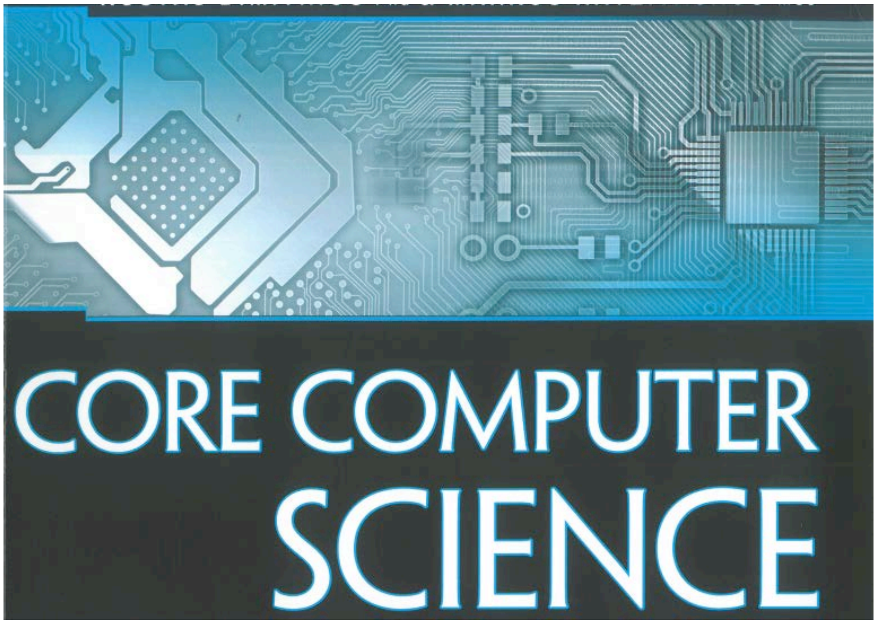


Course Resources

Mr. Alei's web page

DP Compute Science Sylllabus

Michael Brooke Online Course Notes

Coursera

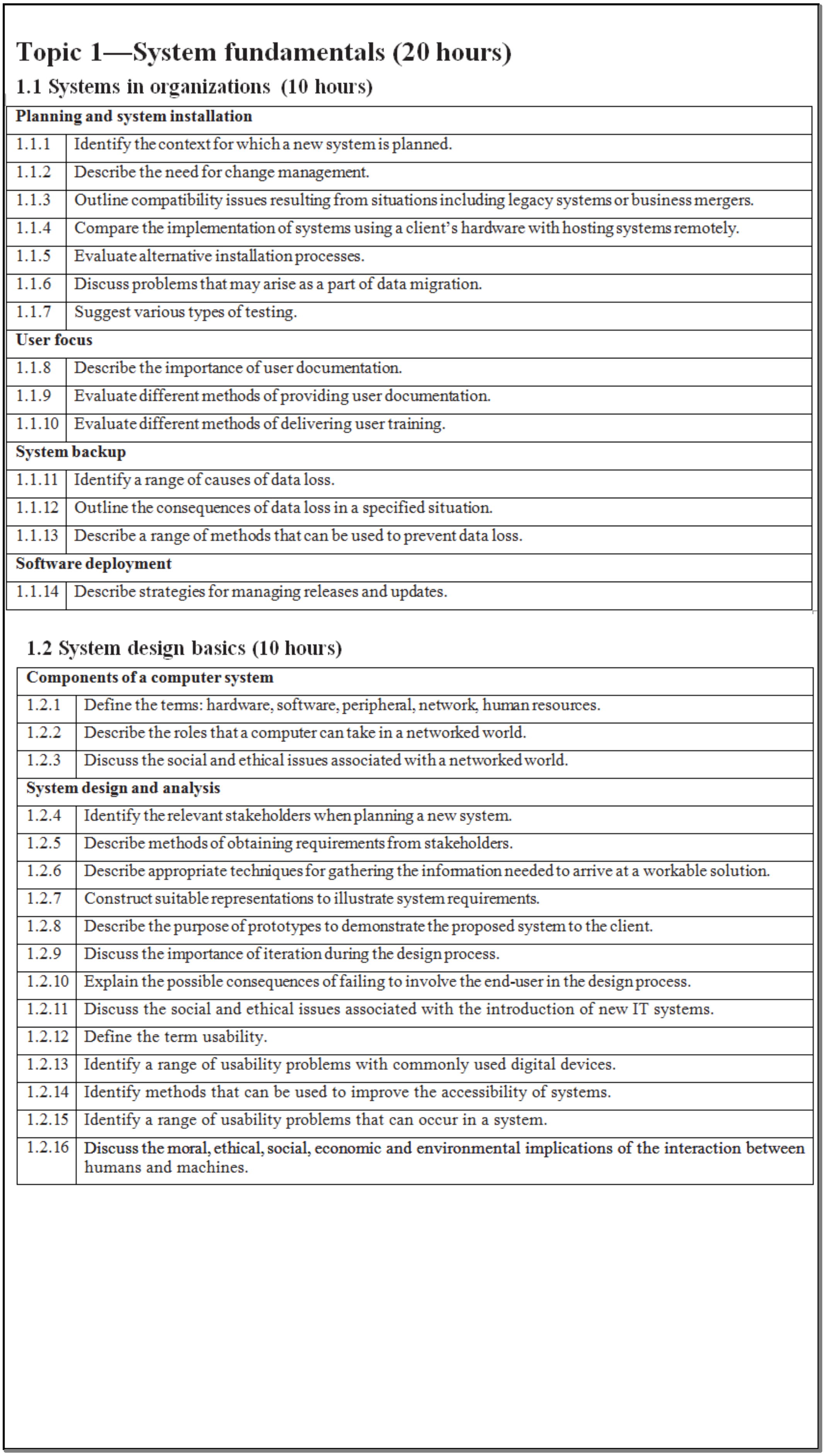
1.1 System Organization

1.2 System Design

2. Computer Architecture

3. Networks

4. Computational Thinking



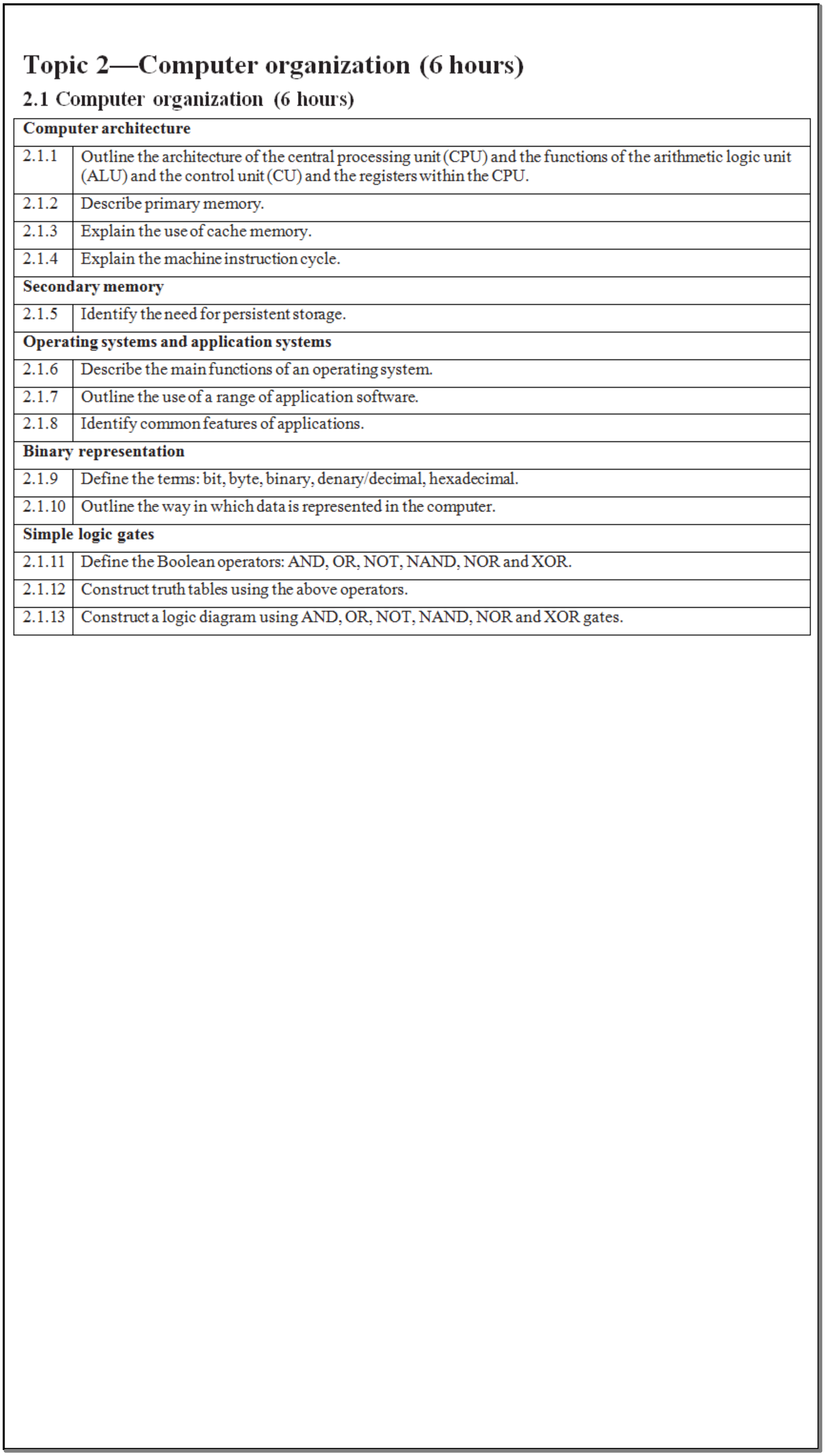
Topic 1—System fundamentals (20 hours)

1.1 Systems in organizations (10 hours)

Planning and system installation	
1.1.1	Identify the context for which a new system is planned.
1.1.2	Describe the need for change management.
1.1.3	Outline compatibility issues resulting from situations including legacy systems or business mergers.
1.1.4	Compare the implementation of systems using a client's hardware with hosting systems remotely.
1.1.5	Evaluate alternative installation processes.
1.1.6	Discuss problems that may arise as a part of data migration.
1.1.7	Suggest various types of testing.
User focus	
1.1.8	Describe the importance of user documentation.
1.1.9	Evaluate different methods of providing user documentation.
1.1.10	Evaluate different methods of delivering user training.
System backup	
1.1.11	Identify a range of causes of data loss.
1.1.12	Outline the consequences of data loss in a specified situation.
1.1.13	Describe a range of methods that can be used to prevent data loss.
Software deployment	
1.1.14	Describe strategies for managing releases and updates.

1.2 System design basics (10 hours)

Components of a computer system	
1.2.1	Define the terms: hardware, software, peripheral, network, human resources.
1.2.2	Describe the roles that a computer can take in a networked world.
1.2.3	Discuss the social and ethical issues associated with a networked world.
System design and analysis	
1.2.4	Identify the relevant stakeholders when planning a new system.
1.2.5	Describe methods of obtaining requirements from stakeholders.
