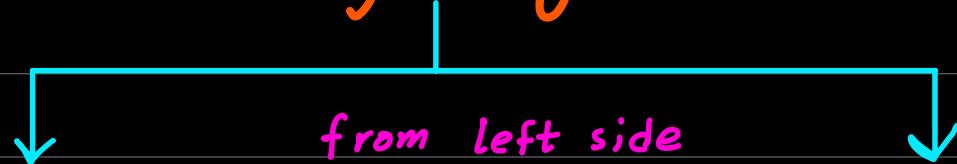


Data Representation

- Data representation refers to the form in which data is processed, stored and transmitted

Binary Magnitude



Numbers starting with
"0" represents Positive
number

(01100011)

↳ first bit is binary magnitude

Numbers starting with
"1" represents negative number
(10001100)

* Using decimal prefixes causes confusion when dealing with bigger numbers

Prefixes



Binary Prefix

- kibibyte
- Mebibyte
- Gibibyte
- Tebibyte

1024

Decimal Prefix

- kilobyte
- Megabyte
- Gigabyte
- Terabyte

1000

Binary Prefixes

$$\begin{aligned} \times 1024 &\rightarrow \text{kibibyte} \\ \times 1024 &\rightarrow \text{Mebibyte} \\ \times 1024 &\rightarrow \text{Gibibyte} \\ \times 1024 &\rightarrow \text{Tebibyte} \end{aligned} \quad \left(\begin{array}{l} \xrightarrow{\quad} \\ \xrightarrow{\quad} \\ \xrightarrow{\quad} \\ \xrightarrow{\quad} \end{array} \right) \div 1024$$

Decimal Prefixes

$$\begin{aligned} \times 1000 &\rightarrow \text{kilobyte} \\ \times 1000 &\rightarrow \text{Megabyte} \\ \times 1000 &\rightarrow \text{Gigabyte} \\ \times 1000 &\rightarrow \text{Terabyte} \end{aligned} \quad \left(\begin{array}{l} \xrightarrow{\quad} \\ \xrightarrow{\quad} \\ \xrightarrow{\quad} \\ \xrightarrow{\quad} \end{array} \right) \div 1000$$

• Smallest unit is bit

• One byte contains 8 bits

0110 1110
1 byte

• 4 bits are known as nibble

0110 1110
1 nibble 1 nibble

• One kilobyte contains 1000 bytes

• One kibibyte contains 1024 bytes

• One Megabyte contains 1000 kilobytes

• One mebibyte contains 1024 kibibytes

• One gigabyte contains 1000 megabytes

• One gibibyte contains 1024 mebibytes

• " Terabyte " 1000 gigabytes

• " Tebibyte " 1024 gibibytes

EXAM STYLE QUESTION

1 (a) Draw one line from each binary value to its equivalent (same) value on the right.

Binary value

8 bits

1024 bytes
8000 bits

1000 kilobytes

1024 mebibytes

8192 bits

1024 bytes

1 kibibyte

1 gigabyte

1 byte

1 kilobyte

1 gibibyte

1 mebibyte

[5]

Number Systems

ways a number can be represented.

Binary
(base-2)

Denary
(base-10)

Hexadecimal
(base-16)

Note: Memorized list (128 64 32 16 8 4 2 1)

CASE 1: Denary to Binary

Q- Convert 65 into 8-bit binary

Step 1: Write down memorized list

Step 2: Place 1 on specific place so total could be the same as denary number given

$$\begin{array}{ccccccccc} & 16+16 & 8+8 & 4+4 & 2+2 \\ 128 & 64 & \overbrace{32}^{\sim} & \overbrace{16}^{\sim} & \overbrace{8}^{\sim} & \overbrace{4}^{\sim} & \overbrace{2}^{\sim} & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \rightarrow \text{Ans} \end{array}$$

EXAM STYLE QUESTION

2 (a) Convert the following denary integer into 8-bit binary.

