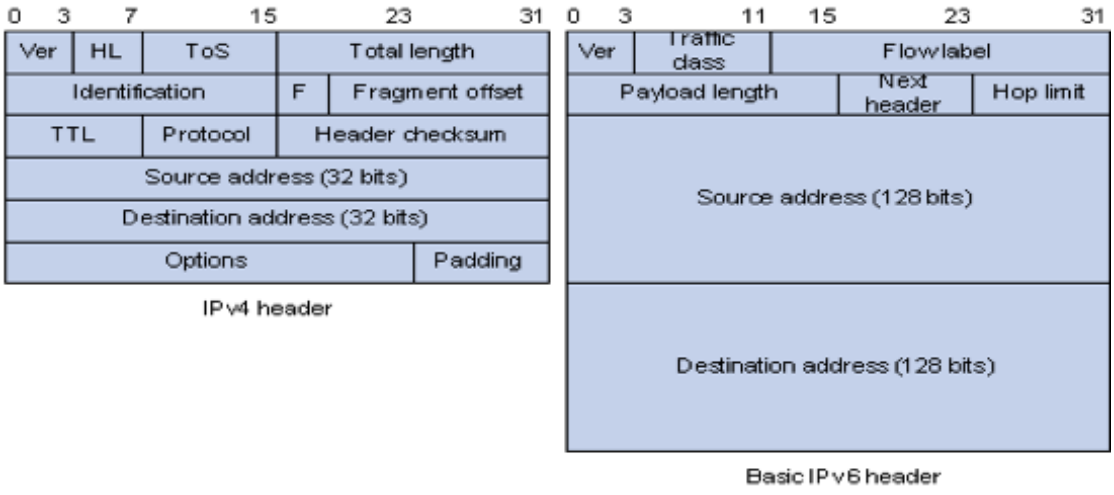
Packet Anatomy

Diagram: IPv4 Packet

Header				Payload	Trailer	
source address	destination address	packet sequence number	protocol	data	error checking information	end of packet marker

Source address – The IP address of the sending computer system.
Destination address – The IP address of the recipient computer system.
Packet sequence number – The position of the packet in the series of packets.
Protocol – The protocol by which the packet is being sent, such as FTP.
Data – The data inside the packet.
Checksum – A calculated summary of such a data portion which is used to ensure the integrity of data portions.
End of packet marker – Defines the end of the packet.

Diagram: IPv6 Packet



Networking Principles

Packet Switched Networks and Circuit Switched Networks

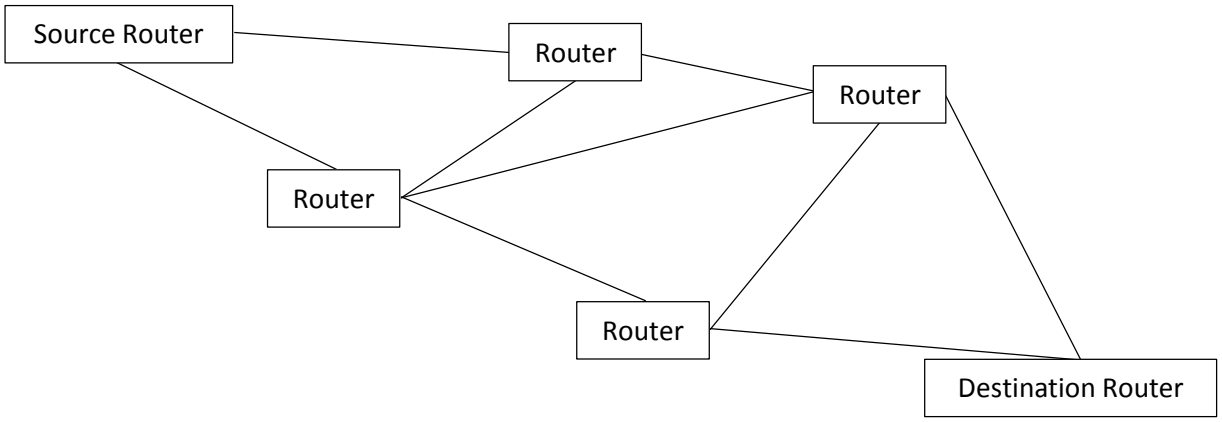
Packet Switching

Definition

Packet switching is a method of transmitting packets of data across a network using inter-connected nodes on which other communications are happening simultaneously.

How it works

Diagram: Packet Switching



Examples

- The Internet – packets are transmitted from one router to another between the Internet layers and link layers in the TCP/IP stack.

Evaluation

Advantages	Disadvantages
Robust because if one router is not available which causes a path to be incomplete, there is likely to be another path which the packet could route.	**Latency can occur** because packets are transmitted along random paths meaning that there is no guarantee that they will arrive within a specific time period.
Efficient because each path is only occupied for the duration of the packet transmission and therefore the path is available to other transmissions once completed. This means that data transmission is possible during high traffic loads on a network.	**Packets can arrive in a random order** because they can be all transmitted along different routes therefore, they must be reordered once received using the packet sequence number in the packet header of each received packet.
Packets are subject to error checking through the use of error correction codes which allow corrupted bits to be corrected.	
Resilient because network downtime will only require the re-transmission of the lost packets rather the entire data.	
Secure because in the event of malicious data interception, only segments of the data are available.	

Networking Principles

Circuit Switching

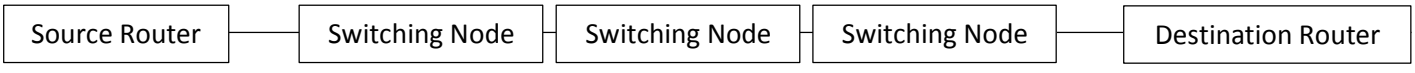
Definition

Packet switching is a method of transmitting data across a network using a direct link between two devices for the duration of the data transmission.

How it works

;

Diagram: Circuit Switching



Examples

- Public Telephone System – when a caller dials a telephone number, various switches in telephone exchanges set up a path between the caller and the recipient.

Evaluation

Advantages	Disadvantages
No latency because the data is guaranteed to arrive within a specific time period.	**Not robust** because if one router fails, communication is lost.
Data arrives in order because it travels sequentially along one route and therefore, the data does not need to be reordered.	**Inefficient** because once a connection is made, the path cannot be used by other transmissions even if no data is currently being transmitted.
Slow setup because initiating the communication can be a slow process.	**Every connection has to support the bandwidth required** and therefore it is not suitable for some data transmissions which require high amounts of bandwidth.
Dedicated route prevents interruption from other data transmissions because there is a guaranteed contention ratio, which is the number of users sharing the same data capacity.	**Insecure** because in the event of malicious data interception, the entire data is available.

TCP/IP Stack

What is the TCP/IP Stack?