1.2.6	Describe appropriate techniques for gathering the information needed to arrive at a workable solution.
1.2.7	Construct suitable representations to illustrate system requirements.
1.2.8	Describe the purpose of prototypes to demonstrate the proposed system to the client.
1.2.9	Discuss the importance of iteration during the design process.
1.2.10	Explain the possible consequences of failing to involve the end-user in the design process.
1.2.11	Discuss the social and ethical issues associated with the introduction of new IT systems.
1.2.12	Define the term usability.
1.2.13	Identify a range of usability problems with commonly used digital devices.
1.2.14	Identify methods that can be used to improve the accessibility of systems.
1.2.15	Identify a range of usability problems that can occur in a system.
1.2.16	Discuss the moral, ethical, social, economic and environmental implications of the interaction between humans and machines.



Topic 2—Computer organization (6 hours)

2.1 Computer organization (6 hours)

Computer architecture	
2.1.1	Outline the architecture of the central processing unit (CPU) and the functions of the arithmetic logic unit (ALU) and the control unit (CU) and the registers within the CPU.
2.1.2	Describe primary memory.
2.1.3	Explain the use of cache memory.
2.1.4	Explain the machine instruction cycle.
Secondary memory	
2.1.5	Identify the need for persistent storage.
Operating systems and application systems	
2.1.6	Describe the main functions of an operating system.
2.1.7	Outline the use of a range of application software.
2.1.8	Identify common features of applications.
Binary representation	
2.1.9	Define the terms: bit, byte, binary, denary/decimal, hexadecimal.
2.1.10	Outline the way in which data is represented in the computer.
Simple logic gates	
2.1.11	Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR.
2.1.12	Construct truth tables using the above operators.
2.1.13	Construct a logic diagram using AND, OR, NOT, NAND, NOR and XOR gates.