55

0	0	1	1	0	1	1	1
128	64	32	16	8	4	2	1

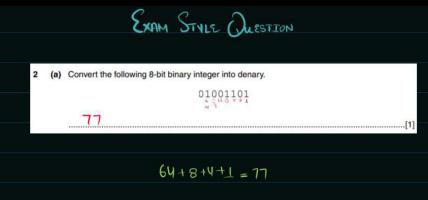
[1]

CASE 2 : Binary to Denary

Q- Convert 0100001 into denary

Step 1: Write down memorized list on the binary number.

Step 2: Add only the numbers with "1"



128 64 32 16 8 4 2 1
0 1 0 0 0 0 0 1

* Write memorized list from right hand side

$$64 + 1 = 65 \text{ Ans}$$

CASE 3 : Binary to Hexadecimal

Q- Convert 11011001 into Hexadecimal

Step 1 : From right side, divide the binary value into groups of four. → 1101 1001

A	=	10
B	=	11
C	=	12
D	=	13
E	=	14
F	=	15

Step 2: Write the memorized list separately for each group

8421 8421
1101 1001

Step 3: Add numbers with 1 and if it is greater than 9, then use value box.

8421 8421
1101 1001

8+4+1 8+1

13 → D9 ← 9
Ans

EXAM STYLE QUESTION

(b) The current contents of a general purpose register (X) are:

X []
8 4 2 1 8 4 2 1

(i) The contents of X represent an unsigned binary integer.

F2

CASE 4 : Denary to Hexadecimal

Q- Convert 195 to Hexadecimal

Step 1 : First convert the denary number to binary

128 64 32 16 8 4 2 1

1 1 0 0 0 0 1 1

Step 2: Repeat steps in CASE 3

- Complete groups

8 4 2 1 8 4 2 1
| | 0 0 0 0 | |

8+4 2+1

12 3
↓ ↓
C 3 Ans

will be made

- If complete groups are not forming, add zeroes at left side.

CASE 5 : Hexadecimal to Denary

- Reverse steps in CASE 3

Q- convert 4E into denary

8 4 2 1 8 4 2 1

0 1 0 0 1 1 1 0

128 64 32 16 8 4 2 1

0 1 0 0 1 1 1 0 → 8+4+2+64 = **78** Ans

(d) Convert the following hexadecimal number into denary. $E=14$

4E

78 [1]

0 1 0 0 1 1 1 0
8 4 2 1 8 4 2 1

0 1 0 0 1 1 1 0
128 64 32 16 8 4 2 1

$64 + 8 + 4 + 2 = 78$

Binary Coded Decimal

- Uses a 4 bit code to represent each denary digit.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

* A type of
format of Binary
number system

Note: Any number greater than 9 can not be represented in BCD (4-bit)

CASE 1: Denary to BCD

Q- Convert 54 into BCD

Step 1: Convert each digit into BCD format.

5	4
8 4 2 1	8 4 2 1
0 1 0 1	0 1 0 0

→ 0101 0100 Ans

EXAM STYLE QUESTION

(b) (i) Convert the following denary integer into Binary Coded Decimal (BCD).

653

..... 0110 0101 0011

[1]

CASE 2 : BCD to Denary

Q- Convert 01010100 BCD to Denary

Step 1: Make groups of 4

* Make groups from right hand side.

0101 0100

Step 2: Write memorized List on each group and add the numbers with 1

$\overbrace{5}^1$ $\overbrace{4}^1$
8 4 2 1 8 4 2 1
0 1 0 1 0 1 0 0 → 54 Ans

EXAM STYLE QUESTION

(b) Convert the following Binary Coded Decimal (BCD) number into denary.

10000011

..... 10000011

83

[1]

Q- Describe how denary integers larger than 9 can be converted into BCD. Give an example in your answer.

* Always give example

- Each Denary digit is written as 4-bit binary number ①
- $46 \rightarrow 0100\ 0110$ ②

Q- Describe how a 8-bit BCD can be converted into a denary integer. Give an example in your answer

- Binary number is split into groups of 4 bits, starting from the right.
- Each group of 4-bits is converted into a denary digit
- E.g: $0011\ 0111 \rightarrow 37$

Q- Identify practical applications where BCD is used.

