B38DF
Computer Architecture and Embedded Systems

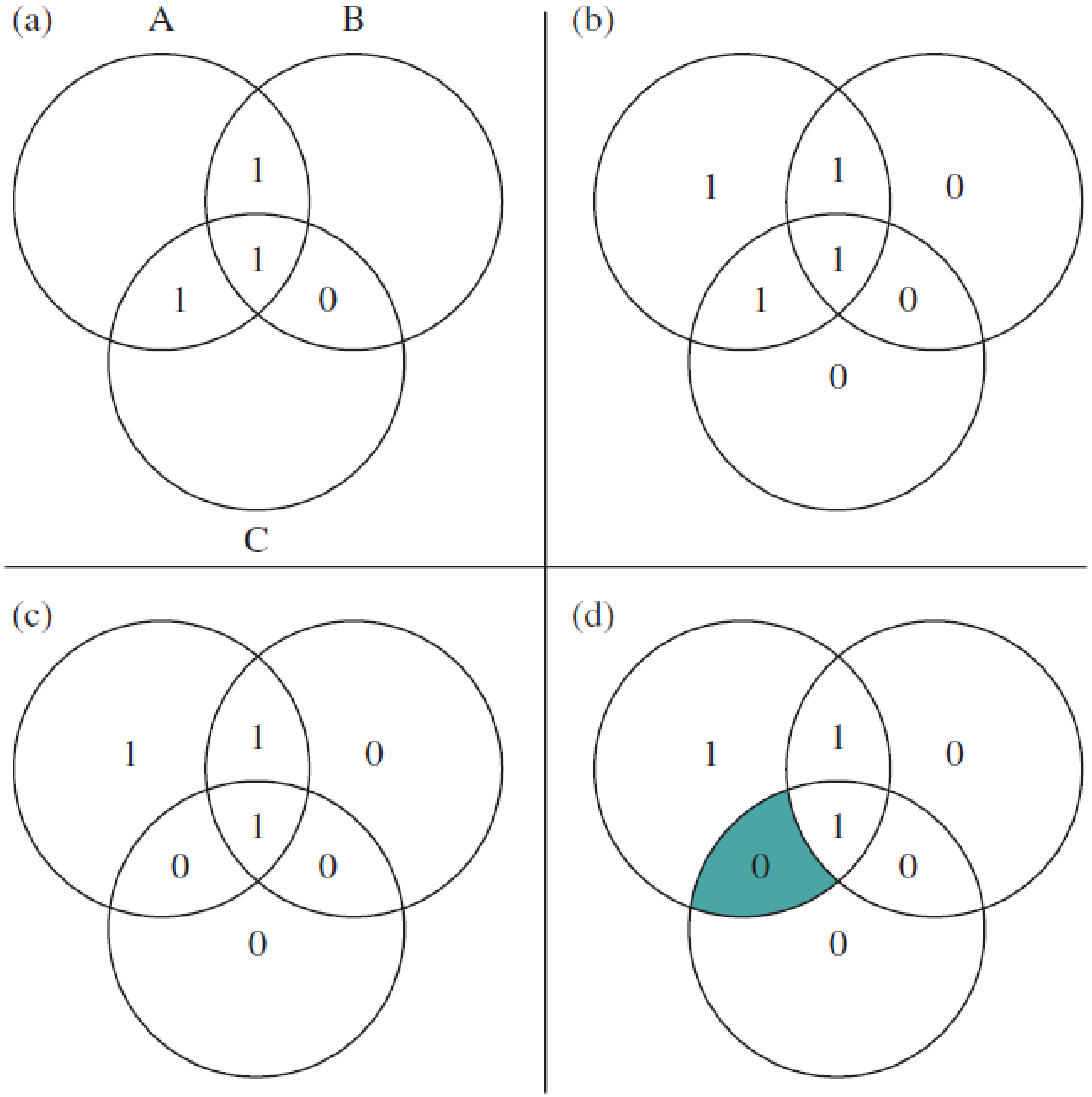
Alexander Belyaev

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

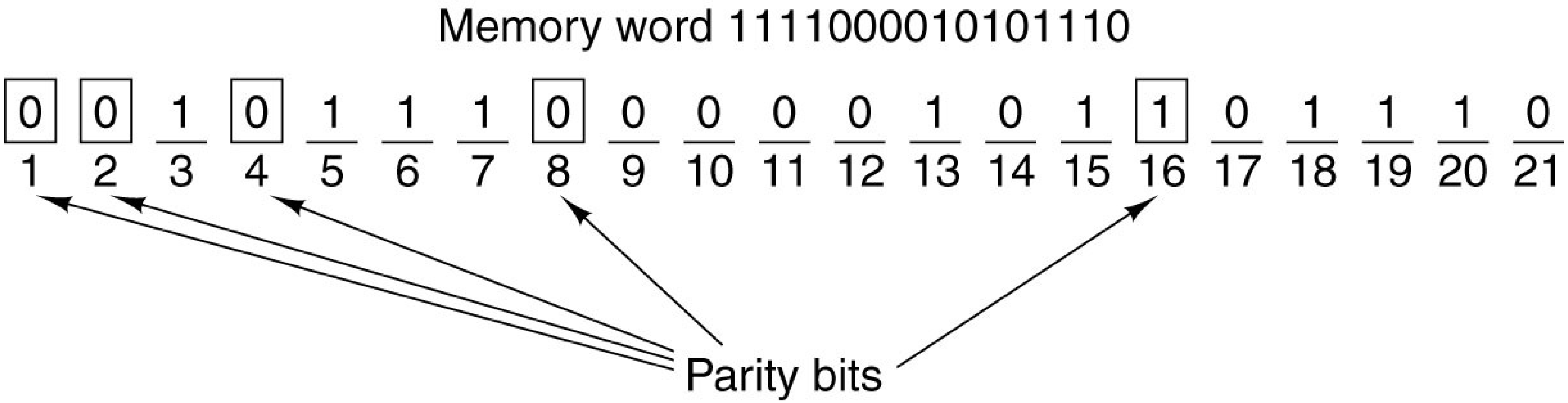
Hamming’s Algorithm – Single error correction



