B38DF
Computer Architecture and Embedded Systems

Alexander Belyaev

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering

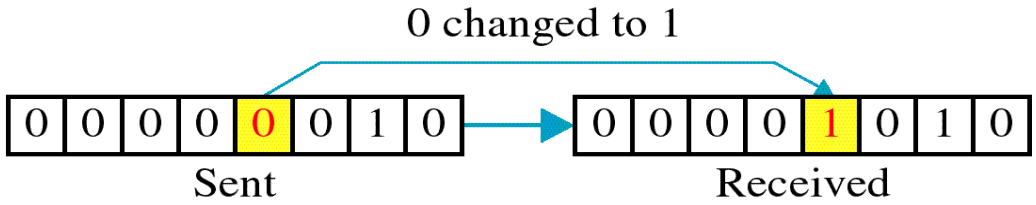
E-mail: a.belyaev@hw.ac.uk
Office: EM2.29

Based on the slides prepared by Dr. Mustafa Suphi Erden

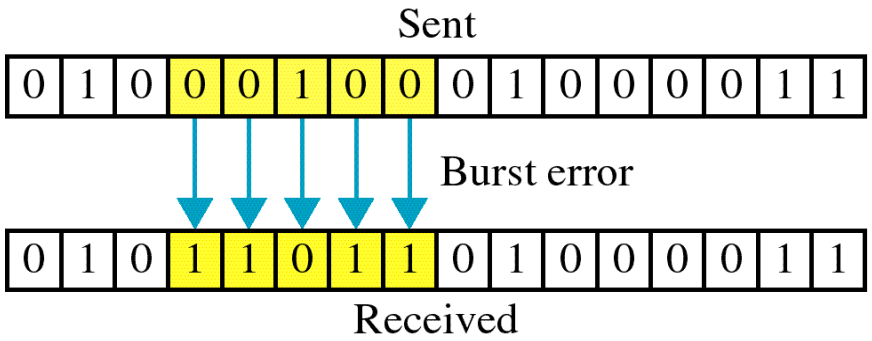
Error-Correcting Codes

- Computer memories can make errors occasionally due to voltage spikes in power line or other causes.
- Error-detecting or error-correcting codes take care of such errors through some extra bits added to the memory for error checking

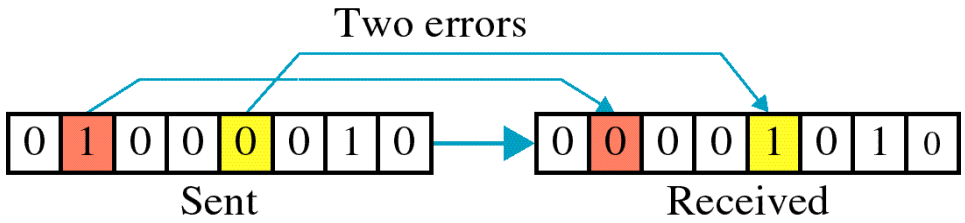
Single-bit error



Burst error

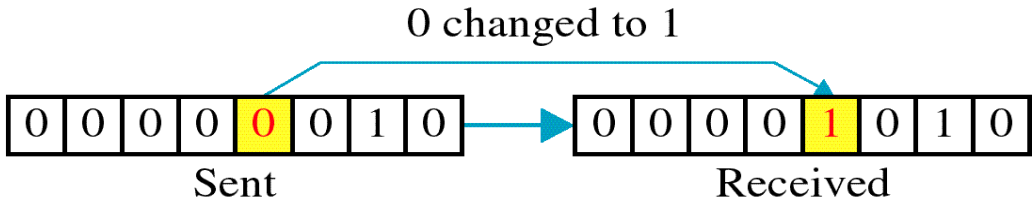


Two errors



Error-Correcting Codes

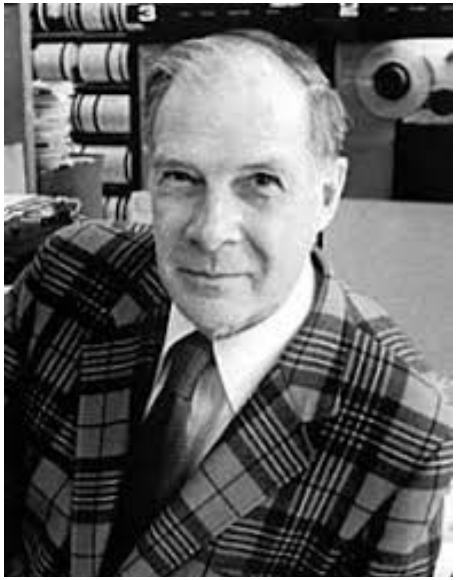
Single-bit error



In order to detect and/or correct errors we have to add some redundancy. The simplest way consists of repeating the original message: for example, instead of sending 1011, we send 1011**1011**. If there is exactly one error, the two halves won't be the same. Note that we cannot correct any errors.

Let us repeat everything 3 times: 1011 is encoded as 1011**10111011** (so 2 out of 3 are redundancy bits). We can now correct a single error.

Can we find a better approach?



Richard Hamming

Richard Wesley Hamming (1915 –1998) was an American mathematician whose work had many implications for computer engineering and telecommunications. His contributions include the **Hamming code** (which makes use of a Hamming matrix), the **Hamming window**, **Hamming numbers**, sphere-packing (or **Hamming bound**), and the **Hamming distance**.

... at Los Alamos ... we were designing atomic bombs. Shortly before the first field test (you realize that no small scale experiment can be done - either you have a critical mass or you do not), a man asked me to check some arithmetic he had done, and I agreed, thinking to fob it off on some subordinate. When I asked what it was, he said, **"It is the probability that the test bomb will ignite the whole atmosphere."** I decided I would check it myself! The next day when he came for the answers I remarked to him, "The arithmetic was apparently correct but I do not know about the formulas for the capture cross sections for oxygen and nitrogen - after all, there could be no experiments at the needed energy levels." He replied, like a physicist talking to a mathematician, that he wanted me to check the arithmetic not the physics, and left. I said to myself, "What have you done, Hamming, you are involved in risking all of life that is known in the Universe, and you do not know much of an essential part?" I was pacing up and down the corridor when a friend asked me what was bothering me. I told him. His reply was, "Never mind, Hamming, **no one will ever blame you**."

