เอกสารประกอบการสอน

วิชา ระบบดิจิตอลเบื้องต้น (Introduction to Digital System)

รหัส 4121703

บทที่ 4 รหัส (CODE)

หลักสูตรระดับปริญญาตรี

พุทธศักราช 2551 (ปรับปรุง 2554)

โดย

จุฑาวุฒิ จันทรมาลี

สาขาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสวนดุสิต

บทที่ 4 รหัส (CODE)

4.1 บทนำ

การประมวลผลในระบบคอมพิวเตอร์คำสั่งและข้อมูลที่จะส่งให้คอมพิวเตอร์ประมวลผลนั้นจะต้องเป็นรหัสเลขฐานสองเท่านั้น แต่การใช้คำสั่งหรือการป้อนข้อมูลผู้ใช้จะใช้ภาษาเขียนหรืออักขระเพื่อง่ายต่อความเข้าใจ จึงทำให้เกิดรหัสต่าง ๆ เพื่อนำมาใช้ใน ระบบคอมพิวเตอร์ พอจะสรุปได้ เป็น 2 กลุ่ม คือ รหัสคำสั่ง เช่น การเขียนภาษาคอมพิวเตอร์สั่งให้คอมพิวเตอร์ทำงานอย่างใดอย่างหนึ่ง ภาษานั้นจะถูกแปลด้วยตัวแปลภาษา ให้เป็นภาษาเครื่อง ซึ่งอยู่ในรูปของรหัสเลขฐานสองส่งให้คอมพิวเตอร์ประมวลผล เมื่อได้ผลลัพธ์ก็จะถูกเปลี่ยนมาอยู่ในรูปสัญลักษณ์ ของตัวอักขระ หรือรูปภาพที่จะสื่อให้ผู้ใช้เข้าใจได้ รหัสสื่อสาร การส่งข้อมูลจากผู้ส่งไปยังผู้รับ และการสื่อสารระหว่างมนุษย์กับคอมพิวเตอร์ ซึ่งส่วนใหญ่จะอยู่ในรูปของตัวเลข ตัวอักษร และสัญลักษณ์พิเศษต่าง ๆ ที่อยู่บนแป้นพิมพ์ จำเป็นที่จะต้องเข้ารหัสให้เป็นรหัสเลขฐานสองดังได้กล่าวมาแล้ว รหัสที่ใช้แทนตัวอักขระเราเรียกว่า Alphanumeric Code ในปัจจุบันมีรหัสมาตรฐานที่ใช้กันอยู่ 2 อย่าง ได้แก่ EBCDIC (Extended BCD Interchange Code) และ ASCII Code (American Standard Code for Information Interchange)

ในการคำนวณเลขจำนวนมาก ๆ การใช้รหัสเลขฐานสองแทน อาจไม่สะดวกเพราะจะต้องใช้เลขฐานสองที่มีจำนวนบิทมาก ในทางปฏิบัติจะใช้รหัสอื่น ๆ มาแทนรหัสเลขฐานสอง รหัสดังกล่าวที่นิยมใช้กันมีอยู่ 3 แบบได้แก่ Binary Coded Decimal (BCD) Code, Excess-3 Code และ Gray Code ซึ่งจะได้กล่าวในรายละเอียด ต่อไป

4.2 Binary Coded Decimal (BCD) Code หรือ 8421 BCD Code

เป็นรหัสเลขฐานสอง 4 บิท ที่นำมาใช้แทนเลขฐานสิบแต่ละหลัก (Digit) ปกติเลขฐานสอง 4 บิท จะมี 16 จำนวน (0000_2 - 1111_2 ซึ่งสามารถแทนเลขฐานสิบได้ตั้งแต่ 0 - 15 แต่การใช้งานในรหัส BCD จะใช้รหัสเลขฐานสองแทนเลขฐานสิบตั้งแต่ 0 - 9 เท่านั้น และเลขฐานสิบหนึ่งหลัก จะใช้เลขฐานสอง 4 บิท ดังตารางข้างล่างนี้

