



UNIVERSITI
TEKNOLOGI
MALAYSIA
www.utm.my

SCSR1013 DIGITAL LOGIC

**MODULE 2b:
DATA ORGANIZATION (CODES)**

FACULTY OF COMPUTING

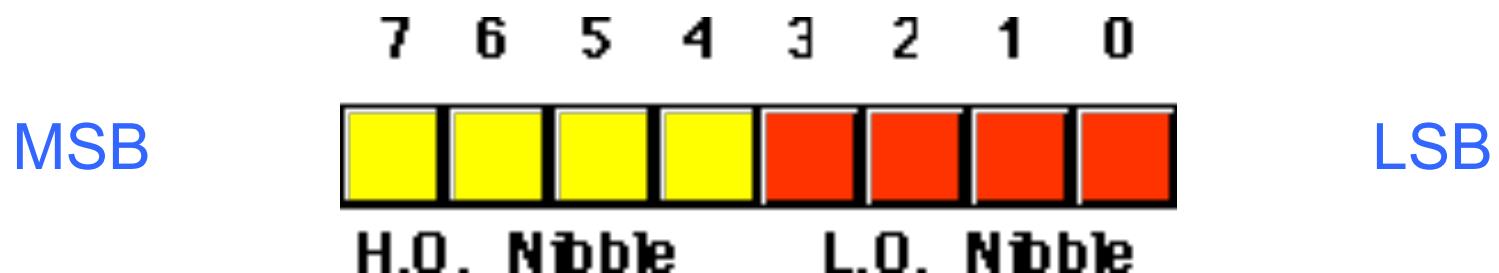
Data Organization

- A value may take an arbitrary number of bits.
- Common collections are single bits
 - smallest "unit" of data on a binary computer is a single [bit](#)
 - groups of four bits called [nibbles](#)
 - groups of eight bits called [bytes](#)
 - groups of 16 bits called [words](#)
- The bits in a byte are normally numbered from zero to seven.



- Bit 0 is the low order bit (rightmost) or least significant bit (LSB) bit 7 is the high order bit (leftmost) or most significant bit (MSB) of the byte.

Note 1 byte also contains exactly 2 nibbles



- 4 bits
- **Major uses:**
 - BCD (Binary Coded Decimal)
 - Hexadecimal numbers



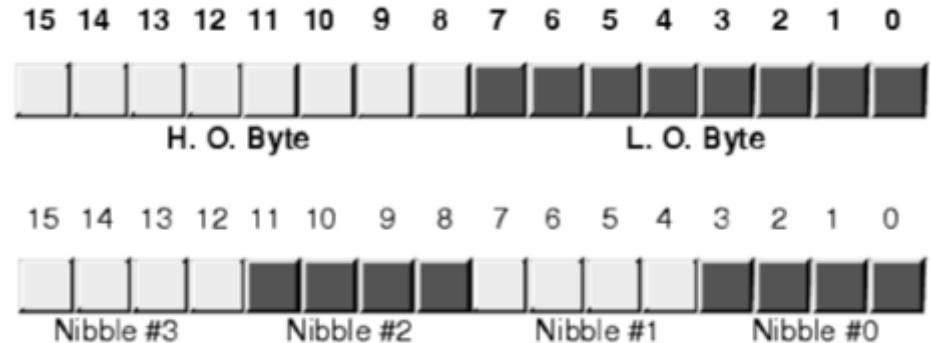
Example: 0111, 1011 and 1111.

7_{16} , B_{16} , F_{16}



- 8 bits
- Total values: $2^8 = 256$
- **Major uses:**
 - Numeric values (0 ... $2^8-1 = 0 ... 255$)
 - Signed numbers: (-128 to +127)

- 16 bits = 2 bytes
- Bit 0 to 15
- **Total values:**
 - $2^{16} = 65,536$



- **Major uses of word:**
 - signed integer (-32,768 ... +32,767)
 - unsigned integer (0 ... $2^{16}-1$) = 0 ... 65,535)
 - UNICODE characters

What are codes?

- Code is a representation of information generated by following a certain rules.
 - In general, we need code because:
 - Code is unique
 - Codes are easy to process
 - Code is easy to represent
 - Codes enable communication in place where ordinary spoken or written language is difficult or impossible, eg Morse Code
 - Due to this, code can simplify the process (such as manipulation and arithmetic operations) of the information in the digital system.
- We will learn:

 - i. BCD codes
 - ii. Gray Codes
 - iii. ASCII codes
 - iv. Parity codes/bit

Binary Coded Decimal (BCD)

- BCD is a way to express each of the decimal digits with a binary code.
- There are only 10 code groups in the BCD system, one for every digit (0000 – 1001)

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

Invalid codes are **1010, 1011, 1100, 1101, 1110, 1111**

Example 1: Convert 3245 to BCD

3 2 4 5
3245 = 0011 0010 0100 0101

Example 2: Convert 7848 to BCD

7 8 4 8
7848 = 0111 1000 0100 1000

Gray Codes

- Designed to prevent false output from electromechanical switches.
- Are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.
- In modern digital communications, Gray codes play an important role in error correction.
- It is arranged so that every transition from one value to the next value involves only one bit change.
- Sometimes referred to as reflected binary, because the first eight values compare with those of the last 8 values, but in reverse order.

