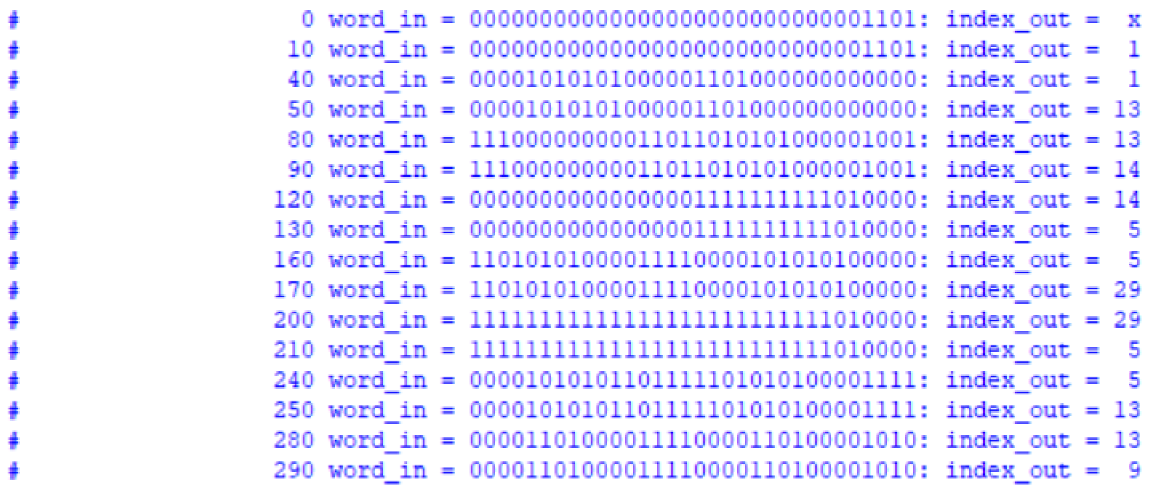
Output Table



```
#                      0 word_in = 00000000000000000000000000001101: index_out =  x
#                     10 word_in = 00000000000000000000000000001101: index_out =  1
#                     40 word_in = 00001010101000001101000000000000: index_out =  1
#                     50 word_in = 00001010101000001101000000000000: index_out = 13
#                     80 word_in = 11100000000011011010101000001001: index_out = 13
#                     90 word_in = 11100000000011011010101000001001: index_out = 14
#                    120 word_in = 00000000000000001111111111010000: index_out = 14
#                    130 word_in = 00000000000000001111111111010000: index_out =  5
#                    160 word_in = 11010101000011110000101010100000: index_out =  5
#                    170 word_in = 11010101000011110000101010100000: index_out = 29
#                    200 word_in = 11111111111111111111111111010000: index_out = 29
#                    210 word_in = 11111111111111111111111111010000: index_out =  5
#                    240 word_in = 00001010101101111101010100001111: index_out =  5
#                    250 word_in = 00001010101101111101010100001111: index_out = 13
#                    280 word_in = 00001101000011110000110100001010: index_out = 13
#                    290 word_in = 00001101000011110000110100001010: index_out =  9
```

Summary

- The design takes in a 32 bits input as word_in
- Checks for a 4-bit set of 1101 and marks the index of the MSB of the set
- If the set is not found in the first 4 bits the index shifts one bit to the right and looks at the next 4 bits
- Once the set is found the index of the MSB is reported as index_out
- If all 32 bits are checked OR a set of 1101 is not found index_out becomes zero