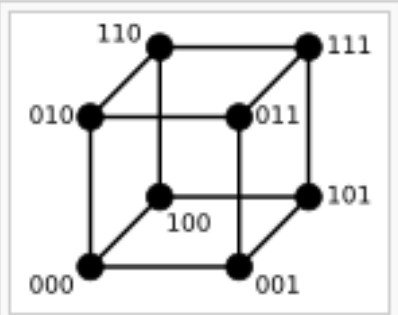
Codeword, Hamming distance

Codeword:

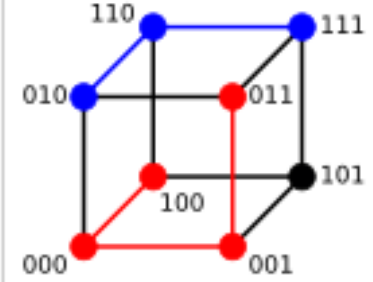
- A memory word consists of *m* data bits to which we add *r* check bits, making the total *n* = *m*+ *r*. The *n*-bit unit containing *m* data and *r* check bits is referred to as an *n*-bit codeword

The Hamming distance is the number of bits that have to be changed to get from one bit pattern to another.

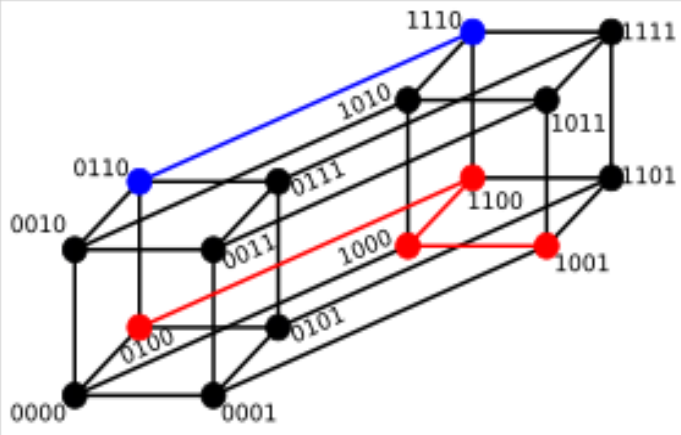
Examples: 10010101 & 1001**10**01 have a Hamming distance of 2.



3-bit binary cube for finding hamming distance



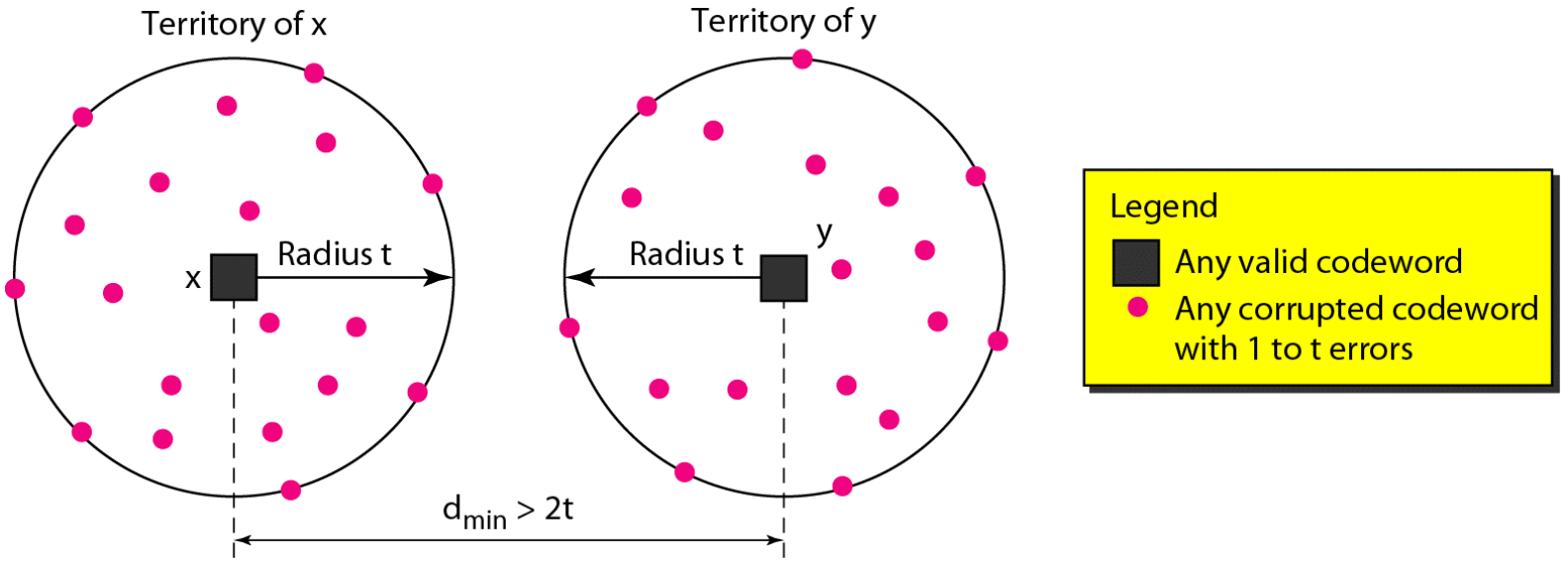
Two example distances: 100->011 has distance 3 (red path); 010->111 has distance 2 (blue path)



Two example distances: 0100->1001 has distance 3 (red path); 0110->1110 has distance 1 (blue path)

Hamming distance

- For any coding whose members (codewords) have a Hamming distance of two, any one bit error can be detected. Why?
- For any coding whose members have a Hamming distance of three, any one bit error can be detected and corrected. Why?
- And any two bit error can be detected. Why?



To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance must be $d_{min} = 2t + 1$.

Hamming distance: An example

Suppose we have the following code:

0 0 0 0 0, 0 1 0 1 1, 1 0 1 1 0, 1 1 1 0 1

Let us determine d_{min}. By examining all possible pairs of code words, we discover that the minimum Hamming distance $d_{min} = 3$. Thus, this code can detect up to two errors and correct one single bit error.

How is correction handled? Suppose we read the invalid code word 10000. There must be at least one error because this does not match any of the valid code words. We now determine the Hamming distance between the observed code word and each legal code word: it differs in 1 bit from the first code word, 4 from the second, 2 from the third, and 3 from the last, resulting in a difference vector of [1,4,2,3]. To make the correction using this code, we automatically correct to the legal code word closest to the observed word.

Here we assume that the minimum number of possible errors is 1. Otherwise it is possible, for example, that the original code word was supposed to be 10110 and was changed to 10000 when two errors occurred.