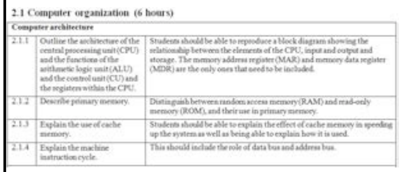
Mr. Alei's web page Michael Brooke Online Course Notes Coursera

CORE COMPUTER SCIENCE

2 - Computer Architecture

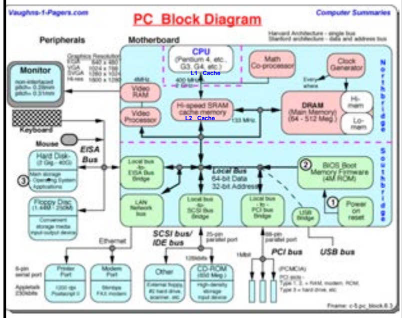
2.1.1 CPU
2.1.2 RAM & ROM
2.1.3 Cache Memory
2.1.4 Instruction Cycle

Key concepts



2.1 Computer organization (6 hours)

Computer architecture		
2.1.1	Outline the architecture of the central processing unit (CPU) and the functions of the arithmetic logic unit (ALU) and the control unit (CU) and the registers within the CPU.	Students should be able to reproduce a block diagram showing the relationship between the elements of the CPU, input and output and storage. The memory address register (MAR) and memory data register (MDR) are the only ones that need to be included.
2.1.2	Describe primary memory.	Distinguish between random access memory (RAM) and read-only memory (ROM), and their use in primary memory.
2.1.3	Explain the use of cache memory.	Students should be able to explain the effect of cache memory in speeding up the system as well as being able to explain how it is used.
2.1.4	Explain the machine instruction cycle.	This should include the role of data bus and address bus.



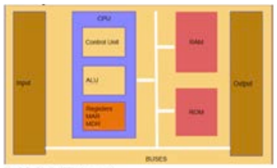
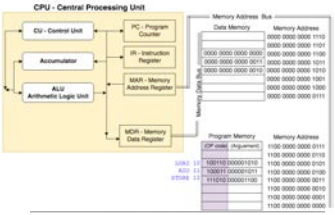
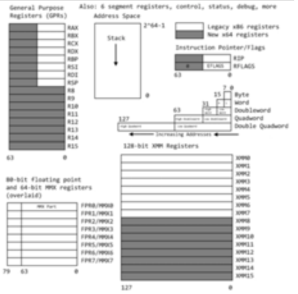


Figure 3 · Block diagram of CPU and main memory

A simplified version at the level of detail you should be able to reproduce

CPU - Central Processing Unit





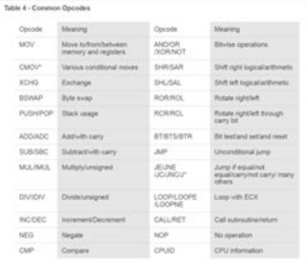


Table 4 - Common Opcodes

Opcode	Meaning	Opcode	Meaning
MOV	Move to/from/between memory and registers	AND/OR /XOR/NOT	Bitwise operations
CMOV*	Various conditional moves	SHR/SAR	Shift right logical/arithmetic
XCHG	Exchange	SHL/SAL	Shift left logical/arithmetic
BSWAP	Byte swap	ROR/ROL	Rotate right/left
PUSH/POP	Stack usage	RCR/RCL	Rotate right/left through carry bit
ADD/ADC	Add/with carry	BT/BTS/BTR	Bit test/and set/and reset
SUB/SBC	Subtract/with carry	JMP	Unconditional jump
MUL/IMUL	Multiply/unsigned	JE/JNE /JC/JNC/J*	Jump if equal/not equal/carry/not carry/ many others
DIV/IDIV	Divide/unsigned	LOOP/LOOPE /LOOPNE	Loop with ECX
INC/DEC	Increment/Decrement	CALL/RET	Call subroutine/return
NEG	Negate	NOP	No operation
CMP	Compare	CPUID	CPU information

```
; Sample x64 Assembly Program
; Chris Lomont 2009 www.lomont.org
extrn ExitProcess: PROC   ; external functions in system libraries
extrn MessageBoxA: PROC
.data
caption db '64-bit hello!', 0
message db 'Hello World!', 0
.code
Start PROC
  sub    rsp,28h      ; shadow space, aligns stack
  mov    rcx, 0       ; hWnd = HWND_DESKTOP
  lea    rdx, message ; LPCSTR lpText
  lea    r8,  caption ; LPCSTR lpCaption
  mov    r9d, 0       ; uType = MB_OK
  call   MessageBoxA  ; call MessageBox API function
  mov    ecx, eax     ; uExitCode = MessageBox(...)
  call ExitProcess
Start ENDP
End
```

2.1.4 Machine Instruction Cycle (YouTube)
Fetch, Decode, Execute, Store

January 28, 2018

Review

Give the name and function of the following acronyms:

CPU -
- **C**entral **P**rocessing **U**nit
- Hardware "brains" of the computer, performs input/output, basic arithmetic & logic operations.

CU
- **C**ontrol **U**nit
- Part of the CPU, it controls retrieval of instructions and data from the primary memory as well as their sequence of execution.

ALU
- **A**rithmetic **L**ogic **U**nit
- Part of the CPU, it performs basic arithmetic, logical, and input/output operations.

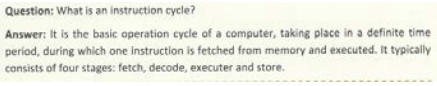
MAR
- **M**emory **A**ddress **R**egister
- Holds the ***address*** of the data being used by the ALU as it performs input/output operations to memory.

MDR
- **M**emory **D**ata **R**egister
- Holds the ***data*** used by the ALU as it performs input/output operations to memory.

Describe the function of the:

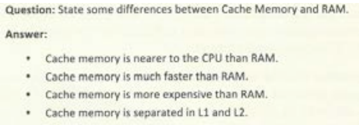
Memory Bus
- The connection for addresses being passed between the memory and the MAR. Also known as the Memory Address Bus

Data Bus
- The connection for data being passed between the memory and the MDR



Question: What is an instruction cycle?

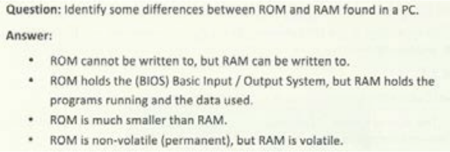
Answer: It is the basic operation cycle of a computer, taking place in a definite time period, during which one instruction is fetched from memory and executed. It typically consists of four stages: fetch, decode, executer and store.



Question: State some differences between Cache Memory and RAM.

Answer:

- Cache memory is nearer to the CPU than RAM.
- Cache memory is much faster than RAM.
- Cache memory is more expensive than RAM.
- Cache memory is separated in L1 and L2.



Question: Identify some differences between ROM and RAM found in a PC.