Hamming's Algorithm – Single error correction for any size word

- <u>Hamming Code</u>

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

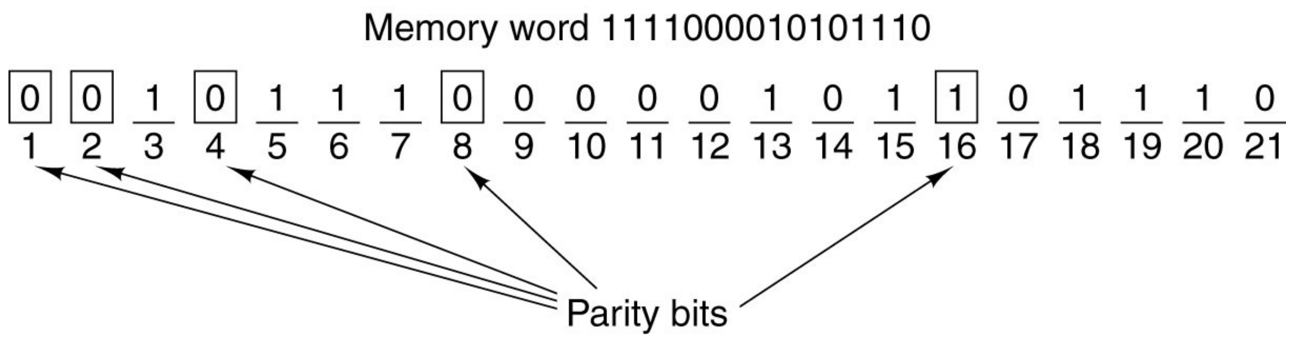


Construction of the Hamming code for the memory word 1111000010101110 by adding 5 check bits to the 16 data bits.

  - ➢ 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 – Single error correction for any size word

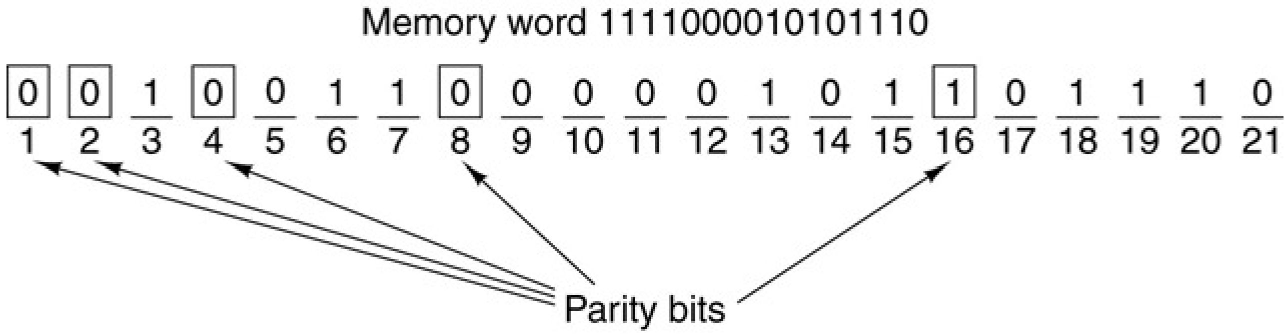


- Each parity bit checks specific bit positions:

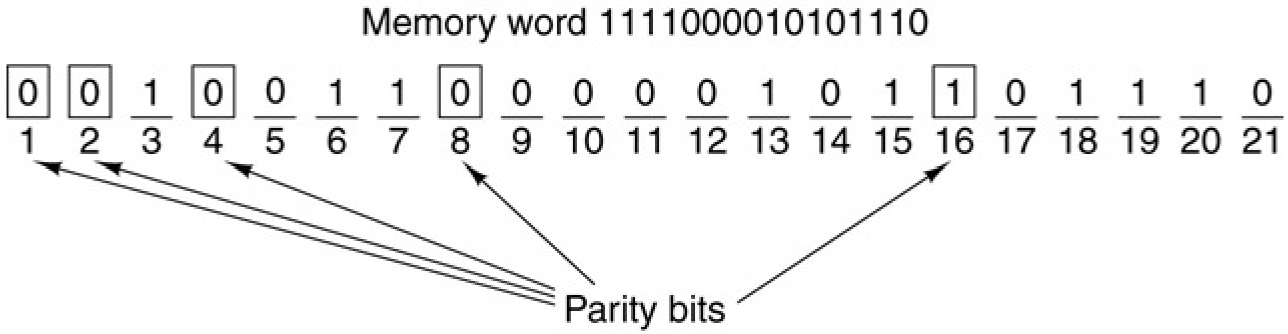
➢ Bit **1** checks bits	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21	* * * * 1
➢ Bit **2** checks bits	2, 3, 6, 7, 10, 11, 14, 15, 18, 19	* * * 1 *
➢ Bit **4** checks bits	4, 5, 6, 7, 12, 13, 14, 15, 20, 21	* * 1 * *
➢ Bit **8** checks bits	8, 9, 10, 11, 12, 13, 14, 15	* 1 * * *
➢ Bit **16** checks bits	16, 17, 18, 19, 20, 21	1 * * * *

- **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

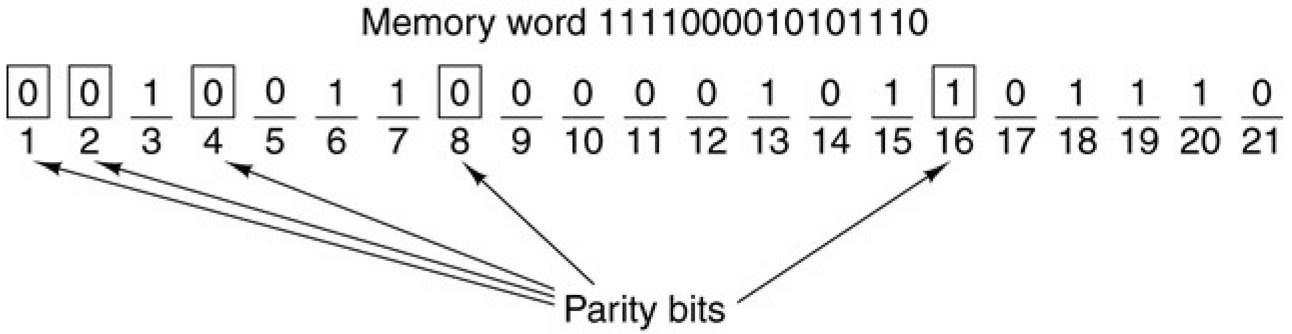


Hamming's Algorithm – Exercise



- 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)

