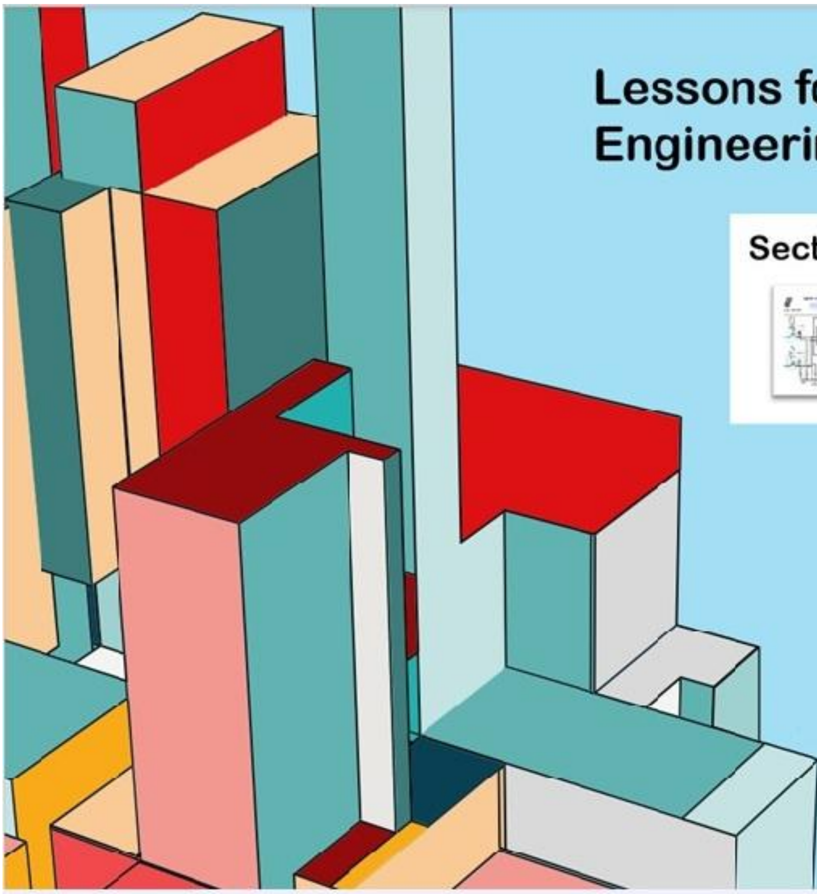
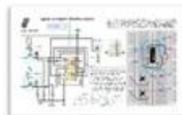


Lessons for Beginning Digital Electronic Engineering

Rev. 003 (MLK Day/2026)

Section 02 – Basics of Chip Based Components



Covers the fundamental concept of bus architecture from small integrated circuit chips to printed circuit boards and how they connect to each other to form a basic computer system. Includes more advanced work on the breadboard kit as well. Also, we cover why digital systems use 1's and 0's instead of base-ten.

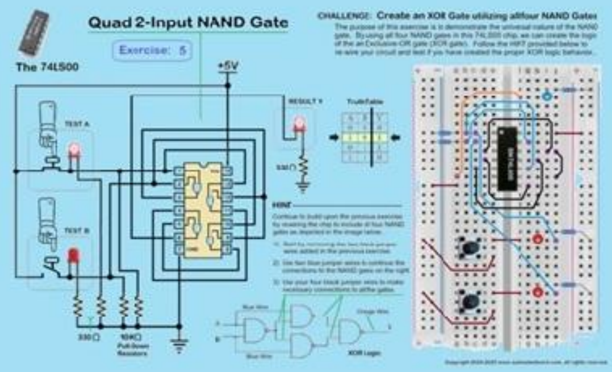
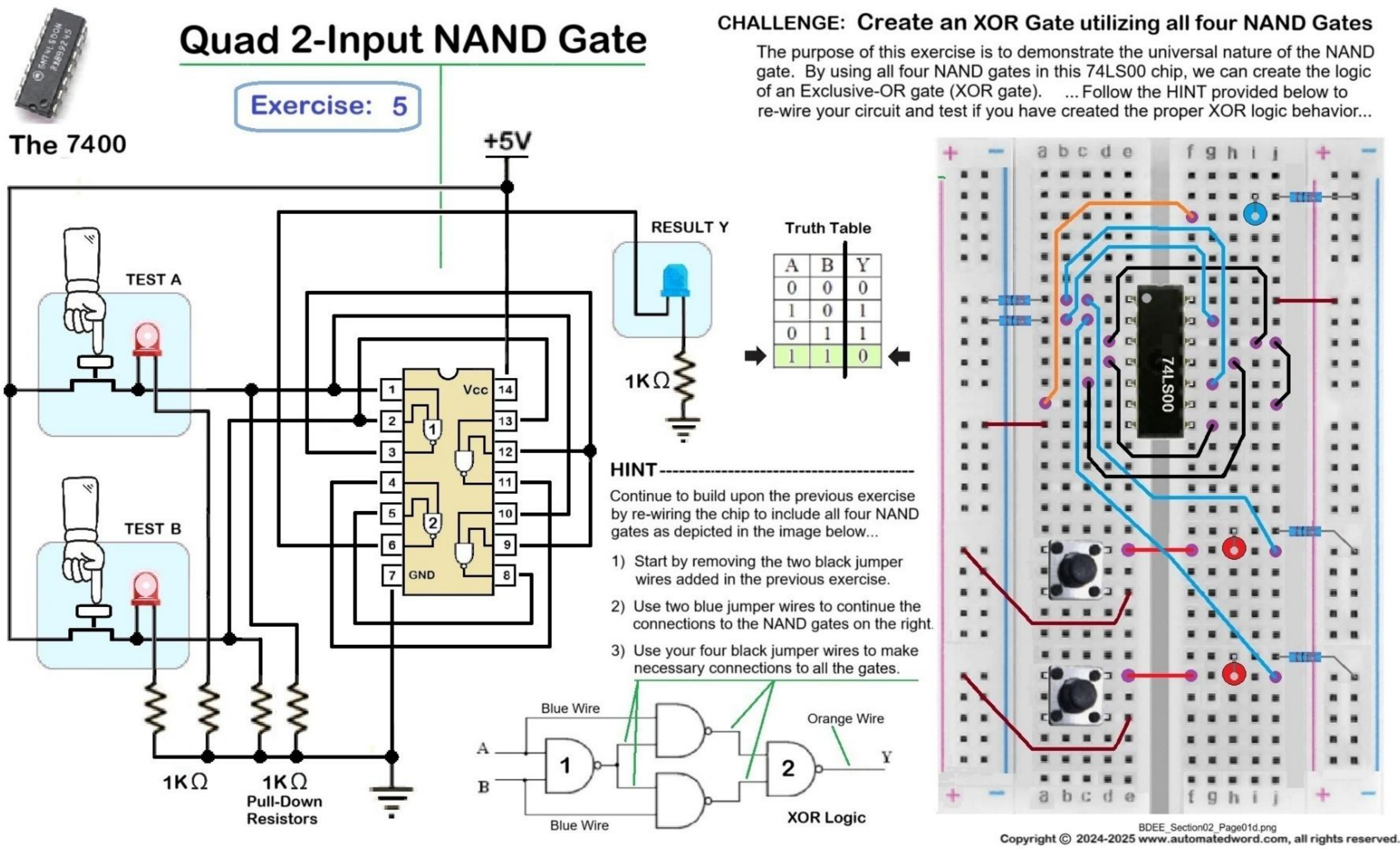


Table of Contents

"Digital parts work together in a circuit like people play their best parts in a community!"



Here is the solution for the CHALLENGE exercise 5 that I chose. I added the two blue wires by connecting one of them from pin 1 to pin 10 and the other blue wire from pin 2 to pin 13 which is shown on the breadboard illustration. I wired pin 3 to pin 12 and then another black jumper wire from pin 12 to pin 9. I then wired output pin 11 to the input of gate 2 at its input pin 4. Finally, I wired output pin 8 to the input of gate 2 at its input pin 5.

Actually, there are several different ways to solve this problem because ultimately, that is left open to interpretation as the HINT diagram doesn't specify which of the two NAND gates on the right side of the chip are to be wired from pins 1 and 2. Likewise, the output pin 3 could be connected to input pins 9 and 12 in several different ways. Also, another possibility as to connect pins 1 and 2 to pins 9 and 12 and connect the output pin 3 to pins 10 and 13. The important thing to understand is that there are several different ways to wire a solution to fulfil the requirements of the HINT diagram.

"What is the XOR gate used for?" - Bing Search "What is the XOR gate used for?" - Google Search

Finally, this note about the NAND gate being the ultimate "Universal Logic Gate"…
While it is true that you can also build every logic circuit utilizing only NOR gates there are several different instances where it takes more NOR gates to accomplish this in some logic structures than could be accomplished with NAND gates but, it has never been shown that the reverse is true – that it would take more NAND gates to accomplish a particular structure than NOR gates.

I plugged the phrase: "why is the NAND gate considered to be the universal logic gate" into a Bing search and got the following… Understanding The NAND Gate - Dyethin

Why is the AND gate a universal gate? - Stack Exchange Universal Logic Gates - GeeksforGeeks

Digital Electronics Uses The Binary Number System

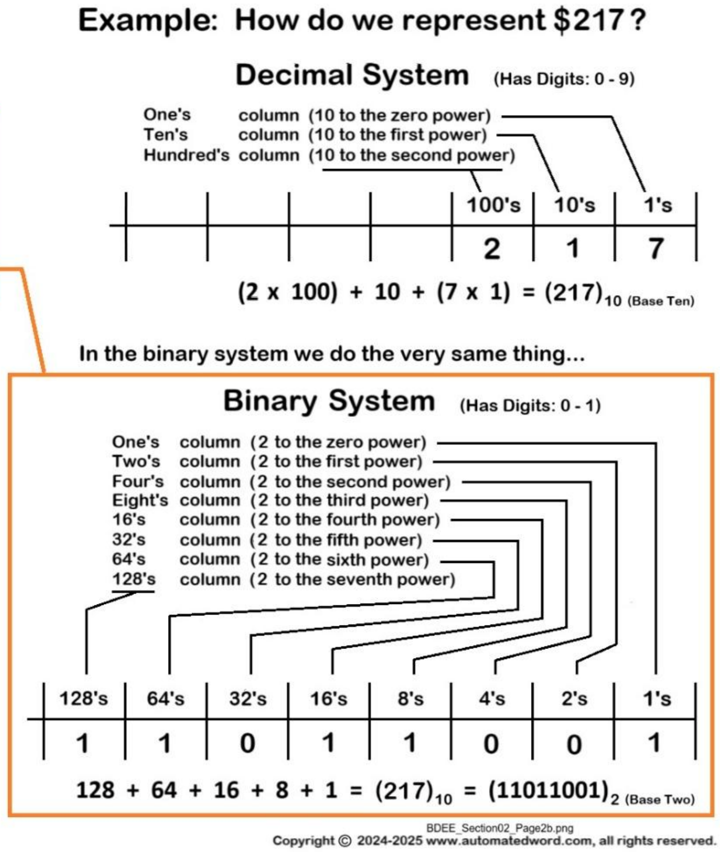
In our daily lives, we tend to think of numbers or quantities of things in, what we call, Base-10 (the good old Dewey Decimal System). Perhaps this comes from the fact that we typically have 10 fingers (who knows).

