From the time of their invention in the 1940s until the turn of the 21st Century, Successive Approximation ADCs were the most common choice for high resolution, low cost, intermediate speed digitization. They remain popular except for the very highest resolution digitization.

It is CRITICAL that a sample and hold stage precede a successive approximations converter! As we will see, the sampled signal must be stable for the entire duration of a conversion.

A successive approximations ADC has much in common with the children's classic, "The Story of the Three Bears." At each stage of the story, results are too hot, too cold, or just right, too big, too small, or just right, etc. Because measurements are hardly ever "just right" (due to noise), a guess at an answer to any question will likely be too high or too low. So if one starts making large amplitude guesses, then takes progressively smaller steps, one will always be too high or too low, but will iterate to "very close to just right."

The key here is that we synthesize a voltage using a DAC, compare the DAC's output voltage to the signal input voltage, then increase or decrease the DAC's output until the code feeding the DAC gives the closest possible match to the potential of the input signal. For the simplest possible example, let's use a 1 bit DAC. The full scale signal will be +5 V, and 0 will correspond to 0. Thus, if the input is over +2.5 V, the output should be with the DAC's bit set. Otherwise, the DAC output should be 0. How do we proceed?

1) Trigger the sample and hold to hold the input value.

2) Guess that the digitized value is 1. Feed that logical 1 to a 1 bit DAC which puts out 1/2 of the ADC's full range, or +2.5 V.

3) Use a comparator to look at the DAC output and the held signal from the sample and hold. If the sampled value is ≥ 2.5 V, the comparator output will be high. If the sampled value is < 2.5 V, the comparator output will be low.

4) If the comparator output is high, the digitized value is 1; otherwise, the single bit is too big to represent the input signal, so we set it back to 0 and we're done.

A picture may help:
Note that if the DAC output is greater than the sample and hold output, the comparator output is high, while if the DAC output is lower than the sampled value, the comparator output is low.

Why stop at 1 bit? Why not 8 bits or 12 bits or 16 bits? In fact, we only need one more insight and all of these possibilities snap into focus. We need to set the most significant bit first. Let's work that out using a 2 bit converter. Again, use +5 volts full scale and have an input that we’re trying to digitize of +3 V. The values for a 2 bit DAC are -- oh, go ahead. Fill in the table:

<table>
<thead>
<tr>
<th>DAC bits</th>
<th>DAC Output V</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

So now we can see how the ADC would arrive at the digitized value for output.

1) Set DAC to 10. DAC puts out 2.5 V

2) Comparator determines that sampled voltage is at least as great as the voltage produced by code 10. The 1 stays set.

3) Set the next DAC bit to 1, for a coding of 11. DAC outputs 3.75 V.

4) Comparator determines that the DAC output is greater than the sampled voltage. The second bit gets turned off.

5) Final encoding: 10.

The digitization error in this case is 0.5 V; the resolution of the measurement is only 1.25 V, so the closest representation of a 3 V input we can have is 2.5 V.

Now let's make the leap to a 12 bit convertor. Resolution is now 1 part in $2^{12}$. Let's keep the conversion unipolar, full scale +5 V, and see how closely we can digitize +3 V.

Fill in the second column in the table, check your work, then work out the third and fourth columns. "MSB" is the most significant bit (what would correspond to the 1 bit example), while "LSB" is the 12th bit. In the third column, you will have a 1 or a 0 for each bit. In the fourth column, sum the values implied by the 1's and 0's in the second and third columns (Thus, if the voltages for the 3 most significant bits were 2, 1, and 0.5 V and the binary values were 101, that would be $2^1 + 1^0 + 0.5^1 = 2.5$ V).
There are several interesting things to see in the encoding. First, look at the pattern of the digitized voltage: 100110011001. In hexadecimal, that's 999. Why is there the repeating pattern? 3 V = 0.6 * 5 V in base 10. But 0.6*16 = 9.6, not an integer. 0.6 times ANY power of 2 (and thus for ANY grouping of bits into nybbles, bytes, or words) does not give an integer, and thus 3 V cannot be exactly represented in binary. Rather, it is represented as a repeating "decimal" (heximal?) fraction.

Second, if you didn't carry enough significant figures with you along the way, you might have rounded off the final result with fewer non-zero bits. For example, if you round off voltage values to 1.0 mV or finer, the coding is as shown above. At 10 mV resolution, the 4 most significant bits, 1001, rounds to 2.810 V. At 10011001, the summed voltage is 2.99 V. One could then get 3.00 V by setting the next bit, for a 10 bit approximation of 100110011. Implicitly, this gives a 12 bit encoding of 100110011000. Inadequate resolution (or noise) during digitization limits the precision of the final encoding.
Hybrid Converters

Suppose there's a signal that is always between 2.8 and 3.2 volts. The first 4 bits of the digitized word will always be 1001. Doesn’t it seem wasteful (4 comparator operations!) to start fresh every time to convert these bits when it is only the less significant bits that are changing? We could probably speed up the conversion if we didn’t waste time on digitizing the slowly-varying, large amplitude part of the potential. Engineers, being clever, have reached the same conclusion and have designed hybrid ADCs that use flash converters for the most significant bits, then successive approximations (or sigma-delta) for less significant bits. The first 8 bits are digitized in a single cycle and feed the 8 most significant bits of the DAC. As long as the output precision of the DAC is good enough, one can then do an analog subtraction of the DAC output from the original (sampled) signal to provide the input for the successive approximations part of the circuit. Here's a sketch:

Why digitize the 8 most-significant bits and then reconvert them to analog before subtracting? Recall that the comparators in the flash ADC are subject to error and thus they only approximately carry out digitization. If the output of the 8 bit DAC is PRECISE to 12-18 bits, then a precision offset can be introduced before the least-significant bits are digitized. The subtraction amplifier (the bottom operational amplifier in the figure) can be designed with gain so that differences of a few millivolts between the sampled voltage and the DAC output are presented to the second ADC at 2×, 10×, or 100× the actual difference, so the second ADC can use higher, less noise-susceptible potentials. Are there additional "gotcha"s? Yes -- and we'll discuss those under Bits, Noise, and Linearity.