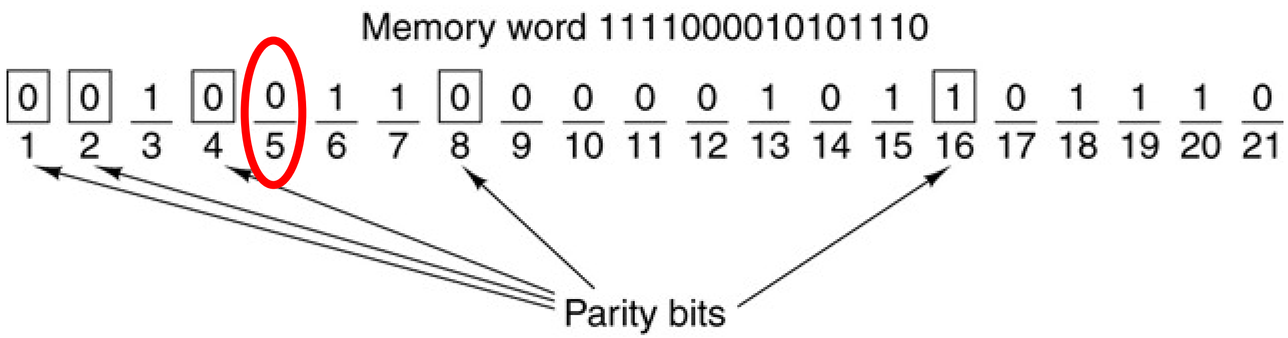
Hamming's Algorithm – Exercise



- 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)

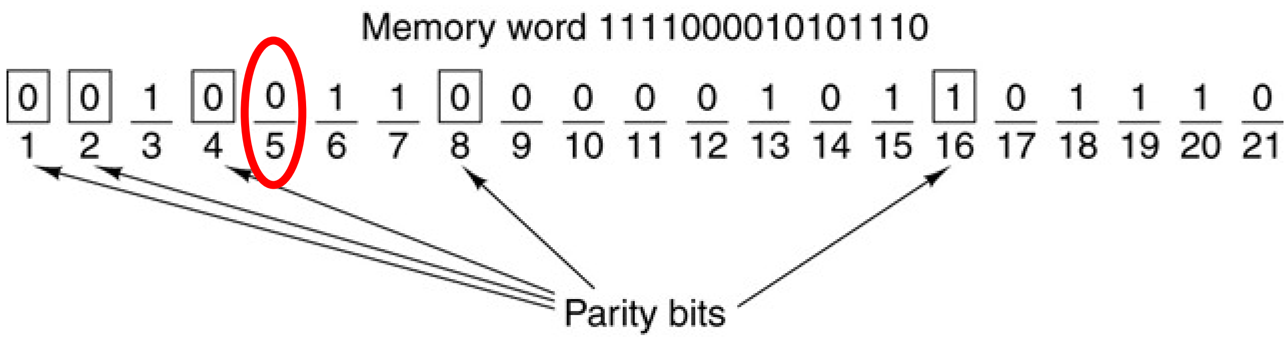
Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.

Hamming's Algorithm – Exercise



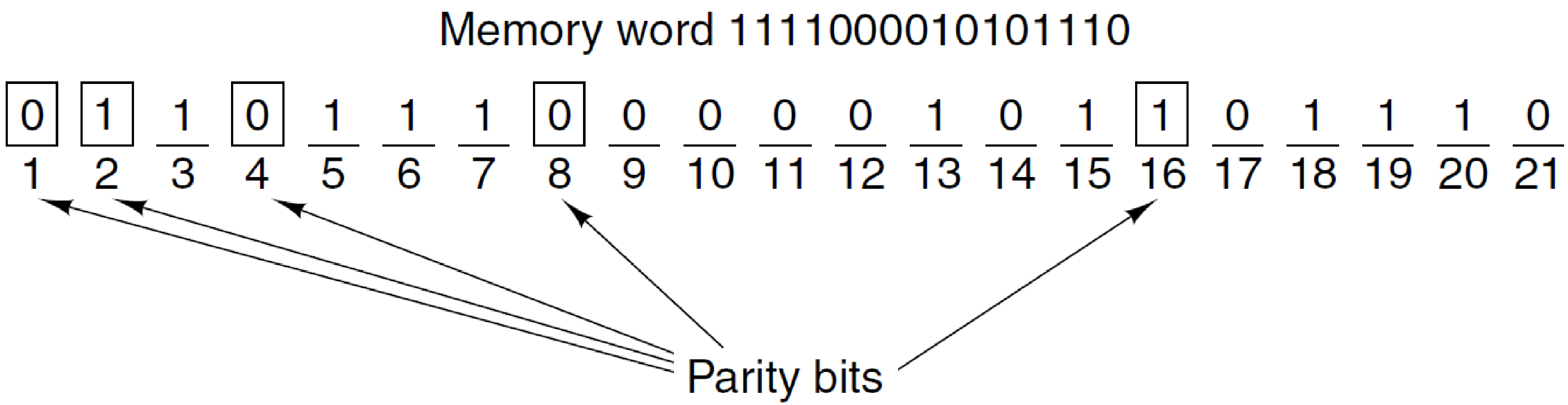
- Where is the error?
  - Parity bit **1** incorrect (1, 3, 5, ~~7~~, 9, 11, ~~13~~, ~~15~~, 17, 19, ~~21~~ contain five 1s)
  - Parity bit **2** correct (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain six 1s)
  - Parity bit **4** incorrect (4, 5, 6, ~~7~~, 12, ~~13~~, 14, ~~15~~, 20, ~~21~~ contains five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