Unfortunately, in digital electronics, including computers and the like, there are only two fingers to work with so to speak. We call them 5-Volts and 0-Volts. We tend to use the shorthand names for these two fingers - "1" and "0" ("One" and "Zero"). Actually, some digital systems run at other voltage levels but, whether they run at 15, 12, 5 or 3.3 volts, in any given stable state of the circuitry there exists either the full system voltage or zero voltage and there is nothing in between.

It should be understood that whether using the decimal system or the binary system, every possible quantity or number can be represented. There isn't any quantity or number that can be represented in one system that cannot be represented in the other. See the example →

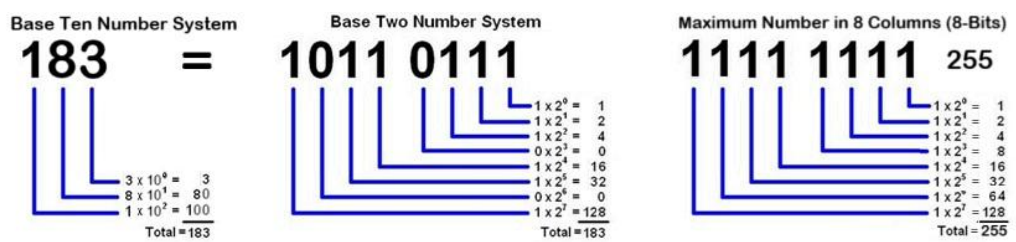
There Are 8 Bits In A Byte...

Without going into too much history, somewhere around the 1940's - 1950's, the term "Byte" was invented to describe a sort of unit of data when dealing with concepts of digital electronics. Since that day we have used the term "byte" to represent eight bits (or columns, if you will) of binary data. Notice in the Binary System to the right, there are 8 columns of 1's and 0's. Eight bits of on/off data that could be represented by digital circuitry. The bits are termed in different ways under certain contexts. They are typically said to be "high" or "low" / "on" or "off" in the electrical context or, "true" / "false" / "0" or "1" in the logical, abstract and arithmetic contexts. The fact that a byte of data is a group of 8 bits is not entirely arbitrary as it was seen to able to hold a value between 0 and 255 - a total of 256 possible numbers. More importantly, it was determined back then to be enough numbers to assign each character on the typewriter a unique value as well as a set of numbers for controlling output to printers.



Here is another way to visualize Base 2 (Binary) numbers...

Remember that in digital electronics, we refer to Columns as "Bits".

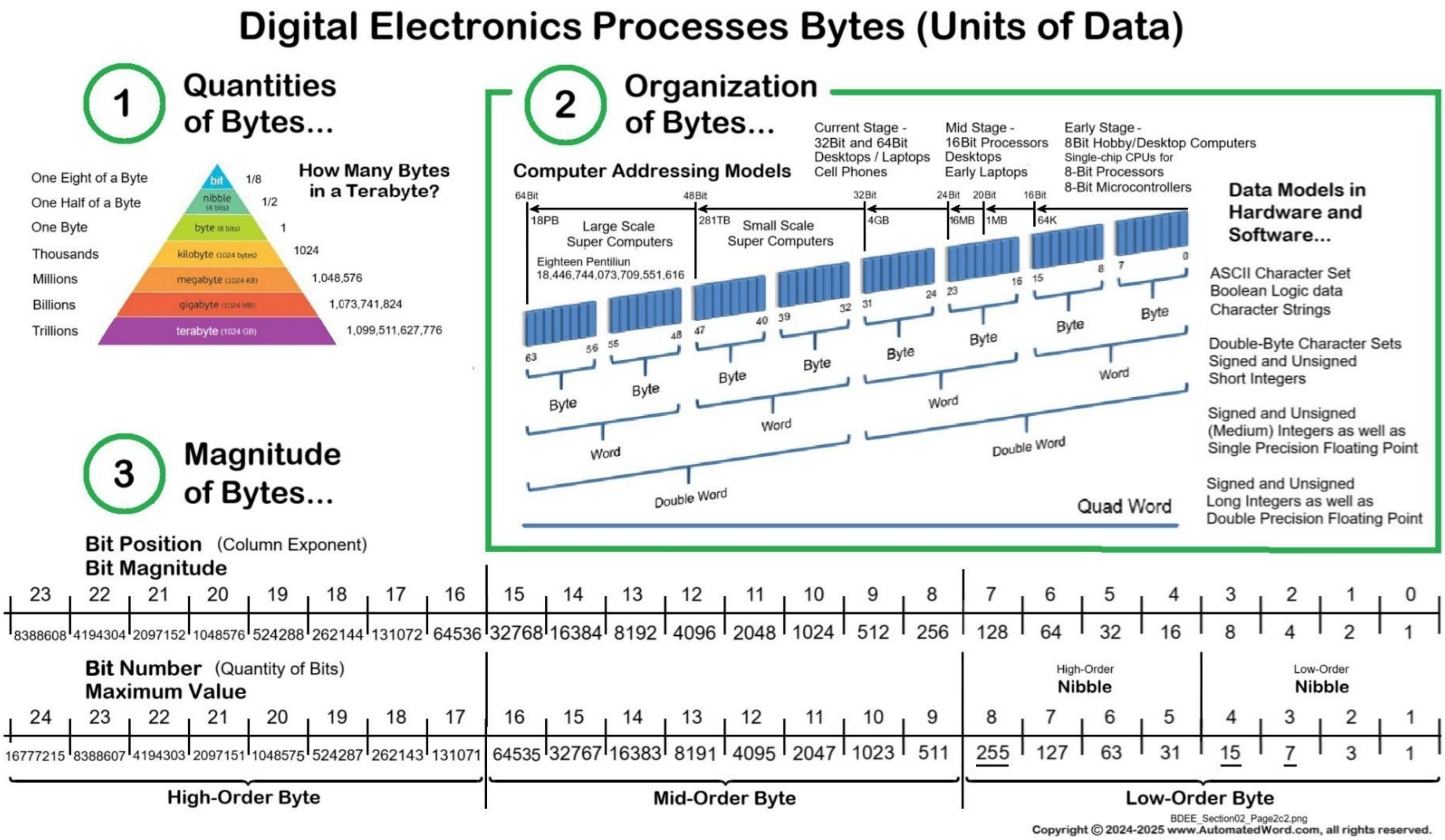


Before you get too panicked about binary numbers, you should know that we will not be making large calculations with them. Even though the computer and other digital systems can deal with very large numbers with as much as 64 columns, we will mostly be concerned with only 4 columns (or bits as we call them). Four bits/columns is enough to represent 16 numbers (from 0 – 15) and that is as much as we will be dealing with for the most part for the rest of this course.

There is only one part where we will be discussing the calculation of 8-bit numbers and that is in Section 3 where we talk about how the CPU of a computer processes numbers - for example, how numbers are added or subtracted together. Even in that section, It is important to remember that an 8-bit number can only represent a number from 0 to 255 for a total of 256 possible values.

So don't worry, It is not important to become proficient in binary operations at all. It is only important to understand how the computer deals with them from a digital electronics standpoint. We will let the electronic circuits do the math work for us.

Here is what I found on the web… [Math is Fun - Number Bases] [Computer Number Systems] [Understand Base in Math - Different Numeral Systems] [List of numeral systems - Wikipedia]



Section 1 – Quantities of Bytes…

The pyramid shows the terms used to identify magnitudes of quantity for bytes. The quantity at each level does not exactly match up with the decimal system because each column in the binary system is a doubling of the previous column from right to left. For example: In the world of digital electronics where the binary number system is utilized, the jump from 512 to 1024 is the closest we will come to 1,000 (in the decimal number system) so we arbitrarily call that 1K. Likewise, the closest we come to 1,000,000 is 1,048,576 and so, we call that 1M, and so on.

Section 2 – Organization of Bytes…

When bytes represent numerical data, they are grouped together as shown in the graph and listed on the right-hand side. Keep in mind that: there are 4 bits in a nibble, 8 bits in a byte, 2 bytes in a word, 2 words in a double-word and 2 double-words in a quad-word (for a total of 8 bytes). When character data is stored as bytes it is usually a string of indefinite length of bytes in ASCII codes or words of double-byte characters. When RAM addresses are stored, they are as a group of bytes which is listed along the top of the chart for types of digital electronic devices/computers such as desktops, laptops, cell phones and microcontrollers.

Section 3 – Magnitudes of Bytes…

This section displays, what I call, two “Bit Rails”. Each rail contains 24 columns in the binary number system for handy lookup convenience. The upper rail is the Bit Position/Bit Magnitude rail. The bit positions within the columns of the binary system are the most commonly used notations on schematic diagrams as well as in digital electronics documentation. The lower rail is the Bit Number/Maximum Value rail. On that rail, the Bit Number represents the number of columns from right to left and the Maximum Value is the maximum amount that can be held by that number of columns (or, that number of bits). Because we will only be dealing with 3 or 4 columns in our study of digital electronics, it is important to remember that when dealing with 3 columns, that number of columns can only hold eight values (0 – 7) and four columns can only hold sixteen values (0 – 15). You don’t have to remember all the bit-patterns for eight columns, it is important to keep in mind that the maximum value that can be represented by eight bits is 255 for a total of 256 possible values.

  (Back to TOC)

Digital Electronics Stores and Processes 8-Bit Bytes

In digital electronics and specifically, as it concerns the design and operation of computers, we think of the computer as a processor of digital data. As stated before, several decades ago, we fell on the "Byte" as a convenient unit of data and, from that time on, we have designed computers to process data in terms of bytes. We will examine in more depth what the definition of a computer is but, for now, let us understand that one of the most important things that makes a computer a powerful data processor is its ability to store and manipulate large amounts of data and this directly translates to large numbers of bytes (or units of data).

The computer's ability to store and retrieve bytes of data from its memory is so important a concept that we will be focused on that for the remainder of this section of the course. Later sections of this course will be more focused on how the computer manipulates data in arithmetic, logical and/or other processes that makes it a useful device in our real world. So, let us begin with memory...

