In this video you'll learn some more technical differences between CPUs. Specifically you'll learn about the Data Word Size, and CPU Bus Size.

CPU Word and Bus Size

Another technical difference between CPUs is the Word size or Data size. This refers to the number of bits used to store data in the CPUs registers and cache. A larger bit size means that the CPU will be able to process larger numbers in a single operation. The 8088 CPU that was used in the original PC could only work on numbers that were 16 bits or 2 bytes in each operation. The CPUs used in PCs evolved to handle 32 bits, and at this point in time are able to handle 64 bits internally.

Does the number of bits a CPU have any effect on performance? The short answer is once again "it depends". For certain operations, such as integer math on very large numbers, a 32 bit CPU will be over twice as fast as a 16 bit CPU; and a 64 bit CPU will be over twice as fast as a 32 bit CPU. Or for math with floating point numbers, or numbers with fractions or decimals, more bits will mean more precise math in fewer operations. But for other CPU tasks it may not make any difference at all. For example, a larger Data Word Size won't make any difference in math with small integers or playing music.

The long answer requires some explanation and some math. A CPU working with 16 bit integer numbers means that 2^{15} or 32768 is the largest value that could be used. (It's not 2^{16}, because one bit is used to indicate whether the number is positive of negative, leaving 15 bits to store the integer value. And for those of you who really understand binary, twos complement would allow a positive value of 32769.)

The CPU could work on larger numbers, but it would require multiple operations. For example if you wanted to add two 4 byte numbers, the CPU would first add the lower 2 bytes of each number. If this addition resulted in a carry over into the 16^{th} bit, then this value would have to be added to one of the upper 2 byte values. The upper 2 bytes of each number would then be added together.

Here's another example using decimal numbers instead of binary, which might make it easier to understand. Let's say we have a CPU that can only handle 2 digit decimal numbers, and let's not worry about whether it's positive or negative; so the largest number we can have is 99. To add larger numbers, for example 175 + 1265 the CPU would use this process. First it would break each number into a lower half, or the first 2 digits, and an upper half. Next it would add the lower halves together, so in this case it would add 75+65, which gives 140. But since it can only store 2 digits, it would again break this into an upper and lower half, or 1 and 40. The upper half from this first addition would have to be "carried over" to the addition of the original two upper halves. Next the CPU would the original two upper halves, or 1 + 12, which results in 13. And last, it would add in the carry over, 1 + 13, which results in 14. The final value would require displaying the results of both halves, or 14 and 40, or 1440.

There are a couple of important concepts that you need to understand here. The first is that the bit size of a CPU limits the range of numbers it can work with in a single operation. The second is that larger values can always be dealt with, but it will take multiple operations and more CPU cycles. The third, is that most real world integer math can be accomplished with either 16 or 32

bit numbers, because these allow for very large numbers. As we discussed with 16 bits you can get numbers up to ~32,000, which is a pretty big number. This is why there's only a performance gain in limited cases, when you're dealing with really big numbers; which doesn't happen very often.

CPU and Max Memory

The biggest impact of the word size may be its relationship with the maximum amount of memory in the system. The CPU uses integers to store the address of where it stores data in RAM or System Memory. Because of this the old 16 bit CPUs could only use maximum of 16 Mbytes of RAM. When these older CPUs were first introduced no one imagined that a computer would ever need or have a Gigabyte of RAM, much less the 16 or 32 GB in current computers. In fact there's a famous myth that Bill Gates said "*640K ought to be enough for anybody"*

http://quoteinvestigator.com/2011/09/08/640k-enough/

So being able to address 16 Mbytes was seen as a limitation. The same thing happened when 32 bit CPUs were introduced. The 32 bit Word Size resulted in a memory address limit of 4 GBytes, but computers at that time had maybe 16 Mbytes, so again it wasn't seen a big deal.

Today's 64 bit CPUs can address what seems like an almost unlimited amount ~ a billion GBytes. But memory is a lot like your garage or closets. Do you remember when you first moved in? Did is seem like you had extra room in your garage or closet? How about now? I know what you're thinking … "I need more room!" They always seem big enough at the start, but sooner or later you fill them up with junk and need more room.