Answer:

- ROM cannot be written to, but RAM can be written to.
- ROM holds the (BIOS) Basic Input / Output System, but RAM holds the programs running and the data used.
- ROM is much smaller than RAM.
- ROM is non-volatile (permanent), but RAM is volatile.

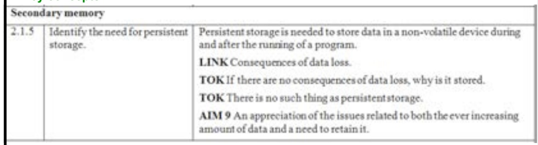
Mr. Alei's web page　　Michael Brooke Online Course Notes　　Coursera

CORE COMPUTER SCIENCE

2 - Computer Architecture

2.1.5 Secondary memory

Key concepts



Secondary memory		
2.1.5	Identify the need for persistent storage.	Persistent storage is needed to store data in a non-volatile device during and after the running of a program. **LINK** Consequences of data loss. **TOK** If there are no consequences of data loss, why is it stored. **TOK** There is no such thing as persistent storage. **AIM 9** An appreciation of the issues related to both the ever increasing amount of data and a need to retain it.

What is ***persistent storage***? How does it differ from other types?
- It is "permanent" or non-volatile - does not disappear on power down.

Name some types of persistent storage.
- Magnetic memory (Hard drives, magnetic tape, floppy disks)
- Solid state (Flash) memory: (SD & CF cards, thumb drives, etc.)
- Optical memory (CD/DVD R/W or ROM)
- Solid state ROM

Image 2.3: An old 5.25 inch floppy disk

Image 2.4: A loaded DVD drive

Image 2.5: An old 3.5 inch floppy disk

Image 2.6: Multiple Compact flash memory cars

Image 2.7: A ZIP drive with a disk

Image 2.8: A Solid state drive for notebook and a hard disk drive for desktop computer

Image 2.9 Various USB sticks

Comparison of Primary and Secondary Memory

Primary Memory
- Expensive
- Fast - SRAM (Fastest)
- Volatile - gone with power off
- Directly accessible by the CPU
- Small amounts of data (a few GB)

Secondary Memory
- Cheap
- Slow
- Permanent (well, semi-permanent)
- Accessed through drivers
- Large amounts of data (a few TB)

Virtual memory

When a computer needs more primary memory, it can use secondary memory as "virtual memory".

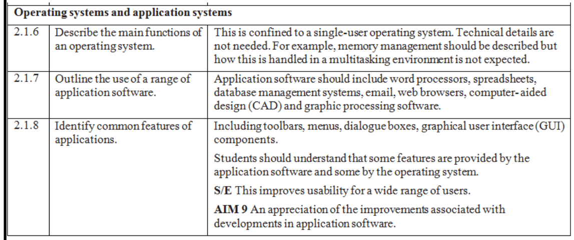
January 28, 2018

Mr. Alei's web page   Michael Brooke Online Course Notes   Coursera

2 - Computer Architecture

2.1.6 Operating systems
2.1.7 Applications
2.1.8 Common Features

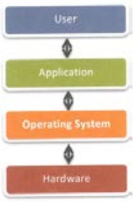
Key concepts



Operating systems and application systems		
2.1.6	Describe the main functions of an operating system.	This is confined to a single-user operating system. Technical details are not needed. For example, memory management should be described but how this is handled in a multitasking environment is not expected.
2.1.7	Outline the use of a range of application software.	Application software should include word processors, spreadsheets, database management systems, email, web browsers, computer-aided design (CAD) and graphic processing software.
2.1.8	Identify common features of applications.	Including toolbars, menus, dialogue boxes, graphical user interface (GUI) components. Students should understand that some features are provided by the application software and some by the operating system. **S/E** This improves usability for a wide range of users. **AIM 9** An appreciation of the improvements associated with developments in application software.

What is an ***operating system (OS)***?
A collection of programs including:

- ***System software***
  - > User/security management
- ***Kernel***
  - > Minimum software required to share hardware between applications
- ***Utilities & Libraries***
  - > Code for applications
- ***Device drivers***
  - > Communicate with external devices
- ***Network connections***
  - > Communicate with external devices
- ***Language translators***



What does an OS do?
Arbitration

1. Load and manage processes
   - > Multi-core processors
2. Manages memory
   - > Virtual memory
3. Manages multitasking
   - > Enables multiple applications to access system hardware "simultaneously"*

Abstraction

1. Provide interface to hardware via *system* calls
   - > Applications can call system routines to
     - – open windows
     - – read mouse position
     - – get user input
     - – write to the monitor
     - – access system variables (date, time, hardware specs, etc.)
2. Provide a file system
   - > Applications call routines to read and write files without worrying about details.
3. Provide a basic user interface.
   - > file navigation and organization
   - > hardware installation and management
   - > internet connectivity
   - > program launching
4. Provide overall security

What are some common operating systems?

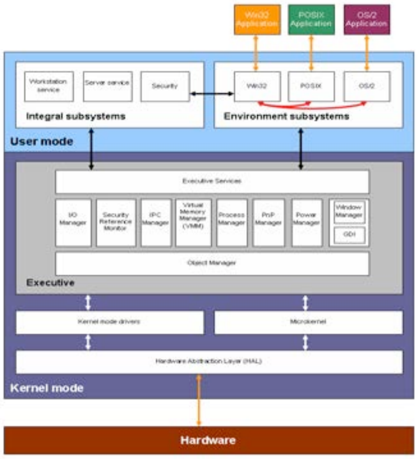
Windows
- A descendant of DOS

Everything else (Unix based)
- Unix - 1970's. The GrandOS
- Mac OSX
- Linux - open source

A good, but very detailed, 23 minute explanation

Mike Murphy: 12:37 - good summary, boring speaker

TechQuickie: 5:15 - humorous, not much detail. Skip the ad at the end!