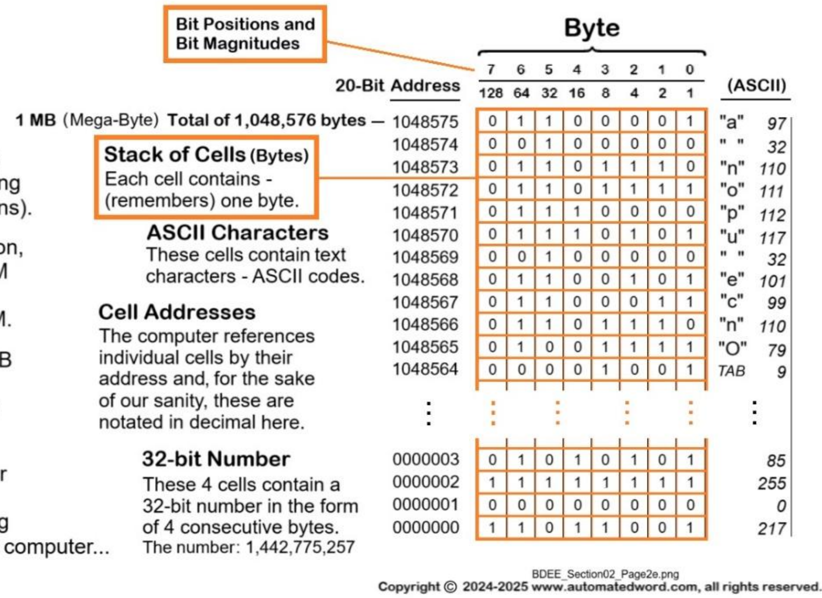
RAM Stands For Random Access Memory...

The reason RAM is important is simple. Once the computer is turned on, much of its activity is retrieving its instructions from memory in order to know how to perform the steps of its operations. The operations typically involve retrieving bytes of data from RAM, performing various manipulations like adding numbers together, comparing them and finally, storing the resulting values as byte-sized data back into RAM – (memory read and write operations).

For the purpose of demonstrating concepts of a computer's memory operation, we will construct a typical model of what a particular computer system's RAM might look like and, we will use this model throughout the remainder of this section of the course. The figure on the right is a good way to visualize RAM.

Imagine, if you will, that the RAM in our system has the capacity to store 1MB (1 mega-byte) of data. One mega-byte, in digital electronic terms is actually over 1 million bytes - 1,048,576 bytes to be exact. Just think of this memory as a stack of cells where each cell can contain (or remember) an 8-bit byte.

The figure to the right should explain in a little more detail the structure of our system's RAM but, before we delve into the exact details and exercises for demonstrating how the computer reads and writes data to RAM, the following pages will cover bus-architecture. We will see how RAM is connected to the computer...



The American Standard Code for Information Interchange (ASCII) codes were developed first as a table of 128 numbers to represent printable characters as well as non-printing control codes for electric printing devices. It was then expanded to the full 8-bit byte with another set of 128 special symbols used in print for a total of 256 numbers which is what the byte can represent with 8 base 2 columns (or, 8 bits). Here are a couple of websites that explain ASCII codes… ASCII Table (7-bit) - ASCII Code, ASCII - Wikipedia
Reference ASCII Table - Character codes in decimal, hexadecimal, octal and binary,

Here is the audio/visual library of RAM topics – after that, it's the good ole reading library…

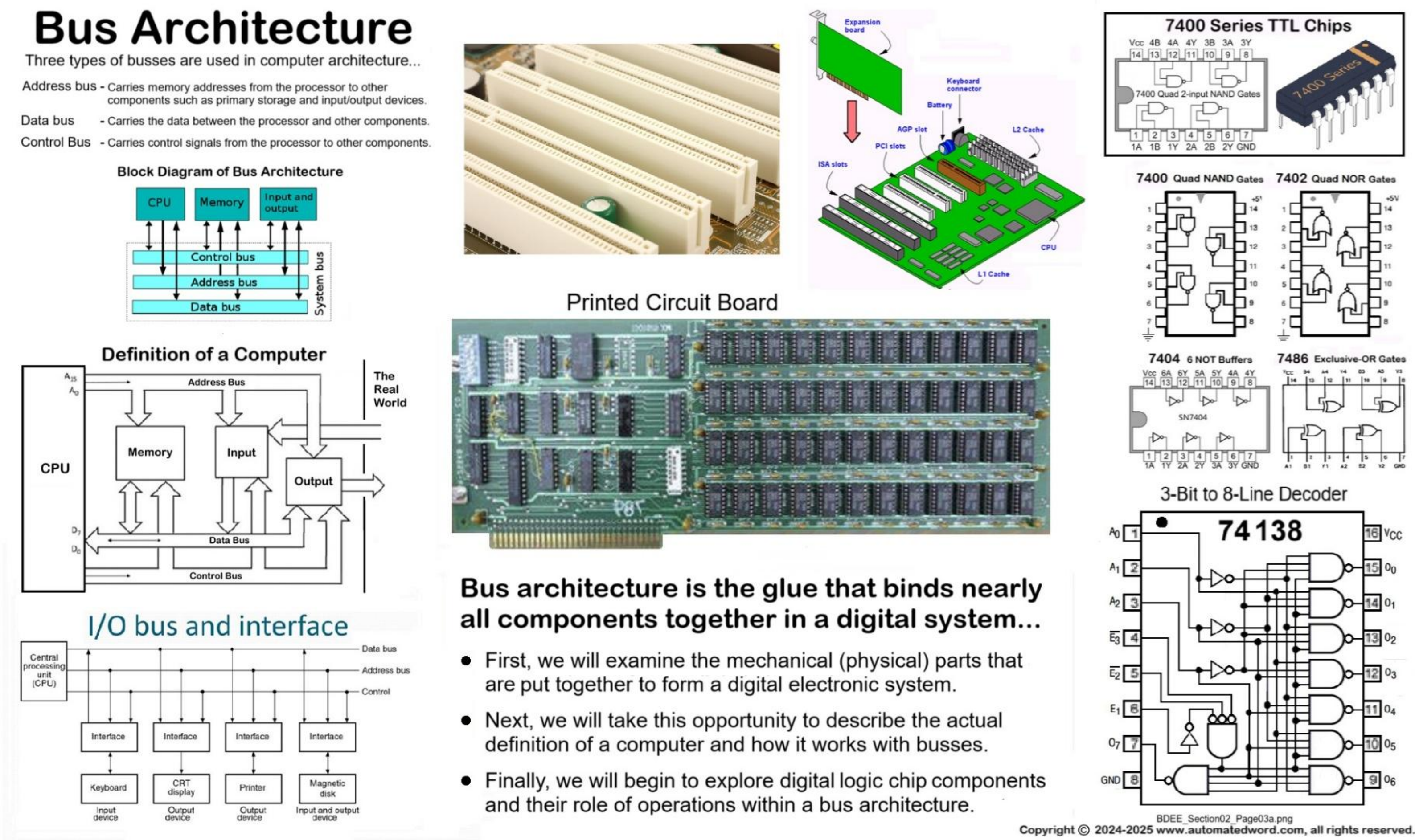
YouTube - Understanding RAM for beginners YouTube - RAM Explained - Random Access Memory
YouTube - RAM Explained for Beginners - Random Access Memory YouTube - Everything about RAM & ROM
YouTube - Numerical data explained YouTube - Fundamental Character Data in RAM
YouTube - How C++ Stores Character Data YouTube - How much RAM does your phone REALLY need

When you first click on the list of websites that I provide you should be aware that the articles or other information they provide may seem redundant at first – when this happens you should skip ahead through them because they usually end up talking about something unique or they provide alternative views or do so in a style that is more likable than others. And, believe me, I am not trying to flood you with advertising. I find the ads overwhelming and annoying sometimes but, occasionally you can see some great information peeking through the clutter.

Here are some more websites that talk about RAM and the different types of data that the computer stores in it…

What Is Stored in RAM? | Robots.net Random Access Memory (RAM) - GeeksforGeeks
Unlocking The Secrets of RAM - Vtechinsider What's Stored in RAM: Key Data Your Computer Needs
Do you know how numeric values are stored in RAM memory? What is RAM and How does it Work?
Is Data Really Stored in RAM? - How to Read RAM? Is Any Data Stored on RAM? - SoftHandTech

Here are two Datasheets for two different Ram chips in the BDEE Quick Viewer: CY7C185 IS61C512



The physical components of Bus Architecture…

The term "Bus" in digital electronics comes from the analogy of a commuter bus going down the street. Providing only one or a handful of wires or leads to transport electronic signals around the circuitry of a digital system is like transporting just a few people around in an automobile. Whereas a commuter bus can transport many more people than an automobile, an electronic "bus" can transport many more electronic signals all in the same timeframe. This translates to moving more information (signals and data) around the system in much swifter time.

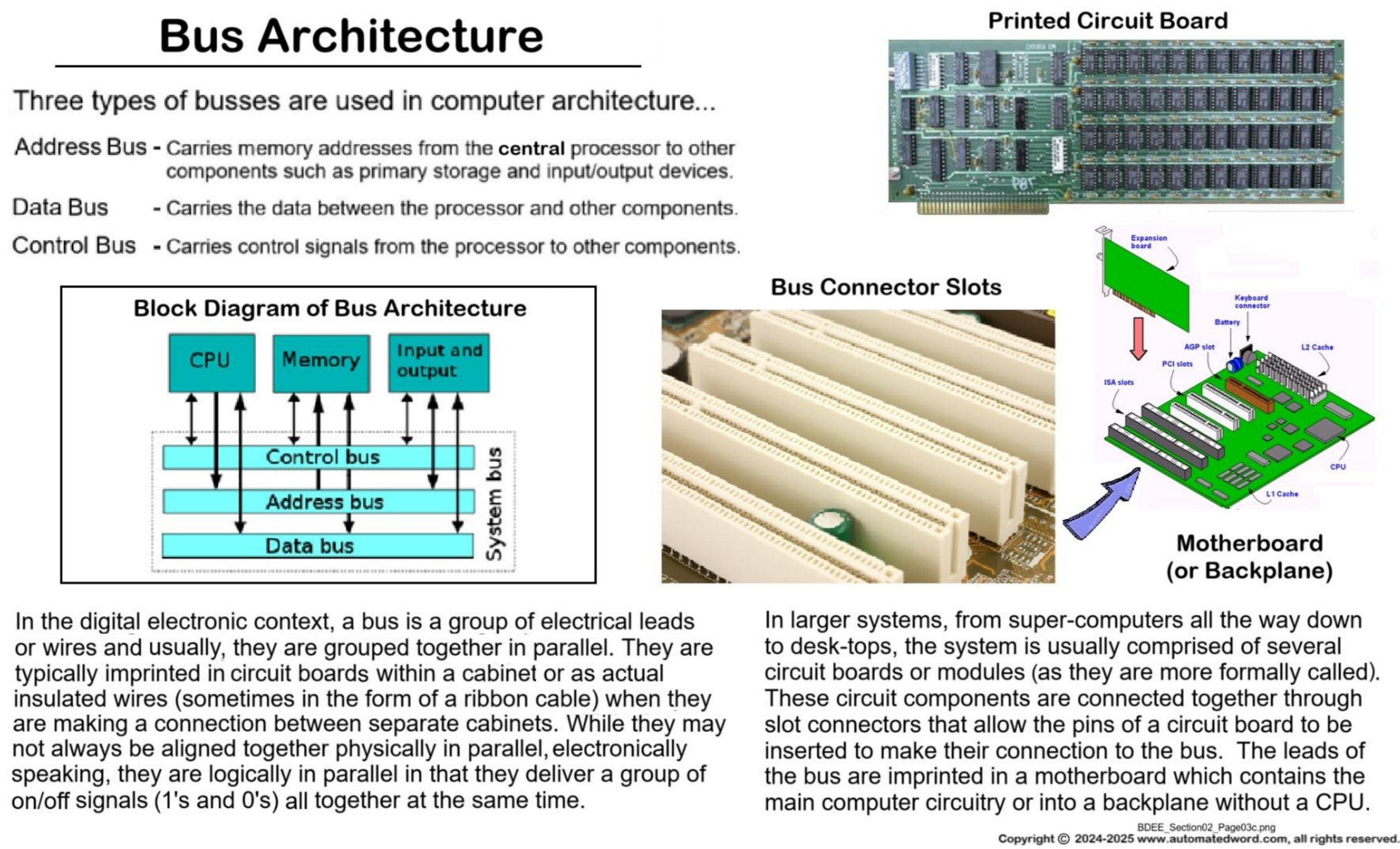
The computer does have an actual formal definition…

In the body of work known as computer science there does exists an actual formal "Definition of a Computer". However, we will discuss this definition in less formal terms for the time being. The formal definition is steeped in very technical language and can be somewhat confusing for our purposes during this course. Ultimately, it is important to understand this definition as it will allow you to put into context the devices that you are likely to encounter during your lifetime even if you never make a career in computer science or digital engineering. We will cover the essential points of this definition which will enable you to discern computer facts from fiction for the rest of your life.