Hamming Distance – Parity Bit

- Given any two codewords, the number of bit positions in which two codewords differ is called the Hamming distance

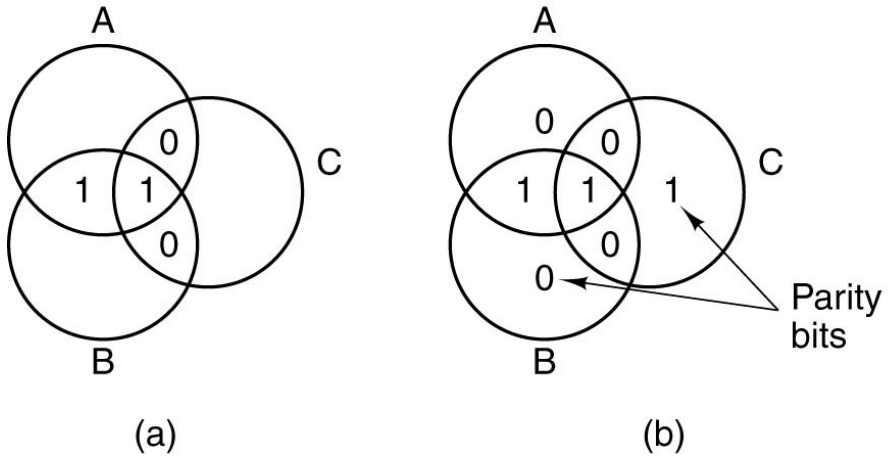
  10001001 10110001 → Hamming distance 3 apart

- A **parity bit**, or **check bit** is a bit added to the end of a string of binary code that indicates whether the number of bits in the string with the value one is even or odd. Parity bits are used as the simplest form of error detecting code. A single parity bit can detect single errors.

- 10111000 10000001 → no errors (assume even parity)
- 10110000 10010001 → there is an error in both codewords!

Hamming's Algorithm – Venn Diagram Explanation

- A graphical representation that illustrates the idea of a single error correcting code for 4-bit words
- **Let us encode the 4-bit memory word 1100 in the regions *AB, ABC, AC, and BC.***



(a) (b)

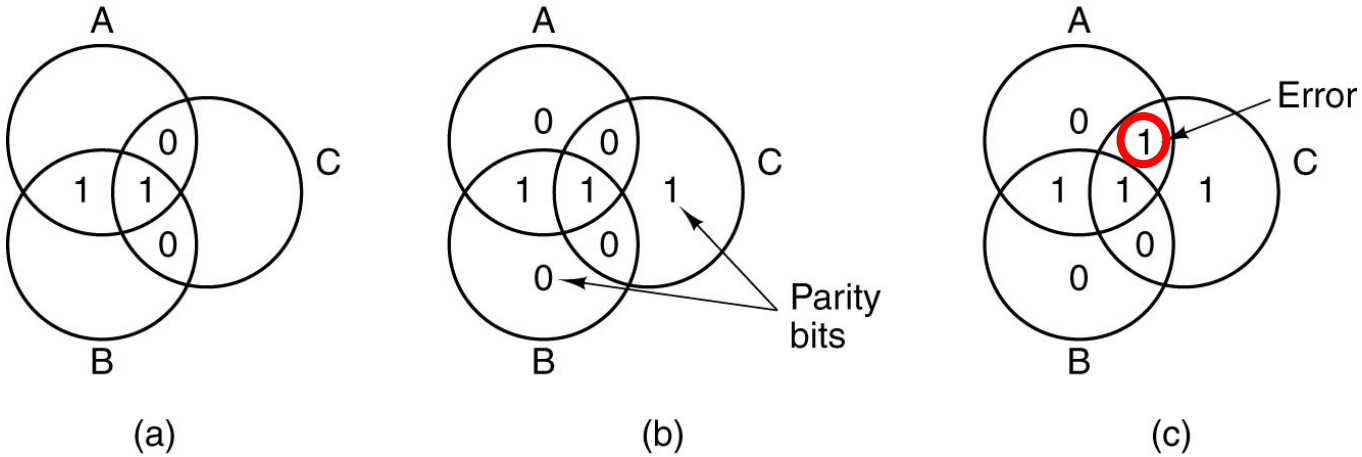
(a) Encoding of 1100 (b) Even parity added

- Add a parity bit to each of the three empty regions to produce even parity, to make the sum of the bits in each circle an even number:

→ A code word with 4 data bits and 3 parity bits

Venn Diagram Explanation – Error

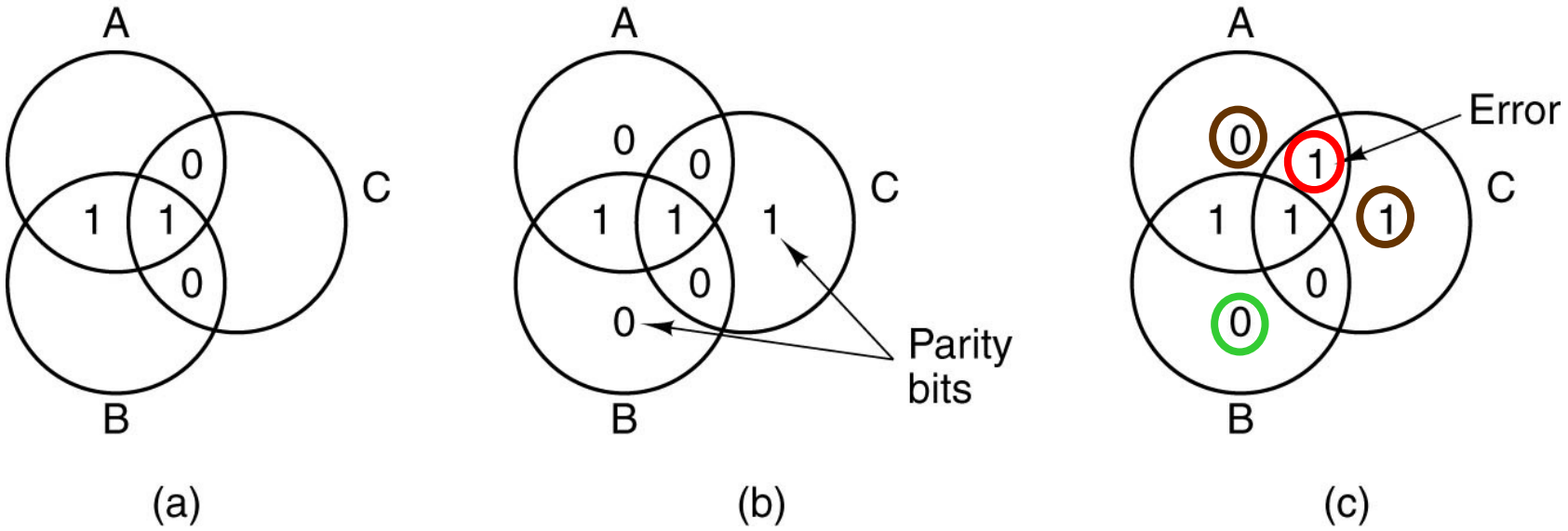
- Assume one bit in AC region goes bad!



(a) Encoding of 1100 (b) Even parity added (c) Error in AC

- Can the computer detect this error and identify the erroneous bit?
- **Yes!** The computer can see that circle B has the correct parity (even), but A and C have the wrong parity (odd). → This corresponds to the region AC!
- Computer can correct the error by changing the 1 in AC region into a 0.