Electromechanical switches



Basics



Toggle



Limit



Decimal	Binary	Gray Code
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0
16	1 0 0 0 0	1 1 0 0 0

Gray Code Conversion

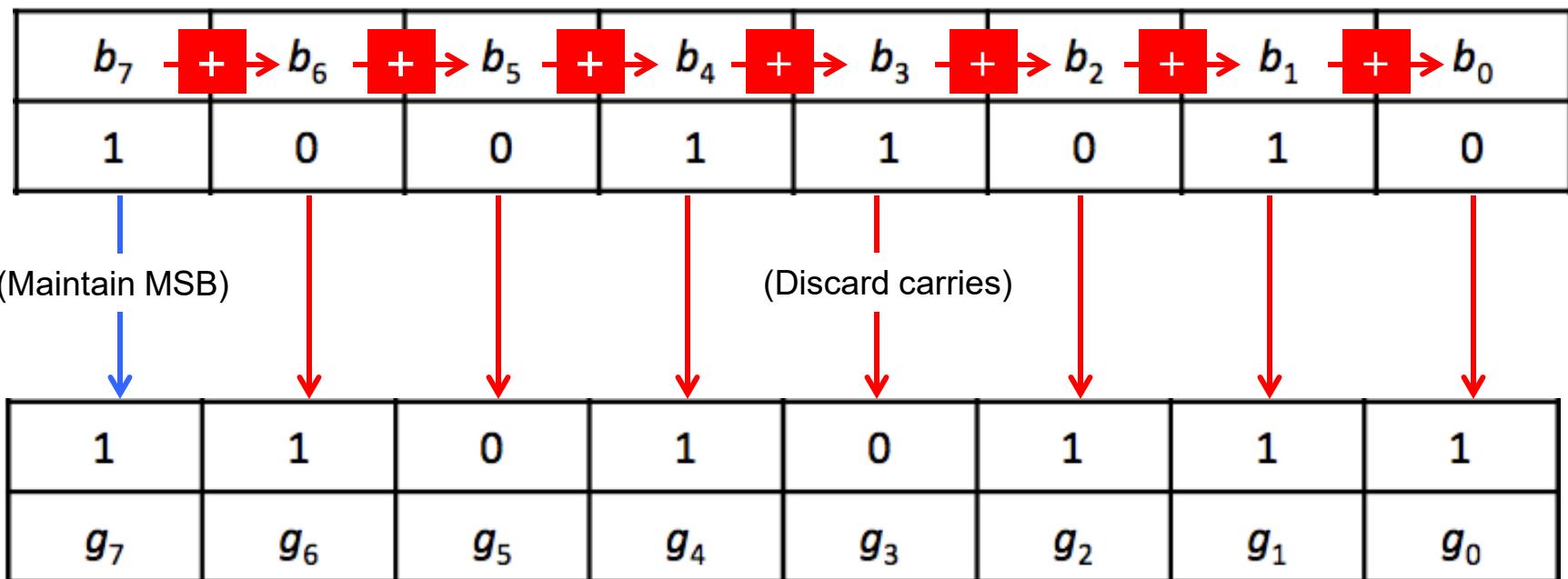
- Binary to Gray Code
 - 1. Record the MSB as it is
 - 2. Add the MSB to the next bit of binary, record the sum and neglect carry.
 - 3. Repeat the process

$\begin{array}{r} 1 \\ \downarrow \\ 1 \\ \downarrow \\ 1 \\ \downarrow \\ 0 \\ \downarrow \\ 1 \\ \downarrow \\ 0 \\ \downarrow \\ 1 \\ \downarrow \\ 1 \\ \downarrow \\ 1 \end{array}$

Example:

Convert ~~100110~~ to its equivalent gray code value

10011010

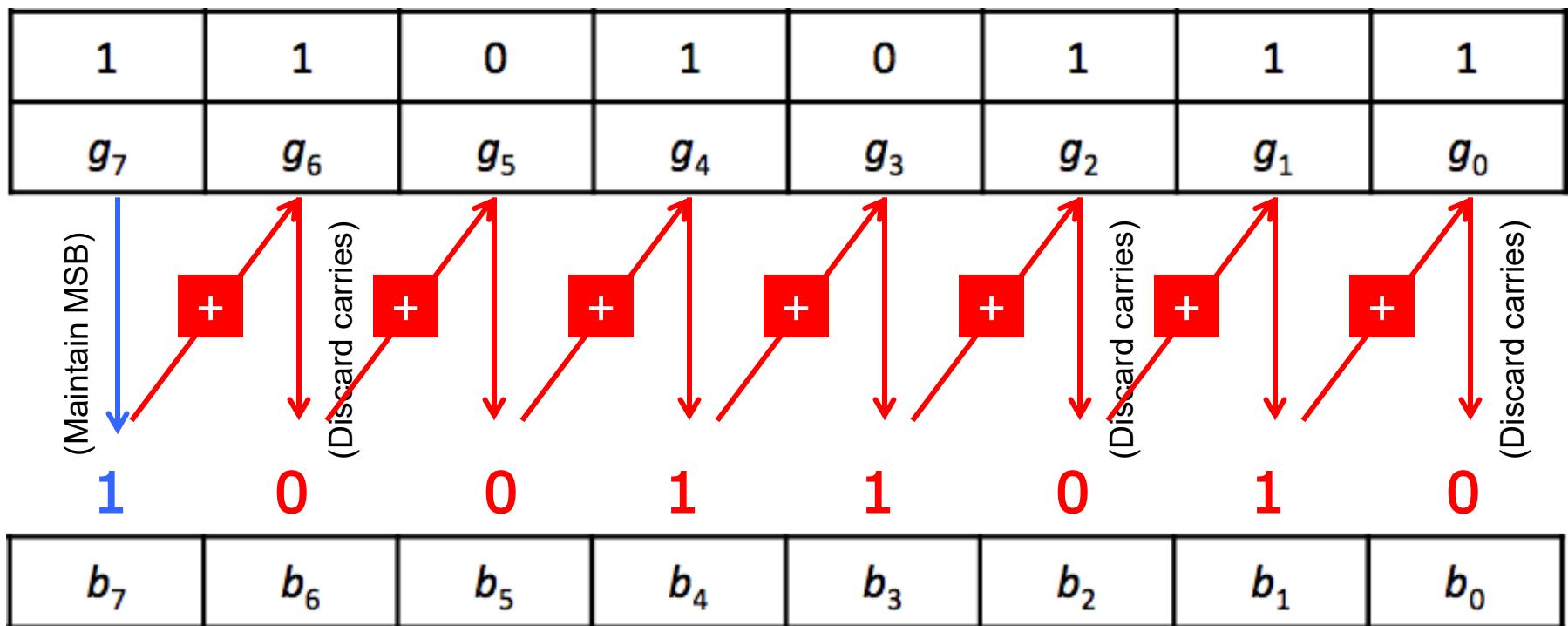


Gray Code Conversion

- Gray Code to Binary
 1. Record the MSB as it is
 2. Add the MSB to the next bit of Gray code, record the sum and neglect carry.
 3. Repeat the process

Example:

Convert the Gray code 11010111 to binary.



Parity Code

- Parity bit used for bit error detection
 - Even parity – total number of 1s even
 - Odd parity – total number of 1s odd
- Parity bit is append to the code at the leftmost position (MSB) .

A parity bit is a bit that is added to ensure that the number of bits with value of 1's in a given set of bits is always even or odd. Parity bits are used as the simplest error detecting code.

bit 1 decide
odd or even

Examples:

1 1 0 1 0 0 1 1 1

Even Parity bit

0 1 0 1 0 0 1 1 1

Odd Parity bit

Number 1s	Even Parity	Odd Parity
Even	0	1
Odd	1	0

(Remember these basic rule)

7 bits of data (number of 1s)	8 bits including parity	
	even	odd
0000000 (0)	00000000	10000000
1010001 (3)	11010001	01010001
1101001 (4)	01101001	11101001
1111111 (7)	11111111	01111111

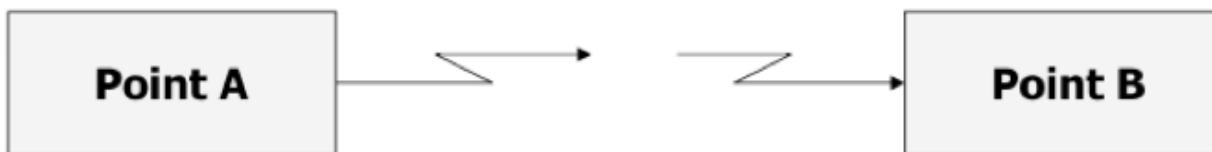
 Parity bit

- Example: Calculate the parity bit for the codes below.