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

ตัวอย่าง เปลี่ยน 5763_{10} ให้เป็นรหัส BCD

5763_{10} = 0101 0111 0110 0011

ตัวอย่าง เปลี่ยน 5763_{10} ให้เป็นรหัส BCD

5763_{10} = 0101 0111 0110 0011

หมายเหตุ จะแทนเลขฐานสิบแต่ละหลักด้วยเลขฐานสอง 4 bit

ตัวอย่าง Convert the number 9630_{10} to its BCD equivalent

(Ans. 1001 0110 0011 0000 BCD)

รหัส BCD code เป็นรหัสที่เข้าใจได้ง่าย แต่ประสิทธิภาพจะด้อยกว่ารหัสฐานสองโดยตรง

ลองพิจารณาเลขฐานสิบ ตั้งแต่ 0000_{10} to 9999_{10}

คำถาม ก. ถ้าเขียนเป็น รหัส BCD จะต้องใช้ทั้งหมดกี่บิท

(Ans. 16 bits (4 bits/digit))

ข. ถ้าเขียนเป็นเลขฐานสองโดยตรงจะต้องใช้ทั้งหมดกี่บิท

(Ans. 13 bits)

การคำนวณด้วยรหัส BCD

การบวกเลขรหัส BCD ให้บวกเหมือนเลขฐานสองปกติ แต่ถ้าผลบวกของทั้งสองหลักมากกว่า 9 จะต้องบวกเพิ่มอีก 6 (0110_2) ผลลัพธ์จึงจะถูกต้อง กรณีที่ผลลัพธ์ของทั้งสองหลักมากกว่า 9 จะเป็นได้เมื่อ

1. 4-bit บวกกันแล้วได้ค่ามากกว่า 9 (1010_2 ถึง $1111_2 = 10_{10}$ ถึง 15_{10})
2. มีตัวทดจากการบวกของ 4-bit (ผลรวมจาก 16_{10} ถึง 18_{10})

Add 45_{10} to 32_{10} using BCD arithmetic

25_{10}	0010 0101	BCD
34_{10}	0011 0100	BCD
59_{10}	0101 1001	BCD

Add 68_{10} to 29_{10} using BCD arithmetic

61_{10}	0110 0001	BCD	
29_{10}	0010 1001	BCD	
80_{10}	1000 1010	BCD	(least significant digit greater than 9 so add 0110_2)
	0110		บวกเข้าอีก 6
	1001 0000	BCD	

Add 68_{10} to 29_{10} using BCD arithmetic

68_{10} 0110 1000 BCD

29_{10} 0010 1001 BCD

97_{10} 1001 0001 BCD (carry from least significant digit so add 0110_2)

0110 บวกเข้าอีก 6

1001 0111 BCD

สรุป จะเห็นรหัส BCD เป็นรหัสเลขฐานสองที่ถูกเปลี่ยนจากเลขฐานสิบแต่ละหลัก โดยหนึ่งหลักของเลขฐานสิบจะต้องใช้รหัส BCD 4 บิท และการเขียนรหัส BCD ให้เขียนแยกเป็นกลุ่ม ๆ ละ 4 บิท

4.3 Excess-3 Code

รหัส Excess-3 หรือรหัสเติม 3 เป็นรหัสที่คล้ายกับรหัส BCD การสร้างรหัส excess-3 ก็เช่นเดียวกันกับรหัส BCD คือเปลี่ยนมาจากเลขฐานสิบ โดยใช้เลขฐานสอง 4-bit แทนเลขฐานสิบแต่ละหลัก แต่รหัส Excees-3 จะต้องบวก 3 เข้าไปแต่ละหลัก

เปลี่ยน 56_{10} ให้เป็น Excess - 3 0101 0110 (BCD)

0011 0011 (บวก 3 แต่ละหลัก)

1000 1001 (Excess 3 code ของ 56_{10})