The digital system's logic gates are connected by busses…

Now, our journey into the realm of digital electronics begins in earnest. We will examine how digital logic is packaged into what we call chips and how digital signals and data move through the circuits to make for a cohesive computer system. Since water pipes have been left far behind, you should notice too, that even transistors are no longer thought of as we transition to the larger picture of block-diagrams. Since memory is an important part of a computer's power and it is relatively simple circuitry, that is what we are going to focus on for the remainder of this section of the course.

The next three pages will examine more detail in each of these three subjects…



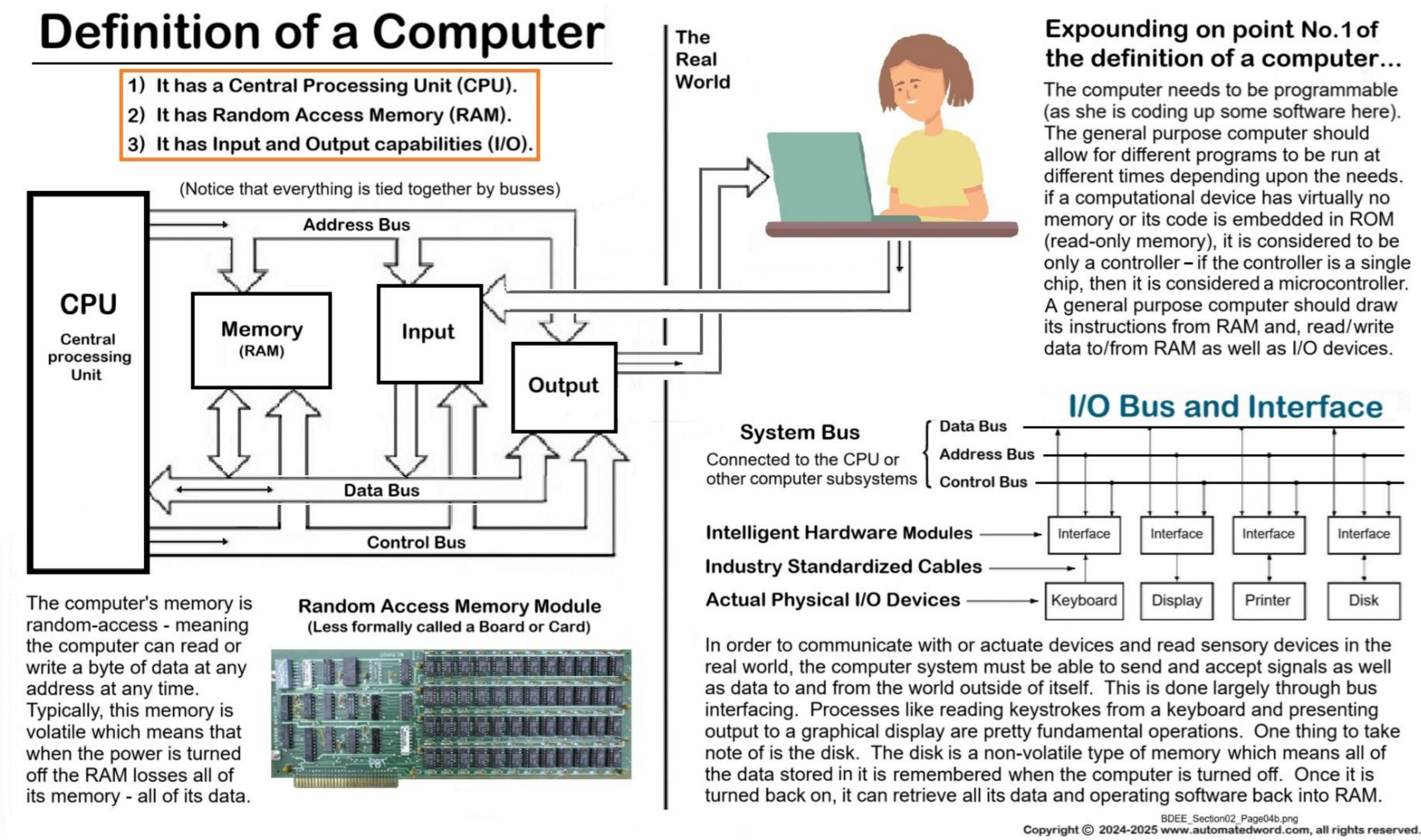
The main idea behind Bus Architecture is the fact that it delivers information around digital circuitry in a parallel fashion as opposed to delivering information one bit after another in serial fashion. This greatly increases the speed of computational devices. For example, if it takes 5 nanoseconds to move a single bit (1 or 0) through a single logic gate in a circuit, it would take (32 x 5 =) 160 nanoseconds to send 32 bits. Or, you can send all 32 bits of data all at the same time in parallel (in 5 nanoseconds) – much faster than in serial mode, No? This is one of the factors that vastly increase the processing power of a computer – processing in parallel. We will cover other factors that add greatly to efficiency in later chapters on our way to describing other components of the system.

Once again here is the total emersion process of learning. Sometimes there is nothing like being totally overwhelmed by a page of links to endless information. You start in one place (even if it is picked at random) and you end up learning something completely different from where you started. I discovered this especially while forming a list of links for the topic of bus architecture. When you see something explained in so many different ways. It seems that the questions that are raised by one source of information fall into place by being answered from another source. And all of the peripheral information starts to put a bigger picture of things into focus. So, here is a nice variety to swim in… Basic Computer Architecture (Beginner's Crash Course)

What is a Computer Bus? - GeeksforGeeks Types of Computer Buses - HubPages
Computer Architecture for beginners: Part 1 | by Chamuditha Kekulawala | Medium CA for beginners: Part 2
Bus (computing) - Wikipedia What is a computer bus: interface, types, architecture | CCM
What is Bus architecture? | Amazing Algorithms LCSO | Functions Of Data Bus , Address Bus , Control Bus

YouTube - Bus Architectures - Professor Messer Course YouTube - Introduction to Bus Architecture
YouTube - Computer Buses – ITFreeTraining YouTube - Computer Architecture - System Bus | TeachingCS
System Bus in Computers - Video | Study.com YouTube - Types of Computer Buses

What's the importance of Buses on Computers | Lenovo US PCB Bus Bar in Modern Electronics - IBE
PCB Bus Routing and Layout | Altium Designer PCB Design: Prioritizing Busses Busses | Acedemic PDF



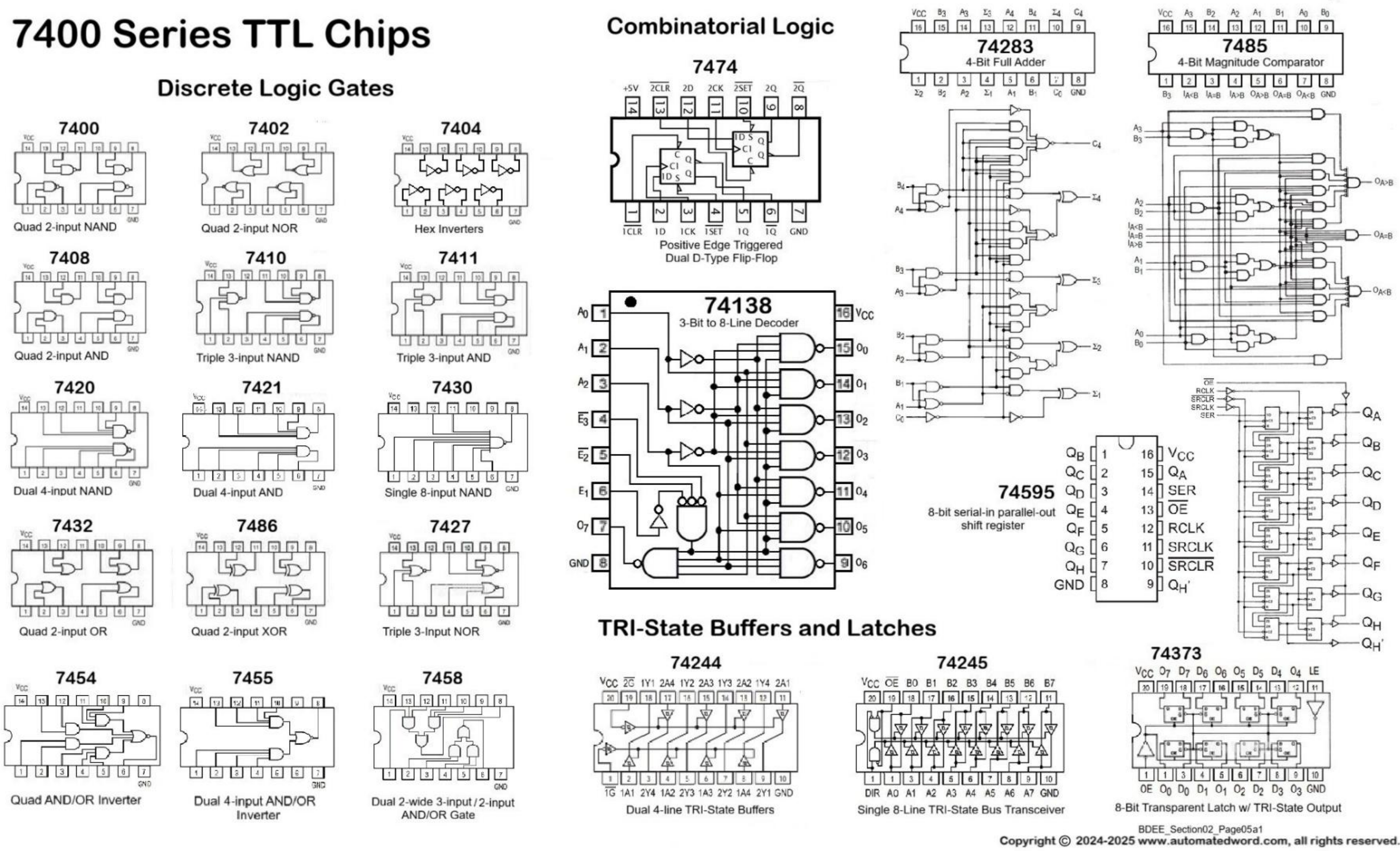
The main idea behind the Definition of a Computer is to be knowledgeable about exactly what is meant when we say "it is a computer" - as an electronic device, what distinguishes it from other electronic devices. WELL, it boils down to this: a computer is a general purpose device. While there are many devices with computational capabilities, and sometimes we call them "computers" as they do perform computations such as hand-held calculators, devices which take sensory inputs and issue variable outputs as in controlling the speed of an electric vehicle. There are "computers", so to speak, that run a typical Coke dispensing machine at your favorite fast-food restaurant, however, these devices are usually referred to as micro-controllers because they are not user programmable. That is the main point – a "computer" is user-programmable. Ultimately, this means the computer needs to be able to run different programs at different times according to the will of the user. It should be able to run any programmed application at any time according to the needs of the user.

