1. A memory has **2048 cells** with **16 bits** stored for each cell.

   **How many bits** can the memory hold?

   **2048 cells X 16 bits/cell = <u>32 768 bits</u> total**

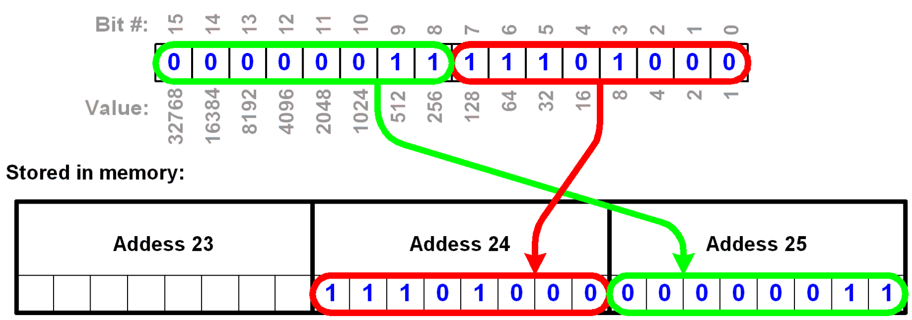
   **How big** does the **memory address** have to be?

   **$2^{11}$ = 2048 combinations, so the address must be 11 bits long**

2. Show how the **decimal number "1000"** would be stored as a **16-bit binary number at memory address 24**. Assume that the memory holds 8 bits in each cell and uses Little Endian byte ordering:

   - **1000 = 512 + 256 + 128 + 64 + 32 + 8**
   - **multibyte number starts at given address and includes as many following cells as necessary**
   - **Little Endian means low-order byte goes first**

**As a 16-bit binary number:**

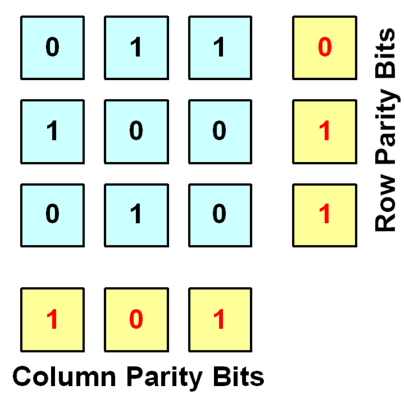



3. (a) **Fill in the parity bits** for the codeword at right so that each row and column has an even parity sum.

   (b) What is the Hamming Distance of this codeword?

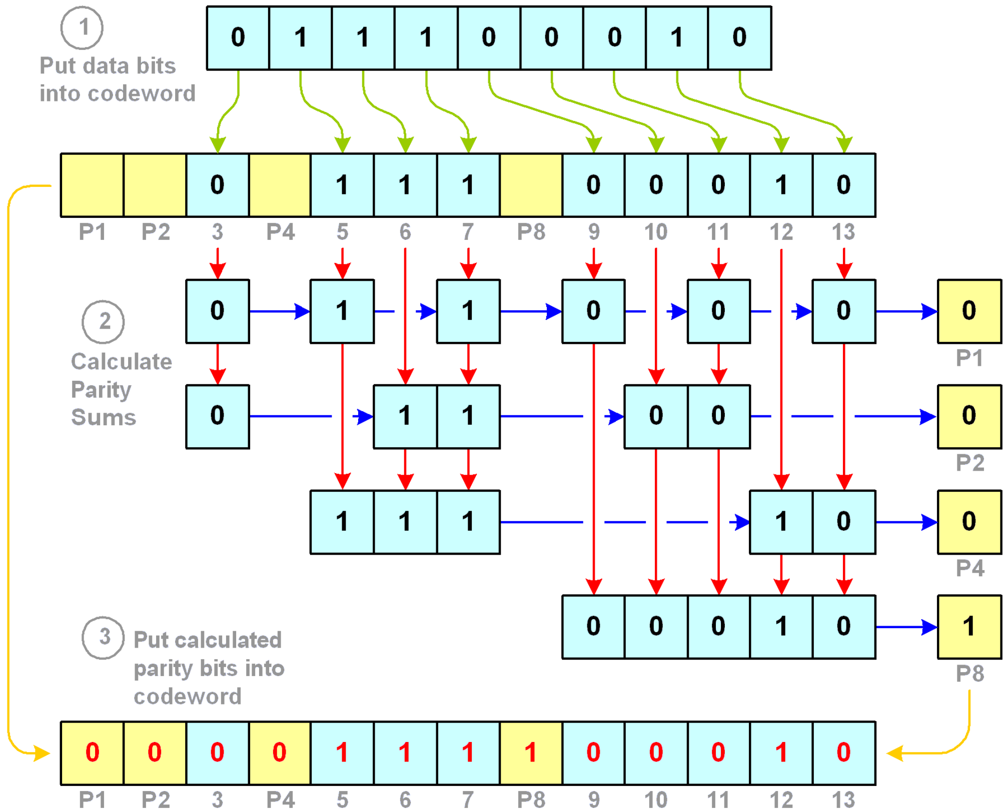
   **<u>3</u> (if a data bit changes, two parity bits must also change)**

   (c) What is the maximum number of error bits that this code could **correct**?