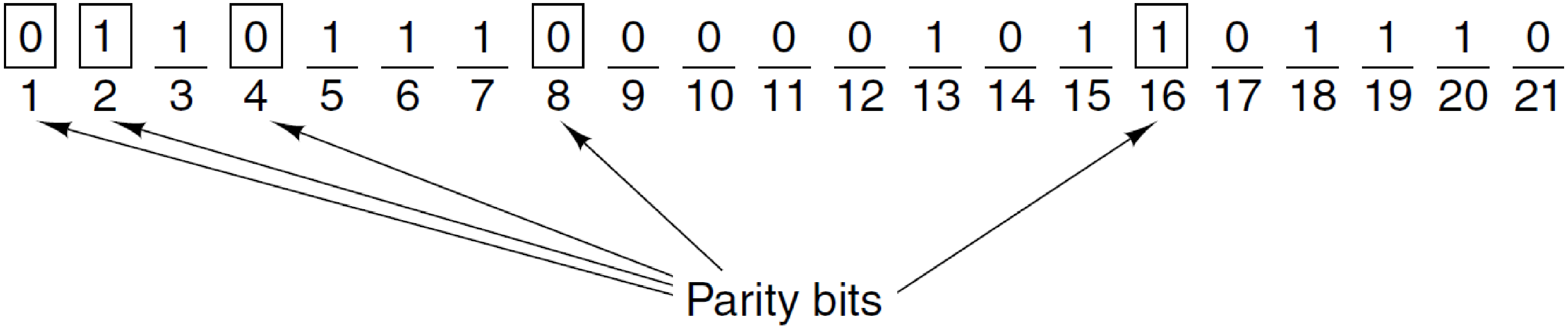
- Where is the error?
  - ➢ Parity bit **1** incorrect (1, 3, 5, ~~7~~, 9, 11, ~~13~~, ~~15~~, 17, 19, ~~21~~ contain five 1s)
  - ➢ Parity bit **2** correct (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contain six 1s)
  - ➢ Parity bit **4** incorrect (4, 5, 6, ~~7~~, 12, ~~13~~, 14, ~~15~~, 20, ~~21~~ contains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

