Solutions to Week 4 Error Correcting Code and BitMasking

Question 1

The bitwise operations AND, OR, NOT and XOR are used to do bit-masking, that is, to set **(made 1)** or reset **(made 0)** particular bits on a byte (or word).

Find the appropriate bitmask(s) M and bitwise operator(s) for any byte A for the following cases showing all your working out and intermediate steps:

1. Reset bit 7 leaving the rest untouched.
2. Make sure that bit 0 is set and only this bit is set in the byte.
3. Flip the MSB and bit 3 leaving the other bits untouched.
4. Reset bits 2 and 6, all other bits are set.

Answer:
A group of 8 bits is called a byte.
The left-end bit of a number represented in binary is called the most significant bit (MSB,), and the right-end bit is called the least significant bit (LSB).
Bit 0 is on the right end of the byte and bit 7 is on the left end.
Bit 0 is the LSB and bit 7 is the MSB.

1. M= 0111 1111 and the operator is AND
2. M1= 0000 0000 operator AND then M2=0000 0001 with OR
3. M = 1000 1000 with XOR
4. M1= 1111 1111 operator OR then M2=1011 1011 with AND

Question 2

Calculate the number of parity bits needed to detect and correct a single-bit error in a string of **8, 16, 32** and **64** bits.

Answer: The formula for Hamming Code is the least number of parity bits, p that must satisfy: 2^p >= m+p+1, where m is the number of data bits.

For m=8, using trial and error

Try p=2:	L.H.S= 2^2 = 4 ,	R.H.S=8+2+1=11	So p=2 is not enough
Try p=3:	L.H.S= 2^3 = 8 ,	R.H.S=8+3+1=12	So p=3 is not enough
Try p=4:	L.H.S= 2^4 = 16 ,	R.H.S=8+4+1=13	So p=4 is enough.

For m=16

Try p=5:	L.H.S= 2^5 = 32 ,	R.H.S=16+5+1=22	So p=5 is enough.

For m=32

Try p=6:	L.H.S= 2^6 = 64 ,	R.H.S=32+6+1=39	So p=6 is enough.

For m=64

Try p=7:	L.H.S= 2^7 = 128 ,	R.H.S=64+7+1=72	So p=7 is enough.

Question 3

Determine the single-bit error correction **Hamming** code using **even**-parity for the 7-bit ASCII character "!"

1. How many Hamming parity bits are required to cover all 7 bits?
2. Encode this character into its own 11-bit even Hamming code.
3. Express that result as a single hexadecimal string.

Answer:
Start by finding the 7-bit ASCII code for "!"; this becomes your 7 data bits.
"!" => $010\ 0001_2$

7 data bits need 4 parity bits. (This working should be shown in assignments.)
The Hamming code for "!" will be 7 + 4 = 11 bits in length. A table can be drawn up like this:

11	10	9	8	7	6	5	4	3	2	1	**Bit Position**
D7	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1	**Parity/data bit**
0	1	0		0	0	0		1			**Value**

Things to note at this stage:

1. The position numbering. It goes from 11 to 1.
2. How we determine what is a parity bit and what is a data bit? Parity bits go at the positions that are powers of 2, so parity bit 4 goes at position$2(4-1)$ = 8. The remaining locations will hold data bits.
3. We fill in the data bits and leave the parity bits blank for the moment.

Now, it is time to work out the values of the parity bits. Parity bits check a fixed number of bits, skip the same number of bits and repeat, starting from the bit's own position. P1 starts at location 1, checks one bit (that bit), skips bit 2, checks bit 3 etc. P2 starts at location 2, checks bits 2 and 3, skips 4 and 5, checks bits 6 and 7 etc.

So:

- P1 covers bits 1, 3, 5, 7, 9, 11
- P2 covers bits 2, 3, 6, 7, 10, 11
- P3 covers bits 4, 5, 6, 7

P4 covers bits 8, 9, 10, 11

For P1, the students can fill 3, 5, 7, 9, 11 and it is easy to calculate P1 — red Similarly, for P2, P3, and P4.

