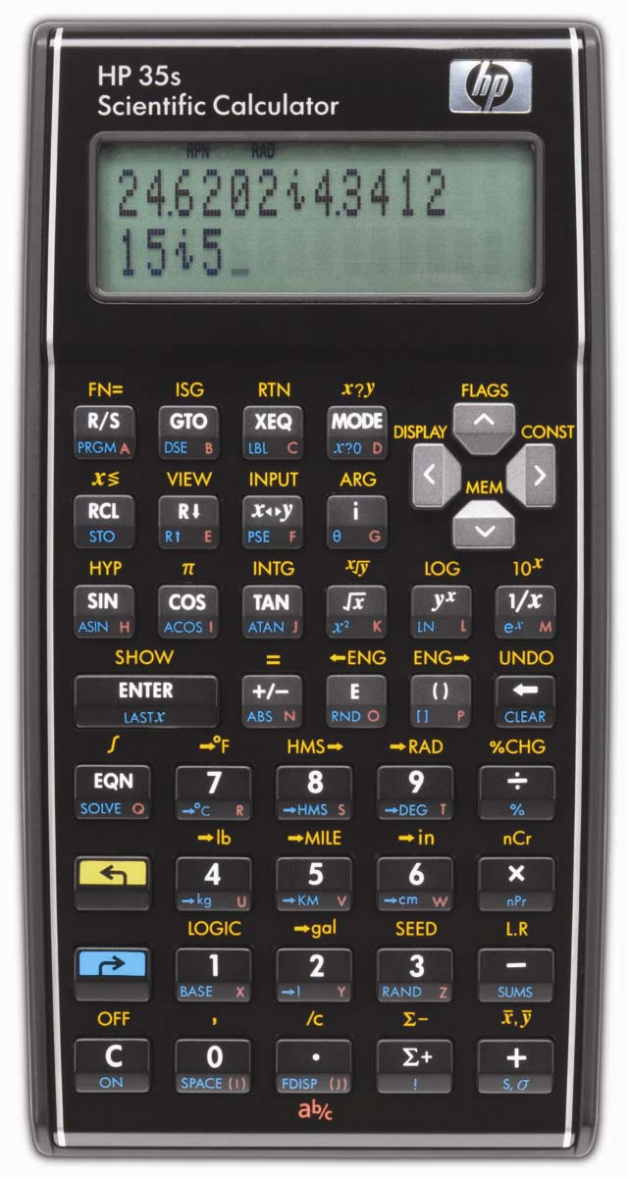
hp calculators

HP 35s Using the LOGIC functions

Numbers in different bases

Operations on binary numbers

Practice manipulating binary numbers



Numbers in different bases

Most numbers we work with day-to-day are in base 10. There are applications within the computer world that require the use of numbers in other bases. The number 24 in base 10 can be translated into base 16 by the following procedure. Just as each digit's location in base 10 can be thought of as a power of ten (the ones' place, the tens' place, the hundreds' place, etc), each digit's location in base 16 can be thought of as a power of 16. Each digit in a base ten number can hold a value from 0 to 9. In base 16, each digit can hold a value from 0 to F, where F corresponds to the value 15 in a base 10 number. Translating 24 from base 10 to base 16 would require a 1 in the second location of the base 16 number (and would convert 16 of the 24 number's value) and an 8 in the second location of the base 16 number. Therefore, 24 base 10 is equal to 18 in base 16. A similar process could be used to convert 24 base 10 to base 8 or base 2.

On the HP 35s, numbers can be represented in bases 2, 8, 10 and 16, or binary, octal, decimal and hexadecimal. The HP 35s can work with numbers in bases 2, 8 and 16 that are 36 bits in length or less. Since the leftmost "bit" is used to indicate a negative number, the largest positive binary number is 0 followed by thirty-five 1's. This means that the largest hexadecimal number that can be entered or generated as an answer is 7FFFFFFFF, (equal to 34,359,738,367 in base 10, and 377777777777 in base 8). This is because the HP 35s uses a 36 bit binary word space to represent numbers in these different bases. Decimal numbers are not limited in this fashion, since they can be represented as floating point numbers. The HP 35s calculator provides the ability to easily work with numbers in different bases, as the following sample problems illustrate.

In RPN mode, the third row of keys on the HP 35s (SIN through 1/x) can be used to enter the hexadecimal digits A through F. In algebraic mode, it is necessary to press RCL and then the appropriate letter key to enter these digits.

Operations on binary numbers

Any binary number, regardless of the base in which it is displayed, can still be thought of as a collection of 1's and 0's. For example, the number 24 in base 10 is also the number 11000 in binary. This is because 11000 in binary is equal to 1 x 24 + 1 x 23, or 24. The HP 35s contains many functions in the ↩ LOGIC menu, shown below, that operate on binary numbers.