According to the definition, a computer needs to have a "scratchpad" so to speak, a component that allows for the temporary recording and retrieval of information (data) for processing. In formal terms, this "scratchpad" is called "Random Access Memory" or "RAM" for short. The RAM component needs to be fast enough to keep up with the computer's processes even if the RAM is volatile (unable to keep stored data when power is turned off).

Also, a computer needs to be able to accept information (input data) from the real world, process it, and issue information back to the real world in the form of output data. For the simplest example: accepting information from a keyboard, processing it, and issuing resulting information to a display terminal of some type. Or, for a more complex example, to accept retail orders from the internet, complete financial transactions and issuing shipment instructions to a warehouse – all possibly through internet or other communications connections.

Finally, the computer needs a central processing unit (CPU) sophisticated enough to orchestrate all the activities of the many components needed for automated processes to occur. This happens most efficiently because the system is tied together through its Bus Architecture Whether directly signaling the timing of data movement from one component to another or, delegating this authority to an I/O component like a magnetic disk for it to temporarily take control of address and data busses, the CPU issues all of its commands on the control bus (which it owns).

  (Back to TOC)



Starting in about the mid-1960s the 7400 family of chips had grown to several hundred by the 1980s and is roughly categorized into two groups. The first group of chips were fairly simple and is comprised of mostly "discrete logic" gates. As the technology of chip manufacturing matured, chips were able to hold many more transistors and more sophisticated logic structures packed into them – these would be called the group of "combinatorial logic" chips.

Here are some links to descriptions, specification sheets and other information on how some chips are used…

Discrete Logic Gates

7400	7402	7404
7408	7410	7411
7420	7421	7430
7432	7486	7451
7454	7455	7458

Combinatorial Logic Structures

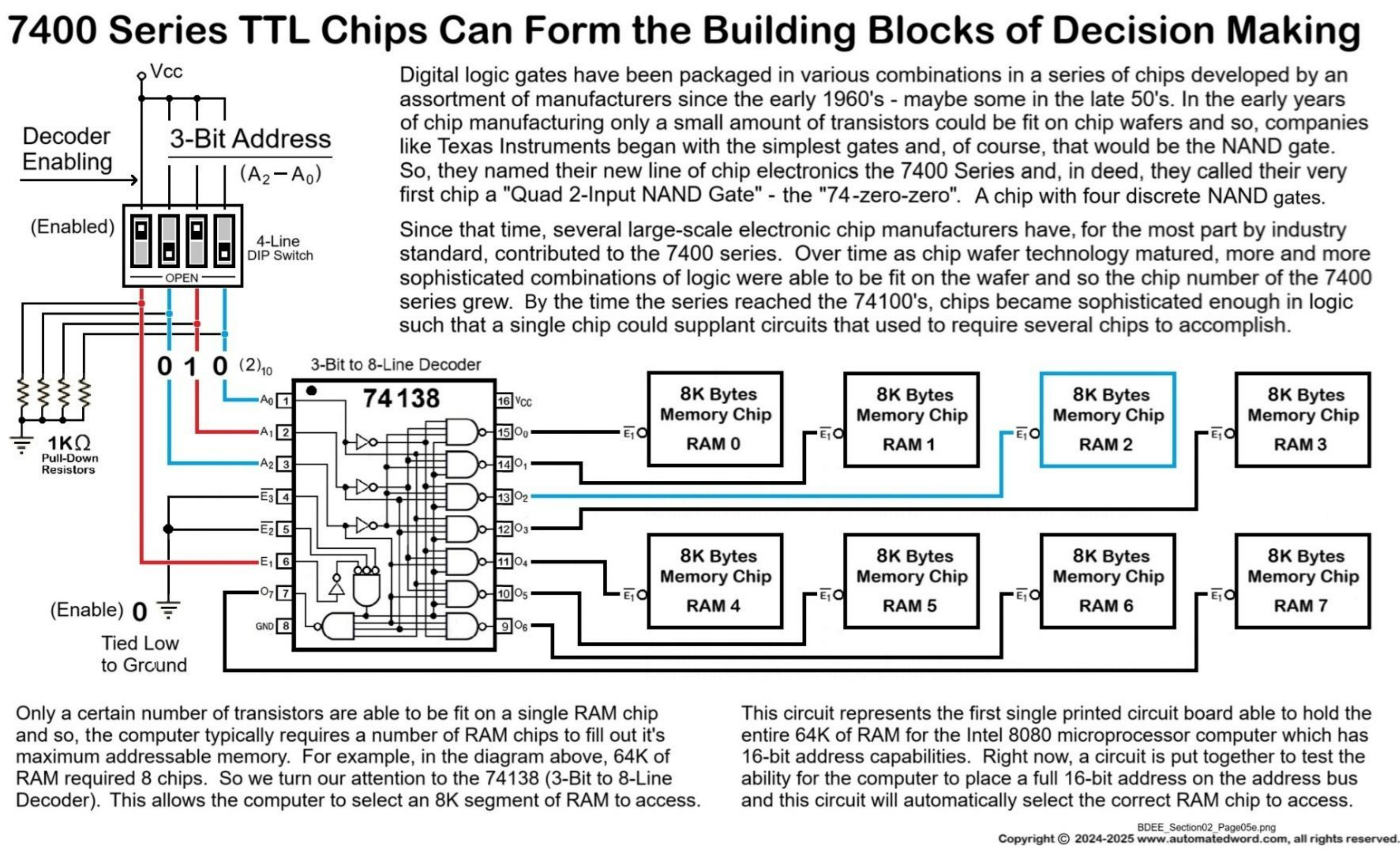
7474	Dual D-Type Flip-Flop (Falling Edge Triggered)
74283	4-Bit Full Adder (with Lookahead Fast Carry)
7485	4-Bit Magnitude Comparator
74595	8-Bit Serial-in Parallel-out Shift Register
74138	3-Bit to 8-Line Decoder
74139	Dual 2-Bit to 4-Line Decoders
74244	Dual 4-Line TRI-State Buffers
74245	8-Line TRI-State Bus Transceiver
74373	8-Bit Transparent Latch (with TRI-State output)

List of 7400-series - Wikipedia   7400-family ICs - Wikipedia   Transistor–Transistor Logic - Wikipedia

YouTube - Digital electronics basics: 7400 logic chips   7400 Series ICs - Pinouts, Examples, and More
Understanding Digital Logic ICs — Part 1 | Nuts & Volts Magazine   Logic ICs Part 2 | Nuts & Volts Magazine

An All-inclusive Guide: 74LS74 Dual Flip-flop – Flex PCB   74 Series Logic ICs | Electronics Club
74LS138 Application Circuits - Kynix   YouTube - Tutorial | Building the 74LS283 4-BIT FULL ADDER
YouTube - Introduction To The 7400 Series Logic Chips   YouTube - Quiz Buzzer Circuit: 74LS373

YouTube - You can learn Arduino in 15 minutes   How a 74HC595 Shift Register Interfaces with Arduino
YouTube - The Shift Register (74HC595N)   Serial to Parallel Shifting with a 74HC595 | Arduino Docs



At this point, without worrying about how the 74138 is to be used in a logic circuit, it is more important to completely understand what it is doing – how it operates so later, you will more easily understand how it is to be used in a given circuit context…

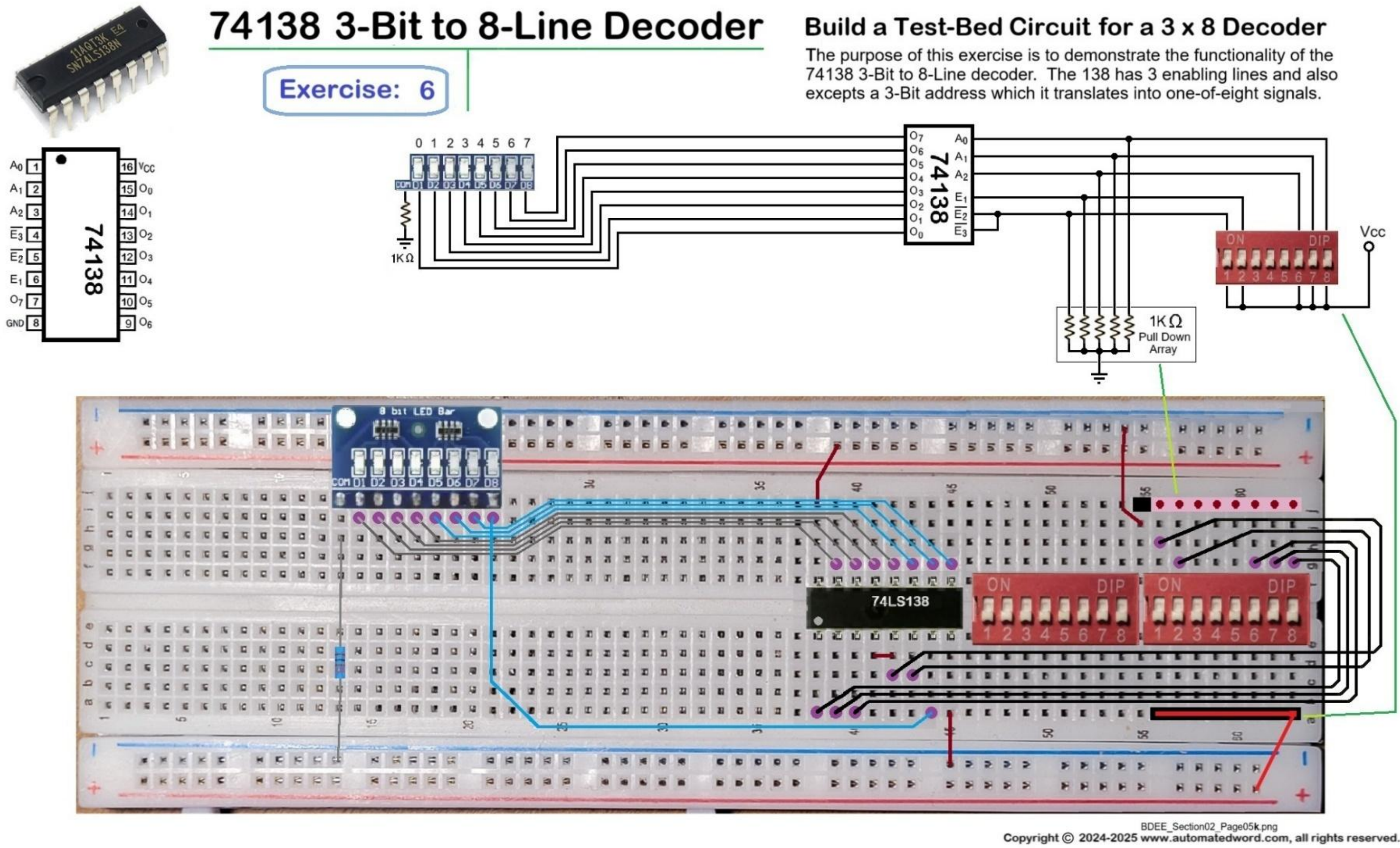
First, it is called a "combinatorial logic" chip. This simply means that it is not a chip with two or more discrete gates but a single block of functional logic structure with a single purpose. This structure is referred to as a "decoder" because it decodes a binary data input (A0, A1 and A2) as a 3-bit address and decides from that number, which of 8 output signals (O0 – O7) to activate. A 3-bit binary number can only represent eight possible numbers (0 – 7) and so, one and only one output signal will be activated at a time.