Decimal	*Binary*	*BCD*	*Excess - 3*	*1's complement*
0	0000	0000	0011	1100
1	0001	0001	0100	1011
2	0010	0010	0101	1010
3	0011	0011	0110	1001
4	0100	0100	0111	1000
5	0101	0101	1000	0111
6	0110	0110	1001	0110
7	0111	0111	1010	0101
8	1000	1000	1011	0100
9	1001	1001	1100	0011

การคำนวณด้วยรหัส Excess-3

ตัวอย่าง จงบวก 3_{10} กับ 5_{10} โดยใช้รหัส Excess-3

3_{10}	0110	Excess-3 ของ 3_{10}
5_{10}	1000	Excess-3 ของ 5_{10}
8_{10}	1110	Excess-6 ของ 8_{10}
	-0110	ลบด้วย 3
	1011	Excess-3 ของ 8_{10}

ตัวอย่าง จงบวก 6_{10} กับ 7_{10} โดยใช้รหัส Excess-3

6_{10}		1001	Excess-3 ของ 6_{10}
7_{10}		1010	Excess-3 ของ 7_{10}
13_{10}	0001	0011	ผลลัพธ์มีตัวทด
	+0011	+0011	บวกด้วย 3 ทั้งสองชุด
	0100	0110	Excess-3 ของ 13_{10}

ตัวอย่าง จงหาผลต่างของ 19_{10} กับ 12_{10} โดยใช้รหัส Excess-3

19_{10}	0100	1100	Excess-3 ของ 19_{10}
-12_{10}	-0100	0101	Excess-3 ของ 12_{10}
7_{10}	0000	0111	ผลลัพธ์เป็น BCD
	+	0011	บวกด้วย 3
		1010	Excess-3 ของ 7_{10}

หมายเหตุ รหัส Excess-3 ไม่เป็น weighted code คือไม่ได้คิดขนาดจากน้ำหนัก (weight) ของแต่ละบิท ส่วนรหัส BCD เป็น weight code บางทีเราเรียกว่า 8421 BCD code รหัส Excess-3 มีจุดเด่นคือ เมื่อนำตารางมาพับครึ่งตัวเลขที่อยู่ห่างจากรอยพับเท่ากันจะเป็นคอมพลีเมนต์กัน

4.4 รหัส 4221 BCD Code

รหัส 4221 BCD code เป็นรหัสชนิด weighted code เป็นการกำหนดน้ำหนัก (weight) ให้แก่ตัวเลขฐานสองแต่ละหลัก 4221 BCD code ใช้ตัวเลขฐานสองจำนวน 4 บิท ใช้แทนตัวเลขฐานสิบตั้งแต่ 0-9 โดยกำหนดน้ำหนักของแต่ละบิทเป็น 4, 2, 2 และ 1 ตามลำดับ ดังตารางข้างล่างนี้

Decimal	*4221*	*1's complement*
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	1000	0111
5	0111	1000
6	1100	0011
7	1101	0010
8	1110	0001
9	1111	0000

กฎของ 4221 BCD code

ต่ำกว่า 5 ใช้ 2 บิททางขวามือก่อน

สูงกว่า 5 ใช้ 2 บิททางซ้ายมือก่อน

สำหรับ 5 = 2+2+1 ไม่ใช้ 4+1

4.5 Gray Code

รหัส Gray code เป็นรหัสชนิด Unweighted code ใช้ตัวเลขฐานสองจำนวน 4 บิท เช่นเดียวกับรหัส BCD, Excess-3, 4221 BCD code และปกติจะใช้แทนตัวเลขฐานสิบ ตั้งแต่ 0-9 แต่เราอาจจะเขียนรหัส Gray แทนเลขฐานสิบได้ตั้งแต่ 0-15 หลักการของ Gray code ตัวเลขที่อยู่ติดกันในลำดับ จะมีค่าต่างกันเพียง 1 บิทเท่านั้น จากตารางข้างล่างนี้ เป็นการเทียบ 4-bit binary code กับ 4-bit Gray code