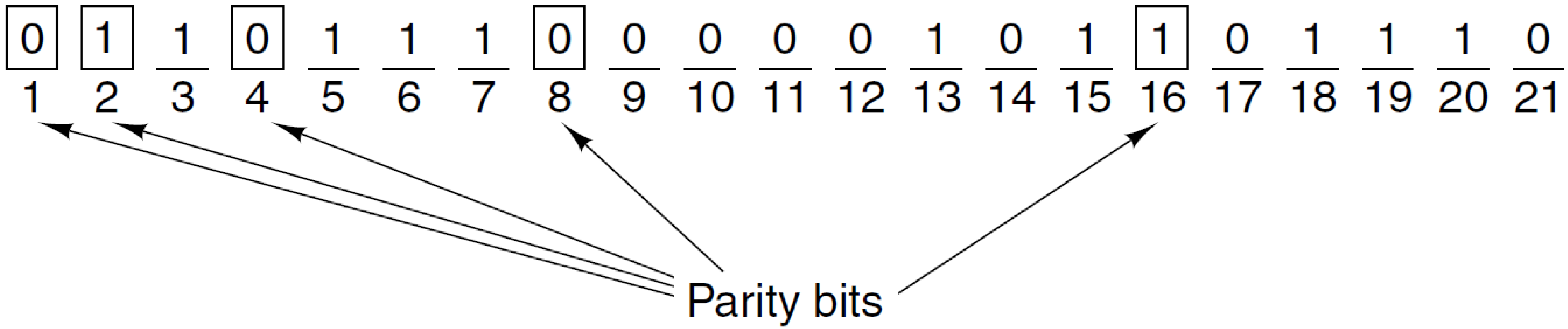
Memory word 1111000010101110



- 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

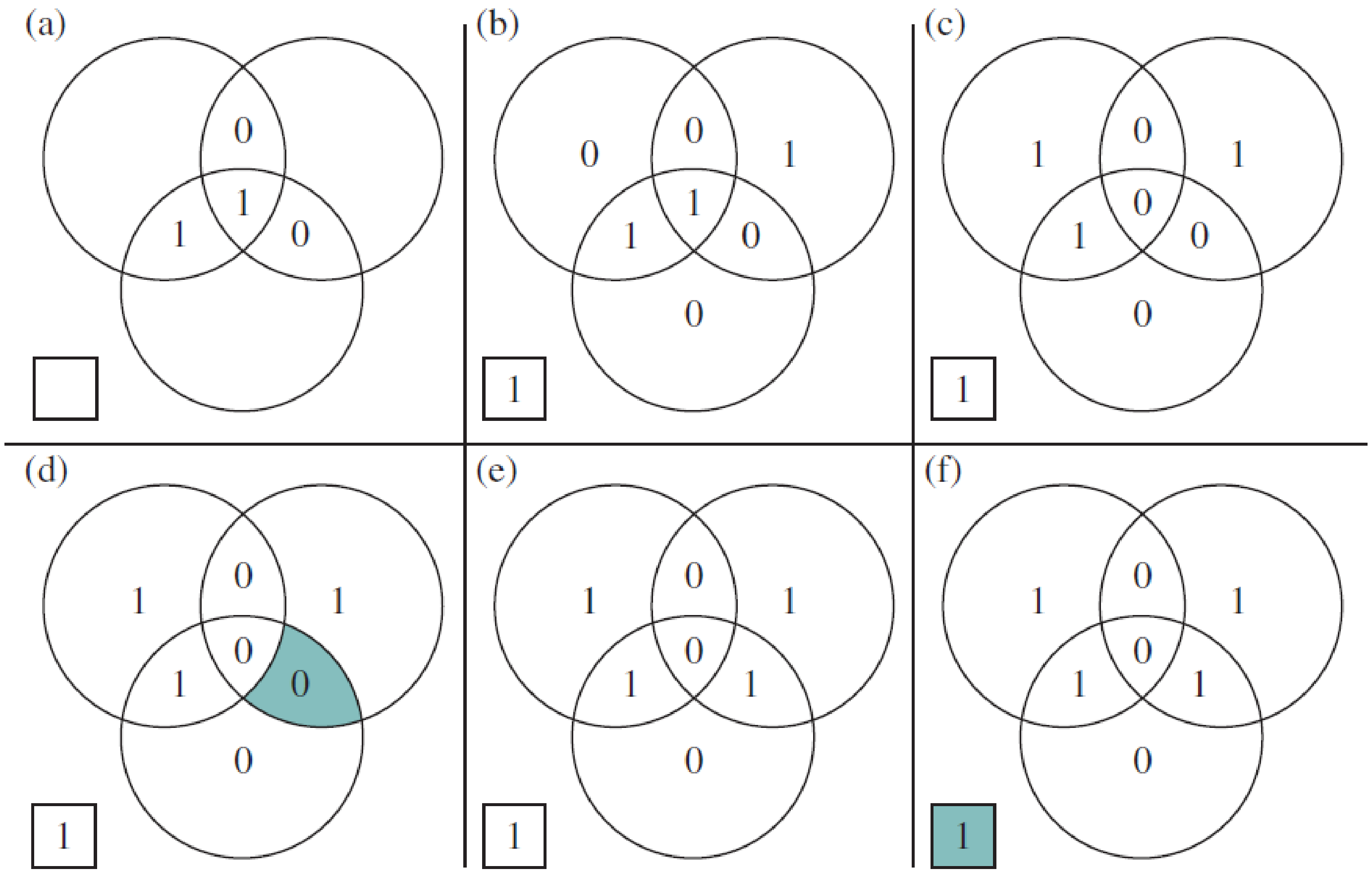
Memory word 1111000010101110



- 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)

Hamming's Algorithm: Single Error Correction (SEC) + Double Error Detection (DED) by adding an extra bit

Word **With Even Parity** **With Two Errors**



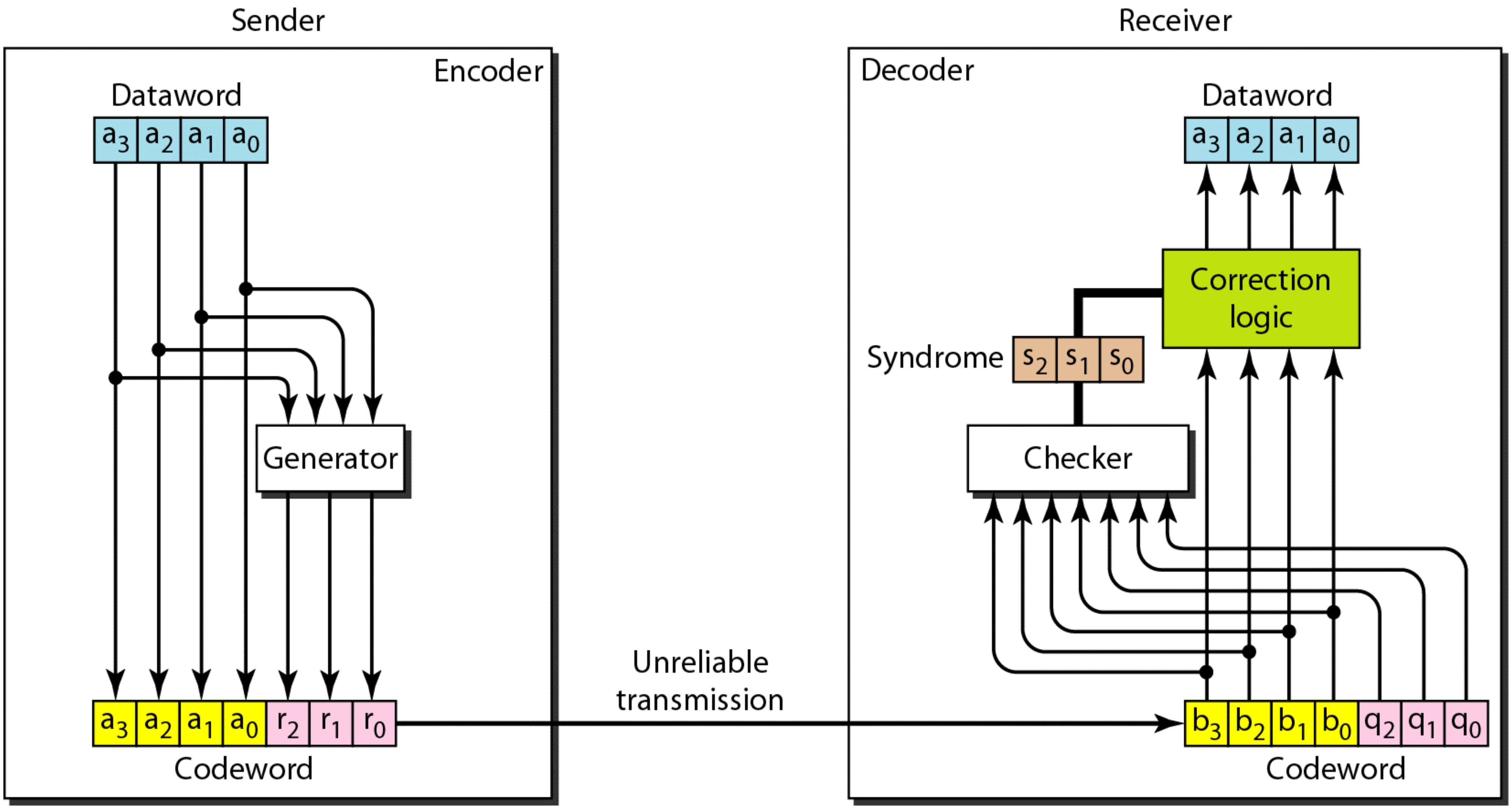
SEC Attempt **IS SEC Correct?** **Extra Bit Confirms DE**

Hamming’s Algorithm: SEC-DED

A single-error-correcting, double-error-detecting (SEC-DED) code is typically used for semiconductor memory.