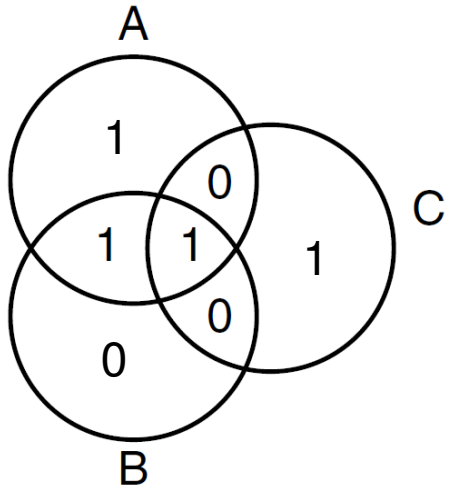
Venn Diagram Explanation – Error



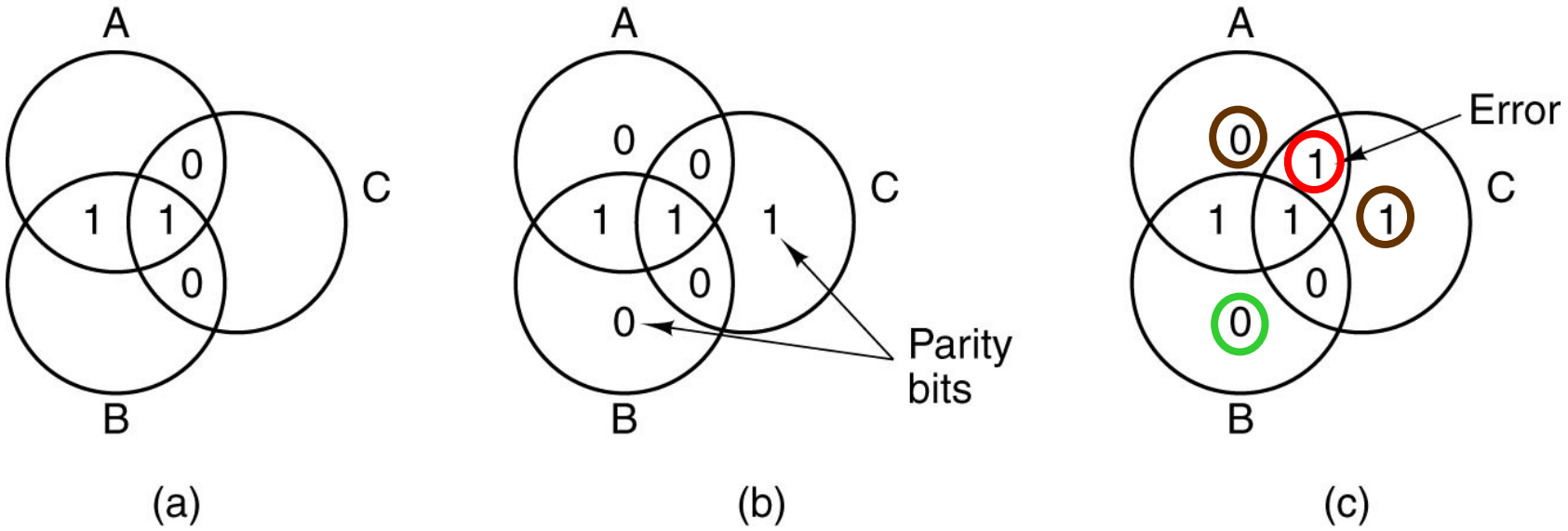
(a) Encoding of 1100 (b) Even parity added (c) Error in AC

Another example:

Where is an error?



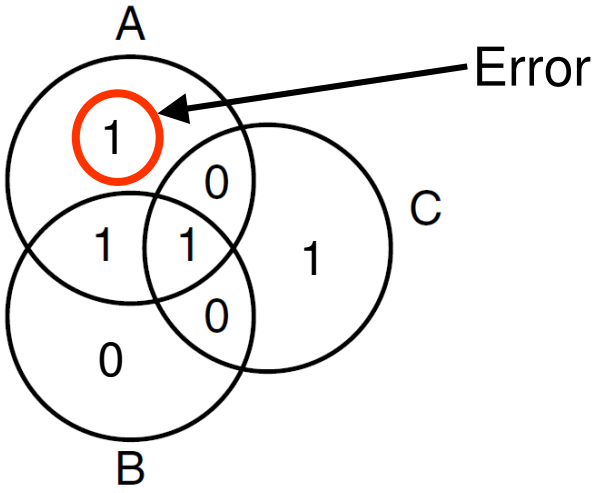
Venn Diagram Explanation – Error



(a) Encoding of 1100 (b) Even parity added (c) Error in AC

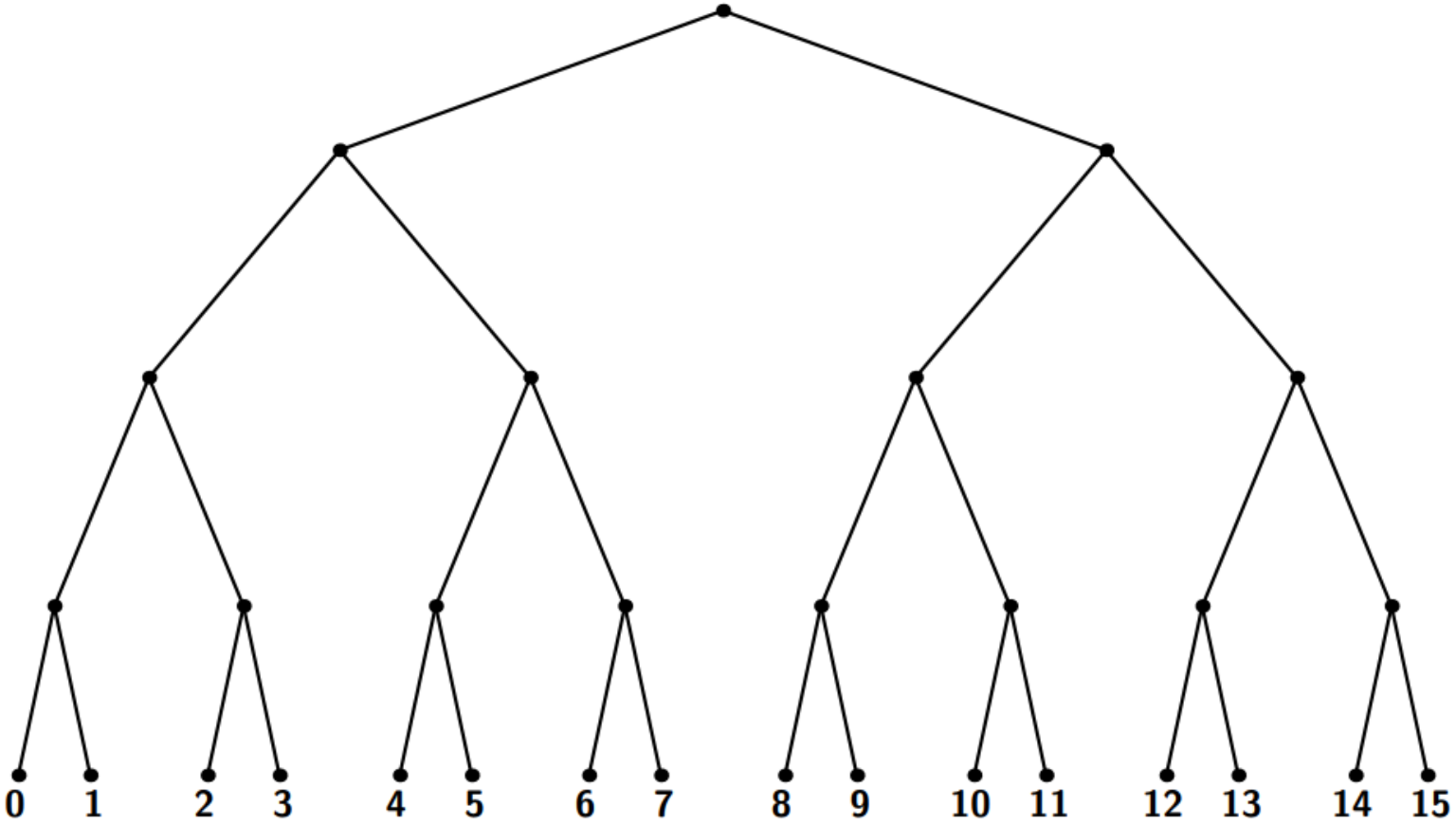
Another example:

Where is an error?



A Number Guessing Game

Ask a friend to thinks of a number between 1 and 15. How many YES/NO questions do you need to ask to find out the secret number?



The Liar Game

Ask a friend to think of a number between 0 and 15.
How many YES/NO questions do you need to ask,
if your friend is permitted to lie at most once?

It is not compulsory to lie.

The Liar Game

A possible set of questions:

Q1.	Is the number	8, 9, 10, 11, 12, 13, 14 or 15 ?
Q2.	Is the number	4, 5, 6, 7, 12, 13, 14 or 15 ?
Q3.	Is the number	2, 3, 6, 7, 10, 11, 14 or 15 ?
Q4.	Is the number	1, 3, 5, 7, 9, 11, 13 or 15 ?
Q5.	Is the number	1, 2, 5, 6, 8, 11, 12 or 15 ?
Q6.	Is the number	1, 3, 4, 6, 8, 10, 13 or 15 ?
Q7.	Is the number	2, 3, 4, 5, 8, 9, 14 or 15 ?

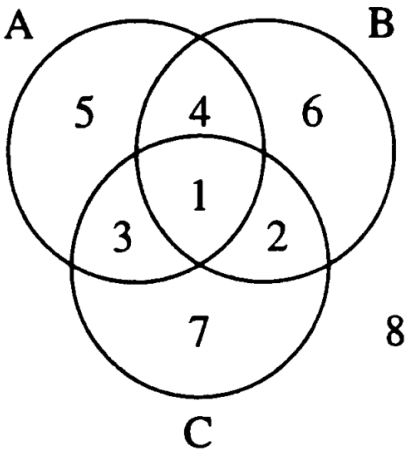
The Liar Game

A possible set of questions:

Q1. Is the number 8, 9, 10, 11, 12, 13, 14 or 15 ?
Q2. Is the number 4, 5, 6, 7, 12, 13, 14 or 15 ?
Q3. Is the number 2, 3, 6, 7, 10, 11, 14 or 15 ?
Q4. Is the number 1, 3, 5, 7, 9, 11, 13 or 15 ?
Q5. Is the number 1, 2, 5, 6, 8, 11, 12 or 15 ?
Q6. Is the number 1, 3, 4, 6, 8, 10, 13 or 15 ?
Q7. Is the number 2, 3, 4, 5, 8, 9, 14 or 15 ?

The first four questions are obvious:

0	1	2	3	4	5	6	7
0000	0001	0010	0011	0100	0101	0110	0111
8	9	10	11	12	13	14	15
1000	1001	1010	1011	1100	1101	1110	1111



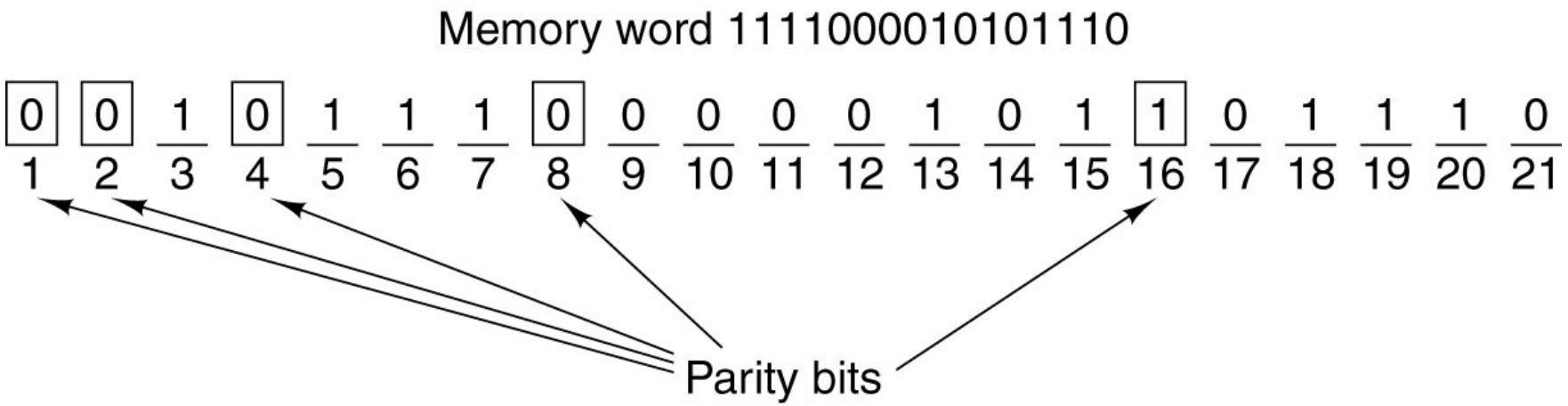
The next three questions correspond to the three parity bits:

0	1	2	3	4	5	6	7
0000000	0001110	0010101	0011011	0100011	0101101	0110110	0111000
8	9	10	11	12	13	14	15
1000111	1001001	1010010	1011100	1100100	1101010	1110001	1111111