Definition

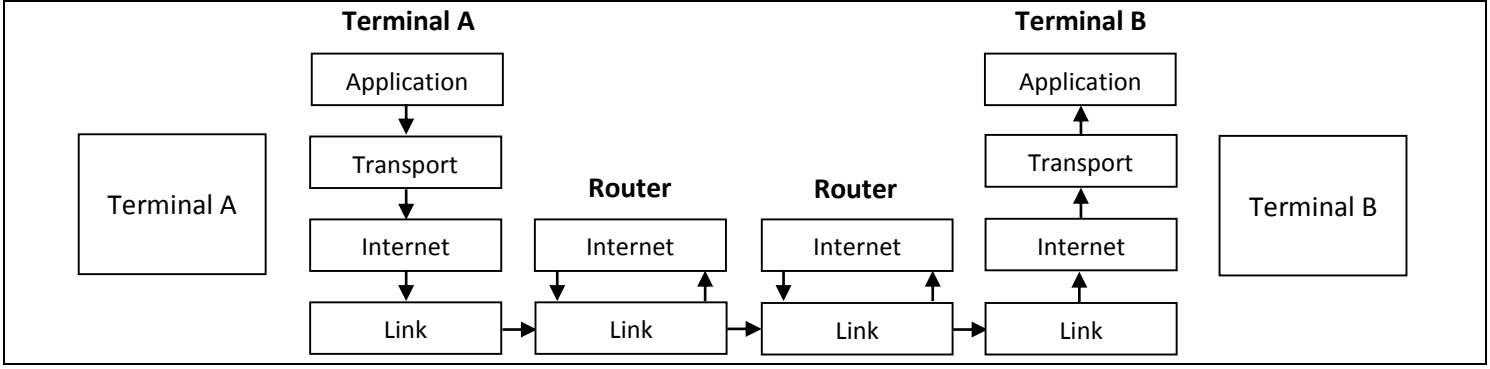
The **Transmission Control Protocol / Internet Protocol (TCP/IP)** stack is a set of networking protocols which work together as four connected layers and pass incoming and outgoing data packets throughout the layers during network communication. The TCP/IP stack contains layers, of software and hardware, which are designed to successfully transfer data across a network.

Layers

Model

The TCP/IP stack contains more than 100 protocols, with the TCP and IP protocols being the most important.

Diagram: TCP/IP Layered Model

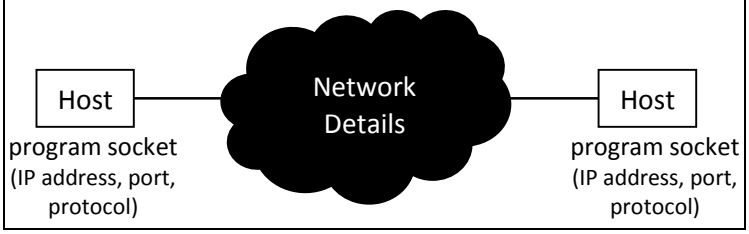


A **protocol data unit (PDU)** is a unit of data which is specified in a protocol of a given layer and consists of protocol-control information. At each layer, a protocol header containing information regarding how the data should be transmitted is added to the PDU.

Abstraction

The protocols operate on different levels of the TCP/IP stack. Each layer appears to communicate with its peer (such as application to application) with its own protocols however, this is an abstraction of how data is transmitted across a network. Instead, data packets are passed and received to/from layers with the lower layers offering services to the layer above. This abstraction allows programmers to only be concerned with a particular layer.

Diagram: Abstraction from Programmers Perspective



Changing Protocols

The structure of the TCP/IP stack allows changes to be made to a layer without necessary changes to other layers. This is true as long as the interface between the layers has not been changed. A protocol may be changed if an update in technology is required due to factors such as security or the scarcity of IP addresses (demonstrated in the IPv4 to IPv6 migration).

TCP/IP Stack

Application Layer – Sockets and Port Numbers

Sockets

Definition

A **socket** is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

Setting up a Socket

A socket can be setup by:

- one host creates a socket, specifying the protocol, IP address and port to listen on;
- if the Transmission Control Protocol (TCP) is being used, a “pipe” for the data must be created; and
- if the User Datagram Protocol (UDP) is being used, no “pipe” has to be created.

Port Numbers

Definition

A **port number** is the logical address of each application or process that uses a network or the Internet to communicate. This is used to uniquely identify a network-based application on a computer system as each application is allocated a 16-bit integer port number. The port number is assigned automatically by the operating system, manually by the user or is set as a default for some popular applications.

Why are they necessary?

Port numbers are necessary because they allow:

- multiple sockets to be used on a computer system at the same time – this requires a port number as packets being delivered to a computer system’s IP address require a port number in order for them to be sent to the correct application;
- servers to use ‘registered’ ports – these are port numbers assigned by the Internet Assigned Numbers Authority (IANA) for use with a certain protocol or application (such as HTTP – 80, FTP – 21 and SFTP – 22).

TCP/IP Stack

Application Layer – HyperText Transfer Protocol (HTTP)

Definition

HyperText Transfer Protocol (HTTP) defines how messages are formatted and transmitted, and what actions web servers and browsers should take in response to various commands. Websites using HTTP have a URL beginning with "http://".

Browser to Web Server Communication

A request for web content is sent by the client browser to the web server and a response is sent by the web server back to the client browser.

Example: Communication

The client enters the URL into their browser.

HTTP **Get**	(default)

The web server responds with the data.

HTTP **OK**	data: index.html

The web page can now be displayed in the client's browser.

The file index.html the file "pic.jpg" from the "images" directory.

HTTP **Get**	/images/pic.jpg

The web server responds with the data.

HTTP **OK**	data: pic.jpg

The image can now be displayed on the web page.

The file index.html the file "logo.gif" from the "images" directory.

HTTP **Get**	/images/logo.gif

The web server responds with the data.

HTTP **OK**	data: logo.gif

The image can now be displayed on the web page.

In the event of an error, the web server may respond with an error status code.

Error Status Codes

Status Code	Meaning
200	Successful request, the webpage exists.
301	Moved permanently, often redirected to a new URL.
401	Unauthorised request, authentication required.
403	Forbidden, access to this page or directory is not permitted.
404	Page does not exist.
500	Internal server error, often caused by an incorrect server configuration.

Transmission Control Protocol (TCP)

Definition

The **Transmission Control Protocol (TCP)** is a network communication protocol designed to send data packets over the Internet. It provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating by an IP network.

Sending Data using TCP

The transport layer uses the Transmission Control Protocol (TCP) to:

- perform a three-way handshake – the host sends a request to the client, the client responds to the request and sends a request back to the host and the host responds to the request, this is achieved in three packets;
- to establish an end-to-end connection with the recipient computer;
- split the data into packets; and
- label the packet with a packet number, the total number of packets and the port number through which the packet should route to ensures that the packet is handled by the correct application on the recipient computer system.

A timer is incremented until a maximum value is reached, called a "time out". When this occurs, a request for the retransmission of the lost packet is sent to the application layer. A receipt of packets received is recorded.

A handshake is an automated process that sets the parameters for communication between two communicating computer systems before packets are transferred.

Handshake Parameters

Physical Protocols	Logical Protocols
These are concerned with how the data will be transmitted.	**These are concerned with the data itself.**
• Transmission medium – wired or wireless (agreeing Wi-Fi frequency, 2.4Ghz or 5Ghz). • Mode of data transmission: ○ simplex – unidirectional data transfer; ○ duplex – bidirectional data transfer; or ○ half duplex – bidirectional data transfer which only allow once direction of data transmission at once time.	• error checking protocols: ○ echo – when a message is received, it sends the message back to compare with the original to see if it has been transmitted correctly; ○ parity check – uses the even or odd scheme to detect whether data has been transmitted correctly; or ○ checksum / check-bit – also uses the even or odd scheme to detect whether data has been transmitted correctly. • Character set – the methods of encoding characters, such as ASCII or Unicode. • Packet size. • Information about data encryption. • Bitrate of data transfer.