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	**Check Bits**	**% Increase**	**Check Bits**	**% Increase**
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

Hamming's Algorithm



Examples

1. Devise a 7-bit even-parity Hamming code for the digits 0 to 9.

Examples

1. Devise a 7-bit even-parity Hamming code for the digits 0 to 9.

$$
\begin{array}{lll}
0 & \rightarrow & 0000 \\
1 & \rightarrow & 0001 \\
2 & \rightarrow & 0010 \\
3 & \rightarrow & 0011 \\
4 & \rightarrow & 0100 \\
5 & \rightarrow & 0101 \\
6 & \rightarrow & 0110 \\
7 & \rightarrow & 0111 \\
8 & \rightarrow & 1000 \\
9 & \rightarrow & 1001
\end{array}
$$

Examples

1. Devise a 7-bit even-parity Hamming code for the digits 0 to 9.

$$
\begin{array}{ccccccc}
0 & \rightarrow & 0000 & \quad & 0 & \rightarrow & **0*000 \\
1 & \rightarrow & 0001 & & 1 & \rightarrow & **0*001 \\
2 & \rightarrow & 0010 & & 2 & \rightarrow & **0*010 \\
3 & \rightarrow & 0011 & & 3 & \rightarrow & **0*011 \\
4 & \rightarrow & 0100 & & 4 & \rightarrow & **0*100 \\
5 & \rightarrow & 0101 & & 5 & \rightarrow & **0*101 \\
6 & \rightarrow & 0110 & & 6 & \rightarrow & **0*110 \\
7 & \rightarrow & 0111 & & 7 & \rightarrow & **0*111 \\
8 & \rightarrow & 1000 & & 8 & \rightarrow & **1*000 \\
9 & \rightarrow & 1001 & & 9 & \rightarrow & **1*001
\end{array}
$$

Bit **1** checks bits 1, 3, 5, 7, 9

Bit **2** checks bits 2, 3, 6, 7

Bit **4** checks bits 4, 5, 6, 7

Examples

1. Devise a 7-bit even-parity Hamming code for the digits 0 to 9.

0	$\rightarrow$	0000	0	$\rightarrow$	$**0*000$	0	$\rightarrow$	$0*0*000$
1	$\rightarrow$	0001	1	$\rightarrow$	$**0*001$	1	$\rightarrow$	$1*0*001$
2	$\rightarrow$	0010	2	$\rightarrow$	$**0*010$	2	$\rightarrow$	$0*0*010$
3	$\rightarrow$	0011	3	$\rightarrow$	$**0*011$	3	$\rightarrow$	$1*0*011$
4	$\rightarrow$	0100	4	$\rightarrow$	$**0*100$	4	$\rightarrow$	$1*0*100$
5	$\rightarrow$	0101	5	$\rightarrow$	$**0*101$	5	$\rightarrow$	$0*0*101$
6	$\rightarrow$	0110	6	$\rightarrow$	$**0*110$	6	$\rightarrow$	$1*0*110$
7	$\rightarrow$	0111	7	$\rightarrow$	$**0*111$	7	$\rightarrow$	$0*0*111$
8	$\rightarrow$	1000	8	$\rightarrow$	$**1*000$	8	$\rightarrow$	$1*1*000$
9	$\rightarrow$	1001	9	$\rightarrow$	$**1*001$	9	$\rightarrow$	$0*1*001$

Bit **1** checks bits 1, 3, 5, 7, 9

Bit **2** checks bits 2, 3, 6, 7

Bit **4** checks bits 4, 5, 6, 7

Examples

1. Devise a 7-bit even-parity Hamming code for the digits 0 to 9.

0	$\rightarrow$	0000	0	$\rightarrow$	$**0*000$	0	$\rightarrow$	$0*0*000$	0	$\rightarrow$	$000*000$
1	$\rightarrow$	0001	1	$\rightarrow$	$**0*001$	1	$\rightarrow$	$1*0*001$	1	$\rightarrow$	$110*001$
2	$\rightarrow$	0010	2	$\rightarrow$	$**0*010$	2	$\rightarrow$	$0*0*010$	2	$\rightarrow$	$010*010$
3	$\rightarrow$	0011	3	$\rightarrow$	$**0*011$	3	$\rightarrow$	$1*0*011$	3	$\rightarrow$	$100*011$
4	$\rightarrow$	0100	4	$\rightarrow$	$**0*100$	4	$\rightarrow$	$1*0*100$	4	$\rightarrow$	$100*100$
5	$\rightarrow$	0101	5	$\rightarrow$	$**0*101$	5	$\rightarrow$	$0*0*101$	5	$\rightarrow$	$010*101$
6	$\rightarrow$	0110	6	$\rightarrow$	$**0*110$	6	$\rightarrow$	$1*0*110$	6	$\rightarrow$	$110*110$
7	$\rightarrow$	0111	7	$\rightarrow$	$**0*111$	7	$\rightarrow$	$0*0*111$	7	$\rightarrow$	$000*111$
8	$\rightarrow$	1000	8	$\rightarrow$	$**1*000$	8	$\rightarrow$	$1*1*000$	8	$\rightarrow$	$111*000$
9	$\rightarrow$	1001	9	$\rightarrow$	$**1*001$	9	$\rightarrow$	$0*1*001$	9	$\rightarrow$	$001*001$