Hamming's Algorithm – Any size word

- Hamming Code

r parity bits added to an *m*-bit word, forming a new word of length *m*+*r*

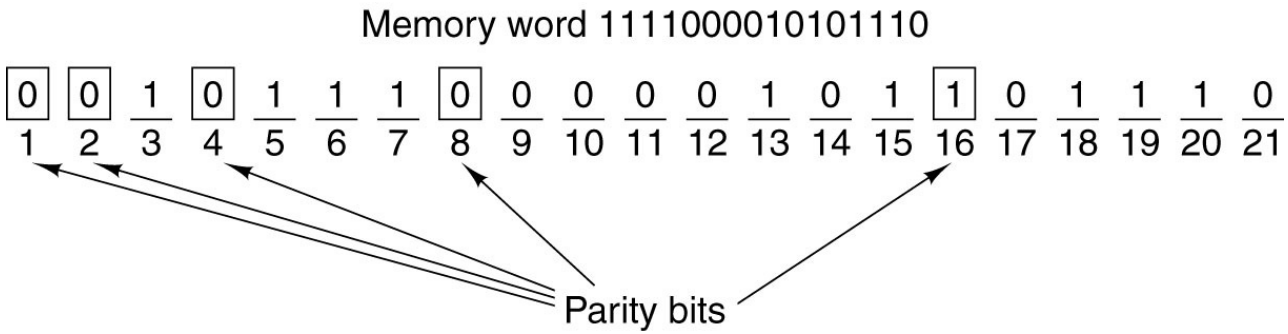


Construction of the Hamming code for the memory word 1111000010101110 by adding 5 check bits to the 16 data bits.

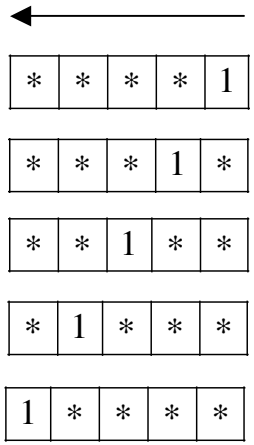
  - Number the bits from left-to-right starting from 1.
  - All bits whose bit number is a power of 2 are parity bits

- For example with a 16-bit word, 5 parity bits are added: bits 1, 2, 4, 8, 16 are parity bits, all the rest are data bits. In all the memory word has 21 bits (16 data, 5 parity).

Hamming's Algorithm – Any size word

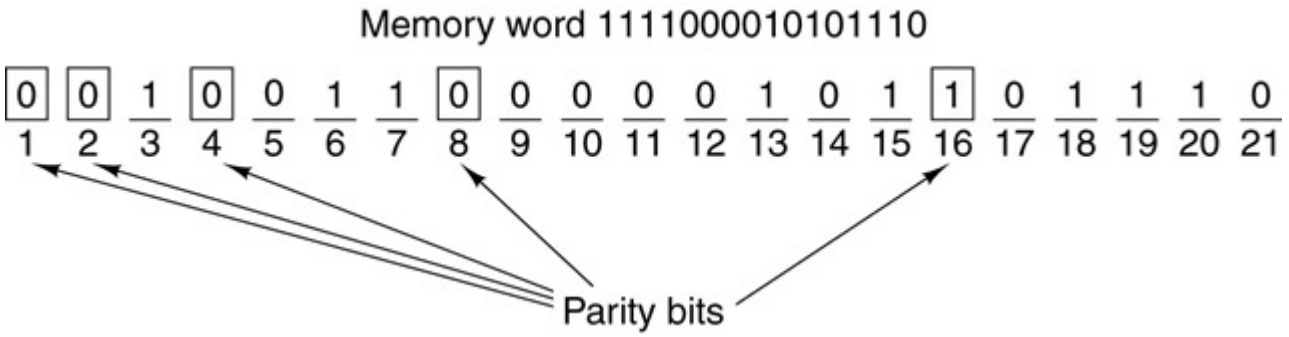


- Each parity bit checks specific bit positions:
  - Bit **1** checks bits 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21
  - Bit **2** checks bits 2, 3, 6, 7, 10, 11, 14, 15, 18, 19
  - Bit **4** checks bits 4, 5, 6, 7, 12, 13, 14, 15, 20, 21
  - Bit **8** checks bits 8, 9, 10, 11, 12, 13, 14, 15
  - Bit **16** checks bits 16, 17, 18, 19, 20, 21

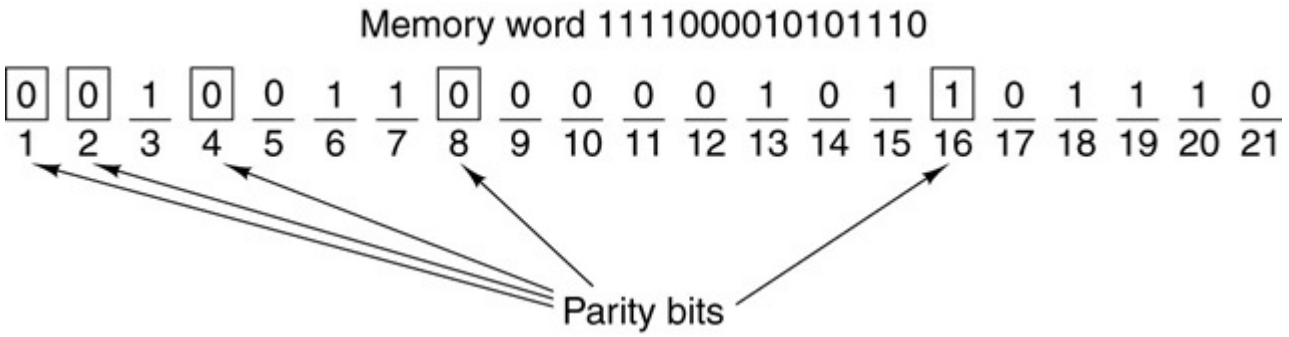


- **Rule:** Bit *b* is checked by those bits *b1, b2, …, bj such that b1+b2+…+bj = b*
- e.g. Bit 5 is checked by bits 1 and 4 because 1+4 = 5; bit 6 is checked by bits 2 and 4 because 2 + 4 = 6.
- The parity bit is set so that the total number of 1s in the checked positions is even.

