# CS 240 Lab 2
# Data as Bits

- **Ice Break!  Tell your lab partner about your weekend**
- **Binary and Hexadecimal Numbers**
- **Logic Diagrams**
- **Exclusive-Or for bitwise comparison**
- **Bit Puzzles**

# Binary and Hexadecimal Numbers

| base 16 | base 10 | base 2 | | | |
|---|---|---|---|---|---|
| **Hex** | **Decimal** | **Binary** | | | |
| | | **QD** | **QC** | **QB** | **QA** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 2 | 0 | 0 | 1 | 0 |
| 3 | 3 | 0 | 0 | 1 | 1 |
| 4 | 4 | 0 | 1 | 0 | 0 |
| 5 | 5 | 0 | 1 | 0 | 1 |
| 6 | 6 | 0 | 1 | 1 | 0 |
| 7 | 7 | 0 | 1 | 1 | 1 |
| 8 | 8 | 1 | 0 | 0 | 0 |
| 9 | 9 | 1 | 0 | 0 | 1 |
| A | 10 | 1 | 0 | 1 | 0 |
| B | 11 | 1 | 0 | 1 | 1 |
| C | 12 | 1 | 1 | 0 | 0 |
| D | 13 | 1 | 1 | 0 | 1 |
| E | 14 | 1 | 1 | 1 | 0 |
| F | 15 | 1 | 1 | 1 | 1 |

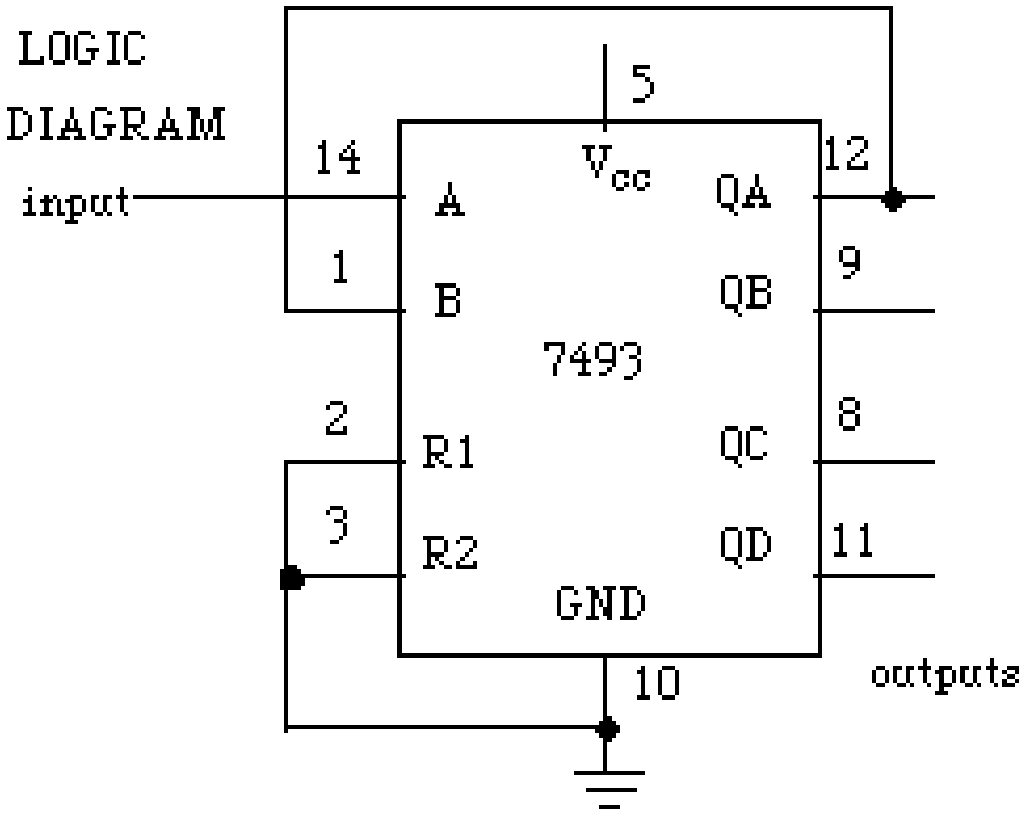
$15 = 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$

**Notice pattern and frequency of change for each digit as values progress numerically**

## Logic Diagrams

NOTE: logic diagram is not the same as pin-outs! Shows information about the logical operation of the device.

- Inputs on left side of diagram
- Outputs on right
- Voltage shown on top
- Ground shown on bottom



**Exclusive-Or function**

Useful for comparisons

A **parity bit** is an extra bit of information which is sent when data is transmitted, to check for errors in transmission. For a given set of bits, the number of bits whose value is 1 is counted. The parity bit is an extra bit which is also sent with the original data. The party bit is set to 0 or 1 to make the total number of 1 bits even.

| A | B | C | $P_{even}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Bit Puzzles**

```
/* isPower2 returns 1 is x is a power of 2, and 0 otherwise
   isPower2(5) = 0, isPower2(8) = 1, isPower2(0) = 0

    No negative value is a power of 2

    Legal operations: ! ~ & ^ | + << >>

    Max operations: 20

    Rating: 4
*/

int isPower2(int x) {
  return 2
}
```

You must write C code to return the correct value for a given input, given the constraints on which and how many operations you are allowed to use in your solution.

Constants must not be larger than 0xFF (decimal 255)

You may not use conditionals or loops

You must also explain your solutions!

**Tips**

Although integers are 32-bit values in this program, assume a smaller number of bits in your handwritten examples to make your binary numbers easier to work with

Handwrite some specific binary values and manipulate them with boolean operators.

Here are some simple manipulations and tips which may help you find a solution:

- Complement the number
- Either add 1 or add -1 (if doing so results in some useful characteristic)
- Mask (bitwise AND with a mask value to isolate bits)
- Shift left and then right again (or vice versa)
- Use Exclusive OR to compare values
- Bitwise OR a general solution with a special case (such as 0)
- !(0) = 1, but !(any other number) = 0