Once you have all the parity bits you can calculate the green row:

11	10	9	8	7	6	5	4	3	2	1	Bit Position
D7	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1	Parity/data bit
0	1	0	1	0	0	0	0	1	0	1	Bit Value
0		0		0		0		1		1	P1
0	1			0	0			1	0		P2
				0	0	0	0				P3
0	1	0	1								P4

Once you have all the parity bits you can calculate the green row:
The Hamming code for ! is 010 1000 0101
= 0010 1000 0101 (in nibbles)
= $285 in hexadecimal

Question 4

Using the **even**-parity **Hamming** code for "!" from Question 3:

1. Flip P4 and show it can be corrected.
2. Flip P4 and D3, show these **cannot be corrected.**

Answer:

1. Flip P4 and show it can be corrected.

11	10	9	8	7	6	5	4	3	2	1	Bit Position
D7	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1	Parity/data bit
0	1	0	0	0	0	0	0	1	0	1	Bit Value
0		0		0		0		1		1	P1
0	1			0	0			1	0		P2
				0	0	0	0				P3
0	1	0	0								P4

P1,P2 and P3 are all correct but P4 should be a 1 for even parity so the flipped bit can be corrected.

- P1 covers bits 1, 3, 5, 7, 9, 11
- P2 covers bits 2, 3, 6, 7, 10, 11
- P3 covers bits 4, 5, 6, 7
- P4 (incorrect) covers bits 8, 9, 10, 11
- All bits covered except 8 so P4=bit 8 has been flipped and so can be corrected.

2. Flip P4 and D3, show these **cannot be corrected.**

11	10	9	8	7	6	5	4	3	2	1	**Bit Position**
D7	D6	D5	**P4**	D4	**D3**	D2	P3	D1	P2	P1	**Parity/data bit**
0	1	0	**0**	0	1	0	0	1	0	1	**Bit Value**
0		0		0		0		1		1	**P1**
0	1			0	1			1	0		**P2**
				0	1	0	0				**P3**
0	1	0	**0**								**P4**

P1 is correct, P2,P3 and P4 are all wrong — cannot determine that P4 and D3 have been flipped because:

- P1 (correct) covers bits 1, 3, 5, 7, 9, 11
- P2 (incorrect) covers bits 2, 3, 6, 7, 10, 11
- P3 (incorrect) covers bits 4, 5, 6, 7
- P4 (incorrect) covers bits 8, 9, 10, 11

Now we know:

- P2 (incorrect) so bits 2, 6, 10 could be in error; since 3, 7, 11 are covered by P1
- P3 (incorrect) so bits 4, 6 could be in error since 5, 7 are covered by P1
- P4 (incorrect) so bits 8, 10 could be in error since 9, 11 are covered by P1

You cannot determine which bits have been flipped.

Question 5

Repeat using **even**-parity **SECDED** for ! from Question 3:

1. Flip P4 and show it can be corrected.
2. Flip P4 and D3, show these **can be detected but corrected.**

Answer:

SECDED (single error correction, double error detection) code — extended hamming code by adding an **overall** parity bit in bit position 0.

- **P0 covers bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11**
- P1 covers bits 1, 3, 5, 7, 9, 11
- P2 covers bits 2, 3, 6, 7, 10, 11
- P3 covers bits 4, 5, 6, 7
- P4 covers bits 8, 9, 10, 11

For P1, the students can fill 3, 5, 7, 9, 11 and it is easy to calculate P1 — red Similarly, for P2, P3, P4.

11	10	9	8	7	6	5	4	3	2	1	0	**Bit Position**
D7	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1	**P0**	**Parity/data bit**
0	1	0	1	0	0	0	0	1	0	1	0	**Bit Value**
0	1	0	1	0	0	0	0	1	0	1	0	**P0**
0		0		0		0		1		1		**P1**
0	1			0	0			1	0			**P2**
				0	0	0	0					**P3**
0	1	0	1									**P4**

Once you have all the parity bits you can calculate the green row.
The SECDED code for "!" is 0101 0000 1010
= 0101 0000 1010 (in nibbles)
= $50A in hexadecimal

1. Flip P4 and show it can be corrected.

11	10	9	8	7	6	5	4	3	2	1	0	**Bit Position**
D7	D6	D5	**P4**	D4	D3	D2	P3	D1	P2	P1	**P0**	**Parity/data bit**
0	1	0	0	0	0	0	0	1	0	1	0	**Bit Value**
0	1	0	0	0	0	0	0	1	0	1	0	**P0**
0		0		0		0		1		1		**P1**
0	1			0	0			1	0			**P2**
				0	0	0	0					**P3**
0	1	0	0									**P4**

P1, P2 and P3 are all correct, P4 shows an error and P0 shows an error.
P1,P2 and P3 are all correct but P4 should be a 1 for even parity so the flipped bit can be corrected.

