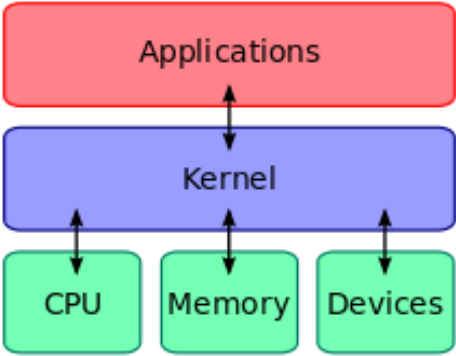
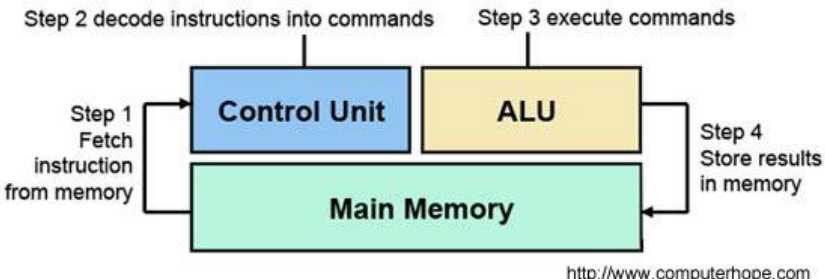


Topic 2—Computer organization (6 hours) *notes by Dave Mulkey, 2015, Germany*

2.1 Computer organization (6 hours)

	Assessment statement	Obj	Teacher's notes	Explanations
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	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.	CPU : The CPU executes programs, which are stored as numbers representing Machine Language Instructions. The Control Unit manages communication with the memory (RAM and ROM). It must FETCH instructions from the memory. After DECODING the instruction, the ALU must perform the command, performing Arithmetic (+,*,-,/) or Logic (and, or,...) operations. The result of the operation is placed in the ACCUMULATOR (a CPU register). Then the CU might send a result back to the RAM. Data Retrieval : The Memory Address Register contains the address (location) of a byte in the memory. This is transmitted

				over the Address Bus. After that the RAM finds the data and sends it back over the Data Bus, and the value is stored in the Memory Data Register. Data Storage : Same as Retrieval, but the data flows in the other direction over the Data Bus. Machine Language Emulator
2.1.2	Describe primary memory.	2	Distinguish between random access memory (RAM) and read-only memory (ROM), and their use in primary memory.	Primary memory contains: - data like numbers, images, text (ASCII codes), etc - machine code instructions for running programs - basic start-up routines ROM = Read Only Memory, permanently encoded on a dedicated chip, contains BIOS (Basic Input/Output System) that runs low level routines when the machine is starting RAM = this is where data and programs are stored This is volatile (temporary) - that means it will erase itself when the power is turned off
2.1.3	Explain the use of cache memory.	3	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.	Cache Memory is a small amount of RAM located between the CPU and the RAM. This is faster (and more expensive) than RAM. It is used as a buffer to temporarily store small amounts of data and/or program instructions. As long as needed data is found in cache, the CPU can get it's data more quickly and hence run faster. This is especially sensible if a program contains a loop, so the same instruction run over and over again.

2.1.4	Explain the machine instruction cycle.	3	This should include the role of data bus and address bus. Fetch - Decode - Execute - Store	Machine Cycle  http://www.computerhope.com Data Bus and Address Bus are explained in 2.1.1
Secondary memory				
2.1.5	Identify the need for persistent storage.	2	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.	Secondary Storage is a permanent storage device, such as hard-disk, tape, CD/DVD, cloud storage. This is a the place to permanently store software, media files and data files. This is non-volatile - not erased when the power goes off. This is needed in any device where the power is shut off and turned back on regularly. If a device is supposed to run constantly (24/7) without shutting down, it MIGHT function without Secondary Storage. But realistically, even an Air-Traffic-Control system needs a backup copy of it's software because it does occasionally need to shut down for maintenance. Secondary storage is also used as Virtual Memory - to simulate primary storage when the memory is full.
Operating systems and				

application systems				
2.1.6	Describe the main functions of an operating system.	2	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.	The main issue here is to recognize the different RESPONSIBILITIES of the OS as opposed to Applications. Operating System : - memory management To load software, the OS must find enough free RAM, then copy the program from secondary memory to primary mem - multi-tasking The computer can have multiple programs loaded and "running". They don't actually run simultaneously. The OS must manage program execution so that one program runs briefly (say 10 ms) and then a different program runs briefly and then a different one - this is called "time-slicing". Some systems allow one program to have "priority" over another. Some unimportant programs run "in the background", like a cloud-storage-synchronization process. It runs whenever nothing important is going on. - device drivers The OS provides a control-interface. Applications make a request, like to use a printer. The app sends data to the OS, then the OS controls the printer and transmits the data. This means that applications don't need to control devices directly, so the application can be simpler. This also provides standardization for controlling the devices. - user interface (GUI or CLI) The OS takes care of drawing all the windows and other GUI controls, as well as accepting keyboard input and producing visual output. Applications use OS modules to control these devices. This provides a standard "look and feel" for the specific platform.