Before it will perform this logic process of selecting one of eight signals based on it's 3-bit input, it must be enabled. Like many other combinatorial logic chips, the 74138 has what are called "enabling inputs". There are three enabling inputs to this chip marked as E1, E2 and E3 and they must be satisfied before the chip will perform it's intended selection process. First, the output lines (O0 – O7) are said to be "active low" signals and this means they are normally high (5 volts – or 1) until they are selected to be activated and when selected they will go to 0 (0 volts). Until the enabling inputs are satisfied all eight output signals will remain at 1 (high).

Notice in the illustration above, that E2 and E3 have a bar above them. This means that they are expecting a 0 (low) input to be satisfied. In order for the chip to be fully enabled, E1 must be satisfied and, since it does not have a bar, it is expecting a 1 (high) input. When all three enable inputs are satisfied the logical decoding will take place and one of the eight output signals will be selected according to the number (0 – 7) provided by the A inputs.

Here is some info from the web… See the Datasheet in the BDEE Quick Datasheet Viewer: [74LS138]

[74LS138 IC: Pin Configuration, Features, Circuit Diagram and Applications]
[7400 Series Guide: 74HC138/74LS138] [74LS138 pinout, features, example, datasheet and applications]
[74LS138 Decoder Working Principle, Application Scenarios and 7AHC138 vs 74LS138]



When you have constructed this exercise make sure you have turned all eight switches to the off position on the 8-line DIP switch. Once you power the board you should note that all the blue lights are on in the 8-bit LED panel. This, of course, is correct because like most decoder type chips, the eight output pins O0 – O7 of the 74138 are issuing a high output (5-volts) and, this will be true as long as the chip is in the disabled state.

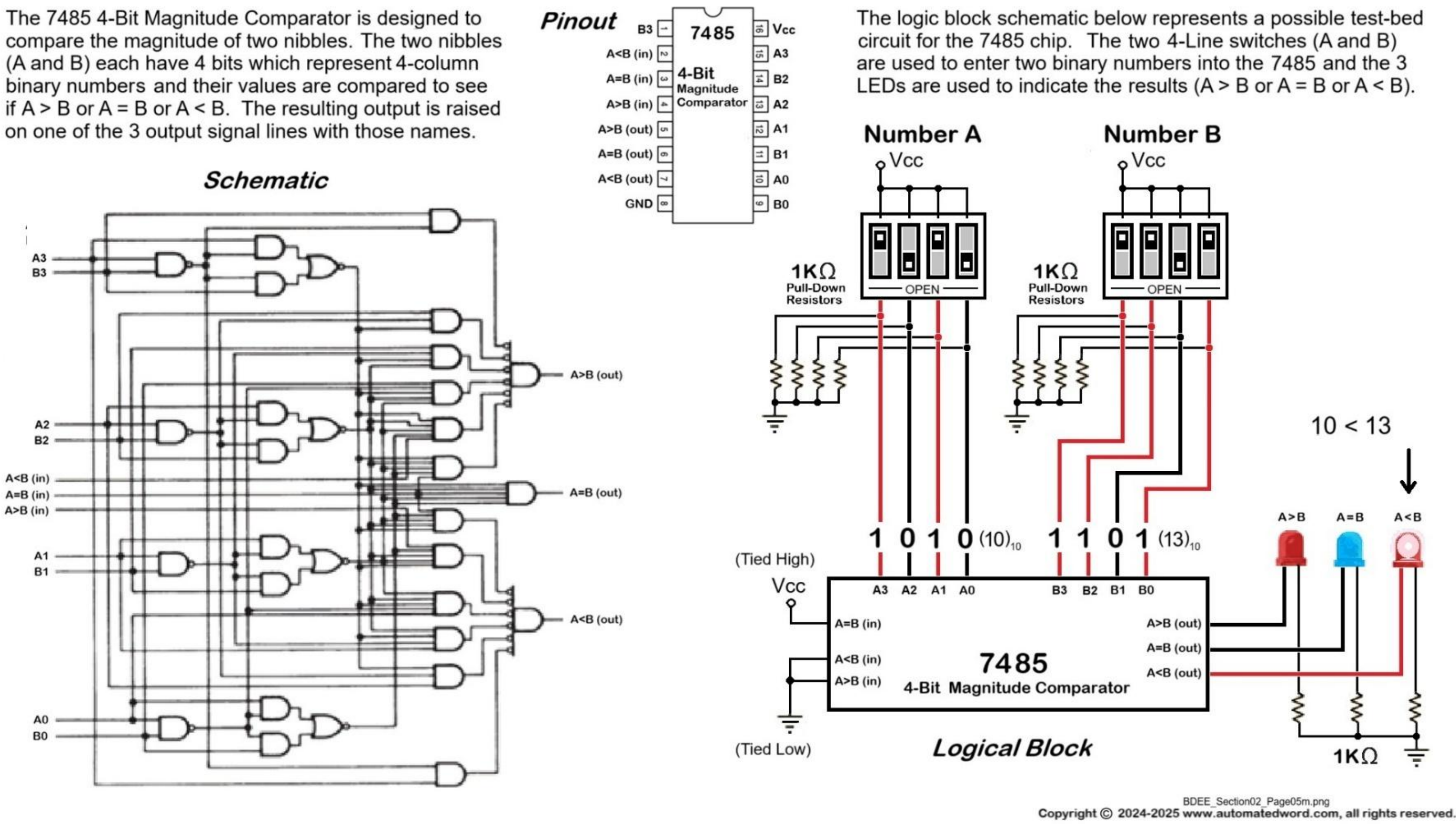
The switch position labelled 1 is off and that means a 0 (0-volts) is being fed to pins 4 and 5 of the 74138. That means that the two enabling pins (4 and 5) are being satisfied for the moment because they are expecting an active low signal - meaning that as far as pins 4 and 5 are concerned, the 74138 can perform its function. However, pin 6 (E1) is also an enabling input pin but it is connected to switch position 2 of the DIP switch and is also receiving a low signal (0-volts) because switch 2 is off but, pin 6 (E1) is expecting an active high (5-volts) signal. This means the 74138 is still disabled because all three enabling inputs must be receiving their expected active signals in order for the 74138 to perform its function.

The first thing to test is to enable the 74138 by applying an active high signal to E1 (pin 6) of the chip by turning on switch at position 2 of the DIP switch. When you do this, it is enabling the 74138 to make a selection of one of its outputs. Because the three address switches on the right-hand side of the DIP switch (switches labelled 6, 7 and 8) lead to pins 1, 2 and 3 of the chip which are A0, A1 and A2 respectively are all off, these input pins are receiving 0-voltage – this means the address being read into the chip is 000. So when the enabling pin E1 is turned on by turning switch position 2 on the 74138 will select O0 to go low (0-volts) because the output from the chip are called active low signals. This, of course, should cause the first LED light on the left-side of the LED panel to turn off. Since the 74138 is designed to have only one output signal to be active (active low that is), you should never see more than one light of the LED panel to turn off.

You should test each light of the LED panel by setting the three address switches to on or off for each of the eight numbers between 0 – 7 and verify that the corresponding LED light goes out (one at a time). Finally, test the "master" enabling switch position 1 (connected to pins 4 and 5) and turn it on and verify that all LED lights are on.

See if you can tell the difference between these two decoder chips in BDEE PDF Viewer: 74LS138 74LS139

Decision Making - Arithmetic Processing (7485 4-Bit Magnitude Comparator)