Hamming's Algorithm – Exercise

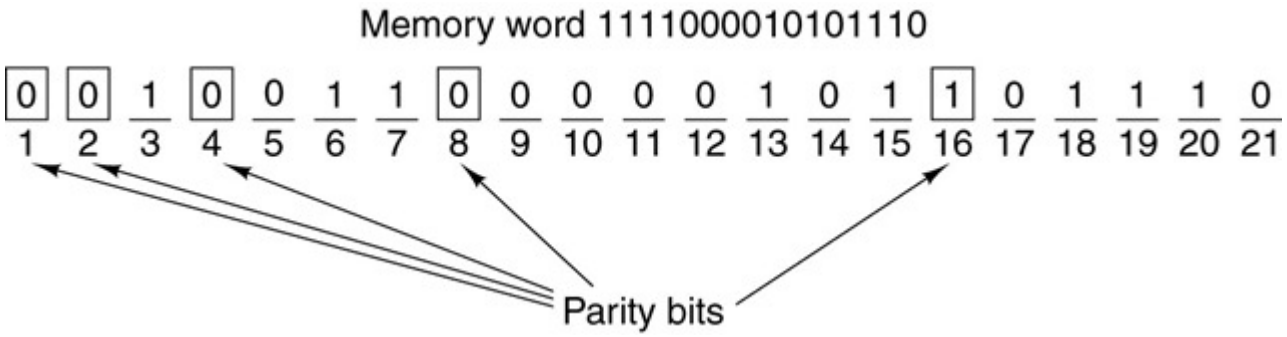


Hamming's Algorithm – Exercise



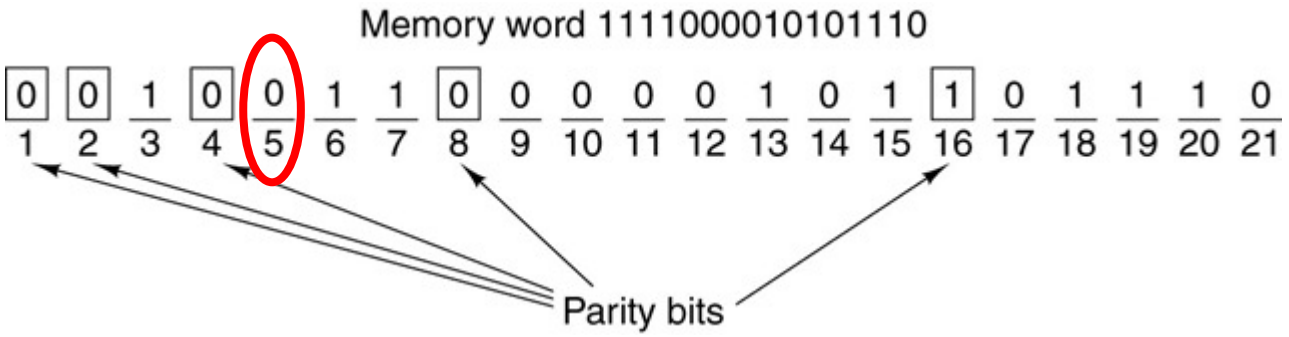
- Where is the error?
  - Parity bit **1** incorrect (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contain five 1s)
  - Parity bit **2** correct (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain six 1s)
  - Parity bit **4** incorrect (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contains five 1s)
  - Parity bit **8** correct (8, 9, 10, 11, 12, 13, 14, 15 contain two 1s)
  - Parity bit **16** correct (16, 17, 18, 19, 20, 21 contain four 1s)

Hamming's Algorithm – Exercise



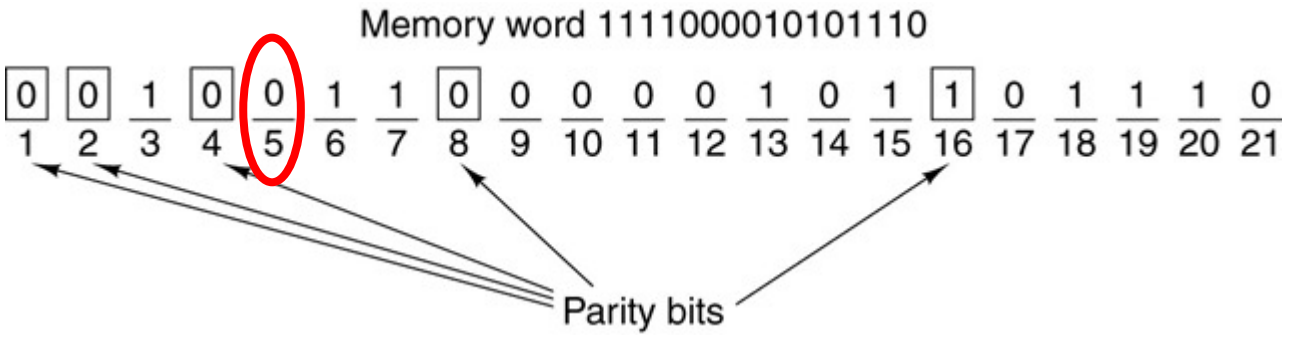
- Where is the error?
  - Parity bit **1** incorrect (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contain five 1s)
  - Parity bit **2** correct (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain six 1s)
  - Parity bit **4** incorrect (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contains five 1s)
  - Parity bit **8** correct (8, 9, 10, 11, 12, 13, 14, 15 contain two 1s)
  - Parity bit **16** correct (16, 17, 18, 19, 20, 21 contain four 1s)

Hamming's Algorithm – Exercise



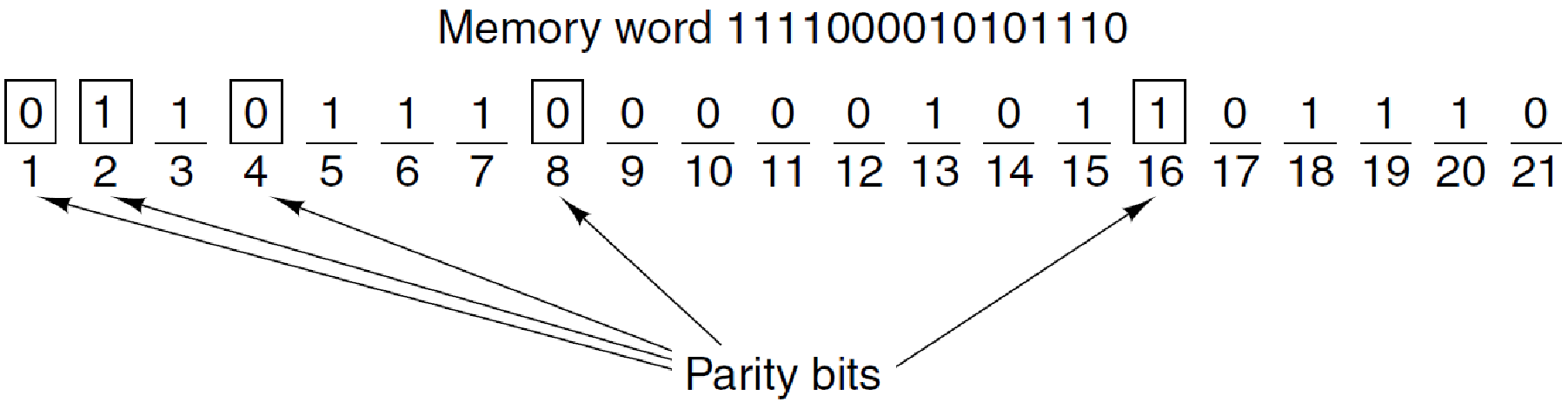
- Where is the error?
  - Parity bit **1** incorrect (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contain five 1s)
  - Parity bit **2** correct (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain six 1s)
  - Parity bit **4** incorrect (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contains five 1s)
  - Parity bit **8** correct (8, 9, 10, 11, 12, 13, 14, 15 contain two 1s)
  - Parity bit **16** correct (16, 17, 18, 19, 20, 21 contain four 1s)