				Applications - Applications implement the actual LOGIC for useful tasks. For example, a word-processor takes care of choosing fonts, copying and pasting, etc. But the actual keyboard input, display, file management and printer management are done by the OS.
2.1.7	Outline the use of a range of application software.	2	Application software should include word processors, spreadsheets, database management systems, email, web browsers, computer-aided design (CAD) and graphic processing software.	word-processor - writing documents spreadsheet - business calculations database - stores and organizes large quantities of data email - sending "mail" to other people web browser - retrieve and render web-pages CAD - draw technical diagrams, for engineers and architects graphics - photo editing, video editing web applications - FaceBook media applications - iTunes calendars communication - Skype, Terminal (FTP) mathematics - graphing, calculating
2.1.8	Identify common features of applications.	2	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.	WIMP = Windows, Icons, Menus, Pointer (GUI OS) GUI Components = Buttons, text-boxes, media, lists, menus CLI = Command Line Interface (DOS prompt, other text boxes)

			S/E This improves usability for a wide range of users. AIM 9 An appreciation of the improvements associated with developments in application software.	
Binary representation				
2.1.9	Define the terms: bit, byte, binary, denary/decimal, hexadecimal.	1		bit = true/false , 1/0 = Binary digit Byte = 8 bits = 0 ... 255 binary = number system Base 2, using bits denary/decimal = base 10 hexadecimal = base 16 using 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F Practice Questions

2.1.10	Outline the way in which data is represented in the computer.	2	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.	numbers = stored in binary, usually displayed in hexadecimal characters = stored in ASCII (8 bits) or UNICODE (16 bits) colours = 24 bit = True Color = Red/Green/Blue 1 Byte each $2^{24} = 16$ million colors 00110011 = 51 87 = 64 + 16 + 4 + 2 + 1 = 01010111																																			
Simple logic gates																																							
2.1.11	Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR.	1	LINK Introduction to programming, approved notation sheet.	Learn the truth tables : <table> <tr> <td>A</td><td>B</td><td>not A</td><td>A and B</td><td>A nand B (opposite of and)</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> </table> <table> <tr> <td>A</td><td>B</td><td>A or B(inclusive)</td><td>A nor B</td><td>A xor B(exclusive)</td></tr> <tr> <td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> </table>	A	B	not A	A and B	A nand B (opposite of and)	0	0	1	0	1	0	1	1	0	1	1	0	0	0	1	1	1	0	1	0	A	B	A or B(inclusive)	A nor B	A xor B(exclusive)	0	0	0	1	0
A	B	not A	A and B	A nand B (opposite of and)																																			
0	0	1	0	1																																			
0	1	1	0	1																																			
1	0	0	0	1																																			
1	1	0	1	0																																			
A	B	A or B(inclusive)	A nor B	A xor B(exclusive)																																			
0	0	0	1	0																																			

				<table><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0 (the exclusive part)</td></tr></table>	0	1	1	0	1	1	0	1	0	1	1	1	1	0	0 (the exclusive part)																																																
0	1	1	0	1																																																															
1	0	1	0	1																																																															
1	1	1	0	0 (the exclusive part)																																																															
2.1.12	Construct truth tables using the above operators.	3	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.	C = Cold R = Raining H = Homework <table><tr><td>C</td><td>R</td><td>H</td><td>C and R</td><td>not H</td><td>(C and R) or not H = not S</td><td>S</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table>	C	R	H	C and R	not H	(C and R) or not H = not S	S	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	1	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	0
C	R	H	C and R	not H	(C and R) or not H = not S	S																																																													
0	0	0	0	1	1	0																																																													
0	0	1	0	0	0	1																																																													
0	1	0	0	1	1	0																																																													
0	1	1	0	0	0	1																																																													
1	0	0	0	1	1	0																																																													
1	0	1	0	0	0	1																																																													
1	1	0	1	1	1	0																																																													
1	1	1	1	0	1	0																																																													
2.1.13	Construct a logic diagram using AND, OR, NOT, NAND, NOR and XOR gates.	3	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:  LINK Thinking logically, connecting computational thinking and program design, introduction to programming.	Not enough space here Logic Gates Intro Video Circuit Gates Emulator Practice exercises NAND Gates and More Questions																																																															