CPU Internal Bus Size

Now let's talk about the CPU internal bus size is also measured in bits, and went from 16 bits, to 32 and is currently at 64 bits. The bus is the set of electrical lines that are used to connect the CPU to the outside world, and to connect the different units, like the ALU and FPU and cache inside the CPU. Each bus line corresponds to one bit of a binary number. So a 16 bit bus will have 16 electrical lines, a 32 bit bus will have 32 lines, etc. Any time data is transferred, the component doing the transfer goes to each line of the bus and either charges it with a voltage or not. The component receiving the transfer looks at the bus and if there's a voltage on the bus line it marks that bit of the corresponding number as a 1. If there is no voltage on the line it marks that bit of the corresponding number as a 0.

Whether or not this results in a gain in performance depends on the specific task being performed. If you're sending 16 bits in a single transfer, then there's no performance gain. But if you need to send 64 bits then it can accomplished in one transfer with a 64 bit bus, or 2 transfers with a 32 bit bus, or 4 transfers with a 16 bit bus.

So once again you'll see general performance gains as you increase the CPU bus width, but the results may vary if you're looking at specific tasks.

32 or 64 Bit Instruction Set

Another difference between CPUs is the instruction set. Most people know that there's some difference between 32 Bit and 64 Bit CPUs, but they don't know exactly why. The difference is almost the same as the difference between the CPU that runs your phone and the CPU that runs your computer. They use different instruction sets.

When a program is compiled or interpreted it's changed from instructions that humans understand to the instructions that the CPU will understand. Each instruction is actually a number, so a program will be series of numbers. When a program is run, the Instruction Decode Unit looks at each instruction number and decides what to do next.

As we discussed earlier all CPUs do pretty much the same things, add and subtract numbers, compare numbers, etc. And all CPUs use numbers for their instructions, but all CPUs don't use the same numbers for each instruction. This is one of the reasons why you can't run Windows software on a Mac, or iPhone Apps on an Android phone. Generally speaking the programs do the same things, but the specific instruction numbers will be much different.

For over 30 years all Intel CPUs and Intel clones such as AMD used the same instruction set. This is why you could still run applications you purchased in the 1990s on computers you bought in the 2000s. With each iteration of their CPUs Intel would add new instructions, but they wouldn't change the numbers for any of the existing instructions. In other words the Pentium and Core Duo CPUs used the same base instruction as the original 8088 CPUs from the late 1970s! However, they added new or advanced instructions with each new generation of chips, so Pentium chips have extra instructions that the 286, 386 and 486 chips don't have; and the Core Duo chips have more instructions than the Pentium chips.

An easy way to understand this is to think about calculators. The original calculators had all the basic math buttons for addition, subtraction, multiplication and division. The next generation added button for calculating x^y and square roots, but they kept the basic math buttons as well. And the next generation added buttons for doing the trig functions, but again they kept all of the existing buttons.

This means you can always use a new calculator to solve even old problems, but you may not be able to use and old calculator to solve a new problem. For example, you wouldn't be able to press a button and find the cosine of 37^o if you used an old calculator. You could look at all the buttons and sooner or later say "I don't have that button!"

The same is true for CPUs. If you send it an instruction number it doesn't recognize it throws up its hands in frustration and says "I don't have that instruction!" But since the CPU doesn't have hands it displays the Blue Screen of Death (BSOD) with the error message "Invalid Instruction".

For many years, there's been little to worry about when purchasing and installing PC software. Some applications, in particular games, do require a specific level of processor or better, because they use some of the advanced instructions. If this is the case, the install program usually checks and won't even install if the program won't run.

Intel introduced a major change to the instruction set when they introduced the 64 bit processors. They decided to no longer use the same base instruction set that they had been dragging around since the late 1970s. This is why there are 32 bit and 64 bit versions of the Windows OS, and some applications. You won't be able to run the 32 bit version of Windows on a 64 bit processor and vice-versa. This is why the install program won't even let you install the wrong version.

You may be able to run some 32 bit applications on a 64 bit processor, but they will run slower because the OS will have to translate the instructions to the 64 bit versions before running the program.