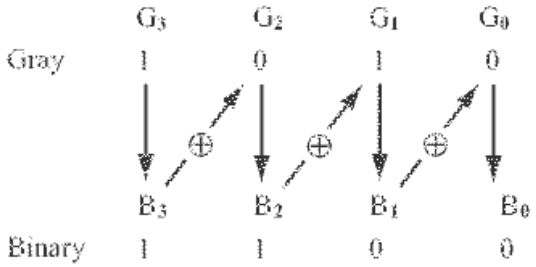
Decimal	*Binary*	*Gray*
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100

9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

การเปลี่ยน Gray Code เป็น Binary Code

ตามที่กล่าวมาแล้ว Gray Code เป็นรหัสชนิด unweighted จึงไม่สะดวกในการประมวลผล ดังนั้นจึงจะต้องเปลี่ยนมาเป็น Binary code ก่อน การเปลี่ยนให้ดูแผนภูมิข้างล่าง ซึ่งจะเป็นการบวกโดยไม่สนใจตัวทดเราเรียกว่า การรวมแบบ modulo sum หรือ ใช้ exclusive OR operator เป็นตัวกระทำ

Convert 1010 GRAY to binary



i.e. $B_3 = G_3$

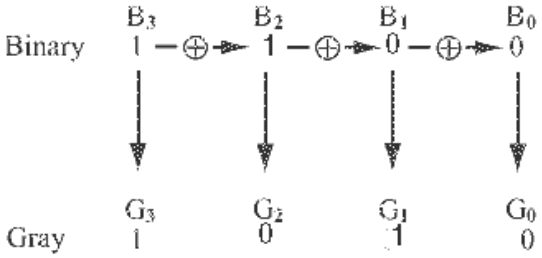
$B_2 = B_3 \oplus G_2$

$B_1 = B_2 \oplus G_1$

$B_0 = B_1 \oplus G_0$

การเปลี่ยน Binary Code เป็น Gray Code

Convert 1100_2 to Gray code



i.e. $G_3 = B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

4.6 Alphanumeric (ASCII) Code

รหัสที่กล่าวมาแล้วทั้งหมดเป็นรหัสเลขฐานสองที่ใช้แทนตัวเลข 0 ถึง 9 แต่ยังมีรหัสเลขฐานสองอีกลุ่มหนึ่งที่ใช้แทนอักขระทั้งตัวอักษร ตัวเลข และสัญลักษณ์พิเศษ เราเรียกว่า Alphanumeric Code ในปัจจุบันมีรหัสมาตรฐานที่ใช้กันอยู่ 2 อย่าง ได้แก่ EBCDID (Extended BCD Interchange Code) และ ASCII Code (American Standard Code for Information Interchange) ดังได้กล่าวนำแล้ว สำหรับเนื้อหาในบทนี้จะขอกล่าวเฉพาะรหัส ASCII Code ซึ่งเป็นรหัสที่ใช้กันมากและเป็นมาตรฐานของ PC ในปัจจุบัน

ASCII Code เป็นรหัสที่ถูกพัฒนาจนเป็นที่ยอมรับให้เป็นมาตรฐานของรหัสที่ใช้ในคอมพิวเตอร์ เป็นรหัสขนาด 7 บิท สามารถใช้แทนอักขระและรหัสควบคุมต่างได้ ถึง 128 รหัส (0-127, 72=128) ซึ่งสามารถใช้แทนอักษรภาษาอังกฤษตัวเล็ก ตัวใหญ่ ตัวเลข 0-9 เครื่องหมายอักขระพิเศษ และรหัสควบคุมอีก 32 รหัส นอกจากนี้ยังเพิ่มรหัสส่วนขยายอีกโดยเพิ่มเป็นรหัส 8 บิท จึงมีส่วนขยายเพิ่มเติมตั้งแต่ 128 - 255 เพื่อใช้แทนกราฟิก สัญลักษณ์ทางคณิตศาสตร์ และเหลือรหัสเพียงพอที่จะใช้แทนอักขระภาษาต่างประเทศอื่น ๆ อีก เช่นภาษาไทย เป็นต้น

ASCII stands for American Standard Code for Information Interchange. ASCII was developed a long time ago and now the characters are not always used in the same way on some machines. Below is the ASCII character table and this includes descriptions of the first 32 characters. ASCII was designed for use with teletypes and so the descriptions are somewhat obscure and their use is frequently not as intended.