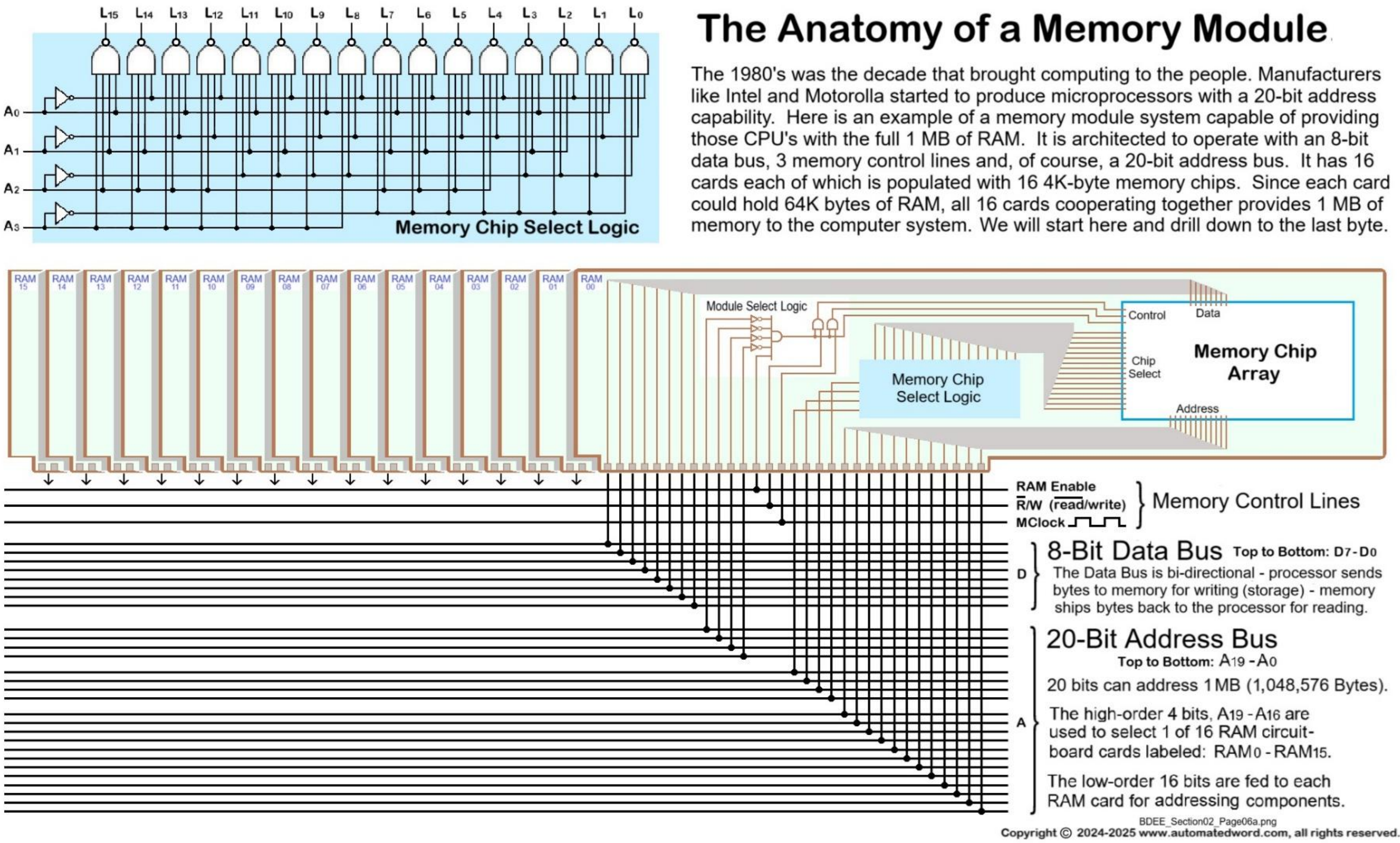


The 7485 4-Bit Magnitude Comparator like all digital logic is constructed with the common AND, OR and XOR logic gates that make up all digital electronic logic structures. As we see here in its schematic design, it is getting too complicated to enclose that schematic drawing into a pinout diagram. And, that's ok because we are not really interested in how the 7485 works – how it does it's job or what it's internal circuitry looks like. For the most part we are no longer interested in schematic diagrams of numerous fundamental logic gates but instead, we will be looking at larger circuit diagrams where each chip is represented as a logic block as in the diagram of the 7485 test-bed circuit above.

In a logic block type diagram or circuit schematic, the inner workings of components are hidden away as if they were black boxes – and that is exactly the way we should look at them. Notice that the logic block diagram of the 7485 does not show us the pin numbers for the physical chip but only the little pin names (or mnemonics as they are sometimes called – a "mnemonic" is a reminder, a prompt or memory aide). Also, in logic block schematics, the logic block diagram of a component, both power (Vcc) and ground connections for the chip are not shown because it is always assumed that those connections would have to be made in a physical circuit implementation in order for the chip to function.

Although the basic concept is that the circuit designer will diagram circuits using logic blocks and pin mnemonics while the electronic engineer or technician who will actually implement a physical circuit will be left to make the physical pin connections by pin-number. So, for the remainder of this course, you will have to associate each pin mnemonic with the actual pin numbers which will give you great practice at doing both jobs. But don't worry a "pinout" diagram for each component will be provided for all the exercises that have a logic block type schematic – which will be all of them.

The logic block does show connections to power and ground for inputs that require a logical setting. These "Cascading" Inputs: "A>B (Input), A=B (Input) and A<B (Input) will be discussed in Section 3.



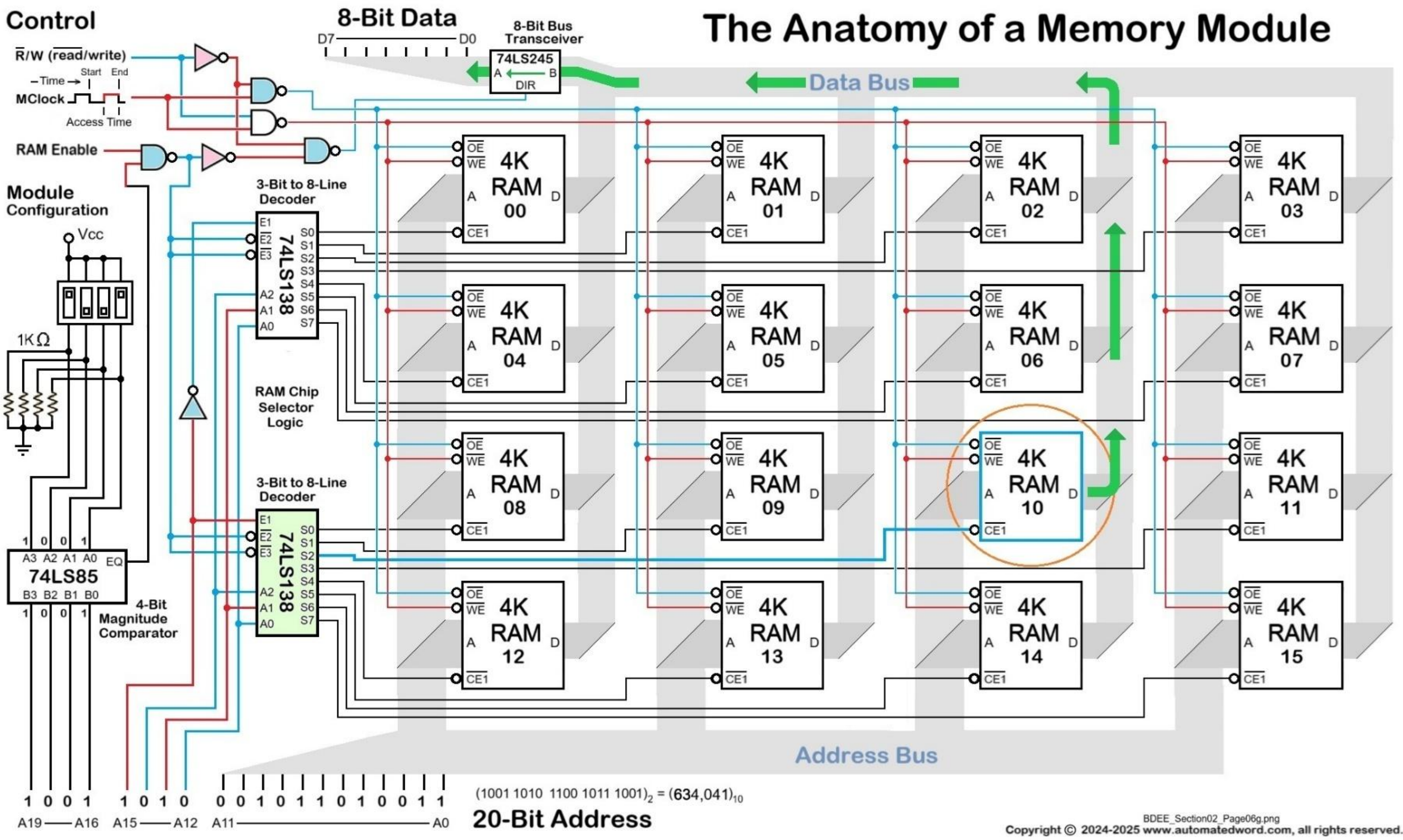
This is a diagram of a 16-Card RAM memory system for an early stage microcomputer – “micro” here means the CPU is packaged on a single chip. This was a huge breakthrough in the late 1970's which led to the large scale proliferation of desktop computers available to the population at large throughout the 1980's. This was the beginning the era of personal computing – no longer a corporate phenomenon.

Electronic manufactures produced the first serious desktop computers like Apple with the Lisa and Macintosh and IBM making the IBM PC (Personal Computer). They were able to do this because the CPU's from Intel and Motorola where able to address over one million bytes of data using a 20-bit address bus in their immediate random access memory modules instead of the measly 65-thousand bytes of the older 16-bit address bus systems.

Actual single chip CPU's started in earnest in the 1970's with a 16-bit address bus. 16-bits means 16 columns of 1's and 0's and this can represent 16 parallel on/off lines capable of holding a maximum value of 65,535 which is 65,536 possible numbers from 0-65,535 (64KB). A 20-bit address is capable of individually addressing 1,048,576 Bytes (over 1 million) and those would be the numbers 0-1,048,575.

In case this was not covered before, it should be understood that one-K, in digital electronic and computer terms means 1,024 and not the value 1,000 as it does throughout the rest of mathematics. This is because of the binary number system that is inherent in digital electronics. 16-bits is 16 columns and this means the maximum number of values that can be represented by 16 columns is 2 to the 16th power - which is 1024. Likewise, a 20-bit address has the capacity of 2 to the 20th power which is 1,048,576 – one mega-byte (1MB).

The schematic on the next page will illustrate the logic structure of a RAM memory module (or card)...



This is the schematic of a 16-Chip RAM memory module (card). It is one of 16 cards each of which contains 16 4KB chips. KB stands for kilo-byte – a kilobyte is 1,024 bytes so 4KB is 4,096 bytes. Since each card has 16 4KB chips then each card can contain 64KB of RAM which is 65,536 bytes. Along the bottom left edge of the diagram is the 20-bit address provided by the computer's CPU when it is going to store or retrieve a single byte of data to or from its RAM system of 16 64KB cards. A 20-bit address is enough binary columns to provide a unique numerical address for 1MB of RAM (1 mega-byte of RAM).