TCP/IP Stack

Protocol Header

Diagram: TCP Protocol Header

2 bytes	*2 bytes*	*4 bytes*	*4 bytes*	*4 bits*	*6 bits*	*6 bits*	*2 bytes*	*2 bytes*	*2 bytes*	*3 bytes*	*1 byte*
Source Port	Destination Port	Sequence Number	Acknowledgement Number	Header Length	Reserved	Flags	Window Size	Checksum	Urgent	Options	Padding

Field	Meaning
Source Port	Identifies the application process on the sending computer that sent the data.
Destination Port	Identifies the application process on the receiving computer for which the data is intended.
Sequence Number	Identifies the first byte of the sent data for the purposes of allowing the receiver to acknowledge receipt of the data and to reorder data as necessary.
Acknowledgement Number	Set in a sent TCP segment, this number notes the sequence number of the next byte the host expects to receive. The Acknowledgement Number field recognises lost packets and flow control.
Header Length	Number of sets of 4 bytes in the TCP header, which allows the receiving host to easily find the end of the TCP header and the data's beginning.
Reserved	Reserved for future use.
Flags	Each bit has different meanings to signal some function. For example connection establishment uses the SYN and ACK flags.
Window	As set in a sent segment, Window signifies the maximum amount of unacknowledged data the host is willing to receive before the other host must wait for an acknowledgement. Used for flow control.
Checksum	Frame Check Sequence (FCS)-like field that can confirm that no errors occurred in the TCP header.
Urgent	Used to point to the sequence number of send data for which the send requests an immediate (urgent) acknowledgement from the receiver.
Options	Additional headers used to expand the protocol in the future. It is seldom used today.
Padding	

User Datagram Protocol (UDP)

Definition

The **User Datagram Protocol (UDP)** is a network communication protocol designed to send data packets over the Internet. It provides ow-latency and loss tolerating connections between applications running on hosts communicating by an IP network.

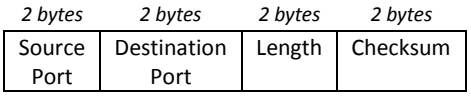
Sending Data using UDP

The data is sent without any prior communications as to how the data should be transmitted.

TCP/IP Stack

Protocol Header

Diagram: TCP Protocol Header



Field	Meaning
Source Port	Identifies the application process on the sending computer that sent the data.
Destination Port	Identifies the application process on the receiving computer for which the data is intended.
Length	Number of octets in the UDP segment, including the data.
Checksum	Frame Check Sequence (FCS)-like field that can confirm that no errors occurred in the UDP header.

TCP and UDP Comparison

Comparison

	TCP	UDP
Connection Type	Connection-oriented	Connectionless
Sequencing	Yes	No
Uses	TCP is used in cases where the loss of data during transmission is detrimental to the consumption of the data at the receiving end. • Email • File sharing • Downloading	TCP is used in cases where the loss of data during transmission is not important or discernible during consumption of the data at the receiving end. • Voice streaming • Video streaming

Internet Protocol

Definition

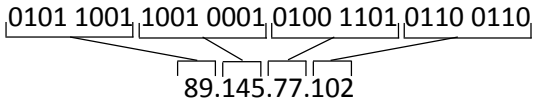
An **Internet Protocol (IP)** address is a numerical address that is assigned to and uniquely identifies a device on a network. The IP address indicates where a packet of data is to be sent or from where it was sent. The IP address indicates where a packet of data is to be sent or from where it was sent. Routers can use IP addresses to direct a packet to its destination.

Network devices recognise IP addresses as binary numbers.

IPv4

Internet Protocol v4 (IPv4) allows each device on a network to have a unique 32-bit binary number address. The binary number is broken down into octets, which are chunks of 8 digits, and each octet is separated by a dot.

Example: IPv4 Address



IPv4 can assign addresses for up to 4.3 billion computer systems. However, there are a growing number of computer systems and therefore more addresses will be required to allow all devices to be connected to the internet – IPv6 was introduced to solve this problem.

IPv6

Internet Protocol v6 (IPv6) allows each device on a network to have a unique 128-bit binary number address. The binary number is arranged into groups of hexadecimal numbers and each group is separated by colons.

Example: IPv6 Address

0000 0000 0000 1010 0000 1011 1010 1101 0000 1101 1010
1101 1111 0000 0000 1101 1111 1010 1100 1110 1101 0000
0000 1101 0000 0000 0000 0000 0000 0000 0000 0000

↓

000A:0BAD:0DAD:F00D:FACE:D00D:0000:0000

IPv6 addresses can be simplified by omitting leading zeros in a group and represent groups of zeros using two colons (::).

Example: Simplified IPv6 Address

000A:0BAD:0DAD:F00D:FACE:D00D:0000:0000

↓

A:BAD:DAD:F00D:FACE:D00D::

This works because:

- if there are two colons, it can be assumed that the group is four zeros; and
- if there are not four characters in a group, it can be assumed that the omitted leading characters are zeros.

TCP/IP Stack

IPv6 can assign addresses for up to 2^{128} computer systems and therefore solves the issue of scarcity of IP addresses.

IPv4 and IPv6 can both coexist and therefore both be used by computer systems on the Internet. However, a packet may only be IPv4 or IPv6.

Assigning IP Addresses

An IP address can be either:

- public – used to assign a router to the Internet and can be accessed through the Internet, usually assigned by an Internet Service Provider (ISP);
- private – used to assign a computer system to a route and cannot be accessed directly through the internet, usually assigned by a router using DHCP or be manually set.

Network Protocols in Web Browsing (Lab)

Configuring a Network

Network Setup

A simple network can be setup using a network cable, known as an Ethernet cable.

Instructions: Setting up a Network on Windows 10

1) Click the "network" icon in the bottom-right corner of the screen.
2) Click "Network & Internet Settings".
3) In the new window, click Ethernet on the left-hand side menu.
4) Click "Change adapter options" on the right-hand side.
5) In the new window, double click "Ethernet".
6) Click "Properties".
7) Double-click "Internet Protocol Version 4 (TCP/IPv4).
8) Choose "Use the following IP address" and configure the IP addresses:
    - choose an IP address for the first computer, such as 192.168.1.2; and
    - choose an IP address for the second computer, such as 192.168.1.3.
9) Click "OK" and close all windows.

Wireshark

Wireshark is an application that enables packets (or frames) on the computer system's network interfaces to be captured and observed.

Instructions: Using Wireshark

1) Double-click the network adapter "Ethernet".
2) Observe the packets being captured.

Example: Wireshark Capture

No.	Source	Destination	Protocol	Info
1	10.10.8.7	10.10.124.12	TCP	41951> http [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2	10.10.124.12	10.10.8.7	TCP	http > 41951 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460
3	10.10.8.7	10.10.124.12	TCP	41951> http [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	10.10.8.7	10.10.124.12	HTTP	GET / HTTP/1.1
5	10.10.124.12	10.10.8.7	HTTP	HTTP/1.1 200 OK (text/html)
6	10.10.8.7	10.10.124.12	HTTP	GET /image.gif HTTP/1.1
7	10.10.124.12	10.10.8.7	TCP	[TCP segment of a reassembled PDU]
8	10.10.124.12	10.10.8.7	TCP	[TCP segment of a reassembled PDU]
9	10.10.124.12	10.10.8.7	HTTP	HTTP/1.1 200 OK (GIF89a)

In this Wireshark capture:

- the IP address of the browser PC is 10.10.8.7;
- frames 1, 2 and 3 are involved with the connection setup as they perform the three-way handshake; and
- frame 6 contains the image file data.

Network Protocols in Web Browsing (Lab)

Web Server

Setting up the Website

A website requires a designated HTML file which usually resides within a directory specifically for the purpose of the website.