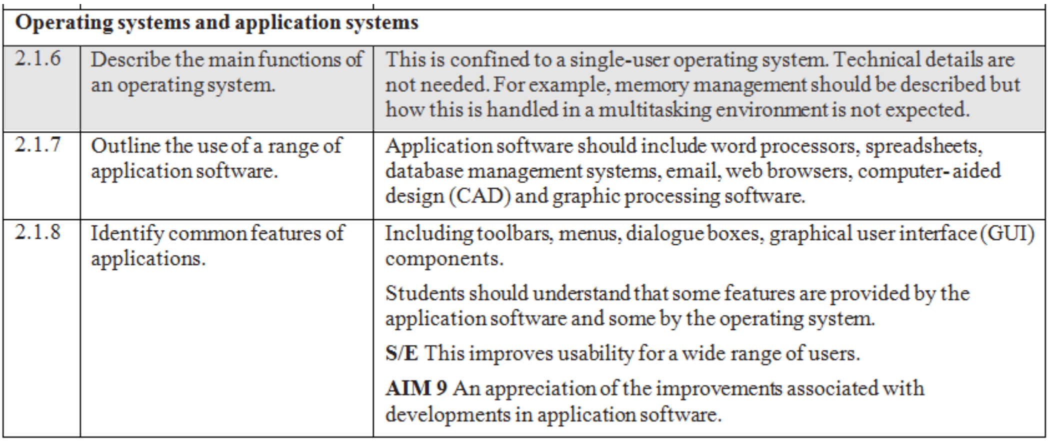
Mr. Alei's web page Michael Brooke Online Course Notes Coursera

CORE COMPUTER SCIENCE

2 - Computer Architecture

2.1.7 Applications
2.1.8 Common Features

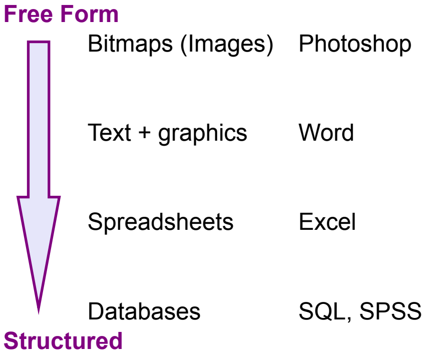
Key concepts



Operating systems and application systems		
2.1.6	Describe the main functions of an operating system.	This is confined to a single-user operating system. Technical details are not needed. For example, memory management should be described but how this is handled in a multitasking environment is not expected.
2.1.7	Outline the use of a range of application software.	Application software should include word processors, spreadsheets, database management systems, email, web browsers, computer- aided design (CAD) and graphic processing software.
2.1.8	Identify common features of applications.	Including toolbars, menus, dialogue boxes, graphical user interface (GUI) components. Students should understand that some features are provided by the application software and some by the operating system. **S/E** This improves usability for a wide range of users. **AIM 9** An appreciation of the improvements associated with developments in application software.

In approximate order of usage
Web browsers
Email
Word processors
Spreadsheets
Graphic processing software.
Database management systems
Computer aided design (CAD)

Data "Structure"



Free Form

Bitmaps (Images)	Photoshop
Text + graphics	Word
Spreadsheets	Excel
Databases	SQL, SPSS

Structured

Common Features?

- Menus
- Toolbars
- Dialogue boxes
- Window management

January 28, 2018

State **one** function of the *operating system* in managing memory. *[1 mark]*

*Award **[1 mark]** for a function stated.*
Allocating storage for data and instructions;
Keeping track of free and occupied parts of memory; ***[1 mark]***

State **one** example of *application software*. *[1 mark]*

*Award **[1 mark]** for a valid example.*
Word processor; spreadsheet; database management system; e-mail; web browser; CAD; graphic processing software; ***[1 mark]***

The operating system in the latest mobile phones allows the user to open more than one application at the same time.

(a) State **three** possible applications that might be open at the same time. *[1 mark]*

(b) Explain the role of the operating system in the management of these applications. *[4 marks]*

The intended uses of a mobile phone influence its design with regard to system resources.

(c) With reference to **two** specific resources, outline how the design of these resources for a mobile phone would differ from those of a standard PC. *[4 marks]*

(a) *Award **[1 mark]** for three or more acceptable possible applications stated.*
Internet browser
Phone application (making a call)
Camera application (taking a picture)
MP3 player
E-mail reader ***[1 mark]***

(b) *Award marks as follows up to **[4 marks max]**.*
*Award **[2 marks]** for a basic description of the operating system in the management of applications.*
*Award **[1 mark]** for elaborating on the description.*
*Award **[1 mark]** for an explanation that is clear, detailed and balanced.*
Answers may include:

Memory management: The OS allocates a certain portion of the memory for each application. The amount of memory needed by each application may change so the OS will need to be able to allocate more memory to an application as needed and to recover memory when the application no longer needs it.

Allocation of processing time for each application: This could involve time-slicing in which each application is given a certain amount of processor time before control is switched to the next application. Alternatively, the OS could use an event-driven model in which control of the processor is passed to the appropriate application as events such as an incoming call, a button press, or an interrupt occur.

Coordination of interfaces: The OS determines which application should be notified if a button is pressed and updates the display based on requests received from each of the applications. ***[4 marks]***

(c) *Award **[1 mark each]** for identifying each valid system resource up to a maximum of **[2 marks]**.*
*Award an additional **[1 mark each]** for describing how the design of each of the identified resources differ between the mobile phone and PC platforms up to a maximum of **[2 marks]**.*
Answers may include:

Memory: The small size of the mobile phone restricts the amount of memory that can be included to less than what can be included in a PC. More importantly, the mobile must use much less power than a PC and this further restricts the amount of memory that can be included and also makes some designers choose slower memory that consumes less power.

Display Screen: The small size of the mobile restricts the size of the screen to something much less than a PC. To compensate, the screen may need to have a much higher resolution. Also, the mobile screen must be readable in direct sunlight, which most PC displays are not.

Input devices: The PC usually uses a full-size keyboard. The mobile must generally use a much smaller keyboard which results either in a greatly reduced number of keys or in very tiny keys such as on a Blackberry. The PC also usually has a pointing device like a mouse which itself is already larger than many mobile phones. Instead, mobile phones may use a touch sensitive screen. ***[4 marks]***

Question: State some difference between a Spreadsheet software application and a Database Management System.

Answer:

- A Database Management System could be used for various data associations that cannot be created with spreadsheets.
- A Database Management System manages databases and serves a lot of users.
- A Database Management System preserves data integrity easier.
- A spreadsheet can be used to produce charts and graphs using automated software tools.
- A spreadsheet can hold a limited amount of information.
- A spreadsheet usually serves the needs of a single person.
- Spreadsheet software can perform a lot of mathematical functions and perform data analysis.

Mr. Alei's web page   Michael Brooke Online Course Notes   Coursera

CORE COMPUTER SCIENCE

2 - Computer Architecture

2.1.9 Binary numbers
2.1.10 Data representations
2.1.11 Logic Gates
2.1.12 Truth Tables
2.1.13 Logic diagrams

Key concepts

Binary representation		
2.1.9	Define the terms: bit, byte, binary, denary/decimal, hexadecimal.	
2.1.10	Outline the way in which data is represented in the computer.	To include strings, integers, characters and colours. This should include considering the space taken by data, for instance the relation between the hexadecimal representation of colours and the number of colours available. **TOK, INT** Does binary represent an example of a lingua franca? **S/E, INT** Comparing the number of characters needed in the Latin alphabet with those in Arabic and Asian languages to understand the need for Unicode.