ASCII Table

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	0	000	NUL (null)	32	20	040	SPACE	64	40	100	@	96	60	140	`
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

The 7-bit ASCII Character Set

The two most significant bits of the ASCII code generally denote the type of information contained in the 5 least significant bits

00	*:*	*Control Characters*
01	:	Numeric/Special Characters
10	:	Upper case alphabet
11	:	Lower case alphabet

The following table illustrates the ASCII codes

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

Extended Ascii Codes

These were added later and are not true ascii

There are different extended sets, but this is the most common

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	⊛	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

4.7 Error Detecting Code

In any electronic system involving the transfer of data (in the form of binary digits) then data transmission errors are possible. The method of parity is widely used as a method of error detection.

An extra bit, known as the parity bit is attached as the least-significant bit to the binary data word (or code group) to be transferred. The new data word to be transmitted (known as the total group) is thus the original code group with the parity bit appended. Two systems are used, namely

even parity	*the value of the parity bit is set such that the total number of 1s in the data word is even*
e.g.	11001 which has an odd number of 1s. The new total group is thus 110011 11110 which has an even number of 1s. The new total group is thus 111100
odd parity	the value of the parity bit is set such that the total number of 1s in the data word is **odd**
e.g.	11001 which has an odd number of 1s. The new total group is thus 110010 11110 which has an even number of 1s. The new total group is thus 111101

At the receiving end, a check is made on the parity of the whole code to detect an error before stippling the parity off to recover the original data word. Examples of circuits for transmitting (coding) and receiving (decoding) parity-encoded data are given below.

The method of parity does not pinpoint the error, rather it acts as an error flag i.e. indicating an error has occurred somewhere. Often the code group has to be re-transmitted when a

parity error is detected. Similarly, this method only safeguards against single errors and not multiple errors which could conspire to leave the parity check valid. In order to detect complex errors and to pinpoint where the errors have occurred then more sophisticated and complex error-checking algorithms have to be employed - which subsequently requires more data bits to be transmitted for each code group.

สรุป

การสื่อสารระหว่างมนุษย์กับคอมพิวเตอร์ ซึ่งส่วนใหญ่จะอยู่ในรูปของตัวเลข ตัวอักษร และสัญลักษณ์พิเศษต่าง ๆ ที่อยู่บนแป้นพิมพ์ จำเป็นที่จะต้องเข้ารหัสให้เป็นรหัสเลขฐานสองดังได้กล่าวมาแล้ว รหัสที่ใช้แทนตัวอักขระเราเรียกว่า Alphanumeric Code ในปัจจุบันมีรหัสมาตรฐานที่ใช้กันอยู่ 2 อย่าง ได้แก่ EBCDIC (Extended BCD Interchange Code) และ ASCII Code (American Standard Code for Information Interchange)

แบบฝึกหัด

1. Convert the following to BCD

1.1 $(100000111)_2$

1.2 $(1011)_{16}$

1.3 $(245)_{10}$

1.4 $(1FD)_{16}$

1.5 $(106)_{16}$

1.6 $(32A)_{16}$

2. Convert the following BCD numbers to binary

2.1 $(126)_{BCD}$

2.2 $(9126)_{BCD}$

2.3 $(6712)_{BCD}$

3. Convert the following to Gray code

3.1 $(1011)_2$

3.2 $(12)_{10}$

3.3 $(1D)_{16}$

4. Convert the following Gray codes to binary

4.1 $(1011)_{GRAY}$

4.2 $(100)_{GRAY}$

4.3 $(110011)_{GRAY}$

5. Convert your name and address to 7-bit ASCII.

6. For each 7-bit ASCII code in question 5, add a parity bit (8 bit) for even parity.