Instructions: Creating a Website Directory

1) Open "Command Prompt".
2) Make a directory called "website" using the command `md website`.
3) Enter the directory `website` using the command `dir website`.
4) Using notepad, create a file containing the code below and save it in the `website` directory with the file name `index.html`.

A web page is structured using HyperText Markup Language (HTML).

HTML Code: Simple Web Page

```
<html>
	<head>
		<title>Networks Lab Webpage</title>
	</head>

	<body>
		<p>Text that you should be able to see transferred over the
		network.</p>
	</body>
</html>
```

Setting up the Web Server

Python 2.7 is capable of running a web server.

Instructions: Setting up a Web Server

1) Open "Command Prompt".
2) Enter the directory `website` using the command `dir website`.
3) Start Python's SimpleHTTPServer using the code `C:\Python27\python -m SimpleHTTPServer 80`.
4) The website can now be viewed on the computer system's network by entering the IP address of the host in a web browser.

Networks and Programs (Lab)

Simple Chat Program

Setting up the Chat Program

Python 2.7 is capable of running a chat program.

Instructions: Setting up a Web Server

1) Open the python 2.7 (command line).
2) Setup the socket by executing the code below.
3) Send or receive messages through the socket by executing the code below.

Python Code: Setting up the Socket

```
# The socket module contains all the necessary details that allow us to 'connect'
# to the networking protocol stack in the operating system and create a socket
# for our communication.
import socket

# We create a new socket object called s.
# In this example, we ask it to use the UDP protocol (instead of TCP).
s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

# We tell our socket object which IP address (it should be the IP address you
# have configured on your PC; it must be in quotes) and port number we want it
# to use (let's use 50000 for our program but we could choose any unused
# value greater than 49151 but less than 65536 - or other unused value <49151).
# Note the address => IP address + port number is in its own brackets.
s.bind(('your IP address', your port number))
```

Python Code: Sending Messages through the Socket

```
# Method (function) of socket objects called sendto, passing parameters - the data
# and address (IP + port) of the remote host. To see if anything has been received
# from the network we can use the recv method.
s.sendto(b'message',('destination IP', destination port))
```

Python Code: Sending Messages through the Socket

```
# Asks the networking software stack for any data (up to 1024 bytes) received from
# the network on our socket object's port and returns it to the program
s.recv(1024)
```

Multiple hosts can be introduced to the network by using a switch. In this configuration, a host can broadcast a message using the reserved host ID (1.1 / 11111111 11111111) to broadcast a message to all other hosts on the network.

Networks and Programs (Lab)

Using a Switch

Definition

A **switch** is a piece of hardware which connects multiple devices together in a network using a physical connection. When a packet is received, the destination address in the packet's header is examined and the packet is only re-broadcast to the device to which it was intended to be sent.

How it works

In wired networks, a switch is used to make physical networks of more than two hosts.

Process: Network Communication using a Switch
4) Packets (or frames) are passed to a physical port on the switch. 5) The destination address in the packet's header is examined. 6) The packet is re-broadcast to the correct host connected to the switch.

Hosts on the network need not be concerned with the switch as it is not an important factor to their method of sending and receiving data on the network. A host on the network need only be concerned with using their network interface card (NIC) to send and receive on the network.

Switches are not concerned with IP networks as they operate on protocol layers below IP, such as the physical addressing layers.

Network Addressing and Routing

Address Mapping

Why is address mapping required?

A **domain name** is a unique name that identifies a website. For example, google.com.

Address mapping is required to turn the name for human use, such as a domain name or computer name, to the corresponding IP address. This can be achieved by using hosts files or a domain name server (DNS).

Address mapping is important as it makes the Internet easier to use for clients as they are able to request a web page using a URL. This is user-friendly and intuitive since URLs are written using natural language and prevents the need for clients to remember an IP address, which are 4 bytes of numbers in IPv4 and 16 bytes of hexadecimal in IPv6.

Hosts File

Definition

A **hosts file** is a plain text file, generated by the operating system, that maps hostnames to IP addresses.

Contents

A hosts file contains the IP address and the respective domain name.

Example: Typical Hosts File

```
127.0.0.1   localhost loopback
::1         localhost
```

Issues

Using the hosts file to map all addresses on the Internet is not feasible as:

- there are millions/billions of mappings between domain names and IP addresses, meaning that the hosts file would be too large; and
- mappings between domain names and IP addresses are often changes and therefore the hosts file is likely to become out-of-date.

Network Addressing and Routing

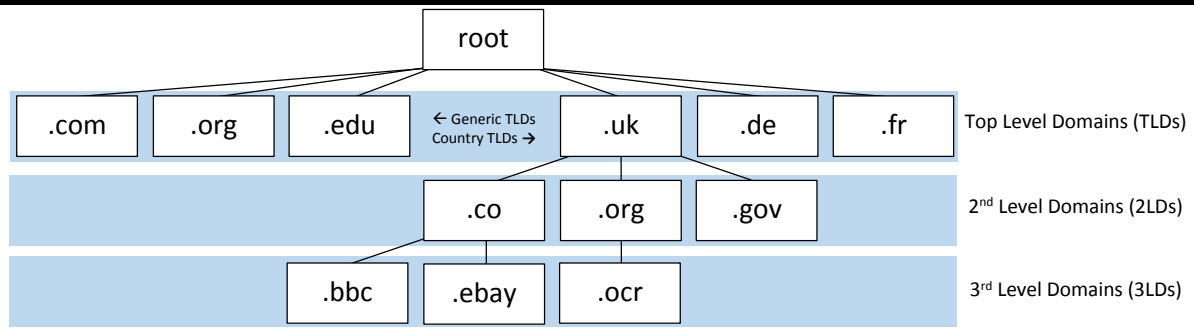
Domain Name Server (DNS)

Definition

A **Domain Name System (DNS)** provides the rules for assigning domains names to IP addresses by structuring the domain names into a hierarchy of smaller domains, the smaller domains are written as a string and separated by full stops.

Hierarchy

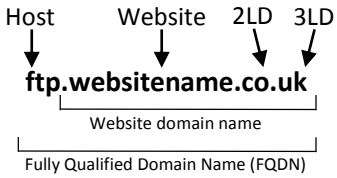
Diagram: DNS Hierarchy



Domain Names

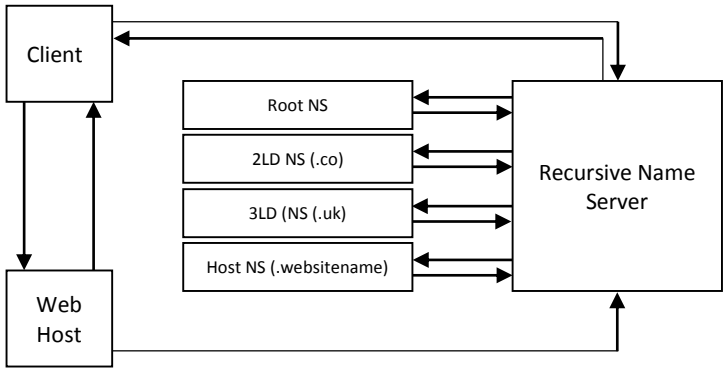
Each domain name has one or more equivalent IP addresses. A DNS server must then retrieve the correct corresponding IP address.

Diagram: Domain Name Structure



A **name server** contains a database of corresponding domain names and IP addresses.