- P0 (incorrect) covers 0, 1, 2 ,3, 4, 5, 6, 7, 8, 9, 10, 11
- P1 covers bits 1, 3, 5, 7, 9, 11
- P2 covers bits 2, 3, 6, 7, 10, 11
- P3 covers bits 4, 5, 6, 7

- P4 (incorrect) covers bits 8, 9, 10, 11

All bits covered except 8 so P4=bit 8 has been flipped and so can be corrected.
Since SECDED parity said there was an error, and Hamming says there is an error, there must be a single error (we know it can't be two bit flips, or SECDED parity would say there was no error. **We assume 3 or more bit flips is so unlikely that it won't happen!**)

2. Flip P4 and D3, show these cannot be corrected but detected.

11	10	9	8	7	6	5	4	3	2	1	0	**Bit Position**
D7	D6	D5	**P4**	D4	**D3**	D2	P3	D1	P2	P1	**P0**	**Parity/data bit**
0	1	0	**0**	0	1	0	**0**	**1**	0	**1**	**0**	**Bit Value**
0	1	0	**0**	0	1	0	**0**	**1**	0	**1**	**0**	**P0**
0		0		0		0		1		**1**		**P1**
0	1			0	1			1	**0**			**P2**
				0	1	0	**0**					**P3**
0	1	0	**0**									**P4**

P1 is correct, P2,P3 and P4 are all wrong —P0 is correct so 2 bits have been flipped.
P0, P1 are correct, P2,P3 and P4 are all wrong — cannot determine that P4 and D3 have been flipped because:

- P0 (correct) covers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
- P1 (correct) covers bits 1, 3, 5, 7, 9, 11
- P2 (incorrect) covers bits 2, 3, 6, 7, 10, 11
- P3 (incorrect) covers bits 4, 5, 6, 7
- P4 (incorrect) covers bits 8, 9, 10, 11

Now we know:

- P0 (correct) covers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
- P2 (incorrect) so bits 2, 6, 10 could be in error since 3, 7, 11 are covered by P1
- P3 (incorrect) so bits 4, 6 could be in error since 5, 7 are covered by P1
- P4 (incorrect) so bits 8, 10 could be in error since 9, 11 are covered by P1

We **cannot** determine the error bits. **But we know more than one error has occurred as SECDED parity hasn't detected an error.** So we can't trust Hamming results.
We have detected that multiple errors have occurred -- but we can't correct it.

Question 6

Data has been encoded using an **odd**-parity **SECDED** code. The binary code was then retrieved as 0111 0110.

1. Has an error occurred? Explain your answer (and show your working).
2. If there was an error, either correct it reporting the correct binary string or explain why it could not be corrected.

Answer:

7	6	5	4	3	2	1	0	**Bit Position**
D4	D3	D2	P3	D1	P2	P1	P0	**Parity/data bit**
0	1	1	1	0	1	1	0	**Bit Value**
0	1	1	1	0	1	1	0	**P0**
0		1		0		1		**P1**
0	1			0	1			**P2**
0	1	1	1					**P3**

- P0: correct.
- P1: incorrect, error in 1, 3, 5, 7.
- P2: incorrect, error in 2, 3, 6, 7.
- P3: correct. 4, 5, 6, 7 are correct.

Since 4, 5, 6, 7 are correct, you can either say 1 or 3 are in error from P1 OR can either say 2 or 3 are in error from P2. Although 3 occurs twice bits 1 and 2 could still be flipped!

So from this simple example, **SECDED has detected 2 bits in error but you cannot correct them!**