Bit **1** checks bits 1, 3, 5, 7, 9

Bit **2** checks bits 2, 3, 6, 7

Bit **4** checks bits 4, 5, 6, 7

Examples

1. Devise a 7-bit even-parity Hamming code for the digits 0 to 9.

0	$\to$	0000	0	$\to$	$**0*000$	0	$\to$	$0*0*000$	0	$\to$	$000*000$	0	$\to$	0000000
1	$\to$	0001	1	$\to$	$**0*001$	1	$\to$	$1*0*001$	1	$\to$	$110*001$	1	$\to$	1101001
2	$\to$	0010	2	$\to$	$**0*010$	2	$\to$	$0*0*010$	2	$\to$	$010*010$	2	$\to$	0101010
3	$\to$	0011	3	$\to$	$**0*011$	3	$\to$	$1*0*011$	3	$\to$	$100*011$	3	$\to$	1000011
4	$\to$	0100	4	$\to$	$**0*100$	4	$\to$	$1*0*100$	4	$\to$	$100*100$	4	$\to$	1001100
5	$\to$	0101	5	$\to$	$**0*101$	5	$\to$	$0*0*101$	5	$\to$	$010*101$	5	$\to$	0100101
6	$\to$	0110	6	$\to$	$**0*110$	6	$\to$	$1*0*110$	6	$\to$	$110*110$	6	$\to$	1100110
7	$\to$	0111	7	$\to$	$**0*111$	7	$\to$	$0*0*111$	7	$\to$	$000*111$	7	$\to$	0001111
8	$\to$	1000	8	$\to$	$**1*000$	8	$\to$	$1*1*000$	8	$\to$	$111*000$	8	$\to$	1110000
9	$\to$	1001	9	$\to$	$**1*001$	9	$\to$	$0*1*001$	9	$\to$	$001*001$	9	$\to$	0011001

Bit **1** checks bits 1, 3, 5, 7, 9

Bit **2** checks bits 2, 3, 6, 7

Bit **4** checks bits 4, 5, 6, 7

Examples

2. Devise a code for the digits 0 to 9 whose Hamming distance is 2.

Examples

2. Devise a code for the digits 0 to 9 whose Hamming distance is 2.

Solution: Just add a parity bit:

Problems

2. Devise a code for the digits 0 to 9 whose Hamming distance is 2.

Solution: Just add a parity bit:

$$
\begin{array}{ccl}
0 & \rightarrow & 0000\boxed{0} \\
1 & \rightarrow & 0001\boxed{1} \\
2 & \rightarrow & 0010\boxed{1} \\
3 & \rightarrow & 0011\boxed{0} \\
4 & \rightarrow & 0100\boxed{1} \\
5 & \rightarrow & 0101\boxed{0} \\
6 & \rightarrow & 0110\boxed{0} \\
7 & \rightarrow & 0111\boxed{1} \\
8 & \rightarrow & 1000\boxed{1} \\
9 & \rightarrow & 1001\boxed{0}
\end{array}
$$

Examples

3. In a Hamming code, some bits are "wasted" in the sense that they are used for checking and not information. What is the percentage of wasted bits for messages whose total length (data + check bits) is $2^n - 1$? Evaluate this expression numerically for values of n from 3 to 10.

Examples

3. In a Hamming code, some bits are ‘‘wasted’’ in the sense that they are used for checking and not information. What is the percentage of wasted bits for messages whose total length (data + check bits) is $2^n - 1$? Evaluate this expression numerically for values of n from 3 to 10.

Solution: If the total length is $2^n - 1$ bits, there are n check bits. Consequently, the percentage of “wasted” bits is $n/(2^n - 1) \times 100\%$. Numerically for n from 3 to 10 we get: 42.9%, 26.7%, 16.1%, 9.5%, 5.5%, 3.1%, 1.8%, and 1.0%.

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char
20	(Space)	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	ˆ	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

ASCII (American Standard Code for Information Interchange)

$\mathrm{E} \rightarrow 45$

$\mathrm{a} \rightarrow 61$

$\mathrm{r} \rightarrow 72$

$\mathrm{t} \rightarrow 74$

$\mathrm{h} \rightarrow 68$

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E $\rightarrow$ 45 $\boxed{0\ 1\ 0\ 0 \mid 0\ 1\ 0\ 1}$

a $\rightarrow$ 61

r $\rightarrow$ 72

t $\rightarrow$ 74

h $\rightarrow$ 68

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E $\rightarrow$ 45

a $\rightarrow$ 61

r $\rightarrow$ 72

t $\rightarrow$ 74

h $\rightarrow$ 68

0	1	0	0	0	1	0	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
*	*	0	*	1	0	0	*	0	1	0	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E	$\rightarrow$	45
a	$\rightarrow$	61
r	$\rightarrow$	72
t	$\rightarrow$	74
h	$\rightarrow$	68

0	1	0	0	0	1	0	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
*	*	0	*	1	0	0	*	0	1	0	1
1	*	0	*	1	0	0	*	0	1	0	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E	$\rightarrow$	45
a	$\rightarrow$	61
r	$\rightarrow$	72
t	$\rightarrow$	74
h	$\rightarrow$	68

0	1	0	0	0	1	0	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
*	*	0	*	1	0	0	*	0	1	0	1
1	*	0	*	1	0	0	*	0	1	0	1
1	1	0	*	1	0	0	*	0	1	0	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E	$\rightarrow$	45
a	$\rightarrow$	61
r	$\rightarrow$	72
t	$\rightarrow$	74
h	$\rightarrow$	68

0	1	0	0	0	1	0	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
*	*	0	*	1	0	0	*	0	1	0	1
1	*	0	*	1	0	0	*	0	1	0	1
1	1	0	*	1	0	0	*	0	1	0	1
1	1	0	0	1	0	0	*	0	1	0	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E → 45

a → 61

r → 72

t → 74

h → 68

0	1	0	0	0	1	0	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
*	*	0	*	1	0	0	*	0	1	0	1
1	*	0	*	1	0	0	*	0	1	0	1
1	1	0	*	1	0	0	*	0	1	0	1
1	1	0	0	1	0	0	*	0	1	0	1
1	1	0	0	1	0	0	0	0	1	0	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

4. An extended ASCII character is represented by an 8-bit quantity. The associated Hamming encoding of each character can then be represented by a string of three hex digits. Encode the following extended five-character ASCII text using an even-parity Hamming code: **`Earth`**. Show your answer as a string of hex digits.