Diagram: Resolving a Web Page

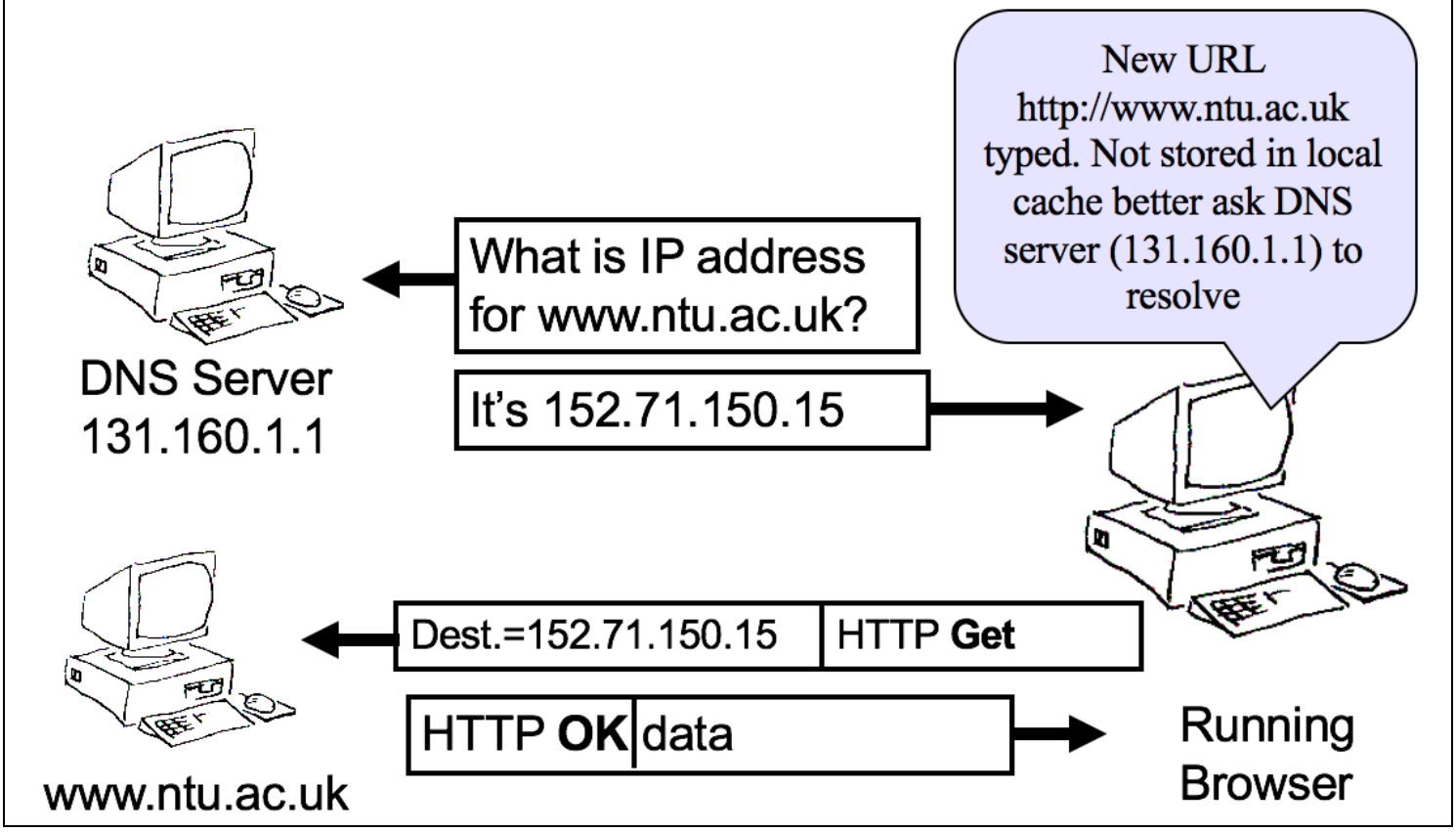


resolving www.websitename.co.uk

- Client inputs a domain name into their web browser.
- Web browser cache is checked for matching web page.
- If web page is not in the cache, domain is transferred to the recursive name server, which is owned by the Internet Service Provider (ISP) and contains the domain names and corresponding IP addresses for web pages previously requested.
- If the web page is not in the recursive name server, the recursive name server will perform queries on the root, 2LD name server, 3LD name server and the web host's name server until a match is found and the domain has been resolved.
- When the domain has been resolved, the name server containing the domain will return the IP address to the recursive name server that returns the IP address to the client's web browser.
- The client's web browser makes a data request to the web host server, if the page is found, the web elements, such as: HTML; CSS; and JavaScript, will be transmitted back to the client's web browser and displayed to the client.

Network Addressing and Routing

Diagram: Requests made when Resolving a Web Page



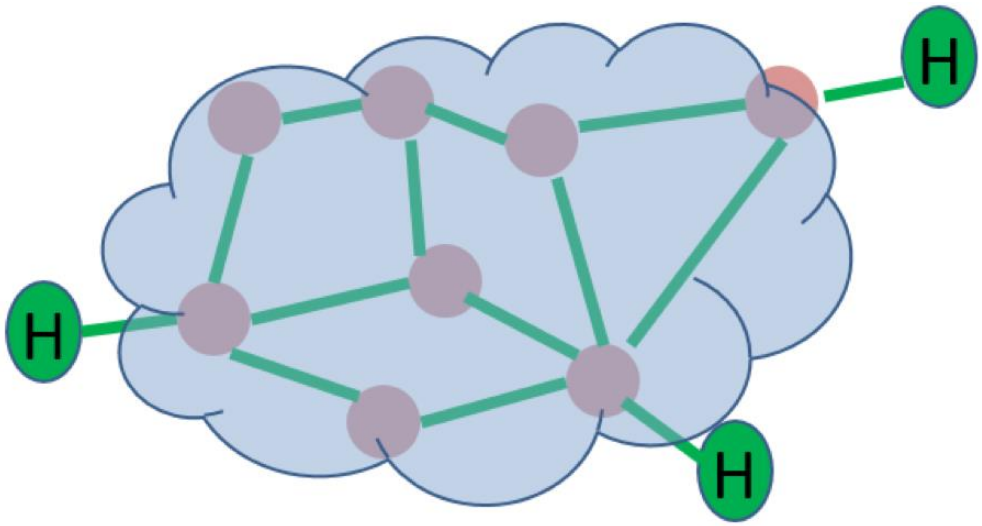
Network Addressing and Routing

Routing Packets

Packet Hops

When a packets is transmitted across the Internet, it is routed through a number of routers connected to the Internet.

Diagram: Routers on the Internet



Packets will “hop” from one router to another when being transmitted. There must be an efficient method of finding the best next “hop” for a given packet – this is achieved using routing tables.

Routing Tables

Definition

A **routing table** is a data table, stored in a router or a networked computer, that lists the routes to particular network destinations.

Possible Approach #1

One possible approach to designing a routing table would be to have a table such that each possible destination is stored along with a corresponding 'next hop'.

Diagram: Routing Table

Destination	Next Hop
1.1.1.1	R1
2.2.2.2	R2
1.2.3.4	R2
....	....