Data representation

Bit - a single one or zero representing a **B**inary Dig**it**
Byte - 8 bits
Word - a group of bytes processed at the same time (***in parallel***)

- Context dependent!
- 16 bit machines used 2 byte words (they're obsolete)
- 32 bit machines used 4 byte words (they're fading fast)
- 64 bit machines use 8 bytes per word

Guess what the name for four bits is...
A nibble!

Number representations

Binary - base 2	digits 0 & 1	in Java: `0b11001101`
Octal - base 8	digits 0 - 7 (one byte)	in Java: `0315`
Decimal - base 10	digits 0 - 9	in Java: `205`
Hexadecimal - base 16	digits 0 - F (two bytes)	in Java: `0xCD`

Base 10. What does 346 mean?

100's 10's 1's
| 3 | 4 | 6 | $= 3x10^2 + 4x10^1 + 6x10^0$
10^2 10^1 10^0

Base 8. What does 346 mean?

64's 8's 1's
| 3 | 4 | 6 | $= 3x8^2 + 4x8^1 + 6x8^0$
8^2 8^1 8^0

$= 192_{10} + 32_{10} + 6_{10} = 230_{10}$

Base 2. What does 1001 1100 mean?

128's 64's 32's 16's 8's 4's 2's 1's
| 1 | 0 | 0 | 1 | | 1 | 1 | 0 | 0 | $= 128_{10} + 16_{10} + 8_{10} + 4_{10}$

$= 156_{10}$

MSB = Most Significant Bit
LSB

Hexadecimal - Base 16

- Same principles as other bases
- Use upper case A-F for "digits" 10-15
- Convenient because one digit can be represented in 4 bits

Fill in the following table

Decimal	Binary	Octal	Hexadecimal
8	1000	10	8
14	1110	16	E
166	1010 0110	246	A6
1010	11 1111 0010	1762	3F2
172	1010 1100	254	AC
34774	1000 0111 1101 0110	103726	87D6
511	1 1111 1111	777	1FF
43981	1010 1011 1100 1101	125715	ABCD

Adding in binary
The MSB

```
  10111   (23)
 + 110    (6)
 ------
  11101   (29)
```

Negative numbers - 2's complement
Complement the digits then add 1

```
0110   (6)
1001   (One's complement)
1010   (Two's complement = -6)
```

Try it again!

Notice that

```
  0110   (6)
 +1010   (-6)
 -----
  0000
```

In a signed binary number, the MSB is the ***sign bit***.
Negative numbers have an MSB of 1 and are in 2's complement form

A 16 bit ***unsigned*** integer has values from 0 to 65535
A 16 bit ***signed*** integer has values from -32768 to 32767

Multiplying binary numbers
Use the same algorithm that you do with base 10

January 28, 2018

	Wed 1/31	Fri 2/2	Wed 2/7	Fri 2/9
Topics	1.1.1 - 1.1.4	1.1.5-1.1.7	1.1.8-1.1.10	1.1.11 - 1.1.14
Presenters	Michaela & Olivia	Anna & Avery	Mlilan, Aidan, & Dominic	

1.1 Systems in organizations (10 hours)	
Planning and system installation	
1.1.1	Identify the context for which a new system is planned.
1.1.2	Describe the need for change management.
1.1.3	Outline compatibility issues resulting from situations including legacy systems or business mergers.
1.1.4	Compare the implementation of systems using a client's hardware with hosting systems remotely.
1.1.5	Evaluate alternative installation processes.
1.1.6	Discuss problems that may arise as a part of data migration.
1.1.7	Suggest various types of testing.
User focus	
1.1.8	Describe the importance of user documentation.
1.1.9	Evaluate different methods of providing user documentation.
1.1.10	Evaluate different methods of delivering user training.
System backup	
1.1.11	Identify a range of causes of data loss.
1.1.12	Outline the consequences of data loss in a specified situation.
1.1.13	Describe a range of methods that can be used to prevent data loss.
Software deployment	
1.1.14	Describe strategies for managing releases and updates.

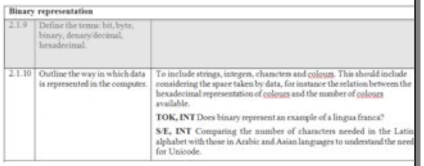
Mr. Alei's web page Michael Brooke Online Course Notes Coursera

CORE COMPUTER SCIENCE

2 - Computer Architecture

2.1.10 Data representations

Key concepts



Binary representation		
2.1.9	Define the terms: bit, byte, binary, denary/decimal, hexadecimal.	
2.1.10	Outline the way in which data is represented in the computer.	To include strings, integers, characters and colours. This should include considering the space taken by data, for instance the relation between the hexadecimal representation of colours and the number of colours available. **TOK, INT** Does binary represent an example of a lingua franca? **S/E, INT** Comparing the number of characters needed in the Latin alphabet with those in Arabic and Asian languages to understand the need for Unicode.

The basic unit of storage is the ***byte*** (8 bits). Everything in a computes is stored in units of bytes. Those bytes can represent numbers, colors, letters and more.

Each piece of data is stored in memory, so we need to know two things:
- How much memory is allotted for the data? (hopefully enough)
- How is the data to be interpreted?

This is accomplished by defining the ***variable type***, either at the time the variable is first encountered or, in some languages, automatically. Java, C++, and other languages require that the programmed explicitly ***declare the type*** of all variables when they are first defined. For example, the line

```
    int i;
```

declares that the variable `i` is of type `int`, meaning integer. In the above line, we have declared but not ***initialized*** the variable. By default, `i` will be given the value 0 (more on subtleties of this later). If we want to start the variable off with a value of, say, 10, we would use the line

```
    int i=10;
```

Java Primitive Data Types

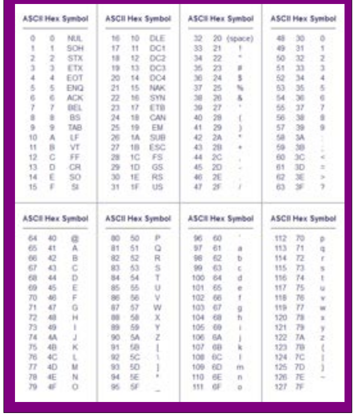
- `boolean`, stored as one bit, the type whose values are either true or false
- `char`, the character type whose values are 16-bit Unicode characters
- the arithmetic types:
  - > the integral types (whole numbers)
    - `byte` 1 byte (8 bits, 256 values)
    - `short` 2 bytes (16 bits, -32768 to 32767)
    - `int` 4 bytes (32 bits, -2,147,483,648 to 2,147,483,647)
    - `long` 8 bytes (64 bits, -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
  - > the floating-point types (numbers that may have a decimal part)
    - `float` 4 bytes (32 bits, ~±3.40282347 x 10^{38})
    - `double` 8 bytes (64 bits, ~±1.797693... x 10^{308})

Text and control characters in documents are stored as `char`.