- Any scenario where a single digit needs to be transmitted / displayed

- calculator, digital clock.

Binary Addition

Sum Carry

$$\begin{array}{r} 0 \ 0 \\ 0 \ 1 \\ 1 \ 0 \\ 1 \ 1 \\ \hline 1 \ 1 \end{array} \quad \begin{array}{r} 0 \ 0 \\ 1 \ 0 \\ 0 \ 1 \\ 1 \ 1 \\ \hline 1 \ 1 \end{array}$$

(ii) The amount of green in binary is 00100011. This has the denary number 15 added to it to create a second colour.

Add the denary number 15 to the binary number 00100011 and give your answer in binary.

Perform the addition in binary. Show your working.

Working $\begin{array}{r} 00100011 \\ + 00001111 \\ \hline 00110010 \end{array}$

Answer (in binary) [3]

(b) (i) Perform the following binary addition. Show your working.

$$\begin{array}{r} 11111 \\ + 10101010 \\ \hline 00110111 \end{array}$$

[2]

① $\begin{array}{r} 010 \\ + 011 \\ \hline 101 \end{array} = 101$

② $\begin{array}{r} 011 \\ + 110 \\ \hline 110 \end{array}$

③ $\begin{array}{r} 01110 \\ + 111 \\ \hline 1010101 \end{array}$

④ $\begin{array}{r} 1100 \\ + 0110 \\ \hline 10010 \end{array}$

⑤ $\begin{array}{r} 1110 \\ + 10111 \\ \hline 10010101 \end{array}$

⑥ $\begin{array}{r} 11111 \\ + 11101 \\ \hline 111100 \end{array}$

Conversion of Negative Numbers

Q- Convert -125 to Binary Form.

Step 1: Ignore '-' sign and convert it to Binary.

128	64	32	16	8	4	2	1
0	1	1	1	1	1	0	1

* If answer is not forming, make different combination.

0	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---

Step 2: Apply 2's compliment method.

0	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

→ Inverse (One's compliment)

1 → 2's compliment

1	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

Answer: 10000011

Q- Convert the denary number -194 into 12-bit two's compliment

128 64 32 16 8 4 2 1

1 1 0 0 0 0 1 0

means two's

compliment of all
12-bits

=> 000011000010

$$\begin{array}{r} 111100111101 \\ + \quad \quad \quad 1 \\ \hline 111100111110 \end{array}$$

111100111110

Note: All **2's compliment** numbers that start with 1 are negative numbers

(iv) State why register H does not currently contain a Binary Coded Decimal (BCD).

The first 4 bits would be 12 which is greater than 9
which is not valid for BCD [1]

(c) H is a register. The current contents of H are:

128	64	32	16	8	4	2	1
1	1	0	0	0	0	0	1

two's compliment is not applied, so treat the binary
The current contents of register H represent an unsigned binary integer, as normal
and ignore sign bit.

(i) Convert the value in register H into denary.

13

(ii) Convert the value in register H into hexadecimal.

C1

(iii) The current contents of register H represent a two's complement binary integer.

11000010
00111110
00111111

Convert the value in register H into denary.

-63

Binary Subtraction

Explanation:

$$40 - 15 = 25$$

OR

$$40 + -15 = 25$$

Note: Apply two's compliment and then same working as binary addition

(iii) Hexadecimal 23 in two's complement representation is 00100011. The denary number 10 needs to be subtracted from this value.

Subtract the denary number 10 from the two's complement representation 00100011.

Give your answer in binary. Show your working.