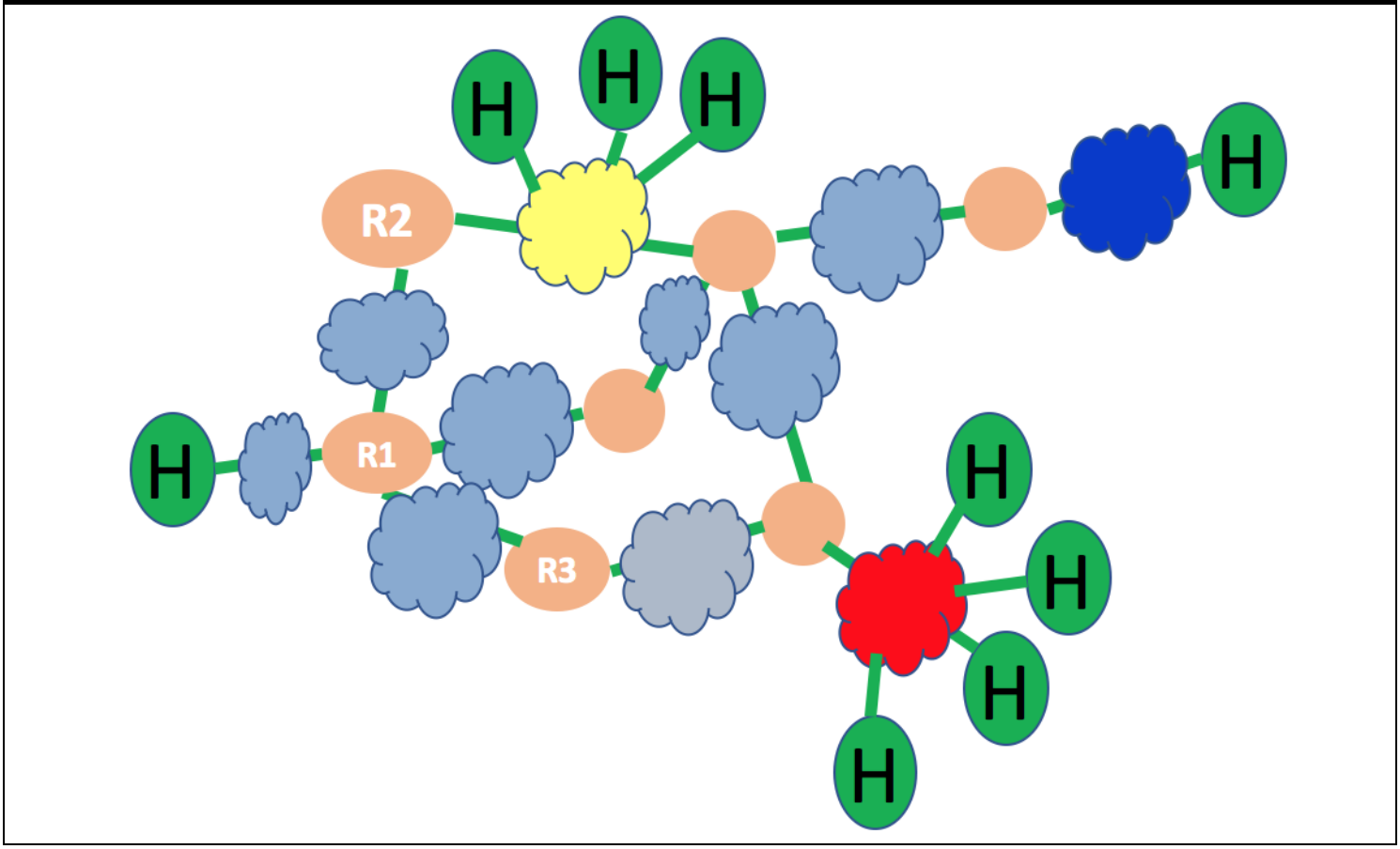
However, this is likely to yield issues as there are potentially 4.3 billion entries in IPv4 and more entries in IPv6. This would make the routing table infeasible to maintain as it would not be kept up-to-date.

Network Addressing and Routing

Possible Approach #2

This approach uses the idea that the Internet is a network of networks.

Diagram: The Internet as a Network of Networks



The Internet can be divided into a number of smaller networks. The diagram visualises these networks using the colours red, blue and yellow.

Diagram: Routing Table

Routing Table for R1

Destination	Next Hop
Yellow	R2
Red	R3
Blue	R2

If the IP address indicates that the destination is on a particular network, the packet can be passed to the corresponding "next hope".

For example, the routing table shows that any packet with a destination IP address that resides within a host connected to the network with the colour code red should be passed to the router R3. It is assumed that the router in the network, which is closer to the destination network or is the actual destination network for the packet, has more information regarding how to effectively route a given packet.

Network Addressing and Routing

Default Gateway

A **default gateway** allows computer systems on a network to communicate with computer systems on another network, such as a router.

As the default gateway is the only interface for communicating with computer systems on another network, it is not necessary to have a routing table that contains individual routes to all possible networks.

Diagram: The Internet as a Network of Networks

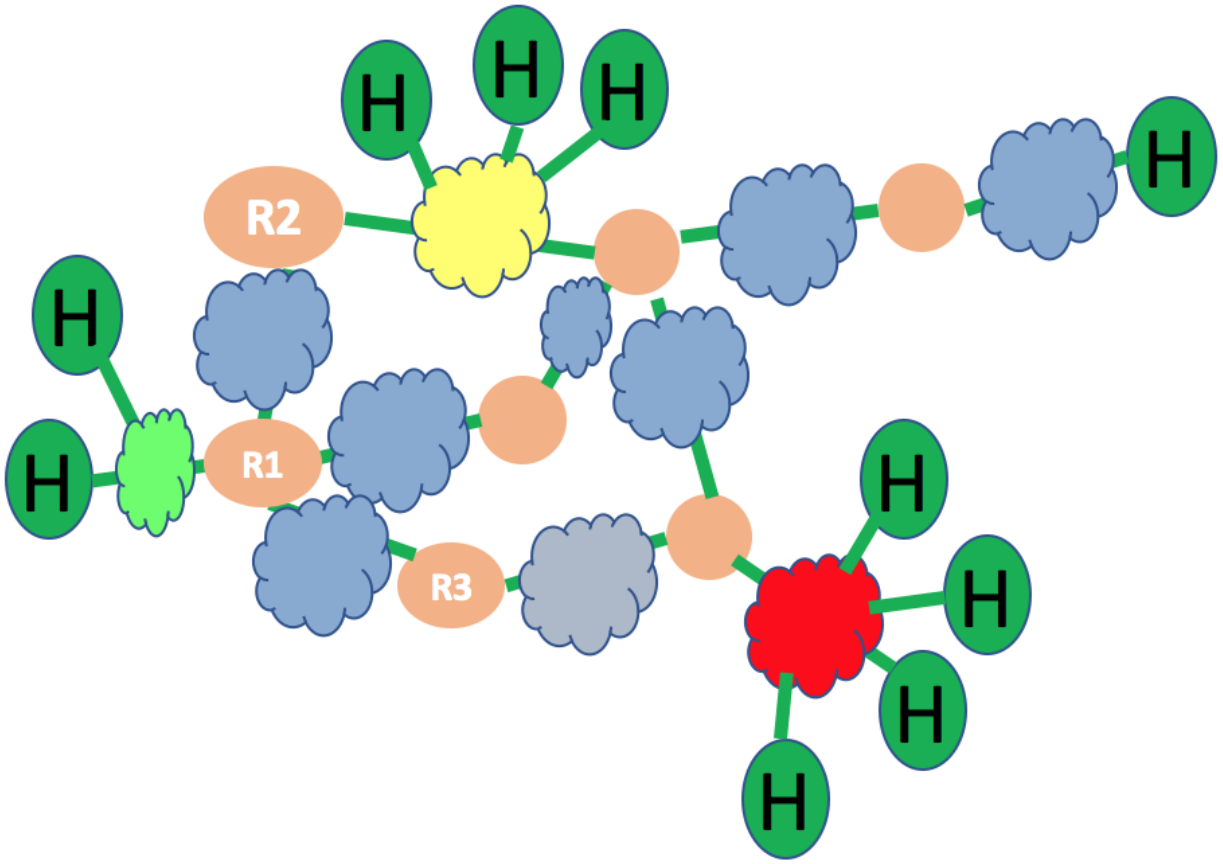


Diagram: Routing Table

Routing Table	
Destination	Next Hop (gateway)
Green	Locally attached/ deliver direct
Any other network	R1

Considering the example where packets are being sent from the network colour coded green, the routing table can consist of the following entries:

- a "next hop" for local routing – local routing occurs when sending a packet from one computer system to another computer system on the same network and therefore it can be delivered directly rather than through the default gateway; and
- a "next hop" for external routing – External routing occurs when sending a packet from one computer system to another computer system on another network and therefore it must be sent to the default gateway in order to interface with the Internet.