Historically, these were stored in 8 bits with **ASCII** codes. Some languages use one byte for a `char`.

Current standard is more complete and uses **Unicode** characters in 2 bytes.

ASCII Codes

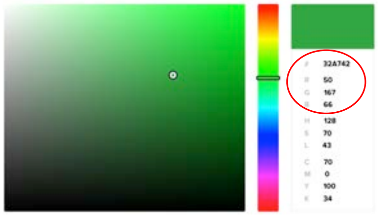


ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

What about colors?

Colors are generally stored as **RGB** values, so named because they are built from a combination of **R**ed, **G**reen, and **B**lue light. Java uses 3 bytes (24 bits) to store colors, one byte each for R, G, & B. For convenience, colors are often written using their hexadecimal codes: 0xFFFFFF is white, 0x000000 is black.



#	32A742
R	50
G	167
B	66
H	128
S	70
L	43
C	70
M	0
Y	100
K	34

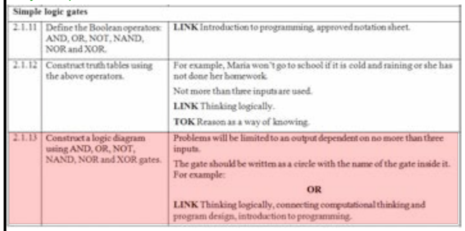
January 28, 2018

Mr. Alei's web page   Michael Brooke Online Course Notes   Coursera

2 - Computer Architecture

2.1.11 - Logical Operators
2.1.12 - Truth tables
2.1.13 - Logic diagrams

Key concepts



Simple logic gates		
2.1.11	Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR.	**LINK** Introduction to programming, approved notation sheet.
2.1.12	Construct truth tables using the above operators.	For example, Maria won't go to school if it is cold and raining or she has not done her homework. Not more than three inputs are used. **LINK** Thinking logically. **TOK** Reason as a way of knowing.
2.1.13	Construct a logic diagram using AND, OR, NOT, NAND, NOR and XOR gates.	Problems will be limited to an output dependent on no more than three inputs. The gate should be written as a circle with the name of the gate inside it. For example: **OR** **LINK** Thinking logically, connecting computational thinking and program design, introduction to programming.

Boolean variables have only two possible values - represented internally as 0 and 1.
In Java, the primitive values are called `true` and `false`.

An ***operator*** is a process that acts on one or more variables to create a result.

Boolean operators are logical combinations of boolean variables. There are 6 operators:

Let `a` and `b` be two boolean variables. Then:
`a AND b` is true if both `a` and `b` are true, otherwise false.

> In a text book, this is written *a* ∩ *b* or sometimes *a* ∧ *b.*
> Algebraically, some books use the notation *a* • *b* or *a* * *b*.
> In Java we write `a && b` or `a & b`

`a OR b` is true if *either* `a` or `b` is true, otherwise false.

> In a text book, this is written *a* ∪ *b* or sometimes *a* ∨ *b.*
> Algebraically, some books use the notation *a* + *b*
> In Java we write `a || b` or `a | b`
> Note that if both a and b are true, the value of `a OR b` is true (see XOR)

`NOT a` is the ***complement*** of `a`; that is, if `a` is true, `NOT a` is false and vice versa

> In a text book, this is written as *ā, ¬a* or *a*'. In general, a bar over a symbol means ***not*** that symbol.
> In Java we write `!a`

`a XOR b` is true if *only one of* `a` or `b` is true, otherwise false.

> In a text book, this is written *a* ⊗ *b*.
> In Java we write `a ^ b`

`a NAND b` is true if (`a` and `b`) is **false,** otherwise false.

> Logically, it's the same as NOT AND or ¬(*a* ∩ *b*) or *a* ⊼ *b*
> In Java there is no single operator for this, but we can write `!(a && b)`

`a NOR b` is true if neither `a` nor `b` are true, otherwise false.

> Logically, it's the same as NOT AND or ¬(*a* ∪ *b*) or *a* ⊽ *b*
> In Java there is no single operator for this but we can write `!(a || b)`

A note: In computer languages, there are two kinds of operators that use these ideas.
For single bit values, you can use either, but for multiple bit values, there are differences:

- ***Logical*** operators
  > Interpret 0 as false, any other number as true
  > `1001 && 0110` = `0001` and `1001 || 0110` = `0001`
- Bitwise (or ***Boolean***) operators - operate on each corresponding pair of bits in two values
  > `1001 & 0110` = `0000` and `1001 | 0110` = `1111`

Just something to have in the back of your mind.

It's easier to understand these using ***truth tables*** which illustrate the results:

AND: *a* ∧ *b*

a	*b*	*a* ∧ *b*
0	0	0
1	0	0
0	1	0
1	1	1

OR: *a* ∨ *b*

a	*b*	*a* ∨ *b*
0	0	0
1	0	1
0	1	1
1	1	1

XOR: *a* ⊗ *b*

a	*b*	*a* ⊗ *b*
0	0	0
1	0	1
0	1	1
1	1	0

NOT: ¬*a*

a	¬*a*
0	1
1	0

NAND: *a* ⊼ *b*

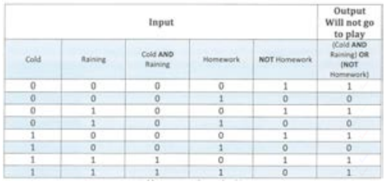
a	*b*	⊼ *b*
0	0	1
1	0	1
0	1	1
1	1	0

NOR: *a* ⊽ *b*

a	*b*	*a* ⊽ *b*
0	0	1
1	0	0
0	1	0
1	1	0

You can use truth tables to solve logic problems. Consider the statement:
"Tania won't go out to play if it's cold and raining or if she hasn't done her homework."

We can represent all the possibilities using a truth table:



Input					**Output Will not go to play**
Cold	Raining	Cold **AND** Raining	Homework	**NOT** Homework	(Cold **AND** Raining) **OR** (**NOT** Homework)
0	0	0	0	1	1
0	0	0	1	0	0
0	1	0	0	1	1
0	1	0	1	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	1	1	0	1	1
1	1	1	1	0	1

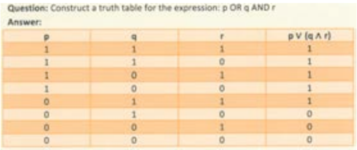
Then, we can write a boolean expression for the result. First, name some variables:

- *C* = it's cold
- *R* = it's raining
- *H* = Tania did HW
- *P* = Tania will ***not*** go outside and play

From the table we see that

P = C' • R' • H' + C' • R • H' + C • R' • H' + C • R • H' + C • R • H

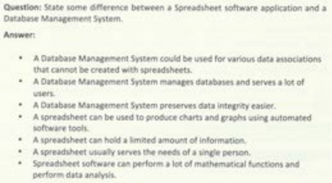
Try a couple:



Question: Construct a truth table for the expression: p OR q AND r
Answer:

p	q	r	p ∨ (q ∧ r)
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

January 28, 2018



Question: State some difference between a Spreadsheet software application and a Database Management System.