Hamming's Algorithm – Exercise

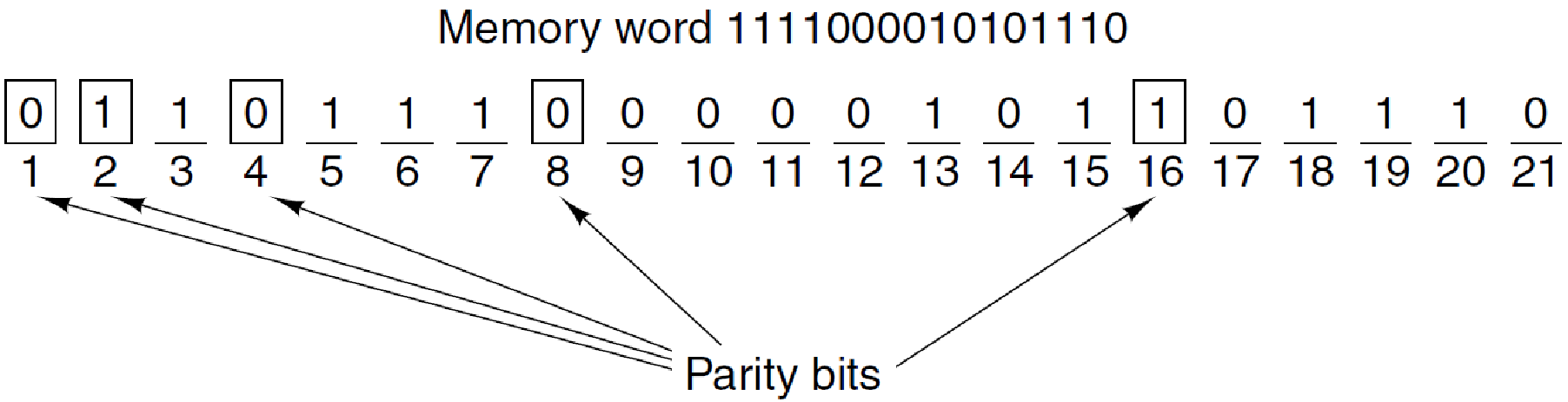


- Where is the error?
  - Parity bit **1** incorrect (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contain five 1s)
  - Parity bit **2** correct (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain six 1s)
  - Parity bit **4** incorrect (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contains five 1s)
  - Parity bit **8** correct (8, 9, 10, 11, 12, 13, 14, 15 contain two 1s)
  - Parity bit **16** correct (16, 17, 18, 19, 20, 21 contain four 1s)
- **Solution:** Add up all the incorrect parity bits. The resulting sum is the position of the incorrect bit! (**1** + **4** = 5)
- **Answer:** Bit 5 → Bit 5 should be corrected by the computer from 0 to 1.

Hamming's Algorithm – Another Exercise

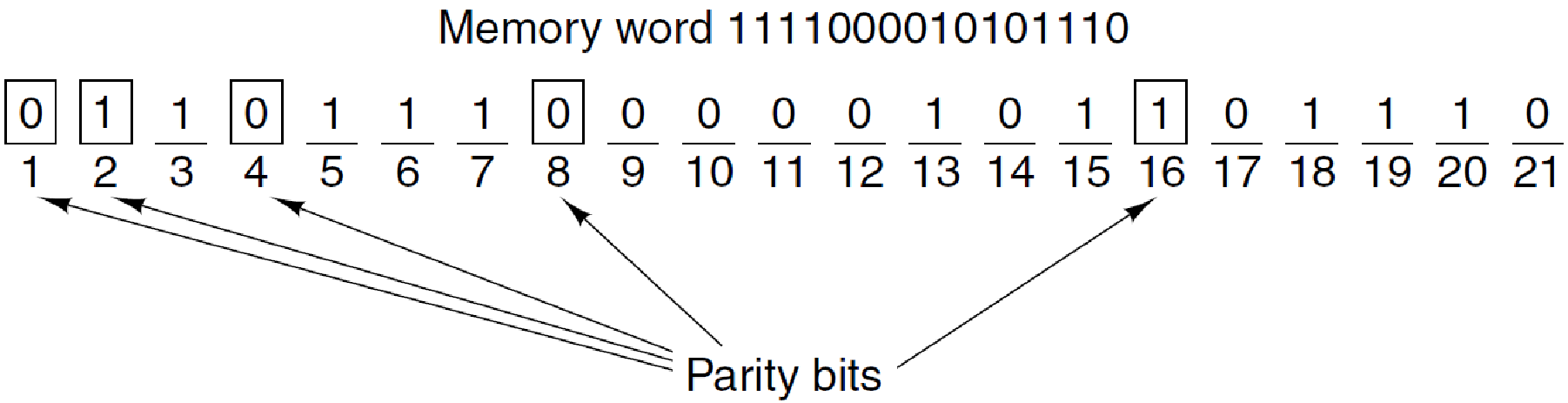


Hamming's Algorithm – Another Exercise



- Where is the error?
  - Parity bit **1** correct (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contain six 1s)
  - Parity bit **2** incorrect (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain seven 1s)
  - Parity bit **4** correct (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contains five 1s)
  - Parity bit **8** correct (8, 9, 10, 11, 12, 13, 14, 15 contain two 1s)
  - Parity bit **16** correct (16, 17, 18, 19, 20, 21 contain four 1s)

Hamming's Algorithm – Another Exercise



- Where is the error?
  - Parity bit **1** correct (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contain six 1s)
  - Parity bit **2** incorrect (2, ~~3, 6, 7, 10, 11, 14, 15, 18, 19~~ contain seven 1s)
  - Parity bit **4** correct (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contains five 1s)
  - Parity bit **8** correct (8, 9, 10, 11, 12, 13, 14, 15 contain two 1s)
  - Parity bit **16** correct (16, 17, 18, 19, 20, 21 contain four 1s)
- **Solution:** Add up all the incorrect parity bits. The resulting sum is the position of the incorrect bit! (**2** = 2)