Network Addressing and Routing

Network Masks

Definition

A **network mask (netmask)** is a 32-bit binary number used to identify the how many bits are used to represent the network identifier (network ID) and therefore how many bits are remaining in order to represent the host identifier (host ID).

Why are network masks required?

Network masks are required because, when routing a packet, it is important to know:

- how to get to the network containing the destination host; and
- how to get to the host within the network.

Format

An IPv4 address is represented using 32 bits.

In IPv4, 1.1.1.1 would be represented as 00000001 00000001 00000001 00000001.

Format of a IPv4 Address and Network Mask

IPv4 Address	00000001	00000001	00000001	00000001
Network Mask	11111111	11111111	11111111	11111111
Result	00000001	00000001	00000001	00000000

The network mask is:

- set to all one's (11111111) in the octets where the network ID (prefix) will be represented; and
- set to all zero's (00000000) in the octets where the host ID (suffix) will be represented.

The result comes from performing an AND operation on the IPv4 address and the network mask. This shows that for any address on the network such that the IPv4 address satisfies 1.1.1.x, the route is required for the network with the IPv4 address 1.1.1.0.

Network Masks in Routing Tables

The network mask is used in routing tables.

Example: Routing Table

Destination	Mask	Next Hop
1.1.1.0	255.255.255.0	Deliver direct
2.2.2.0	255.255.255.0	Deliver direct
3.0.0.0	255.0.0.0	1.1.1.1
1.2.3.96	255.255.255.224	2.2.2.2

Network Addressing and Routing

The network mask has a written represented using denary to improve readability, for example:

- 255 is equivalent to 11111111; and
- 0 is equivalent to 00000000.

However, the processing is completed in binary.

The mask is defined when allocating addresses and configuring routing tables and therefore it cannot be determined from the address alone.

Notation

It is common to write an IPv4 address and network mask using a short-form notation.

Example: Notation
1.1.1.1 /16

The notation consists of:

- the IPv4 address; and
- the number of bits used in the IPv4 address to identify the network ID.

History

In the original internet, a mask could be determined from the CLASS of an address, which would be represented as the first few bits of the address. For example, addresses beginning with the bits 10 are class B and have the mask 255.255.0.0.

Nowadays, masks can have any number (less than 32) of prefix / suffix bits. This offers more flexibility and in turn allows more IP addresses to be allocated.

Network Addressing and Routing

Address Allocation

Network IDs and Host IDs

Hosts on a network must have:

- the same network ID (prefix, identified by the network mask); and
- a different host ID (suffix).

Reserved Host IDs

The following host IDs are reserved in IPv4:

- 0.0 *(00000000 00000000)* – address of the network itself; and
- 0.0 *(00000000 00000000)* – broadcast address, used to send packets to all hosts on the network.

IP Addressing, Internetworks and Security (Lab)

Creating an Internet

Setting up the Networks

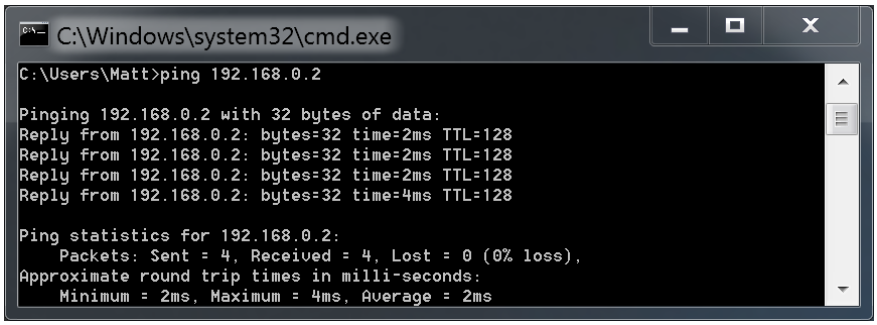
Instructions: Setting up Two Networks

1) Identify a computer system to act as a router.
2) Using a USB Dual Gigabit Ethernet Adapter:
   - connect the adapter to the computer system's Ethernet port;
   - connect one Ethernet cable from the adapter to a switch; and
   - connect the other Ethernet cable from the adapter to another computer system.
3) Connect another two computer systems to the switch using Ethernet cables.
4) Configure the IP addresses for the host computer systems on the network, assuming the network will be 101.202.0.0/16 by choosing appropriate IP addresses (such as 101.202.0.1 and 101.202.0.2).

Checking Connectivity

A host on the network can be checked to see if it is connectable by using the `ping` command.

Example: Using the `ping` command



At this stage, the hosts are unable to ping hosts on the remote network. This can be resolved by "bridging" the two networks.

Bridging the Networks

A host on the network can be checked to see if it is connectable by using the `ping` command.

Instructions: Setting up Two Networks

1) On the router computer system, open the "Network Connections" window.
2) Select the two network interfaces.
3) Right-click on the selected network interfaces.
4) Select "Bridge Connections".

This process will allow hosts on both networks to be connectable and therefore can be "pinged".

IP Addressing, Internetworks and Security (Lab)

Communication Security

Eavesdropping

By using Wireshark, it is possible to see the packets being transmitted across the network. This can be used to intercept information being sent across the network, such as messages being sent using the Python chat program (see page 141).