Code	Number of 1s	Even/Odd	Even Parity	Odd Parity
110010	3	Odd	1 110010	0 110010
101110	4	Even	0 101110	1 101110
101000	2	Even	0 101000	1 101000
110111	5	Odd	1 110111	0 110111
111111	6	Even	0 111111	1 111111
100000	1	Odd	1 100000	0 100000

Error Detection by Parity Checking

- Assume that data = 0101
- It uses even parity.
- Therefore the appended parity bit is 0.
- The data with parity bit: 0 0101
- The data is transmitted.
- The data is received as 00001 → odd no. of 1, not even!!



Data: 0 0101
(even parity)

Data: 0 0001
(even parity)
but does not conform
to even parity

American Standard Code for Information Interchange (ASCII)

- It has 128 characters and symbols represented in 7-bit binary code
- Example :
- $A = 1000001_2$
- $a = 1100001_2$
- A parity bit is added so that the total number of bits is 8 → a byte.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

ASCII codes – More compact table

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	0	0	0	0	1	0	1	0	1	0	1	0	1	1	0	1	1	1	1	
Bits							Column →	0	1	2	3	4	5	6	7											
							Row ↓	0	0	0	0	0	1	0	1	1	0	0	1	0	1	1	0	1	1	1
0	0	0	0	0	0	0	NUL	DLE	SP	0	@	P	‘	p												
0	0	0	0	1	1	1	SOH	DC1	!	1	A	Q	a	q												
0	0	0	1	0	2	2	STX	DC2	”	2	B	R	b	r												
0	0	1	1	1	3	3	ETX	DC3	#	3	C	S	c	s												
0	1	0	0	0	4	4	EOT	DC4	\$	4	D	T	d	t												
0	1	0	1	1	5	5	ENQ	NAK	%	5	E	U	e	u												
0	1	1	0	0	6	6	ACK	SYN	&	6	F	V	f	v												
0	1	1	1	1	7	7	BEL	ETB	‘	7	G	W	g	w												
1	0	0	0	0	8	8	BS	CAN	(8	H	X	h	x												
1	0	0	0	1	9	9	HT	EM)	9	I	Y	i	y												
1	0	1	0	0	10	10	LF	SUB	*	:	J	Z	j	z												
1	0	1	1	1	11	11	VT	ESC	+	:	K	[k	{												
1	1	0	0	0	12	12	FF	FC	,	<	L	\	l	l												
1	1	0	1	1	13	13	CR	GS	-	=	M]	m	}												
1	1	1	0	0	14	14	SO	RS	.	>	N	^	n	~												
1	1	1	1	1	15	15	SI	US	/	?	O	_	o	DEL												

7-bits **binary₂**
ASCII code

Examples:

$b_7b_6b_5 \quad b_4b_3b_2b_1$
110 1101

is represent as
‘m’

Separated

	5	3		
$S =$	0	1 0 1 0 0 1 1	53	53
$C =$	1	1 0 0 0 0 1 1	A3	C3
$R =$	1	1 0 1 0 0 1 0	52	D2
$I =$	1	0 1 1 0 0 1	31	B1
$O \leftarrow$	0	0 0 1 1 0 0 0 0	30	30
$I \Rightarrow$	1	0 0 1 1 0 0 0 1	31	B1
$3 =$	0	0 0 1 1 0 0 1 1	33	33

Exercise 2b.2:

Convert the string SCR1013 to its ASCII hexadecimal value.

$$\text{SCR1013} = 53 \ 43 \ 52 \ 31 \ 30 \ 31 \ 33$$

By using even parity coding, calculate the parity bit and insert this bit at the MSB position. Recalculate the ASCII value in its hexadecimal representation.



Extra

Exercise 2b.3:

Given a string (character) UTM1435.

- a) Convert the string to its ASCII hexadecimal value.
- b) Calculate the odd parity bit and insert as MSB.
- a) Recalculate the ASCII value in hexadecimal.

U = 1 101 0101
 T = 0 101 0100
 M = 1 100 1101
 I = 0 011 0001
 N = 0 011 0100
 3 = 1 011 0011
 5 = 1 011 0101
 h = 0 110 1000

Number 1s	Even Parity	Odd Parity
Even	0	1
Odd	1	0

Extra

Character (ASCII)	ASCII (Hex)	Binary	Odd parity bit + Binary	New ASCII (Hex)
U	55	101 0101	1 101 0101	05
T	54	101 0100	0 101 0100	54
M	4D	100 1101	1 100 1101	C0
I	31	011 0001	0 011 0001	31
4	34	011 0100	0 011 0100	34
3	33	011 0011	1 011 0011	B3
5	35	011 0101	1 011 0101	B5
h	68	110 1000	0 110 1000	68