Working	Converting 10 to -10	Subtracting
00001010	$\rightarrow 10$	1 0 0 1 0 0 0 1 1
11110101		+ 1 1 1 1 0 1 1 0
1		0 0 0 1 1 0 0 1
11110101		

Answer (in binary) 00011001

Overflow

Concept: More bits less space to store

A binary addition diagram on lined paper. The top number is 111111. The bottom number is 10110111. A plus sign (+) is followed by a horizontal line. The sum is 1011000001. A circled '1' is shown above the first '0' in the sum, with a blue arrow pointing from the word 'OVERFLOW' to it. Below the sum, two bullet points explain the consequences of overflow:

- Results in accuracy issue of data
- Ignore overflow bit when writing answers

Q- State how an overflow can occur when adding two binary integers

• The result is a larger number that can not be stored in the given number of bits

ASCII , Extended ASCII , Unicode



Note: The numbers could be easily represented into binary form, but there was no way to represent alphabets / characters

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0	0	32	20	40	[space]	64	40	100	@	96	60	140	
1	1	1	1	33	21	41	!	65	41	101	A	97	61	141	a
2	2	2	2	34	22	42	“	66	42	102	B	98	62	142	b
3	3	3	3	35	23	43	#	67	43	103	C	99	63	143	c
4	4	4	4	36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5	5	37	25	45	%	69	45	105	E	101	65	145	e
6	6	6	6	38	26	46	&	70	46	106	F	102	66	146	f
7	7	7	7	39	27	47	‘	71	47	107	G	103	67	147	g
8	8	8	10	40	28	50	(72	48	110	H	104	68	150	
9	9	9	11	41	29	51)	73	49	111	I	105	69	151	i
10	A	A	12	42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	B	13	43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	C	14	44	2C	54	-	76	4C	114	L	108	6C	154	l
13	D	D	15	45	2D	55	*	77	4D	115	M	109	6D	155	m
14	E	E	16	46	2E	56	*	78	4E	116	N	110	6E	156	n
15	F	F	17	47	2F	57	*	79	4F	117	O	111	6F	157	o
16	0	20	48	30	60	0	80	50	120	P	112	70	160	-	
17	11	21	49	31	61	1	81	51	121	Q	113	71	161	q	
18	12	22	50	32	62	2	82	52	122	R	114	72	162	r	
19	13	23	51	33	63	3	83	53	123	S	115	73	163	s	
20	14	24	52	34	64	4	84	54	124	T	116	74	164	t	
21	15	25	53	35	65	5	85	55	125	U	117	75	165	u	
22	16	26	54	36	66	6	86	56	126	V	118	76	166	v	
23	17	27	55	37	67	7	87	57	127	W	119	77	167	w	
24	18	28	56	38	68	8	88	58	128	X	120	78	168	x	
25	19	29	57	39	69	9	89	59	131	Y	121	79	171	y	
26	1A	32	58	3A	72	:	90	5A	132	Z	122	7A	172	z	
27	1B	33	59	3B	73	:	91	5B	133	I	123	7B	173	{	
28	1C	34	60	3C	74	<	92	5C	134	\	124	7C	174		
29	1D	35	61	3D	75	=	93	5D	135	_	125	7D	175	^	
30	1E	36	62	3E	76	>	94	5E	136	~	126	7E	176	-	
31	1F	37	63	3F	77	?	95	5F	137	-	127	7F	177		



$$A = 65$$

$$a = 97$$

ASCII

- 7 bits are used for character representation
- 128 characters can be represented $(127 + 1)$

$$\begin{array}{ccccccccc} 64 & 32 & 16 & 8 & 4 & 2 & 1 & = A \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Extended ASCII

- 8 bits for character representation
- ASCII codes were 7 bit long, but then it was extended to have bit length of 8
- 256 characters can be represented.

Unicode



Problem:

→ Less number of bits for representation of characters of other languages.

- Uses 16, 24, 32 bits to represent characters.

Q- What are the disadvantages of using ASCII code?

- Only 128 characters can be represented
- Uses values 0 to 127
- Many characters in other languages can not be represented
- In extended ASCII, the characters from 128 to 255 may be coded differently in different systems. (No standard set)

Q- How Unicode is designed to overcome disadvantages of ASCII?

- Uses 16, 24, 32 bits
- Unicode is designed to be superset of ASCII
- Designed so that most characters in other languages can be represented

(b) Explain how a word such as 'HOUSE' is represented by the ASCII character set.

- Each character has its own unique code
- Each character in the word is replaced by its code
- The codes are stored in the order in the word.

[2]