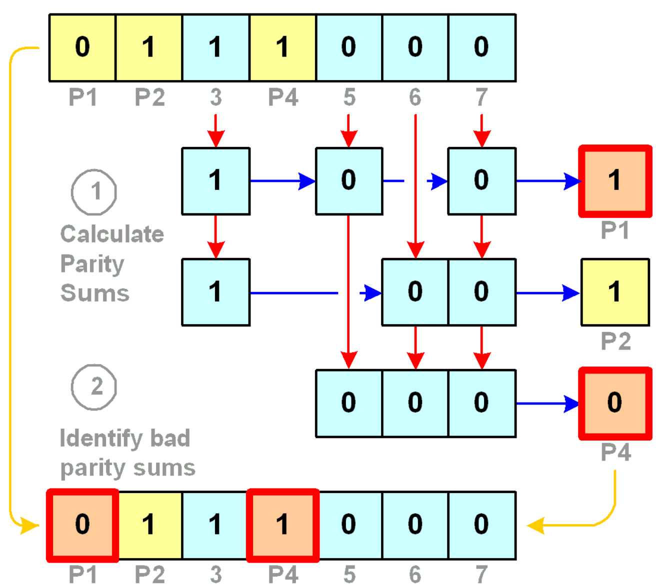



**<u>1</u> ($2^n$+1 = Hamming Dist to correct "n" bits, and $2^1$+1 = 3)**

4. **Create a valid Hamming codeword** for the 9 data bits shown in question 1. Assume that the 9 data bits are arranged as follows:




5. **Which bit is bad** in the following Hamming Codeword?



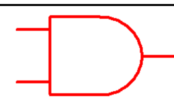
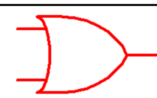
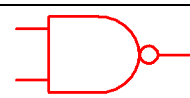
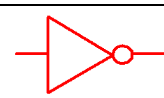
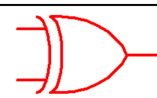

**Parity bits 1 and 4 are bad, therefore the bad bit is: 1+4 = bit 5**

6. **What is the average access time** for a system with a 3-level cache that has the following characteristics::

| Level | Access Time | Hit Rate | Miss Rate | % reads this level | |
|---|---|---|---|---|---|
| 1 | 0.25ns | 75% | 25% | 100% | 0.25ns x 1 = 0.25ns |
| 2 | 3ns | 95% | 5% | 25% x 100% = 25% | 3ns x 0.25 = 0.75ns |
| 3 | 40ns | 100% | 0% | 5% x 25% = 1.25% | 40ns x .0125 = 0.5ns |

0.25ns + 0.75ns + 0.5ns = 1.5ns average access time

7. Draw the **algebraic** and **logic diagramming symbols** for the following types of Boolean operations:

| Boolean Operator | Algebraic Symbol | Logic Diagramming Symbol |
|---|---|---|
| AND | $\bullet$ |  |
| OR | + |  |
| NAND | $\overline{\bullet}$ |  |
| NOT | $\overline{\quad}$ |  |
| XOR | $\oplus$ |  |

8. **Draw a truth table** to show the result of the following expression for all possible input combinations of A, B and C:

$\overline{A} + (B \cdot C)$

| A | B | C | $\overline{A}$ | $B \cdot C$ | $\overline{A} + (B \cdot C)$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

9. **Complete** the following Boolean **identities**:

$$A \cdot 0 = \boxed{0} \quad A \cdot 1 = \boxed{A} \quad A \cdot \overline{A} = \boxed{0}$$

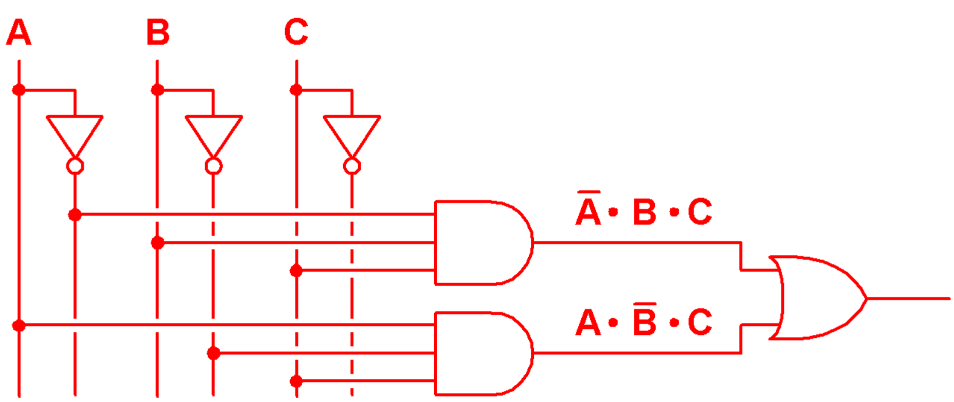
$$A + 0 = \boxed{A} \quad A + 1 = \boxed{1} \quad A + \overline{A} = \boxed{1}$$

10. Write a Boolean expresson in "**Sum of Products**" form for three inputs, A, B and C. The expression must produce a **TRUE** output whenever the **A and B inputs are different** from each other and the **C input is also TRUE**. You don't need to simplify the expression.

| A | B | C | Result | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | → $\overline{A} \cdot B \cdot C$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | → $A \cdot \overline{B} \cdot C$ |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 0 | |

$$\boxed{(\overline{A} \cdot B \cdot C) + (A \cdot \overline{B} \cdot C)}$$

11. **Draw a Boolean logic circuit** for your "Product of Sums" expression for the above question.




(Note – NOT gate from the C input is not really necessary since "Not-C" isn't used anywhere)