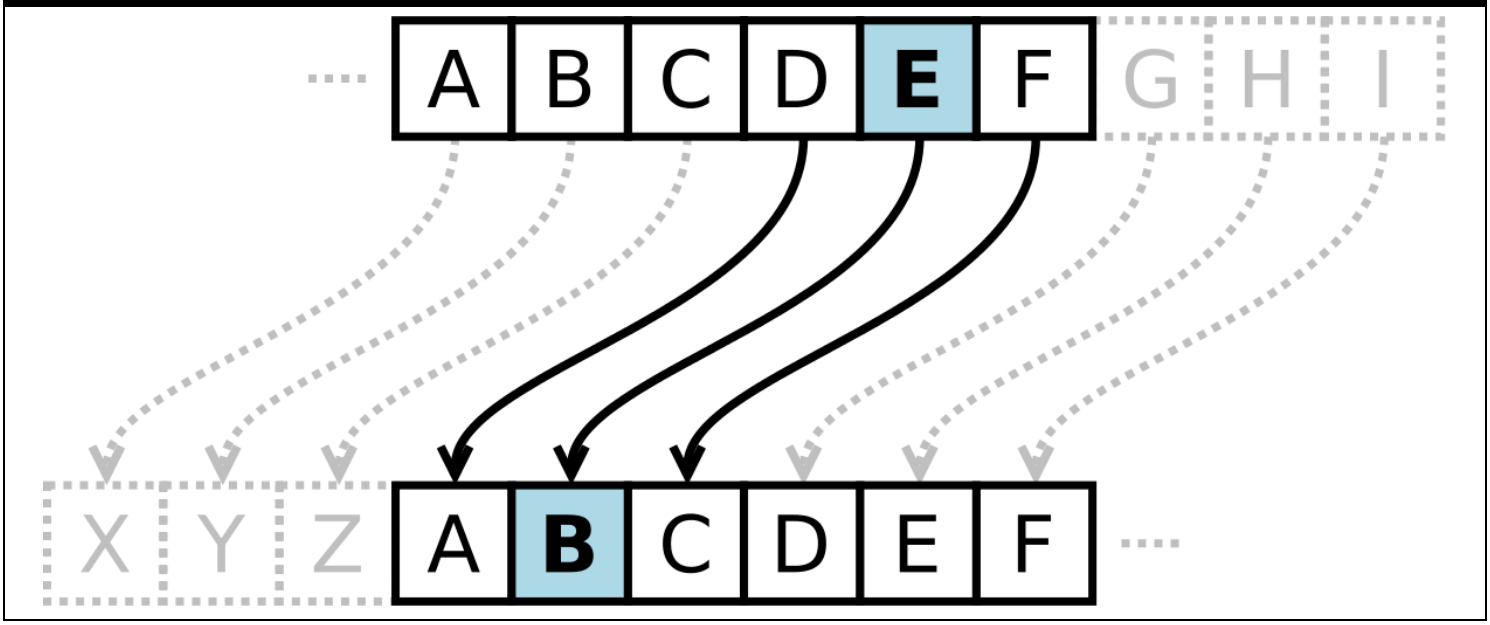
If packets contain confidential information, it is necessary to employ methods of obfuscating the information.

Encryption

If messages are going to be sent over a network, it is possible to encrypt the message by using a Caesar cipher.

The action of a Caesar cipher is to replace each plaintext letter with a different one a fixed number of places down the alphabet. This converts messages into ciphertext, that only the intended recipient of the message has the knowledge of how to decipher the message by changing it back to its original form. To achieve this the sender and receiver must agree the cipher and the key to use in advance and keep this information secret.

Diagram: Caesar Cipher



In the diagram, the cipher uses a left shift of three. For example, each occurrence of E in the plaintext becomes B in the ciphertext.

Mediums of Network Data Transfer

Wireless Access Point (WAP)

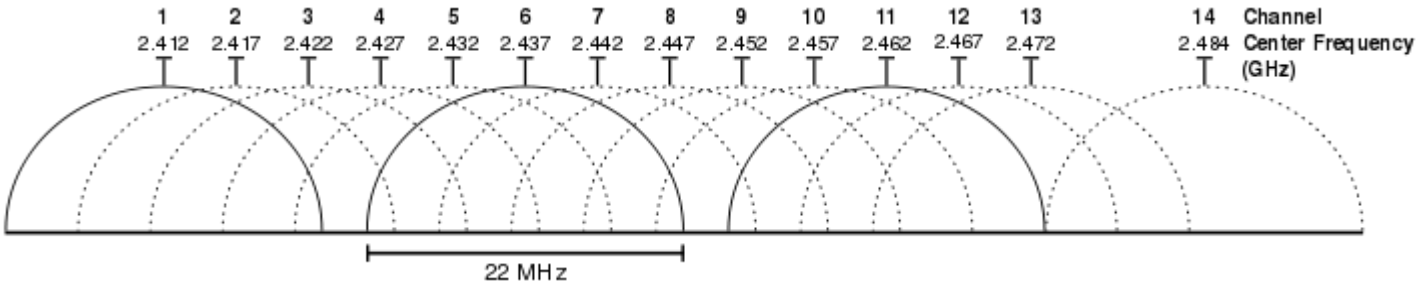
Definition

A **Wireless Access Point (WAP)** is a piece of hardware that has a number of dedicated radio channels which can be received by a computer system with a wireless network adapter. This is used to connect computer systems to a Wi-Fi network which is a local area wireless technology.

How it works

A WAP uses radio channels to broadcast to connected devices and can operate on the 2.4GHz radio frequency, with 14 channels.

Diagram: 2.4GHz Radio Frequency



There are two types of channels:

- overlapping channels – these channels share some of the frequency in the 2.4GHz band (2, 4, 5, 7, 8, 9, 10, 12, 13, 14) and so there is likely to be interference because signals from other channels are going to be present; and
- non-overlapping channels – these channels are separated from other channels by 5MHz in the 2.4GHz band (1, 6, 11) and so there is less likely to be interference because signals from other channels are not present.

However, the 2.4GHz band has become saturated with existing WAPs and therefore modern WAPs offer the option to broadcast signals to devices, with 5GHz compatible wireless network adaptors, on the 5GHz band which has a high number of channels.

Evaluation

Advantages	Disadvantages
Allows portable devices to be connected to the network because no physical connection is required.	**Slower than a dedicated Ethernet connection**.
Less network hardware may be required because devices that are connected wirelessly will not require an Ethernet cable.	**Can only support so many connections** because there are a finite number of radio channels.
	Can be insecure because access to the network can be obtained using a password rather than a physical connection.

Mediums of Network Data Transfer

Fibre Optic

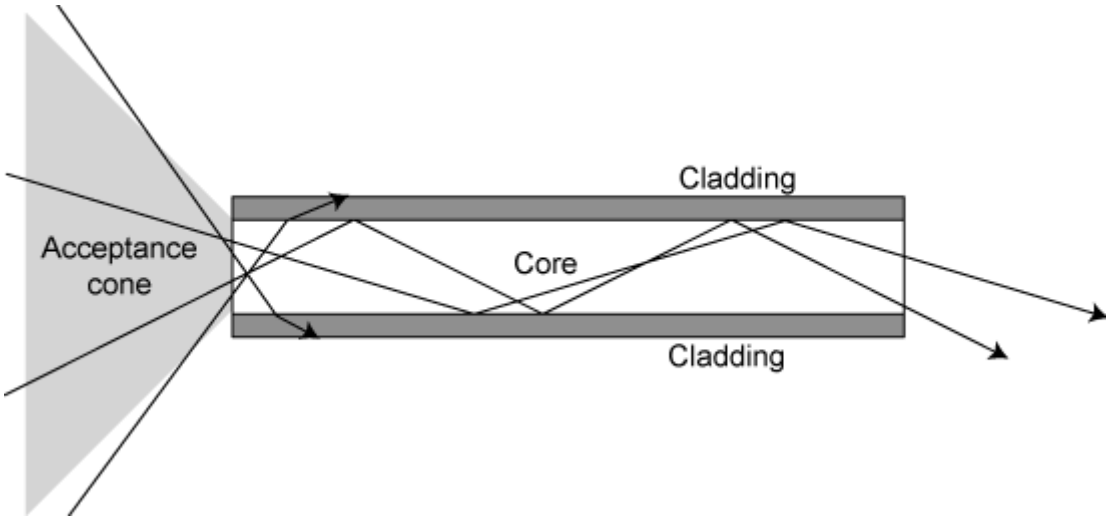
Definition

Fibre optic is a physical technology that uses glass fibres to transmit data.

How it works

Fiber optics transmit data in the form of light particles (or photons) that pulse through a fiber optic cable.

Diagram: Optical Fibre Cable



The glass fiber core and the cladding each have a different refractive index that bends incoming light at a certain angle. When light signals are sent through the fiber optic cable, they reflect off the core and cladding in a series of zig-zag bounces.