Figure 1

Figure 2

The AND function compares two binary numbers at the bit-by-bit level and creates a new binary number with a 1 for each bit position that contains a 1 in both numbers in the same position.

The XOR function (which stands for exclusive OR), does the same thing as the OR function, but only for positions where the original numbers contained a 1 and/or 0, but not where both contain a 1.

The OR function compares two binary numbers at the bit-by-bit level and creates a new binary number with a 1 for each bit position if either original number contains a 1 at the same bit position.

The NOT function creates a new binary number where each bit position's value has been "flipped", with each 1 becoming a 0 and each 0 becoming a 1. This is referred to as the one's complement of the argument.

The NAND function creates a new binary number from two input binary numbers where each bit position's value is based upon a NOT (x AND y). In essence, it returns a bit of one unless both bit positions in the two input binary numbers are ones.

The NOR function creates a new binary number from two input binary numbers where each bit position's value is based upon a NOT (x OR y). In essence, it returns a bit of one ONLY when both of the bit positions in the two input binary numbers are zeroes.

Note: Numbers entered into the HP 35s are assumed to be decimal numbers, regardless of the base mode, unless the proper suffix of "b", "o", or "h" is supplied. These suffix characters are found in the BASE menu as choices 5 through 8, and are shown below in Figure 1. It is not necessary to enter the "d" for decimal numbers, except for clarity in a program.

Figure 3

Practice manipulating binary numbers

Example 1: Evaluate NOT(#4567 d). Make sure the HP 35s is in DEC mode to enter the base 10 number.

Solution: In RPN mode, press: [↱] [BASE] [1] then press [4] [5] [6] [7] [↰] [LOGIC] [4]

Figure 4

In algebraic mode, press: [↱] [BASE] [1] then press [↰] [LOGIC] [4] [4] [5] [6] [7] [ENTER]

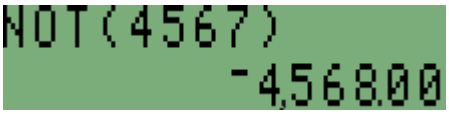


Figure 5

Answer: -4568 base 10. Figure 4 shows the result in RPN mode while figure 5 shows the result in algebraic mode.

Example 2: Perform an OR on these two binary numbers: #70114 o and #57610 o. Make sure the calculator is in OCTAL mode to enter the base 8 numbers.

Solution: [↱] [BASE] [3]

In RPN mode, press:

[7] [0] [1] [1] [4] [↱] [BASE] [7] [ENTER] [5] [7] [6] [1] [0] [↱] [BASE] [7] [↰] [LOGIC] [3]

In algebraic mode, press:

[↶] [LOGIC] [3]
[7] [0] [1] [1] [4] [↷] [BASE] [7] [›] [5] [7] [6] [1] [0] [↷] [BASE] [7] [ENTER]

Figure 6

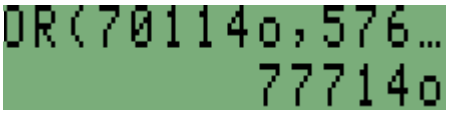


Figure 7

Answer: 77714 base 8. Figure 6 shows the result in RPN mode. Figure 7 shows the result in algebraic mode.

Example 3: Perform an NOR on these two binary numbers: #1011 b and #1001 b. Make sure the calculator is in BINARY mode to enter the base 2 numbers. Do not forget to append the "b" suffix to the binary numbers.

Solution: [↷] [BASE] [4]

In RPN mode, press:

[1] [0] [1] [1] [↷] [BASE] [8] [ENTER] [1] [0] [0] [1] [↷] [BASE] [8] [↶] [LOGIC] [6]

In algebraic mode, press:

[↶] [LOGIC] [6]
[1] [0] [1] [1] [↷] [BASE] [8] [›] [1] [0] [0] [1] [↷] [BASE] [8] [ENTER]

Figure 8

Figure 9

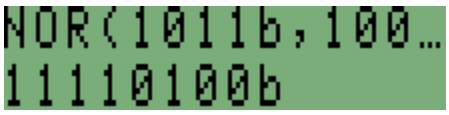


Figure 10

Answer: Figures 8, 9, and 10 show the result in algebraic mode. This display indicates an arrow pointing to the right which indicates that the answer scrolls off the screen in that direction. Press [↷] [›] to view the rest of the result. The leading 1's in the answer are due to the 36 bit word length on the HP 35s. Since both binary numbers would have had all leading bits equal to zero, the NOR function returns a 1 for each of those zero bits. The last four bits reflect the NOR of the entered digits. Only the second entered bit was a zero in both numbers and NOR returns a 1 for that bit location.