Answer:

- A Database Management System could be used for various data associations that cannot be created with spreadsheets.
- A Database Management System manages databases and serves a lot of users.
- A Database Management System preserves data integrity easier.
- A spreadsheet can be used to produce charts and graphs using automated software tools.
- A spreadsheet can hold a limited amount of information.
- A spreadsheet usually serves the needs of a single person.
- Spreadsheet software can perform a lot of mathematical functions and perform data analysis.

	Wed 1/31	Fri 2/2	Wed 2/7	Fri 2/9
Topics	1.1.1 - 1.1.4	1.1.5-1.1.7	1.1.8-1.1.10	1.1.11 - 1.1.14
Presenters	Michaela & Olivia	Anna & Avery	Mlilan, Aidan, & Dominic	

1.1 Systems in organizations (10 hours)	
Planning and system installation	
1.1.1	Identify the context for which a new system is planned.
1.1.2	Describe the need for change management.
1.1.3	Outline compatibility issues resulting from situations including legacy systems or business mergers.
1.1.4	Compare the implementation of systems using a client's hardware with hosting systems remotely.
1.1.5	Evaluate alternative installation processes.
1.1.6	Discuss problems that may arise as a part of data migration.
1.1.7	Suggest various types of testing.
User focus	
1.1.8	Describe the importance of user documentation.
1.1.9	Evaluate different methods of providing user documentation.
1.1.10	Evaluate different methods of delivering user training.
System backup	
1.1.11	Identify a range of causes of data loss.
1.1.12	Outline the consequences of data loss in a specified situation.
1.1.13	Describe a range of methods that can be used to prevent data loss.
Software deployment	
1.1.14	Describe strategies for managing releases and updates.

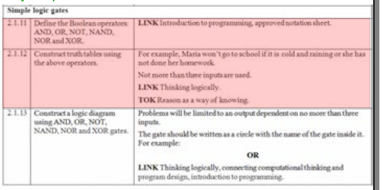
Mr. Alei's web page   Michael Brooke Online Course Notes   Coursera

CORE COMPUTER SCIENCE

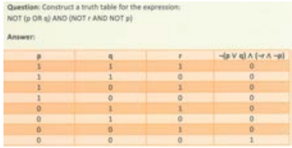
2 - Computer Architecture

2.1.11 - Logical Operators
2.1.12 - Truth tables
2.1.13 - Logic diagrams

Key concepts



Simple logic gates		
2.1.11	Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR.	**LINK** Introduction to programming, approved notation sheet.
2.1.12	Construct truth tables using the above operators.	For example, Maria won't go to school if it is cold and raining or she has not done her homework. Not more than three inputs are used. **LINK** Thinking logically. **TOK** Reason as a way of knowing.
2.1.13	Construct a logic diagram using AND, OR, NOT, NAND, NOR and XOR gates.	Problems will be limited to an output dependent on no more than three inputs. The gate should be written as a circle with the name of the gate inside it. For example: **OR** **LINK** Thinking logically, connecting computational thinking and program design, introduction to programming.

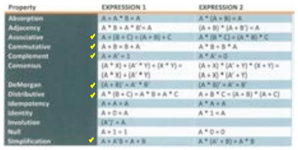


Question: Construct a truth table for the expression:
NOT (p OR q) AND (NOT r AND NOT p)

Answer:

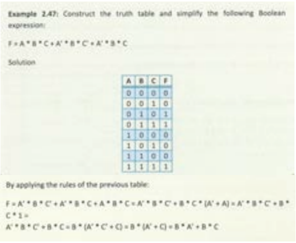
p	q	r	¬(p ∨ q) ∧ (¬r ∧ ¬p)
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

Here are some algebraic properties of Boolean algebra



Property		EXPRESSION 1	EXPRESSION 2
Absorption		A + A * B = A	A * (A + B) = A
Adjacency		A * B + A * B' = A	(A + B) * (A + B') = A
Associative	✓	A + (B + C) = (A + B) + C	A * (B * C) = (A * B) * C
Commutative	✓	A + B = B + A	A * B = B * A
Complement	✓	A + A' = 1	A * A' = 0
Consensus		(A * X) + (A' * Y) + (X * Y) = (A * X) + (A' * Y)	(A + X) * (A' + Y) * (X + Y) = (A + X) * (A' + Y)
DeMorgan	✓	(A + B)' = A' * B'	(A * B)' = A' + B'
Distributive	✓	A * (B + C) = A * B + A * C	A + B * C = (A + B) * (A + C)
Idempotency		A + A = A	A * A = A
Identity		A + 0 = A	A * 1 = A
Involution		(A')' = A	
Null		A + 1 = 1	A * 0 = 0
Simplification	✓	A + A'B = A + B	A * (A' + B) = A * B

The main ones to know are checked...they're pretty intuitive except maybe DeMorgan's and the last one. Try this:



Example 2.47: Construct the truth table and simplify the following Boolean expression:

F = A * B * C + A' * B * C' + A' * B * C

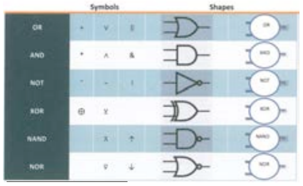
Solution

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

By applying the rules of the previous table:

F = A' * B * C' + A' * B * C + A * B * C = A' * B * C' + B * C * (A' + A) = A' * B * C' + B *
C * 1 =
A' * B * C' + B * C = B * (A' * C' + C) = B * (A' + C) = B * A' + B * C

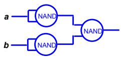
FYI, in electronics, logical operations are executed with "gates" using symbols:



	Symbols			Shapes	
OR	+	∨	|		OR
AND	*	∧	&		AND
NOT	'	¬	!		NOT
XOR	⊕	⊻			XOR
NAND		⊼	↑		NAND
NOR		⊽	↓		NOR

When you draw logic diagrams, use the symbols on the right (circles with names).

Try one: Draw a logic diagram that takes two inputs, *a* and *b*, and, using only NAND gates, creates an output that is equivalent to *a* v *b.*



It turns out you can create all the other logic functions, just using NAND gates.
This can come in very handy in electronics.

HW: Kjell