E → 45

a → 61

r → 72

t → 74

h → 68

0	1	0	0	0	1	0	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
*	*	0	*	1	0	0	*	0	1	0	1
1	*	0	*	1	0	0	*	0	1	0	1
1	1	0	*	1	0	0	*	0	1	0	1
1	1	0	0	1	0	0	*	0	1	0	1
1	1	0	0	1	0	0	0	0	1	0	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Solution: Each 8-bit ASCII character is encoded into three hex digits.

The first set of hex digits is 0D3:

0	0	0	0	1	1	0	1	0	0	1	1

Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Solution: Each 8-bit ASCII character is encoded into three hex digits.

The first set of hex digits is 0D3:

0	0	0	0	1	1	0	1	0	0	1	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
0	0	0	0	1	1	0	1	0	0	1	1

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Solution: Each 8-bit ASCII character is encoded into three hex digits.

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

The first set of hex digits is 0D3:

0	0	0	0	1	1	0	1	0	0	1	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
0	0	0	0	1	1	0	1	0	0	1	1

bit **1** is correct
bit **2** is correct
bit **4** is incorrect (has the wrong parity)
bit **8** is incorrect (has the wrong parity)

Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Solution: Each 8-bit ASCII character is encoded into three hex digits.

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Add up all the incorrect parity bits. The resulting sum is the position of the incorrect bit: **4** + **8** = 12

The first set of hex digits is 0D3:

0	0	0	0	1	1	0	1	0	0	1	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
0	0	0	0	1	1	0	1	0	0	1	1

bit **1** is correct
bit **2** is correct
bit **4** is incorrect (has the wrong parity)
bit **8** is incorrect (has the wrong parity)

corrected code:

0	0	0	0	1	1	0	1	0	0	1	0

Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Solution: Each 8-bit ASCII character is encoded into three hex digits.

Bit **1** checks bits 1, 3, 5, 7, 9, 11

Bit **2** checks bits 2, 3, 6, 7, 10, 11

Bit **4** checks bits 4, 5, 6, 7, 12

Bit **8** checks bits 8, 9, 10, 11, 12

Add up all the incorrect parity bits. The resulting sum is the position of the incorrect bit: **4** + **8** = 12

The first set of hex digits is 0D3:

0	0	0	0	1	1	0	1	0	0	1	1

1	***2***	3	***4***	5	6	7	***8***	9	10	11	12
0	0	0	0	1	1	0	1	0	0	1	1

bit **1** is correct
bit **2** is correct
bit **4** is incorrect (has the wrong parity)
bit **8** is incorrect (has the wrong parity)

corrected code:

0	0	0	0	1	1	0	1	0	0	1	0

0	1	1	0	0	0	1	0

62 $\rightarrow$ b

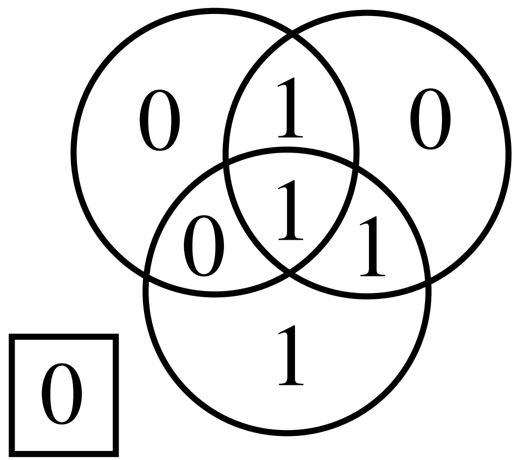
Examples

5. The following string of hex digits encodes extended ASCII characters in an even-parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded.

Solution: Each 8-bit ASCII character is encoded into three hex digits. The first set of hex digits: 0D3, has an error in bit 12 (as indicated by the fact that bit 4 and bit 8 have the wrong parity). The next set, DD3 has bit 11 wrong; the set 0F2 has bit 7 wrong; the set 5C1 has bit 9 wrong; the set 1C5 has bit 1 wrong; the last set CE3 does not contain any errors. After the bit positions are corrected and the data extracted from the code words and looked up in the ASCII table, the encoded characters are: **`babies`**.

Examples

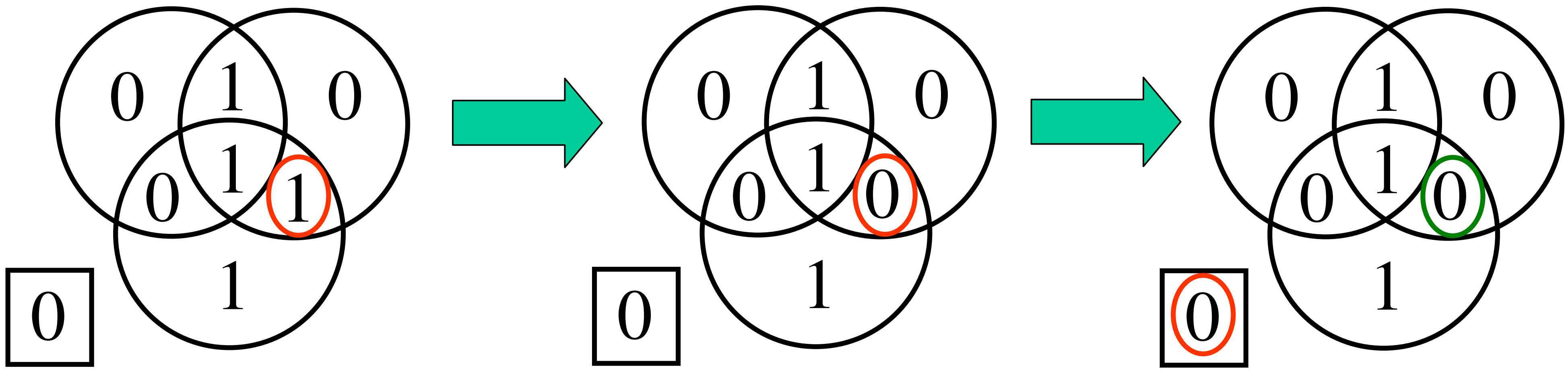
6. Check if there is a single/double error. In the single error case, correct the error.



Examples

6. Check if there is a single/double error. In the single error case, correct the error.

Solution:



We have a double error.