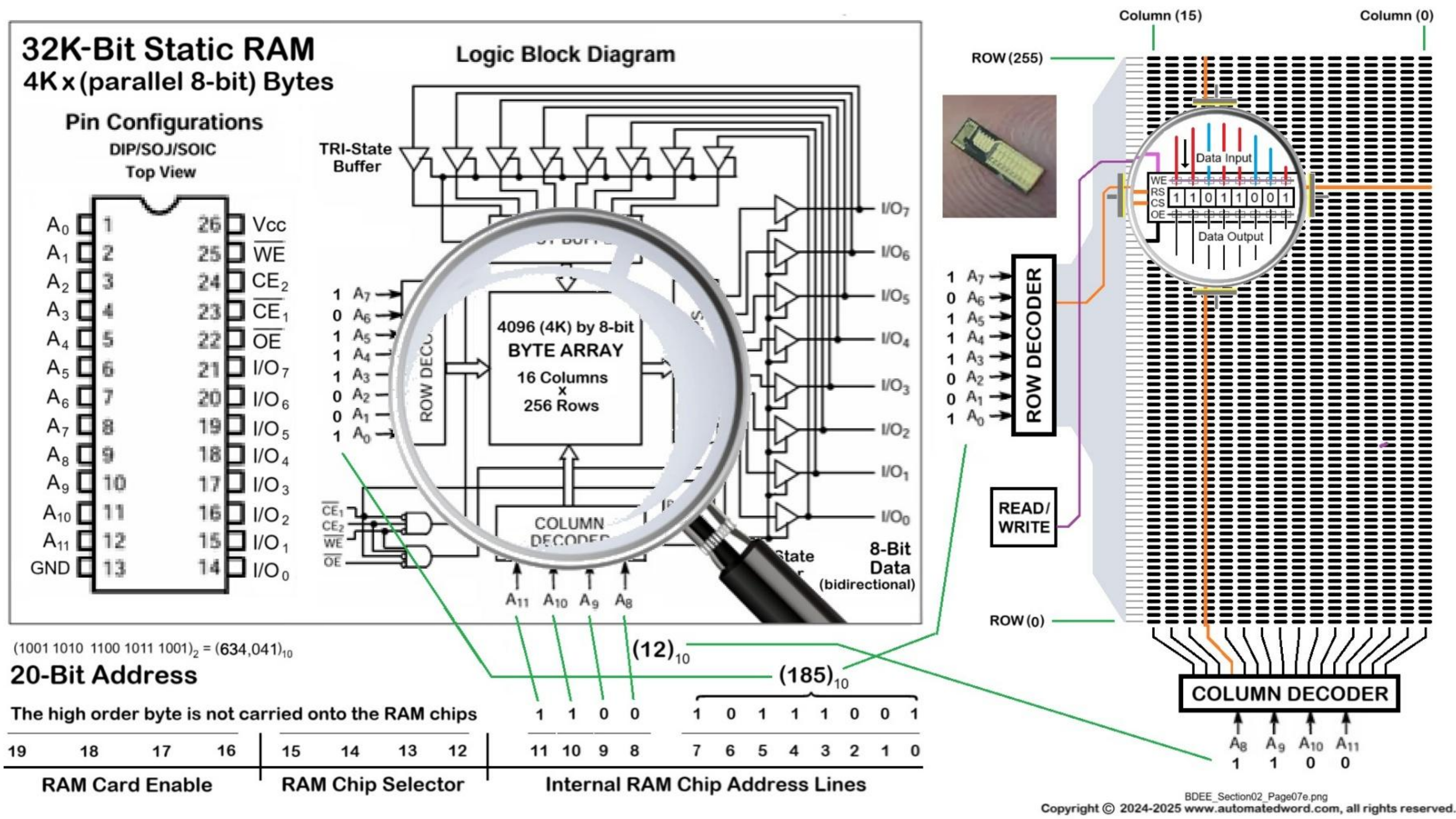
For the purpose of demonstration, an arbitrary address of 634,041 is provided along the bottom edge 20-bits of the address bus lines. During a RAM memory cycle, whether it is a read-cycle or a write-cycle, each card in the 16 card RAM system must determine if the address the CPU has placed on the address bus is on this particular card and, if so, into which of the 16 4KB chips the entire address is referring to. The circuitry on each card is designed to determine whether the address refers to this card by the high-order nibble (a16 – A19) and, if so, to which chip is to be selected by the next high-order nibble (a12-A15).

Each card in the RAM system receives control signals from the CPU on the system control bus lines and these are depicted in the upper left corner of the diagram. These control lines feed into 4 discrete NAND gates and 2 NOT gates which provide logic for proper operation of a read-cycle or a write-cycle. The Read/Write control signal indicates whether the memory cycle will be the CPU reading a byte from RAM or storing a byte into RAM. Note that all 16 RAM system cards are dormant until RAM Enable signal is high.

The overall memory-cycle is driven by MClock for a two-step process. The settling time begins when MClock goes low and at this same moment the CPU places the full 20-bit address on the address bus and sets Read/Write low if this is a memory-read cycle or, sets it high if this is a memory-write cycle. If this is a write-cycle, the CPU will have placed a byte of data on the data bus for storing into RAM. The diagram shows a memory-read cycle where the CPU is waiting for the RAM chip to deliver a byte at execution time.

  (Back to TOC)

A RAM Chip Stores/Fetches a Byte at the Address on it's Address Bus



Here is a page from the manufacturer's specification sheet (spec sheet) for this particular RAM chip. It shows the pinout diagram of a 26-pin DIP package on the left and a very generalized block diagram of it's internal electronic layout on the right.

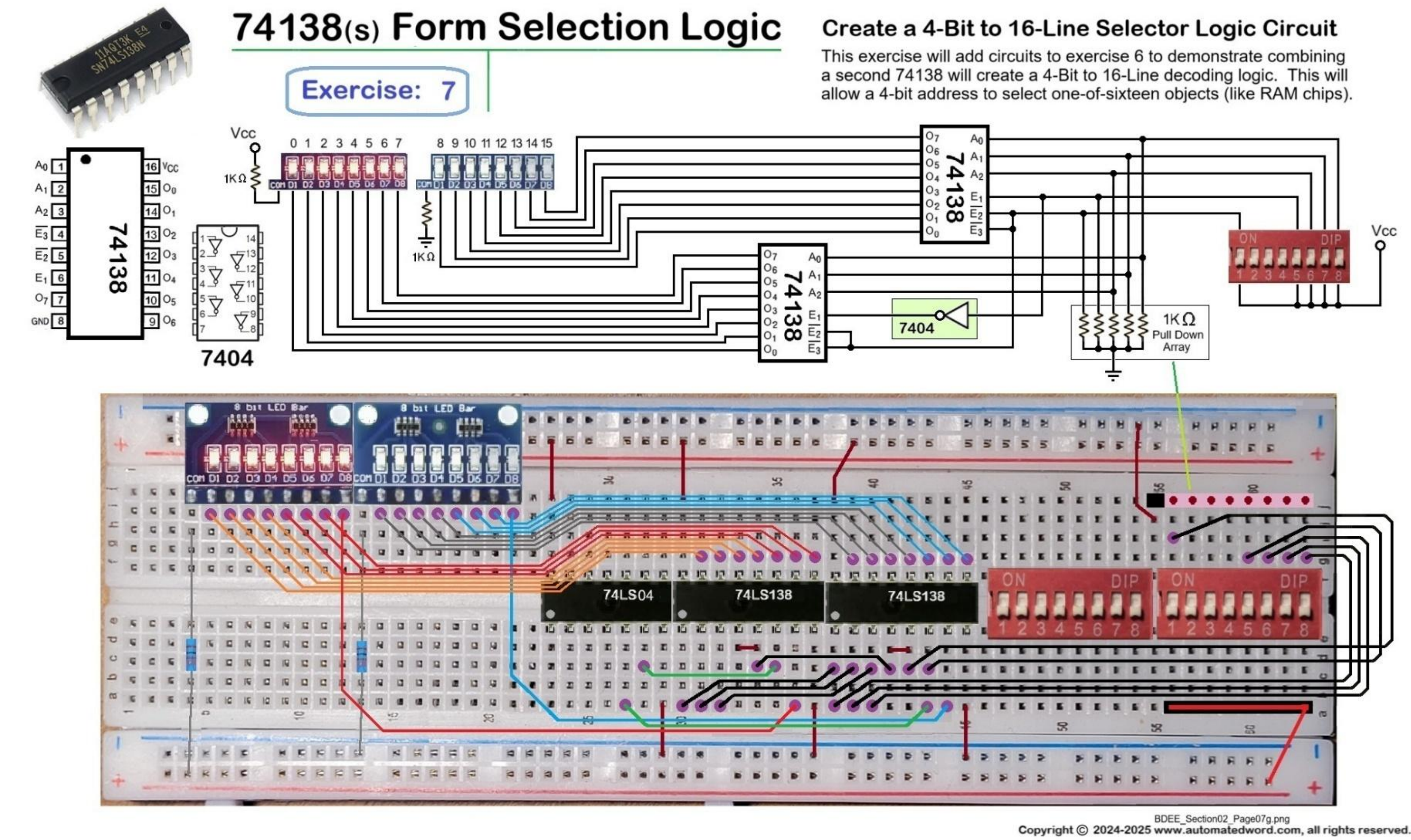
In order for the RAM chip to single out a byte for recording or reading, it must carry on what we started at the module level and that is to decode the address. The low order 12-bits of the system address are carried on to each chip and, in the case of this specific 4-K chip, the high-order nibble is used to decode one of 16 columns and the low-order 8-bits are decoded into one of 256 rows. In this addressing scheme, only the single memory cell who's column and row have been activated will be allowed to write the content of the data bus into itself or place its content onto the data bus for the system to read.

Remember, the data bus in a typical computer system is bi-directional - meaning, in the case of a CPU and it's memory system here, data is allowed to flow from the CPU to the RAM chips when the CPU is writing a byte to memory and when it is reading a byte from memory, the data is allowed to flow from the activated RAM chip all the way back to the CPU on the data bus. This would seemingly be hard to accomplish given the rule, if you remember, that outputs from logic gates cannot be connected to each other. So here is how that exception is accomplished...

The "TRI-State" buffers shown in the block diagram are special buffers that, when activated, will actually disconnect themselves from the data bus when needed. More on this topic will be explained later in Section 3 of this course.

Notice in the above diagram the memory cell contains a byte with the value of 217 and that the Output Enable (OE) line is active so the cell is placing the byte on the data bus for transfer back to the CPU.

See the Datasheet for the 4K-Byte memory chip in the BDEE Quick Datasheet Viewer: 7134SA/LA



This demonstrates how to utilize the enabling aspects of the 74138s in order to form a 4-bit to 16-line decoding mechanism which is efficient for a memory board that needs to select one of 16 chips at a time per memory-cycle. Since each 74138 does exactly the same thing with its 3-bit address input lines, it is just a matter of disabling one decoder while enabling the other from the high-order bit of the incoming 4-bit address. Each 3 x 8 decoder is responsible for marshaling a group of eight RAM chips.

The 8-line DIP switch is there to provide a test-bed simulation of the CPU's control and address bus signals that would be sent by the CPU during a memory cycle. If this exercise is a representation of the chip selection logic as depicted on page 13 (Anatomy of a Memory Module) then the 4 switches labelled 8, 7, 6 and 5 on the right side of the DIP switch represent system address bus lines A12, A13, A14, A15 respectively. The switch at position labelled 1 represents the "master" enabling line as seen on page 13 which is the output from the NAND gate being fed by the CPU's RAM Enable control line and the module select logic coming from the module configuration circuitry.

To perform the test of the constructed logical selection circuit in this exercise, you will run through all the numbers that can be provided very similar to Exercise 6 except, this time, you have 16 numbers to run through (0 – 15). Finally, like Exercise 6, you should turn on the "master" enabling switch (labelled 1) and you should see the circuit go into its dormant state. This, of course, will be indicated by all red LED lights turned off and all blue LED lights turned on. The following is a quick explanation of the red LEDs…

The red LED panel is a wiring trick. Each LED light in this panel is tied in common through their anode leads which means they must be connected to power in order to light up. Ultimately, this means that Only a low signal (0 volts) must be applied to its cathode lead in order for the LED to turn on. This might be fun to see the light indicate one of the 74138 outputs going into the active-low state but, it is deceptive in the sense that while it is indicating that signal is active but not indicating that the signal is active-low.

  (Back to TOC)